Abstract Reduction Systems: Fundamental notions and notations

Definition 8.1. $(U, \to)$ $U \neq \emptyset, \to$ *binary relation* *is called a reduction system.*

- *Notions:*

- $x \in U$ *reducible iff* $\exists y : x \to y$
  irreducible if not reducible.
- $x \xrightarrow{*} y$ *reflexive, transitive closure,* $x \xrightarrow{+} y$ *transitive closure,* $x \xleftrightarrow{*} y$ *reflexive, symmetrical, transitive closure.*
- $x \xrightarrow{i} y$ $i \in \mathbb{N}$ *defined as usual. Notice* $x \xrightarrow{*} y = \bigcup_{i \in \mathbb{N}} x \xrightarrow{i} y$.
- $x \xrightarrow{*} y$, y *irreducible, then* y *is a normal form for* x. *Abb:: NF*
- $\Delta(x) = \{y \mid x \to y\}$, *the set of direct successors of* x.
- $\Delta^{+}(x)$ *proper successors,* $\Delta^{*}(x)$ *successors.*

Notions and notations

- $\Lambda(x) = \max\{i \mid \exists y : x \xrightarrow{i} y\}$ derivational complexity. $\Lambda : U \to \mathbb{N}_\infty$
- $\to$ noetherian (terminating, satisfies the chain condition), in case there is no infinite chain $x_1 \to x_2 \to x_3 \to \cdots$.
- $\to$ bounded, in case that $\Lambda : U \to \mathbb{N}$.
- $\to$ cycle free :: $\neg\exists x \in U : x \xrightarrow{+} x$
- $\to$ locally finite $x \left.\begin{array}{l} \nearrow \\ \to \\ \searrow \end{array}\right\}$, i.e. $\Delta(x)$ finite for every x.

Notions and notations

Simple properties:

- $\rightarrow$ cycle free, then $\xrightarrow{*}$ partial ordering.
- $\rightarrow$ noetherian, then $\rightarrow$ cycle free.
- $\rightarrow$ bounded, so $\rightarrow$ noetherian.
  but not the other way around!
- $\rightarrow \subset \overset{+}{\Rightarrow}$ and $\Rightarrow$ noetherian, then $\rightarrow$ noetherian.

Principle of the Noetherian Induction

Definition 8.2. *$\to$ binary relation on U, P predicate on U.*
P is $\to$-complete, when

$$\forall x[(\forall y \in \Delta^+(x) : P(y)) \supset P(x)]$$

Fact:
PNI: If $\to$ is noetherian and P is $\to$-complete, then $P(x)$ holds for all $x \in U$.

Applications

Lemma 8.3. *$\rightarrow$ noetherian, then each $x \in U$ has at least one normal form.*

More applications to come.... See e.g. König's lemma.

Definition 8.4. *Main properties for $(U, \rightarrow)$*

- $\rightarrow$ *confluent iff* $\overset{*}{\longleftarrow} \circ \overset{*}{\longrightarrow} \;\subseteq\; \overset{*}{\longrightarrow} \circ \overset{*}{\longleftarrow}$
- $\rightarrow$ *Church-Rosser iff* $\overset{*}{\longleftrightarrow} \;\subseteq\; \overset{*}{\longrightarrow} \circ \overset{*}{\longleftarrow}$
- $\rightarrow$ *locally-confluent iff* $\longleftarrow \circ \longrightarrow \;\subseteq\; \overset{*}{\longrightarrow} \circ \overset{*}{\longleftarrow}$
- $\rightarrow$ *strong-confluent iff* $\longleftarrow \circ \longrightarrow \;\subseteq\; \overset{*}{\longrightarrow} \circ \overset{\leq 1}{\longleftarrow}$
- *Abbreviation: joinable $\downarrow$:*

$$\downarrow = \overset{*}{\longrightarrow} \circ \overset{*}{\longleftarrow}$$

Important relations

Lemma 8.5. $\rightarrow$ *confluent iff* $\rightarrow$ *Church-Rosser.*

Theorem 8.6. *(Newmann Lemma) Let* $\rightarrow$ *be noetherian, then*

$$\rightarrow \text{ confluent iff } \rightarrow \text{ locally confluent.}$$

Consequence 8.7. a) *Let* $\rightarrow$ *confluent and* $x \stackrel{*}{\longleftrightarrow} y$.

- i) *If* y *is irreducible, then* $x \stackrel{*}{\longrightarrow} y$. *In particular, when* x, y *irreducible, then* $x = y$.
- ii) $x \stackrel{*}{\longleftrightarrow} y$ *iff* $\Delta^*(x) \cap \Delta^*(y) \neq \varnothing$.
- iii) *If* x *has a NF, then it is unique.*
- iv) *If* $\rightarrow$ *is noetherian, then each* $x \in U$ *has exactly one NF: notation* $x\downarrow$

b) *If in* $(U, \rightarrow)$ *each* $x \in U$ *has exactly one NF, then* $\rightarrow$ *is confluent (in general not noetherian).*

Convergent Reduction Systems

Definition 8.8. $(U, \to)$ *convergent iff* $\to$ *noetherian and confluent.*

Important since: $x \overset{*}{\longleftrightarrow} y$ *iff* $x\downarrow = y\downarrow$

Hence if $\to$ *effective* $\leadsto$ *decision procedure for Word Problem (WP):*

For programming: $x \overset{*}{\longrightarrow} x\downarrow$, $f(t_1, \ldots, t_n) \overset{*}{\longrightarrow}$ *„value“*

As usual these properties are in general undecidable properties.

Task: Find sufficient computable conditions which guarantee these properties.

Termination and Confluence

Sufficient conditions/techniques

Lemma 8.9. *$(U, \to)$, $(M, \succ)$, $\succ$ well founded (WF) partial ordering. If there is $\varphi : U \to M$ with $\varphi(x) \succ \varphi(y)$ for $x \to y$, then $\to$ is noetherian.*

Example 8.10. *Often $(\mathbb{N}, >), (\Sigma^*, >)$ can be used. For $w \in \Sigma^*$ let $|w|$ length, $|w|_a$ a-length $a \in \Sigma$.*

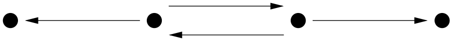
WF-partial orderings on Σ^*

- $x > y$ iff $|x| > |y|$
- $x > y$ iff $|x|_a > |y|_a$
- $x > y$ iff $|x| > |y|$, $|x| = |y| \wedge x \succ_{lex} y$

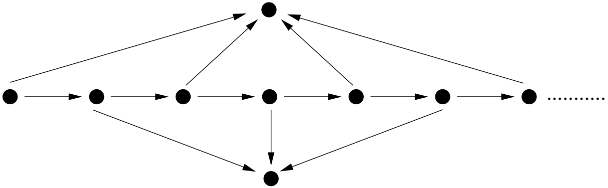
Notice that pure lex-ordering on Σ^* is not noetherian.

Sufficient conditions for confluence

Termination: Confluence *iff* local confluence
Without termination this doesn't hold!



or



Confluence without termination

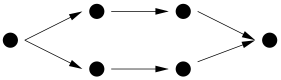
Theorem 8.11. *$\rightarrow$ is confluent iff for every $u \in U$ holds:*

$$\text{from } u \rightarrow x \text{ and } u \xrightarrow{*} y \text{ it follows } x \downarrow y.$$

▷ *one-sided localization of confluence* ◁

Theorem 8.12. *If $\rightarrow$ is strong confluent, then $\rightarrow$ is confluent.*

Not a necessary condition:



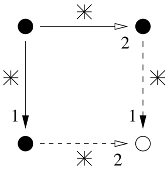
Combination of Relations

Definition 8.13. *Two relations $\rightarrow_1$, $\rightarrow_2$ on U commute, iff*

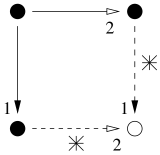
$$ {}_1\overset{*}{\leftarrow} \circ \overset{*}{\rightarrow}_2 \quad \subseteq \quad \overset{*}{\rightarrow}_2 \circ {}_1\overset{*}{\leftarrow} $$

.

They commute locally iff $ {}_1\leftarrow \circ \rightarrow_2 \quad \subseteq \quad \overset{*}{\rightarrow}_2 \circ {}_1\overset{*}{\leftarrow}$.



commutating



locally commutating

Combination of Relations

Lemma 8.14. *Let* $\to \;=\; \to_1 \cup \to_2$

(1) If $\to_1$ *and* $\to_2$ *commute locally and* $\to$ *is noetherian, then* $\to_1$ *and* $\to_2$ *commute.*
(2) If $\to_1$ *and* $\to_2$ *are confluent and commute, then* $\to$ *is also confluent.*

Problem: Non-Orientability:

(a) $x + 0 = x,\ \ x + s(y) = s(x + y)$
(b) $x + y = y + x,\ \ (x + y) + z = x + (y + z)$

▷ *Problem: permutative rules like (b)* ◁

Non-Orientability

Definition 8.15. *Let* $(U, \to, \vdash\!\dashv)$ *with* $\to$ *a binary relation,* $\vdash\!\dashv$ *a symmetrical relation.*

Let $\models\!\!\dashv \;=\; \leftrightarrow \cup \vdash\!\dashv, \qquad \sim \;=\; \overset{*}{\vdash\!\dashv}, \qquad \approx \;=\; \overset{*}{\models\!\!\dashv},$

$\to_\sim \;=\; \sim \circ \to \circ \sim, \qquad \downarrow_\sim \;=\; \overset{*}{\to} \circ \sim \circ \overset{*}{\leftarrow}.$

If $x \downarrow_\sim y$ *holds, then* $x, y \in U$ *are called joinable modulo* $\sim$.

$\to$ *is called Church-Rosser modulo* $\sim$ *iff* $\approx \;\subseteq\; \downarrow_\sim$

$\to$ *is called locally confluent modulo* $\sim$ *iff* $\leftarrow \circ \to \;\subseteq\; \downarrow_\sim$

$\to$ *is called locally coherent modulo* $\sim$ *iff* $\leftarrow \circ \vdash\!\dashv \;\subseteq\; \downarrow_\sim$

Non-Orientability - Reduction Modulo $\vdash\dashv$

Theorem 8.16. *Let $\rightarrow_{\sim}$ be terminating. Then $\rightarrow$ is Church-Rosser modulo $\sim$ iff $\sim$ is local confluent modulo $\sim$ and local coherent modulo $\sim$.*

Most frequent application: Modulo AC (Associativity + Commutativity)

Representation of equivalence relations by convergent reduction relations

Situation: Given: $(U, \vdash\!\dashv)$ and a noetherian PO $>$ on U , find: $(U, \rightarrow)$ with

(i) $\rightarrow \;\subseteq\; >$, $\rightarrow$ convergent on U and

(ii) $\overset{*}{\leftrightarrow} \;=\; \sim$ with $\sim \;=\; \overset{*}{\vdash\!\dashv}$

Idea: Approximation of $\rightarrow$ by stepwise transformations

$$(\vdash\!\dashv, \emptyset) = (\vdash\!\dashv_0, \rightarrow_0) \vdash (\vdash\!\dashv_1, \rightarrow_1) \vdash (\vdash\!\dashv_2, \rightarrow_2) \vdash \dots$$

Invariant in i-th. step:

(i) $\sim \;=\; (\vdash\!\dashv_i \cup \leftrightarrow_i)^*$ and
(ii) $\rightarrow_i \;\subseteq\; >$

Goal: $\vdash\!\dashv_i = \emptyset$ for an i and $\rightarrow_i$ convergent.

Representation of equivalence relations by convergent reduction relations

Allowed operations in i-th. step:

(1) Orient:: $u \rightarrow_{i+1} v$, if $u > v$ and $u \vdash\!\dashv_i v$
(2) New equivalences:: $u \vdash\!\dashv_{i+1} v$, if $u \ {}_i\!\leftarrow w \rightarrow_i v$
(3) Simplify:: $u \vdash\!\dashv_i v$ to $u \vdash\!\dashv_{i+1} w$, if $v \rightarrow_i w$

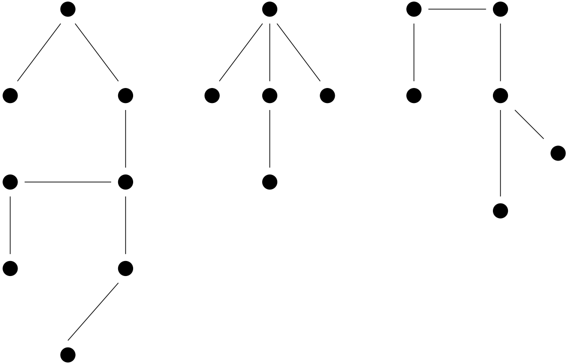
Goal: Limit system

$$\rightarrow \ = \ \rightarrow_\infty \ = \bigcup\{\rightarrow_i \mid i \in \mathbb{N}\} \text{ with } \vdash\!\dashv_\infty \ = \emptyset$$

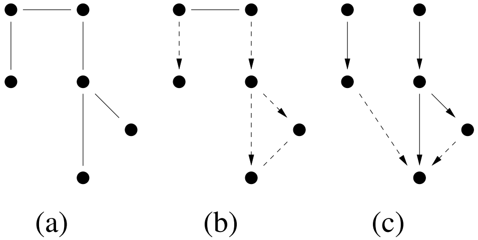
Hence:

- $\longrightarrow_\infty \subseteq >$, i.e. noetherian
- $\overset{*}{\longleftrightarrow} = \sim$
- $\longrightarrow_\infty$ convergent !

Grafical representation of an equivalence relation



Transformation of an equivalence relation



Inference system for the transformation of an equivalence relation

Definition 8.17. *Let $>$ be a noetherian PO on U. The inference system $\mathcal{P}$ on objects $(\vdash\dashv, \rightarrow)$ contains the following rules:*

(1) *Orient*

$$\frac{(\vdash\dashv \cup \{u \vdash\dashv v\}, \rightarrow)}{(\vdash\dashv, \rightarrow \cup \{u \rightarrow v\})} \text{ if } u > v$$

(2) *Introduce new consequence*

$$\frac{(\vdash\dashv, \rightarrow)}{(\vdash\dashv \cup \{u \vdash\dashv v\}, \rightarrow)} \text{ if } u \leftarrow \circ \rightarrow v$$

(3) *Simplify*

$$\frac{(\vdash\dashv \cup \{u \vdash\dashv v\}, \rightarrow)}{(\vdash\dashv \cup \{u \vdash\dashv w\}, \rightarrow)} \text{ if } v \rightarrow w$$

Inference system (Cont.)

(4) Eliminate identities

$$\frac{(\vdash\!\dashv \cup \{u \vdash\!\dashv u\}, \rightarrow)}{(\vdash\!\dashv, \rightarrow)}$$

$(\vdash\!\dashv, \rightarrow) \vdash_{\mathcal{P}} (\vdash\!\dashv', \rightarrow')$ if
$(\vdash\!\dashv, \rightarrow)$ can be transformed in one step with a rule $\mathcal{P}$ into $(\vdash\!\dashv', \rightarrow')$.

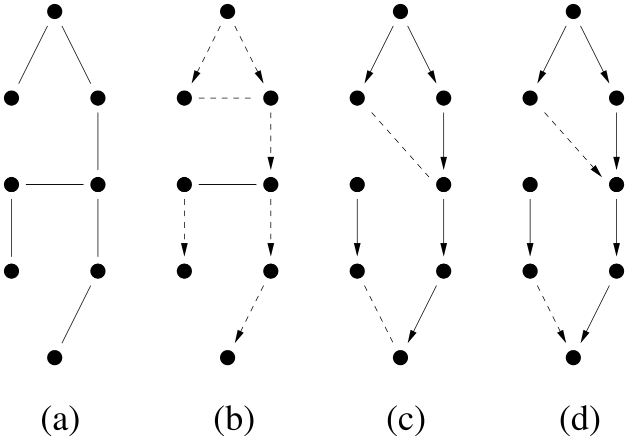
$\vdash^*_{\mathcal{P}}$ transformation relation in finite number of steps with $\mathcal{P}$.

A sequence $((\vdash\!\dashv_i, \rightarrow_i))_{i \in \mathbb{N}}$ is called $\mathcal{P}$-derivation, if

$$(\vdash\!\dashv_i, \rightarrow_i) \vdash_{\mathcal{P}} (\vdash\!\dashv_{i+1}, \rightarrow_{i+1}) \text{ for every } i \in \mathbb{N}$$

.

Transformation with the inference system



Properties of the inference system

Lemma 8.18. *Let* $(\vdash\!\dashv, \to) \vdash_{\mathcal{P}} (\vdash\!\dashv', \to')$

(a) *If* $\to \;\subseteq\; >$, *then* $\to' \;\subseteq\; >$

(b) $(\vdash\!\dashv \cup \leftrightarrow)^* \;=\; (\vdash\!\dashv' \cup \leftrightarrow')^*$

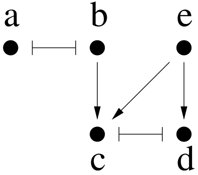
Problem:
When does $\mathcal{P}$ deliver a convergent reduction relation $\to$?
How to measure progress of the transformation?

Idea: Define an ordering $>_{\mathcal{P}}$ on equivalence-proofs, and prove that the inference system $\mathcal{P}$ decreases proofs with respect to $>_{\mathcal{P}}$!

In the proof ordering $\xrightarrow{*} \circ \xleftarrow{*}$ proofs should be minimal.

Equivalence Proofs

Definition 8.19. *Let* $(\vdash\dashv, \rightarrow)$ *be given and* $>$ *a noetherian PO on* U.
Furthermore let $(\vdash\dashv \cup \leftrightarrow)^* = \sim$.
A proof for $u \sim v$ *is a sequence* $u_0 *_1 u_1 *_2 \cdots *_n u_n$ *with* $*_i \in \{\vdash\dashv, \leftarrow, \rightarrow\}$,
$u_i \in U$, $u_0 = u$, $u_n = v$ *and for every* i $u_i *_{i+1} u_{i+1}$ *holds.*
$P(u) = u$ *is proof for* $u \sim u$.
A proof of the form $u \xrightarrow{*} z \xleftarrow{*} v$ *is called V-proof.*



Proofs for $a \sim e$:

$P_1(a,e) = a \vdash\dashv b \rightarrow c \vdash\dashv d \leftarrow e$ $\quad$ $P_2(a,e) = a \vdash\dashv b \rightarrow c \leftarrow e$

Proof orderings

Two proofs in $(\vdash, \rightarrow)$ are called equivalent, if they prove the equivalence of the same pair (u, v). Hence e.g. $P_1(a, e)$ and $P_2(a, e)$ are equivalent.

Notice: If $P_1(u, v), P_2(v, w)$ and $P_3(w, z)$ are proofs, then $P(u, z) = P_1(u, v)P_2(v, w)P_3(w, z)$ is also a proof.

Definition 8.20. *A proof ordering $>_B$ is a PO on the set of proofs that is monotonic, i.e.. $P >_B Q$ for each subproof, and if $P >_B Q$ then $P_1 P P_2 >_B P_1 Q P_2$.*

Lemma 8.21. *Let $>$ be noetherian PO on U and $(\vdash, \rightarrow)$, then there exist noetherian proof orderings on the set of equivalence proofs.*

Proof: Using multiset orderings.

Multisets and the multiset ordering

Instruments: Multiset ordering

Objects: U, $Mult(U)$ Multisets over U

$A \in Mult(U)$ iff $A : U \to \mathbb{N}$ with $\{u \mid A(u) > 0\}$ finite.

Operations: $\cup, \cap, -$

$$(A \cup B)(u) := A(u) + B(u)$$

$$(A \cap B)(u) := min\{A(u), B(u)\}$$

$$(A - B)(u) := max\{0, A(u) - B(u)\}$$

Explicit notation:
$U = \{a, b, c\}$ *e.g.* $A = \{\{a, a, a, b, c, c\}\}, B = \{\{c, c, c\}\}$

Multiset ordering

Definition 8.22. *Extension of $(U, >)$ to $(Mult(U), \gg)$*

$$A \gg B \text{ iff there are } X, Y \in Mult(U) \text{ with } \emptyset \neq X \subseteq A \text{ and}$$
$$B = (A - X) \cup Y, \text{ so that } \forall y \in Y \;\; \exists x \in X \; x > y$$

Properties:

(1) $>$ PO $\rightsquigarrow$ $\gg$ PO
(2) $\{m_1\} \gg \{m_2\}$ iff $m_1 > m_2$
(3) $>$ total $\rightsquigarrow$ $\gg$ total
(4) $A \gg B \rightsquigarrow A \cup C \gg B \cup C$
(5) $B \subset A \;\rightsquigarrow\; A \gg B$
(6) $>$ noetherian iff $\gg$ noetherian

Example: $a < b < c$ then $B \gg A$

Construction of the proof ordering

Let $(\vdash\!\dashv, \rightarrow)$ be given and $>$ a noetherian PO on U with $\rightarrow \subset >$
Assign to each „atomic" proof a complexity

$$c(u * v) = \begin{cases} \{u\} & \text{if } u \rightarrow v \\ \{v\} & \text{if } u \leftarrow v \\ \{\{u, v\}\} & \text{if } u \vdash\!\dashv v \end{cases}$$

Extend this complexity to „composed" proofs through

$$c(P(u)) = \emptyset$$
$$c(P(u, v)) = \{\{c(u_i *_{i+1} u_{i+1}) \mid i = 0, \ldots n - 1\}\}$$

Notice: $c(P(u, v)) \in Mult(Mult(U))$

Define ordering on proofs through

$$P >_{\mathcal{P}} Q \;\; \textit{iff} \;\; c(P) \ggg c(Q)$$

Construction of the proof ordering

Fact : $>_{\mathcal{P}}$ is notherian proof ordering!

Which proof steps are large and which small?

Consider:

(a) $P_1 = x \leftarrow u \rightarrow y$, $P_2 = x \vdash\!\dashv y$

$c(P_1) = \{\{\{u\}, \{u\}\}\} \ggg \{\{x, y\}\} = c(P_2)$ since $u > x$ and $u > y$
$\rightsquigarrow P_1 >_{\mathcal{P}} P_2$

analogously for

(b) $P_1 = x \vdash\!\dashv y$, $P_2 = x \rightarrow y$

(c) $P_1 = u \vdash\!\dashv v$, $P_2 = u \vdash\!\dashv w \leftarrow v$

(d) $P_1 = u \vdash\!\dashv v$, $P_2 = u \rightarrow w \leftarrow v$

Fair Deductions in $\mathcal{P}$

Definition 8.23 (Fair deduction). *Let $(\vdash\dashv_i, \rightarrow_i)_{i\in\mathbb{N}}$ be a $\mathcal{P}$-deduction. Let*

$$\vdash\dashv^\infty = \bigcup_{i\geq 0} \bigcap_{j\geq i} \vdash\dashv_i \text{ and } \rightarrow^\infty = \bigcup_{i\geq 0} \rightarrow_i .$$

The $\mathcal{P}$-Deduction is called fair, in case

(1) $\vdash\dashv^\infty = \emptyset$ and
(2) If $x \;{}^\infty\!\!\leftarrow u \rightarrow^\infty y$, then there exists $k \in \mathbb{N}$ with $x \vdash\dashv_k y$.

Lemma 8.24. *Let $(\vdash\dashv_i, \rightarrow_i)_{i\in\mathbb{N}}$ be a fair $\mathcal{P}$-deduction*

(a) For each proof P in $(\vdash\dashv_i, \rightarrow_i)$ there is an equivalent proof P' in $(\vdash\dashv_{i+1}, \rightarrow_{i+1})$ with $P \geq_{\mathcal{P}} P'$.

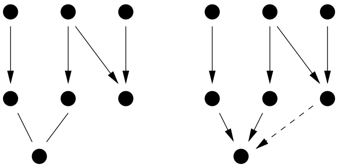
(b) Let $i \in \mathbb{N}$ and P proof in $(\vdash\dashv_i, \rightarrow_i)$ which is not a V-proof. Then there exists a $j > i$ and an equivalent proof P' in $(\vdash\dashv_j, \rightarrow_j)$ with $P >_{\mathcal{P}} P'$.

Main result

Theorem 8.25. *Let* $(\vdash_i, \to_i)_{i \in \mathbb{N}}$ *a fair* $\mathcal{P}$*-Deduction and* $\to \; = \; \to^{\infty}$. *Then*

(a) If $u \sim v$, *then there exists an* $i \in \mathbb{N}$ *with* $u \xrightarrow{*}_i \circ \; {}_i\!\xleftarrow{*} v$

(b) $\to$ *is convergent and* $\xleftrightarrow{*} \; = \; \sim$



Term Rewriting Systems

Goal: Operationalization of specifications and implementation of functional programming languages

Given $spec = (sig, E)$ when is T_{spec} a computable algebra?

$(T_{spec})_s = \{[t]_{=_E} : t \in Term(sig)_s\}$

T_{spec} is a computable Algebra if there is a computable function

$rep : Term(sig) \to Term(sig)$, with $rep(t) \in [t]_{=_E}$ the "unique representative" in its equivalence class.

Paradigm: Choose as representative the minimal object in the equivalence class with respect to an ordering.

$f(x_1, ..., x_n) : ((T_{spec})_{s_1} \times ...(T_{spec})_{s_n}) \to (T_{spec})_s$
$f([r_1], ..., [r_n]) := [rep(f(rep(r_1), ..., (rep(r_n))]$

Term Rewriting Systems

Definition 9.1. *Rules, rule sets, reduction relation*

- *Sets of variables in terms: For $t \in Term_s(F, V)$ let $V(t)$ be the set of the variables in t (Recursive definition! always finite)*
  Notice: $V(t) = \emptyset$ iff t is ground term.
- *A rule is a pair $(l, r), l, r \in Term_s(F, V)$ $(s \in S)$ with $Var(r) \subseteq Var(l)$*
  Write: $l \to r$
- *A rule system R is a set of rules.*
  R defines a reduction relation $\to_R$ over $Term(F, V)$ by:
  $$t_1 \to_R t_2 \quad \text{iff } \exists\, l \to r \in R, p \in O(t_1), \sigma \text{ substitution } : \quad t_1|_p = \sigma(l) \wedge t_2 = t_1[\sigma(r)]_p$$
- *Let $(Term(F, V), \to_R)$ be the reduction system defined by R (term rewriting system).*
- *A rule system R defines a congruence $=_R$ on $Term(F, V)$ just by considering the rules as equations.*

Term Rewriting Systems

Goal: Transform E in R, so that $=_E = \stackrel{*}{\longleftrightarrow}_R$ holds and $\rightarrow_R$ has "sufficiently"good termination and confluence properties.
For instance convergent or confluent. Often it is enough when these properties hold "only" on the set of ground terms.

Notice:

- The condition $V(r) \subseteq V(l)$ in the rule $l \rightarrow r$ is necessary for the termination.
  If neither $V(r) \subseteq V(l)$ nor $V(l) \subseteq V(r)$ in an equation $l = r$ of a specification, we have used superfluous variables in some function's definition.
- $\rightarrow_R$ is compatible with substitutions and term replacement. i.e.
  From $s \rightarrow_R t$ also $\sigma(s) \rightarrow_R \sigma(t)$ and $u[s]_p \rightarrow_R u[t]_p$
- In particular: $=_R = \stackrel{*}{\longleftrightarrow}_R$

Matching substitution

Definition 9.2. *Let $l, t \in Term_s(F, V)$. A substitution σ is called a match (matching substitution) of l on t, if $\sigma(l) = t$.*

Consequence 9.3. ***Properties***:

- $\forall\ \sigma$ *substitution* $O(l) \subseteq O(\sigma(l))$.
- $\exists \sigma : \sigma(l) = t$ *iff for* σ *defined through*
  $\forall u\ O(l) : l|_u = x \in V \rightsquigarrow u \in O(t) \wedge \sigma(x) = t|_u$
  σ *is a substitution* $\wedge\ \sigma(l) = t$.
- *If there is such a substitution, then it is unique on* $V(l)$. *The existence and if possible calculation are effective.*
- *It is decidable whether* t *is reducible with rule* $l \rightarrow r$.
- *If* R *is finite, then* $\Delta(s) = \{t : s \rightarrow_R t\}$ *is finite and computable.*

Examples

Example 9.4. *Integer numbers*

$$\begin{aligned} sig: 0 &:\longrightarrow int \\ s, p &: int \rightarrow int \\ if0 &: int, int, int \rightarrow int \\ F &: int, int \rightarrow int \end{aligned}$$

$$\begin{aligned} eqns: 1 &:: p(0) = 0 \\ 2 &:: p(s(x)) = x \\ 3 &:: if0(0, x, y) = x \\ 4 &:: if0(s(z), x, y) = y \\ 5 &:: F(x, y) = if0(x, 0, F(p(x), F(x, y))) \end{aligned}$$

Interpretation: $\langle \mathbb{N}, ..., \rangle$ *spec- Algebra with functions*
$O_{\mathbb{N}} = 0, s_{\mathbb{N}} = \lambda n.\ n + 1,$
$p_{\mathbb{N}} = \lambda n.\ if \quad n = 0 \quad then \quad 0 \quad else \quad n - 1 \quad fi$
$if0_{\mathbb{N}} = \lambda i, j, k.\ if \quad i = 0\ then \quad j\ else \quad k \quad fi$
$F_{\mathbb{N}} = \lambda m, n.\ 0$

Orient the equations from left to right $\rightsquigarrow$ *rules R (variable condition is fulfilled).*
Is R terminating? Not with a syntactical ordering, since the left side is contained in the right side.

Example (Cont.)

Reduction sequence:

$$F(s(0),0) \to_5 \underbrace{if0(s(0),0,F(\underbrace{\underbrace{p(s(0))}_{2},\underbrace{F(s(0),0)}_{5}}))}_{4}$$

$$\to_4 \underbrace{F(\underbrace{p(s(0))},\underbrace{F(s(0),0)})}_{5}$$

$$\to_2 \underbrace{F(0,\underbrace{F(s(0),0)})}_{5}$$

$$\to_5 \underbrace{if0(0,0,\underbrace{F(\underbrace{p(0)},\underbrace{F(0,\underbrace{F(s(0),0)})})})}_{3} \to_3 0$$

Equivalence

Definition 9.5. *Let* $spec = (sig, E), spec' = (sig, E')$ *be specifications. They are equivalent in case* $=_E \;=\; =_{E'}$, *i.e..* $T_{spec} = T_{spec'}$.
A rule system R *over* sig *is equivalent to* E, *in case* $=_E \;=\; \stackrel{*}{\longleftrightarrow}_R$.

Notice: If R is finite, convergent, equivalent to E, then $=_E$ is decidable

$$s =_E t \;\; \textit{iff} \;\; s\downarrow = t\downarrow \text{ i.e.. identical NF}$$

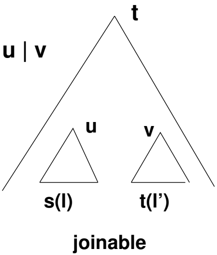
For functional programs and computations in T_{spec} ground convergence is suficient, i.e.. convergence on ground terms.
Problems: Decide whether

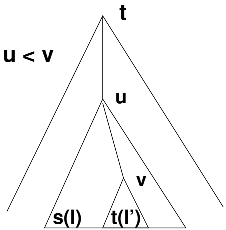
- R noetherian (ground noetherian)
- R confluent (ground confluent)
- How can we transform E in an equivalent R with these properties?

Decidability questions

For finite ground term-rewriting-systems the problems are decidable.

For terminating systems deciding local confluence is sufficient, i.e.. out of $t_1 \leftarrow t \rightarrow t_2$ prove $t_1 \downarrow t_2 \rightsquigarrow$ confluent.





$\rightsquigarrow$ Critical pairs

Critical pairs

Consider the group axioms:

$$\underbrace{(x' \cdot y') \cdot z}_{l_1} \to x' \cdot (y' \cdot z) \quad \text{and} \quad \underbrace{x \cdot x^{-1}}_{l_2} \to 1.$$

"Overlappings" (Superpositions)

$$1 \cdot z \;\overset{l_2}{\swarrow}\; (x \cdot x^{-1}) \cdot z \;\overset{l_1}{\searrow}\; x \cdot (x^{-1} \cdot z) \qquad 1 \;\overset{l_2}{\swarrow}\; (x \cdot y) \cdot (x \cdot y)^{-1} \;\overset{l_1}{\searrow}\; x \cdot (y \cdot (x \cdot y)^{-1})$$

- $l_1|_1$ is "unifiable" with l_2 with substitution $\sigma :: \{x' \leftarrow x, y' \leftarrow x^{-1}, x \leftarrow x\} \rightsquigarrow \sigma(l_1|_1) = \sigma(l_2)$
- l_1 "unifiable" with l_2 with substitution $\sigma :: \{x' \leftarrow x, y' \leftarrow y, z \leftarrow (x \cdot y)^{-1}, x \leftarrow x \cdot y\} \rightsquigarrow \sigma(l_1) = \sigma(l_2)$

Subsumption, unification

Definition 9.6. *Subsumption ordering on terms:*
$s \preceq t$ *iff* $\exists \sigma$ *substitution* $: \sigma(s)$ *subterm of* t
$s \approx t$ *iff* $(s \preceq t \wedge t \preceq s)$
$s \succ t$ *iff* $(t \preceq s \wedge \neg(s \preceq t))$
$\succ$ *is noetherian partial ordering over* $Term(F, V)$ *Proof!.*

Notice:
$O(\sigma(t)) = O(t) \cup \bigcup_{w \in O(t) : t|_w = x \in V} \{wv : v \in O(\sigma(x))\}$

Compatibility properties:
$t|_u = t' \rightsquigarrow \sigma(t)|_u = \sigma(t')$
$t|_u = x \in V \rightsquigarrow \sigma(t)|_{uv} = \sigma(x)|_v \quad (v \in O(\sigma(x)))$
$\sigma(t)[\sigma(t')]_u = \sigma(t[t']_u)$ for $u \in O(t)$

Definition 9.7. $s, t \in Term(F, V)$ *are unifiable iff there is a substitution* σ *with* $\sigma(s) = \sigma(t)$*.* σ *is called a unifier of* s *and* t*.*

Unification, Most General Unifier

Definition 9.8. *Let $V' \subseteq V, \sigma, \tau$ be substitutions.*

- $\sigma \preceq \tau \ (V')$ *iff* $\exists \rho$ *substitution* : $\rho \circ \sigma|_{V'} = \tau|_{V'}$
  Quote: σ *is more general than* τ *over* V'
- $\sigma \approx \tau \ (V')$ *iff* $\sigma \preceq \tau \ (V') \wedge \tau \preceq \sigma \ (V')$
- $\sigma \prec \tau \ (V')$ *iff* $\tau \preceq \sigma \ (V') \wedge \neg(\sigma \preceq \tau \ (V'))$
- *Notice:* $\prec$ *is noetherian partial ordering on the substitutions.*

Question: *Let s, t be unifiable. Is there a most general unifier* $mgu(s,t)$ *over* $V = Var(s) \cup Var(t)$*?*
i.e.. for any unifier σ of s, t always $mgu(s,t) \preceq \sigma \ (V)$ *holds.*

Is $mgu(s,t)$ *unique? (up to variable renaming).*

Unification's problem and its solution

Definition 9.9.

- *A unification's problem is given by a set $E = \{s_i \overset{?}{=} t_i : i = 1, \ldots, n\}$ of equations.*
- *σ is called a solution (or a unifier) in case that $\sigma(s_i) = \sigma(t_i)$ for $i = 1, \ldots, n$.*
- *If $\tau \succeq \sigma\ (Var(E))$ holds for each solution τ of E, then mgu(E) $:= \sigma$ most general solution or most general unifier.*
- *Let $Sol(E)$ be the set of the solutions of E.*
  E and E' are equivalent, if $Sol(E) = Sol(E')$.
- *E' is in solved form, in case that*
  $E' = \{x_j \overset{?}{=} t_j : x_i \neq x_j\ (i \neq j),\ x_i \notin Var(t_j)\ (1 \leq i \leq j \leq m)\}$
- *E' is a solved form for E, iff E' is in solved form and equivalent to E with $Var(E') \subseteq Var(E)$.*

Examples

Example 9.10. *Consider*

- $s = f(x, g(x,a)) \quad \stackrel{?}{=} \quad f(g(y,y), z) = t$
  $\rightsquigarrow x \stackrel{?}{=} g(y,y) \qquad g(x,a) \stackrel{?}{=} z$ *split*
  $\rightsquigarrow x \stackrel{?}{=} g(y,y) \qquad g(g(y,y),a) \stackrel{?}{=} z$ *merge*
  $\rightsquigarrow \sigma :: x \leftarrow g(y,y) \qquad z \leftarrow g(g(y,y),a) \qquad y \leftarrow y$
- $f(x,a) \stackrel{?}{=} g(a,z)$ *unsolvable (not unifiable).*
- $x \stackrel{?}{=} f(x,y)$ *unsolvable, since* $f(x,y)$ *not* x *free.*
- $x \stackrel{?}{=} f(a,y) \rightsquigarrow$ *solution* $\sigma :: x \leftarrow f(a,y)$ *is the most general solution.*

Inference system for the unification

Definition 9.11. *Calculus* ***UNIFY****. Let* $\sigma =$ *be the binding set.*

(1) *Erase*

$$\frac{(E \cup \{s \stackrel{?}{=} s\}, \sigma)}{(E, \sigma)}$$

(2) *Split (Decompose)*

$$\frac{(E \cup \{f(s_1, \ldots, s_m) \stackrel{?}{=} g(t_1, \ldots, t_n)\}, \sigma)}{\lightning \textit{ (unsolvable)}} \quad \textit{if } f \neq g$$

$$\frac{(E \cup \{f(s_1, \ldots, s_m) \stackrel{?}{=} f(t_1, \ldots, t_m)\}, \sigma)}{(E \cup \{s_i \stackrel{?}{=} t_i : i = 1, \ldots, m\}, \sigma)}$$

(3) *Merge (Solve)*

$$\frac{(E \cup \{x \stackrel{?}{=} t\}, \sigma)}{(\tau(E), \sigma \cup \tau)} \quad \textit{if } x \notin Var(t), \tau = \{x \stackrel{?}{=} t\}$$

"occur check"

$$\frac{(E \cup \{x \stackrel{?}{=} t\}, \sigma)}{\lightning \textit{ (unsolvable)}} \quad \textit{if } x \in Var(t) \wedge x \neq t$$

Unification algorithms

Unification algorithms based on UNIFY start always with $(E_0, S_0) := (E, \emptyset)$ and return a sequence $(E_0, S_0) \vdash_{UNIFY} \ldots \vdash_{UNIFY} (E_n, S_n)$
They are successful in case they end with $E_n = \emptyset$, unsuccessful in case they end with $S_n = \lightning$. S_n defines a substitution σ which represents $Sol(S_n)$ and consequently also $Sol(E)$.

Lemma 9.12. *Correctness.*
Each sequence $(E_0, S_0) \vdash_{UNIFY} \ldots \vdash_{UNIFY} (E_n, S_n)$ terminates: either with $\lightning$ (unsolvable, not unifiable) or with $(\emptyset, S)$ and S is a solved form for E.

Notice: Representations in solved form can be quite different (Complexity!!)
$s \stackrel{?}{=} f(x_1, \ldots, x_n) \qquad t \stackrel{?}{=} f(g(x_0, x_0), \ldots, g(x_{n-1}, x_{n-1}))$
$S = \{x_i \stackrel{?}{=} g(x_{i-1}, x_{i-1}) : i = 1, \ldots, n\}$ and
$S_1 = \{x_{i+1} \stackrel{?}{=} t_i : t_0 = g(x_0, x_0), t_{i+1} = g(t_i, t_i) \; i = 0, \ldots, n-1\}$
are both in solved form. The size of t_i grows exponentialy with i.

Example

Example 9.13. *Execution:*

$$f(x, g(a, b)) \stackrel{?}{=} f(g(y, b), x)$$

E_i	S_i	rule
$f(x, g(a, b)) \stackrel{?}{=} f(g(y, b), x)$	$\emptyset$	
$x \stackrel{?}{=} g(y, b), x \stackrel{?}{=} g(a, b)$	$\emptyset$	*split*
$g(y, b) \stackrel{?}{=} g(a, b)$	$x \stackrel{?}{=} g(a, b)$	*solve*
$y \stackrel{?}{=} a, b \stackrel{?}{=} b$	$x \stackrel{?}{=} g(a, b)$	*split*
$b \stackrel{?}{=} b$	$x \stackrel{?}{=} g(a, b), y \stackrel{?}{=} a$	*solve*
	$x \stackrel{?}{=} g(a, b), y \stackrel{?}{=} a$	*delete*

Solution: $mgu = \sigma = \{x \leftarrow g(a, b), y \leftarrow a\}$

Critical pairs - Local confluence

Definition 9.14. *Let R be a rule system and $l_1 \to r_1, l_2 \to r_2 \in R$ with $V(l_1) \cap V(l_2) = \emptyset$ (renaming of variables if necessary, $l_1 \approx l_2$ resp. $l_1 \to r_1 \approx l_2 \to r_2$ are allowed).*

Let $u \in O(l_1)$ with $l_1|_u \notin V$ s.t. $\sigma = mgu(l_1|_u, l_2)$ exists.

$\sigma(l_1)$ is called then a overlap (superposition) of $l_2 \to r_2$ in $l_1 \to r_1$ and $(\sigma(r_1), \sigma(l_1[r_2]_u))$ is the associated critical pair to the overlap $l_1 \to r_1, l_2 \to r_2, u \in O(l_1)$, provided that $\sigma(r_1) \neq \sigma(l_1[r_2]_u)$.

Let $CP(R)$ be the set of all the critical pairs that can be constructed with rules of R.

Notice: The overlaps and consequently the set of critical pairs is unique up to renaming of the variables.

Examples

Example 9.15. *Consider*

- $f(f(\underline{x},\underline{y}),z) \rightarrow f(x,f(y,z))$ $\quad f(\underline{f(x',y')},\underline{z'}) \rightarrow f(x',f(y',z'))$
  unifiable with $x \leftarrow f(x',y'), y \leftarrow z'$

$$f(f(f(x',y'),z'),z)$$
$$\swarrow \qquad \searrow$$
$$t_1 = f(f(x',y'),f(z',z)) \qquad\qquad f(f(x',f(y',z')),z) = t_2$$

- $t = f(x,g(x,a)) \rightarrow h(x)$ $\quad h(x') \rightarrow g(x',x'), t|_1 = t|_{21} = x$
  no critical pairs. Consider variable overlaps:

$$f(h(z),g(h(z),a)))$$
$$\swarrow \qquad \searrow$$
$$t_1 = h(h(z)) \qquad\qquad f(g(z,z),g(h(z),a)) = t_2$$
$$\searrow \qquad\qquad \downarrow$$
$$f(g(z,z),g(g(z,z),a))$$
$$\swarrow$$
$$h(g(z,z))$$

Properties

- Let σ, τ be substitutions, $x \in V$, $\sigma(y) = \tau(y)$ for $y \neq x$ and $\sigma(x) \to_R \tau(x)$. Then for each term t holds:

$$\sigma(t) \xrightarrow{*}_R \tau(t)$$

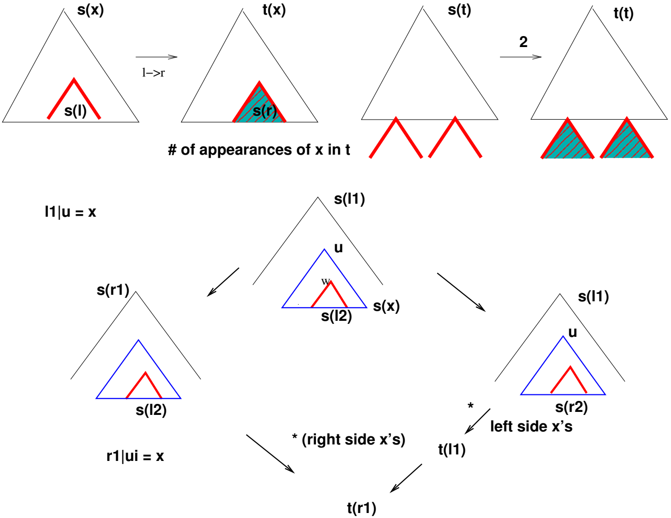
- Let $l_1 \to r_1, l_2 \to r_2$ be rules, $u \in O(l_1), l_1|_u = x \in V$. Let $\sigma(x)|_w = \sigma(l_2)$, i.e.. $\sigma(l_2)$ is introduced by $\sigma(x)$.
  Then $t_1 \downarrow_R t_2$ holds for

$$t_1 := \sigma(r_1) \leftarrow \sigma(l_1) \to \sigma(l_1)[\sigma(r_2)]_{uw} =: t_2$$

Lemma 9.16. *Critical-Pair Lemma of Knuth/Bendix*
Let R be a rule system. Then the following holds:

from $t_1 \leftarrow_R t \to_R t_2$ either $t_1 \downarrow_R t_2$ or $t_1 \leftrightarrow_{CP(R)} t_2$ hold.

Proofs


s(x)
l–>r
s(l)
t(x)
s(r)
s(t)
2
t(t)
# of appearances of x in t
l1|u = x
s(l1)
u
w
s(l2)
s(x)
s(r1)
s(l2)
s(l1)
u
s(r2)
*
left side x's
* (right side x's)
r1|ui = x
t(l1)
t(r1)

Confluence test

Theorem 9.17. *Main result:* *Let R be a rule system.*

- *R is locally confluent iff all the pairs $(t_1, t_2) \in CP(R)$ are joinable.*
- *If R is terminating, then:*
  R confluent iff $(t_1, t_2) \in CP(R) \rightsquigarrow t_1 \downarrow t_2$.
- *Let R be linear (i.e.. for $l, r \in l \to r \in R$ variables appear at most once). If $CP(R) = \emptyset$, then R is confluent.*

Example 9.18.

- *Let $R = \{f(x, x) \to a, f(x, s(x)) \to b, a \to s(a)\}$.*
  R is locally confluent,but not confluent:

$$a \leftarrow f(a, a) \to f(a, s(a)) \to b$$

but not $a \downarrow b$. R is neither terminating nor left-linear.

Example (Cont.)

- $R = \{f(f(x)) \rightarrow g(x)\}$

$$t_1 = g(f(x)) \leftarrow f(f(f(x))) \rightarrow f(g(x)) = t_2$$

  It doesn't hold $t_1 \downarrow_R t_2 \rightsquigarrow R$ not confluent.
  Add rule $t_1 \rightarrow t_2$ to R. R_1 is equivalent to R, terminating and confluent.

$$\begin{array}{ccccc} & & g(f(f(x))) & & \\ & \swarrow & & \searrow & \\ f(g(f(x))) & & & & g(g(x)) \\ & \searrow & & \nearrow & \\ & & f(f(g(x))) & & \end{array}$$

- $R = \{x + 0 \rightarrow x, x + s(y) \rightarrow s(x + y)\}$, linear without critical pairs $\rightsquigarrow$ confluent.
- $R = \{f(x) \rightarrow a, f(x) \rightarrow g(f(x)), g(f(x)) \rightarrow f(h(x)), g(f(x)) \rightarrow b\}$ is locally confluent but not confluent.

Confluence without Termination

Definition 9.19. *$\epsilon - \epsilon$ - Properties.* *Let* $\overset{\epsilon}{\to} \ = \ \overset{0}{\to} \cup \overset{1}{\to}$.

- *R is called $\epsilon - \epsilon$ closed , in case that for each critical pair* $(t_1, t_2) \in CP(R)$ *there exists a t with* $t_1 \overset{\epsilon}{\underset{R}{\to}} t \overset{\epsilon}{\underset{R}{\leftarrow}} t_2$.
- *R is called $\epsilon - \epsilon$ confluent iff* $\underset{R}{\leftarrow} \circ \underset{R}{\to} \ \subseteq \ \overset{\epsilon}{\underset{R}{\to}} \circ \overset{\epsilon}{\underset{R}{\leftarrow}}$

Consequence 9.20.

- $\to$ $\epsilon - \epsilon$ *confluent* $\rightsquigarrow$ $\to$ *strong-confluent.*
- *R $\epsilon - \epsilon$ closed $\not\Rightarrow$ R $\epsilon - \epsilon$ confluent*
  $R = \{f(x,x) \to a, f(x,g(x)) \to b, c \to g(c)\}$. $CP(R) = \emptyset$, *i.e..*
  R $\epsilon - \epsilon$ closed but $a \leftarrow f(c,c) \to f(c,g(c)) \to b$, *i.e.. R not confluent* ↯ .
- *If R is linear and $\epsilon - \epsilon$ closed , then R is strong-confluent, thus confluent (prove that R is $\epsilon - \epsilon$ confluent).*

These conditions are unfortunately too restricting for programming.

Example

Example 9.21. *R left linear $\epsilon - \epsilon$ closed* *is not sufficient:*

$$R = \{f(a,a) \to g(b,b), a \to a', f(a',x) \to f(x,x), f(x,a') \to f(x,x),$$
$$g(b,b) \to f(a,a), b \to b', g(b',x) \to g(x,x), g(x,b') \to g(x,x)\}$$

It holds $f(a',a') \overset{*}{\underset{R}{\longleftrightarrow}} g(b',b')$ *but not* $f(a',a') \downarrow_R g(b',b')$.

R left linear $\epsilon - \epsilon$ closed :

$$\begin{array}{ccccc}
 & & f(a,a) & & \\
 & \swarrow & \downarrow & \searrow & \\
g(b,b) & & f(a',a) & & f(a,a') \\
 & & & & \downarrow \\
\searrow & & \downarrow & \searrow & \swarrow \\
 & & f(a,a) & & f(a',a') \\
 & & & & \downarrow \\
 & & & & f(a',a')
\end{array}$$

.

Parallel reduction

Notice: Let $\rightarrow, \Rightarrow$ with $\xrightarrow{*} = \overset{*}{\Rightarrow}$. (Often: $\rightarrow \subseteq \Rightarrow \subseteq \xrightarrow{*}$).
Then $\rightarrow$ is confluent iff $\Rightarrow$ confluent.

Definition 9.22. *Let R be a rule system.*

- *The parallel reduction, $\mapsto_R$, is defined through $t \mapsto_R t'$ iff $\exists U \subset O(t) : \forall u_i, u_j (u_i \neq u_j \leadsto u_i | u_j)$ $\exists l_i \rightarrow r_i \in R, \sigma_i$ with $t|_{u_i} = \sigma_i(l_i) :: t' = t[\sigma_i(r_i)]_{u_i} (u_i \in U)$ $(t[u_1 \leftarrow \sigma_1(r_1)] \ldots t[u_n \leftarrow \sigma_1(r_n)])$.*
- *A critical pair of $R : (\sigma(r_1), \sigma(l_1[r_2]_u)$ is parallel 0-joinable in case that $\sigma(l_1[r_2]_u) \mapsto_R \sigma(r_1)$.*
- *R is parallel 0-closed in case that each critical pair of R is parallel 0-joinable.*

Properties: $\mapsto_R$ is stable and monotone. It holds $\overset{*}{\mapsto}_R = \xrightarrow{*}_R$ and consequently, if $\mapsto_R$ is confluent then $\rightarrow_R$ too.

Parallel reduction

Theorem 9.23. *If R is left-linear and parallel 0-closed, then $\mapsto_R$ is strong-confluent, thus confluent, and consequently R is also confluent.*

Consequence 9.24.

- *If R fulfills the O'Donnel condition, then R is confluent. O'Donnel's condition: R left-linear, $CP(R) = \emptyset$, R left-sequential (Redexes are unambiguous when reading the terms from left to right: $f(g(x,a),y) \to 0, g(b,c) \to 1$ has not this property).*
  By regrouping of the arguments, the property can frequently be achieved, for instance $f(g(a,x),y) \to 0, g(b,c) \to 1$
- *Orthogonal systems:: R left-linear and $CP(R) = \emptyset$, so R confluent. (In the literature denominated also as regular systems).*
- *Variations: R is strongly-closed, in case that for each critical pair (s,t) there are terms u,v with $s \xrightarrow{*} u \xleftarrow{\leq 1} t$ and $s \xrightarrow{\leq 1} v \xleftarrow{*} t$.*
  R linear and strongly-closed, so R strong-confluent.

Consequences

- Does confluence follow from $CP(R) = \emptyset$? **No**.
  $R = \{f(x,x) \to a, g(x) \to f(x, g(x)), b \to g(b)\}$.
  Consider $g(b) \to f(b, g(b)) \to f(g(b), g(b)) \to a$
  "Outermost" reduction.
  $g(b) \to g(g(b)) \xrightarrow{*} g(a) \to f(a, g(a))$ not joinable.
- Regular systems can be non terminating:
  $\{f(x,b) \to d, a \to b, c \to c\}$. Evidently $CP = \emptyset$.
  $f(c,a) \to f(c,b) \to d$
  $\downarrow *$
  $f(c,a) \to f(c,b)$. Notice that $f(c,a)$ has a normal form. $\rightsquigarrow$
  Reduction strategies that are normalizing or that deliver shortest reduction sequences.
- A context is a term with "holes" $\Box$, e.g. $f(g(\Box, s(0)), \Box, h(\Box))$ as "tree pattern" (pattern) for rule $f(g(x, s(0)), y, h(z)) \to x$. The holes can be filled freely. Sequentiality is defined using this notion.

Termination-Criteria

Theorem 9.25. *R is terminating iff there is a noetherian partial ordering $\succ$ over the ground terms $Term(F)$, that is monotone, so that $\sigma(l) \succ \sigma(r)$ holds for each rule $l \rightarrow r \in R$ and ground substitution σ.*

Proof: $\curvearrowright$ Define $s \succ t$ iff $s \xrightarrow{+} t \quad (s, t \in Term(F))$
$\curvearrowleft$ Asume that $\rightarrow_R$ not terminating, $t_0 \rightarrow t_1 \rightarrow \dots (V(t_i) \subseteq V(t_0))$.
Let σ be a ground substitution with $V(t_0) \subset D(\sigma)$, then
$\sigma(t_0) \succ \sigma(t_1) \succ \dots \lightning$.
Problem: infinite test.

Definition 9.26. *A **reduction ordering** is partial ordering $\succ$ over $Term(F, V)$ with*
(i) $\succ$ is noetherian (ii) $\succ$ is stable and (iii) $\succ$ is monotone.

Theorem 9.27. *R is noetherian iff there exists a reduction ordering $\succ$ with $l \succ r$ for every $l \rightarrow r \in R$*

Termination's criteria

Notice: There are no total reduction orderings for terms with variables..
$x \succ y? \rightsquigarrow \sigma(x) \succ \sigma(y)$
$f(x, y) \succ f(y, x)$? commutativity cannot be oriented.
Examples for reduction orderings:
Knuth-Bendix ordering: Weight for each function symbol and precedence over F.
Recursive path ordering (RPO): precedence over F is recursively extended to paths (words) in the terms that are to be compared.
Lexicographic path ordering(LPO), polynomial interpretations, etc.

$f(f(g(x)))$	$=$	$f(h(x))$	$f(f(x))$	$=$	$g(h(g(x)))$	$f(h(x))$	$=$	$h(g(x$
KB	$\rightarrow$	$l(f) = 3$	$l(g) = 2$	$\rightarrow$	$l(h) =$	1	$\rightarrow$	
RPO	$\leftarrow$	$g > h$	$> f$	$\leftarrow$			$\leftarrow$	

Confluence modulo equivalence relation (e.g. AC):
$R :: f(x, x) \rightarrow g(x) \quad G :: \{(a, b)\} \quad g(a) \leftarrow f(a, a) \sim f(a, b)$ but not $g(a) \downarrow_{\sim} f(a, b)$.

Knuth-Bendix Completion method

Input: E set of equations, $\succ$ reduction ordering, $R = \emptyset$.

Repeat while E not empty
(1) Remove $t = s$ of E with $t \succ s$, $R := R \cup \{t \to s\}$ else abort
(2) Bring the right side of the rules to normal form with R
(3) Extend E with every normalized critical pair generated by $t \to s$ with R
(4) Remove all the rules from R, whose left side is properly larger than t w.r. to the subsumption ordering.
(5) Use R to normalize both sides of equations of E.
Remove identities.

Output: 1) Termination with R convergent, equivalent to E. 2) Abortion 3) not termination (it runs infinitely).

Examples for Knuth-Bendix-Procedure

Example 9.28. ▶ *SRS*:: $\Sigma = \{a, b, c\}, E = \{a^2 = \lambda, b^2 = \lambda, ab = c\}$
$u < v$ *iff* $|u| < |v|$ *or* $|u| = |v|$ *and* $u <_{lex} v$ *with* $a <_{lex} b <_{lex} c$

$E_0 = \{a^2 = \lambda, b^2 = \lambda, ab = c\}, R_0 = \emptyset$

$E_1 = \{b^2 = \lambda, ab = c\}, R_1 = \{a^2 \rightarrow \lambda\}, CP_1 = \emptyset$

$E_2 = \{ab = c\}, R_2 = \{a^2 \rightarrow \lambda, b^2 \rightarrow \lambda\}, CP_2 = \emptyset$

$R_3 = \{a^2 \rightarrow \lambda, b^2 \rightarrow \lambda, ab \rightarrow c\}, NCP_3 = \{(b, ac), (a, cb)\}$
$E_3 = \{b = ac, a = cb\}$

$R_4 = \{a^2 \rightarrow \lambda, b^2 \rightarrow \lambda, ab \rightarrow c, ac \rightarrow b\}, NCP_4 = \emptyset, E_4 = \{a = cb\}$

$R_5 = \{a^2 \rightarrow \lambda, b^2 \rightarrow \lambda, ab \rightarrow c, ac \rightarrow b, cb \rightarrow a\}, NCP_5 = \emptyset, E_5 = \emptyset$

Examples for Knuth-Bendix-Completion

- $E = \{ffg(x) = h(x), ff(x) = x, fh(x) = g(x)\} \quad >: KBO(3, 2, 1)$
  $R_0 = \emptyset, E_0 = E$
  $R_1 = \{ffg(x) \to h(x)\}, KP_1 = \emptyset . E_1 = \{ff(x) = x, fh(x) = g(x)\}$
  $R_2 = \{ffg(x) \to h(x), ff(x) \to x\}, NKP_2 = \{(g(x), h(x))\},$
  $E_2 = \{fh(x) = g(x), g(x) = h(x)\}, R_2 = \{ff(x) \to x\}$
  $R_3 = \{ff(x) \to x, fh(x) \to g(x)\}, NKP_3 = \{(h(x), fg(x))\}, E_3 = \{g(x) = h(x), h(x) = fg(x)\}$
  $R_4 = \{ff(x) \to x, fh(x) \to h(x), g(x) \to h(x)\}, NKP_3 = \emptyset, E_4 = \emptyset$
- $E = \{fgf(x) = gfg(x)\} \quad >: LL :: f > g$
  $R_0 = \emptyset, E_0 = E$
  $R_1 = \{fgf(x) \to gfg(x)\}, NKP_1 = \{(gfggf(x), fggfg(x))\}, E_1 = \{gfggf(x) = fggfg(x)\}$
  $R_1 = \{fgf(x) \to gfg(x), fggfg(x) \to gfggf(x)\}, NKP_2 = \{(gfggfggfg(x), fgggfggfg(x), ..\}...$

Refined Inference system for Completion

Definition 9.29. *Let $>$ be a noetherian PO over $Term(F, V)$. The inference system $\mathcal{P}_{TES}$ is composed by the following rules:*

(1) *Orientate* $\dfrac{(E \cup \{s \doteq t\}, R)}{(E, R \cup \{s \rightarrow t\})}$ *in case that* $s > t$

(2) *Generate* $\dfrac{(E, R)}{(E \cup \{s \doteq t\}, R)}$ *in case that* $s \leftarrow_R \circ \rightarrow_R t$

(3) *Simplify EQ* $\dfrac{(E \cup \{s \doteq t\}, R)}{(E \cup \{u \doteq t\}, R)}$ *in case that* $s \rightarrow_R u$

(4) *Simplify RS* $\dfrac{(E, R \cup \{s \rightarrow t\})}{(E, R \cup \{s \rightarrow u\})}$ *in case that* $t \rightarrow_R u$

(5) *Simplify LS* $\dfrac{(E, R \cup \{s \rightarrow t\})}{(E \cup \{u \doteq t\}, R)}$ *in case that* $s \rightarrow_R u$ *with* $l \rightarrow r$ *and* $s \succ l$ *(SubSumOrd.)*

(6) *Delete identities*

Equational implementations

Programming = Description of algorithms in a formal system

Definition 10.1. *Let* $f : M_1 \times ... \times M_n \rightsquigarrow M_{n+1}$ *be a (partial) function. Let* $T_i, 1 = 1...n+1$ *be decidable sets of ground terms over* Σ*,*
$\hat{f}$ *n-ary function symbol,* E *set of equations.*

A data interpretation $\mathfrak{I}$ *is a function* $\mathfrak{I} : T_i \to M_i$*.*

$\hat{f}$ *implements* f *under the interpretation* $\mathfrak{I}$ *in* E *iff*
1) $\mathfrak{I}(T_i) = M_i \quad (i = 1...n+1)$
2) $f(\mathfrak{I}(t_1), ..., \mathfrak{I}(t_n)) = \mathfrak{I}(t_{n+1})$ *iff* $\hat{f}(t_1, ..., t_n) =_E t_{n+1}$ $(\forall t_i \in T_i)$

$$\begin{array}{ccccc} T_1 \times ... \times T_n & \xrightarrow{\hat{f}} & T_{n+1} \\ \mathfrak{I}\downarrow \quad\quad \mathfrak{I}\downarrow & & \mathfrak{I}\downarrow \\ M_1 \times ... \times M_n & \xrightarrow{f} & M_{n+1} \end{array}$$

Abbreviation: $(\hat{f}, E, \mathfrak{I})$ *implements* f.

Equational implementations

Theorem 10.2. *Let E be set of equations or rules (same notations). For every $i = 1, ..., n+1$ assume*
1) $\mathfrak{I}(T_i) = M_i$
2a) $f(\mathfrak{I}(t_1), ..., \mathfrak{I}(t_n)) = \mathfrak{I}(t_{n+1}) \rightsquigarrow \ \hat{f}(t_1, ..., t_n) =_E t_{n+1}$ $(\forall t_i \in T_i)$

$\hat{f}$ implements the total function f under $\mathfrak{I}$ in E when one of the following conditions holds:

a) $\forall t, t' \in T_{n+1} : t =_E t' \rightsquigarrow \mathfrak{I}(t) = \mathfrak{I}(t')$
b) E confluent and $\forall t \in T_{n+1} : t \rightarrow_E t' \rightsquigarrow t' \in T_{n+1} \wedge \mathfrak{I}(t) = \mathfrak{I}(t')$
c) E confluent and T_{n+1} contains only E-irreducible terms.

Application: Assume $(\hat{f}, E, \mathfrak{I})$ implements the total function f. If E is extended by E_0 under retention of $\mathfrak{I}$, then 1 and 2a still hold. If one of the criteria a, b, c are fullfiled for $E \cup E_0$, then $(\hat{f}, E \cup E_0, \mathfrak{I})$ implements also the function f. This holds specially when $E \cup E_0$ is confluent and T_{n+1} contains only $E \cup E_0$ irreducible terms.

Equational implementations

Theorem 10.3. *Let $(\hat{f}, E, \mathfrak{I})$ implement the (partial) function f. Then*

a) $\forall t, t' \in T_{n+1} :: \mathfrak{I}(t) = \mathfrak{I}(t') \wedge \mathfrak{I}(t) \in Image(f) \rightsquigarrow t =_E t'$
b) Let E be confluent and T_{n+1} contains only normal forms of E. Then $\mathfrak{I}$ is injective on $\{t \in T_{n+1} : \mathfrak{I}(t) \in Image(f)\}$.

Theorem 10.4. *Criterion for the implementation of total functions.*
Assume

1) $\mathfrak{I}(T_i) = M_i \quad (i = 1, \ldots, n+1)$
2) $\forall t, t' \in T_{n+1} :: \mathfrak{I}(t) = \mathfrak{I}(t')$ *iff* $t =_E t'$
3) $\forall_{1 \leq i \leq n} \quad t_i \in T_i \quad \exists t_{n+1} \in T_{n+1} ::$

$$\hat{f}(t_1, \ldots, t_n) =_E t_{n+1} \wedge f(\mathfrak{I}(t_1), \ldots \mathfrak{I}(t_n)) = \mathfrak{I}(t_{n+1})$$

Then $\hat{f}$ implements the function f under $\mathfrak{I}$ in E and f is total.

Notice: If T_{n+1} contains only normal forms and E is confluent, so 2) is fulfilled, in case $\mathfrak{I}$ is injective on T_{n+1}.

Equational implementations

Theorem 10.5. *Let* $(\hat{f}, E, \mathfrak{I})$ *implement* $f : M_1 \times ... \times M_n \to M_{n+1}$. *Let* $S_i = \{t \in T_i :: \exists t_0 \in T_i : t \neq t_0, \mathfrak{I}(t) = \mathfrak{I}(t_0) \;\; t \xrightarrow{+}_E t_0\}$ *be recursive sets.*
Then $\hat{f}$ *implements also* f *with term sets* $T_i' = T_i \setminus S_i$ *under* $\mathfrak{I}|_{T_i'}$ *in* E.

So we can delete terms of T_i that are reducible to other terms of T_i with the same $\mathfrak{I}$-value. Consequently the restriction to E-normal forms is allowed.

Consequence 10.6.

- *Implementations can be composed.*
- *If we extend* E *by* E*- consequences then the implementation property is preserved.*
  This is important for the KB-Completion since only E*-consequences are added.*

Examples: Propositional logic, natural numbers

Example 10.7. *Convention: Equations define the signature. Occasionally variadic functions and overloading. Single sorted.*

Boolean algebra: Let $M = \{true, false\}$ with $\wedge, \vee, \neg, \supset, \dots$.
Constants tt, ff. Term set $Bool := \{tt, ff\}$, $\mathfrak{I}(tt) = true, \mathfrak{I}(ff) = false$.
Strategy: Avoid rules with tt or ff as left side. According to theorem 10.2 c) we can add equations with these restrictions without influencing the implementation property, as long as confluence is achieved.
Consider the following rules:

(1) $cond(tt, x, y) \to x$ (2) $cond(ff, x, y) \to y$. (help function).
(3) $x \ vel \ y \to cond(x, tt, y)$
$E = \{(1), (2), (3)\}$ is confluent. Hence: $tt \ vel \ y =_E cond(tt, tt, y) =_E tt$ holds, i.e.

$$(*_1) \quad tt \ vel \ y = tt \text{ and } \quad (*_2) \quad x \ vel \ tt = cond(x, tt, tt)$$

$x \ vel \ tt = tt$ *cannot* be deduced out of E.
However vel implements the function $\vee$ with E.

Examples: Propositional logic

According to theorem 10.4, we must prove the conditions (1), (2), (3):
$\forall t, t' \in \; Bool \; \exists \bar{t} \in \; Bool :: \mathfrak{I}(t) \vee \mathfrak{I}(t') = \mathfrak{I}(\bar{t}) \wedge t \;\; vel \;\; t' =_E \bar{t}$

For $t = tt$ $(*_1)$ and $t = ff$ (2) since $ff \;\; vel \;\; t' \rightarrow_E cond(ff, tt, t') \rightarrow_E t'$
Thus $x \;\; vel \;\; tt \neq_E \;\; tt$ but $tt \;\; vel \;\; tt =_E \;\; tt, \quad ff \;\; vel \;\; tt =_E \;\; tt$.

MC Carthy's rules for *cond*:

$$(1) \; cond(tt, x, y) = x \quad (2) \; cond(ff, x, y) = y \quad (*) \; cond(x, \; tt, \; tt) = \; tt$$

Notice Not identical with *cond* in Lisp. Difference: Evaluation strategy.
Consider
(**) $cond(x, cond(x, y, z), u) \rightarrow cond(x, y, u)$
$\rightsquigarrow \;\; E' = \{(1), (2), (3), (*), (**)\}$ is terminating and confluent.
Conventions: Sets of equations contain always (1), (2), (3) and
$x \;\; et \;\; y \rightarrow cond(x, y, \;\; ff)$.
Notation: $cond(x, y, z) :: [x \rightarrow y, z]$ or
$[x \rightarrow y_1, x_2 \rightarrow y_2, ..., x_n \rightarrow y_n, z]$ for $[x \rightarrow \;\; [...]..., z]$

Examples: Semantical arguments

Properties of the implementing functions:
$(vel, E, \mathfrak{I})$ implements $\vee$ of BOOL.

Statement: *vel* is associative on *Bool*.
Prove: $\forall t_1, t_2, t_3 \in Bool : t_1 \; vel \; (t_2 \; vel \; t_3) =_E (t_1 \; vel \; t_2) \; vel \; t_3$

There exist $t, t', T, T' \in Bool$ with
$\mathfrak{I}(t_2) \vee \mathfrak{I}(t_3) = \mathfrak{I}(t)$ and $\mathfrak{I}(t_1) \vee \mathfrak{I}(t_2) = \mathfrak{I}(t')$ as well as
$\mathfrak{I}(t_1) \vee \mathfrak{I}(t) = \mathfrak{I}(T)$ and $\mathfrak{I}(t') \vee \mathfrak{I}(t_3) = \mathfrak{I}(T')$

Because of the semantical valid associativity of $\vee$
$\mathfrak{I}(T) = \mathfrak{I}(t_1) \vee \mathfrak{I}(t_2) \vee \mathfrak{I}(t_3) = \mathfrak{I}(T')$ holds.

Since *vel* implements $\vee$ it follows:
$t_1 \; vel \; (t_2 \; vel \; t_3) =_E t_1 \; vel \; t =_E T =_E T' =_E t' \; vel \; t_3 =_E (t_1 \; vel \; t_2) \; vel \; t_3$

Examples: Natural numbers

Function symbols: $\hat{0}, \hat{s}$ Ground terms: $\{\hat{s}^n(\hat{0})\ (n \geq 0)\}$
$\mathfrak{I}$ Interpretation $\mathfrak{I}(\hat{0}) = 0, \mathfrak{I}(\hat{s}) = \lambda x.x + 1$, i.e. $\mathfrak{I}(\hat{s}^n(\hat{0})) = n\ (n \geq 0)$.
Abbreviation: $n \hat{+} 1 := \hat{s}(\hat{n})\ (n \geq 0)$
Number terms. $NAT = \{\hat{n} : n \geq 0\}$ normal forms (Theorem 10.2 c holds).

Important help functions over NAT:

Let $E = \{is\_null(\hat{0}) \rightarrow\ tt, is\_null(\hat{s}(x)) \rightarrow\ ff\}$.
is_null implements the predicate $Is\_Null : \mathbb{N} \rightarrow \{true, false\}$ Zero-test.
Extend E with (non terminating rules)
$\hat{g}(x) \rightarrow [is\_null(x) \rightarrow \hat{0}, \hat{g}(x)]$, $\hat{f}(x) \rightarrow [is\_null(x) \rightarrow \hat{g}(x), \hat{0}]$

Statement:It holds under the standard interpretation $\mathfrak{I}$
$\hat{f}$ implements the null function $f(x) = 0\ \ (x \in \mathbb{N})$ and
$\hat{g}$ implements the function $g(0) = 0$ else undefined.
Because of $\hat{f}(\hat{0}) \rightarrow [is\_null(\hat{0}) \rightarrow \hat{g}(\hat{0}), \hat{0}] \xrightarrow{*} \hat{g}(\hat{0}) \rightarrow [\ldots] \xrightarrow{*} \hat{0}$ and
$\hat{f}(\hat{s}(x)) \rightarrow [is\_null(\hat{s}(x)) \rightarrow \hat{g}(\hat{s}(x)), \hat{0}] \xrightarrow{*} \hat{0}$ (follows from theorem 10.4).

Examples: Natural numbers

Extension of E to E' with rule:

$\hat{f}(x,y) = [is\_null(x) \to y, \hat{0}]$ ($\hat{f}$ overloaded).

$\hat{f}$ implements the function $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$$F(x,y) = \begin{cases} y & x = 0 \\ 0 & x \neq 0 \end{cases} \qquad \begin{array}{l} \hat{f}(\hat{0},\hat{y}) \xrightarrow{*} \hat{y} \\ \hat{f}(\hat{s}(x),\hat{y}) \xrightarrow{*} \hat{0} \end{array}$$

Nevertheless it holds:

$$\hat{f}(x,\hat{g}(x)) =_{E'} [is\_null(x) \to \hat{g}(x), \hat{0}]) =_{E'} \hat{f}(x)$$

But $f(n) = F(n, g(n))$ for $n > 0$ is not true.

If one wants to implement all the computable functions, then the recursion equations of Kleene cannot be directly used, since the composition of partial functions would be needed for it.

Representation of primitive recursive functions

The class $\mathfrak{P}$ contains the functions
$s = \lambda x.x + 1, \pi_i^n = \lambda x_1, \dots, x_n.x_i,$ as well as $c = \lambda x.0$ on $\mathbb{N}$ and is closed w.r. to composition and primitive recursion, i.e.

$$f(x_1, \dots, x_n) = g(h_1(x_1, \dots, x_n), \dots, h_r(x_1, \dots, x_n)) \qquad \text{resp.}$$

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$
$$f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

Statement: $f \in \mathfrak{P}$ is implementable by $(\hat{f}, E_{\hat{f}}, \mathfrak{I})$

Idea: Show for suitable $E_{\hat{f}}$:

$\hat{f}(\hat{k_1}, \dots, \hat{k_n}) \xrightarrow{*}_{E_{\hat{f}}} f(k_1, \widehat{\dots}, k_n)$ with $E_{\hat{f}}$ confluent and terminating.

Assumption: *FUNKT* (signature) contains for every $n \in \mathbb{N}$ a countable number of function symbols of arity n.

Implementation of primitive recursive functions

Theorem 10.8. *For each finite set $A \subset FUNKT \setminus \{\hat{0}, \hat{s}\}$ the exception set, and each function $f : \mathbb{N}^n \to \mathbb{N}, f \in \mathfrak{P}$ there exist $\hat{f} \in FUNKT$ and $E_{\hat{f}}$ finite, confluent and terminating such that $(\hat{f}, E_{\hat{f}}, \mathfrak{I})$ implements f and none of the equations in $E_{\hat{f}}$ contains function symbols from A.*

Proof: Induction over construction of $\mathfrak{P}$: $\hat{0}, \hat{s} \notin A$. Set $A' = A \cup \{\hat{0}, \hat{s}\}$

- $\hat{s}$ implements s with $E_{\hat{s}} = \emptyset$
- $\hat{\pi}_i^n \in FUNKT^n \setminus A'$ implem. π_i^n with $E_{\hat{\pi}_i^n} = \{\hat{\pi}_i^n(x_1, ..., x_n) \to x_i\}$
- $\hat{c} \in FUNKT^1 \setminus A'$ implements c with $E_{\hat{c}} = \{\hat{c}(x) \to 0\}$
- Composition: $[\hat{g}, E_{\hat{g}}, A_0]$, $[\hat{h}_i, E_{\hat{h}_i}, A_i]$ with $A_i = A_{i-1} \cup \{f \in FUNKT : f \in E_{\hat{h}_{i-1}}\} \setminus \{\hat{0}, \hat{s}\}$. Let $\hat{f} \in FUNKT \setminus A'_r$ and $E_{\hat{f}} = E_{\hat{g}} \cup \bigcup_1^r E_{\hat{h}_i} \cup \{\hat{f}(x_1, ..., x_n) \to \hat{g}(\hat{h}_1(...), ..., \hat{h}_r(...))\}$
- Primitive recursion: Analogously with the defining equations.

Implementation of primitive recursive functions

All the rules are left-linear without overlappings $\rightsquigarrow$ confluence.
Termination criteria: Let $\mathfrak{I} : \ FUNKT \rightarrow (\mathbb{N}^* \rightarrow \mathbb{N})$, i.e $\mathfrak{I}(f) : \mathbb{N}^{st(f)} \rightarrow \mathbb{N}$, strictly monotonous in all the arguments. If E is a rule system, $l \rightarrow r \in E, b : VAR \rightarrow \mathbb{N}$ (assignment), if $\mathfrak{I}[b](l) > \mathfrak{I}[b](r)$ holds, then E terminates.
Idea: Use the Ackermann function as bound:
$A(0, y) = y + 1, A(x + 1, 0) = A(x, 1), A(x + 1, y + 1) = A(x, A(x + 1, y))$
A is strictly monotonic,
$A(1, x) = x + 2, A(x, y + 1) \leq A(x + 1, y), A(2, x) = 2x + 3$
For each $n \in \mathbb{N}$ there is a β_n with $\quad \sum_1^n A(x_i, x) \leq A(\beta_n(x_1, ..., x_n), x)$

Define $\mathfrak{I}$ through $\mathfrak{I}(\hat{f})(k_1, ..., k_n) = A(p_{\hat{f}}, \sum k_i)$ with suitable $p_{\hat{f}} \in \mathbb{N}$.

- $p_{\hat{s}} := 1 :: \mathfrak{I}[b](\hat{s}(x)) = A(1, b(x)) = b(x) + 2 > b(x) + 1 = \mathfrak{I}[b](x \hat{+} 1)$
- $p_{\hat{\pi}_i^n} := 1 :: \mathfrak{I}[b](\hat{\pi}_i^n(x_1, ..., x_n)) = A(1, \sum_1^n b(x_i)) > b(x_i) = \mathfrak{I}[b](x_i)$
- $p_{\hat{c}} := 1 :: \mathfrak{I}[b](\hat{c}(x)) = A(1, b(x)) > 0 = \mathfrak{I}[b](\hat{0})$

Implementation of primitive recursive functions

- Composition: $f(x_1, ..., x_n) = g(h_1(...), ..., h_r(...))$.
  Set $c^* = \beta_r(p_{\hat{h}_1}, ..., p_{\hat{h}_r})$ and $p_{\hat{f}} := p_{\hat{g}} + c^* + 2$. Check that
  $\mathfrak{I}[b](\hat{f}(x_1, ..., x_n)) > \mathfrak{I}[b](\hat{g}(\hat{h}_1(x_1, ..., x_n), ..., \hat{h}_r(x_1, ..., x_n)))$
- Primitive recursion:
  Set $m = max(p_{\hat{g}}, p_{\hat{f}})$ and $p_{\hat{f}} := m + 3$. Check that
  $\mathfrak{I}[b](\hat{f}(x_1, ..., x_n, 0)) > \mathfrak{I}[b](\hat{g}(x_1, ..., x_n))$ and
  $\mathfrak{I}[b](\hat{f}(x_1, ..., x_n, \hat{s}(y))) > \mathfrak{I}[b](\hat{g}(....))$.
  Apply $A(m + 3, k + 3) > A(p_{\hat{h}}, k + A(p_{\hat{f}}, k))$
- By induction show that
  $\hat{f}(\hat{k}_1, ..., \hat{k}_n) \to^*_{E_{\hat{f}}} \widehat{f(k_1, ..., k_n)}$
- From the theorem 10.4 the statement follows.

Representation of recursive functions

Minimization:: μ-Operator $\mu_y[g(x_1, ..., x_n, y) = 0] = z$ iff
i) $g(x_1, ..., x_n, i)$ defined $\neq 0$ for $0 \leq i < z$ ii) $g(x_1, ..., x_n, z) = 0$

Regular minimization: μ is applied to total functions for which $\forall x_1, ..., x_n \exists y : g(x_1, ..., x_n, y) = 0$

$\mathfrak{R}$ is closed w.r. to composition, primitive recursion and regular minimization.

Show that: regular minimization is implementable with exception set A.
Assume $\hat{g}, E_{\hat{g}}$ implement g where $\hat{g}(\hat{k}_1, ..., \hat{k}_{n+1}) \xrightarrow{*}_{E_{\hat{g}}} g(k_1, \widehat{...}, k_{n+1})$
Let $\hat{f}, \hat{f}^+, \hat{f}^*$ be new and $E_{\hat{f}} := E_{\hat{g}} \cup \{\hat{f}(x_1, ..., x_n) \to \hat{f}^*(x_1, ..., x_n, \hat{0}),$
$\hat{f}^*(x_1, ..., x_n, y) \to \hat{f}^+(\hat{g}(x_1, ..., x_n, y), x_1, ..., x_n, y),$
$\hat{f}^+(\hat{0}, x_1, ..., x_n, y) \to y, \hat{f}^+(\hat{s}(x), x_1, ..., x_n, y) \to \hat{f}^*(x_1, ..., x_n, \hat{s}(y))\}$

Claim: $(\hat{f}, E_{\hat{f}})$ implements the minimization of g.

Implementation of recursive functions

Assumption: For each $k_1, \dots, k_n \in \mathbb{N}$ there is a smallest $k \in \mathbb{N}$ with $g(k_1, \dots, k_n, k) = 0$

Claim: For every $i \in \mathbb{N}, i \leq k$ $\quad \hat{f}^*(\hat{k}_1, \dots, \hat{k}_n, (k \,\hat{-}\, i)) \xrightarrow{*}_{E_{\hat{f}}} \hat{k}$ holds

Proof: induction over i:

- $i = 0 :: \hat{f}^*(\hat{k}_1, \dots, \hat{k}_n, \hat{k}) \rightarrow \hat{f}^+(\hat{g}(\hat{k}_1, \dots, \hat{k}_n, \hat{k}), \hat{k}_1, \dots, \hat{k}_n, \hat{k}) \xrightarrow{*}_{E_{\hat{g}}}$ $\hat{f}^+(g(k_1, \widehat{\dots}, k_n, k), \hat{k}_1, \dots, \hat{k}_n, \hat{k}) \rightarrow \hat{k}$
- $i > 0 :: \hat{f}^*(\hat{k}_1, \dots, \hat{k}_n, k - (\hat{i} + 1)) \rightarrow$
  $\hat{f}^+(\hat{g}(\hat{k}_1, \dots, \hat{k}_n, k - (\hat{i} + 1)), \hat{k}_1, \dots, \hat{k}_n, k - (\hat{i} + 1)) \xrightarrow{*}_{E_{\hat{g}}}$
  $\hat{f}^+(\hat{s}(\hat{x}), \hat{k}_1, \dots, \hat{k}_n, k - (\hat{i} + 1)) \rightarrow \hat{f}^*(\hat{k}_1, \dots, \hat{k}_n, \hat{s}(k - (\hat{i} + 1))) =$
  $\hat{f}^*(\hat{k}_1, \dots, \hat{k}_n, k \,\hat{-}\, i)) \xrightarrow{*}_{E_{\hat{g}}} \hat{k}$
  For appropiate x and Induction hypothesis.
- $E_{\hat{f}}$ is confluent and according to Theorem 10.4, $(\hat{f}, E_{\hat{f}})$ implements the total function f.
- $E_{\hat{f}}$ is not terminating. $g(k, m) = \delta_{k,m} \rightsquigarrow \hat{f}^*(\hat{k}, k \,\hat{+}\, 1)$ leads to NT-chain. Termination is achievable!.

Representation of partial recursive functions

Problem: Recursion equations (Kleene's normal form) cannot be directly used. Arguments must have "number" as value. (See example). Some arguments can be saved:

Example 10.9.
$f(x, y) = g(h_1(x, y), h_2(x, y), h_3(x, y))$. Let g, h_1, h_2, h_3 be implementable by sets of equations as partial functions.

Claim: f is implementable. Let $\hat{f}, \hat{f}_1, \hat{f}_2$ be new and set:

$$\hat{f}(x, y) = \hat{f}_1(\hat{h}_1(x, y), \hat{h}_2(x, y), \hat{h}_3(x, y), \hat{f}_2(\hat{h}_1(x, y)), \hat{f}_2(\hat{h}_2(x, y)), \hat{f}_2(\hat{h}_3(x, y)))$$

$$\hat{f}_1(x_1, x_2, x_3, \hat{0}, \hat{0}, \hat{0}) = \hat{g}(x_1, x_2, x_3), \quad \hat{f}_2(\hat{0}) = \hat{0}, \quad \hat{f}_2(\hat{s}(x)) = \hat{f}_2(x)$$

$(\hat{f}, E_{\hat{g}}, E_{\hat{h}_1}, E_{\hat{h}_2}, E_{\hat{h}_3} \cup \ REST)$ implements f.

Theorem 10.4 cannot be applied!!.

$(\hat{f}, E_{\hat{g}}, E_{\hat{h}_1}, E_{\hat{h}_2}, E_{\hat{h}_3} \cup \ REST)$ implements f.

Apply definition 10.1:

$\curvearrowright$ For number-terms let $f(\Im(t_1), \Im(t_2)) = \Im(t)$. There are number-terms T_i $(i = 1, 2, 3)$ with
$g(\Im(T_1), \Im(T_2), \Im(T_3)) = \Im(t)$ and $h_i(\Im(t_1), \Im(t_2)) = \Im(T_i)$.
Assumption: $\hat{g}(T_1, T_2, T_3) =_{E_{\hat{f}}} t$ and $\hat{h}_i(t_1, t_2) =_{E_{\hat{f}}} T_i (i = 1, 2, 3)$. The T_i are number-terms:: $\hat{f}_2(T_i) =_{E_{\hat{f}}} \hat{0}$ i.e. $\hat{f}_2(\hat{h}_i(t_1, t_2)) =_{E_{\hat{f}}} \hat{0}$ $(i = 1, 2, 3)$.
Hence
$\hat{f}(t_1, t_2) =_{E_{\hat{f}}} \hat{f}_1(T_1, T_2, T_3, \hat{0}, \hat{0}, \hat{0}) \rightsquigarrow \hat{f}(t_1, t_2) =_{E_{\hat{f}}} t (=_{E_{\hat{f}}} \hat{g}(T_1, T_2, T_3))$

$\curvearrowleft$ For number-terms t_1, t_2, t let $\hat{f}(t_1, t_2) =_{E_{\hat{f}}} t$, so
$\hat{f}_1(\hat{h}_1(t_1, t_2), \hat{h}_2(t_1, t_2), \hat{h}_3(t_1, t_2), \hat{f}_2(\hat{h}_1(t_1, t_2), ...) =_{E_{\hat{f}}} t$. If for an $i = 1, 2, 3$ $\quad \hat{f}_2(\hat{h}_i(t_1, t_2))$ would not be $E_{\hat{f}}$ equal to $\hat{0}$, then the $E_{\hat{f}}$ equivalence class contains only $\hat{f}_1$ terms. So there are number-terms T_1, T_2, T_3 with $\hat{h}_i(t_1, t_2) =_{E_{\hat{f}}} = T_i$ $(i = 1, 2, 3)$ (Otherwise only $\hat{f}_2$ terms equivalent to $\hat{f}_2(\hat{h}_i(t_1, t_2)$). From Assumption:
$\rightsquigarrow h_i(\Im(T_1), \Im(T_2)) = \Im(T_i), \qquad g(\Im(T_1), \Im(T_2), \Im(T_3)) = \Im(t)$

$\mathfrak{R}_p$ and normalized register machines

Definition 10.10. *Program terms for RM: P_n $(n \in \mathbb{N})$ Let $0 \leq i \leq n$*
Function symbols: a_i, s_i constants , $\circ$ binary , W^i unary
Intended interpretation:
a_i :: Increase in one the value of the contents on register i.
s_i :: Decrease in one the value of the contents on register i.$(\dot{-}1)$
$\circ(M_1, M_2)$ :: Concatenation $M_1 M_2$ (First M_1, then M_2)
$W^i(M)$ :: While contents of register i not 0, execute M Abbr.: $(M)_i$

Note: $P_n \subseteq P_m$ for $n \leq m$

Semantics through partial functions: $M_e : P_n \times \mathbb{N}^n \to \mathbb{N}^n$

- $M_e(a_i, \langle x_1, \dots, x_n \rangle) = \langle \dots x_{i-1}, x_i + 1, x_{i+1} \dots \rangle$ $(s_i :: x_i \dot{-} 1)$
- $M_e(M_1 M_2, \langle x_1, \dots, x_n \rangle) = M_e(M_2, M_e(M_1, \langle x_1, \dots, x_n \rangle))$
- $M_e((M)_i, \langle x_1, \dots, x_n \rangle) = \begin{cases} \langle x_1, \dots, x_n \rangle & x_i = 0 \\ M_e((M)_i, M_e(M, \langle x_1, \dots, x_n \rangle)) & \text{otherwise} \end{cases}$

Implementation of normalized register machines

Lemma 10.11. *M_e can be implemented by a system of equations.*

Proof: Let tup_n be n-ary function symbol. For $t_i \in \mathbb{N}$ $(0 < i \leq n)$ let $\langle t_1, ..., t_n \rangle$ be the interpretation for $tup_n(\hat{t}_1, ..., \hat{t}_n)$. Program terms are interpreted by themselves (since they are terms). For $m \geq n$ ::

P_n	$tup_m(\hat{t}_1, ..., \hat{t}_m)$	syntactical level
$\mathfrak{I} \downarrow$	$\mathfrak{I} \downarrow$	
P_n	$\langle t_1, ..., t_m \rangle$	Interpretation

Let *eval* be a binary function symbol for the implementation of M_e and $i \leq n$. Define $E_n := \{$

$eval(a_i,\ tup_n(x_1, ..., x_n)) \longrightarrow\ tup_n(x_1, ..., x_{i-1}, \hat{s}(x_i), x_{i+1}, ..., x_n)$

$eval(s_i,\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1} ...)) \longrightarrow\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1} ...)$

$eval(s_i,\ tup_n(..., x_{i-1}, \hat{s}(x), x_{i+1} ...)) \longrightarrow\ tup_n(..., x_{i-1}, x, x_{i+1} ...)$

$eval(x_1 x_2, t) \longrightarrow\ eval(x_2,\ eval(x_1, t))$

$eval((x)_i,\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1} ...)) \longrightarrow\ tup_n(..., x_{i-1}, \hat{0}, x_{i+1} ...)$

$eval((x)_i,\ tup_n(..., x_{i-1}, \hat{s}(y), x_{i+1} ...) \longrightarrow$

$eval((x)_i, eval(x,\ tup_n(..., x_{i-1}, \hat{s}(y), x_{i+1} ...))\}$

$(eval, E_n, \mathfrak{I})$ implements M_e

Consider program terms that contain at most registers with $1 \leq i \leq n$.

- E_n is confluent (left-linear, without critical pairs).
- Theorem 10.4 not applicable, since M_e is not total.
  Prove conditions of the Definition 10.1.

(1) $\mathfrak{I}(T_i) = M_i$ according to the definition.
(2) $M_e(p, \langle k_1, ..., k_n\rangle) = \langle m_1, ..., m_n\rangle$ iff
$eval(p,\ tup_n(\hat{k}_1, ..., \hat{k}_n)) =_{E_n}\ tup_n(\hat{m}_1, ..., \hat{m}_n)$
$\curvearrowright$ out of the def. of M_e res. E_n. induction on construction of p.
$\curvearrowleft$ Structural induction on p ::
1. $p = a_i(s_i)$ :: $\hat{k}_j = \hat{m}_j (j \neq i), \hat{s}(\hat{k}_i) = \hat{m}_i$ res. $\hat{k}_i = \hat{m}_i = \hat{0}$
$(\hat{k}_i = \hat{s}(\hat{m}_i))$ for s_i
2.Let $p = p_1 p_2$ and
$eval(p_2, eval(p_1,\ tup_n(\hat{k}_1, ..., \hat{k}_n))) \xrightarrow{*}_{E_n}\ tup_n(\hat{m}_1, ..., \hat{m}_n)$
Because of the rules in E_n it holds:

$(eval, E_n, \mathfrak{I})$ implements M_e

There are $i_1, \dots, i_n \in \mathbb{N}$ with $eval(p_1, tup_n(\hat{k}_1, \dots, \hat{k}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{i}_1, \dots, \hat{i}_n)$
hence
$eval(p_2, tup_n(\hat{i}_1, \dots, \hat{i}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{m}_1, \dots, \hat{m}_n)$
According to the induction hypothesis (2-times) the statement holds.
3. Let $p = (p_1)_i$. Then:
$eval((p_1)_i, tup_n(\hat{k}_1, \dots, \hat{k}_n)) \xrightarrow{*}_{E_n} tup_n(\hat{m}_1, \dots, \hat{m}_n)$
There exists a finite sequence $(t_j)_{1 \leq j \leq l}$ with
$t_1 = eval((p_1)_i, tup_n(\hat{k}_1, \dots, \hat{k}_n)), \quad t_j \to t_{j+1}, \quad t_l = tup_n(\hat{m}_1, \dots, \hat{m}_n)$
There exists subsequence $(T_j)_{1 \leq j \leq m}$ of form $eval((p_1)_i, tup_n(\hat{i}_{1,j}, \dots, \hat{i}_{n,j}))$
For T_m $i_{i,m} = 0$ holds, i.e. $i_{1,m} = m_1, \dots, i_{i,m} = 0 = m_i, \dots, i_{n,m} = m_n$.
For $j < m$ always $i_{i,j} \neq 0$ holds and
$eval(p_1, tup_n(\hat{i}_{1,j}, \dots, \hat{i}_{n,j}) \xrightarrow{*}_{E_n} tup_n(\hat{i}_{1,j+1}, \dots, \hat{i}_{n,j+1})$.
The induction hypothesis gives:
$M_e(p_1, \langle i_{1,j}, \dots, i_{n,j} \rangle) = \langle i_{1,j+1}, \dots, i_{n,j+1} \rangle$ for $j = 1, \dots, m$.
But then $M_e((p_1)_i, \langle i_{1,j}, \dots, i_{n,j} \rangle) = \langle m_1, \dots, m_n \rangle \quad (1 \leq j < m)$

Implementation of $\mathfrak{R}_p$

For $f \in \mathfrak{R}_p^{n,1}$ there are $r \in \mathbb{N}$, program term p with at most r-registers $(n+1 \leq r)$, so that for every $k_1, ..., k_n, k \in \mathbb{N}$ holds:
$f(k_1, ..., k_n) = k$ iff $\forall m \geq 0$

$$eval(p, tup_{r+m}(\hat{k}_1, ..., \hat{k}_n, \hat{0}, \hat{0}, ..., \hat{0}, \hat{x}_1, ..., \hat{x}_m)) =_{E_{r+m}} tup_{r+m}(\hat{k}_1, ..., \hat{k}_n, \hat{k}, \hat{0}, ..., \hat{0}, \hat{x}_1, ..., \hat{x}_m) \quad \text{iff}$$

$$eval(p, tup_r(\hat{k}_1, ..., \hat{k}_n, \hat{0}, \hat{0}, ..., \hat{0})) =_{E_r} tup_r(\hat{k}_1, ..., \hat{k}_n, \hat{k}, \hat{0}, ..., \hat{0})$$

Note: $E_r \sqsubset E_{r+m}$ via $tup_r(...) \blacktriangleright tup_{r+m}(..., \hat{0}, ..., \hat{0})$.

Let $\hat{f}, \hat{R}$ be new function symbols, p program for f. Extend E_r by
$\hat{f}(y_1, ..., y_n) \to \hat{R}(eval(p, tup_r(y_1, ..., y_n), \hat{0}, ..., \hat{0}))$ and
$\hat{R}(tup_r(y_1, ..., y_r)) = y_{n+1}$ to $E_{ext(f)}$.

Theorem 10.12. *$f \in \mathfrak{R}_p^{n,1}$ is implemented by $(\hat{f}, E_{ext(f)}, \mathfrak{I})$.*

Non computable functions

Let E be recursive, T_i recursive. Then the predicate

$$P(t_1, ..., t_n, t_{n+1}) \text{ iff } \hat{f}(t_1, ..., t_n) =_E t_{n+1}$$

is a r.a. predicate on $T_1 \times ... \times T_n \times T_{n+1}$

If the function $\hat{f}$ implements f, then P represents the graph of the function $f \rightsquigarrow \quad f \in \mathfrak{R}_p$.

Kleene's normal form theorem:

$f(x_1, ..., x_n) = U(\mu_y[T_n(p, x_1, ..., x_n, y) = 0])$

Let h be the total non recursive function, defined by:

$$h(x) = \begin{cases} \mu_y[T_1(x, x, y) = 0] & \text{in case that } \exists y : T_1(x, x, y) = 0 \\ 0 & \text{otherwise} \end{cases}$$

h is uniquely defined through the following predicate:

(1) $(T_1(x, x, y) = 0 \wedge \forall z(z < y \rightsquigarrow T_1(x, x, z) \neq 0)) \rightsquigarrow h(x) = y$

(2) $(\forall z(z < y \wedge T_1(x, x, z) \neq 0)) \rightsquigarrow (h(x) = 0 \vee h(x) \geq y)$

If $h(x)$ is replaced by u, then these are prim. rec. predicates in x, y, u.

Non computable functions

There are primitive recursive functions P_1, P_2 in x, y, u, so that

$$(1`) \quad P_1(x, y, h(x)) = 0 \text{ and } (2') \quad P_2(x, y, h(x)) = 0$$

represent (1) and (2).
Hence there are an equational system E and function symbols $\hat{P}_1, \hat{P}_2$, that implement P_1, P_2 under the standard interpretation.
(As prim. rec. functions in the Var. x, y, u)
Let $\hat{h}$ be fresh. Add to E the equations

$$\hat{P}_1(x, y, \hat{h}(x)) = \hat{0} \quad \text{and} \quad \hat{P}_2(x, y, \hat{h}(x)) = \hat{0}.$$

The equational system is consistent (there are models) and $\hat{h}$ is interpreted by the function h on the natural numbers.$\rightsquigarrow$
It is possible to specify non recursive functions implicitly with a finite set of equations, in case arbitrary models are accepted as interpretations.
Through non recursive sets of equations any function can be implemented by a confluent, terminating ground system :
$E = \{\hat{h}(\hat{t}) = \hat{t}' : t, t' \in \mathbb{N}, h(t) = t'\}$ (Rule application is not effective).

Computable algebras

Definition 10.13.

- *A sig-Algebra $\mathfrak{A}$ is recursive (effective, computable), if the base sets are recursive and all operations are recursive functions.*
- *A specification spec $= (sig, E)$ is recursive, if T_{spec} is recursive.*

Example 10.14. *Let* $sig = (\{nat, even\}, odd :\rightarrow even, 0 :\rightarrow nat, s : nat \rightarrow nat, red : nat \rightarrow even)$.

As sig-Algebra $\mathfrak{A}$ choose: $A_{even} = \{2n : n \in \mathbb{N}\} \cup \{1\}, A_{nat} = \mathbb{N}$ *with* odd *as* 1, red *as* $\lambda x.if\ x\ even\ then\ x\ else\ 1$, s *successor*

Claim: There is no finite (init-Algebra) specification for $\mathfrak{A}$

- *No equations of the sort nat.*
- $odd, red(s^n(0)), red(s^n(x))$ $(n \geq 0)$ *terms of sort even. No equations of the form* $red(s^n(x)) = red(s^m(x)$ $(n \neq m)$ *are possible.*
- *Infinite number of ground equations are needed.*

Computable algebras

Solution: Enrichment of the signature with:
$even: \; nat \rightarrow \; nat$ and $cond: \; nat \; even \; even \rightarrow \; even$ with interpretation

$$\lambda x.\ if \;\; x \;\; even \;\; then \; 0 \;\; else \; 1, \qquad \lambda x, y, z.\ if \;\; x = 0 \;\; then \;\; y \;\; else \;\; z$$

Equations:
$even(0) = 0, \quad even(s(0)) = s(0), \quad even(s(s(x)) = even(x)$
$cond(0, y, z) = y, \quad cond(s(x), y, z) = z$
$red(x) = cond(even(x), red(x), odd)$

Alternative: Conditional equations:
$red(s(0)) = odd, \;\; red(s(s(x)) = odd \;\; if \;\; red(x) = odd$

Conditional equational systems (term replacement systems) are more "expressive" as pure equational systems. They also define reduction relations. Confluence and termination criteria can be derived. Negated equations in the conditions lead to problems with the initial semantics (non Horn-clause specifications).

Computable algebras: Results

Theorem 10.15. *Let $\mathfrak{A}$ be a recursive term generated sig- Algebra. Then there is a finite enrichment sig' of sig and a finite specification $spec' = (sig', E)$ with $T_{spec'}|_{sig} \cong \mathfrak{A}$.*

Theorem 10.16. *Let $\mathfrak{A}$ be a term generated sig- Algebra. Then there are equivalent:*

- *$\mathfrak{A}$ is recursive.*
- *There is a finite enrichment (without new sorts) sig' of sig and a finite convergent rule system R, so that $\mathfrak{A} \cong T_{spec'}|_{sig}$ for $spec' = (sig', R)$*

See Bergstra, Tucker: Characterization of Computable Data Types (Math. Center Amsterdam 79).

Attention: Does not hold for signatures with only unary function symbols.

Reduction strategies for replacement systems

Main implementation problems for functional programming languages.
Which reduction strategies guarantee the calculation of normal forms, in case these exist. Let R be TES, $t \in term(\Sigma)$.
Assuming that there is $\bar{t}$ irreducible with $t \xrightarrow{*}_R \bar{t}$.

- Which choice of the redexes guarantees a "computation" of $\bar{t}$?
- Which choice of the redexes delivers the "shortest" derivation sequence?
- Let R be terminating. Is there a reduction strategy that delivers always the shortest derivation sequence? How much does it cost?

For $SKI-$calculus and $\lambda-$calculus the Left-Most-Outermost strategy (normal strategy) is normalizing, i.e. calculates a normal form of a term if it exists. It doesn't deliver the shortest derivation sequences. Though it holds: If $t \xrightarrow{k} \bar{t}$ is a shortest derivation sequence, then $t \xrightarrow{\leqq 2^k}_{LMOM} \bar{t}$. By using structure-sharing-methods, the bounds for LMOM can be lowered.

Functional computability models

- Partial recursive functions (Basic functions + Operators)
- Term rewriting systems (Algebraic Specification)
- λ-Calculus and Combinator Calculus
- Graph replacement Systems (Implementation + efficiency)

Central Notion: Application:

Expressions represent (denote) functions.
Application of functions on functions $\rightsquigarrow$ Self application problem

See e.g. Barendregt: Functional Programming and λ-Calculus Handbook of Theoretical Computer Science.

λ-Calculus und Combinator Calculus: Informal

Basic operations:

- Application:: For "expressions" F, A:: $F.A$ or (FA)
  F as program term is "applied" on A as argument term.
- Abstraction:: For an "expression" M, Variable x :: $\lambda x.M$
  Denotes a function which maps x into M, M can "depend"on x .
- Example: $(\lambda x.2 * x + 1).3$ should give as result $2 * 3 + 1$, hence 7.
- β-Equation:: $(\lambda x.M[x])N = M[x := N]$
  "Free" occurrences of x in M are "replaced" by N. β-Conversion

  $$(yx(\lambda x.x))[x := N] \equiv (yN(\lambda x.x))$$

  Notice: Free occurrences of variables in N remain free.
  Renaming of (bound) variables if necessary

  $$(\lambda x.y)[y := xx] \equiv \lambda z.xx\ z \text{ "new"}$$

λ-Calculus und Combinator Calculus: Informal

- α-Equation:: $\lambda x.M = \lambda y.M[x := y]$ with y "new"
  $\lambda x.x = \lambda y.y$. Same effect as "Functions" α-Conversion
- Set of λ- terms in C and V::
  $$\Lambda(C, V) = C|V|(\Lambda\Lambda)|(\lambda V.\Lambda)$$
- Set of free variables of M:: $FV(M)$
- M is closed (Combinator) if $FV(M) = \emptyset$
- Standard Combinators:: $I \equiv \lambda x.x$ $\quad K \equiv \lambda xy.x \equiv \lambda x.(\lambda y.x)$
  $B \equiv \lambda xyz.x(yz)$ $\quad K_* \equiv \lambda xy.y$ $\quad S \equiv \lambda xyz.xz(yz)$
- Following equalities hold:
  $IM = M \quad KMN = M \quad K_*MN = N \quad SMNL = ML(NL)$
  $BLMN = L(M(N))$ left parenthesis !
- Fixpoint Theorem:: $\forall F \exists X \quad FX = X$ with e.g.
  $X \equiv WW$ and $W \equiv \lambda x.F(xx)$

λ-Calculus und Combinator Calculus: Informal

- Representation of functions, numbers $c_n \equiv \lambda fx.f^n(x)$
  F combinator represents f iff $Fz_{n1}...z_{nk} = z_{f(n1,...,nk)}$
- f is partial recursive iff f is represented by a combinator.
- Theorem of Scott: Let $A \subset \Lambda, A$ non trivial and closed under $=$, then A not recursively decidable.
- β-Reduction:: $(\lambda x.M)N \rightarrow_\beta M[x := N]$
- NF = Set of terms which have a normal form is not recursive.
- $(\lambda x.xx)y$ is not in normal form, yy is in normal form.
- $(\lambda x.xx)(\lambda x.xx)$ has no normal form.
- Church Rosser Theorem:: $\rightarrow_\beta$ ist confluent
- Theorem of Curry If M has a normal form then $M \rightarrow_l^* N$, i.e. Leftmost Reduction is normalizing.

Reduction strategies for replacement systems

Definition 11.1. *Let R be a TES.*

- *A one-step reduction strategy* $\mathfrak{S}$ *for R is a mapping* $\mathfrak{S} : term(R, V) \to term(R, V)$ *with* $t = \mathfrak{S}(t)$ *in case that t is in normal form and* $t \to_R \mathfrak{S}(t)$ *otherwise.*
- $\mathfrak{S}$ *is a multiple-step-reduction strategy for R if* $t = \mathfrak{S}(t)$ *in case that t is in normal form and* $t \xrightarrow{+}_R \mathfrak{S}(t)$ *otherwise.*
- *A reduction strategy* $\mathfrak{S}$ *is called normalizing for R, if for each term t with a R- normal form, the sequence* $(\mathfrak{S}^n(t))_{n\geq 0}$ *contains a normal form. (Contains in particular a finite number of terms).*
- *A reduction strategy* $\mathfrak{S}$ *is called cofinal for R, if for each t and* $r \in \Delta^*(t)$ *there is a* $n \in \mathbb{N}$ *with* $r \xrightarrow{*}_R \mathfrak{S}^n(t)$.

Cofinal reduction strategies are optimal in the following sense: they deliver maximal information gain.

Assuming that normal forms contain always maximal information.

Known reduction strategies

Definition 11.2. *Reduction strategies:*

- *Leftmost-Innermost (Call-by-Value). One-step-RS, the redex that appears most left in the term and that contains no proper redex is reduced.*
- *Paralell-Innermost. Multiple-step-RS.* $PI(t) = \bar{t}$, *at which* $t \mapsto \bar{t}$ *(All the innermost redexes are reduced).*
- *Leftmost-Outermost (Call-by-Name). One-step-RS.*
- *Parallel-Outermost. Multiple-step-RS.* $PO(t) = \bar{t}$, *at which* $t \mapsto \bar{t}$ *(All the disjoint outermost redexes are reduced).*
- *Fair-LMOM. A left-most outermost redex in a red-sequence is eventually reduced. (A LMOR in such a strategy doesn't remain unreduced for ever). (Lazy strategy).*

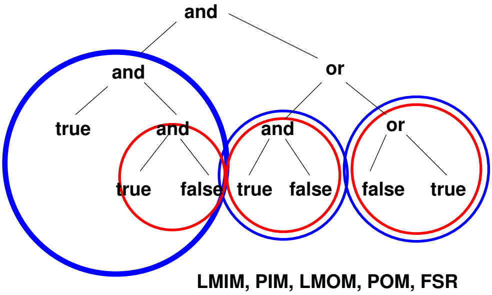
Known reduction strategies

- Full-substitution-rule. (Only for orthogonal systems).
  Multiple-step-RS. $GK(t) :: t \xrightarrow{+} GK(t)$ all the redexes in t are reduced, in case they're not disjunct, then the residuals of the redexes are also reduced.
- Call-By-Need. One-step-RS. It reduces always a necessary redex. A redex in t is necessary, when it must be reduced in order to compute the normal form. (Only for certain TES e.g. LMOM for SKI calculus)
  Problem: How can one decide whether a redex is necessary or not?
- Variable-Delay-Strategy: One-step-RS. Reduce redex, that doesn't appear as redex in the instance of a variable of another redex.

Examples

Example 11.3. *:*

- $and(true, x) \to x,\ and(false, x) \to false,$
  $or(true, x) \to true,\ or(false, x) \to x$
  Orthogonal, strong left sequential (constants "before" the variables).



Examples

- $\Sigma = \{0, s, p, if0, F\}, R = \{p(0) \to 0, p(s(x)) \to x, if0(0, x, y) \to x, if0(s(z), x, y) \to y, F(x, y) \to if0(x, 0, F(p(x), F(x, y)))\}$
  Left-linear, without overlaps. (orthogonal).
  $F(0,0) \to if0(0, 0, F(p(0), F(0,0))) \overset{OM}{\to} 0$
  $\downarrow PIM$
  $if0(0, 0, F(0, if0(0, 0, F(p(0), F(0,0)))))$
  No IM-strategy is for all orthogonal systems normalizing or cofinal.
- FSR (Full-Substitution-Rule): Choose all the redexes in the term and reduce them from innermost to outermost (notice no redex is destroyed). Cofinal for orthogonal systems.
- $\Sigma = \{a, b, c, d_i : i \in \mathbb{N}\}$
  $R := \{a \to b, d_k(x) \to d_{k+1}(x), c(d_k(b)) \to b$
  confluent (left linear parallel 0-closed).
  $c(d_0(a)) \to_1 c(d_1(a)) \to_1 \ldots$. not normalizing (POM).
  $c(d_0(a)) \to_{1,1} c(d_0(b)) \to_0 b$

Examples

- $\Sigma = \{a, b_i, c, d : i \in \mathbb{N}\}$. Non confluent SRS:
  $R = \{ab_0c \to acb_0, ab_0d \to ad, c \to d, cb_i \to d, b_i \to b_{i+1}(i \geq 1)\}$
  $ab_0c \to_{11} ab_0d \to ad$
  $ab_0c \to_0 acb_0 \to_{11} acb_1 \to adb_1 \to \ldots$
- $\Sigma = \{f, a, b, c, d\}$ $R = \{f(x, b) \to d, a \to b, c \to c\}$ Orthogonal.
  LMOM must not be normalizing:
  $f(c, a) \to f(c, a) \to ....$ but $f(c, a) \to f(c, b) \to d$
- $f(a, f(x, y)) \to f(x, f(x, f(b, b)))$ left linear with overlaps.
  $f(a, f(a, f(b, b))) \to_{OUT} f(a, f(a, f(b, b))) \to_{OUT} ....$
  $\downarrow INN$
  $f(a, f(b, f(b, f(b, b)))) \to f(b, f(b, f(b, b)))$
- $R = \{f(g(x), c) \to h(x, d), b \to c\}$
  $f(g(f(a, f(a, \underline{b}))), c) \to_{VD} h(f(a, f(a, \underline{b})), d) \to_{VD}$
  $h(f(a, f(a, c)), d)$

Strategies for orthogonal systems

Theorem 11.4. *For orthogonal systems the following holds:*

- *Full-Substitution-Rule is a cofinal reduction strategy.*
- *POM is a normalizing reduction strategy.*
- *LMOM is normalizing for λ-calculus and CL-calculus.*
- *Every fair-outermost strategy is normalizing.*
- ***Main tools**:*
  Elementary reduction diagrams,*residuals* and *reduction diagrams*

$$
\begin{array}{ccc}
Sab(Ic) & \to & a(Ic)(b(Ic)) \\
\downarrow & & \downarrow \\
 & & ac(b(Ic)) \\
 & & \downarrow \\
Sabc & \to & ac(bc)
\end{array}
\qquad
\begin{array}{ccc}
Ka(Ib) & \to & Kab \\
\downarrow & & \downarrow \\
 & & \\
 & & \\
a & \to_{\emptyset} & a
\end{array}
$$

$$
\begin{array}{ccc}
Ia & \to & a \\
\downarrow_{\emptyset} & & \downarrow_{\emptyset} \\
Ia & \to & a
\end{array}
\qquad
\begin{array}{ccc}
Ia & \to & a \\
\downarrow & & \downarrow_{\emptyset} \\
a & \to_{\emptyset} & a
\end{array}
\qquad
\begin{array}{ccc}
a & \to & a \\
 & & \\
a & & a
\end{array}
$$

Composition of E-reduction diagrams

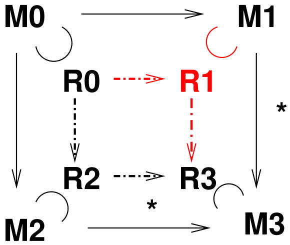
Reduction diagrams and projections:

$$
\begin{array}{ccccccccc}
t_0 & \to & t_1 & \to & \dots & \to & t_n & & \\
\downarrow & & \downarrow * & \xrightarrow{*} & \xrightarrow{*} & & \downarrow * & & \\
t_1' & \xrightarrow{*} & & \xrightarrow{*} & \xrightarrow{*} & & \downarrow * & & \\
\downarrow & & \downarrow * & \xrightarrow{*} & \xrightarrow{*} & & \downarrow * & R_4 = R_2 / R_1 & \\
\downarrow & & \downarrow * & \xrightarrow{*} & \xrightarrow{*} & & \downarrow * & & \\
\vdots & \vdots & \dots & \dots & & & \downarrow * & & \\
\downarrow & & \downarrow * & \xrightarrow{*} & \xrightarrow{*} & & \downarrow * & & \\
t_m' & \xrightarrow{*} & & \xrightarrow{*} & & \xrightarrow{*} & \downarrow * & & \\
 & & & & R_3 = R_1 / R_2 & & & \text{projections} &
\end{array}
$$

Let $R_1 :: t \xrightarrow{+} t'$ and $R_2 :: t \xrightarrow{+} t'$ be two reduction sequences of r from t to t'. They are equivalent $\quad R_1 \cong R_2 \quad$ iff $\quad R_1 / R_2 = R_2 / R_1 = \emptyset$.

Strategies for orthogonal systems

Lemma 11.5. *Let D be an elementary reduction diagram for orthogonal systems, $R_i \subseteq M_i$ $(i = 0, 2, 3)$ redexes with $R_0 - . - . \rightarrow R_2 - . - . \xrightarrow{*} R_3$ i.e. R_2 is residual of R_0 and R_3 is residual of R_2. Then there is a unique redex $R_1 \subseteq M_1$ with $R_0 - . - . \rightarrow R_1 - . - . \xrightarrow{*} R_3$, i.e.*



***Notice**, that in the reduction sequences $M_1 \xrightarrow{*} M_3$ and $M_2 \xrightarrow{*} M_3$ only residuals of the corresponding used redex in the reduction in M_0 are reduced.*

Property of elementary reduction diagrams!

Strategies for orthogonal systems

Definition 11.6. *Let Π be a predicate over term pairs M, R so that $R \subseteq M$ and R is redex (e.g. LMOM, LMIM,...).*

i) Π has property I when for a D like in the lemma it holds:

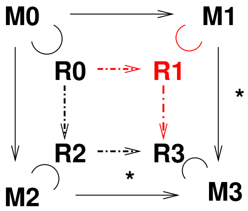
$$\Pi(M_0, R_0) \wedge \Pi(M_2, R_2) \wedge \Pi(M_3, R_3) \rightsquigarrow \Pi(M_1, R_1)$$

ii) Π has property II if in each reduction step $M \rightarrow^{R} M'$ with $\neg\Pi(M, R)$, each redex $S' \subseteq M'$ with $\Pi(M', S')$ has an ancestor-redex $S \subseteq M$ with $\Pi(M, S)$. (i.e. $\neg\Pi$ steps introduce no new Π-redexes).

Lemma 11.7. *Separability of developments. Assume Π has property II. Then each development $\mathfrak{R} :: M_0 \rightarrow \ldots \rightarrow M_n$ can be partitioned in a Π-part followed by a $\neg\Pi$-part.*
More precisely: There are reduction sequences
$\mathfrak{R}_\Pi :: M_0 = N_0 \rightarrow^{R_0} \ldots \rightarrow^{R_{k-1}} N_k$ with $\Pi(N_i, R_i)$ $(i < k)$ and $\mathfrak{R}_{\neg\Pi} :: N_k \rightarrow^{R_k} \ldots \rightarrow^{R_{k+l-1}} N_{k+l}$ with $\neg\Pi(N_j, R_j)$ $(k \leq j < k+l)$ and $\mathfrak{R}$ is equivalent to $\mathfrak{R}_\Pi \times \mathfrak{R}_{\neg\Pi}$.

Example 11.8.

- $\Pi(M,R)$ *iff* R *is redex in* M. *I and II hold.*
- $\Pi(M,R)$ *iff* R *is an outermost redex in* M. *Then properties I and II hold: To I*



R_0, R_2, R_3 *outermost redexes*
Let S_i *be the redex in* $M_0 \to M_i$
Assuming that is not OM $\leadsto$ *In* M_1 *a redex* (P) *is generated by the reduction of* S_1, *that contains* R_1.

In $M_1 \to\!> M_3$ R_1 *becomes again outermost. i.e.* P *is reduced: But in* $M_1 \to\!> M_3$ *only residuals of* S_2 *are reduced and* P *is not residual, since was newly introduced.* ↯ . *II is clear.*

- $\Pi(M,R)$ *iff* R *is left-most redex in* M. *I holds. II not always:* $F(x,b) \to d, a \to b, c \to c :: F(c,a) \to F(c,b)$

Descendants of redexes (residuals)

Definition 11.9. *Traces in reduction sequences:*

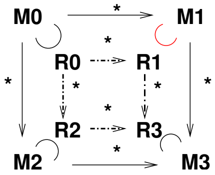
- *Let* $\mathfrak{R} :: M_0 \rightarrow M_1 \rightarrow \ldots$ *be a reduction sequence. Let* M_j *be fixed and* $L_i \subseteq M_i$ $(i \geq j)$ *(provided that* M_i *exists) redexes with* $L_j -.-.\rightarrow L_{j+1} -.-.\rightarrow \ldots.$
  The sequence $\mathfrak{L} = (L_{j+i})_{i \geq 0}$ *is a trace of descendants (residuals) of redexes in* M_j.
- $\mathfrak{L}$ *is called* Π*-trace, in case that* $\forall i \geq j \ \ \Pi(M_i, L_i)$.
- *Let* R *be a reduction sequence,* Π *a predicate.* R *is* Π*-fair, if* R *has no infinite* Π*-Traces.*

Results from Bergstra, Klop :: Conditional Rewrite Rules: Confluence and Termination. JCSS 32 (1986)

Properties of Traces

Lemma 11.10. *Let Π be a predicate with property I.*

- *Let $\mathfrak{D}$ be a reduction diagram with $R_i \subseteq M_i, R_0 -.-.\rightarrow> R_2 -.-.\rightarrow> R_3$ is Π trace.*



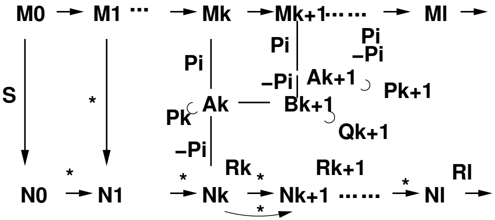
Then $R_0 -.-.\rightarrow> R_1 -.-.\rightarrow> R_3$ via M_1 also a Π trace

- *Let $\mathfrak{R}, \mathfrak{R}'$ be equivalent reduction sequences from M_0 to M. $S \subseteq M_0, S' \subseteq M$ redexes, so that a Π-trace $S -.-.\rightarrow> S'$ via $\mathfrak{R}$ exists. Then there is a unique Π-trace $S -.-.\rightarrow> S'$ via $\mathfrak{R}'$.*

Main Theorem of O'Donnell 77

Theorem 11.11. *Let Π be a predicate with properties I,II. Then the class of Π-fair reduction sequences is closed w.r. to projections.*

Proof Idea:



Let $\mathfrak{R} :: M_0 \to \ldots$ be Π-fair and $\mathfrak{R}' :: N_0 \xrightarrow{*}$ a projection.
$\forall k \exists M_k \xrightarrow{\Pi}> A_k \xrightarrow{\neg\Pi}> N_k$ equivalent to the complete development $M_k \to> N_k$. In the resulting rearrangement both derivations between N_k and N_{k+1} are equivalent. In particular the Π-Traces remain the same.
Results in an echelon form: $A_k - B_{k+1} - A_{k+1} - B_{k+2} - \ldots$.

Main Theorem: Proof

This echelon reaches $\mathfrak{R}$ after a finite number of steps, let's say in M_l::
If not $\mathfrak{R}$ would have an infinite trace of S residuals with property Π.

Let's assume that $\mathfrak{R}'$ is not Π fair. Hence it contains an infinite Π -trace $R_k, ..., R_{k+1} ...$ that starts from N_k.

There are Π-ancestors $P_k \subseteq A_k$ from the Π-redex $R_k \subseteq N_k$, i.e with $\Pi(A_k, P_k)$. Then the Π-trace $P_k - . - . \rightarrow> R_k - . - . \rightarrow> R_{k+1}$ can be lifted via B_{k+1} to the Π-trace $P_k - . - . \rightarrow> Q_{k+1} - . - . \rightarrow> R_{k+1}$.

Iterating this construction until M_l, a redex P_l that is predecessor of R_l with $\Pi(M_l, P_l)$ is obtained. This argument can be now continued with R_{l+1}.
Consequently $\mathfrak{R}$ is not Π-fair. $\lightning$.

Consequences

Lemma 11.12. *Let $\mathfrak{R} :: M_0 \to M_1 \to \ldots$ be an infinite sequence of reductions with infinitely outermost redex-reductions. Let $S \subseteq M_0$ be a redex. Then $\mathfrak{R}' = \mathfrak{R}/\{S\}$ is also infinite.*

Proof: Assume that $\mathfrak{R}'$ is finite with length k. Let $l \geq k$ and R_l be the redex in the reduction of $M_l \to M_{l+1}$ and let $\mathfrak{R}_l$ the reduction sequence from M_l to M_l'

• If R_l is outermost, then $M_l' \xrightarrow{*} M_{l+1}'$ can only be empty if R_l is one of the residuals of S which are reduced in $\mathfrak{R}_l$. Thus $\mathfrak{R}_{l+1}$ has one step less than $\mathfrak{R}_l$.

• Otherwise R_l is properly contained in the residual of S reduced in $\mathfrak{R}_l$.

However given that $\mathfrak{R}$ must contain infinitely many outermost redex-reductions then $\mathfrak{R}_q$ would become empty. Consequently $\mathfrak{R}'$ must coincide with $\mathfrak{R}$ from some position on, hence it is also infinite.

Consequences for orthogonal systems

Theorem 11.13. *Let $\Pi(M, R)$ iff R is outermost redex in M.*

- *The fair outermost reduction sequences are terminating, when they start from a term which has a normal form.*
- *Parallel-Outermost is normalizing for orthogonal systems.*

Proof: If t has a normal form, then there is no infinite Π-fair reduction sequence that starts with t.
Let $\mathfrak{R} :: t \to t_1 \to .... \to$ be an infinite Π-fair and $\mathfrak{R}' :: t \to t'_1 \to ... \to \bar{t}$ a normal form.
$\mathfrak{R}$ contains infinitely many outermost reduction steps (otherwise it would not Π-fair). Then $\mathfrak{R}/\mathfrak{R}'$ also infinite. $\lightning$.

Observe that: The theorem doesn't hold for LMOM-strategy: property II is not fulfilled. Consider for this purpose $a \to b, c \to c, f(x, b) \to d$.

Consequences for orthogonal systems

Definition 11.14. *Let R be orthogonal, $l \to r \in R$ is called left normal, if in l all the function symbols appear left of the variables. R is left normal, if all the rules in R are left normal.*

Consequence 11.15. *Let R be left normal. Then the following holds:*

- *Fair leftmost reduction sequences are terminating for terms with a normal form.*
- *The LMOM-strategy is normalizing.*

Proof: Let $\Pi(M, L)$ iff L is LMO-redex in M. Then the properties I and II hold. For II left normal is needed.
According to theorem 11.11 the Π-fair reduction sequences are closed under projections. From Lemma 11.12 the statement follows.

Summary

A strategy is called perpetual if it can induce infinite reduction sequences.

Strategy	Orthogonal	LN-Ortogonal	Orthogonal-NE
LMIM	*p*	*p*	*p n*
PIM	*p*	*p*	*p n*
LMOM		*n*	*p n*
POM	*n*	*n*	*p n*
FSR	*n c*	*n c*	*p n c*

Classification of TES according to appearances of variables

Definition 11.16. *Let R be TES, $Var(r) \subseteq Var(l)$ for $l \rightarrow r \in R, x \in Var(l)$.*

- *R is called variable reducing, if for every $l \rightarrow r \in R, |l|_x > |r|_x$*
  R is called variable preserving, if for every $l \rightarrow r \in R, |l|_x = |r|_x$
  R is called variable augmenting, if for every $l \rightarrow r \in R, |l|_x \leq |r|_x$
- *Let $D[t, t']$ be a derivation from t to t'. Let $|D[t, t']|$ the length of the reduction sequence. $D[t, t']$ is optimal if it has the minimal length among all the derivations from t to t'.*

Lemma 11.17. *Let R be orthogonal, variable preserving. Then every redex remains in each reduction sequence, unless it is reduced. Each derivation sequence is optimal.*

Proof: Exchange technique: residuals remain as residuals, as long as they are not reduced, i.e. the reduction steps can be interchanged.

Examples

Example 11.18. *Lengths of derivations:*

- *Variable preserving:*
  $R :: f(x,y) \to g(h(x),y)), g(x,y) \to l(x,y), a \to c, b \to d.$
  Consider the term $f(a,b)$ and its derivations.
  All derivation sequences to the normal form are of the same length (4).
- *Variable augmenting (non erasing):*
  $R :: f(x,b) \to g(x,x), a \to b, c \to d.$ *Consider the term $f(c,a)$ and its derivations.*
  Innermost derivation sequences are shorter than the outermost ones.

Further Results

Lemma 11.19. *Let R be overlap free, variable augmenting. Then an innermost redex remains until it is reduced.*

Theorem 11.20. *Let R be orthogonal variable augmenting (ne). Let $D[t, t']$ be a derivation sequence from t to its normal form t', which is non-innermost. Then there is an innermost derivation $D'[t, t']$ with $|D'| \leq |D|$.*

Proof: Let $L(D)$ = derivation length from the first non-innermost reduction in D to t'.
Induction over $L(D) :: t \to t_1 \to \ldots \to t_i \xrightarrow{S} \ldots \to t_j \xrightarrow{*} t'$.
Let i be this position.
S is non-innermost in t_i, hence it contains an innermost redex S_i that must be reduced later on, let's say in the reduction of t_j. Consider the reduction sequence $D' :: t \to t_1 \to \ldots \to t_i \xrightarrow{S_i} t'_{i+1} \xrightarrow{S} \ldots t'_j \overset{<}{\xrightarrow{*}} t'$
$|D'| \leq |D|, L(D') < L(D)$ $\rightsquigarrow$ there is a derivation D' with $L(D') = 0$.

Further Results

Theorem 11.21. *Let R be overlap free, variable augmenting. Every two innermost derivations to a normal form are equally long.*

Sure! given that innermost redexes are disjoint and remain preserved as long as they are not reduced.
Consequence:Let R be left linear, variable augmenting. Then innermost derivations are optimal. Especially LMIM is optimal.

Example 11.22. *If there are several outermost redexes, then the length of the derivation sequences depend on the choice of the redexes. Consider:*
$f(x, c) \to d, a \to d, b \to c$ *and the derivations:*
$f(\underline{a}, b) \to f(d, \underline{b}) \to \underline{f(d, c)} \to d$ *and respectively* $f(a, \underline{b}) \to \underline{f(a, c)} \to d$
$\leadsto$ *variable delay strategy. If an outermost redex after a reduction step is no longer outermost, then it is located below a variable of a redex originated in the reduction. If this rule deletes this variable, then the redex must not be reduced.*

Further Results

Theorem 11.23. *Let R be overlap free.*

- *Let D be an outermost derivation and L a non-variable outermost redex in D. Then L remains a non-variable outermost redex until it is reduced.*
- *Let R be linear. For each outermost derivation $D[t, t']$, t' normal form, exists a variable delaying derivation $D'[t, t']$ with $|D'| \leq |D|$. Consequently the variable delaying derivations are optimal.*

Theorem 11.24. *Ke Li. The following problem is NP-complete:*

Input: A convergent TES R, term t and $D[t, t\downarrow]$.
Question: Is there a derivation $D'[t, t\downarrow]$ with $|D'| < |D|$.

Proof Idea: Reduce 3-SAT to this problem.

Computable Strategies

Definition 11.25. *A reduction strategy $\mathfrak{S}$ is computable, if the mapping $\mathfrak{S} : Term \to Term$ with $t \xrightarrow{*} \mathfrak{S}(t)$ is recursive.*

Observe that: The strategies LMIM, PIM, LMOM, POM, FSR are polynomially computable.

Question: Is there a one-step computable normalizing strategy for orthogonal systems ?.

Example 11.26.

- *(Berry) CL-calculus extended by rules $FABx \to C, FBxA \to C, FxAB \to C$ is orthogonal, non-left-normal. Which argument does one choose for the reduction of FMNL? Each argument can be evaluated to A resp. B, however this is undecidable in CL.*
- *Consider $or(true, x) \to\ true, or(x,\ true) \to true + CL$. Parallel evaluation seems to be necessary!*

Computable Strategies: Counterexample

Example 11.27. *Signature: Constants:* $S, K, S', K', C, 0, 1$
unary: $A, activate$ *binary:* ap, ap' *ternary:* B

Rules:

$ap(ap(ap(S,x),y),z) \to ap(ap(x,y),ap(y,z))$
$ap(ap(K,x),y) \to x$
$activate(S') \to S, \quad activate(K') \to K$
$activate(ap'(x,y)) \to ap(activate(x), activate(y))$
$A(x) \to B(0,x,activate(x)), \quad A(x) \to B(1,x,activate(x))$
$B(0,x,S) \to C, \quad B(1,x,K) \to C, \quad B(x,y,z) \to A(y)$

Terms: *Starting with terms of form* $A(t)$ *where* t *is constructed from* S', K' *and* ap'.

Claim: *R is confluent and has no computable one step strategy which is normalizing.*

A sequential Strategy for paror Systems

Example 11.28. *Let $f, g : \mathbb{N}^+ \to \mathbb{N}$ recursive functions. Define a "term rewriting system" R on $\mathbb{N} \times \mathbb{N}$ with rules:*

- $(x, y) \to (f(x), y)$ *if* $x, y > 0$
- $(x, y) \to (x, g(y))$ *if* $x, y > 0$
- $(x, 0) \to (0, 0)$ *if* $x > 0$
- $(0, y) \to (0, 0)$ *if* $y > 0$

Obviously R is confluent. Unique normal form is $(0, 0)$ and for $x, y > 0$,

$$(x, y) \text{ has a normal form iff } \exists n.\ f^n(x) = 0 \lor g^n(x) = 0.$$

A one step reductions strategy must choose among the application of f res. g in the first res. second argument.
Such a reduction strategy cannot compute first the zeros of $f^n(x)$ res. $g^n(y)$ in order to choose the corresponding argument. One could expect, that there are appropriate functions f and g for which no computable one step strategy exists. But this is not the case!!

A sequential strategy for paror systems

There exists a computable one step reduction strategy which is normalizing.

Lemma 11.29. *Let* $(x, y) \in \mathbb{N} \times \mathbb{N}$. *Then:*

- $x < y$*:: For n either* $f^n(x) = 0$ *or* $f^n(x) \geq y$ *or there exists an* $i < n$ *with* $f^n(x) = f^i(x) \neq 0$ *holds. Choose n minimal with this property. The three alternatives are mutually excluding.*
  If one of the first two holds then $\mathfrak{S}(x, y) = L$ *else R*
- $x \geq y$*:: For n either* $g^n(y) = 0$ *or* $g^n(y) > x$ *or there exists an* $i < n$ *with* $g^n(y) = g^i(y) \neq 0$*. Choose n minimal with this property. The three alternatives are mutually excluding. If one of the first two holds then* $\mathfrak{S}(x, y) = R$ *else L*
- *Claim:* $\mathfrak{S}$ *is a computable one step reduction strategy for R which is normalizing. (Proof: Exercise)*

Computable Strategies

Theorem 11.30. *Kennaway (Annals of Pure and Applied Logic 43(89))*
For each orthogonal system there is a computable sequential (one step) normalising reduction strategy.

Definition 11.31. *Standard reduction sequences*
Let $\mathfrak{R} :: t_0 \to t_1 \to \ldots$ be a reduction sequence in the TES R. Mark in each step in $\mathfrak{R}$ all top-symbols of redexes that appear on the left side of the reduced redex. $\mathfrak{R}$ is a standard reduction sequence if no redex with marked top-symbol is ever reduced.

Theorem 11.32.
Standardization theorem for left-normal orthogonal TES.
Let R be LNO.
If $t \xrightarrow{*} s$ holds, then there exists a standard reduction sequence in R with $t \xrightarrow{*}_{ST} s$.
Especially LMOM is normalizing.

Sequential Orthogonal TES

Example 11.33. *For applicative TES::* $PxQ \to xx, R \to S, Ix \to x$
Consider $\mathfrak{R} :: PR(\underline{IQ}) \to \underline{PRQ} \to \underline{R}R \to SR$
There exists no standard reduction sequence from $PR(IQ)$ *to* SR

Fact: λ-Calculus and CL-Calculus are sequential, i.e. always needed redexes are reduced for computing the normal form.

Definition 11.34. *Let* R *be orthogonal,* $t \in Term(R)$ *with normal form* $t\downarrow$. *A redex* $s \subseteq t$ *is a* ***needed*** *redex, if in every reduction sequence* $t \to \ldots \to t\downarrow$ *some residual of* s *is reduced (contracted).*

Sequential Orthogonal TES: Call-by-need

Theorem 11.35. *Huet- Levy (1979) Let R be orthogonal*

- *Let t with a normal form but reducible , then t contains a needed redex*
- *"Call-by-need" Strategy (needed redexes are contracted) is normalizing*
- *Fair needed-redex reduction sequences are terminating for terms with a normal form.*

Lemma 11.36. *Let R be orthogonal, $t \in Term(R)$, s, s' redexes in t s.t. $s \subseteq s'$. If s is needed, then also s' is.*
In particular:: If t is not in normal form, then a outermost redex is a needed redex.

Let $C[\dots,\dots,\dots]$ be a context with n-places (holes), σ a substitution of the redexes $s_1, \dots, s_n$ in places $1, \dots, n$. The Lemma implies the following property:
$\forall C[\dots,\dots,\dots]$ in normal form, $\forall \sigma \exists i.s_i$ needed in $C[s_1, \dots, s_n]$.
Which one of the s_i is needed, depends on σ .

Sequential Orthogonal TES

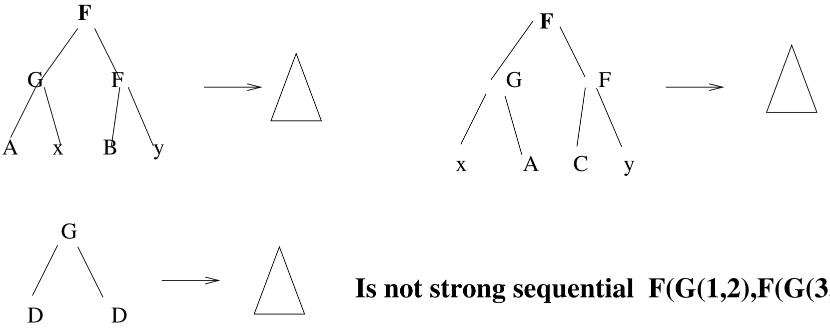
Definition 11.37. *Let R be orthogonal.*

- *R is sequential* iff $\forall C[..., ..., ...]$ in normal form $\exists i \forall \sigma . s_i$ is needed in $C[s_1, ..., s_n]$*
  Unfortunately this property is undecidable
- *Let $C[...]$ context. The reduction relation $\rightarrow_?$ (possible reduction) is defined by*

  $$C[s] \rightarrow_? C[r] \text{ for each redex } s \text{ and arbitrary term } r$$

  $\rightarrow_?^*$ and residuals defined in analogy.
- *A redex s in t is called **strongly needed** if in every reduction sequence $t \rightarrow_? \ldots \rightarrow_? t'$, where t' is a normal form, some descendant of s gets reduced.*
- *R is **strongly sequential** if $\forall C[..., ..., ..]$ in normal form $\exists i \forall \sigma . s_i$ is strongly needed.*

Example



Is not strong sequential F(G(1,2),F(G(3,4),5)

Strong Sequentiality

Lemma 11.38. *Let R be orthogonal.*

- *The property of being strongly sequential is decidable. The needed index i is computable.*
  Proof: See e.g. Huet-Levy
- *Call-by-need is a computable one step reduction strategy for such systems.*

Summary: Formal Specification and Verification Techniques

- What have we learned? $\rightsquigarrow$ See contents of lecture.
- Which were the important notions about FSVT?
- Are formal methods helpful for better software development?
- Can formal methods be integrated in SD-Process models?
- What is needed in order to understand and use formal methods?
- Are there criteria for evaluating formal methods?
- The importance of knowing what one does....

Principles to make a formal method a useful tool in system development

- formal syntax
- formal semantics
- clear conceptual system model
- uniform notion of an interface
- sufficient expressiveness and descriptive power
- concept of development techniques with a proper notion of refinement and implementation

Model oriented specification techniques

- ASM
- VDM
- Z and B-Methods
- SDL
- STATECHARTS
- CSP, Petri-Nets (concurrent)
- ....

Property oriented specification techniques

- Algebraic Specification Techniques (equational logic)
- Logical Specification Techniques (Prolog, temporal- and modal logics)
- Hybrids
- LARCH, OBJ, MAUDE,....
- Tools: http://rewriting.loria.fr/
- ....

Interesting reading:
http://www.comp.lancs.ac.uk/computing/resources/IanS/SE6/Slides/PDF/ch9.
http://libra.msra.cn/ConferenceDetail.aspx?id=1618

Verification techniques

Important: What and where should something hold...
What to do when it does not hold?
Use the proper tools depending on the abstraction level.

- Equational Logic (Term Rewriting ...)
- Equational properties in a single model (Induction methods....)
- First order Logics (General theorem provers...)
- First order properties of single models (Inductive methods...)
- Temporal and modal logics (Propositional part...Model checking)
- Propositional logics (Sat solvers, Davis Putman, tableaux,...)

FSVT

- **Thanks for your attention**