This part focuses on Operational Semantics of formal calculi (and programming languages).

### Resources

- **Reference Books:**
  - R. AMADIO: Operational methods in semantics (available on HAL: https://hal.archives-ouvertes.fr/cel-01422101v1).
  - D. SANGIORGI: Introduction to Bisimulation and Coinduction (Cambridge University Press, 2011)

- **Lecture Notes** (by Middeldorp, Laurent, Ong)

  Please send me an email (with LMFI in the subject) to have the lecture notes on Rewriting Theory.

### Topics

- **Tools to study the operational properties of a system:**
  - Rewrite Theory (rewriting=abstract form of program execution)
  - Induction and Co-induction proof principles.
  - Linear Logic and Proof-Nets.

- **Bridging between lambda-calculus and functional programming:**
  - Call-by-Value and Call-by Name, weak and lazy calculi.
  - Big-Step and Small-Step operational semantics.
  - Observational equivalence

- **Reasoning on programs equivalence:**
  - Bisimulation and coinductive methods.

- **Beyond pure functional:**
  - Probabilistic programming and Bayesian Inference: Probabilistic lambda calculi, Bayesian Proof-Nets

### Rewriting theory

- **Rewriting = abstract form of program execution**
- **Paradigmatic example: λ-calculus** (functional programming language, in its essence)

### Example (Group Theory)

A colony of chameleons includes 20 red, 18 blue, and 16 green individuals. Whenever two chameleons of different color meet, each changes to the third color. Some time passes during which no chameleons are born or die nor do any enter or leave the colony. Is it possible that at the end of this period, all 54 chameleons are the same color?
Abstract Rewriting: motivations

- Concrete rewrite formalisms / concrete operational semantics:
  - $\lambda$-calculus
  - Quantum/ probabilistic/ non-deterministic/... $\lambda$-calculus
  - Proof-nets / graph rewriting
  - Sequent calculus and cut-elimination
  - String rewriting
  - Term rewriting
  - Abstract rewriting
  - Independent from structure of objects that are rewritten
  - Uniform presentation of properties and proofs
Definition 1.1.1. An abstract rewrite system (ARS for short) is a pair $\mathcal{A} = (A, \rightarrow)$ consisting of a set $A$ and a binary relation $\rightarrow$ on $A$. Instead of $(a,b) \in \rightarrow$ we write $a \rightarrow b$ and we say that $a \rightarrow b$ is a rewrite step.

A (finite) rewrite sequence is a non-empty sequence $(a_0, \ldots, a_n)$ of elements in $A$ such that $a_0 \rightarrow a_1 \rightarrow \ldots \rightarrow a_n$. We write $a_0 \rightarrow^* a_n$ or simply $a_0 \rightarrow^* a_n$.

- A rewrite sequence
  - finite: $a \rightarrow e \rightarrow b \rightarrow c \rightarrow f$
  - empty: $\emptyset$
  - infinite: $a \rightarrow e \rightarrow b \rightarrow a \rightarrow b \rightarrow \ldots$

**ARS**

Definition 1.1.1. An abstract rewrite system (ARS for short) is a pair $\mathcal{A} = (A, \rightarrow)$ consisting of a set $A$ and a binary relation $\rightarrow$ on $A$. Instead of $(a,b) \in \rightarrow$ we write $a \rightarrow b$ and we say that $a \rightarrow b$ is a rewrite step.

We denote $\rightarrow^*$ (resp. $\rightarrow^*$) the transitive-reflexive (resp. reflexive) closure of $\rightarrow$.

We denote $\leftrightarrow$ the reverse relation of $\rightarrow$, i.e. $u \leftrightarrow t$ if $t \rightarrow u$.

If $\rightarrow_1, \rightarrow_2$ are binary relations on $A$ then $\rightarrow_1 \cdot \rightarrow_2$ denotes their composition, i.e. $f \rightarrow_1 \cdot \rightarrow_2 g$ if there exists $u \in A$ such that $f \rightarrow_1 u \rightarrow_2 g$.

We write $(A, \{\rightarrow_1, \rightarrow_2\})$ to denote the ARS $(A, \rightarrow)$ where $\rightarrow = \rightarrow_1 \cup \rightarrow_2$.

**Composition**

The transitive-reflexive closure of a relation is a closure operator, i.e. it satisfies:

- $a \rightarrow^* a$;
- $(a \rightarrow^*)^* = a \rightarrow^*$;
- $a \rightarrow^* b$ implies $a \rightarrow^* c$ for $c \rightarrow^* b$.

As a consequence:

- $(a \cup b)^* = (a^* \cup b^*)^*$.

**Closure**

Normal forms = result

Definition 1.1.1. Let $\mathcal{A} = (A, \rightarrow)$ be an ARS. An element $a \in A$ is reducible if there exists an element $b \in A$ with $a \rightarrow b$. A normal form is an element that is not reducible. The set of normal forms of $\mathcal{A}$ is denoted by $\text{NF}(\mathcal{A})$ or $\text{NF}(\rightarrow)$ when $\mathcal{A}$ can be inferred from the context. An element $a \in A$ has a normal form if $a \rightarrow^* b$ for some normal form $b$. In that case we write $a \rightarrow^* b$.

Example

- $a \rightarrow f$  
- $e, f, j, d$ not $g, d$
- $g \rightarrow^* d$

Element $a$ has normal forms?

How many normal forms has this ARS?

**ARS**

$\mathcal{A} = (A, \rightarrow)$

- $d$ is normal form
- $\text{NF}(\mathcal{A}) = \{d, g\}$
- $b \rightarrow g$
Termination

Definition 1.2.1. Let \( A = (A, \rightarrow) \) be an ARS. An element \( a \in A \) is called terminating or strongly normalizing (SN) if there are no infinite rewrite sequences starting at \( a \). The ARS \( A \) is terminating or strongly normalizing if all its elements are terminating. An element \( a \in A \) has unique normal forms (UN) if it does not have different normal forms \( f(a, a) \) and \( a \rightarrow * b \) then \( a = b \). The ARS \( A \) has unique normal forms if all its elements have unique normal forms.

Confluence

Definition 1.2.3. Let \( A = (A, \rightarrow) \) be an ARS. An element \( a \in A \) is confluent if for all elements \( b, c \in A \) with \( a \rightarrow * b \) and \( a \rightarrow * c \) we have \( b \equiv c \). The ARS \( A \) is confluent if all its elements are confluent.

**Termination**

- **SN** strong normalization termination
  - no infinite rewrite sequences
- **WN** (weak) normalization
  - every element has at least one normal form
  - \( \forall a \exists b \ a \rightarrow^1 b \)
- **UN** unique normal forms
  - no element has more than one normal form
  - \( \forall a, b, c \text{ if } a \rightarrow^1 b \text{ and } a \rightarrow^1 c \text{ then } b = c \)

An element \( a \) is weakly normalizing (WN) or simply normalizing (SN) if it has a normal form.

\( a \rightarrow^* b \)

\( c \) is WN? SN?

\( a \) or \( c \) has UN?

The nf are convertible?

\[ \begin{align*}
R &= \begin{cases}
  f(x, x) & \rightarrow c \\
  a & \rightarrow b \\
  f(x, b) & \rightarrow d
\end{cases}
\]

\( f(a, a) \) has normal form?

Can you produce two different nf?

we can compute from the same term \( f(a, a) \) two different normal-forms \( c \) and \( d \)

different meaning for equivalent terms

**Same meaning for *equivalent* terms**

**Confluence**

in an ARS with the property UN every equivalence class of convertible elements contains at most one normal form.

Q: are UN and UNC equivalent?

\[ \begin{align*}
a \leftarrow b \rightarrow c \leftarrow d \rightarrow e
\end{align*} \]
1. $a$ is confluent?
2. $f$ is confluent?

3. Can you add a single arrow so that the resulting ARS has unique normal forms without being confluent?

**Confluence**

A property of term $t$ is **local** if it is quantified over only one-step reductions from $t$; it is **global** if it is quantified over all rewrite sequences from $t$.

- **Locally confluent (WCR)**
- **Strongly confluent**
- **Diamond**

- **Global property:**

An ARS $A = (A, \rightarrow)$ is confluent if and only if $\Rightarrow \subseteq \rightarrow$.

**Confluence**

A property of term $t$ is **local** if it is quantified over only one-step reductions from $t$; it is **global** if it is quantified over all rewrite sequences from $t$.

- **Locally confluent (WCR)**
- **Strongly confluent**
- **Diamond**

An ARS $A = (A, \rightarrow)$ has the diamond property ($\Diamond$) if $\rightarrow \subseteq \Rightarrow$.

**Diamond property**

$\forall a, b, c$ 

$\exists d$

**Every ARS with diamond property is confluent**
What is true?

1. UN implies CR
   \[ a \to b \to c \]

2. Every weakly normalizing ARS with unique normal forms is confluent.

\[ a \to b \to c \to d \]

(i) WCR & WN implies CR

3. WCR & WN implies CR

(ii) WN \nRightarrow SN

Which is true?

1. SN \Rightarrow WN
2. WN \Rightarrow SN
3. Confluence \Rightarrow UN
4. UN \Rightarrow Confluence
5. Confluence \Rightarrow Local confluence
6. Local confluence \Rightarrow Confluence
7. WN & UN \Rightarrow Confluence
8. WN & Local Conf. \Rightarrow Confluence
9. SN & Local Conf. \Rightarrow Confluence

less obvious form

\[ a \to b \]

(ii) WN \nRightarrow SN

R = \{ \frac{f(a) \to c}{f(x) \to f(a)} \}

The system is weakly normalising but not strongly normalising:

Can you find an infinite reduction sequence?

\[ f(b) \to f(a) \to c \]

\[ f(b) \to f(a) \to f(a) \ldots \]

Newman Lemma

By well-founded induction

Newman’s Lemma
Every terminating and locally confluent ARS is confluent.
Memo: Well-founded Induction

**Definition** (Relation bien fondée) Une relation d’ordre $\leq E \times E$ est bien fondée si il n’existe pas de suite infinité d’éléments de $E$ décroissante par rapport à $\leq$.

**Theorem** (Principe d’induction bien fondée) Soient donnés un ensemble $E$ quelconque, un ordre strict $<$ sur $E$ (disjoint de son ensemble d’éléments minimaux), et une propriété $P$ sur $E$.

Si
1. pour tout élément minimal $m \in E$ on a $P(m)$
2. le fait que $P(k)$ soit vérifié pour tout élément $k < a$ implique $P(a)$

alors
pour tout $s \in E$ on a $P(s)$

Recap basics

An abstract rewriting system (ARS) is a pair $(A, \rightarrow)$ consisting of a set $A$ and a binary relation $\rightarrow$ on $A$ whose pairs are written $t \rightarrow s$ and called steps.

We denote $\rightarrow^*$ (resp. $\rightarrow^*$) the transitive-reflexive (resp. reflexive) closure of $\rightarrow$. We write $t \rightarrow u$ if $u \rightarrow v$.

If $\rightarrow_1, \rightarrow_2$ are binary relations on $A$, then $\rightarrow_1 \rightarrow_2$ denotes their composition, i.e., $t \rightarrow_1 s \rightarrow_2 p$ if there exists $u \in A$ such that $t \rightarrow_1 u \rightarrow_2 p$.

We write $(S, \rightarrow_1, \rightarrow_2)$ to denote the compound system $(A, \rightarrow)$ where $\rightarrow = \rightarrow_1 \cup \rightarrow_2$.

A $\rightarrow$-sequence (or reduction sequence) from $t$ is a (possibly infinite) sequence $t_0, t_1, t_2, \ldots$ such that $t_i \rightarrow t_{i+1}$.

$t \rightarrow^*$ $s$ indicates that there is a finite sequence from $t$ to $s$.

$A \rightarrow$-sequence from $t$ is maximal if it is either infinite or ends in $a \rightarrow^* s$.

**Prop.** DIAMOND implies CONFLUENCE

The heart of confluence is a diamond

Newman Lemma

Newman’s Lemma. Every terminating and locally confluent ARS is confluent.

A second proof. Let $A = (A, \rightarrow)$ terminating and locally confluent.

It suffices to show that every element has unique normal forms.

1. Suppose $B = \{ a \in A \mid \text{UN}(a) \neq \emptyset \}$
2. Let $b \in B$ be minimal element (with respect to $\rightarrow$).
3. $b \rightarrow^* n_1$ and $b \rightarrow^* n_2$ with $n_1 \neq n_2$.

Conclusion: showing that it is impossible (absurd).
Closure

$\rightarrow^*$ is the reflexive, transitive closure of $\rightarrow$.

1. $M \rightarrow N \Rightarrow M \rightarrow^* N$.
2. $M \rightarrow^* N$, $N \rightarrow^* L \Rightarrow M \rightarrow^* L$.

The transitive-reflexive closure of a relation is a closure operator, i.e.,

$\rightarrow^* = \bigcup_{n \geq 1} \rightarrow^n$.

As a consequence

$\rightarrow^* \circ \rightarrow^* \subseteq \rightarrow^*$.

Commutation

Commutation. Two relations $\rightarrow_1$ and $\rightarrow_2$ on $A$ commute if

$\rightarrow_1^* \circ \rightarrow_2^* \subseteq \rightarrow_2^* \circ \rightarrow_1^*$.

Confluence. A relation $\rightarrow$ on $A$ is confluent if it commutes with itself.

Proving confluence modularly

Lemma (Hindley-Rosen)

If two relations $\rightarrow_1$ and $\rightarrow_2$ are confluent and commute with each other, then $\rightarrow_1 \cup \rightarrow_2$ is confluent.

An effective usable technique

Lemma (Hindley-Rosen)

If two relations $\rightarrow_1$ and $\rightarrow_2$ are confluent and commute with each other, then $\rightarrow_1 \cup \rightarrow_2$ is confluent.

- Global condition
  - (all sequences)
  - $\rightarrow_1 \cup \rightarrow_2$ is confluent.

- Local condition
  - (one-step test)

Lemma (Hindley's local test)

Strong commutation $\rightarrow_1 \circ \rightarrow_2 \subseteq \rightarrow_1^* \circ \rightarrow_2^*$ implies commutation.

Strategies and subreductions

Lemma (Hindley-Rosen)

If two relations $\rightarrow_1$ and $\rightarrow_2$ are confluent and commute with each other, then $\rightarrow_1 \cup \rightarrow_2$ is confluent.

Global condition
- (all sequences)

Local condition
- (one-step test)

$(\text{Strong Commutation})$

Lemma (Local test). Strong commutation implies commutation.
Normalization

- Def. \((\mathcal{A}, \rightarrow)\) is strongly (weakly, uniformly) normalizing if each \(t \in \mathcal{A}\) is, where the three normalization notions are as follows.
- \(t\) is strongly \(\rightarrow\)-normalizing: every maximal \(\rightarrow\)-sequence from \(t\) ends in a normal form.
- \(t\) is weakly \(\rightarrow\)-normalizing: there exist \(u \rightarrow \) sequence from \(t\) which ends in a normal form.
- \(t\) is uniformly \(\rightarrow\)-normalizing: \(t\) weakly \(\rightarrow\)-normalizing implies \(t\) strongly \(\rightarrow\)-normalizing.

If terms are not strongly normalizing, how do we compute a normal form, or even test if any exist? This is the problem tackled by normalization. By repeatedly performing only specific steps \(\not\rightarrow\), we are guaranteed that a normal form will eventually be computed, if any exists.

Normalizing strategies

- Def. \((\mathcal{A}, \rightarrow)\) is strongly (weakly, uniformly) normalizing if each \(t \in \mathcal{A}\) is, where the three normalization notions are as follows.
- \(t\) is strongly \(\rightarrow\)-normalizing: every maximal \(\rightarrow\)-sequence from \(t\) ends in a normal form.
- \(t\) is weakly \(\rightarrow\)-normalizing: there exist \(u \rightarrow \) sequence from \(t\) which ends in a normal form.
- \(t\) is uniformly \(\rightarrow\)-normalizing: \(t\) weakly \(\rightarrow\)-normalizing implies \(t\) strongly \(\rightarrow\)-normalizing.

Completeness

Completeness. The restriction to a subreduction is a way to control the non-determination which arises from different possible choices of reduction.

In general, we are interested in subreductions which are complete w.r.t. certain subset of interests (i.e., values, normal forms, head normal forms).

Given \(\mathcal{B} \subseteq \mathcal{A}\), we say that \(\mathcal{B}\) is complete if whenever \(t \rightarrow^* u\) with \(u \in \mathcal{B}\), then \(t \rightarrow^n \psi\), with \(\psi \in \mathcal{B}\).

Factorization

Another commutation

Operational properties of interest

- Termination and Confluence
  Existence and uniqueness of normal forms
- How to Compute
  reduction strategies with good properties:
  - standardization,
  - normalization

Factorization

(aka Semi-Standardization, Postponement, or often simply Standardization)

- most basic property about how to compute

\[ t \rightarrow^*_\beta u \Rightarrow t \rightarrow^*_h \cdot \rightarrow^*_h u \text{ head factorization} \]

A key building-block in proofs of more sophisticated how-to-compute properties:

- allows immediate proofs of normalization
  (a reduction strategy reaches a normal form, whenever one exists)
- simplest way to prove standardization, by using Mitschke’s argument
  (left-to-right standardization = iterate head factorization)
Factorization
(also Semi-Standardization, Postponement, or often simply Standardization)

Malliès 97:
the meaning of factorization is that the essential part of a computation can always be separated from its junk.

Assume computations consists of
- steps \( \gamma \) which are in some sense essential, and
- steps \( \gamma \) which are not.

Factorization says that every rewrite sequence can be reorganized/factorized as a sequence of essential steps followed by inessential ones.

\[ I \rightarrow^* U \Rightarrow I \rightarrow^* \gamma \cdot \rightarrow^* U \quad \text{e-factorization} \]

Local test?

We say that \( \rightarrow \) **strongly postpones after** \( \rightarrow \), if

\[ \text{SP}(\rightarrow, \rightarrow) : \quad \rightarrow : \gamma \cdot \rightarrow \subseteq \rightarrow ^* \cdot \rightarrow ^* \quad \text{(Strong Postponement)} \]

Lemma (Local test for postponement [36]). Strong postponement implies postponement:

\[ \text{SP}(\rightarrow, \rightarrow) \implies \text{PP}(\rightarrow, \rightarrow), \] and so \( \text{Fact}(\rightarrow, \rightarrow) \).

Does \( \text{SP} \) hold for \( \lambda \)-calculus?

**Lemma** (Local test for postponement [36]). Strong postponement implies postponement:

\[ \rightarrow^* \subseteq \rightarrow^* \cdot \rightarrow^* \]

\[ \rightarrow^* \cdot \gamma \cdot \rightarrow^* \subseteq \rightarrow^* \cdot \rightarrow^* \cdot \gamma \cdot \rightarrow^* \]

\[ \text{Fact}(\rightarrow, \rightarrow) \implies \text{PP}(\rightarrow, \rightarrow) \]

**Ex** (\( \lambda \)-calculus and strong postponement). \( \beta \) reduction is decomposed into head reduction \( \rightarrow ^H \) and its dual \( \rightarrow ^H \)

\[ \rightarrow ^H \subseteq \rightarrow \text{ and its dual } \rightarrow ^H \]

Consider:

\[ (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H \]

Does \( \text{SP} \) hold for \( \lambda \)-calculus?

**Ex** (\( \lambda \)-calculus and strong postponement). \( \beta \) reduction is decomposed into head reduction \( \rightarrow ^H \) and its dual \( \rightarrow ^H \)

\[ \rightarrow ^H \subseteq \rightarrow \text{ and its dual } \rightarrow ^H \]

Consider:

\[ (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H (\lambda x.x)(x) \rightarrow ^H \]

**Property.** Given a relation \( \rightarrow^* \) such that \( \forall \gamma \rightarrow^* \gamma \),

\[ \text{PP}(\rightarrow, \rightarrow) \text{ if and only if } \text{PP}(\rightarrow, \rightarrow) \]

What if we define a relation such that

\[ \rightarrow^* \rightarrow^* \subseteq \rightarrow^* \cdot \rightarrow^* \]
Examples of uses for factorization

Call-by-Name and Call-by-Value \( \lambda \)-calculus

Terms and values are generated by the following grammars:

- \( V \) := \( x \mid \lambda x.M \) (Values, \( V \))
- \( M \) := \( x \mid e \mid \lambda x.M \mid MM \) (Terms, \( M \))

where \( x \) ranges over a countable set of names, \( e \) over a string (possibly empty) set \( \Sigma \) of constants.

- If the set of constants is empty, the calculus is pure, and the set of terms is denoted \( \Lambda \).
- Otherwise, the calculus is called applied, and the set of terms is often indicated as \( \Lambda_0 \).

Terms are identified up to renaming of bound variables, where \( \lambda x \) is the only binder constructor. \( P(Q/x) \) is the capture-avoiding substitution of \( Q \) for the free occurrences of \( x \) in \( P \).

Call-by-Name and Call-by-Value \( \lambda \)-calculus

\[ \mathcal{C} \{ M \} \text{ (Contexts) \quad (with one hole \{ \}) \quad \text{are generated as follows.} \]

- A rule \( \rho \) is a binary relation on \( \Lambda_0 \), which we also denote \( \Rightarrow_\rho \), writing \( N \Rightarrow_\rho M \). \( \rho \) is called a \( \rho \)-reduction.
- The best known rule is \( \beta \):
  \[ (\lambda x.M)N \Rightarrow_\beta M[N/x] \]
- A reduction step \( \Rightarrow_\rho \) is the closure under context \( \mathcal{C} \) of \( \rho \).
- Explicitly, \( T \Rightarrow T' \) holds if \( T = \mathcal{C}(B) \), \( T' = \mathcal{C}(B') \), and \( B \Rightarrow B' \).

The \( \lambda \)-calculus can be seen both as an equational theory on terms and as an abstract model of computation.

With the functional paradigm point of view, the meaning of any \( \lambda \)-term is the value it evaluates to.
**Call-by-Name and Call-by-Value \(\lambda\)-calculus**

\(Cn\) and \(CbV\) Calculi.
- The (pure) Call-by-Name calculus \(\Lambda_{cbn} = (\lambda, \beta)\) is the set of terms equipped with the contextual closure of the \(\beta\)-rule:
  \[
  (\lambda x. M)N \rightarrow_\beta M[N/x]
  \]
- The (pure) Call-by-Value calculus \(\Lambda_{cbv} = (\lambda, \beta, \eta)\) is the same set equipped with the contextual closure of the \(\beta, \eta\)-rule:
  \[
  (\lambda x. M)[V/x] \rightarrow_\beta M[V/x] \text{ where } V \in V
  \]

**CbN: Head Reduction**

**Head reduction in CbN**

- Head reduction is the closure of \(\beta\) under head context
  \[\lambda x_1...x_n.\{\} \mid M_1 ... M_k\]
- Head normal forms (\(\beta nf\)), whose set is denoted by \(\mathcal{H}\) is its normal forms.
- Given a rule \(\rho\), we write \(\downarrow_\rho\) for its closure under head context.
- A step \(\downarrow_\rho\) is non-head, written \(\downarrow_\rho\) if it is not head.

\[
H ::= \emptyset | \lambda x. H | H M
\]

**CbN Head Factorization**

**Head Factorization**

Head factorization allows for a characterization of the terms which have head normal form.

- According to the function paradigm of computation the goal of every computation is to determine its value
- Since functions are seen as values, it is natural to consider weak evaluation. In practical implementations, weak evaluation is more realistic than the full beta reduction

**CbV: Weak Reduction**

**Weak reductions in CbV**

The result of interest are values (i.e., functions).

In languages, in general the reduction is weak, that is, it does not reduce in the body of a function.

There are three main weak schemes: left, right and in arbitrary order.

- Left contexts \(L\) and right contexts \(R\) and (arbitrary order) weak contexts \(W\) are defined by:
  \[
  L ::= \emptyset | LM | VL
  \]
  \[
  R ::= \emptyset | RM | RV
  \]
  \[
  W ::= \emptyset | WM | MW
  \]
- Given a rule \(\rightarrow\) on \(\Lambda\), weak reduction \(\rightarrow\) is the closure of \(\rightarrow\) under context \(W\).
- A step \(\rightarrow S\) is non-weak, written \(\rightarrow_{\text{weak}} S\) if it is not weak.
- Similarly for left \((\rightarrow_{\text{weak}}\) and \(\rightarrow_{\text{weak}}\)) and right \((\rightarrow_{\text{weak}}\) and \(\rightarrow_{\text{weak}}\)).

**Fact 3** (Weak normal forms). Given a closed term, \(M\) is \(\rightarrow_{\text{weak}}\)-normal iff \(M\) is a value.

**CbV Weak Factorization**

**Weak Factorization**

Let \(s \in (w, r)\)

- weak factorization of \(\rightarrow_{\beta, \eta, \eta}\): \(\rightarrow_{\beta, \eta, \eta} \subseteq \rightarrow_{\beta, \eta, \eta}\)
- Convergence: \(T \rightarrow_{\beta, \eta, \eta} W (W \in V)\) if and only if \(T \rightarrow_{\beta, \eta, \eta} V (V \in V)\)
- Corollary 4. Given \(M\) a closed term, \(M\) has a \(\beta, \eta, \eta\)-reduction to a value, if and only if the \(\beta, \eta, \eta\)-reduction from \(M\) terminates.
Basic properties of the contextual closure

If a step \( T \rightarrow T' \) is obtained by closure under non-empty context of a rule \( \rightarrow \), then \( T \) and \( T' \) have the same shape, i.e., both terms are an application (resp. an abstraction, a variable).

**Fact 5** (Shape preservation).
1. Assume \( T = C[U] \rightarrow C[U'] \rightarrow T' \) and that the context \( C \) is non-empty. Then \( T \) and \( T' \) have the same shape.
2. Hence, for any internal step \( M \rightarrow M' \) (\( \in \{b, w, r, \ldots\} \)) \( M \) and \( M' \) have the same shape.

The following is an easy to verify consequence.

**Lemma 6** (Redexes preservation).
1. \( \beta \text{N}: \text{Assume } T \rightarrow_{\beta} S. \ T \text{ is a } \beta\text{-redex iff so is } S. 
2. \( \beta \text{V}: \text{Assume } T \rightarrow_{\beta} S. \ T \text{ is a } \beta\text{-redex iff so is } S. 

Internal steps preserve head and weak normal nf

Fixed a set of redexes \( R \), \( M \) is \( w\)-normal (resp. \( h\)-normal) if there is no redex \( R \in R \) such that \( M = W[R] \) (resp. \( M = H[R] \)).

**Lemma 7** (Surface normal forms).
1. \( h\text{N}: \text{Let } R \text{ be the set of } h\text{-redexes.}
   \( \text{Assume that } M \rightarrow_{h} M' \text{. } M \text{ is } h\text{-normal } \Rightarrow M' \text{ is } h\text{-normal.} \)
2. \( w\text{V}: \text{Let } R \text{ be the set of } w\text{-redexes.}
   \( \text{Assume that } M \rightarrow_{w} M' \text{. } M \text{ is } w\text{-normal } \Rightarrow M' \text{ is } w\text{-normal.} \)

Back to Factorization

From abstract to concrete system

... but using as little of the specific structure as possible

Recap

**in Call-by-Name:**
- Head Factorization: \( \overline{\text{nf}}^* \subseteq \overline{\text{nf}}^* \).
- Head Normalization: \( M \text{ has nf if and only if } M \rightarrow_{\beta}^* S \text{ (for some } S \in H) \).

**in Call-by-Value:**
- \( \overline{\text{nf}}^* \subseteq \overline{\text{nf}}^* \).
- Convergence: \( T \rightarrow_{\beta} W(W \in V) \text{ if and only if } T \rightarrow_{\beta}^* V \text{ (} V \in V) \).
ARS Recipe

**Property 2 (Criterion).** Given \( \rightarrow \subseteq \rightsquigarrow \cup \triangleleft \), factorization holds

\[ \triangleleft \subseteq \equiv \cdot \triangleleft \subseteq \equiv \triangleleft \]

iff exists \( \psi \)

\[ \triangleleft \equiv \cdot \triangleleft \subseteq \equiv \triangleleft \; (\text{same closure}) \]

\[ \triangleleft \equiv \cdot \triangleleft \subseteq \equiv \triangleleft \; (\text{local postponement}) \]

---

Concretely: CbN and Head Factorization

**Indexed parallel \( \beta \) reduction**

\[
\begin{align*}
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')} \\
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')}
\end{align*}
\]

**Parallel \( \rightarrow_{\text{head}} \) reduction**

\[
\begin{align*}
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')}
\end{align*}
\]

\[ \rightarrow_{\text{head}} \subseteq \Rightarrow \subseteq \Rightarrow \]

1. Merge: if \( t \xrightarrow{\beta} t \Rightarrow \Rightarrow \) \( u \) then \( t \Rightarrow \Rightarrow u \).

2. Split: if \( t \Rightarrow \Rightarrow u \) then \( t \Rightarrow \Rightarrow u \).

---

Concretely: CbV and Weak Factorization

**Indexed parallel \( \beta \) reduction**

\[
\begin{align*}
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')} \\
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')}
\end{align*}
\]

**Parallel \( \rightarrow_{\text{weak}} \) reduction**

\[
\begin{align*}
\frac{t \xrightarrow{\beta} t'}{\lambda x. t \xrightarrow{\beta} \lambda x. t'} & \quad \frac{u \xrightarrow{\beta} u'}{\alpha(t, u) \xrightarrow{\beta} \alpha(t, u')}
\end{align*}
\]

\[ \rightarrow_{\text{weak}} \subseteq \Rightarrow_{\text{weak}} \subseteq \Rightarrow_{\text{weak}} \]

1. Merge: if \( t \xrightarrow{\beta} t \Rightarrow_{\text{weak}} \Rightarrow_{\text{weak}} \) \( u \) then \( t \Rightarrow_{\text{weak}} u \).

2. Split: if \( t \Rightarrow_{\text{weak}} u \) then \( t \Rightarrow_{\text{weak}} u \).

---

You designed a system
You have Factorization
Now what?

From Factorization to Normalization/Standardization
in a few easy steps

---

ARS: more abstract tools...

Decreasing Diagrams:
Lecture Notes, Chapter 6

---

To commute

**Definition 1.** A pair \( (\ll, \gg) \) of rewrite relations commutes if \( \ll \gg \subseteq \gg \ll \), and commutes locally if \( \ll \gg \subseteq \gg \ll \). A rewrite relation \( \rightarrow \) is confluent if \( (\rightarrow, \rightarrow) \) commutes, and locally confluent if \( (\rightarrow, \rightarrow) \) commutes locally.

**Theorem 1 (\( \ll, \gg \)).** A pair of rewrite relations commutes if it is decreasing. A rewrite relation is confluent if it is decreasing.
Decreasing (Van Oostrom)

**Definition 2.** $\rightarrow$ is decreasing, if $\mathfrak{P} = \bigcup_{c \in C} \mathfrak{P}_c = \bigcup_{m \in M} \mathfrak{P}_m$

for families of relations $(\mathfrak{P}_c)_{c \in C} (\mathfrak{P}_m)_{m \in M}$

and some well-founded strict order $<$ on the set of labels $L \cup M$

such that for all labels:

\[
\rho (c, m) \subseteq \bar{\rho} (c, m)
\]

where $\gamma N = \{ n \in L \cup M \mid \exists k \in N \ k > n \}$, and $\gamma_n$ abbreviates $\gamma \{ n \}$

Newman Lemma, again

**Newman’s Lemma.** Every terminating and locally confluent ARS is confluent.

\[
\begin{align*}
\text{WCR} & \quad (A, \rightarrow) \\
\text{LD}_{\text{in}} & \quad (A, \{ \rightarrow \}_{A \in A}) \\
\end{align*}
\]

where $\gamma N = \{ n \in L \cup M \mid \exists k \in N \ k > n \}$