This part focus on **Operational Semantics**
of formal calculi (and programming languages)
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 Networks & proof-nets
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
Operational semantics
of formal calculi and programming languages

Rewriting theory

• Rewriting = abstract form of program execution

• Paradigmatic example: \(\lambda\)-calculus
  (functional programming language, in its essence)
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?
Example (Group Theory)

signature
\[ e \text{ (constant)} \quad - \text{ (unary, postfix)} \quad \cdot \text{ (binary, infix)} \]

equations
\[ e \cdot x \approx x \quad x^- \cdot x \approx e \quad (x \cdot y) \cdot z \approx x \cdot (y \cdot z) \quad \mathcal{E} \]

theorems
\[ e^- \approx_{\mathcal{E}} e \quad (x \cdot y)^- \approx_{\mathcal{E}} y^- \cdot x^- \]

rewrite rules
\[ e \cdot x \rightarrow x \quad x \cdot e \rightarrow x \quad \mathcal{R} \]
\[ x^- \cdot x \rightarrow e \quad x \cdot x^- \rightarrow e \]
\[ (x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z) \quad x^-^- \rightarrow x \]
\[ e^- \rightarrow e \quad (x \cdot y)^- \rightarrow y^- \cdot x^- \]
\[ x^- \cdot (x \cdot y) \rightarrow y \quad x \cdot (x^- \cdot y) \rightarrow y \]

1. \( s \approx t \) is valid in \( \mathcal{E} \) (\( s \approx_{\mathcal{E}} t \)) if and only if \( s \) and \( t \) have same \( \mathcal{R} \)-normal form

2. \( \mathcal{R} \) admits no infinite computations

1 \& 2 \implies \( \mathcal{E} \) has decidable validity problem
Example (Combinatory Logic)

signature  
S  K  I  (constants)  ·  (application, binary, infix)

terms  
S  ((K · I) · I) · S  (x · z) · (y · z)

rewrite rules  
I · x → x
(K · x) · y → x
((S · x) · y) · z → (x · z) · (y · z)

rewriting  
((S · K) · K) · x → (K · x) · (K · x)
→ x

inventor  
Moses Schönfinkel (1924)
Example (Lambda Calculus)

signature \( \lambda \) (binds variables) \( \cdot \) (application, binary, infix)

terms \( M ::= x | (\lambda x. M) | (M \cdot M) \)

\( \alpha \) conversion \( \lambda x. x \cdot y \stackrel{\alpha}{=} \lambda z. z \cdot y \)

\( \beta \) reduction \( (\lambda x. M) \cdot N \rightarrow_\beta M[x := N] \)
replace free occurrences of \( x \) in \( M \) by \( N \)
(and avoid variable capturing)

rewriting \( (\lambda x. x \cdot x) \cdot (\lambda x. x \cdot x) \rightarrow (\lambda x. x \cdot x) \cdot (\lambda x. x \cdot x) \)

inventor Alonzo Church (1932)

both Combinatory Logic and Lambda Calculus are Turing-complete
Operational semantics of formal calculi and programming languages

Rewriting theory

- Rewriting = abstract form of program execution

- Paradigmatic example: $\lambda$-calculus
  (functional programming language, in its essence)
Rewriting

- **Rewrite Theory** provides a powerful set of tools to study **computational and operational properties** of a system: **normalization, termination, confluence, uniqueness of normal forms**
- Tools to study and compare strategies:
  - Is there a strategy guaranteed to lead to normal form, if any (*normalizing strat.*)?
- **Abstract Rewrite Systems** (ARS) capture the common substratum of rewrite theory (*independently from the particular structure of terms*) - can be uses in the study of any calculus or programming language.
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
Abstract Rewriting

Basic language
Definition 1.1.1. An abstract rewrite system (ARS for short) is a pair $\mathcal{A} = \langle A, \rightarrow \rangle$ 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.

ARS $\mathcal{A} = \langle A, \rightarrow \rangle$

- $A = \{ a, b, c, d, e, f, g \}$
- $\rightarrow = \{ (a, e), (b, a), (b, c), (c, d), (c, f) \}$

A (finite) rewrite sequence is a non-empty sequence $(a_0, \ldots, a_n)$ of elements in $A$ such that $a_i \rightarrow a_{i+1}$

We write $a_0 \rightarrow^n a_n$ or simply $a_0 \rightarrow^* a_n$

- rewrite sequence
  - finite $a \rightarrow e \rightarrow b \rightarrow c \rightarrow f$
  - empty $a$
  - infinite $a \rightarrow e \rightarrow b \rightarrow a \rightarrow e \rightarrow b \rightarrow \cdots$
\[ s \leftrightarrow_R t \text{ iff } s \rightarrow_R t \text{ or } t \rightarrow_R s \]
\[ s \leftrightarrow^*_R t \text{ iff } s = s_0 \leftrightarrow_R s_1 \leftrightarrow_R \ldots \leftrightarrow_R s_n = t \text{ for } n \geq 0 \]

- \( \leftrightarrow \) inverse of \( \rightarrow \)
- \( \rightarrow^* \) transitive and reflexive closure of \( \rightarrow \)
- \( \leftrightarrow^* \) inverse of \( \rightarrow^* \)
- \( \leftrightarrow \) symmetric closure of \( \rightarrow \)
- \( \leftrightarrow^* \) conversion (equivalence relation generated by \( \rightarrow \))
- \( \rightarrow^+ \) transitive closure of \( \rightarrow \)
- \( \rightarrow^= \) reflexive closure of \( \rightarrow \)

- is relation composition:
\[ R \cdot S = \{ (a, c) \mid a R b \text{ and } b S c \} \]

\[ \downarrow = \rightarrow^* \cdot \rightarrow^* \]
We denote $\to^*$ (resp. $\to^=\!$) the transitive-reflexive (resp. reflexive) closure of $\to$.

If $\to_1, \to_2$ are binary relations on $A$ then $\to_1 \cdot \to_2$ denotes their composition, i.e. $t \to_1 \cdot \to_2 s$ iff there exists $u \in A$ such that $t \to_1 u \to_2 s$.

We write $(A, \{\to_1, \to_2\})$ to denote the ARS $(A, \to)$ where $\to = \to_1 \cup \to_2$. 
The transitive-reflexive closure of a relation is a closure operator, i.e. satisfies
\[ \rightarrow \subseteq \rightarrow^*, \quad (\rightarrow^*)^* = \rightarrow^*, \quad \rightarrow_1 \subseteq \rightarrow_2 \text{ implies } \rightarrow_1^* \subseteq \rightarrow_2^* \]

As a consequence
\[ (\rightarrow_1 \cup \rightarrow_2)^* = (\rightarrow_1^* \cup \rightarrow_2^*)^* \]
**Terminology**

- If $x \rightarrow^* y$ then $x$ rewrites to $y$ and $y$ is reduct of $x$
- If $x \rightarrow^* z \leftarrow y$ then $z$ is common reduct of $x$ and $y$
- If $x \leftrightarrow^* y$ then $x$ and $y$ are convertible

**Example**

```
\[ a \leftrightarrow b \rightarrow c \rightarrow d \]
```

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

```
\[ e \rightarrow f \rightarrow d \rightarrow g \]
```

```
\[ a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g \]
```
**Definition 1.1.11.** Let $\mathcal{A} = \langle A, \rightarrow \rangle$ 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 $A$ 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 $a \rightarrow^! b$.

Element $a$ has normal forms?  
How many normal forms has this ARS?

ARS $\mathcal{A} = \langle A, \rightarrow \rangle$
- $d$ is normal form
- $\text{NF}(\mathcal{A}) = \{ d, g \}$
- $b \rightarrow^! g$
• **SN**  strong normalization  termination
  - no infinite rewrite sequences

• **WN**  (weak) normalization
  - every element has at least one normal form
  - $\forall a \exists b \ a \rightarrow^! b$

• **UN**  unique normal forms
  - no element has more than one normal form
  - $\forall a, b, c \text{ if } a \rightarrow^! b \text{ and } a \rightarrow^! c \text{ then } b = c$
**Termination**

**Definition 1.2.1.** Let $A = \langle A, \rightarrow \rangle$ 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 ($\forall b, c \in A$ if $a \rightarrow^1 b$ and $a \rightarrow^1 c$ then $b = c$). The ARS $A$ has unique normal forms if all its elements have unique normal forms.

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

```
 a ← b → c → d
    ↑   ↓
    ↓   ↓
  e ← f
    ↓   ↓
    ↓   ↓
    g
```

- $a$ is WN? SN?
- $c$ is WN? SN?
- $a$ or $c$ has UN?
- The nf are convertible?
Definition 1.2.3. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS. An element $a \in A$ is confluent if for all elements $b, c \in A$ with $b \leftrightarrow a \rightarrow^* c$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.

Every confluent ARS has unique normal forms.
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?
Given

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

\( f(a,a) \) 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
(different meaning for same term!)
Same meaning for *equivalent* terms
Definition 1.2.3. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS. An element $a \in A$ is confluent if for all elements $b, c \in A$ with $b \leftarrow a \rightarrow c$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.

An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is confluent if and only if $\leftrightarrow^* \subseteq \downarrow$. 
Definition 1.2.10. An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ has unique normal forms with respect to conversion (UNC) if different normal forms are not convertible ($\forall a, b \in \text{NF}(\mathcal{A})$ if $a \leftrightarrow^* b$ then $a = b$).

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

Q: are UN and UNC equivalent?
Global vs Local
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**

Let $A = \langle A, \rightarrow \rangle$ be an ARS. An element $a \in A$ is *confluent* if for all elements $b, c \in A$ with $b \xleftarrow{*} a \rightarrow{*} c$ we have $b \downarrow c$. The ARS $A$ is confluent if all its elements are confluent.
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**

---

**Local confluence**

Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS. An element $a \in A$ is locally confluent for all elements $b, c \in A$ with $b \leftarrow a \rightarrow c$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.

---

An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ has the diamond property $(\diamond)$ if $\leftarrow \cdot \rightarrow \subseteq \rightarrow \cdot \leftarrow$. 
- diamond property

- $\leftrightarrow \cdot \rightarrow \subseteq \rightarrow \cdot \leftarrow$

- $\forall a, b, c$

- $\exists d$

- every ARS with diamond property is confluent
An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is *strongly confluent* (SCR) if $\leftarrow \cdot \rightarrow \subseteq \rightarrow^\ast \cdot \leftarrow$, see Figure.

a) Show that every strongly confluent ARS is confluent.

b) Does the converse also hold?

c) Show that an ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is confluent if and only if $\leftarrow^\ast \cdot \rightarrow \subseteq \rightarrow^\ast \cdot \leftarrow^\ast$.
Which is true?

1. SN => WN
2. WN => SN
3. Confluence => UN
4. UN => Confluence
5. Confluence => Local confluence
6. Local confluence => Confluence
7. WN & UN => Confluence
8. WN & Local Conf. => Confluence
9. SN & Local Conf. => Confluence
WN vs SN

\[ \mathcal{R} = \begin{cases} f(a) & \rightarrow & c \\ f(x) & \rightarrow & f(a) \end{cases} \]

The system is weakly normalising but not strongly normalising:

Can you find an infinite reduction sequence?

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

\[ f(b) \rightarrow f(a) \rightarrow f(a) \ldots \]
1. SN => WN
2. WN => SN
3. Confluence => UN
4. UN => Confluence
5. Confluence => Local confluence
6. Local confluence => Confluence
7. WN & UN => Confluence
8. WN & Local Conf. => Confluence
9. SN & Local Conf. => Confluence

Newman’s Lemma
Lemma

$\text{WN } \& \text{ UN} \implies \text{ CR}$

Proof

- $\text{WN} \implies \exists n_1, n_2: b_1 \rightarrow^n n_1 \text{ and } b_2 \rightarrow^n n_2$

- $\text{UN} \implies n_1 = n_2 \implies b_1 \downarrow b_2$
Newman’s Lemma. *Every terminating and locally confluent ARS is confluent.*

By well-founded induction
Memo: Well-founded Induction

Définition : [Relation bien fondée] Une relation d’ordre $\succeq E \times E$ est bien fondée si il n’existe pas de suite infinie d’éléments de $E$ décroissante par rapport à $>$.  

Theorem : [Principe d’induction bien fondée] Soient donnés un ensemble $E$ quelconque, un ordre strict $< E$ (dont $\mathcal{M}$ est son ensemble d’éléments minimaux), et une propriété $P$ sur $E$.  

Si  

1. pour tout élément minimal $m \in \mathcal{M}$ on a $P(m)$  
2. le fait que $P(k)$ soit vérifiée pour tout élément $k < x$ implique $P(x)$  

alors  

pour tout $x \in E$ on a $P(x)$

The proof technique of well-founded induction states that a property $\mathcal{P}$ of elements of a terminating ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ holds for all elements in $A$ if the following condition is satisfied: An element $a \in A$ has the property $\mathcal{P}$ if all elements $b$ with $a \rightarrow b$ have the property $\mathcal{P}$. In particular every normal form has to satisfy the property $\mathcal{P}$. 
Newman’s Lemma. Every terminating and locally confluent ARS is confluent.

1. WCR
2. Induction hypothesis \((a \rightarrow b_1 \implies b_1 \text{ is CR})\)
3. Induction hypothesis \((a \rightarrow c_1 \implies c_1 \text{ is CR})\)
Newman’s Lemma. *Every terminating and locally confluent ARS is confluent.*

**A second Proof.**

Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be *terminating and locally confluent*

It suffices to show that every element has unique normal forms

- suppose $B = \{ a \in A \mid \neg \text{UN}(a) \} \neq \emptyset$
- let $b \in B$ be *minimal* element (with respect to $\rightarrow$)
- $b \rightarrow^1 n_1$ and $b \rightarrow^1 n_2$ with $n_1 \neq n_2$

» **Conclude** by showing that it is impossible (absurd)
- **EX** Say which properties hold

1. Confluent
2. Locally confluent
3. Normalizing (weakly normalizing, WN)
4. Terminating (strongly normalizing, SN)
An abstract rewriting system (ARS) is a pair \((\mathcal{A}, \rightarrow)\) consisting of a set \(\mathcal{A}\) and a binary relation \(\rightarrow\) on \(\mathcal{A}\) 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 \(t \leftarrow u\) if \(u \rightarrow t\).

If \(\rightarrow_1, \rightarrow_2\) are binary relations on \(\mathcal{A}\) then \(\rightarrow_1 \cdot \rightarrow_2\) denotes their composition, i.e. \(t \rightarrow_1 \cdot \rightarrow_2 s\) if there exists \(u \in \mathcal{A}\) such that \(t \rightarrow_1 u \rightarrow_2 s\).

We write \((\mathcal{A}, \{\rightarrow_1, \rightarrow_2\})\) to denote the compound system \((\mathcal{A}, \rightarrow)\) where \(\rightarrow = \rightarrow_1 \cup \rightarrow_2\).

A \(\rightarrow\)-sequence (or reduction sequence) from \(t\) is a (possibly infinite) sequence \(t, t_1, t_2, \ldots\) 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 \(\rightarrow\)-nf.
The heart of confluence is a diamond

**Prop.** DIAMOND implies CONFLUENCE

*Can rarely be used directly:*
Most relations of interest do not satisfy it

**Lemma** *(Characterize Confluence).* \( \rightarrow \) is confluent if and only if there exists a relation \( \Rightarrow \) such that

1. \( \Rightarrow^* = \rightarrow^* \),
2. \( \Rightarrow \) is diamond.
You have already seen an example: in the class by Joly

**Definition**  *The development relation is the least reflexive relation* \( \triangleright \) *on \( \Lambda \) such that:

- \( t \triangleright t' \implies \lambda x t \triangleright \lambda x t' \)
- \( t \triangleright t', u \triangleright u' \implies tu \triangleright t'u' \)
- \( t \triangleright t', u \triangleright u' \implies (\lambda x t)u \triangleright t'[x:=u'] \).

**Lemma 1**  \( \rightarrow \subseteq \triangleright \subseteq \rightarrow \).
The transitive-reflexive closure of a relation is a closure operator, i.e. satisfies
\[ \rightarrow \subseteq \rightarrow^*, \quad (\rightarrow^*)^* = \rightarrow^*, \quad \rightarrow_1 \subseteq \rightarrow_2 \text{ implies } \rightarrow_1^* \subseteq \rightarrow_2^* \]

As a consequence
\[ (\rightarrow_1 \cup \rightarrow_2)^* = (\rightarrow_1^* \cup \rightarrow_2^*)^*. \]
**Commutation.** Two relations $\rightarrow_1$ and $\rightarrow_2$ on $A$ commute if $\leftarrow_1 \cdot \rightarrow_2 \subseteq \rightarrow_2 \cdot \leftarrow_1$.

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

Lemma (Hindley’s local test)

*Strong commutation $\leftarrow_1 \cdot \rightarrow_2 \subseteq \rightarrow_2 \cdot \leftarrow_1$ implies commutation.*
Lemma (Hindley-Rosen)

If two relations \( \rightarrow_1 \) and \( \rightarrow_2 \) are confluent and commute with each other, then

\[
\rightarrow_1 \cup \rightarrow_2 \text{ is confluent.}
\]

Global condition (all sequences)

Local condition (one-step test)

\[
\leftarrow_1 \cdot \rightarrow_2 \subseteq \rightarrow_2^* \cdot \leftarrow_1 = \quad \text{(Strong Commutation)}
\]

Lemma (Local test). Strong commutation implies commutation.
Strategies and subreductions
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 a \(\rightarrow\)-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 exists? This is the problem tackled by *normalization*. By repeatedly performing *only specific steps* \(\rightarrow_e\), we are guaranteed that a normal form will eventually be computed, if any exists.
Normalizing strategies

**Def.** \((A, \rightarrow)\) is strongly (weakly, uniformly) normalizing if each \(t \in A\) 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 a \(\rightarrow\)-sequence from \(t\) which ends in a normal form.
- \(t\) is uniformly \(\rightarrow\)-normalizing: \(t\) weakly \(\rightarrow\)-normalizing implies \(t\) strongly \(\rightarrow\)-normalizing.

**Def.**

- \(\overset{e}{\rightarrow}\) is a **strategy for** \(\rightarrow\) if \(\overset{e}{\rightarrow} \subseteq \rightarrow\), and it has the same normal forms as \(\rightarrow\).
- It is a **normalizing strategy** for \(\rightarrow\) if whenever \(t \in A\) has \(\rightarrow\)-normal form, then every maximal \(\overset{e}{\rightarrow}\)-sequence from \(t\) ends in normal form.
Completeness. The restriction to a subreduction is a way to control the non-determinism 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 $\rightarrow_\mathcal{B} \subseteq \rightarrow$ is $\mathcal{B}$-complete if whenever $t \rightarrow^* u$ with $u \in \mathcal{B}$, then $t \rightarrow_\mathcal{B}^* u'$, with $u' \in \mathcal{B}$.
Factorization
(aka weak Standardization)

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

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

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

Melliès 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 $\xrightarrow{e}$ which are in some sense *essential*, and
- steps $\xrightarrow{i}$ which are not.

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

\[ t \xrightarrow{*} u \implies t \xrightarrow{e} \star \cdot \xrightarrow{i} \star u \quad \text{e-factorization} \]
**Factorization.** Let $\mathcal{A} = (A, \{\to_e, \to_i\})$ be an ARS.

- The relation $\to = \to_e \cup \to_i$ satisfies **e-factorization**, written $\text{Fact}(\to_e, \to_i)$, if
  \[
  \text{Fact}(\to_e, \to_i) : \quad (\to_e \cup \to_i)^* \subseteq \to_e^* \cdot \to_i^*.
  \]  
  **(Factorization)**

- The relation $\to_i$ **postpones** after $\to_e$, written $\text{PP}(\to_e, \to_i)$, if
  \[
  \text{PP}(\to_e, \to_i) : \quad \to_i^* \cdot \to_e^* \subseteq \to_e^* \cdot \to_i^*.
  \]  
  **(Postponement)**

**Lemma.** For any two relations $\to_e, \to_i$ the following are equivalent:

1. $\to_i^* \cdot \to_e \subseteq \to_e^* \cdot \to_i^*$
2. $\to_i \cdot \to_e^* \subseteq \to_e^* \cdot \to_i^*$
3. Postponement: $\to_i^* \cdot \to_e^* \subseteq \to_e^* \cdot \to_i^*$
4. Factorization: $(\to_e \cup \to_i)^* \subseteq \to_e^* \cdot \to_i^*$
We say that $\rightarrow_i$ strongly postpones after $\rightarrow$, if

$$\text{SP}(\rightarrow_i, \rightarrow): \rightarrow_i \cdot \rightarrow \subseteq \rightarrow_i^* \cdot \rightarrow = \text{(Strong Postponement)}$$

Lemma (Local test for postponement [26]). Strong postponement implies postponement:

$$\text{SP}(\rightarrow_i, \rightarrow) \text{ implies } \text{PP}(\rightarrow_i, \rightarrow), \text{ and so } \text{Fact}(\rightarrow_i, \rightarrow).$$
Does SP hold for $\lambda$-calculus?

**Ex (\(\lambda\)-calculus and strong postponement).** $\beta$ reduction is decomposed in head reduction $\xrightarrow{h}\beta$ and its dual $\xrightarrow{\neg h}\beta$

$$\xrightarrow{h}\beta = \xrightarrow{h}\beta \cup \xrightarrow{\neg h}\beta$$

Consider:

$$(\lambda x.xxx)(Iz) \xrightarrow{\neg h}\beta (\lambda x.xxx)z \xrightarrow{h}\beta zzz.$$
The relation \( \rightarrow_i \) **postpones** after \( \rightarrow_e \), written \( \text{PP}(\rightarrow_e, \rightarrow_i) \), if

\[
\text{PP}(\rightarrow_e, \rightarrow_i) : \quad \rightarrow_i^* \cdot \rightarrow_e^* \subseteq \rightarrow_e^* \cdot \rightarrow_i^*.
\]

(Postponement)

**Property.** *Given a relation \( \rightarrow_i \) such that \( \rightarrow_i^* = \rightarrow_i^* \),

\( \text{PP}(\rightarrow_e, \rightarrow_i) \) if and only if \( \text{PP}(\rightarrow_e, \rightarrow_i) \).*
Property 2 (Criterion). Given $\rightarrow=\rightarrow_e \cup \rightarrow_i$, e-factorization holds

$$\rightarrow^* \subseteq \rightarrow_e^* \cdot \rightarrow_i^*$$

iff exists $\Rightarrow$

- $\Rightarrow^* = \rightarrow_i^* \ (\text{same closure})$
- $\Rightarrow_i \cdot \rightarrow \subseteq \rightarrow_e^* \cdot \Rightarrow_i \ (\text{strong postponement})$
Examples of uses for factorization
Call-by-Name and Call-by-Value $\lambda$-calculus