M2 LMFI

**Proofs and programs: advanced topics**

**Linear Logic and Quantitative Semantics**

**Teachers:**

Claudia Faggian CNRS (IRIF)
faggian@irif.fr
https://www.irif.fr/~faggian/

Gabriele Vanoni

INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

1

## Organization

- Lectures:
  - Wednesday 14h00-16h00
  - Friday 14h00-16h00
- **Grading:**
  - ➢ **weekly homework** projects

2

## Plan

A foundational study of functional programming languages,

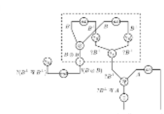
- building on:
  - ➢ *proof theory* (Types, Curry-Howard isomorphism) and
  - ➢ the theory of *lambda-calculus,*
- adopting *the **dynamic and quantitative view brought by Linear Logic.***
  - ➢ Focus first part: a **quantitative view in Operational Semantics**
  - ➢ Focus second prat: a **quantitative view in Denotational Semantics**
  - ➢ Openings towards active research topics: Bayesian learning/ probabilistic programming, ….
- Courses from **LMFI** first term we build on:
  - ➢ *Proof Theory (cut-elimination, lambda calculus,* Curry-Howard *iso)*
- Connected to the **MPRI** course: Semantics of Programming Language (which builds on the models of Linear Logic)

3

## Linear Logic [Girard87] breakthroughs

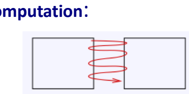
New insights into **proof theory** and
(via the **Curry-Howard correspondence** between proofs and programs )
into the **semantics of programming language**.

- **Proof Nets:** advanced formal system



- representation of proofs (λ-terms, functional programs) by graphs
- tool for the analysis of cut-elimination (= execution) as graph-rewriting process

- Dynamic view, capturing the **flow of computation**:



  - ➢ **Game Semantics**
  - ➢ **Geometry of Interaction**

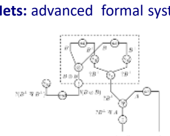
4

## Linear Logic [Girard87] breakthroughs

New insights into **proof** theory and
(via the **Curry-Howard co**rrespondence between proofs and programs )
into the **semantics of pro**gramming language.

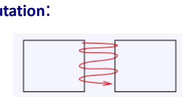
especially suitable for
- Cost analysis (runtime/memory space/ other resources)
- modelling probabilistic & quantum programming

- **Proof Nets:** advanced formal system



- representation of proofs (λ-terms, functional programs) by graphs
- tool for the analysis of cut-elimination (= execution) as graph-rewriting process

- Dynamic view, capturing the **flow of computation**:



  - ➢ **Game Semantics**
  - ➢ **Geometry of Interaction**

- Account for resources
  - ➢ Quantitative Semantics
  - ➢ Quantitative Type Systems

5

## Resource awareness (Quantitative Types)

$$\frac{}{\mathcal{P}, x : P \vdash x : P}\ \text{S-VAR} \qquad \frac{}{\mathcal{P} \vdash \mathtt{b} : \mathtt{B}}\ \text{S-BOOL} \qquad \frac{\mathcal{P} \vdash u : P \quad \mathcal{P}, x : P \vdash t : A}{\mathcal{P} \vdash \mathtt{let}\ x = u\ \mathtt{in}\ t : A}\ \text{S-LET}$$

$$\frac{\mathcal{P} \vdash v : L_1 \quad \mathcal{P} \vdash w : L_2}{\mathcal{P} \vdash \langle v, w \rangle : L_1 \otimes L_2}\ \text{S-PAIR} \qquad \frac{\mathcal{P} \vdash v : L_1 \otimes L_2 \quad \mathcal{P}, x : L_1, y : L_2 \vdash t : A}{\mathcal{P} \vdash \mathtt{letp}\ \langle x, y \rangle = v\ \mathtt{in}\ t : A}\ \text{S-LETP}$$

$$\frac{\mathcal{P}, x : P \vdash t : A}{\mathcal{P} \vdash \lambda x.t : P \multimap A}\ \text{S-ABS} \qquad \frac{\mathcal{P} \vdash t : P \multimap A \quad \mathcal{P} \vdash v : P}{\mathcal{P} \vdash tv : A}\ \text{S-APP}$$

$$\frac{\mathcal{P} \vdash t : A}{\mathcal{P} \vdash !t : !A}\ \text{S-BANG} \qquad \frac{\mathcal{P} \vdash v : !A}{\mathcal{P} \vdash \mathtt{der}\ v : A}\ \text{S-DER}$$

$$\frac{}{\Lambda, x : P \vdash x : P}\ \text{I-VAR} \qquad \frac{\Lambda, \Gamma_1 \vdash u : P \quad \Lambda, \Gamma_2, x : P \vdash t : A}{\Lambda, \Gamma_1 \uplus \Gamma_2 \vdash \mathtt{let}\ x = u\ \mathtt{in}\ t : A}\ \text{I-LET}$$

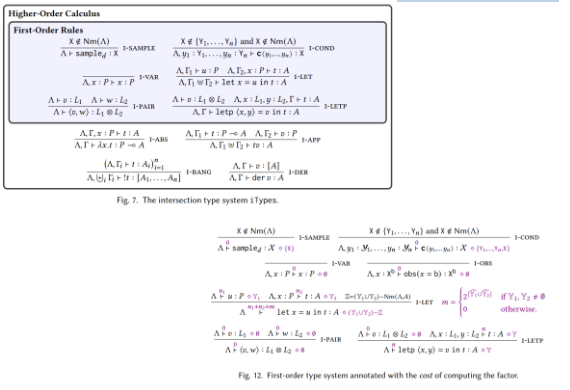
$$\frac{\Lambda \vdash v : L_1 \quad \Lambda \vdash w : L_2}{\Lambda \vdash \langle v, w \rangle : L_1 \otimes L_2}\ \text{I-PAIR} \qquad \frac{\Lambda \vdash v : L_1 \otimes L_2 \quad \Lambda, x : L_1, y : L_2, \Gamma \vdash t : A}{\Lambda, \Gamma \vdash \mathtt{letp}\ \langle x, y \rangle = v\ \mathtt{in}\ t : A}\ \text{I-LETP}$$

$$\frac{\Lambda, \Gamma, x : P \vdash t : A}{\Lambda, \Gamma \vdash \lambda x.t : P \multimap A}\ \text{I-ABS} \qquad \frac{\Lambda, \Gamma_1 \vdash t : P \multimap A \quad \Lambda, \Gamma_2 \vdash v : P}{\Lambda, \Gamma_1 \uplus \Gamma_2 \vdash tv : A}\ \text{I-APP}$$

$$\frac{(\Lambda, \Gamma_i \vdash t : A_i)_{i=1}^{n}}{\Lambda, \biguplus_i \Gamma_i \vdash !t : [A_1, \ldots, A_n]}\ \text{I-BANG} \qquad \frac{\Lambda, \Gamma \vdash v : [A]}{\Lambda, \Gamma \vdash \mathtt{der}\ v : A}\ \text{I-DER}$$

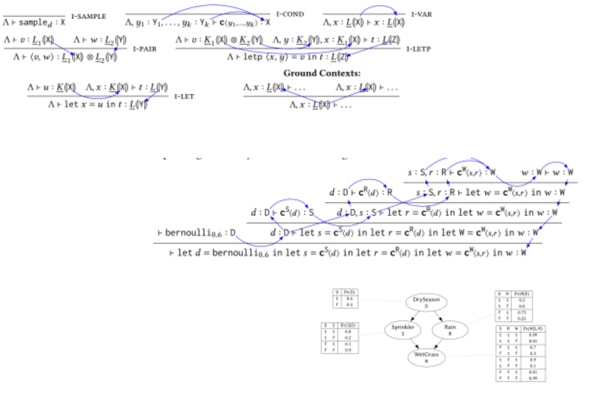
6

## Higher-Order Bayesian Networks



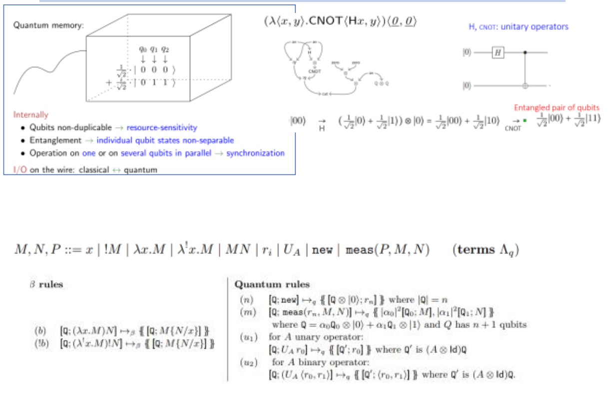

7

## Higher-Order Bayesian Networks




8

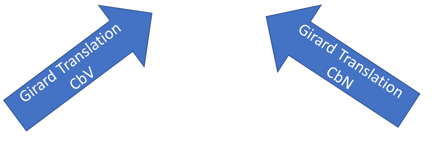
## Quantum lambda calculus




9

**LINEAR LOGIC**

Proof-nets / ! λ-calculus




CbV $\lambda$-calculus

CbN $\lambda$-calculus

10

## Plan / topics for Part 1

HANDS-ON

- Theoretical tools to study the operational properties of a system:
  - Rewrite Theory (rewriting=abstract form of program execution)
- Linear Logic and Proof-Nets.
- Bridging between lambda-calculus and functional programming:
  - Call-by-Value and Call-by Name, weak and lazy calculi.
- Beyond pure functional:
  - Probabilistic programming and Bayesian Inference:
    Probabilistic lambda calculi, Bayesian proof-nets

(Internships possible on operational aspects of probabilistic and quantum computation)

11

## Resources

- **Webpage** https://www.irif.fr/~faggian/LMFI2025
- **Lecture Notes** (by A. Middeldorp, O. Laurent, L. Ong)

12

**Operational semantics**
of formal calculi and programming languages

# Rewriting theory

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

13

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?

14

## Math formalizations…

**Example (Group Theory)**

| | | | |
|---|---|---|---|
| signature | e (constant) | $^-$ (unary, postfix) | $\cdot$ (binary, infix) |
| equations | $\mathrm{e} \cdot x \approx x$   $x^- \cdot x \approx \mathrm{e}$ | $(x \cdot y) \cdot z \approx x \cdot (y \cdot z)$ | $\mathcal{E}$ |
| theorems | $\mathrm{e}^- \approx_{\mathcal{E}} \mathrm{e}$ | $(x \cdot y)^- \approx_{\mathcal{E}} y^- \cdot x^-$ | |

rewrite rules $\mathcal{R}$

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{e} \cdot x & \to & x & x \cdot \mathrm{e} & \to & x \\
x^- \cdot x & \to & \mathrm{e} & x \cdot x^- & \to & \mathrm{e} \\
(x \cdot y) \cdot z & \to & x \cdot (y \cdot z) & x^{--} & \to & x \\
\mathrm{e}^- & \to & \mathrm{e} & (x \cdot y)^- & \to & y^- \cdot x^- \\
x^- \cdot (x \cdot y) & \to & y & x \cdot (x^- \cdot y) & \to & y
\end{array}$$

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

15

## Modelling computation

**Example (Lambda Calculus)**

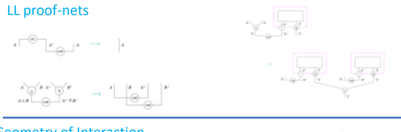
| | |
|---|---|
| signature | $\lambda$ (binds variables)   $\cdot$ (application, binary, infix) |
| terms | $M ::= x \mid (\lambda x.\, M) \mid (M \cdot M)$ |
| $\alpha$ conversion | $\lambda x.\, x \cdot y =_\alpha \lambda z.\, z \cdot y$ |
| $\beta$ reduction | $(\lambda x.\, M) \cdot N \to_\beta M[x := N]$ <br> replace free occurrences of $x$ in $M$ by $N$ <br> (and avoid variable capturing) |
| rewriting | $(\lambda x.\, x \cdot x) \cdot (\lambda x.\, x \cdot x) \to (\lambda x.\, x \cdot x) \cdot (\lambda x.\, x \cdot x)$ |
| inventor | Alonzo Church (1932) |

both Combinatory Logic and Lambda Calculus are Turing-complete

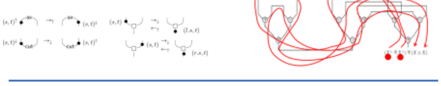
17

## Graph Rewriting

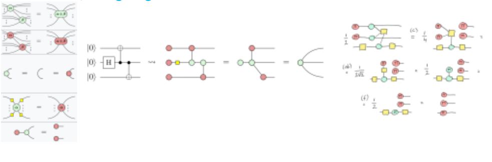
LL proof-nets



Geometry of Interaction



ZX- calculus, string diagrams..



18

## 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.

19

## 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

20

## Why a theory of rewriting matters?

- **Rewriting = abstract form of program execution**

*Rewriting theory* provides a *sound framework* for reasoning about
- **programs transformations**, such as compiler optimizations or parallel implementations,
- **program equivalence**.

21

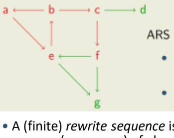
# Abstract Rewriting

Basic language

22

## ARS

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




ARS $\mathcal{A} = \langle A, \to \rangle$
- $A = \{ a, b, c, d, e, f, g \}$
- $\to = \left\{ \begin{matrix} (a, e), (b, a), (b, c), (c, d), (c, f) \\ (e, b), (e, g), (f, e), (f, g) \end{matrix} \right\}$

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

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

- rewrite sequence
  - finite $a \to e \to b \to c \to f$
  - empty $a$
  - infinite $a \to e \to b \to a \to e \to b \to \cdots$

23

- $\leftarrow$ inverse of $\to$
- $\to^*$ transitive and reflexive closure of $\to$
- ${}^*\!\leftarrow$ inverse of $\to^*$

$s \leftrightarrow_{\mathcal{R}} t$ iff $s \to_{\mathcal{R}} t$ or $t \to_{\mathcal{R}} s$

$s \leftrightarrow^*_{\mathcal{R}} t$ iff $s = s_0 \leftrightarrow_{\mathcal{R}} s_1 \leftrightarrow_{\mathcal{R}} \dots \leftrightarrow_{\mathcal{R}} s_n = t$ for $n \geq 0$

- $\leftrightarrow$ symmetric closure of $\to$
- $\leftrightarrow^*$ conversion (equivalence relation generated by $\to$) **
- $\to^+$ transitive closure of $\to$
- $\to^=$ reflexive closure of $\to$

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

$\downarrow = \to^* \cdot {}^*\!\leftarrow$

24

## Composition

- 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 $\to = \to_1 \cup \to_2$.

25

## Closure

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

$$\to \subseteq \to^*, \qquad (\to^*)^* = \to^*, \qquad \to_1 \subseteq \to_2 \text{ implies } \to_1^* \subseteq \to_2^*$$

As a consequence

$$(\to_1 \cup \to_2)^* = (\to_1^* \cup \to_2^*)^*.$$

26

### Terminology

- if $x \to^* y$ then $x$ rewrites to $y$ and $y$ is reduct of $x$
- if $x \to^* z \;{}^*\!\leftarrow y$ then $z$ is common reduct of $x$ and $y$
- if $x \leftrightarrow^* y$ then $x$ and $y$ are convertible

### Example

$$\begin{tikzcd}
a \arrow[dr] & b \arrow[l] \arrow[r] & c \arrow[r] \arrow[d] & d \\
 & e \arrow[u] \arrow[dr] & f \arrow[l] \arrow[d] & \\
 & & g &
\end{tikzcd}$$

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

27

## Normal forms model **results**

**Definition 1.1.11.** Let $\mathcal{A} = \langle A, \to \rangle$ be an ARS. An element $a \in A$ is *reducible* if there exists an element $b \in A$ with $a \to b$. A *normal form* is an element that is not reducible. The set of normal forms of $\mathcal{A}$ is denoted by $\mathsf{NF}(\mathcal{A})$ or $\mathsf{NF}(\to)$ when $A$ can be inferred from the context. An element $a \in A$ *has a normal form* if $a \to^* b$ for some normal form $b$. In that case we write $a \to^! b$.

$$\begin{tikzcd}
a \arrow[dr] & b \arrow[l] \arrow[r] & c \arrow[r] \arrow[d] & d \\
 & e \arrow[u] \arrow[dr] & f \arrow[l] \arrow[d] & \\
 & & g &
\end{tikzcd}$$

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

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

- d is normal form
- $\mathsf{NF}(\mathcal{A}) = \{\, d, g \,\}$
- $b \to^! g$

28

## Operational properties of interest

- Termination and Confluence

Existence and uniqueness of normal forms

- How to Compute

reduction strategies with good properties:

- standardization,
- normalization

29

- SN  strong normalization  termination
  - no infinite rewrite sequences
- WN  (weak) normalization
  - every element has at least one normal form
  - $\forall a \, \exists b \quad a \to^! b$
- UN  unique normal forms
  - no element has more than one normal form
  - $\forall a, b, c \quad$ if $a \to^! b$ and $a \to^! c$ then $b = c$

30

## *Termination*

**Definition 1.2.1.** Let $\mathcal{A} = \langle A, \to \rangle$ 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 $\mathcal{A}$ 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 $a \to^! b$ and $a \to^! c$ then $b = c$). The ARS $\mathcal{A}$ 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.

$$\begin{tikzcd}
a \arrow[dr] & b \arrow[l] \arrow[r] & c \arrow[r] \arrow[d] & d \\
 & e \arrow[u] \arrow[dr] & f \arrow[l] \arrow[d] & \\
 & & g &
\end{tikzcd}$$

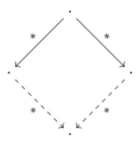
a is WN? SN?
c is WN? SN?
a or c has UN ?

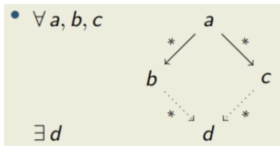
The nf are convertible?

31

## *Confluence*

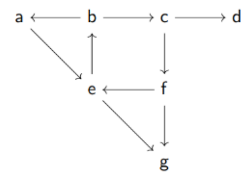
**Definition 1.2.3.** Let $\mathcal{A} = \langle A, \to \rangle$ be an ARS. An element $a \in A$ is *confluent* if for all elements $b, c \in A$ with $b \;{}^{*}\!\!\leftarrow a \to^{*} c$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.






*Every confluent* ARS *has unique normal forms.*

32




1. a is confluent?
2. f is confluent?

Bonus Point

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

33

Given

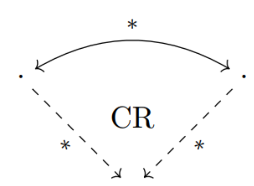
$$\mathcal{R} = \begin{cases} f(x,x) & \to & c \\ a & \to & b \\ f(x,b) & \to & d \end{cases}$$

*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 same term!
(also: different meaning for equivalent terms)

34

## Same meaning for *equivalent* terms

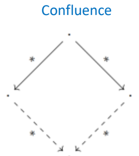



35

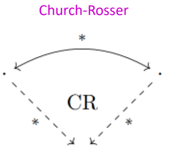
## Confluence & CR

**Definition 1.2.3.** Let $\mathcal{A} = \langle A, \to \rangle$ be an ARS. An element $a \in A$ is *confluent* if for all elements $b, c \in A$ with $b \;{}^{*}\!\!\leftarrow a \to^{*} c$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.

Confluence



Church-Rosser



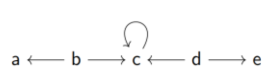

*An* ARS $\mathcal{A} = \langle A, \to \rangle$ *is confluent if and only if* $\leftrightarrow^{*} \subseteq \downarrow$.

36

**Definition 1.2.10.** An ARS $\mathcal{A} = \langle A, \to \rangle$ has *unique normal forms with respect to conversion* (UNC) if different normal forms are not convertible ($\forall a, b \in \mathsf{NF}(\mathcal{A})$ 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?




37

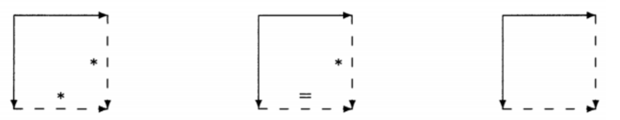
# Global vs Local

38

## Confluence

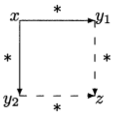
A property of term *t* is *local* if it is quantified over only *one-step reductions* from *t*;
it is *global* if it is quantified over all *rewrite sequences* from *t*.

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




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

Global property:

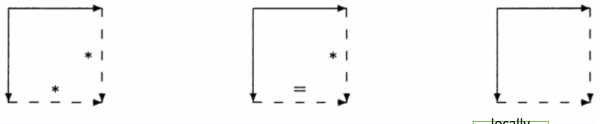



39

## 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**

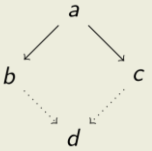



**Local confluence** Let $\mathcal{A} = \langle A, \to \rangle$ be an ARS. An element $a \in A$ is $\boxed{\text{locally confluent}}$ for all elements $b, c \in A$ with $\boxed{b \leftarrow a \to c}$ we have $b \downarrow c$. The ARS $\mathcal{A}$ is confluent if all its elements are confluent.

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

40

- diamond property $\diamond$
  - $\leftarrow \cdot \to \subseteq \to \cdot \leftarrow$
  - $\forall\, a, b, c$




$\exists\, d$

- *every ARS with diamond property is confluent*

Proof by tiling

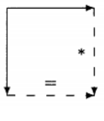
41

An ARS $\mathcal{A} = \langle A, \to \rangle$ is *strongly confluent* (SCR) if $\leftarrow \cdot \to \subseteq \to^= \cdot {}^*\!\!\leftarrow$, see Figure

***a*** Show that every strongly confluent ARS is confluent.

***b*** Does the converse also hold?

***c*** Show that an ARS $\mathcal{A} = \langle A, \to \rangle$ is confluent if and only if $\leftarrow^* \cdot \to \subseteq \to^* \cdot \leftarrow^*$




42

43

44

45

46

47

48

49