プログラミング言語研究のための (高階) 項書換え系入門

浜名 誠

群馬大学 理工学府

PPL 2016 チュートリアル

2016.3.8

書換え系の目的

計算を数学的にモデル化する一手法

必要な数学

- **集合、関係、順序**

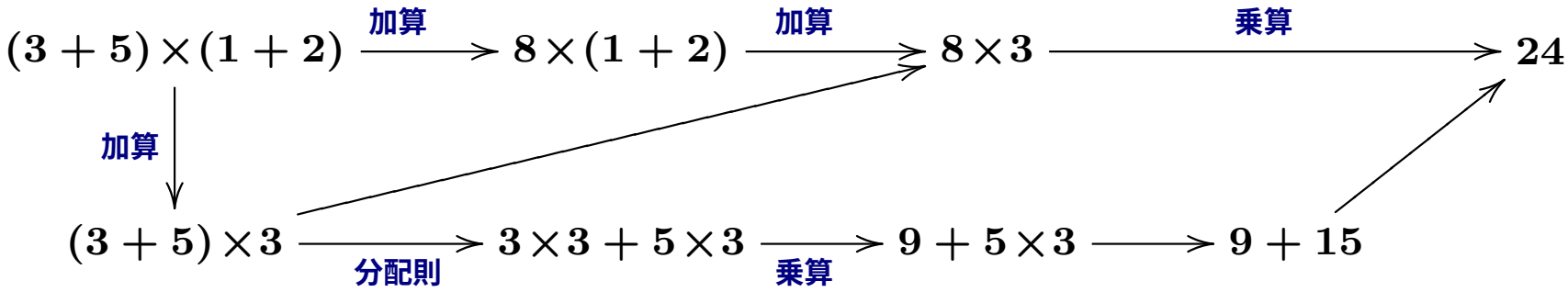
計算の「ステップ」を定式化する

書換えとしての計算

$(3+5)\times(1+2)$ を計算せよ

書換えとしての計算

$(3+5)\times(1+2)=24$ の計算の過程



- ▷ **計算の順序によらず「答え」を得ることができる** (CR)
- ▷ **「答え」は一つしかない** (CR)
- ▷ **「答え」を必ず得ることができる** (SN)

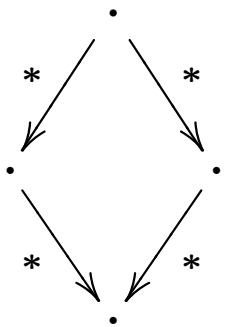
書換え理論の観点: **これらは算術のための「書換え系」の性質である**

合流性

(CR)

confluence

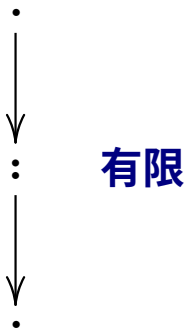
Church-Rosser



停止性

(SN)

strong normalisation



有限

書換え系とは

合流性　と　停止性

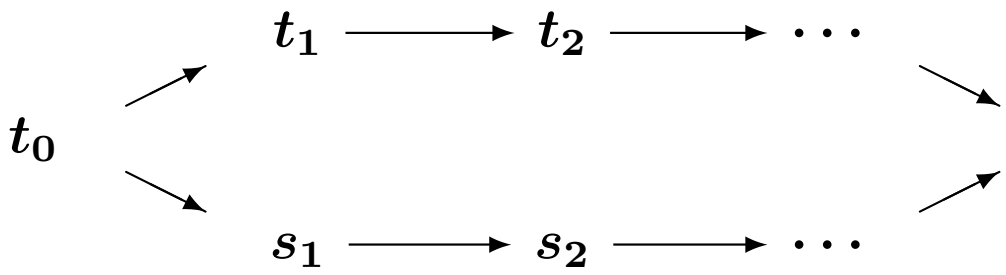
(CR)　(SN)

の学問

なぜ合流性と停止性が重要？

合流性を持ち、停止性を持つ「書換え系」は

1. **計算結果が一意に決まる事の保証**



(正規形 $=$ 計算結果)

2. **等式の決定可能** (decidable) **な証明手続きの提供**

$$s =^{?} t \quad \Leftrightarrow \quad \textbf{証明 or 反証}$$

さまざまな書換え系

- λ計算 — 書換え系の一種
- 項書換え系 (= 一階項書換え系)
- 二階書換え系
- Coq, Agda にある rewrite タクティクや計算規則
- Haskell の rewrite プラグマ、型関数

書換え系の元々の問題意識

- **群の「語の問題」** (Knuth-Bendix'70)

群の公理

$$E = \left\{ \begin{array}{ll} e \cdot x & = x \\ (x \cdot y) \cdot z & = x \cdot (y \cdot z) \\ x^{-1} \cdot x & = e \end{array} \right\}$$

$x \cdot x^{-1} = e$ か？

群の語の問題

$x \cdot x^{-1} = e$ か？

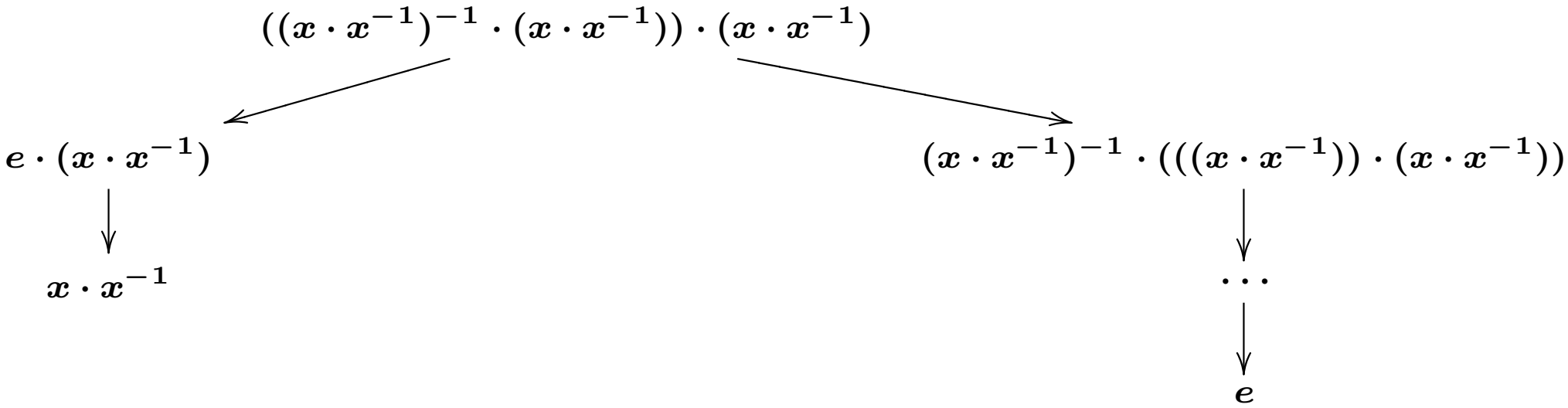
$$
\begin{aligned}
e &= (x \cdot x^{-1})^{-1} \cdot (x \cdot x^{-1}) \\
&= (x \cdot x^{-1})^{-1} \cdot (x \cdot (e \cdot x^{-1})) \\
&= (x \cdot x^{-1})^{-1} \cdot (x \cdot ((x^{-1} \cdot x) \cdot x^{-1})) \\
&= (x \cdot x^{-1})^{-1} \cdot ((x \cdot (x^{-1} \cdot x)) \cdot x^{-1}) \\
&= (x \cdot x^{-1})^{-1} \cdot (((x \cdot x^{-1}) \cdot x) \cdot x^{-1}) \\
&= (x \cdot x^{-1})^{-1} \cdot (((x \cdot x^{-1})) \cdot (x \cdot x^{-1})) \\
&= ((x \cdot x^{-1})^{-1} \cdot (x \cdot x^{-1})) \cdot (x \cdot x^{-1}) \\
&= e \cdot (x \cdot x^{-1}) \\
&= x \cdot x^{-1}
\end{aligned}
$$

どうやってこの導出を得るか？ ▶ 語の問題 (Knuth-Bendix'70)

群 – 書換え規則として

等式を「向き付け」する – 書換え系の誕生

$$E = \left\{ \begin{array}{ll} e \cdot x & \to x \\ x^{-1} \cdot x & \to e \\ (x \cdot y) \cdot z & \to x \cdot (y \cdot z) \end{array} \right\}$$

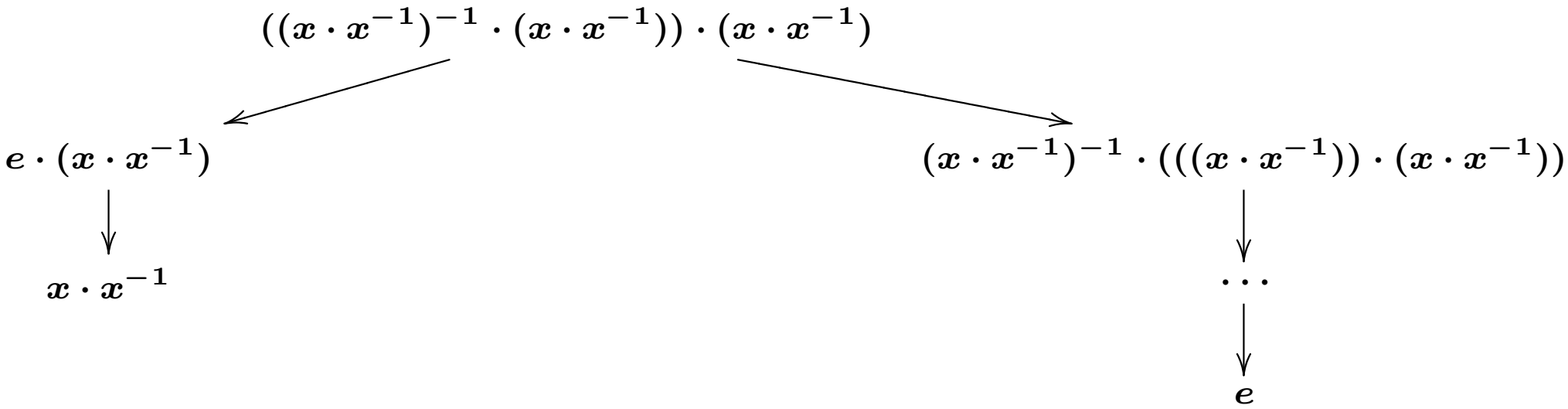


よって $x \cdot x^{-1} = e$

群 – 書換え規則として

等式を「向き付け」する – 書換え系の誕生

$$E = \left\{ \begin{array}{ll} e \cdot x & \to x \\ x^{-1} \cdot x & \to e \\ (x \cdot y) \cdot z & \to x \cdot (y \cdot z) \end{array} \right\}$$



「答え」が二つある

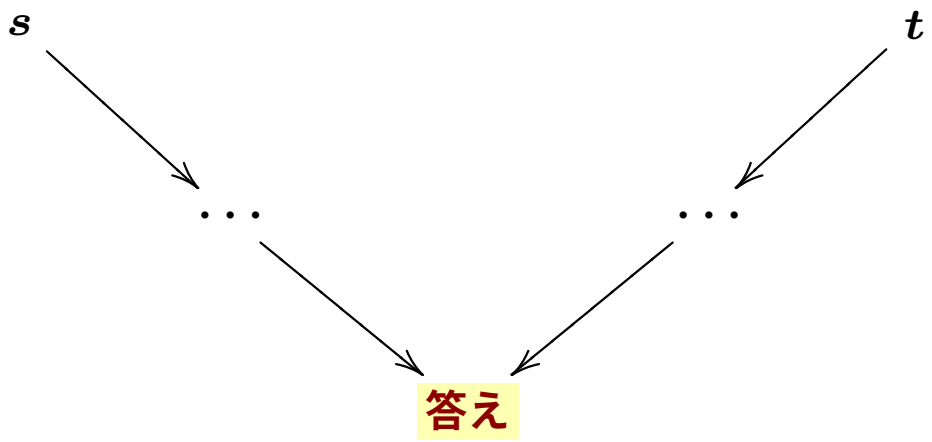
- **先の数の計算とは異なる。⇐ 合流性を持たない。**

もし $s = t$ のときは必ず s と t の「答え」が一つしかなければ？

群 – 書換えによる等式の証明

$$s = t$$

を示すために



とすればよい

答えを正規形と言う

完備化手続きによる等式の証明 [Knuth-Bendix'70]

$$E = \left\{ \begin{array}{ll} e \cdot x & = x \\ x^{-1} \cdot x & = e \\ (x \cdot y) \cdot z & = x \cdot (y \cdot z) \end{array} \right\}$$

↓完備化手続き

$$R = \left\{ \begin{array}{llll} e \cdot x & \to x & x \cdot e & \to x \\ x^{-1} \cdot x & \to e & x \cdot x^{-1} & \to e \\ (x \cdot y) \cdot z & \to x \cdot (y \cdot z) & (x^{-1})^{-1} & \to x \\ e^{-1} & \to e & (x \cdot y)^{-1} & \to y^{-1} \cdot x^{-1} \\ x^{-1} \cdot (x \cdot y) & \to y & x \cdot (x^{-1} \cdot y) & \to y \end{array} \right\}$$

1. R を等式と思うと、それらはすべて E のもとでで正しい (E から導ける)
2. E から $s = t$ が導出できる $\Leftrightarrow$ $s \to^* \exists u \leftarrow^* t$ (合流性 CR)
3. R を使った書換えは必ず止る (停止性 SN)

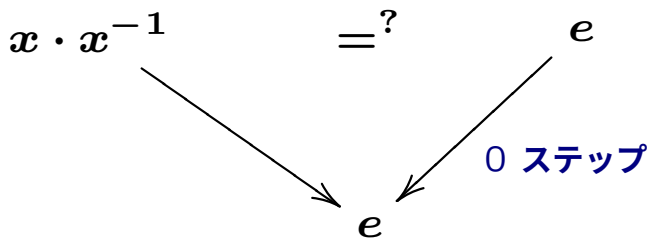
完備化手続きによる等式の証明 [Knuth-Bendix'70]

$$E = \left\{ \begin{array}{ll} e \cdot x & = x \\ x^{-1} \cdot x & = e \\ (x \cdot y) \cdot z & = x \cdot (y \cdot z) \end{array} \right\}$$

↓完備化手続き

$$R = \left\{ \begin{array}{llll} e \cdot x & \to x & x \cdot e & \to x \\ x^{-1} \cdot x & \to e & x \cdot x^{-1} & \to e \\ (x \cdot y) \cdot z & \to x \cdot (y \cdot z) & (x^{-1})^{-1} & \to x \\ e^{-1} & \to e & (x \cdot y)^{-1} & \to y^{-1} \cdot x^{-1} \\ x^{-1} \cdot (x \cdot y) & \to y & x \cdot (x^{-1} \cdot y) & \to y \end{array} \right\}$$

4. **R を使った書換えは唯一の正規形（「答え」）を持つ**



文献

- **坂井 公**. Knuth-Bendix **の完備化手続きとその応用**. **コンピュータソフトウェア**, 4(1), pp.2-22. 1987.
- **外山 芳人**. **項書き換えシステム入門**. **信学技報**, SS98-15, pp.31-38, 1998.
- **外山 芳人**. **完備化による等式証明**. **人工知能学会誌**, Vol.16, No.5, pp.668-674, 2001.

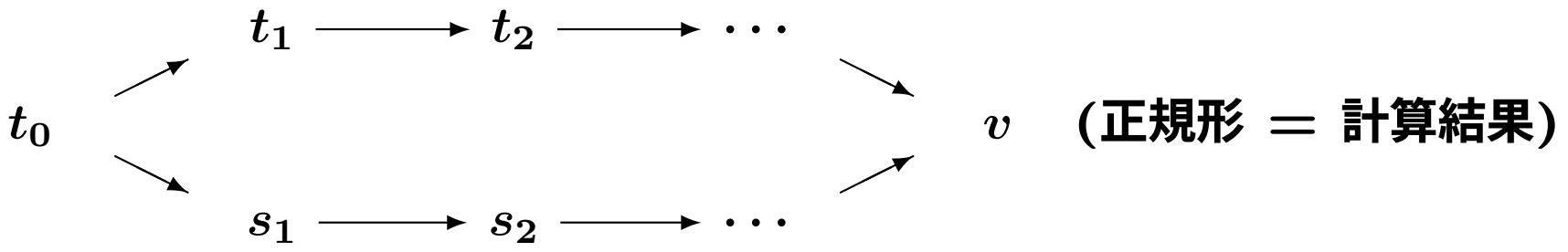
- **完備化のツール**
  - KBCV (**インスブルク大**)
  - Maxcomp (JAIST)
  - mkbTT (**インスブルク大**)
  - Waldmeister (Max-Planck **研究所**)

`http://cl-informatik.uibk.ac.at/users/ami/15isr/tools.php`

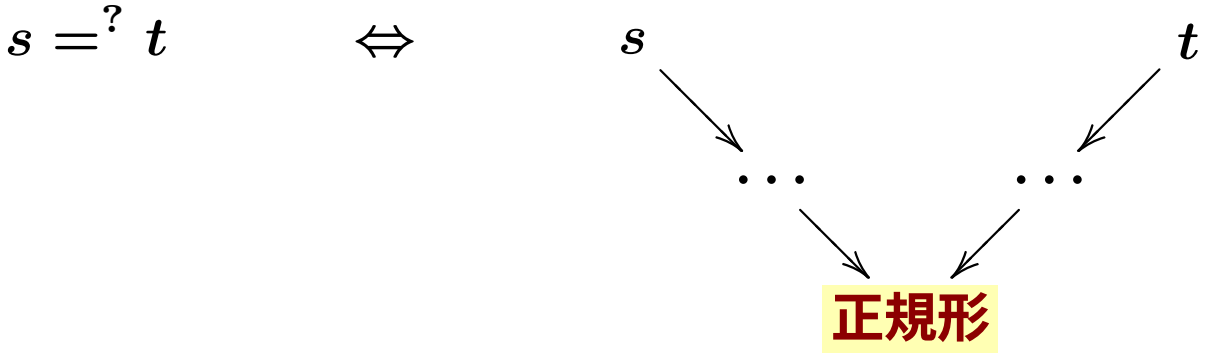
なぜ合流性と停止性が重要？ — プログラム言語の観点

書換え手法の効用

1. **計算**: **結果が一意である事の保証**



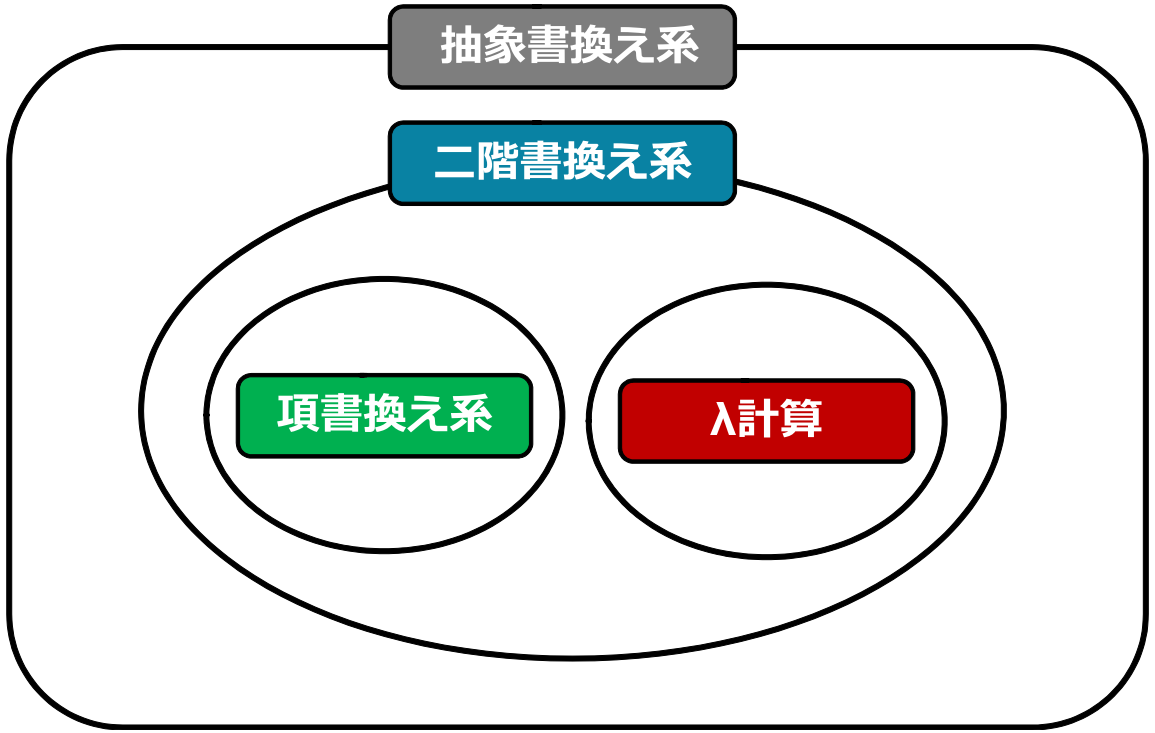
2. **論理 — 検証**: **等式の決定可能** (decidable) **手続きの提供**



目標

どうしたら CR と SN が導けるか？ 条件を探る

1. **抽象書換え系** — **最も一般的な状況**
2. **項書換え系** (Term Rewriting Systems, TRS) $f(x, C(y)) \to g(x, x, A)$
3. **二階書換え系** (Second-order Rewrite System) $(\lambda x.M[x])\ N \to M[x := N]$



抽象書換え系

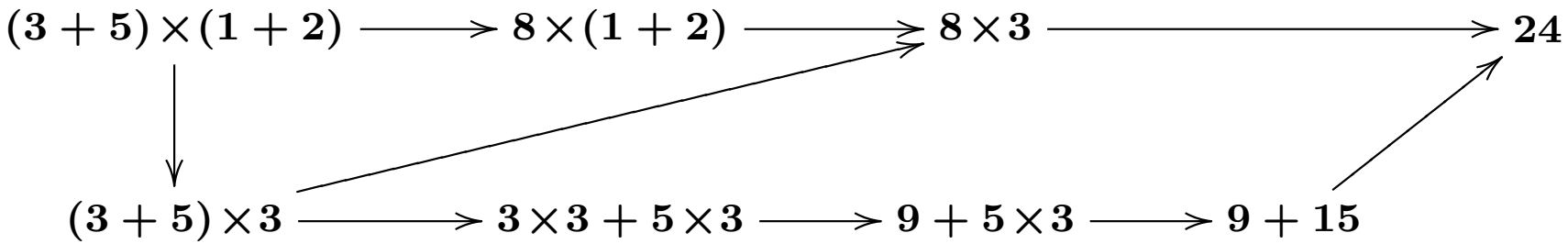
抽象書換え系の定義

抽象化したセッティング・・・ 要素の構造は忘れる

抽象書換え系 $A = (A, \to)$ **とは**

A:**集合**, $\to$: A **上の** 2 **項関係**

例:



$A = \{(3+5)\times(1+2),\ 8\times(1+2), \cdots\}$

A **上の関係** $\to$ **は上の図で定義されるもの、つまり**

$\to\ =\ \{((3+5)\times(1+2), 8\times(1+2)),\ (8\times(1+2), 8\times 3), \ldots\}$

1 ステップの書換え

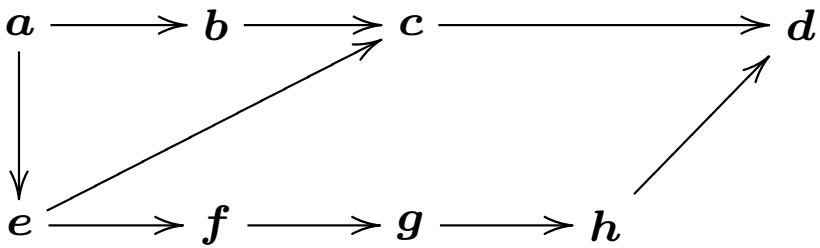
抽象書換え系の定義

抽象化したセッティング · · · 要素の構造は忘れる

抽象書換え系 $A = (A, \to)$ **とは**

A:**集合**, $\to$: A **上の** 2 **項関係**

例:



$A = \{a, b, c, d, e, f, g, h\}$

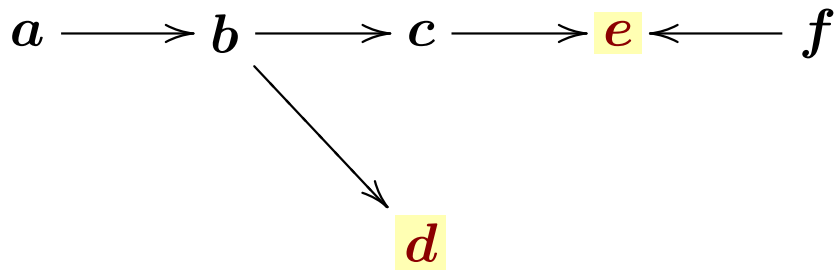
A **上の関係** $\to$ **は上で定義されるもの、つまり**

$\to\ = \{(a, b),\ (b, c), \ldots\}$

正規形

定義　抽象書換え系 $A = (A, \to)$ においてそれ以上書換えられない要素を**正規形**と呼ぶ。

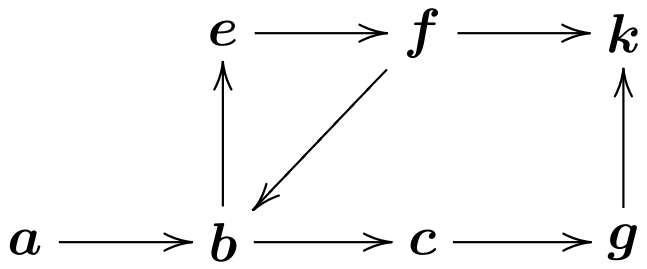
例:



定義 要素 a の正規形がただ一つしかないとき、a は**唯一正規形** (Unique Normal form, UN) を持つ、という。

「A が UN」 $\overset{def}{\iff}$ すべての $a \in A$ が UN

どうやって唯一正規形性 (UN) をチェックするか？



仮定: **計算過程をメモリしない、と仮定する。ループではなく発散するような場合もあるため**

合流性: CR

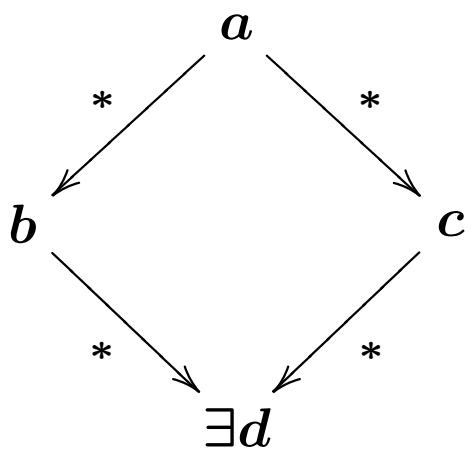
▶ UN を導くための「より強い」条件

反射推移閉包 $\rightarrow^* = \bigcup_{i \geq 0} \rightarrow^i$　　0 **ステップ以上の書換え**

定義 **抽象書換え系** $A = (A, \rightarrow)$、 $a \in A$ **は合流性** (Confluence, CR) **を持つ**

$$a \rightarrow^* b \ \wedge \ a \rightarrow^* c \ \Rightarrow \ \exists d \in A.\ b \rightarrow^* d \ \wedge \ c \rightarrow^* d$$

つまり



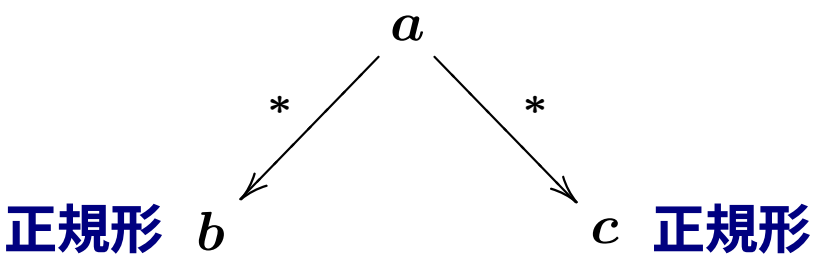
「A **が合流性** (CR) **を持つ**」 $\overset{def}{\Longleftrightarrow}$ **すべての** $a \in A$ **ついて** a **は合流性を持つ**

性質

補題

A **が合流性** (CR) **を持つ** $\Rightarrow$ A **は** UN(**唯一正規形**) **を持つ**

証明:

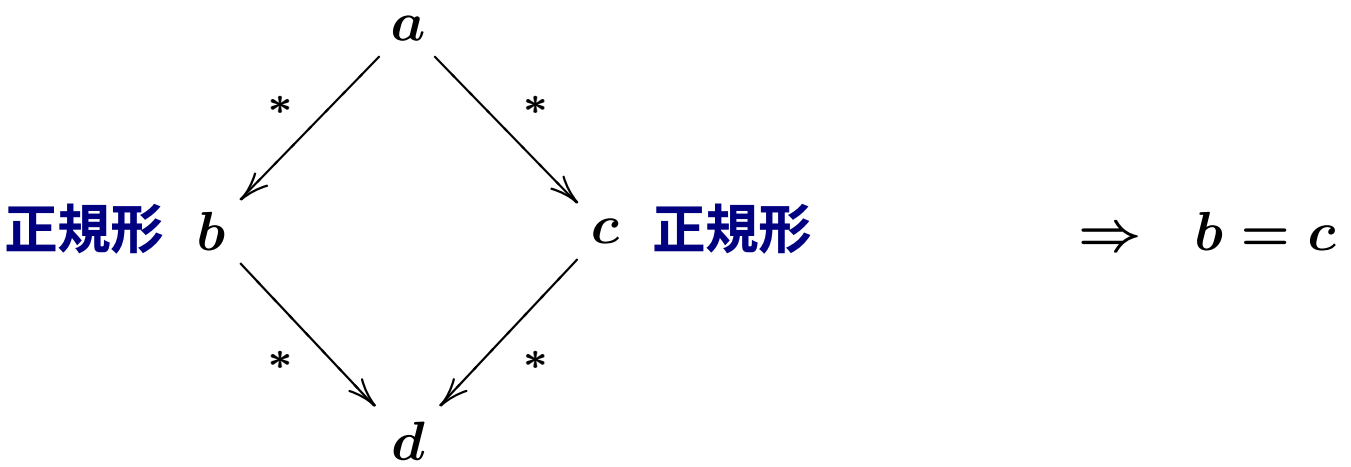


性質

補題

A **が合流性** (CR) **を持つ** $\Rightarrow$ A **は** UN(**唯一正規形**) **を持つ**

証明:



$\Rightarrow \quad b = c$

どのように CR **をチェックするか？**

- CR **を導ける条件を探す** UN $\Leftarrow$ CR $\Leftarrow$??

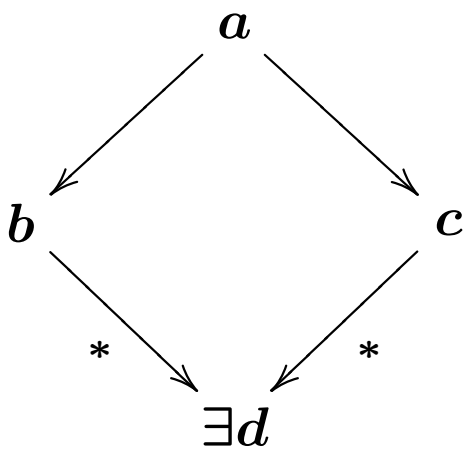
弱合流性: WCR

定義

抽象書換え系 $A = (A, \rightarrow)$**、** $a \in A$ **は弱合流性** (Weak Confluence, WCR) **を持つ**

$$a \rightarrow b \ \wedge \ a \rightarrow c \ \Rightarrow \ \exists d \in A. b \rightarrow^* d \ \wedge \ c \rightarrow^* d$$

1 **ステップだけでよい**



「A **が弱合流性を持つ」** $\overset{def}{\Longleftrightarrow}$ **すべての** $a \in A$ **について** a **は弱合流性** (WCR) **を持つ**

停止性

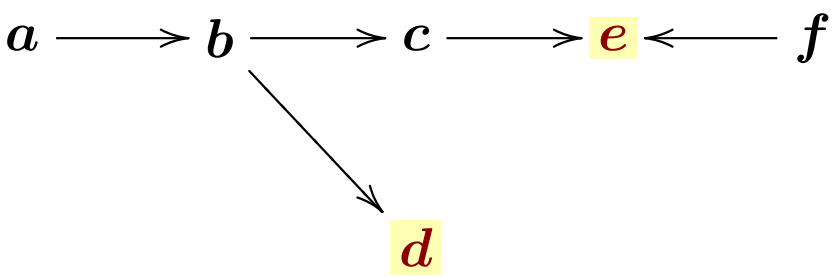
定義 **要素 a は 停止性を持つ** (Strongly Normalization, SN)

$\overset{def}{\Longleftrightarrow}$ **$a \in A$ から始まる無限の書換え列**

$$a \longrightarrow a_1 \longrightarrow a_2 \longrightarrow \ldots$$

が存在しない。

抽象書換え系



は停止性を持つ

停止性

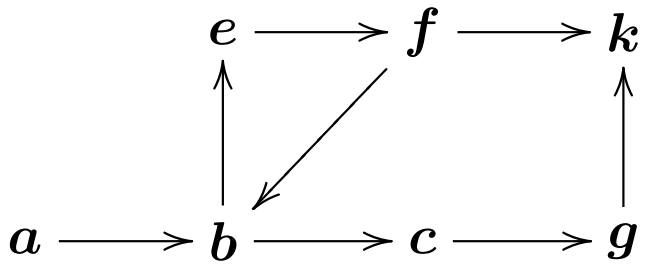
定義 **要素** a **は** **停止性を持つ** (Strongly Normalization, SN)

$\overset{def}{\Longleftrightarrow}$ $a \in A$ **から始まる無限の書換え列**

$$a \longrightarrow a_1 \longrightarrow a_2 \longrightarrow \ldots$$

が存在しない。

抽象書換え系



は停止性を持たない

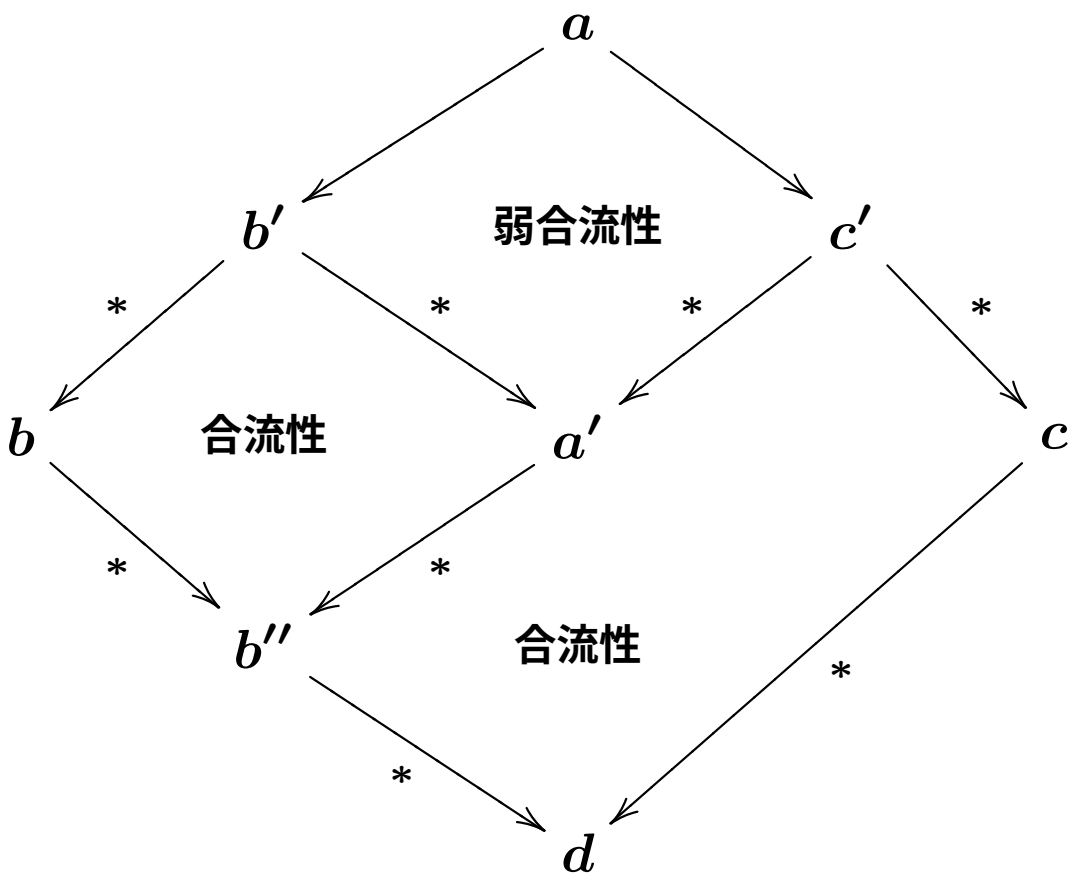
Newmann の補題

定理

抽象書換え系 A が 停止性 (SN) **を持ち** & **弱合流性** (WCR) **を持つ**

$\Rightarrow$ A **は合流性** (CR) **を持つ**

証明: $\longrightarrow$ **についての整礎帰納法** (well-founded induction) **による。**



まとめ

A : **停止性** (SN)&**弱合流性** (WCR) $\Rightarrow$ A : **合流性** (CR) $\Rightarrow$ A : **唯一正規形** (UN)

ではどうやって WCR **や** SN **をチェックするか？▶ 重要な問題**

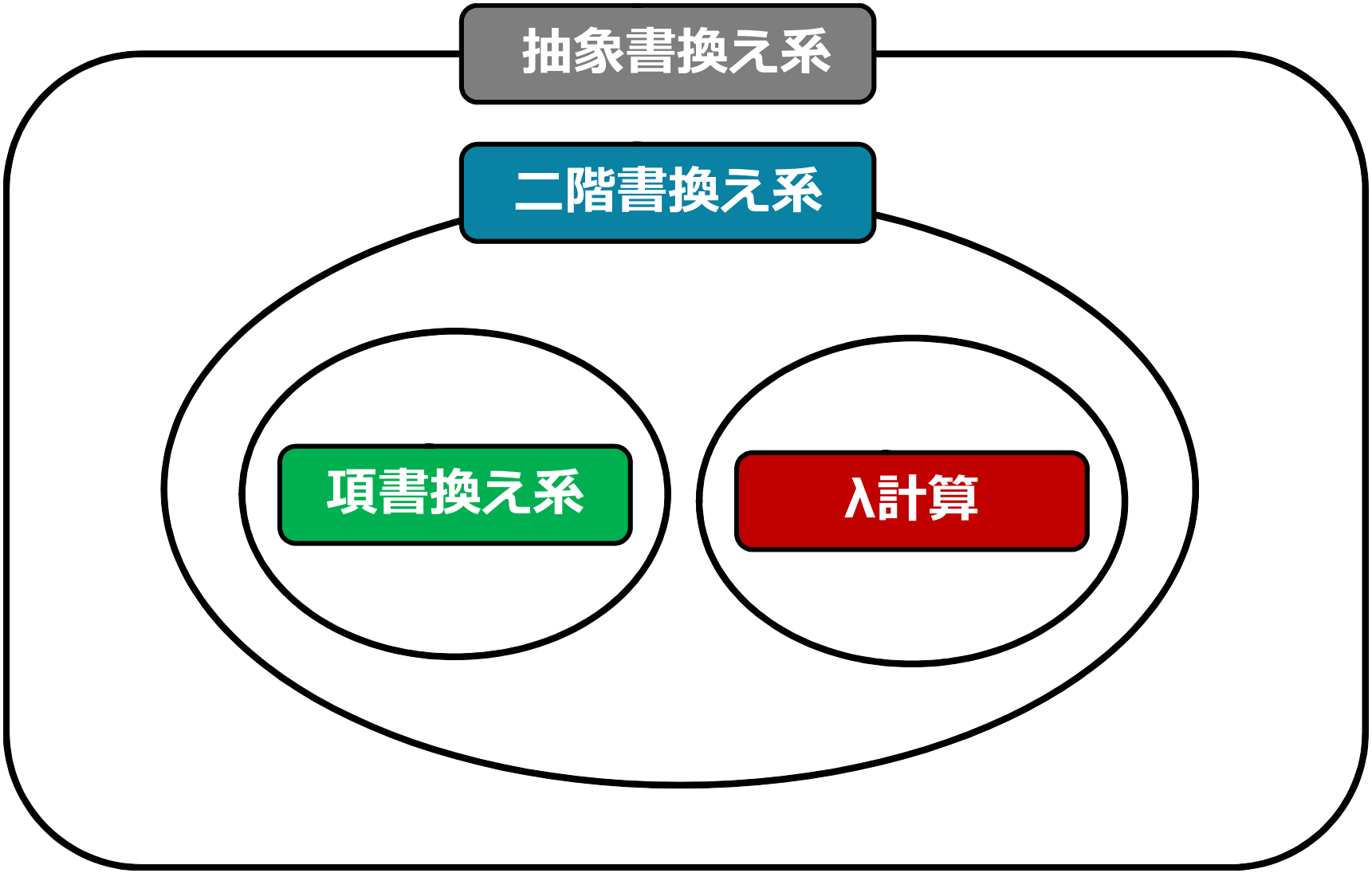
元々の問題

抽象書換え系 は 「具体的」な書換え系の「関係ステップ」だけ着目してる

実際は、書換え関係は $\to_R$ **は (少数の) 書換え規則** R **から生成される**

$$R = \left\{ \begin{array}{llll} e \cdot x & \to x & x \cdot e & \to x \\ x^{-1} \cdot x & \to e & x \cdot x^{-1} & \to e \\ (x \cdot y) \cdot z & \to x \cdot (y \cdot z) & (x^{-1})^{-1} & \to x \\ e^{-1} & \to e & (x \cdot y)^{-1} & \to y^{-1} \cdot x^{-1} \\ x^{-1} \cdot (x \cdot y) & \to y & x \cdot (x^{-1} \cdot y) & \to y \end{array} \right\}$$

演習問題: **これは停止するか？**


抽象書換え系
二階書換え系
項書換え系
λ計算

項書換え系

(Term Rewriting System, TRS)

項書換え系

加算

$$\mathbf{0} + y \to y$$

$$S(x) + y \to S(x + y)$$

階乗

$$fact(\mathbf{0}) \to S(\mathbf{0})$$

$$fact(S(x)) \to fact(x) * S(x)$$

Combinatory Logic

$$((S \cdot x) \cdot y) \cdot z \to (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \to x$$

項書換え系

群

$$e \cdot x \to x \qquad x \cdot e \to x$$

$$x^{-1} \cdot x \to e \qquad x \cdot x^{-1} \to e$$

$$(x \cdot y) \cdot z \to x \cdot (y \cdot z) \qquad (x^{-1})^{-1} \to x$$

$$e^{-1} \to e \qquad (x \cdot y)^{-1} \to y^{-1} \cdot x^{-1}$$

$$x^{-1} \cdot (x \cdot y) \to y \qquad x \cdot (x^{-1} \cdot y) \to y$$

項書換え系

加算のための項書換え系 $\mathcal{R}$

$$0 + y \to y$$
$$S(x) + y \to S(x + y)$$

書換え:

$$0 + (S(S(0)) + S(0)) \to_{\mathcal{R}} 0 + S(S(0) + S(0)) \to_{\mathcal{R}} 0 + S(S(S(0))) \to_{\mathcal{R}} S(S(S(0)))$$

正規形

(一階) 項 $\quad \text{Terms} \ni t ::= x \mid f(t_1, \ldots, t_m)$

書換え関係 $\quad \to_{\mathcal{R}} \subseteq \text{Terms} \times \text{Terms}$

は以下で生成する: **注**: $(\text{Terms}, \to_{\mathcal{R}})$ **は抽象書換え系を成す**

$$\frac{l \to r \in \mathcal{R}}{l\theta \to_{\mathcal{R}} r\theta} \qquad \frac{s \to_{\mathcal{R}} t}{f(\ldots, s, \ldots) \to_{\mathcal{R}} f(\ldots, t, \ldots)}$$

パターンマッチ $\qquad$ **コンテクストで閉じる**

ここで θ は変数を項へ具体化する代入 (matcher)

合流性 (CR) の判定方法

TRS $\mathcal{R}$ の形の判定基準の探求

(i) $\mathcal{R}$ が停止性 (SN) 持ち、すべての危険対が合同 [Knuth,Bendix'70]

(ii) $\mathcal{R}$ が停止性 (SN) 持ち、ルール間に重なりがない

(iii) (Orthogonality) 左線形で重なりがない [Rosen'73]

(iv) 左線形で paralell closed [Huet'80]
…様々な改良 [外山'81,88][van Oostrom'95] や手法多数

(v) (Persistence) $\mathcal{R}$ に型を付けたものが CR なら、元の $\mathcal{R}$ は CR [青戸, 外山'97]

注: (i)(ii) は Newmann の補題 (SN + 弱合流性 (WCR) $\Rightarrow$ CR) を使っている

▶ 合流性の自動判定ツール

`http://cl-informatik.uibk.ac.at/users/ami/15isr/tools.php`

項書換え系

加算

$$0 + y \to y$$

$$S(x) + y \to S(x + y)$$

階乗

$$fact(0) \to S(0)$$

$$fact(S(x)) \to fact(x) * S(x)$$

これらはすべて SN

また「重なりがない」から WCR

ゆえにすべて合流性 (CR) **を持つ**

項書換え系

$$\mathsf{or}(x, \mathsf{true}) \longrightarrow \mathsf{true}$$
$$\mathsf{or}(\mathsf{true}, x) \longrightarrow \mathsf{true}$$
$$\mathsf{or}(\mathsf{false}, \mathsf{false}) \longrightarrow \mathsf{false}$$

SN **である**

重なっているけど、危険対が合同なので合流性 (CR) **を持つ**

応用: Haskell の型関数 — 型レベルの計算

`Elem` は型上の関数 `Elem c` は「コレクション型 `c`」のエレメントの型を返す

```
type family Elem :: * -> *
type instance Elem [e]    = e
```

応用: Haskell の型関数 — 型レベルの計算

```
data Z; data S a
type family Add :: * -> * -> *
type instance Add    Z  y = y
type instance Add (S x) y = S (Add x y)
```

型インスタンスの制限 (GHC のドキュメントより)

1. **任意の二つの左辺に重なりがない、あるいはあった場合は重複するインスタンスの右辺が共通部分の型に対しては一致する事。**
2. `type instance F t1 .. tn = t`
   `t` **中のすべての型族適用** ($G\ \overrightarrow{s}$) **は以下を満たす**
   - ▷ $\overrightarrow{s}$ **に、型族構築子が一つも含まれない。**
   - ▷ $\overrightarrow{s}$ **中の記号の数が、**`t1 .. tn` **における数よりも厳密に小さい。**
   - ▷ **全ての型変数** `a` **について、**`a` **が** $\overrightarrow{s}$ **に現れる回数が、**`t1 .. tn` **に現れる回数を超えない。**

▶ CR **の基準** (i) **と** SN **の保証 の事を言っている**

応用: Haskell の型関数 — 型レベルの計算

1. **任意の二つの左辺に重なりがない、あるいはあった場合は重複するインスタンスの右辺が共通部分の型に対しては一致する事。**

```
type instance F (a, Int) = [a]
type instance F (Int, b) = [b]    -- 重複が許される
```

二階書換え系

Second-order rewriting sysmte

二階書換え系 [Klop'80]

- ▷ **項を 変数束縛** $(x.t)$**、高階変数** ($\mathbf{F}$, $\mathbf{M}$,...) **で拡張**
- ▷ **二階書換え系の例** (1): map

$$\mathsf{map}(x.\mathbf{F}[x], \mathsf{nil}) \longrightarrow \mathsf{nil}$$
$$\mathsf{map}(x.\mathbf{F}[x], \mathbf{X} : \mathbf{XS}) \longrightarrow \mathbf{F}[\mathbf{X}] : \mathsf{map}(x.\mathbf{F}[x], \mathbf{XS})$$

- ▷ **二階書換え系の例** (2): $\boldsymbol{\lambda}$ **計算**

$$\mathsf{app}(\boldsymbol{\lambda}(x.\mathbf{M}[x]),\ \mathbf{N}) \longrightarrow \mathbf{M}[\mathbf{N}]$$
$$\mathsf{app}(\boldsymbol{\lambda}(x.\mathbf{M}),\ x) \longrightarrow \mathbf{M}$$

- ▷ **困難さ**: **一階項書換え系の技術、性質を、単純には持ってこれない**
  成立しなかったり (モジュラ性)、新しい発見の必要性
- ▷ **二階代数理論** (Second-order algebraic theory) [Fiore, et al.'10] **と対応**

書換え系とそのモデル

書換え系	モデル
一階項	Σ-代数 A
一階項書換え系	順序付き Σ-代数 $(A, >_A)$
SN な一階項書換え系	順序付き Σ-代数 $(A, >_A)$ で $>_A$ が整礎 応用 — 解釈による停止性の証明手法
一階項 書換え関係 $\rightarrow_R$ 書換え関係 $\rightarrow_{R \oplus S}$	集合の圏 Set の上の自由モナド T_Σ 前順序の圏 Pre の上の自由モナド T_R モナドの和 $T_{R+S} \cong T_R + T_S$ — 合流性のモジュラリティの圏論的別証明 [Lüth, Ghani'97]
二階項 二階項書換え系	Σ-モノイド $(A, >_A)$ 順序付き Σ-モノイド $(A, >_A)$ [浜名'05] 応用: 高階意味ラベリングによる停止性の証明手法 [浜名'07] cf. International Summer School チュートリアル [浜名'15]

書換え系の応用例

カテゴリカル・コンビネータ

カルテシアン閉圏 $\mathcal{C}$: 圏で積とべきを持つもの:

$$\frac{f : A \times B \longrightarrow C}{\mathrm{cur}(f) : A \longrightarrow B \Rightarrow C}$$

等式による公理化が出来る [Lambek-Scott]

Strong Categorical Combinatory Logic [Curien'86]

(ass)	$(x \circ y) \circ z$	$=$	$x \circ (y \circ z)$
(idL)	$\mathrm{Id} \circ x$	$=$	x
(idR)	$x \circ \mathrm{Id}$	$=$	x
(fst)	$\mathbf{Fst} \circ \langle x, y \rangle$	$=$	x
(snd)	$\mathbf{Snd} \circ \langle x, y \rangle$	$=$	y
(dpair)	$\langle x, y \rangle \circ z$	$=$	$\langle x \circ z, y \circ z \rangle$
(beta)	$\mathbf{App} \circ \langle \mathbf{\Lambda}(x), y \rangle$	$=$	$x \circ \langle \mathrm{Id}, y \rangle$
(d$\mathbf{\Lambda}$)	$\mathbf{\Lambda}(x) \circ y$	$=$	$\mathbf{\Lambda}(x \circ \langle y \circ \mathbf{Fst}, \mathbf{Snd} \rangle)$
(ai)	$\mathbf{\Lambda}(\mathbf{App})$	$=$	Id
(fsi)	$\langle \mathbf{Fst}, \mathbf{Snd} \rangle$	$=$	Id

カテゴリカル・コンビネータ

先の公理をより詳細化。

∘ **の他に「·」という演算子の導入** (explicit substitution) **のようなもの**

Weak Categorical Combinatory Logic [Curien'86]

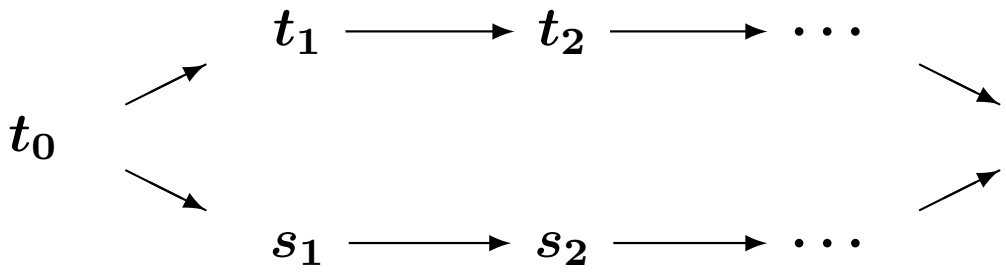
(idL)	$\mathbf{Id} \cdot x$	$\longrightarrow$	x
(ass)	$(x \circ y) \cdot z$	$\longrightarrow$	$x \cdot (y \cdot z)$
(fst)	$\mathbf{Fst} \cdot \langle x, y \rangle$	$\longrightarrow$	x
(snd)	$\mathbf{Snd} \cdot \langle x, y \rangle$	$\longrightarrow$	y
(dpair)	$\langle x, y \rangle \cdot z$	$\longrightarrow$	$\langle x \cdot z, y \cdot z \rangle$
(app)	$\mathbf{App} \circ \langle x, y \rangle$	$\longrightarrow$	$x \circ y$
(d$\mathbf{\Lambda}$)	$(\mathbf{\Lambda}(x) \cdot y) \cdot z$	$\longrightarrow$	$\mathbf{\Lambda}(x \cdot \langle y, z \rangle)$
+(Quote)			

$\rightsquigarrow$ Categorical abstract machine (CAM)　　$\rightsquigarrow$ Caml **の基礎となる**
[Cousineuau, Curien, Mauny, Sci. Comp. Prog.'86]

書換え手法 vs 代替手法

なぜ合流性と停止性が必要だったか

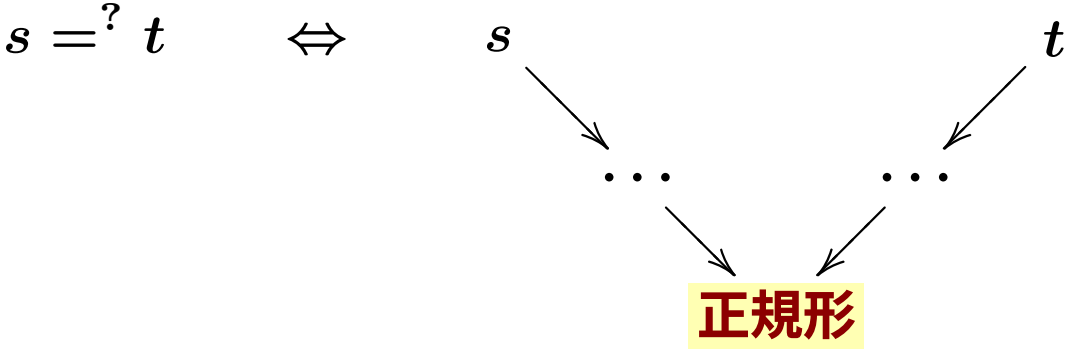
1. **書換え手法**: **計算結果が一意である事の保証**



(正規形 = 計算結果)

代替手法 ▶ 操作的意味論を直接与える、計算戦略の固定 (call-by-value **等**)

2. **等式の証明手続きの提供**



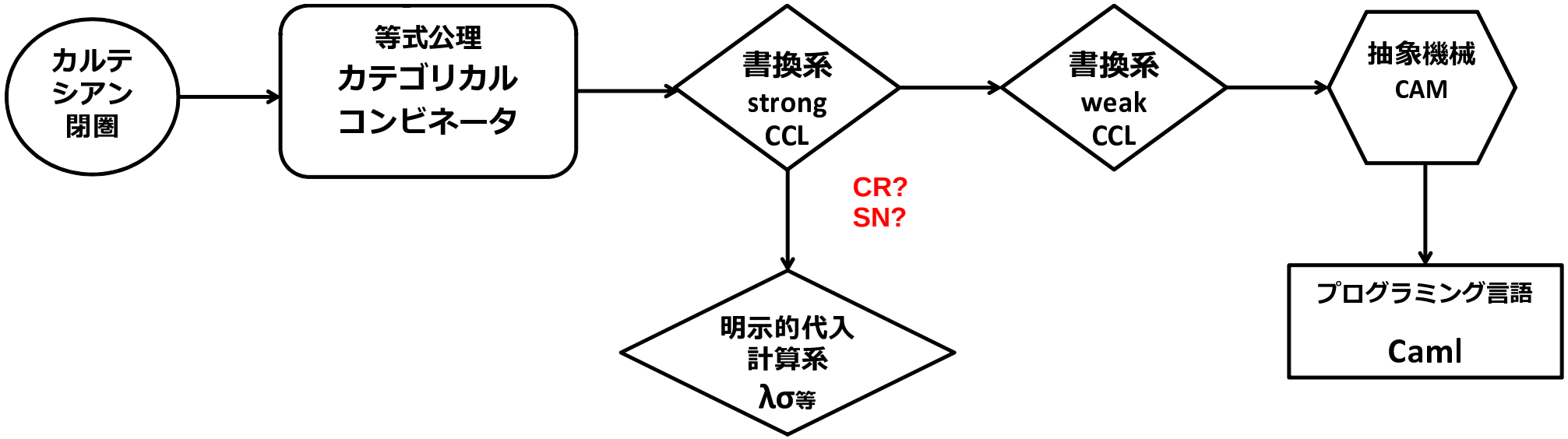
代替手法 ▶ Normalisation by Evaluation**、双模倣、論理関係**

展望: 書換え系とプログラミング言語

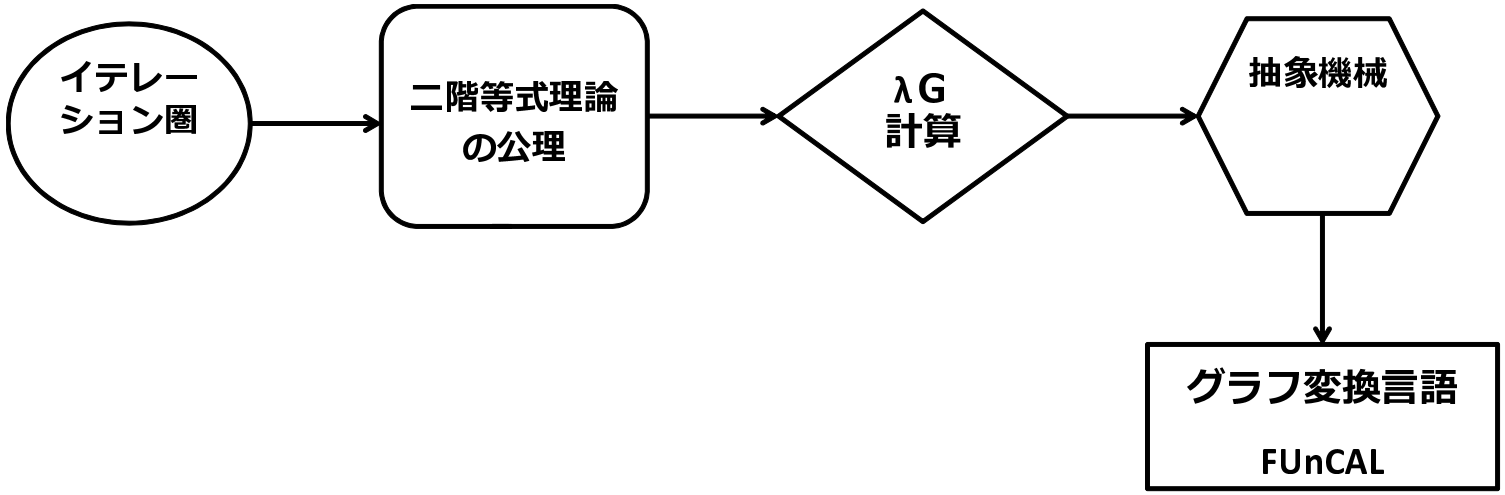
- ▷ **新しい計算体系デザインの基礎、アイデアとしての有用性**:
  等式による定式化 (代数構造) $\rightsquigarrow$ 計算規則の抽出
- ▷ **二階書換え系の展開**
  二階代数理論 (Second-order algebraic theory [Fiore, et al.'10] **と対応**
  - – **モナド的計算エフェクト、ジャンプ、継続、量子計算**
    [Staton LICS'13][Fiore,Staton LICS'14][Staton POPL'15] **は**
    二階代数理論で公理化できる
  - – $\Rightarrow$ **二階書換理論の応用 (等式理論の決定可能性、**SN **な簡約規則など)**

展望: 書換え系とプログラミング言語

カテゴリカル・コンビネータ [Curien,Hardin, **横内ら** 1980 **年代**]

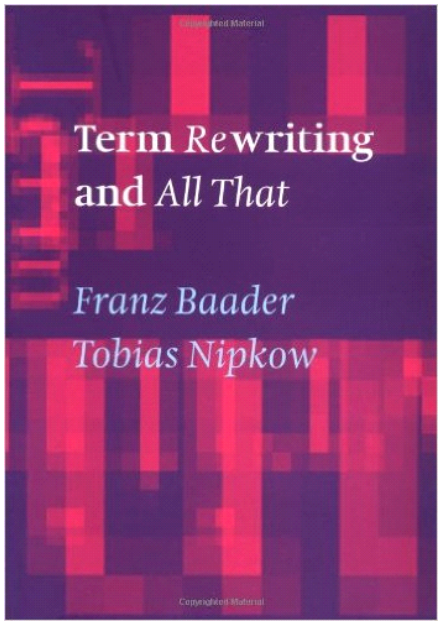


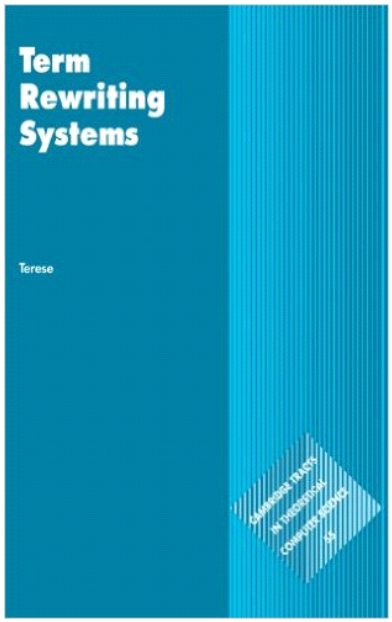
グラフ変換の計算系 [**浜名**, **松田**, **浅田** 2015〜]



Baader-Nipkow

標準的教科書
オススメ



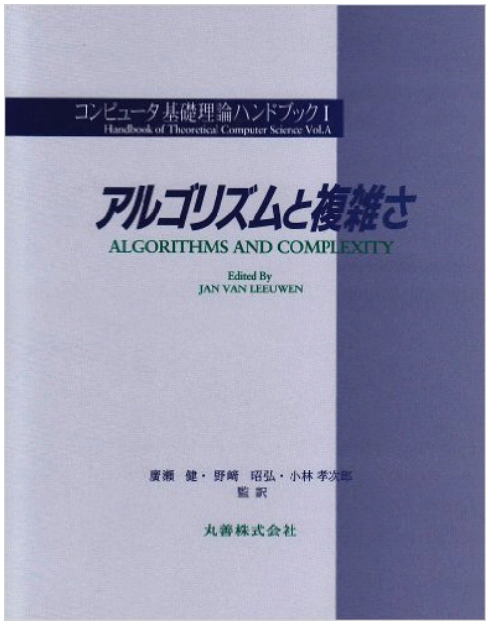


TERSE
オランダの人達

分厚い
辞書的に使う

コンピュータ基礎理論ハンドブック

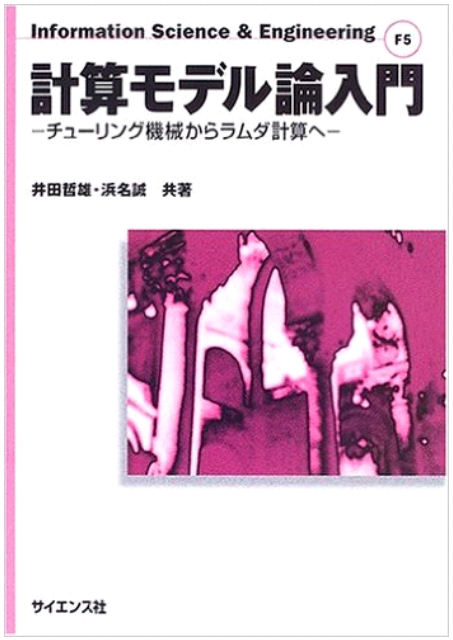
章 書換え系



井田哲雄、岩波書店

項書換え系、等式論理、代数

井田、浜名、サイエンス社



抽象書換え系

付録

CR あるいは SN でない例

合流性 (CR) **を持たない書換え系**

$$\begin{cases} \text{coin} & \to \textbf{裏} \\ \text{coin} & \to \textbf{表} \end{cases}$$

停止性 (SN) **を持たない書換え系**

$$\begin{cases} \Omega & \to \Omega \end{cases} \qquad\qquad \begin{cases} a & \to f(a) \end{cases}$$

書換え: $\Omega \to \Omega \to \cdots$

書換え: $a \to f(a) \to f(f(a)) \to \cdots$

カテゴリカル・コンビネータ

Strong Categorical Combinatory Logic [Curien'86]

(ass)	$(x \circ y) \circ z$	$\longrightarrow$	$x \circ (y \circ z)$
(idL)	$\mathrm{Id} \circ x$	$\longrightarrow$	x
...			
(fst)	$\mathbf{Fst} \circ \langle x, y \rangle$	$\longrightarrow$	x
(beta)	$\mathbf{App} \circ \langle \mathbf{\Lambda}(x), y \rangle$	$\longrightarrow$	$x \circ \langle \mathrm{Id}, y \rangle$
(d$\mathbf{\Lambda}$)	$\mathbf{\Lambda}(x) \circ y$	$\longrightarrow$	$\mathbf{\Lambda}(x \circ \langle y \circ \mathbf{Fst}, \mathbf{Snd} \rangle)$
(fsi)	$\langle \mathbf{Fst}, \mathbf{Snd} \rangle$	$\longrightarrow$	Id

は型無しのとき合流性 (CR) **を持たない** (cf. **型無しλ 計算**+SP **は 合流性がない** [Klop'80])

- ▷ **これを** CR **にできるか？**
  – **型なしの場合**: **項の形を制限する** [**横内** TCS'89] [Hardin TCS'89]
  項にアリティを付ける [**横内** SIAM J.'90]

- ▷ **その後の様々な** explicit substitution **の研究に繋がる**
  Hardin. *From categorical combinators to λ σ-calculi, a quest for confluence*. Research Report, RR-1777, INRIA. 1992.

項書換え系の例: Combinatory Logic

Combinatory Logic

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

SN **ではないが、「左線形で重なりがない」ので合流性を持つ**

Combinatory Logic + equality

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

$$(D \cdot x) \cdot x \longrightarrow E$$

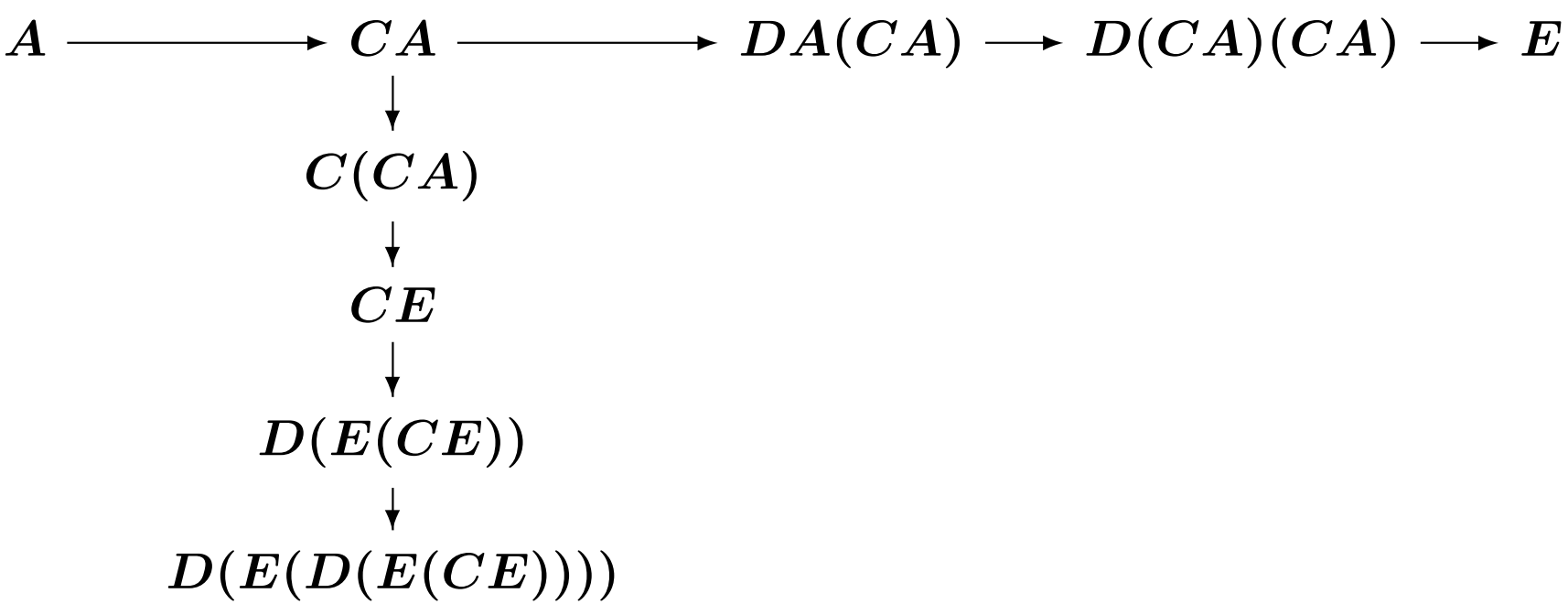
は合流性を持たない！ [Klop'80]

合流性がない例 [Klop'80]

先の TRS の元で

$$\begin{cases} (D \cdot x) \cdot x & \to E \\ (C \cdot x) & \to (D \cdot x) \cdot (C \cdot x) \\ A & \to CA \end{cases}$$

が定義できる



合流性

Combinatory Logic + applicative equality

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

$$(D \cdot x) \cdot x \longrightarrow E$$

合流性を持たない。しかし…

Combinatory Logic + equality

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

$$D(x, x) \longrightarrow E$$

は合流性を持つ

合流性のモジュラ性 [外山'87]

R_1 と R_2 はどちらも合流性を持つ　$\Leftrightarrow$　$R_1 \oplus R_2$ は合流性を持つ

Combinatory Logic + equality

$$((S \cdot x) \cdot y) \cdot z \to (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \to x$$

$$D(x, x) \to E$$

$R_1 \oplus R_2$ は disjoint union (**関数記号を共有していない**)

合流性 (CR) の判定方法

TRS $\mathcal{R}$ の形の判定基準の探求

(i) $\mathcal{R}$ が停止性 (SN) 持ち、すべての危険対が合同 [Knuth,Bendix'70]

(ii) $\mathcal{R}$ が停止性 (SN) 持ち、ルール間に重なりがない

(iii) (Orthogonality) 左線形で重なりがない [Rosen'73]

(iv) 左線形で paralell closed [Huet'80]
…様々な改良 [外山'81,88][van Oostrom'95] や手法多数

(v) (Persistence) $\mathcal{R}$ に型を付けたものが CR なら、元の $\mathcal{R}$ は CR [青戸, 外山'97]

注: (i)(ii) は Newmann の補題 (SN + 弱合流性 (WCR) $\Rightarrow$ CR) を使っている

▶ 合流性の自動判定ツール

`http://cl-informatik.uibk.ac.at/users/ami/15isr/tools.php`

項書換え系の例: Combinatory Logic

Combinatory Logic

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

SN **ではないが、「左線形で重なりがない」ので合流性を持つ**

項書換え系の例: Combinatory Logic

Combinatory Logic + equality

$$((S \cdot x) \cdot y) \cdot z \longrightarrow (x \cdot z)(y \cdot z)$$

$$(K \cdot x) \cdot y \longrightarrow x$$

$$(D \cdot x) \cdot x \longrightarrow E$$

合流性を持たない [Klop'80]

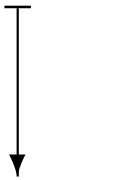
- CR **のモジュラ性** [**外山**'87] **には反さない**: $R_1 \oplus R_2$ **とできない**
- **高階書換え系の場合は、この種のモジュラリティはほとんど成り立たない** [Oostrom'05] $\Rightarrow$ **書換え手法は、プログラミング言語へ応用できない？**
- **例外**: **「堅実」な二階書換え系 と**
  二階再帰プログラム図式 (Recursive Program Schema) **の** disjoint union **は停止性がモジュラー** [**浜名** PPDP'07]
  圏論モデルによるラベル手法

TRS のモデル

基本的なアイデア

TRS $\mathcal{R}$ **のモデル**

$s \to_{\mathcal{R}} t$ **項上の書換え**



$[\![s]\!] >_A [\![t]\!]$ **モデル** $(A, >_A)$

TRS のモデルの概観

モデルは停止性を示すのに役立つ

定理（解釈による停止性証明法）

$\exists$ **モデル** $(A, >_A)$ & $>_A$:**整礎** $\Rightarrow$ $\to_{\mathcal{R}}$ **は停止性を持つ**

[**証明**] **対偶を示す**:

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \quad \text{無限の書換え}$$

$$\downarrow$$

$$[\![t_1]\!] >_A [\![t_2]\!] >_A \cdots \quad \text{整礎でない}$$

典型的な例: $(\mathbb{N}, >)$, $[\![-]\!]$ **をある「重み」を与える写像と取る**

$$f(f(a)) \to_{\mathcal{R}} f(a) \to_{\mathcal{R}} a \qquad \mathcal{R} = \{f(x) \to x\}$$

$$\downarrow [\![-]\!] \quad \text{（項の大きさ）}$$

$$3 > 2 > 1$$

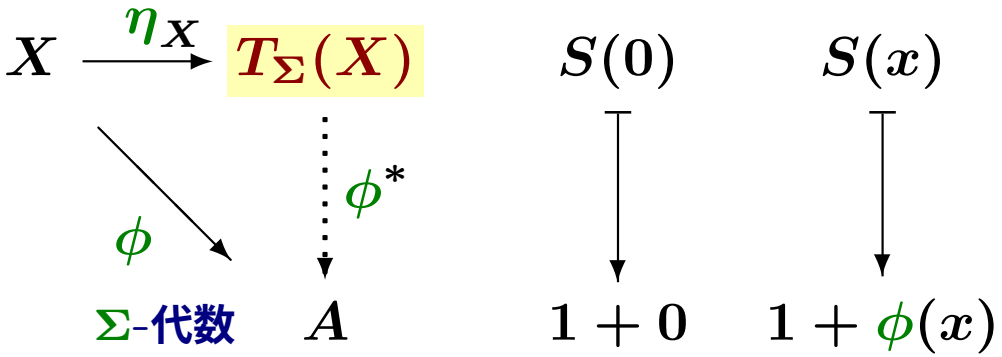
TRS のモデル

TRS のモデル ＝ 順序付き Σ 代数 $(A, >)$

例:
$$0 + y \to S(0)$$
$$S(x) + y \to S(x + y)$$

X **変数の集合**

$T_\Sigma(X)$ **項代数 (項の集合) は 自由 Σ-代数** ϕ^* **準同型写像** (homomorphism)



唯一の準同型写像 ＝ 解釈の写像

反例

A: WCR $\not\Rightarrow$ A:CR

例: **以下の** WCR **な抽象書換え系**