Call-by-need computations in orthogonal TRSs

Irène Durand

Université Bordeaux 1

Habilitation à diriger les recherches

01/07/2005

When		Where	Programming
83-84	DEA	LSI, UPS Toulouse	Functional
84-86	PHD	LSI, UPS Toulouse	Logic
86-88	Post-Doc	PRISM, Univ Maryland	
89-91	MdC	LaBRI, Univ Bordeaux 1	Equational
91-92	Vacataire	Warwick Univ	
92-04	MdC	LaBRI, Univ Bordeaux 1	
04-05	CRCT	FMI, Univ Stuttgart	

1. Theorical framework
   - Term rewriting systems
   - Rewriting strategies
   - Neededness
   - Strong Sequentiality
2. Theorical contribution
   - Call-By-Need classes
   - Complexity
   - Modularity
3. Practical contribution
   - Autowrite
4. Other works
   - Computation to root-stable forms
   - Below strong sequentiality
5. Conclusion
6. Perspectives

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \\ \times(0, x) & \to & 0 \\ \times(\mathsf{s}(x), y) & \to & +(\times(x, y), y) \end{array} \right.$

rewriting

$\mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0))))$ ground term

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

$$\text{rewrite rules } \mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \rightarrow & x \\ +(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(+(x, y)) \\ \times(0, x) & \rightarrow & 0 \\ \times(\mathsf{s}(x), y) & \rightarrow & +(\times(x, y), y) \end{array} \right.$$

rewriting

$\mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0))))$ reducible term

redex

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

$$\text{rewrite rules } \mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \rightarrow & x \\ +(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(+(x, y)) \\ \times(0, x) & \rightarrow & 0 \\ \times(\mathsf{s}(x), y) & \rightarrow & +(\times(x, y), y) \end{array} \right.$$

rewriting

$$\mathsf{s}(\underbrace{\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))}_{\text{redex}}) \rightarrow \mathsf{s}(\underbrace{+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0)))}_{\text{contractum}})$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \rightarrow & x \\ +(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(+(x, y)) \\ \times(0, x) & \rightarrow & 0 \\ \times(\mathsf{s}(x), y) & \rightarrow & +(\times(x, y), y) \end{array} \right.$

rewriting

$$\mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))) \rightarrow \mathsf{s}(+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0))))$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \\ \times(0, x) & \to & 0 \\ \times(\mathsf{s}(x), y) & \to & +(\times(x, y), y) \end{array} \right.$

rewriting

$$\mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))) \to \mathsf{s}(+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0))))$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ $\quad 0$ constant $\quad \mathsf{s}$ unary $\quad + \ \times$ binary

variables $x, y, \ldots$ $\qquad$ terms $\mathsf{s}(\mathsf{s}(0)),\quad +(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \begin{cases} +(0, x) & \rightarrow & x \\ +(\mathsf{s}(x), y) & \rightarrow & \mathsf{s}(+(x, y)) \\ \times(0, x) & \rightarrow & 0 \\ \times(\mathsf{s}(x), y) & \rightarrow & +(\times(x, y), y) \end{cases}$

rewriting

$$\begin{array}{rcl} \mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))) & \rightarrow & \mathsf{s}(+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0)))) \\ & \rightarrow & \mathsf{s}(+(0, \mathsf{s}(\mathsf{s}(0)))) \end{array}$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \left\{ \begin{array}{rcl} +(0,x) & \to & x \\ +(\mathsf{s}(x),y) & \to & \mathsf{s}(+(x,y)) \\ \times(0,x) & \to & 0 \\ \times(\mathsf{s}(x),y) & \to & +(\times(x,y),y) \end{array} \right.$

rewriting

$$\begin{array}{rcl} \mathsf{s}(\times(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)))) & \to & \mathsf{s}(+(\times(0,\mathsf{s}(\mathsf{s}(0))),\mathsf{s}(\mathsf{s}(0)))) \\ & \to & \mathsf{s}(+(0,\mathsf{s}(\mathsf{s}(0)))) \end{array}$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

$$\text{rewrite rules } \mathcal{R} = \left\{ \begin{array}{rcl} +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \\ \times(0, x) & \to & 0 \\ \times(\mathsf{s}(x), y) & \to & +(\times(x, y), y) \end{array} \right.$$

rewriting

$$\begin{array}{rcl} \mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))) & \to & \mathsf{s}(+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0)))) \\ & \to & \mathsf{s}(+(0, \mathsf{s}(\mathsf{s}(0)))) \end{array}$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ — 0 constant, s unary, $+$ $\times$ binary

variables $x, y, \ldots$

terms $\mathsf{s}(\mathsf{s}(0)),\quad +(\mathsf{s}(0), y)$

rewrite rules
$$\mathcal{R} = \begin{cases} +(0, x) & \to & x \\ +(\mathsf{s}(x), y) & \to & \mathsf{s}(+(x, y)) \\ \times(0, x) & \to & 0 \\ \times(\mathsf{s}(x), y) & \to & +(\times(x, y), y) \end{cases}$$

rewriting

$$\begin{aligned} \mathsf{s}(\times(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)))) & \to \mathsf{s}(+(\times(0, \mathsf{s}(\mathsf{s}(0))), \mathsf{s}(\mathsf{s}(0)))) \\ & \to \mathsf{s}(+(0, \mathsf{s}(\mathsf{s}(0)))) \\ & \to \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \end{aligned}$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

$$\text{rewrite rules } \mathcal{R} = \left\{ \begin{array}{rcl} +(0,x) & \to & x \\ +(\mathsf{s}(x),y) & \to & \mathsf{s}(+(x,y)) \\ \times(0,x) & \to & 0 \\ \times(\mathsf{s}(x),y) & \to & +(\times(x,y),y) \end{array} \right.$$

rewriting

$$\begin{array}{rcl} \mathsf{s}(\times(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)))) & \to & \mathsf{s}(+(\times(0,\mathsf{s}(\mathsf{s}(0))),\mathsf{s}(\mathsf{s}(0)))) \\ & \to & \mathsf{s}(+(0,\mathsf{s}(\mathsf{s}(0)))) \\ & \to & \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \quad \text{normal form} \end{array}$$

Term Rewriting System (TRS) $\mathcal{R}$

signature $\mathcal{F} = \{0, \mathsf{s}, +, \times\}$ 0 constant s unary $+$ $\times$ binary

variables $x, y, \ldots$ terms $\mathsf{s}(\mathsf{s}(0))$, $+(\mathsf{s}(0), y)$

rewrite rules $\mathcal{R} = \left\{ \begin{array}{rcl} +(0,x) & \rightarrow & x \\ +(\mathsf{s}(x),y) & \rightarrow & \mathsf{s}(+(x,y)) \\ \times(0,x) & \rightarrow & 0 \\ \times(\mathsf{s}(x),y) & \rightarrow & +(\times(x,y),y) \end{array} \right.$

rewriting

$$\begin{array}{rcl} \mathsf{s}(\times(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)))) & \rightarrow & \mathsf{s}(+(\times(0,\mathsf{s}(\mathsf{s}(0))),\mathsf{s}(\mathsf{s}(0)))) \\ & \rightarrow & \mathsf{s}(+(0,\mathsf{s}(\mathsf{s}(0)))) \\ & \rightarrow & \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \quad \text{normal form} \end{array}$$

$$\mathsf{s}(\times(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)))) \rightarrow^* \mathsf{s}(\mathsf{s}(\mathsf{s}(0))) \in \mathrm{NF}(\mathcal{R})$$

Questions in Rewriting

- Is the TRS terminating? (no infinite rewrite sequences)
- Is the TRS confluent? (implies unicity of normal form)

$$\begin{tikzcd}
 & t \arrow[dl, "*"'] \arrow[dr, "*"] & \\
t_1 \arrow[dr, dashed, "*"'] & & t_2 \arrow[dl, dashed, "*"] \\
 & s &
\end{tikzcd}$$

- How to compute normal forms?

Orthogonal Systems

Definition

An orthogonal TRS is left-linear and non-overlapping (lacks critical pairs)

$$\begin{array}{ll} \mathrm{f}(\mathrm{g}(x,\mathrm{a})) \to x & \mathrm{g}(x,x) \to \mathrm{a} \\ \mathrm{g}(\mathrm{a},x) \to \mathrm{b} & \texttt{not left-linear} \\ \texttt{overlapping} & \end{array}$$

$$\begin{array}{ll} \mathrm{f}(\mathrm{g}(x,\mathrm{a})) \to \mathrm{g}(x,x) & \\ \mathrm{g}(\mathrm{a},\mathrm{b}) \to \mathrm{b} & \texttt{orthogonal} \end{array}$$

Lemma

orthogonality $\Rightarrow$ *confluence* $\Rightarrow$ *unicity of normal form*

signature 0, fib constants s unary $:$, nth, f, $+$ binary

rewrite rules

$$\begin{aligned} +(0,y) &\to y & \mathsf{nth}(0, y:z) &\to y \\ +(\mathsf{s}(x),y) &\to \mathsf{s}(+(x,y)) & \mathsf{nth}(\mathsf{s}(x), y:z) &\to \mathsf{nth}(x,z) \\ \mathsf{f}(x,y) &\to x:\mathsf{f}(y,+(x,y)) & \mathsf{fib} &\to \mathsf{f}(\mathsf{s}(0),\mathsf{s}(0)) \end{aligned}$$

rewriting

$$\begin{array}{lcll} \mathsf{nth}(\mathsf{s}(0),\mathsf{fib}) & \to & \mathsf{nth}(\mathsf{s}(0),\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0))) & \to \\ \mathsf{nth}(\mathsf{s}(0),\mathsf{s}(0):\mathsf{f}(\mathsf{s}(0),+(\mathsf{s}(0),\mathsf{s}(0)))) & \to & \mathsf{nth}(0,\mathsf{f}(\mathsf{s}(0),+(\mathsf{s}(0),\mathsf{s}(0)))) & \to \\ \mathsf{nth}(0,\mathsf{f}(\mathsf{s}(0),\mathsf{s}(+(0,\mathsf{s}(0))))) & \to & \mathsf{nth}(0,\mathsf{f}(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)))) & \to \\ \mathsf{nth}(0,\mathsf{s}(0):\mathsf{f}(\mathsf{s}(\mathsf{s}(0)),+(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0))))) & \to & \mathsf{s}(0) & \end{array}$$

$$\mathsf{nth}(\mathsf{s}(0),\mathsf{fib}) \to \mathsf{nth}(\mathsf{s}(0),\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0))) \to \mathsf{s}(0):\mathsf{f}(\mathsf{s}(0),+(\mathsf{s}(0),\mathsf{s}(0))) \to \cdots$$
$$\to^{\omega} \mathsf{nth}(\mathsf{s}(0),\mathsf{s}(0):\mathsf{s}(0):\mathsf{s}^2(0):\mathsf{s}^3(0):\mathsf{s}^5(0):\cdots:\cdots)$$

Definition

- strategy selects redexes
- strategy is normalizing if it computes the normal form for all terms that have one
- strategy is sequential if it selects a single redex

Examples of strategies

- leftmost outermost sequential
- parallel outermost not sequential

Theorem ([O'Donnell 77])

for orthogonal TRSs

- *parallel-outermost strategy is normalizing*
- *leftmost-outermost strategy is not normalizing*

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

$\mathsf{f}(\mathsf{c}, \mathsf{a})$	$\rightarrow$	$\mathsf{f}(\mathsf{c}, \mathsf{a})$	$\rightarrow$	$\cdots$	leftmost-outermost
$\mathsf{f}(\mathsf{c}, \mathsf{a})$	$\rightarrow^*$	$\mathsf{f}(\mathsf{c}, \mathsf{b})$	$\rightarrow$	b	parallel-outermost

The parallel-outermost strategy is normalizing but not optimal because it performs useless contractions

$$\mathcal{R} = \begin{cases} +(0,x) & \rightarrow & x \\ +(\mathsf{s}(x),y) & \rightarrow & \mathsf{s}(+(x,y)) \\ \times(0,x) & \rightarrow & 0 \\ \times(\mathsf{s}(x),y) & \rightarrow & +(\times(x,y),y) \end{cases}$$

$$\times(\underline{\times(0,\mathsf{s}(0))},\underline{+(0,\mathsf{s}(0))}) \quad \rightarrow^* \quad \underline{\times(0,\mathsf{s}(0))} \quad \rightarrow \quad 0$$

redex $+(0,\mathsf{s}(0))$ is not needed

$$\times(\underline{\times(0,\mathsf{s}(0))},+(0,\mathsf{s}(0))) \quad \rightarrow \quad \underline{\times(0,+(0,\mathsf{s}(0)))} \quad \rightarrow \quad 0$$

Definition ([Huet & Lévy 79])

A redex Δ in a term is needed if a descendant of Δ is contracted in every rewrite sequence from this term to normal form

Theorem ([Huet & Lévy 79])

for orthogonal TRSs ($\perp$)

- *every reducible term has a needed redex*
- *needed rewriting gives an optimal normalizing strategy*

Definition

A strategy which contracts only needed redexes is called a Call-By-Need (CBN) strategy

Strong Sequentiality [HL 79]

- Unfortunately: it is undecidable whether a redex is needed
- find decidable approximation of needed redex

Definition

strongly needed redex: contracted in any rewrite sequence to normal form using arbitrary right-hand sides.

- complicated definition
- notion of index, sequentiality of predicate on term prefixes

In orthogonal systems not every reducible term has a strongly-needed redex

Definition

strongly sequential systems: every reducible term has a strongly needed redex

Theorem ([Huet & Lévy 79])

It is decidable whether a redex in a term is strongly needed

Theorem ([Huet & Lévy 79])

It is decidable whether an orthogonal TRS is strongly sequential.

Proof.

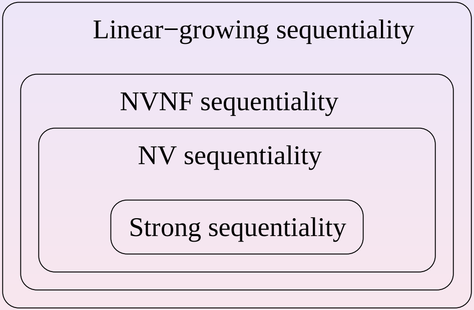
proof is quite difficult (uses the notion of matching dag) □

other proofs

- [Klop & Middeldorp 91] (deltasets)
- [Comon 95,00] (WSkS)

Huet and Lévy's theorem gave rise to several generalizations

Generalization of strong sequentiality

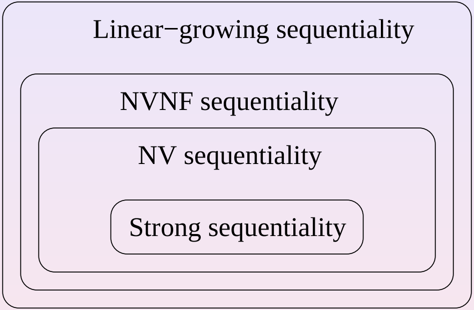


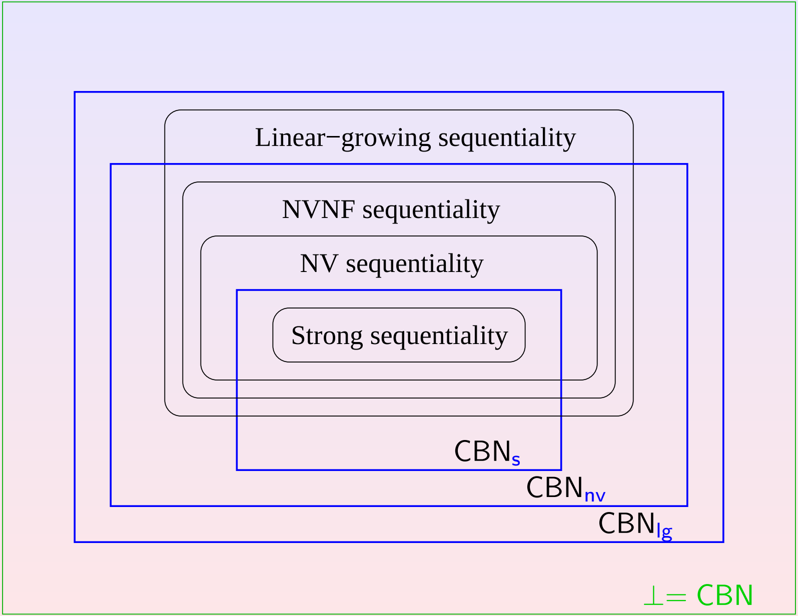
Theorical contribution

One of our main contribution to the domain has been to give a uniform and simplified framework to define classes which admit decidable call-by-need stragegies (joint work with Aart Middeldorp).

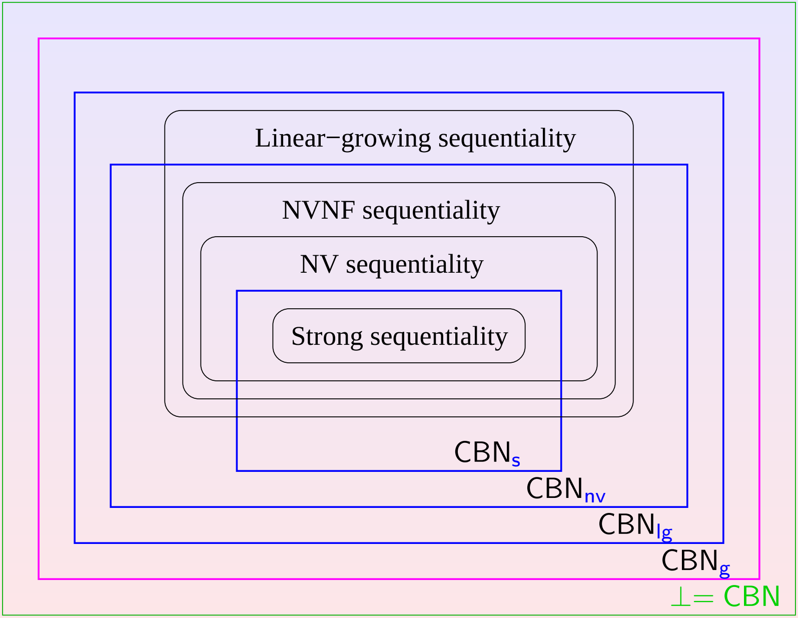
The benefits are

- simpler definitions
- simpler proofs
- bigger classes


Linear−growing sequentiality
NVNF sequentiality
NV sequentiality
Strong sequentiality



Linear−growing sequentiality
NVNF sequentiality
NV sequentiality
Strong sequentiality
CBN_s
CBN_{nv}
CBN_{lg}
$\perp = \mathrm{CBN}$


Linear−growing sequentiality
NVNF sequentiality
NV sequentiality
Strong sequentiality
CBN$_s$
CBN$_{nv}$
CBN$_{lg}$
CBN$_g$
$\perp$= CBN

Notation: $(\xrightarrow[\mathcal{R}]{*})[\mathcal{L}] = \{t \mid t \xrightarrow[\mathcal{R}]{*} s \in \mathcal{L}\}$

Lemma

for an orthogonal TRS $\mathcal{R}$

redex Δ in $C[\Delta]$ needed $\iff C[\bullet] \notin (\xrightarrow[\mathcal{R}]{*})[\mathrm{NF}]$ $\quad$ *$\bullet$-free nf*

Key idea: approximate $\mathcal{R}$ by TRS $\mathcal{R}_\alpha$ such that $C[\bullet] \notin (\xrightarrow[\mathcal{R}]{*})[\mathrm{NF}]$ is decidable.

Definition

TRS $\mathcal{R}_\alpha$ approximates $\mathcal{R}$ if $\xrightarrow[\mathcal{R}]{*} \subseteq \xrightarrow[\mathcal{R}_\alpha]{*}$ and $\mathrm{LHS}_{\mathcal{R}} = \mathrm{LHS}_{\mathcal{R}_\alpha}$

Approximations

- strong (s) [Huet & Lévy 79]
  replace right-hand sides by fresh variables
- non-variable (nv) [Oyamaguchi 93]
  replace variables in right-hand sides by fresh variables
- linear-growing (lg) [Jacquemard 96]
  growing (g) [Nagaya & Toyama 99]
  replace variables in right-hand sides that occur at depth > 1 in left-hand sides by fresh variables

$$\xrightarrow[\mathcal{R}]{*} \subseteq \xrightarrow[\mathcal{R}_{\mathsf{g}}]{*} \subseteq \xrightarrow[\mathcal{R}_{\mathsf{lg}}]{*} \subseteq \xrightarrow[\mathcal{R}_{\mathsf{nv}}]{*} \subseteq \xrightarrow[\mathcal{R}_{\mathsf{s}}]{*}$$

Lemma

if $\mathcal{R}_\alpha$ approximates an orthogonal TRS $\mathcal{R}$ then $\mathcal{R}_\alpha$-needed redexes are $\mathcal{R}$-needed (= needed)

Observation: If every reducible term has an $\mathcal{R}_\alpha$-needed redex, $\mathcal{R}$ admits an optimal and computable sequential call-by-need strategy.

Definition ([Durand-Middeldorp 97])

The class of orthogonal TRSs $\mathcal{R}$ such that every reducible term has an $\mathcal{R}_\alpha$-needed redex is called CBN_α.

$$\mathrm{CBN}_{\mathsf{s}} \subsetneq \mathrm{CBN}_{\mathsf{nv}} \subsetneq \mathrm{CBN}_{\mathsf{lg}} \subsetneq \mathrm{CBN}_{\mathsf{g}} \subsetneq \mathrm{CBN} = \perp$$

Definition

TRS $\mathcal{R}$ is recognizability preserving if for every recognizable set $\mathcal{L}$ $(\xrightarrow[\mathcal{R}]{*})[\mathcal{L}]$ is recognizable.

Theorem ([Jaquemard 96], [Dur-Mid 97], [Nagaya-Toyama 99])

For left-linear $\mathcal{R}$ and $\alpha \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$, $\mathcal{R}_\alpha$ is recognizability preserving.

$\Rightarrow$ $(\xrightarrow[\mathcal{R}_\alpha]{*})[\mathrm{NF}]$ is recognizable

$\Rightarrow$ It is decidable whether $C[\bullet] \notin (\xrightarrow[\mathcal{R}_\alpha]{*})[\mathrm{NF}]$

$\Rightarrow$ It is decidable whether a redex is $\mathcal{R}_\alpha$-needed

$$\xrightarrow[\mathcal{R}]{*} \subseteq \xrightarrow[\mathcal{R}_\mathsf{g}]{*} \subseteq \xrightarrow[\mathcal{R}_\mathsf{lg}]{*} \subseteq \xrightarrow[\mathcal{R}_\mathsf{nv}]{*} \subseteq \xrightarrow[\mathcal{R}_\mathsf{s}]{*}$$

Theorem ([Comon 95])

The set of reducible terms without $\mathcal{R}_{\mathsf{s}}$-needed redex is recognizable.

Theorem ([Durand-Middeldorp 97])

If $\mathcal{R}_\alpha$ is recognizability preserving then the set of reducible terms without $\mathcal{R}_\alpha$-needed redex is recognizable.

Corollary

If is decidable whether a left-linear TRS belongs to CBN_α for $\alpha \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$.

Results

For the CBN_{α} classes, we have obtained

- decidability results
- complexity results
- modularity results

Complexity

- $\boxed{\mathcal{R} \in \mathrm{CBN}_{\mathsf{s}}\,?}$ exponential [Comon 95,00]
- $\boxed{\mathcal{R} \in \mathrm{CBN}_{\mathsf{nv}}\,?}$ [Durand-Middeldorp 98]
  $\boxed{\mathcal{R} \in \mathrm{CBN}_{\mathsf{lg}}\,?}$ double exponential
- $\boxed{\mathcal{R} \in \mathrm{CBN}_{\mathsf{g}}\,?}$ triple exponential [Durand 05]
- $\boxed{\mathcal{R} \in \mathrm{FB}\,?}$ quadratic [Durand 94]
  $\mathrm{FB} \subsetneq \mathrm{CBN}_{\mathsf{s}}$ [Strandh 89]

Modularity

Motivation: Since deciding membership in CBN_α is complex modularity results are important

- Modularity

$$\begin{array}{c}(\mathcal{R}_1,\mathcal{F}_1)\in \mathrm{CBN}_\alpha\\ (\mathcal{R}_2,\mathcal{F}_2)\in \mathrm{CBN}_\alpha\end{array} \stackrel{?}{\Longrightarrow} (\mathcal{R}_1\cup\mathcal{R}_2,\mathcal{F}_1\cup\mathcal{F}_2)\in \mathrm{CBN}_\alpha$$

- First step towards modularity: Signature extension

$$\begin{array}{c}(\mathcal{R},\mathcal{F})\in \mathrm{CBN}_\alpha\\ \mathcal{F}\subsetneq\mathcal{G}\end{array} \stackrel{?}{\Longrightarrow} (\mathcal{R},\mathcal{G})\in \mathrm{CBN}_\alpha$$

- Neither one of these two implications hold in general.

Modularity

Motivation: Since deciding membership in CBN_α is complex modularity results are important

- Modularity

$$\begin{array}{l}(\mathcal{R}_1,\mathcal{F}_1)\in \mathrm{CBN}_\alpha\\(\mathcal{R}_2,\mathcal{F}_2)\in \mathrm{CBN}_\alpha\end{array} \stackrel{?}{\Longrightarrow} (\mathcal{R}_1\cup\mathcal{R}_2,\mathcal{F}_1\cup\mathcal{F}_2)\in \mathrm{CBN}_\alpha$$

- First step towards modularity: Signature extension

$$\begin{array}{c}(\mathcal{R},\mathcal{F})\in \mathrm{CBN}_\alpha\\ \mathcal{F}\subsetneq\mathcal{G}\end{array} \stackrel{?}{\Longrightarrow} (\mathcal{R},\mathcal{G})\in \mathrm{CBN}_\alpha$$

- Neither one of these two implications hold in general.

Signature extension

Results concerning signature extension [Durand-Middeldorp 01]:

$$\begin{tikzcd}
\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l), \forall l \to r \arrow[r, "Y"] \arrow[d, "N"'] & Th\ HL & & \\
\mathrm{ENF}(\mathcal{R}) \neq \varnothing \arrow[r, "Y"] \arrow[d, "N"'] & Th\ DM1 & & \\
\mathrm{WN}(\mathcal{R}_{\alpha}, \mathcal{F}) = \mathrm{WN}(\mathcal{R}_{\alpha}, \mathcal{G}, \mathcal{F}) \arrow[r, "Y"] \arrow[d, "N"'] & \alpha = nv \arrow[r, "Y"] \arrow[d, "N"'] & collapsing \arrow[r, "Y"] \arrow[d, "N"'] & Th\ DM2 \\
CEX\ 1 & CEX\ 2 & CEX\ 3 &
\end{tikzcd}$$

Modularity

Results concerning modularity [Durand-Middeldorp 05]:

$$\begin{tikzcd}
\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l), \forall l \to r \arrow[r, "Y"] \arrow[d, "N"'] & \mathit{Th\ HL} & & \\
\forall i, R_i \ \mathit{noncollapsing} \arrow[r, "Y"] \arrow[dd, "N"'] & \mathrm{ENF} \neq \varnothing \arrow[r, "Y"] \arrow[d, "N"'] & \mathit{Th\ DM3} & \\
 & \mathit{CEX}\ 4 & & \\
\forall i, \mathrm{WN}(\mathcal{R}, \mathcal{G}, \mathcal{F}_i) = \mathrm{WN}(\mathcal{R}_i, \mathcal{F}_i) \arrow[r, "Y"] \arrow[d, "N"'] & \mathit{nv} \arrow[r, "Y"] \arrow[d, "N"'] & \forall i, R_i \ \mathit{collapsing} \arrow[r, "Y"] \arrow[d, "N"'] & \mathit{Th\ DM4} \\
\mathit{CEX}\ 5 & \mathit{CEX}\ 6 & \mathit{CEX}\ 7 &
\end{tikzcd}$$

Also results for constructor sharing combinations

The need for Autowrite

useful properties for obtaining sufficient conditions:

- $\mathrm{NF}(\mathcal{R}, \mathcal{F}) \neq \emptyset$
- $\mathrm{ENF}(\mathcal{R}, \mathcal{F}) \neq \emptyset$
- $\mathrm{WN}(\mathcal{R}_\alpha, \mathcal{G}, \mathcal{F}) = \mathrm{WN}(\mathcal{R}_\alpha, \mathcal{F})$
- is $\mathcal{R}_\alpha$ collapsing? arbitrary?

restriction $\Rightarrow$ counterexample
For each counterexample, we needed to check that $(\mathcal{R}, \mathcal{F}) \in \mathrm{CBN}_\alpha$, $(\mathcal{R}, \mathcal{G}) \notin \mathrm{CBN}_\alpha$ and some of the above conditions.
$\Rightarrow$ many tedious proofs
$\Rightarrow$ Autowrite instead

Autowrite

Main algorithms implemented in Autowrite

Automata

- boolean operations
- emptiness problem
- emptiness of intersection

Term Rewriting Systems

For left-linear $\mathcal{R}$, $\alpha \in \{\mathsf{s}, \mathsf{nv}, \mathsf{lg}, \mathsf{g}\}$ and automaton $\mathcal{A}$,

- Build an automaton $\mathcal{C}_{\mathcal{R}_\alpha, \mathcal{A}}$ such that $L(\mathcal{C}_{\mathcal{R}_\alpha, \mathcal{A}}) = (\xrightarrow[\mathcal{R}_\alpha]{*})[L(\mathcal{A})]$,
- Build an automaton $\mathcal{D}_{\mathcal{R}_\alpha}$ recognizing the set of reducible terms without $\mathcal{R}_\alpha$-needed redexes.

Most of the other operations are combinations of the above.

The outside Autowrite

Autowrite handles a set of specifications
each specification contains

- a signature,
- possibly a set of variables,
- a list of Autowrite objects built upon the signature and variables

The Autowrite objects are:

- Term
- Termset [a set of terms] (named)
- TRS (named)
- Automaton (named)

Example of a specification

```
Ops 0:0 s:1 +:2 *:2
Vars x y
TRS R
; addition
+(0,x) -> x
+(s(x),y) -> s(+(x,y))
; product
*(0,x) -> 0
*(s(x),y) -> +(*(x,y),y)
Automaton EVEN
States odd even
Final States even
```

```
Transitions
0 -> even
s(even) -> odd
s(odd) -> even

Termset RS
0 s(x)

Term *(*(0,s(0)),+(0,s(0)))
Term *(o,+(0,s(0)))
Term *(*(0,s(0)),o)
Term s(s(s(0)))
```

The inside Autowrite

- core of the system written in CLOS (Common Lisp Oriented System)
- the graphical interface is written using McCLIM the free implementation of the CLIM specification
- alltogether about 7500 lines of code

Choices for better performance:

- use sharing rather than copying (especially for automata states)
- compare references not the contents of the objects
- use hash tables
- use memoization

- Autowrite has been used to check many of our examples
- Autowrite has helped us finding counterexamples
- Easy to install (self-contained) and use (no Lisp knowlegde required)
- Autowrite runs faster than other systems implementing tree automata like Timbuk or RX
- Available from my Web page: `http://dept-info.u-bordeaux.fr/~idurand`

Perspectives

- improve performance
- apply to termination
- extend to other types of automata

Computation to root-stable forms

Definition

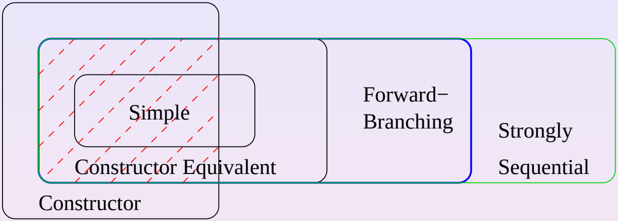
a term is root-stable if it does not rewrite to a redex
(the useful notion for dealing with infinite normal forms)

- [Middeldorp 97] shows that needed redexes are not adequate for computing root-stable forms and proposes the notion root-needed redex
- difficulties for extending our framework to compute root-stable forms
  - undecidability of root-stability
  - root-neededness depends not only the position but also on the redex itself

Results:

- definition of CBN-RS$_{\alpha,\beta}$ classes with decidable call-by-need strategy to compute α-root-stable forms
- decidability and complexity results

Below strong sequentiality



- quadratic decision algorithm for Forward-Branching systems [Durand 94]
- Bruno Salinier PHD's thesis
  - definition of the class of Constructor Equivalent TRSs [Durand-Salinier 93,94]
  - transformation from forward-branching to strong sequential constructor [Salinier-Strandh 96,97]
- implementation of the transformation in Autowrite [Durand 04]

Strong sequentiality

Conjecture ([Middeldorp and Klop 91])

Deciding strong sequentiality is NP-complete
(problem #9 in the list of RTA open problems)

. . .
more than 12 years of research
. . .

Conjecture ([Durand 05])

the problem is both in NP and co-NP

Conclusion

−

- Results on computations to root-stable forms still unpublished
- No result on Klop and Middeldorp's conjecture

+

- Theorical and practical contribution
- Bruno Salinier PHD's thesis
- Journal articles: [IPL 93],[IPL 94], [JSC 94] [ENTCS 05] [IC 05]
- Conferences communications: [PEPM 91], [CADE 97], [FOSSACS 01], [RTA 02], [WRS 04]

Conclusion

—

- Results on computations to root-stable forms still unpublished
- No result on Klop and Middeldorp's conjecture

+

- Theorical and practical contribution
- Bruno Salinier PHD's thesis
- Journal articles: [IPL 93],[IPL 94], [JSC 94] [ENTCS 05] [IC 05]
- Conferences communications: [PEPM 91], [CADE 97], [FOSSACS 01], [RTA 02], [WRS 04]

Perspectives

- generalize systems that preserve recognizability
- apply automata techniques to termination
- computational linguistics
- ?