

Context-sensitive rewriting techniques for programs with strategy annotations

Salvador Lucas

Dep. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

`slucas@dsic.upv.es`

Motivation: programming languages

- **User-defined strategies** in rewriting-based programming languages (e.g., Maude, ELAN, OBJ2, OBJ3, CafeOBJ)
- Often used together with (more) standard features:
 - ◆ Sorts / Types
 - ◆ Conditional equations
 - ◆ AC symbols,
 - ◆ Modules, ...



Motivation: programming languages

*How to analyze the computational properties
of programs written in **such** languages?*

Motivation: programming languages

- **Program Analysis** for such programming languages is an **open** (and **challenging**) research subject!

- Example: **termination of programs:**

A program is terminating if no infinite computation is possible

Motivation: programming languages

■ How to deal with this?

- ◆ As **termination of rewriting**?

A TRS is terminating if there is no infinite rewrite sequence

- ◆ Too **restrictive**! In strategic programming, we do not need/want to consider all possible rewrite sequences...

Motivation: programming languages

■ How to deal with this?

◆ Via **normalizing strategies**?

A rewriting strategy is normalizing if no infinite sequence is issued from terms having a normal form

◆ This does not fit our **notion** of termination! What about terms **without** a normal form?



Motivation: programming languages

*We need to analyze termination of
programs running under strategies!*

Motivation: programming languages

- Termination is only a motivating example.
- Similar things can happen with:
 - ◆ Uniqueness of **computed** canonical forms,
 - ◆ Correctness (e.g., regarding **normalization**),
 - ◆ Completeness,
 - ◆ Efficiency / Optimality,...
- Therefore,...



Motivation: programming languages

*We need to carefully take into account
the role of the strategy language at hand!*

Case study: strategy annotations

■ Strategy annotations:

(basically) lists of integers associated to function symbols which specify the ordering in which the arguments are (eventually) evaluated in function calls

■ Extended Visser's classification [Vis01]:


- ◆ E-strategies (or local strategies) [Eke98],
- ◆ Just-in-time [Pol01],
- ◆ Laziness annotations [FKW00],

- ◆ On-Demand E-strategies [OF00].

Case study: strategy annotations

Example (Local strategies in an OBJ3 program):

```
obj LazyLists is
  sorts Nat LNat .
  op 0      : -> Nat .
  op s      : Nat -> Nat .
  op nil    : -> LNat .
  op cons   : Nat LNat -> LNat [strat (1)] .
  op from   : Nat -> LNat .
  op sel    : Nat LNat -> Nat [strat (1 2 0)] .
  vars N X  : Nat .
  var L     : LNat .
  eq sel(s(N),cons(X,L)) = sel(N,L) .
  eq sel(0,cons(X,L)) = X .
  eq from(N) = cons(N,from(s(N))) .
endo
```



Explicit strategy annotations

Symbols without an explicit annotation are given a default one!

Case study: strategy annotations

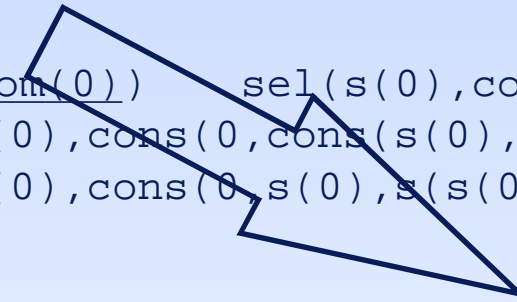
- Informal semantics (for local strategies):
 - ◆ **Positive** integers start the (recursive) evaluation of the corresponding argument of the call
 - ◆ **Zero** indicates that a rule (if any) should be applied to the topmost symbol of the call
- Local strategies are currently used in OBJ3 CafeOBJ, and Maude

Case study: strategy annotations

Example:

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))  
  sel(0, from(s(0)))  
    sel(0, cons(s(0), from(s(s(0)))))  
      s(0)
```

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))  
  sel(s(0), cons(0, cons(s(0), from(s(s(0)))))  
    sel(s(0), cons(0, s(0), s(s(0)), from(s(s(s(0))))))  
      ...
```



Impossible (due to
replacement restrictions)!

Case study: strategy annotations

Example:

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))
```

Terminating!?

```
sel(s(0), from(0))  
sel(s(0), cons(0, cons(s(0), from(s(s(0))))))  
sel(s(0), cons(0, s(0), s(s(0)), from(s(s(s(0)))))))  
...
```

Impossible (due to
replacement restrictions)!

Case study: strategy annotations

Example:

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))
```

Yes, and provable!

```
sel(s(0), from(0))  
sel(s(0), cons(0, cons(s(0), from(s(s(0))))))  
sel(s(0), cons(0, s(0), s(s(0)), from(s(s(s(0)))))))  
...
```

Impossible (due to replacement restrictions)!

Case study: strategy annotations

Example:

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))
```

Key: Syntactic replacement restrictions /
Context-sensitive rewriting

```
sel(s(0), cons(0, cons(s(0), from(s(s(0))))))  
sel(s(0), cons(0, s(0), s(s(0)), from(s(s(s(0))))))  
...
```

Impossible (due to
replacement restrictions)!

Case study: strategy annotations

Example

```
obj LazyLists is
  sorts Nat LNat .
  op 0      : -> Nat .
  op s      : Nat -> Nat .
  op nil    : -> LNat .
  op cons   : Nat LNat -> LNat [strat (1)] .
  op from   : Nat -> LNat .
  op sel    : Nat LNat -> Nat [strat (1 2 0)] .
  vars N X  : Nat .
  var L     : LNat .
  eq sel(s(N),cons(X,L)) = sel(N,L) .
  eq sel(0,cons(X,L)) = X .
  eq from(N) = cons(N,from(s(N))) .
endo
```

Case study: strategy annotations

Example

```
obj LazyLists is
  sorts Nat LNat .
```

Sig {0, s, nil, cons, from, sel}

**Replacement
map**

$\mu(\text{cons})=\{1\}$
 $\mu(\text{sel})=\{1,2\}$

```
vars N X      : Nat .
var L         : LNat .
```

TRS

```
sel(s(N), cons(X, L)) -> sel(N, L)
sel(0, cons(X, L)) -> X
from(N) -> cons(N, from(s(N)))
```

```
endo
```

Case study: strategy annotations

■ Replacement maps:

sets (rather than lists) of positive integers associated to function symbols aimed at specifying which arguments are *reducible* in function calls

■ Context-Sensitive Rewriting (CSR [JFLP98]):

rewriting steps are limited by the replacement restrictions imposed by a replacement map

Summary

■ Context-sensitive rewriting (CSR)

- ◆ Basics of context-sensitive rewriting
- ◆ Head-normalization and (infinitary) normalization
- ◆ Context-sensitive rewriting strategies
- ◆ Confluence and termination
- ◆ Extensions: *Innermost CSR, AC-CSR, modularity,...*

■ CSR and strategy annotations

- ◆ E-strategies
- ◆ Just-in-time
- ◆ Lazy rewriting
- ◆ On-Demand E-strategies



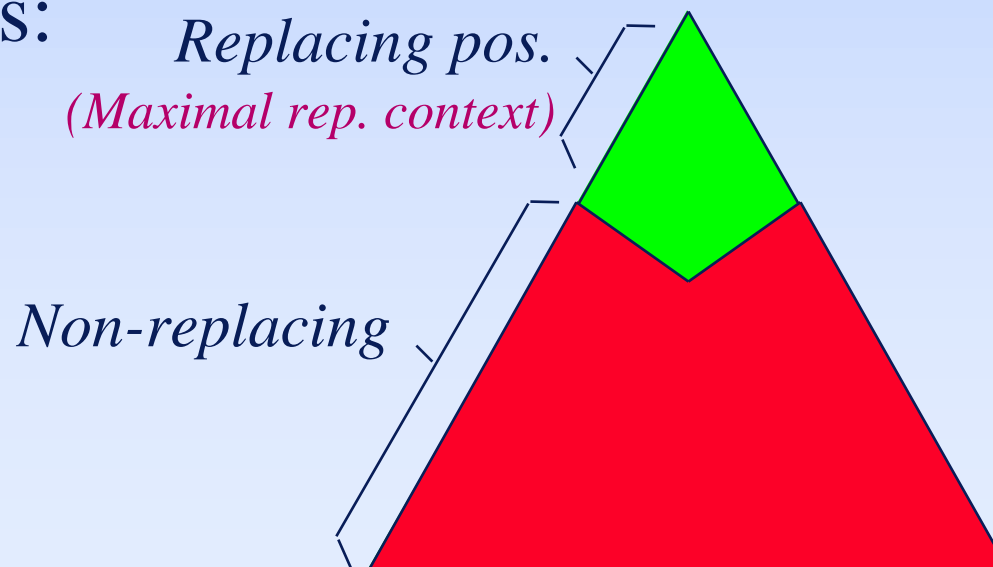
Context-sensitive rewriting

Basics of context-sensitive rewriting

■ Replacement map:

A mapping μ which associates a subset $\mu(f)$ of $\{1, \dots, ar(f)\}$ to each function symbol f

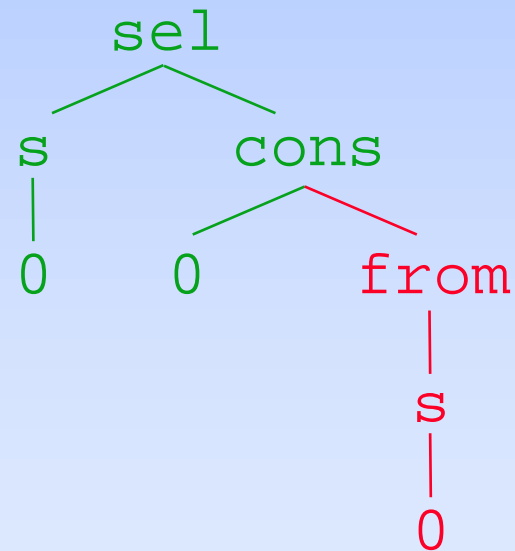
■ The set of positions of a term splits into two disjoint sets:



Basics of context-sensitive rewriting

■ Example: $t = \text{sel}(\text{s}(0), \text{cons}(0, \text{from}(\text{s}(0))))$

| |
|------------------------------|
| $\mu(0) =$ |
| $\mu(\text{s}) = \{1\}$ |
| $\mu(\text{nil}) =$ |
| $\mu(\text{cons}) = \{1\}$ |
| $\mu(\text{from}) = \{1\}$ |
| $\mu(\text{sel}) = \{1, 2\}$ |

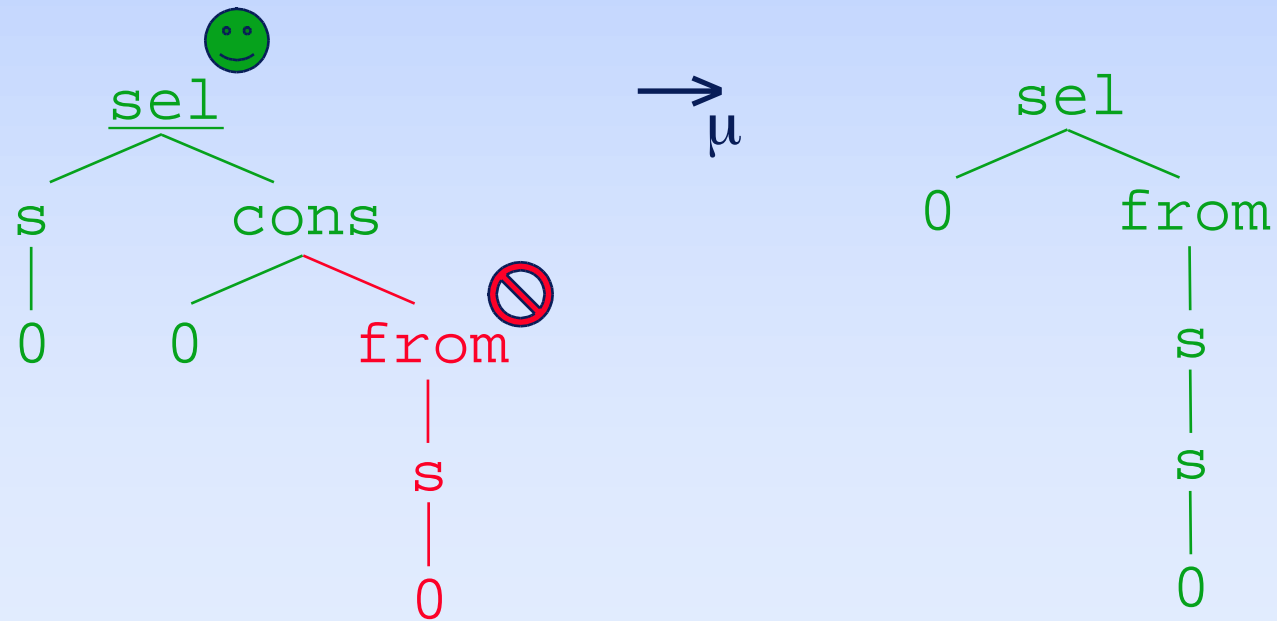


$MRC^u(t) = \text{sel}(\text{s}(0), \text{cons}(0, \quad))$

Basics of context-sensitive rewriting

■ Context-sensitive rewriting:

Given a replacement map μ , *only* rewriting steps at replacing positions are allowed (written $t \rightarrow_{\mu} s$)





(Head- and infinitary) normalization

(Head- and infinitary) normalization

- Consider the following TRS R :

$$\boxed{\begin{array}{l} \mu(f) = \{1\} \\ \mu(g) = \{1\} \end{array}} \quad \left\{ \begin{array}{l} f(x, y) \rightarrow g(x, y) \\ g(b, b) \rightarrow b \\ a \rightarrow b \end{array} \right.$$

- For the term $f(b, a)$, we have derivations

$$\underline{f(b, a)} \rightarrow_{\mu} g(b, a)$$

$$f(b, \underline{a}) \rightarrow \underline{f(b, b)} \rightarrow \underline{g(b, b)} \rightarrow b$$

- CSR is **not able** to obtain a head-normal form (i.e., a term which does **not** reduce to a **redex**)!

(Head- and infinitary) normalization

- In order to compute ‘interesting’ information:
 - ◆ (constructor) head-normal forms,
 - ◆ (infinite) values,
 - ◆ (infinite) normal forms.

we need to make μ (somehow) compatible with unrestricted computations:

$lhs = g(b, b)$

$term = g(b, a)$

Matching impossible!

We should let $\mu(g) = \{1, 2\}$!

(Head- and infinitary) normalization

- The **canonical replacement map** μ_R^{can} for a TRS R is [JFLP98]:

the most restrictive replacement map ensuring that the non-variable subterms of the left-hand sides of the rules of R are replacing

$$\boxed{\begin{array}{l} \mu_R^{can}(f) = \\ \mu_R^{can}(g) = \{1,2\} \end{array}}$$

$$\left\{ \begin{array}{l} f(x, y) \rightarrow g(x, y) \\ g(b, b) \rightarrow b \\ a \rightarrow b \end{array} \right.$$

(Head- and infinitary) normalization

- Let CM_R be the set of replacement maps which are less restrictive than or equally restrictive to μ_R^{can}
- **Theorem** [JFLP98]: Let R be a **left-linear** TRS and μ in CM_R . Every μ -normal form is a **head-normal form**

(Head- and infinitary) normalization

- Left-linearity matters! [JFLP98,IC02]:

$$\boxed{\begin{array}{l} \mu_R^{can}(f) = \{1,2\} \\ \mu_R^{can}(c) = \end{array}} \quad \left\{ \begin{array}{l} f(x, x) \rightarrow c(x) \\ a \rightarrow b \end{array} \right.$$

Term $f(c(a), c(b))$ is a μ -normal form
but it is **not** a head-normal form:

$$f(c(\underline{a}), c(b)) \rightarrow \underline{f(c(b), c(b))} \rightarrow c(c(b))$$

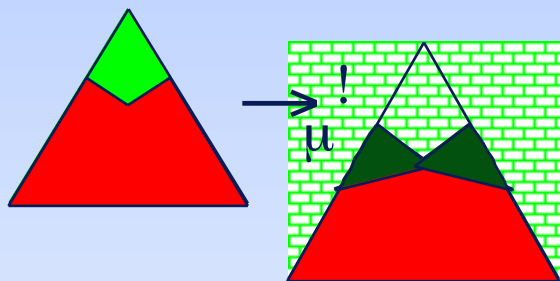
(Head- and infinitary) normalization

- **Theorem** [IC02]: Let R be a **left-linear** TRS and μ in CM_R . If s is a normal form of t , then there is a μ -normal form s' of t which **normalizes** into s

$$t \xrightarrow[\mu]{!} s' \rightarrow! s$$

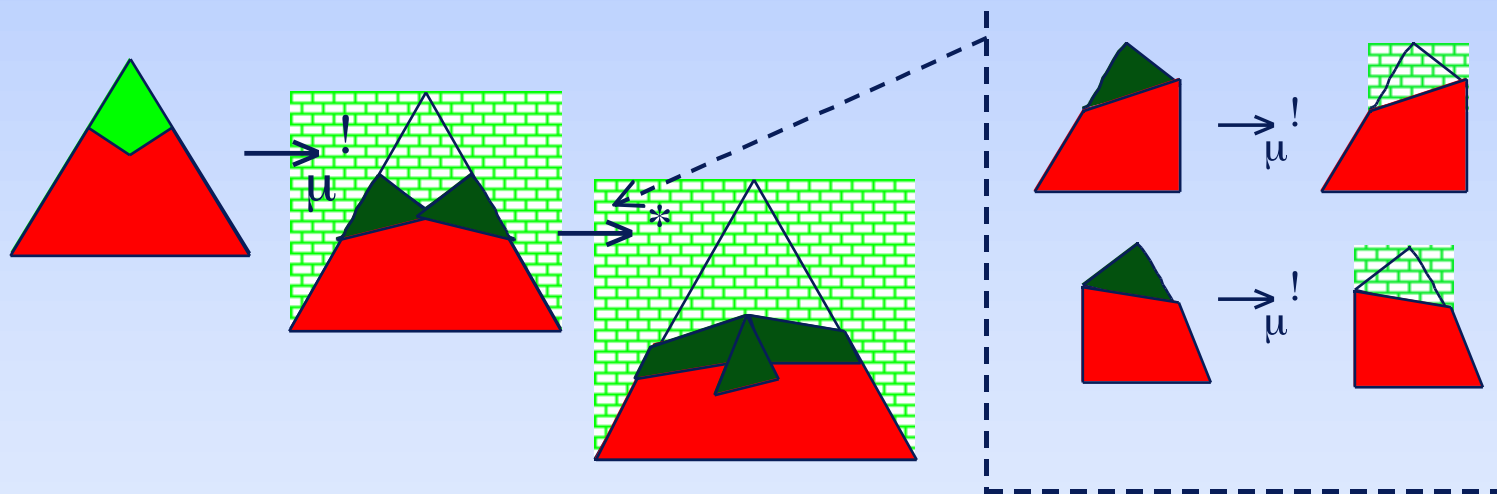
(Head- and infinitary) normalization

- Normalization via μ -normalization [IC02]
(left-linear (confluent) TRSs and μ in CM_R):



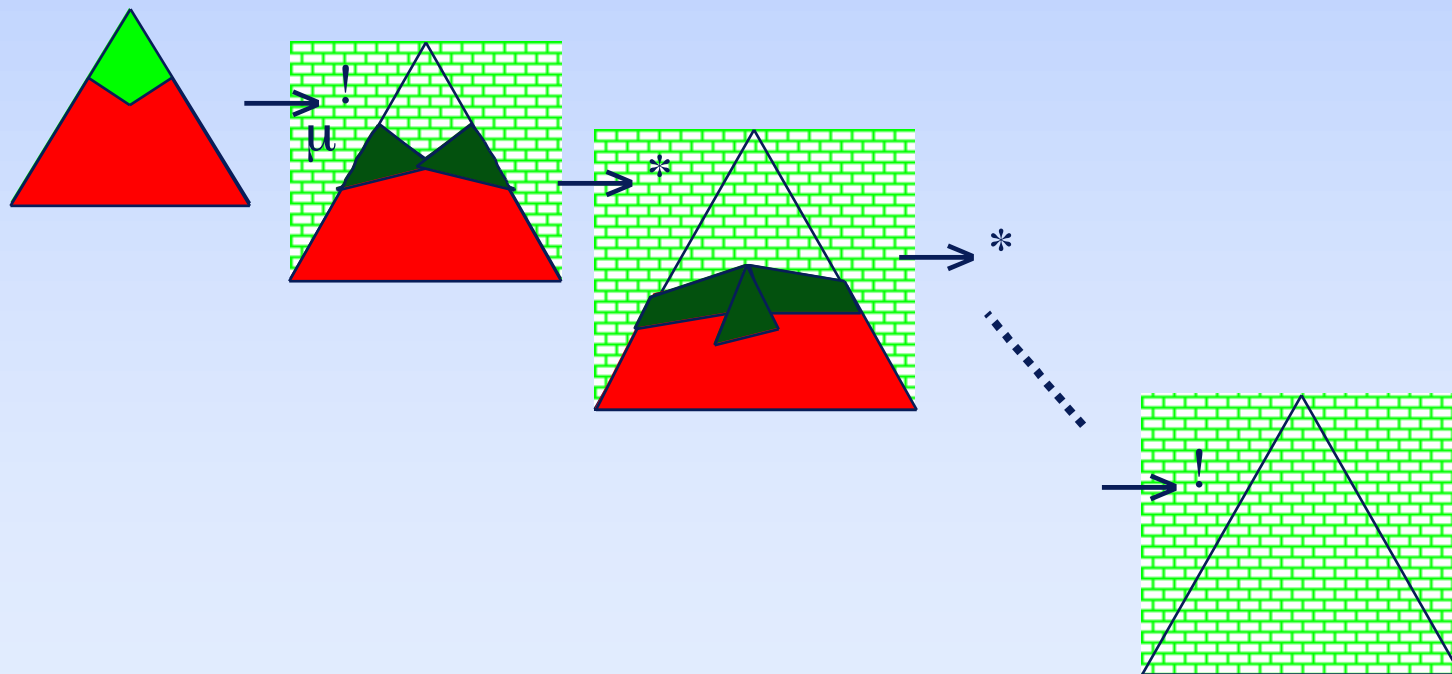
(Head- and infinitary) normalization

- Normalization via μ -normalization [IC02]
(left-linear (confluent) TRSs and μ in CM_R):



(Head- and infinitary) normalization

- Normalization via μ -normalization [IC02]
(left-linear (confluent) TRSs and μ in CM_R):



(Head- and infinitary) normalization

- Normalization via μ -normalization [IC02, RTA02] (**left-linear** (confluent) TRSs and μ in CM_R):

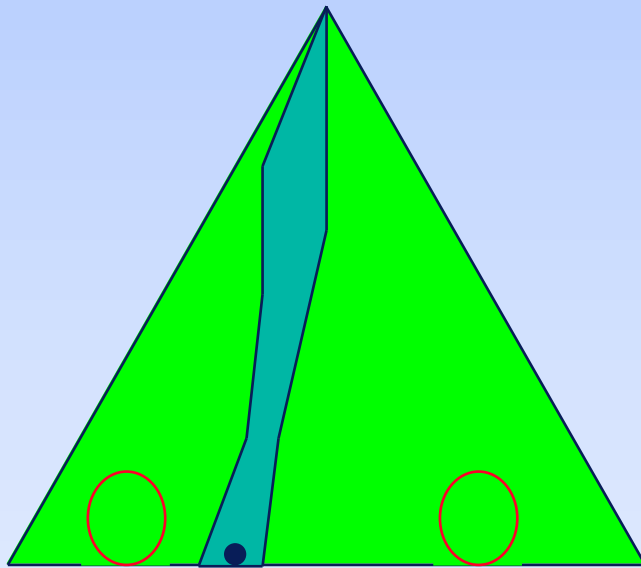
```
Procedure  $norm_{\mu}(T)$   
   $T := \mu\text{-norm}(T)$   
  for each  $t$  in  $T$   
    let  $t = C[t_1, \dots, t_n]$  where  $C[ ] = MRC^{\mu}(t)$   
    for  $i := 1, \dots, n$  do  $S_i := norm_{\mu}(\{t_i\})$   
     $T_t := C[S_1, \dots, S_n]$   
  return  $\bigcup_{t \text{ in } T} T_t$   
end procedure  $norm_{\mu}$ 
```



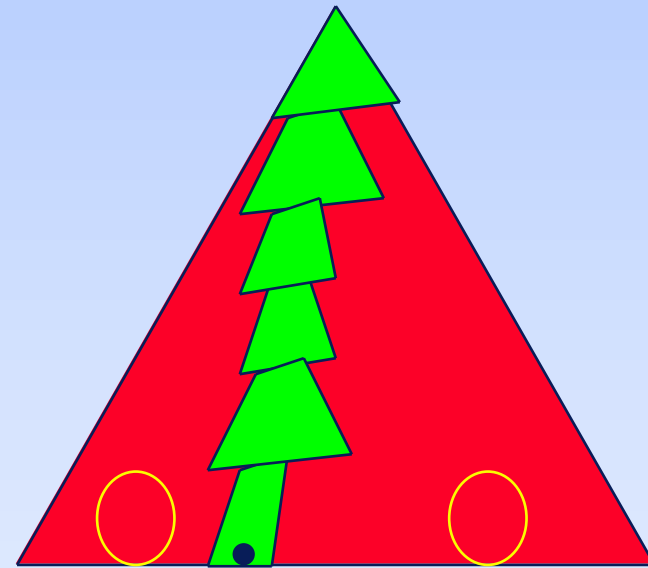
Context-Sensitive Rewriting Strategies

Context-sensitive rewriting strategies

■ Strategies vs computational restrictions



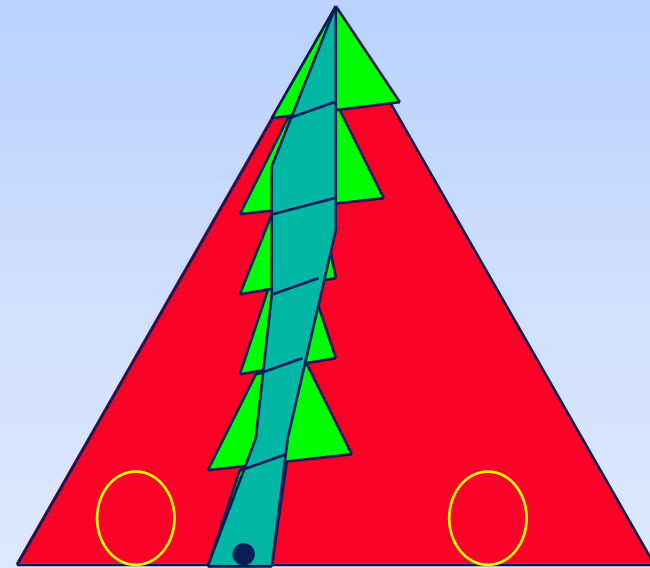
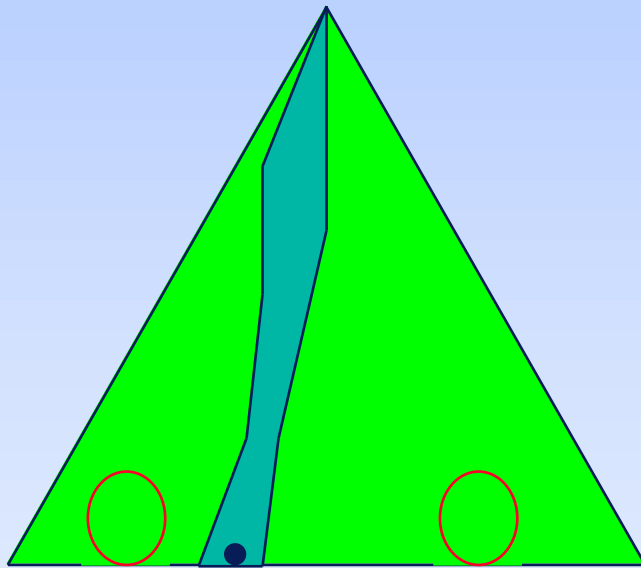
Strategy: build the **straight** way towards the normal form



Comp. rest.: follow a **layered** 'via' towards the normal form

Context-sensitive rewriting strategies

■ Strategies vs computational restrictions



Restricted strategies are still possible and useful!

Context-sensitive rewriting strategies

- Strategies for computational restrictions: In [IC02]:

Salvador Lucas

Context-Sensitive Rewriting Strategies

Information and Computation, to appear.

we discuss in more detail the definition and use of (optimal, μ -normalizing) cs-strategies in programming



Confluence of CSR

Confluence of CSR

**A TRS is μ -confluent
if \rightarrow_{μ} is confluent**

Confluence of CSR

- Sharp differences with unrestricted rewriting
- **Example [JFLP98]:**

Orthogonal!

$$\begin{cases} \mu(f) = \{1\} \\ \mu(g) = \{1\} \\ \mu(h) = \{1\} \end{cases}$$

$$\begin{cases} f(x) \rightarrow g(x, x) \\ h(0) \rightarrow 0 \end{cases}$$

Different μ -normal forms!

$$\underline{f(h(0))} \xrightarrow{\mu} g(\underline{h(0)}, h(0)) \xrightarrow{\mu} g(0, h(0))$$

$$f(\underline{h(0)}) \xrightarrow{\mu} \underline{f(0)} \xrightarrow{\mu} g(0, 0)$$

$$\begin{matrix} g(0, h(0)) \\ g(0, 0) \end{matrix}$$

Confluence of CSR

■ Fortunately, standard results:

- ◆ *Terminating TRSs having joinable critical pairs are confluent [Hue80]*
- ◆ *Orthogonal TRSs are confluent [HL91]*

generalize to CSR after imposing some **additional conditions** and considering the **appropriate notions** (e.g., μ -termination, etc.)

Confluence of CSR

- **Theorem [JFLP98]:** A TRS with **left-homogeneous μ -replacing variables (LHRV)** is **locally μ -confluent** if and only if every critical pair is **μ -joinable**

$$\begin{array}{|l} \mu(f)=\{1\} \\ \mu(g)=\{1\} \\ \mu(h)=\{1\} \end{array} \left\{ \begin{array}{l} f(\mathbf{x}) \rightarrow g(\mathbf{x}, \mathbf{x}) \\ h(0) \rightarrow 0 \end{array} \right. \quad \text{No LHRV}$$

$$\begin{array}{|l} \mu(f)= \\ \mu(g)=\{1\} \\ \mu(h)=\{1\} \end{array} \left\{ \begin{array}{l} f(\mathbf{x}) \rightarrow g(\mathbf{x}, \mathbf{x}) \\ h(0) \rightarrow 0 \end{array} \right. \quad \text{LHRV!}$$

Confluence of CSR

- **Theorem [JFLP98]:** A μ -terminating TRS with left-homogeneous μ -replacing variables is μ -confluent if and only if every critical pair is μ -joinable
- **Example:** A μ -confluent TRS:

$$\boxed{\begin{array}{l} \mu(f)= \\ \mu(g)=\{1\} \\ \mu(h)=\{1\} \end{array}} \quad \left\{ \begin{array}{l} f(x) \rightarrow g(x, x) \\ h(0) \rightarrow 0 \end{array} \right.$$

Confluence of CSR

- **Theorem [JFLP98]:** Orthogonal TRSs with left-homogeneous μ -replacing variables are μ -confluent
- LHRV is **not** necessary for μ -confluence!
- **Example [JFLP98]:**

$$\begin{array}{l} \mu(f)=\{1\} \\ \mu(g)=\{1\} \\ \mu(h)=\{1\} \\ \mu(s)=\{1\} \end{array}$$

$$\left\{ \begin{array}{ll} f(\mathbf{x}) \rightarrow g(\mathbf{x}, \mathbf{x}) & h(0) \rightarrow 0 \\ g(0, \mathbf{x}) \rightarrow 0 & h(s(\mathbf{x})) \rightarrow 0 \\ g(s(\mathbf{x}), \mathbf{y}) \rightarrow s(\mathbf{x}) & \end{array} \right.$$

This is not LHRV but still μ -confluent!

Confluence of CSR

- Testing μ -confluence is not so easy (because of LHRV). Fortunately, **ordinary confluence** also entails interesting properties!
- **Theorem [IC02]:** Let R be a **left-linear, confluent TRS** and μ in CM_R . The μ -normal forms of a term t share the **same maximal replacing context**



Termination of CSR

Termination of CSR

**A TRS is μ -terminating
if \rightarrow_{μ} is terminating**

Termination of CSR

- **Definition** [DKP91]: A TRS is **top-terminating** if no infinitary reduction sequence performs infinitely many rewrites at topmost position
- Top-terminating TRSs only admit infinitary sequences that (if **fairness** is additionally ensured) **ultimately obtain** a (possibly infinite) **normal form**

Termination of CSR

- **Theorem [RTA02]:** Let R be a **left-linear** TRS and μ in CM_R . If R is μ -terminating, then R is **top-terminating**.

SEQUENCES

| | | Finite | Infinite |
|------|-------------|-------------|------------------------|
| TRSs | Normalizing | Normalizing | Inf. normalizing |
| | Terminating | Terminating | Top-terminating |

Termination of CSR

- Proof methods:
 - ◆ μ -reduction orderings
 - ◆ Transformations
 - ◆ Direct methods

Termination of CSR

- A μ -reduction ordering is a well-founded, stable and μ -monotonic ordering $>$

(μ -monotonic: for all symbols f and μ -replacing arguments i ,
 $t > s$ implies $f(t_1, \dots, t_{i-1}, t, \dots, t_k) > f(t_1, \dots, t_{i-1}, s, \dots, t_k)$)

- **Theorem** [Zan97]: A TRS R is μ -terminating if and only if there is a μ -reduction ordering $>$ such that

$$l > r \text{ for all rule } l \rightarrow r \text{ in } R$$

Termination of CSR

- **Transformations**: The μ -termination of a TRS R is demonstrated by proving termination of a TRS R^μ_Θ for a given transformation Θ :
 - ◆ Lucas [ICALP96]
 - ◆ Zantema [RTA97]
 - ◆ Ferreira and Ribeiro [RTA99]
 - ◆ Giesl and Middeldorp [RTA99] (gave a **complete** transformation)

Termination of CSR

- Lucas' transformation [ICALP96]: **remove** all non-replacing subterms from the rules of the TRS R : For instance:

$$R \left\{ \begin{array}{l} \text{first}(0, x) \rightarrow \text{nil} \\ \text{first}(s(x), \text{cons}(y, z)) \rightarrow \text{cons}(y, \text{first}(x, z)) \\ \text{from}(x) \rightarrow \text{cons}(x, \text{from}(s(x))) \end{array} \right.$$

$$\mu(\text{cons}) = \mu(\text{from}) = \mu(s) = \{1\}, \mu(\text{first}) = \{1, 2\}$$

$$R_L^\mu \left\{ \begin{array}{l} \text{first}(0, x) \rightarrow \text{nil} \\ \text{first}(s(x), \text{cons}(y)) \rightarrow \text{cons}(y) \\ \text{from}(x) \rightarrow \text{cons}(x) \end{array} \right.$$

Note that R_L^μ is terminating (e.g., use *RPO*)

Termination of CSR

- Zantema's transformation [RTA97]: the non-replacing subterms of the rules are **marked**:

$$R \left\{ \begin{array}{l} \text{sel}(s(N), \text{cons}(X, L)) \rightarrow \text{sel}(N, L) \\ \text{sel}(0, \text{cons}(X, L)) \rightarrow X \\ \text{from}(N) \rightarrow \text{cons}(N, \text{from}(s(N))) \end{array} \right.$$

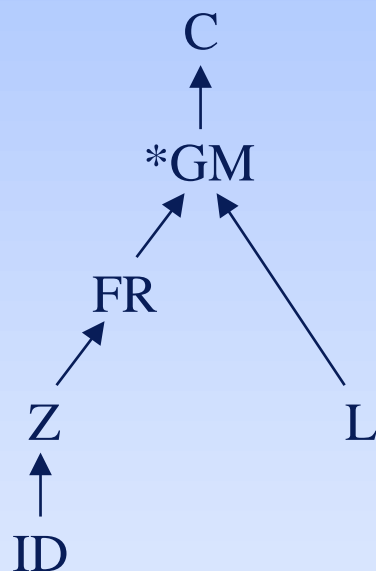
$\mu(\text{cons}) = \mu(\text{from}) = \mu(s) = \{1\}, \mu(\text{sel}) = \{1, 2\}$

$$R_Z^\mu \left\{ \begin{array}{l} \text{sel}(s(N), \text{cons}(X, L)) \rightarrow \text{sel}(N, \text{activate}(L)) \\ \text{sel}(0, \text{cons}(X, L)) \rightarrow X \\ \text{from}(N) \rightarrow \text{cons}(N, \text{from}'(s(N))) \\ \text{from}(N) \rightarrow \text{from}'(N) \\ \text{activate}(\text{from}'(N)) \rightarrow \text{from}(N) \\ \text{activate}(X) \rightarrow X \end{array} \right.$$

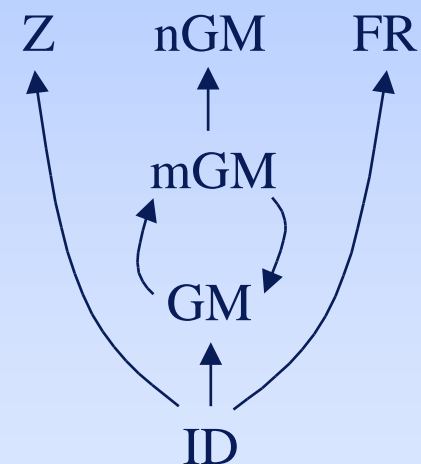
Note that R_Z^μ is terminating (use *RPO* again)

Termination of CSR

■ Comparison:



Termination [GM99]



Simple term. [RTA02]

The transformations are available for use within MU-TERM 1.0:

<http://www.dsic.upv.es/users/elp/slucas/muterm>

Termination of CSR

■ Direct methods:

- ◆ CSRPO (Borralleras, Lucas & Rubio [CADE02])
- ◆ Polynomials (Gramlich & Lucas [RULE02])
- ◆ CSKBO (Borralleras [PhD02])

Termination of CSR

- **Polynomials**: The μ -termination of the following TRS R [GM99]:

$$\boxed{\mu(f)=\{3\}} \quad \left\{ \begin{array}{l} f(a, b, x) \rightarrow f(x, x, x) \\ c \rightarrow a \\ c \rightarrow b \end{array} \right.$$

cannot be proved by using transformations (or CSRPO)!

- We consider polynomial interpretations using **integer** coefficients and returning nonnegative values (and requiring **μ -monotonicity** only)

Termination of CSR

- In the previous example, we let:

$$\begin{aligned} f(x,y,z) &= 1 + (x - y)^2 + z & a &= 2 \\ c &= 3 & b &= 1 \end{aligned}$$

- Then, we have

$$[f(a, b, x)] = 2 + x > 1 + x = [f(x, x, x)]$$

$$[c] = 3 > 2 = [a]$$

$$[c] = 3 > 1 = [b]$$

Therefore, R is μ -terminating.

Termination of CSR

- As for the 'unrestricted' notions (namely: RPO, KBO, etc.), such direct methods are particular cases of the more general notion of **simple termination of CSR**

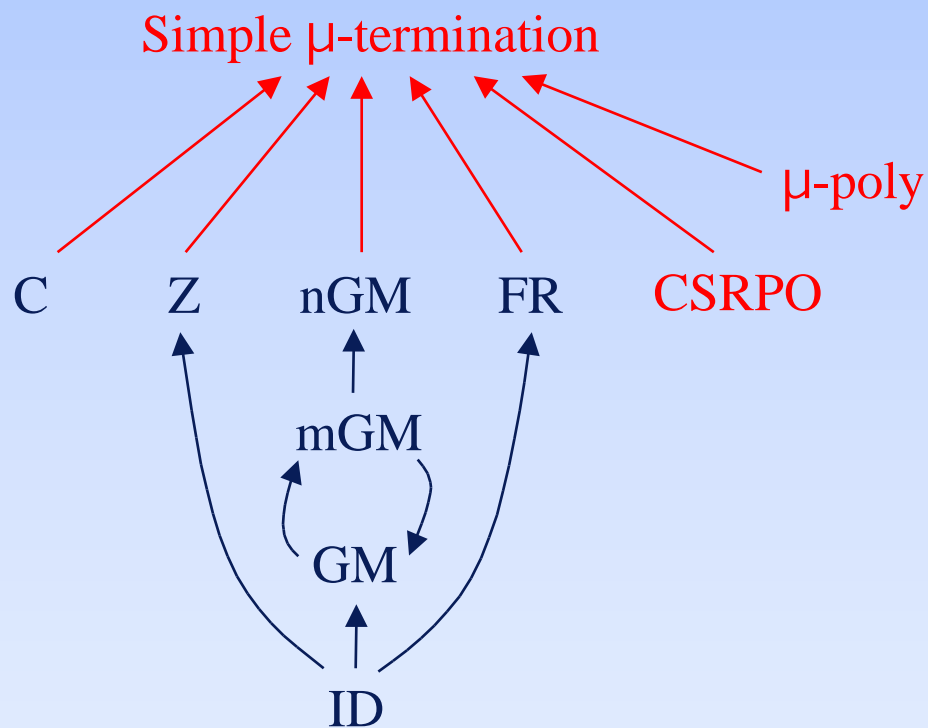
- **Definition [RULE02]:** A TRS R is simply μ -terminating if R plus the rules

$$\{f(x_1, \dots, x_k) \rightarrow x_i \mid i \text{ belongs to } \mu(f)\}$$

is μ -terminating

Termination of CSR

■ Comparison:



Simple termination [RTA02,RULE02]



Extensions of CSR

Extensions of CSR

- The basic theory of CSR has been extended / deepen in several directions:
 - ◆ **Innermost** (termination of) CSR [LPAR01]; Giesl & Middeldorp [DLT02]
 - ◆ **AC**-CSR, Ferreira & Ribeiro [RTA99]; Giesl & Middeldorp [TR02]
 - ◆ **Modularity** issues, Gramlich & Lucas [PPDP02,RULE02]

Extensions of CSR: *innermost* CSR

- **Innermost CSR:** *only redexes at innermost replacing positions are contracted*
- Termination of (ground) innermost CSR is interesting:
 - ◆ for analyzing termination of CSR (as proofs of innermost termination can be easier)
 - ◆ for analyzing termination of **rewriting under strategy annotations** (see below)

Extensions of CSR: *innermost* CSR

- **Theorem** [PPDP02]: A **locally μ -confluent overlay TRS** R satisfying **LHRV** is μ -terminating if and only if R is innermost μ -terminating
- **Theorem** [GM02b]: An **orthogonal CS** R is μ -terminating if and only if R is innermost μ -terminating

Extensions of CSR: *innermost* CSR

- Unfortunately, most of the aforementioned proof techniques (for termination of CSR) are **not** valid for proving innermost termination of CSR [LPAR01,GM02b]
- Fortunately, however, new (even complete) proof techniques have been developed by Giesl and Middeldorp [DLT02]

Extensions of CSR: AC-CSR

- **Associative** and/or **Commutative** operators are often used in rewriting-based engines and programming languages
- The definition of **AC-CSR** is analogous to standard (unrestricted) AC rewriting: just use \rightarrow_{μ} instead of \rightarrow when (AC-equivalent) terms are rewritten

Extensions of CSR: AC-CSR

- Ferreira and Ribeiro [RTA99] introduced the first transformation for proving termination of AC-CSR.
- Giesl and Middeldorp [TR02] have recently introduced more powerful (even complete) transformations

Extensions of CSR: *Modularity*

- Programmers usually organize the programs into components or **modules**
- Computational properties that hold for the whole program if they could be proved for the individual components of the program are said to be **modular**
- Modularity can be very helpful for proving properties of programs in practice

Extensions of CSR: *Modularity*

- Modularity of (**weak, innermost, simple**) **termination** of CSR has been studied in [PPDP02,RULE02]
- Most of the standard results for the unrestricted case extend to CSR
- In some cases, we need extra syntactical conditions (e.g., LHRV) which certainly amounts to make things difficult in practice



CSR and strategy annotations

CSR and strategy annotations

■ Strategy annotations:

(basically) lists of integers associated to function symbols which specify the ordering in which the arguments are (eventually) evaluated in function calls

■ Extended Visser's classification [Vis01]:

- ◆ E-strategies (or local strategies) [Eke98],
- ◆ Just-in-time [Pol01],
- ◆ Laziness annotations [FKW00],

- ◆ On-Demand E-strategies [OF00].



CSR and E-strategies

CSR and *E-strategies*

■ *E-strategies*:

◆ **Syntactically**: lists of **non-negative integers** associated to function symbols

◆ **Use**: OBJ3, Maude, CafeOBJ

◆ **Notation**: $\varphi(\text{if_then_else_fi}) = (1\ 0)$, in practice:

```
op if_then_else_fi : Bool S S -> S [strat (1 0)] .
```

◆ **Semantics**:

◆ **Positive** integers start the (recursive) evaluation of the corresponding argument of the call

◆ **Zero** indicates that a rule (if any) should be applied to the topmost symbol of the call

CSR and *E-strategies*

- Given an E-strategy map φ , we let μ^φ to be:

$$\mu^\varphi(f) = \{i \text{ in } \varphi(f) \mid i > 0\} \text{ for all symbol } f$$

i.e., we **drop zeros** and **disregard from the ordering** of positive integers

- **Theorem** [PPDP01]: Each computation step induced by an E-strategy map φ corresponds to a (possibly empty) context-sensitive reduction step using μ^φ

CSR and *E-strategies*: semantics

■ Semantics of φ :

- ◆ described by means of a mapping $eval_{\varphi}$ from terms to sets of **evaluated terms**
- ◆ evaluated terms are called **E-normal forms**

■ **Theorem** [PPDP01]: For **zero-ended** strategy maps φ , every E-normal form of a term t is a μ^{φ} -normal form

See [Eke98] for further discussion on zero-ended strategy maps
Latest versions of Maude **only** admit zero-ended strategies

CSR and *E-strategies*: semantics

■ Semantics of :

- ◆ In general, E-normal forms are **not** normal forms...

CSR and *E-strategies*: semantics

Example: Consider the following program:

```
obj LazyLists is
  sorts Nat LNat .
  op 0      : -> Nat .
  op s      : Nat -> Nat .
  op nil    : -> LNat .
  op cons   : Nat LNat -> LNat [strat (1)] .
  op from   : Nat -> LNat .
  op sel    : Nat LNat -> Nat [strat (1 2 0)] .
  op first  : Nat LNat -> Nat [strat (1 2 0)] .
  vars N X  : Nat .
  var L     : LNat .
  eq sel(s(N),cons(X,L)) = sel(X,L) .
  eq sel(0,cons(X,L)) = X .
  eq first(0,L) = nil .
  eq first(s(N),cons(X,L)) = cons(X,first(N,L)) .
  eq from(N) = cons(N,from(s(N))) .
endo
```

CSR and *E-strategies*: semantics

Example: The evaluation of `first(s(0), from(0))` yields
(we use version 1.0.5 of the Maude interpreter)

```
=====
obj LazyLists
=====
reduce in LazyLists : first(s(0), from(0)) .
rewrites: 2 in -10ms cpu (0ms real) (~ rewrites/second)
result LNat: cons(0, first(0, from(s(0))))
```

The E-normal form `cons(0, first(0, from(s(0))))`
is not a normal form!

CSR and *E-strategies*: semantics

- **Corollary** [PPDP01]: Let R be a **left-linear** TRS. For **zero-ended** strategy maps φ such that μ^φ is in CM_R , every E-normal form of a term t is a **head-normal form**

Correctness and completeness regarding **normal forms** is discussed in Alpuente, Escobar & Lucas [WRLA02]

CSR and *E-strategies*: termination

- **Theorem** [PPDP01]: A TRS R is φ -terminating if R is μ^φ -terminating
- The ordering of indices in E-strategies can be important...!

CSR and *E-strategies*: termination

- Consider the programs (cf. [FGK01]):

```
obj Example1 is
  sort S .
  op a      : -> S .
  ops g h   : S -> S [strat (1 0)] .
  op f      : S S -> S [strat (0 1 2)] .
  var X     : S .
  eq f(a,g(X)) = f(a,h(X)) .
  eq h(X) = g(X) .
endo
```



Terminating!



Nonterminating!

```
obj Example2 is
  sort S .
  op a      : -> S .
  ops g h   : S -> S [strat (1 0)] .
  op f      : S S -> S [strat (1 2 0)] .
  var X     : S .
  eq f(a,g(X)) = f(a,h(X)) .
  eq h(X) = g(X) .
endo
```

CSR and *E-strategies*: termination

- Under some conditions, though, such an ordering is not relevant...
- **Theorem [LPAR01]:** Let R be a TRS and φ be an elementary strategy map. Then, R is φ -terminating if and only if R is **innermost** μ^φ -terminating

elementary strategy maps are zero-ended strategies
without multiple occurrences of zero

CSR and *E-strategies*: termination

- Without elementarity, the previous result does not hold!
- **Example [LPAR01]:**

```
obj Example1 is
  sort S .
  op a      : -> S [strat (0)] .
  ops b c   : -> S .
  ops g h   : S -> S [strat (1 0)] .
  op f      : S -> S [strat (0 1)] .
  var X     : S .
  eq f(b) = c .
  eq g(X) = h(X) .
  eq h(c) = g(f(a)) .
  eq a = b .
endo
```

Terminating, but not
innermost μ^φ -terminating:

$$\begin{array}{l} \underline{h(c)} \xrightarrow{\mu} g(\underline{f(a)}) \\ \xrightarrow{\mu} g(\underline{f(b)}) \\ \xrightarrow{\mu} g(\underline{c}) \\ \xrightarrow{\mu} \underline{h(c)} \\ \xrightarrow{\mu} \dots \end{array}$$



CSR and just-in-time strategies

CSR and *just-in-time strategies*

■ Van de Pol's strategy annotations [Pol01]:

◆ Syntactically: lists of (**positive integers** and **labels of rules**) associated to function symbols

◆ Use: JITty [Pol02]

◆ Notation: $\zeta(\text{if}) = [1, \dots, 2, 3, \dots]$, in practice:

rules

```
if1([x,y], if(true,x,y), x)
```

```
if2([x,y], if(false,x,y), y)
```

```
if3([x,y], if(x,y,y), y)
```

strategies

```
if([1,if1,if2,2,3,if3])
```

CSR and *just-in-time strategies*

■ Van de Pol's strategy annotations:

◆ Semantics:

- ◆ **Positive** integers start the (recursive) evaluation of the corresponding argument of the call
- ◆ A **rule label** indicates that such a rule should be applied (if possible) to the topmost symbol of the call

CSR and *just-in-time*: semantics

- Given a strategy annotation ζ , we let μ^ζ to be:

$$\mu^\zeta(f) = \{i \text{ in } \zeta(f) \mid i \text{ is a positive integer}\}$$

i.e., we **drop rule information** and **disregard from the ordering** of positive integers

CSR and *just-in-time*: semantics

■ (Small step) Semantics of ζ :

◆ described by means of a mapping $rewr_{\zeta}$ from terms to terms (or λ)

◆ if $rewr_{\zeta}(t) = \lambda$, we say that t is a ζ -normal form

■ **Theorem** [LPAR01]: For Van de Pol's strategy annotations ζ , if $rewr_{\zeta}(t)$ is a term s (different from λ), then there is a μ^{ζ} -rewriting step from t to s

CSR and *just-in-time*: semantics

- **Theorem** [PROLE01]: For **r-full** strategy annotations ζ , if $rewr_{\zeta}(t)$ is μ^{ζ} -normal, then t is a μ^{ζ} -normal form
- Here, *r-fullness* of ζ means that, for each symbol f , every label for a **rule** defining f is present in $\zeta(f)$

CSR and *just-in-time*: semantics

- **Without** r-fullness, the previous result does not hold!
- **Example:**

```
rules
  if1([x,y], if(true,x,y), x)
  if2([x,y], if(false,x,y), y)
  if3([x,y], if(x,y,y), y)
strategies
  if([1,if1,2,3,if3])
```

← Not r-full

Term `if(false,x,y)` is a ζ -normal form
but it is not a μ^ζ -normal form

CSR and *just-in-time*: semantics

- (Big step) Semantics of ζ :
 - ◆ described by means of a partial function $norm_{\zeta}$ from terms to terms
 - ◆ given a term t , $norm_{\zeta}(t)$ **eventually** yields a normal form if ζ is **full and in-time** [Pol01]
- Here, *fullness* of ζ means that, for each symbol f , every argument index of f and label for an **f -rule** are present in $\zeta(f)$

CSR and *just-in-time*: semantics

- Also, *in-time* means that, for each symbol f , the **needed** indices of an f -rule appear **before** the label in $\zeta(f)$.

- An index i in $\zeta(f)$ is **needed** for

$$:f(l_1, \dots, l_k) \rightarrow r$$

if l_i is not a variable or l_i is a variable that occurs in l_j for some j different from i

CSR and *just-in-time*: semantics

■ Example:

rules

```
if1([x,y], if(true,x,y), x)
```

```
if2([x,y], if(false,x,y), y)
```

```
if3([x,y], if(x,y,y), y)
```

strategies

```
if([1,if1,if2,if3,2,3])
```

needed for if3!

Not in-time

- **Theorem [PROLE01]:** For **r-full** and **in-time** strategy annotations ζ , if $norm_{\zeta}(t)$ is a term s , then s is a μ^{ζ} -normal form of t

CSR and *just-in-time*: semantics

- Being ‘in-time’ does not mean that the computed values are head-normal forms...
- **Example [PROLE01]:**

```
rules
  f1([x], f(x,a), f(b,c))
  g1([], g(f(c,c)), c)
  b1([], b, c)
strategies
  f([2,f1])
  g([1,g1])
  b([b1])
```

← r-full and in-time

We have $norm_{\xi}(g(f(b,b))) = g(f(b,c))$
which is **not** a head-normal form

CSR and *just-in-time*: semantics

- **Theorem** [PROLE01]: Let R be a **left-linear** TRS. For **r-full** and **in-time** strategy annotations ζ such that μ^ζ is in CM_R , if $norm_\zeta(t)$ is a term s , then s is a **head-normal form** of t
- Note that $\mu^\zeta(f) = \{1\}$ in the previous example; *i.e.*, μ^ζ is **not** in CM_R

CSR and *just-in-time*: termination

- **Theorem** [LPAR01]: A TRS R is ζ -terminating if R is μ^ζ -terminating
- **Theorem** [LPAR01]: Let R be a TRS and ζ be an r-full, **elementary** strategy annotation. If R is **innermost** μ^ζ -terminating, then R is ζ -terminating

elementary strategy annotations do not have numeric annotations after rule labels

CSR and *just-in-time*: termination

- In general, ζ -termination does not imply innermost μ^ζ -termination (even for r-full, elementary strategy annotations)
- **Example [PROLE01]:**

```
rules
  b1([], b, a)
  b2([], b, c(b))
strategies
  b([b1,b2])
```

Note that R is clearly ζ -terminating but it is **not** innermost μ^ζ -terminating



CSR and lazy rewriting

CSR and *lazy rewriting*

- **Lazy rewriting**: (only for left-linear TRSs)
 - ◆ **Syntactically**: replacement maps; originally predicates on pairs (symbol, argument index) [FKW00]
 - ◆ **Notation**: $\mu(i_f) = \{1\}$
 - ◆ **Semantics**:
 - ◆ **Positive** integers indicate that those arguments can be freely (but recursively) reduced, **but**
 - ◆ **Absent** indices **can still originate reductions** if they are eventually able to improve the matching of the call against a left-hand side of a rule

CSR and *lazy rewriting*

■ Example [Ngu01]:

$$\boxed{\begin{array}{l} \mu(2nd)=\{1\} \\ \mu(:)=\{1\} \\ \mu(s)=\{1\} \\ \mu(from)=\{1\} \end{array}} \quad \left\{ \begin{array}{l} 2nd(x:y:z) \rightarrow y \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

Evaluation of $2nd(from(0))$ using CSR does not yield the desired result (i.e., $s(0)$):

$$2nd(\underline{from}(0)) \xrightarrow{\mu} 2nd(0:from(s(0)))$$

Position of redex $from(s(0))$ should be made replacing 'for a while'!

CSR and *lazy rewriting*

■ Example [Ngu01]:

$$\boxed{\begin{array}{l} \mu(2nd)=\{1\} \\ \mu(:)=\{1\} \\ \mu(s)=\{1\} \\ \mu(from)=\{1\} \end{array}} \quad \left\{ \begin{array}{l} 2nd(x:y:z) \rightarrow y \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

The left-hand side of rule for `2nd` is used for deciding whether a non-replacing subterm of `2nd(0:from(s(0)))` will be activated...

$$\begin{aligned} 2nd(\underline{from(0)}) &\xrightarrow{\mu} 2nd(0:\underline{from(s(0))}) \\ &\rightarrow 2nd(0:s(0):from(s(s(0)))) \end{aligned}$$

CSR and *lazy rewriting*

■ Example [Ngu01]:

$$\boxed{\begin{array}{l} \mu(2nd)=\{1\} \\ \mu(:)=\{1\} \\ \mu(s)=\{1\} \\ \mu(from)=\{1\} \end{array}} \quad \left\{ \begin{array}{l} 2nd(x:y:z) \rightarrow y \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

The **left-hand side** of rule for `2nd` is used for deciding whether a non-replacing subterm of `2nd(0:from(s(0)))` will be **activated**...

$$\begin{aligned} 2nd(\underline{from(0)}) &\rightarrow_{\mu} 2nd(0:\underline{from(s(0))}) \\ &\rightarrow 2nd(0:s(0):from(s(s(0)))) \end{aligned}$$

Further activations here are not allowed anymore!

CSR and *lazy rewriting*

■ Example [Ngu01]:

$$\boxed{\begin{array}{l} \mu(2nd)=\{1\} \\ \mu(:)=\{1\} \\ \mu(s)=\{1\} \\ \mu(from)=\{1\} \end{array}} \quad \left\{ \begin{array}{l} 2nd(x:y:z) \rightarrow y \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

Now, we can perform the evaluation...

$$\begin{aligned} 2nd(\underline{from(0)}) &\xrightarrow{\mu} 2nd(0:\underline{from(s(0))}) \\ &\rightarrow \underline{2nd(0:s(0):from(s(s(0))))} \\ &\xrightarrow{\mu} s(0) \end{aligned}$$

CSR and *lazy rewriting*

- Lazy rewriting can, then, be thought of as CSR plus reduction steps on additionally activated positions [ENTCSv64]
- **Theorem** [ENTCSv64]: Let R be a **left-linear** TRS and μ in CM_R . Then, CSR and lazy rewriting **coincide**.
- We can prove termination of lazy rewriting as termination of CSR in special cases!

CSR and *lazy rewriting*

- In [ENTCSv64], we have developed a transformation from pairs (R, μ) into pairs (R', μ')
- We proved that μ' -termination of R' implies termination of lazy rewriting for R and μ

CSR and *lazy rewriting*

■ Example [ENTCSv64]: From:

$$\boxed{\begin{array}{l} \mu(2nd)=\{1\} \\ \mu(:)=\{1\} \\ \mu(s)=\{1\} \\ \mu(from)=\{1\} \end{array}} \left\{ \begin{array}{l} 2nd(x:y:z) \rightarrow y \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

we obtain:

$$\boxed{\begin{array}{l} \mu'(2nd)=\{1\} \\ \mu'(:)=\{1\} \\ \mu'(s)=\{1\} \\ \mu'(from)=\{1\} \\ \mu'(:')=\{1,2\} \end{array}} \left\{ \begin{array}{l} 2nd(x: '(y:z)) \rightarrow y \\ 2nd(x:y) \rightarrow 2nd(x: 'y) \\ from(x) \rightarrow x:from(s(x)) \end{array} \right.$$

which can now be proved μ' -terminating.



CSR and on-demand E-strategies

CSR and *on-demand E-strategies*

■ On-demand E-strategies:

- ◆ **Syntactically:** lists of **integers** associated to function symbols
- ◆ **Use:** Proposed for OBJ3 and CafeOBJ; **implemented** in CafeOBJ
- ◆ **Notation:** $\varphi(\text{cons}) = (1 -2)$, in practice:
`op cons : S S -> S [strat (1 -2)] .`
- ◆ **Semantics:** [OF00] and [NO01] gave the first formulations. In Alpuente et al. [LPAR02], we discuss some drawbacks in there, and propose a new operational semantics



Conclusions

Conclusions

■ Theoretical aspects:

- ◆ Context-sensitive rewriting provides a good (basic but extensible) formal **framework** for the analysis of simple (**syntactic**) **replacement restrictions** in rewriting

■ Practical aspects:

- ◆ Existing methods for proving properties of CSR can be used for developing **tools for analyzing programs** using strategy languages with a strong emphasis on syntactic replacement restrictions

References

■ Author's references:

- ◆ [CADE02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings Can Be Context-Sensitive. Proc. of CADE'02, Springer LNAI 2392
- ◆ [ENTCSv64] S. Lucas. Lazy Rewriting and Context-Sensitive Rewriting. ENTCS 64, 2002
- ◆ [ICALP96] S. Lucas. Termination of Context-Sensitive Rewriting by Rewriting. Proc. of ICALP'96, Springer LNCS 1099
- ◆ [JFLP98] S. Lucas. Context-sensitive computations in functional and functional-logic programs. JFLP 1998(2)

References

■ Author's references:

- ◆ [IC02] S. Lucas. Context-Sensitive Rewriting Strategies. Inf. and Comp., to appear
- ◆ [LPAR01] S. Lucas. Termination of Rewriting With Strategy Annotations. Proc. of LPAR'01, Springer LNAI 2250
- ◆ [LPAR02] M. Alpuente, S. Escobar, and S. Lucas. Improving On-Demand Strategy Annotations. Proc. of LPAR'02, Springer LNAI 2514
- ◆ [PPDP01] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. Proc. of ACM PPDP'01

References

■ Author's references:

- ◆ [PPDP02] B. Gramlich and S. Lucas. Modular Termination of Context-Sensitive Rewriting. Proc. of ACM PPDP'02
- ◆ [PROLE01] S. Lucas. Computational properties of term rewriting with strategy annotations. Proc. of PROLE'01, UCLM
- ◆ [RTA02] S. Lucas. Termination of (Canonical) Context-Sensitive Rewriting. Proc. of RTA'02, Springer LNCS 2378

References

■ Author's references:

- ◆ [RULE02] B. Gramlich and S. Lucas. Simple Termination of Context-Sensitive Rewriting. Proc. of ACM RULE'02
- ◆ [WRLA02] M. Alpuente, S. Escobar, and S. Lucas. Correct and complete (positive) strategy annotations for OBJ. ENTCS 71

References

■ Further references:

- ◆ [DKP91] N. Dershowitz, S. Kaplan, and D. Plaisted. Rewrite, rewrite, rewrite, rewrite, rewrite... TCS 83:71-96
- ◆ [Eke98] S. Eker. Term Rewriting with Operator Evaluation Strategies. ENTCS 15
- ◆ [FGK01] O. Fissore, I. Gnaedig, and H. Kirchner. Induction for termination with local strategies. ENTCS 58(2)
- ◆ [FKW00] W. Fokkink, J. Kamperman, and P. Walters. Lazy Rewriting on Eager Machinery. ACM TOPLAS 22(1):45-86

References

■ Further references:

- ◆ [FR99] M.C.F. Ferreira and A.L. Ribeiro. Context-Sensitive AC-Rewriting. Proc. of RTA'99, Springer LNCS 1631
- ◆ [GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. Proc. of RTA'99, Springer LNCS 1631
- ◆ [GM02] J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. TR AIB-2002-02, RWTH Aachen, 2002
- ◆ [GM02b] J. Giesl and A. Middeldorp. Termination of Innermost Context-Sensitive Rewriting. Proc. of DLT'02, Springer LNCS to appear.

References

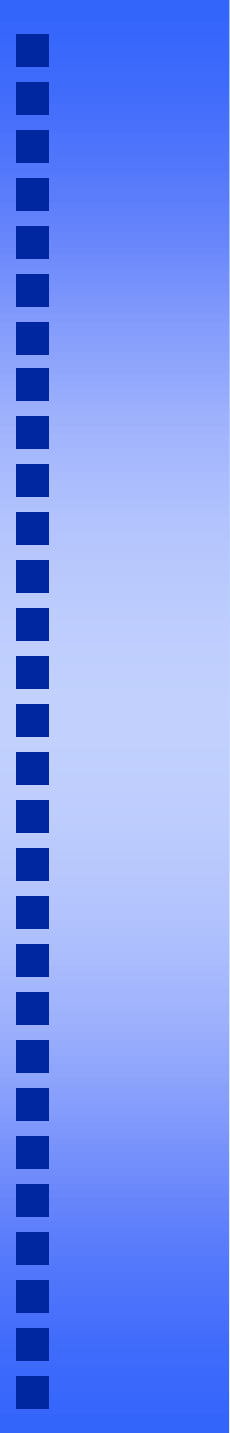
■ Further references:

- ◆ [HL91] G. Huet and J.-J. Lévy. Computations in orthogonal term rewriting systems. Computational logic: essays in honour of Alan Robinson, MIT Press, 1991
- ◆ [Ngu01] Q.-H. Nguyen. Compact Normalisation Trace via Lazy Rewriting. ENTCS 57
- ◆ [NO01] M. Nakamura and K. Ogata. The evaluation strategy for head normal form with and without on-demand flags. ENTCS 36
- ◆ [OF00] K. Ogata and K. Futatsugi. Operational Semantics of Rewriting with the On-demand Evaluation Strategy. Proc of ACM SAC'00

References

■ Further references:

- ◆ [Pol01] J. van de Pol. Just-in-time: on Strategy Annotations. ENTCS 57
- ◆ [Pol02] J. van de Pol. JITty: A Rewriter with Strategy Annotations. Proc. of RTA'02, Springer LNCS 2378
- ◆ [Vis01] E. Visser. A Survey of Strategies in Program Transformation Systems. ENTCS 57
- ◆ [Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. Proc. of RTA'97, Springer LNCS 1232



Context-sensitive rewriting techniques for programs with strategy annotations

Salvador Lucas

Dep. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

`slucas@dsic.upv.es`