Natural Rewriting and Narrowing for General Term Rewriting Systems

Santiago Escobar[1], José Meseguer[2], and Prasanna Thati[2]

[1] UPV, Valencia (Spain) [2] UIUC, Urbana-Champaign (USA)

2nd Workshop on Coq and Rewriting

LIX, Palaiseau (France), September 22-24

Outline

① Motivation
② Demandedness in Natural Rewriting/Narrowing
③ Objective
④ Generalization of demandedness
⑤ Conclusions & Future work

Challenge in Programming Languages

Discovery of sound and complete evaluation strategies:

- optimal w.r.t. efficiency criterion
  ① the number of evaluation steps (needed steps [Huet and Levy])
  ② the avoidance of infinite, failing, or redundant derivations.
- easily implementable
- applicable to a large class of programs

Weakly Needed Rewriting/Narrowing

◇ Sound, complete, and optimal for the class of inductively sequential constructor TRS [Antoy,Hanus,Echahed JACM00].

Weakly Needed Rewriting/Narrowing

- Sound, complete, and optimal for the class of inductively sequential constructor TRS [Antoy,Hanus,Echahed JACM00].
- Some improvement is possible:
  1. Sub-optimal for non-inductively sequential constructor systems.
  2. Non-avoidance of failing computations.

Non-inductively sequential programs

Example:
```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

B(B(T,F,T),$\underline{\texttt{B(F,T,T)}}$,F)

Non-inductively sequential programs

Example:
```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

Optimal computation (only needed steps):

$\texttt{B(B(T,F,T),}\underline{\texttt{B(F,T,T)}}\texttt{,F)} \rightarrow \underline{\texttt{B(B(T,F,T),T,F)}} \rightarrow \texttt{T}$

Non-inductively sequential programs

Example:

```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

Optimal computation (only needed steps):

$\mathtt{B(B(T,F,T),\underline{B(F,T,T)},F)} \rightarrow \underline{\mathtt{B(B(T,F,T),T,F)}} \rightarrow \mathtt{T}$

Needed Rewriting/Narrowing [Antoy,Hanus,Echahed]:

$\mathtt{B(B(T,F,T),\underline{B(F,T,T)},F)} \rightarrow \underline{\mathtt{B(B(T,F,T),T,F)}} \rightarrow \mathtt{T}$

$\mathtt{B(\underline{B(T,F,T)},B(F,T,T),F)} \rightarrow \mathtt{B(T,\underline{B(F,T,T)},F)} \rightarrow \underline{\mathtt{B(T,T,F)}} \rightarrow \mathtt{T}$

Avoidance of failing computations

Example:

$$\texttt{0} \div \texttt{s(N)} = \texttt{0}$$
$$\texttt{s(M)} \div \texttt{s(N)} = \texttt{s((M}-\texttt{N)}\div\texttt{s(N))}$$
$$\vdots$$

$\texttt{10!} \div \texttt{0}$ is a failing term (head-normal form)

Avoidance of failing computations

Example:

$$\begin{aligned} &\texttt{0} \div \texttt{s(N)} = \texttt{0} \\ &\texttt{s(M)} \div \texttt{s(N)} = \texttt{s((M}-\texttt{N)}\div\texttt{s(N))} \\ &\vdots \end{aligned}$$

$\texttt{10!} \div \texttt{0}$ is a failing term (head-normal form)

Needed Rewriting/Narrowing [Antoy,Hanus,Echahed]:

$$\underline{\texttt{10!}} \div \texttt{0} \rightarrow \cdots$$

Natural Rewriting/Narrowing

[Escobar PPDP'03,FLOPS'04]

- Preserves optimal evaluation for sequential parts of a program

$$\texttt{B(B(T,F,T),}\underline{\texttt{B(F,T,T)}}\texttt{,F)} \xrightarrow{\mathfrak{m}} \texttt{B(B(T,F,T),T,F)} \xrightarrow{\mathfrak{m}}{}^{*} \texttt{T}$$

- Avoids unnecessary evaluation for (more) failing terms

$$\texttt{10!} \div \texttt{0} \stackrel{\mathfrak{m}}{\not\rightarrow}$$

Demandedness in Natural Rewriting

1. Reduce a term $\texttt{f(}t_1\texttt{,...,}t_n\texttt{)}$ at top position (if possible)
2. Otherwise, reduce an argument t_i if it might promote the application of a rule (t_i is demanded by a rule)

Example:

```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

Term $t = \texttt{B(B(T,F,T),B(F,T,T),F)}$ is not reducible.
Positions 1 and 2 are demanded.

Failing terms

- Discard demandedness data from rules with a constructor clash

---

Example:

```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

Term $t = \mathtt{B(B(T,F,T),B(F,T,T),F)}$ is not reducible.

Position 1 is no longer demanded by 2nd rule.

Most demanded positions

- Select those demanded positions which are the most (frequently) demanded positions

---

Example:

```
B(T,F,X) = T
B(F,X,T) = T
B(X,T,F) = T
```

Term $t = \texttt{B(B(T,F,T),B(F,T,T),F)}$ is not reducible.

Position 2 is the most demanded position and thus selected for reduction.

Completeness

- Do not miss computations

Example:

$$\begin{aligned} \texttt{True} \vee \texttt{X} &= \texttt{True} \\ \texttt{X} \vee \texttt{True} &= \texttt{True} \\ \texttt{False} \vee \texttt{X} &= \texttt{X} \end{aligned}$$

Term $t = (\texttt{True} \vee \texttt{False}) \vee (\texttt{True} \vee \texttt{False})$ is not reducible.
Position 1 is the most demanded position.
However, both positions are necessary (non-determinism)
and thus reduced by natural rewriting.

Properties of Natural Rewriting/Narrowing

Defined in [Escobar PPDP'03]

1. Natural rewriting and narrowing preserve optimality for a larger class of programs than needed rewriting/narrowing [Antoy,Hanus,Echahed]
2. [Correctness] Natural rewriting and narrowing compute head-normal forms for left-linear constructor systems
3. [Completeness] Natural rewriting and narrowing compute semantically equivalent head-normal forms for left-linear contructor systems

Properties of Natural Rewriting/Narrowing

Defined in [Escobar PPDP'03]

1. Natural rewriting and narrowing preserve optimality for a larger class of programs than needed rewriting/narrowing [Antoy,Hanus,Echahed]
2. [Correctness] Natural rewriting and narrowing compute head-normal forms for left-linear constructor systems
3. [Completeness] Natural rewriting and narrowing compute semantically equivalent head-normal forms for left-linear contructor systems
4. Easily implementable using matching definitional trees [Escobar FLOPS'04] (a generalization of definitional trees [Antoy ALP'92]).

Objective

Left-linearity and constructor conditions are too restrictive
for equational programming languages such as
OBJ, CafeOBJ, ASF+SDF, Maude, or ELAN.

Objective

Left-linearity and constructor conditions are too restrictive
for equational programming languages such as
OBJ, CafeOBJ, ASF+SDF, Maude, or ELAN.

$\Downarrow$

How to preserve good properties and be complete for
general term rewriting systems

That is, how to generalize the demandedness notion

Generalization of Demandedness

- [Non-linear variables] Compute common parts of term t under non-linear variables in lhs l (using least general context)
- [Non-constructor lhs's] Analyze defined symbols above positions of term t w.r.t. lhs l.

  1) when l is matching and 2) when l is not matching

Generalization of Demandedness

- [Non-linear variables] Compute common parts of term t under non-linear variables in lhs l (using least general context)
- [Non-constructor lhs's] Analyze defined symbols above positions of term t w.r.t. lhs l.

  1) when l is matching and 2) when l is not matching
- Adaptation of failing terms and most frequently demanded notions.

Non-linear variables

1. Least general context s for term t and lhs l
2. Least general context for subterms in t under the same variable in l
3. Plug them into s and obtain s'.
4. Demanded positions are variable positions in s'

Non-linear variables

1. Least general context s for term t and lhs l
2. Least general context for subterms in t under the same variable in l
3. Plug them into s and obtain s'.
4. Demanded positions are variable positions in s'

Example: (1) `M % s(N)` $\rightarrow$ `(M-s(N)) % s(N)`
(2) `(0 - s(M)) % s(N)` $\rightarrow$ `N - M` ...
(3) `X` $\approx$ `X` $\rightarrow$ `True`

Term $t =$ `10! % (1-1)` $\approx$ `10! % 0` is not reducible.

Least general context with plugged data: `10! % Y` $\approx$ `10! % Y`

Positions 1.2 and 2.2 are demanded

Non-constructor left-hand side

1. Analyze also defined symbols above demanded positions when lhs l is not matching

---

$\boxed{\text{Example:}}$ (1) $\mathtt{M}\ \%\ \mathtt{s(N)} \rightarrow \mathtt{(M{-}s(N))}\ \%\ \mathtt{s(N)}$
(2) $\mathtt{(0\ -\ s(M))}\ \%\ \mathtt{s(N)} \rightarrow \mathtt{N} - \mathtt{M}$ $\quad \ldots$
(3) $\mathtt{X} \approx \mathtt{X} \rightarrow \mathtt{True}$

Term $t = \mathtt{10!}\ \%\ \mathtt{(1{-}1)} \approx \mathtt{10!}\ \%\ \mathtt{0}$

Positions 1.2 and 2.2 demanded

Non-constructor left-hand side

1. Analyze also defined symbols above demanded positions when lhs l is not matching

---

$\boxed{\text{Example:}}$ (1) `M % s(N) → (M-s(N)) % s(N)`
(2) `(0 - s(M)) % s(N) → N - M` ...
(3) `X ≈ X → True`

Term $t =$ `10! % (1-1) ≈ 10! % 0`

$\Downarrow$ Positions 1.2 and 2.2 demanded $\Downarrow$

Analyze positions 1 and 2 recursively

Non-constructor left-hand side

1. Analyze also defined symbols above demanded positions when lhs l is not matching

---

$\boxed{\text{Example:}}$ (1) $\texttt{M} \ \% \ \texttt{s(N)} \rightarrow \texttt{(M}-\texttt{s(N))} \ \% \ \texttt{s(N)}$
(2) $\texttt{(0} - \texttt{s(M))} \ \% \ \texttt{s(N)} \rightarrow \texttt{N} - \texttt{M}$ $\quad \ldots$
(3) $\texttt{X} \approx \texttt{X} \rightarrow \texttt{True}$

Term $t = \texttt{10!} \,\%\, \texttt{(1}-\texttt{1)} \approx \texttt{10!} \,\%\, \texttt{0}$

$\Downarrow$ Positions 1.2 and 2.2 demanded $\Downarrow$

$\Downarrow$ Analyze positions 1 and 2 recursively $\Downarrow$

(1) Subterm $\texttt{10!} \,\%\, \texttt{0}$ is failing

(2) Position 2 most frequently demanded for $\texttt{10!} \,\%\, \texttt{(1}-\texttt{1)}$

Non-constructor left-hand side

1. Analyze also defined symbols above demanded positions when lhs l is not matching

---

$\boxed{\text{Example:}}$ (1) $\texttt{M} \ \% \ \texttt{s(N)} \rightarrow (\texttt{M}-\texttt{s(N)}) \ \% \ \texttt{s(N)}$
(2) $(\texttt{0} - \texttt{s(M)}) \ \% \ \texttt{s(N)} \rightarrow \texttt{N} - \texttt{M}$ ...
(3) $\texttt{X} \approx \texttt{X} \rightarrow \texttt{True}$

Term $t = \texttt{10!} \, \% \, (\texttt{1}-\texttt{1}) \approx \texttt{10!} \, \% \, \texttt{0}$

$\Downarrow$ Positions 1.2 and 2.2 demanded $\Downarrow$

$\Downarrow$ Analyze positions 1 and 2 recursively $\Downarrow$

(1) Subterm $\texttt{10!} \, \% \, \texttt{0}$ is failing

(2) Position 2 most frequently demanded for $\texttt{10!} \, \% \, (\texttt{1}-\texttt{1})$

Total demanded positions: 1.2 and 2.2

Non-constructor left-hand side

2. Analyze also defined symbols above variables when a lhs is matching

---

$\boxed{\text{Example:}}$ (i) `first(pair(X,Y))` $\longrightarrow$ `X`

(ii) `pair(X,Y)` $\longrightarrow$ `pair(Y,X)`

Term $t =$ `first(pair(a,b))` is matching rule (i)

$\underline{\texttt{first(pair(a,b))}} \longrightarrow \texttt{a}$

Non-constructor left-hand side

2. Analyze also defined symbols above variables when a lhs is matching

---

Example: (i) `first(pair(X,Y))` $\rightarrow$ `X`

(ii) `pair(X,Y)` $\rightarrow$ `pair(Y,X)`

Term $t =$ `first(pair(a,b))` is matching rule (i)

$\underline{\texttt{first(pair(a,b))}} \rightarrow \texttt{a}$

However, an alternative reduction sequence is necessary

$\texttt{first(}\underline{\texttt{pair(a,b)}}\texttt{)} \rightarrow \underline{\texttt{first(pair(b,a))}} \rightarrow \texttt{b}$

Generalized Natural Rewriting

Example: (1) $\mathtt{M}\ \%\ \mathtt{s(N)} \rightarrow (\mathtt{M}-\mathtt{s(N)})\ \%\ \mathtt{s(N)}$
(2) $(\mathtt{0} - \mathtt{s(M)})\ \%\ \mathtt{s(N)} \rightarrow \mathtt{N} - \mathtt{M}$ ...
(3) $\mathtt{X} \approx \mathtt{X} \rightarrow \mathtt{True}$

Term $t = \mathtt{10!}\ \%\ (\mathtt{1}-\mathtt{1}) \approx \mathtt{10!}\ \%\ \mathtt{0}$

Position 1.2 is the only one reducible

Generalized Natural Rewriting

Example: (1) $\texttt{M} \ \% \ \texttt{s(N)} \to \texttt{(M}-\texttt{s(N))} \ \% \ \texttt{s(N)}$
(2) $\texttt{(0} - \texttt{s(M))} \ \% \ \texttt{s(N)} \to \texttt{N} - \texttt{M}$ ...
(3) $\texttt{X} \approx \texttt{X} \to \texttt{True}$

Term $t = \texttt{10!} \, \% \, \texttt{(1}-\texttt{1)} \approx \texttt{10!} \, \% \, \texttt{0}$

$\Downarrow$ Position 1.2 is the only one reducible $\Downarrow$

$\texttt{10!} \, \% \, \texttt{0} \approx \texttt{10!} \, \% \, \texttt{0}$

Reduce at root (no demanded positions)

Generalized Natural Rewriting

Example: (1) $\mathtt{M}\ \%\ \mathtt{s(N)} \to (\mathtt{M}-\mathtt{s(N)})\ \%\ \mathtt{s(N)}$
(2) $(\mathtt{0} - \mathtt{s(M)})\ \%\ \mathtt{s(N)} \to \mathtt{N} - \mathtt{M}$ ...
(3) $\mathtt{X} \approx \mathtt{X} \to \mathtt{True}$

Term $t = \mathtt{10!}\ \%\ (\mathtt{1}-\mathtt{1}) \approx \mathtt{10!}\ \%\ \mathtt{0}$

$\Downarrow$ Position 1.2 is the only one reducible $\Downarrow$

$\mathtt{10!}\ \%\ \mathtt{0} \approx \mathtt{10!}\ \%\ \mathtt{0}$

$\Downarrow$ Reduce at root (no demanded positions) $\Downarrow$

$\mathtt{True}$

No reduction on $\mathtt{10!}$ terms $\Rightarrow$ optimal (lazy) behavior

Generalized Natural Narrowing

- Incremental construction of unifiers to compute only needed narrowing steps [Antoy,Hanus,Echahed JACM00] (counterpart of Huet&Levy needed steps)

Example:

$$
\begin{array}{ll}
\mathtt{min(0,Y)} \to \mathtt{0} & \mathtt{0} + \mathtt{Y} \to \mathtt{Y} \\
\mathtt{min(s(X),0)} \to \mathtt{0} & \mathtt{s(X)} + \mathtt{Y} \to \mathtt{s(X} + \mathtt{Y)} \\
\mathtt{min(s(X),s(Y))} \to \mathtt{s(min(X,Y))} & \mathtt{X} \approx \mathtt{X} \to \mathtt{True}
\end{array}
$$

Term $t = \mathtt{min(X,0} + \mathtt{X)} \approx \mathtt{0}$

Generalized Natural Narrowing

- Incremental construction of unifiers to compute only needed narrowing steps [Antoy,Hanus,Echahed JACM00] (counterpart of Huet&Levy needed steps)

Example:

$$\begin{array}{llll} \mathtt{min(0,Y)} \to \mathtt{0} & & \mathtt{0 + Y} \to \mathtt{Y} \\ \mathtt{min(s(X),0)} \to \mathtt{0} & & \mathtt{s(X) + Y} \to \mathtt{s(X + Y)} \\ \mathtt{min(s(X),s(Y))} \to \mathtt{s(min(X,Y))} & & \mathtt{X} \approx \mathtt{X} \to \mathtt{True} \end{array}$$

Term $t = \mathtt{min(X,0 + X)} \approx \mathtt{0}$

1) $\underline{\mathtt{min(X,0 + X)}} \approx \mathtt{0} \leadsto_{[\mathtt{X} \mapsto \mathtt{0}]} \mathtt{0} \approx \mathtt{0} \leadsto \mathtt{True}$

Generalized Natural Narrowing

- Incremental construction of unifiers to compute only needed narrowing steps [Antoy,Hanus,Echahed JACM00] (counterpart of Huet&Levy needed steps)

Example:

$$\begin{array}{lll} \mathtt{min(0,Y)} \to \mathtt{0} & & \mathtt{0 + Y} \to \mathtt{Y} \\ \mathtt{min(s(X),0)} \to \mathtt{0} & & \mathtt{s(X) + Y} \to \mathtt{s(X + Y)} \\ \mathtt{min(s(X),s(Y))} \to \mathtt{s(min(X,Y))} & & \mathtt{X} \approx \mathtt{X} \to \mathtt{True} \end{array}$$

Term $t = \mathtt{min(X,0 + X)} \approx \mathtt{0}$

1) $\underline{\mathtt{min(X,0 + X)}} \approx \mathtt{0} \leadsto_{[\mathtt{X} \mapsto \mathtt{0}]} \mathtt{0} \approx \mathtt{0} \leadsto \mathtt{True}$

2) $\mathtt{min(X,}\underline{\mathtt{0 + X}}\mathtt{)} \approx \mathtt{0} \leadsto_{[\mathtt{X} \mapsto \mathtt{s(X')}]} \mathtt{min(s(X'),s(X'))} \approx \mathtt{0} \leadsto^{*} \mathtt{s(\cdots)} \approx 0$

Generalized Natural Narrowing

- Incremental construction of unifiers to compute only needed narrowing steps [Antoy,Hanus,Echahed JACM00] (counterpart of Huet&Levy needed steps)

Example:

$$\begin{array}{lll} \mathtt{min(0,Y)} \to \mathtt{0} & \quad & \mathtt{0 + Y} \to \mathtt{Y} \\ \mathtt{min(s(X),0)} \to \mathtt{0} & & \mathtt{s(X) + Y} \to \mathtt{s(X + Y)} \\ \mathtt{min(s(X),s(Y))} \to \mathtt{s(min(X,Y))} & & \mathtt{X} \approx \mathtt{X} \to \mathtt{True} \end{array}$$

Term $t = \mathtt{min(X,0 + X)} \approx \mathtt{0}$

1) $\underline{\mathtt{min(X,0 + X)}} \approx \mathtt{0} \leadsto_{[\mathtt{X} \mapsto \mathtt{0}]} \mathtt{0} \approx \mathtt{0} \leadsto \mathtt{True}$

2) $\mathtt{min(X,}\underline{\mathtt{0 + X}}\mathtt{)} \approx \mathtt{0} \leadsto_{[\mathtt{X} \mapsto \mathtt{s(X')}]} \mathtt{min(s(X'),s(X'))} \approx \mathtt{0} \leadsto^{*} \mathtt{s}(\cdots) \approx 0$

Instantiate X before analyzing any position in t

Avoids 1b) $\mathtt{min(X,}\underline{\mathtt{0 + X}}\mathtt{)} \approx \mathtt{0} \leadsto \mathtt{min(X,X)} \approx \mathtt{0} \cdots$

Properties

1. Conservative extension of natural rewriting/narrowing [Escobar PPDP'03]
   (a) Preserves optimality for sequential parts (now more possibilities)
   (b) Avoids many failing computations (now more possibilities)
2. [Correctness] Natural rewriting/narrowing computes head-normal forms for general TRS's
3. [Completeness] Natural rewriting/narrowing computes semantically equivalent head-normal forms for general TRS's (narrowing only for normalized substitutions)

Conclusions

- Natural rewriting/narrowing [Escobar PPDP'03] best known strategy for left-linear constructor TRS's
- Generalization of Natural Rewriting for general TRS's (non-linear variables & non-constructor lhs's)
- Good properties (inherited from PPDP'03 but applicable to new situations):
  - avoidance failing computations
  - preservation optimality for sequential parts of the program
- Correct and complete for general term rewriting systems

Conclusions

- Optimal narrowing strategies are very important for several research fields:
  - Model checking by reachability analyses [Meseguer&Thati WRLA04]
  - Theorem proving (based on paramodulation or related narrowing methods)
  - Programming languages (functional&logic like Curry or TOY, equational languages like Maude)
  - Several techniques based on narrowing (such as Partial Evaluation and some program transformations)

Future Work

- Study class of programs with optimality results (difficult outside left-linear constructor TRS's)
- Extension for rewriting and narrowing modulo equational theories (AC-rewriting, AC-narrowing)
- Implementation (generalize matching definitional trees from FLOPS'04)

Future work: A-narrowing

- A-narrowing undecidable for some input terms in a program. However, a strategy can improve behavior

Future work: A-narrowing

- A-narrowing undecidable for some input terms in a program. However, a strategy can improve behavior

---

Example:

```
B(T,F,X) = T        T & X = X
B(F,X,T) = T        F & X = F
B(X,T,F) = T
```

& associative

Future work: A-narrowing

- A-narrowing undecidable for some input terms in a program. However, a strategy can improve behavior

---

Example:

`B(T,F,X) = T`	`T & X = X`
`B(F,X,T) = T`	`F & X = F`
`B(X,T,F) = T`	`&` associative

Infinite number of narrowing steps

$\mathtt{X\ \&\ T} \rightsquigarrow_{[\mathtt{X} \mapsto \mathtt{T}]} \mathtt{T}$ / $\mathtt{X\ \&\ T} \rightsquigarrow_{[\mathtt{X} \mapsto \mathtt{T\ \&\ T}]} \mathtt{T\ \&\ T}$ / $\mathtt{X\ \&\ T} \ldots$

Future work: A-narrowing

- A-narrowing undecidable for some input terms in a program. However, a strategy can improve behavior

---

Example:

$$\texttt{B(T,F,X) = T} \qquad \texttt{T \& X = X}$$

$$\texttt{B(F,X,T) = T} \qquad \texttt{F \& X = F}$$

$$\texttt{B(X,T,F) = T} \qquad \text{\& associative}$$

Infinite number of narrowing steps

$$\texttt{X \& T} \leadsto_{[\texttt{X} \mapsto \texttt{T}]} \texttt{T} \;\;/\;\; \texttt{X \& T} \leadsto_{[\texttt{X} \mapsto \texttt{T \& T}]} \texttt{T \& T} \;\;/\;\; \texttt{X \& T} \ldots$$

Optimal computation

$$\texttt{B(X \& T,}\underline{\texttt{T \& T}}\texttt{,F)} \leadsto \underline{\texttt{B(X \& T,T,F)}} \leadsto \texttt{T}$$