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
  - the number of evaluation steps ([needed steps][Huet and Levy])
  - 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:

\[ B(T,F,X) = T \]
\[ B(F,X,T) = T \]
\[ B(X,T,F) = T \]

\[ B(B(T,F,T),B(F,T,T),F) \]
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):
\[ B(B(T,F,T),B(F,T,T),F) \rightarrow B(B(T,F,T),T,F) \rightarrow T \]
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):

\[
B(B(T, F, T), B(F, T, T), F) \rightarrow B(B(T, F, T), T, F) \rightarrow T
\]

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

\[
B(B(T, F, T), B(F, T, T), F) \rightarrow B(B(T, F, T), T, F) \rightarrow T \\
B(B(T, F, T), B(F, T, T), F) \rightarrow B(T, B(F, T, T), F) \rightarrow B(T, T, F) \rightarrow T
\]
Avoidance of failing computations

Example: \[ 0 \div s(N) = 0 \]
\[ s(M) \div s(N) = s((M-N)\div s(N)) \]
\[ \vdots \]

10! \div 0 \text{ is a failing term (head-normal form)}
Avoidance of failing computations

Example:

\[
0 \div s(N) = 0 \\
s(M) \div s(N) = s((M-N) \div s(N)) \\
\vdots
\]

\[
10! \div 0 \quad \text{is a failing term (head-normal form)}
\]

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

\[
10! \div 0 \rightarrow \ldots
\]
Natural Rewriting/Narrowing

[Escobar PPDP’03,FLOPS’04]

- Preserves optimal evaluation for sequential parts of a program

\[ B(B(T,F,T),B(F,T,T),F) \xrightarrow{m} B(B(T,F,T),T,F) \xrightarrow{m^*} T \]

- Avoids unnecessary evaluation for (more) failing terms

\[ 10! \div 0 \xrightarrow{m} \]
Demandedness in Natural Rewriting

1. Reduce a term $f(t_1, \ldots, t_n)$ 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:

\[
\begin{align*}
B(T, F, X) &= T \\
B(F, X, T) &= T \\
B(X, T, F) &= T
\end{align*}
\]

Term $t = B(B(T, F, T), B(F, T, T), F)$ is not reducible. Positions 1 and 2 are demanded.
Failing terms

- **Discard** demandedness data from rules with a constructor clash

Example:

\[
\begin{align*}
B(T,F,X) &= T \\
B(F,X,T) &= T \\
B(X,T,F) &= T
\end{align*}
\]

Term \( t = B(B(T,F,T),B(F,T,T),F) \) 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 \( t = B(B(T, F, T), B(F, T, T), F) \) is not reducible. Position 2 is the most demanded position and thus selected for reduction.
Completeness

- Do not miss computations

Example:  
True \lor X = True
X \lor True = True
False \lor X = X

Term \( t = (\text{True} \lor \text{False}) \lor (\text{True} \lor \text{False}) \) 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 constructor 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 constructor 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.

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. $(0 - s(M)) \% s(N) \rightarrow N - M \ldots$
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.
1. Analyze also **defined** symbols **above demanded positions** whenlhs l is not matching

**Example:**

1. \( M \% s(N) \rightarrow (M - s(N)) \% s(N) \)
2. \( (0 - s(M)) \% s(N) \rightarrow N - M \) \[
\]
3. \( X \approx X \rightarrow \text{True} \)

Term \( t = 10! \% (1 - 1) \approx 10! \% 0 \)

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

---

**Example:**

(1) $M \% s(N) \rightarrow (M - s(N)) \% s(N)$

(2) $(0 - s(M)) \% s(N) \rightarrow N - M \ldots$

(3) $X \approx X \rightarrow \text{True}$

**Term** $t = 10! \% (1+1) \approx 10! \% 0$

$\downarrow$ 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

Example:

(1) \[ M \% s(N) \rightarrow (M\!-\!s(N)) \% s(N) \]
(2) \[ (0 \rightarrow s(M)) \% s(N) \rightarrow N \rightarrow M \ldots \]
(3) \[ X \approx X \rightarrow True \]

Term \( t = 10! \% (1\!-\!1) \approx 10! \% 0 \)

\( \downarrow \) Positions 1.2 and 2.2 demanded \( \downarrow \)

\( \downarrow \) Analyze positions 1 and 2 recursively \( \downarrow \)

(1) Subterm \( 10! \% 0 \) is failing

(2) Position 2 most frequently demanded for \( 10! \% (1\!-\!1) \)
GENERALIZATION OF DEMANDEDNESS

Non-constructor left-hand side

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

Example:

(1) \( M \% s(N) \rightarrow (M-s(N)) \% s(N) \)
(2) \( (0 - s(M)) \% s(N) \rightarrow N - M \ldots \)
(3) \( X \approx X \rightarrow \text{True} \)

Term \( t = 10! \% (1-1) \approx 10! \% 0 \)

\[\Downarrow\] Positions 1.2 and 2.2 demanded \[\Downarrow\]
\[\Downarrow\] Analyze positions 1 and 2 recursively \[\Downarrow\]

(1) Subterm \( 10! \% 0 \) is failing

(2) Position 2 most frequently demanded for \( 10! \% (1-1) \)

Total demanded positions: 1.2 and 2.2
2. Analyze also defined symbols above variables when a lhs is matching

Example: (i) \(\text{first}(\text{pair}(X,Y)) \rightarrow X\)

(ii) \(\text{pair}(X,Y) \rightarrow \text{pair}(Y,X)\)

Term \(t = \text{first}(\text{pair}(a,b))\) is matching rule (i)

\(\text{first}(\text{pair}(a,b)) \rightarrow a\)
Non-constructor left-hand side

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

Example:

(i) \texttt{first(pair(X,Y))} \rightarrow X
(ii) \texttt{pair(X,Y)} \rightarrow \texttt{pair(Y,X)}

Term \( t = \texttt{first(pair(a,b))} \) is matching rule (i)

\[
\texttt{first(pair(a,b))} \rightarrow a
\]

However, an alternative reduction sequence is necessary

\[
\texttt{first(pair(a,b))} \rightarrow \texttt{first(pair(b,a))} \rightarrow b
\]
Generalized Natural Rewriting

Example:

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

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

Position 1.2 is the only one reducible
Generalization of Demandedness

Generalized Natural Rewriting

Example:

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

Term \( t = 10! \% (1 - 1) \approx 10! \% 0 \)

\(\downarrow\) Position 1.2 is the only one reducible \(\downarrow\)

\(10! \% 0 \approx 10! \% 0\)

Reduce at root (no demanded positions)
Generalized Natural Rewriting

Example:

(1) $M \% s(N) \to (M-s(N)) \% s(N)$
(2) $(0 \to s(M)) \% s(N) \to N - M \ldots$
(3) $X \approx X \to True$

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

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

$10! \% 0 \approx 10! \% 0$

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

True

No reduction on 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:**

- \( \min(0, Y) \rightarrow 0 \)
- \( 0 + Y \rightarrow Y \)
- \( \min(s(X), 0) \rightarrow 0 \)
- \( s(X) + Y \rightarrow s(X + Y) \)
- \( \min(s(X), s(Y)) \rightarrow s(\min(X, Y)) \)
- \( X \approx X \rightarrow \text{True} \)

Term \( t = \min(X, 0 + X) \approx 0 \)
Generalized Natural Narrowing

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

Example:

\[
\begin{align*}
\min(0,Y) & \rightarrow 0 & 0 + Y & \rightarrow Y \\
\min(s(X),0) & \rightarrow 0 & s(X) + Y & \rightarrow s(X + Y) \\
\min(s(X),s(Y)) & \rightarrow s(\min(X,Y)) & X & \approx X \rightarrow \text{True}
\end{align*}
\]

Term \( t = \min(X,0 + X) \approx 0 \)

1) \( \min(X,0 + X) \approx 0 \xrightarrow{[X \rightarrow 0]} 0 \approx 0 \xrightarrow{\text{True}} \)
Generalized Natural Narrowing

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

Example:

\[
\begin{align*}
\text{min}(0, Y) & \rightarrow 0 & 0 + Y & \rightarrow Y \\
\text{min}(s(X), 0) & \rightarrow 0 & s(X) + Y & \rightarrow s(X + Y) \\
\text{min}(s(X), s(Y)) & \rightarrow s(\text{min}(X, Y)) & X & \approx X \rightarrow \text{True}
\end{align*}
\]

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

1) \( \text{min}(X, 0 + X) \approx 0 \overset{[X \rightarrow 0]}{\approx} 0 \overset{[X \rightarrow 0]}{\approx} \text{True} \)

2) \( \text{min}(X, 0 + X) \approx 0 \overset{[X \rightarrow s(X')] }{\approx} \text{min}(s(X'), s(X')) \approx 0 \overset{*}{\approx} s(\cdots) \approx 0 \)
Generalized Natural Narrowing

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

**Example:**

\[
\begin{align*}
\text{min}(0, Y) &\rightarrow 0 \\
\text{min}(s(X), 0) &\rightarrow 0 \\
\text{min}(s(X), s(Y)) &\rightarrow s(\text{min}(X, Y)) \\
\end{align*}
\]

\[
\begin{align*}
0 + Y &\rightarrow Y \\
s(X) + Y &\rightarrow s(X + Y) \\
X &\approx X \rightarrow \text{True}
\end{align*}
\]

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

1. \( \text{min}(X, 0 + X) \approx 0 \rightarrow [X \mapsto 0] 0 \approx 0 \rightarrow \text{True} \)

2. \( \text{min}(X, 0 + X) \approx 0 \rightarrow [X \mapsto s(X')] \text{min}(s(X'), s(X')) \approx 0 \rightarrow^* s(\ldots) \approx 0 \)

Instantiate \( x \) before analyzing any position in \( t \)

Avoids 1b) \( \text{min}(X, 0 + X) \approx 0 \rightarrow \text{min}(X, X) \approx 0 \ldots \)
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:

\[
\begin{align*}
B(T,F,X) &= T & T &\land X &= X \\
B(F,X,T) &= T & F &\land X &= F \\
B(X,T,F) &= T & \text{& associative}
\end{align*}
\]
Future work: A-narrowing

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

Example:

\[
\begin{align*}
B(T,F,X) &= T & T \land X &= X \\
B(F,X,T) &= T & F \land X &= F \\
B(X,T,F) &= T & \text{& associative}
\end{align*}
\]

Infinite number of narrowing steps:

\[
\begin{align*}
X \land T \leadsto_{[X \mapsto T]} T & \quad / \quad X \land T \leadsto_{[X \mapsto T \land T]} T \land T & \quad / \quad X \land T \ldots
\end{align*}
\]
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

\[X \& T \leadsto [X \leftarrow T] T \quad / \quad X \& T \leadsto [X \leftarrow T \& T] T \& T \quad / \quad X \& T \ldots\]

Optimal computation

\[B(X \& T,T \& T,F) \leadsto B(X \& T,T,F) \leadsto T\]