Extending Context-Sensitivity in Term Rewriting

AJSW 2010

Felix Schernhammer

Aug 03, 2010

Motivation

Motivation

- Simple context-restrictions have proven to be useful for guiding rewriting derivations.
- Currently, *context-sensitive rewriting* is the only well-established form of restricted rewriting apart from position based reduction strategies.
- Context-sensitive rewriting can only be applied to restricted classes of rewrite systems.

Motivation

Motivation

- Simple context-restrictions have proven to be useful for guiding rewriting derivations.
- Currently, *context-sensitive rewriting* is the only well-established form of restricted rewriting apart from position based reduction strategies.
- Context-sensitive rewriting can only be applied to restricted classes of rewrite systems.

Example

$$\begin{aligned} nats(x) &\rightarrow x : nats(s(x)) \\ 2nd(x : (y : rest)) &\rightarrow y \end{aligned}$$

Motivation cont'd

Problems

- Can we use more general forms of context-restrictions in order to handle more systems?
- How to do it in a simple way, so that restricted rewrite relations remain feasible?

Forbidden patterns

Idea: Use term patterns to determine positions inside a term that are allowed (resp. forbidden) for reduction.

Rewriting with replacement restrictions

What is (at least) needed to ensure feasibility of a restricted rewrite relation?

Completeness

Ability of a rewrite system with restrictions to simulate unrestricted normalizing reductions to a certain extent.

Termination

The restricted rewrite relation should be (provably) terminating at least on *normalizing* terms.

Forbidden patterns by example

Example

$$\begin{aligned} nats(x) &\rightarrow x : nats(s(x)) \\ 2nd(x : (y : rest)) &\rightarrow y \end{aligned}$$

It is not possible to forbid the reduction of the second argument of ":", as completeness is lost in this case.

Idea

Lists may be evaluated up to their second argument but not any further! $\rightarrow$ *Forbidden Patterns*:

$$\langle x : (y : zs), 2.2 \rangle$$

Rewriting with forbidden patterns

Forbidden patterns

A forbidden pattern (w.r.t. to a signature Σ) is a triple $\langle t, q, \lambda \rangle$ where

- t is a term over $\mathcal{T}(\Sigma, V)$,
- q is a position in t, and
- $\lambda \in \{h, b, a\}$ is a flag indicating whether reductions at (h for here), outside, or inside q are forbidden.

Rewriting with forbidden patterns

Forbidden patterns

A forbidden pattern (w.r.t. to a signature Σ) is a triple $\langle t, q, \lambda \rangle$ where

- t is a term over $\mathcal{T}(\Sigma, V)$,
- q is a position in t, and
- $\lambda \in \{h, b, a\}$ is a flag indicating whether reductions at (h for here), outside, or inside q are forbidden.

Forbidden pattern for the "2nd" TRS

$$\langle x : (y : zs), 2.2, h \rangle$$

Rewriting with forbidden patterns

Rewriting with forbidden patterns

Let $\mathcal{R} = (\Sigma, V)$ be a TRS and δ_Σ a set of forbidden patterns.

$s \to_{\delta_\Sigma} t$ if $s \xrightarrow{p}_{\mathcal{R}} t$ and there is no pattern $\langle u, q, \lambda \rangle \in \delta_\Sigma$, such that

- $s = C[u\sigma]_{q'}$ and $p = q'.q$ if $\lambda = h$,
- $s = C[u\sigma]_{q'}$ and $p > q'.q$ for some q', if $\lambda = b$, and
- $s = C[u\sigma]_{q'}$ and $p < q'.q$ for some q', if $\lambda = a$.

Rewriting with forbidden patterns

Rewriting with forbidden patterns

Let $\mathcal{R} = (\Sigma, V)$ be a TRS and δ_Σ a set of forbidden patterns.

$s \to_{\delta_\Sigma} t$ if $s \xrightarrow{p}_{\mathcal{R}} t$ and there is no pattern $\langle u, q, \lambda \rangle \in \delta_\Sigma$, such that

- $s = C[u\sigma]_{q'}$ and $p = q'.q$ if $\lambda = h$,
- $s = C[u\sigma]_{q'}$ and $p > q'.q$ for some q', if $\lambda = b$, and
- $s = C[u\sigma]_{q'}$ and $p < q'.q$ for some q', if $\lambda = a$.

Forbidden pattern relation

$$\begin{aligned} nats(x) &\to x : nats(s(x)) \\ 2nd(x : (y : rest)) &\to y \end{aligned}$$

$\delta = \{\langle x : (y : zs), 2.2, h \rangle\}$, then

$$x : y : nats(z) \not\to_\delta x : y : z : nats(s(z))$$

Another example

Take-app

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Another example

Take-app

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$$\langle x : nats(y), 2, h \rangle$$

Another example

Take-app

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$$\langle x : nats(y), 2, h \rangle$$

$$app(nats(0), 0) \to app(0 : nats(1), 0) \to$$
$$0 : app(nats(1), 0) \to \ldots$$

Another example

Take-app

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$\langle x : app(nats(y), zs), 2.1, h\rangle$ $\quad$ $\langle x : nats(y), 2, h\rangle$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$

$$\begin{aligned}
app(app(nats(0), 0), 0) &\to app(app(0 : nats(1), 0), 0) \to \\
app(0 : app(nats(1), 0), 0) &\to 0 : app(app(nats(1), 0), 0) \to \\
0 : app(app(1 : nats(2), 0), 0) &\to 0 : app(1 : app(nats(2), 0), 0) \to \\
0 : 1 : app(app(nats(2), 0), 0) &\to \ldots
\end{aligned}$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

If one uses the forbidden patterns

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

the induced forbidden pattern rewrite relation has the following properties:

- Termination
- "Full" completeness, i.e. every normal form can be computed.

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(\mathit{nats}(y), zs), 2.1, h\rangle \qquad \langle x : \mathit{nats}(y), 2, h\rangle$$
$$\langle x : \mathit{app}(y : app(z : zs), us), 2, h\rangle$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

$$app(app(nats(0), 0), 0) \quad \to$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

$$app(app(nats(0), 0), 0) \to$$
$$app(app(0 : nats(1), 0), 0) \to$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(\mathit{nats}(y), zs), 2.1, h\rangle \qquad \langle x : \mathit{nats}(y), 2, h\rangle$$
$$\langle x : \mathit{app}(y : app(z : zs), us), 2, h\rangle$$

$$app(app(\mathit{nats}(0), 0), 0) \to$$
$$app(\mathit{app}(0 : \mathit{nats}(1), 0), 0) \to$$
$$\mathit{app}(0 : app(\mathit{nats}(1), 0), 0) \to$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

$$\begin{aligned} app(app(nats(0), 0), 0) &\to \\ app(app(0 : nats(1), 0), 0) &\to \\ app(0 : app(nats(1), 0), 0) &\to \\ 0 : app(app(nats(1), 0), 0) &\to \end{aligned}$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

$$\begin{aligned} app(app(nats(0), 0), 0) &\to \\ app(app(0 : nats(1), 0), 0) &\to \\ app(0 : app(nats(1), 0), 0) &\to \\ 0 : app(app(nats(1), 0), 0) &\to \\ 0 : app(app(1 : nats(2), 0), 0) &\to \end{aligned}$$

Another example

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

$$\begin{aligned}
app(app(nats(0), 0), 0) &\to \\
app(app(0 : nats(1), 0), 0) &\to \\
app(0 : app(nats(1), 0), 0) &\to \\
0 : app(app(nats(1), 0), 0) &\to \\
0 : app(app(1 : nats(2), 0), 0) &\to \\
0 : app(1 : app(nats(2), 0), 0) &\to
\end{aligned}$$

Comparing rewriting with forbidden patterns to related approaches

Forbidden patterns and related notions

- Context-sensitive rewriting is a special case of rewriting with forbidden patterns where δ is of the form $\{\langle t_i, q_i, h\rangle, \langle t_i, q_i, b\rangle \mid 1 \leq i \leq n\}$.

Comparing rewriting with forbidden patterns to related approaches

Forbidden patterns and related notions

- Context-sensitive rewriting is a special case of rewriting with forbidden patterns where δ is of the form $\{\langle t_i, q_i, h\rangle, \langle t_i, q_i, b\rangle \mid 1 \leq i \leq n\}$.
- For a given rewrite system $\mathcal{R}$ the *innermost* rewrite relation $\rightarrow^i_{\mathcal{R}}$ coincides with $\rightarrow_\delta$ if δ is given by

$$\{\langle l, \epsilon, a\rangle \mid l \rightarrow r \in \mathcal{R}\}$$

Comparing rewriting with forbidden patterns to related approaches

Forbidden patterns and related notions

- Context-sensitive rewriting is a special case of rewriting with forbidden patterns where δ is of the form $\{\langle t_i, q_i, h\rangle, \langle t_i, q_i, b\rangle \mid 1 \leq i \leq n\}$.
- For a given rewrite system $\mathcal{R}$ the *innermost* rewrite relation $\to^i_{\mathcal{R}}$ coincides with $\to_\delta$ if δ is given by
$$\{\langle l, \epsilon, a\rangle \mid l \to r \in \mathcal{R}\}$$
- For a given rewrite system $\mathcal{R}$ the *outermost* rewrite relation $\to^o_{\mathcal{R}}$ coincides with $\to_\delta$ if δ is given by
$$\{\langle l, \epsilon, b\rangle \mid l \to r \in \mathcal{R}\}$$

Canonical rewriting with forbidden patterns

Simple Patterns

A pattern $\langle t, q, \lambda \rangle$ is called *simple* if

- t is linear,
- $t|_p$ is a variable or a maximal non-variable position, and
- $t|_q$ is a variable whenever $p \parallel q$.

Simple and non-simple patterns

- $\langle x : (y : \mathit{nats}(z)), 2, h \rangle$ is a simple pattern.
- $\langle x : \mathit{app}(y : \mathit{app}(z : zs), us), 2, h \rangle$ is not simple.

Canonical rewriting with forbidden patterns

Canonical forbidden patterns

Given a TRS $\mathcal{R} = (\Sigma, R)$ and a set of forbidden patterns δ_Σ, δ_Σ is *canonical* if it is simple, it does not contain patterns of the form $(\_, \_, a)$ or $(\_, \epsilon, h)$ and

- no subterm of a pattern term (where the subterm at the forbidden position is replaced by a fresh variable) unifies with the left-hand side l of a rule from $\mathcal{R}$ such that a non-variable subterm of l is forbidden for reduction.
- no subterm of the left-hand side l of a rule unifies with any pattern term (where the subterm at the forbidden position is replaced by a fresh variable) such that a non-variable position of l is forbidden for reduction.

Canonical rewriting with forbidden patterns

Canonicity and normal forms

Let $\mathcal{R}$ be a *left-linear* TRS and let δ_Σ be canonical, then $\to_{\delta_\Sigma}$ normal forms are $\to_{\mathcal{R}}$ head-normal forms.

Example with canonical forbidden patterns

For the TRS

$$\begin{aligned} nats(x) &\to x : nats(s(x)) \\ 2nd(x : (y : rest)) &\to y \end{aligned}$$

$\delta = \{\langle x : (y : zs), 2.2, h\rangle\}$ is canonical.

Completeness of Take-Append

Take-Append

$$app(x : xs, ys) \to x : app(xs, ys) \qquad app(nil, ys) \to ys$$
$$take(s(x), y : ys) \to take(x, ys) \qquad take(0, y : ys) \to y$$
$$nats(x) \to x : nats(s(x))$$

Forbidden patterns:

$$\langle x : app(nats(y), zs), 2.1, h\rangle \qquad \langle x : nats(y), 2, h\rangle$$
$$\langle x : app(y : app(z : zs), us), 2, h\rangle$$

Completeness of Take-Append

- No reduction necessary for a subsequent more outer reduction can be forbidden by one of the first two patterns and no instance of $nats(x)$ is normalizing.
- Whenever the third pattern matches a subterm, then there is also an allowed redex in the term.

Termination of rewriting with forbidden patterns

Termination for "here" forbidden patterns

- In contrast to context-sensitive rewriting, "here"-forbidden patterns impose *local* restrictions.
- Non-local replacement restrictions sometimes make termination analysis hard for context-sensitive rewrite systems (cf. e.g. unhiding contexts).
- Considering only local context-restrictions might be advantageous.

A naive approach

Idea: Identify a (finite) set of *allowed contexts* complementing the forbidden contexts given by the forbidden patterns.

Termination by Transformation Example

Example

$$\begin{aligned} nats(x) &\rightarrow x : nats(s(x)) \\ 2nd(x : y : rest) &\rightarrow y \end{aligned}$$

with a forbidden pattern $\langle x : y : zs, 2.2 \rangle$ would be transformed into

$$\begin{aligned} 2nd(x : (y : zs)) &\rightarrow y \\ 2nd(nats(x)) &\rightarrow 2nd(x : nats(s(x))) \\ s(nats(x)) &\rightarrow s(x : nats(s(x))) \\ nats(nats(x)) &\rightarrow nats(x : nats(s(x))) \\ Top(nats(x)) &\rightarrow Top(x : nats(s(x))) \\ nats(x) : y &\rightarrow (x : nats(s(x))) : y \end{aligned}$$

Termination by Transformation Example

Example

$$\begin{aligned} nats(x) &\rightarrow x : nats(s(x)) \\ 2nd(x : y : rest) &\rightarrow y \end{aligned}$$

with a forbidden pattern $\langle x : y : zs, 2.2 \rangle$ would be transformed into

$$\begin{aligned} (x' : nats(x)) : y &\rightarrow (x' : (x : nats(s(x)))) : y \\ 2nd(x' : nats(x)) &\rightarrow 2nd(x' : x : nats(s(x))) \\ s(x' : nats(x)) &\rightarrow s(x' : (x : nats(s(x)))) \\ nats(x' : nats(x)) &\rightarrow nats(x' : (x : nats(s(x)))) \\ Top(x' : nats(x)) &\rightarrow Top(x' : (x : nats(s(x)))) \end{aligned}$$

Termination by contextual dependency pairs

Contextual dependency pairs

Let $\mathcal{R} = (\mathcal{F} = \mathcal{C} \uplus \mathcal{D}, R)$ be a TRS.
$CDP(\mathcal{R}) = DP_c(\mathcal{R}) \uplus V_c(\mathcal{R}) \uplus A_c(\mathcal{R}) \uplus S_c(\mathcal{R})$, where

$$
\begin{aligned}
DP_c(\mathcal{R}) &= \{l^\# \to r|_p^\# \ [c] \mid l \to r \in R, p \in Pos_{\mathcal{D}}(r), c = r[\Box]_p\} \\
V_c(\mathcal{R}) &= \{l^\# \to T(r|_p) \ [c] \mid l \to r \in R, r|_p = x \in Var, c = r[\Box]_p\} \\
S_c(\mathcal{R}) &= \{T(f(\vec{x})) \to T(x_i) \ [f(\vec{x})[\Box]_i] \mid \\
&\qquad \vec{x} = x_1, \ldots, x_{ar(f)}, l \to r \in R, root(r|_p) = f, i \in \{1, \ldots, ar(f)\}\} \\
A_c(\mathcal{R}) &= \{T(f(x_1, \ldots, x_{ar(f)})) \to f^\#(x_1, \ldots, x_{ar(f)}) \ [\Box] \mid \\
&\qquad l \to r \in R, root(r|_p) = f \in \mathcal{D}\}.
\end{aligned}
$$

Termination by contextual dependency pairs

Example 1

The TRS $\mathcal{R}$ given by

$$a \to f(a) \qquad f(x) \to b$$

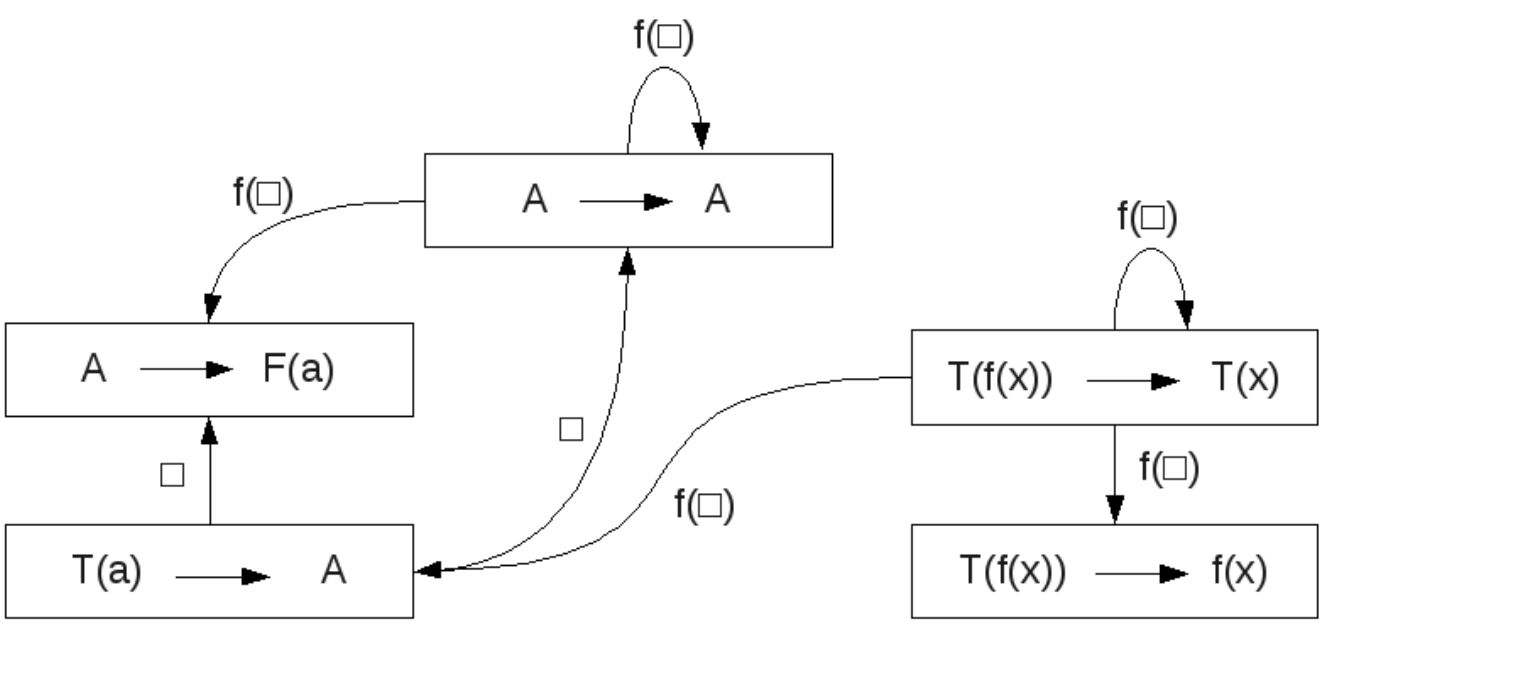
$CDP(\mathcal{R})$ is

$$\begin{array}{rcll} A & \to & A & [f(\Box)] \\ T(f(x)) & \to & T(x) & [f(\Box)] \\ T(a) & \to & A & [\Box] \end{array} \qquad \begin{array}{rcll} A & \to & F(a) & [\Box] \\ T(f(x)) & \to & F(x) & [\Box] \end{array}$$

Potential (not necessarily outermost) DP chains might be

$$\begin{array}{ccccccc} A & \overset{f(\Box)}{\to} & A & \overset{f(\Box)}{\to} & A & \overset{f(\Box)}{\to} & \dots \\ T(f(a)) & \overset{f(\Box)}{\to} & T(a) & \overset{\geq\epsilon}{\to} & T(f(a)) & \overset{f(\Box)}{\to} & \dots \end{array}$$

Example



- Cycles in the graph use edges having $f(\square)$ contexts.

Summary and Outlook

Summary and Outlook

- Context-sensitivity and rewriting under position based strategies are well-analyzed and understood but they turn out to be not expressive enough in certain cases.
- More fine-grained restrictions might lead to better results in this area, although their analysis is harder.
- Identifying feasible classes of patterns and finding acceptable tradeoffs between expressivity and simplicity is crucial.
- Automated generation of appropriate patterns is desirable.