Higher-order Rewriting

Suzan Erven

s4534999

Radboud University Nijmegen

MFoCS Seminar
January 24, 2022

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

- Extends first-order rewrite systems to higher-order (HRS)

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

- Extends first-order rewrite systems to higher-order (HRS)
- Critical pairs & confluence

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

- Extends first-order rewrite systems to higher-order (HRS)
- Critical pairs & confluence

Outermost-Fair Rewriting ('97)
Femke van Raamsdonk

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

- Extends first-order rewrite systems to higher-order (HRS)
- Critical pairs & confluence

Outermost-Fair Rewriting ('97)
Femke van Raamsdonk

- Outermost-fair rewriting is normalising for certain first-order systems

Introduction

Higher-order critical pairs ('91)
Tobias Nipkow

- Extends first-order rewrite systems to higher-order (HRS)
- Critical pairs & confluence

Outermost-Fair Rewriting ('97)
Femke van Raamsdonk

- Outermost-fair rewriting is normalising for certain first-order systems
- Investigates how this can work on an HRS

Outline

Outline

First-order rewriting recap

Extension to higher-order

Confluence

Normalisation

First-order rewriting recap

Example: addition on natural numbers.

$$\begin{aligned} (1): \quad & +(x,0) \rightarrow x \\ (2): \quad & +(x,s(y)) \rightarrow s(+(x,y)) \end{aligned}$$

where $+, s, 0$ functions with arity $1, 2, 0$ respectively, and x, y variables.

First-order rewriting recap

Example: addition on natural numbers.

$$
\begin{aligned}
(1):\quad & +(x,0) \rightarrow x \\
(2):\quad & +(x,s(y)) \rightarrow s(+(x,y))
\end{aligned}
$$

where $+, s, 0$ functions with arity $1, 2, 0$ respectively, and x, y variables.

Rewriting using these rules allows us to prove statements like $1 + 2 = 3$.

Some properties

- (Local) confluence
- (Strong) normalisation
- Termination (= strong normalisation) is undecidable
- If terminating, confluence is decidable

Higher-order?

- First-order accepts only inputs of base type!
- How do we make an extension that makes sense?

Outline

Higher-Order Rewrite System (HRS)

A *rewrite rule* is $l \to r$ such that

- l is a pattern but not η-equivalent to a free variable
- l and r are of the same type
- and all free variables in r also occur in l

An *HRS* is a finite set of rewrite rules.

Patterns (1/2)

Left-hand side of a rule should be a *pattern*

Patterns (1/2)

Left-hand side of a rule should be a *pattern*

Definition: Pattern

A term t in β-normal form is called a *(higher-order) pattern* if every free occurrence of a variable F is in a subterm $F(\overline{u_n})$ of t such that $\overline{u_n}$ is η-equivalent to a list of distinct bound variables.

Patterns (1/2)

Left-hand side of a rule should be a *pattern*

Definition: Pattern

A term t in β-normal form is called a *(higher-order) pattern* if every free occurrence of a variable F is in a subterm $F(\overline{u_n})$ of t such that $\overline{u_n}$ is η-equivalent to a list of distinct bound variables.

Examples

Some patterns are $\lambda x.c(x)$, X, $\lambda x.F(\lambda z.x(z))$, $\lambda x, y.F(y, x)$.
Some non-patterns are $F(c)$, $\lambda x.F(x, x)$, $\lambda x.F(F(x))$.

Patterns (1/2)

Left-hand side of a rule should be a *pattern*

Definition: Pattern

A term t in β-normal form is called a *(higher-order) pattern* if every free occurrence of a variable F is in a subterm $F(\overline{u_n})$ of t such that $\overline{u_n}$ is η-equivalent to a list of distinct bound variables.

Examples

Some patterns are $\lambda x.c(x),\ X,\ \lambda x.F(\lambda z.x(z)),\ \lambda x,y.F(y,x)$.
Some non-patterns are $F(c),\ \lambda x.F(x,x),\ \lambda x.F(F(x))$.

But why?

Patterns (2/2)

Theorem

It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.

Patterns (2/2)

Theorem

It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.

- Rewriting is computable

Patterns (2/2)

Theorem

It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.

- Rewriting is computable
- Critical pairs are computable

Also

- Restriction to patterns ensures no free variables are spawned during rewriting

Patterns (2/2)

Theorem

It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.

- Rewriting is computable
- Critical pairs are computable

Also

- Restriction to patterns ensures no free variables are spawned during rewriting

Consider the rule

$$f(c(F(X), F(a))) \to f(X).$$

(note: lhs not a pattern). Rewriting the term $f(c(a, a))$ with this rule to $f(X)$ spawns a new variable.

Outline

First-order rewriting recap

Extension to higher-order

Confluence

Normalisation

Critical pairs in first-order

Consider (first-order) rules

$$
\begin{array}{lrcl}
(1): & (x \times y) \times z & \to & x \times (y \times z) \\
(2): & i(x \times y) & \to & i(y) \times i(x)
\end{array}
$$

Critical pairs in first-order

Consider (first-order) rules

$$
\begin{array}{ll}
(1): & (x \times y) \times z \;\rightarrow\; x \times (y \times z) \\
(2): & i(x \times y) \;\rightarrow\; i(y) \times i(x)
\end{array}
$$

Unifying gives terms $((x \times y) \times z) \times v$ and $i((x \times y) \times z)$.

Critical pairs in first-order

Consider (first-order) rules

$$
\begin{aligned}
(1):& \quad (x \times y) \times z \;\to\; x \times (y \times z) \\
(2):& \quad i(x \times y) \;\to\; i(y) \times i(x)
\end{aligned}
$$

Unifying gives terms $((x \times y) \times z) \times v$ and $i((x \times y) \times z)$.

Reducing gives critical pairs, both of which converge. So, the system is (at least) locally confluent.

Critical pairs in higher-order

- Idea is the same

Critical pairs in higher-order

- Idea is the same
- Problem: taking a subterm can free bound variables

Critical pairs in higher-order

- Idea is the same
- Problem: taking a subterm can free bound variables
- Solution: Remember which were freed, and bind again before determining the mgu

Critical pairs in higher-order

- Idea is the same
- Problem: taking a subterm can free bound variables
- Solution: Remember which were freed, and bind again before determining the mgu

Critical Pair Lemma

An HRS R where all rules are of base type is *locally confluent* if and only if for each critical pair $u_1 = u_2$ in R, u_1 and u_2 have a common reduct.

(And R terminating $\rightarrow$ decision procedure for confluence)

Outline

First-order rewriting recap

Extension to higher-order

Confluence

Normalisation

Normalisation

- Termination undecidable

Normalisation

- Termination undecidable
- Strategy

Normalisation

- Termination undecidable
- Strategy ($\neq$ always succeed)

Normalisation

- Termination undecidable
- Strategy ($\neq$ always succeed) is normalising if it yields a normal form for any term that has one

Normalisation

- Termination undecidable
- Strategy ($\neq$ always succeed) is normalising if it yields a normal form for any term that has one
- Steal from first-order (...again)

Normalisation

- Termination undecidable
- Strategy ($\neq$ always succeed) is normalising if it yields a normal form for any term that has one
- Steal from first-order (...again)

Definition: Outermost-fair rewriting

A rewrite sequence is *outermost-fair* if every outermost redex is eventually eliminated. I.e. if either it ends in nf, or it's impossible to trace infinitely long an outermost redex occurrence.

Outermost reduction

Consider HRS

$$
\begin{array}{lrcl}
(1): & fax & \to & x.a \\
(2): & fxa & \to & x.a \\
(3): & gx & \to & hxx
\end{array}
$$

Outermost reduction

Consider HRS

$$\begin{array}{lrcl} (1): & fax & \rightarrow & x.a \\ (2): & fxa & \rightarrow & x.a \\ (3): & gx & \rightarrow & hxx \end{array}$$

Term $g(faa)$ has one outermost redex occurrence, which we apply (3) on.

Outermost reduction

Consider HRS

$$
\begin{array}{lrcl}
(1): & fax & \to & x.a \\
(2): & fxa & \to & x.a \\
(3): & gx & \to & hxx
\end{array}
$$

Term $g(faa)$ has one outermost redex occurrence, which we apply (3) on.
Term faa has two: apply either (1) or (2).

Infinite outermost chain

Let $s : s_0 \to s_1 \to s_2 \to \dots$ be an infinite rewrite sequence. An *infinite outermost chain* in s is an infinite sequence of redex occurrences $w_m, m_{m+1}, \dots$ such that

Infinite outermost chain

Let $s : s_0 \to s_1 \to s_2 \to \dots$ be an infinite rewrite sequence. An *infinite outermost chain* in s is an infinite sequence of redex occurrences $w_m, m_{m+1}, \dots$ such that

1. w_p is an outermost redex occurrence in s_p for every $p \geq m$

Infinite outermost chain

Let $s : s_0 \to s_1 \to s_2 \to \dots$ be an infinite rewrite sequence. An *infinite outermost chain* in s is an infinite sequence of redex occurrences $w_m, m_{m+1}, \dots$ such that

1. w_p is an outermost redex occurrence in s_p for every $p \geq m$
2. w_p is a *residual* of w_{p-1} for every $p > m$

Requirements: Almost orthogonal

- Left-linear: In each rule, each bounded variable occurs at most once in the lhs
- Orthogonal: Left-linear and no critical pairs
- Weakly orthogonal: Left-linear and only trivial critical pairs
- Almost orthogonal: Weakly orthogonal + redex occurrence overlaps only at root of redex occurrences

Requirements: Fully extended

A rewrite rule is *fully extended* if every occurrence of a bound variable has (the η-normal form of) every bound variable that it is in the scope of as an argument.
An HRS is fully extended if all of its rules are.

Normalisation strategy

If a system is

1. Almost orthogonal and
2. Fully extended

then the following holds

Theorem

Let s_0 be a weakly normalising term. Every outermost-fair rewrite sequence starting in s_0 eventually ends in a normal form.

Wrap-up

We saw

- What patterns are, and why we want them
- How to use them to extend rewriting systems to higher-order
- How critical pairs work in an HRS
- The consequences w.r.t. confluence
- A normalisation strategy for certain HRSs
- Briefly, what the requirements on an HRS are for the strategy to be reliable

Thank you!