Difference of Constrained Patterns in Logically Constrained Term Rewrite Systems

Naoki Nishida Misaki Kojima Yuto Nakamura

Nagoya University

FroCoS 2025, Reykjavik, Iceland, September 30, 2025

Contents of This Talk

Pattern Completeness

- Non-existence of undefined patterns
  - pattern: $f(t_1, \ldots, t_n)$ with a defined symbol f and constructor terms $t_1, \ldots, t_n$
- Usually checked by compilers/interpreters of programming languages
  - Guards are not taken into account, while warnings may occur
- Equivalent to quasi-reducibility of many-sorted term rewrite systems (TRS)
  - TRS $\mathcal{R}$ is quasi-reducible if all ground patterns are redexes of $\mathcal{R}$
- Usually assumed in using Rewriting Induction [Reddy, 1990]
  - Also used in proving ground confluence via RI [Aoto et al., 2017]

Quasi-reducibility of Rewrite Systems

- Non-existence of undefined patterns $f(t_1, \ldots, t_n)$

Example (list of natural numbers)

- $\mathcal{S} = \{\, nat,\ list,\ bool \,\}$
- $\Sigma = \{\, \mathsf{nil} : list,\ \mathsf{cons} : nat \times list \Rightarrow list,\ 0 : nat,\ \mathsf{s} : nat \Rightarrow nat,\ \mathsf{true}, \mathsf{false} : bool,\ \mathsf{even} : list \Rightarrow bool \,\}$
  - $\mathcal{D} = \{\, \mathsf{even} \,\}$: defined symbols $\quad \mathcal{C} = \{\, 0,\ \mathsf{s},\ \mathsf{nil},\ \mathsf{cons},\ \mathsf{true},\ \mathsf{false} \,\}$: constructors
- $\mathcal{R} = \left\{ \begin{array}{r} \mathsf{even}(\mathsf{nil}) \to \mathsf{true} \\ \mathsf{even}(\mathsf{cons}(x, \mathsf{cons}(y, zs))) \to \mathsf{even}(zs) \end{array} \right\}$ is not quasi-reducible

- Decidable for TRSs [Kapur et al., 1987]
- Complement algorithm for left-linear TRSs [Lazrek et al., 1990, Higashiwada and Aoto, 2019]
- Well-designed formalized algorithm in co-NP for TRSs [Thiemann and Yamada, 2024, Thiemann and Yamada, 2025]
- No result for decidability of quasi-reducibility of constrained systems
  - Some sufficient conditions for Logically Constrained TRSs [Sakata et al., 2009, Kop, 2017]

Logically Constrained Term Rewrite System (LCTRS) [Kop and Nishida, 2013]

- Computation models of functional and imperative programs [Fuhs et al., 2017]
- Calculation rules are implicitly included, e.g., $x + y \to z\ [\, z = x + y \,]$

Example (LCTRS with Integer Theory)

- $\mathcal{S}_{theory} = \{\, bool,\ int \,\}$: theory sorts
- $\mathcal{V}al = \{\, \mathsf{true}, \mathsf{false} : bool \,\} \cup \{\, \mathsf{n} : int \mid n \in \mathbb{Z} \,\}$: values
- $\Sigma_{theory} = \mathcal{V}al \cup \left\{ \begin{array}{l} +, -, \times, /, \ldots : int \times int \Rightarrow int, \\ =_{int}, \neq_{int}, <, \leq, \ldots : int \times int \Rightarrow bool, \\ \wedge, \vee, \ldots : bool \times bool \Rightarrow bool,\ \neg : bool \Rightarrow bool \end{array} \right\}$: theory symbols
- $\Sigma_{terms} = \{\, \mathsf{sum} : int \Rightarrow int \,\}$: user-defined symbols
- $\mathcal{R} = \left\{ \begin{array}{ll} \mathsf{sum}(n) \to n & [\, n \leq 0 \,] \\ \mathsf{sum}(n) \to n + \mathsf{sum}(n+1) & [\, n > 0 \,] \end{array} \right\}$: user-defined rules
- $\mathsf{sum}(3) \to_{\mathcal{R}} 3 + \mathsf{sum}(3 + (-1)) \to_{\mathcal{R}} 3 + \mathsf{sum}(2) \to_{\mathcal{R}} 3 + (2 + \mathsf{sum}(2 + (-1))) \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} 6$

Complement Algorithm for Linear Patterns

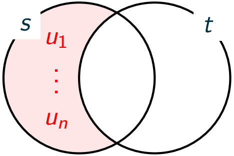
[Lazrek et al., 1990, Higashiwada and Aoto, 2019]

- Based on difference operator $\ominus$ over linear patterns

  $s \ominus t = \{u_1, \ldots, u_n\}$: finite set of linear patterns

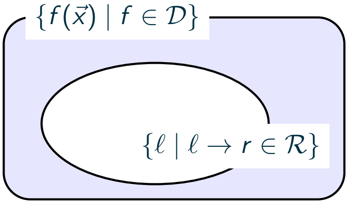
  s.t. $\mathcal{G}(s) \setminus \mathcal{G}(t) = \bigcup_{i=1}^{n} \mathcal{G}(u_i)$

  - $\mathcal{G}(s)$ denotes the set of ground constructor instances



- $\ominus$ is extended to finite sets: $\{s_1, \ldots, s_i\} \oslash \{t_1, \ldots, t_j\} = \{u_1, \ldots, u_k\}$

- $\mathcal{R}$ is quasi-reducible iff $\{f(\vec{x}) \mid f \in \mathcal{D}\} \oslash \{\ell \mid \ell \to r \in \mathcal{R}\} = \emptyset$



Applications to LCTRSs

- Equivalence verification via RI for LCTRSs [Fuhs et al., 2017]
  - Termination and quasi-reducibility of given LCTRSs are assumed
- Proof system for All-Path Reachability (APR) problems $P \Rightarrow^{\forall} Q$ [Ciobâcă and Lucanu, 2018]
  - Difference of constrained terms is computed: Some rule reduces $P \Rightarrow^{\forall} Q$ to $(P \setminus Q) \Rightarrow^{\forall} Q$

Example

- $\mathcal{S} = \{\, \mathit{bool},\ \mathit{int},\ \mathit{list} \,\}$
- $\mathcal{C} = \mathcal{V}\mathit{al} \cup \{\, \mathsf{nil} : \mathit{list},\ \mathsf{cons} : \mathit{int} \times \mathit{list} \Rightarrow \mathit{list} \,\}$
- Is $\left\{ \begin{array}{lrl} (1) & \mathsf{f}(\mathsf{nil}, y_1) \to 0 & [\, y_1 \leq 0 \,] \\ (2) & \mathsf{f}(\mathsf{cons}(x_2, xs_2), y_2) \to \mathsf{f}(xs_2, y_2 - 1) & [\, x_2 \leq 0 \wedge y_2 > 0 \,] \\ (3) & \mathsf{f}(\mathsf{cons}(x_3, \mathsf{cons}(z_3, zs_3)), y_3) \to x_3 + \mathsf{f}(zs_3, y_3 - 2) & [\, x_3 > 0 \wedge y_3 > 1 \,] \end{array} \right\}$ quasi-reducible?

$$\{\, \mathsf{f}(xs, y)\,[\mathsf{true}] \,\} \oslash \left\{ \begin{array}{lrl} (1) & \mathsf{f}(\mathsf{nil}, y_1) & [\, y_1 \leq 0 \,] \\ (2) & \mathsf{f}(\mathsf{cons}(x_2, xs_2), y_2) & [\, x_2 \leq 0 \wedge y_2 > 0 \,] \\ (3) & \mathsf{f}(\mathsf{cons}(x_3, \mathsf{cons}(z_3, zs_3)), y_3) & [\, x_3 > 0 \wedge y_3 > 1 \,] \end{array} \right\} = \emptyset\ ?$$

- Can we decide it?

Goal and Contributions

Goal

Difference operator and Complement Algorithm for Logically Constrained TRSs

Contributions

- $\ominus$ over constrained patterns and constrained linear patterns
  - LHSs of $\ominus$ do not have to be linear, while RHSs are linear
- Complement Algorithm for finite sets of constrained linear patterns
- Quasi-reducibility of left-linear LCTRSs with decidable theories is decidable

LCTRSs in This Talk

- No non-value ground constructor term with a theory sort
  - Example: Declaration of $\mathsf{s} : \mathit{int} \Rightarrow \mathit{int}$ is not allowed for integer LCTRSs
  - All theory sorts are inextensible [Fuhs et al., 2025]
- Finitely many non-theory symbols

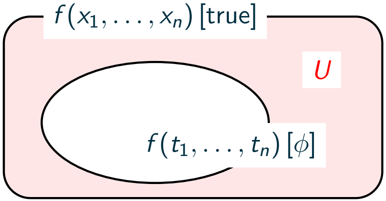
- In practical terms, these are not limitations

Contents of This Talk

Constrained Patterns and Complements

- Constrained pattern $t\,[\phi]$ is a pair of pattern t and constraint ϕ
  - Pattern is a term $f(t_1, \ldots, t_n)$ s.t. $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in T(\mathcal{C}, \mathcal{V})$
  - $\mathcal{G}(t) := \{\, t\sigma \mid \sigma$ is a ground constructor substitution $\}$ and $\mathcal{G}(U) := \bigcup_{u \in U} \mathcal{G}(u)$
- $\mathcal{G}(t\,[\phi]) := \{\, t\sigma \mid \sigma$ is a ground constructor substitution, $\forall x \in \mathcal{V}ar(\phi).\ x\sigma \in \mathcal{V}al,\ [\![\phi\sigma]\!] = \top \,\}$
- Complement of constrained pattern $f(t_1, \ldots, t_n)\,[\phi]$ is a set U of constrained patterns s.t.

$$\mathcal{G}(U) \;=\; \mathcal{G}(f(x_1, \ldots, x_n)\,[\mathsf{true}]) \setminus \mathcal{G}(f(t_1, \ldots, t_n)\,[\phi])$$



- Finite complements are expected

Complement Operator for Linear Constructor Terms

[Lazrek et al., 1990, Higashiwada and Aoto, 2019]

Definition ($\overline{\cdot}$ for linear constructor terms)

- $\overline{x : \iota} = \emptyset$
- $\overline{c(u_1, \ldots, u_n) : \iota} = \{d(y_1, \ldots, y_m) \mid d : \iota_1 \times \cdots \times \iota_m \Rightarrow \iota \in \mathcal{C},\ c \neq d\}$
  $\cup \{c(u_1, \ldots, u_{i-1}, u_i', y_{i+1}, \ldots, y_n) \mid u_i \notin \mathcal{V},\ u_i' \in \overline{u_i}\}$

- $\mathcal{C}$ is assumed to be finite for finiteness of $\overline{u}$

Example

- $\mathcal{S} = \{\, nat,\ bool,\ list,\ pair \,\}$
- $\mathcal{C} = \{\, \mathrm{nil} : list,\ \mathrm{cons} : nat \times list \Rightarrow list,\ 0 : nat,\ \mathrm{s} : nat \Rightarrow nat,\ \mathrm{true}, \mathrm{false} : bool,\ \mathrm{p} : nat \times nat \Rightarrow pair \,\}$
- $\overline{\mathrm{nil}} = \{\, \mathrm{cons}(x, xs) \,\}$
- $\overline{\mathrm{cons}(x, \mathrm{nil})} = \{\, \mathrm{nil},\ \mathrm{cons}(x, \mathrm{cons}(y, ys)) \,\}$

- Linearity is necessary for finite complements of patterns with infinite sorts
  - "Complement of $\mathrm{p}(x, x)$" $= \{\, \mathrm{p}(t_1, t_2) \in T(\mathcal{C}) \mid t_1, t_2 : nat,\ t_1 \neq t_2 \,\}$

Complements of Values in LCTRS Setting

- For finite results, complement operator $\overline{\cdot}$ assumes finiteness of $\mathcal{C}$ and linearity of terms
- $\mathcal{V}al$ ($= \Sigma_{theory} \cap \mathcal{C}$) may be infinite, e.g., $\mathcal{C}$ of integer LCTRSs includes all integers
- Complements of values may be infinite, e.g., $\overline{0} = \mathbb{Z} \setminus \{0\}$ is infinite
- Make s of $s\,[\phi]$ value-free, e.g., $s[0]_p\,[\phi]$ is equivalent to $s[x]_p\,[\phi \wedge x = 0]$
- $\mathcal{C} \setminus \mathcal{V}al$ ($\subseteq \Sigma_{terms}$) should be finite
- Logical Variables in term part can be linearized, e.g, $s[x, x]_p\,[\phi]$ is equivalent to $s[x, y]_p\,[\phi \wedge x = y]$

Proposition [Kop, 2017, Kojima and Nishida, 2024]

For any constrained term $s\,[\phi]$, there exists a value-free LV-linear $s'\,[\phi']$ s.t. $\mathcal{G}(s\,[\phi]) = \mathcal{G}(s'\,[\phi'])$

- $s\,[\phi]$ is assumed to be value-free ($s \in T(\Sigma \setminus \mathcal{V}al, \mathcal{V})$) and LV-linear (linear w.r.t. $\mathcal{V}ar(\phi)$)

LCTRSs in This Talk (repeat)

- ...
- Finitely many non-theory symbols, i.e., Σ_{terms} is finite

Contents of This Talk

1. Background
2. Complement of Patterns
3. Difference Operator over Constrained Patterns
4. Complement Algorithm for Quasi-Reducibility of LCTRSs
5. Conclusion

Difference Operator $\ominus$ over Unconstrained Linear Patterns

[Lazrek et al., 1990, Higashiwada and Aoto, 2019]

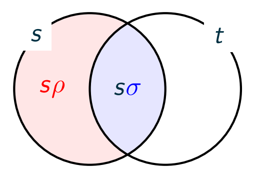
- Assume w.l.o.g. that s, t of $s \ominus t$ have no shared variables: $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \emptyset$

Definition

$$s \ominus t \;=\; \begin{cases} \{\, s\rho \mid \rho \in \overline{\sigma|_{\mathcal{V}ar(s)}} \,\} & \text{if } s, t \text{ are unifiable with mgu } \sigma \\ \{\, s \,\} & \text{o/w} \end{cases}$$

where

$$\overline{\sigma} = \{\, \rho \mid \mathcal{D}om(\rho) = \mathcal{D}om(\sigma),\ \rho \neq \sigma,\ \forall x \in \mathcal{D}om(\sigma).\ x\rho \in \overline{x\sigma} \cup \{x\sigma\} \,\}$$



- $s \ominus t$ is a finite set of patterns s.t. $\mathcal{G}(s \ominus t) = \mathcal{G}(s) \setminus \mathcal{G}(t)$

Example (cont'd)

- $\mathsf{even}(\mathsf{cons}(x, \mathsf{nil})) \ominus \mathsf{even}(\mathsf{cons}(0, ys)) = \{\, \mathsf{even}(\mathsf{cons}(\mathsf{s}(y), \mathsf{nil})) \,\}$
  - $\sigma = \{\, x \mapsto 0,\ ys \mapsto \mathsf{nil} \,\}$ and thus $\overline{\sigma|_{\{x\}}} = \{\, x \mapsto \mathsf{s}(y) \,\}$

- Linearity of s and t ensures linearity of $x\sigma$, but s does not have to be linear [new]

Proposition If t is linear, then $x\sigma$ is linear for any $x \in \mathcal{V}ar(s)$ [new]

Difference Operator $\ominus$ over Value-free LV-linear Constrained Terms

Definition (repeat)

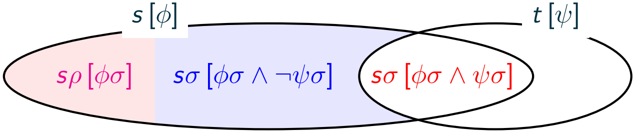
$$s \ominus t \;=\; \begin{cases} \{\, s\rho \mid \rho \in \overline{\sigma|_{\mathcal{V}ar(s)}} \,\} & \text{if } s, t \text{ are unifiable with mgu } \sigma \\ \{\, s \,\} & \text{o/w} \end{cases}$$

where $\overline{\sigma} = \{\, \rho \mid \mathcal{D}om(\rho) = \mathcal{D}om(\sigma),\ \rho \neq \sigma,\ \forall x \in \mathcal{D}om(\sigma).\ x\rho \in \overline{x\sigma} \cup \{x\sigma\} \,\}$

- $\forall x \in \mathcal{V}ar(\phi, \psi).\ x\rho = x\sigma \in \mathcal{V}$ by our 1st assumption on LCTRSs, and thus $\phi\rho = \phi\sigma$ and $\psi\rho = \psi\sigma$
- $\mathcal{G}(s\,[\phi]) = \mathcal{G}(s\rho\,[\phi\sigma]) \uplus \underline{\mathcal{G}(s\sigma\,[\phi\sigma])}$

  $\qquad\qquad\qquad\qquad\qquad\qquad ||$

  $\mathcal{G}(s\sigma\,[\phi\sigma \wedge \neg\psi\sigma]) \uplus \mathcal{G}(s\sigma\,[\phi\sigma \wedge \psi\sigma])$



Definition [new]

$$s\,[\phi] \ominus t\,[\psi] \;=\; \begin{cases} \{\, s\rho\,[\phi\sigma] \mid \rho \in \overline{\sigma|_{\mathcal{V}ar(s)}} \,\} & \text{if } s,\ t \text{ are unifiable with mgu } \sigma \\ \cup \{\, s\sigma\,[\phi\sigma \wedge \neg\psi\sigma] \mid \phi\sigma \wedge \neg\psi\sigma \text{ is SAT} \,\} & \quad \text{and } \phi\sigma \wedge \psi\sigma \text{ is SAT} \\ \{\, s\,[\phi] \,\} & \text{o/w} \end{cases}$$

Example: Difference of Constrained Patterns

Definition (repeat) [new]

$$s\,[\phi] \ominus t\,[\psi] \;=\; \begin{cases} \{\, s\rho\,[\phi\sigma] \mid \rho \in \overline{\sigma|_{\mathcal{V}ar(s)}} \,\} \\ \cup\, \{\, s\sigma\,[\phi\sigma \wedge \neg\psi\sigma] \mid \phi\sigma \wedge \neg\psi\sigma \text{ is SAT} \,\} & \text{if } s, t \text{ are unifiable with mgu } \sigma \text{ and } \phi\sigma \wedge \psi\sigma \text{ is SAT} \\ \{\, s\,[\phi] \,\} & \text{o/w} \end{cases}$$

Example (cont'd)

- $\mathsf{f}(xs, x)\,[\mathsf{true}] \ominus \mathsf{f}(\mathsf{nil}, y_1)\,[y_1 \le 0] = \left\{ \begin{array}{rl} \mathsf{f}(\mathsf{cons}(v, vs), x) & [\mathsf{true}] \\ \mathsf{f}(\mathsf{nil}, x) & [\mathsf{true} \wedge \neg(x \le 0)] \end{array} \right\}$
  - $\sigma = \{\, xs \mapsto \mathsf{nil},\; y_1 \mapsto x \,\}$
  - $\rho = \{\, xs \mapsto \mathsf{cons}(v, vs) \,\}$
- $\mathsf{f}(\mathsf{nil}, y_1)\,[y_1 \le 0] \ominus \mathsf{f}(xs, x)\,[\mathsf{true}] = \emptyset$
  - $\sigma = \{\, xs \mapsto \mathsf{nil},\; y_1 \mapsto x \,\}$
  - $\overline{\{\, y_1 \mapsto x \,\}} = \emptyset$
  - $\phi\sigma \wedge \neg\psi\sigma$ is $x \le 0 \wedge \neg\mathsf{true}$, which is UNSAT

Contents of This Talk

1. Background
2. Complement of Patterns
3. Difference Operator over Constrained Patterns
4. Complement Algorithm for Quasi-Reducibility of LCTRSs
5. Conclusion

Extension to Finite Sets of Unconstrained Linear Patterns

[Lazrek et al., 1990, Higashiwada and Aoto, 2019]

Definition

$$P \oslash Q = \begin{cases} ((P \setminus \{s\}) \cup (s \ominus t)) \oslash ((Q \setminus \{t\}) \cup (t \ominus s)) & \text{if } \exists s \in P, t \in Q.\ s \ominus t \neq \{s\} \\ P & \text{o/w} \end{cases}$$

- Both P and Q are assumed to be sets of linear patterns
  - Not all patterns have to be linear
  - Linearity of s is required for linearity of $t \ominus s$
- Patterns in P are w.l.o.g. assumed to be pairwise disjoint
  - s, t are disjoint if $\mathcal{G}(s) \cap \mathcal{G}(t) = \emptyset$ (i.e., s, t are not unifiable)
  - If s and t are unifiable with mgu σ, then we replace $\{s, t\}$ by $(s \ominus t) \uplus \{s\sigma\} \uplus (t \ominus s)$
- For extension to constrained patterns, replace patterns by constrained ones

Extension of $\oslash$ to Constrained Linear Patterns

Definition

[new]

$$P \oslash Q = \begin{cases} ((P \setminus \{s\,[\phi]\}) \cup (s\,[\phi] \ominus t\,[\psi])) \oslash ((Q \setminus \{t\,[\psi]\}) \cup (t\,[\psi] \ominus s\,[\phi])) & \\ & \text{if } \exists s\,[\phi] \in P, t\,[\psi] \in Q.\ s\,[\phi] \ominus t\,[\psi] \neq \{s\,[\phi]\} \\ P & \text{o/w} \end{cases}$$

Proposition

[new]

- All constrained patterns in $((P \setminus \{s\,[\phi]\}) \cup (s\,[\phi] \ominus t\,[\psi]))$ and $((Q \setminus \{t\}) \cup (t \ominus s))$ are linear
- $\oslash$ is terminating
- $\mathcal{G}(P \oslash Q) = \mathcal{G}(P) \setminus \mathcal{G}(Q)$

Theorem

[new]

Left-linear LCTRS $\mathcal{R}$ is quasi-reducible iff $\{f(\vec{x})\,[\mathsf{true}] \mid f \in \mathcal{D}\} \oslash \{\ell\,[\phi] \mid \ell \to r\ [\phi] \in \mathcal{R}\} = \emptyset$

Corollary

[new]

Quasi-reducibility of left-linear LCTRSs with decidable theories is decidable

Example: Quasi-Reducibility of LCTRSs

Example (cont'd)

$$\left\{\begin{array}{lrl} (1) & \mathsf{f}(\mathsf{nil}, y_1) \to 0 & [\, y_1 \le 0 \,] \\ (2) & \mathsf{f}(\mathsf{cons}(x_2, xs_2), y_2) \to \mathsf{f}(xs_2, y_2 - 1) & [\, x_2 \le 0 \land y_2 > 0 \,] \\ (3) & \mathsf{f}(\mathsf{cons}(x_3, \mathsf{cons}(z_3, zs_3)), y_3) \to x_3 + \mathsf{f}(zs_3, y_3 - 2) & [\, x_3 > 0 \land y_3 > 1 \,] \end{array}\right\} \text{ is not quasi-reducible as}$$

$$\{\, \mathsf{f}(xs, y)\, [\mathsf{true}] \,\} \oslash \left\{\begin{array}{lrl} (1) & \mathsf{f}(\mathsf{nil}, y_1) & [\, y_1 \le 0 \,] \\ (2) & \mathsf{f}(\mathsf{cons}(x_2, xs_2), y_2) & [\, x_2 \le 0 \land y_2 > 0 \,] \\ (3) & \mathsf{f}(\mathsf{cons}(x_3, \mathsf{cons}(z_3, zs_3)), y_3) & [\, x_3 > 0 \land y_3 > 1 \,] \end{array}\right\}$$

$$= \left\{\begin{array}{lrl} (4) & \mathsf{f}(\mathsf{cons}(x, xs), y_1) & [\, y_1 \le 0 \,] \\ (5) & \mathsf{f}(\mathsf{nil}, y_1) & [\, \neg(y_1 \le 0) \,] \end{array}\right\} \oslash \left\{\begin{array}{ll} (2) & \dots \\ (3) & \dots \end{array}\right\}$$

$$= \left\{\begin{array}{lrl} (6) & \mathsf{f}(\mathsf{cons}(x, xs), y_1) & [\, y_1 \le 0 \land \neg(x \le 0 \land y_1 > 0) \,] \\ (5) & & \dots \end{array}\right\} \oslash \left\{\begin{array}{lrl} (7) & \mathsf{f}(\mathsf{cons}(x_2, xs_2), y_2) & [\, \dots \,] \\ (3) & & \dots \end{array}\right\}$$

$$= \left\{\begin{array}{lrl} (8) & \mathsf{f}(\mathsf{cons}(x, \mathsf{nil}), y_1) & [\, y_1 \le 0 \land \neg(x \le 0 \land y_1 > 0) \,] \\ (9) & \mathsf{f}(\mathsf{cons}(x, \mathsf{cons}(z, zs)), y_1) & [\, y_1 \le 0 \land \neg(x \le 0 \land y_1 > 0) \land \neg(x > 0 \land y_1 > 1) \,] \\ (5) & & \dots \end{array}\right\} \oslash \left\{\begin{array}{ll} (7) & \dots \\ (3c) & \dots \end{array}\right\}$$

$$= \{\, (8),\ (9),\ (5) \,\} \neq \emptyset$$

Contents of This Talk

1. Background
2. Complement of Patterns
3. Difference Operator over Constrained Patterns
4. Complement Algorithm for Quasi-Reducibility of LCTRSs
5. Conclusion

Conclusion

Summary

- $\ominus$ over constrained patterns and constrained linear patterns
  - LHSs of $\ominus$ do not have to be linear, while RHSs are linear
- Complement Algorithm for finite sets of constrained linear patterns
- Quasi-reducibility of left-linear LCTRSs with decidable theories is decidable

Future Work

- Extension of co-NP Algorithm [Thiemann and Yamada, 2024] to LCTRSs
- Implementation

References

Aoto, T., Toyama, Y., and Kimura, Y. (2017).
Improving rewriting induction approach for proving ground confluence.
In Miller, D., editor, *Proceedings of the 2nd International Conference on Formal Structures for Computation and Deduction*, volume 84 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Ciobâcă, Ş. and Lucanu, D. (2018).
A coinductive approach to proving reachability properties in logically constrained term rewriting systems.
In Galmiche, D., Schulz, S., and Sebastiani, R., editors, *Proceedings of the 9th International Joint Conference on Automated Reasoning*, volume 10900 of *Lecture Notes in Computer Science*, pages 295–311. Springer.

Fuhs, C., Guo, L., and Kop, C. (2025).
An innermost DP framework for constrained higher-order rewriting.
In Fernández, M., editor, *Proceedings of the 10th International Conference on Formal Structures for Computation and Deduction*, volume 337 of *LIPIcs*, pages 20:1–20:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Fuhs, C., Kop, C., and Nishida, N. (2017).
Verifying procedural programs via constrained rewriting induction.
ACM Transactions on Computational Logic, 18(2):14:1–14:50.

Higashiwada, N. and Aoto, T. (2019).
Automatically proving sufficient completeness of conditional term rewriting systems.
In *Manuscript for the presentation at the 124th Workshop of IPSJ Special Interest Group on Programming*, pages 1–6. in Japanese.

References (cont.)

Kapur, D., Narendran, P., and Zhang, H. (1987).
On sufficient-completeness and related properties of term rewriting systems.
Acta Informatica, 24(4):395–415.

Kojima, M. and Nishida, N. (2024).
A sufficient condition of logically constrained term rewrite systems for decidability of all-path reachability problems with constant destinations.
Journal of Information Processing, 32:417–435.

Kop, C. (2017).
Quasi-reductivity of logically constrained term rewriting systems.
CoRR, abs/1702.02397.

Kop, C. and Nishida, N. (2013).
Term rewriting with logical constraints.
In Fontaine, P., Ringeissen, C., and Schmidt, R. A., editors, *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 343–358. Springer.

Lazrek, A., Lescanne, P., and Thiel, J.-J. (1990).
Tools for proving inductive equalities, relative completeness, and ω-completeness.
Information and Computation, 84(1):47–70.

References (cont.)

Reddy, U. S. (1990).
Term rewriting induction.
In Stickel, M. E., editor, *Proceedings of the 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 162–177. Springer.

Sakata, T., Nishida, N., Sakabe, T., Sakai, M., and Kusakari, K. (2009).
Rewriting induction for constrained term rewriting systems.
IPSJ Transactions on Programming, 2(2):80–96.
in Japanese (a translated summary is available from `https://www.trs.css.i.nagoya-u.ac.jp/crisys/`).

Thiemann, R. and Yamada, A. (2024).
A verified algorithm for deciding pattern completeness.
In Rehof, J., editor, *Proceedings of the 9th International Conference on Formal Structures for Computation and Deduction*, volume 299 of *LIPIcs*, pages 27:1–27:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Thiemann, R. and Yamada, A. (2025).
Deciding pattern completeness in non-deterministic polynomial time.
In *Proceedings of the 14th International Workshop on Confluence*, pages 31–37.