Confluence Properties on Open Terms in the First-Order Theory of Rewriting

Franziska Rapp
joint work with
Aart Middeldorp

University of Innsbruck
IWC 2016, Obergurgl, September 8
FORT

property \rightarrow \text{decision mode} \rightarrow \text{TRS}

yes | no | ?

FORT is based on tree automata techniques (Dauchet and Tison, LICS 1990)
FORT

property \rightarrow \text{synthesis mode} \rightarrow \text{TRS}

\frac{\exists s \forall t (s \rightarrow ^* t \land \neg \exists u (t \rightarrow u)) \Rightarrow \exists v (s \not\rightarrow \parallel v \lor v \in \not\rightarrow t)}{\text{FORT}}

\text{FORT is based on tree automata techniques (Dauchet and Tison, LICS 1990)}
∀ \(s \in \mathcal{T}\) \(s \to^* t \land \neg \exists u \ (t \to u) \implies \exists v \ (s \not\to v \lor v \not\to^\epsilon t)\)
FORT is based on tree automata techniques (Dauchet and Tison, LICS 1990)
Outline

- First-Order Theory of Rewriting
- Automation
- Properties on Open Terms
- Experiments
- Future Work
First-Order Theory of Rewriting

- first-order logic over a language $\mathcal{L}$ without function symbols
First-Order Theory of Rewriting

- first-order logic over a language $\mathcal{L}$ without function symbols
- $\mathcal{L}$ contains the following binary predicate symbols:

\[
\rightarrow \quad \rightarrow^+ \quad \rightarrow^* \quad \rightarrow! \quad \equiv \quad \epsilon \quad >\epsilon \quad \leftrightarrow \quad \leftrightarrow^* \quad \downarrow \quad =
\]
First-Order Theory of Rewriting

- first-order logic over a language $\mathcal{L}$ without function symbols
- $\mathcal{L}$ contains the following binary predicate symbols:
  \[
  \rightarrow, \rightarrow^+, \rightarrow^*, \rightarrow!, \rightarrow\ll, \epsilon\rightarrow, \epsilon\rightarrow^+, \leftrightarrow, \leftrightarrow^*, \downarrow, =
  \]
- models of $\mathcal{L}$ are finite left-linear right-ground TRSs
First-Order Theory of Rewriting

- first-order logic over a language $\mathcal{L}$ without function symbols
- $\mathcal{L}$ contains the following binary predicate symbols:

  \[ \rightarrow, \rightarrow^+, \rightarrow^*, \rightarrow!, \rightarrow\rightarrow, \epsilon\rightarrow, \geq\epsilon\rightarrow, \leftrightarrow, \leftrightarrow^*, \downarrow, = \]

- models of $\mathcal{L}$ are finite left-linear right-ground TRSs
- set of ground terms serves as domain for variables in formulas over $\mathcal{L}$
First-Order Theory of Rewriting

- first-order logic over a language $\mathcal{L}$ without function symbols
- $\mathcal{L}$ contains the following binary predicate symbols:
  \[
  \rightarrow, \rightarrow^+, \rightarrow^*, \rightarrow^! \quad \# \quad \epsilon \quad >\epsilon \quad \leftrightarrow \quad \leftrightarrow^* \quad \downarrow \quad =
  \]
- models of $\mathcal{L}$ are finite left-linear right-ground TRSs
- set of ground terms serves as domain for variables in formulas over $\mathcal{L}$
Derived Predicates

\[ s \rightarrow^* t \iff s \rightarrow^+ t \lor s = t \]
\[ s \leftrightarrow t \iff s \rightarrow t \lor t \rightarrow s \]
\[ s \rightarrow! t \iff s \rightarrow^* t \land \neg \exists u \ (t \rightarrow u) \]
\[ s \downarrow t \iff \exists u \ (s \rightarrow^* u \land t \rightarrow^* u) \]
Derived Predicates

\[
\begin{align*}
 s \rightarrow^* t & \iff s \rightarrow^+ t \lor s = t & s \leftrightarrow t & \iff s \rightarrow t \lor t \rightarrow s \\
 s \rightarrow^1 t & \iff s \rightarrow^* t \land \neg \exists u (t \rightarrow u) & s \downarrow t & \iff \exists u (s \rightarrow u \land t \rightarrow^* u)
\end{align*}
\]

\[\text{CR}(t) \iff \forall u \forall v (t \rightarrow^* u \land t \rightarrow v \implies u \downarrow v)\]
Derived Predicates

\[ s \rightarrow^* t \iff s \rightarrow^+ t \lor s = t \]
\[ s \leftrightarrow t \iff s \rightarrow t \lor t \rightarrow s \]
\[ s \rightarrow! t \iff s \rightarrow^* t \land \neg \exists u (t \rightarrow u) \]
\[ s \downarrow t \iff \exists u (s \rightarrow^* u \land t \rightarrow^* u) \]
\[ \text{CR}(t) \iff \forall u \forall v (t \rightarrow^* u \land t \rightarrow v \implies u \downarrow v) \]
\[ \text{CR} \iff \forall t \text{ CR}(t) \]
### Derived Predicates

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s \rightarrow^* t$</td>
<td>$s \rightarrow^+ t \lor s = t$</td>
</tr>
<tr>
<td>$s \rightarrow^! t$</td>
<td>$s \rightarrow^* t \land \neg \exists u \ (t \rightarrow u)$</td>
</tr>
<tr>
<td>CR($t$)</td>
<td>$\forall u \forall v \ (t \rightarrow^* u \land t \rightarrow v \implies u \downarrow v)$</td>
</tr>
<tr>
<td>WCR($t$)</td>
<td>$\forall u \forall v \ (t \rightarrow u \land t \rightarrow v \implies u \downarrow v)$</td>
</tr>
<tr>
<td>WN($t$)</td>
<td>$\exists u \ (t \rightarrow^! u)$</td>
</tr>
<tr>
<td>UN($t$)</td>
<td>$\forall u \forall v \ (t \rightarrow^! u \land t \rightarrow^! v \implies u = v)$</td>
</tr>
<tr>
<td>NFP($t$)</td>
<td>$\forall u \forall v \ (t \rightarrow u \land t \rightarrow^! v \implies u \rightarrow^! v)$</td>
</tr>
<tr>
<td>NF($t$)</td>
<td>$\neg \exists u \ (t \rightarrow u)$</td>
</tr>
<tr>
<td>UNC</td>
<td>$\forall t \forall u \ (t \leftrightarrow^* u \land NF(t) \land NF(u) \implies t = u)$</td>
</tr>
</tbody>
</table>

**Example:**

Let $t$ and $s$ be terms in a rewriting system. Then, $s \rightarrow^* t$ means that $s$ can be rewritten into $t$ through a sequence of applications of rewrite rules. Similarly, $s \rightarrow^! t$ means that $s$ can be rewritten into $t$ in a single step, without the possibility of further rewriting.

CR($t$) denotes the correctness of term $t$, meaning that $t$ cannot be rewritten further. WCR($t$) is a weaker form of correctness, allowing for a single step of rewriting. WN($t$) indicates that $t$ is a normal form, i.e., it cannot be rewritten any further. UN($t$) signifies that $t$ is a unique normal form, meaning that any two normal forms for $t$ are identical. NFP($t$) indicates that $t$ is not a fixed point, i.e., $t$ can be rewritten into another term. NF($t$) means that $t$ is not a normal form, indicating it can be rewritten further.

These definitions provide a formal framework for reasoning about the rewriting process and the properties of terms within the system.
Outline

- First-Order Theory of Rewriting
- Automation
- Properties on Open Terms
- Experiments
- Future Work
Translation

• **binary predicates** are RR$_2$ relations and implemented via tree automata
  • ground tree transducers (GTTs) for $\iff$
  • RR$_n$ automata for all relations
Translation

- Binary predicates are RR$_2$ relations and implemented via tree automata
  - Ground tree transducers (GTTs) for $\iff$
  - RR$_n$ automata for all relations

- Implications and universal quantifiers are eliminated

\[
\varphi \Rightarrow \psi \equiv \neg \varphi \lor \psi \quad \forall x \varphi \equiv \neg \exists x \neg \varphi
\]
Translation

- binary predicates are RR$_2$ relations and implemented via tree automata
  - ground tree transducers (GTTs) for $\iff$
  - RR$_n$ automata for all relations
- implications and universal quantifiers are eliminated
  \[
  \varphi \Rightarrow \psi \equiv \neg \varphi \lor \psi \quad \quad \quad \forall x \varphi \equiv \neg \exists x \neg \varphi
  \]
- negations are pushed inside and double negations are eliminated
Translation

- binary predicates are RR$_2$ relations and implemented via tree automata
  - ground tree transducers (GTTs) for $\equiv$
  - RR$_n$ automata for all relations
- implications and universal quantifiers are eliminated
  \[
  \varphi \Rightarrow \psi \equiv \neg \varphi \lor \psi \quad \forall x \varphi \equiv \neg \exists x \neg \varphi
  \]
- negations are pushed inside and double negations are eliminated
- remaining propositional connectives are implemented by corresponding closure operations on RR$_n$ automata
Translation

- binary predicates are RR₂ relations and implemented via tree automata
  - ground tree transducers (GTTs) for \( \iff \)
  - RRₙ automata for all relations
- implications and universal quantifiers are eliminated
  \[
  \varphi \Rightarrow \psi \equiv \neg \varphi \lor \psi \quad \forall x \varphi \equiv \neg \exists x \neg \varphi
  \]
- negations are pushed inside and double negations are eliminated
- remaining propositional connectives are implemented by corresponding closure operations on RRₙ automata
- existential quantifiers are implemented using projection
Translation

- binary predicates are RR\(_2\) relations and implemented via tree automata
  - ground tree transducers (GTTs) for \(\equiv\)
  - RR\(_n\) automata for all relations
- implications and universal quantifiers are eliminated
  \[ \varphi \Rightarrow \psi \equiv \neg \varphi \lor \psi \]
  \[ \forall x \varphi \equiv \neg \exists x \neg \varphi \]
- negations are pushed inside and double negations are eliminated
- remaining propositional connectives are implemented by corresponding closure operations on RR\(_n\) automata
- existential quantifiers are implemented using projection

Remark

formulas are not transformed into prenex normal form, since this increases the dimension of involved relations
Outline

- First-Order Theory of Rewriting
- Automation
- Properties on Open Terms
- Experiments
- Future Work
Definition

TRS is confluent if \( \forall s \forall t \forall u \left( s \rightarrow^* t \land s \rightarrow^* u \implies \exists v \left( t \rightarrow^* v \land u \rightarrow^* v \right) \right) \)
Definition

TRS is confluent if \( \forall s \forall t \forall u \left( s \rightarrow^* t \land s \rightarrow^* u \implies \exists v \left( t \rightarrow^* v \land u \rightarrow^* v \right) \right) \)

Remarks

- variables \( s, t, u, v \) range over all terms
**Definition**

TRS is confluent if \( \forall s \forall t \forall u \left( s \rightarrow^* t \land s \rightarrow^* u \Rightarrow \exists v \left( t \rightarrow^* v \land u \rightarrow^* v \right) \right) \)

**Remarks**

- variables \( s, t, u, v \) range over all terms
- FORT is based on tree automata techniques and (hence) variables range over ground terms
**Definition**

TRS is confluent if

\[
\forall s \forall t \forall u \left( s \xrightarrow{*} t \land s \xrightarrow{*} u \implies \exists v \left( t \xrightarrow{*} v \land u \xrightarrow{*} v \right) \right)
\]

**Remarks**

- variables \( s, t, u, v \) range over all terms
- FORT is based on tree automata techniques and (hence) variables range over ground terms
- confluence \( \neq \) ground-confluence
Properties on Open Terms

**Definition**

TRS is confluent if \( \forall s \forall t \forall u \ (s \rightarrow^* t \land s \rightarrow^* u \implies \exists v \ (t \rightarrow^* v \land u \rightarrow^* v)) \)

**Remarks**

- variables \( s, t, u, v \) range over all terms
- FORT is based on tree automata techniques and (hence) variables range over ground terms
- confluence \( \neq \) ground-confluence

**FSCD 2016 submission**

*It should be stressed that the above properties are restricted to ground terms. So CR stands for ground-confluence, which is different from confluence, even in the presence of ground terms; consider e.g. the rules \( f(x) \rightarrow x \) and \( f(x) \rightarrow c \).*
Definition

TRS is confluent if $\forall s \forall t \forall u \left( s \rightarrow^* t \land s \rightarrow^* u \implies \exists v \left( t \rightarrow^* v \land u \rightarrow^* v \right) \right)$

Remarks

- variables $s$, $t$, $u$, $v$ range over all terms
- FORT is based on tree automata techniques and (hence) variables range over ground terms
- confluence $\neq$ ground-confluence

FSCD 2016 submission

It should be stressed that the above properties are restricted to ground terms. So CR stands for ground-confluence, which is different from confluence, even in the presence of ground terms; consider e.g. the rules $f(x) \rightarrow x$ and $f(x) \rightarrow c$. For (left-linear) right-ground TRSs there is no difference.
Example

TRS

\[ a \rightarrow b \quad f(a, x) \rightarrow b \quad f(b, b) \rightarrow b \]
Example

TRS

\[ a \rightarrow b \quad f(a, x) \rightarrow b \quad f(b, b) \rightarrow b \]

is ground-confluent but not confluent
Example

TRS

\[
\begin{align*}
    a & \rightarrow b \\
    f(a, x) & \rightarrow b \\
    f(b, b) & \rightarrow b
\end{align*}
\]

is ground-confluent but not confluent

Confluence Related Properties

\[
\begin{align*}
    \text{CR: } & \quad \forall s \forall t \forall u (s \rightarrow^* t \land s \rightarrow u \implies t \downarrow u) \\
    \text{WCR: } & \quad \forall s \forall t \forall u (s \rightarrow^! t \land s \rightarrow^! u \implies t \downarrow u) \\
    \text{UN: } & \quad \forall s \forall t \forall u (s \rightarrow^! t \land s \rightarrow^! u \implies t = u) \\
    \text{UNC: } & \quad \forall t \forall u (t \leftrightarrow^* u \land \text{NF}(t) \land \text{NF}(u) \implies t = u) \\
    \text{NFP: } & \quad \forall s \forall t \forall u (s \rightarrow^! t \land s \rightarrow^! u \implies t \rightarrow^! u) \\
    \text{SCR: } & \quad \forall s \forall t \forall u (s \rightarrow t \land s \rightarrow u \implies \exists v (t \rightarrow^= v \land u \rightarrow^* v))
\end{align*}
\]
Example

TRS

\[ a \rightarrow b \quad f(a, x) \rightarrow b \quad f(b, b) \rightarrow b \]

is ground-confluent but not confluent

Confluence Related Properties

- **CR:** \( \forall s \forall t \forall u (s \rightarrow^* t \land s \rightarrow u \implies t \downarrow u) \)
- **WCR:** \( \forall s \forall t \forall u (s \rightarrow t \land s \rightarrow u \implies t \downarrow u) \)
- **UN:** \( \forall s \forall t \forall u (s \rightarrow! t \land s \rightarrow! u \implies t = u) \)
- **UNC:** \( \forall t \forall u (t \leftrightarrow^* u \land NF(t) \land NF(u) \implies t = u) \)
- **NFP:** \( \forall s \forall t \forall u (s \rightarrow t \land s \rightarrow! u \implies t \rightarrow! u) \)
- **SCR:** \( \forall s \forall t \forall u (s \rightarrow t \land s \rightarrow u \implies \exists v (t \rightarrow= v \land u \rightarrow^* v)) \)

\( \mathcal{P} = \{ \text{CR, WCR, UN, UNC, NFP, SCR} \} \)
Properties on Open Terms

Relationships

WCR ↔ CR → NFP → UNC → UN

SCR

Remark: ∀P ∈ P_G \Rightarrow P
Properties on Open Terms

Relationships

\[ \text{WCR} \iff \text{CR} \implies \text{NFP} \implies \text{UNC} \implies \text{UN} \]

\[ \text{SCR} \]

Notation

\( G_P \) denotes property \( P \) restricted to ground terms
Properties on Open Terms

Relationships

WCR ↔ CR ⟷ NFP ⟷ UNC ⟷ UN

SCR

GSCR

GWCR ↔ GCR ⟷ GNFP ⟷ GUNC ⟷ GUN

Notation

$GP$ denotes property $P$ restricted to ground terms
Properties on Open Terms

**Relationships**

```
WCR ↔ CR → NFP → UNC → UN

SCR

GSCR

GWCR ↔ GCR → GNFP → GUNC → GUN
```

**Notation**

GP denotes property $P$ restricted to ground terms
Properties on Open Terms

Relationships

\[
\begin{align*}
\text{WCR} & \iff \text{CR} \rightarrow \text{NFP} \rightarrow \text{UNC} \rightarrow \text{UN} \\
\text{SCR} & \rightarrow \text{GSCR} \rightarrow \text{GWCR} \leftarrow \text{GCR} \rightarrow \text{GNFP} \rightarrow \text{GUNC} \rightarrow \text{GUN}
\end{align*}
\]

Notation

\( GP \) denotes property \( P \) restricted to ground terms

Remark

\[ \forall P \in \mathbb{P} \quad GP \not\supseteq P \]
Lemma

\forall\ \text{left-linear right-ground } \text{TRS } (\mathcal{F}, \mathcal{R})

1. \quad (\mathcal{F}, \mathcal{R}) \vdash P \iff (\mathcal{F} \cup \{ c \}, \mathcal{R}) \vdash GP \quad \forall P \in \mathcal{P} \setminus \{ \text{UNC} \}

\text{with fresh constant } c
Lemma

∀ left-linear right-ground TRS $(\mathcal{F}, \mathcal{R})$

1. $(\mathcal{F}, \mathcal{R}) \vDash P \iff (\mathcal{F} \cup \{c\}, \mathcal{R}) \vDash GP \quad \forall P \in \mathcal{P} \setminus \{UNC\}$

2. $(\mathcal{F}, \mathcal{R}) \vDash \text{UNC} \iff (\mathcal{F} \cup \{c, c'\}, \mathcal{R}) \vDash \text{GUNC}$

with fresh constants $c$ and $c'$
Lemma

∀ left-linear right-ground TRS \((F, R)\)

1. \((F, R) \models P \iff (F \cup \{ c \}, R) \models GP\) \(\forall P \in \mathcal{P} \setminus \{UNC\}\)

2. \((F, R) \models UNC \iff (F \cup \{ c, c' \}, R) \models GUNC\)

with fresh constants \(c\) and \(c'\)

Example

left-linear right-ground TRS

\[
\begin{align*}
  a &\rightarrow b \\
  f(x, a) &\rightarrow f(b, b) \\
  f(b, x) &\rightarrow f(b, b) \\
  f(f(x, y), z) &\rightarrow f(b, b)
\end{align*}
\]

- does not satisfy UNC: \(f(x, b) \leftarrow f(x, a) \rightarrow f(b, b) \leftarrow f(y, a) \rightarrow f(y, b)\)
Lemma

∀ left-linear right-ground TRS \((\mathcal{F}, \mathcal{R})\)

1. \((\mathcal{F}, \mathcal{R}) \models P \iff (\mathcal{F} \cup \{c\}, \mathcal{R}) \models GP \quad \forall P \in \mathcal{P} \setminus \{\text{UNC}\}

2. \((\mathcal{F}, \mathcal{R}) \models \text{UNC} \iff (\mathcal{F} \cup \{c, c'\}, \mathcal{R}) \models \text{GUNC}

with fresh constants \(c\) and \(c'\)

Example

left-linear right-ground TRS

\[
\begin{align*}
a & \rightarrow b \\
f(x, a) & \rightarrow f(b, b) \\
f(b, x) & \rightarrow f(b, b) \\
f(f(x, y), z) & \rightarrow f(b, b)
\end{align*}
\]

- does not satisfy UNC: \(f(x, b) \leftarrow f(x, a) \rightarrow f(b, b) \leftarrow f(y, a) \rightarrow f(y, b)\)
- adding single fresh constant \(c\) is not enough to violate GUNC
Lemma

∀ left-linear right-ground TRS \((\mathcal{F}, \mathcal{R})\)

1. \((\mathcal{F}, \mathcal{R}) \models P \iff (\mathcal{F} \cup \{ c \}, \mathcal{R}) \models GP \quad \forall P \in \mathcal{P} \setminus \{ \text{UNC} \}

2. \((\mathcal{F}, \mathcal{R}) \models \text{UNC} \iff (\mathcal{F} \cup \{ c, c' \}, \mathcal{R}) \models \text{GUNC}

with fresh constants \(c\) and \(c'\)

Example

left-linear right-ground TRS

\[
\begin{align*}
a & \rightarrow b \\
f(x, a) & \rightarrow f(b, b) \\
f(b, x) & \rightarrow f(b, b) \\
f(f(x, y), z) & \rightarrow f(b, b)
\end{align*}
\]

- does not satisfy UNC: \(f(x, b) \leftarrow f(x, a) \rightarrow f(b, b) \leftarrow f(y, a) \rightarrow f(y, b)\)
- adding single fresh constant \(c\) is not enough to violate GUNC
- GUNC is violated by adding another fresh constant \(c'\):

\[
\begin{align*}
f(c, b) & \leftarrow f(c, a) \rightarrow f(b, b) \leftarrow f(c', a) \rightarrow f(c', b)
\end{align*}
\]
Remarks

- for termination (SN) and normalization (WN) no fresh constants are needed.
### Remarks

- for termination (SN) and normalization (WN) no fresh constants are needed
- for other properties expressible in first-order theory of rewriting adding one or two constants may be insufficient
Remarks

- for termination (SN) and normalization (WN) no fresh constants are needed
- for other properties expressible in first-order theory of rewriting adding one or two constants may be insufficient: 

TRS consisting of rule $f(x) \rightarrow a$ satisfies

$$P: \neg \exists s \exists t \exists u \forall v (v \leftrightarrow^* s \lor v \leftrightarrow^* t \lor v \leftrightarrow^* u)$$
Remarks

- for termination (SN) and normalization (WN) no fresh constants are needed
- for other properties expressible in first-order theory of rewriting adding one or two constants may be insufficient:

  TRS consisting of rule $f(x) \rightarrow a$ satisfies

  $$P: \neg \exists s \exists t \exists u \forall v (v \leftrightarrow^* s \lor v \leftrightarrow^* t \lor v \leftrightarrow^* u)$$

  but GP does not hold after adding additional constants $c$ and $c'$
Remarks

- for termination (SN) and normalization (WN) no fresh constants are needed
- for other properties expressible in first-order theory of rewriting adding one or two constants may be insufficient:

TRS consisting of rule $f(x) \rightarrow a$ satisfies

$$P: \neg \exists s \exists t \exists u \forall v (v \leftrightarrow^* s \lor v \leftrightarrow^* t \lor v \leftrightarrow^* u)$$

but $GP$ does not hold after adding additional constants $c$ and $c'$

- adding fresh unary function symbol $g$ and fresh constant $c$ in order to create infinitely many ground normal forms
Properties on Open Terms

Remarks

• for termination (SN) and normalization (WN) no fresh constants are needed
• for other properties expressible in first-order theory of rewriting adding one or two constants may be insufficient:

TRS consisting of rule $f(x) \rightarrow a$ satisfies

$$P: \neg \exists s \exists t \exists u \forall v \ (v \leftrightarrow^* s \lor v \leftrightarrow^* t \lor v \leftrightarrow^* u)$$

but $GP$ does not hold after adding additional constants $c$ and $c'$

• adding fresh unary function symbol $g$ and fresh constant $c$ in order to create infinitely many ground normal forms is unsound in general:

consider $a \rightarrow b$ and $\forall s \forall t \ (s \rightarrow t \implies s \xrightarrow{c} t)$
Properties on Open Terms

Definition

signature $\mathcal{F}$ is **monadic** if $\mathcal{F}$ contains no function symbols of arity $> 1$

Example

taking 0.85 seconds but 1.73 seconds if fresh constant is added
Definition

signature $\mathcal{F}$ is monadic if $\mathcal{F}$ contains no function symbols of arity $> 1$

Lemma

$\forall$ left-linear right-ground TRS $(\mathcal{F}, \mathcal{R})$ such that $\mathcal{R}$ is ground or $\mathcal{F}$ is monadic

$(\mathcal{F}, \mathcal{R}) \models P \iff (\mathcal{F}, \mathcal{R}) \models GP \quad \forall P \in \mathcal{P}$
**Definition**

signature $\mathcal{F}$ is monadic if $\mathcal{F}$ contains no function symbols of arity $> 1$

**Lemma**

∀ left-linear right-ground TRS $(\mathcal{F}, \mathcal{R})$ such that $\mathcal{R}$ is ground or $\mathcal{F}$ is monadic

$(\mathcal{F}, \mathcal{R}) \models P \iff (\mathcal{F}, \mathcal{R}) \models GP \quad \forall P \in \mathcal{P}$

**Example**

checking GCR of TRS

\[
\begin{align*}
  f(f(f(x))) & \rightarrow a &
  f(f(a)) & \rightarrow a &
  f(a) & \rightarrow a \\
  f(f(g(g(x)))) & \rightarrow f(a) &
  g(f(a)) & \rightarrow a &
  g(a) & \rightarrow a
\end{align*}
\]

takes 0.85 seconds but 1.73 seconds if fresh constant is added
Outline

- First-Order Theory of Rewriting
- Automation
- Properties on Open Terms
- Experiments
- Future Work
Synthesis Experiments with FORT 0.2

- FW: a:0 b:0 f:2

GWCR &~WCR &~GCR

a → b
f(x, a) → a
a → f(a, a)

109 s

GCR &~CR &~GSCR

a → b
f(x, a) → b
b → f(a, a)

16 s

GNFP &~NFP &~GCR

a → b
f(x, a) → f(a, a)
f(b, b) → f(a, a)

16 s

GUNC &~UNC &~GNFP

a → a
f(x, a) → a
f(b, x) → b

95 s
Experiments

Relationships

- WCR
- CR
- NFP
- UNC
- UN
- SCR
- GSCR
- GWCR
- GCR
- GNFP
- GUNC
- GUN

Synthesis Experiments with **FORT 0.2**

```
- S -f "a:0 b:0 f:2"
```
Synthesis Experiments with **FORT 0.2**

```
-S -f "a:0 b:0 f:2"

"GWCR & ~WCR & ~GCR"
```
Synthesis Experiments with **FORT 0.2**

- `S -f "a:0 b:0 f:2"

- "GWCR & ~WCR & ~GCR"  
  \[ a \rightarrow b \quad f(x, a) \rightarrow a \quad a \rightarrow f(a, a) \]  
  \[ 80 \text{ s} \]
Experiments

Relationships

WGCR ← CR ← NFP ← S ← "GWCR & ~WCR & ~GCR" a → b f(x, a) → a a → f(a, a) 80 s

GNFP ← GCR ← GNFP ← "GCR & ~CR & ~GSCR" a → b f(x, a) → f(a, a) 16 s

GNFP ← GCR ← GNFP ← "GNFP & ~NFP & ~GCR" a → b f(x, a) → f(a, a) f(b, b) → f(a, a) 16 s

GUNC ← UN ← "GUNC & ~UNC & ~GNFP" a → f(b, x) f(b, b) → b 95 s

NFP ← UN ← "NFP & ~UN & ~UNF" a → f(b, x) f(b, b) → b 80 s

UN ← UN ← "UN & ~UN & ~UNF" a → f(b, x) f(b, b) → b 80 s

UN ← UN ← "UN & ~UN & ~UNF" a → f(b, x) f(b, b) → b 80 s

Experiments with FORT 0.2

-S -f "a:0 b:0 f:2"

"GWCR & ~WCR & ~GCR" a → b f(x, a) → a a → f(a, a) 80 s

"GCR & ~CR & ~GSCR"

FR (ICS @ UIBK)
Synthesis Experiments with **FORT 0.2**

- \(-S \ -f \ "a:0 \ b:0 \ f:2"\)

- "GWCR & ~WCR & ~GCR" \(a \rightarrow b\) \(f(x, a) \rightarrow a\) \(a \rightarrow f(a, a)\) 80 s

- "GCR & ~CR & ~GSCR" \(a \rightarrow b\) \(f(x, a) \rightarrow b\) \(b \rightarrow f(a, a)\) 109 s
### Relationships

![Diagram showing relationships]

### Synthesis Experiments with **FORT 0.2**

```
-S -f "a:0 b:0 f:2"

"GWCR & ~WCR & ~GCR"  a → b  f(x, a) → a  a → f(a, a)  80 s
"GCR & ~CR & ~GSCR"  a → b  f(x, a) → b  b → f(a, a)  109 s
"GNFP & ~NFP & ~GCR"
```
Synthesis Experiments with **FORT 0.2**

\[ -S \ -f \ "a:0 \ b:0 \ f:2" \]

- "GWCR & ~WCR & ~GCR"  \( a \rightarrow b \)  \( f(x, a) \rightarrow a \)  \( a \rightarrow f(a, a) \)  \( 80 \text{ s} \)
- "GCR & ~CR & ~GSCR"  \( a \rightarrow b \)  \( f(x, a) \rightarrow b \)  \( b \rightarrow f(a, a) \)  \( 109 \text{ s} \)
- "GNFP & ~NFP & ~GCR"  \( a \rightarrow b \)  \( f(x, a) \rightarrow f(a, a) \)  \( f(b, b) \rightarrow f(a, a) \)  \( 16 \text{ s} \)
**Experiments**

### Relationships

```
<table>
<thead>
<tr>
<th>WCR</th>
<th>CR</th>
<th>NFP</th>
<th>UNC</th>
<th>UN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GWCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNFP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUNC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### Synthesis Experiments with **FORT 0.2**

```
-S -f "a:0 b:0 f:2"

"GWCR & ~WCR & ~GCR"  a → b  f(x, a) → a  a → f(a, a)  80 s
"GCR & ~CR & ~GSCR"  a → b  f(x, a) → b  b → f(a, a)  109 s
"GNFP & ~NFP & ~GCR"  a → b  f(x, a) → f(a, a)  f(b, b) → f(a, a)  16 s
"GUNC & ~UNC & ~GNFP"
```
Experiments

Relationships

WCR ↔ CR → NFP → UNC → UN
SCR ↓ GSCR ↑
GWCR ↔ GCR → GNFP → GUNC → GUN

Synthesis Experiments with FORT 0.2

-S -f "a:0 b:0 f:2"

"GWCR & ~WCR & ~GCR"  a → b  f(x, a) → a  a → f(a, a)  80 s
"GCR & ~CR & ~GSCR"  a → b  f(x, a) → b  b → f(a, a)  109 s
"GNFP & ~NFP & ~GCR"  a → b  f(x, a) → f(a, a)  f(b, b) → f(a, a)  16 s
"GUNC & ~UNC & ~GNFP"  a → a  f(x, a) → a  f(b, x) → b  95 s
Synthesis Experiments with **FORT 0.2**

- **GWCR & ~WCR & ~GCR**
  \[ a \rightarrow b \quad f(x, a) \rightarrow a \quad a \rightarrow f(a, a) \]  80 s

- **GCR & ~CR & ~GSCR**
  \[ a \rightarrow b \quad f(x, a) \rightarrow b \quad b \rightarrow f(a, a) \]  109 s

- **GNFP & ~NFP & ~GCR**
  \[ a \rightarrow b \quad f(x, a) \rightarrow f(a, a) \quad f(b, b) \rightarrow f(a, a) \]  16 s

- **GUNC & ~UNC & ~GNFP**
  \[ a \rightarrow a \quad f(x, a) \rightarrow a \quad f(b, x) \rightarrow b \]  95 s
Experiments

Relationships

- WCR
- CR
- NFP
- UNC
- UN
- SCR
- GSCR
- GWCR
- GCR
- GNFP
- GUNC
- GUN

Synthesis Experiments with FORT 0.2

- `S -f "a:0 b:0 f:2"

- "GWCR & ~WCR & ~GCR"  
  \[a \rightarrow b \ f(x, a) \rightarrow a\]  
  \[a \rightarrow f(a, a)\]  
  \[80 \text{ s}\]

- "GCR & ~CR & ~GSCR"  
  \[a \rightarrow b \ f(x, a) \rightarrow b\]  
  \[b \rightarrow f(a, a)\]  
  \[109 \text{ s}\]

- "GNFP & ~NFP & ~GCR"  
  \[a \rightarrow b \ f(x, a) \rightarrow f(a, a)\]  
  \[f(b, b) \rightarrow f(a, a)\]  
  \[16 \text{ s}\]

- "~GNFP & GUNC & ~UNC"  
  \[a \rightarrow a \ f(x, a) \rightarrow a\]  
  \[f(b, x) \rightarrow b\]  
  \[21 \text{ s}\]
### Experiments

#### Relationships

- **WCR** ↔ **CR** → **NFP** → **UNC** → **UN**
- **SCR** ↔ **GSCR** ↔ **GWCR** ↔ **GCR** ↔ **GNFP** → **GUNC** → **GUN**

#### Synthesis Experiments with **FORT 1.0**

- `-S -f "a:0 b:0 f:2"
- "GWCR & ~WCR & ~GCR"  
  \[a \rightarrow b  
  f(x, a) \rightarrow a  
  a \rightarrow f(a, a)\]  
  \[8 - 20 \text{ s}\]
- "GCR & ~CR & ~GSCR"  
  \[a \rightarrow b  
  f(x, a) \rightarrow b  
  b \rightarrow f(a, a)\]  
  \[8 - 10 \text{ s}\]
- "GNFP & ~NFP & ~GCR"  
  \[a \rightarrow b  
  f(x, a) \rightarrow f(a, a)  
  f(b, b) \rightarrow f(a, a)\]  
  \[9 - 11 \text{ s}\]
- "~GNFP & GUNC & ~UNC"  
  \[a \rightarrow a  
  f(x, a) \rightarrow a  
  f(b, x) \rightarrow b\]  
  \[2 - 4 \text{ s}\]
### Example

"GUN & ~UN & ~GUNC"

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b \to a$</td>
<td>$d \to c$</td>
</tr>
<tr>
<td>$b \to c$</td>
<td>$d \to e$</td>
</tr>
<tr>
<td>$c \to c$</td>
<td>$f(x, a) \to A$</td>
</tr>
</tbody>
</table>
### Example

"GUN & ~UN & ~GUNC"

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rewrite</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>b → a</td>
<td>d → c</td>
<td>f(x, e) → A</td>
</tr>
<tr>
<td>b → c</td>
<td>d → e</td>
<td>f(x, A) → A</td>
</tr>
<tr>
<td>c → c</td>
<td>f(x, a) → A</td>
<td>f(c, x) → A</td>
</tr>
</tbody>
</table>

### Comparison (GCR)

AGCP is recent tool for ground-confluence of many-sorted TRSs based on rewriting induction (Aoto and Toyama, FSCD 2016)
Example

"GUN & ~UN & ~GUNC"

\[ b \to a \quad d \to c \quad f(x, e) \to A \\
\]
\[ b \to c \quad d \to e \quad f(x, A) \to A \\
\]
\[ c \to c \quad f(x, a) \to A \quad f(c, x) \to A \\
\]

Comparison (GCR)

AGCP is recent tool for ground-confluence of many-sorted TRSs based on rewriting induction (Aoto and Toyama, FSCD 2016)

<table>
<thead>
<tr>
<th>65 TRSs</th>
<th>AGCP (∅ time)</th>
<th>FORT 0.2 (∅ time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>8 (0.02 s)</td>
<td>42 (0.42 s)</td>
</tr>
<tr>
<td>no</td>
<td>–</td>
<td>14 (3.88 s)</td>
</tr>
<tr>
<td>maybe</td>
<td>56 (0.19 s)</td>
<td>–</td>
</tr>
<tr>
<td>timeout</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>total time</td>
<td>71 s</td>
<td>612 s</td>
</tr>
</tbody>
</table>
**Example**

"GUN & ~UN & ~GUNC"

\[
\begin{align*}
  b &\rightarrow a \\
  b &\rightarrow c \\
  c &\rightarrow c
\end{align*}
\]

\[
\begin{align*}
  d &\rightarrow c \\
  d &\rightarrow e \\
  f(x, a) &\rightarrow A
\end{align*}
\]

\[
\begin{align*}
  f(x, e) &\rightarrow A \\
  f(x, A) &\rightarrow A \\
  f(c, x) &\rightarrow A
\end{align*}
\]

**Comparison (GCR)**

AGCP is recent tool for ground-confluence of many-sorted TRSs based on rewriting induction (Aoto and Toyama, FSCD 2016)

<table>
<thead>
<tr>
<th>65 TRSs</th>
<th>AGCP (∅ time)</th>
<th>FORT 0.2 (∅ time)</th>
<th>FORT 1.0 (∅ time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>8 (0.02 s)</td>
<td>42 (0.42 s)</td>
<td>43 (0.26 s)</td>
</tr>
<tr>
<td>no</td>
<td>–</td>
<td>14 (3.88 s)</td>
<td>18 (0.96 s)</td>
</tr>
<tr>
<td>maybe</td>
<td>56 (0.19 s)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>timeout</td>
<td>1</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>total time</td>
<td>71 s</td>
<td>612 s</td>
<td>268 s</td>
</tr>
</tbody>
</table>


**Experiments**

Example

"GUN & ~UN & ~GU"

Let's see what happens at **COCO**

\[ f(x, e) \rightarrow A \]
\[ f(x, A) \rightarrow A \]
\[ f(c, x) \rightarrow A \]

**Comparison (GCR)**

AGCP is recent tool for ground-confluence of many-sorted TRSs based on rewriting induction (Aoto and Toyama, FSCD 2016)

<table>
<thead>
<tr>
<th>65 TRSs</th>
<th>AGCP (∅ time)</th>
<th>FORT 0.2 (∅ time)</th>
<th>FORT 1.0 (∅ time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>8 (0.02 s)</td>
<td>42 (0.42 s)</td>
<td>43 (0.26 s)</td>
</tr>
<tr>
<td>no</td>
<td>–</td>
<td>14 (3.88 s)</td>
<td>18 (0.96 s)</td>
</tr>
<tr>
<td>maybe</td>
<td>56 (0.19 s)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>timeout</td>
<td>1</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>total time</td>
<td>71 s</td>
<td>612 s</td>
<td>268 s</td>
</tr>
</tbody>
</table>
Future Work

Outline

- First-Order Theory of Rewriting
- Automation
- Properties on Open Terms
- Experiments
- Future Work
Future Work

Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting ($\rightarrow$) is undecidable
Future Work

Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of **one-step** rewriting ($\rightarrow$) is undecidable

- ... even for linear non-erasing TRSs (Treinen, 1998)
Future Work

Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting ($\rightarrow$) is undecidable

- … even for linear non-erasing TRSs (Treinen, 1998)
- … even for complete right-ground TRSs (Vorobyov, 2002)
Limitation

Restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting ($\rightarrow$) is undecidable

- ... even for linear non-erasing TRSs (Treinen, 1998)
- ... even for complete right-ground TRSs (Vorobyov, 2002)

Ongoing and Future Work

- generating witnesses for existential formulas and formulas with free variables
Future Work

Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting ($\rightarrow$) is undecidable

- ... even for linear non-erasing TRSs (Treinen, 1998)
- ... even for complete right-ground TRSs (Vorobyov, 2002)

Ongoing and Future Work

- generating witnesses for existential formulas and formulas with free variables
- support for combinations of TRSs (e.g., to express commutation)
Future Work

Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting ($\rightarrow$) is undecidable

- ... even for linear non-erasing TRSs (Treinen, 1998)
- ... even for complete right-ground TRSs (Vorobyov, 2002)

Ongoing and Future Work

- generating witnesses for existential formulas and formulas with free variables
- support for combinations of TRSs (e.g., to express commutation)
- incorporating rewrite strategies
Limitation

restriction to left-linear right-ground TRSs is hard to overcome because first-order theory of one-step rewriting (→) is undecidable

- ... even for linear non-erasing TRSs (Treinen, 1998)
- ... even for complete right-ground TRSs (Vorobyov, 2002)

Ongoing and Future Work

- generating witnesses for existential formulas and formulas with free variables
- support for combinations of TRSs (e.g., to express commutation)
- incorporating rewrite strategies
- formalizing underlying theory in Isabelle/HOL