VRAIN Valencian Research Institute for Artificial Intelligence

On Proving Confluence of Generalized Term Rewriting Systems Using CONFident

Raúl Gutiérrez

Salvador Lucas

LISBON, PORTUGAL, JULY 24TH, 2026

Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
Spain

UPV

Motivation

Motivation

- We want to extend CONFident to prove and disprove **confluence** of Generalized Term Rewriting Systems (GTRSs).
- A GTRS is a tuple $\mathcal{R} = (\Omega, \mu, H, R)$, where:
  - $\Omega = (\mathcal{F}, \Pi)$ is a signature with predicates.
  - $\mu \in M_{\mathcal{F}}$.
  - H is a set of auxiliary clauses (H is used to model the semantics of conditions).
  - R is a set of rewrite rules $\ell \to r \Leftarrow c$.

Motivation

Example (Horn-clauses)

$$0 \neq s(n) \tag{1}$$

$$s(n) \neq 0 \tag{2}$$

$$s(n) \neq s(m) \Leftarrow n \neq m \tag{3}$$

Motivation

Example (Rules)

$$from(n) \to n : from(s(n)) \tag{4}$$

$$rm(x, nil) \to nil \tag{5}$$

$$rm(x, x : xs) \to rm(x, xs) \tag{6}$$

$$rm(x, y : xs) \to y : rm(x, xs) \Leftarrow x \neq y \tag{7}$$

$$edups(nil) \to nil \tag{8}$$

$$edups(x : nil) \to x : nil \tag{9}$$

$$edups(x : x : xs) \to edups(x : xs) \tag{10}$$

$$edups(x : y : xs) \to x : rm(x, edups(y : xs)) \Leftarrow x \neq y \tag{11}$$

Motivation

Example (Replacement map & Horn-Clauses - COPS)

```
(REPLACEMENT-MAP
(from )
(rm 2)
(cons 2)
)

(VAR m n x xs y)
(HORN-CLAUSES
Neq(0,s(n))
Neq(s(n),0)
Neq(s(m),s(n)) | Neq(m,n)
)
```

Motivation

Example (Rules - COPS)

```
(RULES
from(n) -> cons(n,from(s(n)))
rm(x,nil) -> nil
rm(x,cons(x,xs)) -> rm(x,xs)
rm(x,cons(y,xs)) -> cons(y,rm(x,xs)) | Neq(x,y)
edups(nil) -> nil
edups(cons(x,nil)) -> cons(x,nil)
edups(cons(x,cons(x,xs))) -> edups(cons(x,xs))
edups(cons(x,cons(y,xs))) ->
   cons(x,rm(x,edups(cons(y,xs)))) | Neq(x,y)
)
```

Well-Formed Proof Trees

(Rf)
$$\frac{}{x \to^* x}$$

(Co)
$$\frac{x \to y \qquad y \to^* z}{x \to^* z}$$

(Re)$_\beta$
$$\frac{B_1 \quad \ldots \quad B_n}{\ell \to r}$$
for $\beta : \ell \to r \Leftarrow B_1, \ldots, B_n \in \mathcal{R}$

(Pr)$_{f,i}$
$$\frac{x_i \to y_i}{f(x_1, \ldots, x_i, \ldots, x_k) \to f(x_1, \ldots, y_i, \ldots, x_k)}$$
for $f \in \mathcal{F}^{(k)}$ and $i \in \mu(f)$

(HC)$_\alpha$
$$\frac{A_1 \quad \ldots \quad A_n}{A}$$
for $\alpha : A \Leftarrow A_1 \wedge \cdots \wedge A_n \in \mathcal{H}$

First-Order Theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$ (1/2)

$$(\forall x)\ x \to^* x \tag{12}$$

$$(\forall x, y, z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z \tag{13}$$

$$(\forall x, y)\ x \to y \Rightarrow s(x) \to s(y) \tag{14}$$

$$(\forall x, y, z)\ x \to y \Rightarrow z : x \to z : y \tag{15}$$

$$(\forall x, y)\ x \to y \Rightarrow edups(x) \to edups(y) \tag{16}$$

$$(\forall n)\ from(n) \to n : from(s(x)) \tag{17}$$

$$(\forall x)\ rm(x, nil) \to nil \tag{18}$$

$$(\forall x, xs)\ rm(x, x : xs) \to rm(x, xs) \tag{19}$$

$$(\forall x, y, xs)\ Neq(x, y) \Rightarrow rm(x, y : xs) \to y : rm(x, xs) \tag{20}$$

$$edups(nil) \to nil \tag{21}$$

$$edups(x : nil) \to x : nil \tag{22}$$

$$(\forall x, xs)\ edups(x : x : xs) \to edups(x : xs) \tag{23}$$

$$(\forall x, y, xs)\ Neq(x, y) \Rightarrow edups(x : y : xs) \to x : rm(x, edups(y : xs)) \tag{24}$$

First-Order Theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$ (2/2)

$$(\forall n)\ Neq(0, s(n)) \tag{25}$$

$$(\forall n)\ Neq(s(n), 0) \tag{26}$$

$$(\forall n, m)\ Neq(n, m) \Rightarrow Neq(s(n), s(m)) \tag{27}$$

Well-Formed Proof Trees (from Rules to Horn-Clauses)

(Rf) $$\frac{}{x \to^* x}$$

(Co) $$\frac{x \to y \qquad y \to^* z}{x \to^* z}$$

(Pr)$_{f,i}$ $$\frac{x_i \to y_i}{f(x_1, \ldots, x_i, \ldots, x_k) \to f(x_1, \ldots, y_i, \ldots, x_k)}$$
for $f \in \mathcal{F}^{(k)}$ and $i \in \mu(f)$

(HC)$_\alpha$ $$\frac{A_1 \quad \ldots \quad A_n}{A}$$
for $\alpha : A \Leftarrow A_1 \wedge \cdots \wedge A_n \in \mathcal{H}$

Confluence Framework

Confluence Framework

Confluence Problem

Let $\mathcal{R}$ be a GTRS. A (local, strong) **confluence problem**, denoted $CR(\mathcal{R})$ ($WCR(\mathcal{R})$,$SCR(\mathcal{R})$), is **positive** if $\mathcal{R}$ is (local, strong) confluent; otherwise, it is *negative*.

Joinability Problem

Let $\mathcal{R}$ be a GTRS and π be a **conditional pair**. A (strong) **joinability problem**, denoted $JO(\mathcal{R}, \pi)$ ($SJO(\mathcal{R}, \pi)$), is **positive** if π is (strongly) joinable in $\mathcal{R}$; otherwise, it is *negative*.

GTRSs

```
1  data Signature idf
2    = Signature { defined         :: Set idf
3                , constructors   :: Set idf
4                , predicates     :: Set idf
5                } deriving (Eq,Show)
6
7  data GTRS idf idv term
8    = GTRS { gtrsSignature    :: Signature idf
9           , gtrsVariables    :: Variables idf
10          , gtrsRules        :: Set (CRule term)
11          , gtrsHornClauses :: Set (HornClause term)
12          }
```

Type Classes

- `IsProblem`
  - `getProblemType`
  - `getR`
- `IsConfProblem`
  - `getConfProblem`
  - `setConfProblem`
- `IsJProblem`
  - `getCPairs`
  - `setCPairs`

Processor

A *processor* P is a partial function from problems into sets of problems; alternatively it can return “no”. The domain of P (i.e., the set of problems on which P is defined) is denoted $\mathcal{D}om(\mathsf{P})$. We say that P is

- *sound* if for all $\tau \in \mathcal{D}om(\mathsf{P})$, τ is positive whenever $\mathsf{P}(\tau) \neq$ “no” and all $\tau' \in \mathsf{P}(\tau)$ are positive.
- *complete* if for all $\tau \in \mathcal{D}om(\mathsf{P})$, τ is negative whenever $\mathsf{P}(\tau) =$ “no” or τ' is negative for some $\tau' \in \mathsf{P}(\tau)$.

Processors

```
13 -- Infeasibility Rule Processor
14 data InfGTRSProcessor = InfGTRSProcessor {timeout :: Maybe Int}
15
16 instance (... functional dependencies ...)
17   => Processor InfGTRSProcessor (Problem GRewriting conftrs)
18                                 (Problem GRewriting conftrs) where
19   apply InfGTRSProcessor {timeout = to} inP = ...
20
21 -- | Critical Pairs Processor
22 data CriticalPairGTRSProcessor = CriticalPairGTRSProcessor
23
24 instance (... functional dependencies ...)
25   => Processor CriticalPairGTRSProcessor (Problem GRewriting conftrs1)
26                                          (Problem JGRewriting conftrs2) where
27   apply CriticalPairsCProcessor inP = ...
```

Adapted processors

- Extra variables check processor.
- Simplification processor.
  - Removing trivial rules.
  - Removing trivial conditions.
  - Removing infeasible conditional rules.
  - Removing ground atoms.
- Inlining procesor.
- Local/Strong confluence processors.
- Move between confluence and local/strong confluence problem processors.
- Confluence of terminating systems as local confluence processor.
- Orthogonality procesors (partially).
- Joinability processor.

Not adapted processors

- Modularity processor.
- Transformational processors.
- Orthogonality procesor.

Strategy

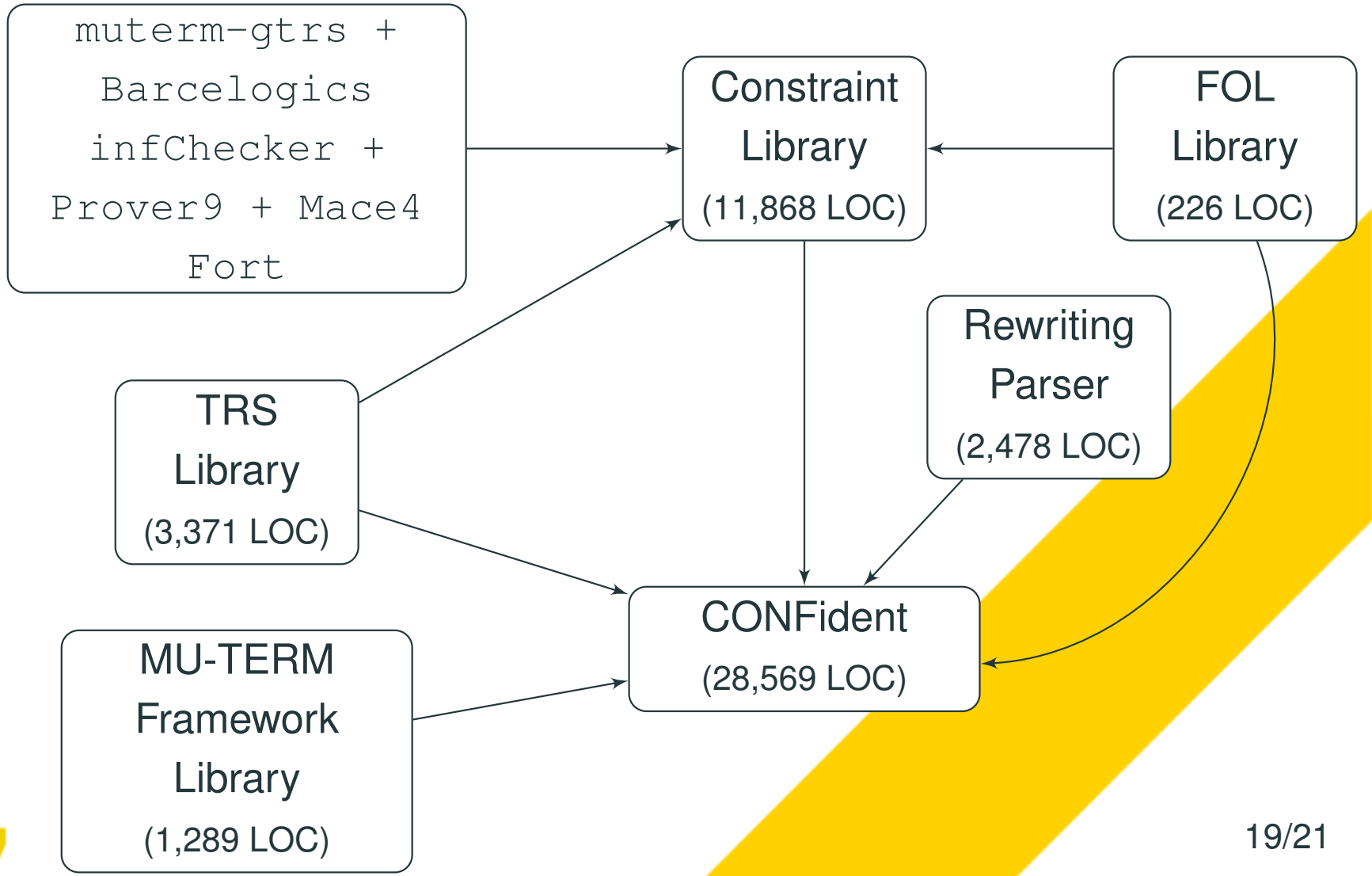
1. Extra variables check processor is applied.
2. Simplification and inlining processor are applied to simplify the input system or its rules.
3. Processors that obtain joinability problems are applied (with and without termination); and either
   1. the proof branch is *finished* (if the system is terminating and there is no critical pairs), or
   2. a set with a single WCR-problem is obtained; then the local confluence processor to obtain joinability problems; joinability processor is eventually used on each of them to *finish* the proof branch; or else,
   3. a set of joinability problems is obtained (if the system is terminating); then joinability processor is used on each of them to *finish* the proof branch.

Benchmarks

http://zenon.dsic.upv.es/confident/benchmarks/results-gtrs-20250504/results.html

Internal Structure

- URL: `http://zenon.dsic.upv.es/confident/`



Solving our Leading Example

- ...

Future Work

Future Work

- **Non adapted processors**?
- **Dedicated processors**?