Proving Confluence with CONFident

Raúl Gutiérrez[1] Miguel Vítores[2] Salvador Lucas[2]

HAIFA, AUGUST 1ST, 2022

[1] Universidad Politécnica de Madrid
Spain
[2] Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
Spain

Description

- CONFident is a tool for checking **(non-)confluence** of systems based on rewriting by means of its logical representation.
- Specially designed to (dis)prove confluence of
  - Term Rewriting Systems.
  - Conditional Term Rewriting Systems (join, oriented and semi-equational).
  - Context-Sensitive Term Rewriting Systems.

Confluence on reduction relations (1/2)

1. **Confluence** is the property of reduction relations guaranteeing that whenever s has two different reducts t and t' (i.e., $s \to^* t$ and $s \to^* t'$), both t and t' are joinable, i.e., they have a common reduct u (hence $t \to^* u$ and $t' \to^* u$ holds for some u).
2. Confluence is one of the most important properties of reduction relations: for instance,
   1. it ensures that for all expressions s, at most one irreducible reduct t of s can be obtained;
   2. it ensures that two divergent computations can always join in the future.
3. Thus, the semantics and implementation of rewriting-based languages is less dependent on specific strategies to implement reductions.

Confluence on reduction relations (2/2)

1. Local confluence is a property weaker than confluence that can be used to prove confluence in the presence of termination.

$$\begin{tikzcd}
 & t \arrow[dr, dotted, "*"] & \\
s \arrow[ur] \arrow[dr] & & u \\
 & t' \arrow[ur, dotted, "*"'] &
\end{tikzcd}$$

Local confluence

$$\begin{tikzcd}
 & t \arrow[dr, dotted, "*"] & \\
s \arrow[ur, "*"] \arrow[dr, "*"'] & & u \\
 & t' \arrow[ur, dotted, "*"'] &
\end{tikzcd}$$

confluence

Supported variants of term rewriting systems

Term rewriting systems

COPS - Example 669

$$
\begin{aligned}
a &\to c \\
f(a) &\to b \\
b &\to b \\
b &\to h(b, h(c, a))
\end{aligned}
$$

Conditional term rewriting systems

COPS - Example 409

$$
\begin{array}{rcl}
f(x,y) & \to & g(s(x)) \mid c(g(x)) \approx c(a) \\
f(x,y) & \to & h(s(x)) \mid c(h(x)) \approx c(a) \\
g(s(x)) & \to & x \\
h(s(x)) & \to & x \\
b & \to & b
\end{array}
$$

- $\approx$ can be interpreted as **reachability** ($\to^*$), **joinability** ($\downarrow$) or **conversion** ($\leftrightarrow^*$)

Context-sensitive term rewriting systems

TPDB - Ex_14_AEGL02

$$
\begin{aligned}
from(x) &\to cons(x, from(s(x))) \\
length(nil) &\to 0 \\
length(cons(x, y)) &\to s(length1(y)) \\
length1(x) &\to length(x)
\end{aligned}
$$

- $\mu(from) = \mu(cons) = \mu(s) = 1$ and $\mu(f) = \varnothing$ for $f \in \{length, length1, 0, nil\}$.

Confluence Framework

Confluence Framework

Confluence Problem

- Let $\mathcal{R}$ be a CTRS (TRSs are included). A *confluence problem*, denoted $CR(\mathcal{R})$, is *positive* if $\mathcal{R}$ is confluent; otherwise, it is *negative*.
- Let $\mathcal{R}$ be a TRS and μ be a replacement map. A μ-confluence problem, denoted $CR(\mathcal{R}, \mu)$, is *positive* if $\mathcal{R}$ is μ-confluent; otherwise, it is *negative*.

Confluence Framework

Joinability Problem

- Let $\mathcal{R}$ be a CTRS and π be a *conditional pair* where the conditional part has no occurrence of $\hookrightarrow$. A *joinability problem*, denoted $JO(\mathcal{R}, \pi)$, is *positive* if π is joinable in $\mathcal{R}$; otherwise, it is *negative*.
- Let $\mathcal{R}$ be a TRS, μ be a replacement map, and π be a *conditional pair* where the conditional part contains at most an occurrence of $\hookrightarrow$ [1]. A μ*-joinability problem*, denoted $JO(\mathcal{R}, \mu, \pi)$, is *positive* if π is μ-joinable in $\mathcal{R}$; otherwise, it is *negative*.

[1] This is necessary to deal with the extended μ-critical pairs

Confluence Framework

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 $\tau' \in \mathsf{P}(\tau)$.

List of processors

Simplification processor

$$\mathsf{P}_{\mathsf{Simp}}(CR(\mathcal{R})) = \{CR(\mathcal{R}')\}$$

Simplifications

1. *Removing or transforming rules (CS-TRSs and CTRSs).* All rules $t \to t$ or $t \to t \mid c$ for some term t are removed.
2. *Removing infeasible rules (CTRSs only).* Rules $\ell \to r \mid c$ with an infeasible condition c (i.e., such that no substitution σ makes $\sigma(c)$ to hold in $\mathcal{R}$) are removed. We use infChecker for that purpose.
3. *Inlining conditional rules (CTRSs only).* We can often reduce the number of conditions of a rule, by using *inlining*.

Modular decomposition processor

$$\mathsf{P}_{\mathsf{MD}}(CR(\mathcal{R})) = \{CR(\mathcal{R}_1), CR(\mathcal{R}_2)\}$$

Modularity conditions

1. Disjoint TRSs.
2. Constructor-sharing and left-linear TRSs.
3. Constructor-sharing and layer-preserving TRSs.

Huet&Levi processor

$$\mathsf{P}_{\mathsf{HL}}(\mathcal{CR}(\mathcal{R})) = \varnothing$$

Orthogonality conditions

1. Weakly orthogonal TRSs are confluent.
2. Left-linear and has no extended μ-critical pairs.
3. Orthogonal, properly oriented and right-stable.

Extended Huet processor

$$\mathsf{P}_{\mathsf{Huet}}(CR(\mathcal{R})) = \{JO(\mathcal{R}, \pi_1), \ldots, JO(\mathcal{R}, \pi_n)\}$$

$$\mathsf{P}_{\mathsf{Huet}}(CR(\mathcal{R}, \mu)) = \{JO(\mathcal{R}, \mu, \pi_1), \ldots, JO(\mathcal{R}, \mu, \pi_n)\}$$

Conditions

1. $\pi_1, \ldots, \pi_n$ are (extended μ-)critical pairs.

Extended Huet&Newman processor

$$\mathsf{P}_{\mathsf{HN}}(CR(\mathcal{R})) = \{JO(\mathcal{R}, \pi_1), \ldots, JO(\mathcal{R}, \pi_n)\}$$

$$\mathsf{P}_{\mathsf{HN}}(CR(\mathcal{R}, \mu)) = \{JO(\mathcal{R}, \mu, \pi_1), \ldots, JO(\mathcal{R}, \mu, \pi_n)\}$$

Conditions

1. $\mathcal{R}$ is (μ-)terminating.
2. $\pi_1, \ldots, \pi_n$ are (extended μ-)critical pairs.
3. For normal CTRSs, if $\mathcal{R}$ is terminating and left-linear.

Canonical joinability processor

$$\mathsf{P}_{\mathsf{CanJ}}(CR(\mathcal{R})) = \{JO(\mathcal{R}, \mu, \pi_1), \ldots, JO(\mathcal{R}, \mu, \pi_n)\}$$

Conditions

1. $\mathcal{R}$ is left-linear and level decreasing.
2. μ is the canonical replacement map.
3. $\mathcal{R}$ is μ-terminating.
4. $\pi_1, \ldots, \pi_n$ are μ-critical pairs.

Canonical μ-confluence processor

$$\mathsf{P}_{\mathsf{CanCR}}(CR(\mathcal{R})) = \{CR(\mathcal{R}, \mu)\}$$

Conditions

1. $\mathcal{R}$ is left-linear and normalizing.
2. μ is the canonical replacement map.

Unravelings processor

$$\mathsf{P}_{\mathsf{U}}(CR(\mathcal{R})) = \{CR(U(\mathcal{R}))\}$$

Conditions

1. $\mathcal{R}$ is terminating.
2. $\mathcal{R}$ is a deterministic 3-CTRS.

Joinability processor

$$\mathsf{P}_{\mathsf{JO}}(JO(\mathcal{R},\pi)) = \begin{cases} \varnothing & \text{if } \pi \text{ is joinable w.r.t. } \mathcal{R} \\ \text{no} & \text{if } \pi \text{ is not joinable w.r.t. } \mathcal{R} \end{cases}$$

$$\mathsf{P}_{\mathsf{JO}}(JO(\mathcal{R},\mu,\pi)) = \begin{cases} \varnothing & \text{if } \pi \text{ is } \mu\text{-joinable w.r.t. } \mathcal{R} \\ \text{no} & \text{if } \pi \text{ is not } \mu\text{-joinable w.r.t. } \mathcal{R} \end{cases}$$

External tools

1. We use tools like Prover9, AGES or Mace4.

Processors in the confluence framework

	P_{Simp}	P_{MD}	P_{Huet}	P_{HL}	P_{HN}	P_{CanJ}	P_{CanCR}	P_{U}
TRSs	✓	✓	✓	✓	✓	✓	✓	×
CS-TRSs	✓	×	✓	✓	✓	×	×	×
CTRSs	✓	×	✓	✓	✓	×	×	✓
Sound	✓	✓	×	✓	✓	✓	✓	✓
Complete	✓	✓	✓	✓	✓	×	×	×

Confluence proof tree

Confluence proof tree

Let $\mathcal{R}$ be a CTRS. A confluence proof tree $\mathcal{T}$ for $\mathcal{R}$ is a tree whose root label is $CR(\mathcal{R})$, whose inner nodes are labeled with problems, and whose leaves are labeled either with problems, "yes" or "no". For every inner node n labeled with τ, there is a processor P such that $\tau \in \mathcal{D}om(\mathsf{P})$ and:

1. if $\mathsf{P}(\tau) =$ "no" then n has just one child, labeled with"no".
2. if $\mathsf{P}(\tau) = \varnothing$ then n has just one child, labeled with "yes".
3. if $\mathsf{P}(\tau) = \{\tau_1, \ldots, \tau_m\}$ with $m > 0$, then n has m children labeled with the problems $\tau_1, \ldots, \tau_m$.

Confluence framework

Confluence framework

Let $\mathcal{R}$ be a CTRS and $\mathcal{T}$ be a confluence proof tree for $\mathcal{R}$. Then:

1. if all leaves in $\mathcal{T}$ are labeled with "yes" and all involved processors are sound for the problems they are applied to, then $\mathcal{R}$ is confluent.
2. if $\mathcal{T}$ has a leaf labeled with "no" and all processors in the path from the root to such a leaf are complete for the problems they are applied to, then $\mathcal{R}$ is non-confluent.

Implementation

- The tool is available here: `http://zenon.dsic.upv.es/confident/`
- It is written in Haskell and it is based on a divide-and-conquer schema where different techniques are applied recursively using a pre-defined strategy.
- Different techniques are defined based on the logical representation of the systems to check **(non-)joinability** of (conditional) critical pairs:
  - Proving **conditinal joinability** as a logical consequence using Theorem Provers.
  - Disproving **conditinal joinability** as a feasibility problem using infChecker.

Strategy and Results

- Our **proof strategy** is:
  1. we apply simplification and modular techniques;
  2. we analyze the system to extract good properties (external tools like MU-TERM can be used to check termination and operational termination);
  3. we compute the set of conditional critical pairs;
  4. we apply the semantic based techniques to prove or disprove the joinability of the conditional critial pairs.
- This year there is a **CSR subcategory** in CoCo 2022.