Proving and disproving feasibility with infChecker

Raúl Gutiérrez
Salvador Lucas

LEIPZIG, GERMANY, SEPTEMBER 3RD, 2025

Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
Spain

Motivation

Feasibility/Infeasibility

Question

Given a rewrite system $\mathcal{R}$ and terms s and t. Is there a substitution σ instantiating the variables in s and t such that the reachibility test $\sigma(s) \to_{\mathcal{R}}^{*} \sigma(t)$ succeeds?

Feasibility/Infeasibility

If such substitution does not exists, we say that the problem is **infeasible**; otherwise, we call it **feasible**.

Goal

Improve infChecker to give full support to a more general notion of feasibility.

Generalize Feasibility

Expressions

- **f-condition** (an atomic formula, e.g., $s \to^* t$, $s \downarrow t$, $s \leftrightarrow^* t, \ldots$),
- **f-sequence** (of f-conditions, interpreted as a *conjunction of atoms*), and
- **f-goal** (interpreted as *disjunctions of f-sequences*).

Feasibility/Infeasibility

These expressions F (viewed as boolean combinations of atoms) are said to be **feasible** with respect to a given first-order theory Th if there is a substitution σ such that $Th \vdash \sigma(F)$ holds. Otherwise, F is said to be **infeasible**.

From Conditional Term Rewriting Systems (CTRSs)

Example

$$
\begin{array}{rcl}
le(0, s(y)) & \to & true \\
le(s(x), s(y)) & \to & le(x, y) \\
le(x, 0) & \to & false \\
min(cons(x, nil)) & \to & x \\
min(cons(x, xs)) & \to & x \Leftarrow min(xs) \to^* y, le(x, y) \to^* true \\
min(cons(x, xs)) & \to & y \Leftarrow min(xs) \to^* y, le(x, y) \to^* false
\end{array}
$$

Well-Formed Proof Trees

Example

$$
\begin{array}{rcl}
le(0, s(y)) & \to & true \\
le(s(x), s(y)) & \to & le(x, y) \\
le(x, 0) & \to & false \\
min(cons(x, nil)) & \to & x \\
min(cons(x, xs)) & \to & x \Leftarrow min(xs) \to^* y, le(x, y) \to^* true \\
min(cons(x, xs)) & \to & y \Leftarrow min(xs) \to^* y, le(x, y) \to^* false
\end{array}
$$

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

(Re)$_\beta$ $$\frac{s_1 \to^* t_1 \quad \ldots \quad s_n \to^* t_n}{\ell \to r}$$

for $\beta : \ell \to r \Leftarrow s_1 \to^* t_1, \ldots, s_1 \to^* t_n \in \mathcal{R}$

(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 $1 \leq i \leq k$

First-Order Theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$

$$(\forall x)\ x \to^* x \tag{1}$$

$$(\forall x, y, z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z \tag{2}$$

$$(\forall x, y)\ x \to y \Rightarrow s(x) \to s(y) \tag{3}$$

$$(\forall x, y, z)\ x \to y \Rightarrow cons(x, z) \to cons(y, z) \tag{4}$$

$$(\forall x, y, z)\ x \to y \Rightarrow cons(z, x) \to cons(z, y) \tag{5}$$

$$(\forall x, y, z)\ x \to y \Rightarrow le(x, z) \to le(y, z) \tag{6}$$

$$(\forall x, y, z)\ x \to y \Rightarrow le(z, x) \to le(z, y) \tag{7}$$

$$(\forall x, y)\ x \to y \Rightarrow min(x) \to min(y) \tag{8}$$

$$(\forall y)\ le(0, s(y)) \to true \tag{9}$$

$$(\forall x, y)\ le(s(x), s(y)) \to le(x, y) \tag{10}$$

$$(\forall x)\ le(x, 0) \to false \tag{11}$$

$$(\forall x)\ min(cons(x, nil)) \to x \tag{12}$$

$$(\forall x, y, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* true \Rightarrow min(cons(x, xs)) \to x \tag{13}$$

$$(\forall x, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* false \Rightarrow min(cons(x, xs)) \to y \tag{14}$$

Variants of Term Rewriting Systems - CS-CTRSs

Example

$$
\begin{array}{rcl}
le(0, s(y)) & \to & true \\
le(s(x), s(y)) & \to & le(x, y) \\
le(x, 0) & \to & false \\
min(cons(x, nil)) & \to & x \\
min(cons(x, xs)) & \to & x \Leftarrow min(xs) \to^* y, le(x, y) \to^* true \\
min(cons(x, xs)) & \to & y \Leftarrow min(xs) \to^* y, le(x, y) \to^* false
\end{array}
$$

where $\mu(cons) = 1$ and $\mu(f) = \{1, \ldots, ar(f)\}$

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

(Re)$_\beta$ $\dfrac{s_1 \to^* t_1 \quad \ldots \quad s_n \to^* t_n}{\ell \to r}$ for $\beta : \ell \to r \Leftarrow s_1 \to^* t_1, \ldots, s_1 \to^* t_n \in \mathcal{R}$

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

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

First-Order Theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$

$$(\forall x)\ x \to^* x \tag{15}$$

$$(\forall x, y, z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z \tag{16}$$

$$(\forall x, y)\ x \to y \Rightarrow s(x) \to s(y) \tag{17}$$

$$(\forall x, y, z)\ x \to y \Rightarrow cons(x, z) \to cons(y, z) \tag{18}$$

$$(\forall x, y, z)\ x \to y \Rightarrow le(x, z) \to le(y, z) \tag{19}$$

$$(\forall x, y, z)\ x \to y \Rightarrow le(z, x) \to le(z, y) \tag{20}$$

$$(\forall x, y)\ x \to y \Rightarrow min(x) \to min(y) \tag{21}$$

$$(\forall y)\ le(0, s(y)) \to true \tag{22}$$

$$(\forall x, y)\ le(s(x), s(y)) \to le(x, y) \tag{23}$$

$$(\forall x)\ le(x, 0) \to false \tag{24}$$

$$(\forall x)\ min(cons(x, nil)) \to x \tag{25}$$

$$(\forall x, y, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* true \Rightarrow min(cons(x, xs)) \to x \tag{26}$$

$$(\forall x, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* false \Rightarrow min(cons(x, xs)) \to y \tag{27}$$

Variants of Term Rewriting Systems - Generalized TRSs

Example

$$
\begin{array}{rcl}
isEven(0) & & \\
isEven(s(s(n))) & & \Leftarrow isEven(n) \\
add(0,x) & \to & x \\
add(x,0) & \to & x \\
add(s(x),y) & \to & s(add(x,y)) \\
add(x,y) & \to & add(y,x) \\
test(x) & \to & even(x) \Leftarrow isEven(x) \\
test(x) & \to & odd(x) \Leftarrow isEven(s(x))
\end{array}
$$

(Re)$_\beta$ $\dfrac{B_1 \quad \ldots \quad B_n}{\ell \to r}$

for $\beta : \ell \to r \Leftarrow A_1, \ldots, A_n \in \mathcal{R}$

B_i is an atom or a rew. relation

(HC)$_\beta$ $\dfrac{A_1 \quad \ldots \quad A_n}{A}$

for $\alpha : A \Leftarrow A_1 \wedge \cdots \wedge A_n \in \mathcal{H}$

First-Order Theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$

$$(\forall x)\ x \to^* x \tag{28}$$

$$(\forall x, y, z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z \tag{29}$$

$$(\forall x, y)\ x \to y \Rightarrow s(x) \to s(y) \tag{30}$$

$$(\forall x, y, z)\ x \to y \Rightarrow add(x, z) \to add(y, z) \tag{31}$$

$$(\forall x, y, z)\ x \to y \Rightarrow add(z, x) \to add(z, y) \tag{32}$$

$$(\forall x, y)\ x \to y \Rightarrow test(x) \to test(y) \tag{33}$$

$$(\forall x)\ add(0, x) \to x \tag{34}$$

$$(\forall x)\ add(x, 0) \to x \tag{35}$$

$$(\forall x, y)\ add(s(x), y) \to s(add(x, y)) \tag{36}$$

$$(\forall x, y)\ add(x, y) \to add(y, x) \tag{37}$$

$$(\forall x)\ isEven(x) \Rightarrow test(x) \to even(x) \tag{38}$$

$$(\forall x)\ isEven(s(x)) \Rightarrow test(x) \to odd(x) \tag{39}$$

$$isEven(0) \tag{40}$$

$$(\forall x)\ isEven(x) \Rightarrow isEven(s(s(x))) \tag{41}$$

Predefined relations in infChecker

- One rewriting step (`->`)
- One CS-rewriting step (`\->`)
- Zero or one rewriting step (`->=`)
- Zero or one CS-rewriting step (`\->=`)
- Zero or more rewriting steps (`->*`)
- Zero or more CS-rewriting steps (`\->*`)
- One or more rewriting steps (`->+`)
- One or more CS-rewriting steps (`\->+`)
- Subterm (`|>=`) and strict subterm (`|>`)
- Joinability (`->*<-`)
- CS-joinability (`\->*<-/`)
- One convertibility step (`<->`)
- One CS-convertivility step (`<-/\->`)
- Zero or more convertibility steps (`<->*`)
- Zero or more CS-convertibility steps (`<-/\->*`)

Predefined Relations

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

(Inc) $$\frac{s \to t}{s \to^{=} t}$$

(Rf) $$\frac{}{x \trianglerighteq x}$$

$(\mathrm{Pr})_{f,i}$ $$\frac{}{f(x_1, \ldots, x_i, \ldots, x_k) \trianglerighteq x_i}$$

for $f \in \mathcal{F}^{(k)}$ and $1 \leq i \leq k$

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

(Inc) $$\frac{s \to^{*} t}{s \leftrightarrow t}$$

(Inc) $$\frac{t \to^{*} s}{s \leftrightarrow t}$$

Feasibility/Infeasibility

Definition

A f-condition $\bowtie(s_1, \ldots, s_n)$ is $(\mathbb{T}, \sigma)$-**feasible** if $\mathrm{Th}_{\bowtie} \vdash \bowtie(\sigma(s_1), \ldots, \sigma(s_n))$ holds; otherwise, it is $(\mathbb{T}, \sigma)$-**infeasible**. We also say that $\bowtie(s_1, \ldots, s_n)$ is $\mathbb{T}$-**feasible** (or $\mathrm{Th}_{\bowtie}$-feasible, or just feasible if no confusion arises) if it is $(\mathbb{T}, \sigma)$-feasible for some substitution σ; otherwise, we call it **infeasible**.

A sequence F is $\mathbb{T}$-feasible (or just **feasible**) iff there is a substitution σ such that, for all $\gamma \in$ F, γ is $(\mathbb{T}, \sigma)$-feasible. Note that $()$ is trivially feasible. A *goal* $\mathcal{G}$ is **feasible** iff it contains a f-sequence F $\in \mathcal{G}$. Now, $\{\}$ is trivially **infeasible**.

Feasibility Framework

Feasibility Framework

fProblem

An **fProblem** τ is a pair $\tau = (\mathbb{T}, \mathcal{G})$, where $\mathbb{T}$ is a theory and $\mathcal{G}$ is a f-goal. The **fProblem** τ is **feasible** if $\mathcal{G}$ is **$\mathbb{T}$-feasible**; otherwise it is **$\mathbb{T}$-infeasible**.

fProcessor

An **fProcessor** P is a partial function from fProblems into sets of fProblems. Alternatively, it can return "yes".

- An fProcessor P is **sound** if for all $\tau \in \mathcal{D}om(\mathsf{P})$, τ is feasible whenever either $\mathsf{P}(\tau) =$"yes" or $\exists \tau' \in \mathsf{P}(\tau)$, such that τ' is feasible.
- An fProcessor P is **complete** if for all $\tau \in \mathcal{D}om(\mathsf{P})$, τ is infeasible whenever $\forall \tau' \in \mathsf{P}(\tau)$, τ' is infeasible.

Feasibility Proof Tree (1/2)

Definition

Let τ be an fProblem. A **feasibility proof tree** $\mathcal{T}$ for τ is a tree whose inner nodes are labeled with fProblems and the leaves are labeled with **fProblems**, "yes" or "no".

The root of $\mathcal{T}$ is labeled with τ and for every inner node τ', there is a fProcessor P such that $\tau' \in \mathcal{D}om(\mathsf{P})$ and:

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

Feasibility Proof Tree (2/2)

Theorem

Let $\mathcal{T}$ be a feasibility proof tree for $\tau_I = (\mathbb{T}, \mathcal{G})$. Then:

1. if all leaves in $\mathcal{T}$ are labeled with “no” and all involved fProcessors are **complete** for the fProblems they are applied to, then $\mathcal{G}$ is $\mathbb{T}$**-infeasible**.
2. if $\mathcal{T}$ has a leaf labeled with “yes” and all fProcessors in the path from τ_I to the leaf are **sound** for the fProblems they are applied to, then $\mathcal{G}$ is $\mathbb{T}$**-feasible**.

Splitting fProcessor

Theorem

Let $\tau = (\mathbb{T}, \mathcal{G})$ be an fProblem and $\mathsf{F} \in \mathcal{G}$ are the f-sequences of $\mathcal{G}$. The fProcessor $\mathsf{P}^{\mathsf{Spl}}$ given by $\mathsf{P}^{\mathsf{Spl}}(\tau) = \{(\mathbb{T}, \mathsf{F}) \mid \mathsf{F} \in \mathcal{G}\}$ is **sound** and **complete**.

Satisfiability fProcessor

Theorem

Let $\tau = (\mathbb{T}, \mathcal{G})$ be an fProblem with $\mathcal{G} = (\bowtie_i (s_1, \ldots, s_k))_{i=1}^n$. Let $\mathcal{A}$ be a structure such that $\mathcal{A} \neq \varnothing$ and $\mathcal{A} \models \overline{\mathcal{R}} \cup \{\neg(\exists\vec{x}) \bigwedge_{i=1}^n \bowtie_i (s_1, \ldots, s_n)\}$. The fProcessor $\mathsf{P}^{\mathsf{Sat}}$ given by $\mathsf{P}^{\mathsf{Sat}}(\tau) = \varnothing$ is **sound** and **complete**.

Example 903.trs - Satisfiability fProcessor

$$(Rf), (T), (C)_{f,i}$$
$$(\forall y)\ le(0, s(y)) \to true$$
$$(\forall x, y)\ le(s(x), s(y)) \to le(x, y)$$
$$(\forall x)\ le(x, 0) \to false$$
$$(\forall x)\ min(cons(x, nil)) \to x$$
$$(\forall x, y, xs)\ min(xs) \to^* y \wedge$$
$$le(x, y) \to^* true \Rightarrow min(cons(x, xs)) \to x$$
$$(\forall x, xs)\ min(xs) \to^* y \wedge$$
$$le(x, y) \to^* false \Rightarrow min(cons(x, xs)) \to y$$
$$\neg((\exists x, y) min(nil) \to^* x \wedge le(y, x) \to^* true)$$

- AGES output:

```
Domain: |N U {-1}

Function Interpretations:
[le(x,y)] = y         [0] = 1
[min(x)] = x          [false] = 1
[s(x)] = 1 + x        [nil] =  - 1
[cons(x,y)] = 1 + x + y
[true] = 0

Predicate Interpretations:
x ->* y <=> (x >= y)
x -> y <=> ((x >= y) /\
            (1 + y >= 0))
```

Example 903.trs - Satisfiability fProcessor

$$(Rf), (T), (C)_{f,i}$$

$$(\forall y)\ le(0, s(y)) \to true$$

$$(\forall x, y)\ le(s(x), s(y)) \to le(x, y)$$

$$(\forall x)\ le(x, 0) \to false$$

$$(\forall x)\ min(cons(x, nil)) \to x$$

$$(\forall x, y, xs)\ min(xs) \to^* y \wedge$$

$$le(x, y) \to^* true \Rightarrow min(cons(x, xs)) \to x$$

$$(\forall x, xs)\ min(xs) \to^* y \wedge$$

$$le(x, y) \to^* false \Rightarrow min(cons(x, xs)) \to y$$

$$(\forall x, y)\ \neg(min(nil) \to^* x) \vee \neg(le(y, x) \to^* true)$$

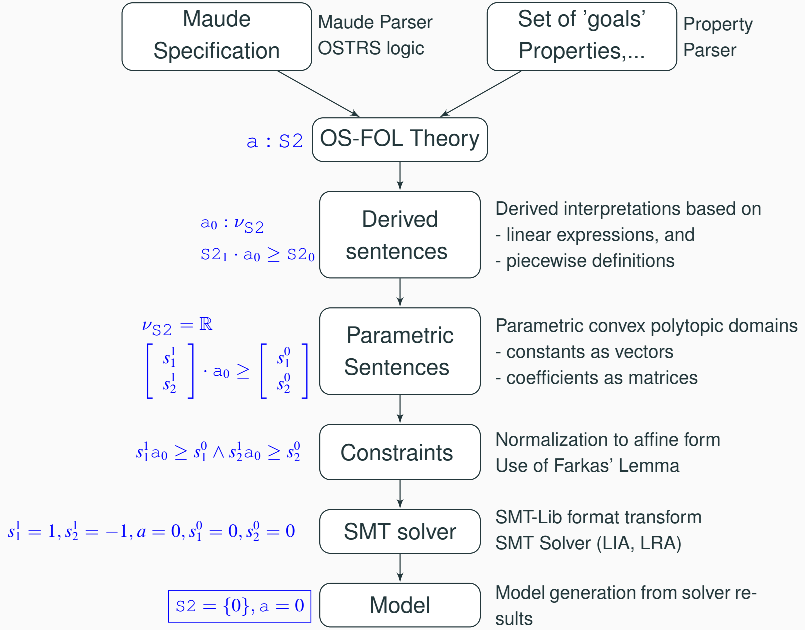
- AGES output:

```
Domain: |N U {-1}

Function Interpretations:
[le(x,y)] = y        [0] = 1
[min(x)] = x         [false] = 1
[s(x)] = 1 + x       [nil] =  - 1
[cons(x,y)] = 1 + x + y
[true] = 0

Predicate Interpretations:
x ->* y <=> (x >= y)
x -> y <=> ((x >= y) /\
            (1 + y >= 0))
```

AGES: Automatic Generation of OS-FOL Models



Provability fProcessor

Theorem

Let $\tau = (\mathbb{T}, \mathcal{G})$ be an fProblem with $\mathcal{G} = (\bowtie_i (s_1, \ldots, s_k))_{i=1}^{n}$ such that $\overline{\mathcal{R}} \vdash (\exists \vec{x}) \bigwedge_{i=1}^{n} \bowtie_i (s_1, \ldots, s_k)$ holds. The fProcessor $\mathsf{P}^{\mathsf{Prov}}$ given by $\mathsf{P}^{\mathsf{Prov}}(\tau) =$ "yes" is **sound** and **complete**.

Provability fProcessor

Example 836.trs

$$
\begin{array}{rcl}
le(0, s(y)) & \to & true \\
le(s(x), s(y)) & \to & le(x, y) \\
le(x, 0) & \to & false \\
min(cons(x, nil)) & \to & x \\
min(cons(x, xs)) & \to & x \Leftarrow le(x, min(xs)) \to^* true \\
min(cons(x, xs)) & \to & min(xs) \Leftarrow le(x, min(xs)) \to^* false \\
min(cons(x, xs)) & \to & min(xs) \Leftarrow min(xs) \to^* x
\end{array}
$$

Goal

$$le(x, min(xs)) \to^* false, min(xs) \to^* x$$

Example 836.trs - Provability fProcessor (Prover9 output)

```
(1) (x ->* x) # [reflexivity]
(2) -(x -> y) | -(y ->* z) | (x ->* z) # [transitivity]
(7) -(x -> y) | le(z,x) -> le(z,y) # [congruence]
(11) le(x,0) -> false # [replacement]
(12) min(cons(x,nil)) -> x # [replacement]
(16) (exists x (exists y (le(x,min(y)) ->* false &
     min(y) ->* x))) # [goal]
------------------------------------------------------------
(17) -(le(x,min(y)) ->* false) |
     -(min(y) ->* x) # [deny(16)]
(18) le(x,0) ->* false [ur(2,11,1)]
(19) -(le(min(x),min(x)) ->* false) [resolve(17,1)]
(20) -(le(min(x),min(x)) -> y) |
     -(y ->* false) [resolve(19,2)]
(21) -(le(min(x),y) ->* false) |
     -(min(x) -> y) [resolve(20,7)]
(22) -(le(min(cons(x,nil)),x) ->* false) [resolve(21,12)]
(23) $F [resolve(22,18)]
```

Narrowing on f-Conditions fProcessor

Definition

Let $\tau = (\mathbb{T}, \mathcal{G})$ be an fProblem, $s_i \to^* t_i \in \mathcal{G}$, and $\mathcal{N} \subseteq \overline{\mathcal{N}}(\mathbb{T}, \mathcal{G}, i)$ finite. $\mathsf{P}^{\mathsf{NarrCond}}$ is given by $\mathsf{P}^{\mathsf{NarrCond}}(\tau) = \{(\mathbb{T}, \mathcal{N})\}$.

Theorem

$\mathsf{P}^{\mathsf{NarrCond}}$ is sound. If $\mathcal{N} = \overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, i)$ and $s_i \to^* t_i \in \mathcal{G}$ is such that s_i and t_i do *not* unify and either s_i is *ground* or (1) $\mathit{NRules}(\mathcal{R}, s_i)$ is a TRS and (2) s_i is linear, then $\mathsf{P}^{\mathsf{NarrCond}}$ is complete.

Example 896.trs - Narrowing fProcessor

Example (896.trs)

$$
\begin{array}{ll}
add(0,x) \to x & add(s(x),y) \to s(add(x,y)) \\
 & \quad div(minus(x,y),s(y)) \to^* q \\
lte(0,y) \to true & lte(s(x),0) \to false \\
lte(s(x),s(y)) \to lte(x,y) & minus(0,s(y)) \to 0 \\
minus(s(x),s(y)) \to minus(x,y) & minus(x,0) \to x \\
mod(0,y) \to 0 & mod(x,0) \to x \\
mod(x,s(y)) \to mod(minus(x,s(y)),s(y)) & \Leftarrow lte(s(y),x) \to^* true \\
mod(x,s(y)) \to x & \Leftarrow lte(s(y),x) \to^* false \\
mult(0,y) \to 0 & mult(s(x),y) \to add(mult(x,y),y) \\
div(s(x),s(y)) \to 0 & \Leftarrow lte(s(x),y) \to^* true \\
div(s(x),s(y)) \to s(q) & \Leftarrow lte(s(x),y) \to^* false, \\
div(0,s(x)) \to 0 & power(x,0) \to s(0) \\
power(x,n) \to mult(mult(y,y),s(0)) & \Leftarrow n \to^* s(n'), mod(n,s(s(0))) \to^* 0, \\
 & \quad power(x,div(n,s(s(0)))) \to^* y \\
power(x,n) \to mult(mult(y,y),x) & \Leftarrow n \to^* s(n'), mod(n,s(s(0))) \to^* s(z), \\
 & \quad power(x,div(n,s(s(0)))) \to^* y
\end{array}
$$

Example (896.trs)

$add(0,x) \to x$	$add(s(x),y) \to s(add(x,y))$
	$div(minus(x,y),s(y)) \to^* q$
$lte(0,y) \to true$	$lte(s(x),0) \to false$
$lte(s(x),s(y)) \to lte(x,y)$	$minus(0,s(y)) \to 0$
$minus(s(x),s(y)) \to minus(x,y)$	$minus(x,0) \to x$
$mod(0,y) \to 0$	
$mod(x,s(y)) \to$	
$mod(x,s(y)) \to$	
$mult(0,y) \to 0$	$x,y),y)$
$div(s(x),s(y)) \to 0$	$\Leftarrow lte(s(x),y) \to^* true$
$div(s(x),s(y)) \to s(q)$	$\Leftarrow lte(s(x),y) \to^* false,$
$div(0,s(x)) \to 0$	$power(x,0) \to s(0)$
$power(x,n) \to mult(mult(y,y),s(0))$	$\Leftarrow n \to^* s(n'), mod(n,s(s(0))) \to^* 0,$
	$power(x,div(n,s(s(0)))) \to^* y$
$power(x,n) \to mult(mult(y,y),x)$	$\Leftarrow n \to^* s(n'), mod(n,s(s(0))) \to^* s(z),$
	$power(x,div(n,s(s(0)))) \to^* y$

Feasibility Conditions

$$\{lte(s(x),0) \to^* true\}$$

Example 896.trs - Narrowing fProcessor

Example (896.trs)

$$add(0, x) \to x$$

$$add(s(x), y) \to s(add(x, y))$$

$$div(minus(x, y), s(y)) \to^* q$$

$$minus(s(x), s(y)) \to minus(x, y)$$

$$minus(x, 0) \to x$$

$$mod(0, y) \to 0$$

$$mod(x, 0) \to x$$

$$mod(x, s(y)) \to$$

$$mod(x, s(y)) \to$$

$$mult(0, y) \to 0$$

$$x, y), y)$$

$$div(s(x), s(y)) -$$

$$div(s(x), s(y)) -$$

$$div(0, s(x)) \to 0$$

$$power(x, 0) \to s(0)$$

$$power(x, n) \to mult(mult(y, y), s(0)) \Leftarrow n \to^* s(n'), mod(n, s(s(0))) \to^* 0, power(x, div(n, s(s(0)))) \to^* y$$

$$power(x, n) \to mult(mult(y, y), x) \Leftarrow n \to^* s(n'), mod(n, s(s(0))) \to^* s(z), power(x, div(n, s(s(0)))) \to^* y$$

Feasibility Conditions

$$\{false \to^* true\}$$

Example 896.trs - Satisfiability fProcessor (Mace4 output)

```
Domain: {0,1}

Function Interpretations:
[false] = 0          [true] = 1          [0] = 0
[s](0) = 0           [s](1) = 1          [add](0,0) = 0
[add](0,1) = 1       [add](1,0) = 0      [add](1,1) = 1
[div](0,0) = 0       [div](0,1) = 0      [div](1,0) = 0
[div](1,1) = 0       [lte](0,0) = 1      [lte](0,1) = 1
[lte](1,0) = 1       [lte](1,1) = 1      [minus](0,0) = 0
[minus](0,1) = 0     [minus](1,0) = 1    [minus](1,1) = 1
[mod](0,0) = 0       [mod](0,1) = 0      [mod](1,0) = 1
[mod](1,1) = 1       [mult](0,0) = 0     [mult](0,1) = 1
[mult](1,0) = 0      [mult](1,1) = 1     [power](0,0) = 0
[power](0,1) = 0     [power](1,0) = 1    [power](1,1) = 1
Predicate Interpretations:
0 ->* 0 = true       0 ->* 1 = false     1 ->* 0 = true
1 ->* 1 = true       0 -> 0 = true       0 -> 1 = false
1 -> 0 = true        1 -> 1 = true
```

Leading Example

Strong Joinability

Two terms s and t are **strongly joinable** if there are terms u and u' such that $s \to^{=} u \;\; {}^{*}\!\!\leftarrow t$ and $s \to^{*} u' \;\; {}^{=}\!\!\leftarrow t$, where $\to^{=}$ is $\to \cup =$

CCPs

$$
\begin{array}{ll}
\pi_{(34),\Lambda,(35)}: & \langle 0, 0 \rangle \\
\pi_{(34),\Lambda,(37)}: & \langle add(x, 0), x \rangle \\
\pi_{(35),\Lambda,(37)}: & \langle add(0, x), x \rangle \\
\pi_{(38),\Lambda,(39)}: & \langle odd(x), even(x) \rangle \Leftarrow isEven(x), isEven(s(x))
\end{array}
$$

Implementation

Implementation

- Written in Haskell. Consists of 28 new Haskell modules with more than 3500 lines of code.
- Accessible via CoCoWeb3 platform.
- Input format: extended version of TPDB format, allowing the declaration of a list of feasibility conditions using the reserved word `CONDITION`.

Strategy

1. we apply modularity results with P^{Spl};
2. we try to prove feasibility using P^{Prov};
3. if P^{Prov} fails, we apply P^{Sat};
4. if P^{Sat} fails, we apply $P^{NarrCond}$;
5. if $P^{NarrCond}$ succeeds and modifies the feasibility sequence, we go to Item 3, otherwise we return MAYBE.

Uses of Infeasibility

Infeasible rules

Use of infeasibility

Disable/remove the use of **conditional rules** in reductions.

Example

$$\begin{aligned} b &\to c \\ a &\to b \Leftarrow a \to^* b \end{aligned}$$

Infeasible rules

Use of infeasibility

Discard **conditional dependency pairs** $u \to v \Leftarrow c$ in the analysis of operational termination of GTRSs.

Example

$$DP_H = \{A \to A \Leftarrow b \to x, c \to x\}$$

$$DP_V = \{A \to B, A \to C \Leftarrow b \to x\}$$

$$R =$$

$$\begin{array}{rcl} \{a & \to & a \Leftarrow b \to x, c \to x \\ b & \to & d \Leftarrow d \to x, e \to x \\ c & \to & d \Leftarrow d \to x, e \to x\} \end{array}$$

Infeasible rules

Use of infeasibility

Discard **conditional critical pairs** $u \downarrow v \Leftarrow c$ in the analysis of confluence of GTRSs.

Conditional critical pair

$$< even(x), odd(x) > \Leftarrow isEven(x), isEven(s(x))$$

Infeasible rules

Use of infeasibility

Prove **root-stability** of a term t (t cannot be reduced to a redex)

Goal

$$t \to^* \ell_1 \vee \cdots \vee t \to^* \ell_n$$

Infeasible rules

Use of infeasibility

Prove **irreducibility** of ground terms t (which is undecidable for CTRSs)

Goal

$$t \to x$$

Infeasible rules

Use of infeasibility

Prove the **non-joinability** of terms s and t

Goal

$$s \to^* x, t \to^* x$$

Infeasible rules

Use of infeasibility

Discard **arcs** in the **dependency graphs** that are obtained during the analysis of termination using dependency pairs.

Goal

$$t_i \to^* s_j$$

Infeasible rules

Use of infeasibility

Obtain **feasibility of conditional variable pairs**.

Example

$$\begin{array}{rcl} a & \to & b \\ f(x) & \to & c \Leftarrow x \to^* a \end{array}$$

Critical pair

$$< f(x'), c > \Leftarrow x \to x', x \to^* a$$

Goal

$$x \to x', x \to^* a, f(x') \to^* z, c \to^* z$$

Infeasible rules

Use of infeasibility

Prove **feasibility** of combinations of different relations.

Infeasible rules

Use of infeasibility

Teaching!

Future Work

Future Work

- We would like to **extend** our tools CONFident and MU-TERM to deal with GTRSs.