infChecker at the 2021 Confluence Competition*

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

[1] Universidad Politécnica de Madrid, Madrid, Spain
r.gutierrez@upm.es
[2] VRAIN, Universitat Politècnica de València, Valencia, Spain
slucas@dsic.upv.es
mvitvic@posgrado.upv.es

1 Overview

infChecker 1.0 is a tool for checking *(in)feasibility* of goals $\mathcal{G} = \{F_i\}_{i=1}^m$ where $F_i = (s_{ij} \bowtie_{ij} t_{ij})_{i=1}^{n_i}$ and $\bowtie_{ij} \in \{\to, \to^*, \to^+, \hookrightarrow, \hookrightarrow^*, \hookrightarrow^+, \rhd, \trianglerighteq, \downarrow, \Downarrow, \leftrightarrow, \leftrightarrow\!\!\!\!\hookrightarrow, \leftrightarrow^*, \leftrightarrow\!\!\!\!\hookrightarrow^*\}$ where predicates $\bowtie_{ij}$ represent binary relations on terms (most of them well-known or easy generalizations of well-known relations) defined by provability of goals $s \bowtie_{ij} t$ with respect to a *first-order theories* $\mathsf{Th}_{\bowtie_{ij}}$ [1, 2].

The tool is available here: http://zenon.dsic.upv.es/infChecker/. It is written in Haskell and implements the *Feasibility Framework* [1], that describes:

- *f-problems*, by a tuple $\tau = (\mathbb{T}, \mathcal{G})$, where $\mathbb{T}$ is a $\mathbb{P}$-indexed theory, $\mathbb{P}$ is a set of predicates and $\mathcal{G}$ is a sequence of $\mathbb{T}$-conditions. We say that τ is feasible if $\mathcal{G}$ is $\mathbb{T}$-feasible; otherwise, it is infeasible.
- *f-processor*, as partial functions from f-problems into set of f-problems. Alternatively, it can return "yes". An f-processor $\mathsf{P}(\tau)$ is sound iff $\mathsf{P}(\tau) =$ "yes" or exists $\tau' \in \mathsf{P}(\tau)$ such that τ' is feasible. An f-processor $\mathsf{P}(\tau)$ is complete iff τ is infeasible whenever $\mathsf{P}(\tau) \neq$ "yes" and for all $\tau' \in \mathsf{P}(\tau)$, τ' is infeasible.

We implement five processors: $\mathsf{P}^{\mathsf{Sat}}$ to prove infeasibility, $\mathsf{P}^{\mathsf{Prov}}$ to prove feasibility, P^{NC} to apply narrowing, $\mathsf{P}^{\mathsf{Spl}}$ to decompose a goal and P^{UR} to simplify the set of rules.

Our feasibility problems accepts different variants of TRS: Term Rewriting Systems, Conditional Term Rewriting Systems, Context-Sensitive Term Rewriting Systems and Conditional Context-Sensitive Term Rewriting Systems.

By using the proper relation (straight arrows for rewriting and curly arrows for context-sensitive rewriting) we can use and combine different forms of rewriting.

References

[1] R. Gutiérrez and S. Lucas. Automatically Proving and Disproving Feasibility Conditions. In N. Peltier and V. Sofronie-Stokkermans, editor, *Proc. of IJCAR'2020*, LNCS 12167:416–435. Springer, 2020.

[2] S. Lucas and R. Gutiérrez. Use of Logical Models for Proving Infeasibility in Term Rewriting. *Information Processing Letters*, 136:90–95, 2018.

*Partially supported by the EU (FEDER) and the Spanish MCIU under grant RTI2018-094403-B-C32 and by the Spanish Generalitat Valenciana under grant PROMETEO/2019/098.