International Journal on Software Tools for Technology Transfer (2021) 23:905–916
https://doi.org/10.1007/s10009-021-00620-4

COMPETITIONS AND CHALLENGES

Special Issue: TOOLympics 2019

CoCo 2019: report on the eighth confluence competition

Aart Middeldorp[1] · **Julian Nagele**[2] · **Kiraku Shintani**[3]

Accepted: 6 May 2021 / Published online: 28 May 2021

Abstract
We report on the 2019 edition of the Confluence Competition, a competition of software tools that aim to prove or disprove confluence and related (undecidable) properties of rewrite systems automatically.

Keywords Confluence · Term rewriting · Automation · Mechanized reasoning · Software competition

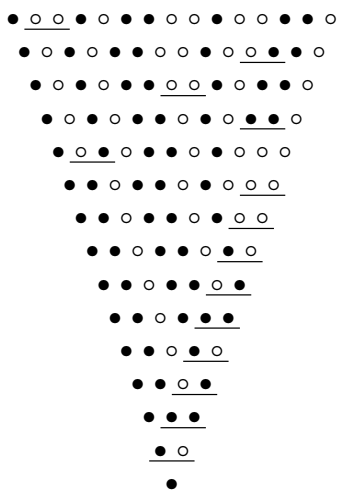
1 Introduction

Term rewriting is a Turing-complete model of computation, which underlies much of declarative programming and automated theorem proving. Confluence provides a general notion of determinism and has been conceived as one of the central properties of rewriting. A rewrite system R is a set of directed equations, so-called rewrite rules, which induces a rewrite relation $\to_R$ on terms. It is called confluent if for all terms s, t and u such that $s \to_R^* t$ and $s \to_R^* u$ there exists a term v such that $t \to_R^* v$ and $u \to_R^* v$. Confluence is equivalent to the Church–Rosser property, introduced in 1936 by Church and Rosser [8] to show the consistency of the λI-calculus, and guarantees that normal forms (which are terms t such that $t \to_R u$ for no term u) are unique.

We provide two examples and refer to standard textbooks for comprehensive surveys [7,31,44]. The first rewrite system describes the *Coffee Bean Game*, a variant of the *Grecian Urn* described in [11].

Example 1 Coffee beans come in two kinds called black (●) and white (○). A two-player game starts with a random sequence of black and white beans. In a move, a player must take two adjacent beans and put back one bean, according to the following set of rules R_1:

The player who puts the last white bean wins. For instance, the following is a valid game:



In this case the player who started won, since the last white bean was put in the 13th move. It turns out that the moves of the players do not affect the outcome of the game, because the rules constitute a confluent system; the outcome depends solely on the initial configuration.

The second example is attributed to Henk Barendregt in [17].

✉ Aart Middeldorp
aart.middeldorp@uibk.ac.at

Julian Nagele
mail@jnagele.net

Kiraku Shintani
s1820017@jaist.ac.jp

[1] Department of Computer Science, University of Innsbruck, Innsbruck, Austria

[2] Bank of America, London, UK

[3] School of Information Science, JAIST, Ishikawa, Japan

Example 2 Consider the rewrite system R consisting of the following three rewrite rules

$$\mathsf{c} \to \mathsf{g}(\mathsf{c}) \qquad \mathsf{f}(x,x) \to \mathsf{a} \qquad \mathsf{g}(x) \to \mathsf{f}(x,\mathsf{g}(x))$$

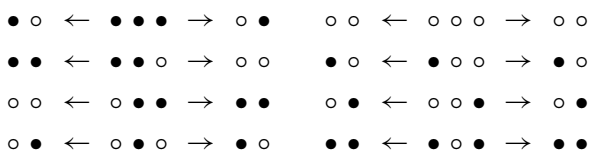
The constant c rewrites in four steps to a: $\mathsf{c} \to_R \mathsf{g}(\mathsf{c}) \to_R \mathsf{f}(\mathsf{c},\mathsf{g}(\mathsf{c})) \to_R \mathsf{f}(\mathsf{g}(\mathsf{c}),\mathsf{g}(\mathsf{c})) \to_R \mathsf{a}$. Hence $\mathsf{c} \to_R^* \mathsf{a}$ and thus also $\mathsf{c} \to_R \mathsf{g}(\mathsf{c}) \to_R^* \mathsf{g}(\mathsf{a})$. Therefore, c rewrites to both a and $\mathsf{g}(\mathsf{a})$. The constant a is a normal form as none of the rewrite rules applies. The term $\mathsf{g}(\mathsf{a})$ admits exactly one (infinite) rewrite sequence:

$$\begin{aligned}\mathsf{g}(\mathsf{a}) &\to_R \mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{a}))\\ &\to_R \mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{a})))\\ &\to_R \mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{a}))))\\ &\to_R \cdots\end{aligned}$$

Since the term a is not reached, R is not confluent. The weaker property of unique normal forms (both with respect to conversion and reduction) is satisfied.

Another property of rewrite systems that has received much attention, including a designated competition (termCOMP), is termination. [1] A rewrite system R is terminating if its rewrite relation $\to_R$ is well-founded. The rewrite system in Example 1 is terminating because in each step the number of beans decreases.

For terminating rewrite systems, confluence is decidable. The decision procedure (Knuth and Bendix [18]) is a landmark result in rewriting and implemented in all confluence tools. It amounts to checking whether all critical pairs are joinable. Critical pairs are formed by overlapping left-hand sides of rewrite rules to create (for finite rewrite systems) a finite number of local peaks $t \;{}_R\!\leftarrow s \to_R u$. In Example 1, we have the following critical peaks:



One easily checks that the resulting critical pairs (the end points of the peaks) are joinable, meaning that they can be rewritten to the same bean configuration. Hence, confluence is established.

In general, confluence and termination are undecidable properties of rewrite systems. As a consequence, no single automatable technique is sufficient to determine the status of every possible input problem. Tools implement a number of different techniques that are suitably combined to determine the status of a problem. Often, this falls short, also because of imposed time limits in competitions.

The remainder of this competition report is organized as follows. In the next section, we present a short overview of the organization of CoCo, including a description of the supporting infrastructure. The competition categories of CoCo 2019 are described in Sect. 3, and Sect. 4 briefly describes the participating tools. Section 5 presents the results of CoCo 2019, and we conclude in Sect. 6 with ideas for future editions of CoCo.

2 Competition

The focus on confluence research has shifted toward automation in the past decade. To stimulate these developments, the Confluence Competition (CoCo)[2] has been set up in 2012. Since its creation with 4 tools competing in 2 categories, CoCo has grown steadily and featured 12 categories in 2019, ranging from confluence of various rewrite formalisms to commutation and infeasibility. These are described in the next section. Since 2012 a total of 21 tools have participated in CoCo. Many of the tools participated in multiple categories. Tools operate on problems from the online database of *confluence problems* (COPS)[3] and a number of *secret* problems submitted shortly before the competition, in a format suitable for the category in which the tools participate. For each category, 100 problems consisting of all secret problems and a random selection from COPS are collected.

CoCo is executed on the cross-community competition platform StarExec [43]. Tool authors upload their tools to StarExec two weeks before the competition, after which a test run is conducted involving a few selected problems for each category. This allows tool authors to fix last-minute bugs before the live competition. The steering committee of CoCo is responsible for running the competition on StarExec and exporting the results. Each tool has access to a single node and is given 60 s per problem. For a given problem, tools must answer YES (proved) or NO (disproved), followed by a justification that is understandable by a human expert; any other output signals that the tool could not determine the status of the problem. As human expertise is insufficient to guarantee correctness, CoCo supports certification categories, in which tool output is checked by an independent and formally verified certifier. The possibility in StarExec to reserve a large number of computing nodes allows to complete CoCo within a single slot of a workshop or conference. This live event of CoCo is shared with the audience via the online service *LiveView* [16] which continuously polls new results from StarExec while the competition is running. A

[1] http://termination-portal.org/wiki/Termination_Competition.

[2] http://project-coco.uibk.ac.at/.

[3] https://cops.uibk.ac.at/.

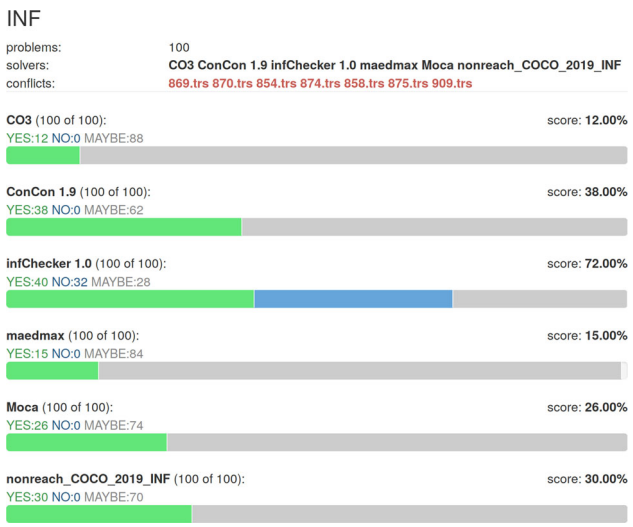


Fig. 1 Part of the LiveView of CoCo 2019 upon completion

screenshot of part of the LiveView of CoCo 2019 is shown in Fig. 1. Since all categories deal with undecidable problems, and developing software tools is error-prone, YES/NO conflicts (which are situations where tools produce contradictory answers) appear once in a while. The real-time display of conflicts allows the CoCo steering committee to take action before winners are announced. Soon after each competition, the results are made available from the results page.[4] A few weeks after each live competition, there is a full run of tools on all eligible problems in the COPS database. Authors of tools with incorrect results have the possibility to submit a corrected version for the full run.

COPS

All problems in CoCo are selected from COPS, an online database for confluence and related properties in term rewriting. At the time of writing, COPS contains 1155 problems, including 471 collected from the literature. The problems are numbered consecutively starting from COPS #1. COPS supports several formats, to cater for the various CoCo categories. Via its web interface, everyone can retrieve and download problems and also upload new problems. The interface is designed in a way that novice users can easily learn problem formats. At the same time experts and tool builders can conveniently retrieve problem sets for their research and experiments. The former is achieved by *syntax highlighting*; for the latter a *tagging* mechanism is used. Tags are combined into queries for selecting problem sets. Different kinds of tags are supported. On the one hand, properties of rewrite systems like left-linearity, groundness, and termination are useful to filter the database for those problems that are supported by a particular tool or technique. These include tags to distinguish the different input formats, which are automatically assigned when problems are submitted. For example, "`trs !confluent !non_confluent`" is the query to select all first-order rewrite systems whose confluence status is unknown, meaning that no tool produced a YES or NO answer. (At CoCo 2019 this query returned 292 problems. If we include the secret problems, the number is 299.) A second category of tags refers to problems that were used in full runs of CoCo. The `literature` tag is assigned to problems that appear in the literature, which includes papers presented at informal workshops like the International Workshop on Confluence and Ph.D. theses.

The data in COPS consist of problems and tags. Most of the tag files are generated automatically or updated by a collection of scripts that call external tools. To prevent duplicate problems in COPS, a *duplicate checker* is used, which is based on a program that transforms problems into a canonical form which is invariant under permutation of rules and renaming of function symbols. [5] Currently, only problems in the basic TRS format (first-order, no conditions, no sorts) are supported.

CoCoWeb

Most of the tools that participate in CoCo can be downloaded, installed, and run on one's local machine, but this can be a painful process. Only few confluence tools—we are aware of CO3 [29], ConCon [41], and CSI [27,48]—provide a convenient web interface to easily test the status of a confluence problem that is provided by the user. In [16] CoCoWeb[6] is presented, a web interface to execute confluence tools on confluence problems. This provides a single entry point to all tools that participate in CoCo. The typical use of CoCoWeb is to test whether a given confluence problem is known to be confluent or not. This is useful when preparing or reviewing an article, preparing or correcting exams about term rewriting, and when contemplating submitting a challenging problem to COPS. In particular, CoCoWeb is useful when crafting or looking for examples to illustrate a new technique. Using CoCoWeb on the rewrite system from Example 2 (COPS #47), we learn that (automatically) disproving confluence is much harder than showing unique normal forms (UNC); only a single confluence tool (CSI) answers NO on this problem (and only since 2018). This answer is certified by CeTA (see the description under CPF-TRS in Sect. 3).

[4] http://project-coco.uibk.ac.at/results/.

[5] https://github.com/haskell-rewriting/canonical-trs.

[6] http://cocoweb.uibk.ac.at/.

3 Categories

In this section, we briefly describe the 12 categories of CoCo 2019. For each category, we list the participating tools, and for most we provide one or two example problems.

TRS

The category TRS is about confluence of *first-order term rewriting* and has been part of CoCo from the very beginning. We give two examples. The first one

```
COPS #7
(VAR x)
(RULES
  f(f(x)) -> g(x)
)
```

is not confluent because the peak $\mathsf{f}(\mathsf{g}(x)) \leftarrow \mathsf{f}(\mathsf{f}(\mathsf{f}(x))) \rightarrow \mathsf{g}(\mathsf{f}(x))$ involves different normal forms. The second example

```
COPS #104
(VAR x y z)
(RULES
  Ap(Ap(Ap(S,x),y),z) -> Ap(Ap(x,z),Ap(y,z))
  Ap(Ap(K,x),y) -> x
  Ap(I,x) -> x
)
```

is Combinatory Logic, which is confluent because it satisfies the orthogonality criterion. In 2019, three tools contested the TRS category: ACP, CoLL-Saigawa, and CSI.

CPF-TRS

CPF-TRS is a category for *certified* confluence proofs. CPF stands for Certification Problem Format, [7] an extendable format to express not only confluence but also termination and complexity proofs of first-order rewrite systems [37]. The purpose of the certification categories (CPF-TRS and CPF-CTRS) is to ensure that tools produce correct answers. In these categories, tools have to produce certified proofs with their answers. The predominant approach to achieve this uses a combination of a confluence prover and independent certifier. First, the confluence prover analyzes confluence as usual, restricting itself to criteria supported by the certifier. If it is successful, the prover outputs its proof in CPF, which is then checked by the certifier. In our case, this is CeTA [45], a state-of-the-art certifier for rewriting techniques generated from IsaFoR, [8] a formalization of first-order term rewriting in the Isabelle/HOL proof assistant [28]. Consequently, certificates must be expressed in CPF. Also this category has been part of CoCo from 2012. For CoCo 2019 the tools ACP and CSI teamed up with CeTA.

CTRS and CPF-CTRS

The categories CTRS and CPF-CTRS, introduced respectively, in 2014 and 2015, are concerned with (certified) confluence of *conditional term rewriting*, a formalism in which rewrite rules come equipped with conditions that are evaluated recursively using the rewrite relation.

```
COPS #308
(CONDITIONTYPE ORIENTED)
(VAR x)
(RULES
  not(x) -> false | x == true
  not(x) -> true  | x == false
)
```

The declaration `(CONDITIONTYPE ORIENTED)` in the above example problem specifies that the conditions (`x == true` and `x == false`) of the rules are interpreted as reachability ($\rightarrow^*$); a term `not(t)` can be rewritten to `false` using the first rule provided the argument term `t` rewrites to `true`. The competition restricts to this kind of conditional rewriting since the tools do so. In 2019, three tools contested the CTRS category: ACP, CO3, and ConCon. The combination of ConCon and CeTA was the only participant in the CPF-CTRS category.

HRS

The HRS category, introduced in 2015, deals with confluence of *higher-order rewriting*, i.e., rewriting with binders and functional variables, like in the following example:

```
(FUN
  f : o -> o -> o
  s : o -> o
  a : o
  b : o
  mu : (o -> o) -> o
)
(VAR
  x : o
  Z : o -> o
)
(RULES
  f x x -> a,
  f x (s x) -> b,
  mu (\x. Z x) -> Z (mu (\x. Z x))
)
```

[7] http://cl-informatik.uibk.ac.at/software/cpf/.

[8] http://cl-informatik.uibk.ac.at/software/ceta/.

Here, `Z` is a higher-order variable, which is apparent from the variable declaration `Z : o -> o`. The example is not confluent because the term

```
f (mu (\x. s x)) (mu (\x. s x))
```

can be rewritten to both `a` and `b`. The format supported by CoCo goes back to the higher-order rewrite systems of Mayr and Nipkow [21], with small modifications for increased readability. In 2019, the tool CSI^ho was the only participant of the HRS category.

GCR

This category is about *ground*-confluence of *many-sorted* term rewrite systems and was also introduced in 2015. The signature declaration `(f 0 0 -> 1)` in the example below (COPS #558) ensures that the binary function symbol `f` can only appear at the root of terms. Note that the `(c -> 0)` declaration specifies the constant symbol `c`, which does not appear in the rewrite rules, but is used to build the set of ground terms.

```
(VAR x)
(SIG
  (a -> 0)
  (b -> 0)
  (c -> 0)
  (f 0 0 -> 1)
)
(RULES
  a -> b
  f(b,b) -> f(a,a)
  f(x,a) -> f(a,a)
)
```

If `(c -> 0)` is omitted, then the system is ground confluent because the unjoinable peak $\mathsf{f}(\mathsf{c},\mathsf{b}) \leftarrow \mathsf{f}(\mathsf{c},\mathsf{a}) \rightarrow \mathsf{f}(\mathsf{a},\mathsf{a})$ does not exist. In 2019, the tools AGCP and FORT participated in the GCR category.

NFP, UNC, and UNR

The three categories NFP, UNC, and UNR were introduced in 2016 and are about properties of first-order term rewrite systems related to unique normal forms. A rewrite system R has the *normal form property* (NFP) if every term that is convertible to a normal form, rewrites to that normal form (for all terms t and u, if $t \leftrightarrow_R^* u$ and u is a normal form then $t \rightarrow_R^* u$). We say that R has *unique normal forms with respect to conversion* (UNC) if different normal forms are not convertible (for all normal forms t and u, if $t \leftrightarrow_R^* u$ then $t = u$). Finally, R has *unique normal forms with respect to reduction* (UNR) if no term rewrites to different normal forms. These three properties are weaker than confluence (CR):

$$\mathsf{CR} \implies \mathsf{NFP} \implies \mathsf{UNC} \implies \mathsf{UNR}$$

The rewrite system of Example 2

COPS #47

```
(VAR x)
(RULES
  c -> g(c)
  f(x,x) -> a
  g(x) -> f(x,g(x))
)
```

is not confluent but satisfies the three weaker properties. In 2019 CSI and FORT participated in all three categories whereas ACP joined the UNC category.

COM

The category COM is about *commutation* of first-order rewrite systems and was introduced in 2019. Two rewrite systems R and S commute if the inclusion ${}_R^*\!\leftarrow \cdot \rightarrow_S^* \subseteq \rightarrow_S^* \cdot {}_R^*\!\leftarrow$ holds. Here, $\cdot$ denotes relation composition. Commutation is an important generalization of confluence. Apart from direct applications in rewriting, e.g., for confluence, standardization, normalization, and relative termination, commutation is the basis of many results in computer science, like correctness of program transformations [17], and bisimulation up-to [33].

To ensure compatibility of the signatures of the rewrite systems R and S, function symbols and variables in S are renamed on demand. We give an example of a commutation problem that illustrates the problem. Consider COPS #82 (consisting of the rewrite rules $\mathsf{f}(\mathsf{a}) \rightarrow \mathsf{f}(\mathsf{f}(\mathsf{a}))$ and $\mathsf{f}(x) \rightarrow \mathsf{f}(\mathsf{a})$) and COPS #80 (consisting of $\mathsf{a} \rightarrow \mathsf{f}(\mathsf{a},\mathsf{b})$ and $\mathsf{f}(\mathsf{a},\mathsf{b}) \rightarrow \mathsf{f}(\mathsf{b},\mathsf{a})$). Since function symbol f is unary in the first and binary in the second rewrite system, it is renamed to f' in COPS #80:

```
(PROBLEM COMMUTATION)
(COMMENT COPS 82 80)
(VAR x)
(RULES
  f(a) -> f(f(a))
  f(x) -> f(a)
)
(VAR )
(RULES
  a -> f'(a,b)
  f'(a,b) -> f'(b,a)
)
```

The correct answer of this commutation problem is YES since the critical peak of R and S can be closed to a decreasing diagram [1]. To reuse existing systems and avoid duplication, in COPS this problem is given as

```
(PROBLEM COMMUTATION)
(COPS 82 80)
```

and an inlining tool generates the earlier problem (by replacing the `(COPS 82 80)` declaration with the content of COPS #82 and COPS #80, with f in the latter renamed into f′ as described above) before it is passed to tools participating in the commutation category. The COM category was contested by ACP, CoLL, and FORT.

INF

The INF category is about *infeasibility* problems. It was also introduced in 2019. Infeasibility problems originate from different sources. Critical pairs in a conditional rewrite system are equipped with conditions. If no satisfying substitution for the variables in the conditions exists, the critical pair is harmless and can be ignored when analyzing confluence of the rewrite system in question. In this case, the critical pair is said to be *infeasible* [31, Definition 7.1.8]. Sufficient conditions for infeasibility of conditional critical pairs are reported in [19,42].

Another source of infeasibility problems is the dependency graph in termination analysis of rewrite systems [6]. An edge from dependency pair $\ell_1 \to r_1$ to dependency pair $\ell_2 \to r_2$ exists in the dependency graph if two substitutions σ and τ can be found such that $r_1\sigma$ rewrites to $\ell_2\tau$. (By renaming the variables in the dependency pairs apart, a single substitution suffices.) If no such substitutions exist, there is no edge, which may ease the task of proving termination of the underlying rewrite system [13,24].

We provide two example problems. The first one stems from the conditional critical pair between the two conditional rewrite rules in COPS #547:

```
COPS #936 inlined version
(PROBLEM INFEASIBILITY)
(COMMENT COPS 547)
(CONDITIONTYPE ORIENTED)
(VAR x)
(RULES
  f(x) -> a | x == a
  f(x) -> b | x == b
)
(VAR x)
(CONDITION x == a, x == b)
```

The correct answer of this infeasibility problem is YES since no term in the underlying conditional rewrite system rewrites to both `a` and `b`. In COPS, this problem is given as

```
COPS #936
(PROBLEM INFEASIBILITY)
(COPS 547)
(VAR x)
(CONDITION x == a, x == b)
```

and an inlining tool generates the earlier problem before it is passed to tools participating in the infeasibility category. The == sign in the condition of infeasibility problems is interpreted as reachability ($\to^*$) if the rewrite system referenced in the `(COPS n)` declaration is a TRS or an oriented CTRS. If it is *semi-equational* CTRS, then == is interpreted as convertibility ($\leftrightarrow^*$).

The second example is related to Example 2 from the introduction and is a special case since the condition in the infeasibility problem contains no variables:

```
(PROBLEM INFEASIBILITY)
(COMMENT COPS 47)
(VAR x)
(RULES
  F(x,x) -> A
  G(x) -> F(x,G(x))
  C -> G(C)
)
(CONDITION G(A) == A)
```

It has YES as correct answer since the term `G(A)` does not rewrite to `A`. This answer can be used to conclude that the underlying rewrite system is not confluent.

The INF category was contested in 2019 by six tools: CO3, ConCon, Moca, infChecker, MaedMax, and nonreach.

SRS

The category SRS is about confluence of *string rewriting*. String rewrite systems are term rewrite systems in which terms are strings. To ensure that the infrastructure developed for TRSs can be reused, the TRS format is used with the restriction that all function symbols are unary. So a string rewrite rule ab $\to$ ba is rendered as a(b(x)) $\to$ b(a(x)) where x is a variable. A concrete example is given below:

```
COPS #442
(VAR x)
(RULES
  f(f(x)) -> x
  f(x) -> f(f(x))
)
(COMMENT
doi:10.4230/LIPIcs.RTA.2015.257
```

```
[81] Example 1
)
```

The correct answer of this problem is YES since the addition of the redundant rules [26] `f(x) -> f(f(f(x)))` and `f(x) -> x` makes the critical pairs of the SRS development closed [32].

The SRS category was created to foster research on confluence techniques for string rewriting. In the Termination Competition, there is an active community developing powerful techniques for (relative) termination of string rewrite systems. We anticipate that these are beneficial when applied to confluence analysis.

The tools ACP, CSI, CoLL-Saigawa, and noko-leipzig participated in the SRS category.

4 Tools

In this section, we briefly present the tools that participated in CoCo 2019. More detailed descriptions are available online. [9] All tools are available for testing via CoCoWeb.

ACP

The tool ACP [10] has been participating in CoCo from the beginning [5]. In 2019, it participated in the COM, CPF-TRS, CTRS, SRS, TRS and UNC categories, winning three of them. New techniques for the latter category are described in [3]. For the TRS category, ACP supports ordered rewriting [20]. ACP is written in SML/NJ.

AGCP

The tool AGCP [11] participated in the GCR category. It uses rewriting induction to (dis)prove ground confluence of many-sorted rewrite systems [2,4]. AGCP is written in SML/NJ.

CeTA

CeTA[12] is a certifier for (non-)confluence (and other properties) of rewrite systems with and without conditions [45]. It is used by ACP, CSI and ConCon to certify their generated (non-)confluence proofs. The combinations CSI+CeTA and ConCon+CeTA won the respective TRS-CPS and CTRS-CPF categories. New in 2019 is the support for ordered completion proofs for infeasibility of conditional rules and critical pairs [38]. CeTA is code-generated from IsaFoR [37], the Isabelle Formalization of Rewriting.

CO3

The tool CO3 [13] participated in the CTRS and INF categories. CO3 is written in OCaml. It incorporates the new technique of narrowing trees [30]. An early description can be found in [29].

CoLL

The tool CoLL [14] participated in the new COM category. It is written in OCaml and implements various commutation criteria for left-linear rewrite systems [36].

CoLL-Saigawa

The tool CoLL-Saigawa [15] participated in the SRS and TRS categories. It is a combination of CoLL, described above, and the earlier tool Saigawa [15] that participated in CoCo from the very start. CoLL-Saigawa is written in OCaml.

ConCon

The tool ConCon [16] participated in the CTRS, CTRS-CPF and INF categories. ConCon implements several techniques for oriented conditional rewrite systems [40] and employs Maed-Max [46] for infeasibility. ConCon is written in Scala.

CSI

The tool CSI [17] has been participating in CoCo from the beginning [27,48]. In 2019, it participated in the CPF-TRS, NFP, SRS, TRS, UNC and UNR categories, winning four of them (the CPF-TRS category in combination with CeTA). CSI is written in OCaml.

CSIˆho

The tool CSIˆho [18] was the only participant of the HRS category. It implements several techniques for (dis)proving confluence of higher-order rewrite systems [25]. CSIˆho is based on CSI and written in OCaml.

[9] http://project-coco.uibk.ac.at/2019/participants/.

[10] http://www.nue.ie.niigata-u.ac.jp/tools/acp/.

[11] http://www.nue.ie.niigata-u.ac.jp/tools/agcp/.

[12] http://cl-informatik.uibk.ac.at/ceta/.

[13] https://www.trs.css.i.nagoya-u.ac.jp/co3/.

[14] https://www.jaist.ac.jp/project/saigawa/coll/.

[15] https://www.jaist.ac.jp/project/saigawa/.

[16] http://cl-informatik.uibk.ac.at/software/concon/.

[17] http://cl-informatik.uibk.ac.at/software/csi/.

[18] http://cl-informatik.uibk.ac.at/software/csi/ho/.

FORT

The tool FORT [19] is a decision and synthesis tool [34,35] for the first order theory of rewriting for finite left-linear, right-ground rewrite systems. It implements the decision procedure for this theory [10], which uses tree automata techniques. In 2019 it participated in the COM, GCR, NFP, UNC and UNR categories, surprisingly winning the COM category. FORT is written in Java.

infChecker

The tool infChecker [20] is a new participant of CoCo. It uses the theorem prover Prover9 [22] and the model finding tools AGES [14] and Mace4 [22]. Due to the latter, it is the only tool in the INF category that supports NO answers. The tool infChecker is written in Haskell.

MaedMax

The new tool MaedMax [21] participated in the INF category. It implements maximal ordered completion [46] and can output certificates [38] that can be checked by CeTA. The tool was developed as a completion tool and also works as a first-order theorem prover. Given an infeasibility problem, MaedMax translates it into an equivalent satisfiability problem. MaedMax is written in OCaml.

Moca

The tool Moca [22] is a first-order theorem prover and another new participant of CoCo, joining the INF category. It implements maximal ordered completion [46] and the split-if encoding of [9]. Moca is written in Haskell.

noko-leipzig

The new tool noko-leipzig [23] participated in the SRS category. It uses arctically weighted automata [12] for disproving confluence and is written in Haskell.

nonreach

The new tool nonreach [24] participated in the INF category. Among others [23] it implements decomposition techniques based on narrowing [39] for proving infeasibility. The tool nonreach is written in Haskell.

5 Results

In this section, we present the results of CoCo 2019. For each category, we mention problem selection and summarize the competition data. For every category, a problem set consists of 100 problems, including all *secret problems* and a certain number of unresolved problems in the last full run. These problems were randomly selected from the COPS database with the seed number 273 to control the selection. The number was composed of the three seed digits 2 (Hubert Garavel), 7 (Geoff Sutcliffe), 3 (Akihisa Yamada) provided by the panel members. For each category, tools are ranked based on the total number of YES and NO answers. The time tools spent on the problems have no effect on the score.

Full details are available online. [25]

TRS

The TRS category had one secret problem COPS #1133. String rewrite systems were excluded from the selection due to the creation of the SRS category. The results of the TRS category are summarized in the following table: The column

rank	tool	total	yes	no	?	!	∅
1.	ACP	79	44	35	0	5	9.45
2.	CSI	75	42	33	0	0	13.80
	CoLL-Saigawa	57	36	21	1	0	12.69

? lists the number of erroneous answers, and ! lists the number of unique answers, which are the answers that no other tool produced. Moreover, the column ∅ gives the average time spent on each problem (including timeouts). ACP was ahead with 4 problems, breaking the 3-year hegemony of CSI. Due to a wrong answer for COPS #538, CoLL-Saigawa is not ranked.

In total, 82 problems were solved and 18 problems including 12 non-left-linear systems were unsolved. One of the oldest unsolved problems is COPS #126, consisting of the single rule $\mathsf{f}(\mathsf{f}(x, y), z) \to \mathsf{f}(\mathsf{f}(x, z), \mathsf{f}(y, z))$.

CPF-TRS

For the CPF-TRS category, the same problems as in the TRS category were selected. The results are summarized below:

[19] http://cl-informatik.uibk.ac.at/software/FORT/.

[20] http://zenon.dsic.upv.es/infChecker/.

[21] http://cl-informatik.uibk.ac.at/software/maedmax/.

[22] https://www.jaist.ac.jp/project/maxcomp/.

[23] https://tinyurl.com/t6j262m.

[24] https://bitbucket.org/fmessner/nonreach/.

[25] http://project-coco.uibk.ac.at/2019/results.php.

Rank	Tool	Total	Yes	No	?	!	∅
1.	CSI+CeTA	62	28	34	0	62	17.74
2.	ACP+CeTA	0	0	0	0	0	6.37

The win of CSI+CeTA is no surprise since many of the techniques implemented in CSI have been certified. The numbers for ACP+CeTA are explained by a change in the CPF format that was missed by the ACP developers. (From the last column, we infer that ACP spent an average of about 6 s to produce a proof, which then could not be certified by CeTA.) For the full run of CoCo 2019, this was corrected, resulting in the following numbers (out of 501 problems):

Tool	Total	Yes	No	?	!
CSI+CeTA	368	181	187	0	167
ACP+CeTA	204	62	142	0	3

CTRS

The CTRS category had a surprise winner in 2019. Due to wrong answers by ConCon and CO3, the first and second ranked tools of every earlier CoCo, the relative newcomer ACP (participating in the CTRS category since 2018) won.

Rank	Tool	Total	Yes	No	?	!	∅
1.	ACP	49	35	14	0	2	0.76
	ConCon	67	41	26	5	13	5.28
	CO3	53	36	17	1	4	0.01

CPF-CTRS

No surprises in the CPF-CTRS category in 2019, but note the small gap between answers (in the CTRS category) and certified answers:

Rank	Tool	Total	Yes	No	?	∅
1.	ConCon+CeTA	64	38	26	0	4.84

HRS

With three tools participating in 2017, two in 2018, and only one in 2019, [26] the outcome is clear:

[26] CoCo 2020 featured two tools in the HRS category.

Rank	Tool	Total	Yes	No	?	∅
1.	CSI ^ho	52	35	17	0	11.41

GCR

The ranking of the GCR category is no surprise since FORT is a decision tool restricted to TRSs that are both left-linear and right-ground: The very low numbers are explained by

Rank	Tool	Total	Yes	No	?	!	∅
1.	AGCP	4	1	3	0	3	19.96
2.	FORT	1	0	1	0	0	1.41

the COPS selection query, which excluded problems solved in the 2018 full run. The numbers for the full run of CoCo 2019 are as follows (out of 606 problems):

Tool	Total	Yes	No	?	!
AGCP	475	352	123	0	373
FORT	121	38	83	0	19

NFP

The outcome of the NFP category is as expected. Two of the NO answers by FORT are unique:

Rank	Tool	Total	Yes	No	?	!	∅
1.	CSI	51	24	37	0	32	15.77
2.	FORT	31	5	26	0	2	0.30

UNC

The gap between ACP and CSI narrowed to 6 problems (from 14 problems in the 2018 competition):

Rank	Tool	Total	Yes	No	?	!	∅
1.	ACP	74	32	42	0	9	11.51
2.	CSI	68	28	40	0	2	17.15
3.	FORT	30	8	22	0	0	0.31

UNR

Of the 100 selected problems, 32 are left-linear and right-ground, and hence in the scope of FORT:

Rank	Tool	Total	Yes	No	?	!	∅
1.	CSI	63	16	47	0	35	16.59
2.	FORT	32	14	18	0	4	0.27

COM

The outcome of the new COM category was a surprise. CoLL is a designated tool for commutation of left-linear rewrite systems and ACP has support for arbitrary rewrite systems. Due to erroneous answers by these tools, FORT came out on top:

Rank	Tool	Total	Yes	No	?	!	∅
1.	FORT	33	16	17	0	10	3.91
	ACP	52	17	35	5	14	2.23
	CoLL	39	22	17	3	5	22.76

INF

The new INF category had the highest number of contestants, including four new tools, and infChecker won by a large margin. It was the only tool capable of producing NO answers: A total of six secret problems (COPS #1125–#1137) were

Rank	Tool	Total	Yes	No	?	!	∅
1.	infChecker	72	40	32	0	37	21.40
2.	nonreach	30	30	0	0	2	0.07
3.	Moca	26	26	0	0	6	24.10
4.	MaedMax	15	15	0	0	0	7.24
5.	CO3	12	12	0	0	0	0.01
	ConCon	31	31	0	7	2	1.62

submitted by several participants.

SRS

In the SRS category, two secret problems (COPS #1131 and COPS #1132) were submitted. The new tool noko-leipzig produced the most NO answers, but the YES answers by CSI made the difference:

Rank	Tool	Total	Yes	No	?	!	∅
1.	CSI	50	22	28	0	7	32.67
2.	noko-leipzig	41	7	34	0	6	27.95
3.	ACP	35	22	13	0	7	30.42
4.	CoLL-Saigawa	22	11	11	0	3	40.12

6 Outlook

In the near future, we plan to merge CoCo with COPS and CoCoWeb, to achieve a single entry point for confluence problems, tools, and competitions. Moreover, the COPS submission interface will be extended with functionality to support submitters of new problems as well as the CoCo SC.

We plan to reimplement the LiveView software for real-time visualization of CoCo runs, taking into account current limitations, future developments and demands. We will implement flexible scoring schemes and support joint categories based on ordered lists of properties. We will also investigate what additional features are needed to support our sister competition termCOMP.

We anticipate that in the years ahead new categories will be added to CoCo. Natural candidates are rewriting modulo AC, nominal rewriting, and constraint rewriting. Also, we will consider measures to increase the number of tools participating in the HRS category, which is the only CoCo category devoted to higher-order rewriting. Given the large research activity in this area, we are keen to keep the HRS category alive. One possibility is to allow a dependently typed higher-order formalism for expressing problems.

Apart from the improvements mentioned in the preceding paragraphs, the competition serves to highlight progress and challenges in confluence research. On the one hand, the gap between the certified categories and their uncertified counterparts is steadily diminishing, showcasing the progress on the verification front as well as suggesting which techniques are suitable candidates for formal verification to close the gap. On the other hand, problems whose status (YES or NO) is unknown or whose status is known from the literature but out of reach of tools, lead to further research into (automatable) techniques for (dis)proving confluence and related properties. Examples include [26,30,47].

Acknowledgements We are grateful to Nao Hirokawa for continuous support for the infrastructure of CoCo. Fabian Mitterwallner contributed to the inlining and renaming tools for the new commutation and infeasibility categories. Raúl Gutiérrez, Naoki Nishida, and Salvador Lucas contributed the initial set of infeasibility problems (COPS #818–#936). Johannes Waldmann contributed challenging SRS problems (COPS #987–#1036). We acknowledge the TOOLympics 2019 initiators for giving us the opportunity to present CoCo 2019. Finally, the comments by the reviewers helped to improve the presentation.

Funding Open access funding provided by University of Innsbruck and Medical University of Innsbruck.

References

1. Aoto, T.: Automated confluence proof by decreasing diagrams based on rule-labelling. In: Proceedings of 21st International Conference on Rewriting Techniques and Applications, LIPIcs, vol. 6, pp. 7–16 (2010). https://doi.org/10.4230/LIPIcs.RTA.2010.7
2. Aoto, T., Toyama, Y.: Ground confluence prover based on rewriting induction. In: Proceedings of 1st International Conference on Formal Structures for Computation and Deduction, LIPIcs, vol. 52, pp. 33:1–33:12 (2016). https://doi.org/10.4230/LIPIcs.FSCD.2016.33
3. Aoto, T., Toyama, Y.: Automated proofs of unique normal forms with respect to conversion for term rewriting systems. In: Proceedings of 12th International Symposium on Frontiers of Combining Systems, LNCS, vol. 11715 (2019). https://doi.org/10.1007/978-3-030-29007-8_19
4. Aoto, T., Toyama, Y., Kimura, Y.: Improving rewriting induction approach for proving ground confluence. In: Proceedings of 2nd International Conference on Formal Structures for Computation and Deduction, LIPIcs, vol. 84, pp. 7:1–7:18 (2017). https://doi.org/10.4230/LIPIcs.FSCD.2017.7
5. Aoto, T., Yoshida, J., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proceedings of 20th International Conference on Rewriting Techniques and Applications, LNCS, vol. 5595, pp. 93–102 (2009). https://doi.org/10.1007/978-3-642-02348-4_7
6. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Comput. Sci. **236**, 133–178 (2000). https://doi.org/10.1016/S0304-3975(99)00207-8
7. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998). https://doi.org/10.1017/CBO9781139172752
8. Church, A., Rosser, J.B.: Some properties of conversion. Trans. Am. Math. Soc. **39**(3), 472–482 (1936). https://doi.org/10.1090/S0002-9947-1936-1501858-0
9. Claessen, K., Smallbone, N.: Efficient encodings of first-order horn formulas in equational logic. In: Proceedings of 9th International Joint Conference on Automated Reasoning, LNAI, vol. 10900, pp. 388–404 (2018). https://doi.org/10.1007/978-3-319-94205-6_2
10. Dauchet, M., Tison, S.: The theory of ground rewrite systems is decidable. In: Proceedings of 5th IEEE Symposium on Logic in Computer Science, pp. 242–248 (1990). https://doi.org/10.1109/LICS.1990.113750
11. Dershowitz, N., Plaisted, D.A.: Chapter 9 – rewriting. In: Handbook of Automated Reasoning, pp. 535–610. North-Holland (2001). https://doi.org/10.1016/B978-044450813-3/50011-4
12. Felgenhauer, B., Waldmann, J.: Proving non-joinability using weakly monotone algebras. In: Joint Proceedings of the 10th Workshop on Higher-Order Rewriting and the 8th International Workshop on Confluence, pp. 28–32 (2019). Available from http://cl-informatik.uibk.ac.at/iwc/hor-iwc2019.pdf
13. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proceedings of the 5th International Workshop on Frontiers of Combining Systems, LNAI, vol. 3717, pp. 216–231 (2005). https://doi.org/10.1007/11559306_12
14. Gutiérrez, R., Lucas, S.: Automatic generation of logical models with AGES. In: Proceedings of the 27th International Conference on Automated Deduction, LNAI, vol. 11716, pp. 287–299 (2019). https://doi.org/10.1007/978-3-030-29436-6_17
15. Hirokawa, N., Klein, D.: Saigawa: A confluence tool. In: Proceedings of the 1st International Workshop on Confluence, p. 49 (2012). Available from http://cl-informatik.uibk.ac.at/iwc/iwc2012.pdf
16. Hirokawa, N., Nagele, J., Middeldorp, A.: Cops and CoCoWeb – Infrastructure for confluence tools. In: Proceedings of the 9th International Joint Conference on Automated Reasoning, LNAI, vol. 10900, pp. 346–353 (2018). https://doi.org/10.1007/978-3-319-94205-6_23
17. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems. J. ACM **27**(4), 797–821 (1980). https://doi.org/10.1145/322217.322230
18. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech J, (ed,) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
19. Lucas, S., Gutiérrez, R.: Use of logical models for proving infeasibility in term rewriting. Inf. Process. Lett. **136**, 90–95 (2018). https://doi.org/10.1016/j.ipl.2018.04.002
20. Martin, U., Nipkow, T.: Ordered rewriting and confluence. In: Proceedings of the 10th International Conference on Automated Deduction, LNCS, vol. 449, pp. 366–380 (1990). https://doi.org/10.1007/3-540-52885-7_100
21. Mayr, R., Nipkow, T.: Higher-order rewrite systems and their confluence. Theor. Comput. Sci. **192**(1), 3–29 (1998). https://doi.org/10.1016/S0304-3975(97)00143-6
22. McCune, W.: Prover9 and Mace4 (2005–2010). http://www.cs.unm.edu/~mccune/prover9/
23. Meßner, F., Sternagel, C.: nonreach – A tool for nonreachability analysis. In: Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Part I), *LNCS*, vol. 11427, pp. 337–343 (2019). https://doi.org/10.1007/978-3-030-17462-0_19
24. Middeldorp, A.: Approximating dependency graphs using tree automata techniques. In: Proceedings of the 1st International Joint Conference on Automated Reasoning, *LNAI*, vol. 2083, pp. 593–610 (2001). https://doi.org/10.1007/3-540-45744-5_49
25. Nagele, J.: Mechanizing confluence: Automated and certified analysis of first- and higher-order rewrite systems. Ph.D. Thesis, University of Innsbruck (2017). http://resolver.obvsg.at/urn:nbn:at:at-ubi:1-13305
26. Nagele, J., Felgenhauer, B., Middeldorp, A.: Improving automatic confluence analysis of rewrite systems by redundant rules. In: Proceedings of the 26th International Conference on Rewriting Techniques and Applications, LIPIcs, vol. 36, pp. 257–268 (2015). https://doi.org/10.4230/LIPIcs.RTA.2015.257
27. Nagele, J., Felgenhauer, B., Middeldorp, A.: CSI: New evidence – a progress report. In: Proceedings of the 26th International Conference on Automated Deduction, LNAI, vol. 10395, pp. 385–397 (2017). https://doi.org/10.1007/978-3-319-63046-5_24
28. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL – A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002). https://doi.org/10.1007/3-540-45949-9
29. Nishida, N., Kuroda, T., Yanagisawa, M., Gmeiner, K.: CO3: A COnverter for proving COnfluence of COnditional TRSs. In: Proceedings of the 4th International Workshop on Confluence, p. 42 (2015). Available from http://cl-informatik.uibk.ac.at/iwc/iwc2015.pdf

30. Nishida, N., Maeda, Y.: Narrowing trees for syntactically deterministic conditional term rewriting systems. In: Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction, LIPIcs, vol. 108, pp. 26:1–26:20 (2018). https://doi.org/10.4230/LIPIcs.FSCD.2018.26
31. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer, Berlin (2002). https://doi.org/10.1007/978-1-4757-3661-8
32. van Oostrom, V.: Developing developments. Theor. Comput. Sci. **175**(1), 159–181 (1997). https://doi.org/10.1016/S0304-3975(96)00173-9
33. Pous, D.: New up-to techniques for weak bisimulation. Theor. Comput. Sci. **380**(1), 164–180 (2007). https://doi.org/10.1016/j.tcs.2007.02.060
34. Rapp, F., Middeldorp, A.: Automating the first-order theory of rewriting for left-linear right-ground rewrite systems. In: Proceedings of the 1st International Conference on Formal Structures for Computation and Deduction, LIPIcs, vol. 52, pp. 36:1–36:12 (2016). https://doi.org/10.4230/LIPIcs.FSCD.2016.36
35. Rapp, F., Middeldorp, A.: FORT 2.0. In: Proceedings of the 9th International Joint Conference on Automated Reasoning, LNAI, vol. 10900, pp. 81–88 (2018). https://doi.org/10.1007/978-3-319-94205-6_6
36. Shintani, K., Hirokawa, N.: CoLL: A confluence tool for left-linear term rewrite systems. In: Proceedings of the 25th International Conference on Automated Deduction, LNCS, vol. 9195, pp. 127–136 (2015). https://doi.org/10.1007/978-3-319-21401-6_8
37. Sternagel, C., Thiemann, R.: The certification problem format. In: Proceedings of the 11th Workshop on User Interfaces for Theorem Provers, Electronic Proceedings in Theoretical Computer Science, vol. 167, pp. 61–72 (2014). https://doi.org/10.4204/EPTCS.167.8
38. Sternagel, C., Winkler, S.: Certified equational reasoning via ordered completion. In: Proceedings of the 27th International Conference on Automated Deduction, LNAI, vol. 11716, pp. 508–525 (2019). https://doi.org/10.1007/978-3-030-29436-6_30
39. Sternagel, C., Yamada, A.: Reachability analysis for termination and confluence of rewriting. In: Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (Part I), LNCS, vol. 11427, pp. 262–278 (2019). https://doi.org/10.1007/978-3-030-17462-0_15
40. Sternagel, T.: Reliable confluence analysis of conditional term rewrite systems. Ph.D. Thesis, University of Innsbruck (2017). urn:nbn:at:at-ubi:1-9288
41. Sternagel, T., Middeldorp, A.: Conditional confluence (system description). In: Proceedings of the 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications, LNCS (ARCoSS), vol. 8560, pp. 456–465 (2014). https://doi.org/10.1007/978-3-319-08918-8_31
42. Sternagel, T., Middeldorp, A.: Infeasible conditional critical pairs. In: Proceedings of the 4th International Workshop on Confluence, pp. 13–17 (2015)
43. Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: A cross-community infrastructure for logic solving. In: Proceedings of the 7th International Joint Conference on Automated Reasoning, LNAI, vol. 8562, pp. 367–373 (2014). https://doi.org/10.1007/978-3-319-08587-6_28
44. Terese: Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press (2003)
45. Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics, LNCS, vol. 5674, pp. 452–468 (2009). https://doi.org/10.1007/978-3-642-03359-9_31
46. Winkler, S., Moser, G.: MaedMax: A maximal ordered completion tool. In: Proceedings of the 9th International Joint Conference on Automated Reasoning, LNAI, vol. 10900, pp. 472–480 (2018). https://doi.org/10.1007/978-3-319-94205-6_31
47. Yamaguchi, M., Aoto, T.: A fast decision procedure for uniqueness of normal forms w.r.t. conversion of shallow term rewriting systems. In: Proceedings of the 5th International Conference on Formal Structures for Computation and Deduction, LIPIcs, vol. 167, pp. 11:1–11:23 (2020). https://doi.org/10.4230/LIPIcs.FSCD.2020.11
48. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proceedings of the 23th International Conference on Automated Deduction, LNAI, vol. 6803, pp. 499–505 (2011). https://doi.org/10.1007/978-3-642-22438-6_38