Layer Systems for Confluence — Formalized*

Bertram Felgenhauer[1] and Franziska Rapp[2]

[1] Department of Computer Science, University of Innsbruck, Austria
bertram.felgenhauer@uibk.ac.at
[2] Allgemeines Rechenzentrum Innsbruck, Austria

Abstract. Toyama's theorem states that the union of two confluent term rewrite systems with disjoint signatures is again confluent. This is a fundamental result in term rewriting, and several proofs appear in the literature. The underlying proof technique has been adapted to prove further results like persistence of confluence (if a many-sorted term rewrite system is confluent, then the underlying unsorted system is confluent) or the preservation of confluence by currying.
In this paper we present a formalization of modularity and related results in Isabelle/HOL. The formalization is based on layer systems, which cover modularity, persistence, currying (and more) in a single framework. The persistence result has been integrated into the certifier CeTA and the confluence tool CSI, allowing us to check confluence proofs based on persistent decomposition, of which modularity is a special case.

1 Introduction

Toyama's theorem [13,17,19] states that confluence is modular, i.e., that the union of two confluent term rewrite systems (TRSs) over disjoint signatures is confluent if and only if the two TRSs themselves are confluent. For example, Combinatory Logic extended with an equality test

$$@(@(\mathsf{K}, x), y) \to x \quad @(@(@(\mathsf{S}, x), y), z) \to @(@(x, z), @(y, z)) \quad \mathsf{e}(x, x) \to \top$$

is confluent because the first two rules are orthogonal, the last rule is terminating and has no critical pairs, and the signatures of these two sets of rules are disjoint. As the example shows, modularity opens up a decomposition approach to proving confluence, which is attractive, because different confluence criteria may apply to the constituent TRSs that do not apply to their union. By adapting the modularity proof, several other results have been proved in the literature.

- Confluence is persistent [1], i.e., a TRS is confluent if and only if it is confluent as a many-sorted TRS. This gives rise to a decomposition technique, and fully subsumes modularity.
- Confluence is preserved by currying [11]. Currying is useful, for example, as a preprocessing step for deciding ground confluence.

* This work is supported by FWF (Austrian Science Fund) project P27528.

– The notion of modularity has been generalized as well, by weakening the assumption that the signatures of the two TRSs are disjoint; for example, confluence is modular for layer-preserving composable TRSs [16], and for quasi-ground systems [12].

The list goes on. All of these proofs are based on decomposing terms into a maximal top and remaining aliens, but with different sets of admissible tops. In each case, confluence is established by induction on the number of nested tops in that decomposition (the *rank* of a term). Layer systems [7] were introduced as an abstraction from these proofs. A layer system $\mathfrak{L}$ is simply the set of admissible tops; for modularity, those are homogeneous multi-hole contexts, i.e., multi-hole contexts whose function symbols all belong to the signature of only one of the two given TRSs. At the heart of layer systems lies an adaptation of the modularity proof in [17]. When establishing confluence by layer systems, as remaining proof obligations, one has to check that a layer system satisfies so called layer conditions, which is easier than doing a full adaptation of the modularity proof.

Isabelle/HOL [15] is an interactive proof assistant based on higher-order logic with a Hindley-Milner type system, extended with type classes. It follows the LCF tradition [9] in having a trusted kernel, which ensures that theorems follow from the axioms by construction. Isabelle features a structured proof language [20]. Another useful feature are locales, which allow bundling of functions and assumptions that are shared by several definitions and theorems. (For example, locales are used to model groups in Isabelle/HOL). The locale mechanism in Isabelle is quite powerful; in particular, locales can be instantiated (so $\mathbb{Z}$ with addition, 0 as unit, and negation is a group) and extended (for example, the group locale is an extension of a semigroup locale, with additional operations (unit and inverse) and assumptions). Our main reason for using Isabelle/HOL is the existing **Isa**belle **Fo**rmalization of **R**ewriting, IsaFoR [18]. In addition to fundamental notions of term rewriting like terms, substitutions, contexts, multi-hole contexts, and so on, IsaFoR is also the foundation of CeTA (**Ce**rtified **T**ool **A**ssertions), which can certify termination and confluence proofs, among other things.

In this paper we describe a formalization of layer systems in Isabelle/HOL as part of IsaFoR. In fact, the prospect of formalization was one of the selling points of layer systems, with the idea of making large parts of the proof reusable. Note that whereas adapting existing proofs is convenient on paper, it becomes a burden when done in a formalization. The resulting duplication of code (that is, theorem statements and proofs) would decrease maintainability and is therefore best avoided. Our effort covers modularity of confluence, persistence of confluence, and preservation of confluence by currying for first order term rewrite systems. To the best of our knowledge, this is the first time that any of these results has been fully formalized in a proof assistant.

From a practical perspective, our interest in formalization is motivated by our work on an automated confluence prover, CSI [14]. As with all software, CSI potentially contains bugs. In order to increase the trust in CSI, proof output in

a machine readable format is supported, which can be checked using CeTA [18]. As part of our formalization effort, we have extended CeTA with support for a decomposition technique based on persistence of confluence, allowing CSI and potentially other confluence tools to produce certifiable proofs using this technique. We have prepared a website with examples and information about the used software at

http://cl-informatik.uibk.ac.at/software/lisa/ictac2018/

For most theorems and many definitions, we provide the corresponding identifiers in the formalization; in the PDF version of this paper, they link to the HTML version of the formalization itself.

The remainder of this paper is structured as follows. We recall notations and basic definitions in Section 2. Then we present the layer conditions, which are central to our formalization, in Section 3. The next two sections are about persistence. Section 4 uses persistence as an example to illustrate how layer systems can be applied to obtain a confluence result, while Section 5 focuses on the persistent decomposition. In Section 6, we present details of the currying application. Finally, we conclude in Section 7.

2 Preliminaries

We use standard notation from term rewriting [3]. Let $\mathcal{F}$ be a signature and $\mathcal{V}$ be a set of variables. Then $\mathcal{T}(\mathcal{F},\mathcal{V})$ is the set of terms over that signature. We denote by $\mathcal{P}os(t)$ the set of positions of t. The subterm of t at position p is $t|_p$, and $t[s]_p$ is the result of replacing the subterm at position p in t by s. We also write $\mathcal{P}os_X(r)$ for the set of positions p of t such that the root symbol of $t|_p$ is in X. If $X = \{x\}$ is a singleton set, we may omit the outer curly braces and write $\mathcal{P}os_x(t)$. The set of variables of t is $\mathcal{V}ar(t)$. The set of multi-hole contexts over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{C}(\mathcal{F},\mathcal{V})$. (Multi-hole contexts are terms that may contain occurrences of an extra constant $\Box$, representing their holes.) If C is a multi-hole context with n holes, then $C[t_1,\ldots,t_n]$ denotes the term obtained by replacing the i-th hole in C by t_i for $1 \leqslant i \leqslant n$. On multi-hole contexts, we have a partial order $\sqsubseteq$ which is generated by $\Box \sqsubseteq C$ and closure under contexts ($D \sqsubseteq D'$ implies $C[D] \sqsubseteq C[D']$). The corresponding partial supremum operation is denoted by $\sqcup$; intuitively it merges two multi-hole contexts.

A substitution $\sigma,\tau,\ldots$ is a map from variables to terms. The result of applying the substitution σ to the term t is denoted by $t\sigma$. A term rewrite system (TRS) $\mathcal{R}$ is a set of rules $\ell \to r$, where ℓ and r are terms, ℓ is not a variable, and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$. There is a rewrite step from s to t ($s \to_{\mathcal{R}} t$) if $s = s[\ell\sigma]_p$ and $t = s[r\sigma]_p$ for a position $p \in \mathcal{P}os(s)$ and substitution σ.

Given a relation $\to$, we write $\leftarrow$ and $\to^*$ for its inverse and its reflexive transitive closure, respectively. A relation $\to$ is confluent if $t \,{}^*\!\leftarrow s \to^* u$ implies

$t \to^* \cdot {}^*\!\leftarrow u$. It is confluent on X if for all $s \in X$, $t {}^*\!\leftarrow s \to^* u$ implies $t \to^* \cdot {}^*\!\leftarrow u$.[3]

3 Layer Conditions

In the layer system approach to confluence, one sets up a layer system for a TRS $\mathcal{R}$ that satisfies the so-called layer conditions. These layer conditions constitute the interface between the reusable part of the formalization and the parts that are specific to a particular application of layer systems (e.g., modularity). Since they are central to the formalization, we recall the basic constructions and the layer conditions here. For full details please refer to [7].

Recall that modularity of confluence states that the union of two TRSs over disjoint signatures is confluent if each of the two TRSs is confluent (the converse is also true and fairly easy to prove). Modularity is proved by induction on the *rank* of a term; to obtain the rank, one decomposes the term into alternating layers of multi-hole contexts over the two signatures; the rank is the maximum nesting depth of the resulting layers.

Example 1. Let $\mathcal{F}_1 = \{\mathsf{A}, \mathsf{F}\}$ and $\mathcal{F}_2 = \{\mathsf{b}, \mathsf{g}\}$. Then $\mathrm{rank}(\mathsf{F}(\mathsf{F}(\mathsf{A}))) = 1$, while $\mathrm{rank}(\mathsf{g}(\mathsf{b}, \mathsf{F}(\mathsf{b}))) = 3$; the latter term is decomposed into $\mathsf{g}(\mathsf{b}, \Box)$, $\mathsf{F}(\Box)$ and b.

Layer systems abstract from this situation by considering all possible multi-hole contexts at the top of such a decomposition. So a layer system is a set of multi-hole contexts, and gives rise to tops and maximal tops as follows.

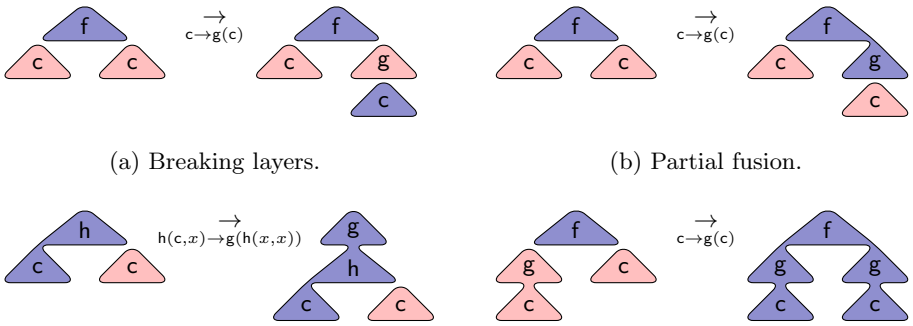
Definition 2 ([7, Definition 3.1]). *Let $\mathcal{F}$ be a signature and $\mathcal{V}$ be an infinite set of variables. Let $\mathfrak{L} \subseteq \mathcal{C}(\mathcal{F}, \mathcal{V})$ be a set of multi-hole contexts over $\mathcal{F}$. Then $L \in \mathfrak{L}$ is called a* top *of a context $C \in \mathcal{C}(\mathcal{F}, \mathcal{V})$ (according to $\mathfrak{L}$) if $L \sqsubseteq C$. A top is a* max-top *of C if it is maximal with respect to $\sqsubseteq$ among the tops of C.*

We want to prove that all terms are confluent, provided that terms of rank 1 are confluent. To this end we have to impose certain restrictions on the layer system.

- the rank must be well-defined, which is ensured if any term has a unique max-top that is not empty (i.e., not equal to $\Box$);
- rewrite steps must not span several layers (so it can be mimicked by a suitable rank 1 term); and
- the rank must not increase by rewriting.

Example 3. We illustrate a few obstructions to proving confluence in Figure 1. (This example is an abridged version of [7, Example 3.4].)

[3] Another reasonable definition for "$\to$ is confluent on X" would be that $\to \cap (X \times X)$ is confluent; this is equivalent to the given definition whenever X is closed under rewriting by $\to$.



(a) Breaking layers.

(b) Partial fusion.

(c) Fusion from above.

(d) Conspiring aliens.

Fig. 1: Undesired behavior on layers.

(a) Here, we have the rewrite step $\mathsf{f}(\mathsf{c},\mathsf{c}) \to \mathsf{f}(\mathsf{c},\mathsf{g}(\mathsf{c}))$, decomposed by some set of layers $\mathfrak{L}$. However, the c subterm becomes two layers after the rewrite step, increasing the rank. So rewriting a layer must again result in a layer.

(b) This is the same rewrite step as in (a). In this example, $\mathsf{g}(\mathsf{c})$ may be a layer. However, the resulting term merges with the layer above (a phenomenon we call *fusion*). In the example, the fusion is *partial*; the fused context is broken apart. This is caused by there being a layer $\mathsf{f}(\Box,\mathsf{g}(\Box))$ but no layer $\mathsf{f}(\Box,\mathsf{g}(\mathsf{c}))$.

(c) In this example, there is a root step $\mathsf{h}(\mathsf{c},\mathsf{c}) \to \mathsf{g}(\mathsf{h}(\mathsf{c},\mathsf{c}))$. Note that both c constants in the result originate in the isolated c, but nevertheless, one of them has fused with the top in the result (so the rewrite step takes place above the point where fusion happens, hence *fusion from above*). In [7, Example 3.4] we show that the TRS

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

has a set of layers such that fusion from above is the sole reason for the system being non-confluent despite being confluent on terms of rank 1.

(d) Finally, it may happen that a rewrite step triggers fusion in a position that is parallel to the rewrite step. (*aliens* are what remains of a term after taking away its max-top; here a rewrite step in one alien causes another alien to fuse, hence *conspiring aliens*). As far as we know, this is not actually an obstruction to confluence, but nevertheless absence of conspiring aliens is required for our proof.

Definition 4 ([7, Definition 3.3]). *Let $\mathcal{F}$ be a signature. A set $\mathfrak{L} \subseteq \mathcal{C}(\mathcal{F},\mathcal{V})$ of contexts is called a* layer system[4] *if it satisfies properties (L_1), (L_2), and (L_3). The elements of $\mathfrak{L}$ are called* layers. *A TRS $\mathcal{R}$ over $\mathcal{F}$ is* weakly layered

[4] In [7] we use $\mathbb{L}$ for layer systems. We use $\mathfrak{L}$ here to be consistent with snippets like Figure 2 that are generated from our Isabelle formalization, where $\mathbb{L}$ is not available.

(according to a layer system $\mathfrak{L}$) *if condition (W) is satisfied for each* $\ell \to r \in \mathcal{R}$. *It is* layered (according to a layer system $\mathfrak{L}$) *if conditions (W), (*C_1*), and (*C_2*) are satisfied. The conditions are as follows.*

(L_1) *Each term in* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *has a non-empty top.*
(L_2) *If* $x \in \mathcal{V}$ *and* $C \in \mathcal{C}(\mathcal{F}, \mathcal{V})$ *then* $C[x]_p \in \mathfrak{L}$ *if and only if* $C[\Box]_p \in \mathfrak{L}$.
(L_3) *If* $L, N \in \mathfrak{L}$, $p \in \mathcal{P}os_{\mathcal{F}}(L)$, *and* $L|_p \sqcup N$ *is defined then* $L[L|_p \sqcup N]_p \in \mathfrak{L}$.
(W) *If* M *is a max-top of* s, $p \in \mathcal{P}os_{\mathcal{F}}(M)$, *and* $s \to_{p,\ell\to r} t$ *then* $M \to_{p,\ell\to r} L$ *for some* $L \in \mathfrak{L}$.
(C_1) *In (W) either* L *is a max-top of* t *or* $L = \Box$.
(C_2) *If* $L, N \in \mathfrak{L}$ *and* $L \sqsubseteq N$ *then* $L[N|_p]_p \in \mathfrak{L}$ *for any* $p \in \mathcal{P}os_{\Box}(L)$.

In a nutshell, (L_1) and (L_3) ensure that the rank is well-defined. Property (L_2) is a technical property that ensures that aliens can always be represented by suitable variables in the confluence proof. Condition (W) prevents breaking layers, and together with (L_3), fusion from above. The final two conditions, (C_1) and (C_2), prevent fusion from above and conspiring aliens, respectively. Now, let us formally define the rank and aliens of a term.

Definition 5 ([7, Definition 3.6]). *Let* $t = M[t_1, \ldots, t_n]$ *with* M *the max-top of* t. *We define* $\mathrm{rank}(t) = 1 + \max\{\mathrm{rank}(t_i) \mid 1 \leqslant i \leqslant n\}$, *where* $\max(\varnothing) = 0$ *(*$t_1, \ldots, t_n$ *are the* aliens *of* t*).*

The main theorems of [7] are as follows (we omit [7, Theorem 4.3] because it has yet to be formalized).

Theorem 6 ([7, Theorem 4.1]). *Let* $\mathcal{R}$ *be a weakly layered TRS that is confluent on terms of rank one. If* $\mathcal{R}$ *is left-linear then* $\mathcal{R}$ *is confluent.*

Theorem 7 ([7, Theorem 4.6]). *Let* $\mathcal{R}$ *be a layered TRS that is confluent on terms of rank one. Then* $\mathcal{R}$ *is confluent.*

```
locale layer_system_sig = fixes F :: 'f sig and L :: ('f, 'v) mctxt set

locale layer_system = layer_system_sig F L for F :: 'f sig and
  L :: ('f, 'v :: infinite) mctxt set +
  assumes L_sig: L ⊆ C
  and L1: t ∈ T ⟹ ∃ L ∈ L. L ≠ MHole ∧ L ≤ mctxt_of_term t
  and L2: p ∈ poss_mctxt C ⟹
    mreplace_at C p (MVar x) ∈ L ⟷ mreplace_at C p MHole ∈ L
  and L3: L ∈ L ⟹ N ∈ L ⟹ p ∈ funposs_mctxt L ⟹
    (subm_at L p, N) ∈ comp_mctxt ⟹
    mreplace_at L p (subm_at L p ⊔ N) ∈ L
```

Fig. 2: Definitions of the *layer_system_sig* and *layer_system* locales in IsaFoR.

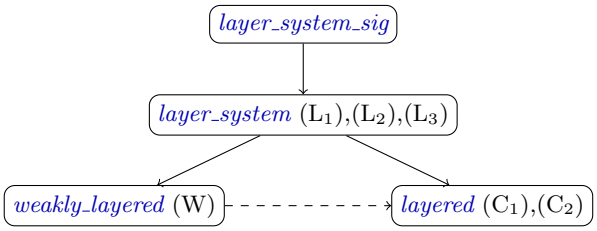


Fig. 3: Hierarchy of locales.

In Isabelle, we bundle these assumptions in locales [4]. Figure 2 shows how the first three layer conditions have been formalized in Isabelle. (A locale is declared using the **locale** keyword, followed by the locale name. It may declare constants using **fixes**, and make assumptions (often about those constants) using **assumes**. Furthermore, a locale may extend other locales; this is the case for *layer_system*, which extends *layer_system_sig*. In order to use a result from a locale, it has to be interpreted, meaning that one provides definitions for the types and constants that the locale depends on and prove that they satisfy the locale assumptions.) Inside the *layer_system_sig* locale, we define $\mathcal{T}$ and $\mathcal{C}$, the set of terms and multi-hole contexts over $\mathcal{F}$, and the concept of max-tops. In fact, max-tops are defined separately for terms and for multi-hole contexts, because while on paper, multi-hole contexts are just terms which may contain an extra constant $\Box$, in IsaFoR they have their own type. In total, four locales are defined, capturing the layer conditions, cf. Figure 3. Note that condition (W) is not part of the *layered* locale; it would be redundant because (C_1) implies (W). In Isabelle we have encoded this fact by proving that *layered* is a sublocale of *weakly_layered*, as indicated by the dashed arrow. (Basically, a locale A is a sublocale of another locale B if the assumptions of B imply those of A.)

Within the formalization, Theorem 6 is established inside the *weakly_layered* locale as theorem *weakly_layered.CR_ll*, whereas Theorem 7 is holds in the *layered* locale as theorem *layered.CR*. (In fact these statements are declared as locale assumptions; they become theorems by proving suitable sublocale relationships. This is done in `LS_Left_Linear.thy` and `LS_General.thy`). The proofs of these main results correspond to Section 4 of [7]. The (lengthy) proof works by induction on the rank: assuming that terms of rank r are confluent, several auxiliary results are derived, and finally, confluence of terms of rank $r+1$ follows. To this end, we use two more locales *weakly_layered_induct* and *weakly_layered_induct_dd* that capture the induction hypothesis, and an auxiliary assumption (namely that local peaks of so called *short steps* are joinable in a suitable way), respectively. For this use of locales it is crucial that they can be interpreted inside of a proof, since the induction hypothesis cannot be established for arbitrary r outside of an induction proof. This happens in the proof of the main lemma [7, Lemma 4.27] which we give in Figure 4. Note that it does induction on the rank (called *rk* in the proof), and that it uses an **interpret** command to instantiate

```
lemma (in weakly_layered) CR_main_lemma:
  assumes base: CR_on (rstep' R) {t. mctxt_of_term t ∈ 𝔏}
  and step: ⋀rk. CR_on (rstep' R) {t ∈ 𝒯. rank t ≤ Suc rk} ⟹
    weakly_layered_induct_dd ℱ 𝔏 R rk (R rk)
  shows CR_on (rstep' R) 𝒯
proof −
  have CR_on (rstep' R) {t ∈ 𝒯. rank t ≤ Suc rk} for rk
  proof (induct rk)
    case 0
    have t ∈ 𝒯 ∧ rank t ≤ Suc 0 ⟷ mctxt_of_term t ∈ 𝔏 for t
      using rank_1[of t] rank_gt_0[of t] 𝔏_sig by (fastforce simp: 𝒞_def 𝒯_def)
    then show ?case using base by simp
  next
    case (Suc rk)
    then interpret weakly_layered_induct_dd ℱ 𝔏 R rk R rk by (rule step)
    show ?case using CR_Suc_rk by (simp only: native_terms_def)
  qed
  then show ?thesis by (auto simp: CR_on_def)
    (metis less_Suc_eq less_Suc_eq_le less_Suc_eq_le)
qed
```

Fig. 4: Proof of the "Main Lemma" for layer systems [7, Lemma 4.27]

the *weakly_layered_induct_dd* locale based on the induction hypothesis inside the proof.

One major benefit of using locales is separation of concerns; thanks to the abstraction of the layer conditions as locales, we could already work on the applications like modularity and currying before the proofs of the main results were complete, without having to worry about working with different assumptions. Basically, each application is an instantiation of these locales, which we could establish independently of the main results.

4 Persistence

To give an impression of what an application of layer systems entails, let us consider the case of persistence. This section overlaps with [7, Section 5.5], but here we focus on interesting aspects in the context of our formalization. In fact, given that the results presented here are both formalized and previously published, we focus on ideas rather than giving full proofs.

Definition 8 (*many_sorted_terms*, *persistent_cr_infinite_vars*)**.** *Let $\mathcal{S}$ be a set of* sorts*. A* many-sorted signature *$\mathcal{F}$ associates with each function symbol f of arity n a signature $f : \beta_1 \times \cdots \times \beta_n \to \alpha$, where $\beta_1, \ldots, \beta_n, \alpha \in \mathcal{S}$. Furthermore we assume that there are pairwise disjoint, infinite sets of variables $\mathcal{V}_\alpha$ for $\alpha \in \mathcal{S}$. The sets of of* terms of sort *α for $\alpha \in \mathcal{S}$ are defined inductively by*

$$\mathcal{T}_\alpha ::= \mathcal{V}_\alpha \cup \{f(t_1, \ldots, t_n) \mid f : \beta_1 \times \cdots \times \beta_n \to \alpha, t_1 \in \mathcal{T}_{\beta_1}, \ldots, t_n \in \mathcal{T}_{\beta_n}\}$$

A many-sorted *TRS $\mathcal{R}$ is a TRS such that for every $\ell \to r \in \mathcal{R}$, $\ell, r \in \mathcal{T}_\alpha$ for some $\alpha \in \mathcal{S}$.*

We wish to establish the following theorem using layer systems.

Theorem 9 (many-sorted persistence, *CR_persist*). *Let $\mathcal{R}$ be a many-sorted TRS. We let $\mathcal{V} = \bigcup_{\alpha\in\mathcal{S}} \mathcal{V}_\alpha$. Then $\mathcal{R}$ is confluent on $\mathcal{T}_\alpha$ for all $\alpha \in \mathcal{S}$ if and only if $\mathcal{R}$ is confluent on $\mathcal{T}(\mathcal{F}, \mathcal{V})$.*

To this end we define a layer system $\mathfrak{L}$ as follows.

$$\begin{aligned}\mathfrak{L}_\alpha ::= {} & \mathcal{V} \cup \{\Box\} \cup \\ & \{f(C_1, \ldots, C_n) \mid f : \beta_1 \times \cdots \times \beta_n \to \alpha, C_1 \in \mathfrak{L}_{\beta_1}, \ldots, C_n \in \mathfrak{L}_{\beta_n}\} \\ \mathfrak{L} = {} & \bigcup_{\alpha\in\mathcal{S}} \mathfrak{L}_\alpha\end{aligned}$$

Showing that $\mathfrak{L}$ layers $\mathcal{R}$ is mostly straightforward. However, in order to show (W) (which is a prerequisite for showing (C_1)), one has to establish that if a rewrite step is applicable to a term at a position that is part of its max-top, then it is also applicable to the max-top itself. In order to obtain the substitution for the second rewrite step, it is helpful to define functions that compute the max-top:

$$\mathsf{mt}_\alpha(x) = x \qquad \text{for } x \in \mathcal{V}$$

$$\mathsf{mt}_\alpha(f(t_1, \ldots, t_n)) = \begin{cases} f(\mathsf{mt}_{\beta_1}(t_1), \ldots, \mathsf{mt}_{\beta_n}(t_n)) & \text{if } f : \beta_1 \times \cdots \times \beta_n \to \alpha \\ \Box & \text{if } f : \beta_1 \times \cdots \times \beta_n \to \alpha' \\ & \text{and } \alpha \neq \alpha' \end{cases}$$

The max-top of a term t equals $\mathsf{mt}_\alpha(t)$ for some $\alpha \in \mathcal{S}$ that can be obtained by looking at the root symbol of t.

Lemma 10 (*push_mt_subst*, *push_mt_ctxt*). *The following properties hold for mt_α.*

- *if $s \in \mathcal{T}_\alpha$ then $\mathsf{mt}_\alpha(s\sigma) = s\sigma'$ where $\sigma'(x) = \mathsf{mt}_\alpha(\sigma(x))$ for $x \in \mathcal{V}_\alpha$; and*
- *if $p \in \mathcal{P}os(\mathsf{mt}_\alpha(t))$, then for some $\beta \in \mathcal{S}$, all terms s satisfy $\mathsf{mt}_\alpha(t[s]_p) = \mathsf{mt}_\alpha(t)[\mathsf{mt}_\beta(s)]$.*

Now, given a rewrite step $s[\ell\sigma]_p \to s[r\sigma]_p$, with $p \in \mathcal{P}os_{\mathcal{F}}(\mathsf{mt}_\alpha(s))$ (as in (W)), the lemma entails

$$\begin{aligned}\mathsf{mt}_\alpha(s[\ell\sigma]_p) = \mathsf{mt}_\alpha(s)[\mathsf{mt}_\beta(\ell\sigma)]_p &= \mathsf{mt}_\alpha(s)[\ell\sigma']_p \\ \to \mathsf{mt}_\alpha(s)[r\sigma']_p &= \mathsf{mt}_\alpha(s)[\mathsf{mt}_\beta(r\sigma)]_p = \mathsf{mt}_\alpha(s[r\sigma]_p)\end{aligned}$$

where $\ell, r \in \mathcal{T}_\beta$; this gives the desired rewrite step for (W). For (C_1) note that $s[r]_p$ can be a variable, in which case it is possible that $\mathsf{mt}_\alpha(s[r\sigma]_p) = \Box$, whereas the max-top is larger.

Remark 11. This idea of defining the max-top as a function is a recurring theme; it features in the formalizations of modularity and currying as well. The main benefit of (recursive) functions is that they come with an induction principle that is not available for the implicit notion of a "maximal top".

After showing that $\mathfrak{L}$ layers $\mathcal{R}$, Theorem 7 yields the following corollary.

Corollary 12 (*CR_on_union*)**.** *If $\mathcal{R}$ is confluent on $\mathfrak{L} \cap \mathcal{T}(\mathcal{F}, \mathcal{V})$,*[5] *then $\mathcal{R}$ is confluent on $\mathcal{T}(\mathcal{F}, \mathcal{V})$.*

Let us now sketch a proof of Theorem 9. First note that if $\mathcal{R}$ is a many-sorted TRS, then the sets $\mathcal{T}_\alpha$ are closed under rewriting by $\mathcal{R}$; hence confluence of $\mathcal{R}$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ implies confluence of $\mathcal{R}$ on $\mathcal{T}_\alpha$ for any $\alpha \in \mathcal{S}$. For the converse, we want to use Corollary 12. We need to show that $\mathcal{R}$ is confluent on $\mathfrak{L} \cap \mathcal{T}(\mathcal{F}, \mathcal{V})$. To this end, assume that $s \in \mathfrak{L} \cap \mathcal{T}(\mathcal{F}, \mathcal{V})$, and we have a peak $t \mathrel{{}^*\!\leftarrow} s \to^* t$. If s is a variable then $s = t = u$ and we're done. Otherwise, we can read off the sort α of s from its root symbol. Note that s is not necessarily an element of $\mathcal{T}_\alpha$, because $\mathfrak{L}$ disregards the sorts of variables. We modify s in two steps; first we annotate each variable with the type that is induced by its context (i.e., if x is the i-th argument of $f : \beta_1 \times \cdots \times \beta_n \to \gamma$, then we replace it by (x, β_i));[6] and secondly we rename the annotated variables in such a way that each (v, β) is replaced by an element of $\mathcal{V}_\beta$. In this fashion, we obtain a peak $t' \mathrel{{}^*\!\leftarrow} s' \to^* u'$, where $s', t', u' \in \mathcal{T}_\alpha$, and a substitution σ with $s = s'\sigma$, $t = t'\sigma$ and $u = u'\sigma$. By confluence of $\mathcal{R}$ on $\mathcal{T}_\alpha$, there is a valley $t' \to^* v' \mathrel{{}^*\!\leftarrow} u'$, and hence a corresponding valley $t = t'\sigma \to^* v'\sigma \mathrel{{}^*\!\leftarrow} u'\sigma = u$ in $\mathfrak{L} \cap \mathcal{T}(\mathcal{F}, \mathcal{V})$.

5 Persistent Decomposition

Aoto and Toyama [1] pointed out that persistence gives rise to a decomposition technique for proving confluence. The basic idea is to attach sorts to a TRS. To obtain a decomposition, for each sort of the many-sorted TRS obtained in that way, the set of rules that are applicable to terms of that sort is computed. By persistence, if all of the resulting systems are confluent, the original TRS is confluent as well. In [2] a refined version of the persistent decomposition is presented, wherein only the maximal systems w.r.t. the subset relation are considered.

Example 13 ([1, Example 1]). Consider the TRS $\mathcal{R}$ consisting of the rules

$$\begin{aligned} \mathsf{f}(x, y) &\to \mathsf{f}(\mathsf{g}(x), \mathsf{g}(y)) & \mathsf{F}(\mathsf{g}(x), x) &\to \mathsf{F}(x, \mathsf{g}(x)) \\ \mathsf{g}(x) &\to \mathsf{h}(x) & \mathsf{F}(\mathsf{h}(x), x) &\to \mathsf{F}(x, \mathsf{h}(x)) \end{aligned}$$

The following sort attachment makes the TRS $\mathcal{R}$ many-sorted:

$$\mathsf{f} : 2 \times 2 \to 0 \qquad \mathsf{g} : 2 \to 2 \qquad \mathsf{h} : 2 \to 2 \qquad \mathsf{F} : 2 \times 2 \to 1$$

[5] Because multi-hole contexts are not terms, this is $\{t.\ \mathit{mctxt\_of\_term}\ t \in \mathfrak{L}\}$ in the formalization.

[6] This annotation procedure formalizes the following sentence in the proof of [7, Theorem 5.13]: "Note that for each p the sort of $s'|_p$ is uniquely determined by s."

Looking at the sorts of possible subterms of terms of sort 0 (namely 0 and 2), 1 (1 and 2) and 2 (only 2), we obtain three induced TRSs, consisting of the first two rules, the last three rules, and only the second rule of $\mathcal{R}$, respectively. The last TRS is contained in the other two, and hence does not have to be considered. Confluence of $\mathcal{R}$ follows from confluence of the two systems

$$\mathsf{g}(x) \to \mathsf{h}(x) \qquad\qquad \mathsf{f}(x,y) \to \mathsf{f}(\mathsf{g}(x),\mathsf{g}(y))$$

(which is orthogonal) and

$$\mathsf{g}(x) \to \mathsf{h}(x) \qquad \mathsf{F}(\mathsf{g}(x),x) \to \mathsf{F}(x,\mathsf{g}(x)) \qquad \mathsf{F}(\mathsf{h}(x),x) \to \mathsf{F}(x,\mathsf{h}(x))$$

(which is terminating and has joinable critical pairs). Non-confluence of $\mathcal{R}$ would follow if any of the three TRSs induced by the sorts 0, 1, or 2 was non-confluent.

$$\frac{}{\alpha \trianglerighteq \alpha}\ \mathsf{refl} \qquad \frac{\alpha \trianglerighteq \beta \quad \beta \trianglerighteq \gamma}{\alpha \trianglerighteq \gamma}\ \mathsf{trans} \qquad \frac{f : \beta_1 \times \cdots \times \beta_n \to \alpha \quad 1 \leqslant i \leqslant n}{\alpha \trianglerighteq \beta_i}\ \mathsf{arg}$$

Fig. 5: Syntactic order on sorts.

Definition 14. *Let $\mathcal{R}$ be a many-sorted TRS. Based on the signature, we define an order $\trianglerighteq$ on sorts by the rules in Figure 5. The* TRS $\mathcal{R}_\alpha$ induced by $\alpha \in \mathcal{S}$ *is given by*

$$\mathcal{R}_\alpha = \{\ell \to r \mid \ell \to r \in \mathcal{R},\ \ell \in \mathcal{T}_\beta,\ \alpha \trianglerighteq \beta\}$$

Remark 15. The notation $\trianglerighteq$ is justified by the fact that $\mathcal{T}_\alpha \ni s \trianglerighteq t \in \mathcal{T}_\beta$ implies $\alpha \trianglerighteq \beta$. Note further that $\alpha \trianglerighteq \beta$ implies $\mathcal{R}_\alpha \supseteq \mathcal{R}_\beta$, so the maximal induced TRSs $\mathcal{R}_\alpha$ w.r.t. subsets are induced by the maximal sorts α w.r.t. $\trianglerighteq$.

Since only rules from $\mathcal{R}_\alpha$ are applicable to terms in $\mathcal{T}_\alpha$, we have the following lemma.

Lemma 16 (*CR_on_$\mathcal{T}_\alpha$_by_needed_rules*). *The system $\mathcal{R}$ is confluent on $\mathcal{T}_\alpha$ if and only if $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}_\alpha$.*

We formalize the persistent decomposition result as follows.

Theorem 17 (*persistent_decomposition*). *Let $\Sigma \subseteq \mathcal{S}$ be a set of sorts with the property that for each $\beta \in \mathcal{S}$, either $\mathcal{R}_\beta = \varnothing$, or $\alpha \in \Sigma$ for some $\alpha \trianglerighteq \beta$. Then $\mathcal{R}$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$ if and only if $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$ for all $\alpha \in \Sigma$.*

Since no proof has been given in the literature[7] (as far as we know), we include one here.

[7] The proof is not difficult, but as a system description, [2] lacked space for a proof.

Proof. First assume that $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$ for all $\alpha \in \Sigma$. By Theorem 9, confluence of $\mathcal{R}$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$ follows if we can show that $\mathcal{R}$ is confluent on $\mathcal{T}_\beta$ for any $\beta \in \mathcal{S}$. By Lemma 16, this is equivalent to $\mathcal{R}_\beta$ being confluent on $\mathcal{T}_\beta$. If $\mathcal{R}_\beta = \varnothing$, we are done. Otherwise, by assumption, there is a sort $\alpha \trianglerighteq \beta$ such that $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$. Because $\mathcal{T}_\beta$ is closed under rewriting by $\mathcal{R}_\alpha$, $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}_\beta$, which implies that $(\mathcal{R}_\alpha)_\beta = \mathcal{R}_\beta$ is confluent on $\mathcal{T}_\beta$ by Lemma 16 and the fact that $\mathcal{R}_\alpha$ is a many-sorted TRS using the same signature as $\mathcal{R}$.

For the other direction, assume that $\mathcal{R}$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$. We show that $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}(\mathcal{F},\mathcal{V})$ for all $\alpha \in \mathcal{S}$ (and in particular those in Σ). Since $\mathcal{R}_\alpha$ is a many-sorted TRS, it is persistent (Theorem 9), so it suffices to show that $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}_\beta$ for all $\beta \in \mathcal{S}$. So consider a peak $t \,{}_{\mathcal{R}_\alpha}^{*}\!\leftarrow s \rightarrow_{\mathcal{R}_\alpha}^{*} u$. We proceed by induction on $s \in \mathcal{T}_\beta$.

If $s \in \mathcal{V}$ then $s = t = u$ and we are done. Otherwise, $s = f(s_1,\dots,s_n)$ for some $f : \beta_1 \times \cdots \times \beta_n \to \beta$, and $s_1 \in \mathcal{T}_{\beta_1}$, …, $s_n \in \mathcal{T}_{\beta_n}$. There are two cases.

1. If $\alpha \trianglerighteq \beta$, then since $\mathcal{R}$ is confluent on $\mathcal{T}_\beta$, $\mathcal{R}_\beta$ is confluent on $\mathcal{T}_\beta$. By Lemma 16 applied to $(\mathcal{R}_\alpha)_\beta = \mathcal{R}_\beta$, $\mathcal{R}_\alpha$ is confluent on $\mathcal{T}_\beta$ as well.
2. If $\alpha \not\trianglerighteq \beta$, then $\mathcal{R}_\alpha$ contains no rules whose root symbol has result sort β. Consequently there cannot be any root steps in $t \,{}_{\mathcal{R}_\alpha}^{*}\!\leftarrow s \,{}_{\mathcal{R}_\alpha}^{*}\!\leftarrow u$. Hence we obtain $t_1,\dots,t_n$ and $u_1,\dots,u_n$ with $t_i \,{}_{\mathcal{R}_\alpha}^{*}\!\leftarrow s_i \rightarrow_{\mathcal{R}_\alpha}^{*} u_i$ for $1 \leqslant i \leqslant n$, $t = f(t_1,\dots,t_n)$, and $u = f(u_1,\dots,u_n)$. We conclude by the induction hypothesis (s_i is confluent for $1 \leqslant i \leqslant n$). □

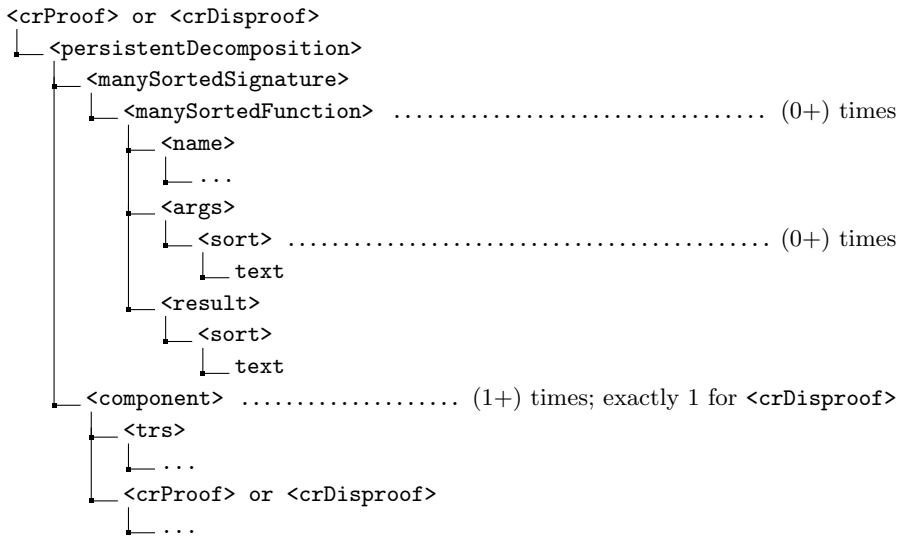


Fig. 6: CPF fragment for persistent decomposition proofs

We further integrated this result into CeTA. To this end, we implemented a function that computes the maximal sorts (with respect to $\trianglerighteq$) for a given signature, a check function that checks the preconditions of Theorem 17, and extended CeTA's CPF parser with a certificate format for a persistent decomposition (CPF is an XML format. The fragment for persistent decomposition is given in Figure 6, and may be of interest to tool authors who want to incorporate certifiable persistent decomposition into their confluence tools).

6 Currying

Currying is the most complicated application of layer systems that we have formalized so far. Currying is a transformation of term rewrite systems in which applications of n-ary functions are replaced by n applications of a single fresh binary function symbol to a constant, thereby applying arguments to the function one by one. More formally, we introduce a fresh function symbol $\bullet$ to denote application, whereas every other function symbol becomes a constant. We adopt the convention of writing f_n to denote a function symbol of arity n. Moreover, we denote the arity of a function symbol f with respect to the signature $\mathcal{F}$ by $\mathsf{a}_{\mathcal{F}}(f)$. We identify $f_{\mathsf{a}_{\mathcal{F}}(f)}$ with f.

Definition 18. *Given a TRS $\mathcal{R}$ over a signature $\mathcal{F}$, its* curried version $\mathsf{Cu}(\mathcal{R})$ *consists of rules $\{\mathsf{Cu}(l) \to \mathsf{Cu}(r) \mid \ell \to r \in \mathcal{R}\}$, where $\mathsf{Cu}(t) = t$ if t is a variable and $\mathsf{Cu}(f(t_1, \ldots, t_n)) = f_0 \bullet \mathsf{Cu}(t_1) \bullet \cdots \bullet \mathsf{Cu}(t_n)$. Here $\bullet$ is a fresh left-associative function symbol.*

Currying is useful for deciding properties such as confluence [5] or termination [10]. For analyzing confluence by currying, the following result is important.

Theorem 19 (*main_result_complete*)**.** *Let $\mathcal{R}$ be a TRS. If $\mathcal{R}$ is confluent, then $\mathsf{Cu}(\mathcal{R})$ is confluent.*

This result was proved by Kahrs [11]. Rather than working directly with $\mathsf{Cu}(\mathcal{R})$, Kahrs works with the *partial parametrization* of $\mathcal{R}$, which is given by $\mathsf{PP}(\mathcal{R}) = \mathcal{R} \cup \mathcal{U}_{\mathcal{F}}$, where $\mathcal{U}_{\mathcal{F}}$ is the set of uncurrying rules for $\mathcal{F}$ (see Definition 20). Confluence of $\mathsf{PP}(\mathcal{R})$ and $\mathsf{Cu}(\mathcal{R})$ are closely related, cf. Lemma 21.

Definition 20. *Given a signature $\mathcal{F}$, the* uncurrying *rules $\mathcal{U}_{\mathcal{F}}$ are rules*

$$f_i(x_1, \ldots, x_i) \bullet x_{i+1} \to f_{i+1}(x_1, \ldots, x_{i+1})$$

for every function symbol $f \in \mathcal{F}$ and $0 \leqslant i < \mathsf{a}_{\mathcal{F}}(f)$.

Lemma 21 ([11, Proposition 3.1]). *Let $\mathcal{R}$ be a TRS. Then $\mathsf{Cu}(\mathcal{R})$ is confluent if $\mathsf{PP}(\mathcal{R})$ is.*

Hence in order to prove Theorem 19 it suffices to prove that $\mathsf{PP}(\mathcal{R})$ is confluent. To this end, we make use of Theorem 7. Hence we need to show that $\mathsf{PP}(\mathcal{R})$ is

layered according to some set of layers $\mathfrak{L}$, and confluent on terms of rank one. First of all we have to define a suitable set of layers. We choose $\mathfrak{L} = \mathfrak{L}_1 \cup \mathfrak{L}_2$ letting $\mathcal{V}_\Box = \mathcal{V} \cup \{\Box\}$ and

$$\begin{aligned}\mathfrak{L}_1 ::= \mathcal{V}_\Box \cup \{ & f_m(s_1, \ldots, s_m) \bullet s_{m+1} \bullet \cdots \bullet s_n \mid \\ & f \in \mathcal{F},\ 0 \leqslant m \leqslant n \leqslant \mathsf{a}_\mathcal{F}(f) \text{ and } s_1, \ldots, s_n \in \mathfrak{L}_1\} \\ \mathfrak{L}_2 = \{ & x \bullet t \mid x \in \mathcal{V}_\Box \text{ and } t \in \mathfrak{L}_1\}\end{aligned}$$

This definition realizes a separation between well-formed terms ($\mathfrak{L}_1$), whose $\mathcal{U}_\mathcal{F}$-normal form contains no $\bullet$ symbol, and ill-formed terms ($\mathfrak{L}_2$), whose $\mathcal{U}_\mathcal{F}$-normal form contains exactly one $\bullet$ symbol at the root. As required for condition (L_1), variables and holes are treated interchangeably.

Whereas for Lemma 21 we could follow the lines of the paper proof, the formalization of the fact that $\mathsf{PP}(\mathcal{R})$ is layered according to $\mathfrak{L}$ turned out to be much more tedious. As with the modularity and persistence applications, we found it convenient to define functions that *compute* the max-top of a term, since the abstract definition of max-tops in the layer framework is not really suitable for proofs in Isabelle.

Definition 22. *The following function checks whether the number of arguments applied to the first non-$\bullet$ function symbol f is at most the arity $\mathsf{a}_\mathcal{F}(f)$ according to the original signature $\mathcal{F}$*

$$\mathsf{check}(t, m) = \begin{cases} \mathit{false} & \mathit{if}\ t \in \mathcal{V} \\ \mathsf{check}(t_1, m+1) & \mathit{if}\ t = t_1 \bullet t_2 \\ \mathsf{a}_\mathcal{F}(f) \geqslant m + n & \mathit{if}\ t = f_n(t_1, \ldots, t_n) \end{cases}$$

Let $\mathcal{F}^\bullet = \mathcal{F} \cup \{\bullet\}$. The max-top $\mathsf{mt}_{\mathsf{Cu}}$ of a term $t \in \mathcal{T}(\mathcal{F}^\bullet, \mathcal{V})$ with respect to $\mathfrak{L}$ is computed as

$$\mathsf{mt}_{\mathsf{Cu}}(t) = \begin{cases} t & \mathit{if}\ t \in \mathcal{V} \\ f(\mathsf{mt}_1(t_1, 0), \ldots, \mathsf{mt}_1(t_n, 0)) & \mathit{if}\ t = f(t_1, \ldots, t_n) \\ & \mathit{and}\ (\mathsf{check}(t, 0)\ \mathit{or}\ t_1 \in \mathcal{V}) \\ \Box \bullet \mathsf{mt}_1(t_2, 0) & \mathit{otherwise}\ (\mathit{in\ which\ case}\ t = t_1 \bullet t_2) \end{cases}$$

Here $\mathsf{mt}_1(t, m)$ computes the max-top of t with respect to $\mathfrak{L}_1$, where m is the number of already applied arguments:

$$\mathsf{mt}_1(t, m) = \begin{cases} t & \mathit{if}\ t \in \mathcal{V} \\ \mathsf{mt}_1(t_1, m+1) \bullet \mathsf{mt}_1(t_2, 0) & \mathit{if}\ t = t_1 \bullet t_2\ \mathit{and}\ \mathsf{check}(t, m) \\ f(\mathsf{mt}_1(t_1, 0), \ldots, \mathsf{mt}_1(t_n, 0)) & \mathit{if}\ t = f(t_1, \ldots, t_n),\ f \neq \bullet \\ & \mathit{and}\ \mathsf{check}(t, m) \\ \Box & \mathit{otherwise} \end{cases}$$

Note that there is some redundancy, since the check *function does the same counting several times. It turns out, however, that this redundancy simplifies later proofs.*

After proving the correctness of mt_1 and $\mathsf{mt}_{\mathsf{Cu}}$, the main difficulty was the proof of condition (C_1) for $\mathfrak{L}$ and $\mathsf{PP}(\mathcal{R})$. Similar to Lemma 10, we proved facts about the interaction of mt_1 (and hence $\mathsf{mt}_{\mathsf{Cu}}$) with contexts and substitutions, in order to analyze a rewrite step $s = C[l\sigma]_p \to C[r\sigma]_p$ with p a function position of the max-top M of s.

Lemma 23 (*push_mt_in_ctxt*)**.** *Let s be a term and p the hole position of context C such that $C[s]_p \in \mathcal{T}(\mathcal{F}^\bullet, \mathcal{V})$ and $p \in \mathcal{P}os_{\mathcal{F}^\bullet}(\mathsf{mt}_1(C[s], j))$. Then there exists a context D and a natural number k such that $\mathsf{mt}_1(C[s], j) = D[\mathsf{mt}_1(s, k)]$, and $\mathsf{mt}_1(C[t], j) = D[\mathsf{mt}_1(t, k)]$ for any term $t \in \mathcal{T}(\mathcal{F}^\bullet, \mathcal{V})$ having the same number of missing arguments as s.*

Lemma 24 (*push_mt_in_subst*)**.** *Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Then $\mathsf{mt}_1(t \cdot \sigma, 0) = \mathsf{mt}_1(t, 0) \cdot \sigma'$ with $\sigma' = (\lambda x.\, \mathsf{mt}_1(x, 0)) \circ \sigma$.*

Using these two lemmas, we can obtain the desired rewrite step from M by the following computation, where for simplicity we only consider the case $M \in \mathfrak{L}_1$ and $l \to r \in \mathcal{R}$:

$$M = \mathsf{mt}(s) = \mathsf{mt}_1(C[l \cdot \sigma], 0) \overset{23}{=} D[\mathsf{mt}_1(l \cdot \sigma, k)] \overset{24}{=} D[\mathsf{mt}_1(l, 0) \cdot \sigma'] = D[l \cdot \sigma']$$
$$\to_{p,\ell\to r} D[r \cdot \sigma'] = D[\mathsf{mt}_1(r, 0) \cdot \sigma'] \overset{24}{=} D[\mathsf{mt}_1(r \cdot \sigma, k)] \overset{23}{=} \mathsf{mt}_1(C[r \cdot \sigma], 0)$$

The uses of the previous two lemmas are indicated above the equalities. Note that the number of missing arguments of r and l are equal (namely 0), so we can use Lemma 23 in both directions. For the same reason we must have $k = 0$, because otherwise $\mathsf{mt}_1(l \cdot \sigma, k) = \Box$, contradicting the fact that the rewrite step would take place at a function position of M. Hence Lemma 24 is applicable. Furthermore, we use $\mathsf{mt}_1(l, 0) = l$ and $\mathsf{mt}_1(r, 0) = r$, using that l and r are well-formed. At this point we have established (W). For (C_1), we analyze the term $\mathsf{mt}_1(C[r \cdot \sigma], 0)$ some more: If $C = \Box$, r is a variable and $\mathsf{check}(r \cdot \sigma)$ is false, $\mathsf{mt}_1(C[r \cdot \sigma], 0) = \Box$. Otherwise, the max-top of $C[r \cdot \sigma]$ is equal to $\mathsf{mt}_1(C[r \cdot \sigma], 0)$.

Remark 25. As an anonymous reviewer suggested, it would most likely have been easier to use a different layer system, where each $\bullet$ symbol starts a new layer:

$$\begin{aligned}
\mathfrak{L}'_1 &= \mathcal{T}(\mathcal{F}, \mathcal{V}_\Box) \\
\mathfrak{L}'_2 &= \{f_m(s_1, \ldots, s_m) \bullet s_{m+1} \mid f \in \mathcal{F}, 0 \leqslant m < \mathsf{a}_{\mathcal{F}}(f) \text{ and } s_1, \ldots, s_{m+1} \in \mathfrak{L}'_1\} \\
\mathfrak{L}'_3 &= \{x \bullet y \mid x, y \in \mathcal{V}_\Box\} \cup \\
&\qquad \{f_m(x_1, \ldots, x_m) \mid f \in \mathcal{F}, 0 \leqslant m < \mathsf{a}_{\mathcal{F}}(f) \text{ and } x_1, \ldots, x_m \in \mathcal{V}_\Box\}
\end{aligned}$$

This would have avoided the complications of counting the number of "missing" arguments in the check function. Unfortunately we did not find this idea before starting our formalization. Adapting the existing formalization accordingly would be a substantial effort with no obvious gain—the final result would still be that currying preserves confluence.

topic	lines	dB factor
definitions, basic facts about layers	3.2k	20
Theorem 7	2.0k	13
modularity	0.8k	30
persistence	1.5k	55
currying	3.8k	40
executable persistence check	0.6k	—
total	12k	

Fig. 7: Formalization effort (dB = de Bruijn)

7 Conclusion

We have presented a formalization of modularity, persistence, and currying, in the Isabelle proof assistant. The formalization spans about 12k lines of theory files and took approximately 9 person-months to develop. A breakdown of the effort is given in Figure 7. (Note that modularity is subsumed by persistence. We formalized modularity first because it is the easiest application. Many proof ideas for modularity carried over to the other, more difficult applications.) The de Bruijn factor (which compares the size of the formalized proof to the paper version) varies wildly. We believe that the main reason for this is that the level of detail for proofs in [7] varies greatly; the core confluence proof (leading up to Theorem 7) is carried out in much more detail than the applications, where large parts of the proofs rely on the reader's intuition. A second contributing factor is that two people worked on different parts of the formalization.

As far as we know, this is the first formalization of modularity of confluence in any proof assistant. We would like to point out that even though the confluence proof for layer systems is based on a constructive proof of modularity of confluence [17], the formalized result is not constructive. This is because Isabelle/HOL is a classical logic. Producing a constructive proof in Isabelle/HOL would have to rely on discipline (including the avoidance of proof automation tools like Metis that are based on Skolemization). In fact, since the proof factors through decreasing diagrams (which were already part of the Archive of Formal Proofs [6]), we would first need a constructive proof for confluence by decreasing diagrams. In the end we would not reap any benefits from having a constructive proof (namely, an executable confluence result).

We integrated the persistence result into our theorem prover CSI (which already supported order-sorted persistence, so the main effort for extending CSI was adding the XML output.) We present experimental results in Figure 8. The check mark ✓ indicates certified strategies; CSI✓and +pd✓are the certified strategies with and without persistent decomposition, respectively, while CSI refers to the uncertified, full strategy of CSI. As can be seen from the data, we have achieved a modest improvement in certified proofs over the Cops database

	CSI✓	+pd✓	CSI
yes	148	154	244
no	162	162	162
maybe	127	121	31
total	437	437	437

Fig. 8: Impact of persistent decomposition on certifiable proofs by CSI.

of confluence problems.[8] It is worth noting that there is no progress in certified non-confluence proofs; in fact, there is no certification gap for non-confluence at all. For non-confluence, CSI employs tree automata [8], which (in theory, and evidently also in practice) subsume the many-sorted decomposition result, because many-sorted terms are a regular tree language.

There are several parts of [7] that have not yet been formalized. For one, there are two more applications of layer systems, namely modularity of layer-preserving composable TRSs, and a modularity result for quasi-ground systems. The bigger missing part are *variable-restricted layer systems*, which are the foundation for a generalized persistence result with ordered sorts [7, Theorem 6.3]. Furthermore, while we have formalized preservation of confluence by currying, this is not integrated into CeTA. As far as we know, no confluence tool currently uses currying directly. However, currying is the basis of efficient decision procedures for ground TRSs, which are implemented in CSI, and are a target for future formalization efforts.

References

1. Aoto, T., Toyama, Y.: Extending persistency of confluence with ordered sorts. Tech. Rep. IS-RR-96-0025F, School of Information Science, JAIST (1996)
2. Aoto, T., Yoshida, J., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proc. 20th RTA. LNCS, vol. 5595, pp. 93–102 (2009), doi: 10.1007/978-3-642-02348-4_7
3. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998), doi: 10.1017/CBO9781139172752
4. Ballarin, C.: Locales: A module system for mathematical theories. JAR **52**(2), 123–153 (2014)
5. Felgenhauer, B.: Deciding confluence of ground term rewrite systems in cubic time. In: Proc. 23rd RTA. LIPIcs, vol. 15, pp. 165–175 (2012), doi: 10.4230/LIPIcs.RTA.2012.165
6. Felgenhauer, B.: Decreasing diagrams II. AFP (Aug 2015), formal proof development, https://www.isa-afp.org/entries/Decreasing-Diagrams-II.html
7. Felgenhauer, B., Middeldorp, A., Zankl, H., van Oostrom, V.: Layer systems for proving confluence. ACM TOCL **16**(2:14), 1–32 (2015), doi: 10.1145/2710017

[8] full results are available at http://cl-informatik.uibk.ac.at/software/lisa/ictac2018/.

8. Felgenhauer, B., Thiemann, R.: Reachability, confluence, and termination analysis with state-compatible automata. I&C **253**(3), 467–483 (2017), doi: 10.1016/j.ic.2016.06.011
9. Gordon, M., Milner, R., Wadsworth, C.: Edinburgh LCF, LNCS, vol. 78. Springer (1979), doi: 10.1007/3-540-09724-4
10. Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination. In: Proc. 15th LPAR. pp. 667–681 (2008)
11. Kahrs, S.: Confluence of curried term-rewriting systems. JSC **19**(6), 601–623 (1995), doi: 10.1006/jsco.1995.1035
12. Kitahara, A., Sakai, M., Toyama, Y.: On the modularity of confluent term rewriting systems with shared constructors. Technical Reports of the Information Processing Society of Japan **95**(15), 11–20 (1995), in Japanese
13. Klop, J., Middeldorp, A., Toyama, Y., de Vrijer, R.: Modularity of confluence: A simplified proof. IPL **49**, 101–109 (1994), doi: 10.1016/0020-0190(94)90034-5
14. Nagele, J., Felgenhauer, B., Middeldorp, A.: CSI: New evidence – A progress report. In: Proc. 26th CADE. LNCS (LNAI), vol. 10395, pp. 385–397 (2017), doi: 10.1007/978-3-319-63046-5_24
15. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL – A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002), doi: 10.1007/3-540-45949-9
16. Ohlebusch, E.: Modular Properties of Composable Term Rewriting Systems. Ph.D. thesis, Universität Bielefeld (1994)
17. van Oostrom, V.: Modularity of confluence constructed. In: Proc. 4th IJCAR. LNCS (LNAI), vol. 5195, pp. 348–363 (2008), doi: 10.1007/978-3-540-71070-7_31
18. Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Proc. 22nd TPHOLs. LNCS, vol. 5674, pp. 452–468 (2009), doi: 10.1007/978-3-642-03359-9_31
19. Toyama, Y.: On the Church-Rosser property for the direct sum of term rewriting systems. JACM **34**(1), 128–143 (1987), doi: 10.1145/7531.7534
20. Wenzel, M.: Isar - A generic interpretative approach to readable formal proof documents. In: Proc. 12th TPHOLs. pp. 167–184 (1999), doi: 10.1007/3-540-48256-3_12