Type Introduction for Runtime Complexity Analysis*

Martin Avanzini[1] **and Bertram Felgenhauer**[1]

1 **Institute of Computer Science,**
University of Innsbruck, Austria
{martin.avanzini,bertram.felgenhauer}@uibk.ac.at

1 Introduction

Runtime complexity analysis is a natural refinement of *termination* analysis. Instead of asking whether all reductions yield a result eventually, we are interested in how long the reduction process takes. In order to measure the runtime complexity of a term rewrite system (TRS for short) it is natural to look at the maximal length of derivation sequences, a program first suggested by Hofbauer and Lautemann [5]. The resulting notion of complexity is called *derivational complexity*. Hirokawa and Moser [4] introduced a variation, called *runtime complexity*, that only takes *basic* or *constructor-based* terms as start terms into account. This notion of complexity accurately express the complexity of a program through the runtime complexity of a TRS, and constitutes an *invariant cost model* for rewrite systems [2].

Advanced techniques developed in the context of program complexity analysis essentially rely on sort information. For instance Hoffmann et al. [6] define an elegant and powerful calculus to infer various complexity properties of *resource aware ML programs*, essentially *sorted rewrite systems*, automatically. It is inherently difficult to transfer these techniques into an untyped setting.

In this note we show that the runtime complexity function of a sorted rewrite system $\mathcal{R}$ coincides with the runtime complexity function of the unsorted rewrite system $\Theta(\mathcal{R})$, obtained by forgetting sort information. Hence our result states that *sort-introduction*, a process that is easily carried out via unification, is sound for runtime complexity analysis. Our result thus provides the foundation for exploiting sort information in analysis of TRSs.

Our main research is tightly related to the research on *persistent* properties [9] of rewrite systems, e.g. [1, 7]. Here a property on rewrite systems is called persistent if it holds for the sorted TRS $\mathcal{R}$ if and only if it holds on the unsorted variant $\Theta(\mathcal{R})$. As trivial corollary to our main result we obtain that *innermost termination* is persistent, a result that has been previously established in [3].

2 Preliminaries

We assume familiarity with rewriting [8]. We denote by $\mathcal{V}$ a countable infinite set of *variables*, $\mathcal{F}$ denotes a signature and $\mathcal{T}(\mathcal{F},\mathcal{V})$ denotes the set of terms with symbols in $\mathcal{F}$ and variables in $\mathcal{V}$. We denote by $\mathsf{Var}(t)$ the set of variables occurring in t. Let $\mathcal{R}$ be a TRS. Roots of left-hand sides in $\mathcal{R}$ are called *defined*, symbols that are not defined are called constructors and are collected in $\mathcal{C}_\mathcal{R}$. Terms $t = f(t_1,\dots,t_k)$ with $t_i \in \mathcal{T}(\mathcal{C}_\mathcal{R},\mathcal{V})$ for all $i = 1,\dots,k$ are called *basic*. The *rewrite relation* of a term rewrite system $\mathcal{R}$ is denoted by $\to_\mathcal{R}$, by $\xrightarrow{\mathsf{i}}_\mathcal{R}$ we denote the innermost rewrite relation of $\mathcal{R}$. The *runtime complexity (function)* $\mathrm{rc}_\mathcal{R} : \mathbb{N} \to \mathbb{N}$

* This work is supported by FWF (Austrian Science Fund) projects J3563 and P22467.

of $\mathcal{R}$ is defined by

$$\mathrm{rc}_{\mathcal{R}}(n) := \max\{\ell \mid \exists t_0,\dots,t_\ell.\ t_0 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_\ell \text{ and } t_0 \text{ is a basic term of size up to } n\}\ .$$

Note that $\mathrm{rc}_{\mathcal{R}}$ is well-defined when $\mathcal{R}$ is terminating. The *innermost runtime complexity (function)* $\mathrm{rci}_{\mathcal{R}}$ of $\mathcal{R}$ is defined analogously, considering innermost reductions only.

To simplify notations, we employ the notion of *$\mathcal{S}$-sorted rewriting* of Aoto and Toyama [1]. Let $\mathcal{S}$ be a set of *sorts*. Sorts are denoted by $\alpha, \beta, \dots$, possibly followed by subscripts. A *sort-attachment* is a mapping τ from $\mathcal{V} \cup \mathcal{F}$ to $\mathcal{S}^*$ such that $\tau(x) \in \mathcal{S}$ for $x \in \mathcal{V}$ and $\tau(x) \in \mathcal{S}^{k+1}$ for every k-ary $f \in \mathcal{F}$. In the latter case we write $f : \alpha_1,\dots,\alpha_k \to \alpha$ instead of $\tau(f) = \alpha_1,\dots,\alpha_k,\alpha$. Without loss of generality, we assume that for each $\alpha \in \mathcal{S}$ the sets $\mathcal{V}_\alpha := \{x \mid \tau(x) = \alpha\}$ are countable infinite. The sort $\mathsf{sort}(t)$ of a term t is defined by the root symbol only. We set $\mathsf{sort}(x) := \tau(x)$ for variables x and $\mathsf{sort}(f(t_1,\dots,t_k)) = \alpha$ where $f : \alpha_1,\dots,\alpha_k \to \alpha$.

A term t is *well-sorted* (under τ) with sort α if $t : \alpha$ is derivable by the following rules: (i) $t = x$ and $\tau(x) = \alpha$, or (ii) $t = f(t_1,\dots,t_k)$, $f : \alpha_1,\dots,\alpha_k \to \alpha$ and $t_i : \alpha_i$ $(i = 1,\dots,k)$. We denote by $\mathcal{T}(\mathcal{F},\mathcal{V})^\tau \subseteq \mathcal{T}(\mathcal{F},\mathcal{V})$ the set of all term which are well-sorted under τ.

An *$\mathcal{S}$-sorted TRS $\mathcal{R}$* is given by an unsorted TRS $\Theta(\mathcal{R})$ and sort-attachment τ such that every rule $l \to r \in \Theta(\mathcal{R})$, $l : \alpha$ and $r : \alpha$ holds for some sort $\alpha \in \mathcal{S}$. In the following, $\mathcal{R}$ always denotes an $\mathcal{S}$-sorted TRS. The rewrite relation $\to_{\mathcal{R}}$ of an $\mathcal{S}$-sorted TRS is given by the restriction of $\to_{\Theta(\mathcal{R})}$ to well-sorted terms $\mathcal{T}(\mathcal{F},\mathcal{V})^\tau$. We extend the notion of runtime complexity function in the obvious way to $\mathcal{S}$-sorted TRSs. A property P of TRSs is called *persistent* if for each rewrite system $\mathcal{R}$, $\mathcal{R}$ has property P if and only if $\Theta(\mathcal{R})$ has property P. Notice that our notion of persistency coincides with the standard notion formulated on many-sorted TRSs, see [1].

3 Bounded Runtime Complexity is a Persistent Property of TRSs

In the following we show that the *bounded runtime complexity problem*, which asks for a TRS $\mathcal{R}$ and function $f : \mathbb{N} \to \mathbb{N}$ whether $\mathrm{rc}_{\mathcal{R}}(n) \leqslant f(n)$ for all $n \in \mathbb{N}$ holds, is persistent. We even show a stronger property, viz, $\mathrm{rc}_{\mathcal{R}}(n) = \mathrm{rc}_{\Theta(\mathcal{R})}(n)$ for all $n \in \mathbb{N}$. It is clear that every $\mathcal{R}$-derivation is also an $\Theta(\mathcal{R})$-derivation, hence $\mathrm{rc}_{\mathcal{R}}(n) \leqslant \mathrm{rc}_{\Theta(\mathcal{R})}(n)$ holds trivially. The converse is however not true in general. Consider the sorted TRS $\mathcal{R}_1$ consisting of rules

$$\mathsf{f}(0,1,x) \to \mathsf{f}(x,x,x) \qquad \mathsf{g}(y,z) \to y \qquad \mathsf{g}(y,z) \to z\ ,$$

and sort-attachment so that $0, 1 : \alpha$, $\mathsf{f} : \alpha,\alpha,\alpha \to \alpha$ and $\mathsf{g} : \beta,\beta \to \beta$. Notice that $\mathcal{R}_1$ is terminating, since sorting excludes the formation of terms involving both f and g symbols. On the other hand, the TRS $\Theta(\mathcal{R}_1)$ gives rise to a cycle

$$t := \underline{\mathsf{f}(0,1,\mathsf{g}(0,1))} \to_{\Theta(\mathcal{R}_1)} \mathsf{f}(\underline{\mathsf{g}(0,1)},\mathsf{g}(0,1),\mathsf{g}(0,1)) \to_{\Theta(\mathcal{R}_1)} \mathsf{f}(0,\underline{\mathsf{g}(0,1)},\mathsf{g}(0,1)) \to_{\Theta(\mathcal{R}_1)} t\ ,$$

and is thus not terminating. The TRS $\mathcal{R}_1$ is the prototypical example that shows that termination is not persistent, it is however not a counterexample to our claim. The notion of runtime complexity considers only basic, i.e. *argument normalised* terms. Indeed, the runtime complexity function of the sorted TRS $\mathcal{R}_1$ and its unsorted version $\Theta(\mathcal{R}_1)$ coincide.

Our central observation is that in a $\Theta(\mathcal{R})$-derivation D starting from argument normalised term t, subterms that lead to a sort conflict (called *aliens* of t below) do not contribute to the derivation D itself. Although the (normalised) aliens might get duplicated or erased, the sorting condition on $\mathcal{R}$ ensures that aliens never contribute to a pattern which triggers the application of a rule. This suggests that such aliens in t can be replaced by fresh variables so that the resulting term s is well-sorted. Although some care has to be taken in the assignment of variables to aliens for non-left-linear systems, the derivation D of t can be simulated step-wise by a $\mathcal{R}$-derivation starting from the modified term s.

Fix a set of sorts $\mathcal{S}$ and an $\mathcal{S}$-sorted TRS $\mathcal{R}$. To define sorted contexts, we assume the presence of fresh constants $\Box_\alpha$, the *holes*, for each sort $\alpha \in \mathcal{S}$. We extend the type assignment τ underlying $\mathcal{R}$ so that $\tau(\Box_\alpha) = \alpha$. A *multi-holed context* $C[\Box_{\alpha_1}, \dots, \Box_{\alpha_n}]$ is a sorted term that contains each hole $\Box_{\alpha_i}$ $(i = 1, \dots, n)$ exactly once. With $C[t_1, \dots, t_n]$ we denote the term obtained by replacing holes α_i with t_i in $C[\Box_{\alpha_1}, \dots, \Box_{\alpha_n}]$.

We write $s = C[\![s_1, \dots, s_n]\!]$ for the *unique* decomposition $s = C[s_1, \dots, s_n]$ into a well-sorted context $C[\Box_{\alpha_1}, \dots, \Box_{\alpha_n}]$ and terms s_i with and $\mathsf{sort}(s_i) \neq \alpha_i$ for every $i = 1, \dots, n$. The subterms $s_1, \dots, s_n$ of s are called the *aliens* of s. The set of all aliens $\{s_1, \dots, s_n\}$ in s is denoted by $\mathsf{alien}(s)$. Note that when s is well-sorted, the context C degenerates to s.

▶ **Definition 3.1.** Let s be a term. Consider a family $\gamma = (\gamma_\alpha : T_{\neq\alpha} \to V_\alpha)_{\alpha \in \mathcal{S}}$ of *bijective* mappings from terms $T_{\neq\alpha} \subseteq \{t \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \mid \mathsf{sort}(t) \neq \alpha\}$ to variables $V_\alpha \subseteq \mathcal{V}_\alpha$. We define the *domain* and *range* of γ by $\mathsf{dom}(\gamma) := \cup_{\alpha \in \mathcal{S}} T_{\neq\alpha}$ and $\mathsf{range}(\gamma) := \cup_{\alpha \in \mathcal{S}} V_\alpha$ respectively. Then γ is called an *alien replacement* for s if $\mathsf{alien}(s) \subseteq \mathsf{dom}(\gamma)$ and $\mathsf{range}(\gamma) \cap \mathsf{Var}(s) = \varnothing$. We denote by $\bar{\gamma}$ the *inverse* of γ: $\bar{\gamma}(x) := t$ where $\gamma_{\mathsf{sort}(x)}(t) = x$ for all $x \in \mathsf{range}(\gamma)$.

We define $s \gtrdot_\gamma t$ if $s = C[\![s_1, \dots, s_n]\!]$ for context $C[\Box_{\alpha_1}, \dots, \Box_{\alpha_n}]$, γ is an alien replacement for s and $t = C[\gamma_{\alpha_1}(s_1), \dots, \gamma_{\alpha_n}(s_n)]$ is well-sorted.

Notice that to each term $s = C[\![s_1, \dots, s_n]\!]$ we can associate an alien replacement γ and term t such that $s \gtrdot_\gamma t$ holds: Start from a well-sorted term $C[x_1, \dots, x_n]$ for pairwise disjoint and fresh variables of appropriate sort. Identify variables x_i and x_j $(i, j \in \{1, \dots, n\})$ when $\mathsf{sort}(x_i) = \mathsf{sort}(x_j)$ and $s_i = s_j$. The fixpoint of this construction yields the well-sorted term $t := C[y_1, \dots, y_n]$. The family $\gamma = (\gamma_\alpha)_{\alpha \in \mathcal{S}}$, defined by $\gamma_{\mathsf{sort}(x_i)}(y_i) := s_i$ for $i = 1, \dots, n$, is an alien replacement where by construction $s \gtrdot_\gamma t$ holds.

Consider $s \gtrdot_\gamma t$. By the conditions on $\mathsf{range}(\gamma)$ it follows that t matches s with substitution $\bar{\gamma}$, i.e. $s = t\bar{\gamma}$. Provided γ is an alien replacement, we can also state the inverse correspondence.

▶ **Lemma 3.2.** *Let γ denote an alien replacement for a term s. Then $s \gtrdot_\gamma t$ if and only if $s = t\bar{\gamma}$ and t is well-sorted.*

The following lemma confirms that t is a *maximal* well-sorted pattern that matches s.

▶ **Lemma 3.3.** *Suppose $s \gtrdot_\gamma t$ holds, and let u be a well-sorted term with $\mathsf{Var}(u) \cap \mathsf{range}(\gamma) = \varnothing$. If u matches s then u matches also t.*

Proof. Let σ be a substitution with $s = u\sigma$. Without loss of generality, we suppose $\mathsf{dom}(\sigma) \subseteq \mathsf{Var}(u)$. Observe that since u is well-sorted, aliens of s occur only in the substitution part. We define the substitution σ_γ as follows, for all $x \in \mathsf{dom}(\sigma)$: suppose $\mathsf{sort}(x) \neq \mathsf{sort}(\sigma(x))$, thus $\sigma(x) \in \mathsf{alien}(s)$ and $\gamma_{\mathsf{sort}(x)}(s)$ is well-defined. Then we set $\sigma_\gamma(x) := \gamma_{\mathsf{sort}(x)}(\sigma(x))$. Otherwise, suppose $\sigma(x) = C_x[\![s_1, \dots, s_m]\!]$ for some non-empty context $C_x[\Box_{\alpha_1}, \dots, \Box_{\alpha_m}]$ and aliens $\{s_1, \dots, s_m\} \subseteq \mathsf{alien}(s)$. Then we set $\sigma_\gamma(x) := C_x[\gamma_{\alpha_1}(s_1), \dots, \gamma_{\alpha_1}(s_m)]$.

By definition of σ_γ, $\sigma_\gamma(x)\bar{\gamma} = \sigma(x)$ for $x \in \mathsf{dom}(\sigma)$. By the variable condition on u, we have $(u\sigma_\gamma)\bar{\gamma} = s$. Note that $u\sigma_\gamma$ is by construction well-sorted, Lemma 3.2 thus gives $s \gtrdot_\gamma u\sigma_\gamma$. By definition of $\gtrdot_\gamma$ we see that $s \gtrdot_\gamma u\sigma_\gamma$ and $s \gtrdot_\gamma t$ implies $u\sigma_\gamma = t$. ◀

The following lemma provides our central simulation result. Since we consider derivations from argument normalised terms only, it suffices to consider only *outer steps* in the simulation.

▶ **Definition 3.4.** A rewrite step $s \rightarrow_{\Theta(\mathcal{R})} t$ is called *inner* if it takes place in one of the aliens of s. The step $s \rightarrow_{\Theta(\mathcal{R})} t$ is called *outer* if it is not an inner rewrite step.

▶ **Lemma 3.5.** *Suppose $s_1 \gtrdot_\gamma t_1$ holds for an alien replacement γ with $\mathsf{range}(\gamma)$ disjoint from the set of variables occurring in $\mathcal{R}$.*

1) If $s_1 \rightarrow_{\Theta(\mathcal{R})} s_2$ is an outer step then $t_1 \rightarrow_{\mathcal{R}} t_2$ for some term t_2 with either (i) $s_2 \gtrdot_\gamma t_2$ or (ii) $s_2 \in \mathsf{alien}(s_1)$ with $t_2 = \gamma_\alpha(s_2)$ for some $\alpha \in \mathcal{S}$.
2) If $s_1 \xrightarrow{\mathsf{i}}_{\Theta(\mathcal{R})} s_2$ is an outer step then $t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_2$ for some term t_2 with either (i) $s_2 \gtrdot_\gamma t_2$ or (ii) $s_2 \in \mathsf{alien}(s_1)$ and $t_2 = \gamma_\alpha(s_2)$ for some $\alpha \in \mathcal{S}$.

Proof. We consider Proposition 1 first. Suppose $s_1 \gtrdot_\gamma t_1$ for γ as above. Consider an outer rewrite step $s_1 \rightarrow_{\Theta(\mathcal{R})} s_2$. The proof is by induction on the rewrite context.

In the base case, $s_1 = l\sigma$ and $s_2 = r\sigma$ for some substitution σ and rewrite rule $l \rightarrow r \in \mathcal{R}$. By Lemma 3.3 we obtain a substitution σ_γ such that $t_1 = l\sigma_\gamma \rightarrow_{\mathcal{R}} r\sigma_\gamma$. We verify that either condition (i) or (ii) holds for $t_2 = r\sigma_\gamma$.

Reconsider the substitution σ_γ constructed in Lemma 3.3. Suppose first that the applied rewrite rule is collapsing, i.e. $r \in \mathsf{Var}(l)$. We distinguish the two cases in construction of σ_γ. In the first case, $r\sigma_\gamma = \gamma_{\mathsf{sort}(r)}(r\sigma)$ with $r\sigma \in \mathsf{alien}(s_1)$, i.e. (ii) holds. In the second case $r\sigma_\gamma = C_x[\![u_1,\dots,u_m]\!]$ for some non-empty context $C_x[\Box_{\alpha_1},\dots,\Box_{\alpha_m}]$ and aliens $\{u_1,\dots,u_m\} \subseteq \mathsf{alien}(s_1)$. This yields $\mathsf{alien}(r\sigma) \subseteq \mathsf{alien}(l\sigma)$, as moreover $\mathsf{Var}(r\sigma) \subseteq \mathsf{Var}(l\sigma)$ we conclude that γ is also an alien replacement for $r\sigma$. As the side conditions on $\mathsf{range}(\gamma)$ gives $(r\sigma_\gamma)\bar{\gamma} = r\sigma$, we conclude $r\sigma \gtrdot_\gamma r\sigma_\gamma$ by Lemma 3.2.

Now suppose that the applied rewrite rule is non-collapsing. Since $l \rightarrow r$ is well-sorted, any alien in $r\sigma$ occurs in the substitution, and hence is an alien of $l\sigma$. Hence again γ is an alien replacement for $r\sigma$, since $(r\sigma_\gamma)\bar{\gamma} = r\sigma$ we obtain $r\sigma \gtrdot_\gamma r\sigma_\gamma$ using Lemma 3.2. This finishes the base case.

For the inductive step, consider an outer rewrite step

$$s_1 = f(u_1,\dots,u_i,\dots,u_k) \rightarrow_{\Theta(\mathcal{R})} f(u_1,\dots,v_i,\dots,u_k) = s_2 \ ,$$

with $u_i \rightarrow_{\Theta(\mathcal{R})} v_i$ outer. Using $s_1 \gtrdot_\gamma t_1$, Lemma 3.2 gives $t_1 = f(u'_1,\dots,u'_i,\dots,u'_k)$ with $u_i = u'_i\bar{\gamma}$ for all $i = 1,\dots,k$. Hence by induction hypothesis, $u'_i \rightarrow_{\mathcal{R}} v'_i$ for some well-sorted term v'_i with either $v_i \gtrdot_\gamma v'_i$ or $v'_i = \gamma_\alpha(v_i)$ for $v_i \in \mathsf{alien}(u_i)$ and $\alpha \in \mathcal{S}$. Set $t_2 := f(u'_1,\dots,v'_i,\dots,u'_k)$ and thus $t_1 \rightarrow_{\mathcal{R}} t_2$. If $v_i \gtrdot_\gamma v'_i$ holds then $s_2 \gtrdot_\gamma t_2$ follows by applying Lemma 3.2 immediately. Hence suppose $v'_i = \gamma_\alpha(v_i)$. Since t_2 is well-sorted and $\mathsf{sort}(v_i) \neq \mathsf{sort}(\gamma_\alpha(v_i))$ by definition, it follows that v_i is an alien in s_2. Again we conclude $s_2 = t_2\bar{\gamma}$ and thus $s_2 \gtrdot_\gamma t_2$ by Lemma 3.2. We conclude Proposition 1.

For Proposition 2, observe that if $l\sigma \rightarrow_{\Theta(\mathcal{R})} r\sigma$ is an innermost step, i.e $l\sigma$ is argument normalised, then so is $l\sigma_\gamma$ and hence $l\sigma_\gamma \rightarrow_{\mathcal{R}} r\sigma_\gamma$ is an innermost rewrite step. Proposition 2 follows then by reasoning identical to above. ◀

▶ **Lemma 3.6.** *If $s_1 \rightarrow_{\Theta(\mathcal{R})} s_2$ is outer, then either $\mathsf{alien}(s_2) \subseteq \mathsf{alien}(s_1)$ or $s_2 \in \mathsf{alien}(s_1)$.*

Proof. Consider an outer step $s_1 \rightarrow_{\Theta(\mathcal{R})} s_2$, and let t_1 be such that $s_1 \gtrdot_\gamma t_1$ holds. Then by Lemma 3.5, either $s_2 \gtrdot_\gamma t_2$ for some term t_2 or $s_2 \in \mathsf{alien}(s_1)$. In the former case, $s_2 \gtrdot_\gamma t_2$ witnesses that aliens of s_2 occur as aliens in s_1, in the latter case we conclude directly. ◀

▶ **Theorem 3.7.** *Let s be a term such that all aliens in s are in $\Theta(\mathcal{R})$ normal-form. Then any (innermost) $\Theta(\mathcal{R})$-derivation of s is simulated step-wise by an (innermost) $\mathcal{R}$-derivation starting from some t with $s \gtrdot_\gamma t$.*

Proof. Consider a derivation $D : s = s_0 \rightarrow_{\Theta(\mathcal{R})} s_1 \rightarrow_{\Theta(\mathcal{R})} s_2 \rightarrow_{\Theta(\mathcal{R})} \cdots$. Using Lemma 3.6, a standard induction shows that $\mathsf{alien}(s_i) \subseteq \mathsf{alien}(s)$ for all but possibly the last term in D, and that the steps $s_i \rightarrow_{\Theta(\mathcal{R})} s_{i+1}$ are outer. We conclude the by Lemma 3.5(1) (Lemma 3.5(2) respectively). ◀

Any basic term s satisfies trivially that aliens of s are in $\Theta(\mathcal{R})$ normal-form. The above theorem thus shows that the $\mathsf{dh}(s, \rightarrow_{\Theta(\mathcal{R})}) \leqslant \mathsf{dh}(s', \rightarrow_{\mathcal{R}})$, whenever s is basic. Since $s \gtrdot_\gamma s'$ implies that $|s| \geqslant |s'|$ it follows that $\mathrm{rc}_{\Theta(\mathcal{R})}(n) \leqslant \mathrm{rc}_{\mathcal{R}}(n)$. By identical reasoning, $\mathrm{rci}_{\Theta(\mathcal{R})}(n) \leqslant \mathrm{rci}_{\mathcal{R}}(n)$. Thus we obtain the following corollary.

▶ **Corollary 3.8.** *The (innermost) runtime complexity functions of $\mathcal{R}$ and $\Theta(\mathcal{R})$ coincide. In particular, the bounded (innermost) runtime complexity problem is persistent.*

Observe that if a TRS $\mathcal{R}$ is innermost non-terminating, then there exists a minimal non-terminating term $s = f(s_1, \ldots, s_n)$ in the sense that all arguments are in normal-form. In particular, the aliens of s are normalised. We thus re-obtain the following result from [3].

▶ **Corollary 3.9.** *Innermost termination is a persistent property.*

4 Conclusion

In this abstract we have shown that sort-introduction is sound for runtime complexity analysis. We considered the most simple form of sorted rewriting. It is expected that our result can be extended to more general forms, allowing for instance *polymorphism* or *ordered sorts*. Such extensions are subject to future research. To which extent sort information can be exploited in runtime complexity analysis is also subject to further research.

References

1 T. Aoto and Y. Toyama. Persistency of Confluence. *JUCS*, 3(11):1134–1147, 1997.

2 M. Avanzini and G. Moser. Closing the Gap Between Runtime Complexity and Polytime Computability. In *Proc. of 21st RTA*, volume 6 of *LIPIcs*, pages 33–48. Dagstuhl, 2010.

3 C. Fuhs, J. Giesl, M. Parting, P. Schneider-Kamp, and S. Swiderski. Proving Termination by Dependency Pairs and Inductive Theorem Proving. *JAR*, 47(2):133–160, 2011.

4 N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. In *Proc. of 4th IJCAR*, volume 5195 of *LNAI*, pages 364–380. Springer, 2008.

5 D. Hofbauer and C. Lautemann. Termination Proofs and the Length of Derivations. In *Proc. of 3rd RTA*, volume 355 of *LNCS*, pages 167–177. Springer, 1989.

6 J. Hoffmann, K. Aehlig, and M. Hofmann. Multivariate Amortized Resource Analysis. In *Proc. of 38th POPL*, pages 357–370. ACM, 2011.

7 A. Middeldorp and H. Ohsaki. Type Introduction for Equational Rewriting. *AI*, 36(12): 1007–1029, 2000.

8 TeReSe. *Term Rewriting Systems*, volume 55 of *CTTCS*. Cambridge University Press, 2003.

9 H. Zantema. Termination of Term Rewriting: Interpretation and Type Elimination. *JSC*, 17(1):23–50, 1994.