There Is No Best β-Normalization Strategy for Higher-Order Reasoners*

Alexander Steen and Christoph Benzmüller
a.steen|c.benzmueller@fu-berlin.de

Freie Universität Berlin, Institute of Computer Science

Abstract. The choice of data structures for the internal representation of terms in logical frameworks and higher-order theorem provers is a crucial low-level factor for their performance. We propose a representation of terms based on a polymorphically typed nameless spine data structure in conjunction with perfect term sharing and explicit substitutions.
In related systems the choice of a β-normalization method is usually statically fixed and cannot be adjusted to the input problem at runtime. The predominant strategies are hereby implementation specific adaptions of leftmost-outermost normalization. We introduce several different β-normalization strategies and empirically evaluate their performance by reduction step measurement on about 7000 heterogeneous problems from different (TPTP) domains.
Our study shows that there is no generally best β-normalization strategy and that for different problem domains, different best strategies can be identified. The evaluation results suggest a problem-dependent choice of a preferred β-normalization strategy for higher-order reasoning systems.

1 Introduction

Higher-order (HO) automated theorem proving (ATP) is, in many ways, more complex and involved than ATP in first-order or propositional logic. This additional complexity can be found on the proof search layer as well as on the layer of terms respectively formulas. However, one advantage is that the increased practical expressiveness of higher-order logic often enables more intuitive and concise problem representations and solutions. Many interactive and automated theorem provers for higher-order logic are based on Church's simple type theory [7] – also called classical higher-order logic (HOL) – or extensions of it.

In automated reasoning systems, terms are the most general and common pieces of information that are accessed, manipulated and created by most routines of the reasoning system. It is therefore not surprising, that the internal representation of terms is a crucial detail which has direct consequences on the efficiency of the whole system.

We present a combination of term representation techniques for HO ATP systems that is based on locally nameless spine terms [4] and explicit treatment

* This work has been supported by the German National Research Foundation (DFG) under grant BE 2501/11-1 (Leo-III).

of substitutions [1]. These base choices are appropriately adjusted to meet the requirements of HO ATP systems. In particular, our representation natively admits an expressive typing system, efficient term operations and reasonable memory consumption through term sharing in a combination that is novel to HO reasoners.

The support for efficient term operations hereby not only covers those adopted from the first-order universe, but also the essential operation of β-normalization. To this end, we differ from prominent other reasoning systems in proposing several new (modified) β-normalization strategies that allow a problem-dependent handling of β-reduction. Thus, we do not hard-wire a single, preferred β-normalization strategy that we anticipate to perform best over all possible problem inputs. We think that this approach can in fact increase the overall performance of HO ATP systems in which β-(re-)normalization has to be repeatedly carried out during proof search.

This research is motivated by previous observations [18] that suggest that there is no single best normalization strategy. The here proposed strategies have been empirically evaluated using a representative set of benchmark problems for theorem proving. This evaluation confirms that there are problem classes at which the de-facto standard leftmost-outermost strategy is outperformed by our rather simple alternative strategies. The evaluation has been conducted within the LeoPARD [21] system platform for HOL reasoners.[1]

2 HOL Term Representation

HOL is an elegant and expressive formal system that extends first-order logic with quantification over arbitrary sets and functions. We consider Alonzo Church's *simple type theory* [7] which is a formulation of HOL that is built on top of the simply typed λ-calculus [5,6].

The simply typed λ-calculus, denoted $\lambda_{\rightarrow}$, augments the untyped λ-calculus with *simple types*, which are freely generated from a set of base types and the function type constructor $\rightarrow$. In HOL, the set of base types is usually taken as a superset of $\{\iota, o\}$ with ι and o for individuals and truth values, respectively.

The work presented here focuses on an extended variant of $\lambda_{\rightarrow}$ that natively supports *parametric polymorphism* and incorporates a locally nameless representation using de-Bruijn indices for bound variables [3]. The notion of de-Bruijn indices is extended for nameless type variables to keep up the guarantee of syntactical uniqueness of α-equivalent terms. Types (denoted by τ or ν) are thus given by

$$\tau, \nu ::= s \in T \mid \underline{i} \in \mathbb{N} \mid \tau \rightarrow \nu \mid \forall .\, \tau$$

where T is a non-empty set of base type symbols and $\underline{i}$ is a nameless type variable.

The term data structure presented next adopts, combines and extends techniques that are employed in state-of-the-art HO reasoning systems, such as

[1] The LeoPARD framework is freely available under BSD license and can be downloaded at `https://github.com/cbenzmueller/LeoPARD`.

Teyjus λProlog [12] (which is based on explicit substitutions of the *Suspension Calculus* [11]), the logical frameworks *TWELF* [13] and *Beluga* [14], and the interactive *Abella* prover [8]. In particular, the combination of techniques for term data structures presented here is, up to our knowledge, novel in the context of HO ATP and not employed in any modern system.

On the basis of nameless terms, spine notation [4] in conjunction with explicit substitutions [1] is employed. The first technique allows quick head access and a left-to-right traversal method that is more efficient than in classical curried representation. The latter method's explicit treatment of substitutions enables the combination of substitution runs which in turn permits a more efficient β-normalization procedure.

More specifically, the internal representation of polymorphic HOL syntax is given by (types are partially omitted for simplicity):

$$\begin{aligned} s &::= (h \cdot S) \mid (s \cdot S) \mid (\lambda_\tau.\ s) \mid (\Lambda.\ s) \mid s[\sigma] \\ h &::= i_\tau \mid c_\tau \mid h[\sigma] \\ S &::= \textsc{Nil} \mid s_\tau ; S \mid \tau ; S \mid S[\sigma] \\ \sigma_{term} &::= \uparrow^i \ \mid s_\tau \cdot \sigma_{term} \qquad \sigma_{type} ::= \uparrow^i \ \mid \tau \cdot \sigma_{type} \end{aligned}$$

where the terms s are either *roots*, redexes, term and type abstraction, or closures (respectively) with *heads* h (that are bound indices i, constants $c_\tau \in \Sigma$ from the signature Σ or itself closures) and *spines* S. We support defined constants c_τ and their expansion using directed equation axioms ($c_\tau := d_\tau$). The spines collect arguments in a linear sequence, concatenated by the ; constructor. A substitution $\sigma = (\sigma_{term}, \sigma_{type})$ is internally represented by a pair of a term- and a type substitution, for which each individual substitution exclusively contains substitutes for the corresponding de-Bruijn indices. In the current version, closures cannot occur within types. This is because the number of type variables within current common ATP problems is typically very low (often zero), and, hence, merging of substitution runs in types is not crucial.

We extend the notion of β-normalization to substitutions $\sigma = (\sigma_{term}, \sigma_{type})$ by $\sigma{\downarrow_\beta} = (\sigma_{term}{\downarrow_\beta}, \sigma_{type})$ where $\sigma_{term}{\downarrow_\beta}$ denotes the substitution ρ for which it holds that $\rho(i) = \sigma_{term}(i){\downarrow_\beta}$, i.e. all components of the substitution are β-normalized individually.

The type abstraction mechanism ($\Lambda.\ s$) is due to Girard and Reynolds, who independently developed a polymorphically typed λ-calculus today widely known as System F [9,16]. We use a *Church-style* λ-calculus in which each type is considered a part of the term's name and thus intrinsic to it. This has several advantages over the extrinsic, or *Curry-style*, interpretation, but comes with some downsides, e.g., wrt. typing flexibility.

3 Normalization Strategies

We now introduce corresponding strategies, two of them novel (wrt. earlier experiments in [18]), and present them along with a brief discussion of possible benefits (and downsides). Subsequently, these strategies are empirically evaluated using an extensive benchmark set. The strategies are:

1. **DEFAULT** *(Leftmost-outermost)*: This normalization method corresponds to the standard *normal-order* strategy, that is, the leftmost-outermost redex is processed first at each step during β-normalization. We use **DEFAULT** as starting point for the presentation and explanation of further strategies below. The complete rules for **DEFAULT** can be found in Fig. 1. Here, $s{\downarrow}^{\sigma}_{\beta}$ denotes β-normalization relative to substitution σ. The computation of the β-normal form of term s is initiated by $s{\downarrow}_{\beta} := s{\downarrow}^{(id,id)}_{\beta}$, where id is the identity substitution $id := \uparrow^0$.
2. **HSUBST**n *($n > 0$, Heuristic application of substitution in* RxApp*)*: If the size of the term to be prepended onto the substitution is smaller than n, it is normalized strictly. Otherwise, the substitution is postponed using closures as before. The rule RxApp from Fig. 1 is thus replaced by the two rules

$$\frac{(s \cdot t; S_{tail}) \downarrow^{\sigma,\sigma'}_{\beta} \qquad s = \lambda_{\tau}.\, s' \qquad |t| \geq n}{(s' \cdot S_{tail}) \downarrow^{(t[\sigma]\cdot\sigma_{term},\sigma_{type}),\sigma'}_{\beta}}\ \textsc{RxApp}_{\geq}$$

$$\frac{(s \cdot t; S_{tail}) \downarrow^{\sigma,\sigma'}_{\beta} \qquad s = \lambda_{\tau}.\, s' \qquad |t| < n}{(s' \cdot S_{tail}) \downarrow^{(t\downarrow^{\sigma}_{\beta}\cdot\sigma_{term},\sigma_{type}),\sigma'}_{\beta}}\ \textsc{RxApp}_{<}$$

   where $|t|$ denotes the *size of term t* (i.e. the number of term nodes in internal representation).
3. **WHNF** *(Normalize substitution once WHNF is obtained)*: When arrived at weak head normal form $c{\cdot}S$ of the current (sub-)term during β-normalization, the substitution σ is normalized and then used to further β-normalize the spine S. Thus, the rule RAtom (cf. Fig. 1) is replaced by

$$\frac{(c \cdot S) \downarrow^{\sigma}_{\beta} \qquad c \in \Sigma \qquad \sigma' = \sigma{\downarrow}_{\beta}}{c \cdot S{\downarrow}^{\sigma'}_{\beta}}\ \textsc{RAtom}'$$

4. **STRCOMP** *(Strict composition of term-substitutions)*: The standard (meta-operation) of term-substitution composition with closures is given by

$$(s_{\tau} \cdot \sigma_{term}) \circ \rho_{term} \longrightarrow s_{\tau}[\rho_{term}] \cdot (\sigma_{term} \circ \rho_{term}) \tag{1}$$

   In **STRCOMP** it is instead calculated strictly:

$$(s_{\tau} \cdot \sigma_{term}) \circ \rho_{term} \longrightarrow s_{\tau}{\downarrow}^{(\rho_{term},id)}_{\beta} \cdot (\sigma_{term} \circ \rho_{term}) \tag{2}$$

   In contrast to (1), the application of substitution ρ_{term} in (2) is not postponed using closures but applied immediately by β-normalization.

Root rules

$$\frac{(c\cdot S)\downarrow^{\sigma}_{\beta} \qquad c\in\Sigma}{c\cdot S\downarrow^{\sigma}_{\beta}}\ \textsc{RAtom}$$

$$\frac{(i_\tau\cdot S)\downarrow^{\sigma}_{\beta} \qquad \sigma_{term}(i)=j}{j_{\tau[\sigma_{type}]}\cdot S\downarrow^{\sigma}_{\beta}}\ \textsc{RBndSub} \qquad \frac{(i\cdot S)\downarrow^{\sigma}_{\beta} \qquad \sigma_{term}(i)=s}{(s\cdot S)\downarrow^{(id,\sigma_{type}),\sigma}_{\beta}}\ \textsc{RTermSub}$$

$$\frac{(h[\rho'][\rho]\cdot S)\downarrow^{\sigma}_{\beta}}{(h[\rho'\circ\rho]\cdot S)\downarrow^{\sigma}_{\beta}}\ \textsc{RClosClos} \qquad \frac{(c[\rho]\cdot S)\downarrow^{\sigma}_{\beta} \qquad c\in\Sigma}{c\cdot S\downarrow^{\sigma}_{\beta}}\ \textsc{RAtomClos}$$

$$\frac{(i_\tau[\rho]\cdot S)\downarrow^{\sigma}_{\beta} \qquad (\rho_{term}\circ\sigma_{term})(i)=j}{j_{\tau[\rho_{type}\circ\sigma_{type}]}\cdot S\downarrow^{\sigma}_{\beta}}\ \textsc{RBndClos}$$

$$\frac{(i[\rho]\cdot S)\downarrow^{\sigma}_{\beta} \qquad (\rho_{term}\circ\sigma_{term})(i)=s}{(s\cdot S)\downarrow^{(id,\rho_{type}\circ\sigma_{type}),\sigma}_{\beta}}\ \textsc{RTermClos}$$

Abstraction/Closure rule

$$\frac{(\lambda_\tau.\ s)\downarrow^{\sigma}_{\beta}}{\lambda_{\tau[\sigma_{type}]}.\ s\downarrow^{(1\cdot\sigma_{term}\circ\uparrow,\sigma_{type})}_{\beta}}\ \textsc{Abs} \qquad \frac{(\Lambda.\ s)\downarrow^{\sigma}_{\beta}}{\Lambda.\ s\downarrow^{(\sigma_{term},1\cdot\sigma_{type}\circ\uparrow)}_{\beta}}\ \textsc{TyAbs} \qquad \frac{(s[\sigma'])\downarrow^{\sigma}_{\beta}}{s\downarrow^{\sigma'\circ\sigma}_{\beta}}\ \textsc{Clos}$$

Redex rules

$$\frac{(s\cdot \textsc{Nil})\downarrow^{\sigma,\sigma'}_{\beta}}{s\downarrow^{\sigma}_{\beta}}\ \textsc{RxSpNil} \qquad \frac{(s\cdot S[\rho])\downarrow^{\sigma,\sigma'}_{\beta}}{(s\cdot S)\downarrow^{\sigma,\rho\circ\sigma'}_{\beta}}\ \textsc{RxSpClos}$$

$$\frac{(s\cdot t;S_{tail})\downarrow^{\sigma,\sigma'}_{\beta} \qquad s=\lambda_\tau.\ s'}{(s'\cdot S_{tail})\downarrow^{(t[\sigma]\cdot\sigma_{term},\sigma_{type}),\sigma'}_{\beta}}\ \textsc{RxApp} \qquad \frac{(s\cdot \tau;S_{tail})\downarrow^{\sigma,\sigma'}_{\beta} \qquad s=\Lambda.\ t}{(t\cdot S_{tail})\downarrow^{(\sigma_{term},\tau[\sigma'_{type}]\cdot\sigma_{type}),\sigma'}_{\beta}}\ \textsc{RxTyApp}$$

$$\frac{(s\cdot S)\downarrow^{\sigma,\sigma'}_{\beta} \qquad s=h\cdot S'}{(h[\sigma]\cdot S'[\sigma] +\!\!+ S[\sigma'])\downarrow^{(id,id)}_{\beta}}\ \textsc{RxRMrg} \qquad \frac{(s\cdot S)\downarrow^{\sigma,\sigma'}_{\beta} \qquad s=t\cdot S'}{(t\cdot S'[\sigma] +\!\!+ S[\sigma'])\downarrow^{\sigma,(id,id)}_{\beta}}\ \textsc{RxRxMrg}$$

$$\frac{(s\cdot S)\downarrow^{\sigma,\sigma'}_{\beta} \qquad s=t[\rho]}{(t\cdot S)\downarrow^{\rho\circ\sigma,\sigma'}_{\beta}}\ \textsc{RxClos}$$

Spine rules

$$\frac{\textsc{Nil}\downarrow^{\sigma}_{\beta}}{\textsc{Nil}}\ \textsc{SpNil} \qquad \frac{(S[\rho])\downarrow^{\sigma}_{\beta}}{S\downarrow^{\rho\circ\sigma}_{\beta}}\ \textsc{SpClos} \qquad \frac{(s_0;S_{tail})\downarrow^{\sigma}_{\beta}}{(s_0\downarrow^{\sigma}_{\beta});S_{tail}\downarrow^{\sigma}_{\beta}}\ \textsc{SpApp} \qquad \frac{(\tau;S_{tail})\downarrow^{\sigma}_{\beta}}{(\tau[\sigma_{type}]);S_{tail}\downarrow^{\rho\circ\sigma}_{\beta}}\ \textsc{SpTyApp}$$

Fig. 1: β-normalization strategy **DEFAULT**

5. **WEAK** *(Weakly normalize substitutions on demand)*: Before application of RTermSub or RTermClos, β-normalize the term before substituting, and

update σ accordingly. This means that each time a term is supposed to be substituted, its β-normal form is substituted instead. Also, in order to avoid re-computations, the original term is replaced by its β-normal form in the substitution σ, too. Thus, rule RTermSub from Fig. 1 is replaced by

$$\frac{(i \cdot S) \downarrow_\beta^\sigma \qquad \sigma_{term}(i) = s \qquad s' = s{\downarrow_\beta}}{(s' \cdot S) \downarrow_\beta^{(id,\sigma_{type}),(\sigma_{term}[i \leftarrow t{\downarrow_\beta}],\sigma_{type})}} \text{ RTermSub}'$$

and RTermClos is replaced analogously. Here, when term t is substituted for de-Bruijn index i, the substitution σ is updated to hold the normalized t at position i, i.e. $\sigma'(j) = t{\downarrow_\beta}$ iff $j = i$ and $\sigma'(j) = \sigma(j)$ otherwise.

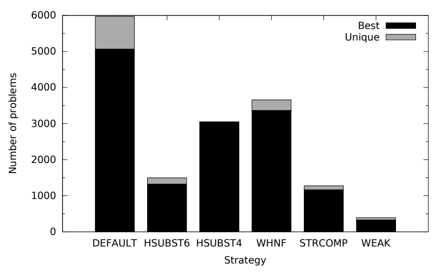
4 Evaluation and Further Work

In order to estimate the expected effects of using different β-normalization strategies in practical scenarios of automated reasoning, a worst-case analysis seems inappropriate and is therefore omitted. In lieu thereof, a representative set of problems for (HO) theorem proving has been chosen for which the number of β-normalization reduction steps has been compared empirically between all strategies. Since the proposed strategies do not include costly heuristics (e.g. based on structural properties of terms), a decrease in reduction counts can directly be translated to a speed-up with respect to actual time consumption. The evaluation has been conducted with the LeoPARD system platform, in which the term data structures from §2 and the strategies from §3 have been implemented.

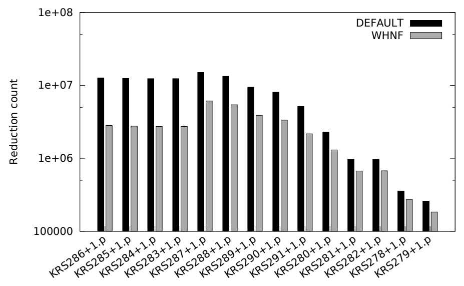
The benchmarks. The benchmark problems were chosen from a relatively broad field of diversity: The first three benchmark domains are the sets denoted Church I, Church II and Church III that contain reducible arithmetic terms (of the form $mult(i,i)$, $power(i,3)$, $power(3,i)$ respectively) in polymorphic Church numerals encoding [17]. The domains S4E and S4F contain a total of 3480 HO problems, converted from propositional and first-order modal logic problems from the QMLTP library [15]. Both domains differ wrt. to the details of the employed semantic embedding of logic S4 in HOL [2].[2]

The remaining benchmarks (a total of 3246 problems) are (typed) first-order and HO problems from the TPTP problem library [19,20]. These benchmark domains are denoted according to their problem domain name as given by the TPTP library. Generally, first-order CNF problems, as well as TPTP domains that only contain them, were not considered for the evaluation, since the contained formulae are already given in clause normal form which results in likewise β-normalized internal clause representations in LeoPARD.

The benchmark problem selection embodies, in its sum, a representative set of nearly 7000 practical inputs for reasoning systems and a heterogeneous set of (syntactic and semantic) term characteristics is covered.



(a) Distribution of best performance over all benchmark problems



(b) Highest reduction count differences in problem domain KRS

Fig. 2: Evaluation results

Results and Discussion. Fig. 2a shows the number of benchmark problems (throughout all domains) that were β-normalized (uniquely) best using the given strategy. It can be seen that, in our benchmark set, the **DEFAULT** strategy has the higher number of problems normalized with minimal reduction count (compared to the other strategies). Nevertheless, **HSUBST4** and **WHNF** are competitive alternatives, and there are even problems that are uniquely normalized best in the remaining strategies. It should be pointed out again that the competing strategies are relatively simple, since they do not use sophisticated term structure heuristics and yet already admit a fair effectiveness in certain domains.

In order to give a brief idea of the amount of potential reduction count savings, a quantitative comparison of 14 problems from KRS with highest reduction count differences between default leftmost-outermost and the alternative **WHNF** strategy is shown in Figure 2b. These difference are, in the most striking cases, up to factor 4.5 which is considerable in magnitudes of 10^6 reduction steps and above.

More detailed results that underline our observations can be found in Table 1. Here, for selected problem domains[3], and each relevant β-normalization strategy, the number of problems that performed best and worst are displayed (i.e. the number of problems that had the lowest respectively highest overall reduction count for this strategy). Additionally, the number of unique problems – denoted (u) – which normalized strictly faster in this strategy than in any other strategy within the domain is given. The sum of all reduction steps, denoted Σr_i, throughout the whole problem domain, as well as the maximal number of reduction steps (for a single problem) are given. The remaining three values, $\widetilde{r_i}$, $\overline{r_i}$ and

[2] The archive of semantically embedded S4-formulae from QMLTP can be found at `http://page.mi.fu-berlin.de/cbenzmueller/papers/THF-S4-ALL.zip`

[3] The complete evaluation results can be found at `http://inf.fu-berlin.de/~lex/files/betaresults.pdf`

σ, denote the arithmetic mean, the median value and the standard derivation of the measurement results (respectively).

Strategy	Best (u)	Worst (u)	Σr_i	$\min r_i$	$\max r_i$	$\overline{r_i}$	$\widetilde{r_i}$	σ
STRCOMP	88 (88)	3 (3)	1151350	79	32650	$11{\cdot}10^3$	9014.5	$9{\cdot}10^3$
DEFAULT	12 (0)	0 (0)	1712750	17	50408	17127.5	12962.5	$15{\cdot}10^3$
WHNF	12 (0)	0 (0)	1712750	17	50408	17127.5	12962.5	$15{\cdot}10^3$
HSUBST4	0 (0)	0 (0)	1712850	18	50409	17128.5	12963.5	$15{\cdot}10^3$
HSUBST6	0 (0)	0 (0)	1733050	22	50809	17330.5	13165.5	$15{\cdot}10^3$
WEAK	0 (0)	97 (0)	39838425	33	1546215	398384.2	205336.5	$445{\cdot}10^3$

(a) Domain CHURCH1 (100 problems)

Strategy	Best (u)	Worst (u)	Σr_i	$\min r_i$	$\max r_i$	$\overline{r_i}$	$\widetilde{r_i}$	σ
STRCOMP	17 (17)	0 (0)	147516	150	25782	7764.0	4704.0	$7{\cdot}10^3$
DEFAULT	2 (0)	0 (0)	236075	80	42110	12425.0	7325.0	$12{\cdot}10^3$
WHNF	2 (0)	0 (0)	236075	80	42110	12425.0	7325.0	$12{\cdot}10^3$
HSUBST6	0 (0)	0 (0)	1262759	107	271331	66461.0	27665.0	$80{\cdot}10^3$
HSUBST4	0 (0)	0 (0)	1262759	107	271331	66461.0	27665.0	$80{\cdot}10^3$
WEAK	0 (0)	1 (0)	1359621	171	289215	71559.0	30483.0	$86{\cdot}10^3$

(b) Domain CHURCH2 (19 problems)

Strategy	Best (u)	Worst (u)	Σr_i	$\min r_i$	$\max r_i$	$\overline{r_i}$	$\widetilde{r_i}$	σ
WHNF	14 (0)	0 (0)	704503	511	14846	5636.0	5378.0	$3{\cdot}10^3$
HSUBST6	0 (0)	0 (0)	834604	513	22883	6676.8	5475.0	$5{\cdot}10^3$
DEFAULT	80 (16)	0 (0)	848536	511	23663	6788.3	5472.0	$5{\cdot}10^3$
HSUBST4	50 (0)	0 (0)	848599	513	23663	6788.8	5472.0	$5{\cdot}10^3$
STRCOMP	14 (0)	3 (0)	8443020	511	419068	67544.2	13622.0	$106{\cdot}10^3$
WEAK	0 (0)	14 (0)	23354287	913	1069897	186834.3	89193.0	$252{\cdot}10^3$

(c) Domain GRA (125 problems)

Strategy	Best (u)	Worst (u)	Σr_i	$\min r_i$	$\max r_i$	$\overline{r_i}$	$\widetilde{r_i}$	σ
WHNF	173 (14)	95 (0)	35695447	0	6106161	129801.6	689.0	$67{\cdot}10^4$
HSUBST6	96 (0)	85 (0)	106292434	0	15011396	386517.9	835.0	$206{\cdot}10^4$
DEFAULT	254 (12)	95 (0)	106316948	0	15028663	386607.1	689.0	$206{\cdot}10^4$
HSUBST4	239 (0)	109 (14)	106317316	0	15028665	386608.4	689.0	$206{\cdot}10^4$
STRCOMP	*Unfeasible*							
WEAK	*Unfeasible*							

(d) Domain KRS (275 problems)

Table 1: Selected results of reduction count measurements

As an example, in benchmark domain CHURCH I (cf. Table 1a) **STRCOMP** performs drastically better than in any other domain: Although **DEFAULT** and **WHNF** have the lowest minimum value, **STRCOMP** is by far the best strategy (in problem count and overall reduction sum) with 88 of 100 problems (uniquely) normalized best. In terms of reduction steps per problem, **STR-COMP** takes only roughly 70% of the number of steps required by **DEFAULT** (in both average and mean). Similar results also apply for the remaining Church domains. Also, in the GRA domain (cf. Table 1c), the mean normalization step count $\overline{r_i}$ is more than 100 steps lower in the **WHNF** strategy than when using **DEFAULT**. These results demonstrate that the alternative normalization strategies can in fact perform better (wrt. reduction count per problem) than default leftmost-outermost in certain problem domains.

Further Work. While the present evaluation grouped problems by a (given, practically motivated) semantic classification, further investigations need to identify syntactic criteria in order to group problems with similar properties (with respect to β-normalization performance) for a specific strategy.

Based on observation and some preliminary experiments, we are positive that methods based on syntactic criteria such as the following can be employed for choosing an appropriate normalization strategy at runtime:

- Recognition of regular patterns in terms
- The term's size and depth
- The number of abstractions not occurring at top-level
- The number of bound indices

For future work, not only concrete (syntactical) heuristics but also machine learning techniques could be employed to study representative sets of problems.

5 Conclusion

A sophisticated internal representation mechanism for (second-order) polymorphically typed HO terms, including a locally nameless spine notation combined with explicit substitutions and perfect term sharing, has been presented.

Using the above representation, several new β-normalization strategies have been introduced. These strategies vary in their extent of laziness and strictness in certain normalization rules, e.g. during composition of substitutions. They have subsequently been implemented and evaluated within the LeoPARD framework. The conducted evaluation was based on a representative benchmark set.

For logical frameworks and meta languages, the representation of objects such as programs and proofs in λProlog has previously been studied [10]. However, a fine-grained evaluation of normalization strategies in context of HO ATP as reported here has not been carried out before. Extending previous studies in a rather orthogonal manner (wrt. application domain, granularity, and system of explicit substitutions), our benchmarks reveal that there is no single best β-normalization strategy for a relevant set of problem classes. In particular, our findings show that the performance of a strategy rather depends on some (syntactic) characteristics of the input problem. The reduction count difference between the default leftmost-outermost strategy and the leading strategy can, in fact, be as high as factor four.

References

1. Abadi, M., Cardelli, L., Curien, P.L., Levy, J.J.: Explicit substitutions. In: Proc. of the 17th Symp. on Principles of Programming Languages. pp. 31–46. POPL '90, ACM, New York, NY, USA (1990)
2. Benzmüller, C., Raths, T.: HOL based first-order modal logic provers. In: LPAR. LNCS, vol. 8312, pp. 127–136. Springer (2013)
3. Bruijn, N.G.D.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. INDAG. MATH 34, 381–392 (1972)
4. Cervesato, I., Pfenning, F.: A linear spine calculus. J. Logic and Computation 13(5), 639–688 (2003)
5. Church, A.: A Set of Postulates for the Foundation of Logic. Annals of Mathematics 33(2), 346–366 (1932)
6. Church, A.: A Set of Postulates for the Foundation of Logic, Second Paper. The Annals of Mathematics 34(4), 839–864 (Oct 1933)
7. Church, A.: A formulation of the simple theory of types. J. Symb. Log. 5(2) (1940)
8. Gacek, A.: The Abella interactive theorem prover (system description). In: Automated Reasoning, IJCAR. LNCS, vol. 5195, pp. 154–161. Springer (2008)
9. Girard, J.: Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Ph.D. thesis, Université Paris VII (1972)
10. Liang, C., Nadathur, G., Qi, X.: Choices in representation and reduction strategies for lambda terms in intensional contexts. J. Autom. Reasoning 33(2), 89–132 (2004)
11. Nadathur, G.: A fine-grained notation for lambda terms and its use in intensional operations. Journal of Functional and Logic Programming 1999(2) (1999)
12. Nadathur, G., Mitchell, D.: System Description: Teyjus - A Compiler and Abstract Machine Based Implementation of λProlog. In: Automated Deduction, CADE, LNAI, vol. 1632, pp. 287–291. Springer (1999)
13. Pfenning, F., Schürmann, C.: System description: Twelf - A meta-logical framework for deductive systems. In: Automated Deduction, CADE. LNAI, vol. 1632, pp. 202–206. Springer (1999)
14. Pientka, B., Dunfield, J.: Beluga: A framework for programming and reasoning with deductive systems (system description). In: Automated Reasoning, IJCAR. LNCS, vol. 6173, pp. 15–21. Springer (2010)
15. Raths, T., Otten, J.: The qmltp problem library for first-order modal logics. In: Automated Reasoning, IJCAR, LNCS, vol. 7364, pp. 454–461. Springer (2012)
16. Reynolds, J.C.: Towards a theory of type structure. In: Symposium on Programming. LNCS, vol. 19, pp. 408–423. Springer (1974)
17. Reynolds, J.C.: An introduction to polymorphic lambda calculus. In: Logical Foundations of Functional Programming. pp. 77–86. Addison-Wesley (1994)
18. Steen, A.: Efficient Data Structures for Automated Theorem Proving in Expressive Higher-Order Logics. Master's thesis, Freie Universität Berlin, Berlin (2014)
19. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. Journal of Automated Reasoning 43(4), 337–362 (2009)
20. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. Journal of Formalized Reasoning 3(1), 1–27 (2010)
21. Wisniewski, M., Steen, A., Benzmüller, C.: LeoPARD — A Generic Platform for the Implementation of Higher-Order Reasoners. In: Intelligent Computer Mathematics, CICM. LNCS, vol. 9150, pp. 325–330. Springer (2015)