Smyth Experiment Tables

ICFP 2020 Artifact Evaluation (May 22, 2020)

Changes to SMYTH. Since the submission, we have made two algorithmic improvements:

- (1) Change to search order: trying to synthesize case expressions before constructors (still subject to the same staging parameters). This change tends to improve performance and produce more readable solutions for some benchmarks.
- (2) From Section 6.3: "Future work should consider how to automatically reconfigure staging parameters to account for the structure of user-provided sketches." We have implemented such "responsive" staging parameters.

We have also since removed the client-side portion of our implementation that had been written in Elm. The new (command-line) interface is written in OCaml.

Changes to Synquid. From Section 6.4: "Discussion with the authors of Synquid has revealed an implementation issue involving the axiomatization of recursive datatypes in the underlying logic. As a result, desired solutions for many benchmarks—even non-recursive ones—failed to typecheck. When this issue is addressed, Synquid may very well synthesize many of these tasks (conservatively marked?)."

Nadia Polikarpova has recently fixed this issue (https://github.com/nadia-polikarpova/synquid/commit/20e6d62e151e314cc0e5b36983fded12e6e6c8c4), so we re-ran the Synquid experiments.

Updated Figure 10 Results. Based on the above, our Figure 10 results have changed slightly. The following page replicates Figure 10 from our submission. Then Tables 1, 2, 3, and 4 describe the differences for each of the four experiments.

Tables 1, 2, and 3 include a *Revised Artifact (Ours)* column to show our updated results, using blue highlights to emphasize differences compared to the *Submission*. The *Revised Artifact (Yours)* column shows the results that you obtained on your machine; differences from ours are highlighted in red. If you have not run the experiments, the column should be filled with dots.

LEON and SYNQUID Benchmarks (experiments/exp-4-logic/): For Experiment 4, we wrote a script (generate-benchmarks.py) to generate LEON and SYNQUID tasks (generated/), which we copied and pasted into the respective web editors. Our labeled results are in results/. The helper script show-results. sh summarizes the labels, and also includes comments with some shell commands we found useful during this experiment. The sketching benchmarks we tried (discussed on lines 1033-1034 of the submission) can be found in sketches/.

Leon: If https://leon.epfl.ch/ doesn't work, you might try https://leon.epfl.ch/, a URL that Viktor Kuncak provided. In either case, look for the "Synthesis" menu on the right and click "Search."

SYNQUID: The web editor http://comcom.csail.mit.edu/comcom/#Synquid currently serves the version with Nadia's recent changes. Nadia suggested running our benchmarks with the -e=True flag; we did, but found that -e=False was needed for a few benchmarks to succeed.

.

Ermanin	1	1	SMYTH	214	20	3b		ON 4	_	QUID 4
Experiment			l	2a 2b				4		4
Sketch			None		Base Case 24/28 rec. benchmarks					
#Benchmarks	T-		TH benchmarks							
Objective		p-1		op-1		p-1-R	1	0 -	-	0.
Name	Expert	Time	Expert	Random (50%, 90%)	Expert	Random (50%, 90%)	1	2a	1	2a
bool band	4	0.004	3 (75%)	(4, 4)	_	_	/	/	/	/
bool bor	4	0.004	3 (75%)	(4, 4)	_	_	/	/	/	/
bool impl	4	0.005	3 (75%)	(3, 4)	_	_	/	/	/	/
bool neg	2	0.002	2 (100%)	(2, 2)	_	_	1	/	/	/
bool_xor	4	0.011	3 (75%)	(4, 4)	_	-	1	X^1	1	✓
list_append	6	0.008	4 (67%)	(3, 5)	1+1 (33%)	(1+3, 1+5)	1	\mathbf{X}^{1}	?	\mathcal{X}^4
list_compress	13	timeout	_	_	_	_	\mathbf{x}^2	_	?	_
list_concat	6	0.008	3 (50%)	(2, 3)	1+2 (50%)	(1+2, 1+3)	1	X^1	?	\mathcal{X}^4
list_drop	11	0.030	5 (45%)	(6, 9)	1+2 (27%)	(1+8, 1+15)	/	/	?	\mathcal{X}^4
list_even_parity	7	0.051	5 (71%)	failed	overspec	failed	1	\mathcal{X}^1	?	\mathcal{X}^4
list_filter	8	0.130	4 (50%)	_	overspec	_	X ³	χ^3	X ³	χ^3
list_fold	9	0.765	3 (33%)	_	1+3 (44%)	_	X ³	χ^3	X 3	χ^3
list_hd	3	0.003	2 (67%)	(2, 3)	_	_	✓	/	?	\mathcal{X}^1
list_inc	4	0.184	2 (50%)	(2, 2)	_	_	✓	/	?	?
list_last	6	0.007	4 (67%)	(5, 12)	1+2 (50%)	(1+5, 1+8)	1	/	?	\mathcal{X}^4
list_length	3	0.003	3 (100%)	(2, 3)	1+1 (67%)	(1+2, 1+3)	/	_	?	_
list_map	8	0.039	4 (50%)	_	1+2 (38%)	_	X ³	χ^3	X 3	X ³
list_nth	13	0.113	5 (38%)	(8, 15)	1+2 (23%)	(1+8, 1+16)	/	/	?	χ^4
list_pairwise_swap	7	4.229	5 (71%)	timeout	overspec	timeout	/	/	?	χ^4
list_rev_append	5	0.097	3 (60%)	(5, 9)	1+2 (60%)	(1+3, 1+14)	1	/	?	X^4
list_rev_fold	5	0.027	2 (40%)	(2, 3)	_	_	/	/	?	?
list_rev_snoc	5	0.009	3 (60%)	(3, 7)	1+1 (40%)	(1+3, 1+5)	√	/	?	χ^4
list_rev_tailcall	8	0.007	3 (38%)	(3, 6)	1+1 (25%)	(1+3, 1+6)	X ¹	/	?	X ⁴
list_snoc	8	0.012	4 (50%)	(3, 4)	1+2 (38%)	(1+2, 1+4)	1	/	?	X ⁴
list_sort_sorted_insert	7	0.015	3 (43%)	(3, 6)	1+1 (29%)	(1+3, 1+6)	√	√	?	χ^4
list_sorted_insert	12	10.964	7 (58%)	timeout	overspec	timeout	\mathbf{X}^2	χ^2	?	χ^4
list_stutter	3	0.004	2 (67%)	(3, 3)	1+1 (67%)	(1+3, 1+4)	1	/	?	χ^4
list_sum	3	0.023	2 (67%)	(2, 3)			1	X ¹	?	?
list_take	12	0.075	6 (50%)	(7, 10)	1+3 (33%)	(1+8, 1+15)	1	1	?	X ⁴
list_tl	3	0.003	2 (67%)	(2, 3)	_	_	1	1	X ¹	X ¹
nat_add	9	0.007	4 (44%)	(4, 6)	1+1 (22%)	(1+4, 1+6)	1	/	?	\mathcal{X}^4
nat_iseven	4	0.004	3 (75%)	(3, 4)	1+2 (75%)	(1+3, 1+4)	1	/	?	\mathcal{X}^4
nat_max	9	0.039	9 (100%)	(9, 11)	1+4 (56%)	(1+8, 1+12)	\mathcal{X}^1	_	?	_
nat_pred	3	0.002	2 (67%)	(2, 3)	_	_	1	✓	X ¹	X ¹
tree_binsert	20	timeout	_		_	_	X ²	_	?	_
tree_collect_leaves	6	0.066	3 (50%)	$(3, 4)^3$	1+2 (50%)	(1+3, 1+3)	/	/	?	χ^4
tree_count_leaves	7	3.009	3 (43%)	timeout	1+1 (29%)	timeout	✓	/	?	X ⁴
tree_count_nodes	6	0.323	3 (50%)	$(4, \downarrow)^{10}$	1+2 (50%)	$(1+4, 1+5)^{10}$	/	/	?	X ⁴
tree_inorder	5	0.114	4 (80%)	(3, 4)	1+2 (60%)	(1+3, 1+3)	√	√	?	χ^4
tree_map	7	0.055	4 (57%)	_	1+3 (57%)	_	X ³	χ^3	X 3	χ^3
tree_nodes_at_level	11	timeout	_	_	_	_	X ²	_	?	_
tree_postorder	20	timeout			_	_	✓	_	?	
tree_preorder	5	0.145	3 (60%)	$(3, 3)^3$	1+2 (60%)	(1+3, 1+3)	/	✓	?	X^4
Averages			61%*		45%					

Fig. 10. Experiments. **Top-1(-R)**: 1st (recursive) solution valid. **Time**: Average of 10 runs, in seconds. **Averages**: Non-blank, non-error rows. *Upper bound: 65% for all 43 rows.

$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$			nent 1				
Fig. 10 Ours Yours Fig. 10 Ours Yours	Name	Expert			Time		
Bool_band		Submission:	Revised	Artifact:	Submission:	Revised	Artifact:
bool_bor d		Fig. 10	Ours	Yours	Fig. 10	Ours	Yours
bool_impl 4	bool_band	4	4	•	0.004	0.004	•
bool_neg 2 2 0.002 0.001 bool_xor list_append 6 6 0.008 0.008 list_compress 13 • timeout • timeout • timeout • 0.008 0.010 list_concat 6 6 0.0051 • 0.011 1 1 1 0.003 0.003 1 1 1 1 0.051 • 0.051 • 0.051 • 0.051 • 0.051 • 0.003 0.003 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	bool_bor	4	4	•	0.004	0.003	•
bool_xor	bool impl	4	4	•	0.005	0.004	•
State Stat	bool_neg	2	2	•	0.002	0.001	•
list_concat	bool_xor	4	4	•	0.011	0.009	•
list_concat 6 6 0.008 0.010 list_drop 11 11 0.030 0.092 list_drop 11 11 0.030 0.092 list_filter 8 9 0.130 0.144 list_filter 8 9 0.130 0.144 list_filter 8 9 0.130 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.007 0.008 list_tery_strain_strain_strain_strain_strain_strain_strai	list_append	6	6	•	0.008	0.008	•
list_drop 11 11 0.030 0.092 list_even_parity 7 • 0.051 • • list_filter 8 9 0.130 0.144 list_fold 9 9 0.765 0.838 list_list_fold 9 9 0.765 0.838 list_list_fold 3 3 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.007 0.008 0.007 0.008 0.007 0.008 0.007 0.008 0.007	list_compress	13	•	•	timeout	•	•
list_even_parity 7 • 0.051 • • list_filter 8 9 0.130 0.144 • list_fold 9 9 0.765 0.838 • list_fold 9 9 0.765 0.838 • list_fold 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.007 0.008 list_rev_fold 5 5 0.009 0.010 <td< td=""><td>list_concat</td><td>6</td><td>6</td><td>•</td><td>0.008</td><td>0.010</td><td>•</td></td<>	list_concat	6	6	•	0.008	0.010	•
list_filter 8 9 0.130 0.144 list_fold 9 9 0.765 0.838 list_hd 3 3 0.003 0.003 0.003 list_list_list 6 6 6 0.007 0.007 0.007 list_list_list_list_list_list_list_list_	list_drop	11	11	•	0.030	0.092	•
list_fold 9 9 0.765 0.838 list_hd 3 3 0.003 0.003 list_linc 4 4 0.184 0.018 list_linc 4 4 0.184 0.018 list_linc 4 4 0.184 0.003 0.007 0.007 list_list_linc 4 4 0.184 0.018 list_list_linc 4 4 0.184 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.009 0.019 0.113 0.124 1 0.012 0.014 0.024 0.034 0.007 0.034 0.007 0.034 0.007 0.003 0.007 0.008 0.007 0.008 0.007 0.008 0.007 0.008 0.007 0.008 0.007 0.008 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001	list_even_parity	7	•	•	0.051	•	•
list_hd 3 3 0.003 0.003 lound list_inc 4 4 0.184 0.018 list_last 6 6 0.007 0.007 0.007 list_last 6 6 0.007 0.007 0.007 list_last 6 6 6 0.007 0.007 0.007 0.002 list_last 13 3 0.003 0.002 0.049 list_last 13 13 0.113 0.124 list_last 13 13 0.113 0.124 list_last_pairwise_swap 17 7 4.229 0.634 list_strey_append 5 5 0.097 0.107 0.034 list_strey_fold 5 5 0.097 0.107 0.035 list_strey_fold 5 5 0.027 0.035 list_strey_fold 8 8 0.0027 0.035 list_strey_fold 8 8 0.0027 0.008 list_strey_fold 8 8 0.002 0.001 list_strey_fold 0.002 0.001 0.002	list_filter	8	9	•	0.130	0.144	•
list_inc 4 4 0.184 0.018 list_last 6 6 0.007 0.007 list_length 3 3 0.003 0.002 list_length 3 3 0.003 0.002 list_map 8 8 0.039 0.049 list_not 13 13 0.113 0.124 list_pairwise_swap 7 7 4.229 0.634 list_rev_append 5 5 0.097 0.107 list_rev_snoc 5 5 0.027 0.035 list_rev_snoc 5 5 0.009 0.010 list_snoc 8 8 0.012 0.012 list_snot 8 8 0.012 0.012 list_sort_sorted_insert 12 12 12 10.064 2.902 list_stutter 3 3 0.023 0.029 13 list_stutter 3 3 0.023 0.029 13<	list_fold	9	9	•	0.765	0.838	•
list_last 6 6 0.007 0.007 0.007 list_length 3 3 0.003 0.002 0.049 0.012 0.012 0.012 0.012 0.012 0.012 0.012 0.027 0.035 0.041 0.027 0.035 0.007 0.005 0.009 0.010 0.005 0.009 0.010 0.000 0.000 0.010 0.002 0.001 0.002 0.002 0.001 0.002 0.002 0.001 0.002 0.015 0.002 0.015 0.002 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.002 0.002 0.002 0.003 <td>list_hd</td> <td>3</td> <td>3</td> <td>•</td> <td>0.003</td> <td>0.003</td> <td>•</td>	list_hd	3	3	•	0.003	0.003	•
list_length 3 3 0.003 0.002 list_map 8 8 0.039 0.049 list_nth 13 13 0.113 0.124 list_nth 13 13 0.113 0.124 list_rev_and 13 13 0.113 0.124 list_rev_and 15 15 0.097 0.034 list_rev_and 15 5 0.097 0.0107 0.0107 0.0107 0.035 list_rev_snoc 15 5 0.0097 0.003 0.001 0.007 0.008 list_rev_tailcall 8 8 0.007 0.008 list_snoc 8 8 0.0012 0.012 0.012 0.012 0.012 0.012 0.015 0.002 0.003 0.002 0.003 0.00	list inc	4	4	•	0.184	0.018	•
list_map 8 8 0.039 0.049 list_nth list_nth 13 13 0.113 0.124 list_pairwise_swap 7 7 4.229 0.634 list_rev_append 5 5 0.097 0.107 list_rev_fold 5 5 0.027 0.035 list_rev_fold 5 5 0.009 0.010 list_rev_tailcall 8 8 0.007 0.008 list_srow_ass 8 8 0.007 0.008 list_srow_ass 0.012 0.012 list_srow_ass 0.012 0.012 list_srow_ass 0.012 0.012 0.015 0.029 0.02 0.02 <t< td=""><td>list last</td><td>6</td><td>6</td><td>•</td><td>0.007</td><td>0.007</td><td>•</td></t<>	list last	6	6	•	0.007	0.007	•
list_nth 13 13 0.113 0.124 Islist_pairwise_swap 7 7 4.229 0.634 Islist_rev_append 5 5 0.097 0.107 Islist_rev_append 5 5 0.097 0.107 Islist_rev_fold 5 5 0.027 0.035 Islist_rev_snoc 5 5 0.009 0.010 Islist_srev_tailcall 8 8 0.007 0.008 Islist_snoc 8 8 0.012 0.012 0.012 0.015 0.021 0.022 0.022 0.022 0.022 0.023 0	list length	3	3	•	0.003	0.002	•
list_pairwise_swap 7 7 4.229 0.634 list_rev_append 5 5 0.097 0.107 list_rev_lond 5 5 0.027 0.035 list_rev_snoc 5 5 0.009 0.010 list_snoc 8 8 0.007 0.008 list_snoc 8 8 0.012 0.012 list_sort_sorted_insert 12 12 10.964 2.902 list_sorted_insert 12 12 10.964 2.902 list_stutter 3 3 0.004 0.003 list_stutter 3 3 0.023 0.029 list_take 12 12 12 0.075 0.065 list_tl 3 3 0.003 0.002 0.005 nat_add 9 9 0.007 0.006 0.006 nat_seven 4 4 0.004 0.003 0.002 nat_max 9 9	list map	8	8	•	0.039	0.049	•
list_pairwise_swap 7 7 4.229 0.634 list_rev_append 5 5 0.097 0.107 list_rev_append 5 5 0.097 0.107 list_rev_tailcall 5 5 0.027 0.035 list_rev_tailcall 8 8 0.009 0.010 list_sorc 0.008 list_sorc 0.007 0.008 list_sorc 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.0015 0.001 0.0015 0.002 0.002 0.002 0.002 0.002 0.003 0.002 0.003 0.002 0.003 0.002 0.005 0.006 0.006 0.006 0.004 0.003 0.002 0.001 0.006 0.004 0.003 0.002 0.001 0.006	list nth	13	13	•	0.113	0.124	•
list_rev_append 5 5 0.097 0.107 • list_rev_fold 1 0.027 0.035 • 0.027 0.035 • 0.027 0.035 • 0.009 0.010 • 0.009 0.010 • 0.009 0.010 • 0.009 0.010 • 0.009 0.010 • 0.009 0.010 • 0.004 0.008 • 0.004 0.008 • 0.012 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.012 • 0.015 • 0.002 • 0.003 • 0.029 • 0.023 • 0.029 • 0.029 • 0.003 • 0.002 • 0.006 • 0.004 • 0.003 • 0.002 • 0.001 • 0.004 • 0.003 • 0.0		7	7	•	4.229	0.634	•
list_rev_fold 5 5 0.027 0.035 • list_rev_snoc 5 5 0.009 0.010 • list_rev_tailcall 8 8 0.007 0.008 • list_sorced_insert 7 7 0.015 0.012 • list_sorted_insert 12 12 10.964 2.902 • list_stutter 3 3 0.004 0.003 • list_sum 3 3 0.023 0.029 • list_take 12 12 0.075 0.065 • list_tl 3 3 0.003 0.002 • nat_add 9 9 0.007 0.006 • nat_max 9 9 0.039 0.041 • nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • <td></td> <td>5</td> <td>5</td> <td>•</td> <td>0.097</td> <td>0.107</td> <td>•</td>		5	5	•	0.097	0.107	•
list_rev_snoc 5 5 0.009 0.010 Ist_rev_tailcall 8 8 0.007 0.008 Ist_snoc 8 8 0.012 0.012 Ist_snoc Ist_snoc 8 8 0.012 0.012 Ist_snoc Ist_snoc 8 8 0.012 0.015 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.02 0.001 0.004 0.003 0.002 0.001 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.003		5	5	•	0.027	0.035	•
list_rev_tailcall 8 8 0.007 0.008 • list_snoc list_snoc 8 8 0.012 0.012 • list_snoc list_sort_sorted_insert 12 12 10.964 2.902 • list_stutter 3 3 0.004 0.003 • list_stutter 3 3 0.004 0.003 • list_stutter 12 12 12 0.075 0.065 • list_take 12 12 0.075 0.065 • list_take 12 12 0.075 0.065 • list_take 0.003 0.002 • list_take 12 12 0.075 0.065 • list_take 0.003 0.002 • list_take 0.003 0.002 • list_take 0.003 0.002 • list_take 0.003 0.002 • list_take 0.006 • list_take 0.006 • list_take 0.006 • list_take 0.007 0.006 • list_take 0.007 0.006 • list_take 0.007 0.006 • list_take 0.007 0.006 • list_take		5	5	•	0.009	0.010	•
list_snoc 8 8 0.012 0.012 0.012 0.012 0.015 0.004 0.003 0.002 0.002 0.023 0.029 0.029 0.023 0.029 0.065 0.065 0.065 0.065 0.065 0.005 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.003 0.002 0.001 0.003 0.002 0.001 0.001 0.003 0.002 0.001 0.0				•			•
list_sort_sorted_insert 7 7 0.015 0.015 list_sorted_insert 12 12 10.964 2.902 list_stutter 3 3 0.004 0.003 list_stutter 3 3 0.023 0.029 list_stutter 12 12 12 0.075 0.065 list_take 12 12 0.075 0.065 list_take list_take 12 12 0.007 0.065 list_take list_take 12 12 0.007 0.066 list_take 0.003 0.002 0.002 0.002 0.002 0.002 0.006 0.002 0.001 nat_add 9 9 0.007 0.006 0.003 0.002 0.001 nat_max 9 9 0.039 0.041 0.003 nat_max 9 9 0.039 0.041 0.004 0.003 0.002 0.001 tree_count_leave 0.002 0.001 0.001 0.001 0.002 0.001 0.001 0.002 0.001		8		•			•
list_sorted_insert 12 12 10.964 2.902 Islist_stutter 3 3 0.004 0.003 Islist_stum 3 3 0.023 0.029 Islist_stum 12 12 0.075 0.065 Islist_take 12 12 0.075 0.065 Islist_take 12 12 0.007 0.006 0.002 0.002 0.002 0.002 0.002 0.002 0.003 0.002 0.003 0.002 0.001 0.003 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.003 0.004 0.001		7	7	•	0.015	0.015	•
list_stutter 3 3 0.004 0.003 • list_sum 3 3 0.023 0.029 • list_take 12 12 0.075 0.065 • list_tt 3 3 0.003 0.002 • nat_add 9 9 0.007 0.006 • nat_iseven 4 4 0.004 0.003 • nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 0.066 0.074 • tree_collect_leaves 7 7 3.009 2.660 • tree_count_leaves 7 7 3.009 2.660 • tree_inorder 5 5 0.114 0.123 • tree_inorder 5 5 0.014 0.055 0.061 • tree_postorder 20		12	12	•	10.964	2.902	•
list_sum 3 3 0.023 0.029 list_take 12 12 0.075 0.065 list_tl 3 3 0.003 0.002 nat_add 9 9 0.007 0.006 nat_iseven 4 4 0.004 0.003 nat_max 9 9 0.039 0.041 nat_pred 3 3 0.002 0.001 tree_binsert 20 timeout • tree_collect_leaves 6 6 0.066 0.074 tree_count_leaves 7 7 3.009 2.660 tree_count_nodes 6 6 0.323 0.351 tree_inorder 5 5 0.114 0.123 tree_map 7 7 0.055 0.061 tree_nodes_at_level 11 • timeout • tree_postorder 20 • timeout •		3	3	•	0.004		•
list_take 12 12 0.075 0.065 • list_tl 3 3 0.003 0.002 • nat_add 9 9 0.007 0.006 • nat_iseven 4 4 0.004 0.003 • nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 0.066 0.074 • tree_count_leaves 7 7 3.009 2.660 • tree_count_nodes 6 6 0.323 0.351 • tree_inorder 5 5 0.114 0.123 • tree_map 7 7 0.055 0.061 • tree_postorder 20 • timeout • •	_	3	3	•	0.023		•
list_tl 3 3 0.003 0.002 nat_add 9 9 0.007 0.006 nat_iseven 4 4 0.004 0.003 nat_max 9 9 0.039 0.041 nat_pred 3 3 0.002 0.001 tree_binsert 20 • timeout • tree_collect_leaves 6 6 0.066 0.074 tree_count_leaves 7 7 3.009 2.660 • tree_count_nodes 6 6 0.323 0.351 • tree_inorder 5 5 0.114 0.123 • tree_map 7 7 0.055 0.061 • tree_postorder 20 • timeout • timeout •		12	12	•			•
nat_iseven 4 4 0.004 0.003 • nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 0.066 0.074 • tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_postorder 20 • timeout • •				•			•
nat_iseven 4 4 0.004 0.003 • nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 0.066 0.074 • tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_postorder 20 • timeout • •	nat add	9	9	•	0.007	0.006	•
nat_max 9 9 0.039 0.041 • nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 0.066 0.074 • tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • • • • • tree_postorder 20 • • • • • •		4	4	•	0.004	0.003	•
nat_pred 3 3 0.002 0.001 • tree_binsert 20 • timeout • • tree_collect_leaves 6 6 • 0.066 0.074 • tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • • • • tree_postorder 20 • • • •	_	9	9	•			•
tree_collect_leaves 6 6 0.066 0.074 • tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • • timeout • • tree_postorder 20 • • timeout • •	_	3	3	•	0.002	0.001	•
tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • timeout • • tree_postorder 20 • timeout • •	tree_binsert	20	•	•	timeout	•	•
tree_count_leaves 7 7 • 3.009 2.660 • tree_count_nodes 6 6 • 0.323 0.351 • tree_inorder 5 5 • 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • timeout • • • tree_postorder 20 • timeout • • •	tree_collect_leaves	6	6	•	0.066	0.074	•
tree_count_nodes 6 6 0.323 0.351 • tree_inorder 5 5 0.114 0.123 • tree_map 7 7 0.055 0.061 • tree_nodes_at_level 11 • timeout • • tree_postorder 20 • timeout • •		7	7	•	3.009	2.660	•
tree_inorder 5 5 0.114 0.123 • tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • timeout • • tree_postorder 20 • timeout • •		6	6	•	0.323	0.351	•
tree_map 7 7 • 0.055 0.061 • tree_nodes_at_level 11 • timeout • • tree_postorder 20 • timeout • •		5	5	•	0.114	0.123	•
tree_nodes_at_level 11 • timeout • tree_postorder 20 • timeout •	_	7	7	•	0.055	0.061	•
tree_postorder 20 • timeout •		11	•	•	timeout	•	•
		20	•	•	timeout	•	•
		5	5	•	0.145	0.153	•

Table 1. Experiment 1. Differences (in blue) between results from SMYTH at submission and SMYTH now:

list_compress, tree_binsert, tree_nodes_at_level: Not automatically run by our scripts because they timeout.

Expert: list_even_parity: SMYTH now finds a smaller solution that is consistent with the MYTH examples. We are not sure why MYTH did not also find that solution (which would have meant that they would have added more examples until the desired solution was synthesized). We will mark this as overspec.

Expert: list_filter: We noticed that, given the MYTH expert examples, the term synthesized by MYTH (Osera's thesis [35], p.171) is not actually a correct implementation of the list filter function. We added trace-complete examples until SMYTH synthesizes a correct solution; there are now 9 examples instead of the 8 reported in [35]. This approach is a proxy for what the MYTH developers might have done to obtain their expert set of examples. (We did not contact the MYTH developers nor try MYTH given these new examples.)

Time: Small variations in running time are expected. Two benchmarks (list_pairwise_swap, list_sorted_insert) are now significantly faster due to the improvements in search order.

		eriment 2a		Experiment 2b			
Name	Expert Exper		Expert	Random	Random Random		
	Submission:	Revised Artifact:		Submission:	Revised Artifact:		
	Fig. 10	Ours	Yours	Fig. 10	Ours	Yours	
bool_band	3 (75%)	3 (75%)	•	(4,4)	(4,4)	•	
bool_bor	3 (75%)	3 (75%)	•	(4,4)	(4,4)	•	
bool_impl	3 (75%)	3 (75%)	•	(3,4)	(4,4)	•	
bool_neg	2 (100%)	2 (100%)	•	(2,2)	(2,2)	•	
bool_xor	3 (75%)	4 (100%)	•	(4,4)	(4,4)	•	
list_append	4 (67%)	4 (67%)	•	(3,5)	(3,4)	•	
list_compress	_	•	•	_	•	•	
list_concat	3 (50%)	3 (50%)	•	(2,3)	(2,4)	•	
list_drop	5 (45%)	5 (45%)	•	(6,9)	(6,9)	•	
list_even_parity	5 (71%)	•	•	failed	(-,-)	•	
list_filter	4 (50%)	5 (63%)	•	_	•	•	
list_fold	3 (33%)	3 (33%)	•	_	•	•	
list_hd	2 (67%)	2 (67%)	•	(2,3)	(2,3)	•	
list_inc	2 (50%)	2 (50%)	•	(2,2)	(2,2)	•	
list_last	4 (67%)	4 (67%)	•	(5,12)	(5,9)	•	
list_length	3 (100%)	3 (100%)	•	(2,3)	(3,4)	•	
list_map	4 (50%)	4 (50%)	•	_	•	•	
list_nth	5 (38%)	5 (38%)	•	(8,15)	(7,14)	•	
list_pairwise_swap	5 (71%)	5 (71%)	•	timeout	•	•	
list_rev_append	3 (60%)	3 (60%)	•	(5,9)	(5,8)	•	
list_rev_fold	2 (40%)	2 (40%)	•	(2,3)	(2,4)	•	
list_rev_snoc	3 (60%)	3 (60%)	•	(3,7)	(3,6)	•	
list_rev_tailcall	3 (38%)	3 (38%)	•	(3,6)	(3,4)	•	
list_snoc	4 (50%)	3 (38%)	•	(3,4)	(3,4)	•	
list_sort_sorted_insert	3 (43%)	3 (43%)	•	(3,6)	(3,6)	•	
list sorted insert	7 (58%)	7 (58%)	•	timeout	•	•	
list stutter	2 (67%)	2 (67%)	•	(3,3)	(3,3)	•	
list sum	2 (67%)	2 (67%)	•	(2,3)	(2,2)	•	
list take	6 (50%)	5 (42%)	•	(7,10)	(6,9)	•	
list_tl	2 (67%)	2 (67%)	•	(2,3)	(2,3)	•	
nat add	4 (44%)	4 (44%)	•	(4,6)	(5,6)	•	
nat_iseven	3 (75%)	3 (75%)	•	(3,4)	(4,4)	•	
nat max	9 (100%)	9 (100%)	•	(9,11)	(8,12)	•	
nat_pred	2 (67%)	2 (67%)	•	(2,3)	(2,3)	•	
tree_binsert	_	•	•	_	•	•	
tree collect leaves	3 (50%)	3 (50%)	•	$(3,4)^3$	(3,4)	•	
tree count leaves	3 (43%)	3 (43%)	•	timeout	•	•	
tree count nodes	3 (50%)	3 (50%)	•	$(4,\downarrow)^{10}$	(4,-)	•	
tree inorder	4 (80%)	4 (80%)	•	(3,4)	(3,4)	•	
tree map	4 (57%)	4 (57%)	•		•	•	
tree_nodes_at_level	_	• (=)	•	_	•	•	
tree_postorder	_	•	•	_	•	•	
tree preorder	3 (60%)	3 (60%)	•	$(3,3)^3$	(3,4)	•	
Lice_preorder	3 (00/0)	3 (0070)	-	(3,3)	(3,1)	•	

Table 2. Experiment 2. Differences (in blue) between results from SMYTH at submission and SMYTH now: list_compress, tree_binsert, tree_nodes_at_level: Not run because they failed in Experiment 1.

Expert: bool_xor: With the algorithmic changes, SMYTH now requires (all) 4 examples. (Small changes to search order and search parameters can change the results of synthesis tools.)

Expert: list_filter: The Experiment 1 expert examples were extended by one; the Experiment 2 expert examples were, too.

Expert: list_snoc, **list_take**: When looking through our tasks again, we noticed an opportunity to try removing another example from these benchmarks; Sмутн produces correct solutions given the fewer examples.

Random: Small variations in k50 and k90 are expected because the examples are generated randomly. There are some blue dots and dashes because our scripts for benchmarking and generating the table differences do not automatically display failed, timeout, superscripts 3 and 10, or the ↓ arrow presented in Figure 10.

		periment 3a		Experiment 3b			
Name	Expert	Expert	Expert	Random	Random	Random	
	Submission:	Revised Artifact:		Submission:	Revised Artifact:		
	Fig. 10	Ours	Yours	Fig. 10	Ours	Yours	
bool_band	_	•	•	_	•	•	
bool_bor	_	•	•	_	•	•	
bool_impl	_	•	•	_	•	•	
bool_neg	_	•	•	_	•	•	
bool_xor	_	•	•	_	•	•	
list_append	1+1 (33%)	1+1 (33%)	•	(1+3,1+5)	(1+3,1+4)	•	
list_compress	_	•	•	_	•	•	
list_concat	1+2 (50%)	•	•	(1+2,1+3)	(1+3,1+5)	•	
list_drop	1+2 (27%)	1+2 (27%)	•	(1+8,1+15)	(1+7,-)	•	
list even parity	overspec	•	•	failed	(-,-)	•	
list filter	overspec	1+4 (63%)	•	_	•	•	
list fold	1+3 (44%)	1+3 (44%)	•	_	•	•	
list_hd		•	•	_	•	•	
list inc	_	•	•	_	•	•	
list last	1+2 (50%)	1+2 (50%)	•	(1+5,1+8)	(1+5,1+10)	•	
list_length	1+1 (67%)	1+1 (67%)	•	(1+2,1+3)	(1+2,1+2)	•	
list_map	1+2 (38%)	1+2 (38%)	•		•	•	
list nth	1+2 (23%)	1+2 (23%)	•	(1+8,1+16)	(1+7,1+15)	•	
list pairwise swap	overspec	• ′	•	timeout	•	•	
list_rev_append	1+2 (60%)	1+2 (60%)	•	(1+3,1+14)	(1+3,1+4)	•	
list_rev_fold		• 1	•		•	•	
list rev snoc	1+1 (40%)	1+1 (40%)	•	(1+3,1+5)	(1+2,1+4)	•	
list rev tailcall	1+1 (25%)	1+1 (25%)	•	(1+3,1+6)	(1+3,1+5)	•	
list_snoc	1+2 (38%)	1+1 (25%)	•	(1+2,1+4)	(1+3,1+4)	•	
list_sort_sorted_insert	1+1 (29%)	1+1 (29%)	•	(1+3,1+6)	(1+2,1+4)	•	
list sorted insert	overspec	1+7 (67%)	•	timeout	•	•	
list_stutter	1+1 (67%)	1+1 (67%)	•	(1+3,1+4)	(1+2,1+3)	•	
list_sum		• ′	•		•	•	
list take	1+3 (33%)	1+3 (33%)	•	(1+8,1+15)	(1+7,1+16)	•	
list_tl		•	•		•	•	
nat add	1+1 (22%)	1+1 (22%)	•	(1+4,1+6)	(1+3,1+4)	•	
nat iseven	1+2 (75%)	1+2 (75%)	•	(1+3,1+4)	(1+3,1+4)	•	
nat max	1+4 (56%)	1+4 (56%)	•	(1+8,1+12)	(1+8,1+12)	•	
nat_pred	_	•	•	_	•	•	
tree binsert	_	•	•	_	•	•	
tree collect leaves	1+2 (50%)	1+2 (50%)	•	(1+3,1+3)	(1+3,1+3)	•	
tree_count_leaves	1+1 (29%)	1+1 (29%)	•	timeout	•	•	
tree_count_nodes	1+2 (50%)	1+2 (50%)	•	$(1+4,1+5)^{10}$	(1+3,1+5)	•	
tree inorder	1+2 (60%)	1+2 (60%)	•	(1+3,1+3)	(1+3,1+4)	•	
tree map	1+3 (57%)	1+3 (57%)	•	-	•	•	
tree_nodes_at_level	- (37.0)	• (3770)	•	_	•	•	
tree_postorder	_	•	•	_	•	•	
tree preorder	1+2 (60%)	1+2 (60%)	•	(1+3,1+3)	(1+3,1+3)	•	
nec_preorder	112 (00%)	1 12 (00/0)	-	(113,113)	(1 : 3,1 : 3)	•	

Table 3. Experiment 3. Differences (in blue) between results from Sмүтн at submission and Sмүтн now:

Benchmarks with dashes (—) in Figure 10 are not included in this experiment either because they are not recursive or because they failed in Experiment 1.

Expert: list_filter, list_sorted_insert: These tasks succeed because of the newly added responsive staging parameters.

Expert: list_snoc: When looking through our tasks again, we noticed an opportunity to try removing another example from these benchmarks; SMYTH produces a correct solution given the fewer examples.

Expert: list_even_parity: As in Experiment 1, Smyth now finds a smaller solution, so we will mark this as overspec. (The desired solution is ranked second by Smyth.)

Expert: list_concat: The first solution now returned by SMYTH is a "rev_concat" function (a generalization of snoc) rather than the desired concat function (a generalization of cons). The second solution returned by SMYTH is the desired concat. Both of these solutions have the same AST size, and so SMYTH arbitrarily chooses. The MYTH expert examples do not distinguish between these two functions. Nevertheless, we will mark this as incorrect.

Random: Small variations in k50 and k90 are expected because the examples are generated randomly. There are some blue dots and a "(-,-)" because the provided scripts do not automatically display all the labels in Figure 10.

	LEON 4		Synquid 4		
Name	1	2a	1	2a	
Nume	1	2	•	2	
bool_band	/	✓	✓	✓	
bool_bor	/	✓	✓	✓	
bool_impl	/	✓	✓	✓	
bool_neg	/	$\checkmark \Rightarrow -$	✓	$\checkmark \Rightarrow -$	
bool_xor	1	$X^1 \Rightarrow -$	1	$\checkmark \Rightarrow -$	
list_append	/	\mathbf{X}^1	? ⇒ ✓	$\mathbf{X}^4 \Longrightarrow \mathbf{X}^1$	
list_compress	χ^2	_	$? \Rightarrow X^2$. —	
list_concat	/	\mathcal{X}^1	$? \Rightarrow X^1$	$X^4 \Rightarrow X^1$	
list_drop	/	√	? ⇒ ✓	$X^4 \Rightarrow X^2$	
list_even_parity	√	$X^1 \Rightarrow -$	$? \Rightarrow X^2$	$X^4 \Longrightarrow -$	
list_filter	X ³	χ^3	X ³	χ^3	
list_fold	X 3	χ^3	χ^3	X ³	
list_hd	/	✓	? ⇒ ✓	$\chi^1 \Rightarrow \checkmark$	
list_inc	/	✓	$? \Rightarrow X^2$	$? \Rightarrow X^1$	
list_last	✓	✓	? ⇒ ✓	$X^4 \Rightarrow X^2$	
list_length	√	_	? ⇒ ✓	_	
list_map	X ³	χ^3	X ³	X ³	
list_nth	1	√	? ⇒ ✓	$X^4 \Rightarrow X^2$	
list_pairwise_swap	1	✓ <u> </u>	$? \Rightarrow X^2$	$X^4 \Rightarrow X^2$	
list_rev_append	1	/	$? \Rightarrow X^2$	$X^4 \Rightarrow X^2$	
list_rev_fold	1	√	$? \Rightarrow X^2$	$? \Rightarrow X^2$ $X^4 \Rightarrow X^2$	
list_rev_snoc	✓ X ¹	√	$? \Rightarrow X^1$	$X^4 \Rightarrow X^2$ $X^4 \Rightarrow X^1$	
list_rev_tailcall		√	? ⇒ ✓	$X^4 \Rightarrow X^2$ $X^4 \Rightarrow X^2$	
list_snoc	1	/	$? \Rightarrow \checkmark$ $? \Rightarrow X^2$	$X^4 \Rightarrow X^2$ $X^4 \Rightarrow X^1$	
list_sort_sorted_insert	X ²	✓ X ²	$? \Rightarrow X^2$	$X^4 \Rightarrow X^2$	
list_sorted_insert	<i></i>	<u>^</u>	? ⇒ ✓	$X^4 \Rightarrow X^1$	
list_stutter	1	X ¹	$? \Rightarrow X^2$	$? \Rightarrow X^2$	
list_sum	1	<u>^</u>	? ⇒ ✓	$f \Rightarrow \land$ $X^4 \Rightarrow X^2$	
list_take list_tl	1	/	$Y^1 \Rightarrow \checkmark$	$ \stackrel{\wedge}{X} \Rightarrow \stackrel{\wedge}{X} $	
list_ti	•	✓	^ -> V	^ <i>→</i> v	
nat_add	1	✓	? ⇒ ✓	$X^4 \Rightarrow X^1$	
nat_iseven	/	/	? ⇒ ✓	$X^4 \Rightarrow X^2$	
nat_max	X^1	_	? ⇒ ✓	_	
nat_pred	1	/	$X^1 \Rightarrow \checkmark$	$X^1 \Rightarrow \checkmark$	
tree_binsert	X ²	_	$? \Rightarrow X^2$	_	
tree_collect_leaves	/	✓	$? \Rightarrow X^1$	$X^4 \Longrightarrow X^1$	
tree_count_leaves	/	✓	$? \Rightarrow X^2$	$\mathbf{X}^4 \Longrightarrow \mathbf{X}^2$	
tree_count_nodes	/	✓	$? \Rightarrow X^1$	$X^4 \Rightarrow X^2$	
tree_inorder	/	✓	$? \Rightarrow X^1$	$X^4 \Rightarrow X^2$	
tree_map	χ^3	χ^3	χ^3	χ^3	
tree_nodes_at_level	χ^2	_	$? \Rightarrow X^2$	_	
tree_postorder	/	_	$? \Rightarrow X^2$		
tree_preorder	✓	/	$? \Rightarrow X^1$	$X^4 \Rightarrow X^1$	

Table 4. Experiment 4. Differences (in blue) between results from submission and now:

2a: **bool_neg**: The SMYTH expert examples include all of the MYTH expert examples, so this experiment should have been marked "—".

2a: bool_xor: The Sмүтн expert examples changed and now include all of the Мүтн expert examples, so this task is no longer applicable ("—").

2a: list_even_parity: This is now considered a failure in Experiment 1, so this task is no longer applicable ("-").

Synquid: In the submission, ? and X^4 were used to mark tasks that could not be run due to a Synquid implementation issue that has since been fixed. As expected, some of the ? tasks now succeed (\checkmark), some produce an incorrect solution (X^1), and some terminate with zero solutions or do not find a solution within a timeout (X^2). Also as expected, all of the X^4 tasks—where the examples are not trace-complete, i.e., do not form an inductive- specification—fail (X^1 or X^2).