Type Feedback vs. Concrete Type Inference

Ole Agesen
Stanford University

Urs Hölzle
UC, Santa Barbara

OOPSLA'95

Type Feedback (TF) vs. Type Interence (TI)

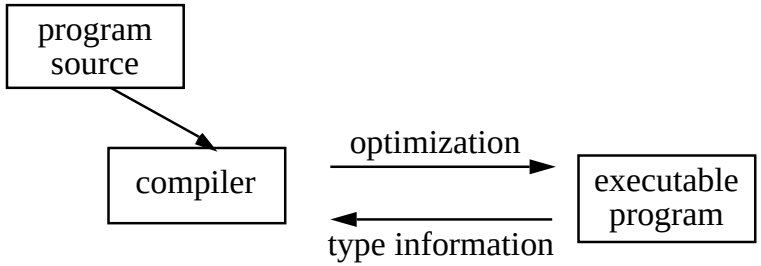


Figure 1. Overview of Type Feedback

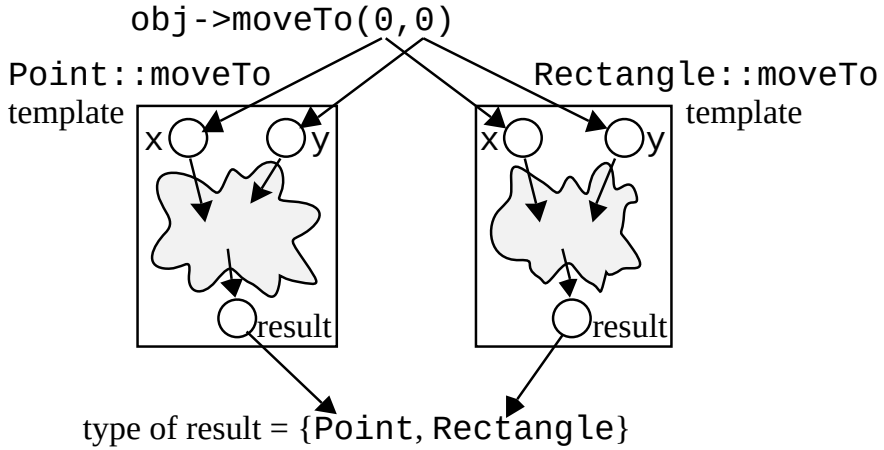
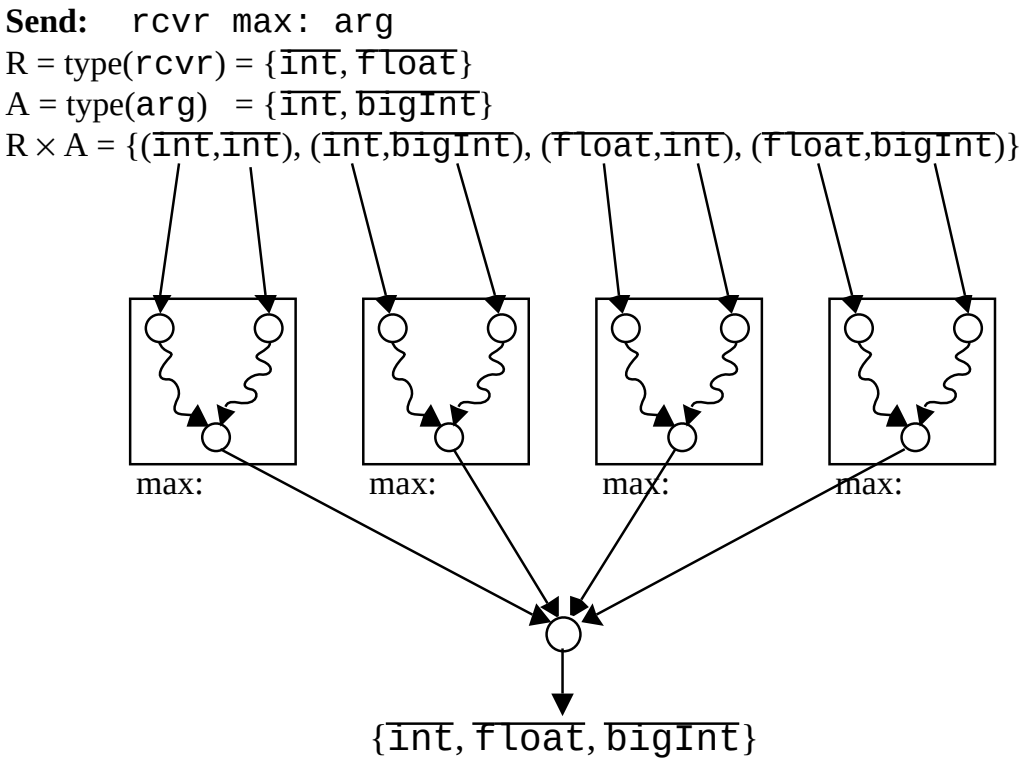


Figure 3. Send invoking several methods

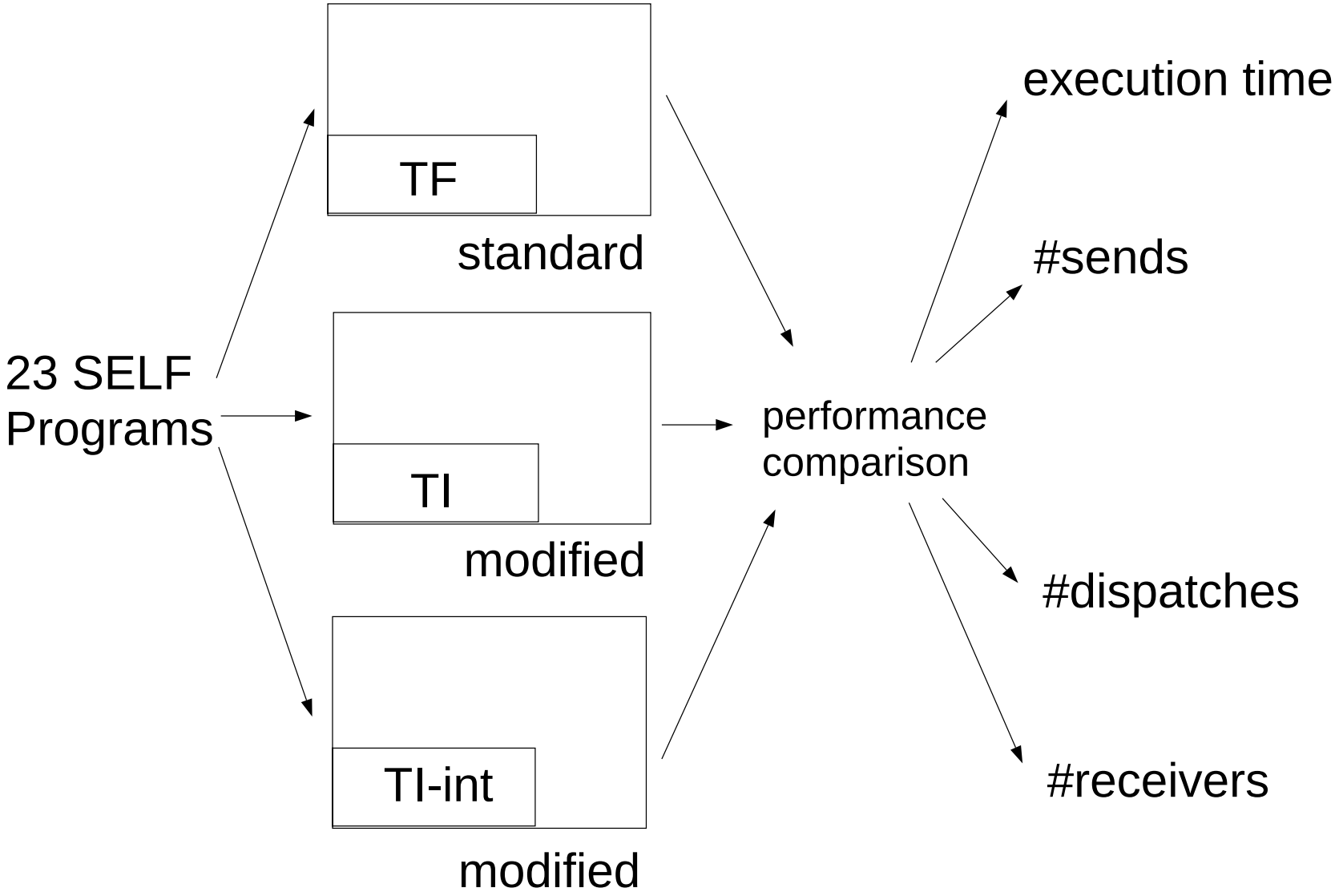
Receiver Classes at Call Sites

Cartesian Product Algorithm

Cartesian Product Algorithm (CPA)



Evaluation Methodology



23 SELF Benchmark Programs

	Name	Appl. size[a]	Total size[b]	calls	static calls	dispatches	avg. arity	time
"Tiny"	AtAllPut	3	1,059	1,600,054	1,000,034	1,200,030	2.17	16.24
	Recur	3	1,047	3,932,167	2,949,082	2,949,110	2.44	31.83
	SumTo	3	1,049	15,003,637	8,002,122	17,002,921	2.29	76.61
	Tak	10	1,055	572,495	333,954	572,488	2.31	8.16
"Small"	Bubble	20	1,089	4,949,143	3,047,322	3,917,196	2.17	44.35
	Detabify	21	1,071	2,323,037	1,295,022	1,673,021	1.92	26.92
	Intmm	30	1,092	2,191,730	1,340,659	1,478,013	2.52	19.43
	Mergesort	50	1,169	13,787,215	8,050,676	13,471,760	2.06	171.86
	Perm	25	1,082	2,482,648	1,677,576	1,617,831	2.33	30.05
	Puzzle	170	1,309	21,704,318	12,577,749	17,269,270	5.12	256.93
	Queens	35	1,094	1,873,879	1,060,564	1,543,504	2.86	17.25
	Quick	35	1,101	2,394,931	1,481,793	1,791,956	2.17	21.85
	Quick2	35	1,180	3,974,032	2,702,648	2,756,506	2.56	33.90
	Sieve	25	1,053	6,086,477	3,723,942	4,975,860	2.15	45.68
	Towers	60	1,116	2,029,491	1,342,206	1,145,272	2.41	18.627
	Tree	25	1,108	1,667,752	620,558	1,891,346	1.97	34.21
"Large"	Deltablue	500	1,358	2,200,062	1,250,828	1,443,229	2.59	39.95
	Diff	300	1,992	9,639,507	5,766,326	6,385,795	4.72	144.10
	SParser	400	1,442	319,256	185,806	173,674	5.15	9.01
	PrimMaker	1,100	2,241	2,331,941	1,291,980	1,516,633	4.67	50.80
	Richards	400	1,284	8,504,293	4,741,143	5,908,623	2.06	116.31
	RSA	300	1,541	212,595,415	124,215,649	157,813,545	5.56	n/a
	CParser	7,000	10,984	4,146,255	2,544,773	2,529,170	4.76	31.79

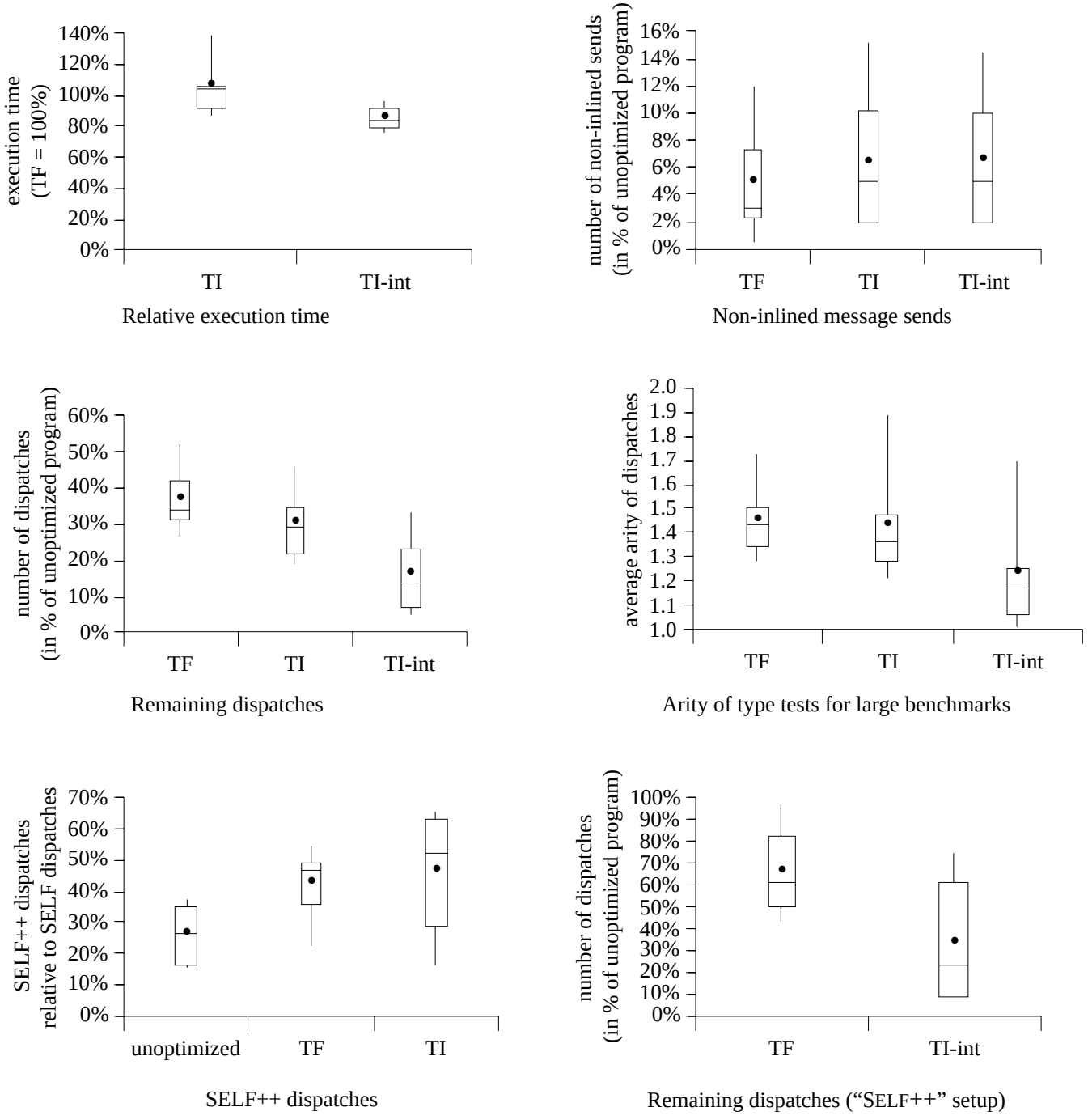


Figure A-1. Box-chart summaries of data for large benchmarks

Box charts show the range of data (vertical lines) as well as the 25% and 75% percentiles (end of the boxes) and the median (horizontal lines). The mean is indicated with a dot (•).

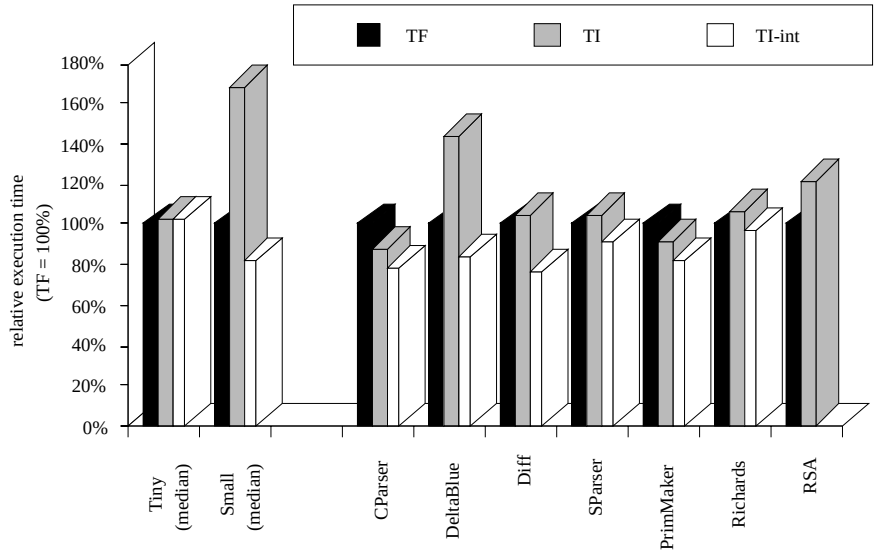


Figure 4. Relative execution time of benchmarks

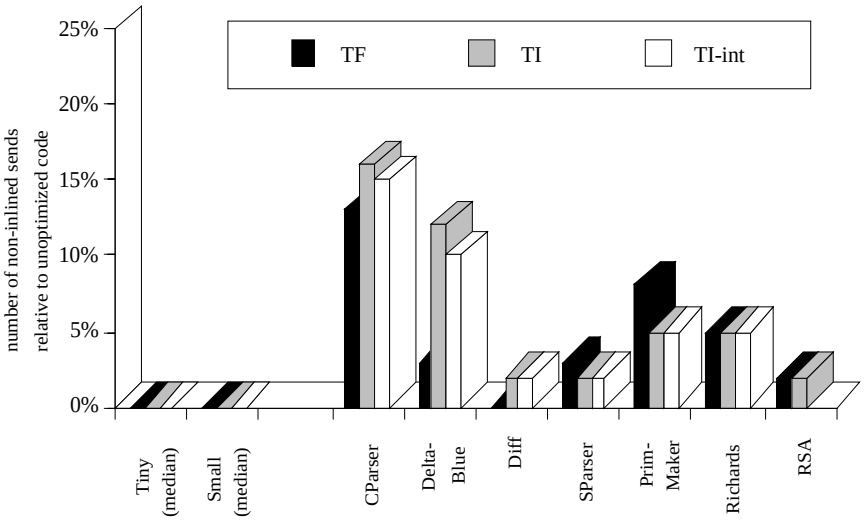


Figure 5. Number of non-inlined message sends relative to unoptimized SELF

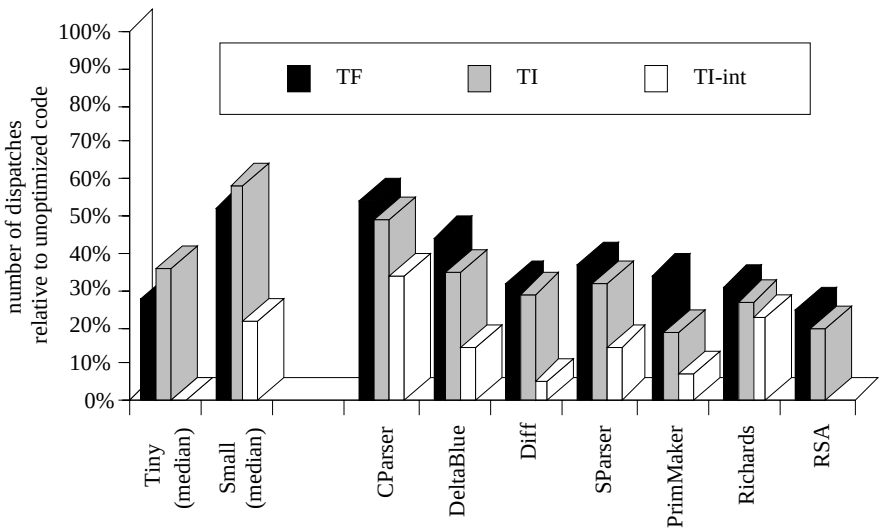


Figure 6. Remaining run-time dispatches

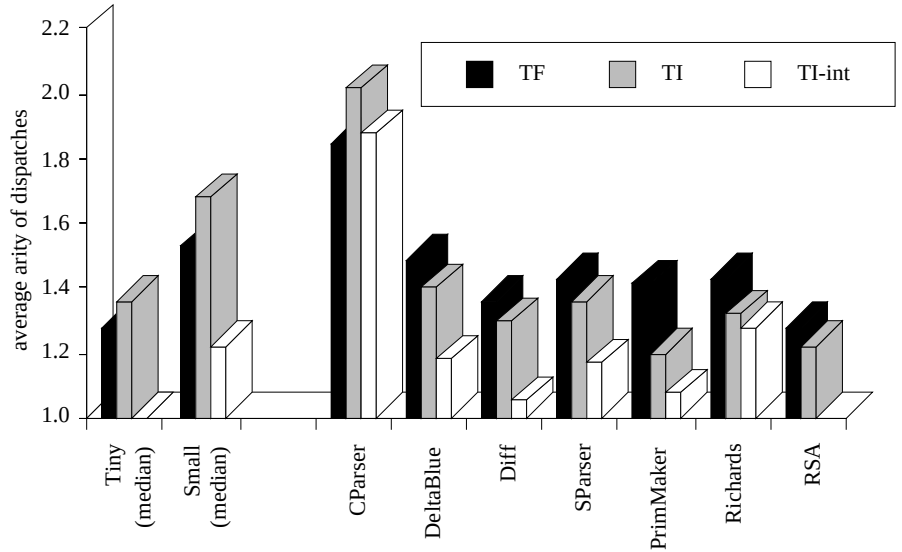


Figure 9. Arity of type tests

The Effect of Message Splitting

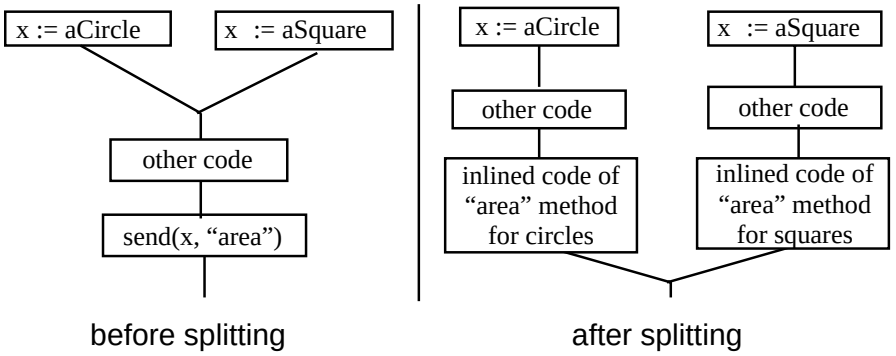


Figure 7. Splitting

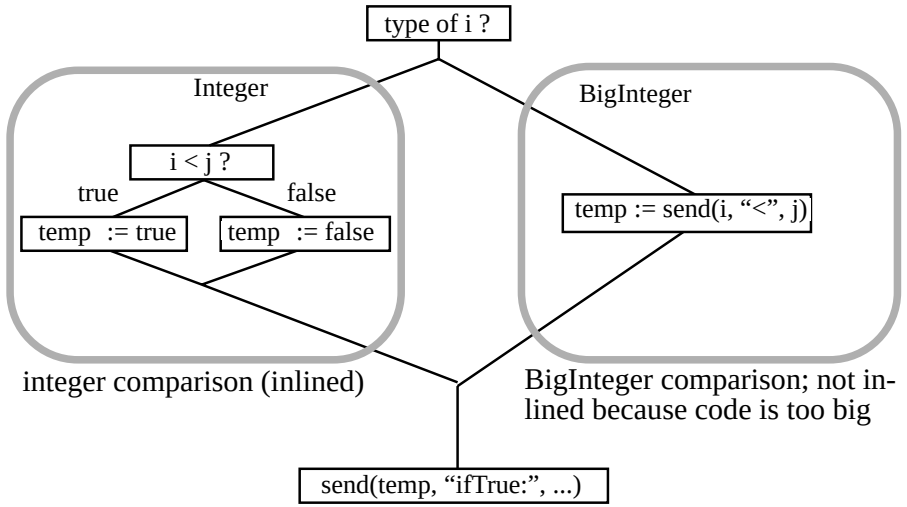


Figure 8. Intermediate code for the expression “i < j” in TI

The Impact of Purely Object Oriented

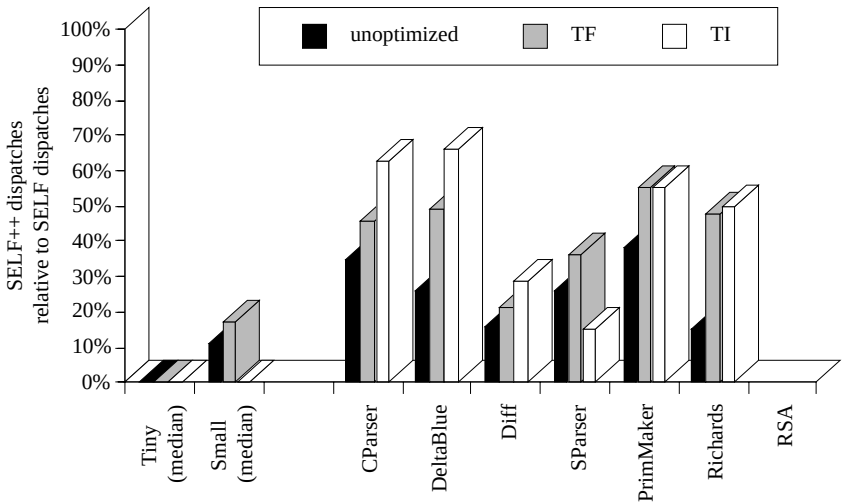


Figure 10. Dispatches in SELF++ relative to SELF

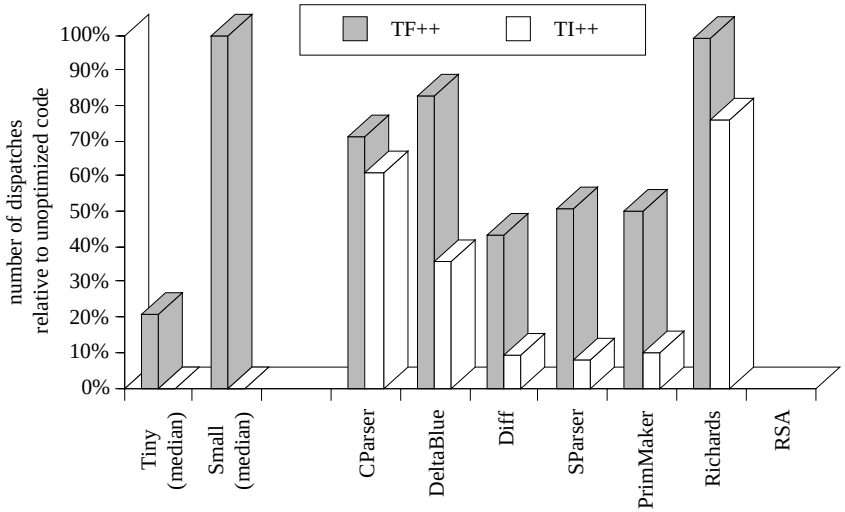


Figure 11. Remaining run-time dispatches in SELF++

Qualitative Comparison

Characteristic	Type Feedback	Type Inference
Responsiveness	incremental, sub-second pauses; suitable for interactive systems	“batch style”, multi-second to multi-minute pauses, not (yet) suitable for interactive systems
Performance	may perform poorly if statically compiled with non-representative profile data	may perform poorly without dynamic information
Application delivery	only need to generate compiled code for cases that actually occur	compiled code can cover all cases
	may need compiler or interpreter at runtime	can generate self-contained executable
	may need source code (or equivalent) at run time	no need to keep around source code
Generality	handles entire language; supports extensible systems	cannot handle entire language; may not scale to very large programs; doesn’t support extensible systems
Implementation	can use same compiler for both development and delivery	probably needs two separate compilers (one for development, one for delivery)

Table 4: Main characteristics of type inference and type feedback

Summary and Conclusion

TI removes more dispatches than TF

Both TF and TI inline equally well

Runtime frequency info is important

Arbitrary-precision is a key performance factor

None of the metrics is a good predictor

Later optimizer de-optimizes the previous effort

Purely OO programs are "easier" to optimize

Combine TI and TF ???