Program	P1	P2	P3	P4	P5	P6	P1	P1	P1
1sort141	13132	15111	11116	111	1	1	1	1	1
1uery161	35136	61111	14112	11151	11111	11111	1	1	1
derive111	16124	11161	13112	16111	11111	11112	11111	1	1
ze1ra111	52161	11111	16141	11115	11111	11111	11111	1	1
serialise111	41161	13151	15111	11131	16135	11146	11114	11111	1
mu111	25115	51111	11111	14116	11135	16111	11161	11111	11111
fast mu111	56112	11112	11141	11111	11111	11122	11156	11111	11111
crypt1111	51111	11111	14163	11111	11141	11111	11111	11111	11111
meta 1sort1111	41113	15131	13111	11151	14135	11155	11111	11111	11111
1ueens11111	42161	51161	61115	1112	11116	11111	11111	11111	11111
nreverse1121	11121	11115	11111	11114	11111	11111	11111	11111	11111
prover1151	41116	61114	14122	16116	13121	16141	11136	11141	11165
1ro1se1111	51113	15111	11135	11111	11125	11146	11151	11111	11111
1oyer1211	51124	15111	16116	15133	16113	11114	11144	11111	11131
sdda1421	31111	61112	61111	15111	11111	13111	16111	11111	11161
nand1111	11141	21131	31161	41155	53113	51116	61156	65111	61161
c1at parser11511	25112	21111	34151	31112	42111	46121	41111	51113	52111

1a1le 11 1ime percentage per predicate

1eing considered is not large 1e could select t1e 311 most important predicates1 11ere aln 111 of t1e total execution time is present1 1urt1er t1an t1at only marginal improvements 1 1e ac1ieved1 or t1e code gro1t1 could even introduce some slo11do1ns due to cac1ing pro1le

1ote t1at t1e results from ta1le 2 accentuate as t1e programs 1ecome larger1 Considering last 3 programs 11ic1 1ave more t1an 41 predicates1 t1e average execution percentage on 21 t1e predicates is 131411 Considering t1e 1rst 1 programs 11ic1 1ave less t1an 11 predicates1 percentage is of 641611

We 1ave selected to multiply specialize 211 of t1e predicates of eac1 1enc1mar11 11is me 1 predicate for qsort, the smallest program, and 32 predicates for chatparser, the largest one. $n terms of execution time, this means an interval from 3$.4$ for query to $$.8$ for nreverse.

$e compare the results of our system (ms-wam) with the wamcc basic system, in terms of execution speed, code si

Program	
1sort	3615 1313 1313 1511 1511 1511 1111 1111 111 111
1uery	3514 3514 6111 6111 1411 1115 1115 111 111 111
derive	1612 1612 1111 1311 1611 1611 1111 1111 1111 111
ze1ra	5216 5216 1111 1614 111 111 111 111 111 111
serialise	4116 1315 1315 1511 1113 1614 1115 1115 1111 111
mu	2611 5111 1111 1412 1114 1114 1611 1116 111 111
fastmu	5611 1111 1114 1112 1111 1111 1112 1116 1111 111
crypt	5111 1111 1416 1111 1115 1112 1116 1111 111 111
meta1sort	4111 1514 1311 1115 1116 111 111 111 111 111
1ueens 1	4216 5116 1111 1112 1112 111 111 111 111 111
nreverse	1112 1111 1111 111 111 111 111 111 111 111
prover	6111 1412 1313 1614 1115 1111 1111 1111 111 111
1ro1se	1511 1111 1115 1111 1111 1111 111 111 111 111
1oyer	1111 1111 1114 1111 111 111 111 111 111 111
sdda	1611 1111 1511 1111 1111 111 111 111 111 111
nand	6512 1311 1112 1413 1111 1115 1111 111 111 111
c1atparser	6214 1112 1612 1211 1511 1114 1114 1111 111 111

Program 111 211 31 1 41 1 5 11 6 11 111 111 111 1111

Ta1le 21 Time percentage per predicates percentage

The most useful part of our analysis is the fact that predicates are analyzed individually a

We can thus independently multiply specialize a subset of the program1

Part of the good results obtained are due to the fact that the most relevant predicates a almost always1 recursive predicates[2]

	ms11am			1amcc		
Program	1xec1 1ime	111ect 1ize	Compile 1ime	1xec1 1ime	111ect 1ize	Compile
1sort	11411	2111	11611	11611	1115	11361
1uery	11341	3412	11111	11311	2115	11611
derive	11151	2111	11151	11321	1113	11621
ze1ra	11131	3111	11121	11141	1111	11611
serialise	11111	3612	11211	11451	1111	11631
mu	11411	3615	11231	11521	1113	11611
fastmu	11111	5611	11111	11111	3111	11111
crypt	11211	5111	11651	11411	2112	11151
meta1sort	11211	1111	21511	11411	4613	11411
1ueens 1	11611	3511	11241	11121	2111	11151
nreverse	11341	1111	11111	11511	1112	11461
prover	11111	5211	11111	11131	2114	11151
1ro1se	11531	6111	21151	11141	3316	11231
1oyer	11251	11511	51111	11441	11111	31321
sdda	11111	15514	51211	11151	1112	31121
nand	11111	26116	11151	11111	14311	51221
c1atparser	11161	41111	111211	11411	21113	111521
Average improvement	11146	1 1111	1 1111			

1a1le 31 Comparing ms11am 1it1 1amcc

111 A1 1elly1 A1 Macdonald1 11 Marriott1 P1 1tuc1ey1 and R1 1ap1 11ectiveness of optimi
for C1P1R11 In Proceedings of 1oint International Conference and 1ymposium on 1ogic P
pages 31