A Semantics-Driven Approach to Improving DataRaceBench's OpenMP Standard Coverage

Chunhua “Leo“ Liao, Pei-Hung Lin, Markus Schordan, and Ian Karlin

IWOMP’18, Sept. 28 Friday
Barcelona, Spain

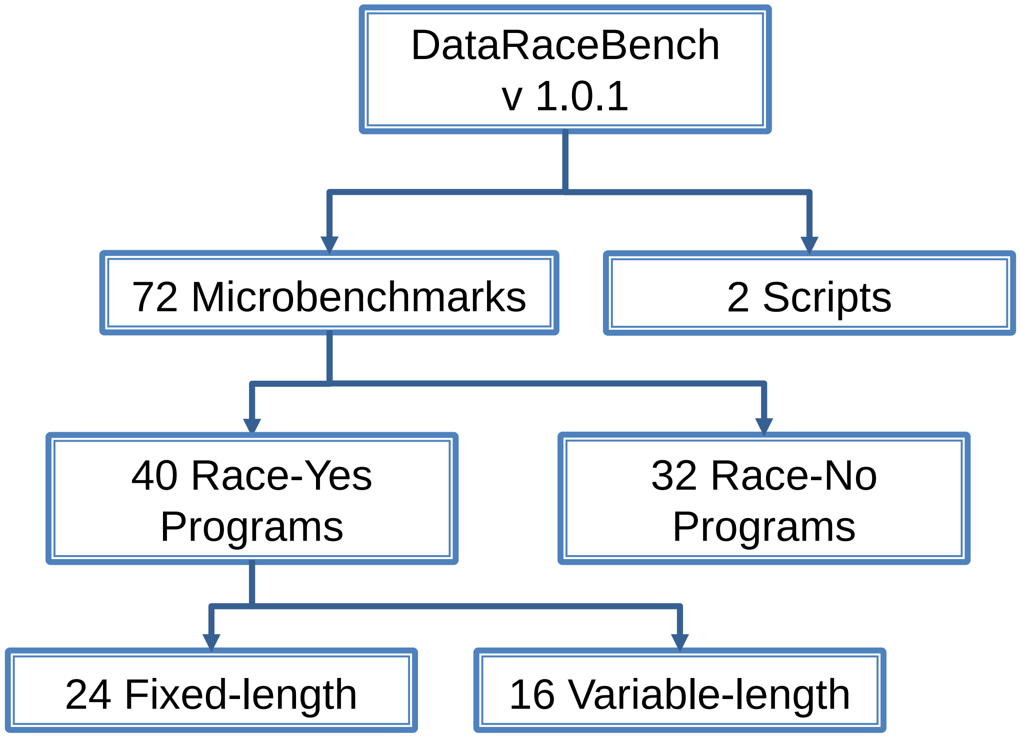
LLNL-PRES-758498

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

DataRaceBench

- DataRaceBench v 1.0.1 released in Aug, 2017
  - Learning from LINPACK for HPC
- Includes typical OpenMP code patterns with or without data races
- Generates quantitative metrics based on true/false X positives/negatives
- Discovers strengths and limitations of data race detection tools



https://github.com/LLNL/dataracebench

Example microbenchmarks

```
1.      …
2.      int i,x;
3.      #pragma omp parallel for
4.       for (i=0;i<100;i++)
5.       {   x=i;  }
6.       printf("x=%d",x);
7.      …
```

lastprivatemissing-orig-yes.c

Race-yes: missing data sharing clauses

One data race pair
x@5 vs. x@5

```
1.      …
2.      int i,x;
3.      #pragma omp parallel for lastprivate (x)
4.      for (i=0;i<100;i++)
5.       {   x=i; }
6.      printf("x=%d",x);
7.      …
```

lastprivate-orig-no.c

Race-no: use of data sharing clauses

A common question to us

- How comprehensive is this benchmark suite?

- Contributions of this paper
  — A novel semantics-driven approach to defining a space to be covered
  — Analyze DataRaceBench's coverage of the space
  — Add new microbenchmarks to improve the coverage
  — Re-evaluate some tools for new insights

Semantic analysis of data races

"A data race can occur when two concurrent threads access a shared variable and when at least one access is a write, and the threads use no explicit (synchronization) mechanism to prevent the accesses from being simultaneous."

- A complete benchmark should cover all semantic dimensions and their combinations: cartesian product   D1 x D2 x D3 ....

Narrow down the scope

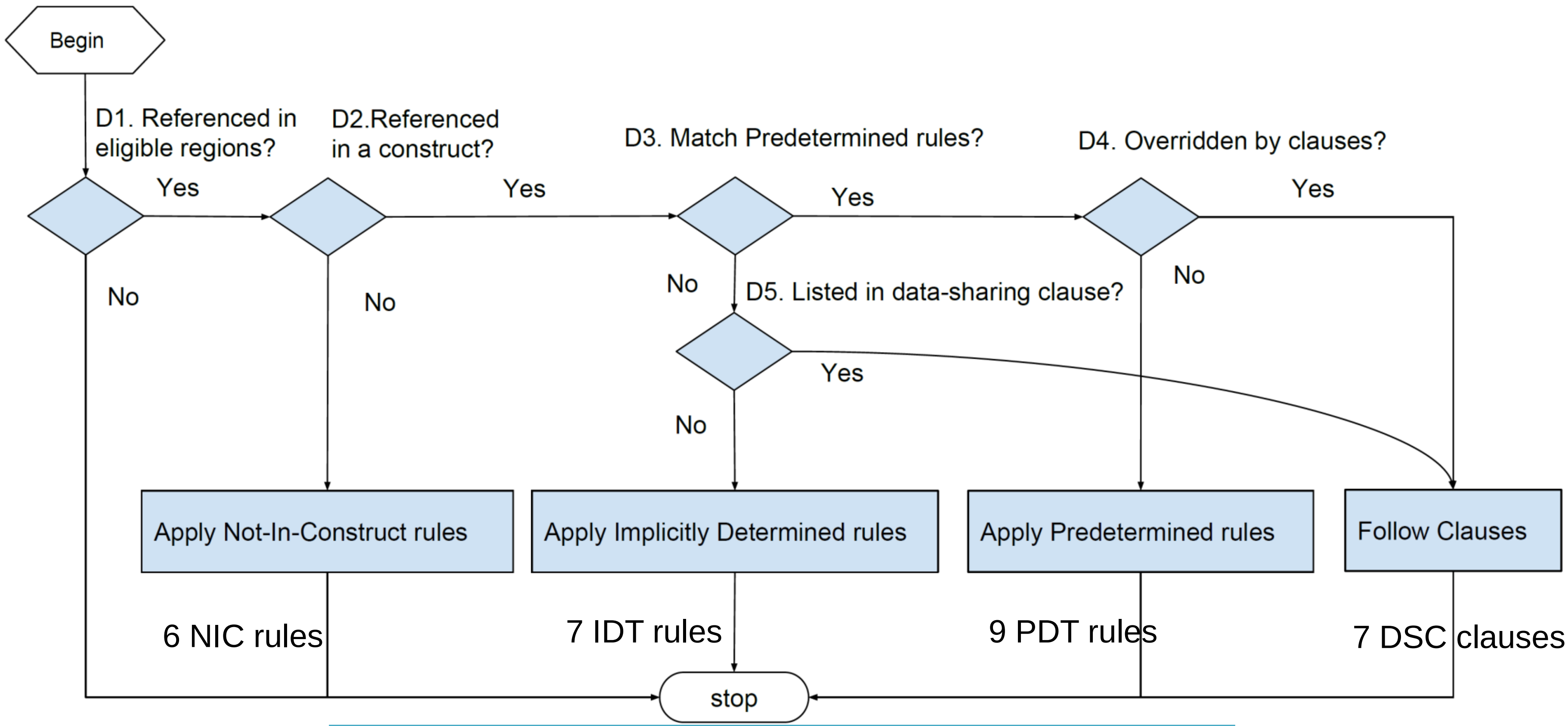
- Parallel (concurrent) semantics: information indicating if a code region will be executed concurrently or not
- Shared semantics: information describing if a variable is visible and accessible by multiple threads or not.
- Synchronization semantics: any information deciding if there is any synchronization mechanism to prevent the shared accesses to a variable from being simultaneous or not.

Semantic labels

Categorize relevant OpenMP 4.5 directives, clauses, or rules and assign labels

Semantics Category	Semantic Labels
Parallel	PD01-PD26: parallel, for, sections, single, master, simd, for simd, task, taskloop, taskloop simd, parallel for, parallel sections, target parallel, target teams, etc. PC01-PC04: if, num_threads, collapse(), num_teams()
Shared	NIC01-NIC06: not referenced in a construct rules IDT01-IDT07: implicitly determined rules PDT01-PDT09: predetermined attribute rules DSC01-DSC07: explicit data-sharing clauses, such as default, shared, private, firstprivate, lastprivate, reduction, and linear.
Synchronization	N01-N10: nowait, critical, barrier, taskwait, taskgroup, atomic, flush, ordered (both clause and directive), depend

Flowchart of data-sharing attribute rules



NIC01-NIC06: Not referenced in a construct rules
IDT01-IDT07: implicitly determined rules
PDT01-PDT09: predetermined attribute rules
DSC01-DSC07: data-sharing clauses

Example NIC rules: not referenced in a construct

The order of the six rules (NIC 1-6) in the specification is rather ad-hoc, we re-organized them, sometimes split them, as follows:

- Declared inside the called routine
  - NIC1: if the variable uses static storage, it is shared
  - NIC6: otherwise, it is private
- File-scope or namespace-scope variable
  - NIC2.1: threadprivate if the variable is in a threadprivate directive
  - NIC2.2: shared otherwise
- Function arguments in C++
  - NIC5.1: same as actual arguments if passed by reference
  - NIC5.2: private if passed by values (not explicitly listed in OpenMP)
- Dynamic storage:
  - NIC3 - objects with dynamic storage duration are shared.
- Static data members
  - NIC4.1: threadprivate if within a threadprivate directive
  - NIC4.2: shared otherwise

Coverage Analysis

- Definition: for each semantic label
  - It is covered if there is a microbenchmark using the corresponding construct, clause or rule
  - For simplicity, a combined construct is treated as covered if the corresponding individual constructs are covered

- Methods
  - Simple keyword search: if some directives or clauses are used in the benchmark suite or not
    - E.g. collapse, depend, taskgroup, etc.
  - Source analysis tool, CoverageAnalyzer, to recognize code patterns
    - Built using the ROSE source-to-source compiler framework
    - PDT8: locally declared variables , if they have static storage duration-> shared
    - NIC rules: Not In a construct rules (a variable referenced within a region, but not within a construct)

Analysis results for DataRaceBench v1.0.1

	Parallel	Shared				Sync.
		NIC	PDT	IDT	DSC	
Semantic Label Count	30	9	12	8	7	10
Covered Labels	PD1-4,6,8,11 12,14,15,PC02		2,3,5.1 5.2,6	2,3,4.1,6	2-6	1,10
Covered Label Count	11	0	5	4	5	2
Coverage Ratio	36.67%	0.0%	41.67%	50.0%	71.43%	20.0%

NIC01-NIC06: Not referenced in a construct rules
IDT01-IDT07: implicitly determined rules
PDT01-PDT09: predetermined attribute rules
DSC01-DSC07: data-sharing clauses

Improved coverage for v1.2.0

- 44 new microbenchmarks added into v1.2.0 to cover the missed semantic labels (72+44=116)

	Parallel	Shared				Sync.
		NIC	PDT	IDT	DSC	
Semantic Label Count	30	9	12	8	7	10
Covered Labels	all	all	all	all	all	all
Covered Label Count	30	9	12	8	7	10
Coverage Ratio	100%	100%	100%	100%	100%	100%

NIC01-NIC06: Not referenced in a construct rules
IDT01-IDT07: implicitly determined rules
PDT01-PDT09: predetermined attribute rules
DSC01-DSC07: data-sharing clauses

Example new microbenchmarks

```
class A {
public:
  static int ctr;
  static int pctr;
#pragma omp threadprivate(pctr)
};
int A::ctr=0;
int A::pctr=0;
A a;
void foo()
{
    a.ctr++;
    a.pctr++;
}
int main()
{
#pragma omp parallel
      foo();
//...
}
```

```
#include <stdio.h>
int a[100][100];
int main()
{
  int i, j;
#pragma omp parallel for ordered(2)
  for (i = 0; i < 100; i++)
    for (j = 0; j < 100; j++)
      {
        a[i][j] = a[i][j] + 1;
#pragma omp ordered depend(sink:i-1,j) \
            depend (sink:i,j-1)
        printf ("test i=%d j=%d\n",i,j);
#pragma omp ordered depend(source)
      }
  return 0;
}
```

Race-yes using static data members (NIC4)
DRB086-static-data-member-orig-yes.cpp

Race-no using ordered (N08, N09)
DRB094-doall2-ordered-orig-no.c

More example new microbenchmarks

```
  int result = 0;
#pragma omp parallel
#pragma omp single
  {
#pragma omp taskgroup
#pragma omp task
      {
        sleep(3); result = 1;
      }
#pragma omp task
    result = 2;
  }
  assert (result==2);
```

Race-no using taskgroup (N05)
DRB107-taskgroup-orig-no.c

```
// ...
  double a[len];

  /* Initialize with some values*/
  for (i=0; i<len; i++)
    a[i]= ((double)i)/2.0;

#pragma omp target map(tofrom: a[0:len])
#pragma omp teams num_teams(2)
  {
    a[50]*=2.0;
  }
```

Race-yes using target+teams(PD18)
DRB116-target-teams-orig-yes.c

Evaluation

Tool	Version	Compiler
Archer	towards_tr4 branch	Clang/LLVM 4.0.1
Intel Inspector	2018 (build 522981)	Intel Compiler 18.0.1

- DataRaceBench
  - https://github.com/LLNL/dataracebench v1.2.0
- Hardware
  - Quartz cluster of the Livermore Computing Center @LLNL
  - Each node: 2 Intel Xeon E5-2695 v4 processors, 18-core each, hyper threading support
- Execution configuration
  - Intel Inspector: inspxe-cl -collect ti3 -knob scope=extreme -knob stack-depth=16 -knob use-maximum-resources=true
  - 72 threads, repeat 5 times, 10-minutes timeout

Numbers of positive, negative and N/A results

Term	Meaning in our context
True Positive (TP)	Detecting data races in a race-yes program
False Positive (FP)	Detecting data races in a race-no program
True Negative (TN)	Not detecting data races in a race-no program
False Negative (FN)	Not detecting data races in a race-yes program
TP/FN	Mixed results with true positives and false negatives
TN/FP	Mixed results with true negatives and false positives
Not Available (N/A)	Compile-time seg. fault (CSF), Unsupported feature (CUN) Runtime seg. fault (RSF), Runtime timeout (RTO)

Tool	Race:Yes				Race:No			
	TP	TP/FN	FN	N/A	TN	TN/FP	FP	N/A
Archer	15	0	0	4	17	0	0	8
Intel Inspector	19	0	0	0	21	1	2	1

Evaluation report: part 1

Microbenchmark Program	R	Archer min - max race		Archer type	Intel Inspector min - max race		Intel Inspector type
DRB073-doall2-orig-yes.c	Y	84 -	92	TP	2 -	2	TP
DRB074-flush-orig-yes.c	Y	1 -	3	TP	1 -	1	TP
DRB075-getthreadnum-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB076-flush-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB077-single-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB078-taskdep2-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB079-taskdep3-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB080-func-arg-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB081-func-arg-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB082-declared-in-func-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB083-declared-in-func-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB084-threadprivatemissing-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB085-threadprivate-orig-no.c	N	-		CSF	0 -	0	TN
DRB086-static-data-member-orig-yes.cpp	Y	-		CSF	1 -	1	TP
DRB087-static-data-member2-orig-yes.cpp	Y	-		CSF	1 -	1	TP
DRB088-dynamic-storage-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB089-dynamic-storage2-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB090-static-local-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB091-threadprivate2-orig-no.c	N	-		CSF	0 -	0	TN
DRB092-threadprivatemissing2-orig-yes.c	Y	71 -	71	TP	1 -	1	TP
DRB093-doall2-collapse-orig-no.c	N	0 -	0	TN	0 -	0	TN
DRB094-doall2-ordered-orig-no.c	N	-		CUN	0 -	0	RTO

Evaluation report: part 2

Microbenchmark Program	R	Data Race Detection Tools					
		Archer			Intel Inspector		
		min race	max race	type	min race	max race	type
DRB095-doall2-taskloop-orig-yes.c	Y	-		CUN	2	2	TP
DRB096-doall2-taskloop-collapse-orig-no.c	N	-		CUN	0	4	FP TN
DRB097-target-teams-distribute-orig-no.c	N	0	0	RSF	0	0	TN
DRB098-simd2-orig-no.c	N	0	0	TN	0	0	TN
DRB099-targetparallelfor2-orig-no.c	N	0	0	TN	0	0	TN
DRB100-task-reference-orig-no.cpp	N	-		CUN	0	0	TN
DRB101-task-value-orig-no.cpp	N	0	0	TN	0	0	TN
DRB102-copyprivate-orig-no.c	N	-		CSF	0	0	TN
DRB103-master-orig-no.c	N	0	0	TN	0	0	TN
DRB104-nowait-barrier-orig-no.c	N	0	0	TN	0	0	TN
DRB105-taskwait-orig-no.c	N	0	0	TN	3	4	FP
DRB106-taskwaitmissing-orig-yes.c	Y	35	48	RTO TP	4	6	TP
DRB107-taskgroup-orig-no.c	N	0	0	TN	1	1	FP
DRB108-atomic-orig-no.c	N	0	0	TN	0	0	TN
DRB109-orderedmissing-orig-yes.c	Y	71	71	TP	1	1	TP
DRB110-ordered-orig-no.c	N	0	0	TN	0	0	TN
DRB111-linearmissing-orig-yes.c	Y	73	85	TP	1	2	TP
DRB112-linear-orig-no.c	N	-		CUN	0	0	TN
DRB113-default-orig-no.c	N	0	0	TN	0	0	TN
DRB114-if-orig-yes.c	Y	42	48	TP	1	1	TP
DRB115-forsimd-orig-yes.c	Y	44	47	TP	1	1	TP
DRB116-target-teams-orig-yes.c	Y	0	0	RSF	1	1	TP

Conclusion

- A semantics-driven approach to coverage analysis and improvement of DataRaceBench v.1.0.1
  - Defined semantics labels in three categories
  - Developed coverage analysis using a source-to-source analyzer
  - 44 new microbenchmarks were added to improve the coverage
  - https://github.com/LLNL/dataracebench v.1.2.0 released
- Re-evaluated two tools
  - Intel Inspector supported more microbenchmarks without compilation or runtime errors than Archer did
  - Archer did better than Inspector to support taskwait and taskgroup
- Surprises
  - Evaluations of dynamic tools also evaluate compilers
  - Found a misuse of the term construct in OpenMP 4.5
    - Declare simd is called construct: conflicting executable vs. non-executable semantics
  - The data-sharing attribute rules: surprisingly difficult
    - Need an official refined algorithm, like the one deciding the number of threads

Future work

- OpenMP runtime library routines and environment variables
- Better coverage analysis for cartesian product of the semantics dimensions
- Automatic approach to microbenchmark generation

Acknowledgement

Lawrence Livermore National Security, LLC, via projects LDRD 17-ERD-023.
Archer developers: Joachim Protze@RWTH Aachen Univ., Dong Ahn@LLNL
John-Mellor Crummey @ Rice Univ.