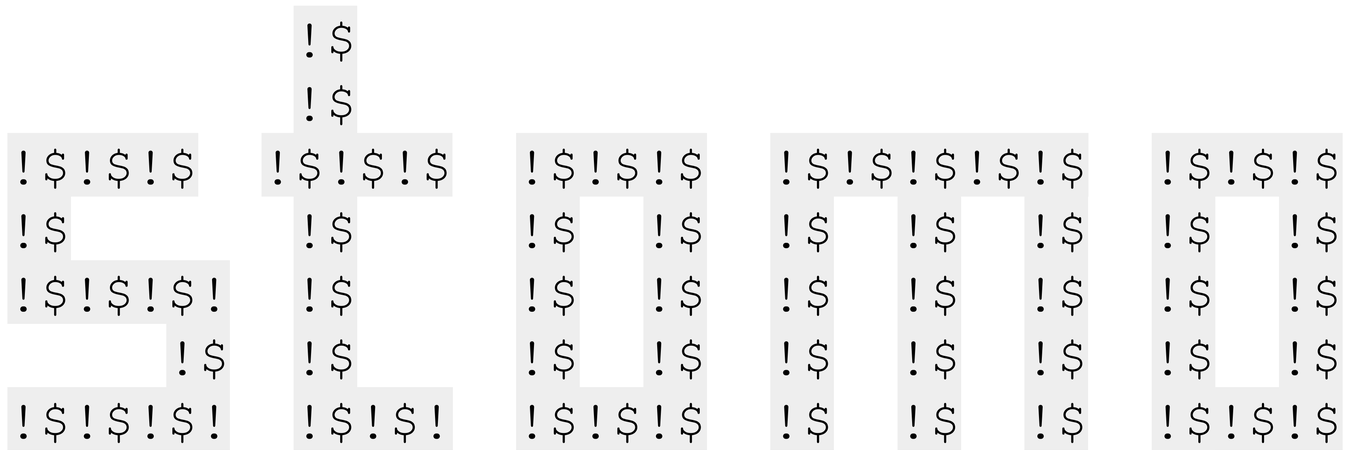


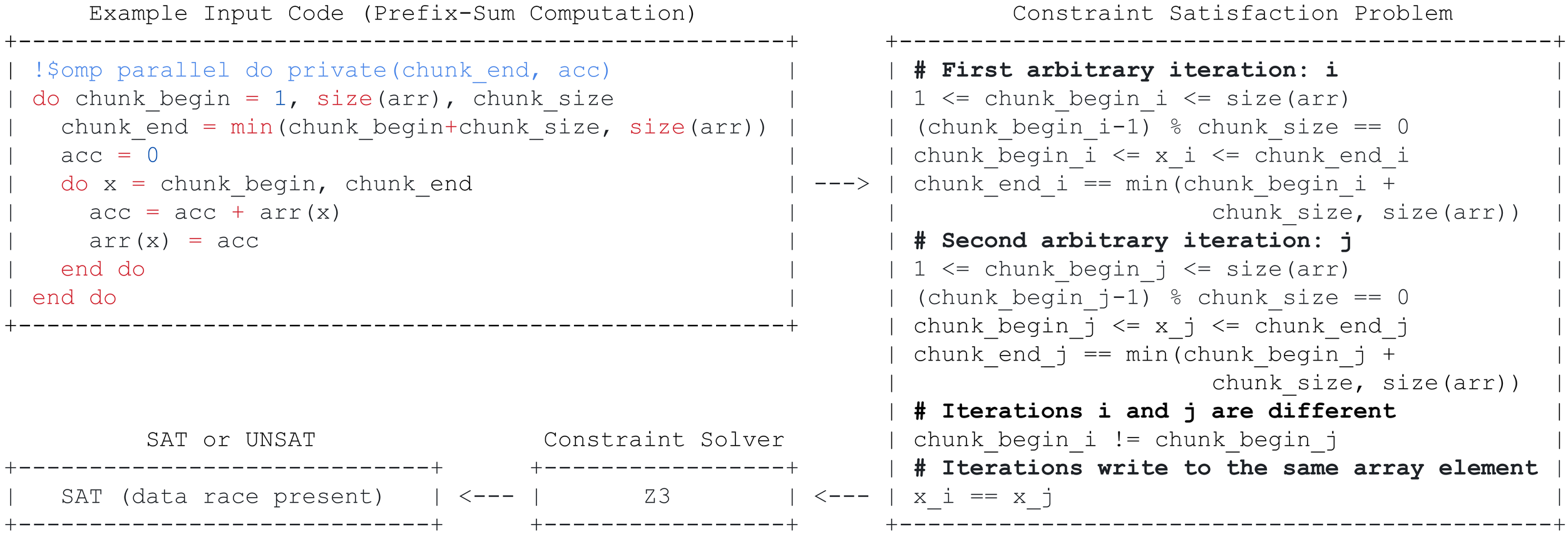
Matthew Naylor and Chris Edsall
Research Computing Services
University of Cambridge

Static Checking for Fortran OpenMP https://github.com/RSE-Cambridge/stomp

Introduction

Stomp is static checker for Fortran OpenMP combining **PSyclone** (a Python library for processing Fortran code developed by Met Office partners) and **Z3** (a theorem prover from Microsoft Research) to catch common parallel programming mistakes, such as data races and out-of-bounds array accesses, at compile time, avoiding time-consuming debug sessions.

How does it work?



Supported OpenMP constructs

Directives			Clauses			Functions
target	teams	distribute	shared	private	firstprivate	omp_get_thread_num()
parallel	do	barrier	reduction	default	lastprivate	omp_get_team_num()
atomic	critical	single	schedule	collapse	nowait	omp_get_num_threads()
master	threadprivate	sections	num_threads	num_teams	thread_limit	omp_get_num_teams()
section						

Stomp directives

```
!$omp parallel do private(j)
do i = 1, size(arr)
  j = some_array(i)
  !$stomp unique(j)
  arr(j) = 1
end do
```

<-- A key feature of Stomp is that it allows the user to resolve false positives by adding various !$stomp directives to the code. The user can specify (and document) their assumptions, and then check that the code is indeed safe under those assumptions. In the example (left), without the !$stomp directive, Stomp would be unable to determine that parallel accesses to 'arr' are non-conflicting.

Results

Stomp has been applied to the **Data Race Benchmark Suite 1.4.1** and **solves 140/166 Fortran problems**, rivalling Coderrect, a best-in-class commercial tool (no longer available). Stomp has also been successfully applied to GPU benchmarking, climate modelling, and nuclear fusion applications.

Tool	Available	Accuracy
Coderrect	No	81.4%
LLOV	Yes	67.7%
Stomp	Yes	84.3%