Goblint: Mixed Flow-Sensitive Abstract Interpretation

Simmo Saan[1] Julian Erhard[2,3] **Michael Schwarz**[2] Karoliine Holter[1] Michael Petter[2]
Vesal Vojdani[1] Helmut Seidl[2]

SV-COMP Community Meeting 2025

[1]University of Tartu [2]Technical University of Munich [3]Ludwig-Maximilians Universität München
`m.schwarz@tum.de`

Goblint in 4 Bullet Points

- Static Analyzer for C programs
- Based on abstract interpretation — sound!
- Overapproximating — no violations!
- Specializes in multi-threaded programs

Goblint in 4 Bullet Points

- Static Analyzer for C programs
- Based on abstract interpretation — sound!
- Overapproximating — no violations!
- Specializes in multi-threaded programs

A basic example approach:

- Accumulate all possible values for globals
- Track local states per program point

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g]$:

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g]:$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 0]$$

```
int g = 0;

main:                 ⊤
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 0]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g] : [0,0]$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g] : [0,0]$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0,8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0,8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g] : [0, 8]$

```
int g = 0;

main:                    ⊤
    x = 3;               {x ↦ [3,3]}
    g = 8;               {x ↦ [3,3]}
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

$\top$

$\{x \mapsto [3,3]\}$

$\{x \mapsto [3,3]\}$

$\{x \mapsto [3,3]\}$

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 8]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 10]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 10]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 10]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 42]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 42]$$

```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$[g] : [0, 42]$

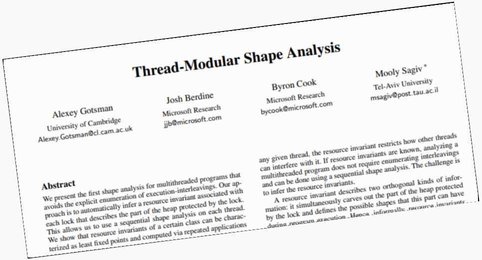
```
int g = 0;

main:
    x = 3;
    g = 8;
    if(g > 10) {
      error();
    }
    ...
```

```
t1:
    g = 10;
    ...
    g = 42;
```

$$[g] : [0, 42]$$

Zooming out:

- Accumulate information for global variables
- Track local states per program point



Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research
jjb@microsoft.com

Byron Cook
Microsoft Research
bycook@microsoft.com

Mooly Sagiv [*]
Tel-Aviv University
msagiv@post.tau.ac.il

Abstract

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications

any given thread, the resource invariant restricts how other threads can interfere with it. If resource invariants are known, analyzing a multithreaded program does not require enumerating interleavings and can be done using a sequential shape analysis. The challenge is to infer the resource invariants.

A resource invariant describes two orthogonal kinds of information: it simultaneously carves out the part of the heap protected by the lock and defines the possible shapes that this part can have

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research
jjb@microsoft.com

Byron Cook
Microsoft Research
bycook@microsoft.com

Mooly Sagiv *
Tel-Aviv University
msagiv@post.tau.ac.il

Abstract

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications

any given thread, the resource invariant restricts how other threads can interfere with it. If resource invariants are known, analyzing a multithreaded program does not require enumerating interleavings and can be done using a sequential shape analysis. The challenge is to infer the resource invariants.

A resource invariant describes two orthogonal kinds of information: it simultaneously carves out the part of the heap protected by the lock and defines the possible shapes that this part can have during program execution. Hence, informally, resource invariants

STATIC ANALYSIS OF RUN-TIME ERRORS IN EMBEDDED REAL-TIME PARALLEL C PROGRAMS

ANTOINE MINÉ

CNRS & École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France
e-mail address: mine@di.ens.fr

Abstract. We present a static analysis by Abstract Interpretation to check for run-time errors in parallel and multi-threaded C programs. Following our work on Astrée, we focus on embedded critical programs without recursion nor dynamic memory allocation, but extend the analysis to a static set of threads communicating implicitly through a shared memory and explicitly using a finite set of mutual exclusion locks, and scheduled according to a real-time scheduling policy and fixed priorities. Our method is thread-modular. It is

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research
jjb@microsoft.com

Byron Cook
Microsoft Research
bycook@microsoft.com

Mooly Sagiv *
Tel-Aviv University
msagiv@post.tau.ac.il

Abstract

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications

any given thread, the resource invariant restricts how other threads can interfere with it. If resource invariants are known, analyzing a multithreaded program does not require enumerating interleavings and can be done using a sequential shape analysis. The challenge is to infer the resource invariants.

A resource invariant describes two orthogonal kinds of information: it simultaneously carves out the part of the heap protected by the lock and defines the possible shapes that this part can have

STATIC ANALYSIS OF RUN-TIME ERRORS IN EMBEDDED REAL-TIME PARALLEL C PROGRAMS

ANTOINE MINÉ

CNRS & École Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France
e-mail address: mine@di.ens.fr

ABSTRACT. We present a static analysis by Abstract Interpretation to check for run-time errors in parallel and multi-threaded C programs. Following our work on Astrée, we focus on embedded critical programs without recursion nor dynamic memory allocation, but extend the analysis to a static set of threads communicating implicitly through a shared memory and explicitly using a finite set of mutual exclusion locks, and scheduled according

Relational Thread-Modular Static Value Analysis by Abstract Interpretation*

Antoine Miné

CNRS & École Normale Supérieure
45, rue d'Ulm
75005 Paris, France
mine@di.ens.fr

Abstract. We study thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research
jjb@microsoft.com

Byron Cook
Microsoft Research
bycook@microsoft.com

Abstract

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications

any given thread, the resource invariant can interfere with it. If resource invariant multithreaded program does not require and can be done using a sequential shap to infer the resource invariants.
A resource invariant describes two mation: it simultaneously carves out th by the lock and defines the possible during program execution. Hence, in

Improving Thread-Modular
Abstract Interpretation

Michael Schwarz[1], Simmo Saan[2], Helmut Seidl[1], Kalmer Apinis[2],
Julian Erhard[1], and Vesal Vojdani[2]

[1] Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
[2] University of Tartu, Tartu, Estonia
{simmo.saan, kalmer.apinis, vesal.vojdani}@ut.ee

Abstract. We give thread-modular non-relational value analyses as abstractions of a local trace semantics. The semantics as well as the analyses are formulated by means of global invariants and side-effecting constraint systems. We show that a generalization of the analysis provided by the static analyzer GOBLINT as well as a natural improvement of Antoine Miné's approach can be obtained as instances of this general scheme. We show that these two analyses are incomparable w.r.t. precision and

BEDDED

for run-time
e, we focus
cation, but
gh a shared
d according

Static Value
rpretation*

CNRS & École Normale Supérieure
45, rue d'Ulm
75005 Paris, France
mine@di.ens.fr

Abstract. We study thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research
jjb@microsoft.com

Byron Cook
Microsoft Research
bycook@microsoft.com

Abstract

We present the first shape analysis for multithreaded programs that avoids the explicit enumeration of execution-interleavings. Our approach is to automatically infer a resource invariant associated with each lock that describes the part of the heap protected by the lock. This allows us to use a sequential shape analysis on each thread. We show that resource invariants of a certain class can be characterized as least fixed points and computed via repeated applications

any given thread, the resource invariant
can interfere with it. If resource invarian
multithreaded program does not require
and can be done using a sequential shap
to infer the resource invariants.

A resource invariant describes two
mation: it simultaneously carves out th
by the lock and defines the possible s

Improving Thread-Modular Abstract Interpretation

Michael Schwarz[1], Simmo Saan[2], Helmut Seidl[1], Kalmer Apinis[2], Julian Erhard[1], and Vesal Vojdani[2]

[1] Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
[2] University of Tartu, Tartu, Estonia
{simmo.saan, kalmer.apinis, vesal.vojdani}@ut.ee

Abstract. We give thread-modular non-relational value analyses as abstractions of a local trace semantics. The semantics as well as the analyses are formulated by means of global invariants and side-effecting constraint systems. We show that a generalization of the analysis provided by the static analyzer GOBLINT as well as a natural improvement of Antoine Miné's approach can be obtained as instances of this general scheme. We show that these two analyses are incomparable w.r.t. precision and

BEDDED

for run-time
ee, we focus
cation, but
gh a shared
d according

Static Value
rpretation*

CNRS & École Normale Supérieure
45, rue d'Ulm
75005 Paris, France
mine@di.ens.fr

Abstract. We study thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety

Clustered Relational Thread-Modular Abstract Interpretation with Local Traces

Michael Schwarz[1](✉), Simmo Saan[2], Helmut Seidl[1], Julian Erhard[1], and Vesal Vojdani[2]

[1] Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
[2] University of Tartu, Tartu, Estonia
{simmo.saan, vesal.vojdani}@ut.ee

Abstract. We construct novel thread-modular analyses that track relational information for potentially overlapping clusters of global variables – given that they are protected by common mutexes. We provide a framework to systematically increase the precision of clustered relational analyses by splitting control locations based on abstractions of *local traces*. As

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research

Byron Cook
Microsoft Research

Improving Thread-Modular Abstract Interpretation

Global Invariants for Analyzing Multi-threaded Applications

Helmut Seidl*
Universität Trier, Germany

Varmo Vene†
Tartu University, Estonia

Markus Müller-Olm‡
FernUniversität Hagen, Germany

Abstract

We exhibit an interprocedural framework for the analysis of multi-threaded programs based on *partial invariants* of a new kind of constraint systems which we call *side-effecting*. We explore the formal properties of these constraint systems and provide general techniques for computing partial invariants. We demonstrate the practicality of this approach by designing and implementing a reasonably efficient flow- and context-sensitive interprocedural data-race analyzer of multi-threaded C.

Clustered … Abstract Interpretation with Local Traces

Michael Schwarz[1(✉)], Simmo Saan[2], Helmut Seidl[1], Julian Erhard[1], and Vesal Vojdani[2]

[1] Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
[2] University of Tartu, Tartu, Estonia
{simmo.saan, vesal.vojdani}@ut.ee

Abstract. We construct novel thread-modular analyses that track relational information for potentially overlapping clusters of global variables – given that they are protected by common mutexes. We provide a framework to systematically increase the precision of clustered relational analyses by splitting control locations based on abstractions of *local traces*. As

… Static Value … rpretation*

CNRS & École Normale Supérieure
45, rue d'Ulm
75005 Paris, France
mine@di.ens.fr

Abstract. We study thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety

Thread-Modular Shape Analysis

Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk

Josh Berdine
Microsoft Research

Byron Cook
Microsoft Research
bycook@microsoft.com

Abstract

We present the first shape analysis for multithr… avoids the explicit enumeration of execution-in… proach is to automatically infer a resource inva… each lock that describes the part of the heap p… This allows us to use a sequential shape ana… We show that resource invariants of a certain… terized as least fixed points and computed via…

A general method for rendering static analyses for diverse concurrency models modular

Quentin Stiévenart, Jens Nicolay, Wolfgang De Meuter, Coen De Roover

Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/j.jss.2018.10.001

Global Invariants for Analyzing Multi-threaded Applications

Helmut Seidl*
Universität Trier, Germany

Varmo Vene†
Tartu University, Estonia

Markus Müller-Olm‡
FernUniversität Hagen, Germany

Abstract

We exhibit an interprocedural framework for the analysis of multi-threaded programs based on *partial invariants* of a new kind of constraint systems which we call *side-effecting*. We explore the formal properties of these constraint systems and provide general techniques for computing partial invariants. We demonstrate the practicality of this approach by designing and implementing a reasonably efficient flow- and context-sensitive interprocedural data-race analyzer of multi-threaded C.

Clustered … Abstract Interpretation with Local Traces

Michael Schwarz[1(✉)], Simmo Saan[2], Helmut Seidl[1], Julian Erhard[1], and Vesal Vojdani[2]

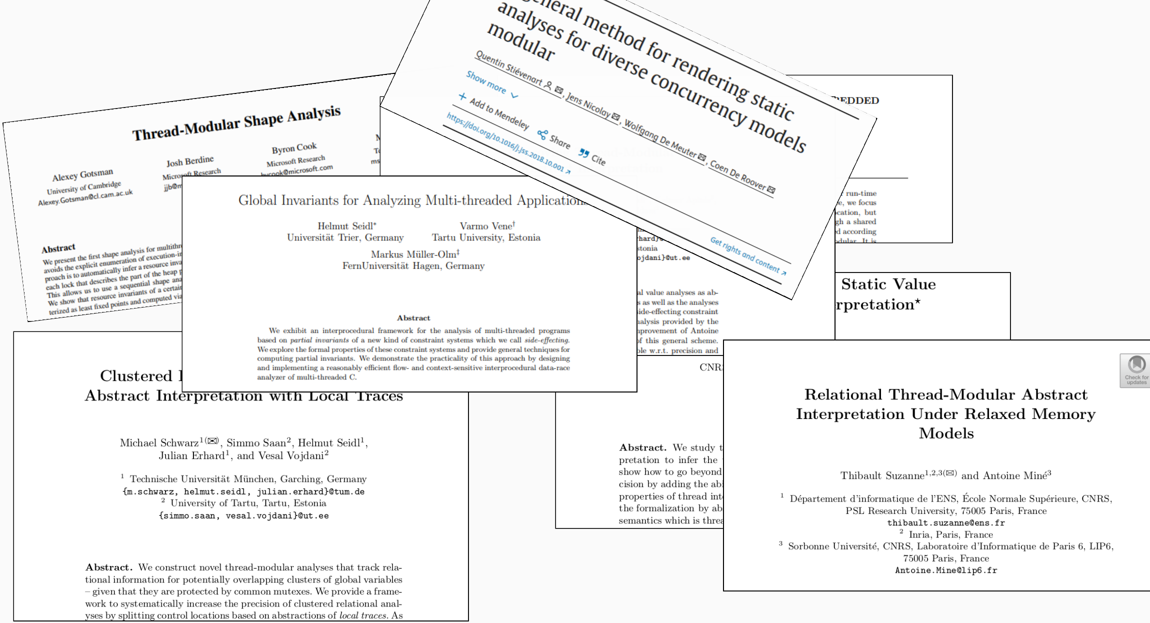
[1] Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
[2] University of Tartu, Tartu, Estonia
{simmo.saan, vesal.vojdani}@ut.ee

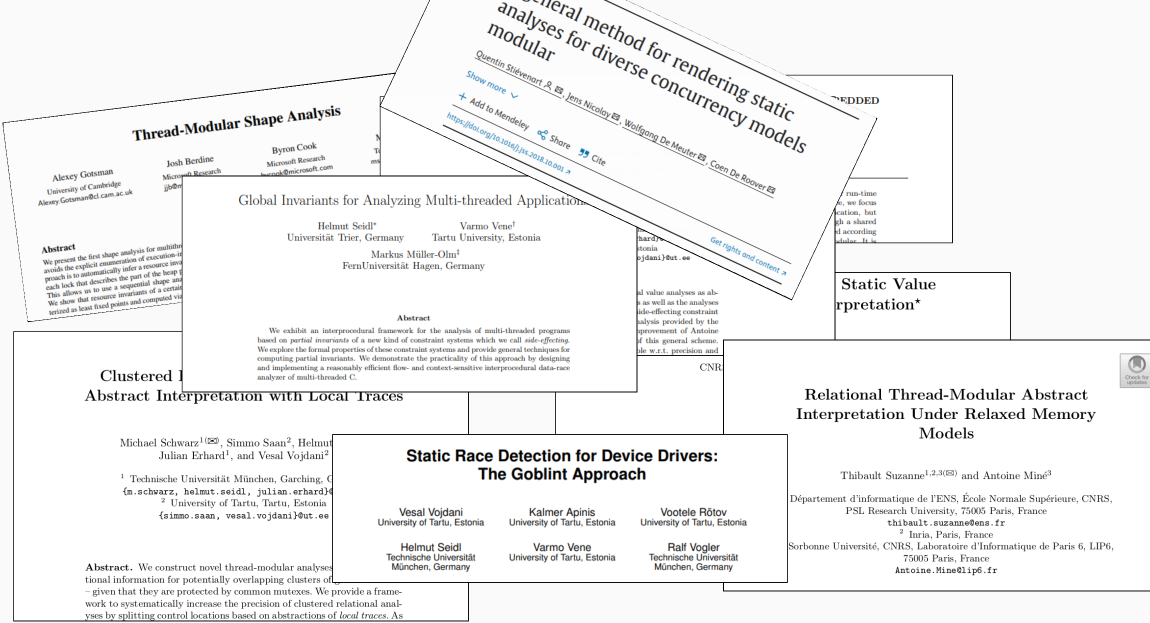
Abstract. We construct novel thread-modular analyses that track relational information for potentially overlapping clusters of global variables – given that they are protected by common mutexes. We provide a framework to systematically increase the precision of clustered relational analyses by splitting control locations based on abstractions of *local traces*. As

… Static Value … rpretation*

CNRS & École Normale Supérieure
45, rue d'Ulm
75005 Paris, France
mine@di.ens.fr

Abstract. We study thread-modular static analysis by abstract interpretation to infer the values of variables in concurrent programs. We show how to go beyond the state of the art and increase an analysis precision by adding the ability to infer some relational and history-sensitive properties of thread interferences. The fundamental basis of this work is the formalization by abstract interpretation of a rely-guarantee concrete semantics which is thread-modular, constructive, and complete for safety


Thread-Modular Shape Analysis
Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk
Josh Berdine
Microsoft Research
Byron Cook
Microsoft Research
bycook@microsoft.com
Abstract
A general method for rendering static analyses for diverse concurrency models modular
Quentin Stiévenart, Jens Nicolay, Wolfgang De Meuter, Coen De Roover
Show more
Add to Mendeley
Share
Cite
https://doi.org/10.1016/j.jss.2018.10.001
Get rights and content
Global Invariants for Analyzing Multi-threaded Applications
Helmut Seidl*
Universität Trier, Germany
Varmo Vene†
Tartu University, Estonia
Markus Müller-Olm‡
FernUniversität Hagen, Germany
Abstract
We exhibit an interprocedural framework for the analysis of multi-threaded programs based on partial invariants of a new kind of constraint systems which we call side-effecting. We explore the formal properties of these constraint systems and provide general techniques for computing partial invariants. We demonstrate the practicality of this approach by designing and implementing a reasonably efficient flow- and context-sensitive interprocedural data-race analyzer of multi-threaded C.
Clustered
Abstract Interpretation with Local Traces
Michael Schwarz1(✉), Simmo Saan2, Helmut Seidl1, Julian Erhard1, and Vesal Vojdani2
1 Technische Universität München, Garching, Germany
{m.schwarz, helmut.seidl, julian.erhard}@tum.de
2 University of Tartu, Tartu, Estonia
{simmo.saan, vesal.vojdani}@ut.ee
Abstract. We construct novel thread-modular analyses that track relational information for potentially overlapping clusters of global variables – given that they are protected by common mutexes. We provide a framework to systematically increase the precision of clustered relational analyses by splitting control locations based on abstractions of local traces. As
Static Value
rpretation★
Relational Thread-Modular Abstract Interpretation Under Relaxed Memory Models
Thibault Suzanne1,2,3(✉) and Antoine Miné3
1 Département d'informatique de l'ENS, École Normale Supérieure, CNRS, PSL Research University, 75005 Paris, France
thibault.suzanne@ens.fr
2 Inria, Paris, France
3 Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, 75005 Paris, France
Antoine.Mine@lip6.fr
Check for updates



Thread-Modular Shape Analysis
Alexey Gotsman
University of Cambridge
Alexey.Gotsman@cl.cam.ac.uk
Josh Berdine
Microsoft Research
Byron Cook
Microsoft Research
Abstract
A general method for rendering static analyses for diverse concurrency models modular
Quentin Stiévenart, Jens Nicolay, Wolfgang De Meuter, Coen De Roover
Show more
Add to Mendeley
Share
Cite
https://doi.org/10.1016/j.jss.2018.10.001
Get rights and content
Global Invariants for Analyzing Multi-threaded Applications
Helmut Seidl*
Universität Trier, Germany
Varmo Vene†
Tartu University, Estonia
Markus Müller-Olm‡
FernUniversität Hagen, Germany
Abstract
We exhibit an interprocedural framework for the analysis of multi-threaded programs based on partial invariants of a new kind of constraint systems which we call side-effecting. We explore the formal properties of these constraint systems and provide general techniques for computing partial invariants. We demonstrate the practicality of this approach by designing and implementing a reasonably efficient flow- and context-sensitive interprocedural data-race analyzer of multi-threaded C.
Static Value
rpretation*
Clustered
Abstract Interpretation with Local Traces
Michael Schwarz1(✉), Simmo Saan2, Helmut
Julian Erhard1, and Vesal Vojdani2
1 Technische Universität München, Garching,
{m.schwarz, helmut.seidl, julian.erhard}@
2 University of Tartu, Tartu, Estonia
{simmo.saan, vesal.vojdani}@ut.ee
Abstract. We construct novel thread-modular analyses
tional information for potentially overlapping clusters of
– given that they are protected by common mutexes. We provide a frame-
work to systematically increase the precision of clustered relational anal-
yses by splitting control locations based on abstractions of local traces. As
Static Race Detection for Device Drivers:
The Goblint Approach
Vesal Vojdani
University of Tartu, Estonia
Kalmer Apinis
University of Tartu, Estonia
Vootele Rõtov
University of Tartu, Estonia
Helmut Seidl
Technische Universität München, Germany
Varmo Vene
University of Tartu, Estonia
Ralf Vogler
Technische Universität München, Germany
Relational Thread-Modular Abstract Interpretation Under Relaxed Memory Models
Thibault Suzanne1,2,3(✉) and Antoine Miné3
Département d'informatique de l'ENS, École Normale Supérieure, CNRS, PSL Research University, 75005 Paris, France
thibault.suzanne@ens.fr
2 Inria, Paris, France
Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, 75005 Paris, France
Antoine.Mine@lip6.fr
Check for updates

Zooming out further

Differentiate:

- Flow-insensitive for some information:
- Flow-sensitive for other information:

Zooming out further

Differentiate:

- Flow-insensitive for some information: **Globals**
- Flow-sensitive for other information: **Locals**

Zooming out further

Differentiate:

- Flow-insensitive for some information: **Globals**
- Flow-sensitive for other information: **Locals**

Mixed Flow-Sensitivity

Example: Thread-Modular Analysis of Multi-Threaded Programs

- **Globals**: program globals
- **Locals**: program points of threads

Further Examples within Goblint

Partial Contexts / Context Lifters

- **Globals**: Start points of procedures
- **Locals**: Other program points of procedures

Further Examples within Goblint

Non-Local Control Flow via `setjmp/longjmp`

- **Globals**: Targets of longjumps
- **Locals**: Other program points of procedures

Mixed Flow-Sensitivity

Differentiate:

- Flow-insensitive for some information: **Globals**
- Flow-sensitive for other information: **Locals**

Generic concept arising in many settings

Mixed Flow-Sensitivity

Differentiate:

- Flow-insensitive for some information: **Globals**
- Flow-sensitive for other information: **Locals**

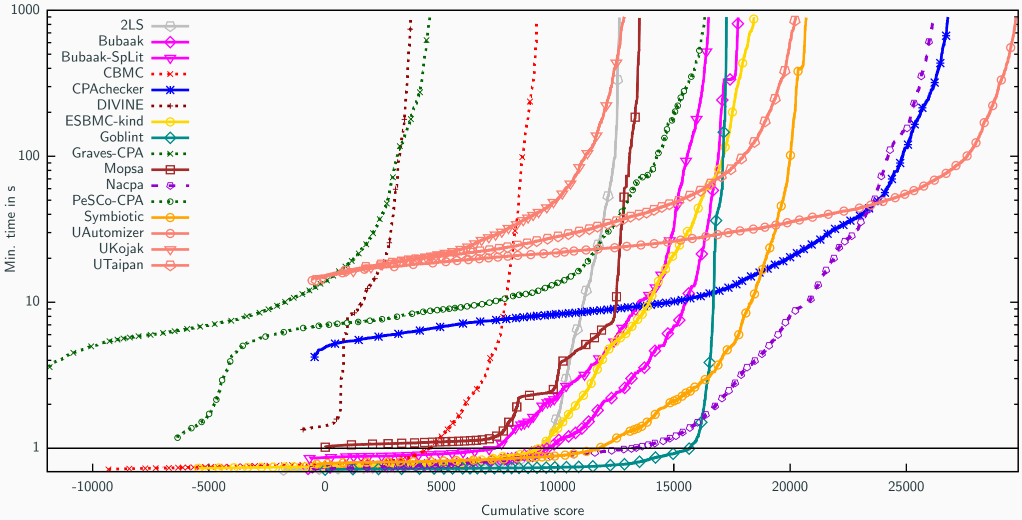
Generic concept arising in many settings

An implementation using side-effecting constraint systems lies at the heart of GOBLINT

Results in this year's competition

- Best in class for data race freedom
  - behind metaverifier CoOpeRace which incorporates Goblint
- Only tool to support all properties without producing any incorrect verdicts
  - (**Second year in a row**, this needs to change!)

Overall



2LS
Bubaak
Bubaak-SpLit
CBMC
CPAchecker
DIVINE
ESBMC-kind
Goblint
Graves-CPA
Mopsa
Nacpa
PeSCo-CPA
Symbiotic
UAutomizer
UKojak
UTaipan
Min. time in s
1000
100
10
1
-10000
-5000
0
5000
10000
15000
20000
25000
Cumulative score

Ockham Criterion

Proposed by [Black and Ribeiro, '16] for SATE V:

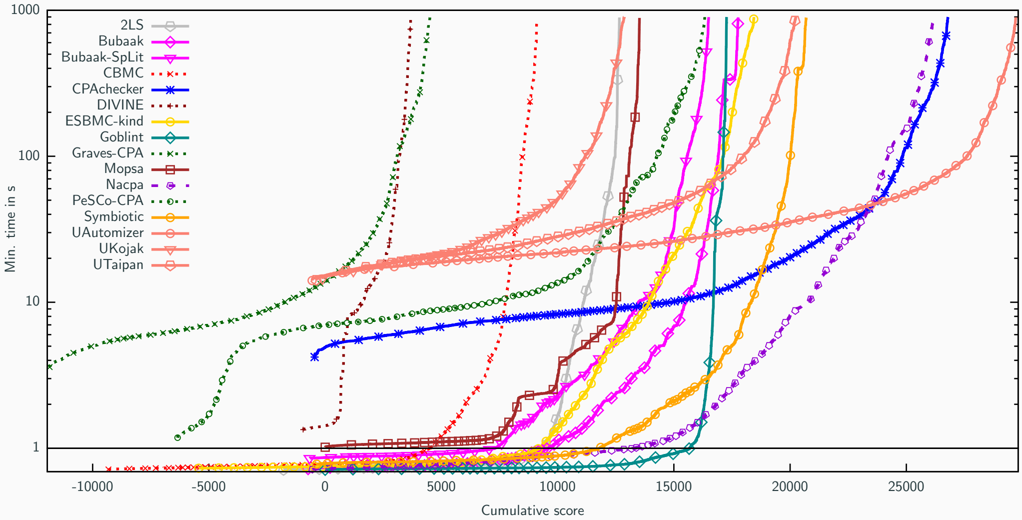
(1) The analyzer's findings are claimed to always be correct.

(2) It produces findings for most of a program.

(3) **Even one incorrect finding disqualifies an analyzer**.

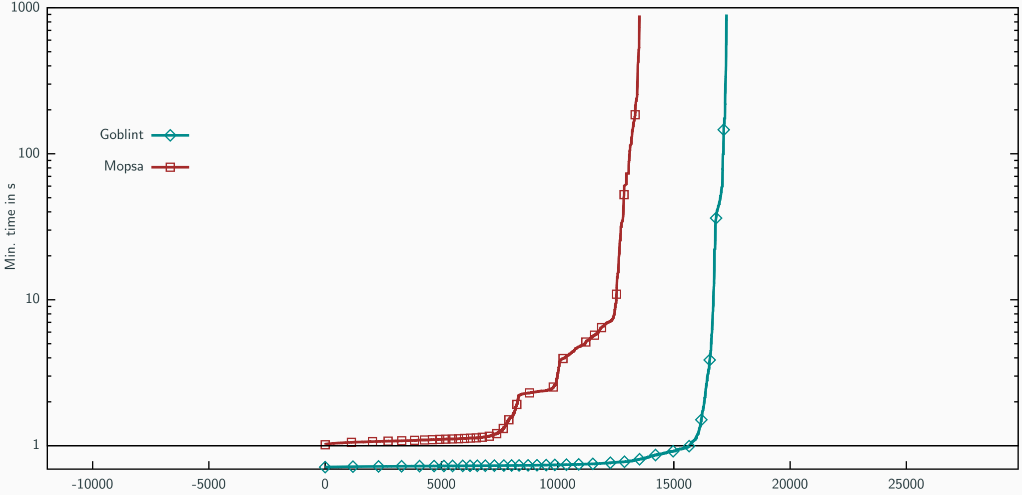
William of Ockham
(c. 1287-1347 CE)

Overall



1000
100
10
1
Min. time in s
-10000
-5000
0
5000
10000
15000
20000
25000
Cumulative score
2LS
Bubaak
Bubaak-SpLit
CBMC
CPAchecker
DIVINE
ESBMC-kind
Goblint
Graves-CPA
Mopsa
Nacpa
PeSCo-CPA
Symbiotic
UAutomizer
UKojak
UTaipan

Overall (Ockham)



Thank you!

- Mixed Flow-Sensitivity
  - Flow-sensitive for some unknowns
  - Flow-insensitive for others
- Best contestant for data races
- Best overall score without incorrect verdicts

/goblint/analyzer

Deutsche Forschungsgemeinschaft

This work was supported in part by Deutsche Forschungsgemeinschaft (DFG) – 378803395/2428 ConVeY.