Making Numerical Program Analysis Fast

Gagandeep Singh Markus Püschel Martin Vechev
Department of Computer Science
ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Static Program Analysis

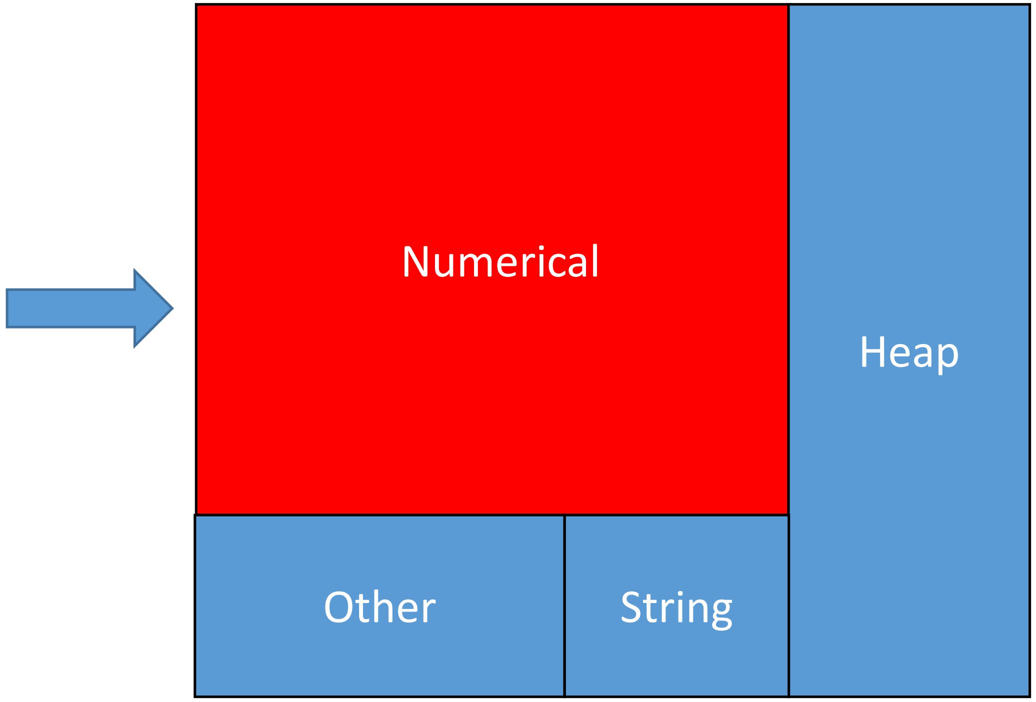
Static Program Analysis

```
public static void verify() {
  int[] ptr = new int[8];
  int start = 0;
  for(int i0 = 0; i0 < 8; ++i0) {
      int x1 = i0 | start;
      for(int x2 = 0; x2<100000;++x2) {
          int y3 = 2*x1;
          int index4 = 0;
          if (y3 == 0) { index4 = 1; }
          if (y3 == 49) { index4 = 8; }
          if (y3 == 36) { index4 = 8; }
          if (y3 == -1) { index4 = 0; }
          if (y3 == 50) { index4 = 9; }
          ptr[index4] = 1;
      }
    }
  }
```

Static Program Analysis

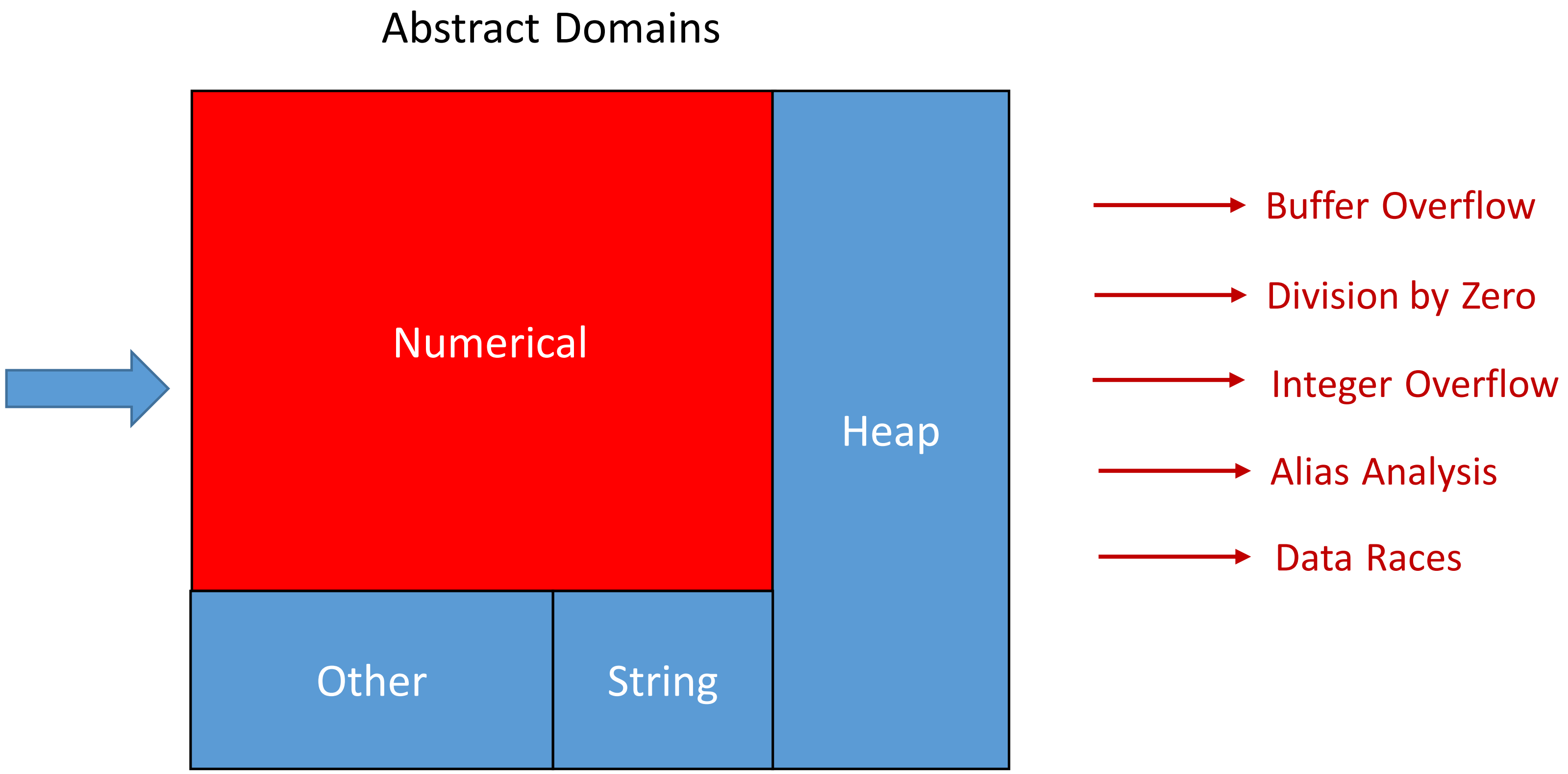
```
public static void verify() {
  int[] ptr = new int[8];
  int start = 0;
  for(int i0 = 0; i0 < 8; ++i0) {
      int x1 = i0 | start;
      for(int x2 = 0; x2<100000;++x2) {
          int y3 = 2*x1;
          int index4 = 0;
          if (y3 == 0) { index4 = 1; }
          if (y3 == 49) { index4 = 8; }
          if (y3 == 36) { index4 = 8; }
          if (y3 == -1) { index4 = 0; }
          if (y3 == 50) { index4 = 9; }
          ptr[index4] = 1;
      }
    }
  }
```

Abstract Domains



Static Program Analysis

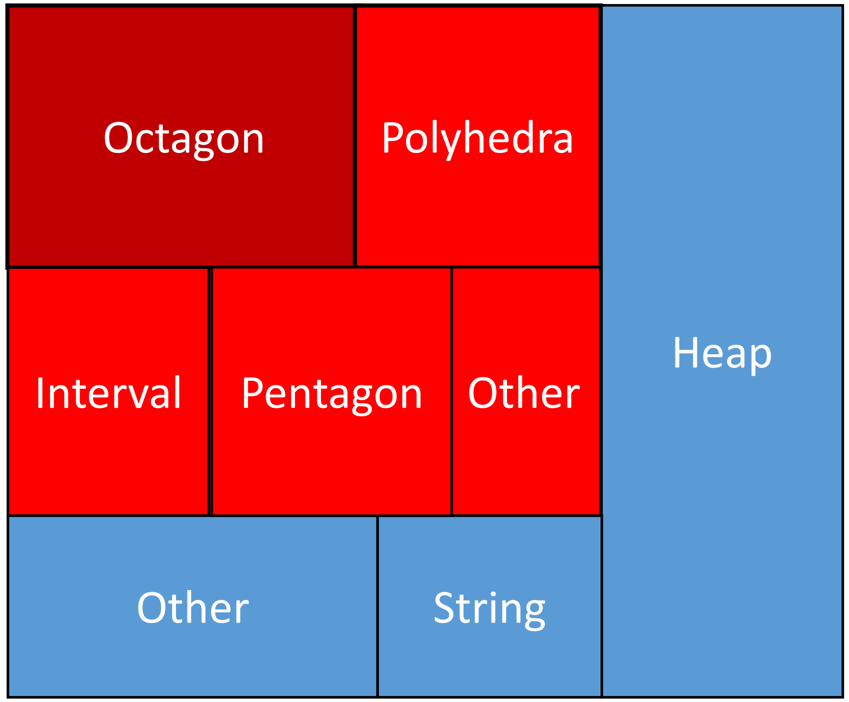
```
public static void verify() {
  int[] ptr = new int[8];
  int start = 0;
  for(int i0 = 0; i0 < 8; ++i0) {
      int x1 = i0 | start;
      for(int x2 = 0; x2<100000;++x2) {
         int y3 = 2*x1;
         int index4 = 0;
         if (y3 == 0) { index4 = 1; }
         if (y3 == 49) { index4 = 8; }
         if (y3 == 36) { index4 = 8; }
         if (y3 == -1) { index4 = 0; }
         if (y3 == 50) { index4 = 9; }
         ptr[index4] = 1;
      }
    }
  }
```



Static Program Analysis

```
public static void verify() {
  int[] ptr = new int[8];
  int start = 0;
  for(int i0 = 0; i0 < 8; ++i0) {
      int x1 = i0 | start;
      for(int x2 = 0; x2<100000;++x2) {
          int y3 = 2*x1;
          int index4 = 0;
          if (y3 == 0) { index4 = 1; }
          if (y3 == 49) { index4 = 8; }
          if (y3 == 36) { index4 = 8; }
          if (y3 == -1) { index4 = 0; }
          if (y3 == 50) { index4 = 9; }
          ptr[index4] = 1;
      }
    }
  }
```

Abstract Domains



→ Buffer Overflow

→ Division by Zero

→ Integer Overflow

→ Alias Analysis

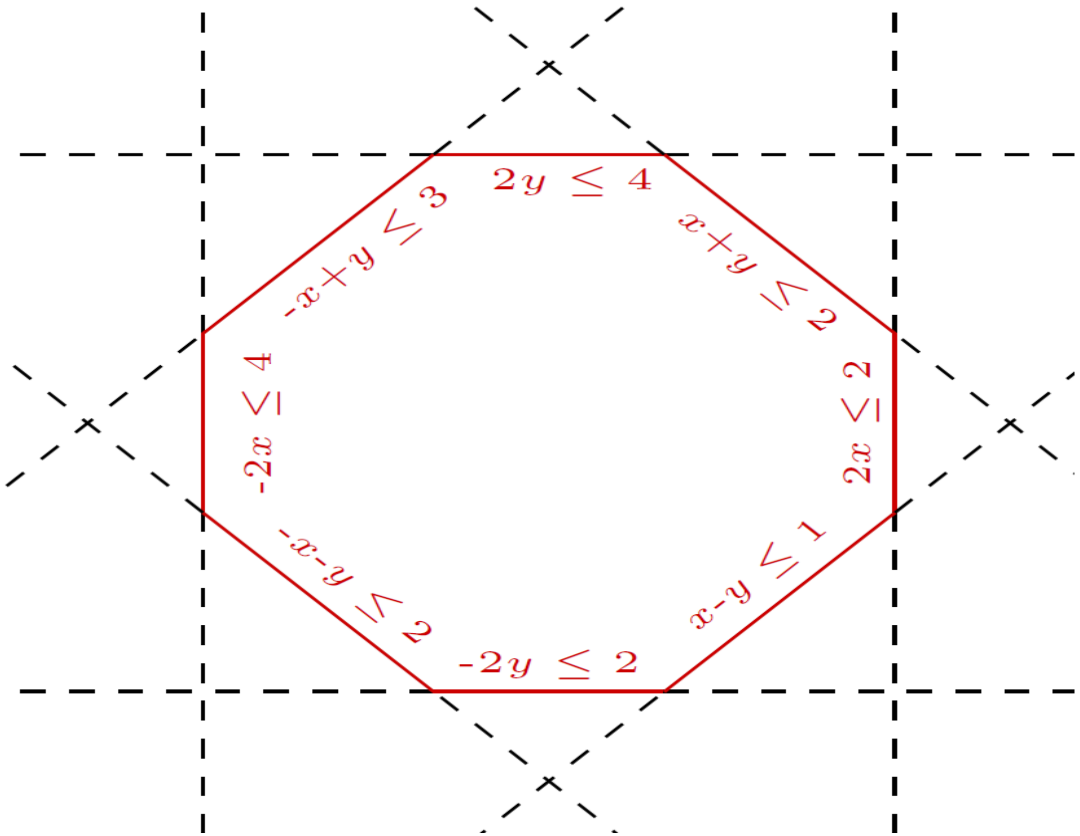
→ Data Races

Octagon Abstract Domain
(Miné, HOSC, 2006)

- Octagonal Inequalities:
  - Binary: $\pm x \pm y \leq c$, $x \neq y$
  - Unary: $\pm 2x \leq 2d$
  - $c, d \in \mathbb{R} \cup \{\infty\}$

Octagon Abstract Domain
(Miné, HOSC, 2006)

- Octagonal Inequalities:
  - Binary: ±x ± y ≤ c, $x \neq y$
  - Unary: ±2x ≤ 2d
  - c, d ∈ ℝ ∪ {∞}



Octagon

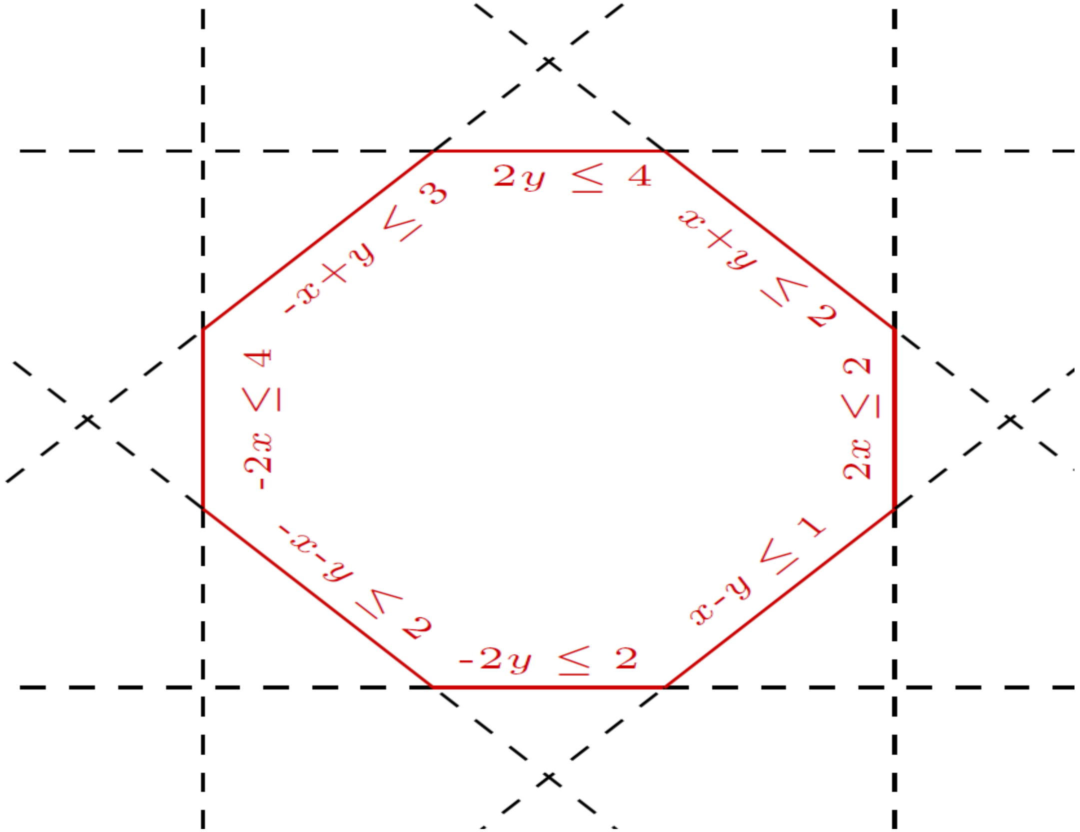
Octagon Abstract Domain

(Miné, HOSC, 2006)

- Octagonal Inequalities:
  - Binary: $\pm x \pm y \leq c$, $x \neq y$
  - Unary: $\pm 2x \leq 2d$
  - $c, d \in \mathbb{R} \cup \{\infty\}$

	x^+	x^-	y^+	y^-
x^+	0	4	3	2
x^-	2	0	2	1
y^+	1	2	0	2
y^-	2	3	4	0

Difference Bound Matrix (DBM)

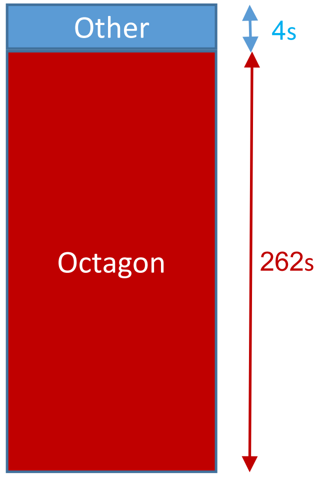


Octagon

Octagon Analysis is Expensive

Example: Static analyzer for TouchDevelop
(Brutschy et al. OOPSLA, 2014)

Using APRON

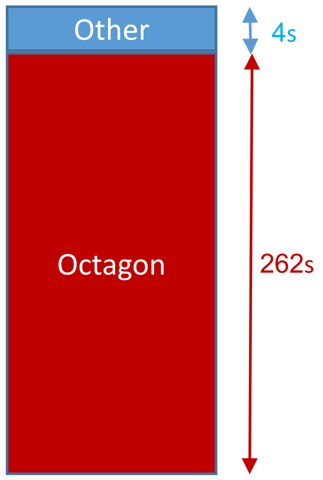


Single Core

Octagon Analysis is Expensive

Example: Static analyzer for TouchDevelop
(Brutschy et al. OOPSLA, 2014)

Using APRON



Single Core

Using ELINA

Single Core

Octagon Analysis is Expensive

Our Contribution: drop-in replacement for APRON

Example: Static analyzer for TouchDevelop
(Brutschy et al. OOPSLA, 2014)

Using APRON

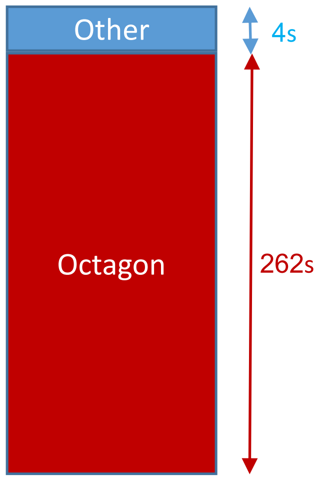
Using ELINA

Octagon Analysis is Expensive

Example: Static analyzer for TouchDevelop
(Brutschy et al. OOPSLA, 2014)

Our Contribution: drop-in replacement for APRON

Using APRON



Single Core

Using ELINA

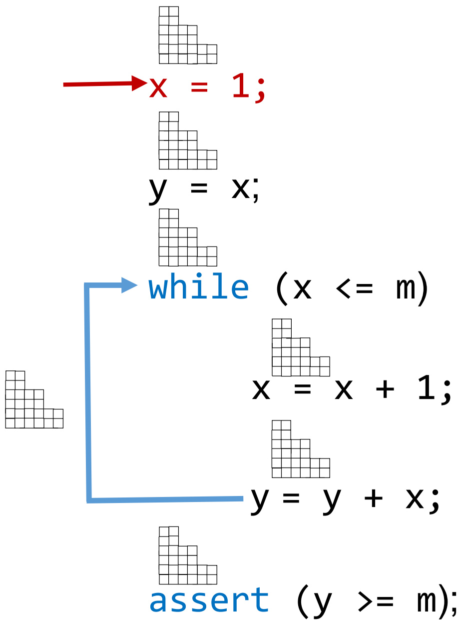
- Octagon Speedup: 26x
- Overall Speedup: 19x
- No loss in precision

Single Core

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

Octagon Analysis



Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	.				
x^-	.	0				
y^+	.	.	0	.		
y^-	.	.	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{}

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
        x = x + 1;
        y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	.				
x^-	.	0				
y^+	.	.	0	.		
y^-	.	.	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{}

$\sqcap$ { x ≤ 1, -x ≤ -1 }

$\sqcap$ { x ≤ 1, -x ≤ -1 }

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	.	.	0	.		
y^-	.	.		0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{2x ≤ 2, -2x ≤ -2 }

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
        x = x + 1;
        y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	.	.	0	.		
y^-	.	.	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

$\{2x \leq 2, -2x \leq -2\}$

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	.	.	0	.		
y^-	.	.	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{2x ≤ 2, -2x ≤ -2}

⊓ { y-x ≤ 0, x-y ≤ 0 }

⊓ { y -x≤ 0, x-y ≤ 0 }

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0		0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{2x ≤ 2, -2x ≤ -2, y - x ≤ 0, x – y ≤ 0}

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
        x = x + 1;
        y = y + x;
assert (y >= m);
```

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
      x = x + 1;
      y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

$\{2x \leq 2, -2x \leq -2, y - x \leq 0, x - y \leq 0\}$

Octagon Analysis

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{2x ≤ 2, -2x ≤ -2, y - x ≤ 0, x – y ≤ 0}

⊓ { x-m ≤ 0 }

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0		0		
m^+	0	.	.	.	0	.
m^-	.	.	.	.	.	0

⊓ {x-m ≤ 0 }

{2x ≤ 2, -2x ≤ -2, y - x ≤ 0, x – y ≤ 0,
x – m ≤ 0 }

Closure (*) increases precision of Join (⊔ (operator

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

Closure (*) increases precision of Join (⊔ (operator

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

$\{2x \leq 2, -2x \leq -2, y - x \leq 0, x - y \leq 0\}$

Closure (*) increases precision of Join (⊔ (operator

```
x = 1;
y = x;
while (x <= m)
        x = x + 1;
        y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	.	0	.		
y^-	.	0	.	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

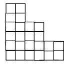
{2x ≤ 2, -2x ≤ -2, y - x ≤ 0, x – y ≤ 0}

↓ *

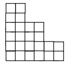
	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	-2	0	-2		
y^-	2	0	2	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

{2x ≤ 2, -2x ≤ -2, y - x ≤ 0, x – y ≤ 0,
-x – y ≤ -2, x + y ≤ 2, -2y ≤ -2, 2y ≤ 2 }

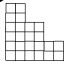
Join (⊔) of two closed Octagons



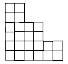
```
x = 1;
```



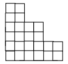
```
y = x;
```

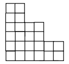


```
while (x <= m)
```

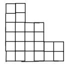


```
    x = x + 1;
```





```
    y = y + x;
```



```
assert (y >= m);
```

Join (⊔) of two closed Octagons

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	-2	0	-2		
y^-	2	0	2	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

Join (⊔) of two closed Octagons

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	-2	0	-2		
y^-	2	0	2	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-4				
x^-	4	0				
y^+	-1	-5	0	-6		
y^-	5	1	6	0		
m^+	1	-3	2	-4	0	-2
m^-	.	.	.	.	.	0

Join (⊔) of two closed Octagons

```
x = 1;
y = x;
while (x <= m)
    x = x + 1;
    y = y + x;
assert (y >= m);
```

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	2	0				
y^+	0	-2	0	-2		
y^-	2	0	2	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

$\sqcup$

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-4				
x^-	4	0				
y^+	-1	-5	0	-6		
y^-	5	1	6	0		
m^+	1	-3	2	-4	0	-2
m^-	.	.	.	.	.	0

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	4	0				
y^+	0	-2	0	-2		
y^-	5	1	6	0		
m^+	.	.	.	.	0	.
m^-	.	.	.	.	.	0

Time Complexity of Octagon Operators

Octagon Operator	Time Complexity
Meet (⊓)	O(n^2)
Join (⊔)	O(n^2)
Inclusion (⊆)	O(n^2)
Equality (=)	O(n^2)
Widening ($\overline{\nabla}$)	O(n^2)
Closure (*)	O(n^3)

Key Idea: Online Decomposition

- The set of program variables can be partitioned into disjoint subsets called independent components.
- Each independent component corresponds to a smaller octagon.
- Transitive closure can be applied independently on smaller octagons.
- Maintain the decomposition dynamically throughout the analysis.

	x^+	x^-	y^+	y^-	m^+	m^-
x^+	0	-2				
x^-	4	0				
y^+	0	-2	0	-2		
y^-	5	1	6	0		
m^+	.	.	.	.	0	2
m^-	.	.	.	.	.	0

	x^+	x^-	y^+	y^-
x^+	0	-2		
x^-	4	0		
y^+	0	-2	0	-2
y^-	5	1	6	0

	m^+	m^-
m^+	0	2
m^-	.	0

Other Improvements

- We reduced operation count of closure by half.
- We designed sparse closure for very sparse matrices that runs in O(n^2) time.
- We performed cache optimizations and vectorization for all octagon operators.
- If the matrix becomes dense, keeping decomposition is not feasible.
    - We designed different octagon types and their corresponding operators.
    - We keep track of sparsity and switch dynamically between different types.

Implementation

- ELINA is implemented in C using double precision.
- Provides interface for analyzing program written in C++ and Java.
- Supports SSE and AVX intrinsics.
- Can be directly plugged into any existing static analyzer using APRON.

Experimental Evaluation

- CPAchecker (Beyer et al. CAV, 2011)
  - participates in software verification competitions.
- TOUCHBOOST (Brutschy et al. OOPSLA, 2014)
  - analyzes eventdriven TouchDevelop applications.
- DPS (Raychev et al. SAS, 2013)
  - analyzes parallel programs and introduces synchronization for determinism.
- DIZY (Partush et al. SAS, 2013)
  - computes semantic differences between a program and its patched version.

Experimental Results: CPAchecker
(Beyer et al., CAV, 2011)

Using APRON

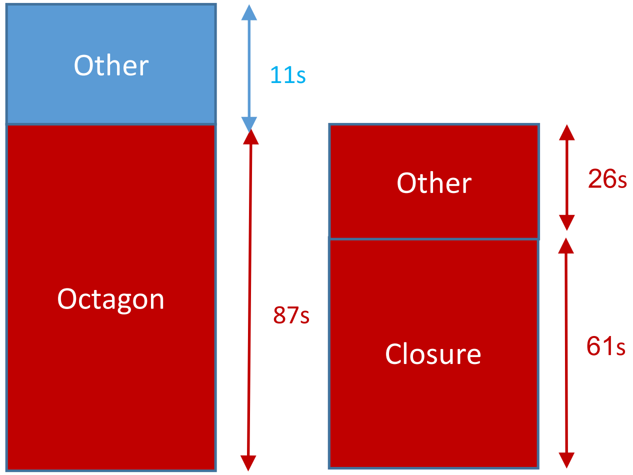
Using ELINA

15s

Experimental Results: CPAchecker

(Beyer et al., CAV, 2011)

Using APRON



Single Core

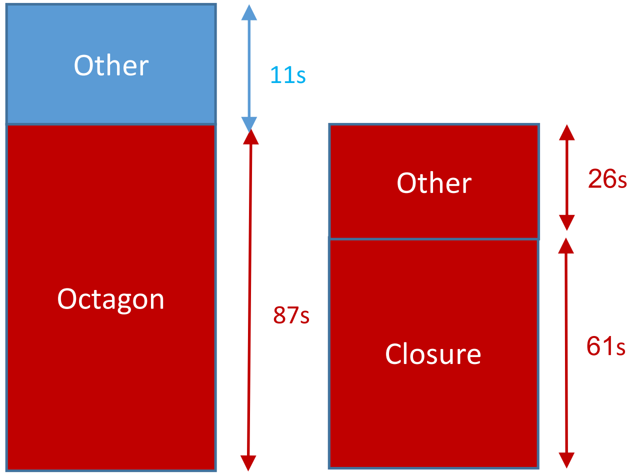
Using ELINA

15s

Experimental Results: CPAchecker

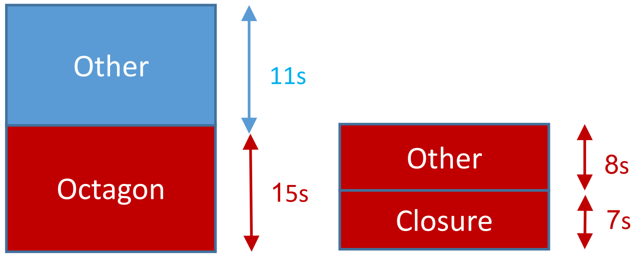
(Beyer et al., CAV, 2011)

Using APRON



Single Core

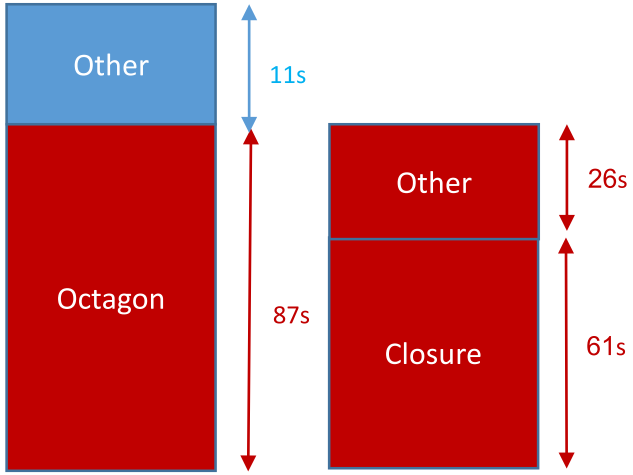
Using ELINA



Single Core

Experimental Results: CPAchecker
(Beyer et al., CAV, 2011)

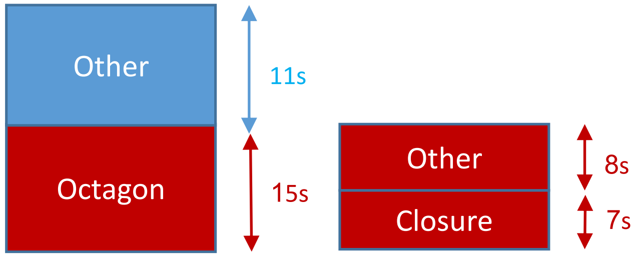
Using APRON



Single Core

Using ELINA

- Closure Speedup: 8.4x
- Octagon Speedup: 6x
- Overall Speedup: 3.7x



Single Core

Experimental Results: DPS

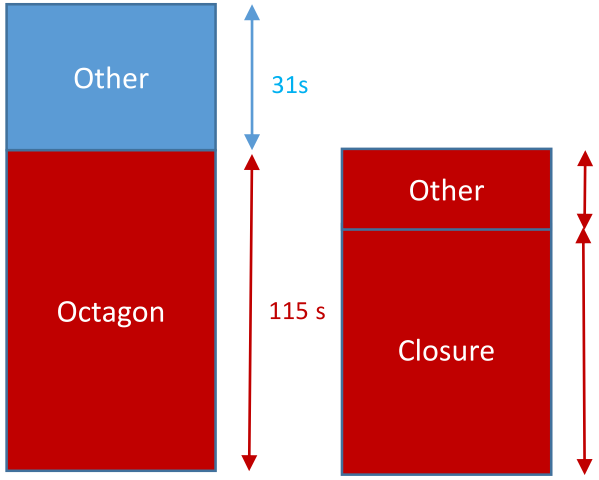
(Raychev et al, SAS, 2013)

Using APRON

Using ELINA

Experimental Results: DPS
(Raychev et al, SAS, 2013)

Using APRON



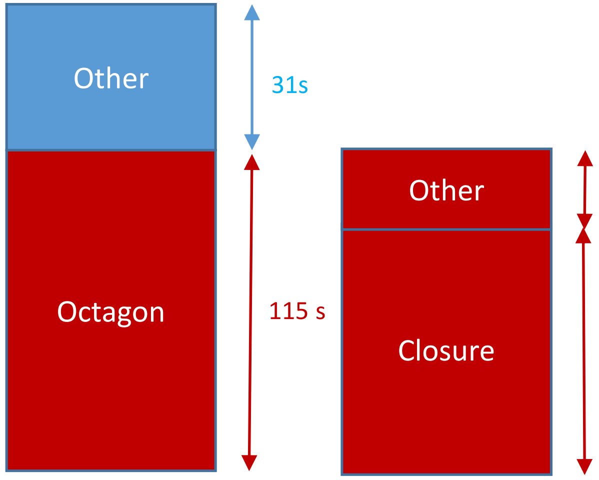
Single Core

Using ELINA

Experimental Results: DPS

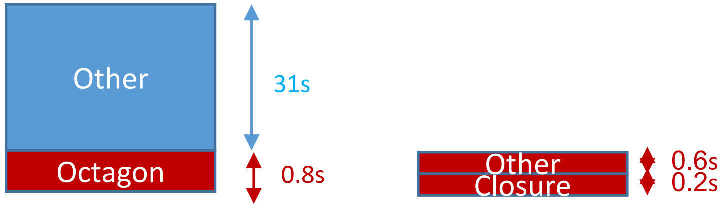
(Raychev et al, SAS, 2013)

Using APRON



Single Core

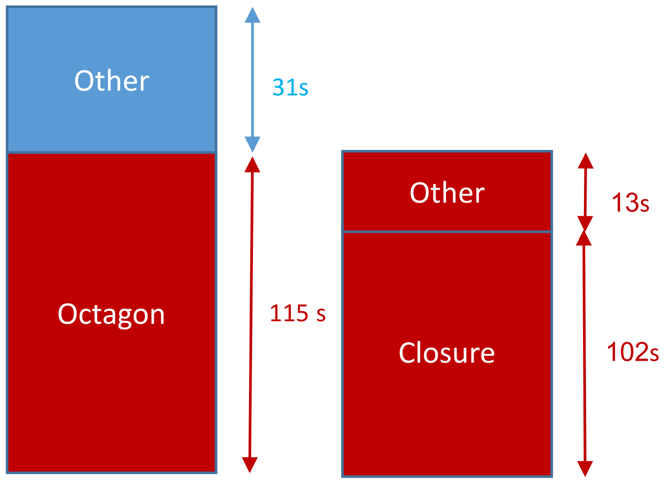
Using ELINA



Single Core

Experimental Results: DPS
(Raychev et al, SAS, 2013)

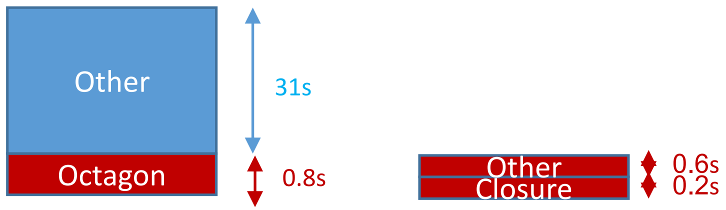
Using APRON



Single Core

Using ELINA

- Closure Speedup: 665x
- Octagon Speedup: 146x
- Overall Speedup: 4.2x

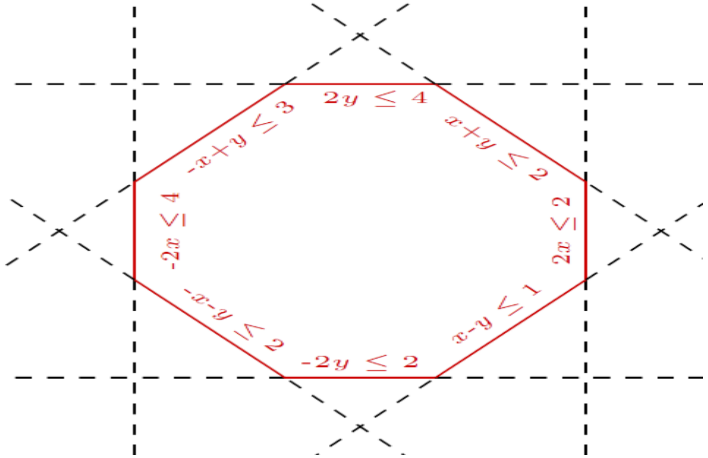


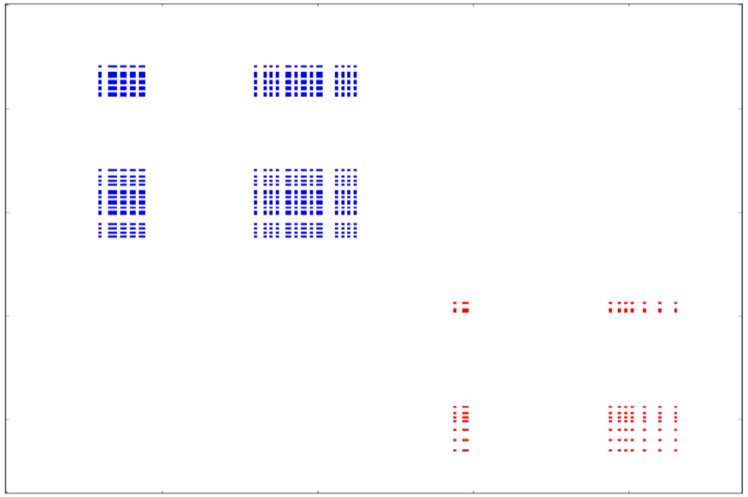
Single Core

Related Work

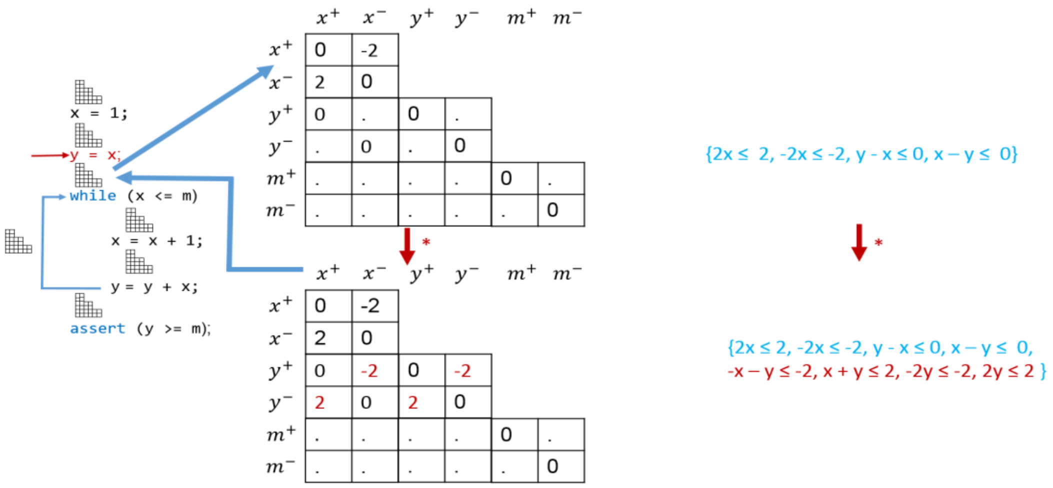
- Variable Packing (Venet et al. PLDI, 2004)
    - Loses precision, may take more iterations to converge.
- Octagon operators on GPUs (Banterle et al. SAS, 2007)
    - Our optimized library will run much faster on GPUs.

Conclusion





Closure (*) increases precision of Join (⊔) operator



Octagon Analysis is Expensive

Example: Static analyzer for TouchDevelop
(Brutschy et al. OOPSLA, 2014)

Using APRON **Using ELINA**

Single Core Single Core

https://github.com/eth-srl/OptOctagon