Lightweight Verification of Array Indexing

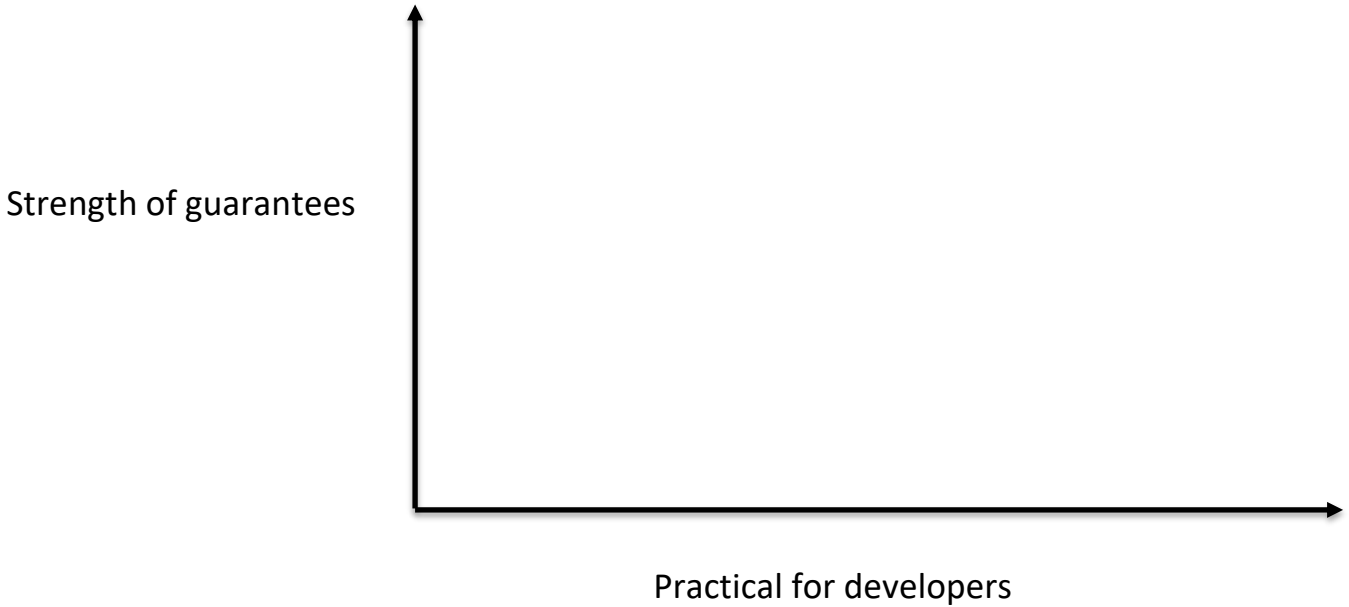
Martin Kellogg*, Vlastimil Dort**, Suzanne Millstein*, Michael D. Ernst*

* University of Washington, Seattle
** Charles University, Prague

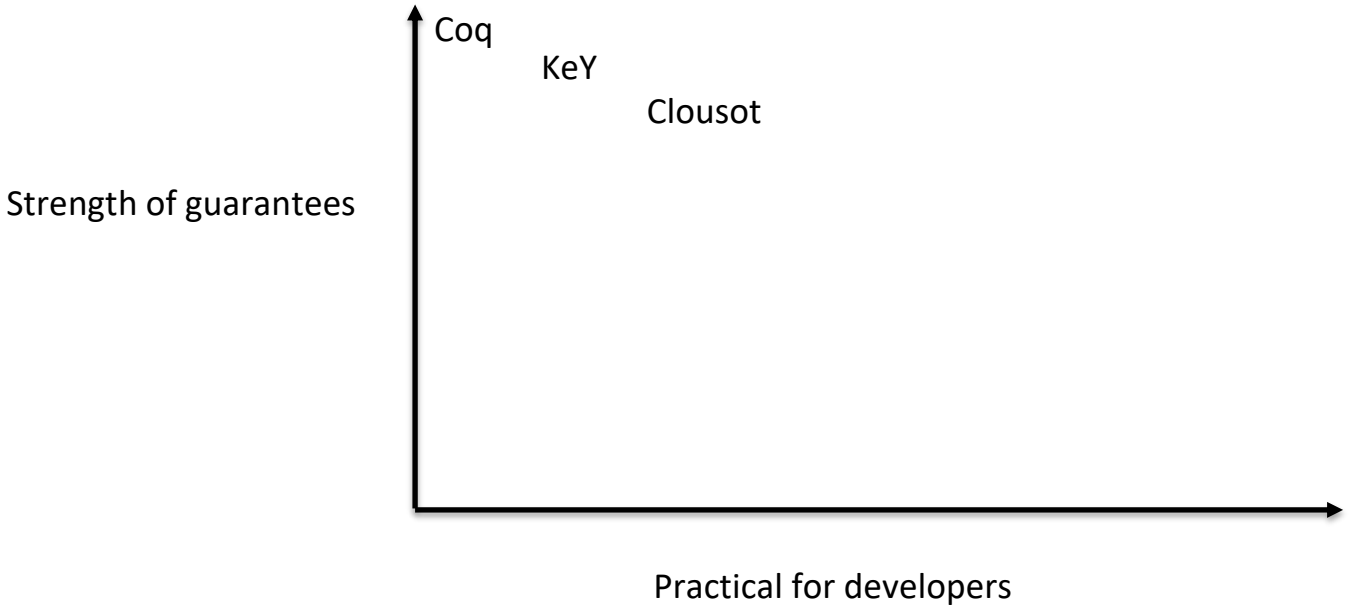
The problem: unsafe array indexing

- In unsafe languages (C): buffer overflow!
- In managed languages (Java, C#, etc.): exception, program crashes

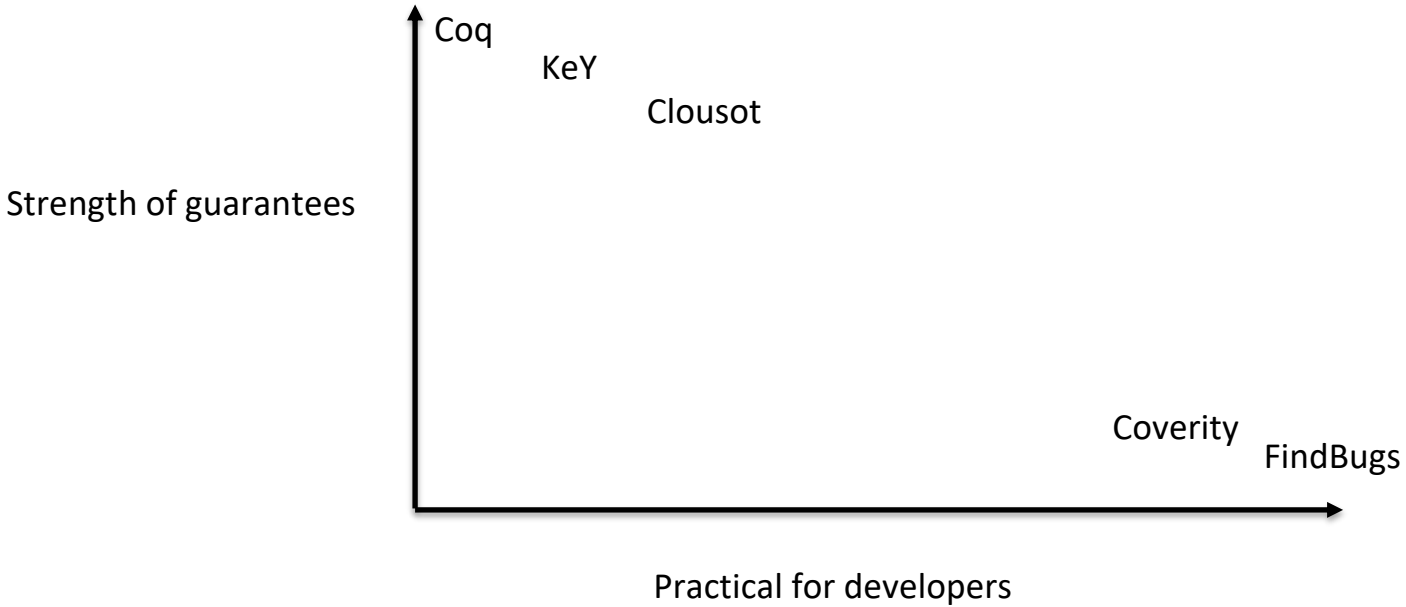
The state of the art



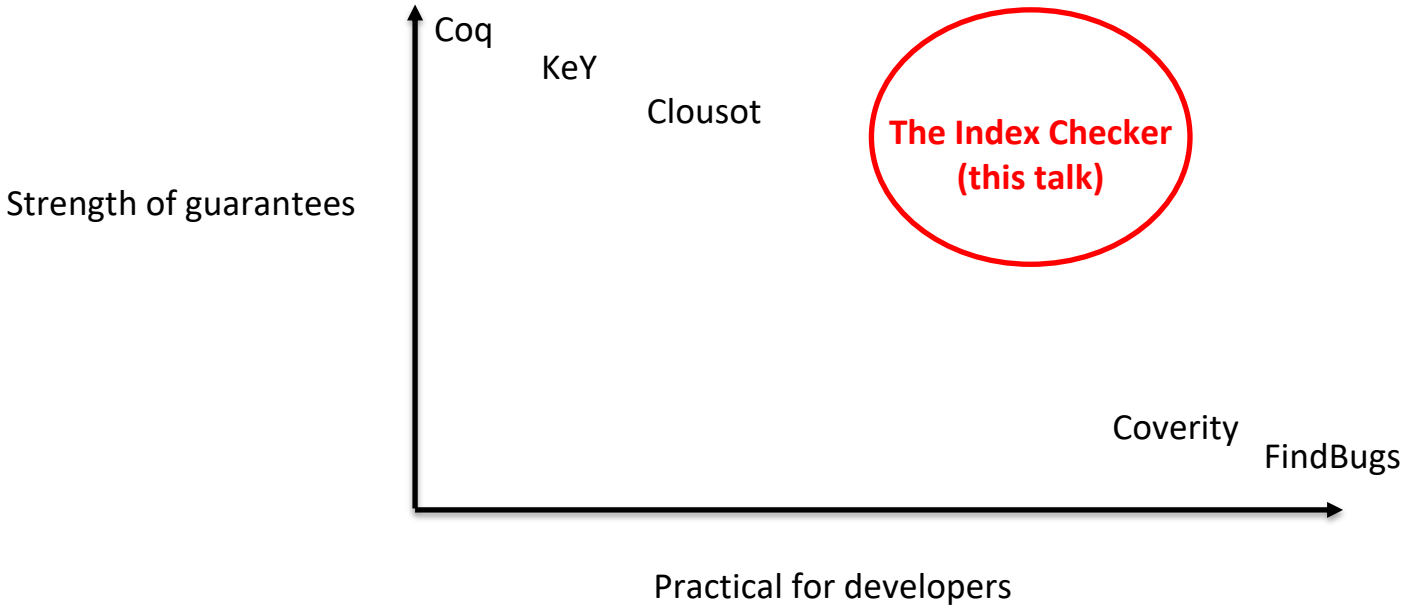
The state of the art



The state of the art



The state of the art



Problems with complex analyses

- false positives

- annotation burden

- complex analyses are hard to predict

Problems with complex analyses

- false positives
    - bounds checking is hard → complex analysis
    - complex analysis → harder to implement
    - harder to implement → more false positives
- annotation burden

- complex analyses are hard to predict

Problems with complex analyses

- false positives
    - bounds checking is hard → complex analysis
    - complex analysis → harder to implement
    - harder to implement → more false positives
- annotation burden
    - complex analysis → complex annotations

- complex analyses are hard to predict

Problems with complex analyses

- false positives
    - bounds checking is hard → **complex analysis**
    - **complex analysis** → harder to implement
    - harder to implement → more false positives
- annotation burden
    - **complex analysis** → complex annotations

- **complex analyses** are hard to predict

Insight:

Fundamental problem is complex analyses!

Cooperating simple analyses

Solve all three problems:

Cooperating simple analyses

Solve all three problems:

- **simpler implementation → fewer false positives**

Cooperating simple analyses

Solve all three problems:

- **simpler implementation → fewer false positives**
- **simpler abstractions → easier to write annotations**

Cooperating simple analyses

Solve all three problems:

- **simpler implementation → fewer false positives**
- **simpler abstractions → easier to write annotations**
- **simpler analysis → simpler to predict**

Proving an array access safe

```
T[] a = …;
int i = …;
... a[i] ...
```

We need to show that:

- `i` is an index for a

Proving an array access safe

```
T[] a = …;
int i = …;
... a[i] ...
```

We need to show that:

- ~~i is an index for a~~
- i ≥ 0
- i < a.length

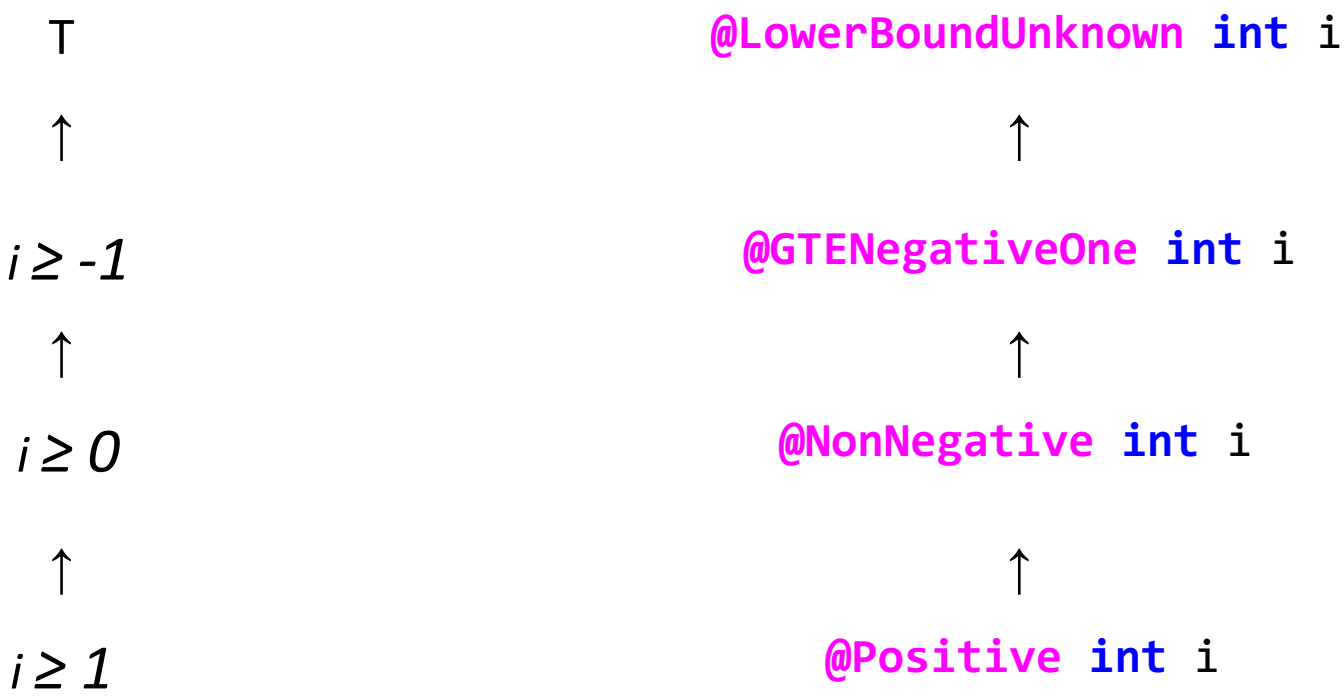
Proving an array access safe

```
T[] a = …;
int i = …;
... a[i] ...
```

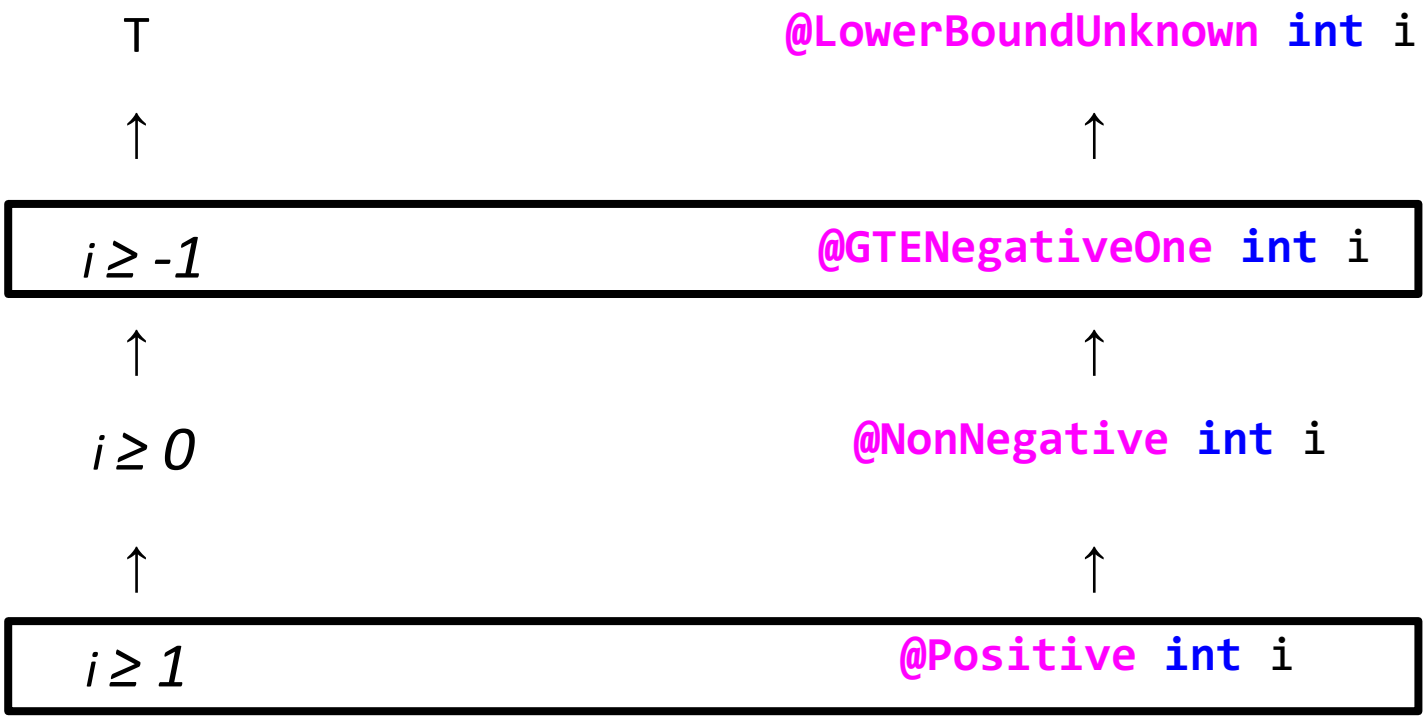
We need to show that:

- ~~`i` is an index for `a`~~
- `i ≥ 0` ← A lower bound on `i`
- `i < a.length` ← An upper bound on `i`

A type system for lower bounds



A type system for lower bounds



A type system for upper bounds

```
if (i >= 0 && i < a.length) {
        a[i] = ...
}
```

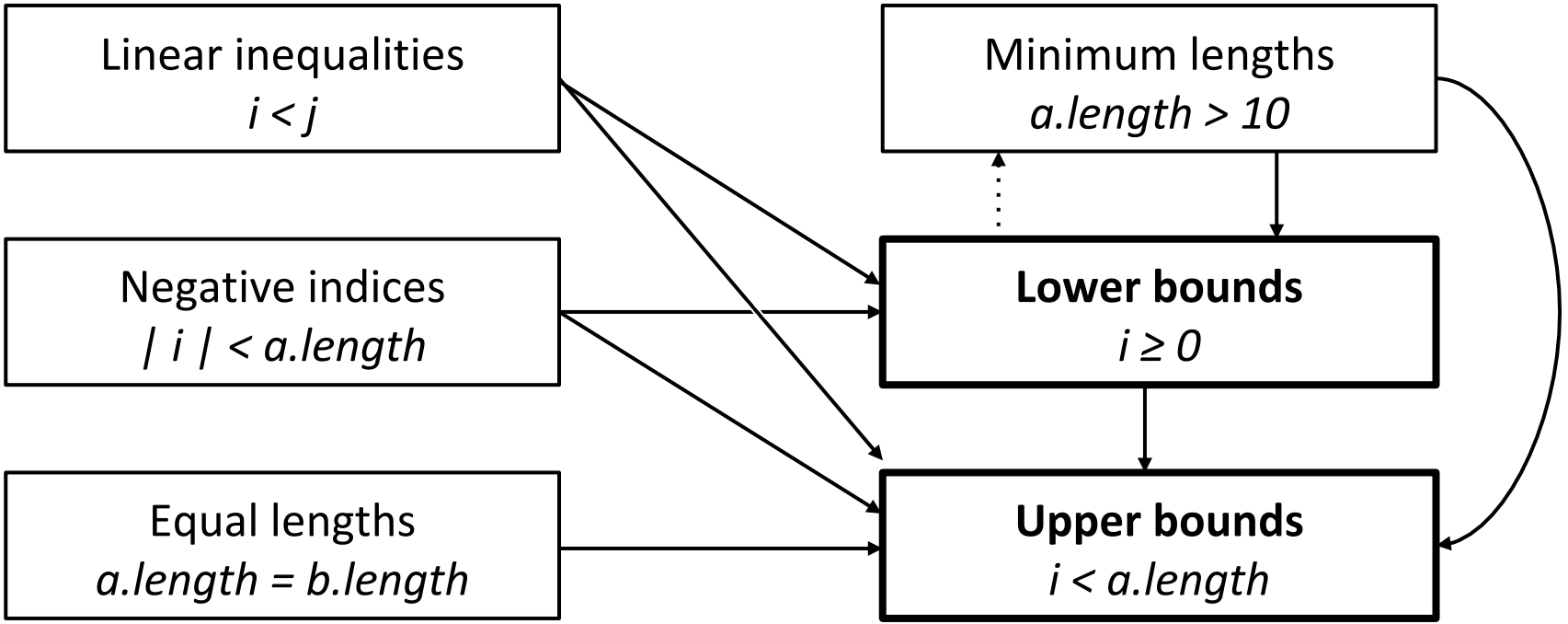
A type system for upper bounds

```
if (i >= 0 && i < a.length) {
        a[i] = ...
}
```

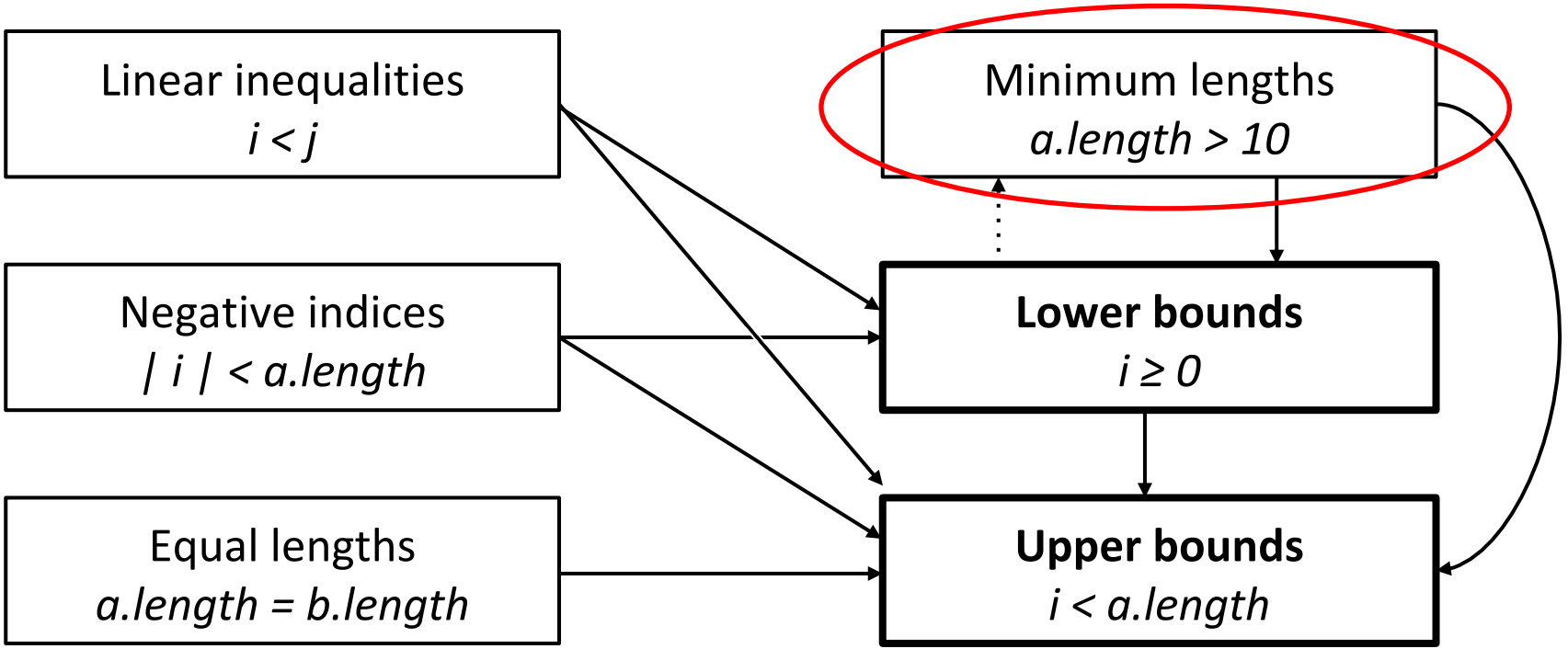
i < a.length

```
@LTLengthOf("a") int i
```

Type systems



Type systems



A type system for minimum array lengths

```
if (a.length >= 3) {
        a[2] = ...;
}
```

A type system for minimum array lengths

```
if (a.length >= 3) {
        a[2] = ...;
}
```

a.length ≥ i

```
T @MinLen(i) [] a
```

Evaluation

Three case studies:

- **Google Guava (two packages)**
- **JFreeChart**
- **plume-lib**

Comparison to existing tools:

- **FindBugs, KeY, Clousot**

Case Studies

	Guava	JFreeChart	plume-lib	Total
Lines of code	10,694	94,233	14,586	119,503
Bugs found	5	64	20	89
Annotations	510	2,938	241	3,689
False positives	138	386	43	567
Java casts	222	2,740	219	3,181

Comparison to other tools: confirmed bugs

Approach	Types	Bug finder	Verif. w/ solver	Abs. interpret.
Tool	**Index Checker**	**FindBugs**	**KeY**	**Clousot**
True Positives				
False Negatives				
Time (100k LoC)				

Comparison to other tools: confirmed bugs

Approach	Types	Bug finder	Verif. w/ solver	Abs. interpret.
Tool	**Index Checker**	**FindBugs**	**KeY**	**Clousot**
True Positives				
False Negatives				
Time (100k LoC)	~10 minutes	~1 minute	cannot scale	~200 minutes

Comparison to other tools: confirmed bugs

Approach	Types	Bug finder	Verif. w/ solver	Abs. interpret.
Tool	**Index Checker**	**FindBugs**	**KeY**	**Clousot**
True Positives	18/18	0/18	9/18	16/18
False Negatives	0/18	18/18	1/18	2/18
Time (100k LoC)	~10 minutes	~1 minute	cannot scale	~200 minutes

Using the Index Checker

- Distributed with Checker Framework

www.checkerframework.org

Contributions

- A methodology: simple, cooperative type systems
- An analysis: abstractions for array indexing
- An implementation and evaluation for Java
- Verifying the absence of array bounds errors in real codebases (and finding bugs in the process!)