Why Semantic Analysis is better for Identifying Deadcode

Program analysis - 2023

Arianna Bianchi, Kajsa Aviaja Pedersen, Kári Sveinsson, Mariana Santos

Outline

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

Dead Code?

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

What Is Deadcode

Also known as Software bloat (Debloating)

Section of source code whose result is never used in any other computation

Can have impact on the whole program if used incorrectly, for eg. division by zero

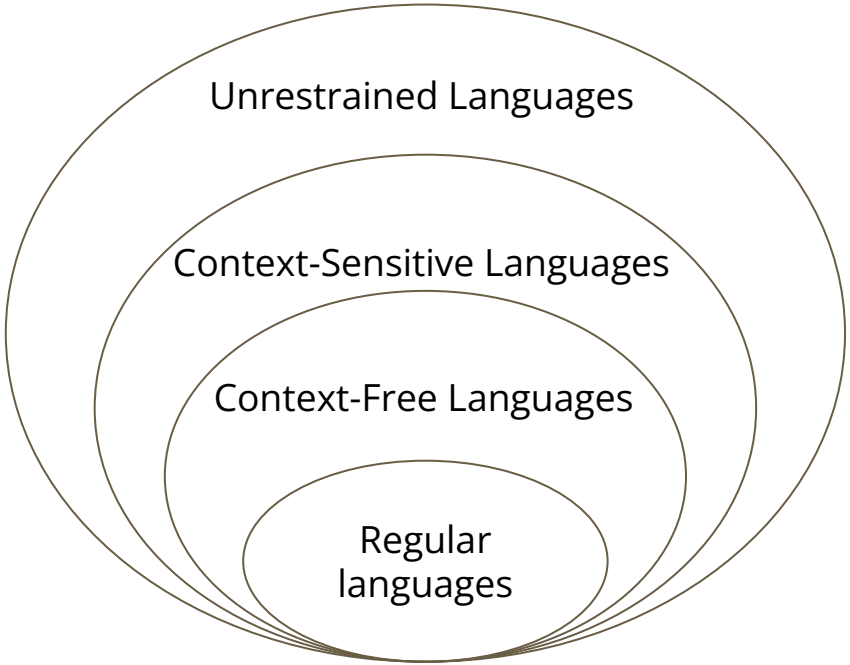
Can be hard to maintain and hard to read/understand

Our Syntactic Analysis

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

Syntactic Analysis

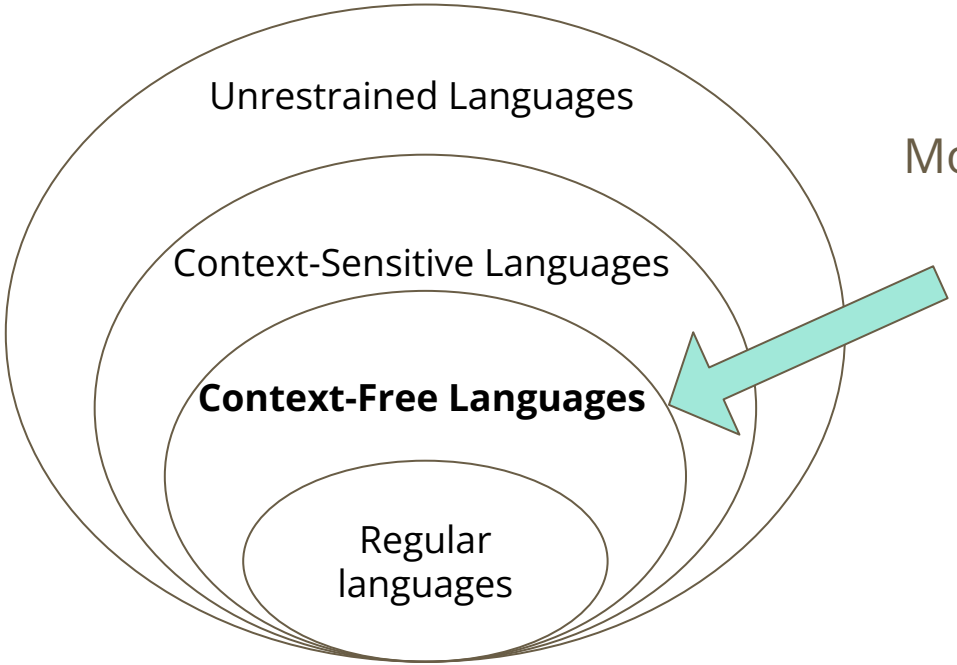
Chomsky Hierarchy: grammar divided into 4 types



Most programming languages are

Syntactic Analysis

Chomsky Hierarchy: grammar divided into 4 types



Most programming languages are

Syntactic Analysis - Parse Trees

Regular expressions

Can correctly only parse regular languages

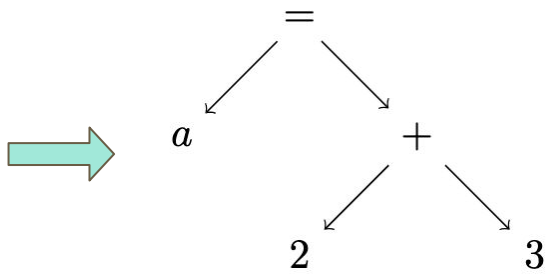
Used to match patterns

Context free Languages

Used to produce Abstract Syntax Trees

Checks the structure of the statement is correct

a = 2 + 3



Syntactic Analysis

We assume our AST is a **true** representation of the Java source code

Creating an **over-approximation** of function calls

Pre-generated parse tree tool that we used:

Tree Sitter

Our Abstractions

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

Java Bytecode

Simplifies the code

Parsing is easier - simpler grammar

Our Interpretation:

- Supports integers, booleans and references
- We only use a small subset of JVM operations
- Easier to abstract

Abstractions

Types:

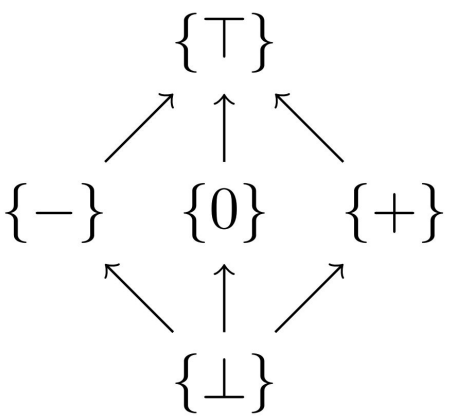
- Integers
- Booleans
- References

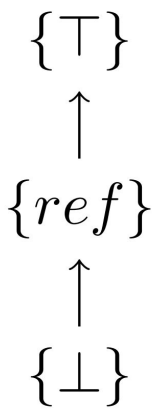
Sound abstraction / over-approximation

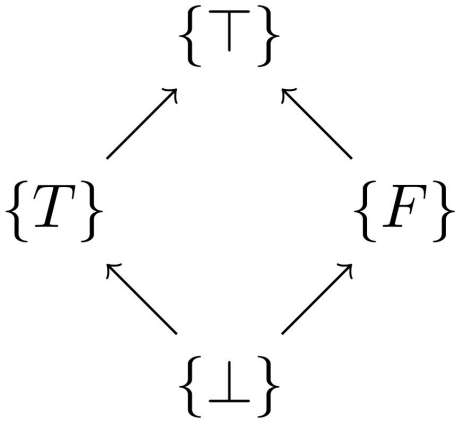
- Complete Lattices
- Finite height
- Monotone functions
- Galois Connections

Lattices

These are all complete lattices, with finite height







Monotone Functions

All abstract functions should be sound

- i.e. produce an overestimate.

So we use least fixed point so we ensure we have the most precise solution.

$$lfp(f) = \bigsqcup_{i \geq 0} f^i(\bot)$$

$\times$	$\top$	$+$	0	$-$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$
$+$	$\top$	$+$	0	$-$	$\bot$
0	$\top$	0	0	0	$\bot$
$-$	$\top$	$-$	0	$+$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$+$	$\top$	$+$	0	$-$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$
$+$	$\top$	$+$	$+$	$\top$	$\bot$
0	$\top$	$+$	0	$-$	$\bot$
$-$	$\top$	$\top$	$-$	$-$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Sign-Abstraction

Galois connections

1. $\forall x \in L_1 : x \sqsubseteq \gamma(\alpha(x))$
2. $\forall y \in L_2 : \alpha(\gamma(y)) \sqsubseteq y$

Ensures:

- Safe, but can lose precision
- Most precise abstraction - identity function

$$\alpha_{Int}(i) = \begin{cases} \bot, & \text{if } i = \emptyset \\ -, & \text{if } i \sqsubseteq \{-1, -2, -3, ...\} \\ 0, & \text{if } i = 0 \\ +, & \text{if } i \sqsubseteq \{1, 2, 3...\} \\ \top, & \text{otherwise} \end{cases}$$

$$\gamma_{Int}(s) = \begin{cases} \emptyset, & \text{if} s = \bot \\ \{-1, -2, -3...\}, & \text{if } s = - \\ 0, & \text{if } s = 0 \\ \{1, 2, 3...\}, & \text{if } s = + \\ \mathbb{Z}, & \text{otherwise} \end{cases}$$

Main Differences between semantic and static analysis

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

Syntactics vs. Semantics

Syntactic analysis:

- Only looks at the structure or grammar
- Cannot see what elements does, only what kind of element it is

Semantic analysis:

- Looks at the meaning or behaviour of the program
- Can see how the code acts

Evaluation

- What is deadcode?
- Syntactic analysis
- Abstract Interpretation
- Main Differences
- Evaluation

Evaluation

Accuracy

Performance

Complexity

Output Example

```
(dk.dtu.pa.payment.Order.calcTotal() , dk.dtu.pa.Util.sum(int,int))
```

Evaluation: Accuracy

- **Precision**
  How close the results are to the correct values
- **Recall**
  Rate of edges correctly detected
- **F-score**
  Harmonic mean of **precision** and **recall**

	Syntactic Analysis	Semantic Analysis
Precision	0.4231	0.6957
Recall	0.5789	1.0000
F-score	0.4889	0.8205

F-score increase of 68%!

Evaluation: Performance

- Run 50 times to get a time reading
- Run 100 times to get 100 results
- Compute mean and STD

Syntactic Analysis	Semantic Analysis
$6.897 \pm 0.369ms$	$8.778 \pm 0.405ms$

Semantic Analysis increase of 27%!

Evaluation: Complexity

How to measure?

- Syntactic Analysis:
  - Parse trees
  - More easily scalable to other language features

- Semantic Analysis:
  - *Jvm2Json* (bytecode represented in JSON)
  - Not as easy to extend to other features and operators

Thank you

Any Questions...?