Tempest

ACADEMY

Conference

2023

# Detecção de vulnerabilidades em código-fonte com IA

Classificação da Informação: PÚBLICA
Autor da apresentação: Maria Fernanda Rodrigues Vieira e Silva


01 Introdução

02 (Muito breve) Revisão sobre Machine Learning

03 Em busca dos bugs

04 Resultados

05 Trabalhos Futuros

Tempest

ACADEMY

Conference

# Introdução

# Introdução

## Detecção de Vulnerabilidades

- **Static**:  Se baseiam em técnicas de análise de código-fonte.
- **Dynamic**: Técnicas de análise de Software em Runtime.
- **Hybrid:** Utiliza técnicas estáticas e dinâmicas.

Tempest
ACADEMY
Conference

# Introdução

## Detecção de Vulnerabilidades

- **Static**: Se baseiam em técnicas de análise de código-fonte.
- **Dynamic**: Técnicas de análise de Software em Runtime.
- **Hybrid:** Utiliza técnicas estáticas e dinâmicas.

Tempest
ACADEMY
Conference

# Introdução

## Detecção de Vulnerabilidades Estática

- Detecção baseada em assinatura
- Detecção baseada em similaridade de código
- Detecção baseada em execução simbólica

Tempest
ACADEMY
Conference

# Introdução

## Detecção de Vulnerabilidades Estática

- Detecção baseada em assinatura
- Detecção baseada em similaridade de código
- Detecção baseada em execução simbólica

Tempest
ACADEMY
Conference

Tempest

ACADEMY

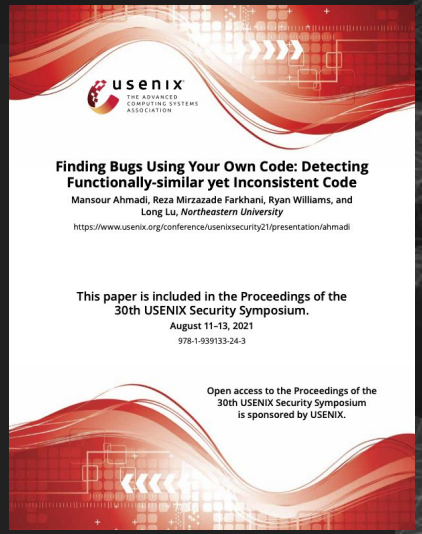
Conference

# Por que IA?


# Por que IA?

- Grande Potencial
- Excelentes resultados no processamento de Linguagem Natural
- Escalabilidade
- Abordagem inovadora

# Qual o estado da arte?




**Finding Bugs Using your Own Code: Detecting Functionally-similar yet Inconsistent Code**

Por:

**Mansour Ahmadi** (Northeastern University)

**Reza Mirzazade Farkhani** (Northeastern University)

**Ryan Williams** (Northeastern University)

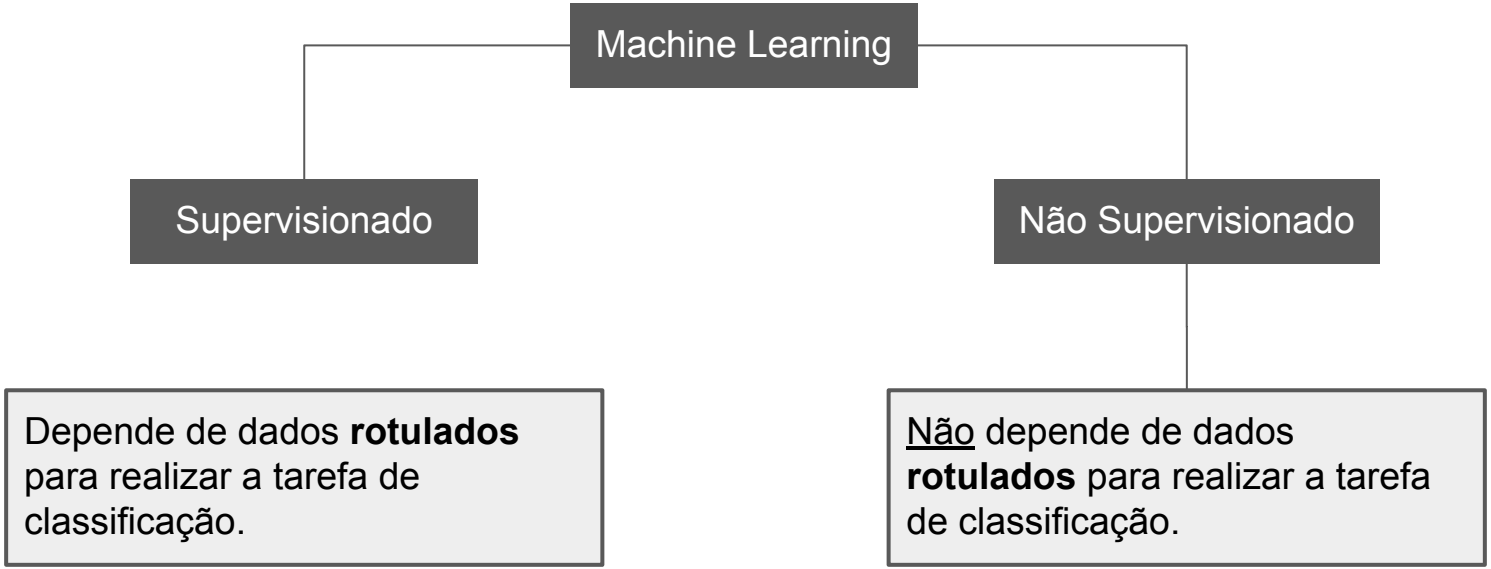
**Long Lu** (Northeastern University)

Tempest ACADEMY Conference


# (Muito breve) Revisão sobre Machine Learning

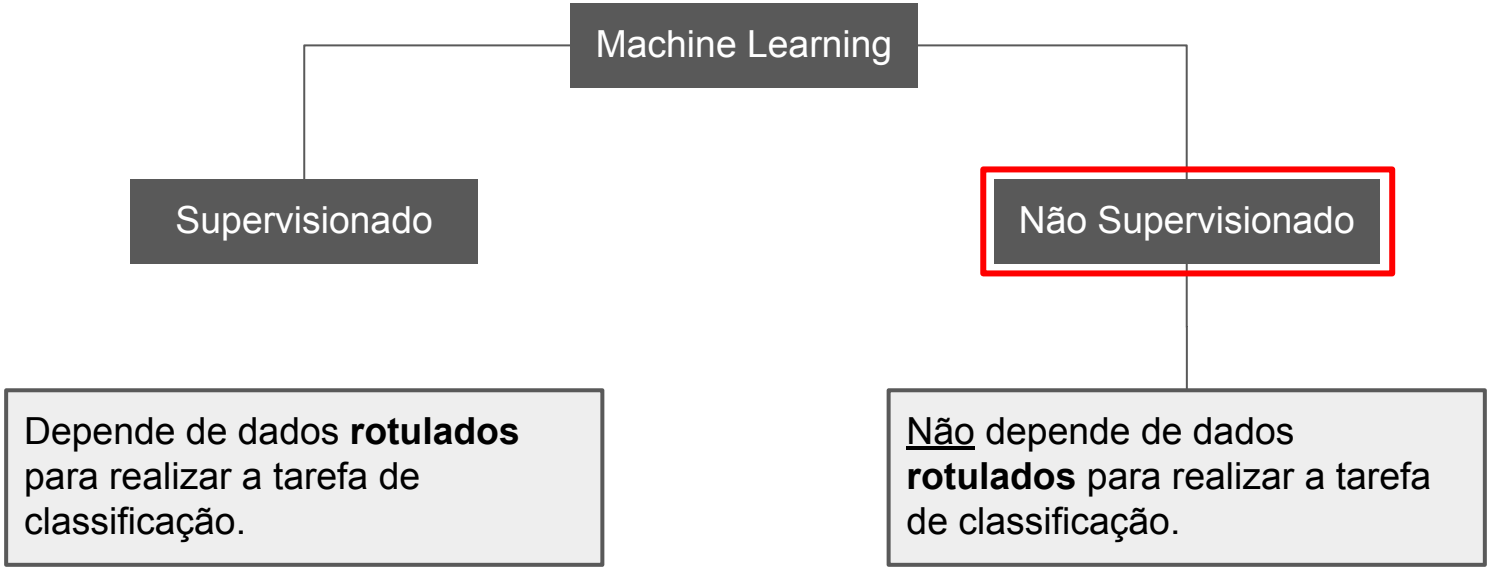
# Revisão sobre Machine Learning

Tempest
ACADEMY
Conference



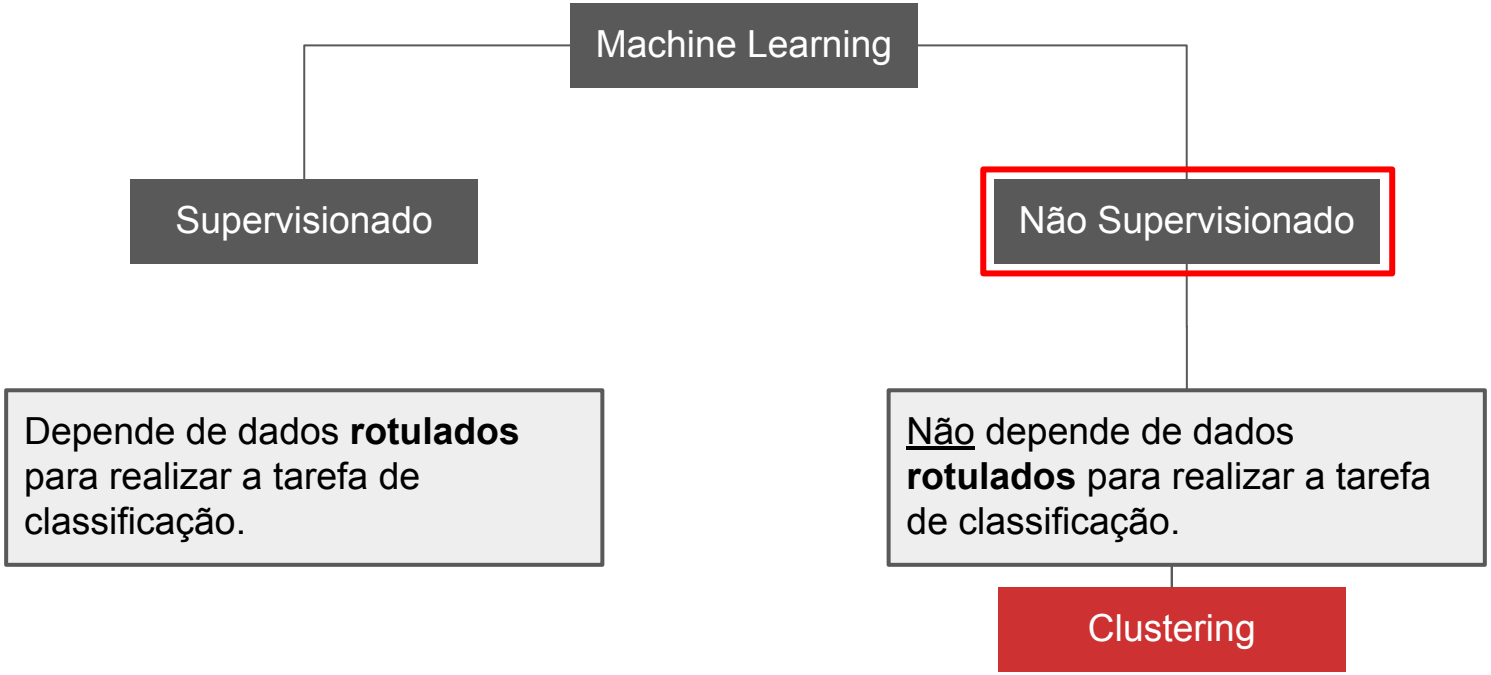
# Revisão sobre Machine Learning

Tempest ACADEMY Conference



# Revisão sobre Machine Learning

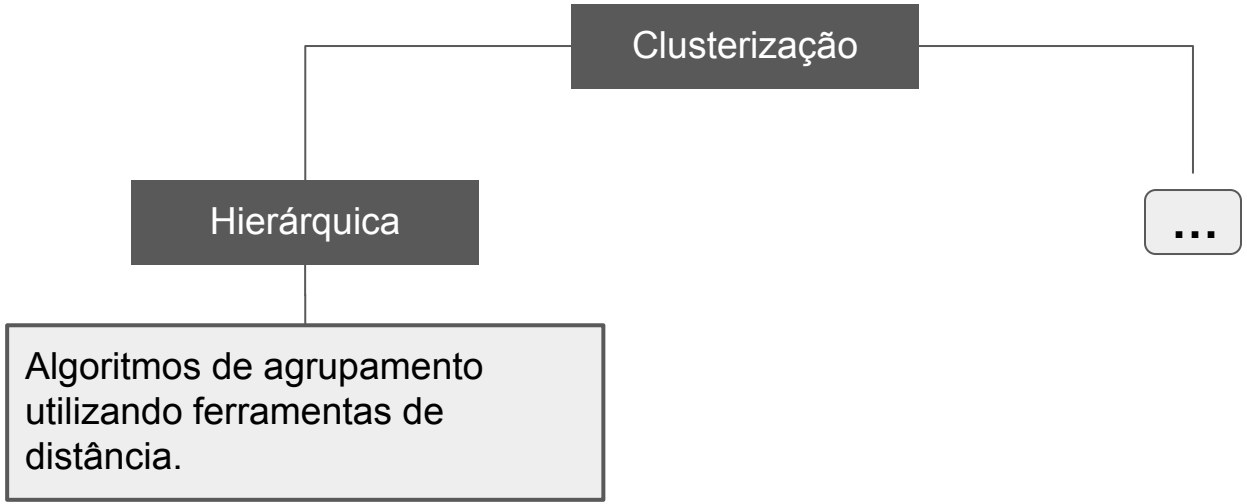
Tempest ACADEMY Conference



# Clustering Não Supervisionado

Tempest
ACADEMY
Conference

Algoritmos que buscam identificar padrões em conjunto de dados não rotulados.



Tempest
[ACADEMY]
Conference

**Logo, a abordagem utilizada no FICS precisa apenas do próprio código-fonte do projeto para detectar vulnerabilidades.**

Tempest
[ACADEMY]
Conference

# Vejamos o modelo...


# Em busca dos bugs

# FICS

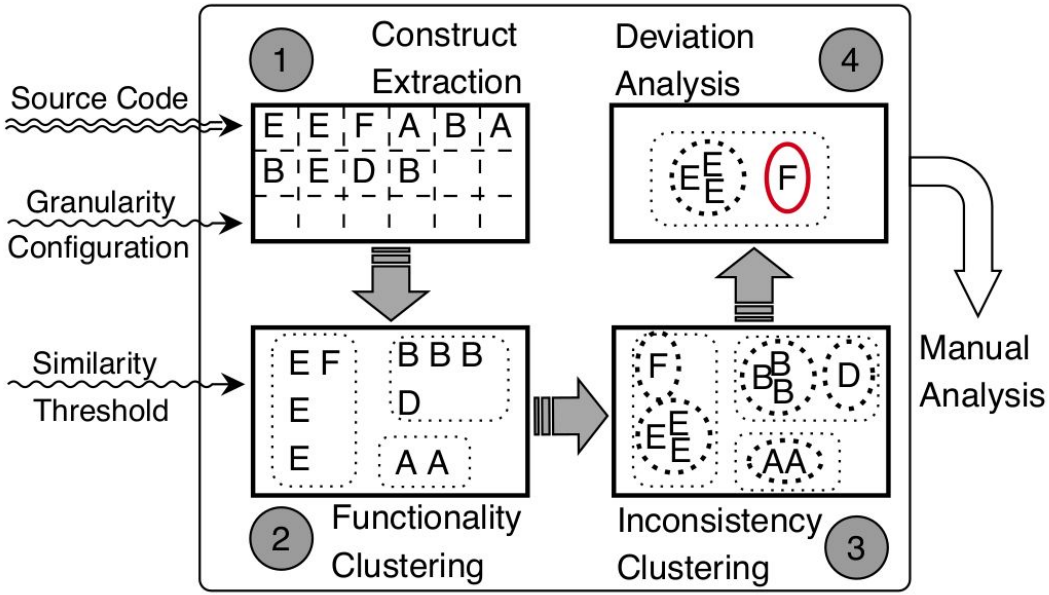
## Functionally-similar yet Inconsistent Code

Tempest

ACADEMY

Conference

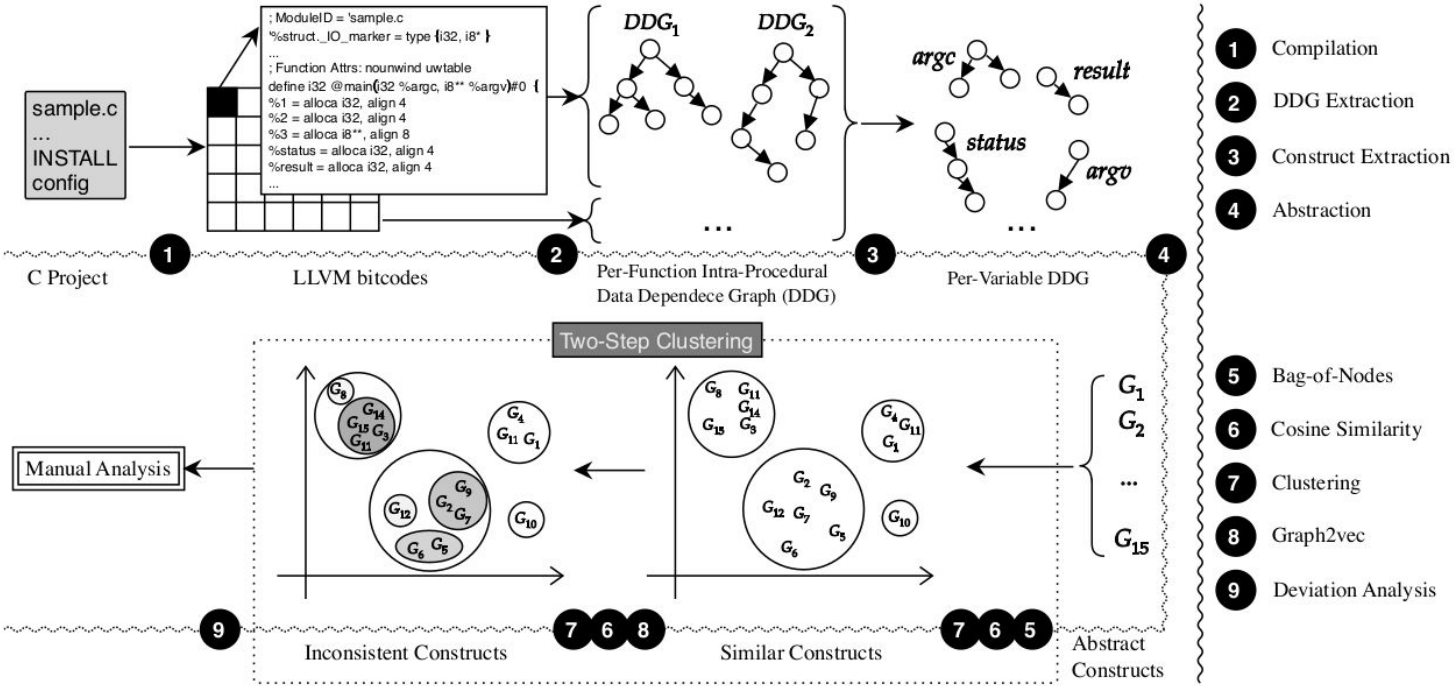
# Visão Macro do Modelo



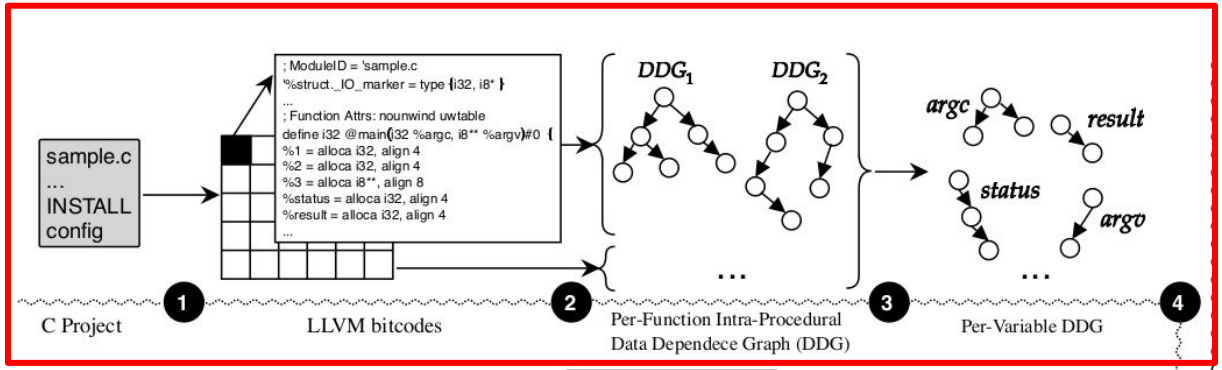
Tempest
[ACADEMY]
Conference

# Vamos detalhar mais a imagem...

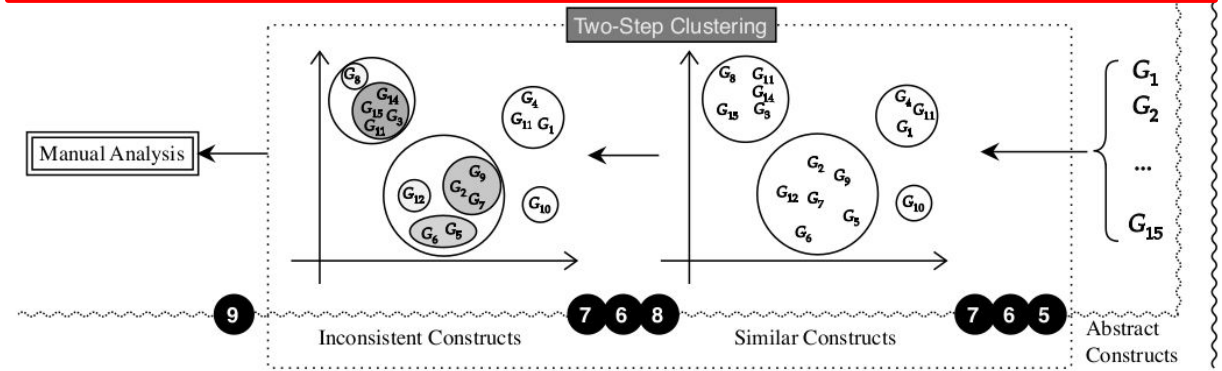
# Passo a passo:



# Passo a passo:




1. Compilation
2. DDG Extraction
3. Construct Extraction
4. Abstraction

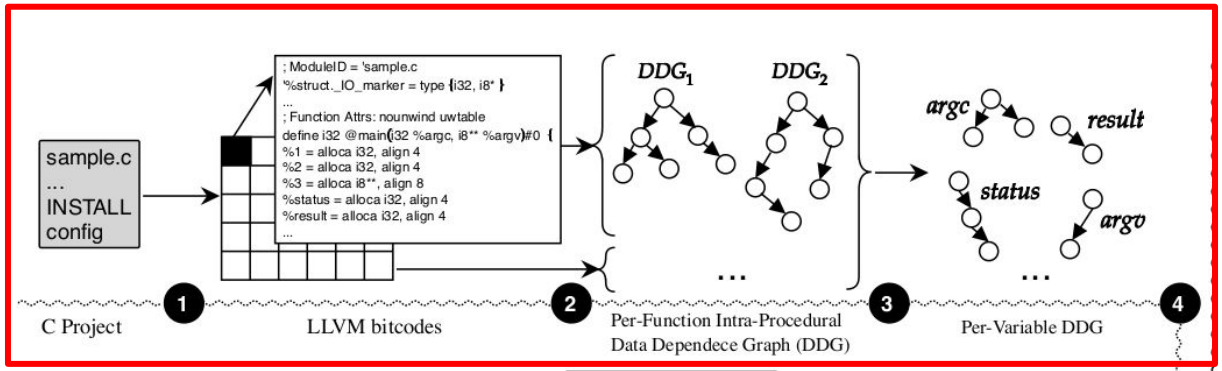



5. Bag-of-Nodes
6. Cosine Similarity
7. Clustering
8. Graph2vec
9. Deviation Analysis

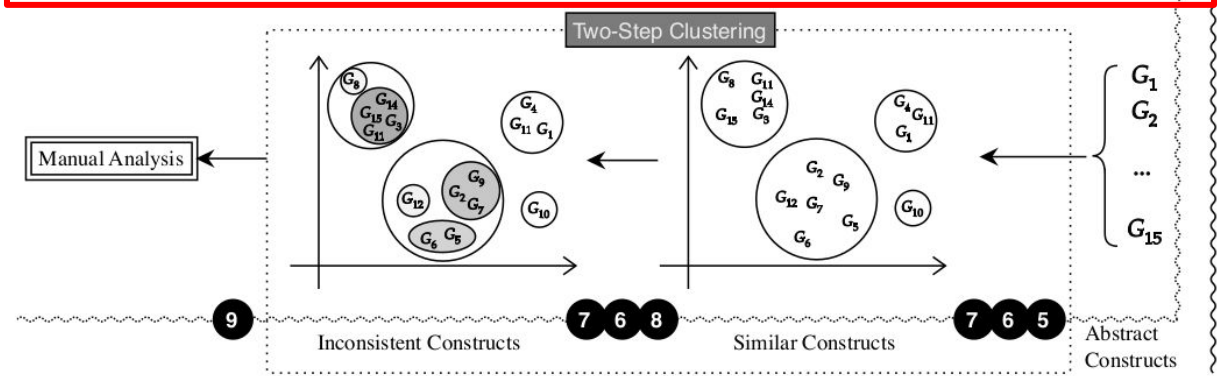
# Passo a passo:

Tempest ACADEMY Conference

**Parte 1**




1. Compilation
2. DDG Extraction
3. Construct Extraction
4. Abstraction

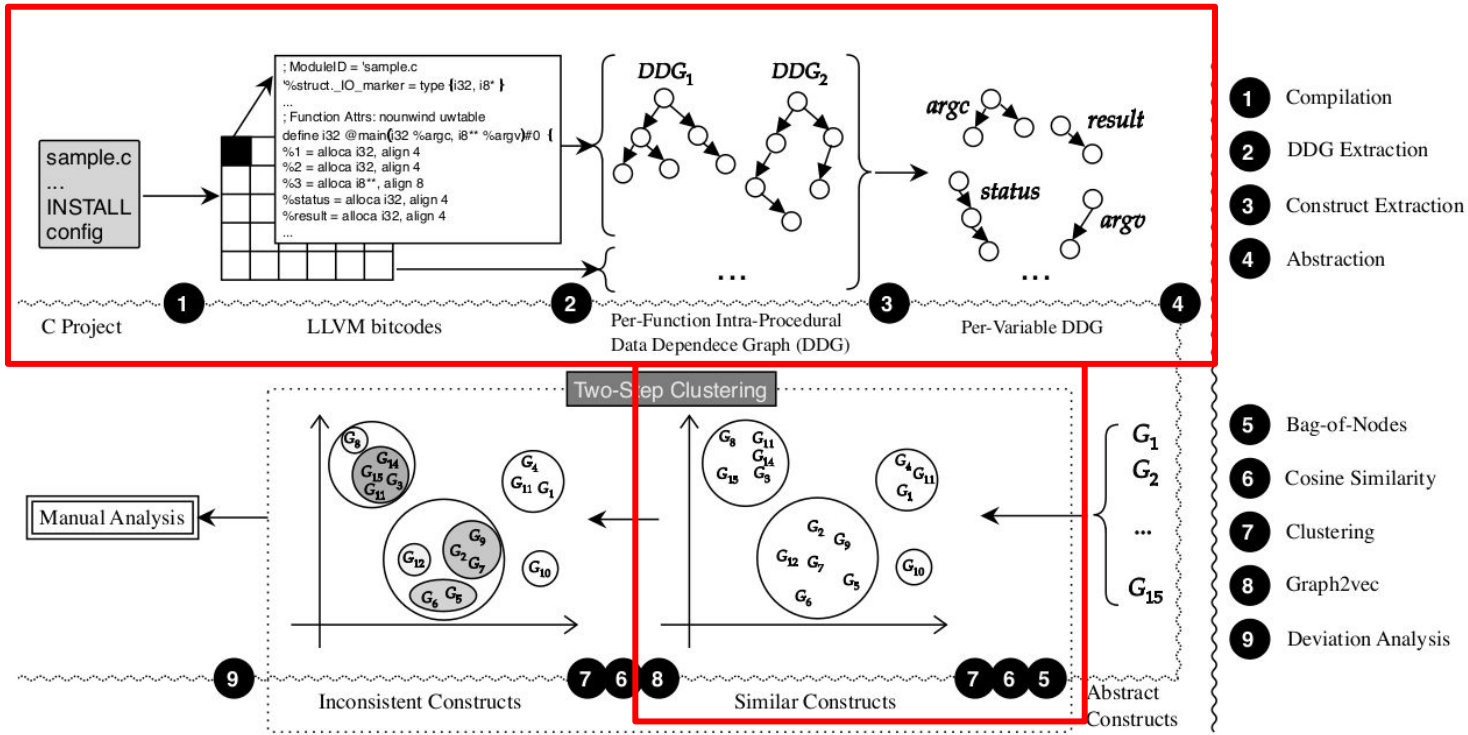



5. Bag-of-Nodes
6. Cosine Similarity
7. Clustering
8. Graph2vec
9. Deviation Analysis

# Passo a passo:

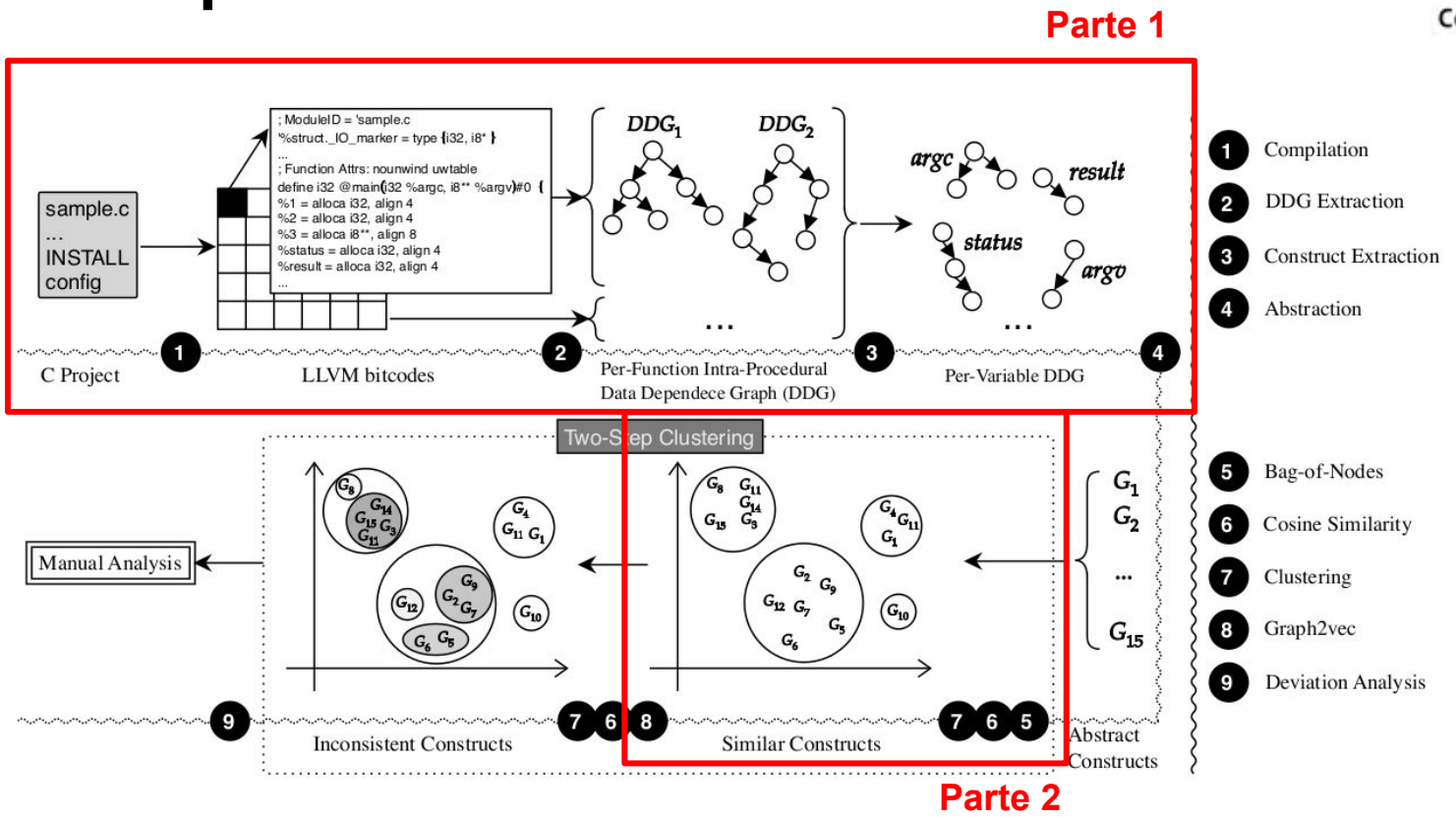
Tempest ACADEMY Conference

Parte 1



# Passo a passo:

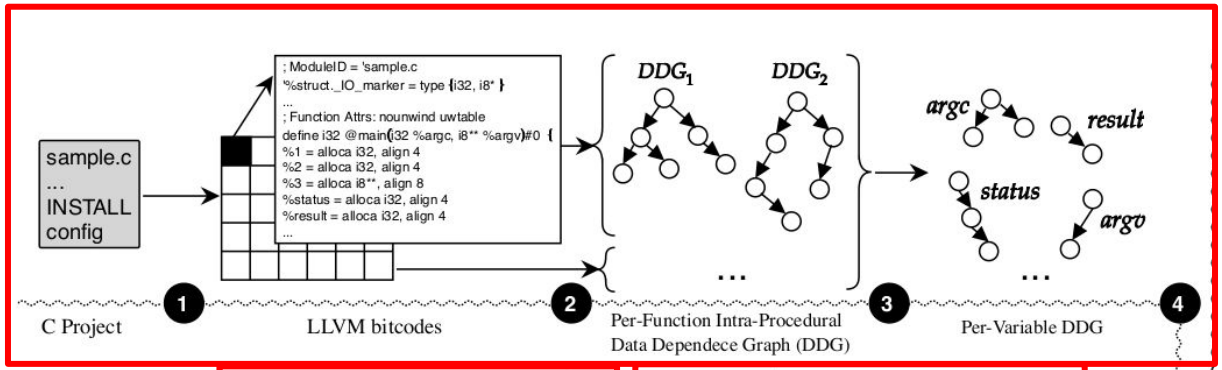
Tempest ACADEMY Conference

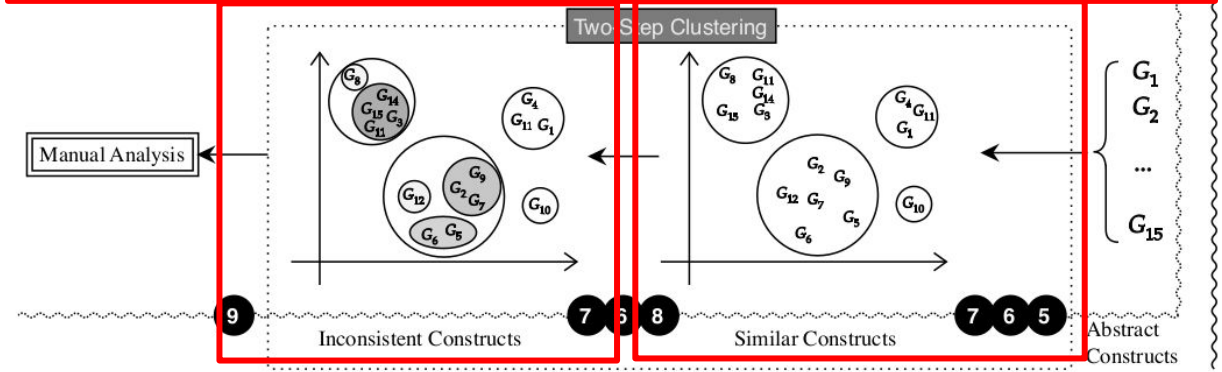


# Passo a passo:

Tempest ACADEMY Conference

Parte 1

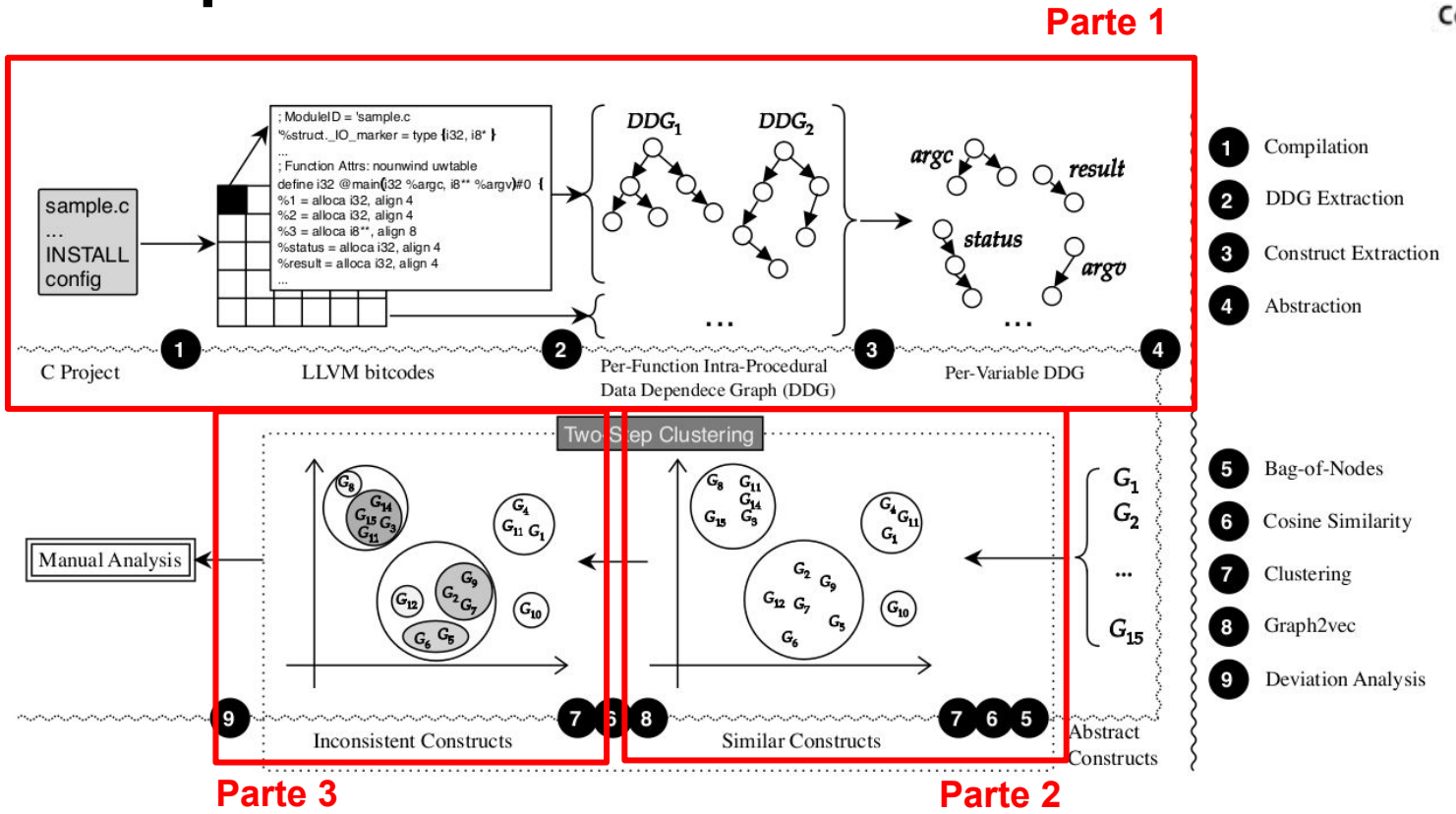



Parte 2

# Passo a passo:

Tempest ACADEMY Conference



# Parte 1

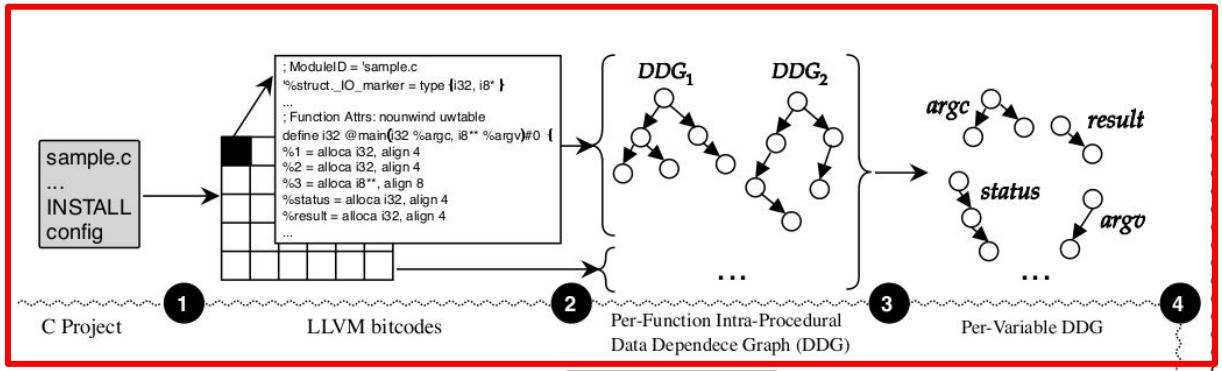
## Transformando Código em Vetores

Tempest
ACADEMY
Conference

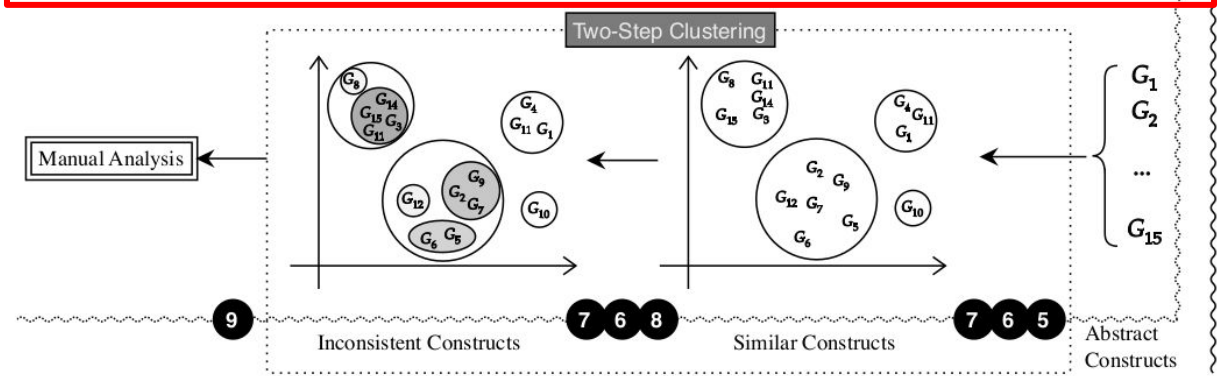
# Passo a passo:

Tempest ACADEMY Conference

**Parte 1**

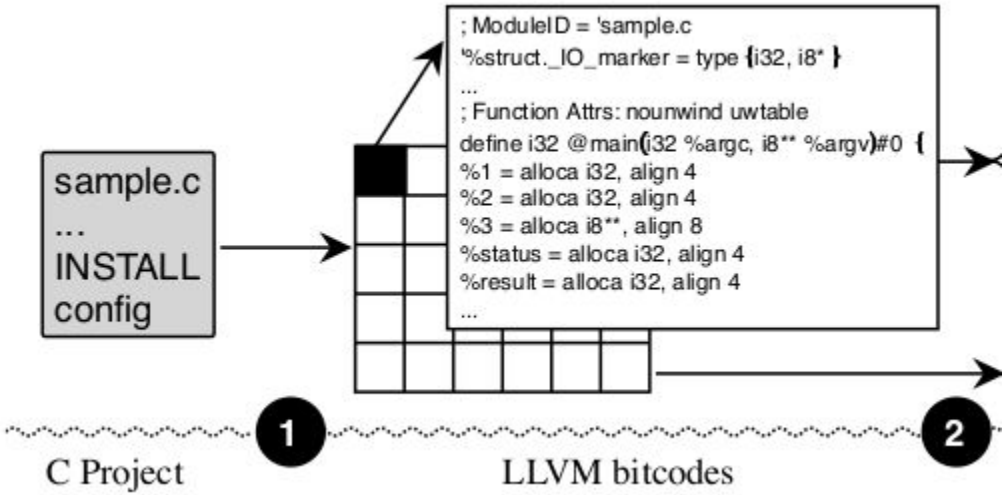



1. Compilation
2. DDG Extraction
3. Construct Extraction
4. Abstraction




5. Bag-of-Nodes
6. Cosine Similarity
7. Clustering
8. Graph2vec
9. Deviation Analysis

# Compilação

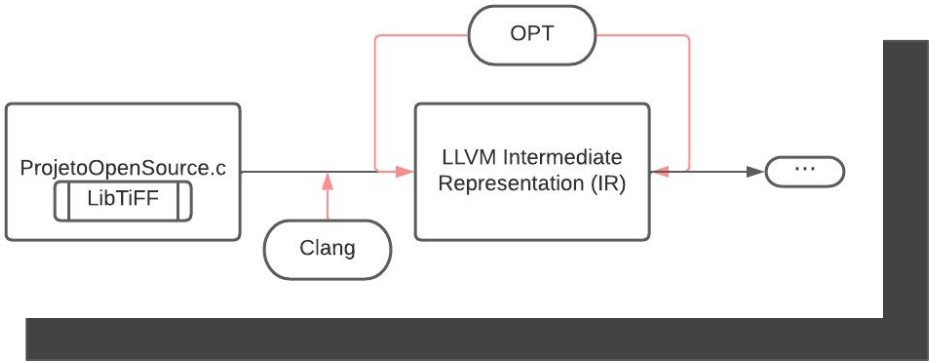


# Compilação do projetoOpenSource.c

Tempest
[ACADEMY]
Conference

**Ferramentas:**

- Clang: Tradutor de linguagem C para LLVM Bitcode (.bc) ou LLVM Assembly Language Format (.ll)
- Opt: Analisador e Otimizador de código fonte LLVM



# LLVM-IR

- Binary Operations
- Memory Operations
- Terminator Operations
- Cast Operations
- Other Operations

```c
#include <stdio.h>

int main(){
   int variavel=10;
   return variavel;
}
```

CLANG e OPT

```
define i32 @main() #0 !dbg !4 {
 %1 = alloca i32, align 4
 %variavel = alloca i32, align 4
 store i32 0, i32* %1, align 4
 call void @llvm.dbg.declare(metadata i32* %variavel, metadata !11, metadata !12),
!dbg !13
 store i32 10, i32* %variavel, align 4, !dbg !13
 %2 = load i32, i32* %variavel, align 4, !dbg !14
 ret i32 %2, !dbg !15
}
declare void @llvm.dbg.declare(metadata, metadata, metadata) #1

attributes #0 = { norecurse nounwind uwtable "disable-tail-calls"="false"
"less-precise-fp mad"="false" "no-frame-pointer-elim"="true"
"no-frame-pointer-elim-non-leaf""no-infs-fp-math"="false" "no-nans-fp-math"="false"
"stack-protector-buffer-size"="8" "target-cpu"="x86-64"
"target-features"="+fxsr,+mmx,+sse,+sse2" "unsafe-fp-math"="false"
"use-soft-float"="false" }
attributes #1 = { nounwind readnone }
```

LLVM Bitcode (.bc)

Documentação: https://llvm.org/docs/LangRef.html#instruction-reference

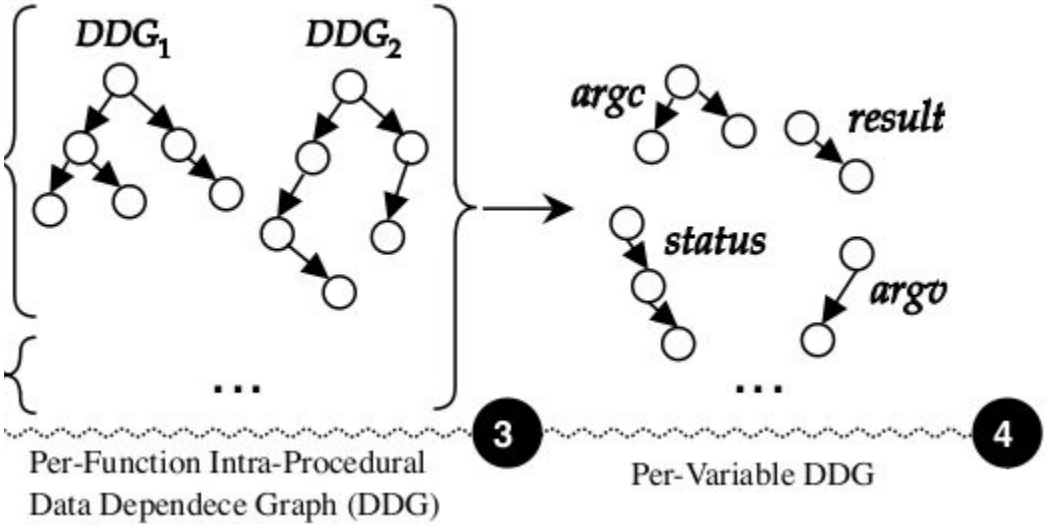
Tempest

ACADEMY

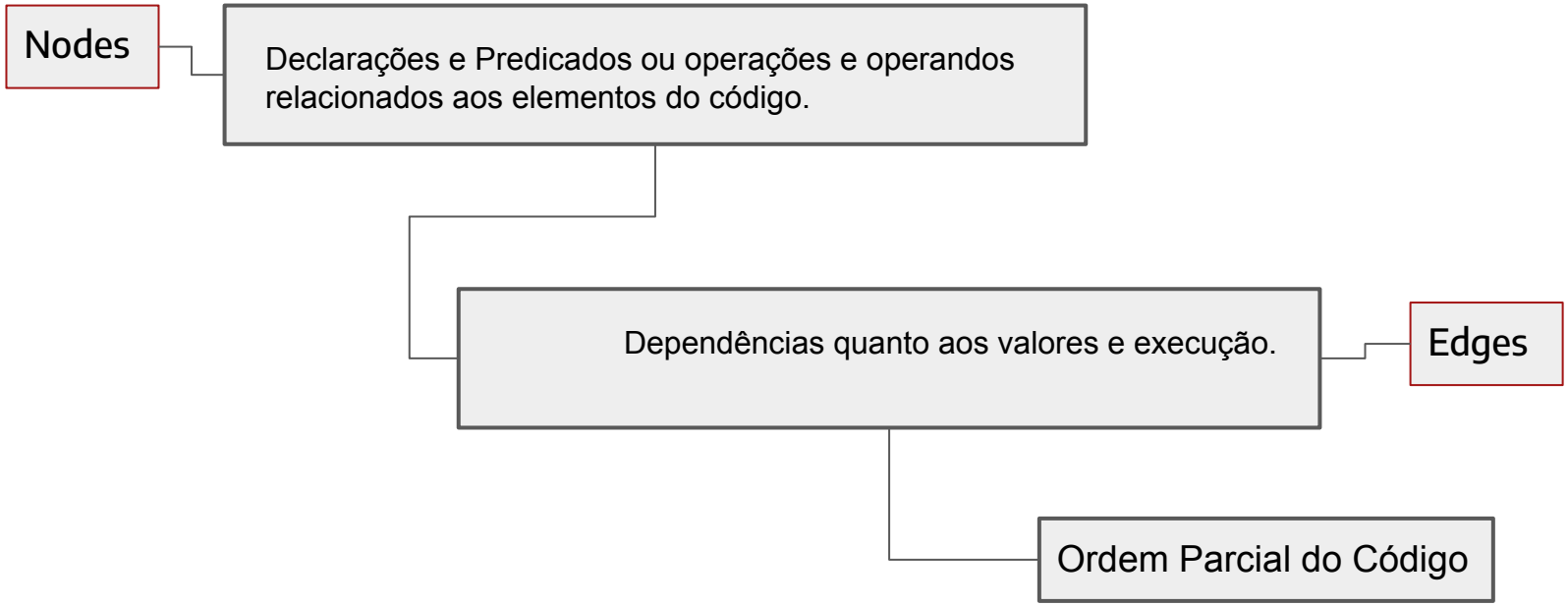
Conference

# Extração dos Data Dependence Graph e Formação dos Constructs

Tempest
ACADEMY
Conference

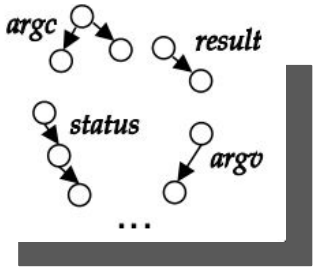


# Elementos



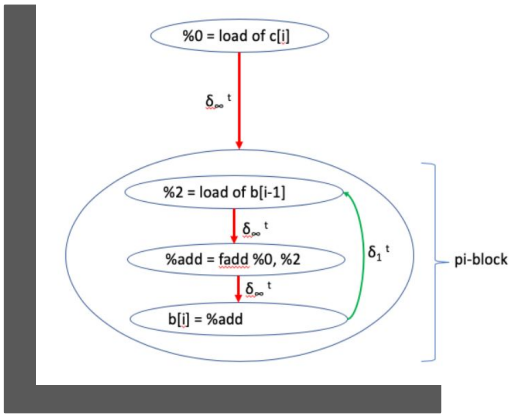
# Transformações dos Grafos

Extração das sub-rotinas do DDG, em todas as variáveis locais e globais




PDG — — — — — — — — — DDG — — — — — — — — — CONSTRUCT

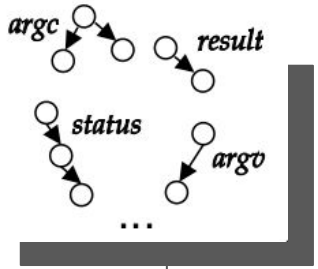
Omissão das dependências de controle do código




Tempest
ACADEMY
Conference

# Construct

Subgrafo do Data Dependence Graph. Contém todas as variáveis locais, parâmetros e variáveis globais de suas respectivas funções.




Através de um DDG, a extração começa de um Node específico e atravessa todos os subsequentes que dependem desse.

Definindo os constructs, o problema muda de:

**Detectar inconsistências no código**

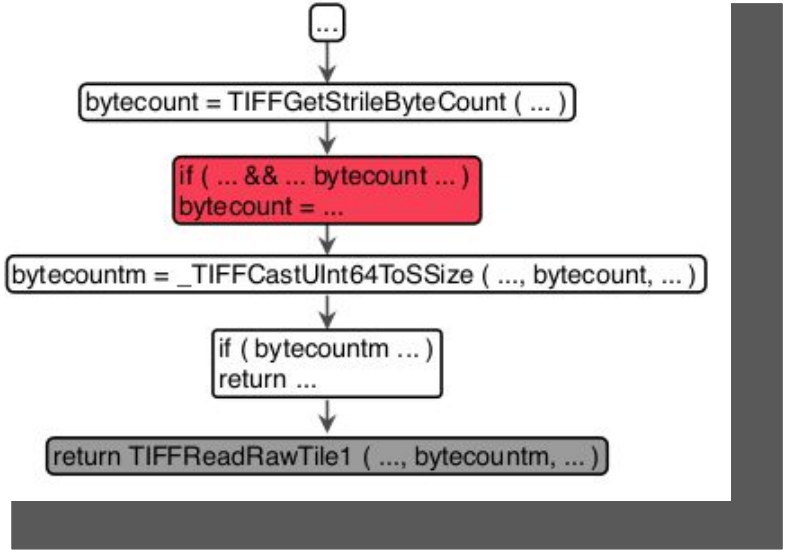
para:

**Detectar inconsistências em operações ou computações em variáveis.**

Tempest
ACADEMY
Conference

# Construct Extraction

Tempest
ACADEMY
Conference



# Overview

Tempest ACADEMY Conference

Program Dependence Graph (PDG)

Control Dependence

Inter-relação entre predicados e declarações.

```
if(predicado){

   declaração;

}
```

Data Dependence

Inter-relação entre valores e suas dependências.

```
A = B * C;

D = A + E;
```

Data Dependence Graph

Construct Abstraction

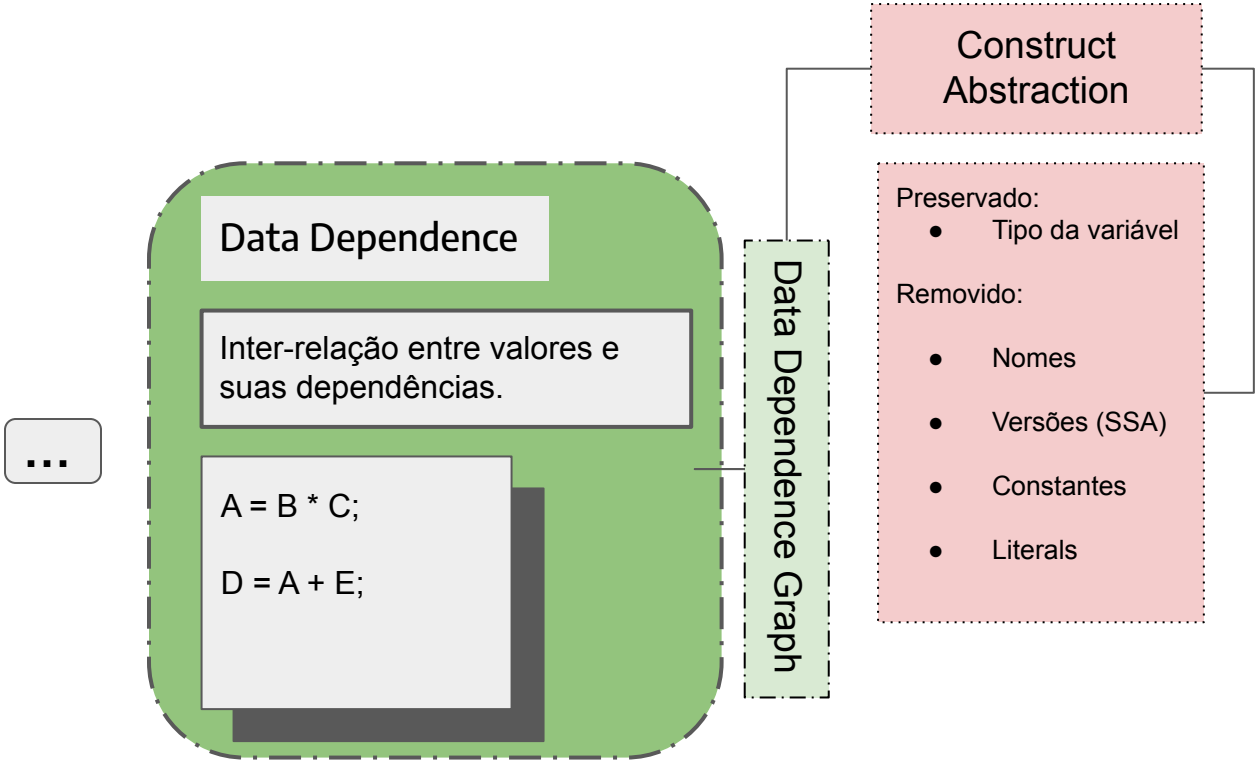
Preservado:

- Tipo da variável

Removido:

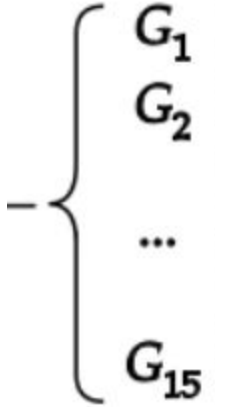
- Nomes
- Versões (SSA)
- Constantes
- Literals

# Overview

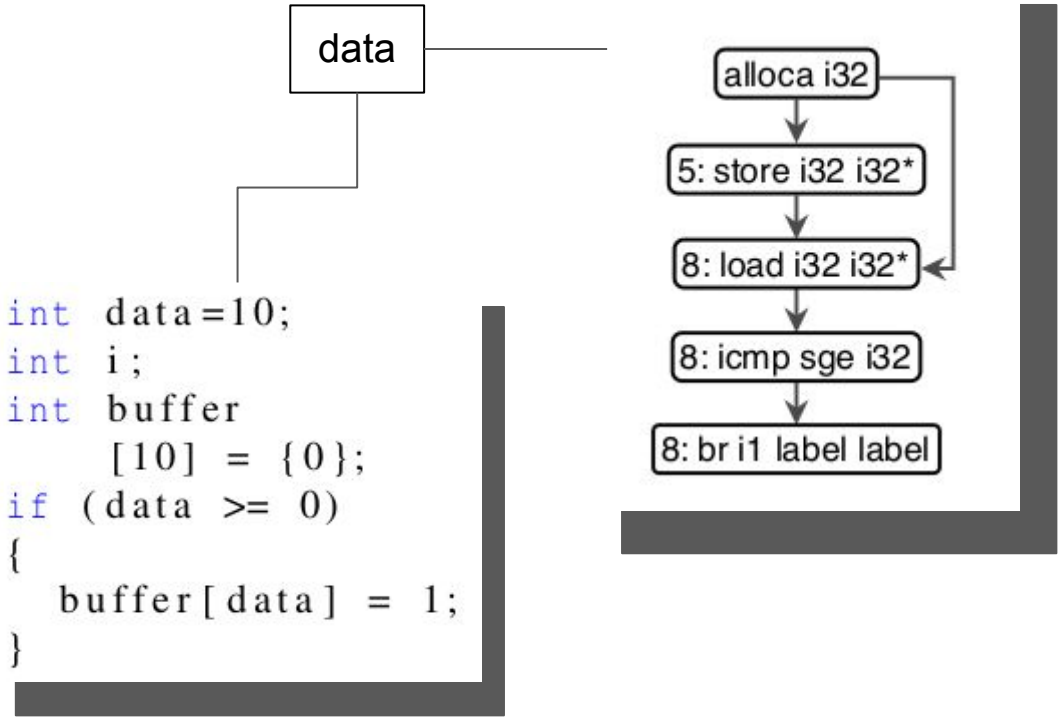


# Abstração do Construct

Tempest
[ACADEMY]
Conference




# Construct Abstraction




**alloca**: aloca memória na pilha de execução da função.

**store**: escreve na memória.

**load**: lê da memória.

**icmp**: retorna booleano ou um vetor de booleanos, baseado na comparação de inteiros.

**br**: controle de fluxo.

# Bag Of Nodes

Tempest
ACADEMY
Conference

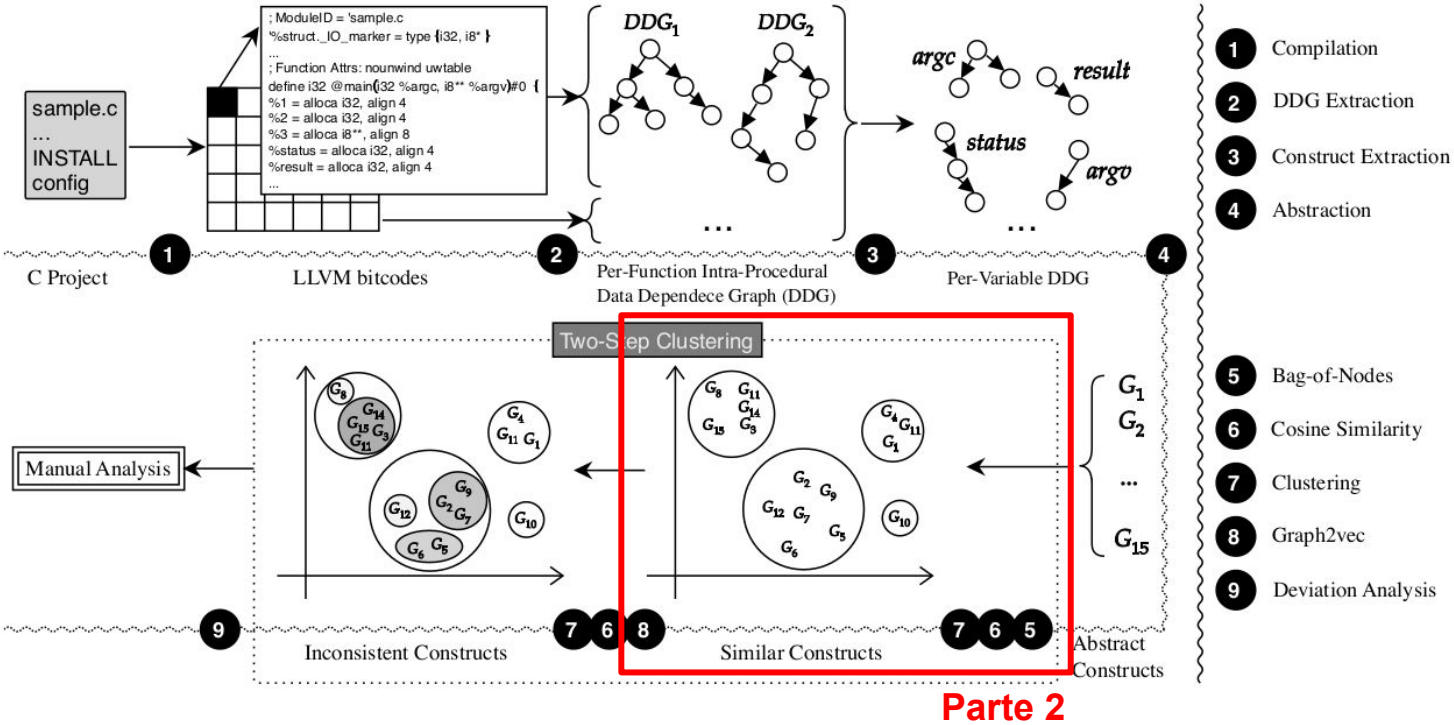
# Parte 2

## Primeira Clusterização

Tempest
ACADEMY
Conference

# Passo a passo:

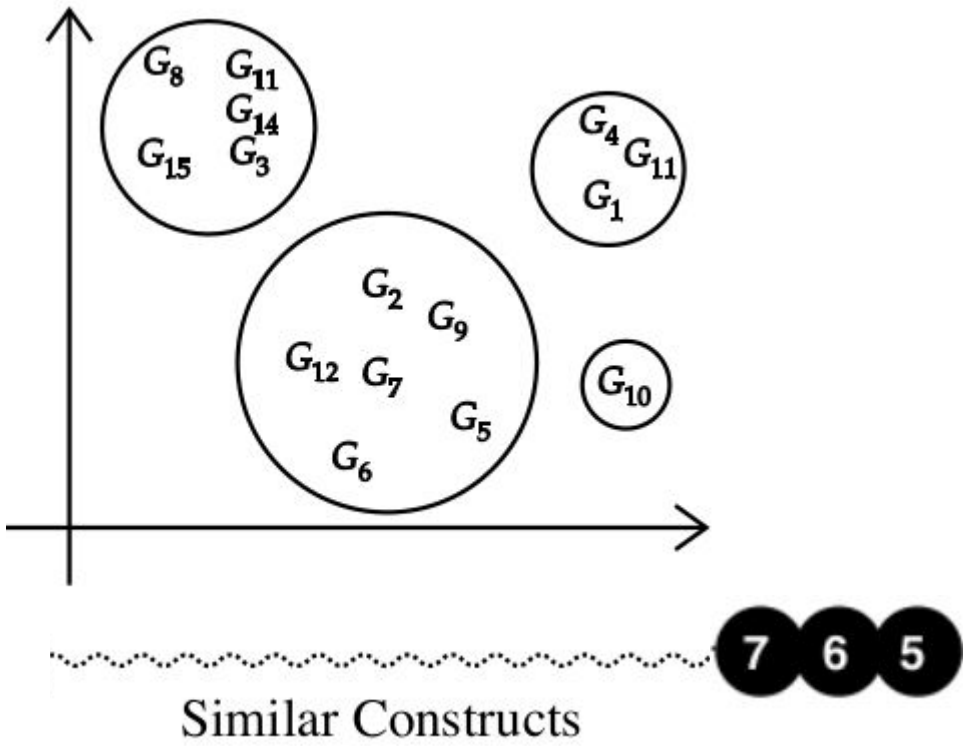
Tempest
ACADEMY
Conference



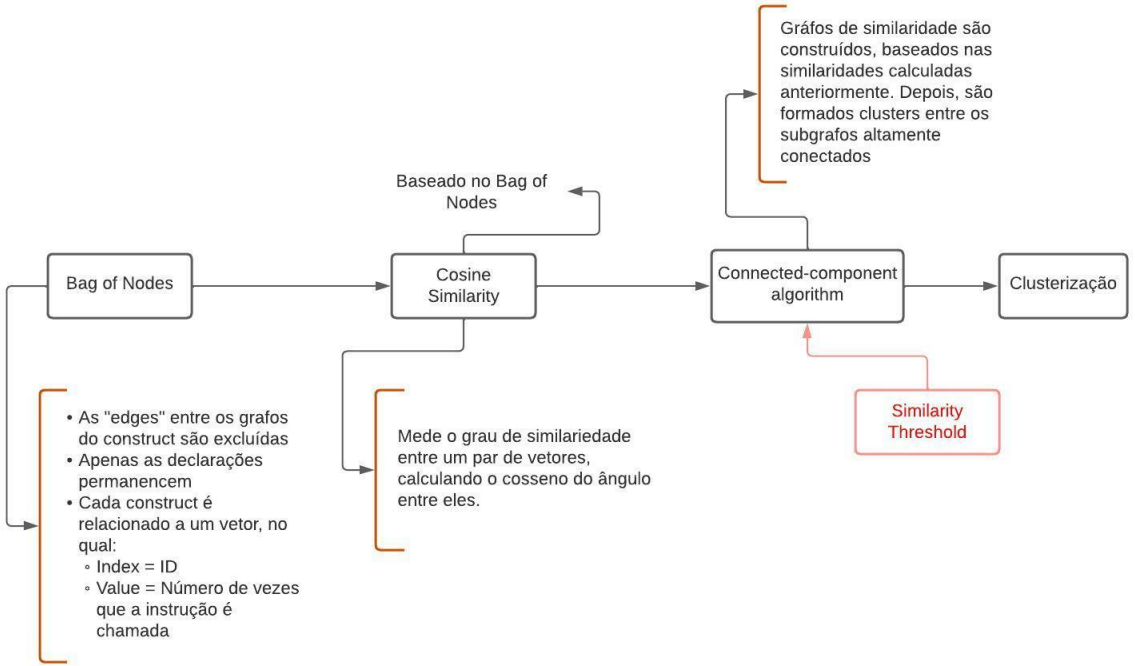

**Parte 2**

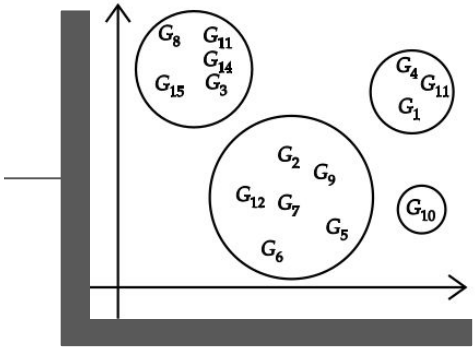
# Granulação Grossa

Tempest
ACADEMY
Conference



# 1° Clustering




Similarity Threshold

range 0,8 - 0.95

↓ Menos clusters, porém maiores

↑ Mais clusters, porém menores

# Bag of Nodes - Cosine - Connected-Component

Tempest
[ACADEMY]
Conference

construct_1 = | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | ...

construct_2 = | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ...

```
from scipy.spatial.distance import cosine

cosine_distance  =
distance.cosine(construct_1,construct_2)
print(cosine_distance)
```

Valor da distância dos Constructs

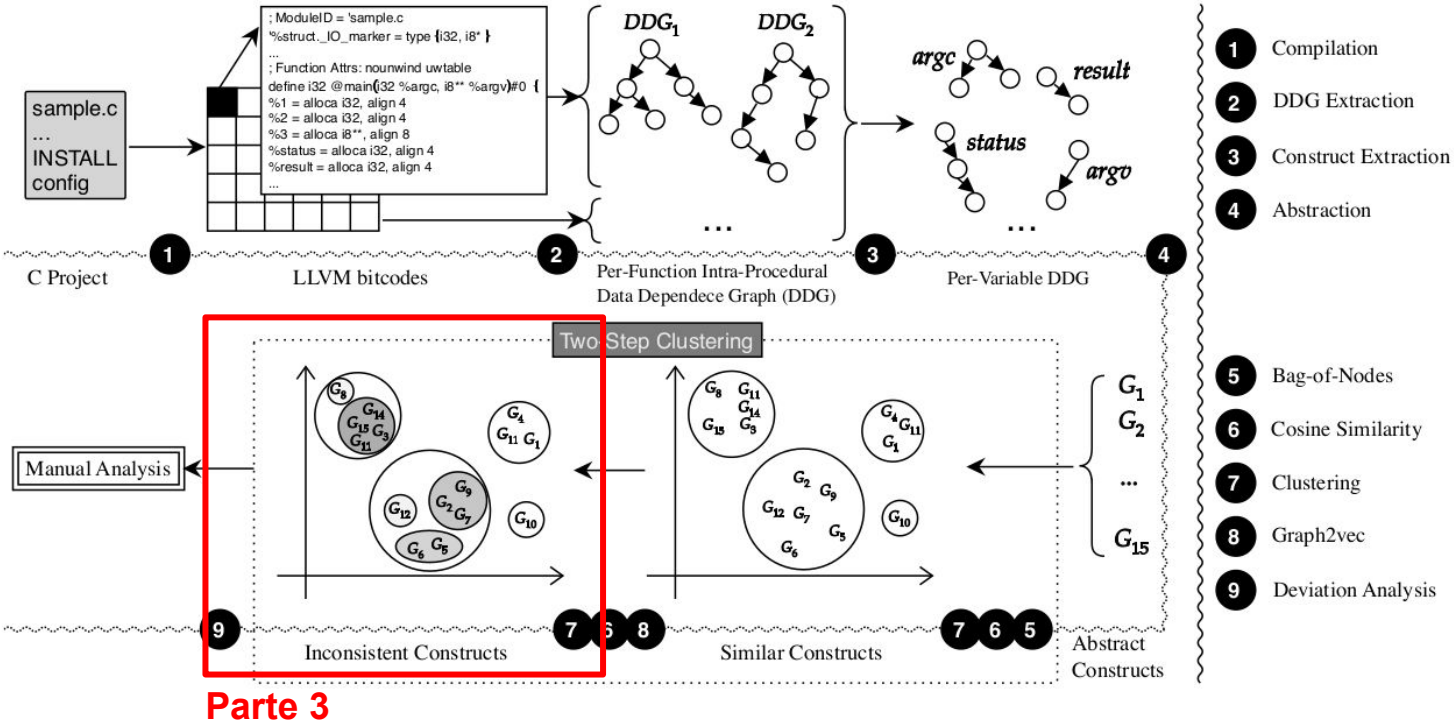
connected_components(array ou sparse matrix, connection=”strong”)

# Parte 3

## Segunda  Clusterização

Tempest
ACADEMY
Conference

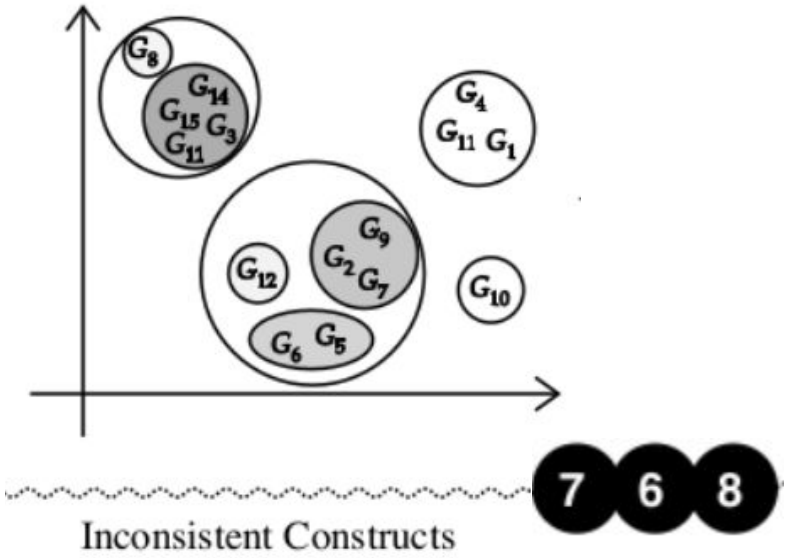
# Passo a passo:



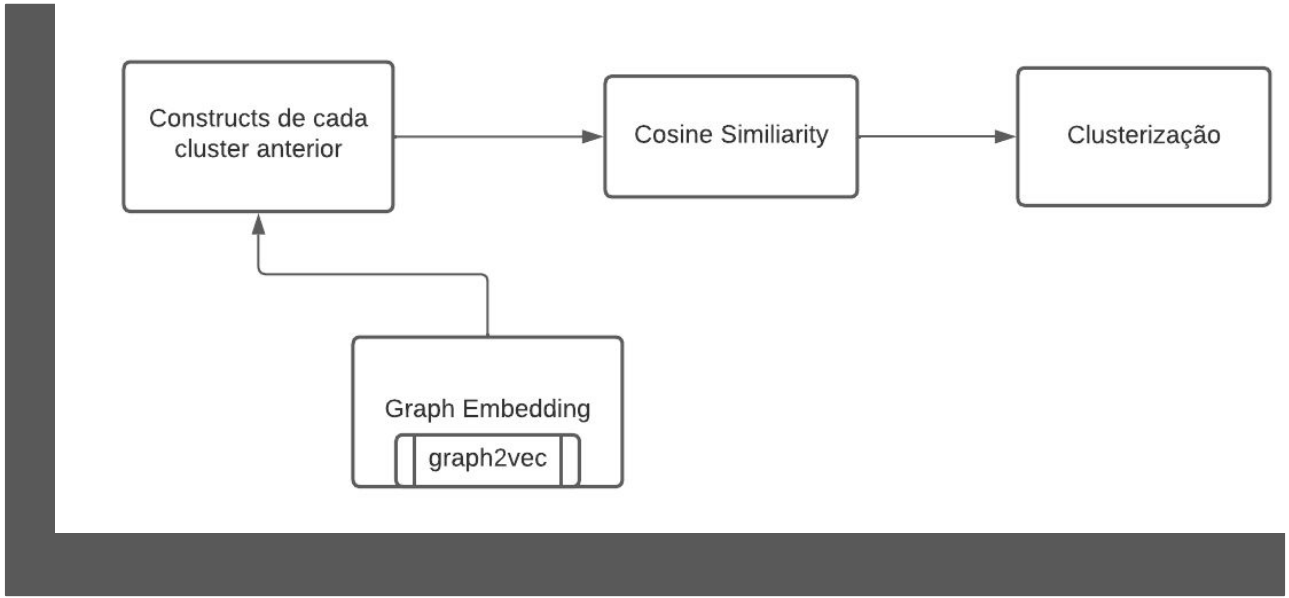

**Parte 3**

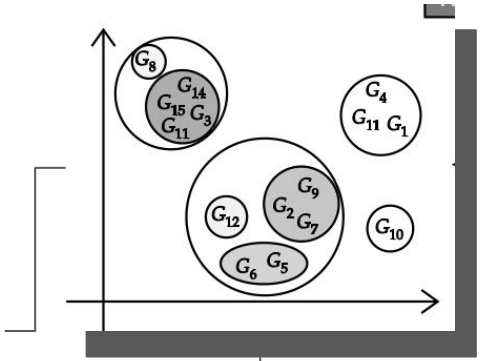
# Granulação Fina

Tempest
[ACADEMY]
Conference



# 2° Clustering

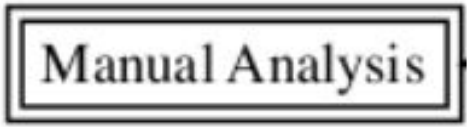



Análise Manual

# Análise Manual

Tempest
ACADEMY
Conference




Tempest

ACADEMY

Conference

# Resultados

# Resultados Obtidos

Os pesquisadores ao final encontraram um total de **22** vulnerabilidades que tiveram CVE assinado em **5** projetos diferentes.

Tempest
ACADEMY
Conference

## Resultados Obtidos

- LibTiff: **5** bugs, **93** inconsistências.
- OpenSSH: **1** bugs, **121** inconsistências.
- OpenSSL: **9** bugs, **310** inconsistências.
- QEMU: **4** bugs, **1206** inconsistências.
- WolfSSL: **3** bugs, **91** inconsistências.

Tempest
ACADEMY
Conference

# Lista de Bugs Encontrados

Tempest
ACADEMY
Conference

| Codebase | OpenSSL |
|---|---|
| Missing check | Report/Patch |
| Missing check | Patch |
| Wrong use of clear_free | Report/Patch |
| Null dereference | Report/Patch |
| Null dereference | Report/Patch |
| Inconsistent Check | Report/Patch |
| Memory Leak | Report/Patch |
| Missing clear_free | Report/Patch |

| Codebase | QEMU |
|---|---|
| 2 Missing checks | Report/Patch |
| Undefined Behaviour | Report/Patch |
| Uninitialized variable | Report/Patch |

| Codebase | LibTIFF |
|---|---|
| Missing checks | Patch |
| Mislocated check - Bad casting | Report/Patch |
| Missing TIFFClose | Report/Patch |

# Lista de Bugs Encontrados

Tempest
[ACADEMY]
Conference

| | |
|---|---|
| Codebase | wolfSSL |
| Missing check | Report/Patch |
| Missing check | Report/Patch |
| Memory exhaustion | Report/Patch |
| Codebase | OpenSSH |
| Missing bzero | Patch |

Código: https://github.com/RiS3-Lab/FICS


# Trabalhos Futuros

# Limitações

- Funciona apenas em projetos escritos em C
- O projeto não pode ser muito pequeno
- O projeto não pode ser muito grande
- Alguns grupos de vulnerabilidades não são detectados pelo modelo a depender do projeto
- Não faz uso de Transfer Knowledge
- Vulnerabilidades em nível de statements não são detectadas

Tempest
ACADEMY
Conference

# Propostas de trabalhos

- Avaliar a utilização de outras estruturas de representação de código
- Avaliar outras formas de representação do código-fonte
- Avaliar a possibilidade de utilização de Transfer Knowledge
- Re-implementar o modelo utilizando Python 3
- Executar o modelo em outros projetos
- Realizar um estudo comparativo com ferramentas tradicionais para detecção de vulnerabilidades
- Avaliar outros algoritmos de Clusterização
- Utilizar outros thresholds
- Portar o modelo para outras linguagens de programação

Tempest
ACADEMY
Conference

# Obrigada!!

Tempest
ACADEMY
Conference

## Artigo no Sidechannel:

## LinkedIn

Tempest
ACADEMY
Conference

# POC

Tempest
ACADEMY
Conference