Universidade do Minho
Escola de Engenharia

Análise Estática

e compreensão de programas

Tiago Baptista
&
Pedro Rangel Henriques

Conteúdo

- Control Flow Graphs **(CFG)**
- Data Dependency Graphs **(DDG)**
- System Dependency Graphs **(SDG)**

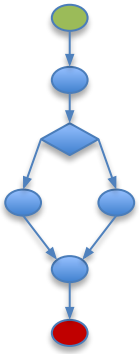
Control Flow Graphs - Definição

- Gráfico que modela uma função através da representação das estruturas de controlo.
  - Modela todos os possíveis caminhos de execução de uma função.

- CFG=(V,E)
  - V (vertices) guarda os elementos básicos das funções.
    - Simplificação: Cada vértice representa um statement.
  - E (edges/arestas) conexões entre os statements, chamados the flow paths (caminhos do fluxo).

- Cada estrutura de controlo tem gráficos com uma estrutura pré-definida e constante.

Control Flow Graphs - Exemplos

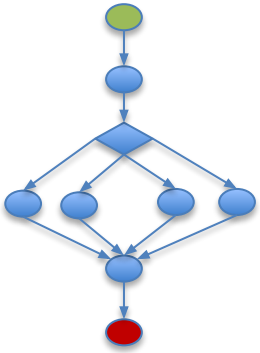


if...else

if
…

while/for

do…whil
e



switch…cas
e

Control Flow Graphs - Aplicações

- Reconhecer todos os possíveis caminhos de execução de uma função/programa :
  - Determinar a complexidade de uma função :
    - McCabe's complexity = E – V + 2 (McCabe's Cyclomatic Complexity with Flow Graph (Example))
  - Detetar possíveis comportamentos indesejáveis.
  - Desenhar e calcular a cobertura de testes.
- Localizar código "morto"
  - Grafos de ilha, sem qualquer ligação a outros pontos.
- Entendimento do código através da observação do seu fluxo geral.

Data Dependency Graphs - Definição

- Gráfico que modela uma função através da representação das dependências entre dados
- DFG=(V,E)
  - V (vertices) guardam dados (variáveis, constantes, operadores, etc)
  - E (edges/arestas) definem a dependência entre os dados.
- Tem como base o formato de código de **atribuição única (single assignment).**

Data Dependency Graphs - Exemplo

Código original:

```
x = 5;
x = x - 3;
if(x<3) {
   y = x + 2;
   w = y;
} else {
   y = x – 3;
}
w = x - y;
z = x + y;
```

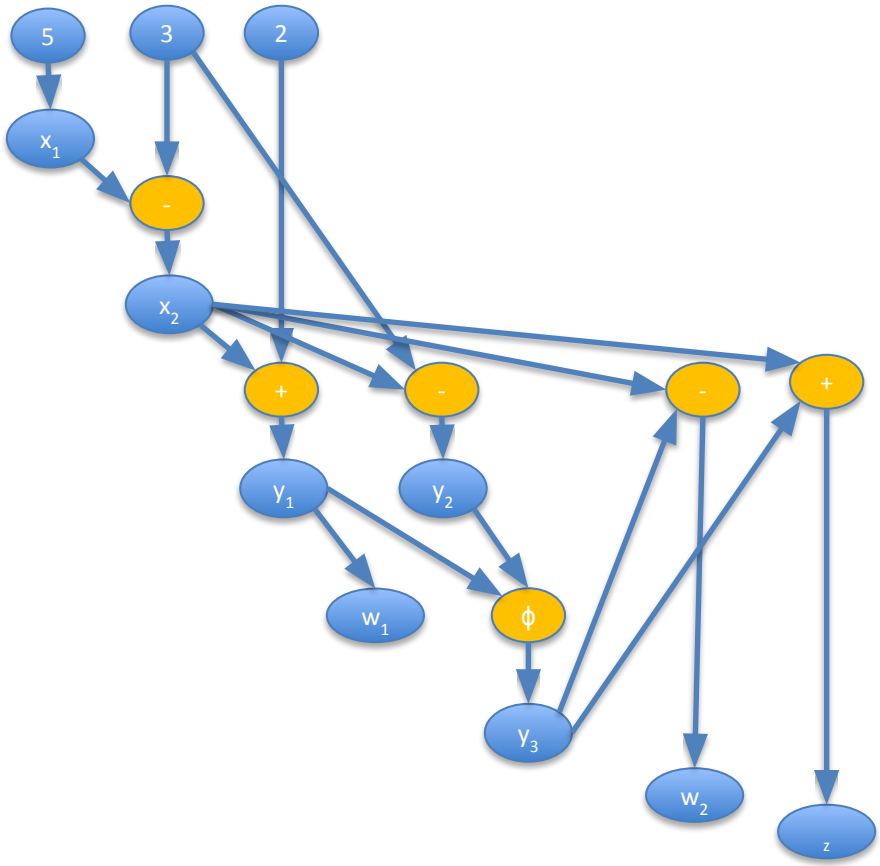
Formato single assignment:

```
x1 = 5;
x2 = x1 - 3;
if ( x2 < 3 ) {
   y1 = x2 + 2;
   w1 = y1;
} else {
   y2 = x2 – 3;
}
y3 = ϕ(y1,y2);
w2 = x2 – y3;
z = x2 + y3;
```

Data Dependency Graphs - Exemplo

Formato single assignment:

```
x1 = 5;
x2 = x1 - 3;
if ( x2 < 3 ) {
   y1 = x2 + 2;
   w1 = y1;
} else {
   y2 = x2 – 3;
}
y3 = ϕ(y1,y2);
w2 = x2 – y3;
z = x2 + y3;
```



Data Dependency Graphs - Aplicações

- Identificação de dependência no código (entre variáveis e até funções)
  - Detectar o uso de variáveis e propriedades de liveness
- Detectar código morto
  - Variáveis não utilizadas
  - Grafos de ilha
- Importante na construção de compiladores
  - Detetar a ordem de execução
  - Gerir alocação de memória/registos

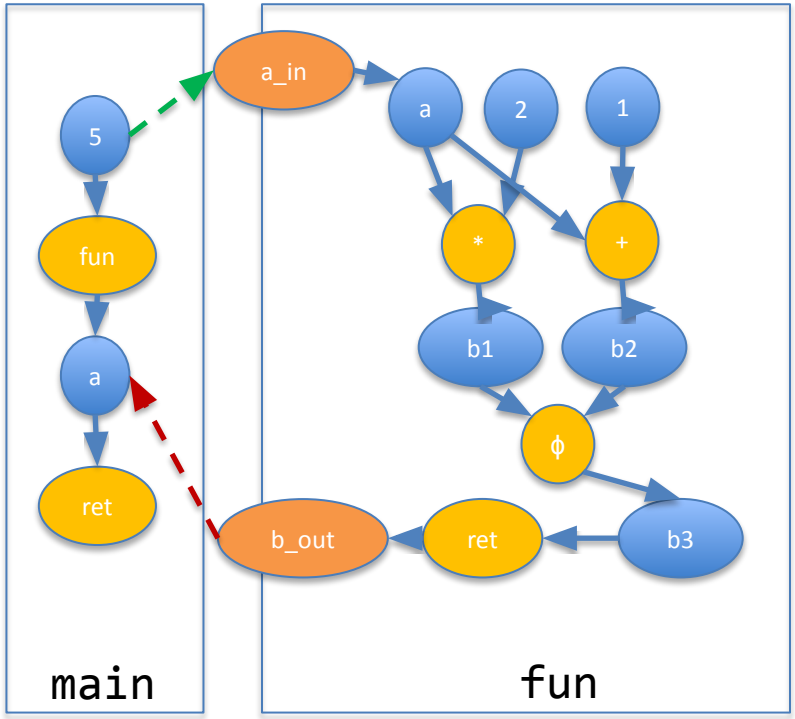
System Dependency Graphs - Definição

- Gráfico que modela um sistema (conjunto de funções) através da representação do fluxo e dependência de dados das funções que o constituem.
- SDG = (PV,V,E,IPE)
  - PV (procedure vertices) representa as funções/procedures;
  - V (vertices) representa os dados de cada função/procedure;
  - E (edges) representa as dependências de dados entre os vértices V;
  - IPE (inter-procedural edges) representa o fluxo de dados para os PV.

System Dependency Graphs - Exemplo(1)

```
main() {
 a = fun(5);
 return a;
}

fun(a) {
 b1 = a * 2;
 if(b < 10) {
  b2 = a + 1;
 }
 b3 = ϕ(b1,b2);
 return b3;
}
```

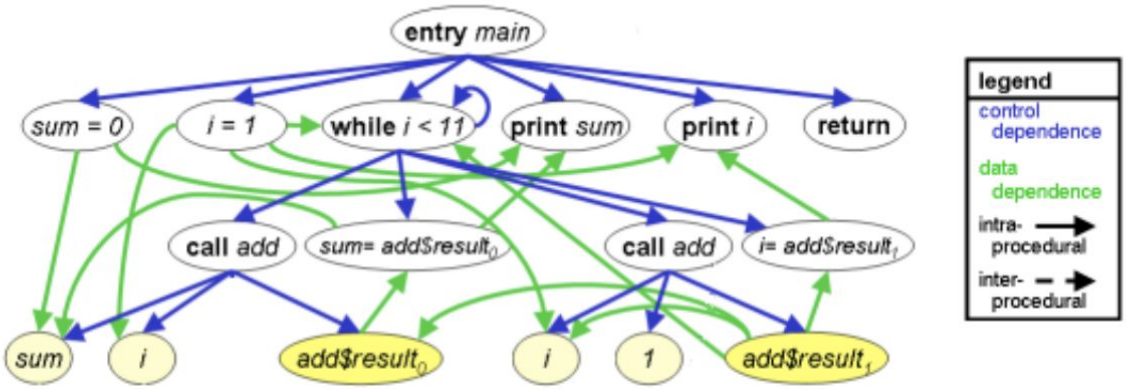


O controlo de fluxo não está a ser considerado (para simplificação)

System Dependency Graphs - Exemplo(2)

```
public static void main(String[] args) {
    int sum, i;
    sum = 0;
    i = 1;
    while (i<11) {
      sum = add(sum, i);
      i = add(i, 1);
    }
    System.out.println("sum = " + sum);
    System.out.println("i = " + i);
  }
```

Listing 2.3: Excerpt of a Java program



Fonte : https://epl.di.uminho.pt/~gepl/GEPL_DS/DariusSDG/files/MSc_NPereira_Dissertation.pdf

System Dependency Graphs - Aplicações

- Dão uma visão geral sobre um sistema
  - Controlo de fluxo
  - Dependência entre dados
- Análise completa de um sistema num só grafo
  - Análise de segurança (via taint analysis)
  - Procura de padrões (boas práticas no código/ linters)
  - Compreensão de código (descobrir onde fazer mudanças)

Extra :Testes

- Unitários
  - Teste de porções de código
  - Normalmente funcionalidade a funcionalidade
- Integração
  - Teste de vários módulos/funcionalidades e interação entre eles

Extra :Testes Exemplo

```
it( name: 'KicsRealtime Successful case ', fn: async () => {
    const auth = new CxWrapper(cxScanConfig);
    const cxCommandOutput: CxCommandOutput = await auth.kicsRealtimeScan( fileSources: "dist/tests/data/Dockerfile", additionalParams: "-v");
    const scanObject = cxCommandOutput.payload.pop();
    expect(scanObject.results.length).toBeGreaterThan( expected: 0);
    expect(scanObject.count).toBeGreaterThan( expected: 0);
})
);
```

Universidade do Minho
Escola de Engenharia

Análise Estática

e compreensão de programas

Tiago Baptista
&
Pedro Rangel Henriques