数据流测试

Data Flow Test

目录

CONTENTS

目录

CONTENTS

目录

CONTENTS

5.1 一般概念

随着程序沿着一条路径运行，存在一种**数据流**，在变量与变量之间进行**数据交换**

5.1 一般概念

进行数据流测试的两个动机：

1、程序变量所对应的内存位置应该按照期望的方式访问

2、为一个变量生成的数值需要验证其正确性

5.1 一般概念

数据流测试的两种方式：

1、静态数据流测试：分析源代码，寻找潜在的程序缺陷（称为数据流异常）

2、动态数据流测试：根据一组数据流测试标准，从源代码中识别出程序执行路径

目录

CONTENTS

5.2 数据流异常

一个异常是指不正常的或是偏离正常来做一些事情。**数据流异常**说明需要检查这些语句并消除给代码阅读者带来的歧义

5.2 数据流异常-3种类型的异常

1、二次定义错误（类型1）： 重复定义（赋值）变量。（给赋值后未使用的变量再次赋值）

2、未定义但被使用（类型2）： 在计算中使用了一个未定义的变量

3、定义但未被使用（类型3）： 定义一个变量后，没有被任何的计算引用即被销毁

5.2 数据流异常-3种类型的异常-例子

x=f1(y)
x=f2(z)

- 如果第二条语句进行了预期的计算，则第一条语句的计算是冗余的
- 第一条语句包含一个故障。例如：第一条语句可能应该是：w=f1(y)
- 第二条语句包含一个故障。例如：第二条语句可能应该是：v=f2(z)
- 第四种可能的错误是，在第一条和第二条语句之间丢掉了一条语句。例如，有语句v=f3(x)是这两条语句之间的丢掉的语句

5.2 数据流异常

如何识别数据流异常?

5.2 数据流异常-识别异常的方法

程序变量状态

1、“未定义”（U）状态：即已给变量分配空间，但未赋值

2、“定义但未被引用”（D）状态：给变量赋值之后未被引用的状态

3、“已定义已引用”（R）状态：引用该变量的值，但未重新赋值之前

4、“非正常”状态（A）：意味着一个程序异常，而且不可转换为正常状态

5.2 数据流异常-识别异常的方法

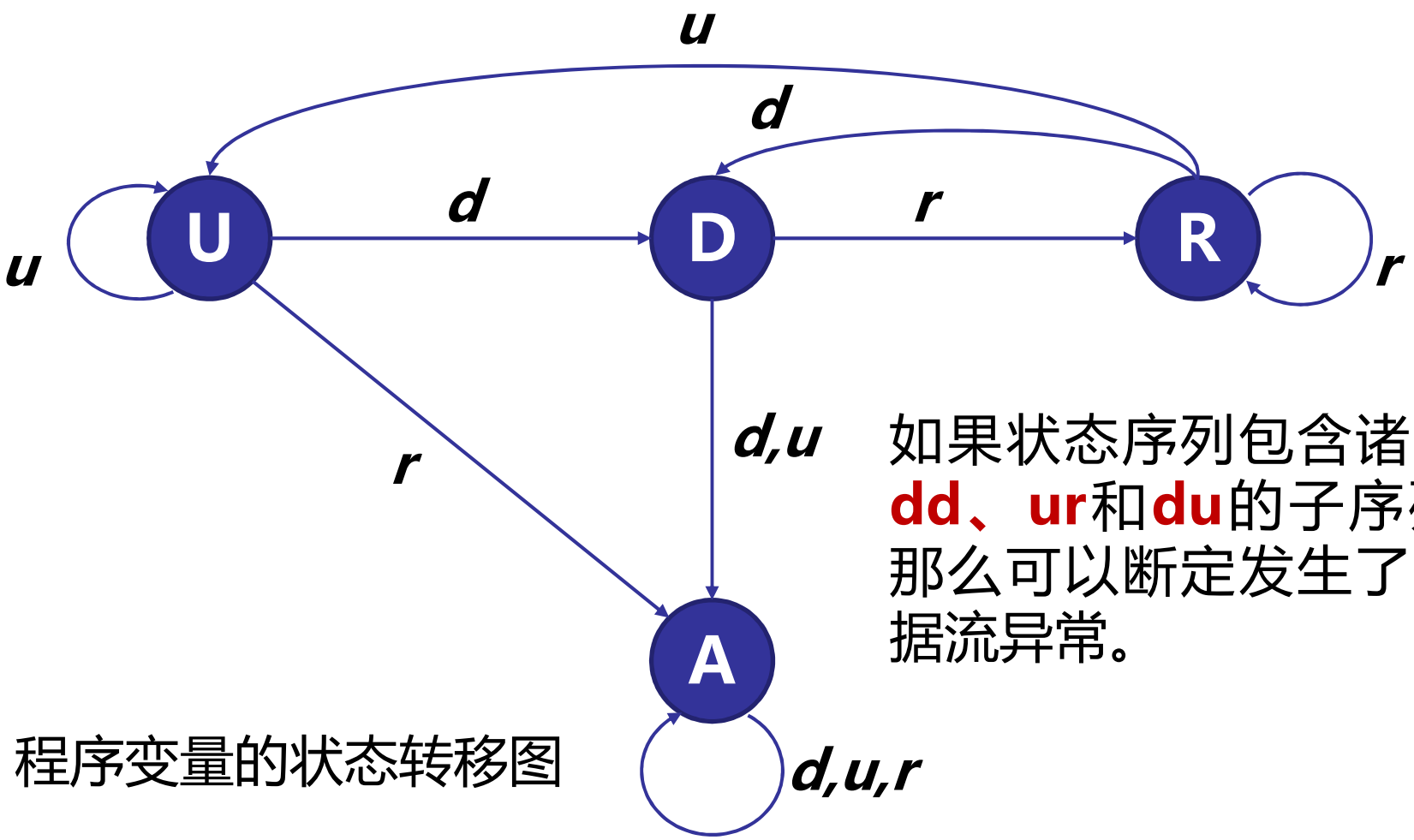
动作

1、定义（d）： 即赋值操作

2、引用（r）： 即读取操作

3、取消定义（u）： 即清空变量的内容，但不撤销变量的空间

5.2 数据流异常-识别异常的方法



程序变量的状态转移图

如果状态序列包含诸如**dd**、**ur**和**du**的子序列，那么可以断定发生了数据流异常。

目录

CONTENTS

5.3 动态数据流测试概述

对于一个变量，如果没有测试用例可以执行从该变量的赋值语句到使用该变量的语句的路径，那么应该认为这个变量**“未被赋予正确的值”**

5.3 动态数据流测试概述

进行数据流测试的两个动机：

1、给变量“赋予正确的值”意味着这个变量的值是否正确的生成出来

2、变量的使用指的是为同一个变量、其他变量及程序流控制变量生成新的值

5.3 动态数据流测试概述

数据流测试涉及某种程序执行路径。数据流测试包括选择由入口至出口的路径，目的是要覆盖一定的数据定义和使用的模式 **（数据流测试标准）**

5.3 动态数据流测试概述

数据流测试概要

1、画出程序单元数据流图

2、选择若干数据流测试标准

3、在数据流图中识别出满足数据流测试标准的路径

4、从选择的路径推导出路径谓词表达式，并求解这些表达式得到测试输入

5.4 数据流图

画数据流图的**目的**是用于识别**数据定义**及它们的**使用**

5.4 数据流图

数据变量的每一次出现可按照如下**分类**：

定义：发生在一个值移动到变量的内存地址的时刻

取消定义或销毁：发生在释放变量的值和内存地址时

使用：当一个变量的值从该变量的内存地址中取出时，使用发生

5.4 数据流图

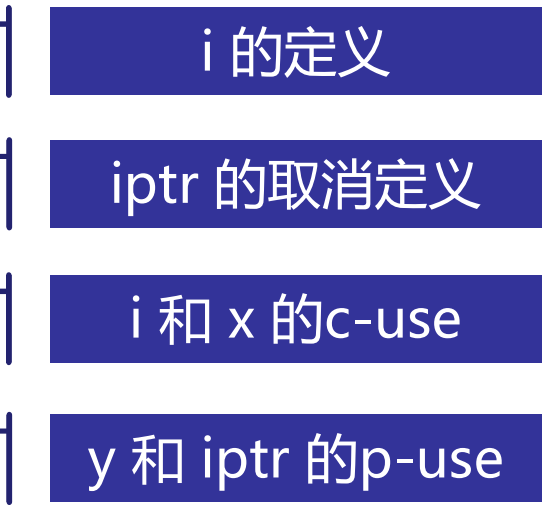
有两种**使用**形式：

计算使用（c-use）：直接影响正在进行的计算，会计算出同一个变量或其他变量的潜在的新值

谓词使用（p-use）：指在一个谓词中变量的使用，用于控制程序的执行顺序

5.4 数据流图

```
int VarTypes(int x,int y){
        int i;
        int *iptr;
        i=x;
        iptr=malloc(sizeof(int));
        *iptr=i+x;
        if(*iptr>y)
                return (x);
        else {
                iptr=malloc(sizeof(int));
                *iptr=x+y;
                return(*iptr);
                }
}
```



5.4 数据流图

数据流图的构建步骤

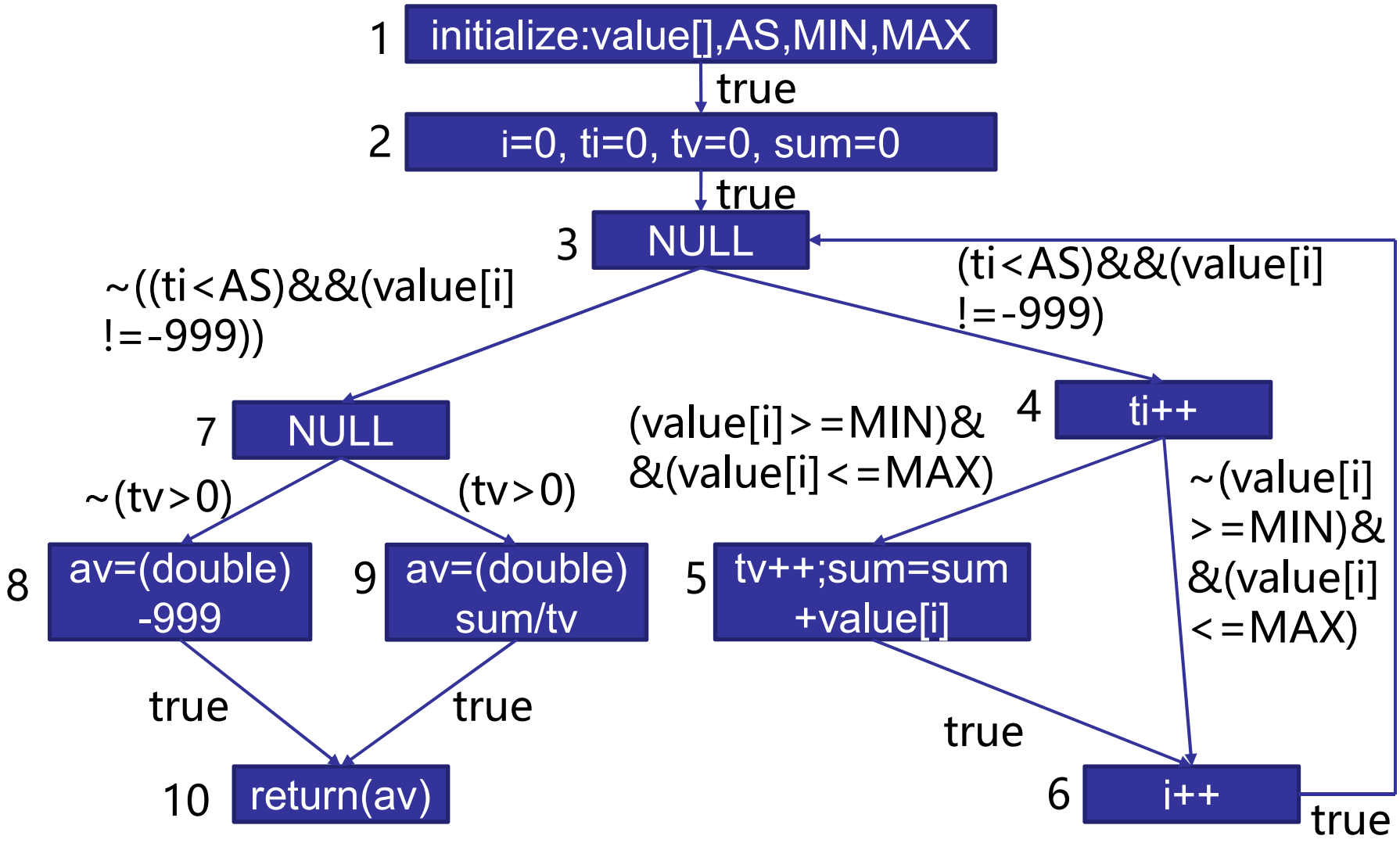
1、**定义**和**c-use序列**与图的每一个节点关联（**定义**和**c-use序列**构成图的节点，分支节点用NULL节点表示）

2、**p-use集合**与图中的每个边关联（**p-use集合**构成图的边，或用true边表示不含条件的控制流）

3、入口节点有子程序中每个参数的**定义**和每个非局部变量的**定义**

4、出口节点对每个局部变量**取消定义**

5.4 数据流图--图5.4 ReturnAverage函数的数据流图



目录

CONTENTS

5.5 数据流术语

本节介绍一些**术语**，然后使用这些术语来解释若干数据流测试**路径选取标准**

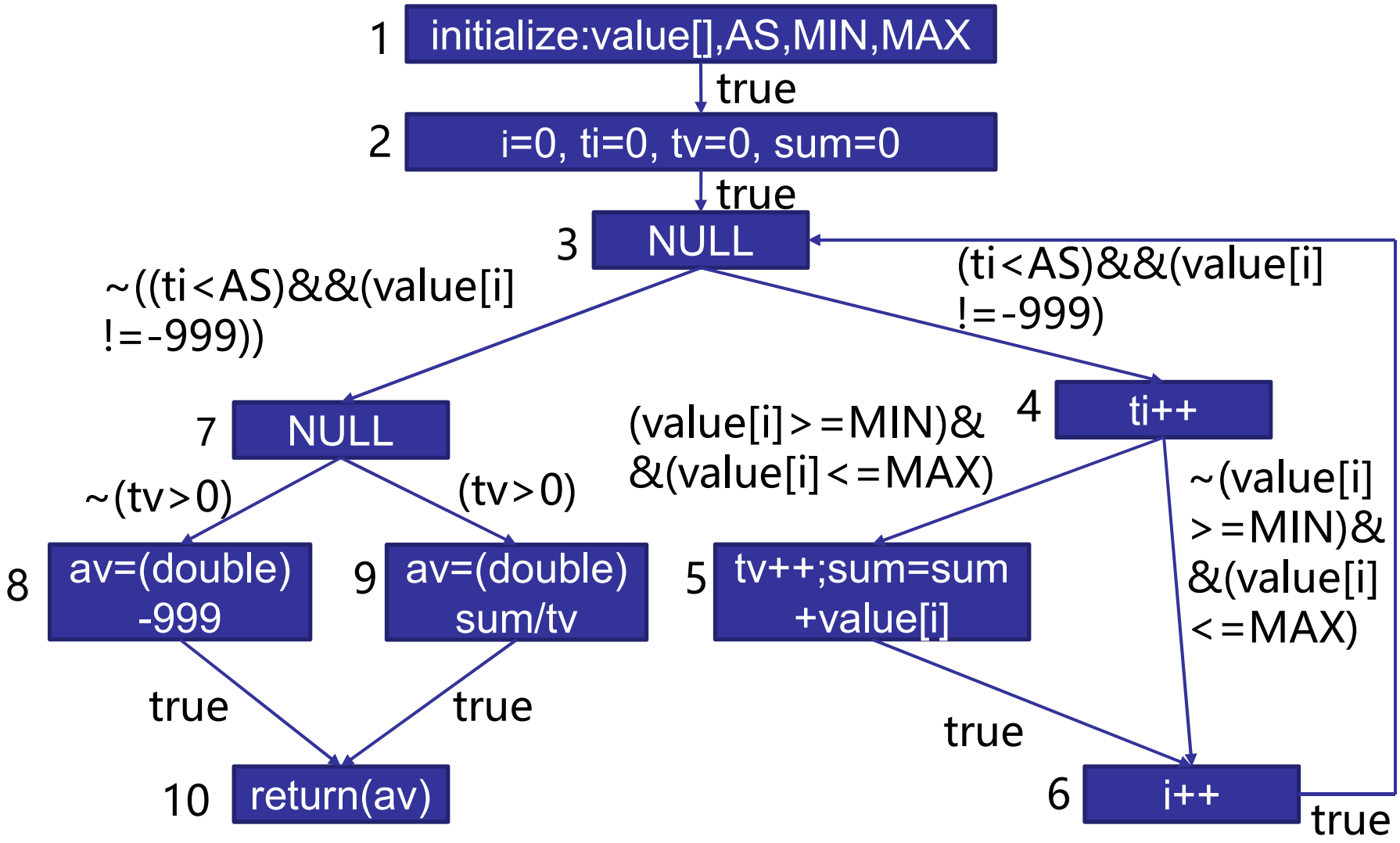
5.5 数据流术语

全局c-use

如果在节点 i 之前定义变量 x，那么节点 i 中的变量 x 的c-use称为**全局c-use**

例：图5.4中节点9中的tv是一个全局c-use

图5.4 ReturnAverage函数的数据流图



5.5 数据流术语

定义清纯路径（无定义路径）

如果变量 x 在节点 $n_1 \ldots n_m$ 中既不被定义，也没有被取消定义，则路径（**i**-n_1-…-n_m-**j**）（m≥0）称为对于变量 x 的**“定义清纯路径”（def-clear path）**：从节点 i 到节点 j；或者从节点 i 到边（n_m，j）

例：图5.4中路径2-3-4-5、2-3-4-6和2-3-4-6-3-4-6-3-4-5均是tv的定义清纯路径

5.5 数据流术语

全局定义

节点 i 是变量 x 的全局定义，当满足如下条件时，

节点 i 包含变量 x 的定义，从节点 i 到以下某处对于变量 x 有“定义清纯路径”

1.包含一个全局c-use的节点；

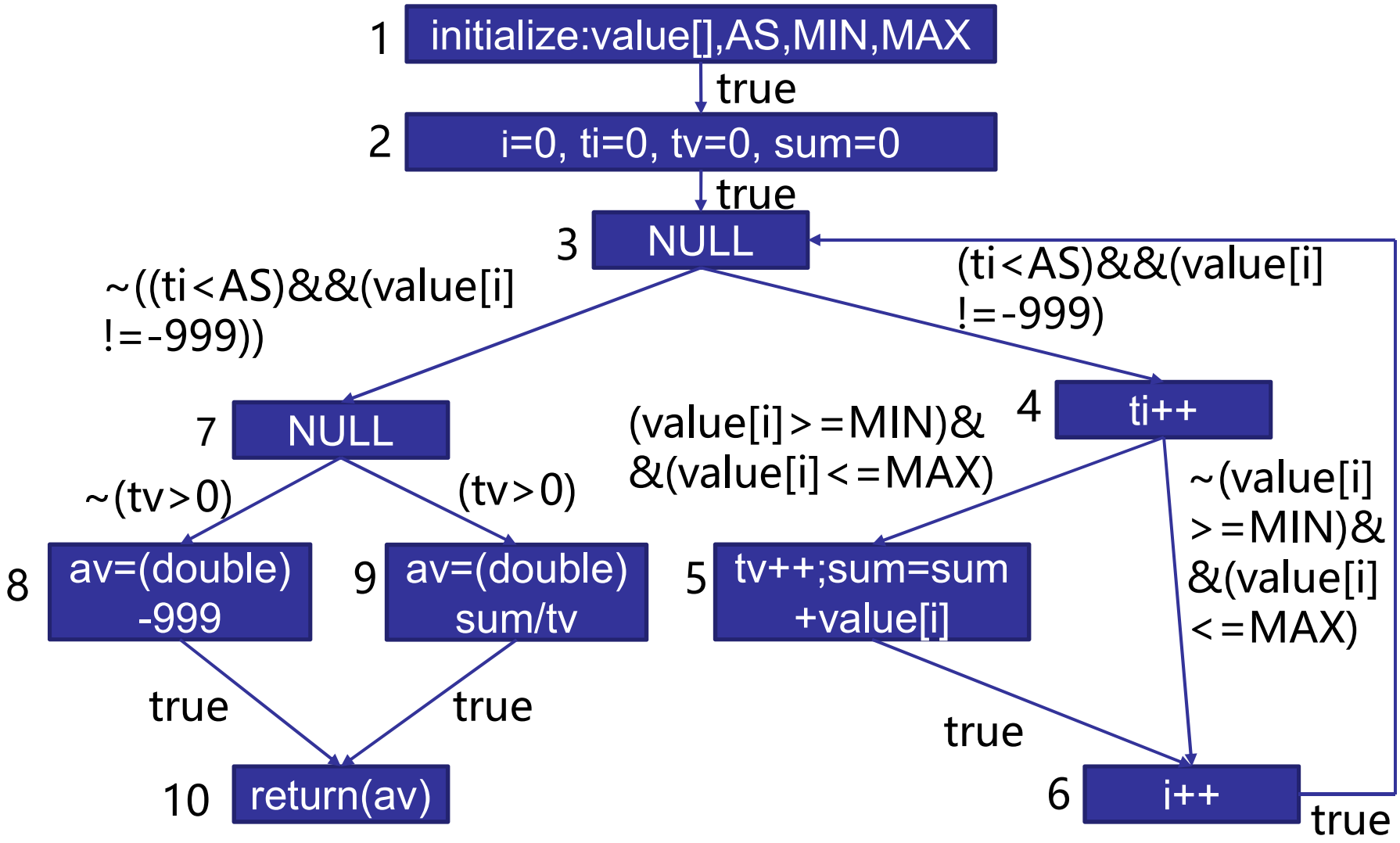
2.或者包含变量 x 的p-use的边

5.5 数据流术语

表5.1 图5.4中节点的全局定义def()和全局c-use()集合

节点i	def(i)	c-use(i)
1	{ value, AS, MIN, MAX }	{ }
2	{ i, ti, tv, sum }	{ }
3	{ }	{ }
4	{ ti }	{ ti }
5	{ tv, sum }	{ tv, i, sum, value }
6	{ i }	{ i }
7	{ }	{ }
8	{ av }	{ }
9	{ av }	{ sum, tv }
10	{ }	{ av }

图5.4 ReturnAverage函数的数据流图



5.5 数据流术语-表5.2 图5.4中边的谓词和p-use()集合

边(i,j)	predicate(i,j)	p-use(i,j)
(1,2)	True	{}
(2,3)	True	{}
(3,4)	(ti<AS)&&(value[i]!=-999)	{i,ti,AS,value}
(4,5)	(value[i]<=MIN)&&(value[i]>=MAX)	{i,MIN,MAX,value}
(4,6)	~((value[i]<=MIN)&&(value[i]>=MAX))	{i,MIN,MAX,value}
(5,6)	True	{}
(6,3)	True	{}
(3,7)	~((ti<AS)&&(value[i]!=-999))	{i,ti,AS,value}
(7,8)	~(tv>0)	{tv}
(7,9)	(tv>0)	{tv}
(8,10)	True	{}
(9,10)	True	{}

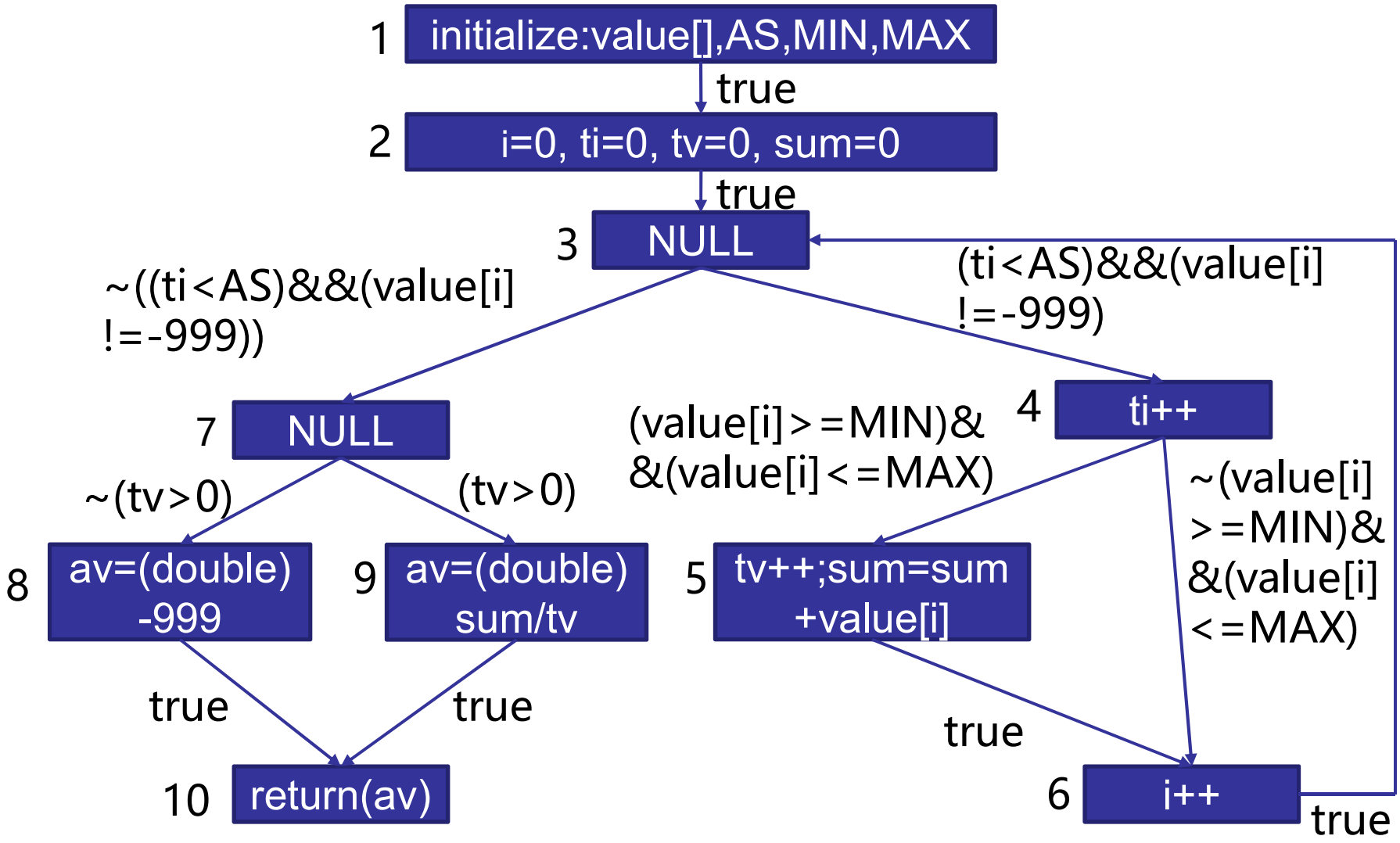
5.5 数据流术语

简单路径

是除了第一个和最后一个节点没有要求以外，所有节点都是截然不同的节点

例：图5.4中，路径2-3-4-5和3-4-6-3均是简单路径

图5.4 ReturnAverage函数的数据流图



5.5 数据流术语

无循环路径

路径上所有节点都是截然不同的节点

5.5 数据流术语

完全路径

起始于入口节点，并从出口节点退出的路径

5.5 数据流术语

定义-使用路径（du-path）

如果节点n_1含有x的一个全局定义且满足以下任一条件，则路径（n_1-n_2-...-n_j-n_k）是相对于变量x的**定义-使用路径**。满足如下任一条件：

节点n_k含有节点x的一个全局c-use，并且（n_1-n_2-...-n_j-n_k）相对于变量x是一条定义清纯简单路径

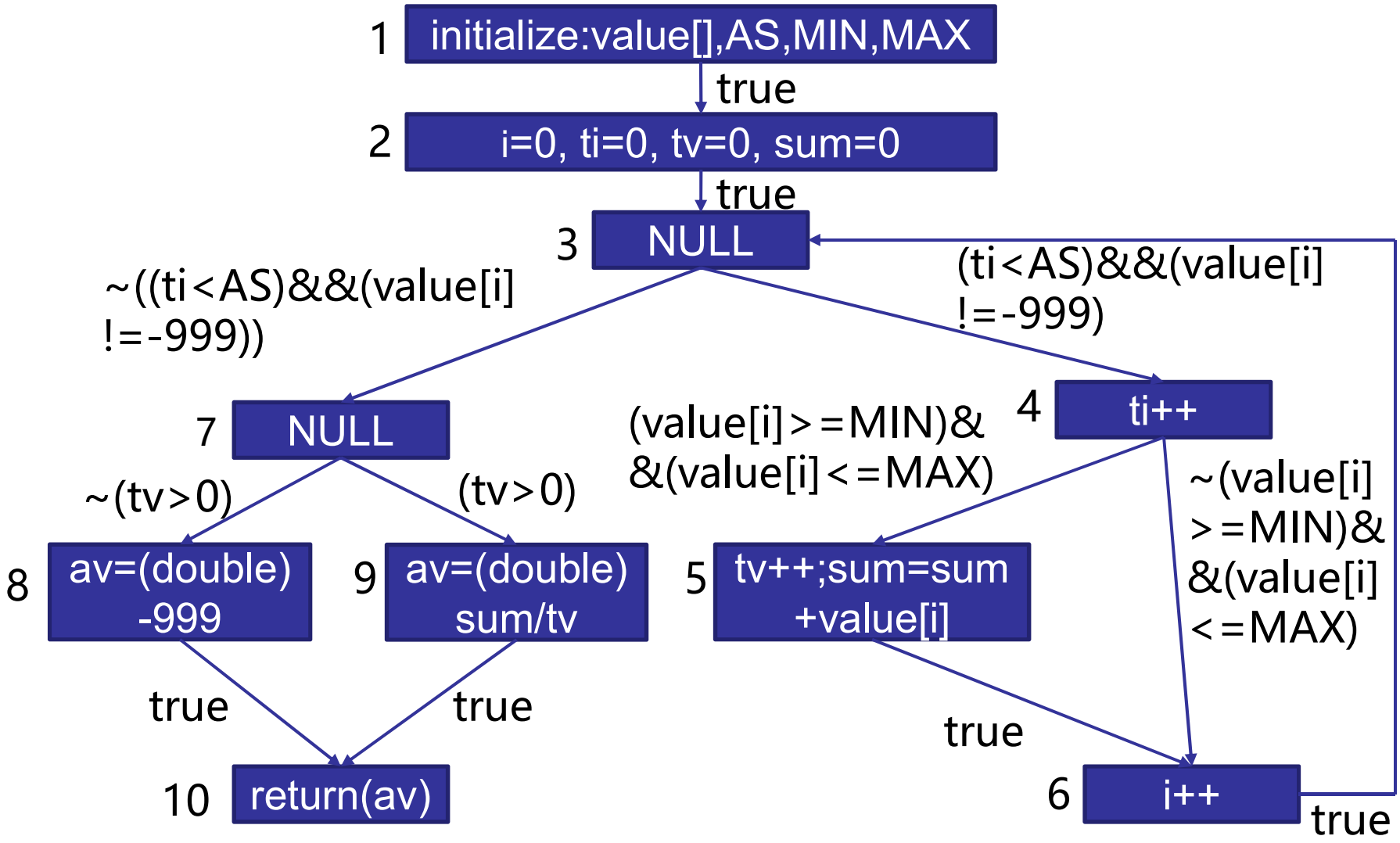
边（n_j，n_k）含有x的p-use，并且（n_1-n_2-...-n_j）相对于变量x是一条定义清纯无循环路径

定义-使用路径（du-path）--例子

示例：考虑节点2和节点5分别含有变量tv的全局定义和c-use。因此2-3-4-5是一条相对于变量tv的定义-使用路径

示例：考虑节点2和边（7,9）分别含有变量tv的全局定义和p-use。因此2-3-7-9是一条相对于变量tv的定义-使用路径

图5.4 ReturnAverage函数的数据流图



数据流测试标准

5.6 数据流测试标准

所有定义（all-defs）

对于**每个**变量x和有x的全局定义的节点i，选择包含一条**定义清纯完全路径**，从节点i到**(1)**含有x的全局c-use的节点j或者到**(2)**含有x的p-use的边（j，k）

示例：对于变量**tv**，符合all-defs标准的路径就有(1-2-3-4-5-6-3-7-9-10)、(1-2-3-7-8-10)、(1-2-3-4-5-6-3-7-9-10)。（**为了满足all-defs标准，关于变量i，ti，sum的相似路径也要找到**）

5.6 数据流测试标准

所有计算使用（all-c-uses）

对于**每个**变量x和每个节点i，如果x在节点i有全局变量定义，选择从节点i到**所有**节点j包含一条**定义清纯路径的完全路径**，使得在节点j有变量x的全局c-use

示例：对于变量ti，符合all-c-uses标准的路径就有(1-2-3-4-5-6-3-7-8-10)、(1-2-3-4-5-6-3-7-9-10)、(1-2-3-4-6-3-7-8-10)、(1-2-3-4-6-3-7-9-10)

5.6 数据流测试标准

所有谓词使用（all-p-uses）

对于**每个**变量x和每个节点i，如果在节点i上x有全局定义，选择从节点i到**所有**边（j，k）包含一条**定义清纯路径的完全路径**，使得在边（j，k）有变量x的全局p-use

示例：对于变量tv，符合all-p-uses标准的路径就有(1-2-3-7-8-10)、(1-2-3-7-9-10)、(1-2-3-4-5-6-3-7-8-10)、(1-2-3-4-5-6-3-7-9-10)

5.6 数据流测试标准

所有谓词使用/部分计算使用（all-p-uses/some-c-uses)

如果变量x无p-use，则等同于**部分计算使用（some-c-uses）**：对于每个变量x和每个节点i，如果x在节点i有全局定义，选择从节点i到**某些**节点j的**定义清纯完全路径**，使得在节点j有变量x的全局c-use

示例：对于变量i，符合all-p-uses/some-c-uses标准的路径就有(1-2-3-4-5-6-3-7-9-10)

5.6 数据流测试标准

所有计算使用/部分谓词使用（all-c-uses/some-p-uses）

如变量x无全局c-use，则等同于**部分谓词使用（some-p-uses）**：对于每个变量x和每个节点i，如果x在节点i有全局定义，选择从节点i到**某些**边（j，k）的**定义清纯完全路径**，使得在边（j，k）有变量x的p-use

示例：对于变量AS，符合标准的路径就有(1-2-3-4-5-6-3-7-9-10)

5.6 数据流测试标准

所有使用（all-uses)

=所有计算使用（all-c-uses）+所有谓词使用(all-p-uses)

5.6 数据流测试标准

所有定义-使用路径（all-du-path）

对于**每个**变量x和每个节点i，如果节点i里有变量x的全局定义，选择包含all-du-path的**完全路径**，从节点i出发到

(1)到**所有**节点j，使得节点j有变量x的全局c-use；

(2)到**所有**边（j，k），使得边（j，k）有变量x的全局p-use

目录

CONTENTS

5.7 数据流测试选择标准的比较

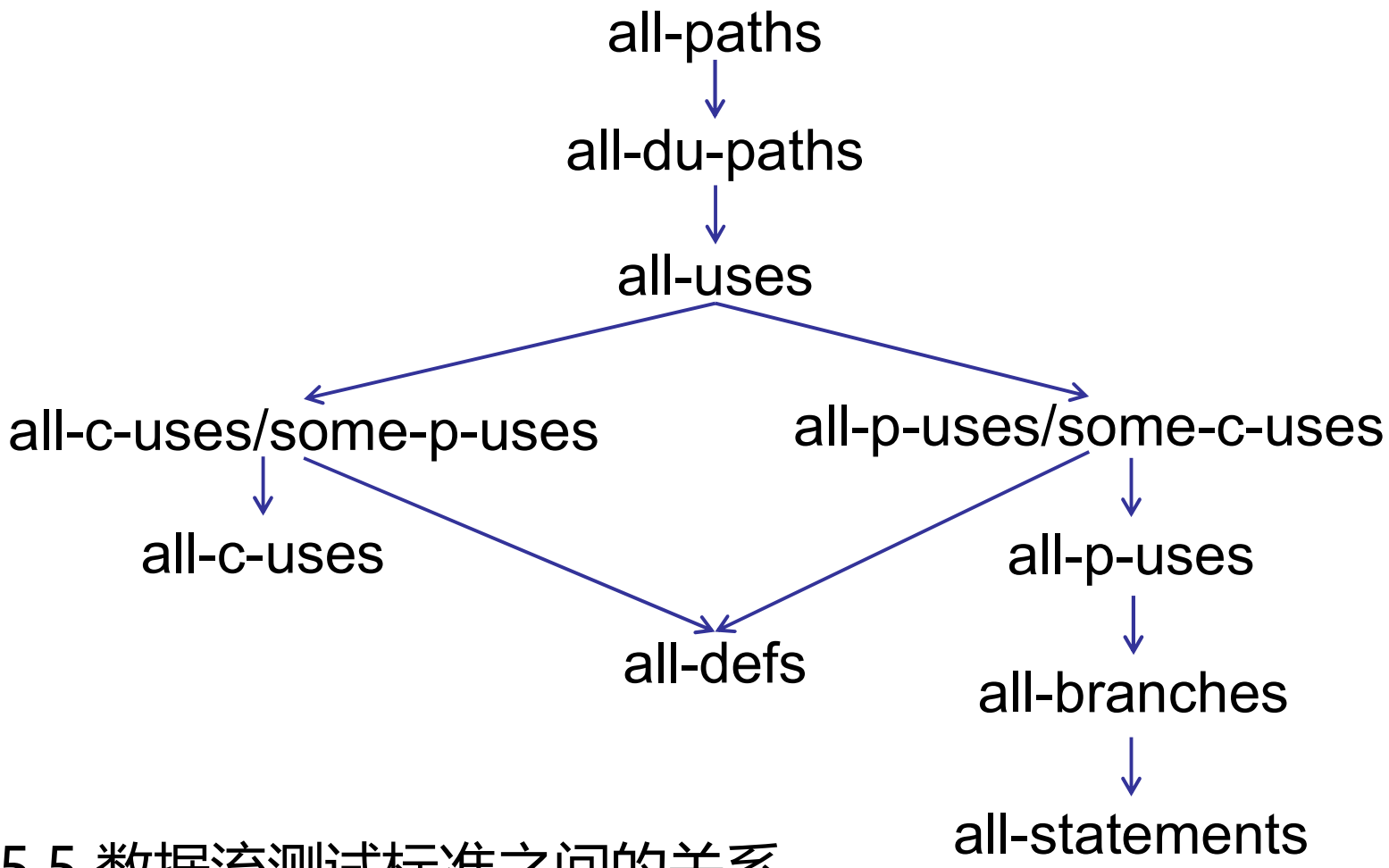


图5.5 数据流测试标准之间的关系

5.7 数据流测试选择标准的比较

all-p-uses标准严格包含**all-branches**标准。这意味着从数据流图中能够比从控制流图中选择更多的路径

5.8 可行路径和测试选择标准

一个完全路径是**可执行(executable)**的，如果存在对它的输入变量和全局变量的赋值操作，并且路径上的所有谓词都为true。这样的路径也称为**可行(feasible)路径**

5.8 可行路径和测试选择标准

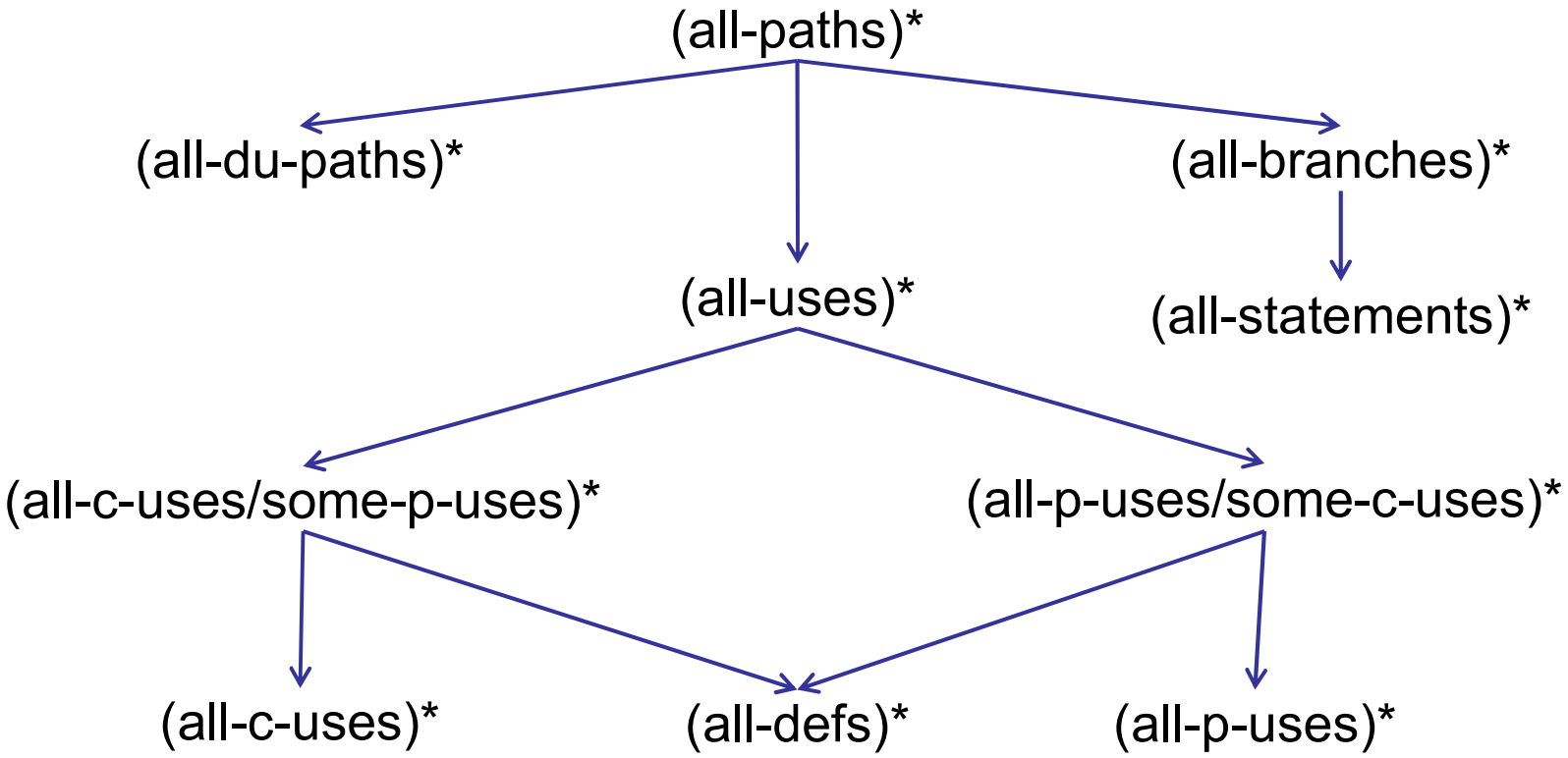


图5.6 FDF测试选择标准之间的关系

5.9 测试技术的比较

5.9 测试技术的比较

Ntafos报告了这三种技术的比较实验结果。这个实验包含了7个有已知程序错误的数学软件。对于基于控制流的技术，选择了**分支覆盖标准**。在基于数据流的测试中，使用了**all-uses标准**。并且随机测试方法也应用到程序测试中。

5.9 测试技术的比较

数据流测试、**分支**测试和**随机**测试分别检查出 **90%、85.5%、79.5%**的已知错误。设计**84个**测试用例用于实现all-uses标准，设计 **34个**测试用例用于实现分支覆盖标准，设计**100个**测试用例用于随机测试方法。

数据流测试

End