北京理工大学

# 软件测试与质量保证

3.3 因果图测试

张宇霞 副研究员

# 目录

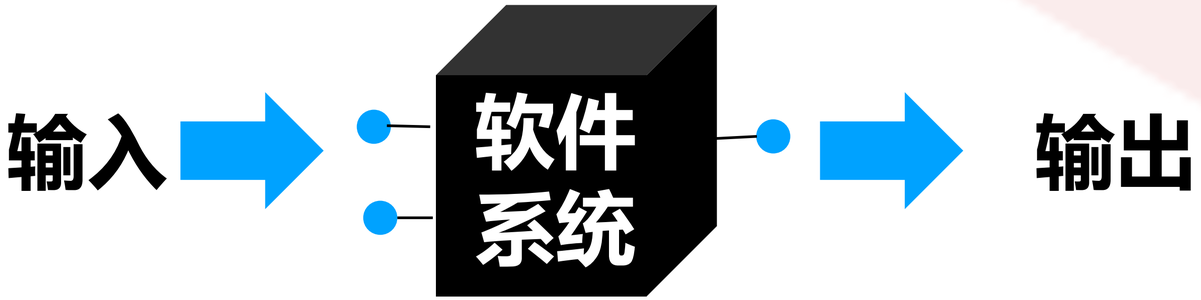
CONTENTS

# 目录

CONTENTS

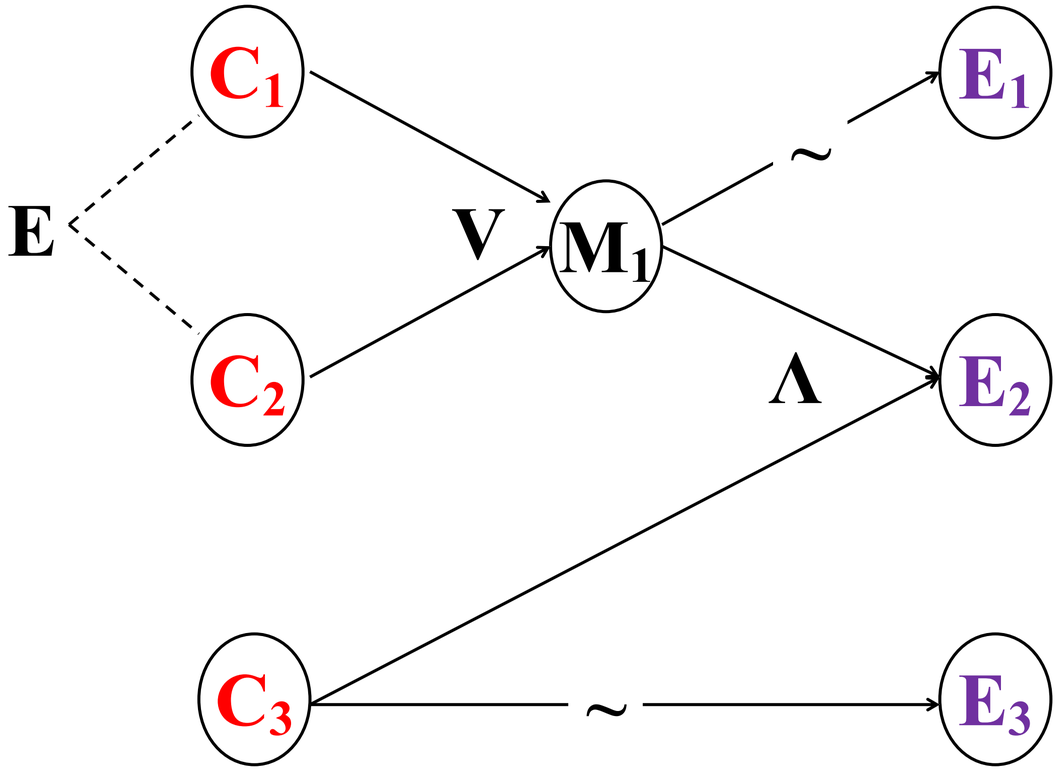
# 01 因果图

- **黑盒测试技术**
  - 等价类划分
  - 边界值分析
  - 因果图
  - 输入组合法
  - 基于状态测试



01

# 因果图

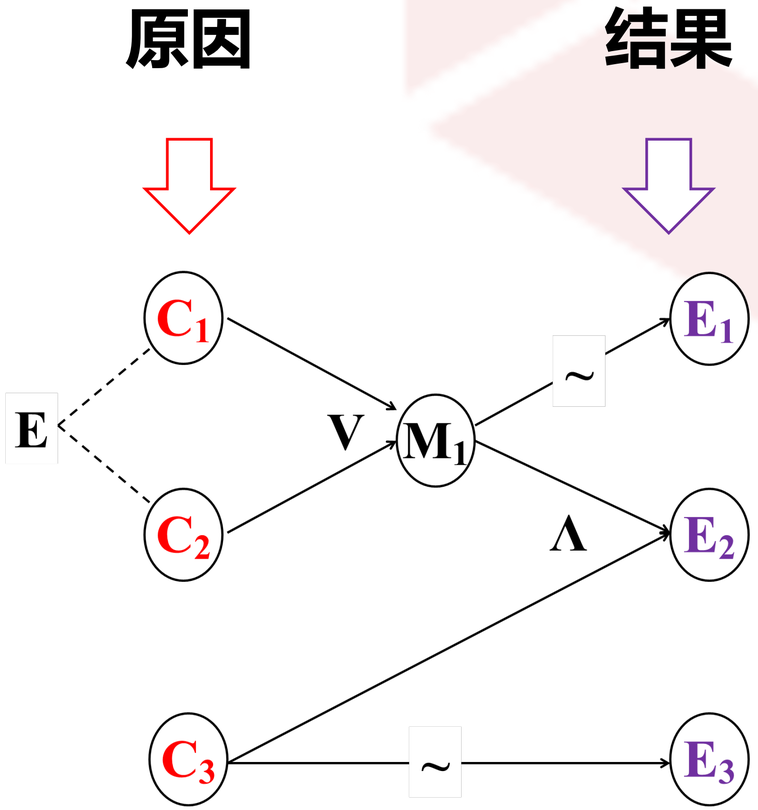


01

# 因果图

## ■ 因果图的基本元素

- 原因： $C_i$
- 结果： $E_j$
- 原因和结果之间的因果关系
- 原因和原因之间的关系
- 结果和结果之间的关系




原因：比如说输入里面带有特殊字符，或者你按了某个按钮等等。

结果：程序的输出满足了某个有意义的触发条件，比如程序异常，程序中断，或者从ATM吐出钞票等等

01

# 因果图

■ **因果图的基本元素**

- 原因： $C_i$
- 结果： $E_j$

$C_i == 0$ → 该原因没有出现

$C_i == 1$ → 该原因出现/发生

$E_j == 0$ → 该结果没有出现

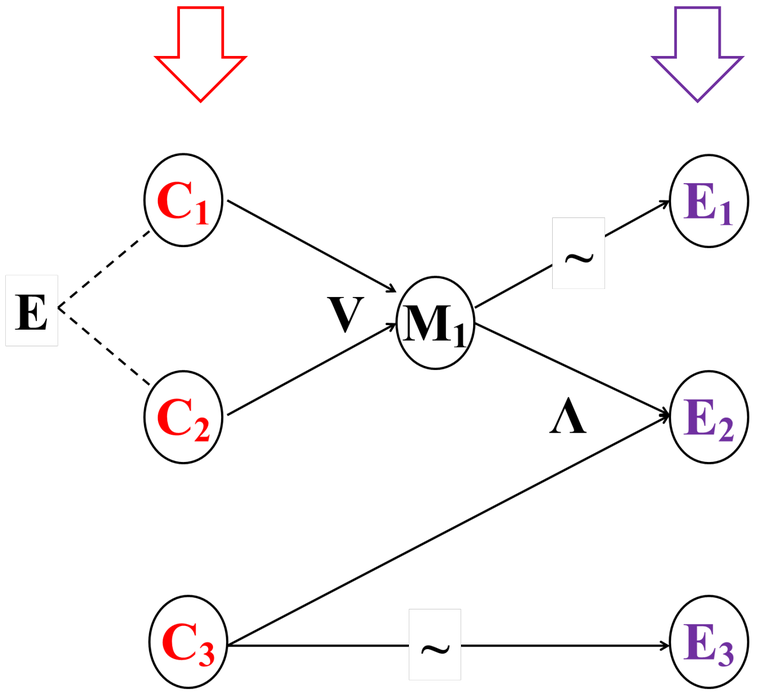
$E_j == 1$ → 该结果出现

01

# 因果图

■ **原因与结果之间的关系**

- 恒等
- 非
- 或
- 与



# 01 因果图

■ 恒等：若$C_i$是1，则$E_j$也为1，否则$E_j$为0；

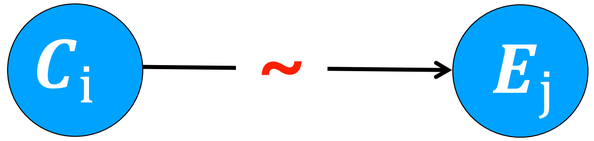
若$C_i$出现，则$E_j$必出现，否则$E_j$不可出现。

01

# 因果图

■ 非：若$C_i$是1，则$E_j$也为0，否则$E_j$为1；

若$C_i$出现，则$E_j$不可出现，否则$E_j$必出现。

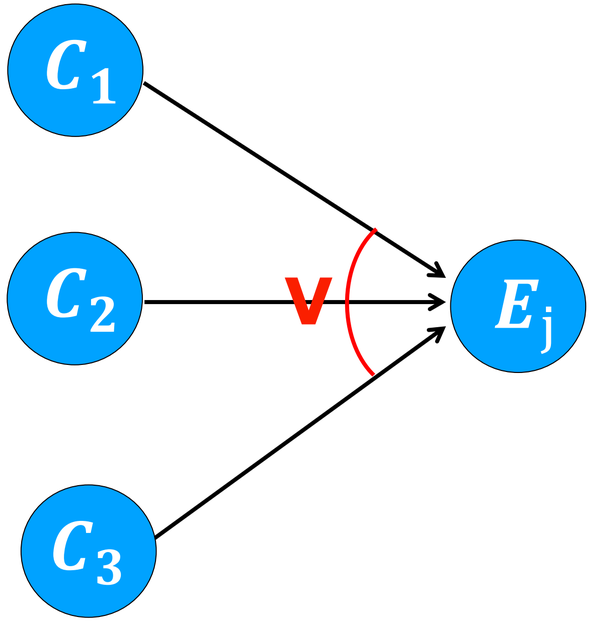


01

# 因果图

■ 或：若$C_1$或$C_2$或$C_3$是1，则$E_j$也为1，否则$E_j$为0；

若$C_1$或$C_2$或$C_3$中的任意一个原因出现，则$E_j$必出现，否则$E_j$不会出现。

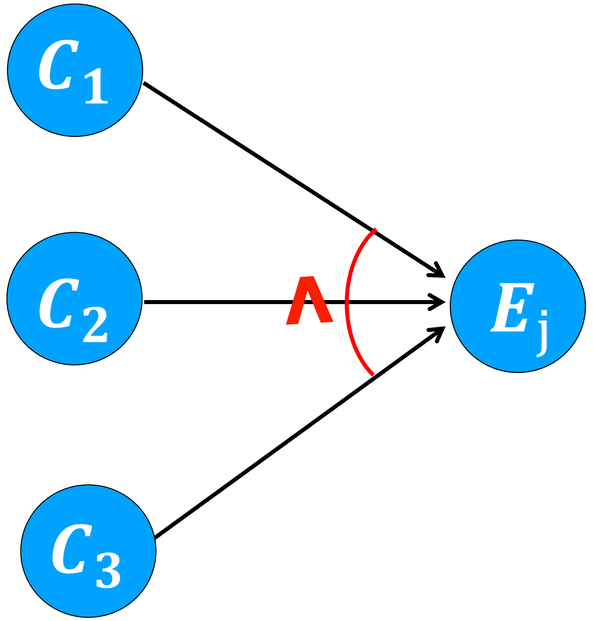


01

# 因果图

■ 与：若$C_1$、$C_2$和$C_3$都是1，则$E_j$也为1，否则$E_j$为0；

若$C_1$、$C_2$和$C_3$都出现，则$E_j$必出现，否则$E_j$不会出现。



01

# 因果图

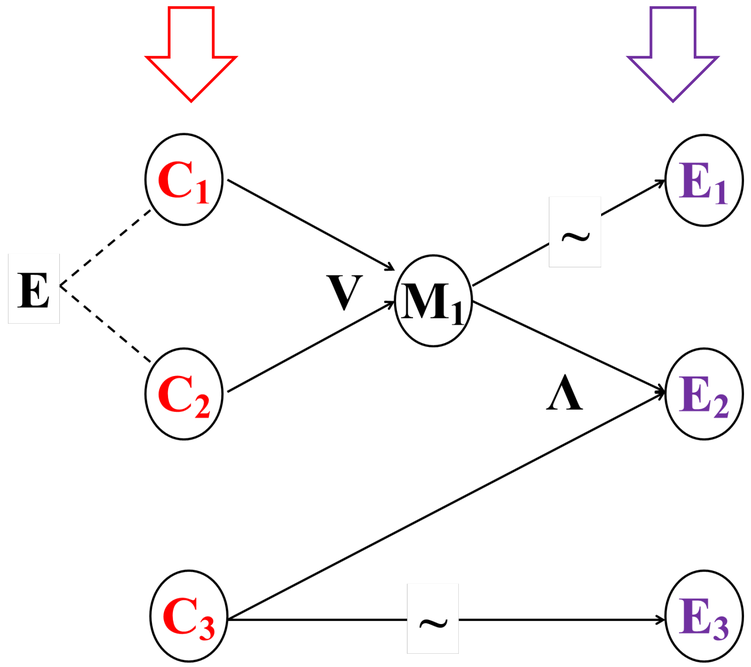
- 恒等：若$C_i$是1，则$E_j$也为1，否则$E_j$为0；
- 非：若$C_i$是1，则$E_j$也为0，否则$E_j$为1；
- 或：若$C_1$或$C_2$或$C_3$是1，则$E_j$也为1，否则$E_j$为0；
- 与：若$C_1$、$C_2$和$C_3$都是1，则$E_j$也为1，否则$E_j$为0。

01

# 因果图

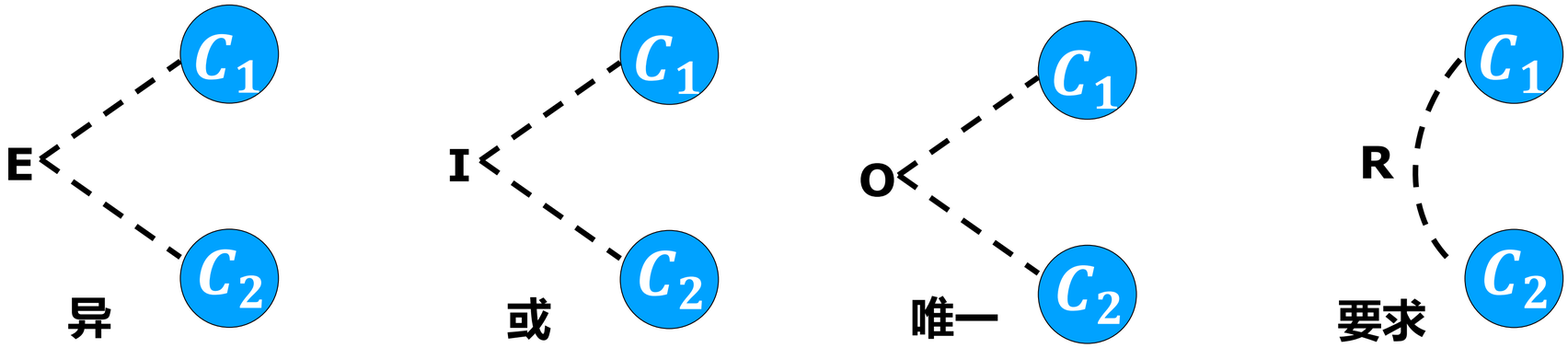
■ **原因与原因之间的关系**

- 异
- 或
- 唯一
- 要求



# 01 因果图

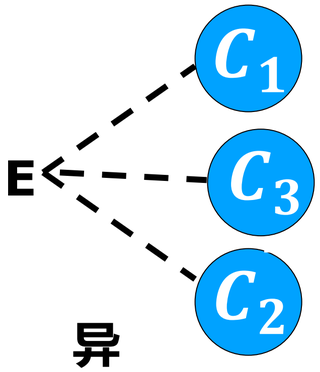
## ■ 原因与原因之间的关系



# 01 因果图

■ E约束（异）： $C_1$、$C_2$中最多有一个可能为1，即$C_1$、$C_2$不能同时为1

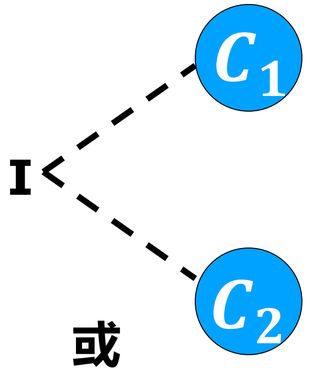
$C_1$、$C_2$不能同时出现



# 01 因果图

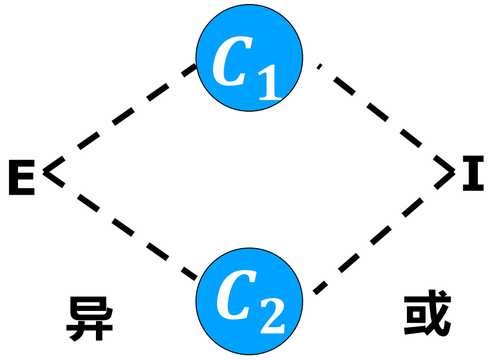
■ I约束（或）： $C_1$、$C_2$中最少有一个为1，即$C_1$、$C_2$不能同时为0

$C_1$、$C_2$不能同时都不出现



# 01 因果图

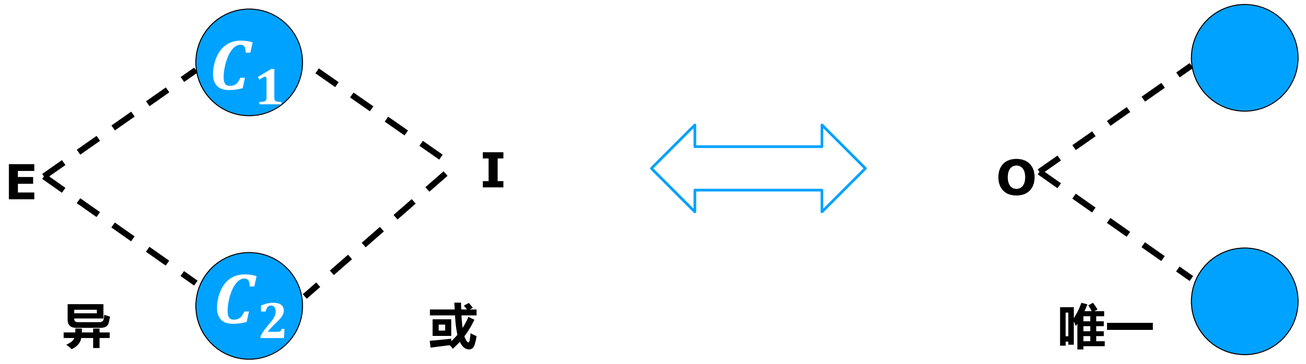
■ E/I约束： $C_1$、$C_2$中有且只有一个为1



01

# 因果图

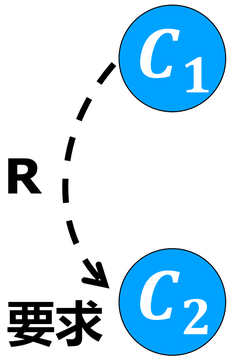
■ O约束（唯一）： $C_1$、$C_2$中有且只有一个为1



# 01 因果图

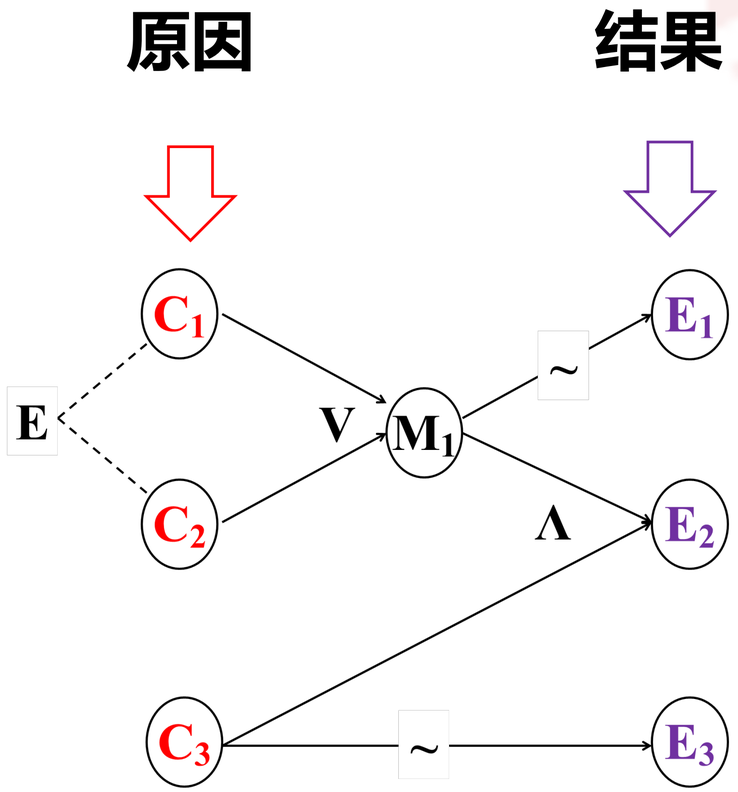
■ R约束（要求）： $C_1$为1时，$C_2$必须为1

$C_1$出现，则$C_2$必同时出现



# 01 因果图

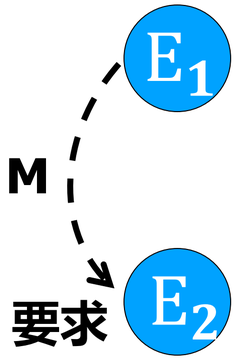
- 结果与结果之间的关系
  - 强制（M约束）



01

# 因果图

■ M约束（强制）： $E_1$为1则要求$E_2$必须为0



01

# 因果图

■ **原因和原因之间的：**

- E约束（异）： $C_1$、$C_2$中最多有一个可能为1，即$C_1$、$C_2$不能同时为1。
- I约束（或）： $C_1$、$C_2$中最少有一个为1，即$C_1$、$C_2$不能同时为0。
- O约束（唯一）： $C_1$、$C_2$中有且只有一个为1。
- R约束（要求）： $C_1$为1时，$C_2$必须为1。

■ **结果和结果之间的：**

- M约束（强制）： $E_1$为1则要求$E_2$必须为0。

# 目录

CONTENTS

02

# 因果图测试

- **因果图测试方法：利用图解法分析输入的各种组合情况，进而据此设计测试用例。**
  - 检查输入条件的各种组合以及输入条件之间的互相制约关系。
  - 特别适合程序输入之间具有复杂关系的那些软件的测试工作。

    （检查程序输入条件的各种组合）

02

# 因果图测试

■ **因果图测试方法的具体步骤：**

① 首先从程序规格说明书的描述中，找出**因**(输入条件)和**果**(输出结果或者程序状态的改变)；

② 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图；

③ 然后通过因果图转换为**判定表**；

④ 最后为判定表中的每一列设计一个测试用例。

02

# 因果图测试

■ **程序规格说明书：**

- 输入的第一个字符必须是#或!，第二个字符必须是F，在此情况下进行文件的修改；
- 如果第一个字符不是#或!，则给出信息：“错误命令”；
- 如果第二个字符不是F，则给出信息：“错误操作类型”。

02

# 因果图测试

■ **因果图测试方法的具体步骤：**

**① 首先从程序规格说明书的描述中，找出因(输入条件)和果(输出结果或者程序状态的改变)；**

② 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图；

③ 然后通过因果图转换为判定表；

④ 最后为判定表中的每一列设计一个测试用例。

02

# 因果图测试

■ **程序规格说明书：**

- 输入的第一个字符必须是#或!，第二个字符必须是F，在此情况下进行文件的修改；
- 如果第一个字符不是#或!，则给出信息：“错误命令”；
- 如果第二个字符不是F，则给出信息：“错误操作类型”。

02

# 因果图测试

## ■ 原因

- $C_1$：第一个字符是 #。
- $C_2$：第一个字符是！。
- $C_3$：第二个字符是 F。

02

# 因果图测试

■ **程序规格说明书：**

- 输入的第一个字符必须是#或!，第二个字符必须是F，在此情况下进行文件的修改；
- 如果第一个字符不是#或!，则给出信息：“错误命令”；
- 如果第二个字符不是F，则给出信息：“错误操作类型”。

02

# 因果图测试

## ■ 结果

- $E_1$：给出信息：“错误命令”。
- $E_2$：进行文件的修改。
- $E_3$：给出信息：“错误操作类型”。

02

# 因果图测试

## ■ 原因

- $C_1$：第一个字符是 #。
- $C_2$：第一个字符是！。
- $C_3$：第二个字符是 F。

## ■ 结果

- $E_1$：给出信息：“错误命令”。
- $E_2$：进行文件的修改。
- $E_3$：给出信息：“错误操作类型”。

02

# 因果图测试

■ **因果图测试方法的具体步骤：**

① 首先从程序规格说明书的描述中，找出因(输入条件)和果(输出结果或者程序状态的改变)；

**② 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图；**

③ 然后通过因果图转换为判定表；

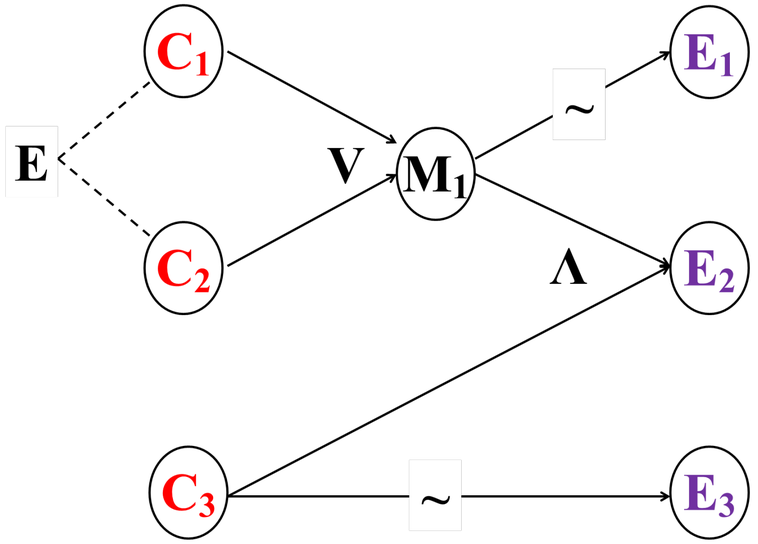
④ 最后为判定表中的每一列设计一个测试用例。

02

# 因果图

■ 程序规格说明书：

- 输入的第一个字符必须是#或!，第二个字符必须是F，在此情况下进行文件的修改；
- 如果第一个字符不是#或!，则给出信息：“错误命令”；
- 如果第二个字符不是F，则给出信息：“错误操作类型”。




■ 原因：

- $C_1$：第一个字符是#
- $C_2$：第一个字符是!
- $C_3$：第二个字符是F

■ 结果

- $E_1$：给出信息：“错误命令”
- $E_2$：进行文件的修改。
- $E_3$：给出信息：“错误操作类型”

02

# 因果图测试

■ **因果图测试方法的具体步骤：**

① 首先从程序规格说明书的描述中，找出因(输入条件)和果(输出结果或者程序状态的改变)；

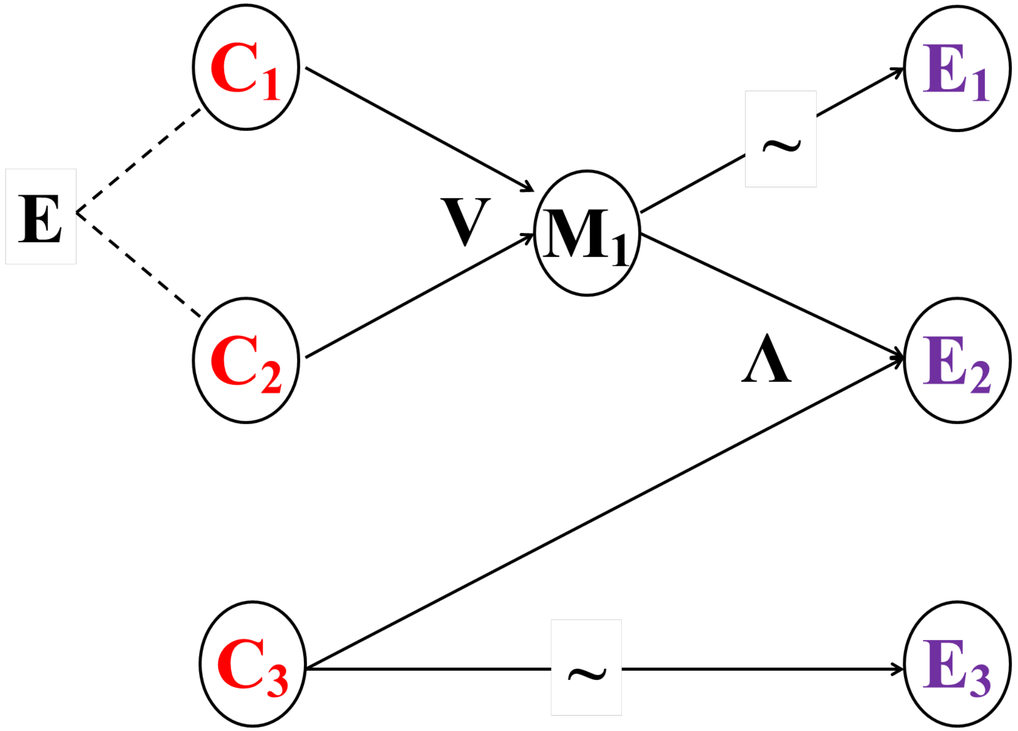
② 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图；

**③ 然后通过因果图转换为判定表；**

④ 最后为判定表中的每一列设计一个测试用例。

02

# 因果图



02

# 因果图

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $C_2$ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| $C_3$ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $M_1$ | | | 1 | 1 | 1 | 1 | 0 | 0 |

02

# 因果图测试

■ **因果图测试方法的具体步骤：**

① 首先从程序规格说明书的描述中，找出因(输入条件)和果(输出结果或者程序状态的改变)；

② 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图；

③ 然后通过因果图转换为判定表；

**④ 最后为判定表中的每一列设计一个测试用例。**

02

# 因果图

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $C_2$ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| $C_3$ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $M_1$ | | | 1 | 1 | 1 | 1 | 0 | 0 |
| $E_1$ | | | | | | | √ | √ |
| $E_2$ | | | √ | | √ | | | |
| $E_3$ | | | | √ | | √ | | √ |
| 不可能 | √ | √ | | | | | | |

02

# 因果图

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $C_2$ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| $C_3$ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $M_1$ | | | 1 | 1 | 1 | 1 | 0 | 0 |
| $E_1$ | | | | | | | √ | √ |
| $E_2$ | | | √ | | √ | | | |
| $E_3$ | | | | √ | | √ | | √ |
| 不可能 | √ | √ | | | | | | |
| 测试用例 | | | #F | #B | !F | !M | CF | CM |

02

# 因果图

- Test1：输入数据 - #F　预期输出 -　- 修改文件
- Test2：输入数据 - #B　预期输出 -　- 给出信息M
- Test3：输入数据 - !F　预期输出 -　- 修改文件
- Test4：输入数据 - !M　预期输出 -　- 给出信息M
- Test5：输入数据 - CF　预期输出 -　- 给出信息N
- Test6：输入数据 - CM　预期输出 -　- 给出信息M和N

02

# 因果图测试

## ■ 饮料的自动售货机

➢ 若投入5角钱或1元钱的硬币，押下〖橙汁〗或〖啤酒〗的按钮，则相应的饮料就送出来。若售货机没有零钱找，则一个显示〖零钱找完〗的红灯亮，这时在投入1元硬币并押下按钮后，饮料不送出来而且1元硬币也退出来；若有零钱找，则显示〖零钱找完〗的红灯灭，在送出饮料的同时退还5角硬币。

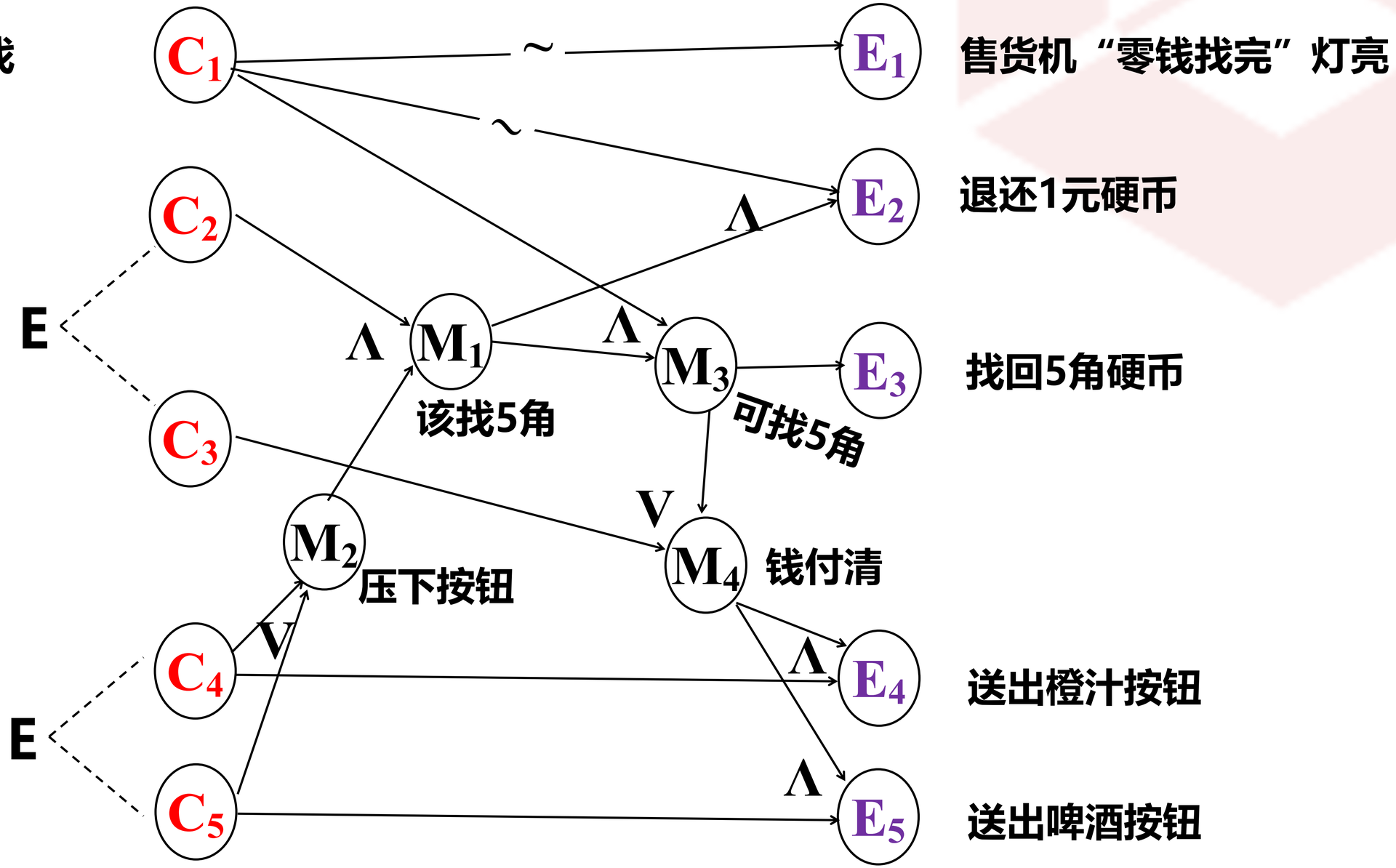
02

# 因果图测试

| | | | |
|---|---|---|---|
| 原因： | 1.售货机有零钱找 | 3.投入5角硬币 | 5.压下啤酒按钮 |
| | 2.投入1元硬币 | 4.压下橙汁按钮 | |
| 结果： | 21. 售货机〖零钱找完〗灯亮 | | 24. 送出橙汁饮料 |
| | 22. 退还1元硬币 | | 25. 送出啤酒饮料 |
| | 23. 退还5角硬币 | | |

02

# 因果图测试


售货机有零钱找
投入1元硬币
投入5角硬币
压下橙汁按钮
压下啤酒按钮
C1
C2
C3
C4
C5
E
E
M1
该找5角
M2
压下按钮
M3
可找5角
M4
钱付清
E1
售货机“零钱找完”灯亮
E2
退还1元硬币
E3
找回5角硬币
E4
送出橙汁按钮
E5
送出啤酒按钮
~
~
Λ
Λ
Λ
V
V
Λ
Λ

02

# 因果图测试

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 20 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 30 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $C_2$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $C_3$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $C_4$ | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| $C_5$ | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $M_1$ | | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| $M_2$ | | | | | | 1 | 1 | 0 | | 1 | 1 | 0 | | 1 | 1 | 0 | | | | | 1 | 1 | 0 | | 1 | 1 | 0 | | 1 | 1 | 0 |
| $M_3$ | | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| $M_4$ | | | | | | 1 | 1 | 0 | | 1 | 1 | 1 | | 0 | 0 | 0 | | | | | 0 | 0 | 0 | | 1 | 1 | 1 | | 0 | 0 | 0 |
| $E_1$ | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | | | | 1 | 1 | 1 | | 1 | 1 | 1 | | 1 | 1 | 1 |
| $E_2$ | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| $E_3$ | | | | | | 1 | 1 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | | | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 |
| $E_4$ | | | | | | 1 | 0 | 0 | | 1 | 0 | 0 | | 0 | 0 | 0 | | | | | 0 | 0 | 0 | | 1 | 0 | 0 | | 0 | 0 | 0 |
| $E_5$ | | | | | | 0 | 1 | 0 | | 0 | 1 | 0 | | 0 | 0 | 0 | | | | | 0 | 0 | 0 | | 0 | 1 | 0 | | 0 | 0 | 0 |
| 测试用例 | | | | | | Y | Y | Y | | Y | Y | Y | | Y | Y | Y | | | | | Y | Y | Y | | Y | Y | Y | | Y | Y | Y |

02

# 因果图测试

## ■ 因果图法的优点

- 考虑到了输入的各种组合以及各个输入情况之间的相互制约关系。
- 能够帮助测试人员按照一定的步骤，高效率的开发测试用例。
- 因果图法是将自然语言规格说明转化成形式语言规格说明的一种严格的方法，可以指出规格说明存在的不完整性和二义性。

# 目录

CONTENTS

# 04 小结

因果图测试是黑盒测试的重要手段

因果图测试方法：利用图解法分析输入的各种组合情况，
进而据此设计测试用例

因果图测试方法的具体步骤：
①找因果②画关系③转判定表④选用例

# 谢谢！