软件分析与测试
符号执行补充

2023.11

中国科学院大学
University of Chinese Academy of Sciences
ISCAS 中国科学院软件研究所
Institute of Software, Chinese Academy of Sciences

符号执行：课堂作业

- 对于以下求最大公约数的代码（$m \geq 0, n \geq 0$），回答问题：

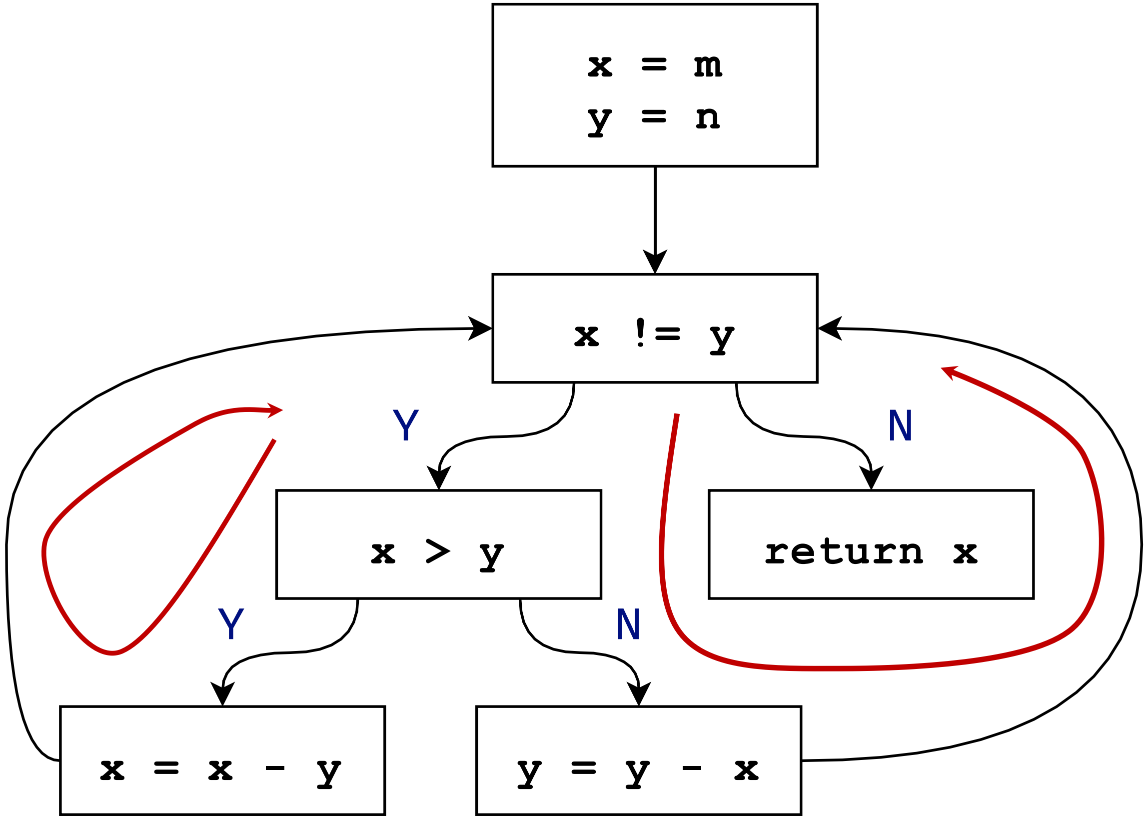
```
1   uint32_t gcd(
2     uint32_t m, uint32_t n
3   ) {
4     uint32_t x, y;
5     x = m;
6     y = n;
7     while (x != y) {
8       if (x > y)
9         x = x - y;
10      else
11        y = y - x;
12    }
13    return x;
14  }
```

- 1. 画出控制流图，并求该程序的圈复杂度
- 2. 使用符号执行的方法找到**达到分支和语句覆盖的**路径，写出行号表示的路径，以及用符号值表示的路径约束
- 3. 针对以上路径采用符号执行技术分别给出一组可行输入数据（测试用例）
- 4. 仔细观察程序结构和 m、n 的取值范围，分析是否存在缺陷？
  结合符号执行的特点简要阐述。

符号执行：题解

■ 1. 求该程序的圈复杂度

```
uint32_t gcd(
  uint32_t m, uint32_t n
) {
  uint32_t x, y;
  x = m;
  y = n;
  while (x != y) {
    if (x > y)
      x = x - y;
    else
      y = y - x;
  }
  return x;
}
```



圈复杂度 = “空间”个数 （7 人， 25.9%）

= 判断节点数 + 1 （2 人， 7.4%）

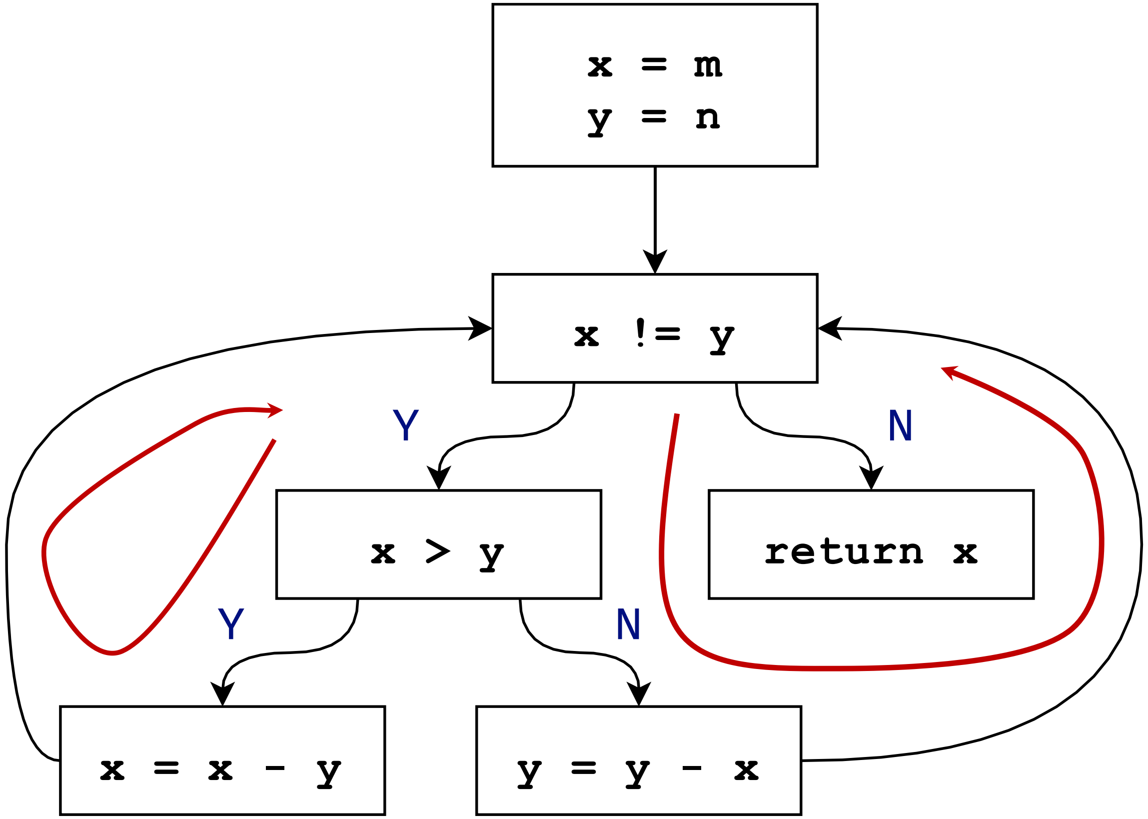
= E – N + 2P （14 人， 51.9%）

= 3 （23 人，85.2%）

符号执行：题解

- 1. 求该程序的圈复杂度

```
uint32_t gcd(
  uint32_t m, uint32_t n
) {
  uint32_t x, y;
  x = m;
  y = n;
  while (x != y) {
    if (x > y)
      x = x - y;
    else
      y = y - x;
  }
  return x;
}
```



主要问题：画成流程图或状态转移图

（12 人，44.4%）

符号执行：题解

- 2. 使用符号执行的方法找到**达到分支和语句覆盖的**路径，写出行号表示的路径，以及用符号值表示的路径约束

```
1  uint32_t gcd(
2    uint32_t m, uint32_t n
3  ) {
4    uint32_t x, y;
5    x = m;
6    y = n;
7    while (x != y) {
8      if (x > y)
9        x = x - y;
10     else
11       y = y - x;
12   }
13   return x;
14 }
```

- 3. 针对以上路径分别给出一组可行输入数据（测试用例）

路径 #1：5→6→7→8→9→7→8→11→7→13

路径约束：

$m \neq n$

$m > n$

$m - n \neq n \Leftrightarrow m \neq 2n$

$m - n > n \Leftrightarrow m > 2n$

$m - n = 2n - m \Leftrightarrow 2m = 3n$

可行输入：m=3, n=2

符号执行：题解

- 2. 使用符号执行的方法找到**达到分支和语句覆盖的**路径，写出行号表示的路径，以及用符号值表示的路径约束

```
1   uint32_t gcd(
2     uint32_t m, uint32_t n
3   ) {
4     uint32_t x, y;
5     x = m;
6     y = n;
7     while (x != y) {
8       if (x > y)
9         x = x - y;
10      else
11        y = y - x;
12    }
13    return x;
14  }
```

- 3. 针对以上路径分别给出一组可行输入数据（测试用例）

路径 #2：5→6→7→8→11→7→8→9→7→13

路径约束：

$m \neq n$

$m > n$

$m \neq n - m \Leftrightarrow 2m \neq n$

$m > n - m \Leftrightarrow 2m > n$

$m - (n - m) = n - m \Leftrightarrow 3m = 2n$

可行输入：m=2, n=3

符号执行：题解

- 2. 使用符号执行的方法找到**达到分支和语句覆盖的**路径，写出行号表示的路径，以及用符号值表示的路径约束

```
1   uint32_t gcd(
2     uint32_t m, uint32_t n
3   ) {
4     uint32_t x, y;
5     x = m;
6     y = n;
7     while (x != y) {
8       if (x > y)
9         x = x - y;
10      else
11        y = y - x;
12    }
13    return x;
14  }
```

- 3. 针对以上路径分别给出一组可行输入数据（测试用例）

主要问题：

①找到的路径**没有达到**分支和语句覆盖

（12 人，44.4%）

②路径到达第 13 行之前**没有回到**第 7 行的 while

路径 #1：5→6→7→8→9→7→8→11→**7**→13

路径 #2：5→6→7→8→11→7→8→9→**7**→13

（6 人，22.2%）

③约束的字母**没有用参数的 m 和 n 表示（用的 x 和 y）**

（4 人，14.8%）

国科大　ISCAS

符号执行：题解

- 4. 仔细观察程序结构和 m、n 的取值范围（$m \geq 0, n \geq 0$），分析是否存在符号执行无法发现的缺陷？结合符号执行的特点简要阐述。

```
1  uint32_t gcd(
2    uint32_t m, uint32_t n
3  ) {
4    uint32_t x, y;
5    x = m;
6    y = n;
7    while (x != y) {
8      if (x > y)
9        x = x - y;
10     else
11       y = y - x;
12   }
13   return x;
14 }
```

路径 #1：5→6→7→8→9→7→8→11→**7**→13

路径 #2：5→6→7→8→11→7→8→9→**7**→13

通过符号执行，以上两条路径已经达到分支和语句覆盖，但不能发现无限循环问题（**14 人，51.9%**）：

当 $x = 0$, $y > 0$ 或 $x > 0, y = 0$ 时，

程序将无法终止！

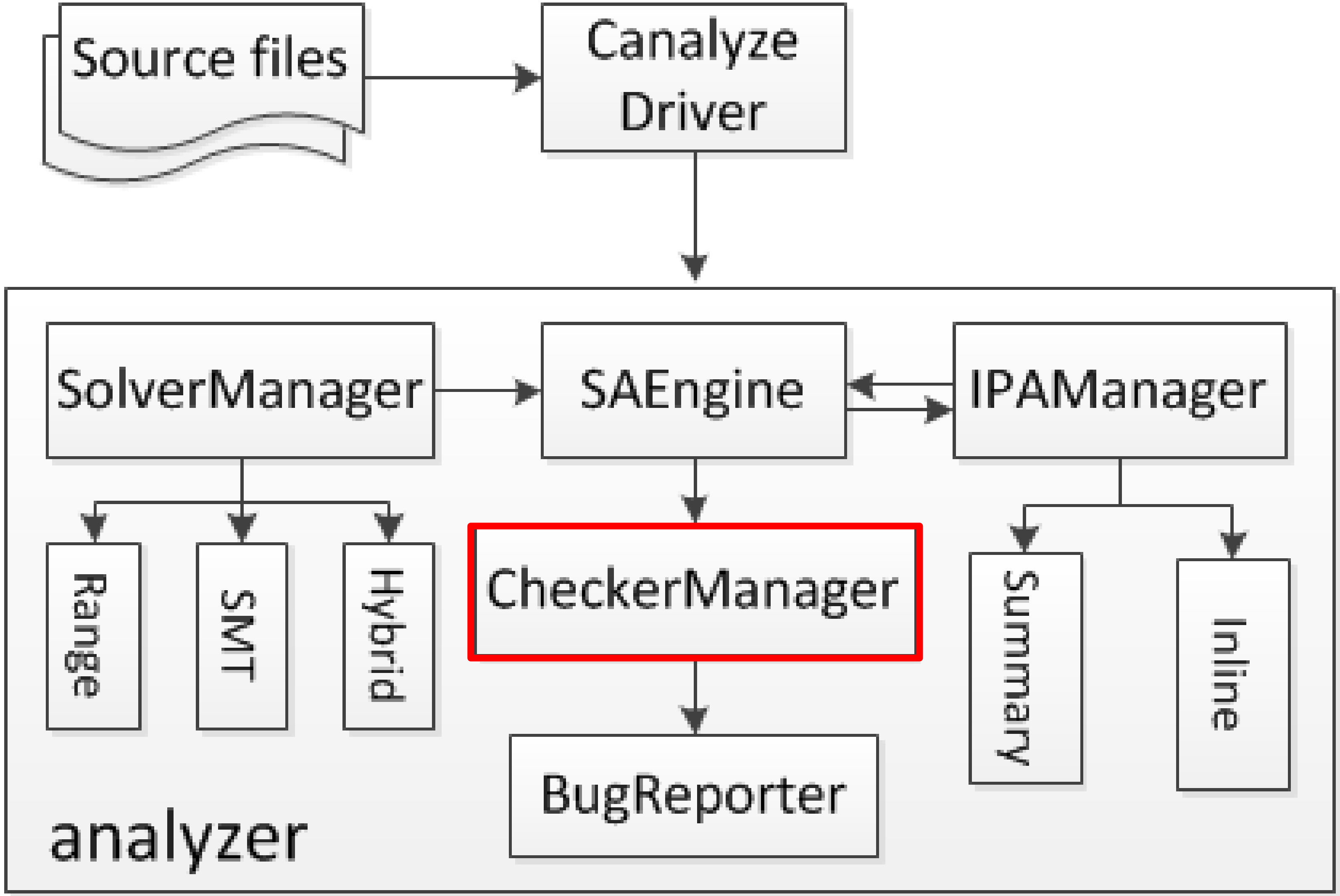
（注：uint32_t 类型为无符号整型，因此不存在 < 0 的情形）

符号执行：小结

反观本次课堂作业，大家需要强化以下方面的知识：

1. 辨析【控制流图】、【流程图】和【状态转移图】等的区别
2. 进一步理解和巩固【圈复杂度】的计算方法
3. 进一步理解【循环展开】的概念和做法，理解通过符号执行【收集路径约束】、【寻找全覆盖路径】等的方式方法
4. 注意区分【函数参数】和【局部变量】之于【路径约束】的关系

基于符号执行的静态分析



插入检查点[ZJ VSTTE2005]

```
int main () {
  int i;
  int x[3], y[2];
  x[0] = 1;
  for (i=0; i<3; i++)
     x[i] = i-1;
  for (i=1; i<3; i++)
     y[x[i]] = i;
}
```

Path 1:

```
x[0] = 1;      i = 0;
@(i < 3);       x[i] = i-1;  i = i+1;
@(i < 3);       x[i] = i-1;  i = i+1;
@(i < 3);       x[i] = i-1;  i = i+1;
@!(i < 3);
i = 1;
@(i < 3);
@(x[i] < 0); // assertion
```

Path 2:

```
...
@(x[i] >= 2);
```

国科大 ISCAS

基于路径分析的死循环检测

- **首先假设一个最简单的情况，要检查一个循环：**

```
while(C){
    P
}
```

  - 其中**C**为循环条件，**P**为循环体，假设循环体只有一条路径且没有**break**，这条路径用P_0表示。
  - 若循环不是无限循环，那么就意味着会在某次循环之后，循环条件**C**将会不满足。这次循环就是最后一次循环。

Jian Zhang: A Path-Based Approach to the Detection of Infinite Looping. APAQS 2001: 88-96

最后一次循环

- **我们关注最后一次循环：**
  - 循环开始时，循环条件C必然满足
  - 执行P_0，
  - 满足了!C的条件，跳出循环
- **于是我们定义一个“扩展路径” $C;P_0;!C$ 它表示了最后一次循环的情况。**
- **对扩展路径检查可行性，**
  - 若可行，那么说明循环可能会跳出。
  - 否则必然无限循环（循环的充分条件）

```
sum = 0;
i = 4;
while (i >= 0) {
        sum = sum + a[i];
        i= i + 1 ;
}
```

扩展路径

```
@ i >= 0;
sum = sum + a[il;
i = i+1;
@ i < 0;
```

更多情况

- **若循环体P有多条路径，用P_0,P_1,P_2…表示**

  对每一条P_j，都对扩展路径C;P_j;!C做可行性判断，若都不满足才认为一定无限循环。

- **若某条路径以break或return结尾**

  这就说明这条路径执行结束后，无需满足!C的条件即可跳出循环，则它的扩展路径为C;P_j，无需加上!C

```
int i;
i = 0;
while (i < 5) {
    if (i == 4)
        i = 0;
    i= i + 1 ;
}
```

```
P1:

int i;
@ i < 5;
@ i == 4;
i = 0;
i = i+1;
@ i >= 5;
```

```
P2:

int i;
@i < 5;
@ i != 4;
@ i >= 5;
i = i+1;
```

国科大 ISCAS

对GCD程序的分析

```
x = m; //m >= 0
y = n; //n >= 0
while (x != y) {
  if (x > y)
    x = x - y;
  else
    y = y - x;
}
```

一条扩展路径

```
@ (x != y)
@ (x > y)
    x = x - y;
@ (x == y)
```

符号执行结果

x == 2y, x > y

➔ x == 2y, x > 0, y >0

如果在循环之前加入前置条件

y <=0 || x<= 0

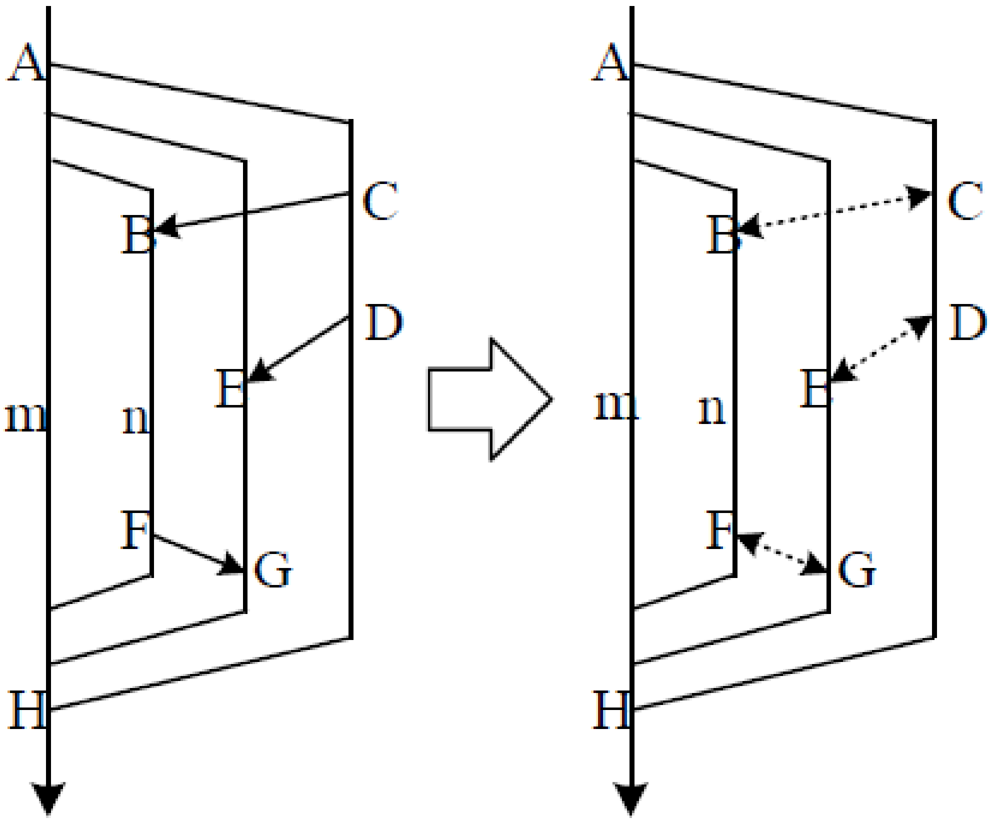
- **基于路径分析方法**
  - 仅为充分条件
  - 难以检测复杂的情况
- **更多关于无限循环的研究**

基于路径分析的死循环检测，计算机学报，2009

Large-Scale Analysis of Non-Termination Bugs in Real-World OSS Projects，ESEC/FSE ’22

- 多线程程序的符号执行

Interleaving



(a) 一条交织　　(b) 交织空间

国科大　ISCAS

Happen-Before规则

- 操作A happen-before操作B时，我们其实是在说在发生操作B之前，操作A对内存施加的影响能够被观测到
  - 同一个线程中，书写在前面的操作happen-before书写在后面的操作。
  - 对锁的unlock操作happen-before后续的对同一个锁的lock操作。
  - 如果操作A happen-before操作B，操作B happen-before操作C，那么操作A happen-before操作C。这条规则也称为传递规则。
-

采用符号执行的方法进行编码

$\varphi = \varphi_{sm} \wedge \varphi_{io} \wedge \varphi_{rw}$，其中

- φ_{sm} 是程序语义约束描述了线程内的数据流关系，以及线程的行为；
- φ_{io} 是语句时序约束限定了执行语句读写关系约束（指令）在线程内与线程间应该满足的时序关系；
- φ_{rw} 是读写关系约束描述了共享变量的读操作与写操作的匹配关系。

程序语义约束

```
线程 0:
0:   x=3, y=1; // intial
1:   create (1) ;
2:   create (2) ;
3:   join (1) ;
4:   join (2) ;
```

```
线程 1:
5:   a=x;
6:   b=y;
7:   lock(m);
8:   x=a+2;
9:   y=b+3;
10:  unlock(m);
```

```
线程 2:
11:  lock(m);
12:  x++;
13:  y++;
14:  unlock(m);
```

$Thread\,0$

$0 : x_w^0 = 3, y_w^0 = 1;$

$1 : create(1);$

$2 : create(2);$

$3 : join(1);$

$4 : join(2);$

$Thread\,1$

$5 : a^0 = x_r^0;$

$6 : b^0 = y_r^0;$

$7 : lock(m);$

$8 : x_w^1 = a^0 + 2;$

$9 : y_w^1 = b^0 + 3;$

$10 : unlock(m);$

$Thread\,2$

$11 : lock(m);$

$12 : x_w^2 = x_r^1 + 1;$

$13 : y_w^2 = y_r^1 + 1;$

$14 : unlock(m);$

$$x_w^0 = 3 \wedge y_w^0 = 1 \wedge$$
$$a^0 = x_r^0 \wedge b^0 = y_r^0 \wedge$$
$$x_w^1 = a^0 + 2 \wedge y_w^1 = b^0 + 3 \wedge$$
$$x_w^2 = x_r^1 + 1 \wedge y_w^2 = y_r^1 + 1$$

语句时序约束

$Thread\,0$
$0: x_w^0 = 3, y_w^0 = 1;$
$1: create(1);$
$2: create(2);$
$3: join(1);$
$4: join(2);$

$Thread\,1$
$5: a^0 = x_r^0;$
$6: b^0 = y_r^0;$
$7: lock(m);$
$8: x_w^1 = a^0 + 2;$
$9: y_w^1 = b^0 + 3;$
$10: unlock(m);$

$Thread\,2$
$11: lock(m);$
$12: x_w^2 = x_r^1 + 1;$
$13: y_w^2 = y_r^1 + 1;$
$14: unlock(m);$

$$(o_1 < o_2 < o_3 < o_4) \wedge$$
$$(o_5 < o_6 < o_7 < o_8 < o_9 < o_{10}) \wedge$$
$$(o_{11} < o_{12} < o_{13} < o_{14})$$

线程内

$$o_1 < o_5 \wedge o_2 < o_{11}$$
$$o_{10} < o_3 \wedge o_{14} < o_4$$

线程创建与终止

$$o_{10} < o_{11} \vee o_{14} < o_7$$

锁

读写约束

Thread 0	Thread 1	Thread 2
$0: x_w^0 = 3, y_w^0 = 1;$	$5: a^0 = x_r^0;$	$11: lock(m);$
$1: create(1);$	$6: b^0 = y_r^0;$	$12: x_w^2 = x_r^1 + 1;$
$2: create(2);$	$7: lock(m);$	$13: y_w^2 = y_r^1 + 1;$
$3: join(1);$	$8: x_w^1 = a^0 + 2;$	$14: unlock(m);$
$4: join(2);$	$9: y_w^1 = b^0 + 3;$	
	$10: unlock(m);$	

$$\{x_r^0 = x_w^0 \wedge o_0 < o_5 \wedge (o_{12} < o_0 \vee o_{12} > o_5)\} \vee$$
$$\{x_r^0 = x_w^2 \wedge o_{12} < o_5 \wedge (o_0 < o_{12} \vee o_0 > o_5)\}$$

- 约束求解简介

约束求解

- 约束满足问题

Constraint Satisfaction Problem (CSP)

未知量：$x_1, x_2, \ldots$

给定 值域： $D_1, D_2, \ldots$

约束条件：$c_1, c_2, \ldots$

找出变量的值 $x_i \in D_i$ 使得每个c_j 成立。

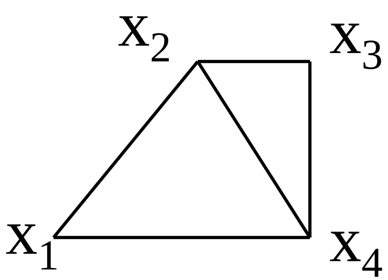
约束条件：等式、不等式、逻辑表达式、…

具体的约束满足问题

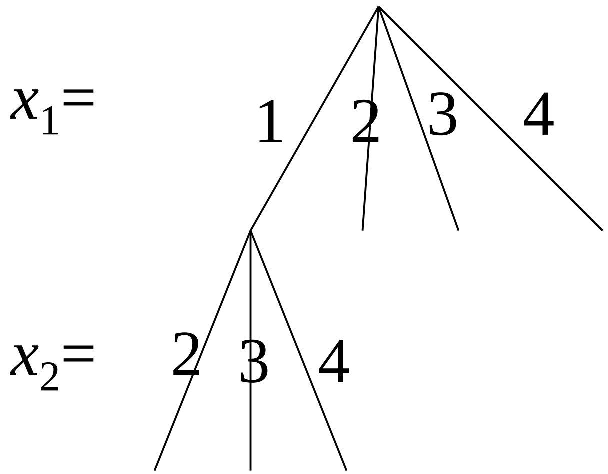
- **图的顶点染色问题 (Graph coloring)**

$$x_i : 1..n;$$

$$x_i \neq x_j \quad \text{(if vertices } i \text{ and } j \text{ are connected)}$$



回溯算法（Backtracking）



搜索树

- **搜索策略：**
  在有多个变量未取值时，先选哪个进行试探？
  在每个节点，如何通过有效推理及时发现矛盾？
- **推理规则**

SAT

- 给定的布尔/命题逻辑公式，判断它是否可满足？

$(p \vee \neg q) \wedge (q \vee \neg p)$

可满足：p = **TRUE**, q = **TRUE**. ***model***

- 命题逻辑—可判定
- 很多问题可转化成 SAT. 但它是 NP难。

SAT

- 合取
  - and ∧ &&
- 析取
  - Or ∨ ||
- 其他二元运算
  - → ↔
- CNF 合取范式
- DNF 析取范式
  - 子句 clause

DIMACS input format

例. (x1 or x3 or not x4); x4; (x2 or not x3)

```
p cnf 4 3
1 3 -4 0
4 0
2 -3 0
```

CNF

- **将一般的公式转成CNF**
  - 采用De Morgan律
    – 子句数量爆炸
  - 引入额外的变量

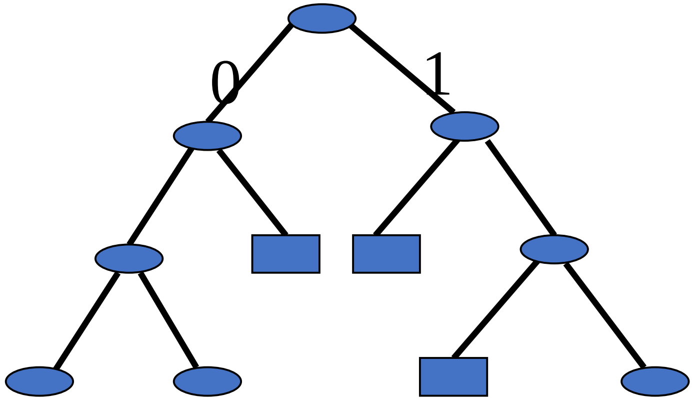
J M. Wilson: Compact normal forms in propositional logic and integer programming formulations. Comput. Oper. Res. 17(3): 309-314 (1990)

文献 [110] 提出了一种引入额外的布尔变量来避免组合爆炸的翻译方法. 令 $c_1, \dots, c_n$ 表示约束, 我们用 $C(c_i)$ 表示由 c_i 翻译而来的 SAT 子句. 我们引入 CNF 中的文字 c_i' 表示 c_i. 对于一个公式 S, 可以采用如下的翻译规则.

1. 如果 S 是一个文字, 那么 $C(S)$ 为空并且 $S' = S$.
2. 如果 S 是一个合取式 $c_1 \wedge \cdots \wedge c_n$, 那么 $C(S)$ 包含 $\bigcup_{i=1}^{n} C(c_i)$ 以及如下两类子句:
   $$\neg c_1' \vee \cdots \vee \neg c_n' \vee S' \text{ 以及 } c_i' \vee \neg S' \ (1 \le i \le n);$$
3. 如果 S 是一个析取式 $c_1 \vee \cdots \vee c_n$, $C(S)$ 包含 $\bigcup_{i=1}^{n} C(c_i)$ 以及如下类型子句:
   $$c_1' \vee \cdots \vee c_n' \vee \neg S' \text{ 以及 } \neg c_i' \vee S' \ (1 \le i \le n);$$
   这一规则常被用于翻译测试覆盖.
4. 如果 $S = \neg c$, 那么 $C(S) = C(c)$ 并且 $S' = \neg c'$.

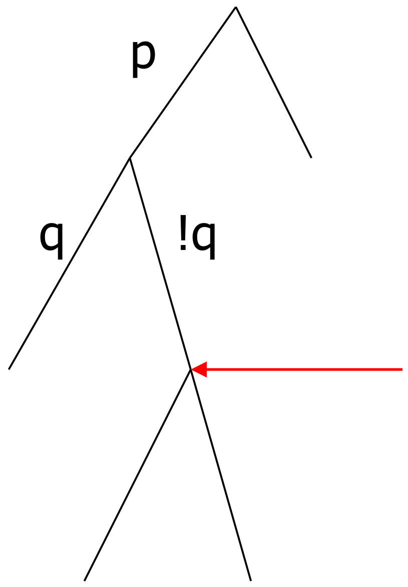
SAT solving

- 选取变量的策略；快速有效的推导
- DPLL 算法
  - 重言式规则
    $S = \{p \vee \neg p,\ q \vee r\}$
  - 单子句规则
    $S = \{p, p \vee \neg q, \neg p \vee r\}$
  - 纯文字规则
    $S = \{p \vee q \vee \neg r \vee s,$
    $\neg p \vee q \vee \neg r \vee \neg s,$
    $\neg p \vee \neg q\}$
  - 分裂规则 二叉搜索树
    $S = \{A_1 \vee l, \ldots, A_m \vee l, B_1 \vee \neg l, \ldots, B_n \vee \neg l, C_1 \ldots, C_t\}$



- 求解工具: zChaff, minisat, …, sat4j
-

搜索过程



求解$\{p \vee q, \neg p \vee \neg q\}$
其中：
p: (x>3)
q: (2*x > 5)

Check the feasibility of:
x>3; 2*x <= 5

Linear programming:
lp_solve

Satisfiability modulo theories (SMT)

- **多种理论：bit vectors, arrays, …**

SMT solvers:

- **CVC3/CVC4 (NewYork, Iowa)**
- **Yices (SRI)**
- **Z3 (Microsoft)**

Ordering Constraint

$$(o_1 < o_2 < o_3 < o_4) \wedge$$
$$(o_5 < o_6 < o_7 < o_8 < o_9 < o_{10}) \wedge$$
$$(o_{11} < o_{12} < o_{13} < o_{14})$$

$$o_1 < o_5 \wedge o_2 < o_{11}$$
$$o_{10} < o_3 \wedge o_{14} < o_4$$

$$o_{10} < o_{11} \vee o_{14} < o_7$$

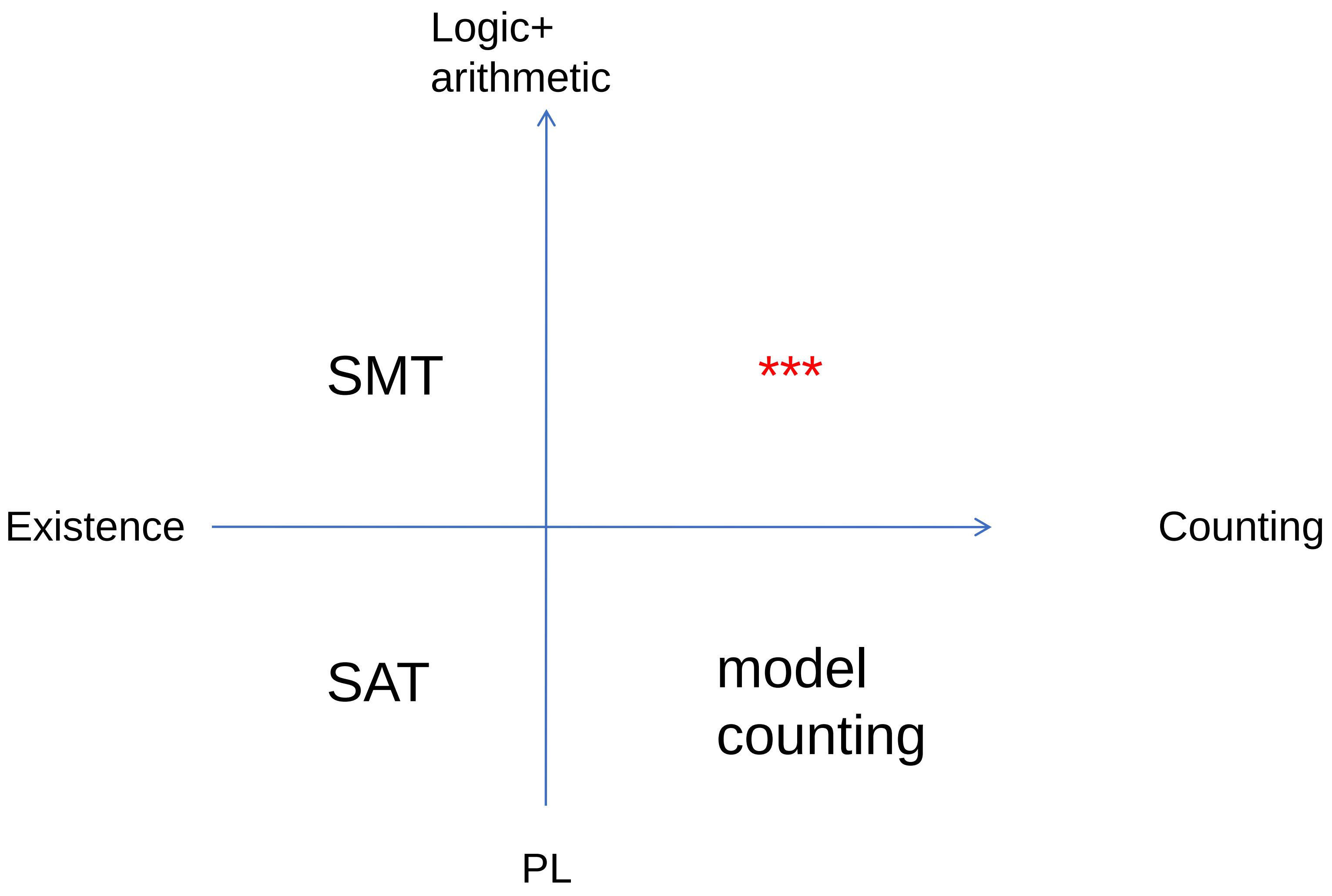
Input Format: SMT-LIB 2

```
(declare-const a Int)
(declare-const b Int)
(declare-const c Int)
(assert (> a (+ b 2)))
(assert (= a (+ (* 4 c) 1)))
(check-sat)
(get-model)

(declare-fun x () Real)
```

- (<= (+ x 3) (* 2 x))

SMT求解问题的扩展 (I)



SMT约束的计数问题

	求解问题	计数问题
逻辑+理论（如算术、位向量）	SMT	#SMT
命题逻辑	SAT	Model Counting
	NP	#P even for 2-SAT

Model counting

- 输入：命题逻辑公式
- 输出：解（model）的个数

- 例. (p OR q)

- AI的一个重要研究问题

计算（估算）解空间的大小/解的密度

- **Given an SMT formula (a set of constraints), compute the volume of its solution space (or its solution density).**

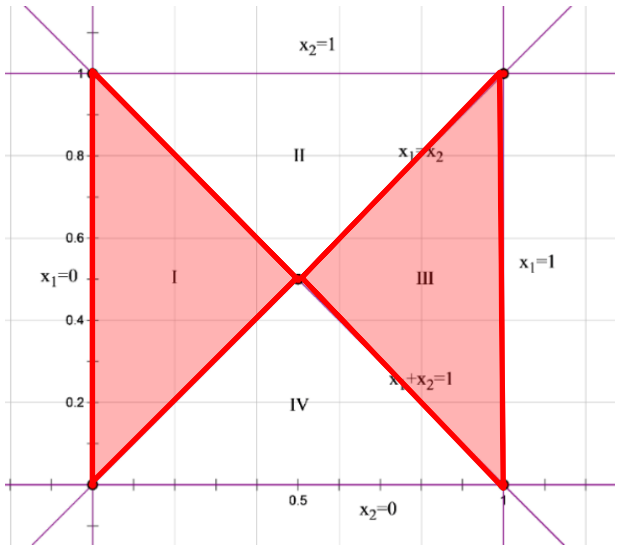
- **Example.** $\Phi :=$

$$(((y+3x<1) \rightarrow (30<y)) \vee (x \leq 60)) \wedge ((30<y) \rightarrow \neg(x>3) \wedge (x \leq 60))$$

We count the number of its solutions in some bounded space.

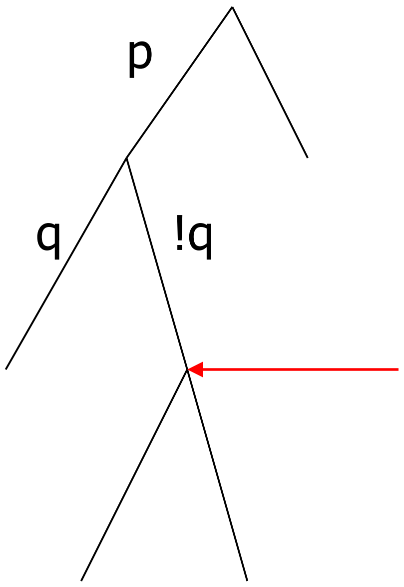
一个#SMT(LA)问题的例子

- $(x_1 < x_2 \vee x_1 + x_2 \geq 1) \wedge$
- $(b \vee x_1 < x_2 \vee x_1 + x_2 < 1) \wedge$
- $(x_1 \geq x_2 \vee x_1 + x_2 < 1) \wedge$
- $x_1 \leq 1 \wedge$
- $x_2 \leq 1 \wedge$
- $x_1 \geq 0 \wedge$
- $x_2 \geq 0$



$2 \times \mathrm{vol(I)} + \mathrm{vol(III)}$

Solution counting [MaLiuZhang 2009]



p: (x>3)
q: (2*x > 5)

•Check the #of solutions of:
x>3; 2*x <= 5

•Vol. computing for polytopes:
• **vinci**

SMT求解问题的扩展 (II)

逻辑与算术	SMT	??
线性算术约束（不等式）	LP, …	Linear programming

复杂约束下的优化问题

- **Linear Programming**

  $$\min.\ f(X)$$
  $$s.t.\ Ax <= b$$

- **SMT 优化**

  $$\min.\ f(X)$$
  s.t. **SMT约束**

一个小例子

min. x - y
subject to:
$(((y + 3x < 3) \rightarrow (30 < y)) \lor (x <= 60))$
$\land ((30 < y) \rightarrow \neg (x > 3) \land (x <= 60))$

局部最优 vs 全局优化

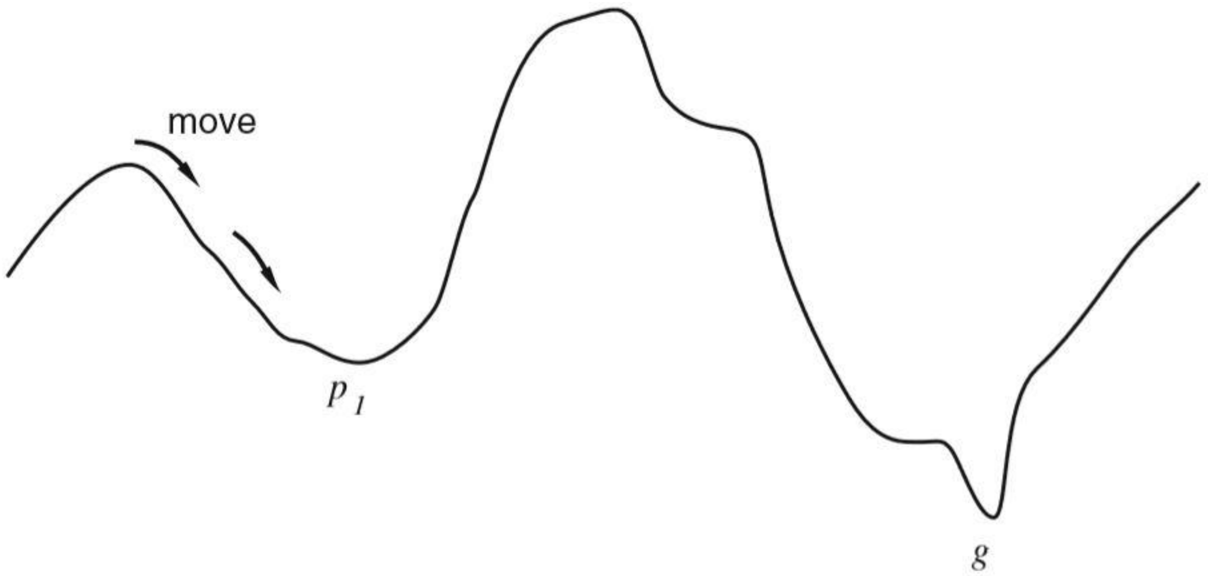


Fig. 5.1 Local minimum