软件分析

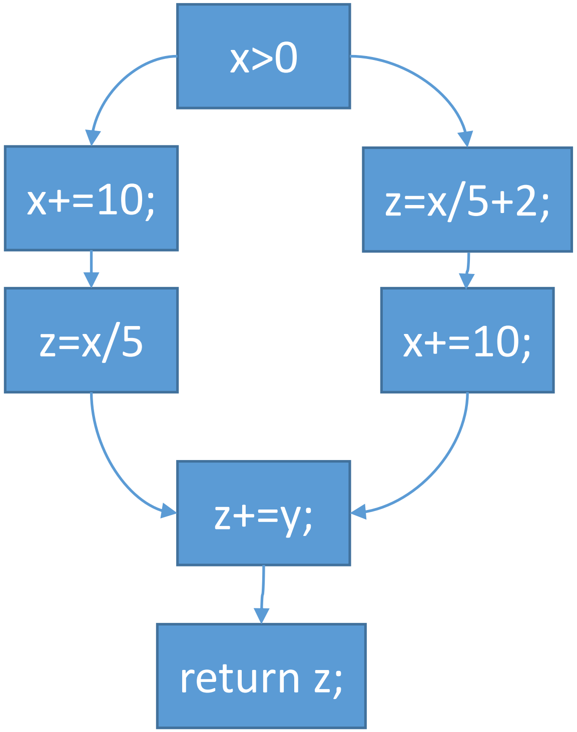
符号执行

熊英飞
北京大学

2016

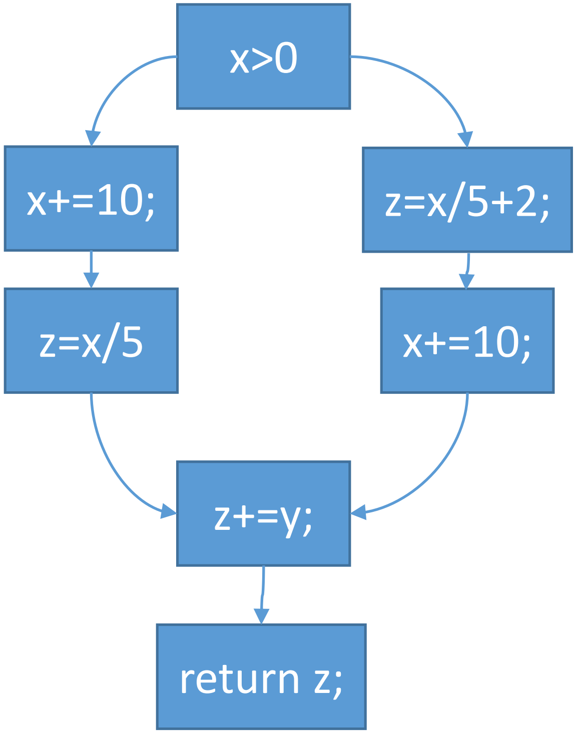
符号执行

```
• int main(x,y) {
•   if (x>0) {
•     x+=10;
•     z=x/5;
•   }
•   else {
•     z=x/5+2;
•     x+=10;
•   }
•   z+=y;
•   return z;
• }
```



符号执行

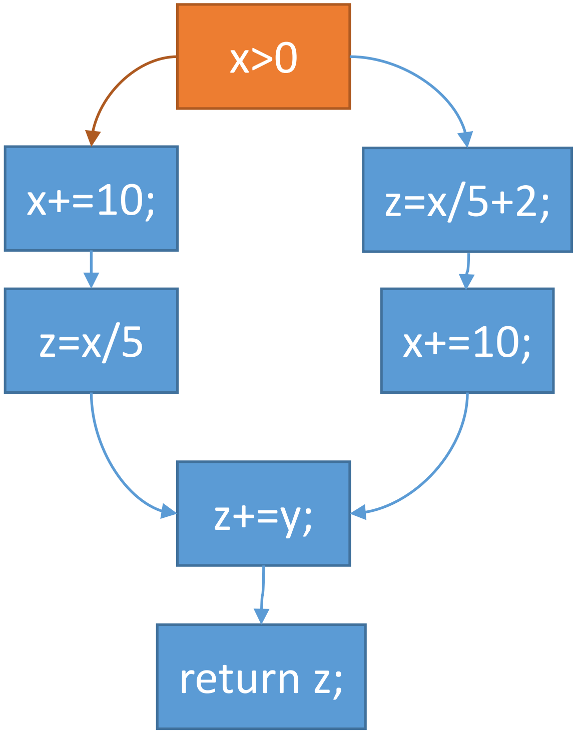
```
int main(x,y) {
  if (x>0) {
    x+=10;
    z=x/5;
  }
  else {
    z=x/5+2;
    x+=10;
  }
  z+=y;
  return z;
}
```



x=a
y=b
z=?

符号执行

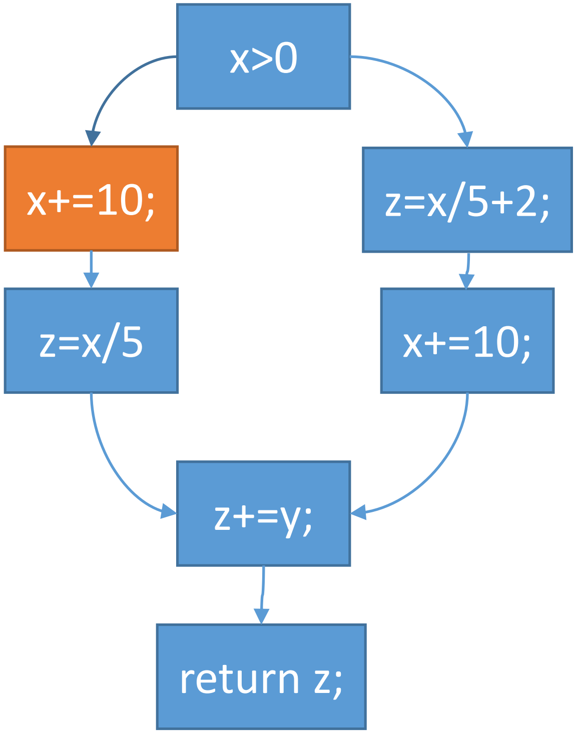
```
int main(x,y) {
  if (x>0) {
    x+=10;
    z=x/5;
  }
  else {
    z=x/5+2;
    x+=10;
  }
  z+=y;
  return z;
}
```



x=a
y=b
z=?
a>0

符号执行

```
int main(x,y) {
  if (x>0) {
    x+=10;
    z=x/5;
  }
  else {
    z=x/5+2;
    x+=10;
  }
  z+=y;
  return z;
}
```



x>0

x+=10;

z=x/5+2;

z=x/5

x+=10;

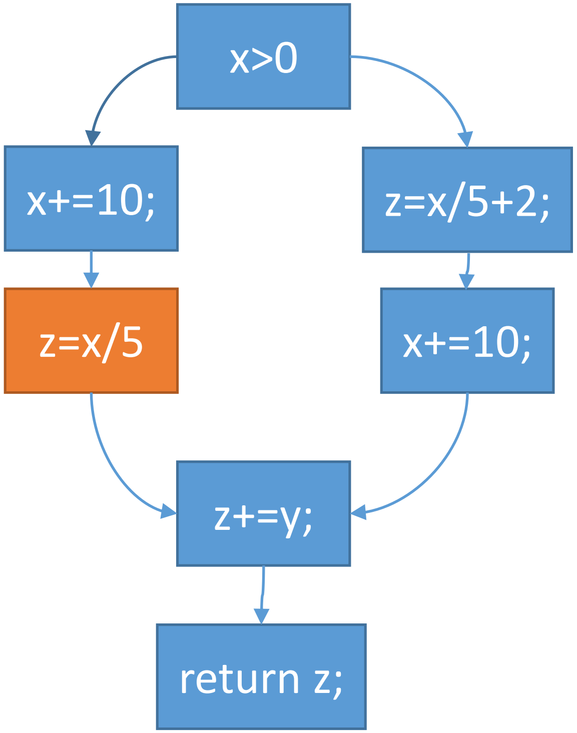
z+=y;

return z;

x=a+10
y=b
z=?
a>0

符号执行

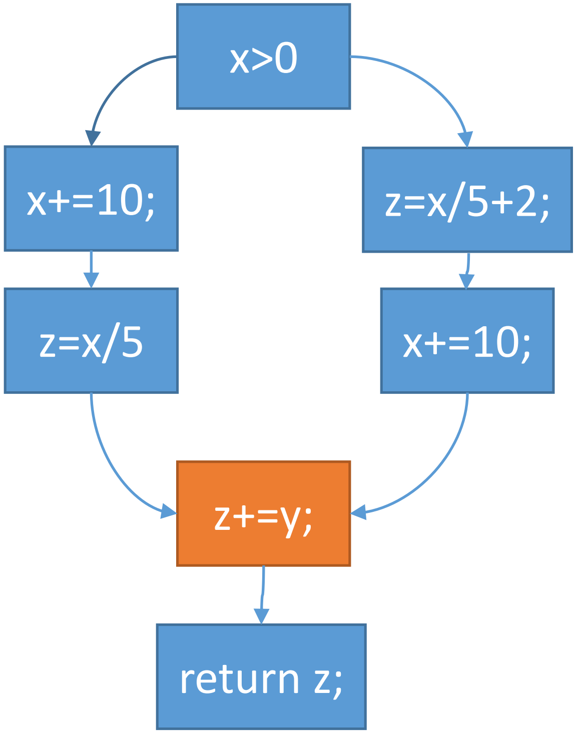
```
int main(x,y) {
  if (x>0) {
    x+=10;
    z=x/5;
  }
  else {
    z=x/5+2;
    x+=10;
  }
  z+=y;
  return z;
}
```



x=a+10
y=b
z=(a+10)/5
a>0

符号执行

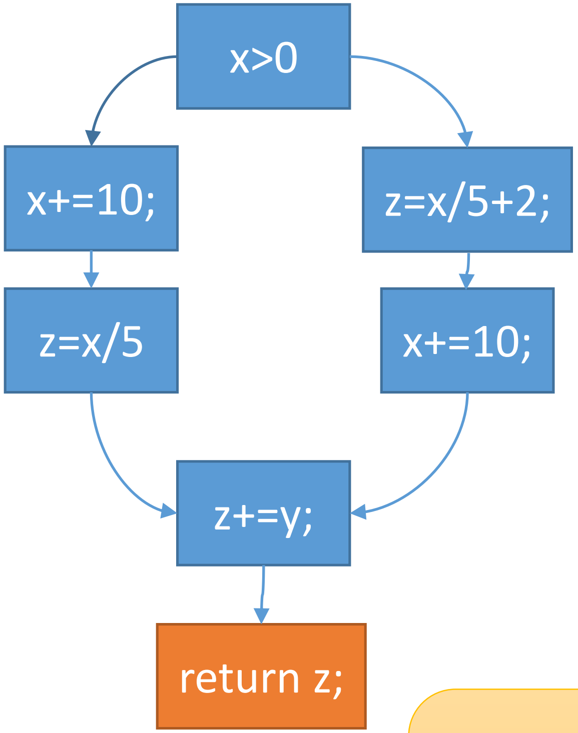
```
• int main(x,y) {
•   if (x>0) {
•     x+=10;
•     z=x/5;
•   }
•   else {
•     z=x/5+2;
•     x+=10;
•   }
•   z+=y;
•   return z;
• }
```



x=a+10
y=b
z=(a+10)/5+b
a>0

符号执行

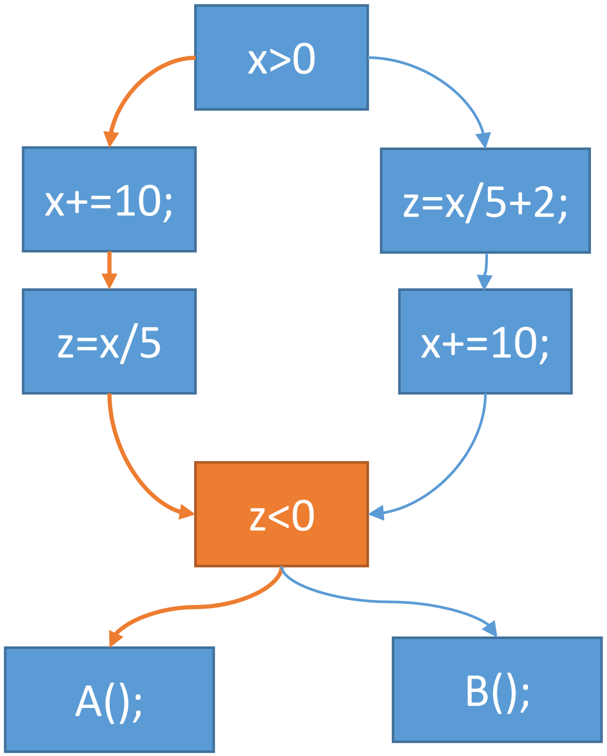
```
int main(x,y) {
  if (x>0) {
    x+=10;
    z=x/5;
  }
  else {
    z=x/5+2;
    x+=10;
  }
  z+=y;
  return z;
}
```



x=a+10
y=b
z=(a+10)/5+b
a>0

返回值为(a+10)/5+b
且a>0

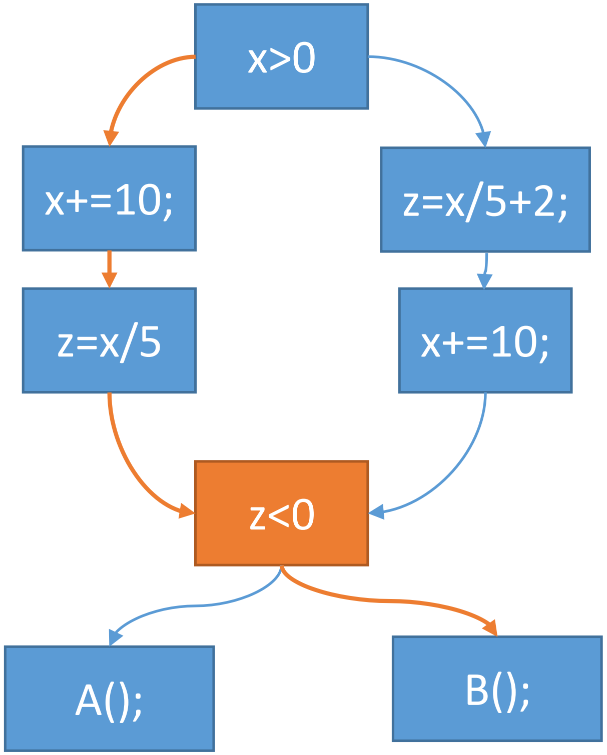
路径可行性



x=a+10
y=b
z=(a+10)/5
a>0

a>0/\ (a+10)/5<0不可满足

路径可行性



x=a+10
y=b
z=(a+10)/5
a>0

a>0/\ (a+10)/5<0不可满足

符号执行

- 程序的规约通常表示为前条件和后条件
  - 前条件：a>0, b>0
  - 后条件：return > 0
- 形成命题：
  - a>0 /\ b>0 => (a+10)/5+b > 0
  - 命题成立=逆命题不可满足
  - 用SMT Solver可求解
- 规约被违反=任意路径对应的命题不成立
- 规范被满足=所有路径对应的命题都成立
  - 通常做不到
  - 对于循环，遍历有限次

符号执行的应用

除0错误

```
if (i !=0)
   x = 3 / i;
else
   ERROR;
```

缓冲区溢出错误

```
if (0<=i  &&  i < a.length)
   a[i] = 4;
else
   ERROR;
```

约束求解失败的情况

- 形成了复杂条件
  - $x^5 + 3x^3 == y$
  - p->next->value == x
- 调用了系统调用
  - If (file.read()==x)

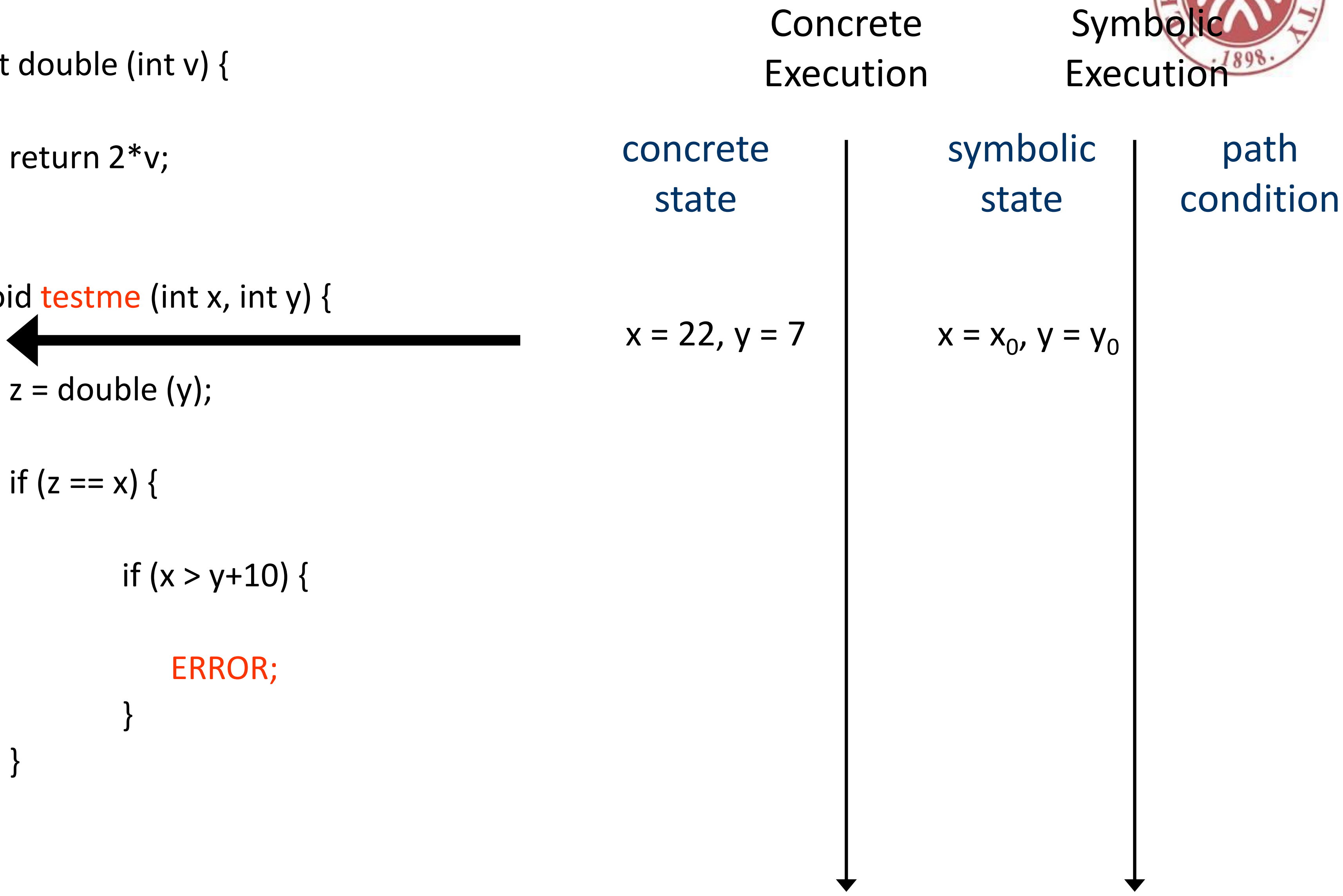
- 动态符号执行
  - 混合程序的真实执行和符号执行
  - 在约束求解无法进行的时候，用真实值代替符号值
    - 如果真实值x=10，则$x^5 + 3x^3 == y$变为103000==y，可满足

动态符号执行

- 动态符号执行主要用于生成测试输入
- 代表性工作：
  - Concolic Testing, Koushik Sen
    - 主要工具：CUTE
  - Execution-Generated Testing, Cristian Cadar
    - 主要工具：KLEE

Concolic Testing Approach

```
int double (int v) {

   return 2*v;
}

void testme (int x, int y) {

   z = double (y);

   if (z == x) {

         if (x > y+10) {

              ERROR;
         }
   }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

x = 22, y = 7

$x = x_0$, $y = y_0$

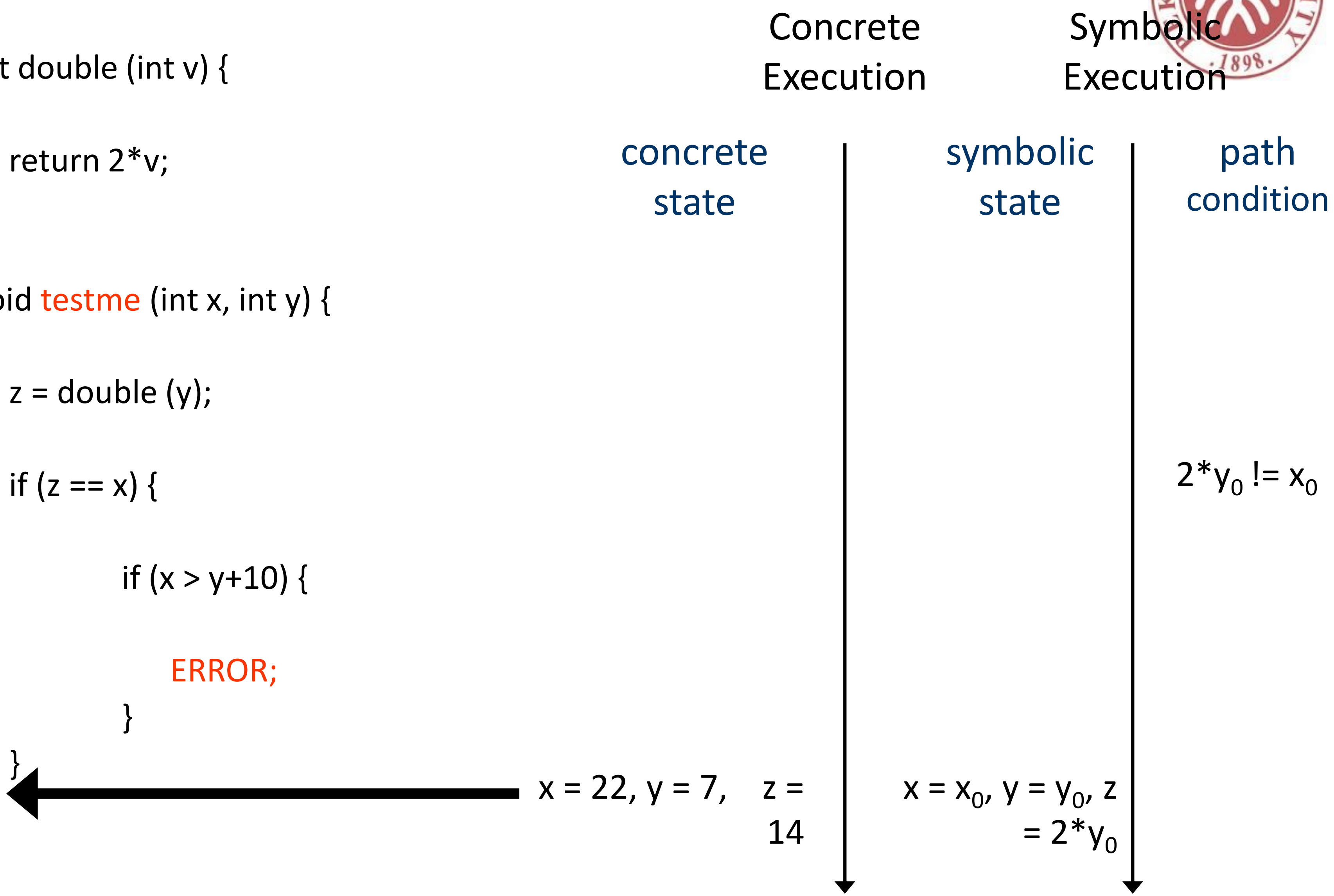
Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 22, y = 7, z = 14	$x = x_0$, $y = y_0$, $z = 2*y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

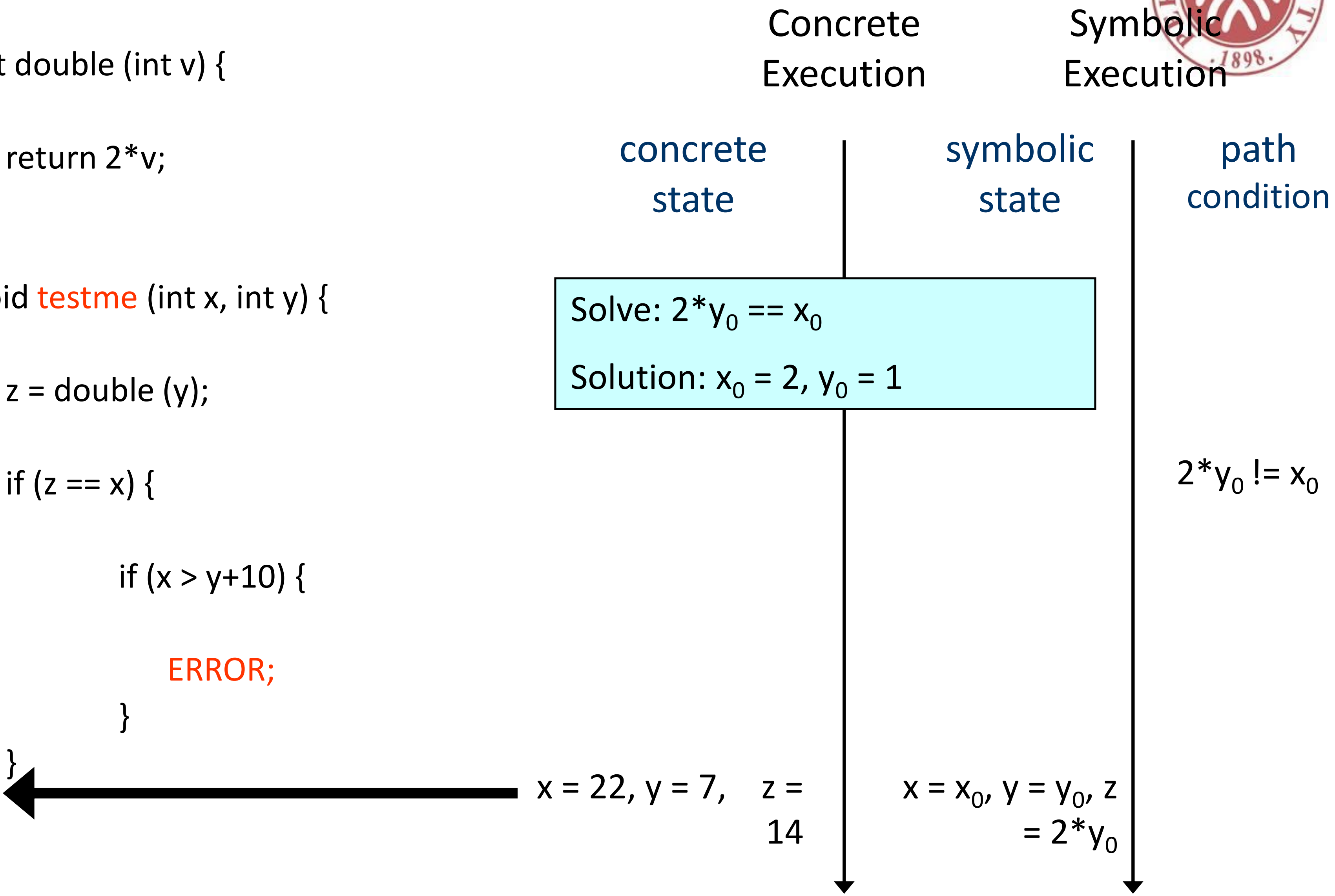
$2*y_0 != x_0$

x = 22, y = 7, z = 14

$x = x_0$, $y = y_0$, $z = 2*y_0$

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }
}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $2*y_0 == x_0$

Solution: $x_0 = 2$, $y_0 = 1$

$2*y_0 != x_0$

x = 22, y = 7, z = 14

$x = x_0$, $y = y_0$, $z = 2*y_0$

Concolic Testing Approach

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2, y = 1	$x = x_0$, $y = y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2, y = 1, z = 2	$x = x_0$, $y = y_0$, $z = 2*y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
		$2*y_0 == x_0$
x = 2, y = 1, z = 2	x = x_0, y = y_0, z = $2*y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                  ERROR;
            }
    }

}
```

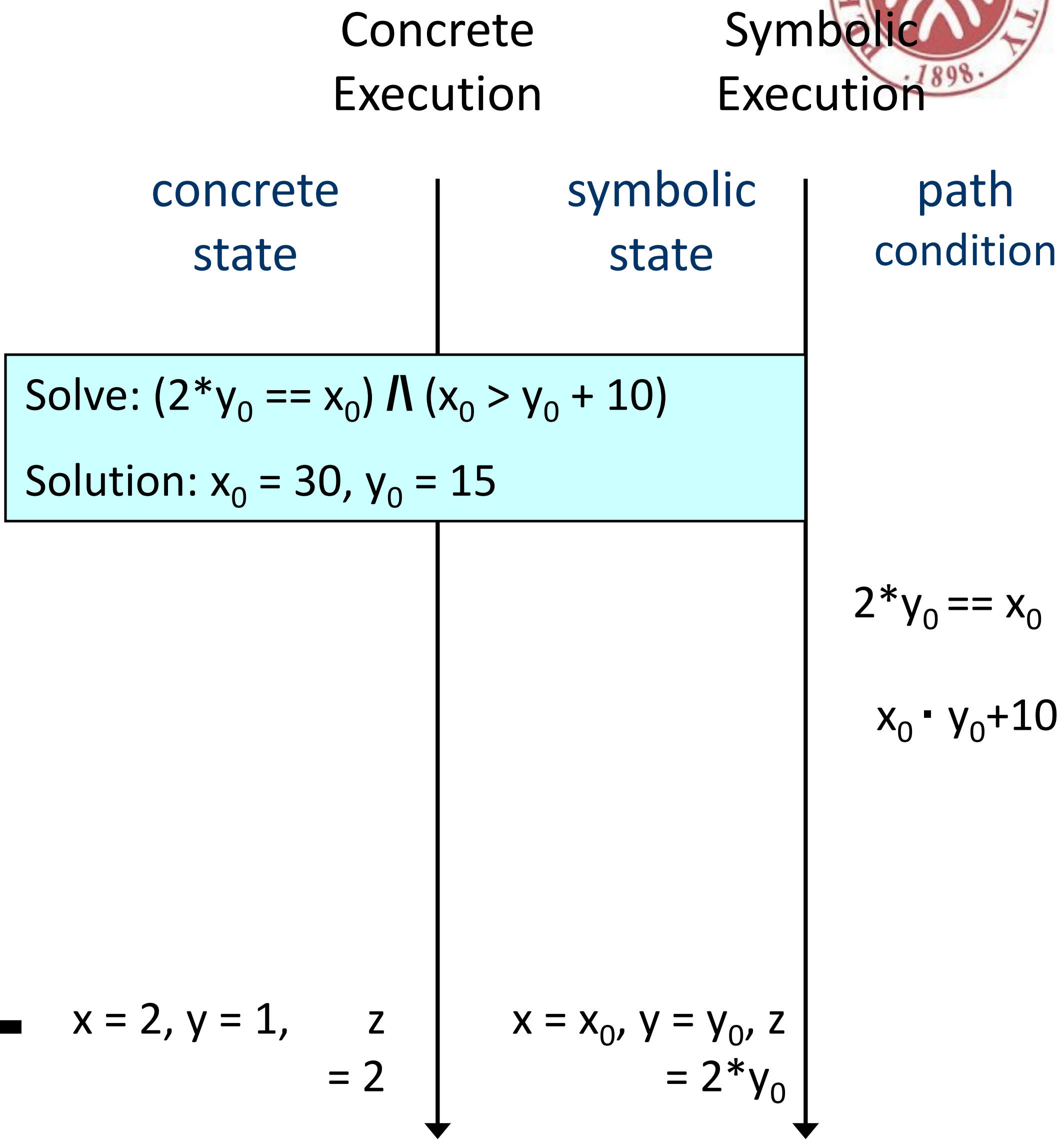
Concrete Execution

Symbolic Execution

concrete state	symbolic state	path condition
		$2*y_0 == x_0$
		x_0 ▪ y_0+10
x = 2, y = 1, z = 2	$x = x_0$, $y = y_0$, $z = 2*y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $(2*y_0 == x_0) \land (x_0 > y_0 + 10)$

Solution: $x_0 = 30$, $y_0 = 15$

$2*y_0 == x_0$

x_0 ▪ y_0+10

x = 2, y = 1, z = 2

$x = x_0$, $y = y_0$, $z = 2*y_0$

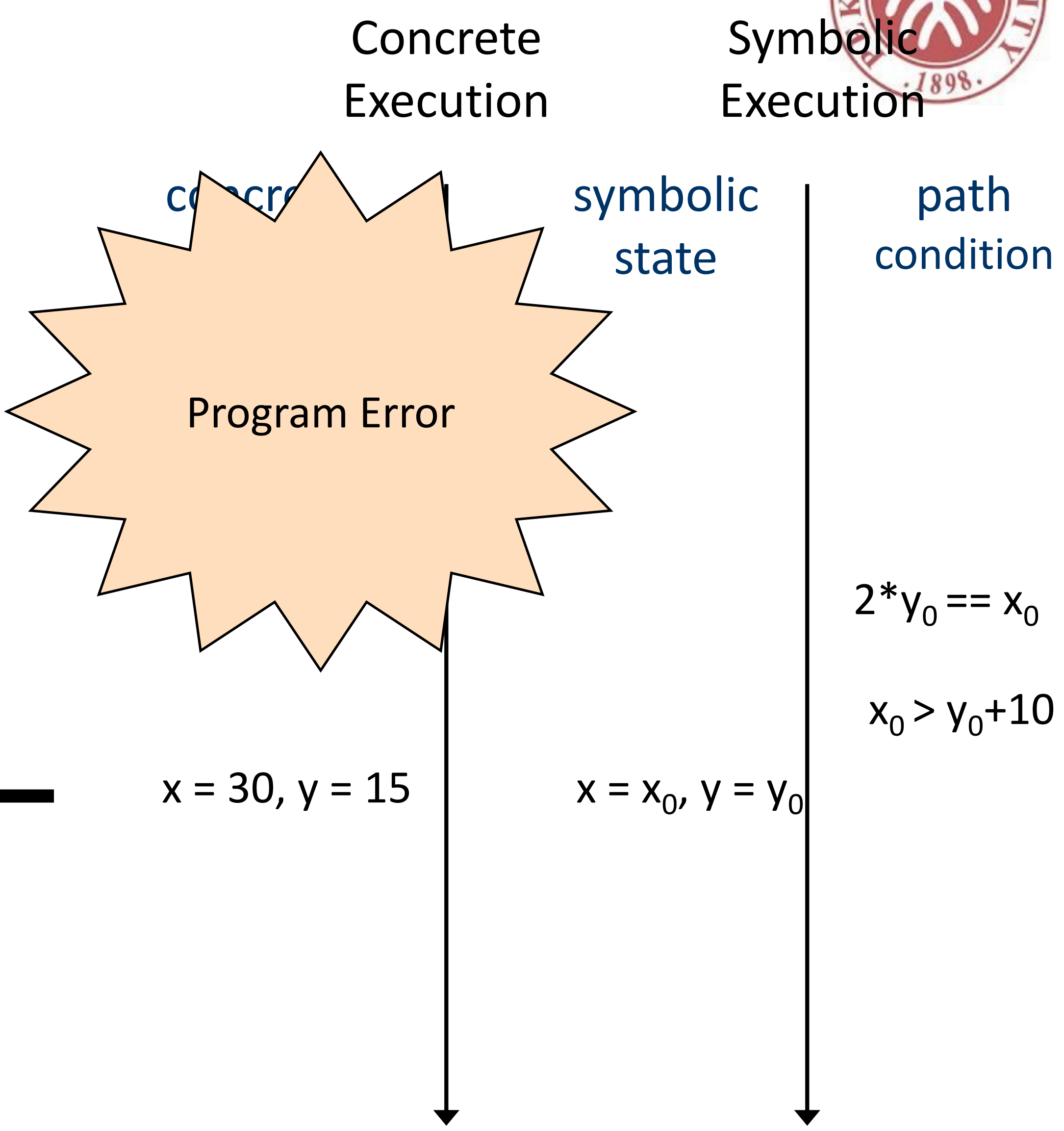
Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 30, y = 15	$x = x_0$, $y = y_0$	

Concolic Testing Approach

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```



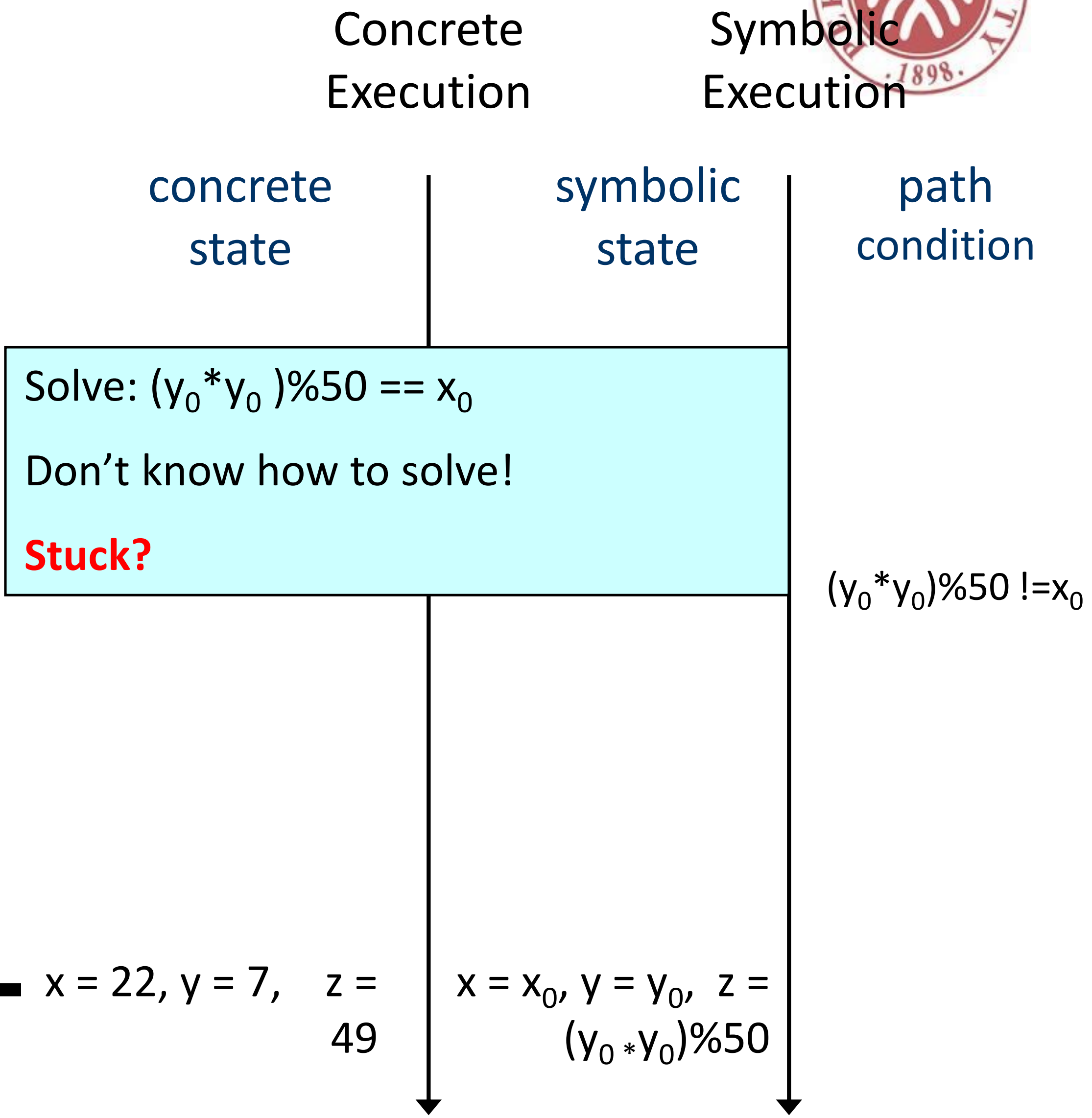
Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {

    z = foo (y);

    if (z == x) {

            if (x > y+10) {

                ERROR;
            }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 22, y = 7	$x = x_0$, $y = y_0$	

Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
            if (x > y+10) {
                ERROR;
            }
    }
}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $(y_0 * y_0)\%50 == x_0$

Don’t know how to solve!

Stuck?

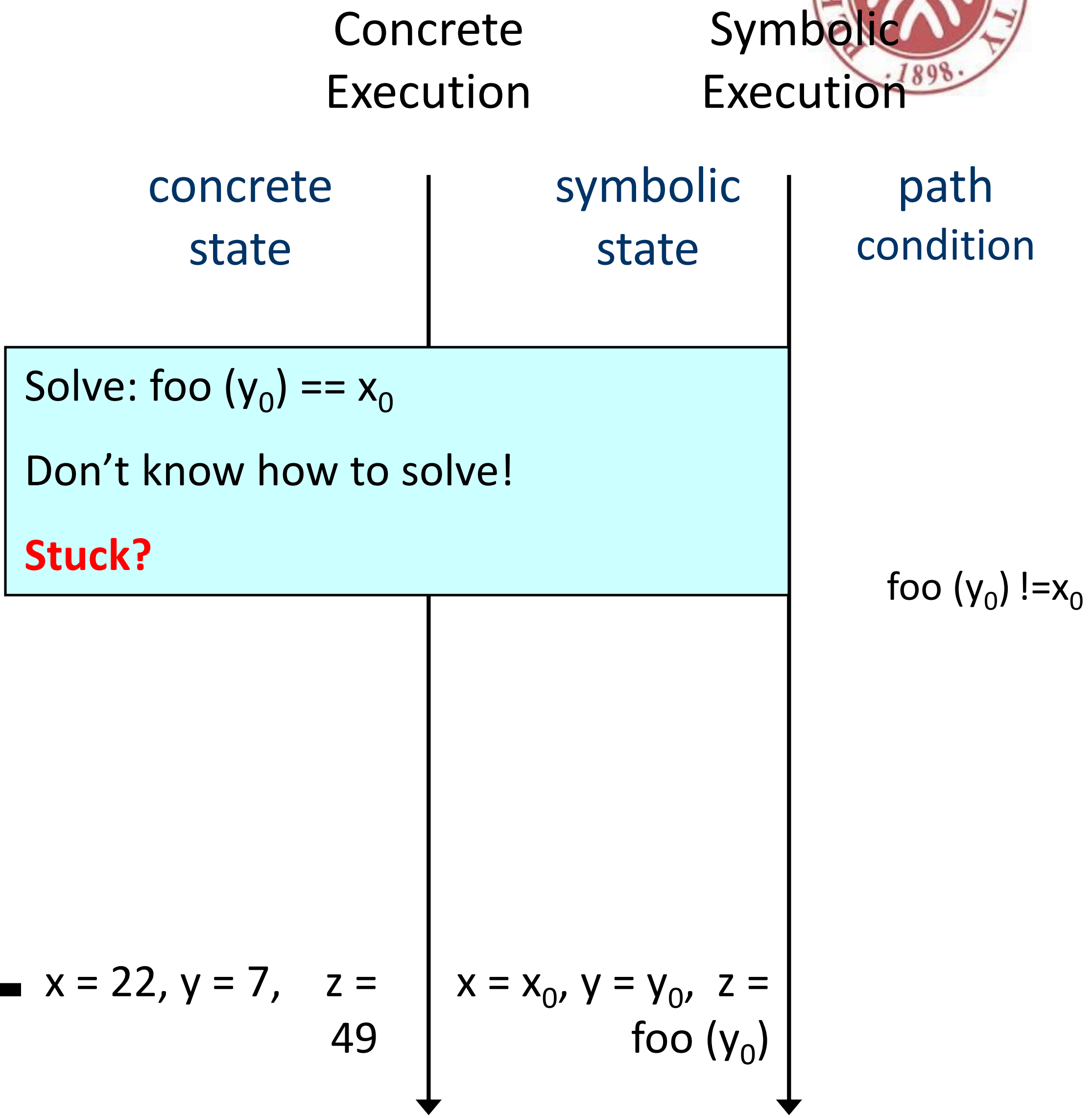
$(y_0 * y_0)\%50 \, != x_0$

x = 22, y = 7, z = 49

$x = x_0$, $y = y_0$, $z = (y_0 * y_0)\%50$

Novelty : Simultaneous Concrete and Symbolic Execution

```
void testme (int x, int y) {

    z = foo (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: foo (y_0) == x_0

Don’t know how to solve!

Stuck?

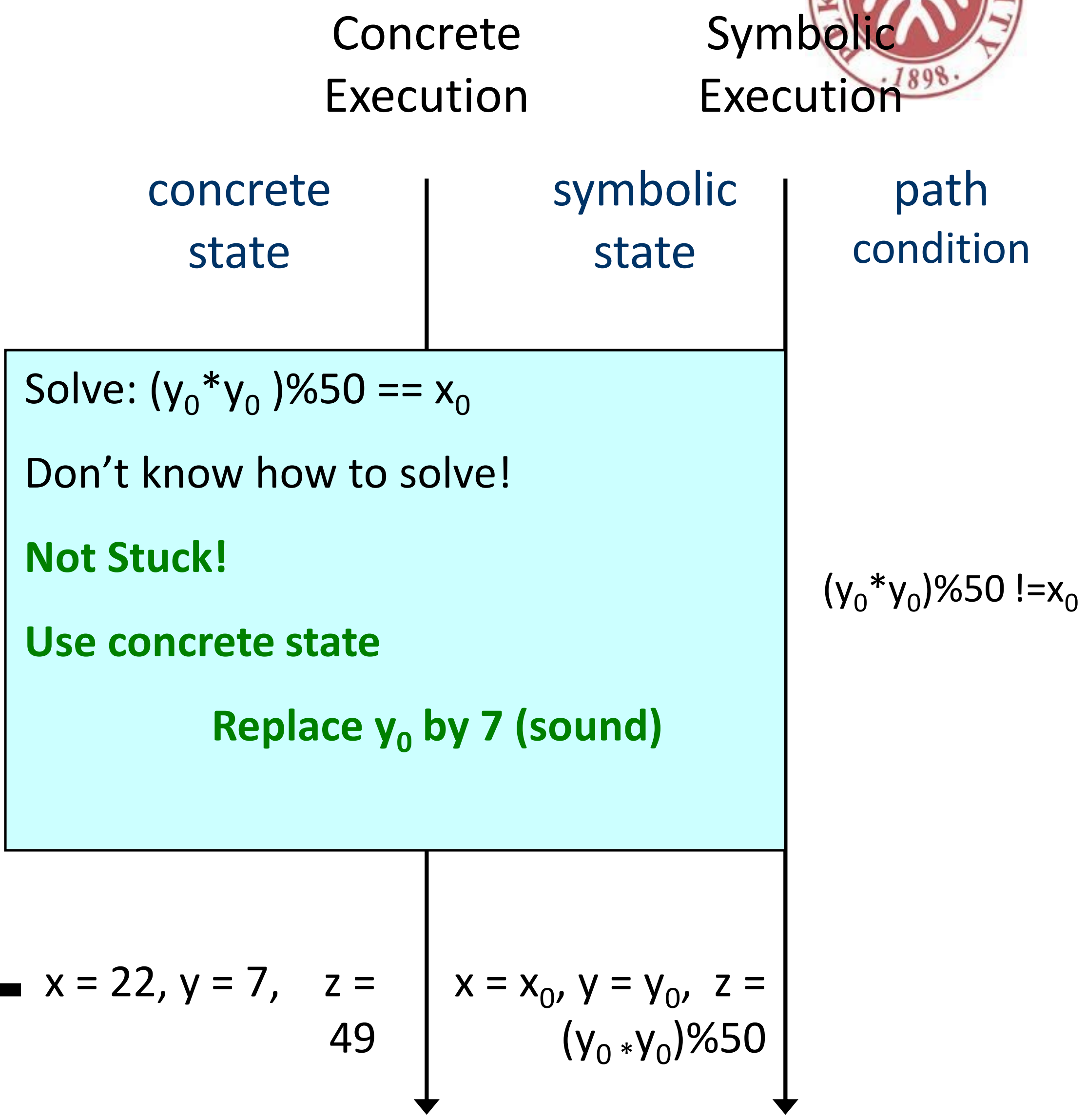
foo (y_0) !=x_0

x = 22, y = 7, z = 49

x = x_0, y = y_0, z = foo (y_0)

Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $(y_0*y_0)\%50 == x_0$

Don’t know how to solve!

Not Stuck!

Use concrete state

Replace y_0 by 7 (sound)

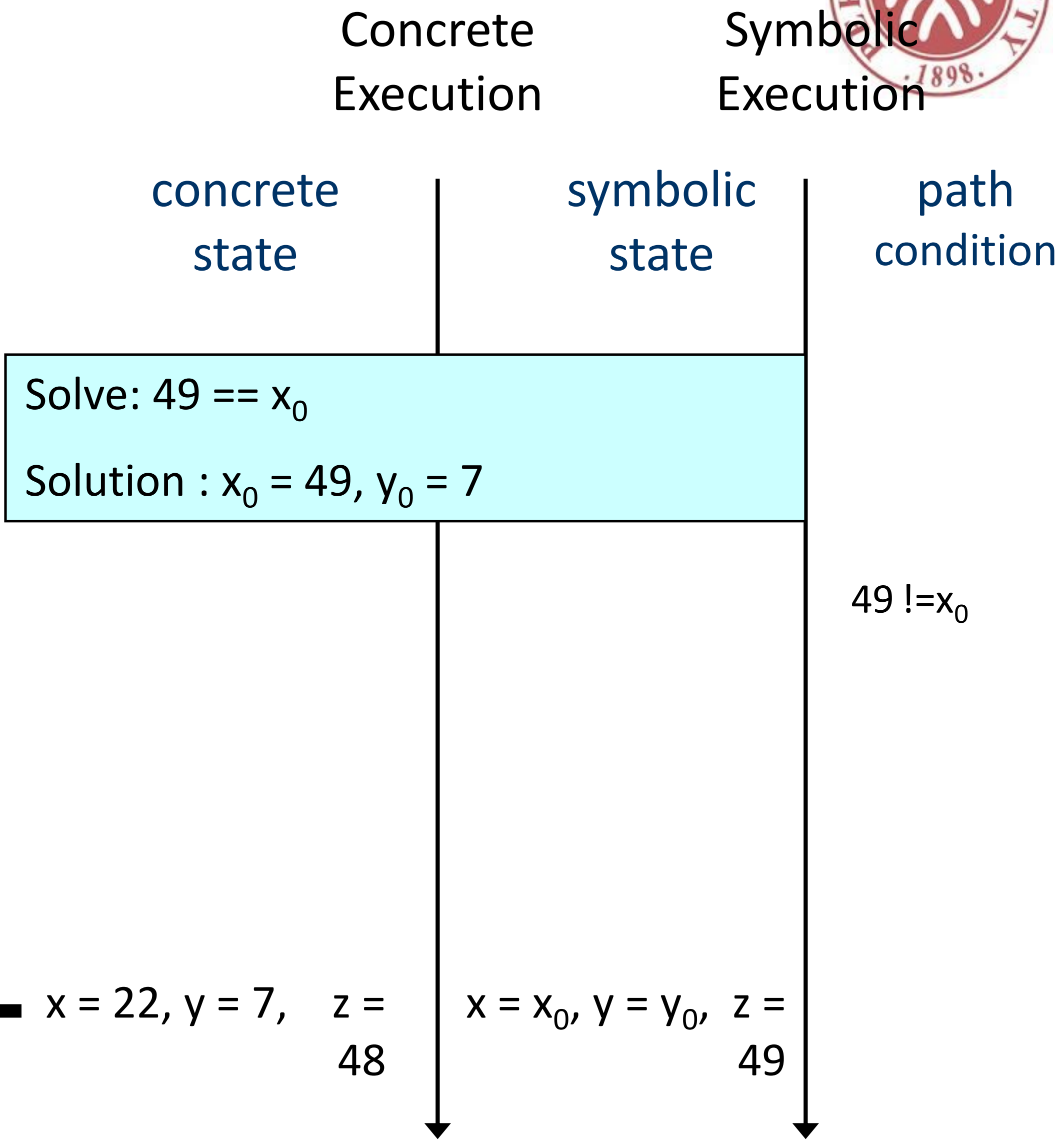
$(y_0*y_0)\%50 \,!= x_0$

x = 22, y = 7, z = 49

$x = x_0$, $y = y_0$, $z = (y_0 * y_0)\%50$

Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
            if (x > y+10) {
                ERROR;
            }
    }
}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: 49 == x_0

Solution : x_0 = 49, y_0 = 7

49 !=x_0

x = 22, y = 7, z = 48

x = x_0, y = y_0, z = 49

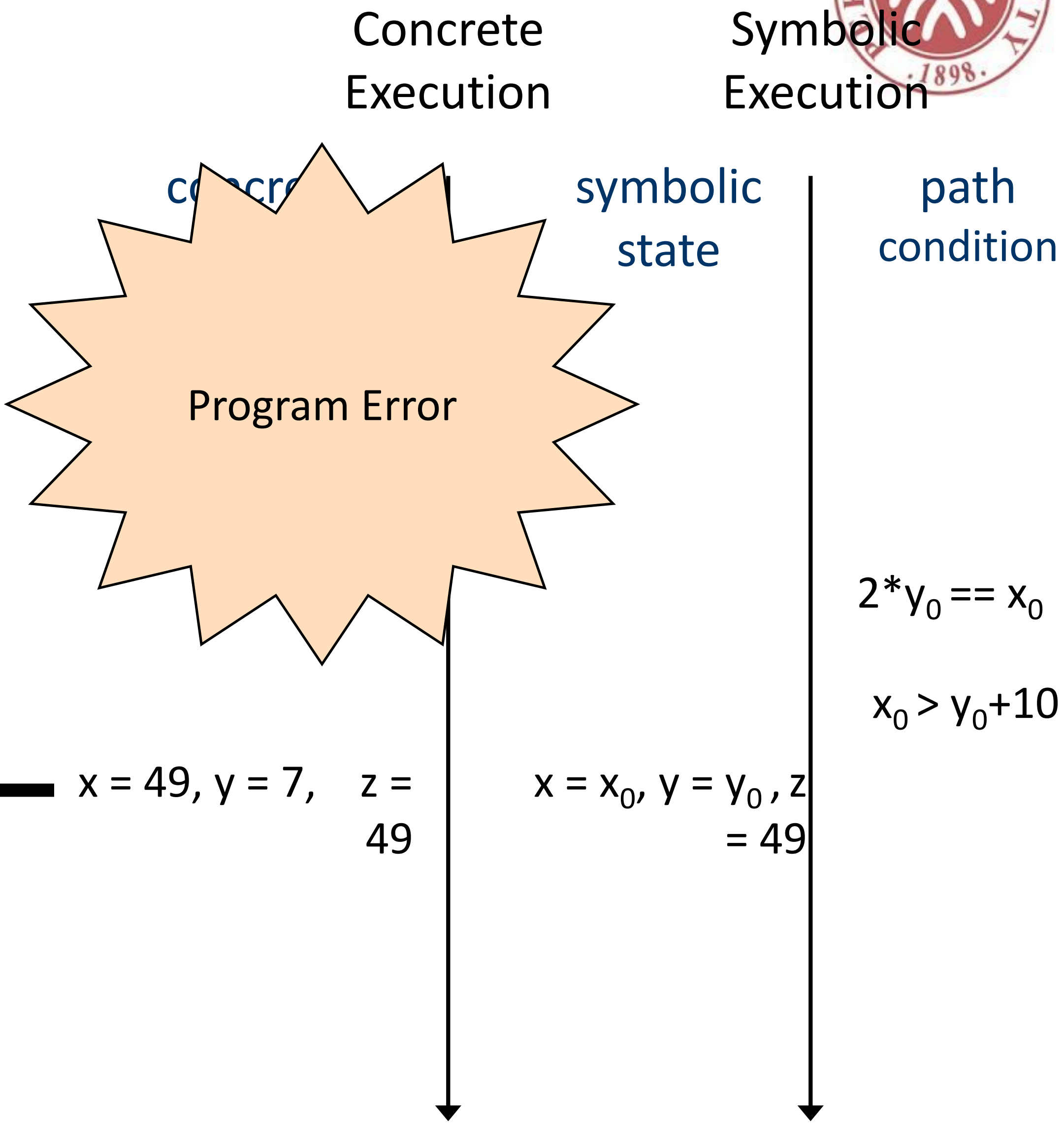
Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
            if (x > y+10) {
                ERROR;
            }
    }
}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 49, y = 7	$x = x_0$, $y = y_0$	

Novelty : Simultaneous Concrete and Symbolic Execution

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
            if (x > y+10) {
                  ERROR;
            }
    }
}
```



下周课程

- 由王博同学介绍LLVM
- 方法一
  - 从 http://pan.baidu.com/s/1c1EujRU 下载虚拟机，使用 VMWare 12 加载。
  - 密码为 123456。LLVM 根目录为 /home/llvm/llvm/ ，build 目录为/home/llvm/llvm/build/
- 方法二
  - 使用 Linux 或 Mac OS 操作系统（若使用Linux虚拟机推荐内存至少4G）。
  - 按照 http://llvm.org/docs/GettingStarted.html 中的Getting Started Quickly 提示进行编译。如无调试需要，请在 cmake 时按手册说明使用选项

    -DCMAKE_BUILD_TYPE=Release

    以加快编译速度。