lecture 23:
background on symbolic execution
5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

symbolic execution and program testing

http://dl.acm.org/citation.cfm?id=360252

the symbolic execution approach

the symbolic execution approach

towards large production of reliable programs

- two extreme alternatives

the symbolic execution approach

towards large production of reliable programs

- two extreme alternatives

program testing

- best effort: run program on sample inputs
- sample inputs?

the symbolic execution approach

towards large production of reliable programs

- two extreme alternatives

program testing

- best effort: run program on sample inputs
- sample inputs?

program verification

- logical proof based on precise specification
- not practical for routine use

the symbolic execution approach

towards large production of reliable programs

- two extreme alternatives

program testing

- best effort: run program on sample inputs
- sample inputs?

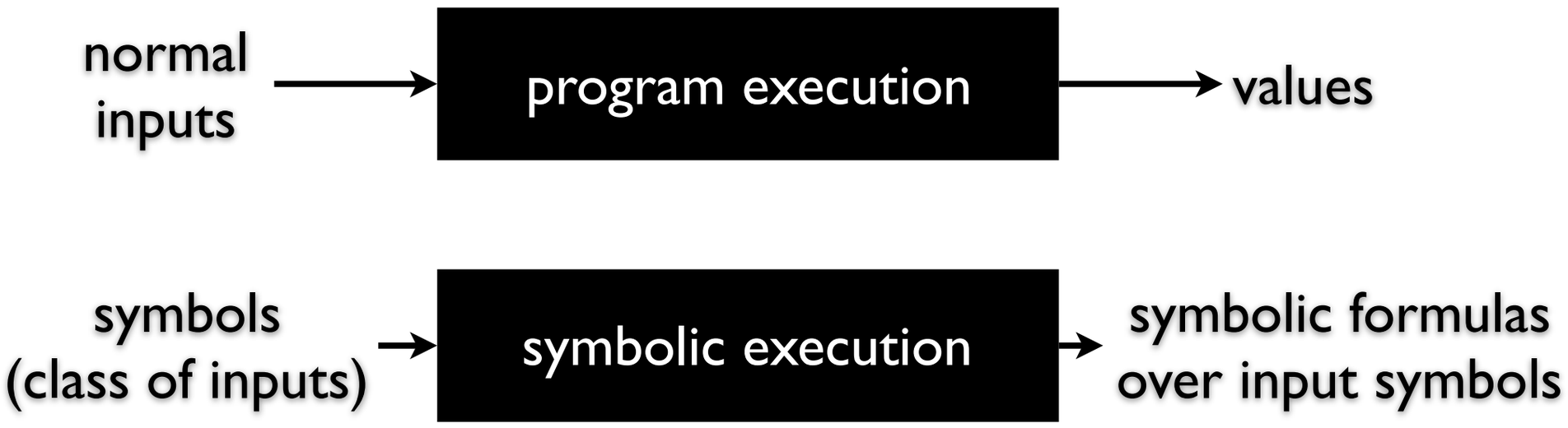
program verification

- logical proof based on precise specification
- not practical for routine use

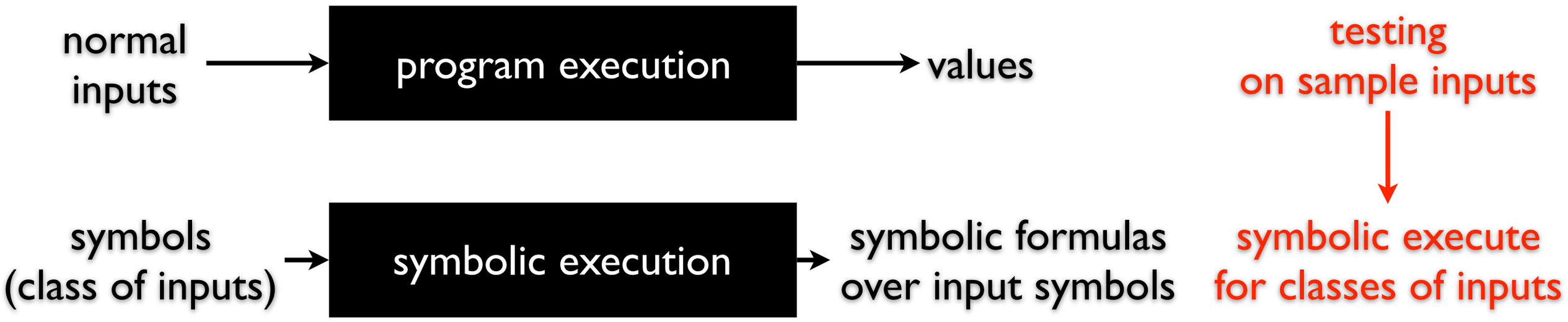
symbolic execution: a compromise

- assures "program meets its requirement" even when formal specifications are not given

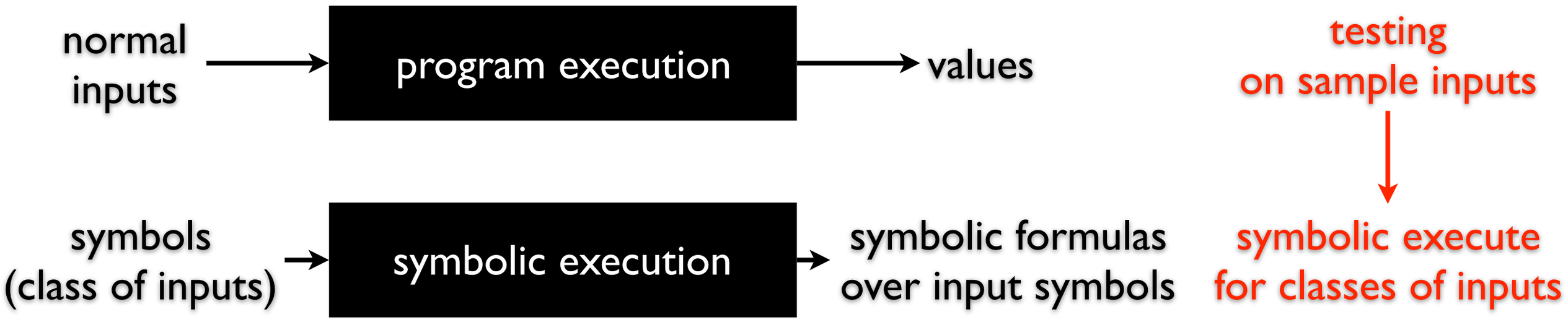
symbolic execution



symbolic execution



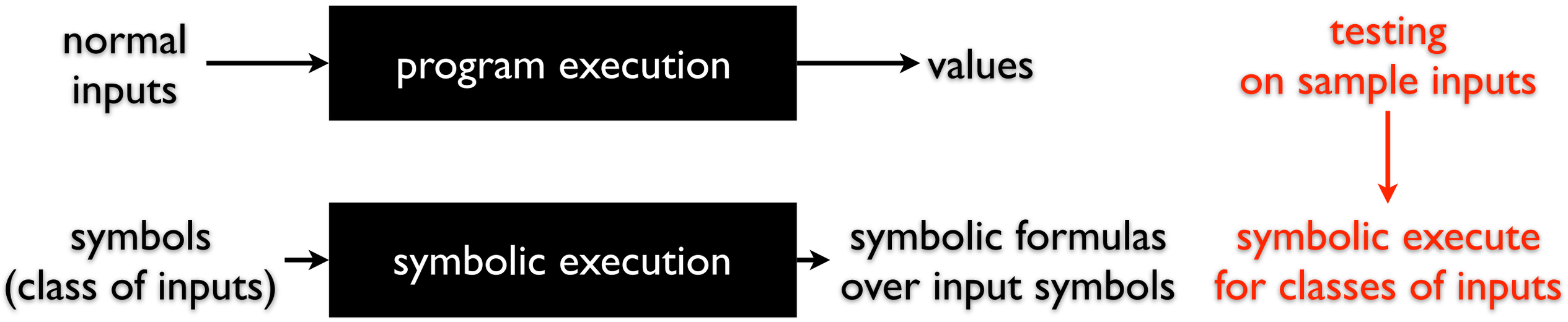
symbolic execution



class of inputs

- characterized by each symbol execution

symbolic execution



class of inputs

- characterized by each symbol execution

coverage

- determined by the dependence of program's control
  - e.g., control flow independent of inputs — a single symbolic execution

execution semantics

for normal program execution

- data objects
- how statements manipulate data objects
- how control flows through the statements

state of a program execution

- values of program variables
- statement counter

execution semantics

for symbolic execution

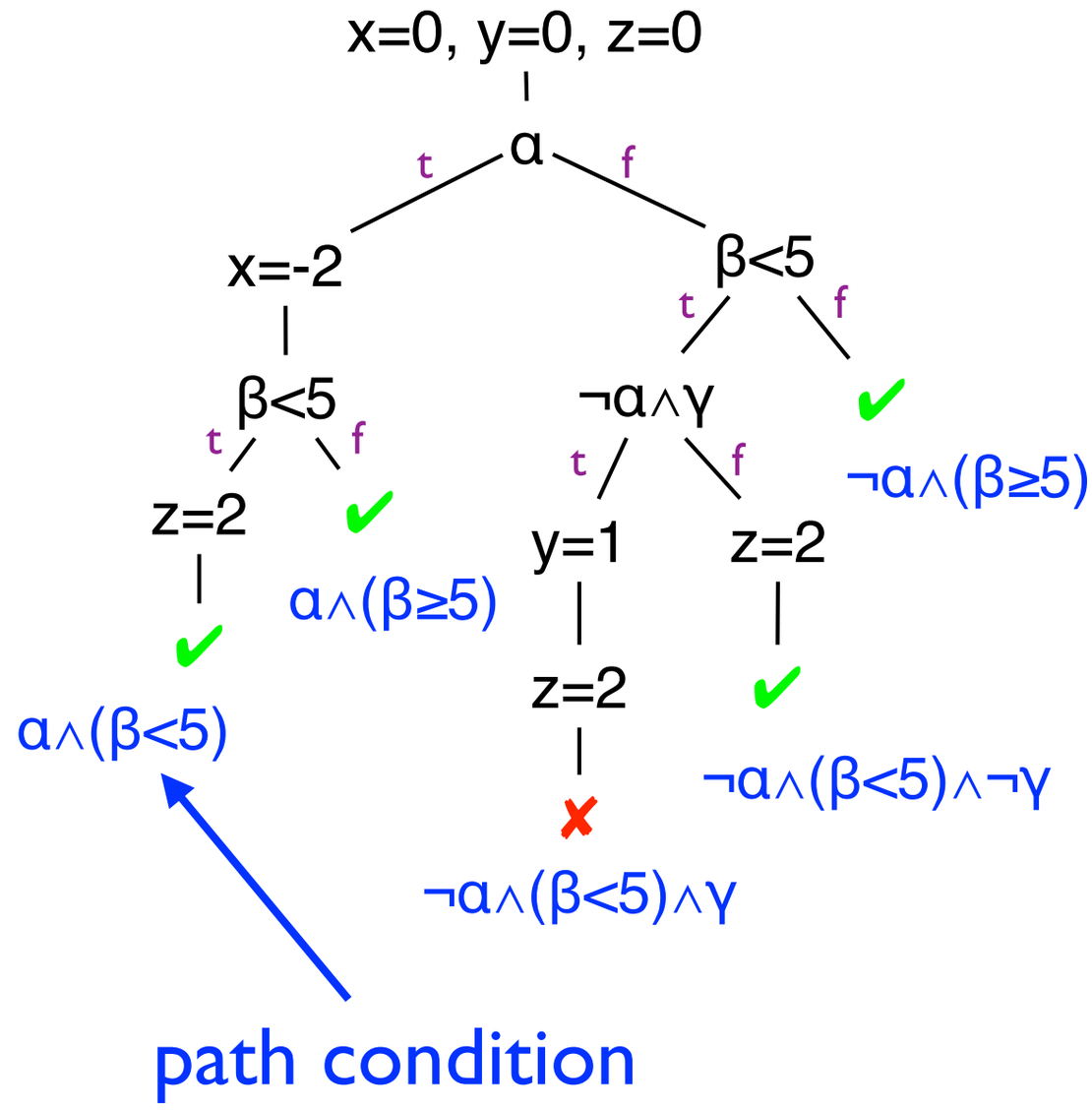
- symbolic data objects
- evaluation rules

state

- path condition (pc)
- accumulator of properties which the inputs must satisfy for an execution to follow the path

symbolic execution tree

```
1. int a = α, b = β, c = γ;
2.                    // symbolic
3. int x = 0, y = 0, z = 0;
4. if (a) {
5.   x = -2;
6. }
7. if (b < 5) {
8.   if (!a && c)  { y = 1; }
9.   z = 2;
10.}
11.assert(x+y+z!=3)
```



IF-statements

IF-statements

IF q(inputs…)

- in chossing between alternative paths, assumptions about the inputs are made and aded (conjoined) to pc
- pc initialized to TRUE

IF-statements

IF q(inputs…)

- in chossing between alternative paths, assumptions about the inputs are made and aded (conjoined) to pc
- pc initialized to TRUE

non-forking

- if pc ⊃ q or pc ⊃ ¬q

IF-statements

IF q(inputs…)

- in chossing between alternative paths, assumptions about the inputs are made and aded (conjoined) to pc
- pc initialized to TRUE

non-forking

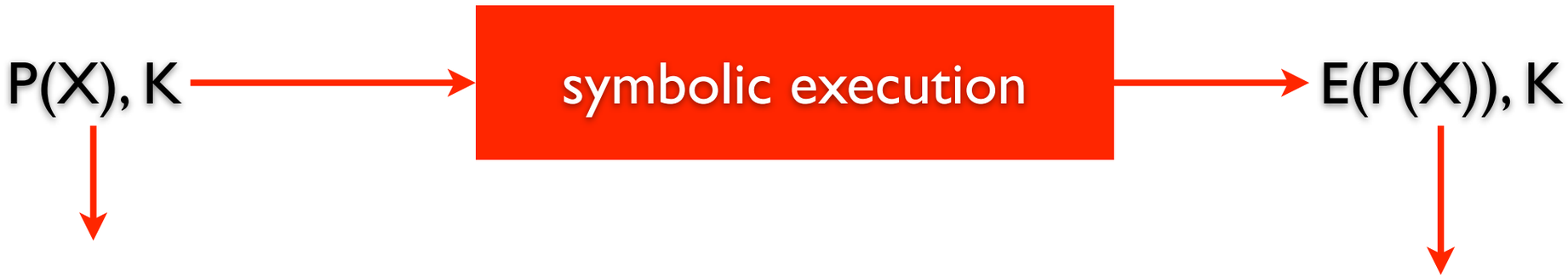
- if $pc \supset q$ or $pc \supset \neg q$

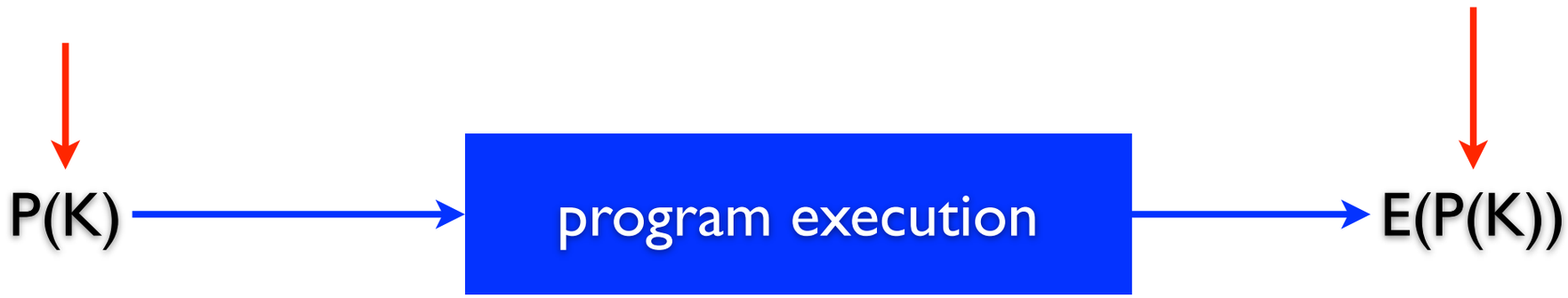
forking

- if neither $pc \supset q$ or $pc \supset \neg q$ is true
- THEN branch: $pc \longleftarrow pc \wedge q$
- ELSE branch: $pc \longleftarrow pc \wedge \neg q$

commutativity

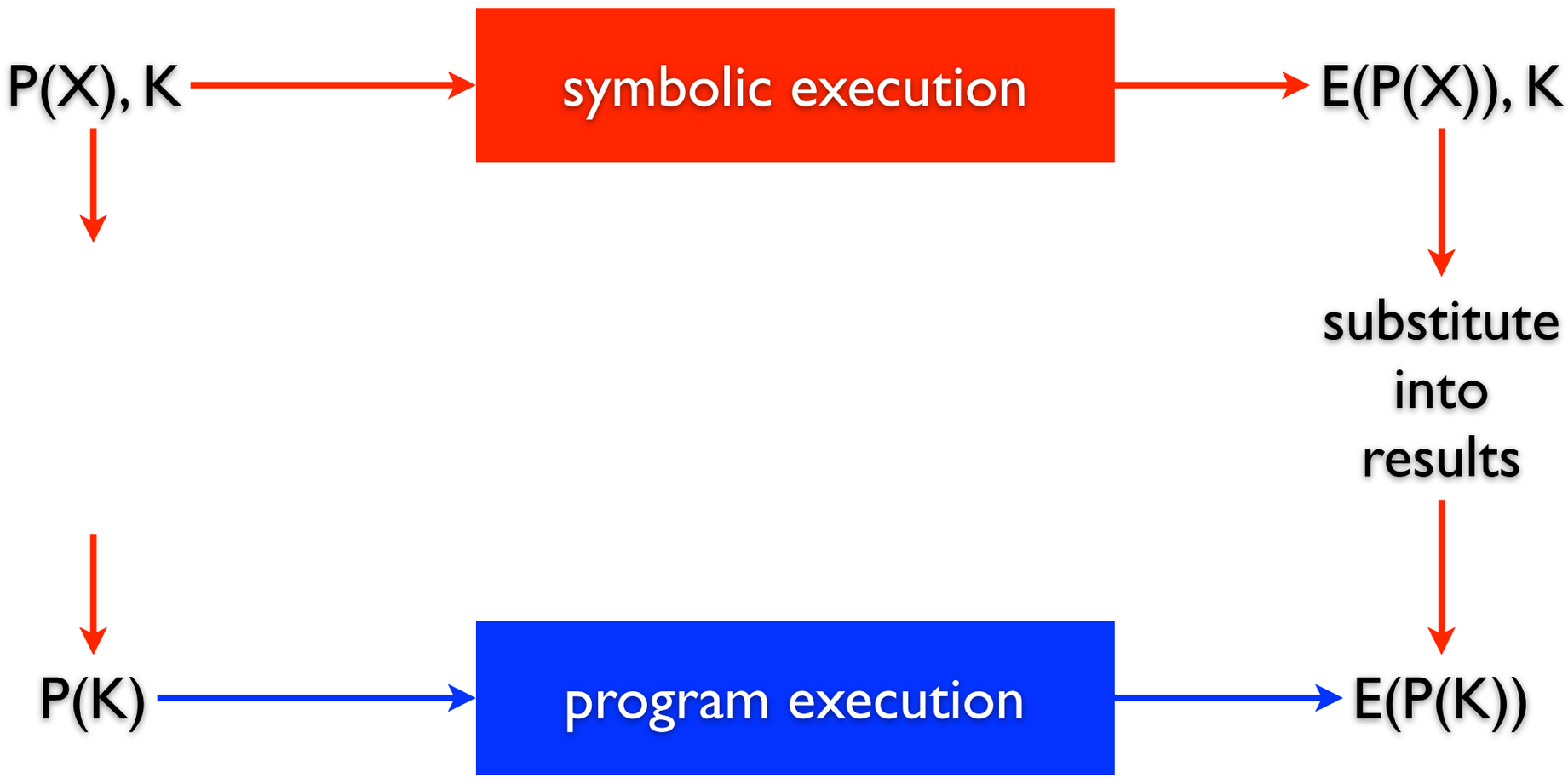
P — program, E(P(X)) — execute P on input symbol X, K — concrete inputs





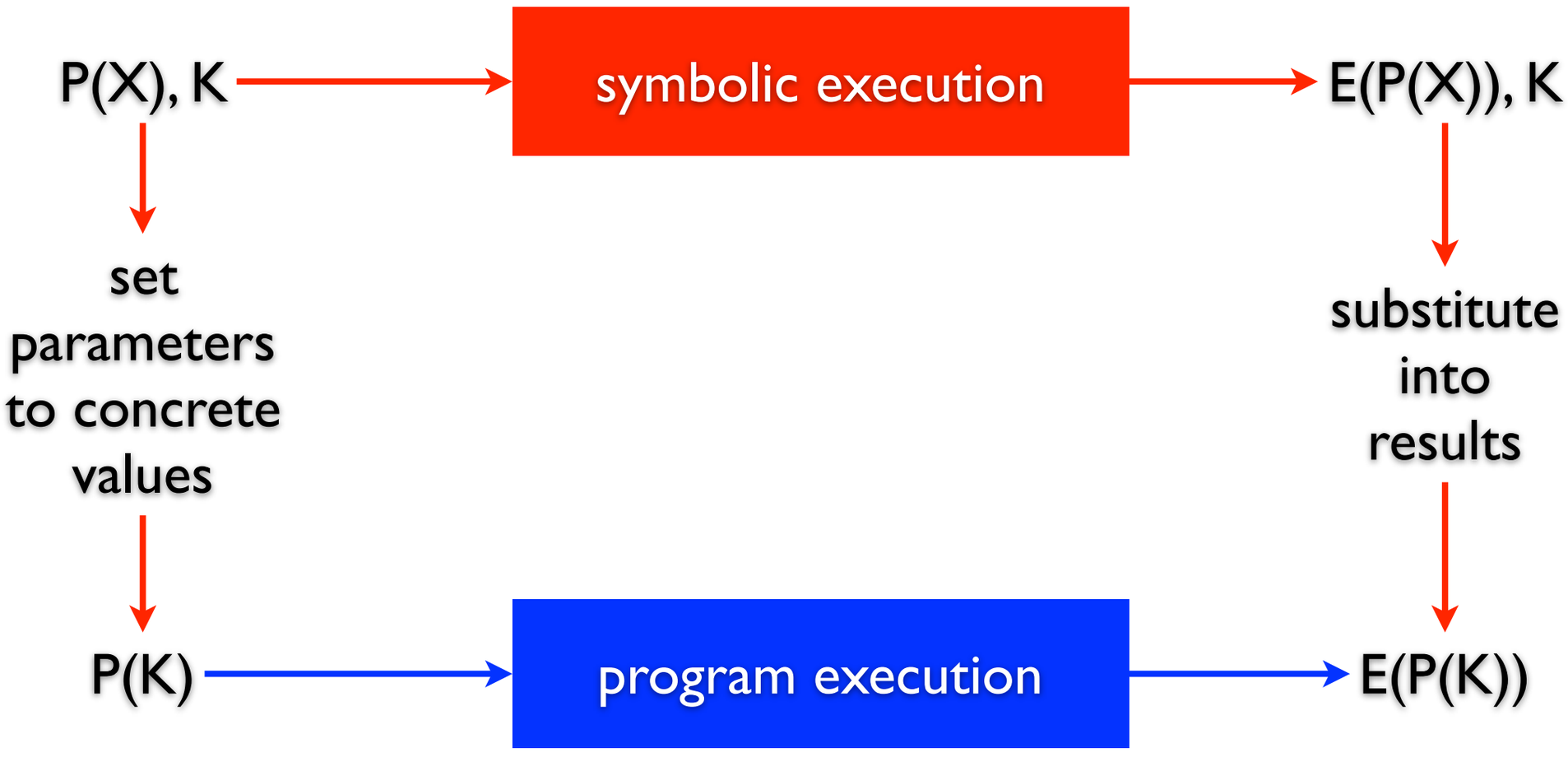
commutativity

P — program, E(P(X)) — execute P on input symbol X, K — concrete inputs



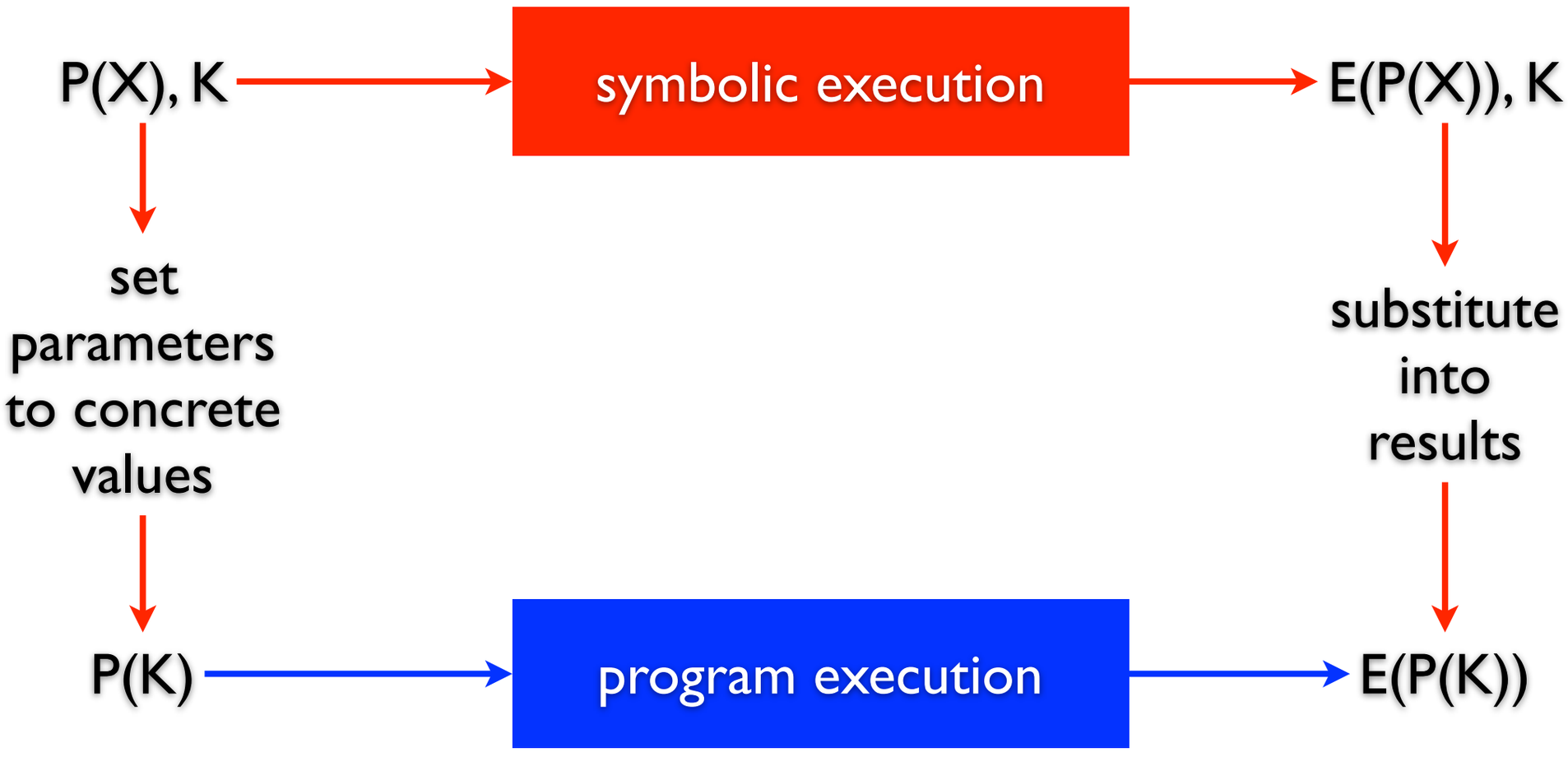
commutativity

P — program, E(P(X)) — execute P on input symbol X, K — concrete inputs



commutativity

P — program, E(P(X)) — execute P on input symbol X, K — concrete inputs



- symbolic execution captures the same effect as conventional execution
- the specific computation of the program are generalized and delayed

the challenges

the interesting case — brach type statements

- executing "IF statement" requires theorem proving

still an enhanced testing methodology

- input classes needed (to exhaust all possible cases) is practically infinite
  - lots of program paths
  - program state has many bits

industrial strength tool

- Klee

final report

formatting requirements

no more than two (2) single-spaced page SIGCOMM style

- 10pt font on 12pt leading formatted for printing on Letter-sized (8.5” by 11”)

latex template

- http://conferences.sigcomm.org/sigcomm/2017/misc/sig-alternate-10pt.cls

what to present

motivation

- introduce the problem
- if a concrete problem, place it in a larger problem space

introduction, approach

- present your solution — the method, design, architecture
  - describe your solution and the rational behind it

status

- preliminary results? what are the next steps?
- challenges and discussions?

examples

- https://www.usenix.org/conference/ons2014/technical-sessions/

online Student Feedback

Forms …

To complete your e-SFFs on line, simply log-in to TUportal, or the e-SFF website directly at http://esff.temple.edu and complete your feedback for this course.