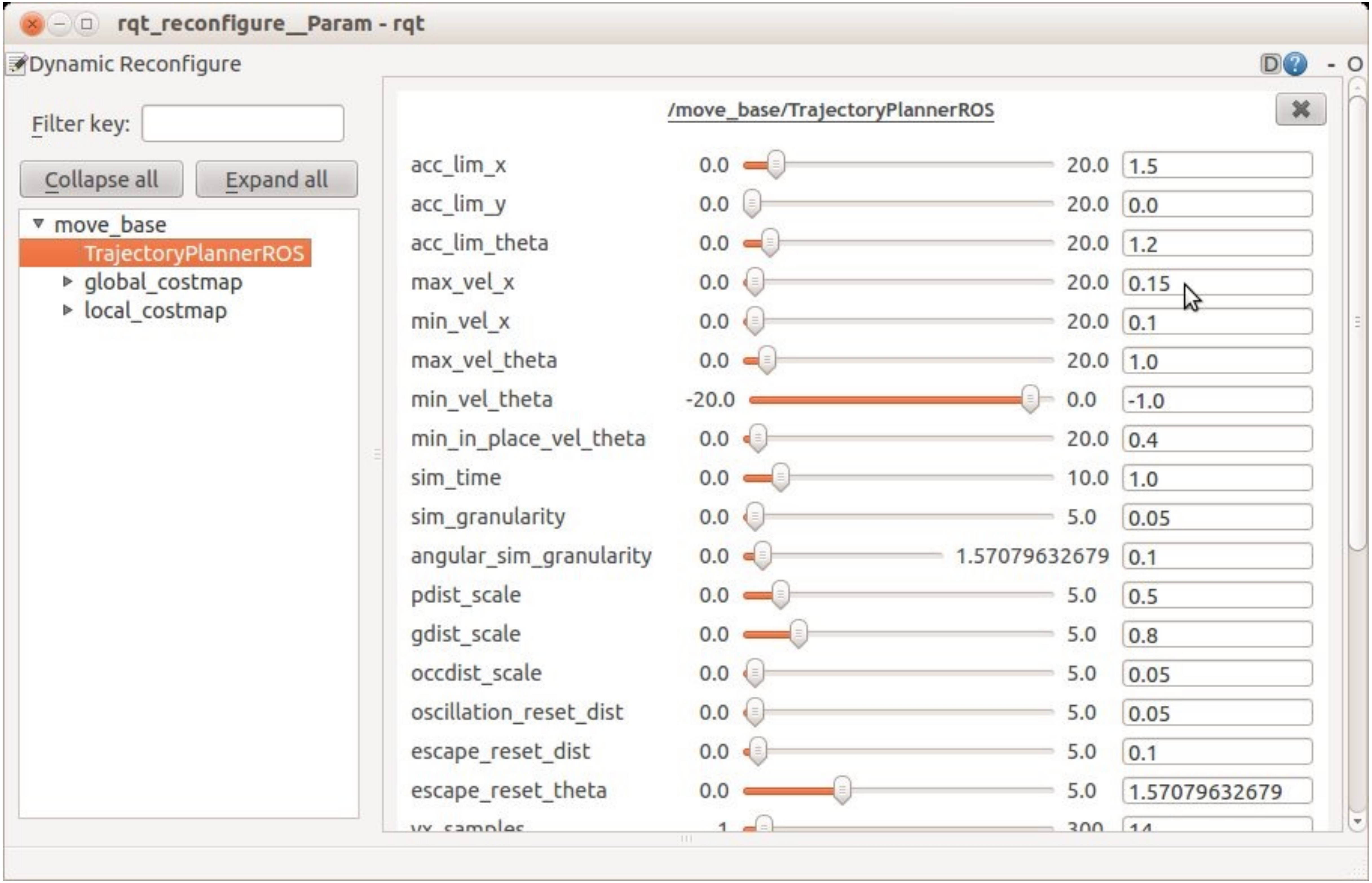
Exploiting Structure and Behavior of Highly Configurable Systems to Measure Performance

Miguel Velez
Software Engineering Ph.D. student
Carnegie Mellon University
Joint work with: Christian Kästner, Pooyan Jamshidi, and Norbert Siegmund

Most software is highly configurable



/move_base/TrajectoryPlannerROS

acc_lim_x	0.0	20.0	1.5
acc_lim_y	0.0	20.0	0.0
acc_lim_theta	0.0	20.0	1.2
max_vel_x	0.0	20.0	0.15
min_vel_x	0.0	20.0	0.1
max_vel_theta	0.0	20.0	1.0
min_vel_theta	-20.0	0.0	-1.0
min_in_place_vel_theta	0.0	20.0	0.4
sim_time	0.0	10.0	1.0
sim_granularity	0.0	5.0	0.05
angular_sim_granularity	0.0	1.57079632679	0.1
pdist_scale	0.0	5.0	0.5
gdist_scale	0.0	5.0	0.8
occdist_scale	0.0	5.0	0.05
oscillation_reset_dist	0.0	5.0	0.05
escape_reset_dist	0.0	5.0	0.1
escape_reset_theta	0.0	5.0	1.57079632679
vx_samples	1	300	14

Measuring performance in HCS is difficult

For developers and users

Overwhelmed by choices

Lose track of interactions

In an ideal world

"For each valid configuration, *x* is the *y* you will get"

y = quality attribute

x = actual value

Configuration	Execution Time
A=0 B=0 C=0 D=0	6s
A=0 B=0 C=1 D=1	6s
A=1 B=0 C=0 D=1	9s
…	…

$$T = 3A + 1AB + 6$$

Black-Box Approach

	A	B	C	D	T
1	0	0	0	0	6
2	0	0	0	1	6
3	0	0	1	0	6
4	0	0	1	1	6
5	0	1	0	0	6
6	0	1	0	1	6
7	0	1	1	0	6
8	0	1	1	1	6
9	1	0	0	0	9
10	1	0	0	1	9
11	1	0	1	0	9
12	1	0	1	1	9
13	1	1	0	0	10
14	1	1	0	1	10
15	1	1	1	0	10
16	1	1	1	1	10

Black-Box Approach

	A	B	C	D	T
1	0	0	0	0	6
2	0	0	0	1	6
3	0	0	1	0	6
4	0	0	1	1	6
5	0	1	0	0	6
6	0	1	0	1	6
7	0	1	1	0	6
8	0	1	1	1	6
9	1	0	0	0	9
10	1	0	0	1	9
11	1	0	1	0	9
12	1	0	1	1	9
13	1	1	0	0	10
14	1	1	0	1	10
15	1	1	1	0	10
16	1	1	1	1	10

Black-Box Approach

	A	B	C	D	T
1	0	0	0	0	6
2	0	0	0	1	6
3	0	0	1	0	6
4	0	0	1	1	6
5	0	1	0	0	6
6	0	1	0	1	6
7	0	1	1	0	6
8	0	1	1	1	6
9	1	0	0	0	9
10	1	0	0	1	9
11	1	0	1	0	9
12	1	0	1	1	9
13	1	1	0	0	10
14	1	1	0	1	10
15	1	1	1	0	10
16	1	1	1	1	10

Black-Box in practice

	A	B	C	D	T
1	0	0	0	0	6
2	0	0	0	1	6
3	0	0	1	0	6
4	0	0	1	1	6
5	0	1	0	0	6
6	0	1	0	1	6
7	0	1	1	0	6
8	0	1	1	1	6
9	1	0	0	0	9
10	1	0	0	1	9
11	1	0	1	0	9
12	1	0	1	1	9
13	1	1	0	0	10
14	1	1	0	1	10
15	1	1	1	0	10
16	1	1	1	1	10

Black-Box in practice

	A	B	C	D	T
1	0	0	0	0	6
2	0	0	0	1	6
3	0	0	1	0	6
4	0	0	1	1	6
5	0	1	0	0	6
6	0	1	0	1	6
7	0	1	1	0	6
8	0	1	1	1	6
9	1	0	0	0	9
10	1	0	0	1	9
11	1	0	1	0	9
12	1	0	1	1	9
13	1	1	0	0	10
14	1	1	0	1	10
15	1	1	1	0	10
16	1	1	1	1	10

Ideally

Want to find interactions

Identify what we need to measure

Determine if we do not need to measure all configurations

Our Proposal: Exploit Structure and Behavior of the Program to Measure Performance

White-Box Analysis

A B C D

```
x=A
y=B
sleep(6)
if(x)
    sleep(3)
if(y)
    sleep(1)
```

What We Can Do With Structure and Behavior

Determine how options are used

Identify interactions

Compare execution traces based on configurations

Know if we need to execute all configurations or subset

Approach: Combine Static Taint Analysis with Dynamic Analysis

Goal

AB

```
    x = A
    y = B
    sleep(6)
    if(x)
A?      sleep(3)
    if(y)
B?      sleep(1)
```

Static Taint Analysis

Static Taint Analysis

AB

```
x = A
y = B
sleep(6)
if(x)
    sleep(3)
if(y)
    sleep(1)
```

Static Taint Analysis

A B

```
x = A      {x_A}
y = B
sleep(6)
if (x)
    sleep(3)
if (y)
    sleep(1)
```

Static Taint Analysis

A B

```
x = A      $\{x_A\}$
y = B      $\{x_A y_B\}$
sleep(6)
if(x)
    sleep(3)
if(y)
    sleep(1)
```

Static Taint Analysis

AB

```
x = A       $\{x_A\}$
y = B       $\{x_A y_B\}$
sleep(6)    $\{x_A y_B\}$
if (x)
    sleep(3)
if (y)
    sleep(1)
```

Static Taint Analysis

A B

```
x = A          $\{x_A\}$
y = B          $\{x_A \ y_B\}$
sleep(6)       $\{x_A \ y_B\}$
if(x)          $\{x_A \ y_B\}$
    sleep(3)
if(y)
    sleep(1)
```

Static Taint Analysis

```
          x = A          $\{x_A\}$
AB        y = B          $\{x_A\ y_B\}$
          sleep(6)       $\{x_A\ y_B\}$
①         if (x) ↔ $\{x_A\ y_B\}$
A?            sleep(3)
          if (y)
              sleep(1)
```

Static Taint Analysis

AB

```
        x = A          {x_A}
        y = B          {x_A y_B}
        sleep(6)       {x_A y_B}
①   [   if (x)    <->  {x_A y_B}
A?  [      sleep(3)    {x_A y_B}
        if(y)
           sleep(1)
```

Static Taint Analysis

AB

```
        x=A           $\{x_A\}$
        y=B           $\{x_A\ y_B\}$
        sleep(6)      $\{x_A\ y_B\}$
①      if(x) ↔       $\{x_A\ y_B\}$
A?          sleep(3)  $\{x_A\ y_B\}$
        if(y)         $\{x_A\ y_B\}$
            sleep(1)
```

Static Taint Analysis

A B

```
      x = A              $\{x_A\}$
      y = B              $\{x_A y_B\}$
      sleep(6)           $\{x_A y_B\}$
①     if (x) ↔           $\{x_A y_B\}$
A?        sleep(3)       $\{x_A y_B\}$
②     if (y) ↔           $\{x_A y_B\}$
B?        sleep(1)
```

Static Taint Analysis

AB

x = A $\{x_A\}$

y = B $\{x_A\ y_B\}$

sleep(6) $\{x_A\ y_B\}$

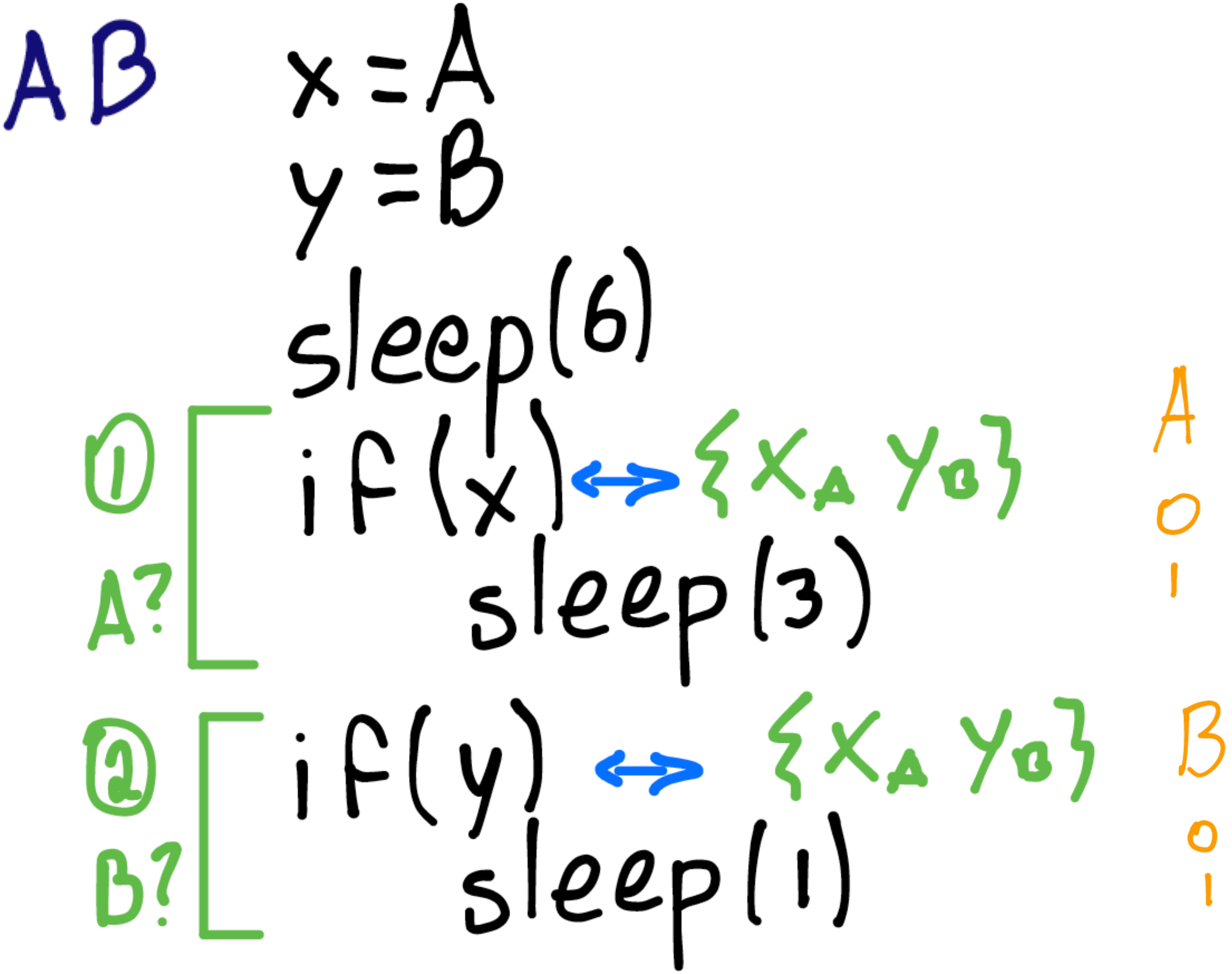
① [if(x) ↔ $\{x_A\ y_B\}$

A? [sleep(3) $\{x_A\ y_B\}$

② [if(y) ↔ $\{x_A\ y_B\}$

B? [sleep(1) $\{x_A\ y_B\}$

Relevant Statements and Options



```
x = A
y = B
sleep(6)
if(x)
    sleep(3)
if(y)
    sleep(1)
```

Relevant Statements and Options

AB

```
x = A
y = B
sleep(6)
① [ if(x)  ↔ {x_A y_B}
A?  [     sleep(3)
② [ if(y)  ↔ {x_A y_B}
B?  [     sleep(1)
```

AB

0 0

1 1

Dynamic Analysis

Transform AST

①
A?

```
if(x)
    t=ct
    sleep(3)
    ①←ct-t
```

②
B?

```
if(y)
    t=ct
    sleep(1)
    ②←ct-t
```

A	B	①	②
0	0		
1	1		

Building Performance Table

AB

+ = then   – = else

```
     x = A
     y = B
     sleep(6)
①  [ if(x) ↔ {X_A Y_B}
A?  [     sleep(3)
②  [ if(y) ↔ {X_A Y_B}
B?  [     sleep(1)
```

	A	B	①	②	T
1	0	0			
2	1	1			

Building Performance Table

AB

```
    x = A
    y = B
    sleep(6)
①  [ if(x)  <->  {X_A Y_B}
A?  [     sleep(3)
②  [ if(y)  <->  {X_A Y_B}
B?  [     sleep(1)
```

+ = then   - = else

	A	B	①	②	T
1	0	0	-0	-0	6
2	1	1			

Building Performance Table

AB

+ = then   − = else

```
      x = A
      y = B
      sleep(6)
①     if(x) ↔ {X_A Y_B}
A?        sleep(3)
②     if(y) ↔ {X_A Y_B}
B?        sleep(1)
```

	A	B	①	②	T
1	0	0	−0	−0	6
2	1	1	+3	+1	10

Building Performance Table

AB

```
     x = A
     y = B
     sleep(6)
①  [ if(x) ↔ {X_A Y_B}
A?  [     sleep(3)
②  [ if(y) ↔ {X_A Y_B}
B?  [     sleep(1)
```

+ = then   - = else

	A	B	①	②	T
1	0	0	-0	-0	6
2	1	1	+3	+1	10
3	1	0	+3	-0	9

Building Performance Table

AB

```
x = A
y = B
sleep(6)
① [ if(x)  ↔ {X_A Y_B}
A?        sleep(3)
② [ if(y)  ↔ {X_A Y_B}
B?        sleep(1)
```

+ = then   − = else

	AB		①	②	T
1	0	0	−0	−0	6
2	1	1	+3	+1	10
3	1	0	+3	−0	9
4	0	1	−0	+1	7

Building Performance Table

AB

```
     x = A
     y = B
     sleep(6)
①  [ if(x)   ↔ $\{x_A\ y_B\}$
A?       sleep(3)
②  [ if(y)   ↔ $\{x_A\ y_B\}$
B?       sleep(1)
```

+ = then   − = else

	A	B	①	②	T
1	0	0	−0	−0	6
2	1	1	+3	+1	10
3	1	0	+3	−0	9
4	0	1	−0	+1	7

without executing

Future Directions

Apply in Real Programs

C

Java

RQ: How often can we exploit lack of interactions to measure performance?

Augment Tracking Information

```
    x = A
    y = B
    sleep(6)
    if(x)
A?      sleep(3)

B?  if(y)
        sleep(0)
```

Augment Tracking Information

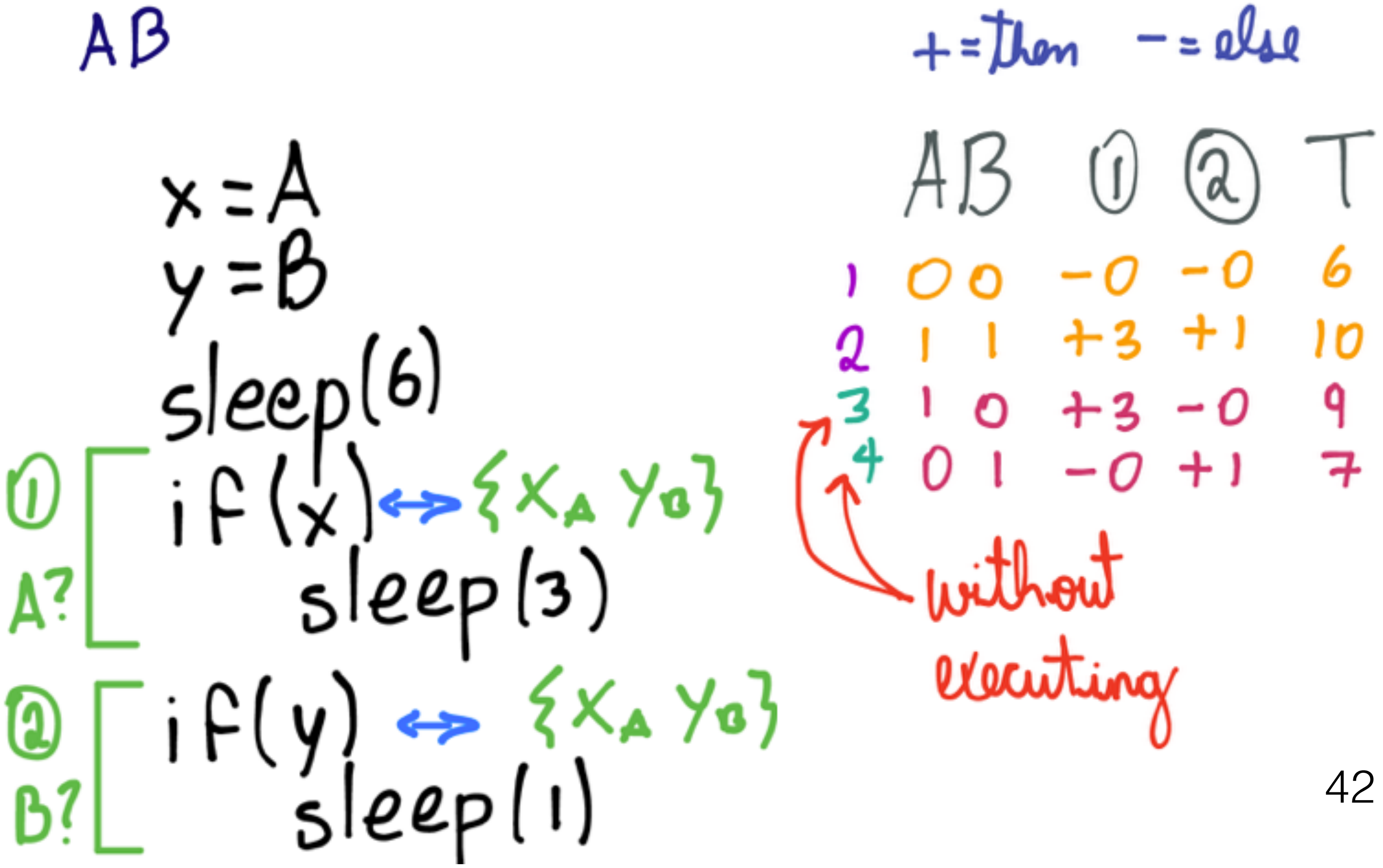
```
      x = A
      y = B
      sleep(6)
A  [  if(x)
          sleep(3)
B? [  if(y)
          sleep(0)
```

Augment Tracking Information

```
    x=A
    y=B
    sleep(6)
A   if(x)
        sleep(3)
B?  if(y)
        sleep(0)
```

Summary

- From Black-Box to White-Box
- Static Taint Analysis with Dynamic Analysis
- New pruning strategies in real programs



```
x=A
y=B
sleep(6)
if(x)
    sleep(3)
if(y)
    sleep(1)
```

+=then -=else

	AB	①	②	T
1	0 0	−0	−0	6
2	1 1	+3	+1	10
3	1 0	+3	−0	9
4	0 1	−0	+1	7

Miguel Velez
mvelezce@cs.cmu.edu

Data-flow Example

A B C D

```
          x = A
          y = B
          sleep(6)
 ①     [  if(x)        ↔ {x_A y_B}
 A?    |      sleep(3)
       [      y = 1    ↔ {x_A y_A}
 ②     [  if(y)        ↔ {x_A y_B y_A}
 A B?  [      sleep(1)
```