A Trusted Safety Verifier for Process Controller Code

Stephen McLaughlin, Devin Pohly, Patrick McDaniel and Saman Zonouz

February 24, 2014

Control Systems Under Threat

PENNSTATE

Programmable Controllers are Insecure By Design

digital bond

Control Systems Under Threat

PENNSTATE

Programmable Controllers are Insecure By Design

digital bond

Control systems not robust enough for security patches

Control Systems Under Threat

PENNSTATE

Programmable Controllers are Insecure By Design

digital bond

Control systems not robust enough for security patches

Google stuxnet

Web News Videos Images Books More Search tools

About 1,560 results (0.20 seconds)

South Korea to develop **Stuxnet**-like cyberweapons
BBC News - 11 hours ago
The country's defence ministry wants to develop weapons similar to **Stuxnet**, the software designed to attack Iranian nuclear enrichment plants.
South Korea plans **Stuxnet**-style cyber weapons to sabotage North's

A new perspective on SCADA

PENNSTATE

Supervisory Control and Data Acquisition

A new perspective on SCADA

PENNSTATE 1855

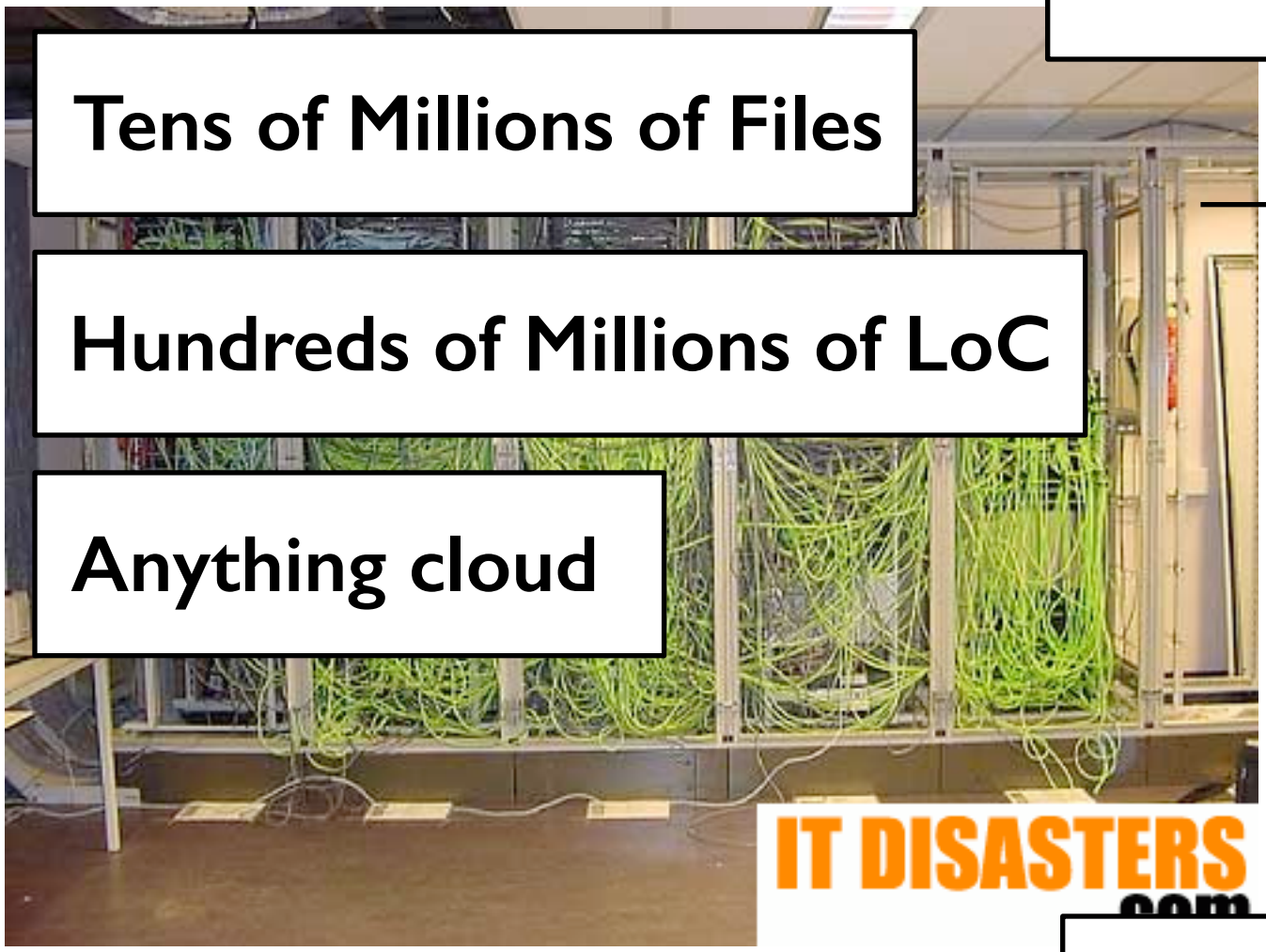
A new perspective on SCADA

PENNSTATE 1855

Tens of Millions of Files

Hundreds of Millions of LoC

Anything cloud



Far smaller codebase

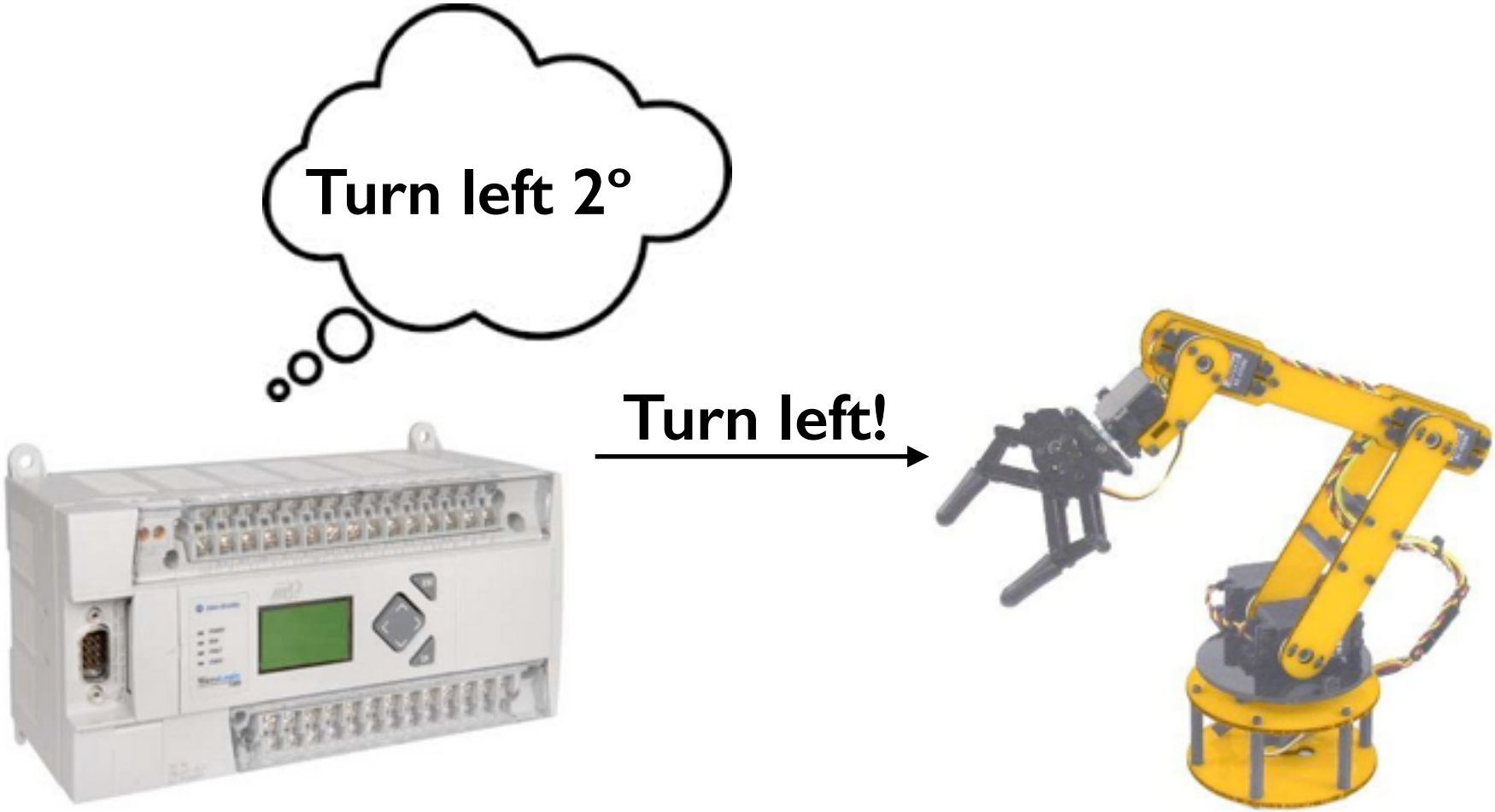
Hundreds of Devices

We would like to directly protect the physical process, regardless of the integrity of the IT perimeter.

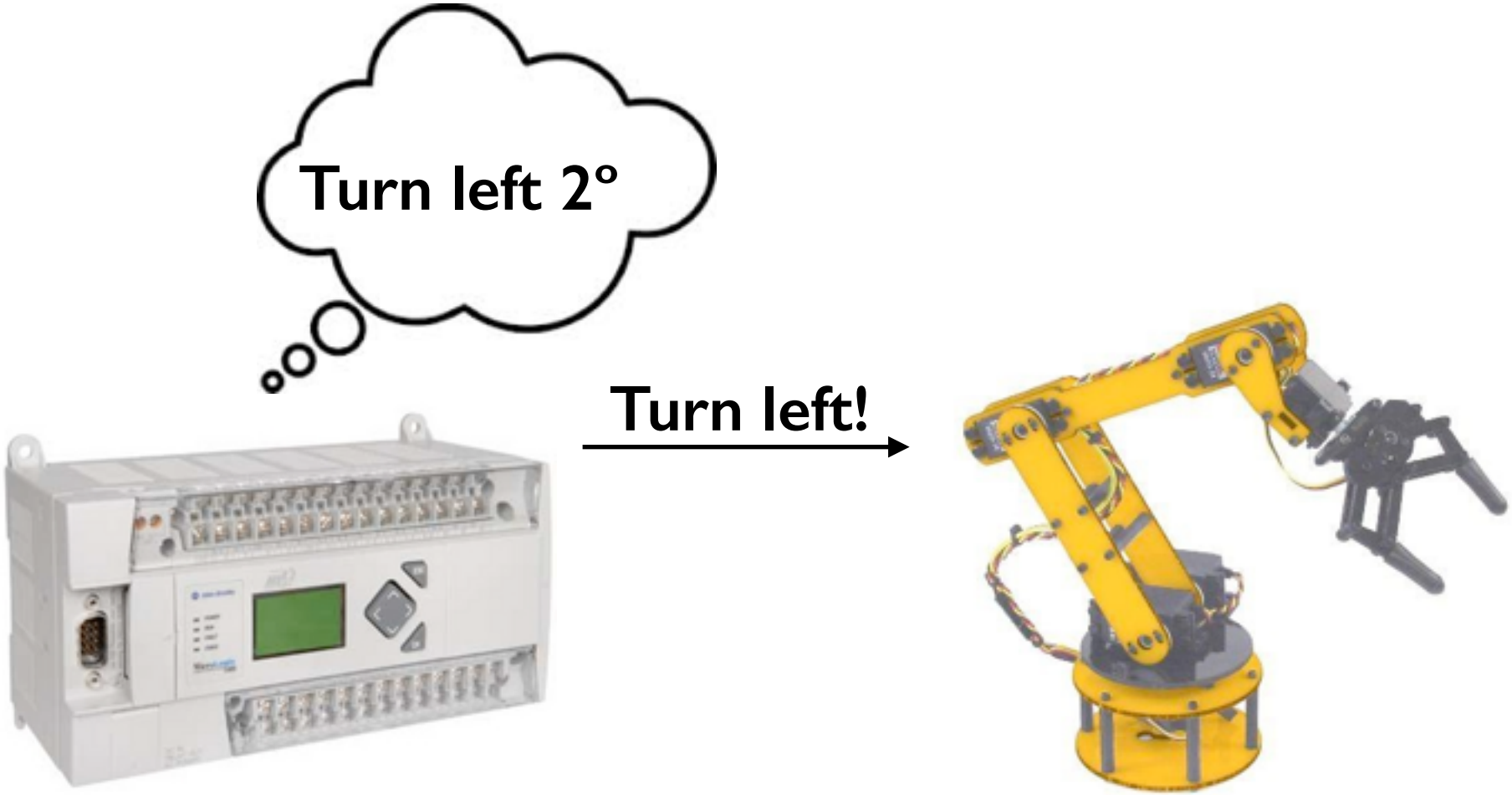
A PLC's life

A PLC's life

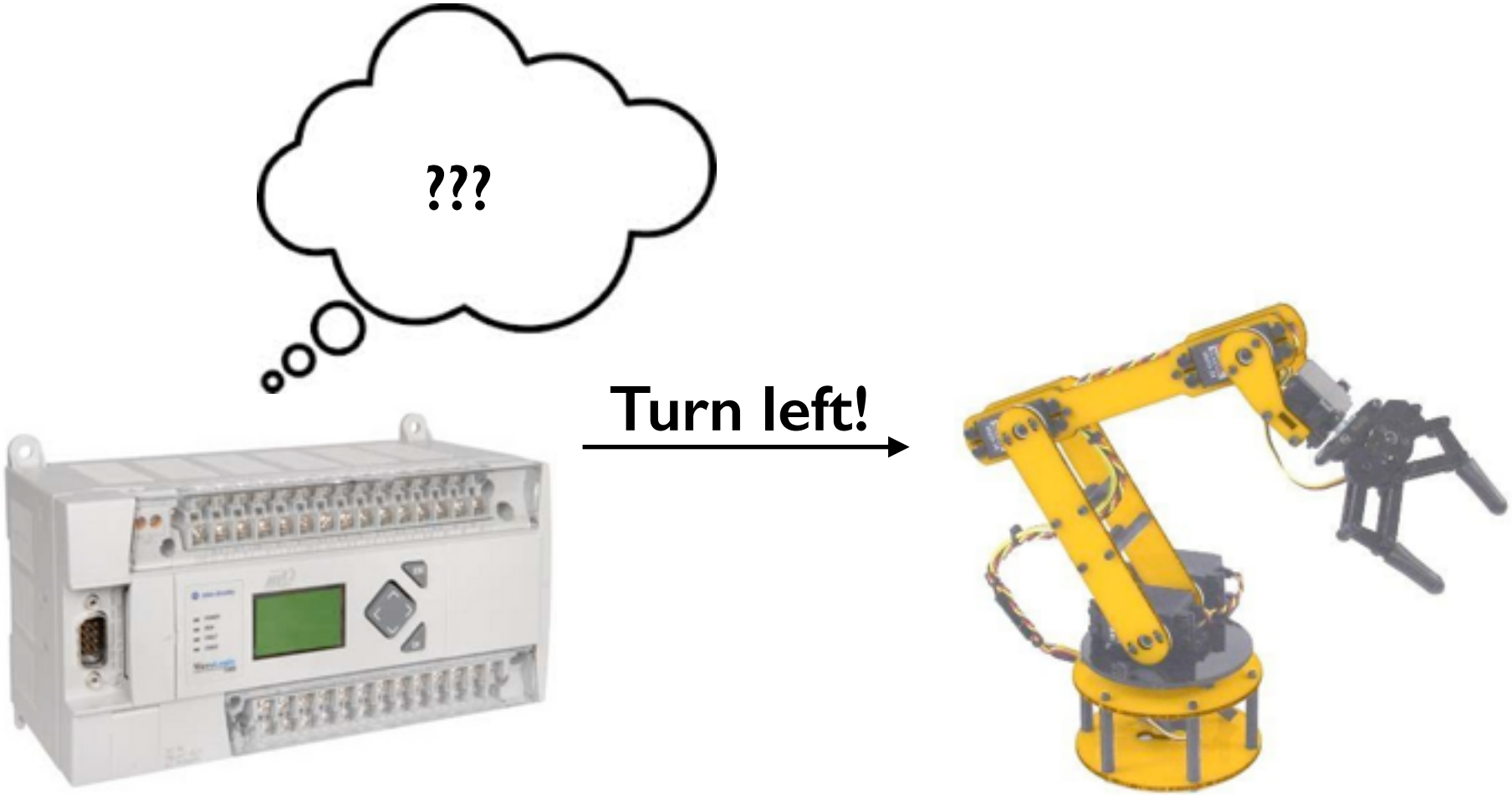
A PLC's life



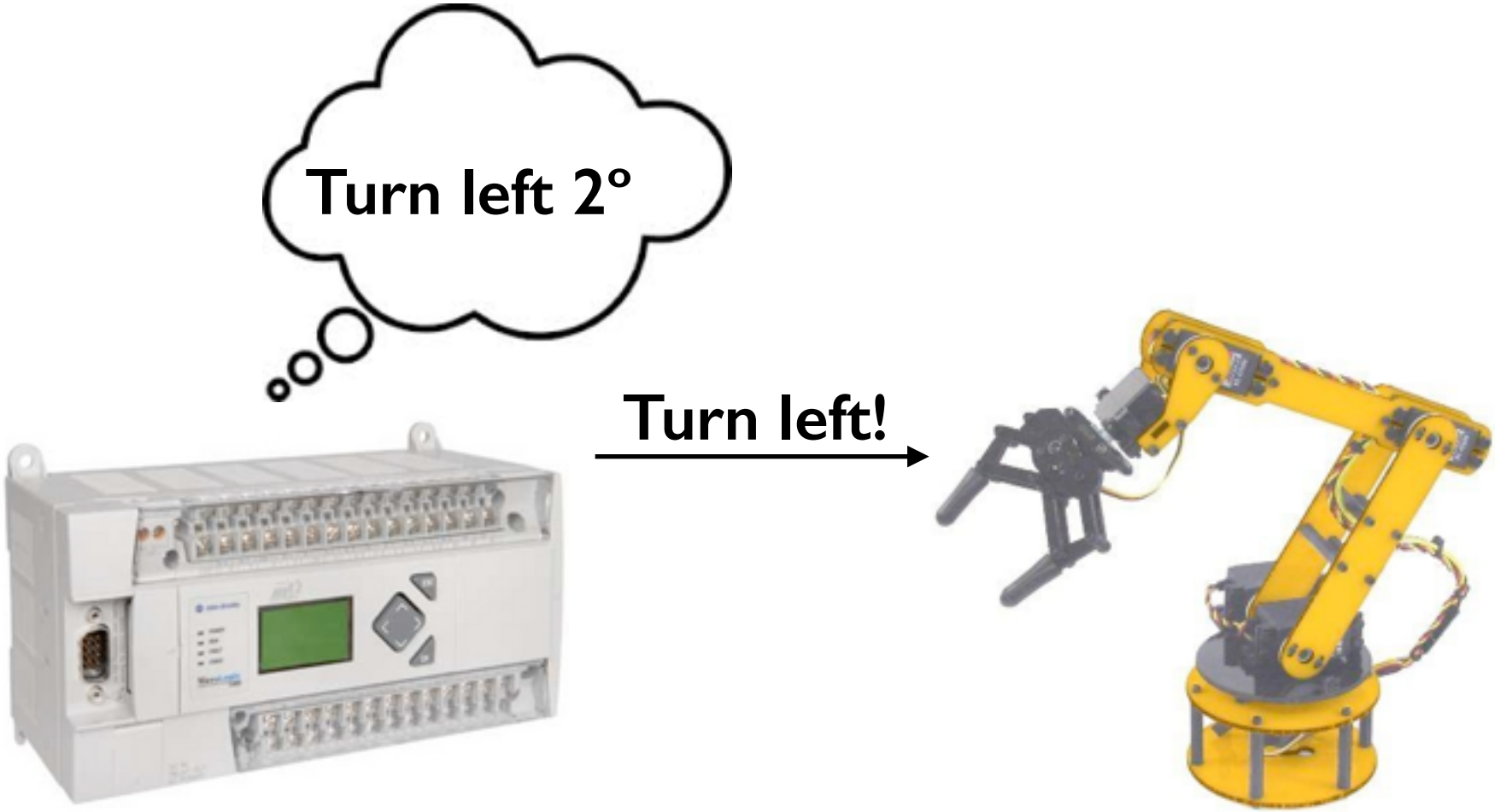
A PLC's life



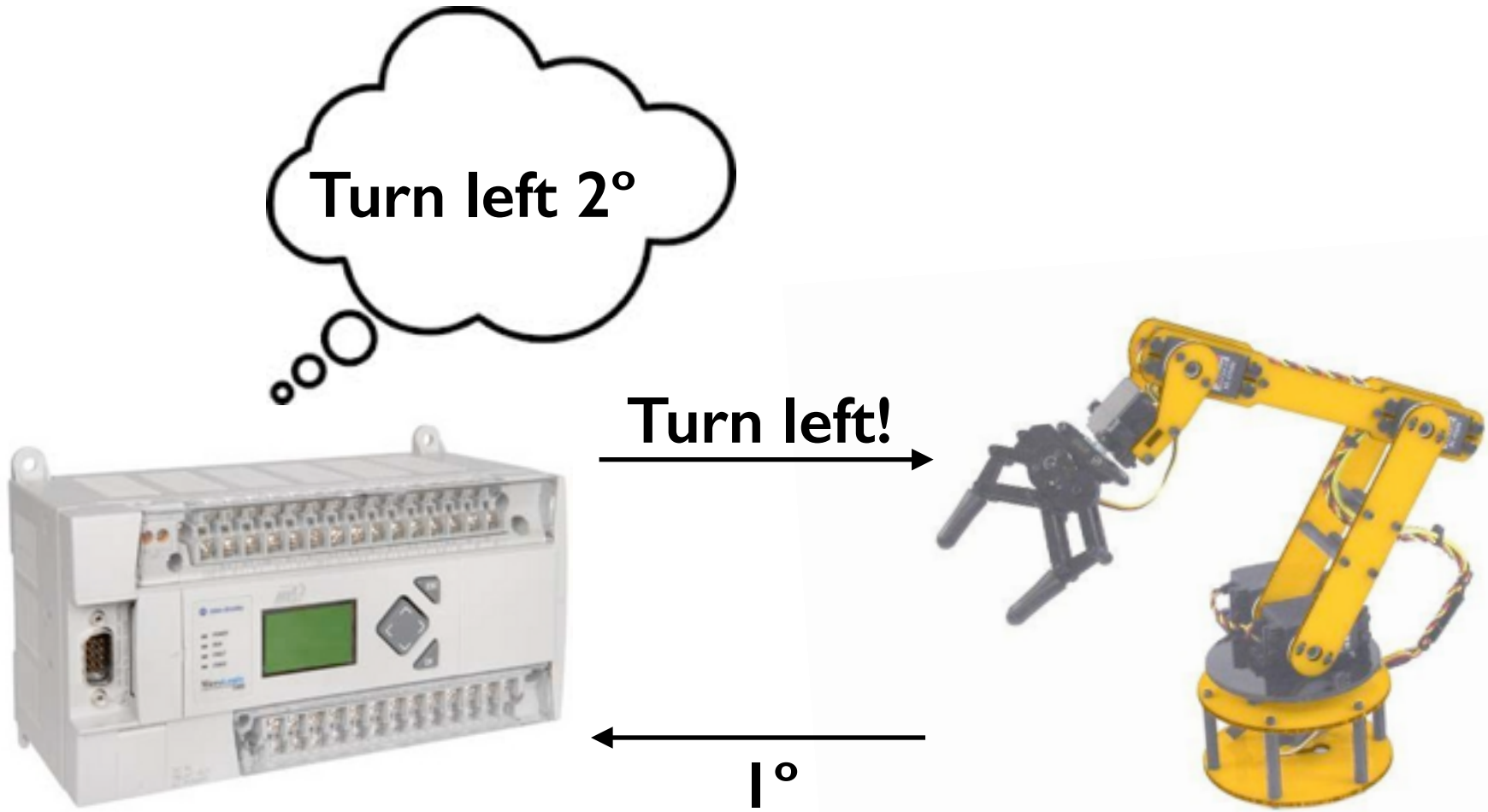
A PLC's life



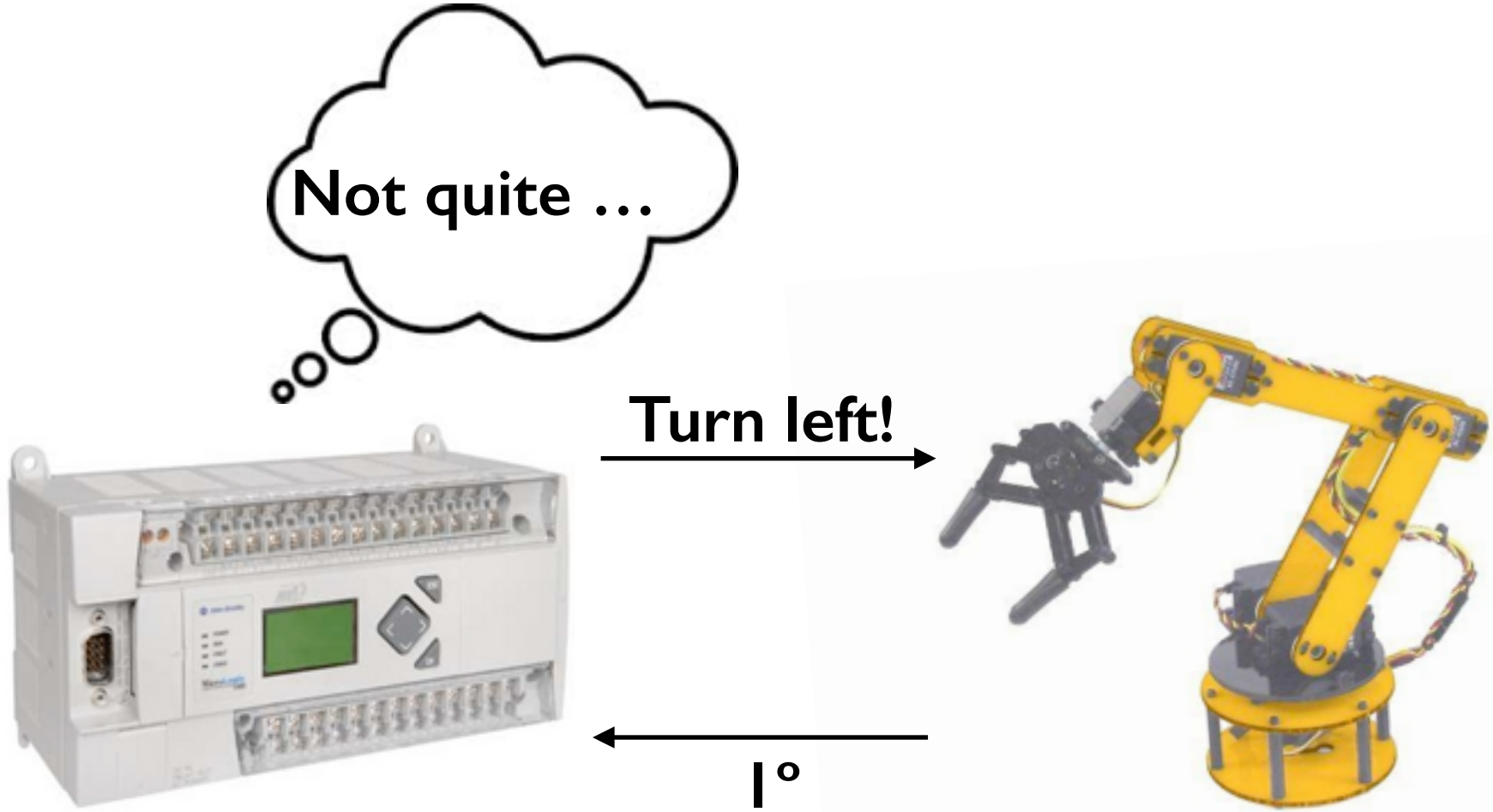
A PLC's life



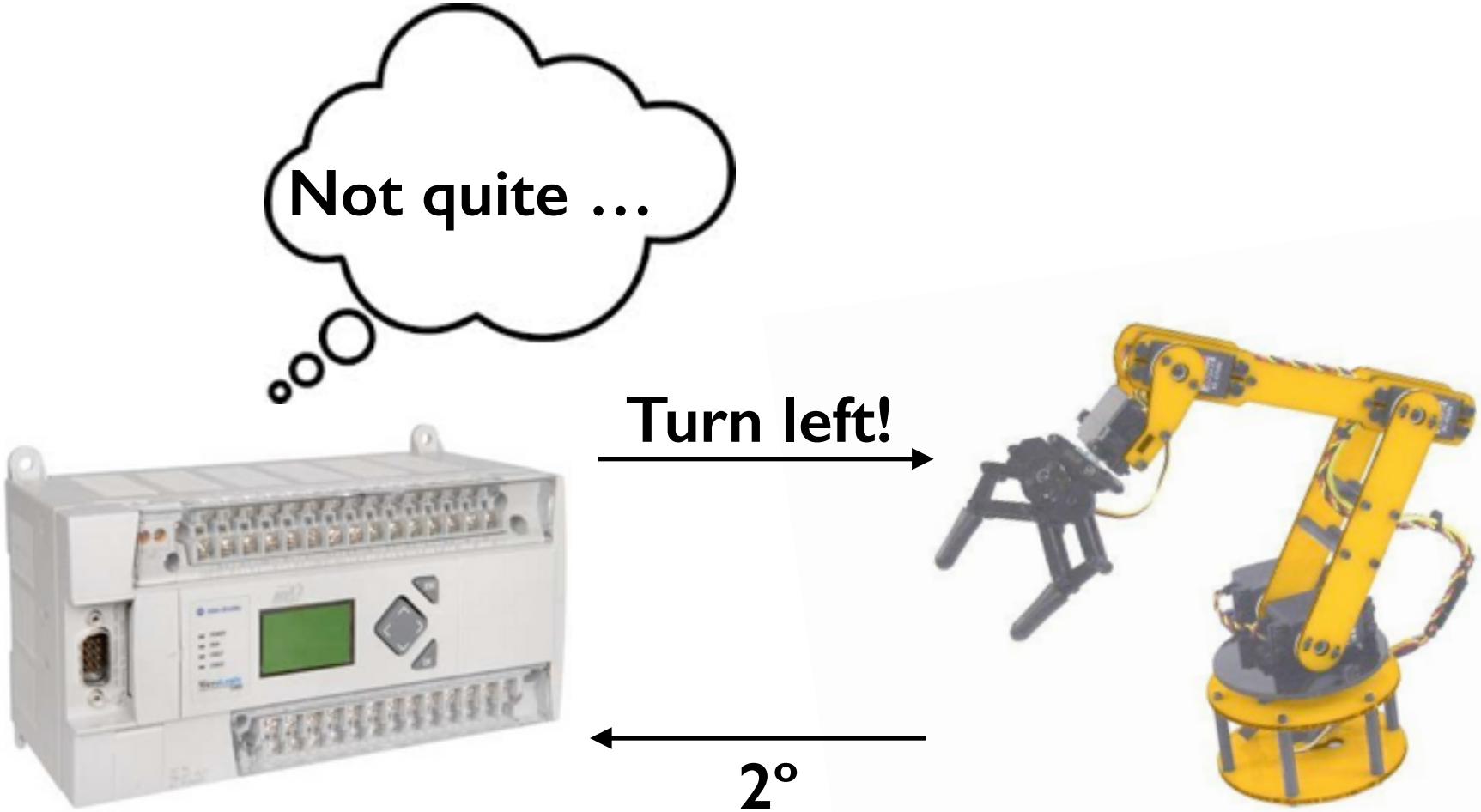
A PLC's life



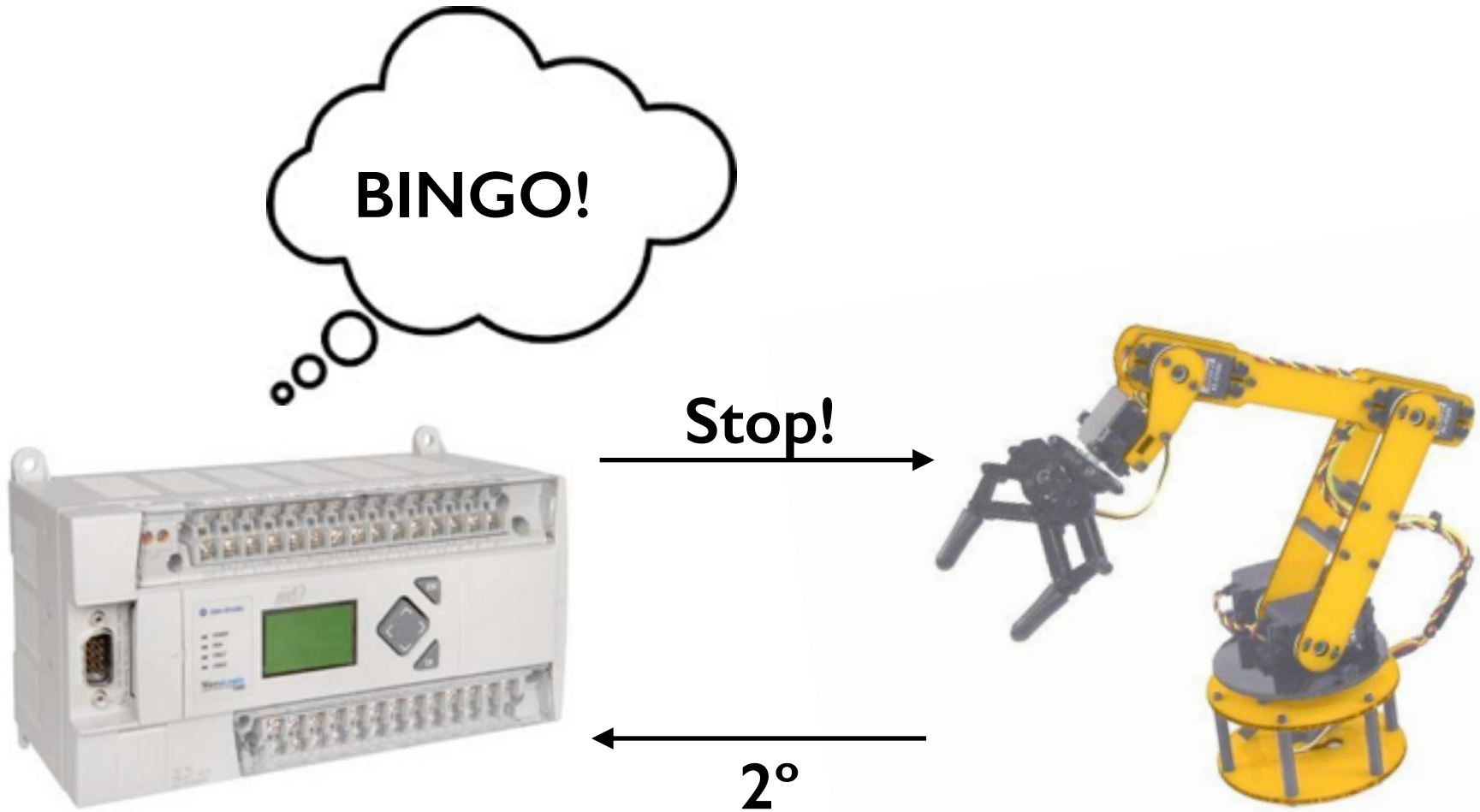
A PLC's life



A PLC's life



A PLC's life



A PLC's life

A PLC's life

Control Logic

```
L DW2        ;; Degrees
< ID255
ST Q 3.2
```

A PLC's life

Control Logic

```
L DW2        ;; Degrees
< ID255
ST Q 3.2
```

Control Signal

A PLC's life

Control Logic

```
L DW2        ;; Degrees
< ID255
ST Q 3.2
```

Control Signal

Sensor Measurements

A PLC's life

Control Logic

```
L DW2        ;; Degrees
< ID255
ST Q 3.2
```

Control Signal

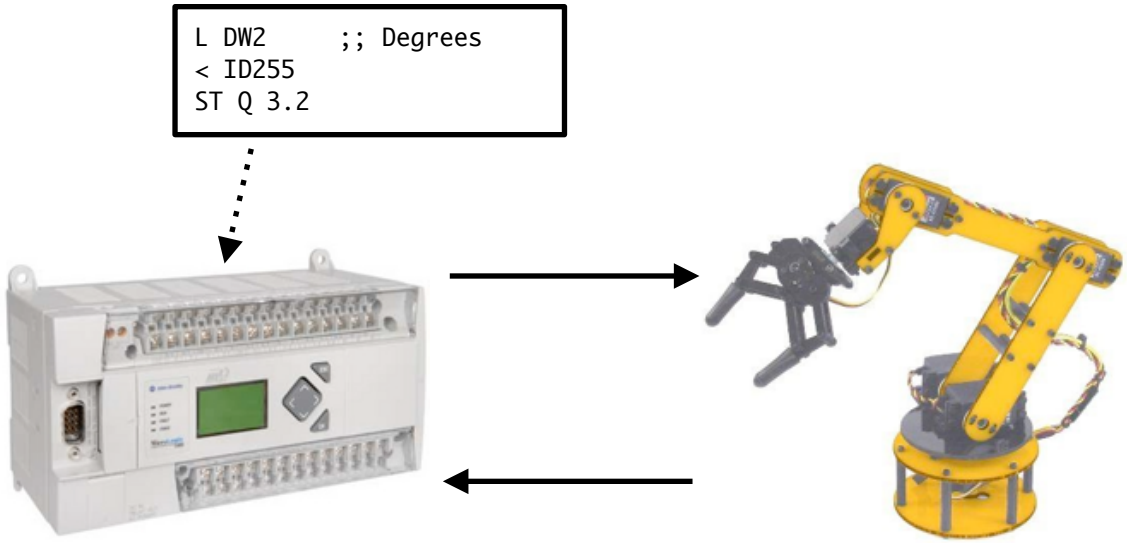
Scan Cycle

Sensor Measurements

A PLC's life

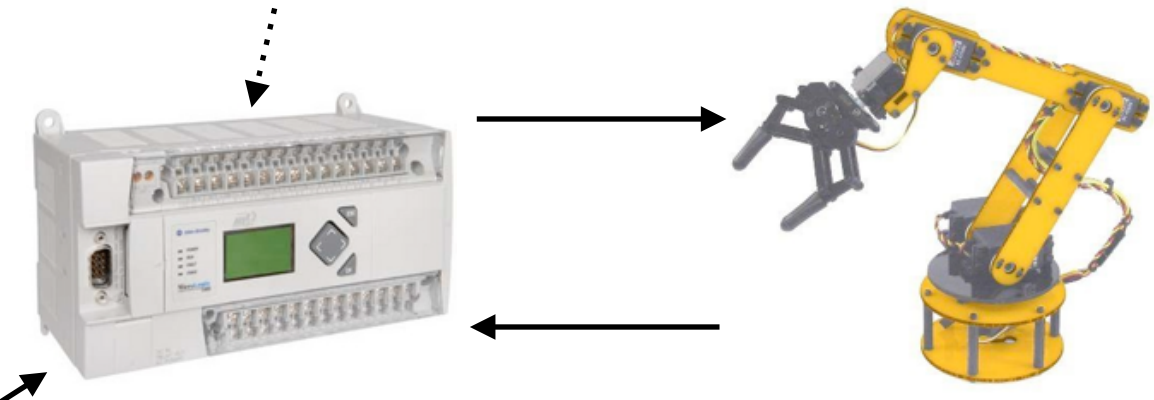
Applications

- Chemical processing
- Railroad safety
- Manufacturing
- Traffic Control
- PID Control



A PLC's life

```
L DW2      ;; Degrees
< ID255
ST Q 3.2
```



```
;; Spin out
;; of control!

S Q 3.2
```

A PLC's life

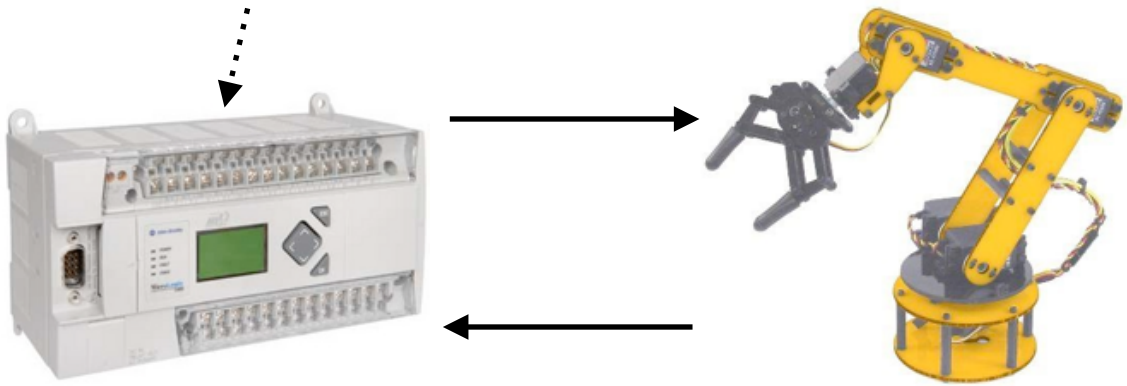
```
L DW2      ;; Degrees
< ID255
ST Q 3.2
```

```
;; Spin out
;; of control!

S Q 3.2
```

A PLC's life

```
L DW2      ;; Degrees
< ID255
ST Q 3.2
```



TSV

```
;; Spin out
;; of control!

S Q 3.2
```

"The Arm must remain within a quarter hemisphere at all times"

Trusted Safety Verifier

PENNSTATE

- **Goal: O**nly allow code to be run on a PLC if it satisfies a set of engineer-supplied safety properties.
- Challenges:
  - Existing tools not up to the task.
  - Control systems are *stateful*, requiring temporal properties.
  - State space explosion with existing analysis techniques.

Consider some PLC code

```
A I   0.5  ;; And input bit 5
= Q   0.1  ;; Store at output bit 1
. . .
```

Consider some PLC code

```
A I   0.5  ;; And input bit 5
= Q   0.1  ;; Store at output bit 1
. . .
```

- **Side effects**

Consider some PLC code

```
A I   0.5  ;; And input bit 5
= Q   0.1  ;; Store at output bit 1
. . .
```

- **Side effects**
- **PLC special features**

Consider some PLC code

```
A I   0.5  ;; And input bit 5
= Q   0.1  ;; Store at output bit 1
. . .
```

- **Side effects**
- **PLC special features**
- **Architecture dependent**

Consider some PLC code

```
A I   0.5  ;; And input bit 5
= Q   0.1  ;; Store at output bit 1
. . .
```

- **Side effects**
- **PLC special features**
- **Architecture dependent**

Instruction List
Intermediate Language

```
// A I 0.5
STA := load(mem, [I::0::0::0::5]);
cjmp FC == 0 : reg1_t,L1,L2;
label L1;
RLO := STA;
label L2;
RLO := RLO && STA;
FC :=1 : reg1_t;

// = Q 0.1
STA := RLO;
mem := store(mem, [Q::0::0::0::1], RLO);
FC := 0 : reg1_t;
```

ILIL

PENNSTATE

Based on the Vine IL

$prog ::= inst^* \; fun^*$

$fun ::= \texttt{ident}(var)\{inst^*\}$

$inst ::= \texttt{cjmp}\ e,e,e \mid \texttt{jmp}\ e \mid \texttt{label ident} \mid \texttt{ident} := e$

$\mid \texttt{call ident}(var{=}e) \mid \texttt{ret} \mid \texttt{assert}\ e$

$e ::= \texttt{load}(\texttt{ident}, addr) \mid \texttt{store}(\texttt{ident}, addr, \texttt{int}) \mid e\ binop\ e$

$\mid unop\ e \mid var \mid val \mid (e)$

$binop ::= +, -, *, /, \texttt{mod}, \&, \&\&, <<, \ldots$ (And signed versions.)

$unop ::= -$ (Negate), $\sim$ (Bitwise), ! (Boolean)

$var ::= \texttt{ident}\ (:\ \tau)$

$val ::= mem \mid addr \mid \texttt{int}\ (:\ \tau)$

$mem ::= \{addr \mapsto \texttt{int},\ addr \mapsto \texttt{int}, \ldots\}$

$addr ::= [\texttt{int} :: \texttt{int} :: \ldots]$

$\tau ::= \texttt{reg1\_t} \ldots \texttt{reg64\_t} \mid \texttt{mem\_t}(\texttt{int}) \mid \texttt{addr\_t}$

ILIL Analysis

```
// A I 0.5
STA := load(mem, [I::0::0::0::5]);
cjmp FC == 0 : reg1_t,L1,L2;
label L1;
RLO := STA;
label L2;
RLO := RLO && STA;
FC :=1 : reg1_t;

// = Q 0.1
STA := RLO;
mem := store(mem, [Q::0::0::0::1], RLO);
FC := 0 : reg1_t;
```

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                      (not X_3))
 [X_0] -> (X_2)
```

ILIL Analysis

```
// A I 0.5
STA := load(mem, [I::0::0::0::5]);
cjmp FC == 0 : reg1_t,L1,L2;
label L1;
RLO := STA;
label L2;
RLO := RLO && STA;
FC :=1 : reg1_t;

// = Q 0.1
STA := RLO;
mem := store(mem, [Q::0::0::0::1], RLO);
FC := 0 : reg1_t;
```

Symbolic Sensor Values

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

ILIL Analysis

```
// A I 0.5
STA := load(mem, [I::0::0::0::5]);
cjmp FC == 0 : reg1_t,L1,L2;
label L1;
RLO := STA;
label L2;
RLO := RLO && STA;
FC :=1 : reg1_t;

// = Q 0.1
STA := RLO;
mem := store(mem, [Q::0::0::0::1], RLO);
FC := 0 : reg1_t;
```

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                      (not X_3))
 [X_0] -> (X_2)
```

Symbolic Control Signal

Checking Temporal Properties

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

```
G (!a)
a: (and X_0 X_5)
```

Checking Temporal Properties

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

Linear Temporal Logic

```
G (!a)
a: (and X_0 X_5)
```

$$\varphi ::= \mathit{true} \mid b \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi,$$

Checking Temporal Properties

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

```
G (!a)
a: (and X_0 X_5)
```

Checking Temporal Properties

- **Input Variables**
- **State Variables**

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

```
G (!a)
a: (and X_0 X_5)
```

???

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 . . .
```

???

```
(assert ((and (not (and X_3 X_2))
              (X_5)))) :
 [X_5] -> (X_2)
 . . .
```

Checking Temporal Properties

- **Input Variables**
- **State Variables**

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                       (not X_3))
 [X_0] -> (X_2)
```

```
G (!a)
a: (and X_0 X_5)
```

???

???

Reachability determined by SMT solver

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 . . .
```

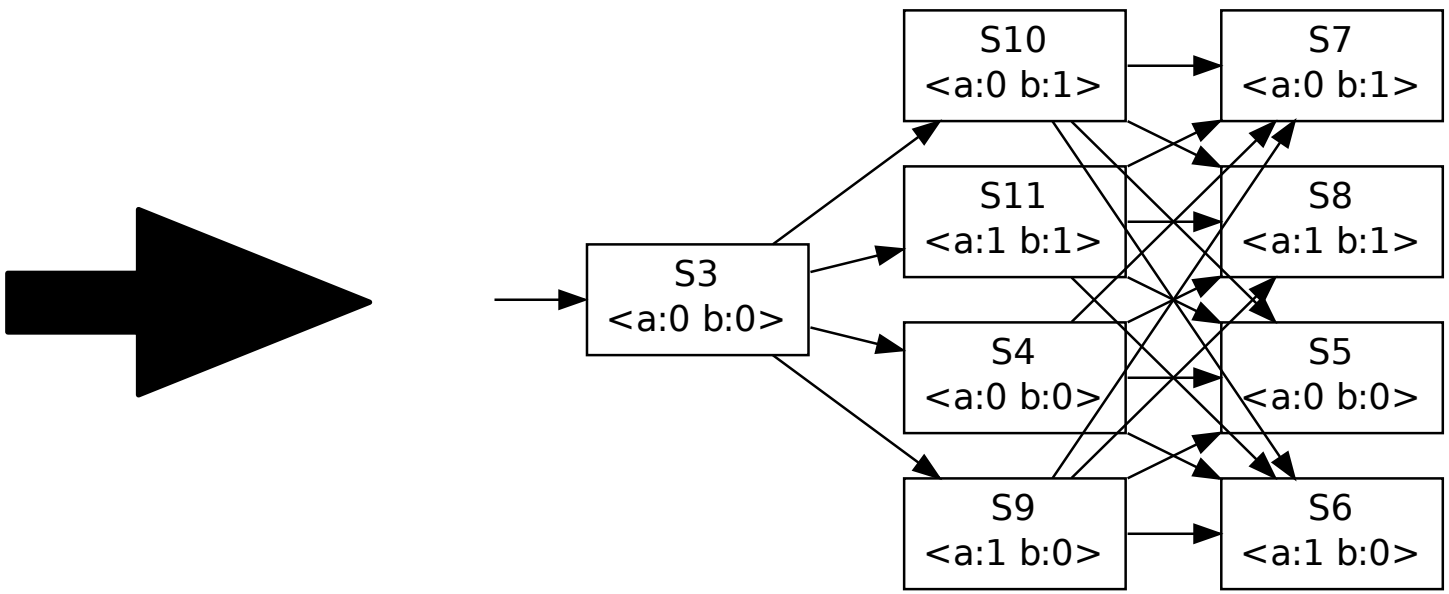
```
(assert ((and (not (and X_3 X_2))
              (X_5)))) :
 [X_5] -> (X_2)
 . . .
```

Temporal Execution Graph (TEG)

```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 [X_7] -> (and (and X_1 true)
                    (not X_3))
 [X_0] -> (X_2)
```

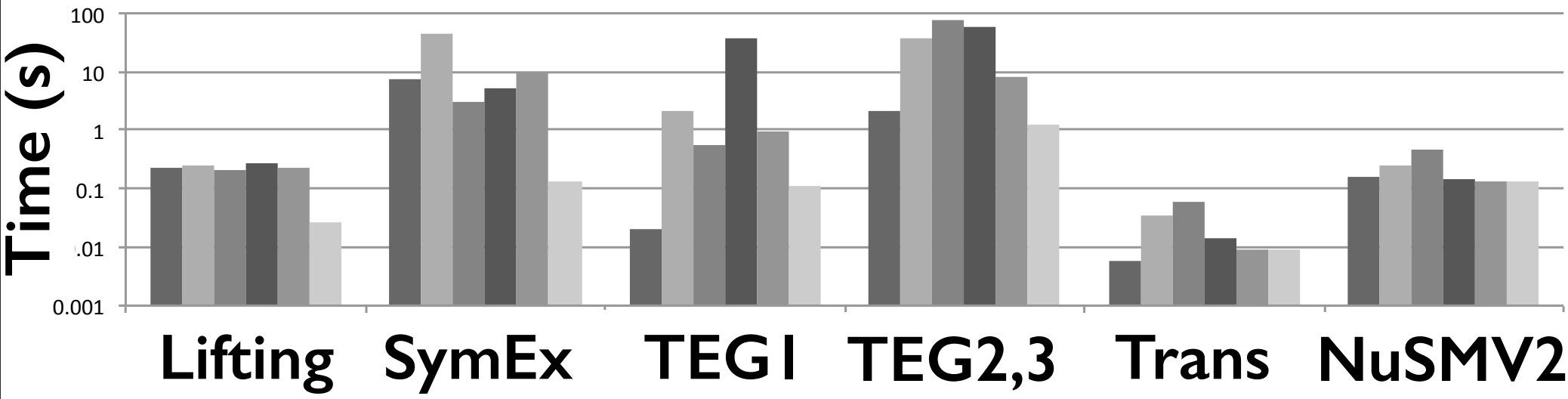
```
(assert ((and (not (and X_3 X_2))
              (not X_5)))) :
 [X_5] -> (X_2)
 . . .
```

```
(assert ((and (not (and X_3 X_2))
              (X_5)))) :
 [X_5] -> (X_2)
 . . .
```

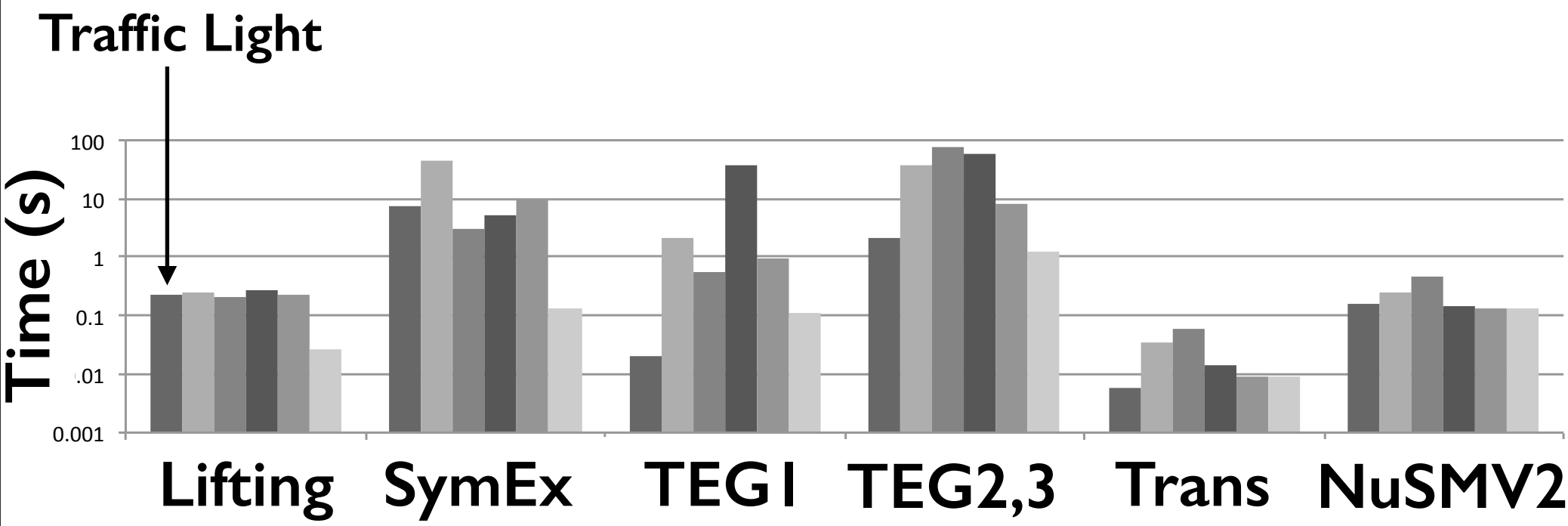


Performance

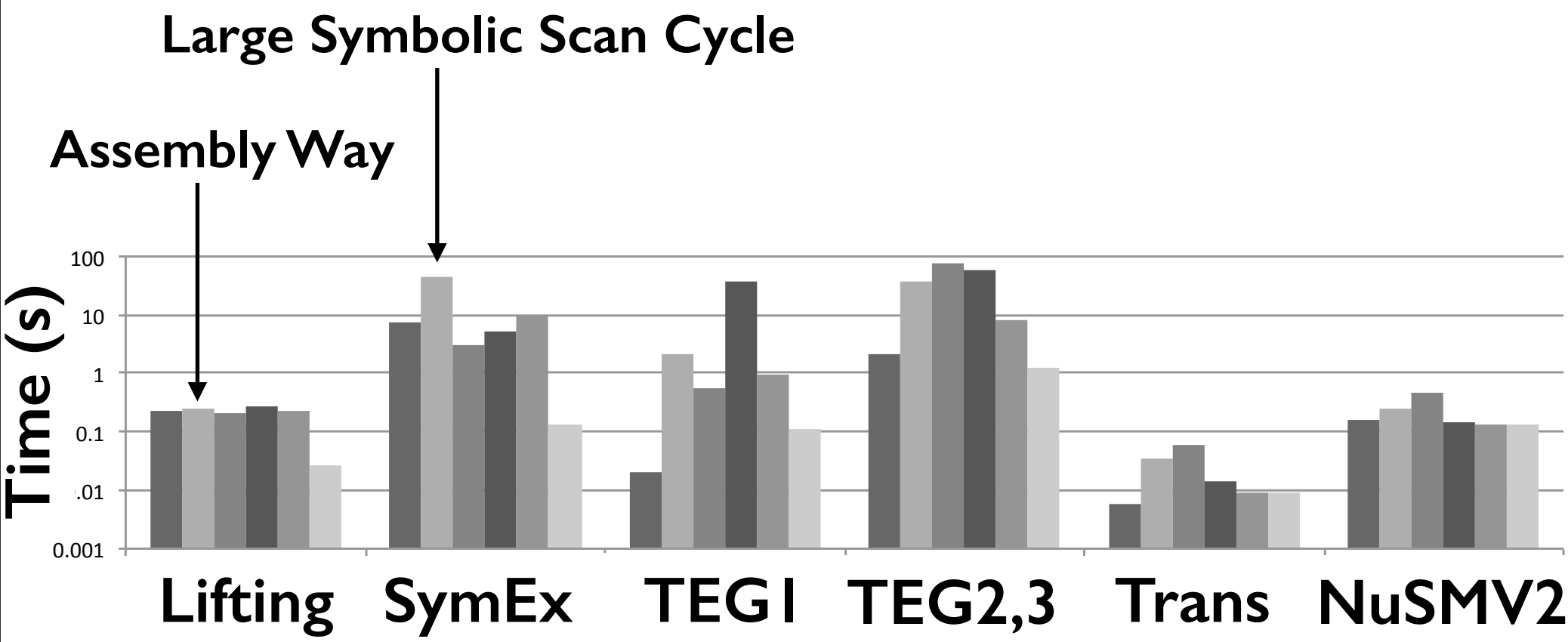
TEG Depth bounded at 14



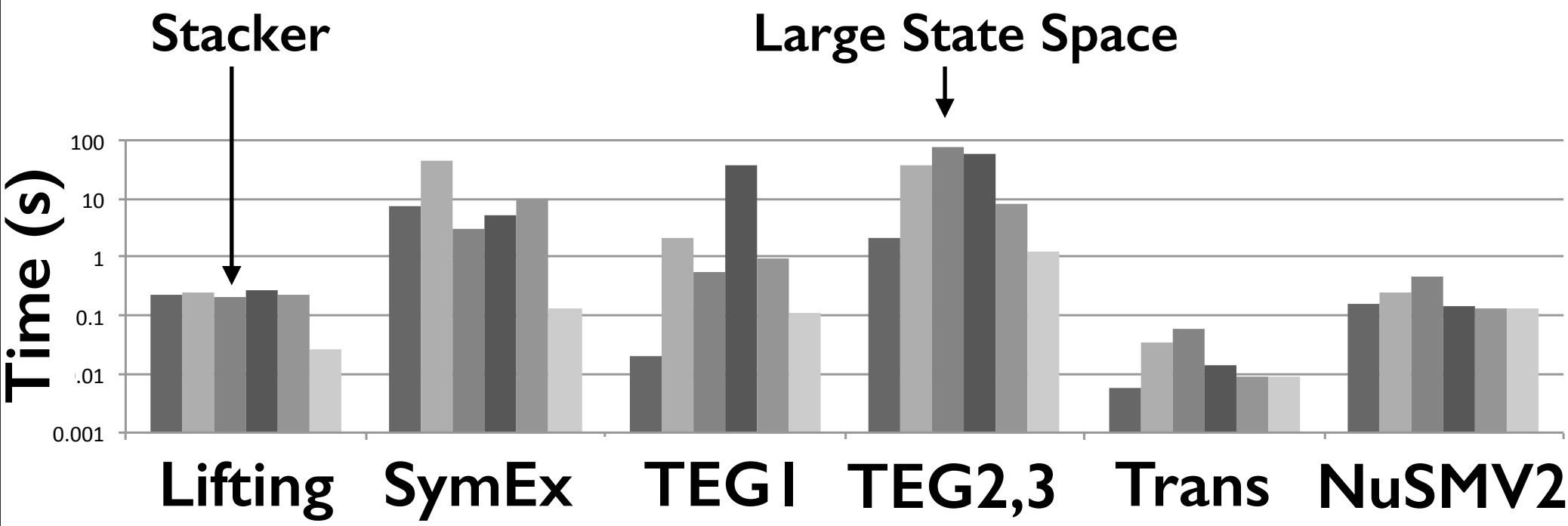
Performance



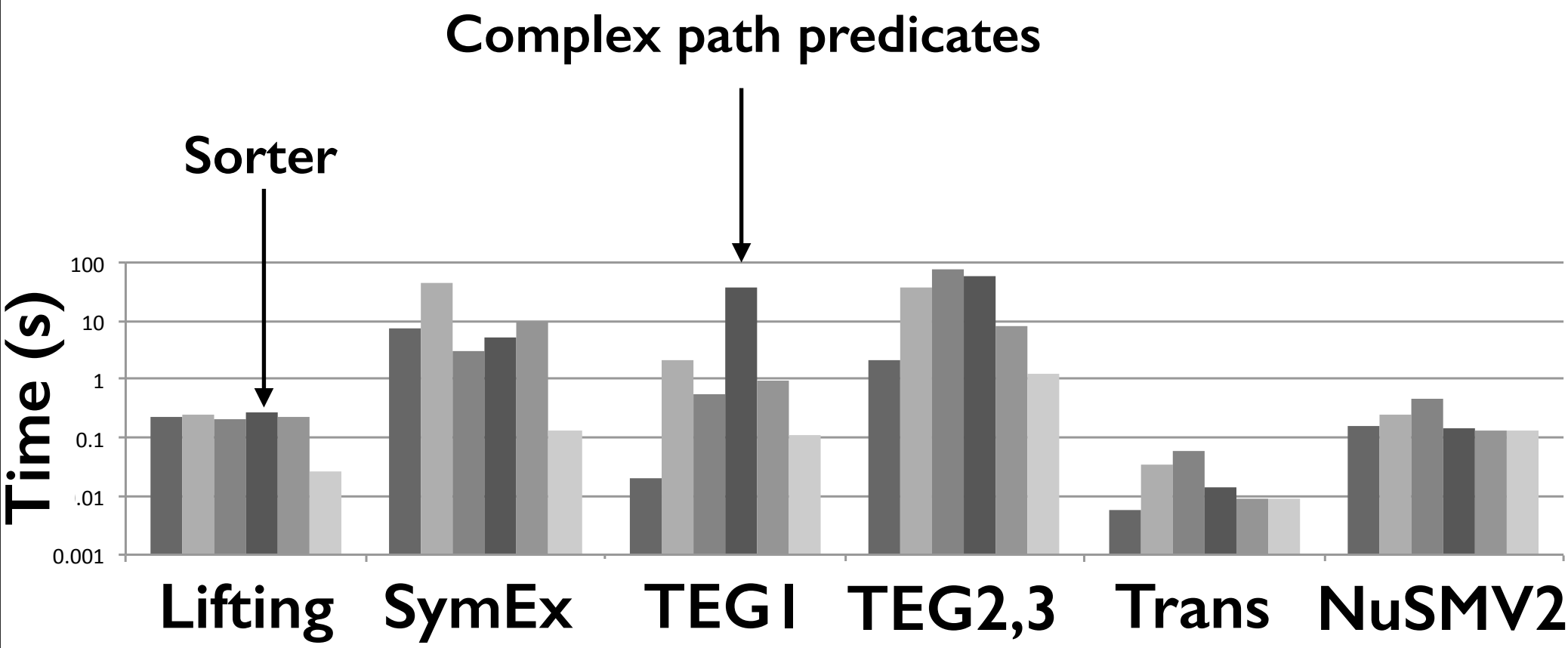
Performance



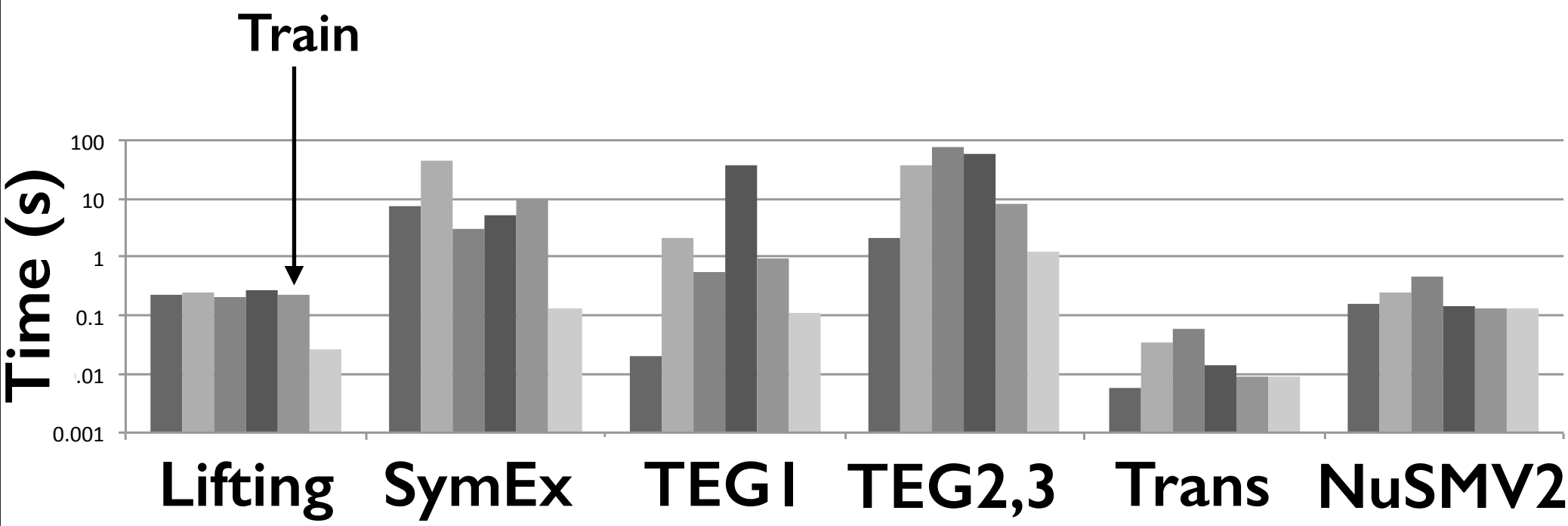
Performance



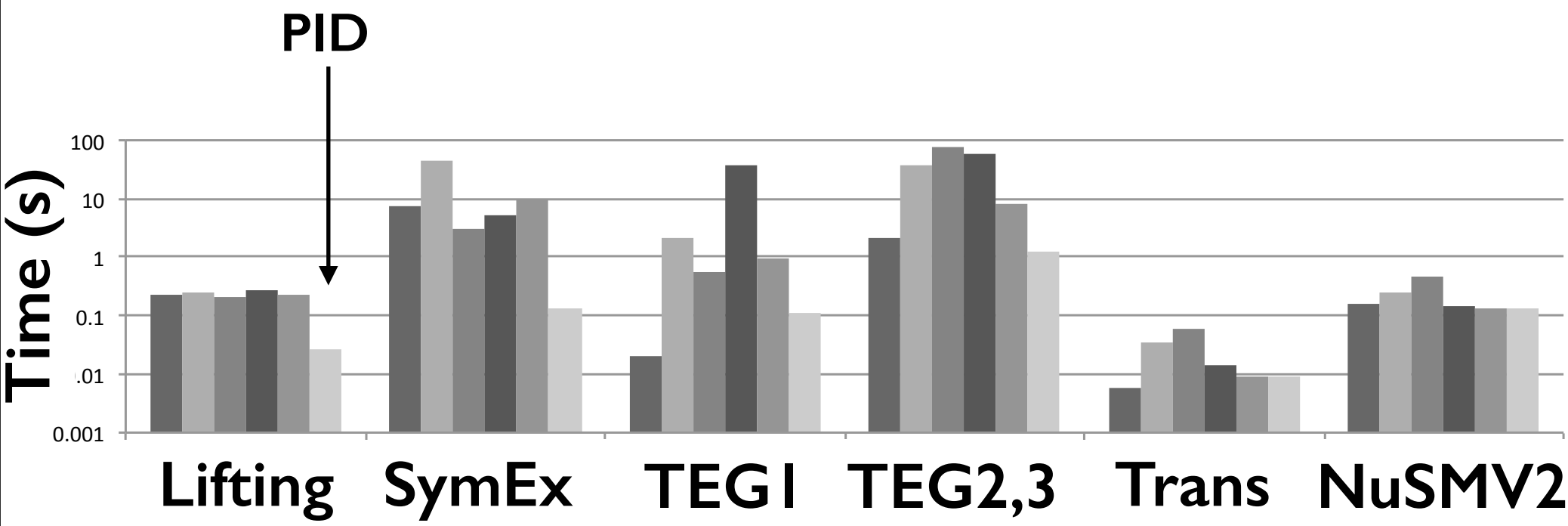
Performance



Performance



Performance



Related Work

	Approach	Boolean Logic	Enum	Numeric	Cond. Branching	Function Blocks	MCR	Nested Logic	Timers	Counters	Pointers	Data Blocks	Edge Detection
Park et al. [22]	SAT	✓											
Groote et al. [14]	SAT	✓							✓				
Homer [15]	Thm	✓	✓	✓	✓								
Biha [21]	Thm	✓	✓	✓	✓				✓				
SABOT [18]	Mod	✓							✓				
Canet et al. [6]	Mod	✓	✓		✓								
TSV	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Related Work

No efficient reduction

	Approach	Boolean Logic	Enum	Numeric	Cond. Branching	Function Blocks	MCR	Nested Logic	Timers	Counters	Pointers	Data Blocks	Edge Detection
Park et al. [22]	SAT	✓											
Groote et al. [14]	SAT	✓							✓				
Homer [15]	Thm	✓	✓	✓	✓								
Biha [21]	Thm	✓	✓	✓	✓				✓				
SABOT [18]	Mod	✓							✓				
Canet et al. [6]	Mod	✓	✓		✓								
TSV	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Related Work

PENNSTATE

Hard to formalize IL instructions with side-effects

	Approach	Boolean Logic	Enum	Numeric	Cond. Branching	Function Blocks	MCR	Nested Logic	Timers	Counters	Pointers	Data Blocks	Edge Detection
Park et al. [22]	SAT	✓											
Groote et al. [14]	SAT	✓							✓				
Homer [15]	Thm	✓	✓	✓	✓								
Biha [21]	Thm	✓	✓	✓	✓				✓				
SABOT [18]	Mod	✓							✓				
Canet et al. [6]	Mod	✓	✓		✓								
TSV	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Related Work

No symbolic state lumping

	Approach	Boolean Logic	Enum	Numeric	Cond. Branching	Function Blocks	MCR	Nested Logic	Timers	Counters	Pointers	Data Blocks	Edge Detection
Park et al. [22]	SAT	✓											
Groote et al. [14]	SAT	✓							✓				
Homer [15]	Thm	✓	✓	✓	✓								
Biha [21]	Thm	✓	✓	✓	✓				✓				
SABOT [18]	Mod	✓							✓				
Canet et al. [6]	Mod	✓	✓		✓								
TSV	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Thanks!

Steve: smclaugh@cse.psu.edu
Saman: s.zonouz@miami.edu