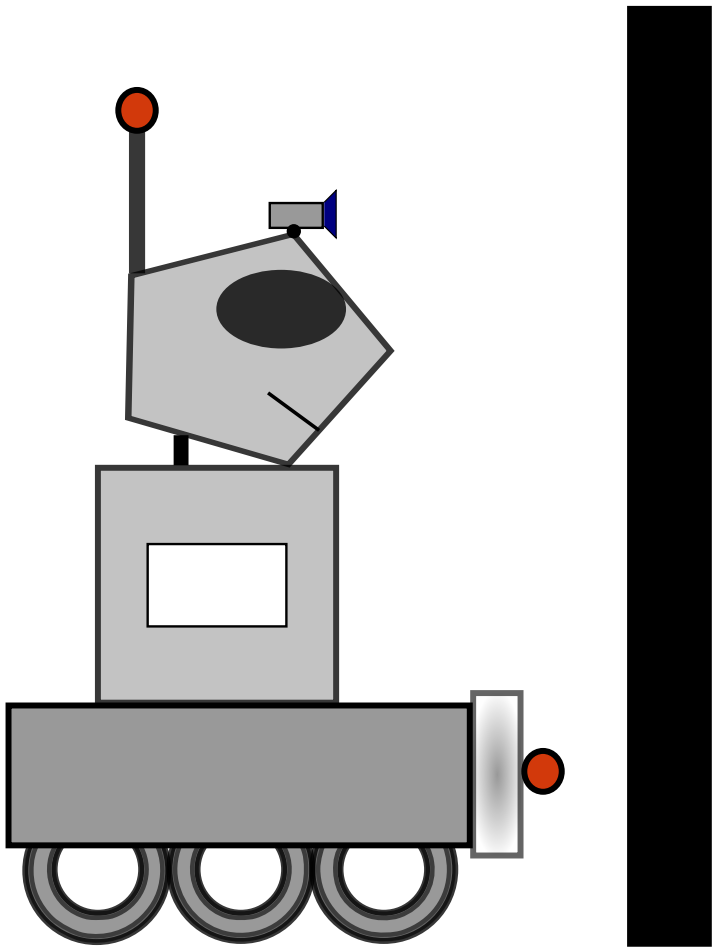
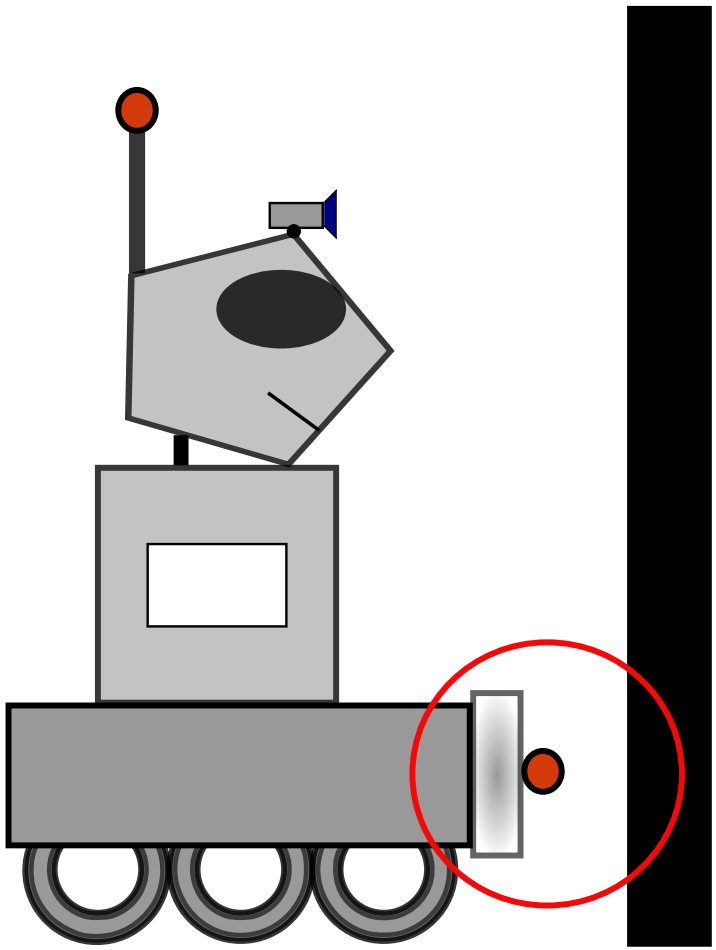
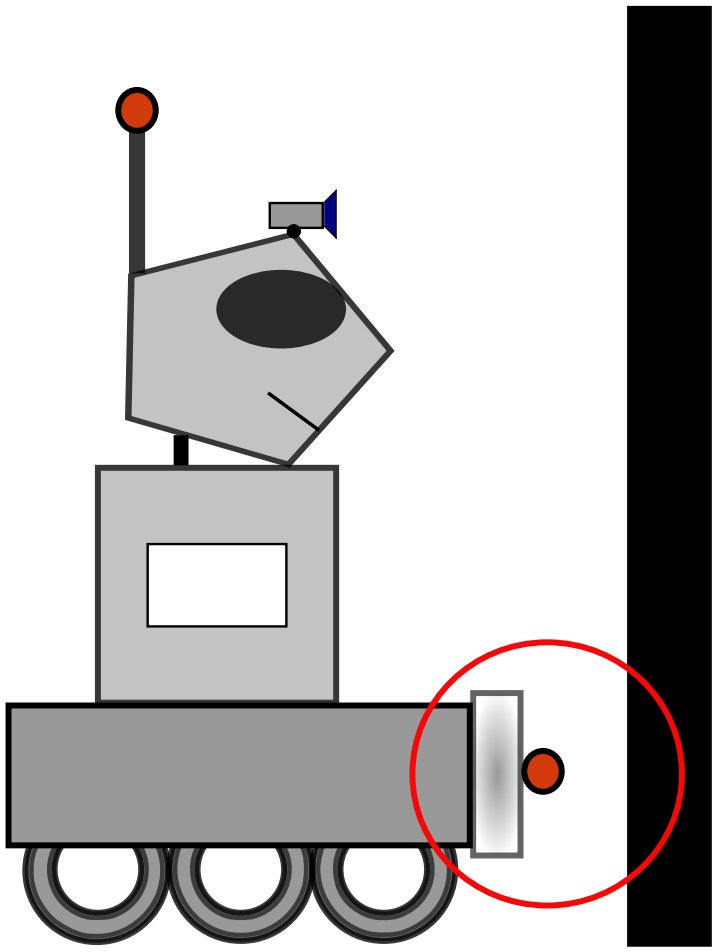
Interactive Timing Analysis for Designing Reactive Systems

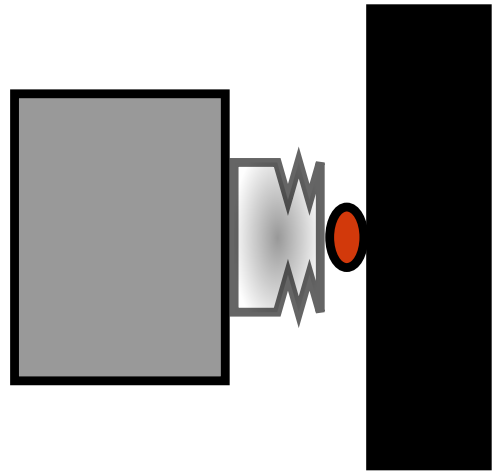
Insa Fuhrmann, David Broman, Reinhard von Hanxleden

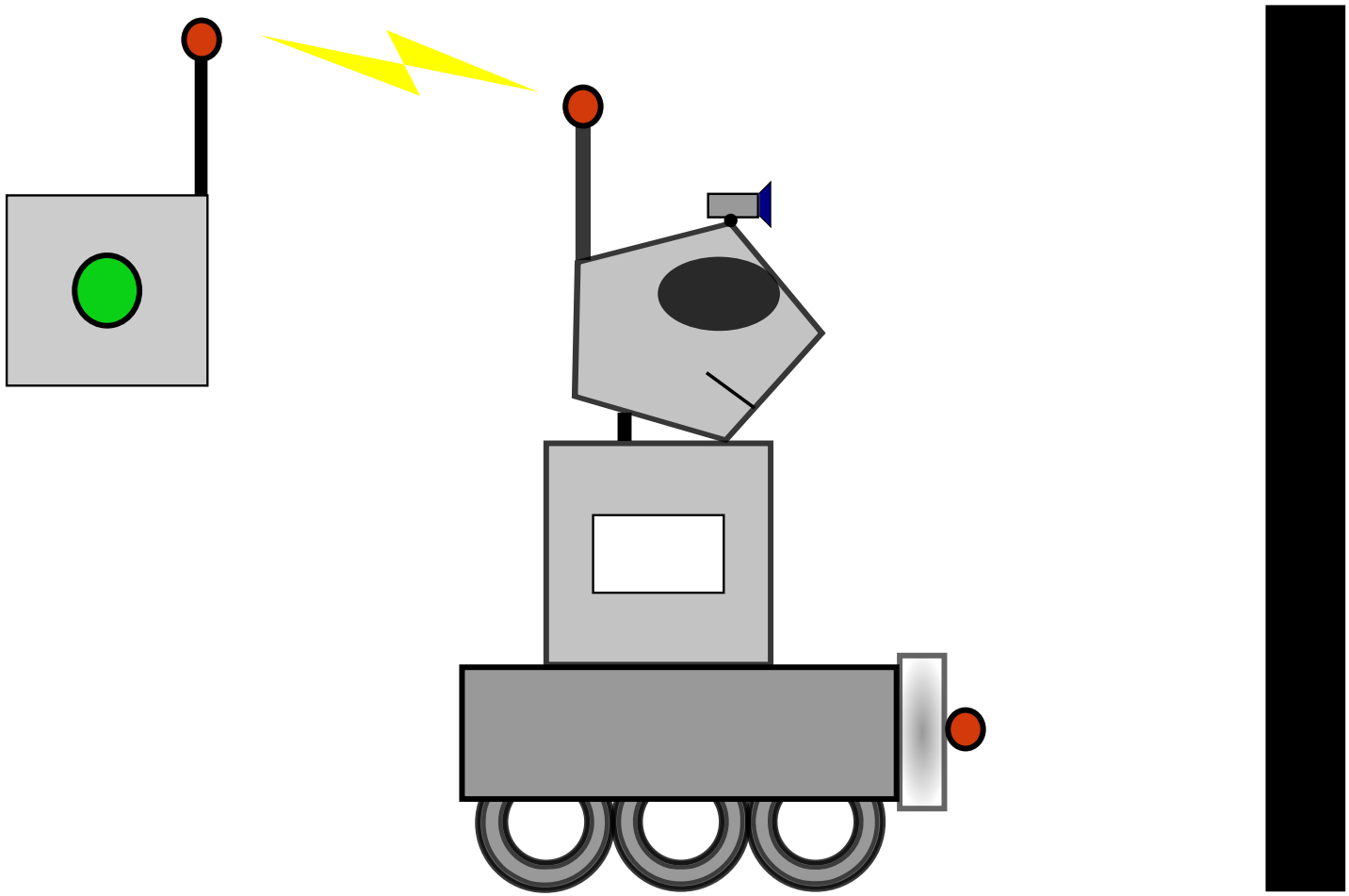
SYNCHRON 2015

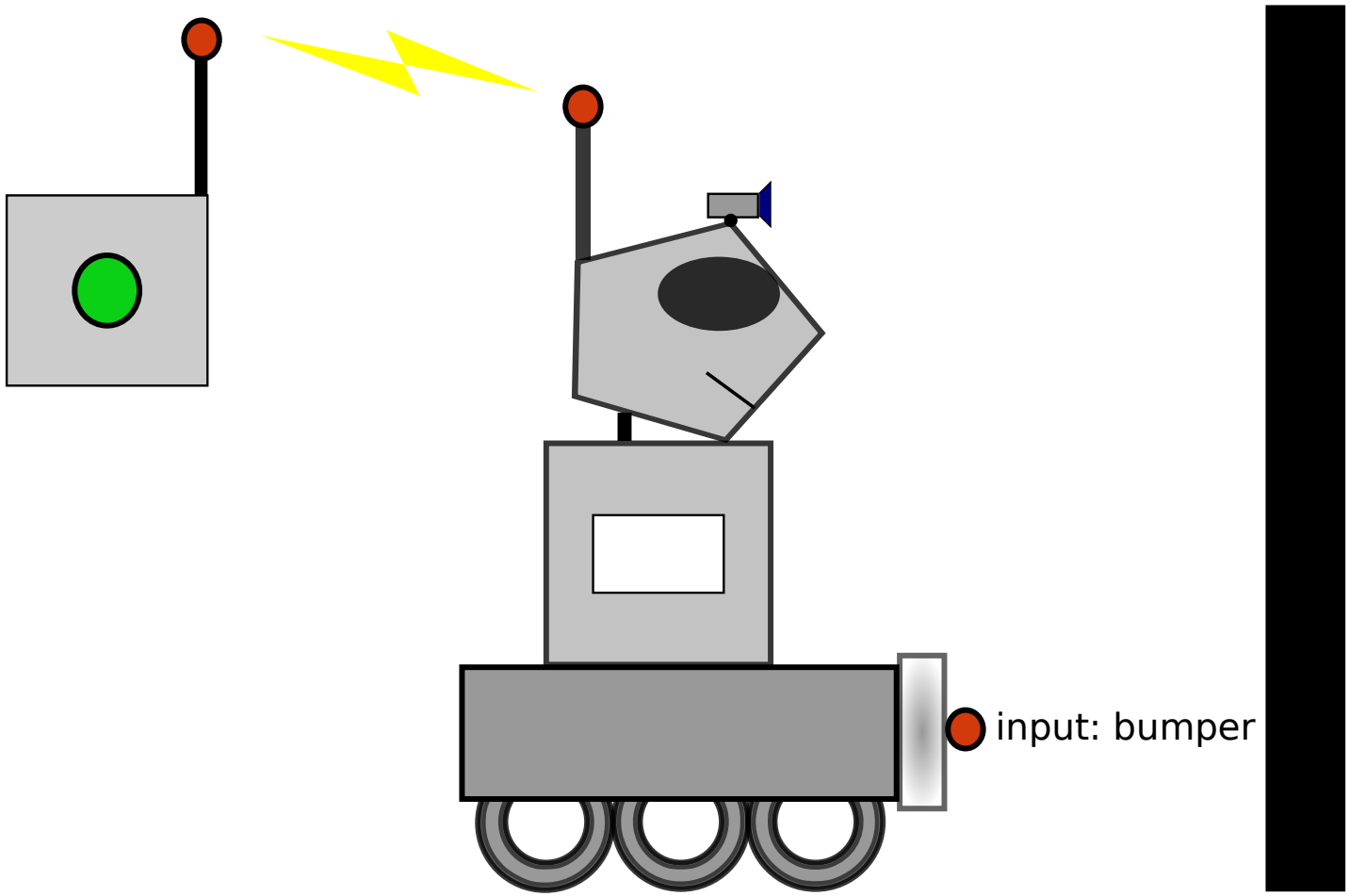


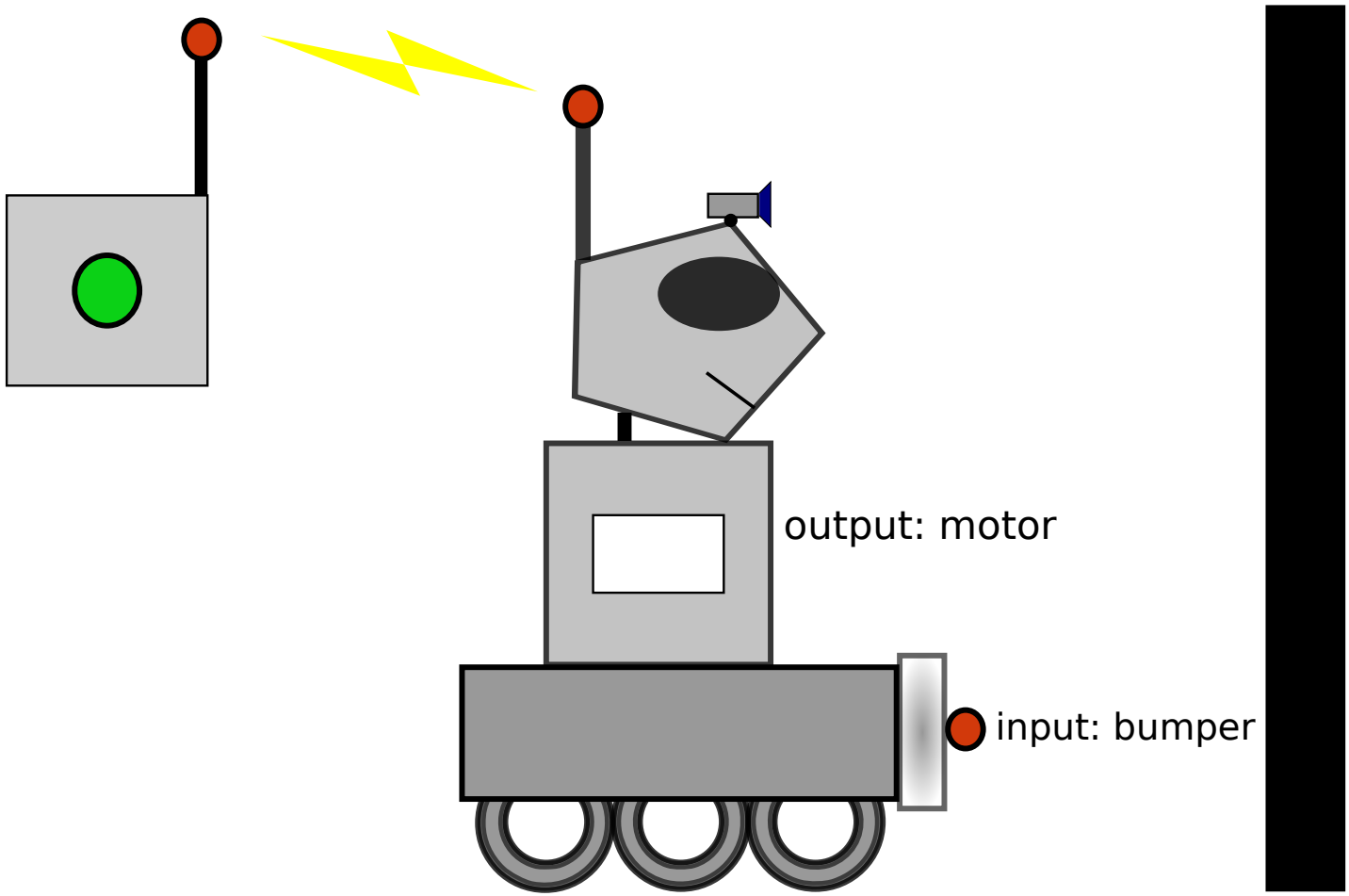


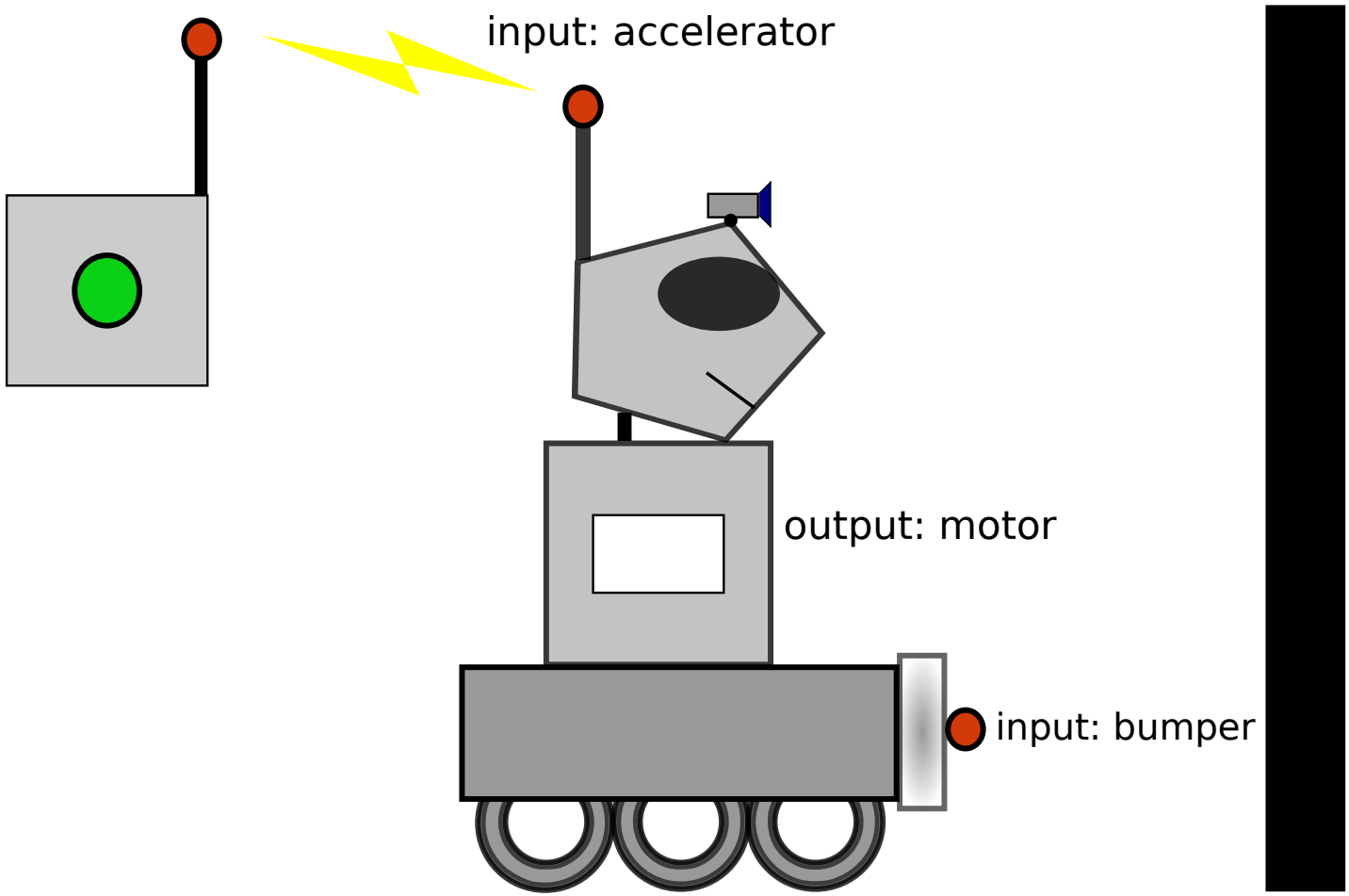


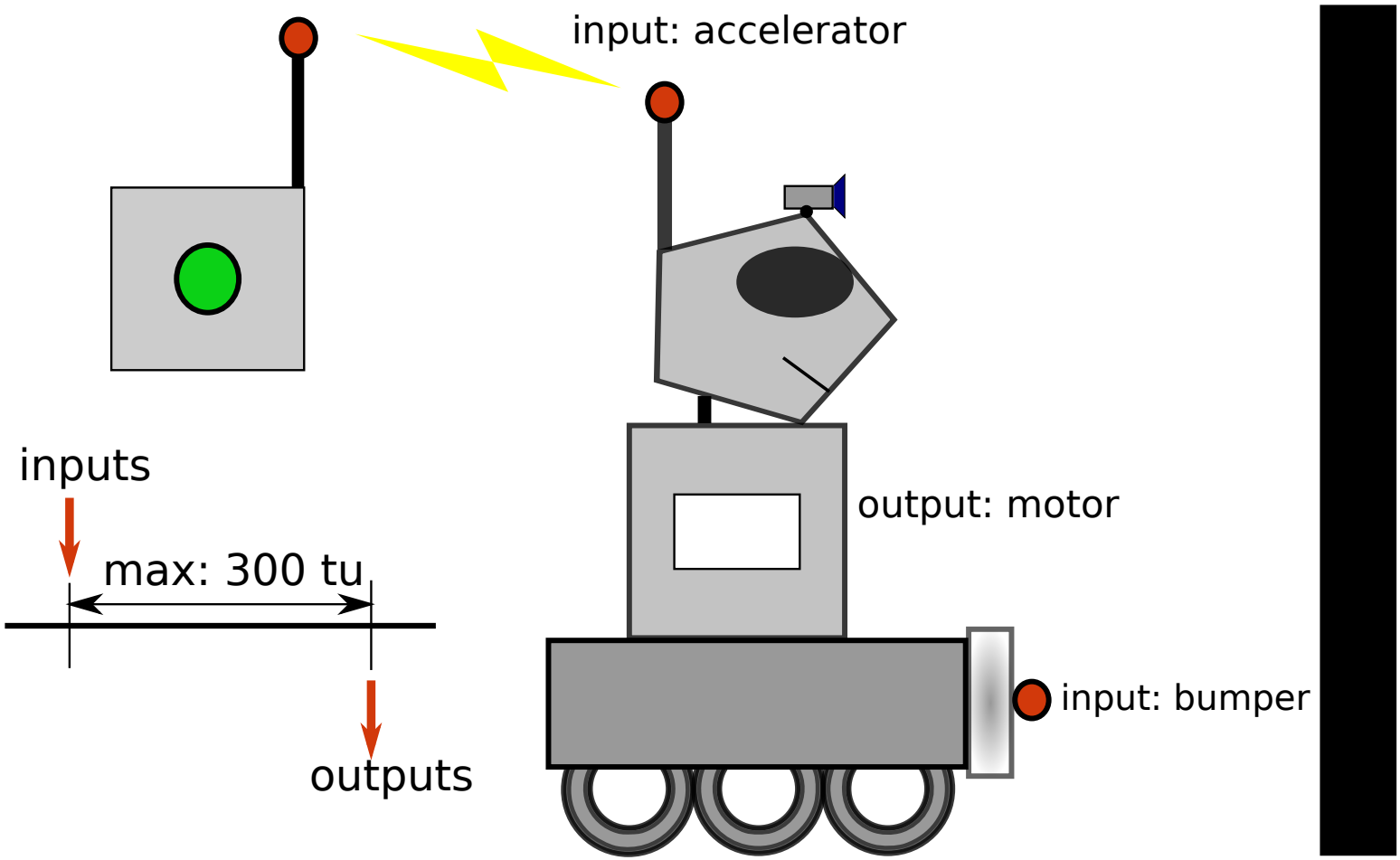


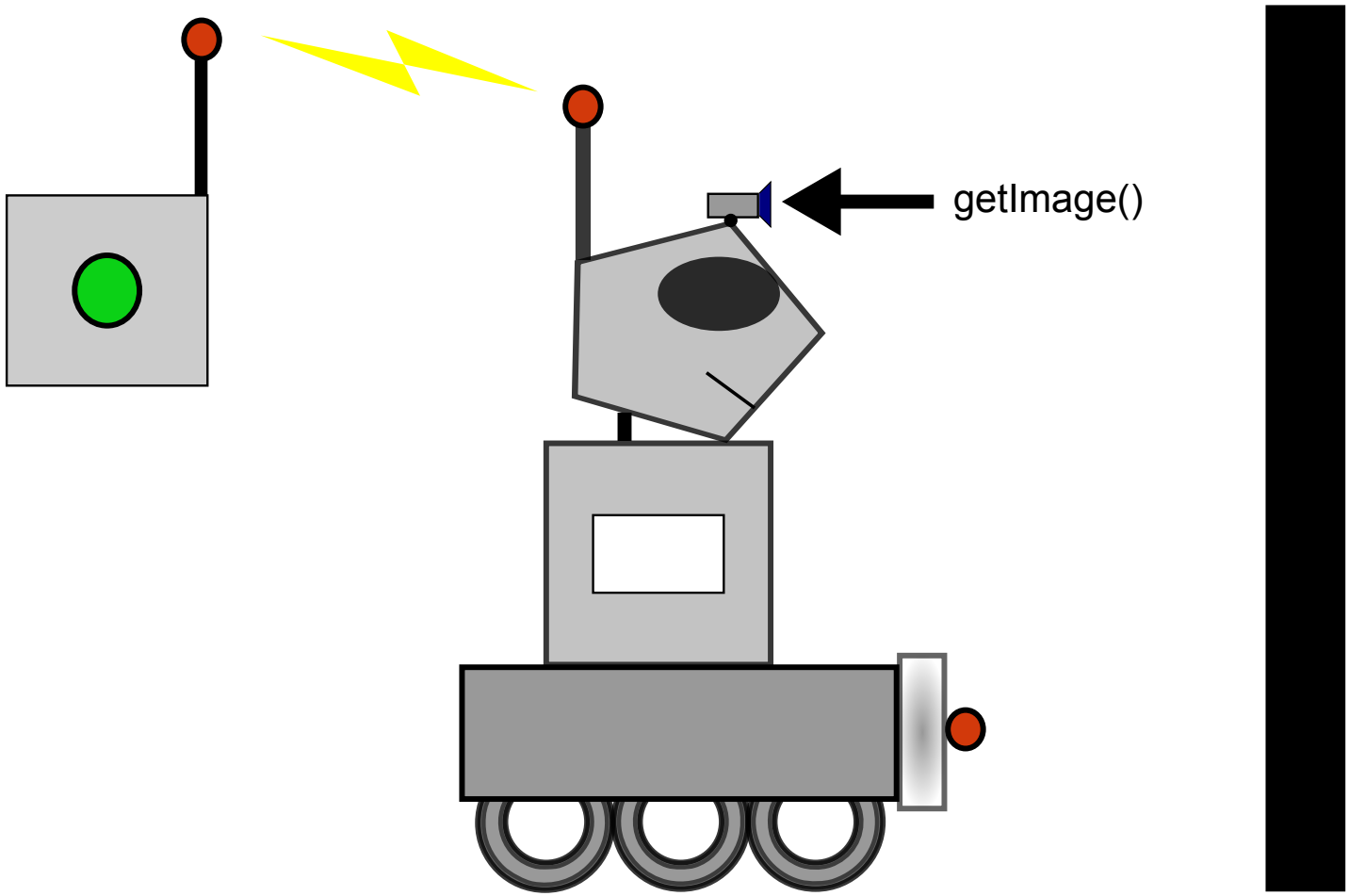


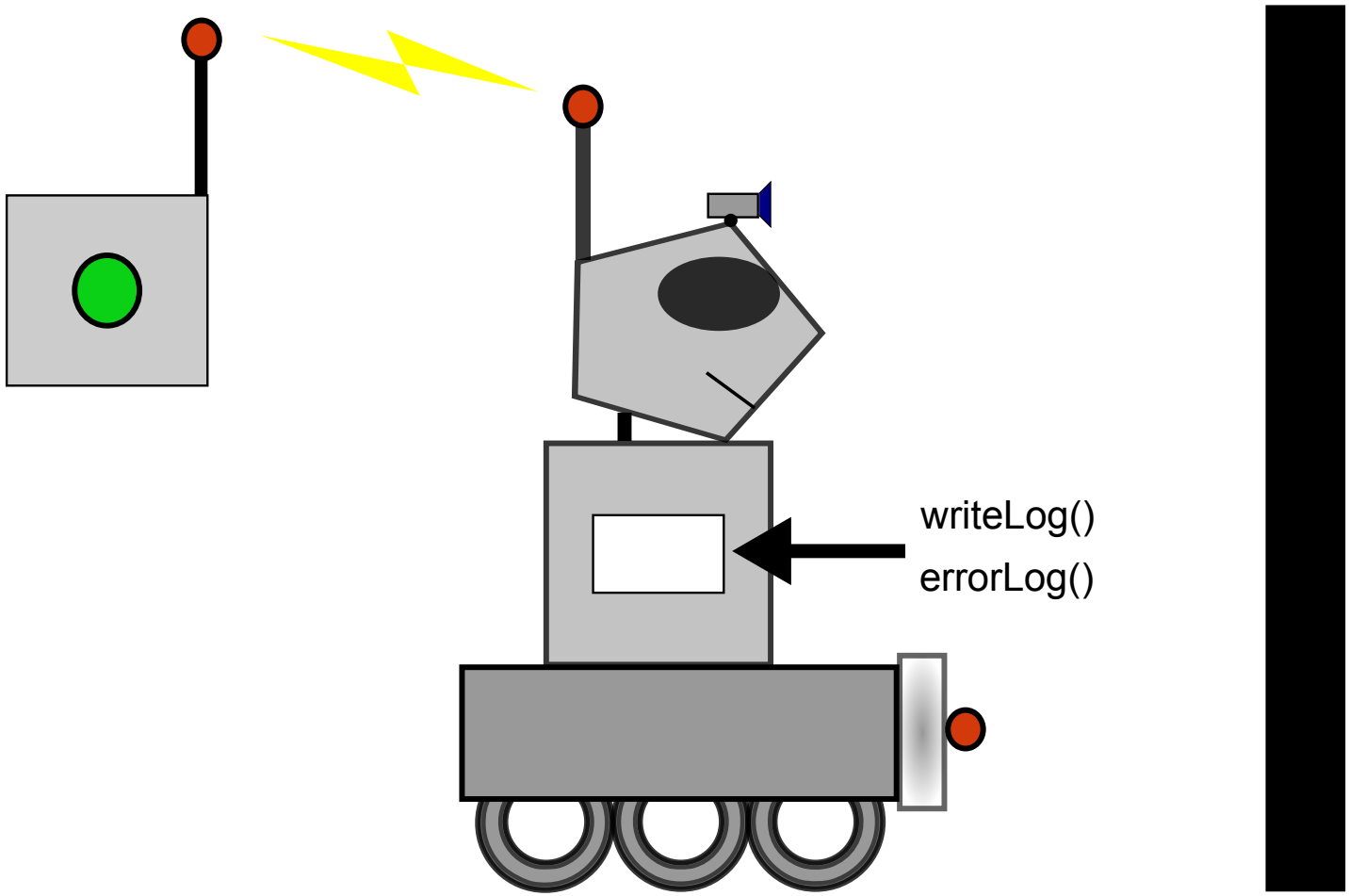

input: bumper

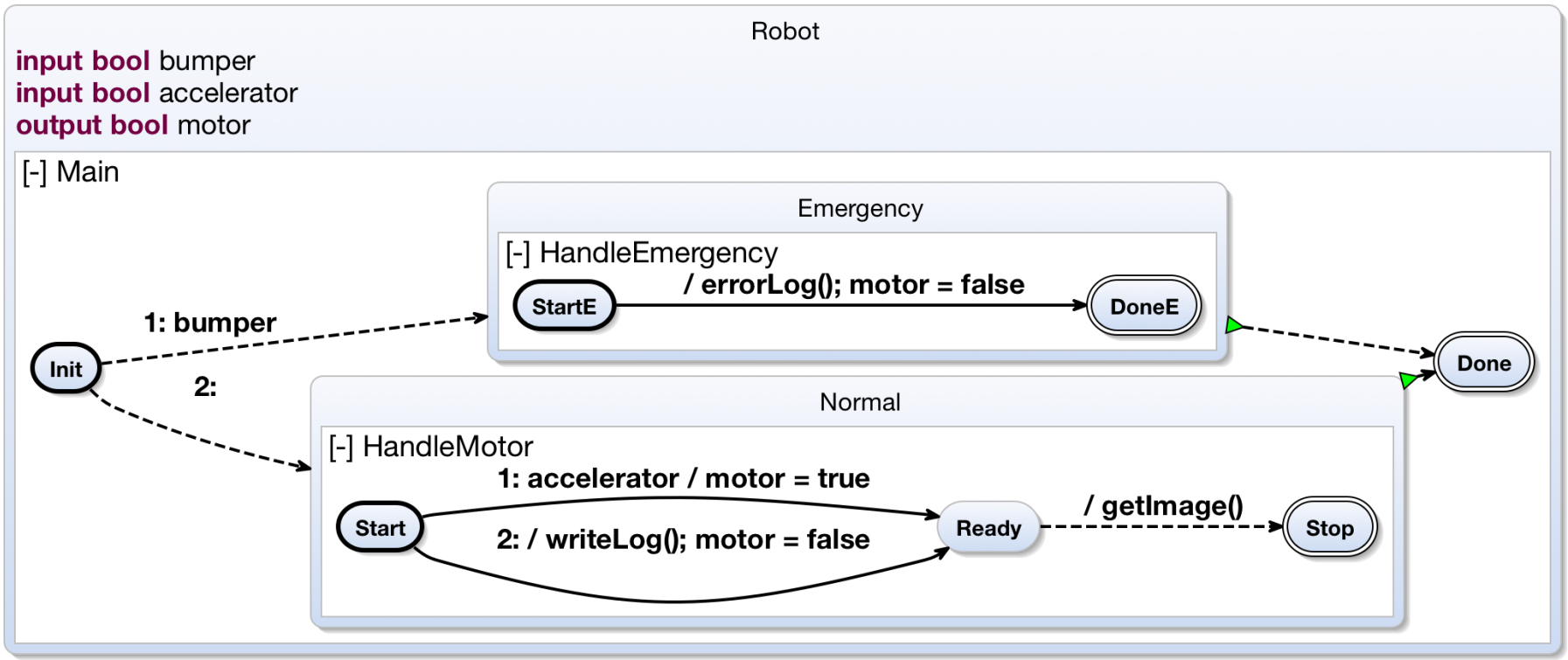

output: motor
input: bumper

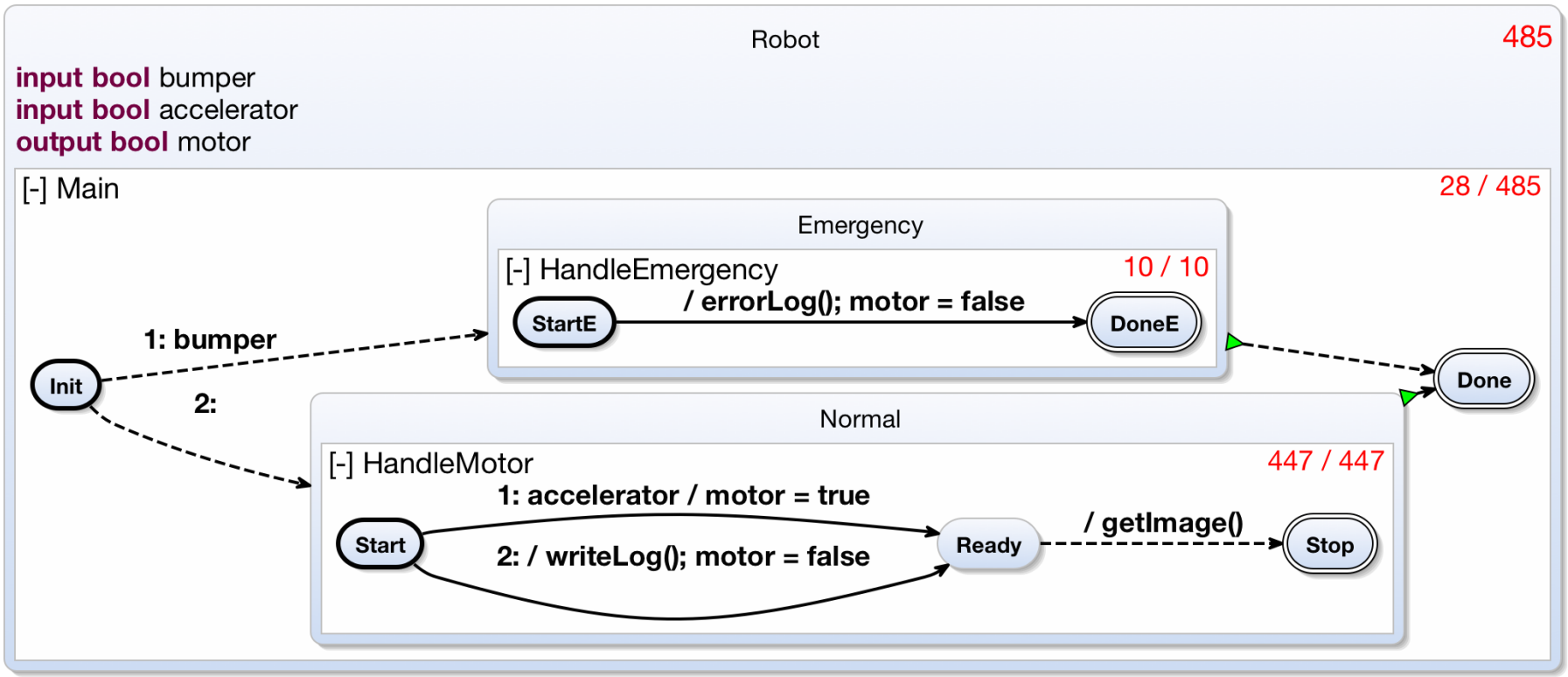

input: accelerator
output: motor
input: bumper


input: accelerator
inputs
max: 300 tu
outputs
output: motor
input: bumper


getImage()

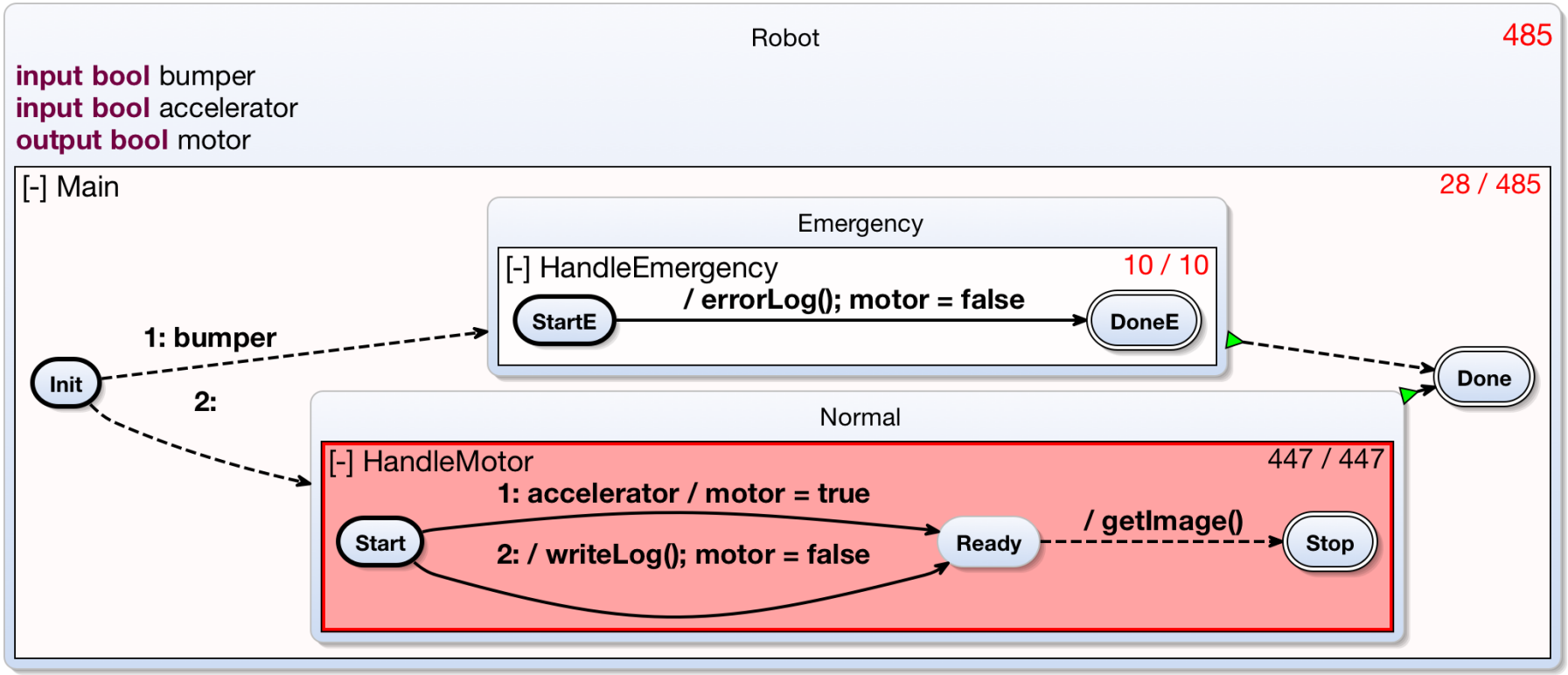

writeLog()
errorLog()

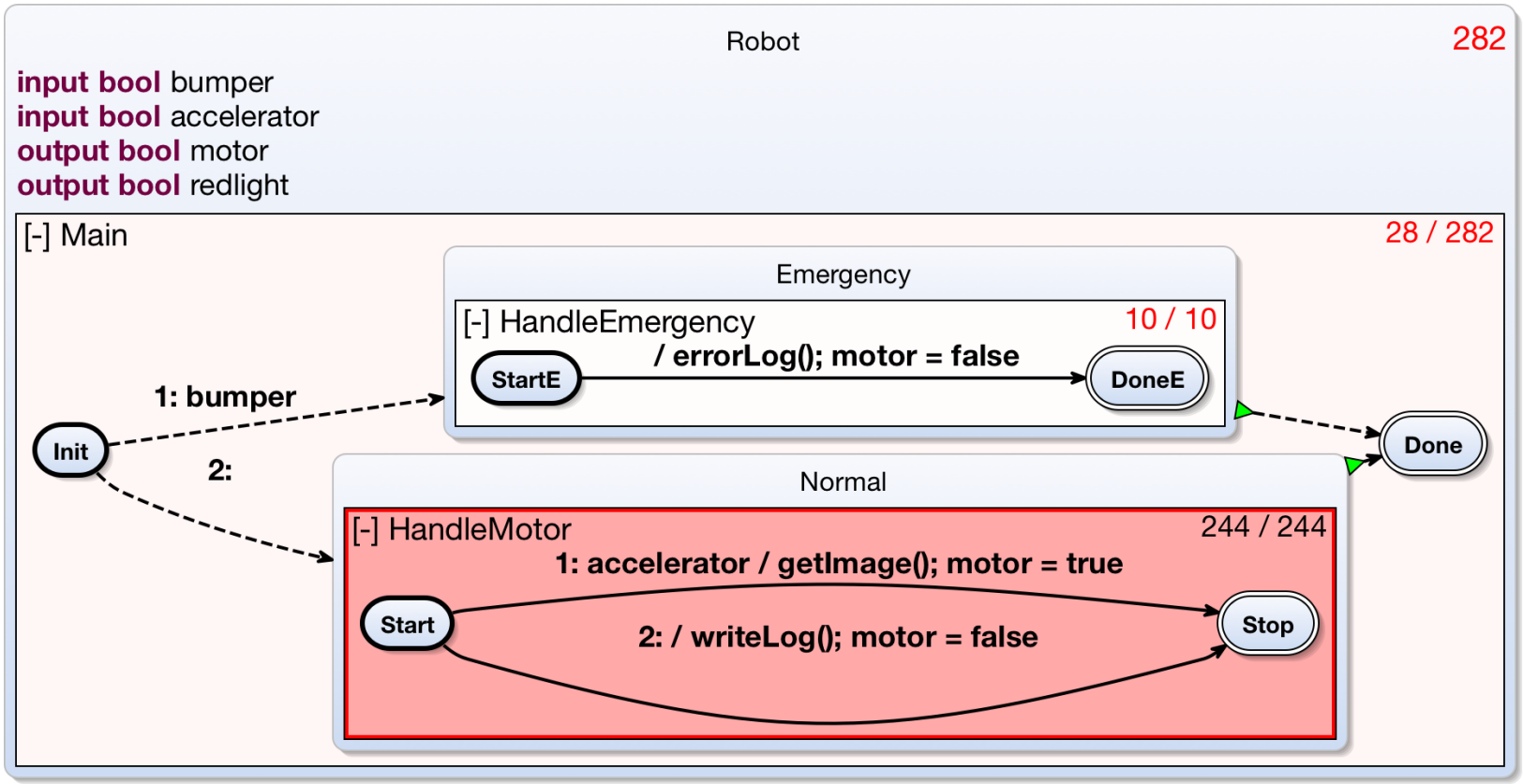

Robot
input bool bumper
input bool accelerator
output bool motor
[-] Main
Emergency
[-] HandleEmergency
StartE
/ errorLog(); motor = false
DoneE
1: bumper
Init
2:
Done
Normal
[-] HandleMotor
1: accelerator / motor = true
Start
2: / writeLog(); motor = false
Ready
/ getImage()
Stop

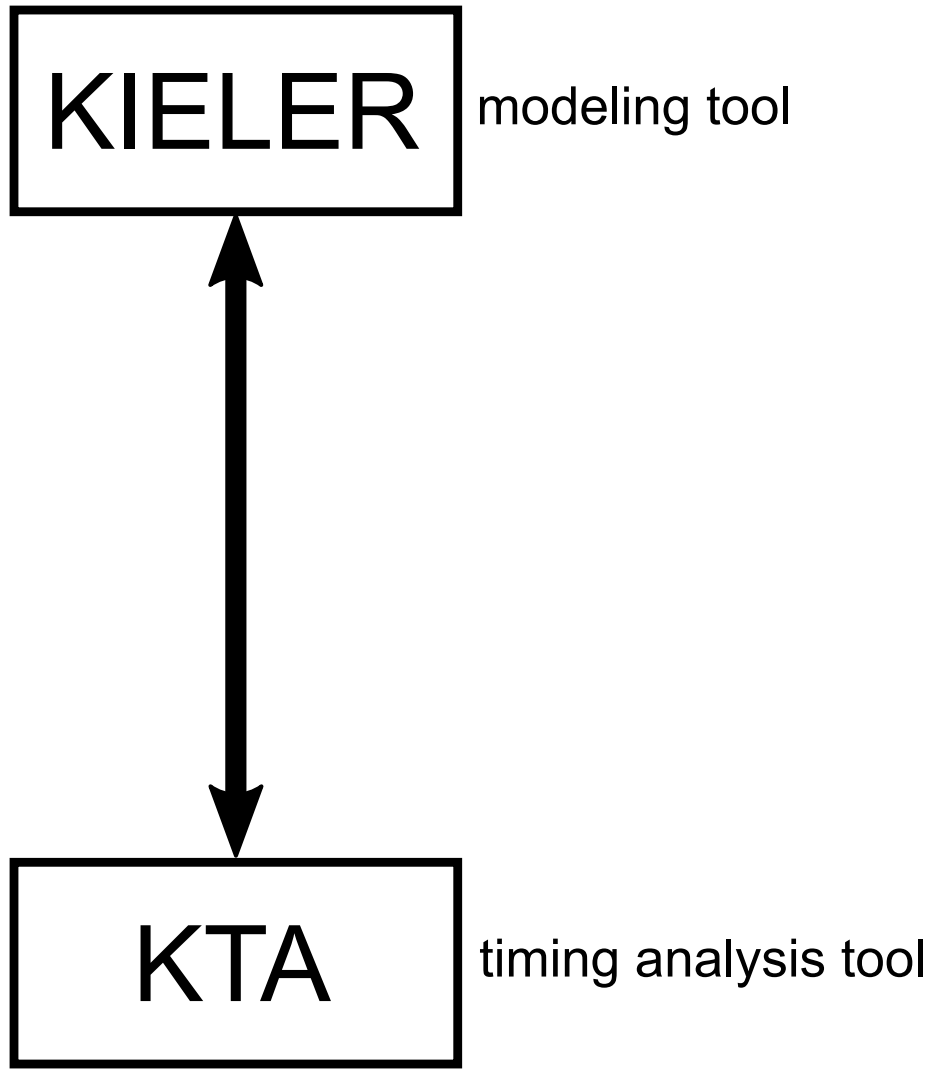

Robot
485
input bool bumper
input bool accelerator
output bool motor
[-] Main
28 / 485
Emergency
[-] HandleEmergency
10 / 10
StartE
/ errorLog(); motor = false
DoneE
1: bumper
Init
2:
Done
Normal
[-] HandleMotor
447 / 447
1: accelerator / motor = true
Start
2: / writeLog(); motor = false
Ready
/ getImage()
Stop

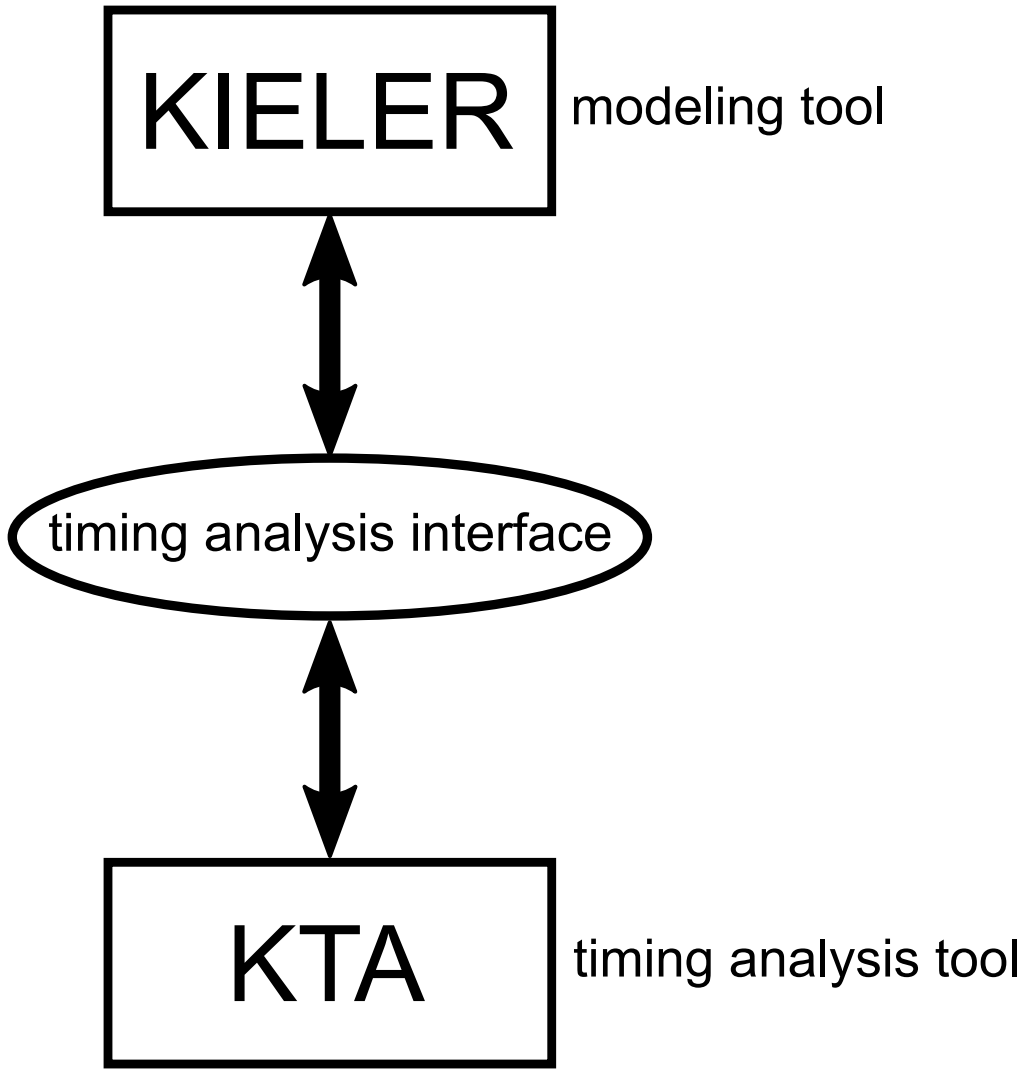
"...hotspots accounted for only 1.2% of the lines of code but contributed 29% of the overall execution time"

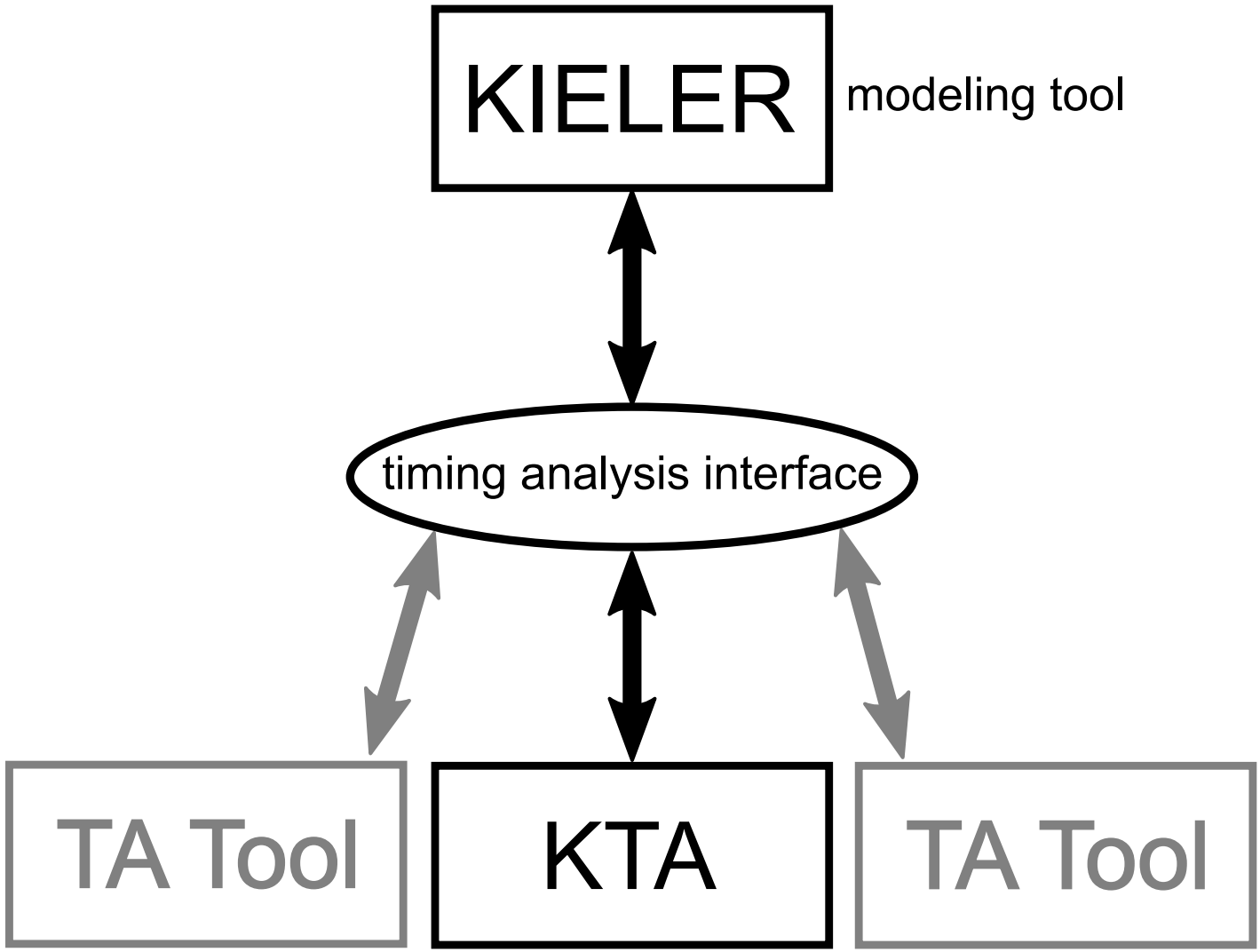
G.Bernat et al., Identifying opportunities for worst-case exectuion time reduction in an avionics system, Ada User Journal, 2007

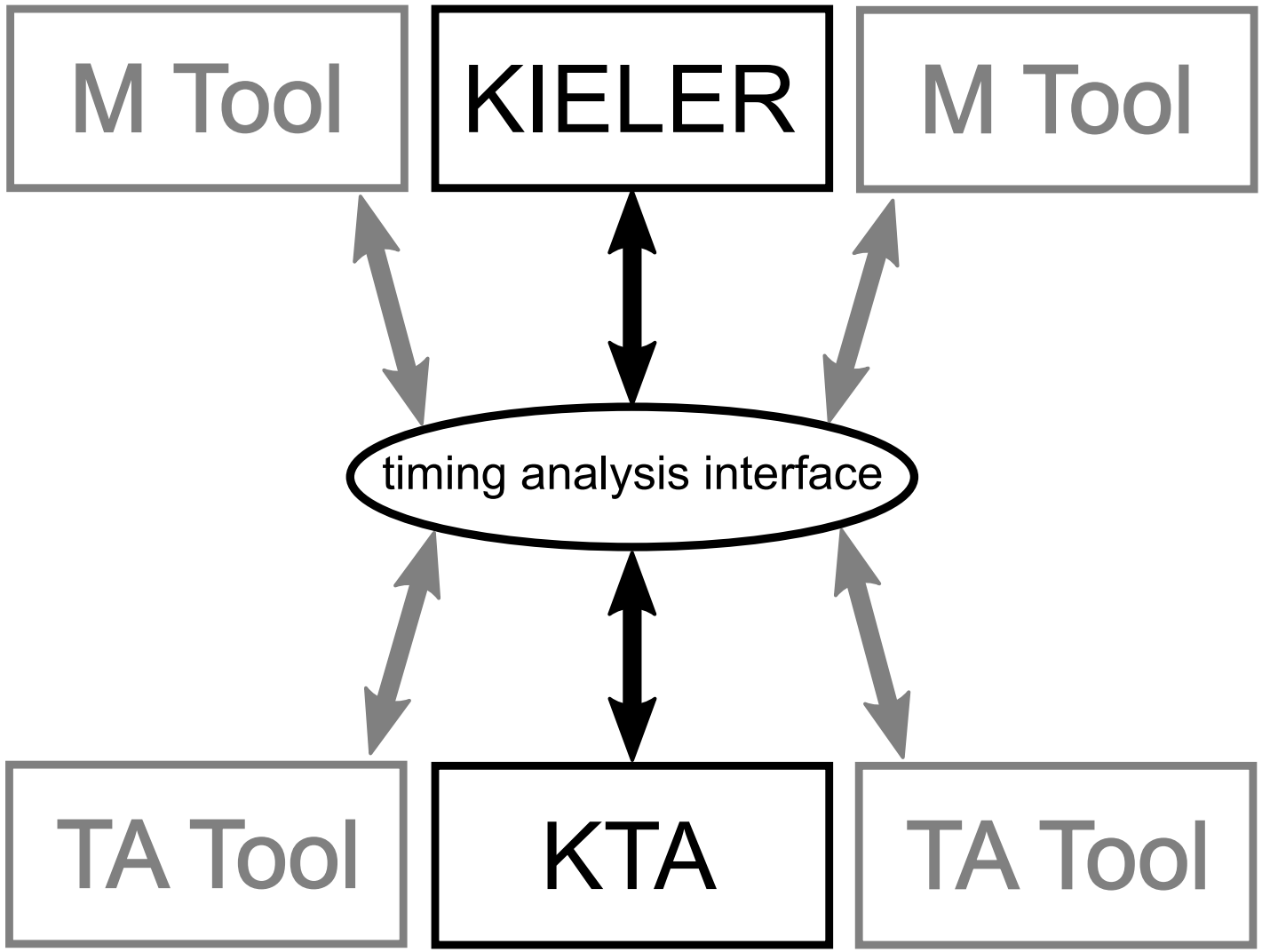

Robot
485
input bool bumper
input bool accelerator
output bool motor
[-] Main
28 / 485
Emergency
[-] HandleEmergency
10 / 10
StartE
/ errorLog(); motor = false
DoneE
1: bumper
Init
2:
Done
Normal
[-] HandleMotor
447 / 447
1: accelerator / motor = true
Start
2: / writeLog(); motor = false
Ready
/ getImage()
Stop

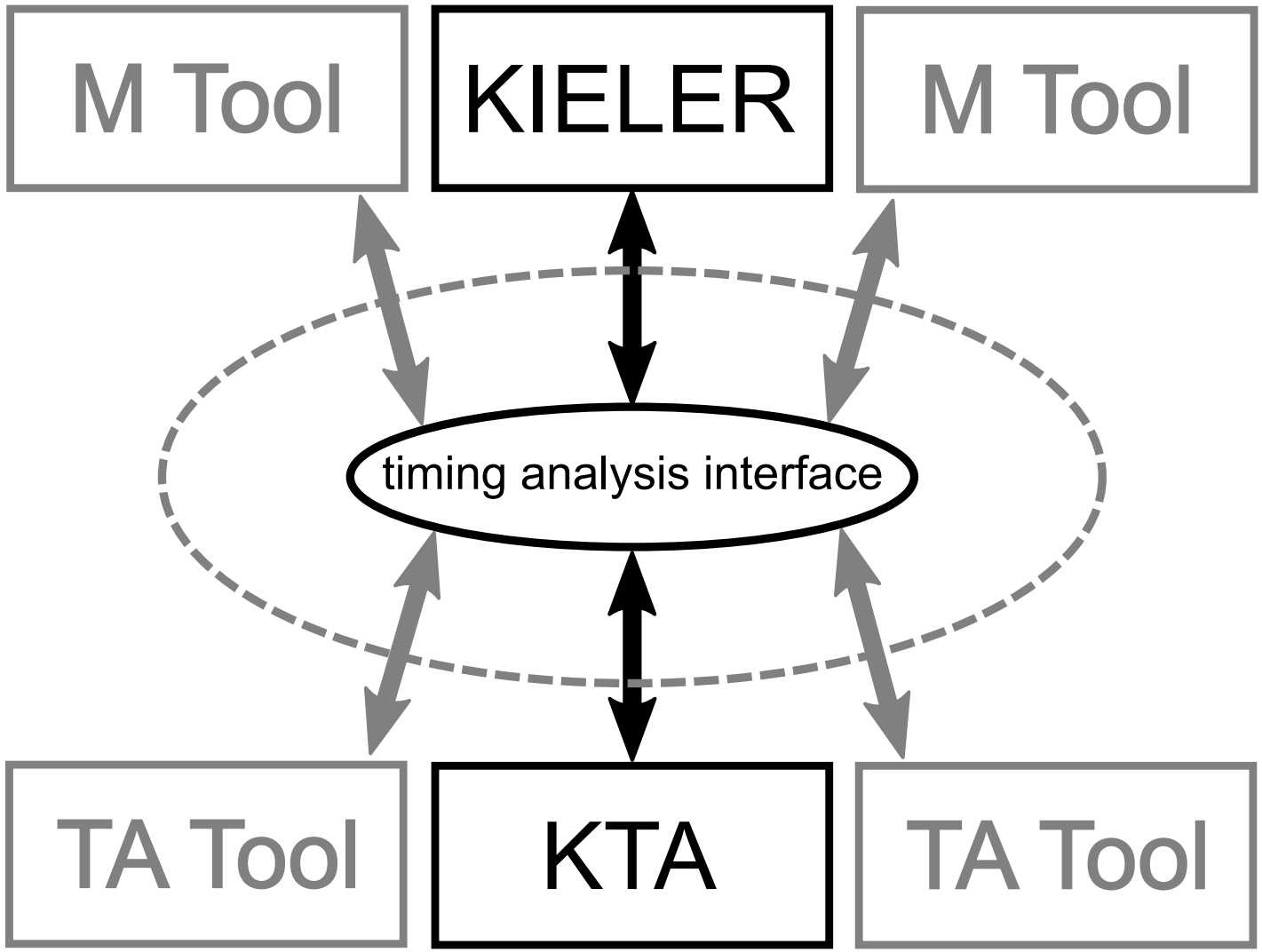

Robot
282
input bool bumper
input bool accelerator
output bool motor
output bool redlight
[-] Main
28 / 282
Emergency
[-] HandleEmergency
10 / 10
/ errorLog(); motor = false
StartE
DoneE
1: bumper
Init
2:
Done
Normal
[-] HandleMotor
244 / 244
1: accelerator / getImage(); motor = true
Start
2: / writeLog(); motor = false
Stop

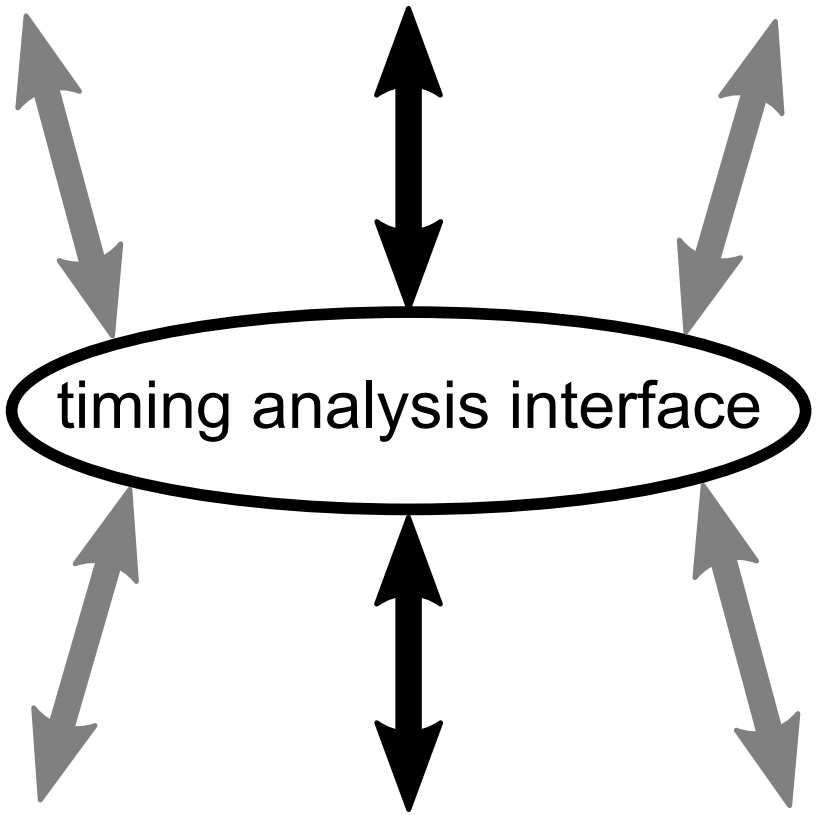

KIELER
modeling tool
KTA
timing analysis tool


KIELER
modeling tool
timing analysis interface
KTA
timing analysis tool

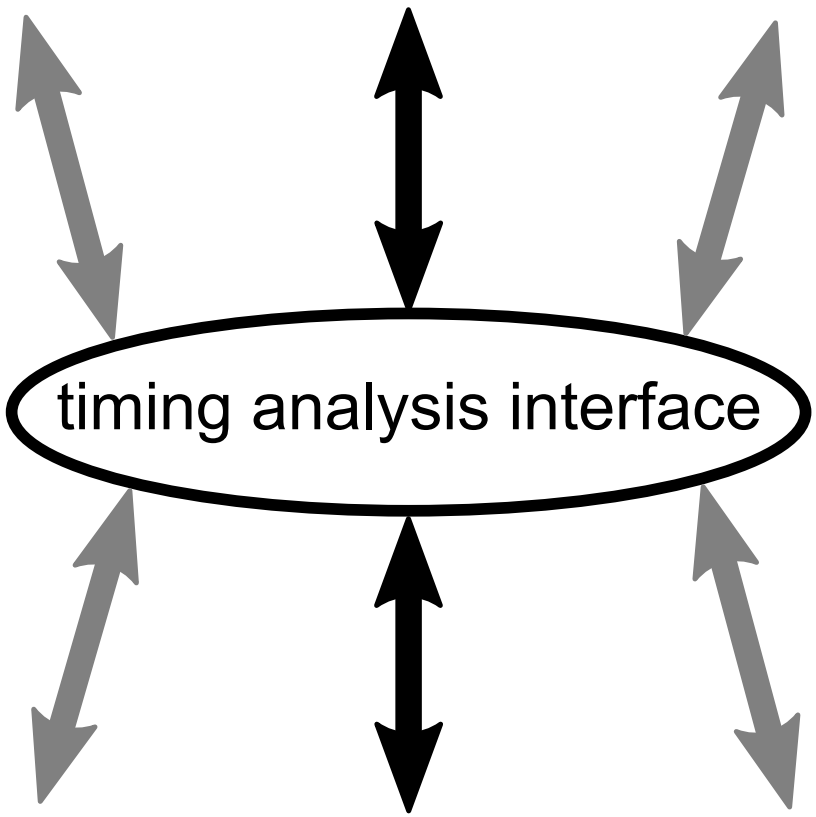

KIELER
modeling tool
timing analysis interface
TA Tool
KTA
TA Tool


M Tool
KIELER
M Tool
timing analysis interface
TA Tool
KTA
TA Tool

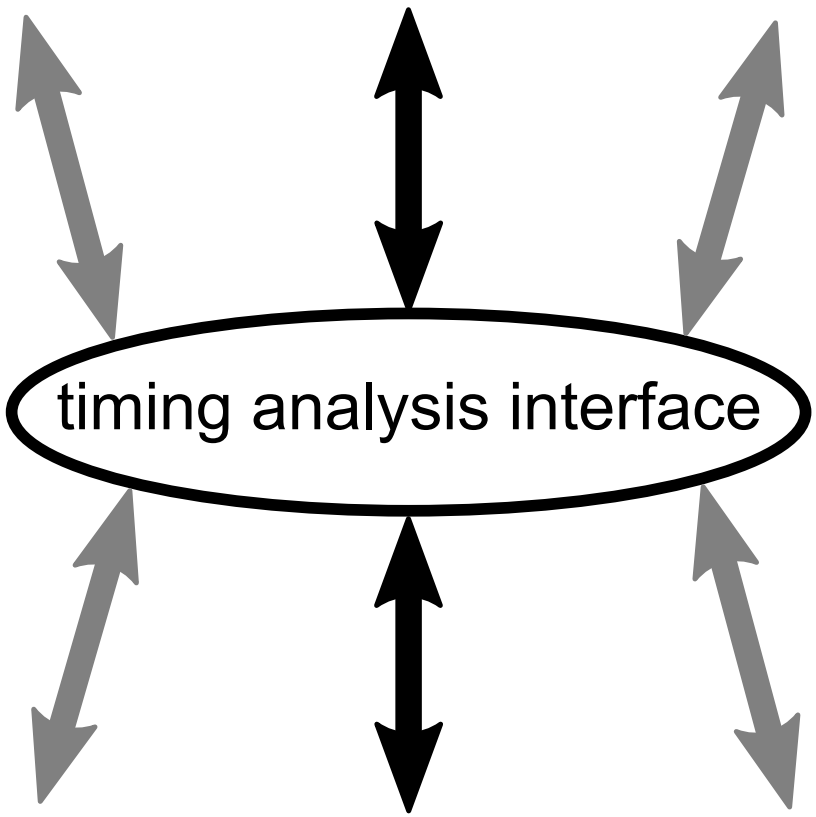

M Tool
KIELER
M Tool
timing analysis interface
TA Tool
KTA
TA Tool


timing analysis interface

$$t_{req} = ()$$

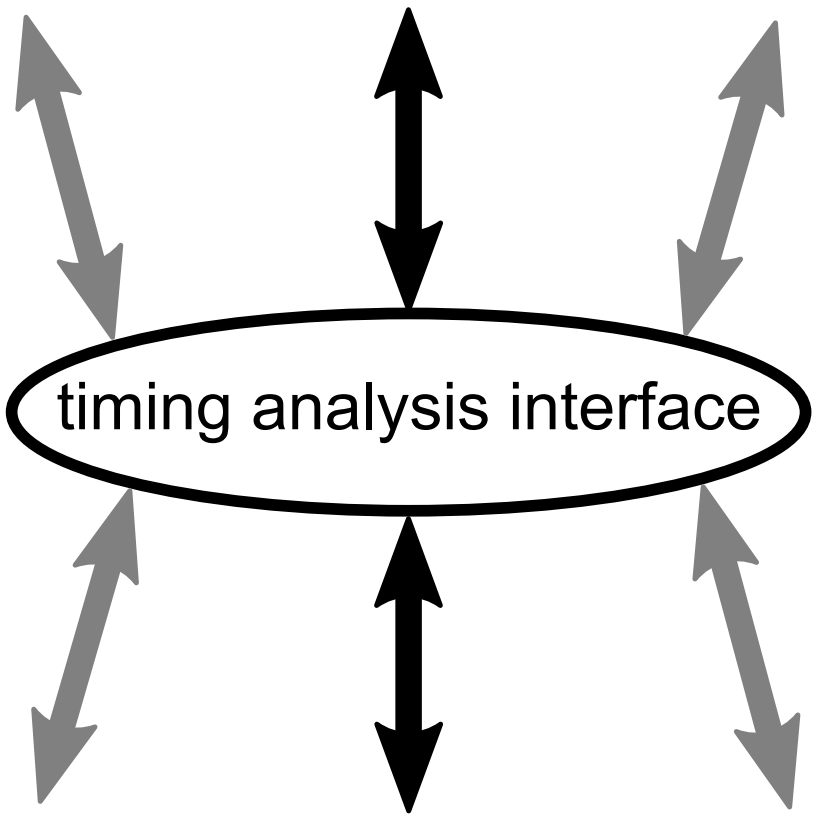


$t_{req} = (f)$



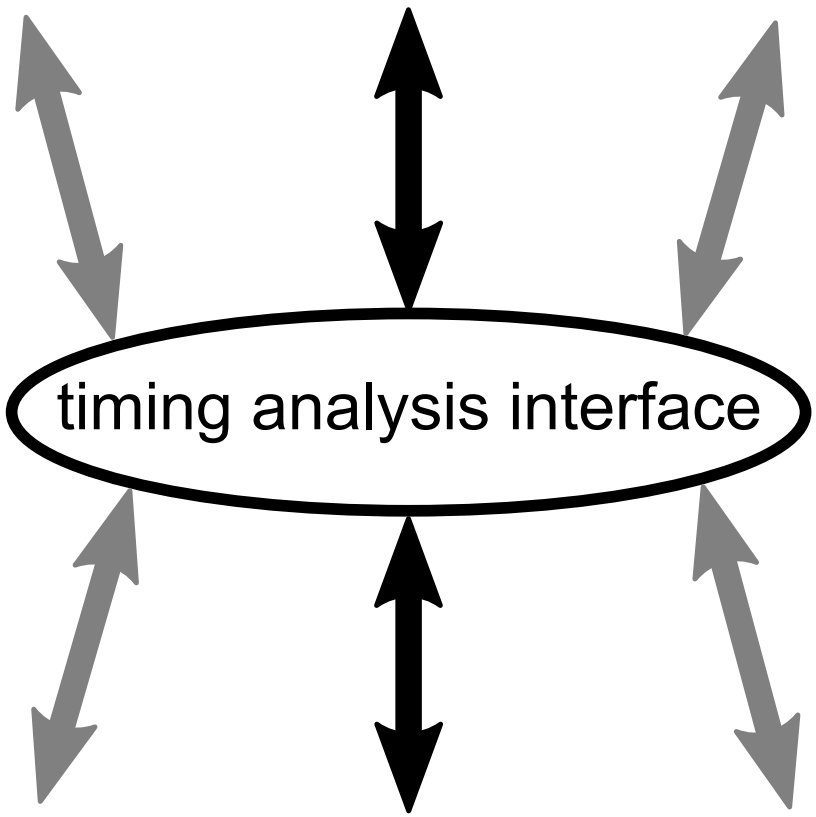
- Function to be analysed

$t_{req} = (f, a)$



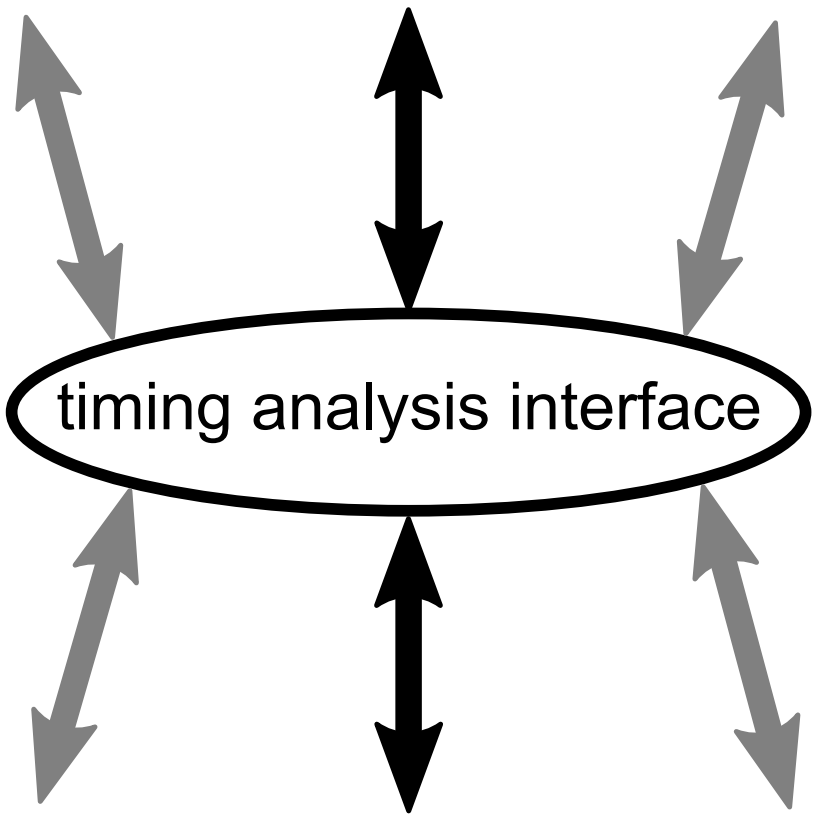
- Function to be analysed
- Assumptions on arguments

$t_{req} = (f, a, g)$



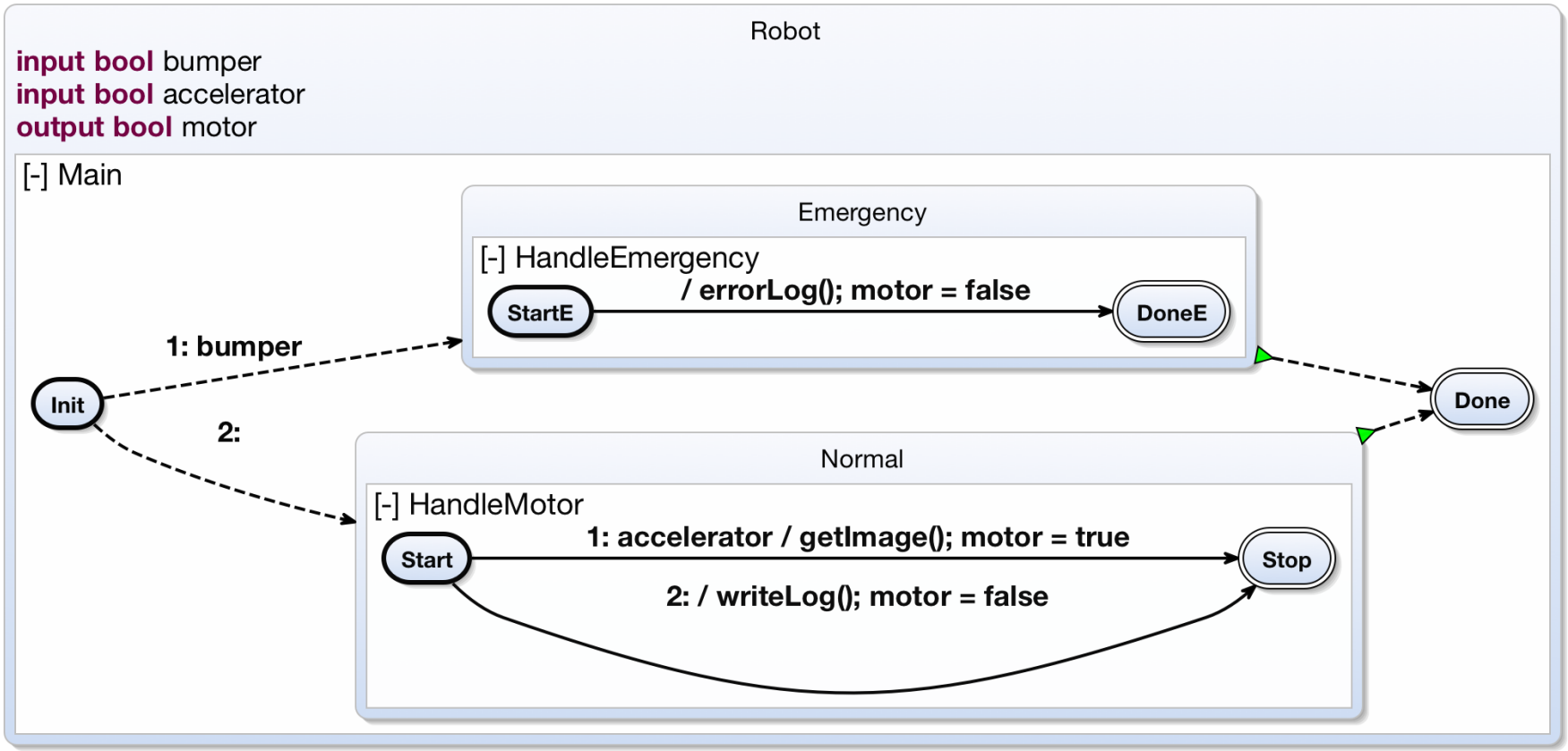
- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables

$t_{req} = (f, a, g, S)$



- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables

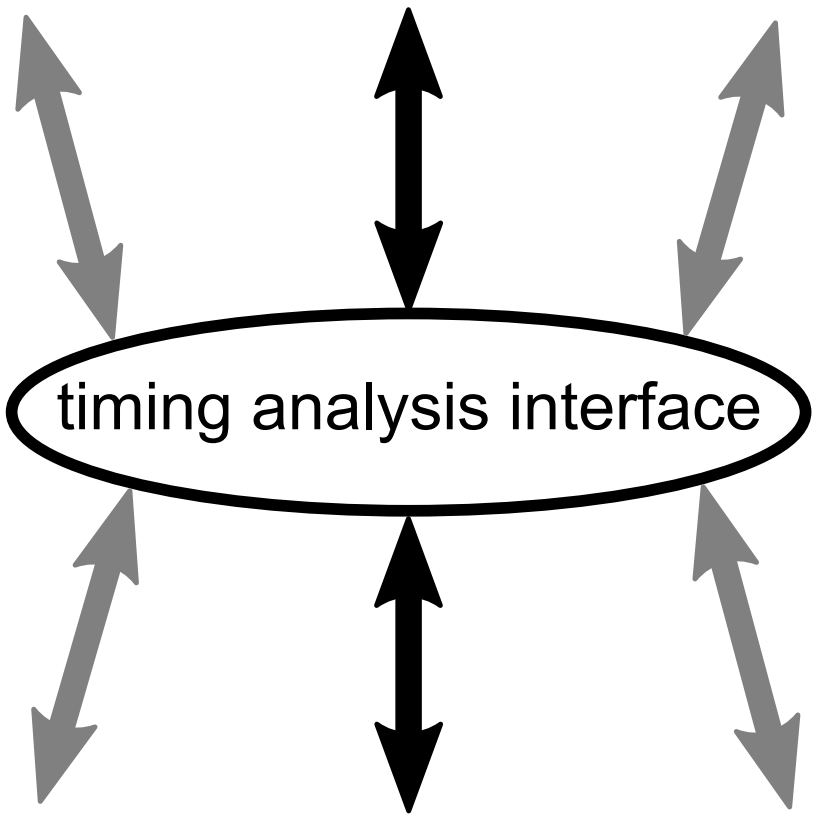
State variables - reactive infeasible paths



State variables - reactive infeasible paths II

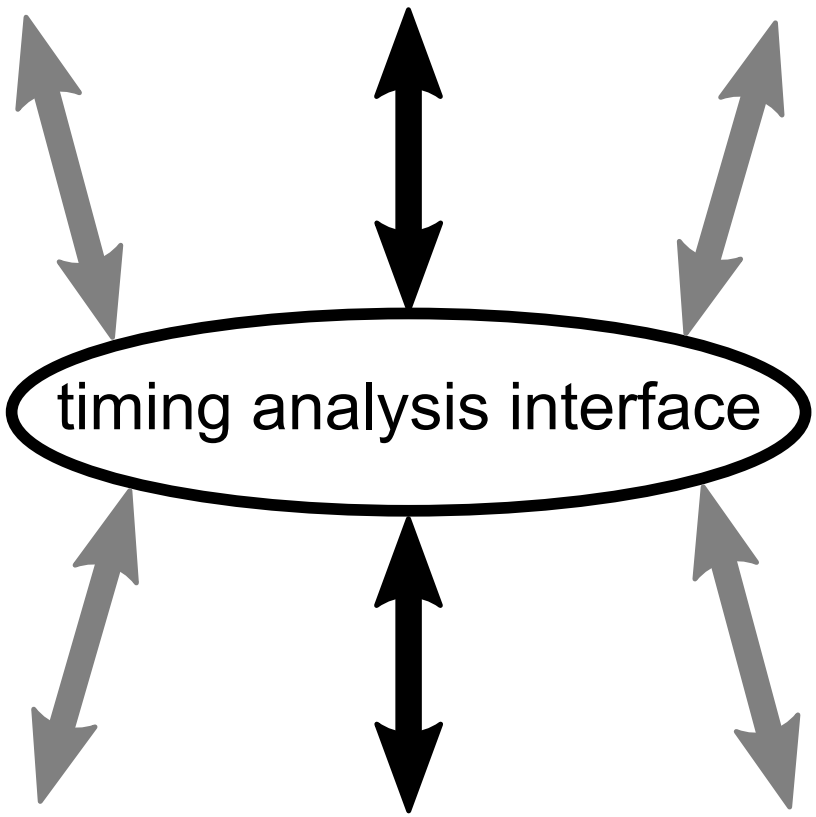
```
1   void tick(){
2     g0 = _GO;
3     g1 =(_GO && bumper);
4     g2 =(PRE_g1);
5     if(g2){
6       errorLog();
7       motor = 0;}
8     g5 =(PRE_g4);
9     g6 =(g5 && accelerator);
10    if(g6){
11      getImage();
12      motor = 1;}
19    g7 =(g5 && (!(accelerator)));
20    if(g7){
21      writeLog();
22      motor = 0;}
23    g3 =(g2 || (g6 || g7));
24    g4 =(_GO && (!(bumper)));
25    PRE_g1 = g1;
26    PRE_g4 = g4;
27    _GO = 0;
28  return;}
```

$$t_{req} = (f, a, g, S)$$



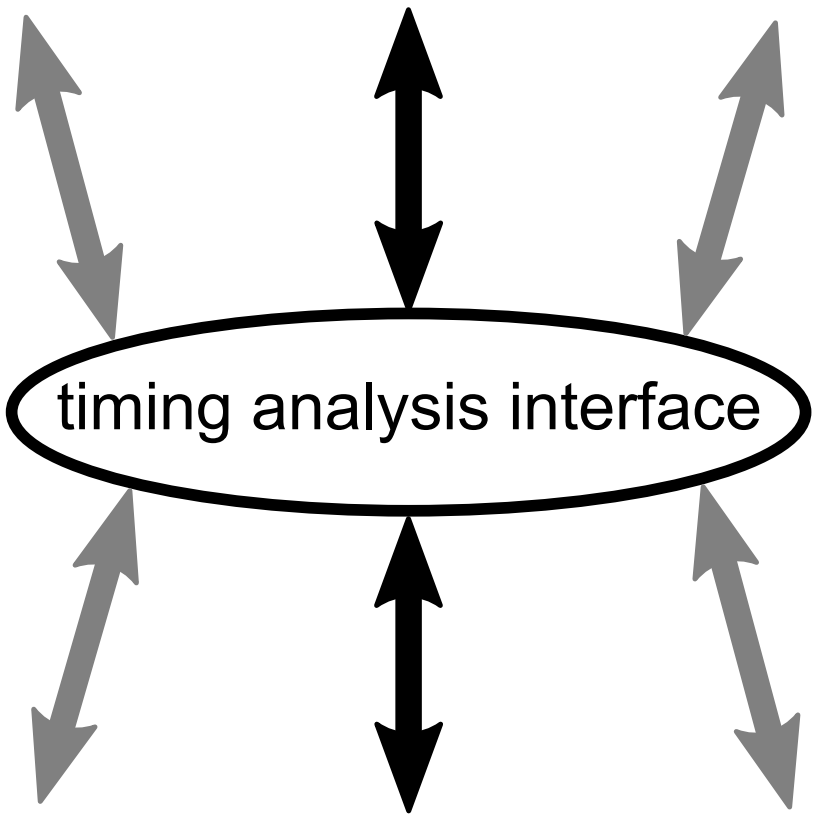
- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables

$$t_{req} = (f, a, g, S, e)$$



- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables
- Assumptions on called functions

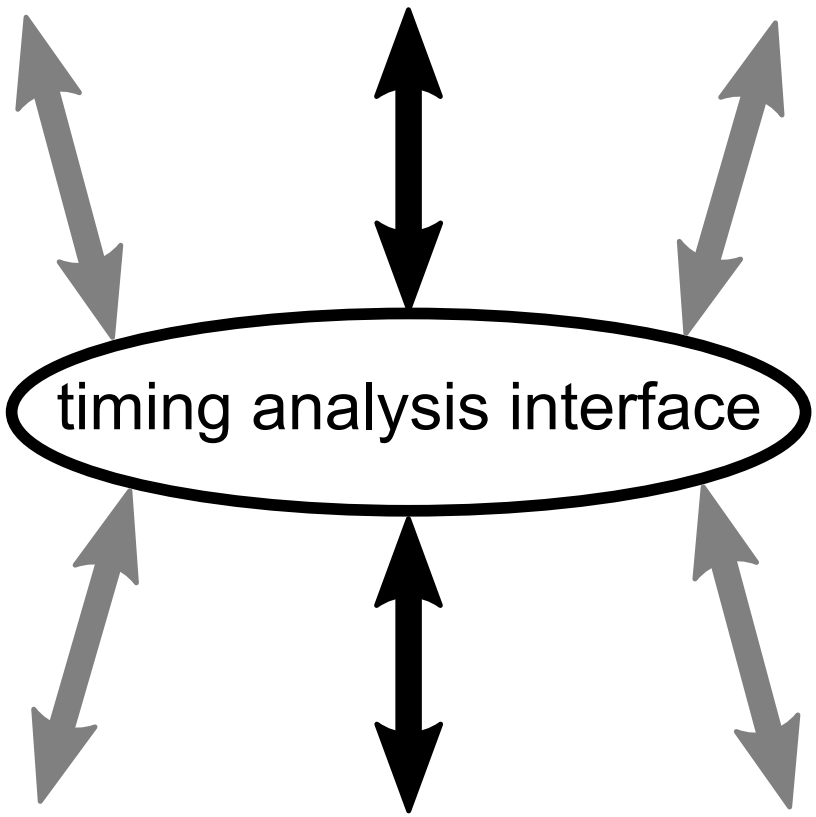
$$t_{req} = (f, a, g, S, e, P)$$



- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables
- Assumptions on called functions
- Timing Program Points

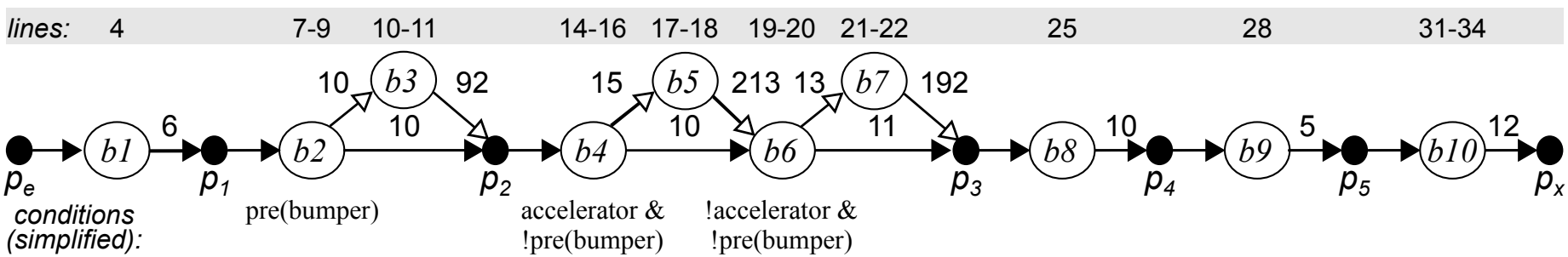
```
1  void tick(){
2  //Main
3  // implicit TPP
4      g0 = _GO;
5  //HandleEmergency
6      TPP(1);
7      g1 =(_GO && bumper);
8      g2 =(PRE_g1);
9      if(g2){
10       errorLog();
11       motor = 0;}
12 //HandleMotor
13     TPP(2);
14     g5 =(PRE_g4);
15     g6 =(g5 && accelerator);
16     if(g6){
17       getImage();
18       motor = 1;}
19     g7 =(g5 && (!(accelerator)));
20     if(g7){
21       writeLog();
22       motor = 0;}
23 //Main
24     TPP(3);
25     g3 =(g2 || (g6 || g7));
26 //HandleMotor
27     TPP(4);
28     g4 =(_GO && (!(bumper)));
29 //Main
30     TPP(5);
31     PRE_g1 = g1;
32     PRE_g4 = g4;
33     _GO = 0;
34 return;}
35 // implicit TPP}
```

$t_{req} = (f, a, g, S, e, P, R)$

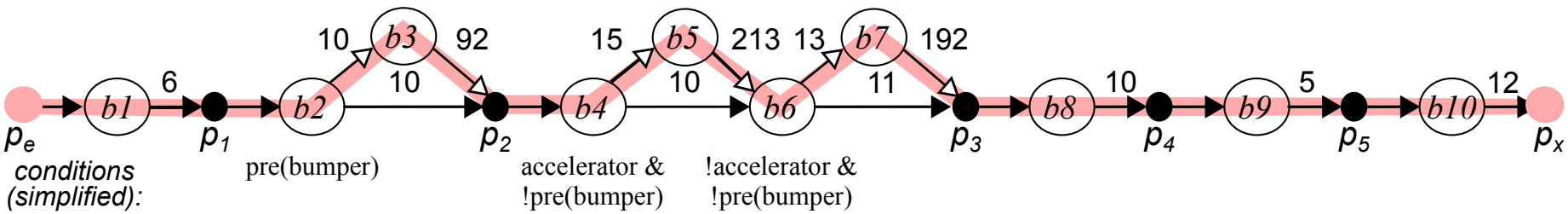


- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables
- Assumptions on called functions
- Timing Program Points
- Analysis Requests

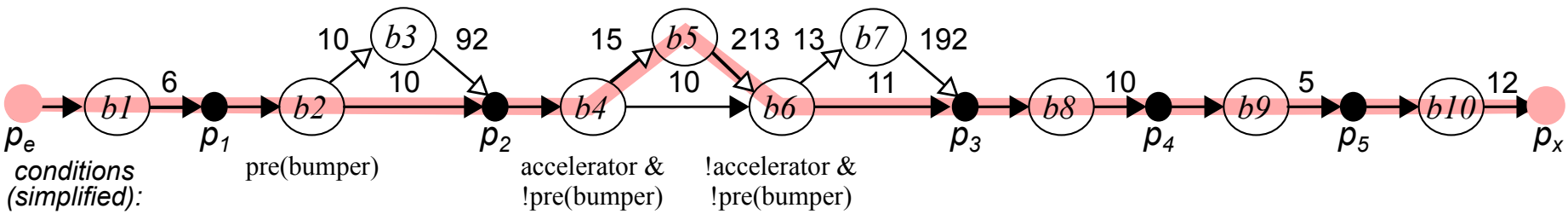
```
1   void tick(){
2   //Main
3   // implicit TPP
4     g0 = _GO;
5   //HandleEmergency
6     TPP(1);
7     g1 =(_GO && bumper);
8     g2 =(PRE_g1);
9     if(g2){
10      errorLog();
11      motor = 0;}
12  //HandleMotor
13    TPP(2);
14    g5 =(PRE_g4);
15    g6 =(g5 && accelerator);
16    if(g6){
17      getImage();
18      motor = 1;}
19    g7 =(g5 && (!(accelerator)));
20    if(g7){
21      writeLog();
22      motor = 0;}
23  //Main
24    TPP(3);
25    g3 =(g2 || (g6 || g7));
26  //HandleMotor
27    TPP(4);
28    g4 =(_GO && (!(bumper)));
29  //Main
30    TPP(5);
31    PRE_g1 = g1;
32    PRE_g4 = g4;
33    _GO = 0;
34  return;}
35  // implicit TPP}
```



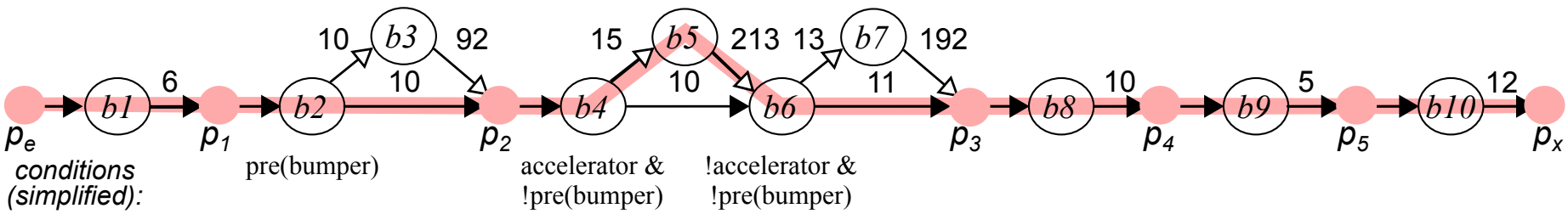
Worst Case Path



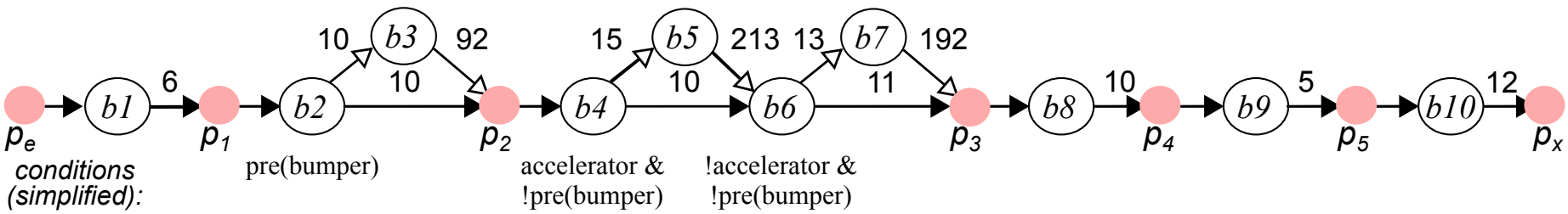
Worst Case Path



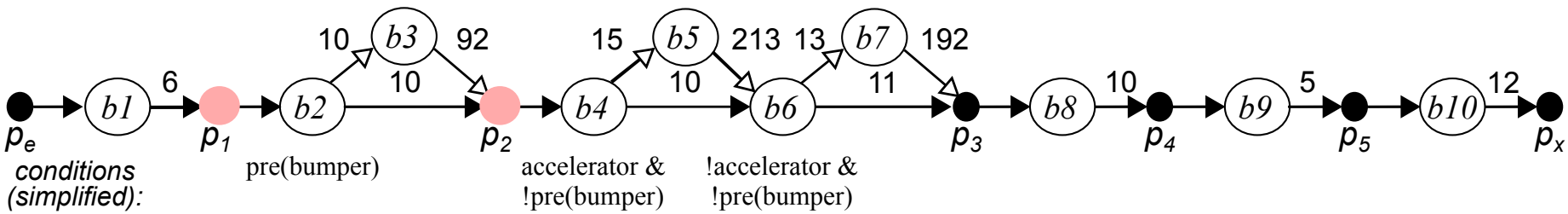
Worst Case Path



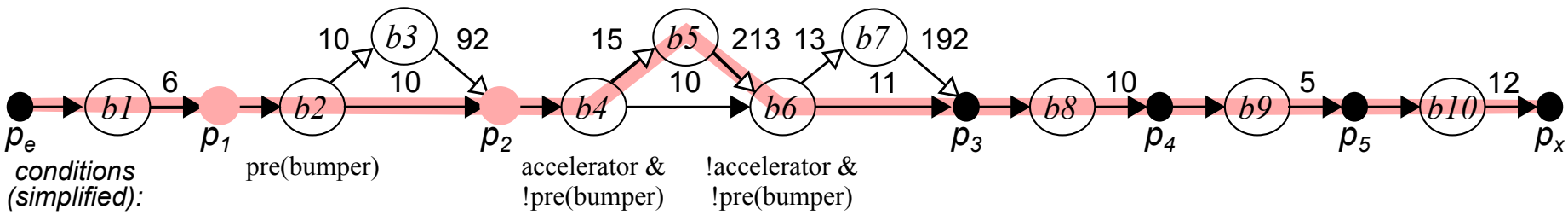
Worst Case Path



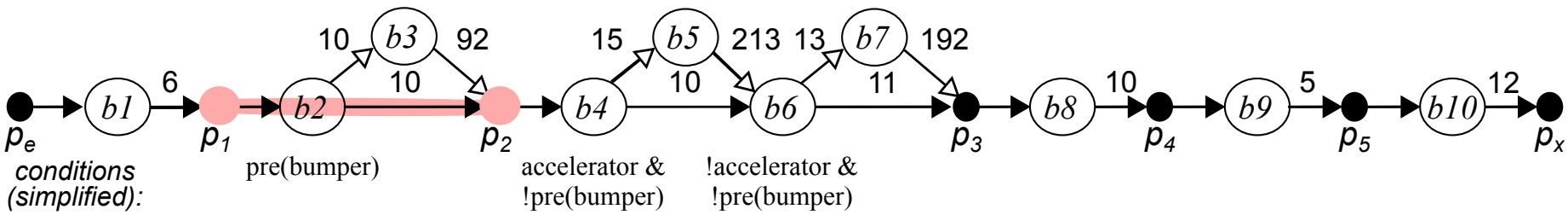
Fractional WCET



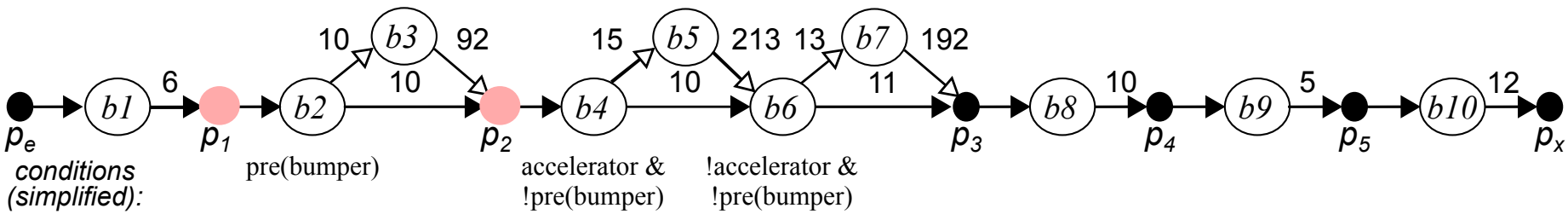
Fractional WCET



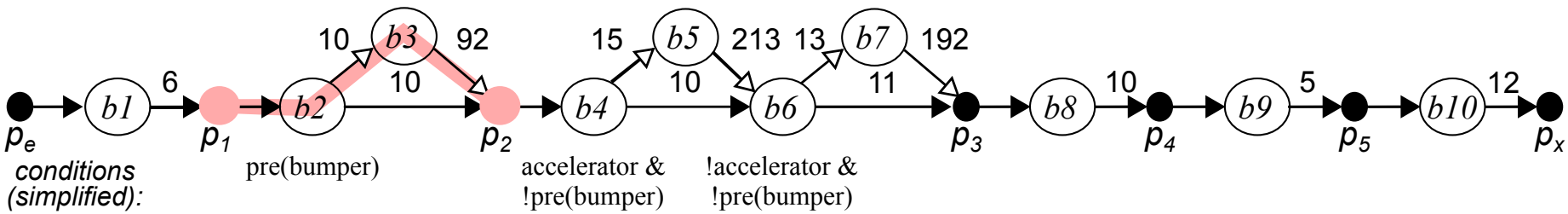
Fractional WCET



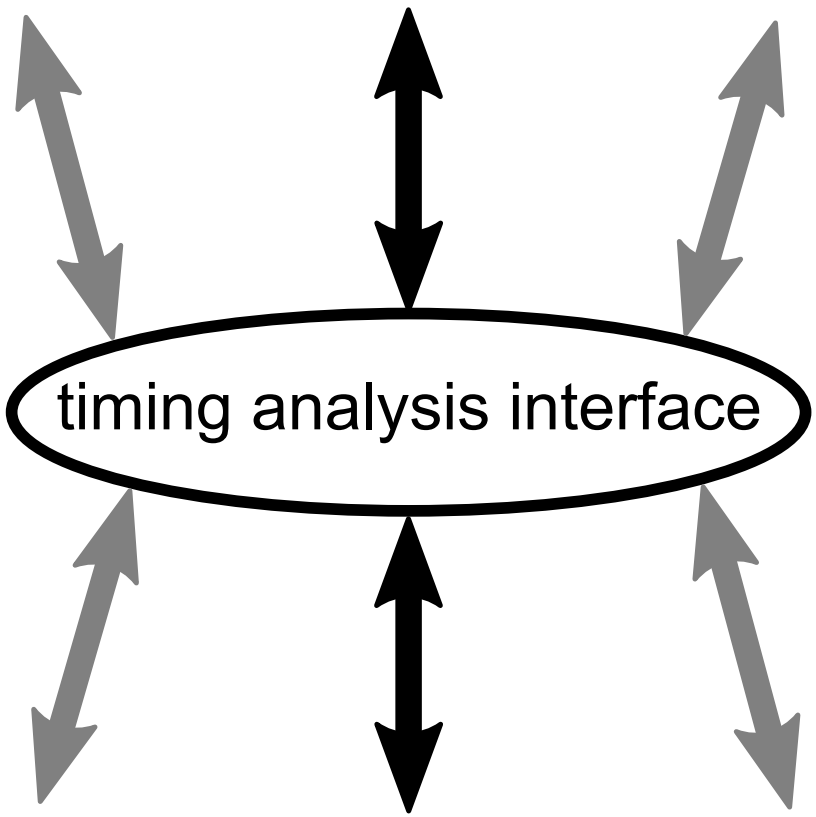
Local WCET


b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
6
10
92
10
15
213
10
13
192
11
10
5
12
p_e
p_1
p_2
p_3
p_4
p_5
p_x
conditions
(simplified):
pre(bumper)
accelerator &
!pre(bumper)
!accelerator &
!pre(bumper)

Local WCET

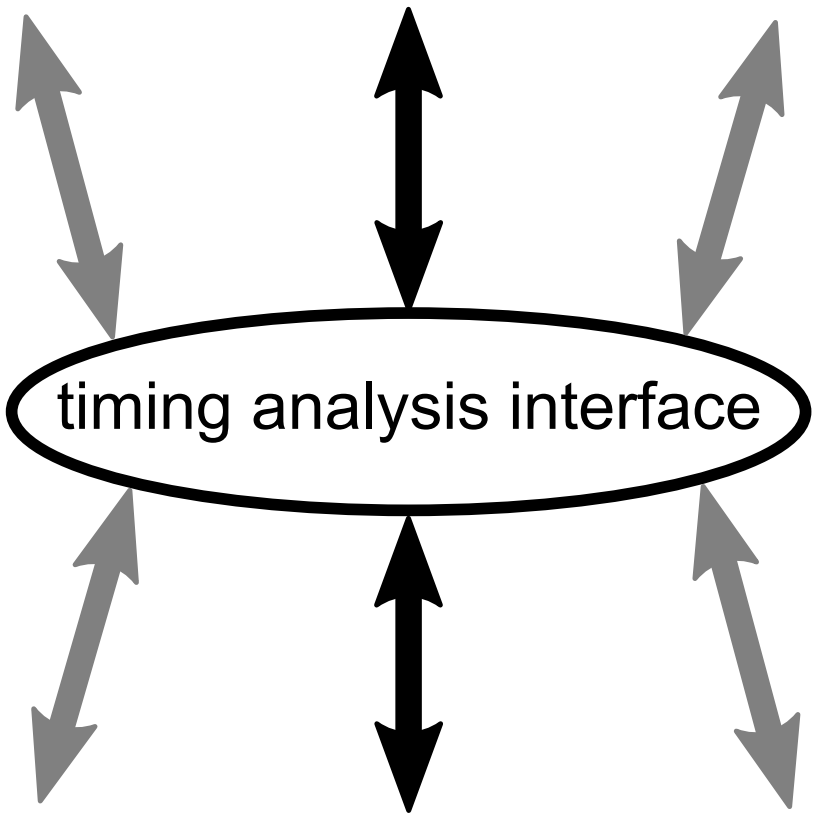

b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
6
10
92
10
15
213
10
13
192
11
10
5
12
p_e
p_1
p_2
p_3
p_4
p_5
p_x
conditions
(simplified):
pre(bumper)
accelerator &
!pre(bumper)
!accelerator &
!pre(bumper)

$t_{req} = (f, a, g, S, e, P, R)$



- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables
- Assumptions on called functions
- Timing Program Points
- Analysis Requests

$t_{req} = (f, a, g, S, e, P, R)$



- Function to be analysed
- Assumptions on arguments
- Assumptions on global variables
- State variables
- Assumptions on called functions
- Timing Program Points
- Analysis Requests

$t_{res} : R \to \mathbb{N}_{\perp} \cup \mathcal{P}(\bar{p})$

Related Work

Fast WCET Analysis: Harmon et. al.: Fast, interactive worst-case execution time analysis with back-annotation. Industrial IEEE Transactions on Informatics 2012

Interactive C-Code analysis: Ko et. al.: Supporting the specification and analysis of timing constraints. IEEE Real-Time Technology and Applications Symposium 1996

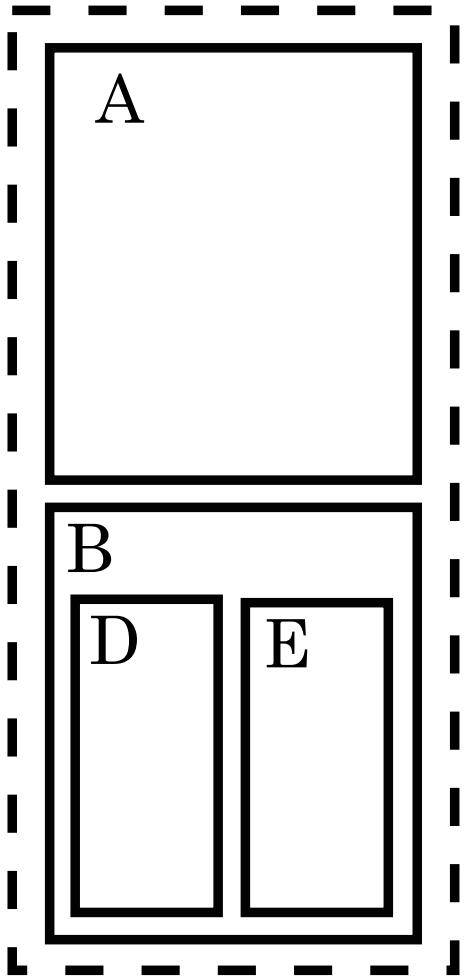
Analysis of Java Code: Persson, Hedin: Interactive execution time predictions using reference attributed grammars. WAGA99: 1999

Matlab/Simulink analysis: Kirner et. al.: Fully automatic worst-case execution time analysis for Matlab/Simulink models. In: Proceedings of the 14th Euromicro Conference on Real-Time Systems 2002

SCADE, aiT Ferdinand et.al.: Combining a high-level design tool for safety-critical systems with a tool for wcet analysis on executables. In: Embedded Real Time Software (ERTS), 2008.

Contributions

- Design flow for interactive timing analysis with *hotspot highlighting*, and the separation of *deep*, *flat*, *local* and *fractional* timing values
- *Timing analysis interface* with seperation of concerns of timing analysis for external function calls and for the tick function
- Concept of Timing Program Points
- Concept of *reactive infeasible paths*



...

TPP(1)

B

TPP(2)

D

TPP(3)

A

TPP(4)

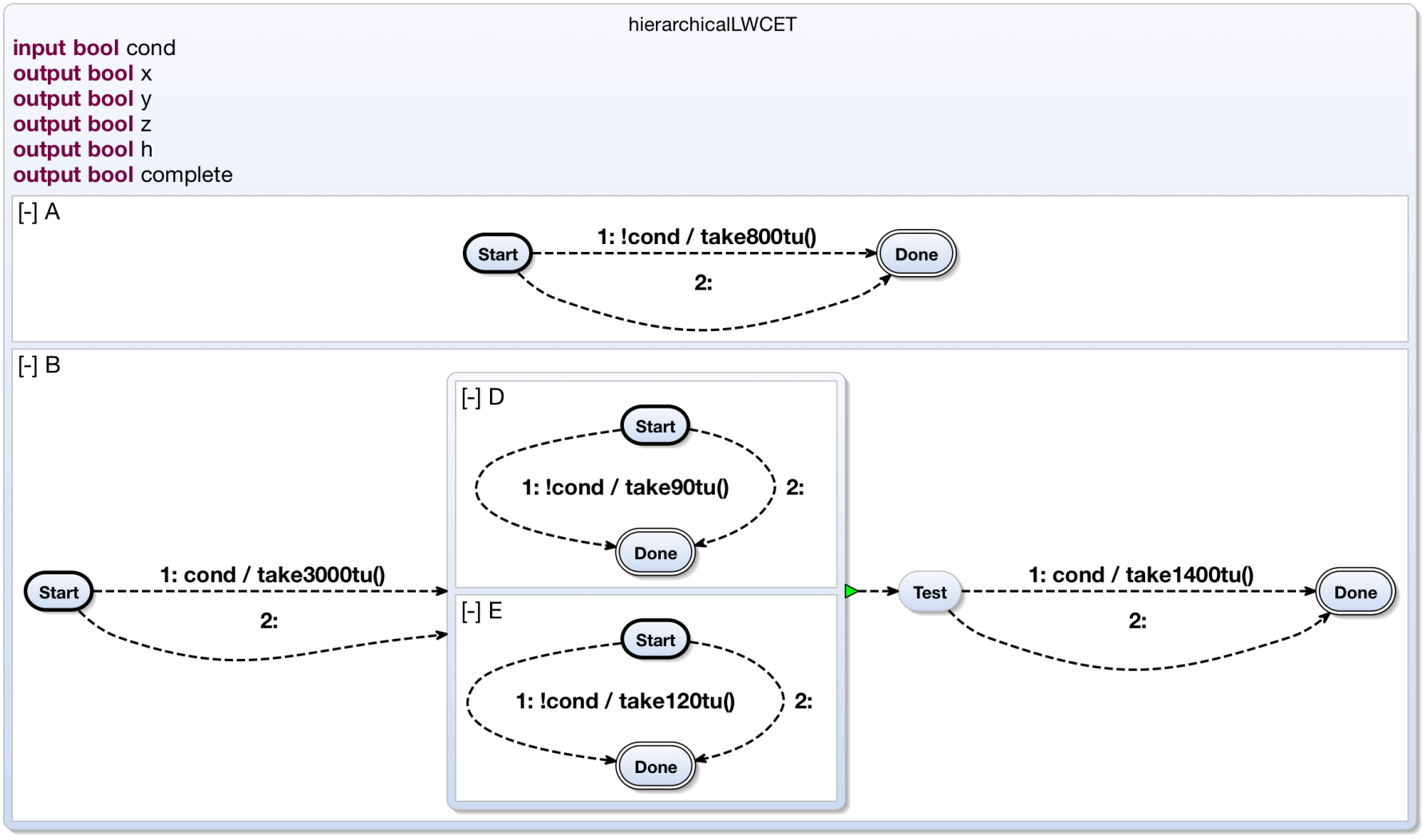
E

TPP(5)

B

TPP(6)

...


hierarchicalLWCET
input bool cond
output bool x
output bool y
output bool z
output bool h
output bool complete
[-] A
Start
1: !cond / take800tu()
2:
Done
[-] B
Start
1: cond / take3000tu()
2:
[-] D
Start
1: !cond / take90tu()
2:
Done
[-] E
Start
1: !cond / take120tu()
2:
Done
Test
1: cond / take1400tu()
2:
Done