Engineering the Hardware/Software Interface for Robotic Platforms - A Comparison of Applied Model Checking with Prolog and Alloy

Md Abdullah Al Mamun

Christian Berger

Jörgen Hansson

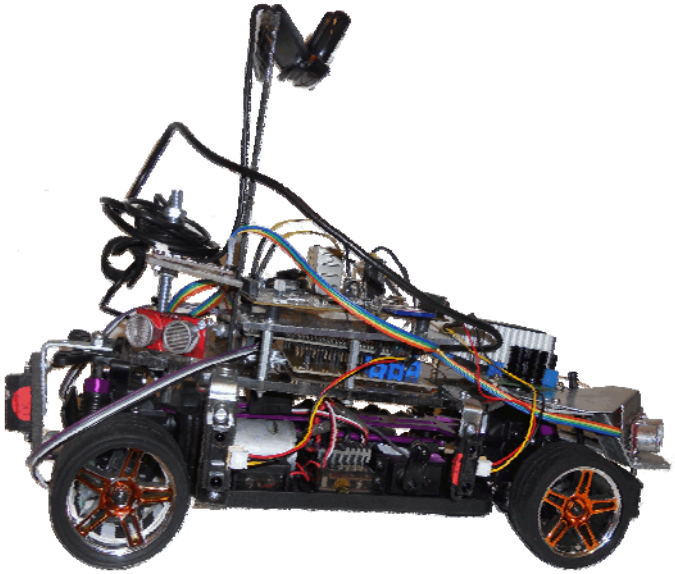
Division of Software Engineering
Department of Computer Science & Engineering

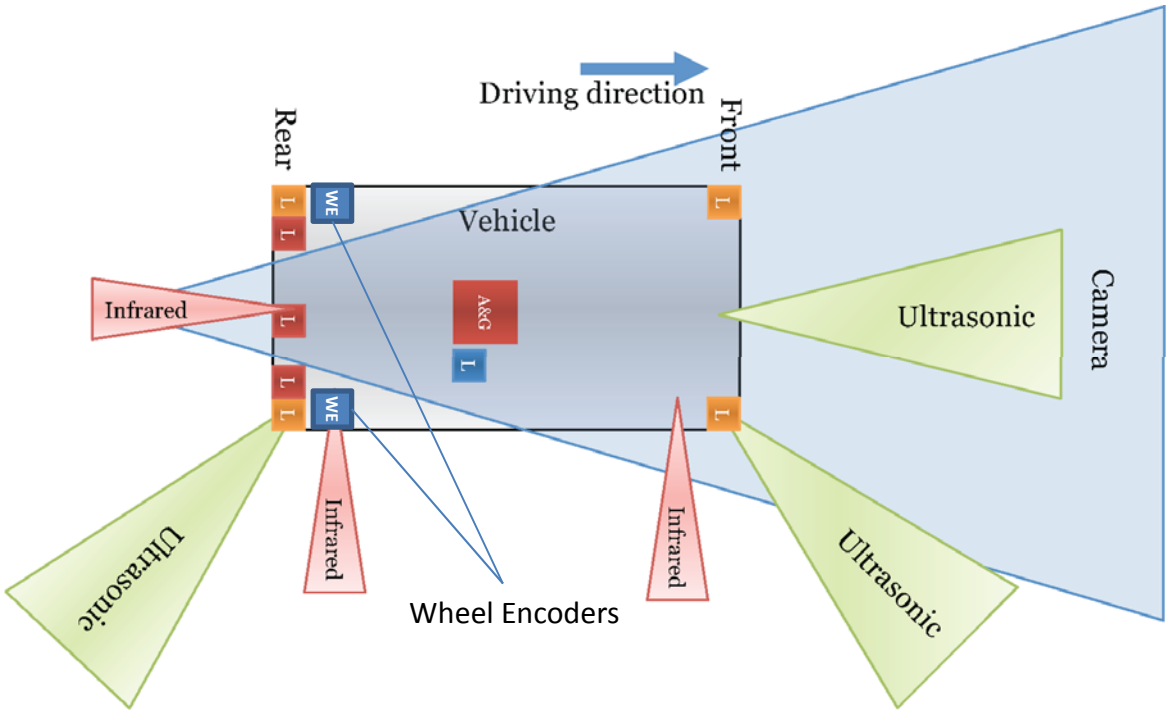
The 4th Workshop on
DSLRob 2013
Tokyo, Japan

Outline

- Problem Domain
- Research Questions
- Overall Workflow
- Prolog Approach
- Alloy Approach
- Result
- Other Observations
- Future Work
- Summary & Conclusion

Problem Domain: Sensor Layout

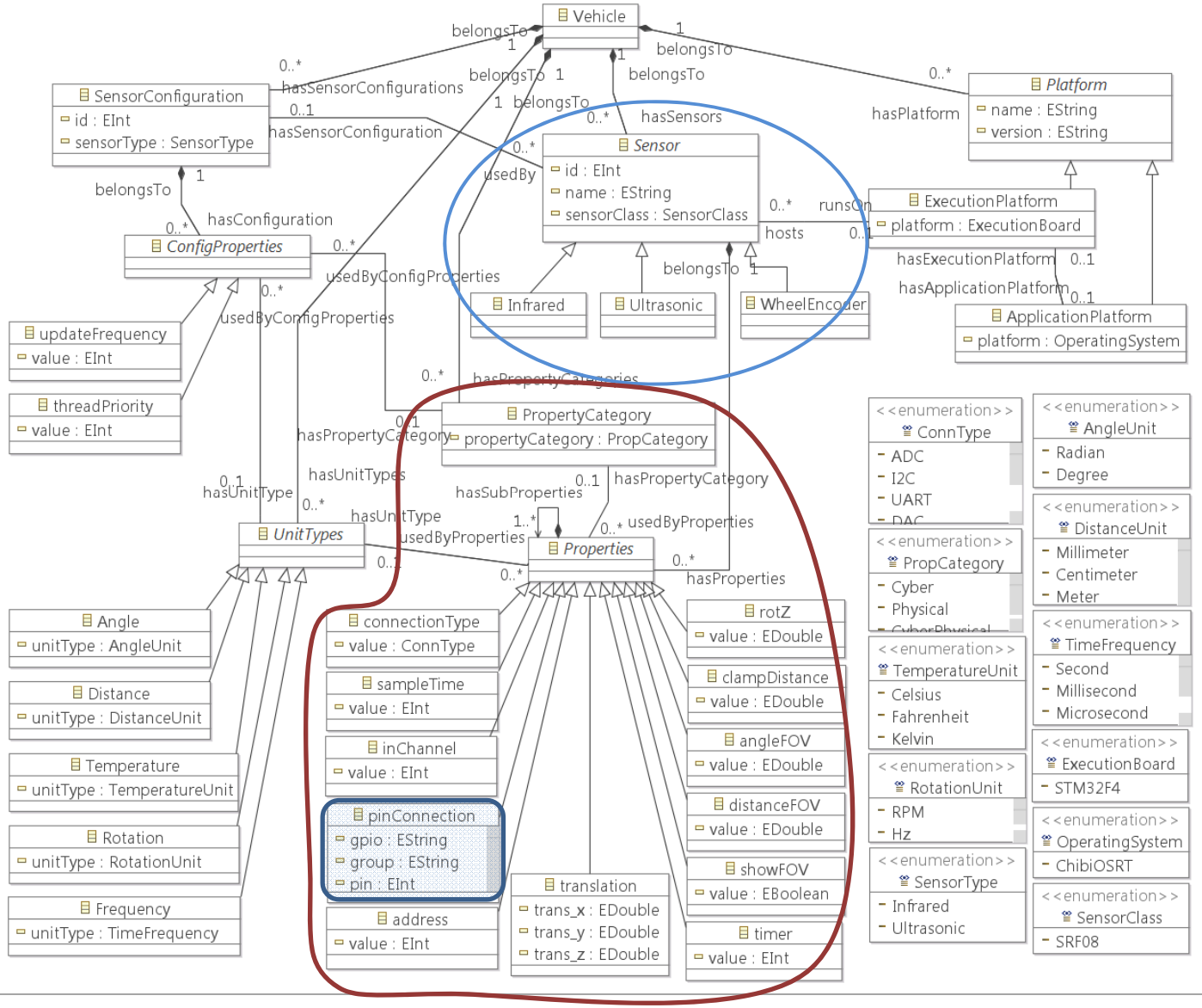




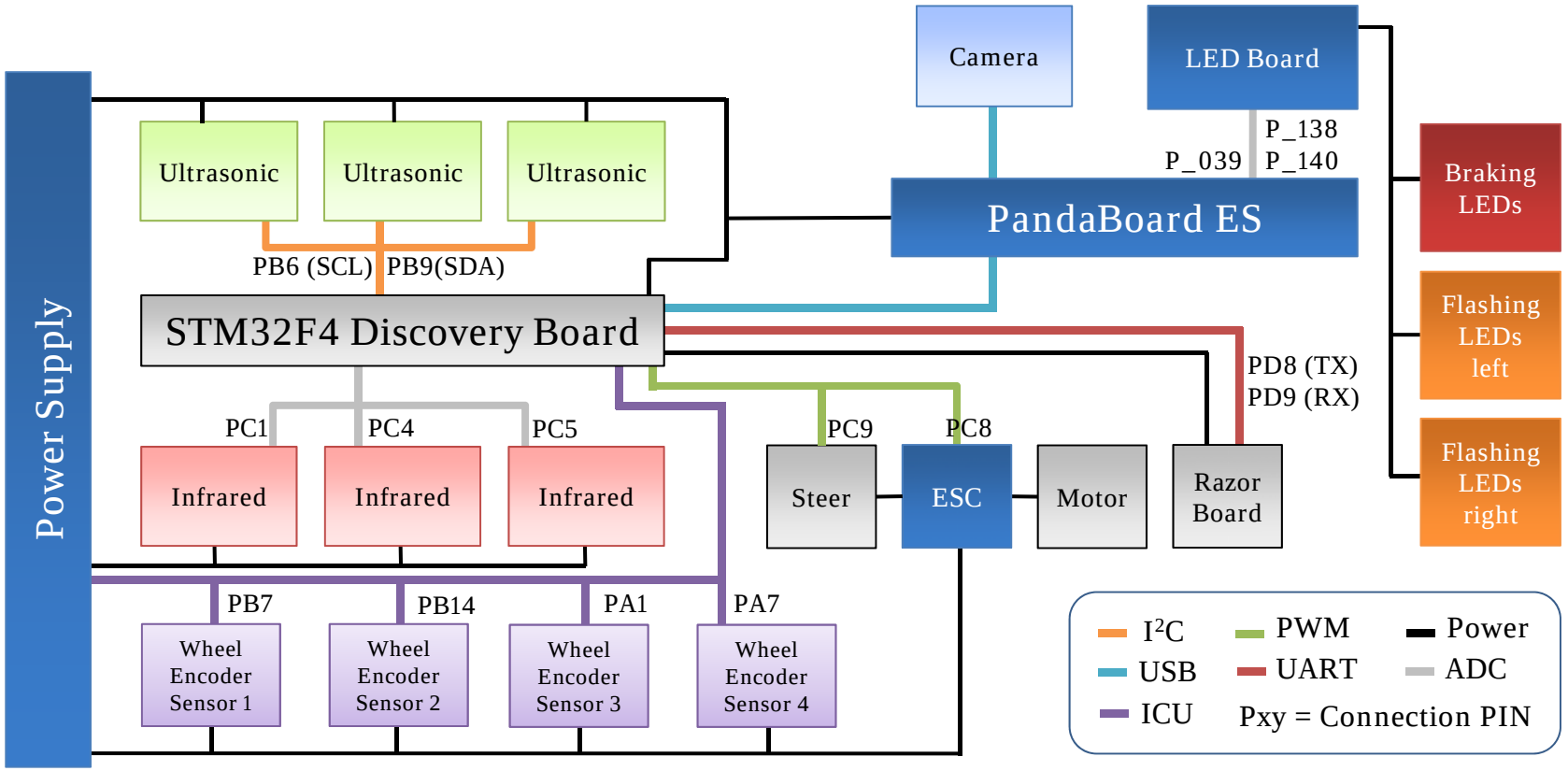
Problem Domain: DSL Meta-Model

- **EMF ECORE Model**

- Sensors
- Sensor Properties
- Configuration Properties
- Execution Platform
- Application Platform
- Unit Types
- Enumerations



Problem Domain: Solution for a Desired Configuration

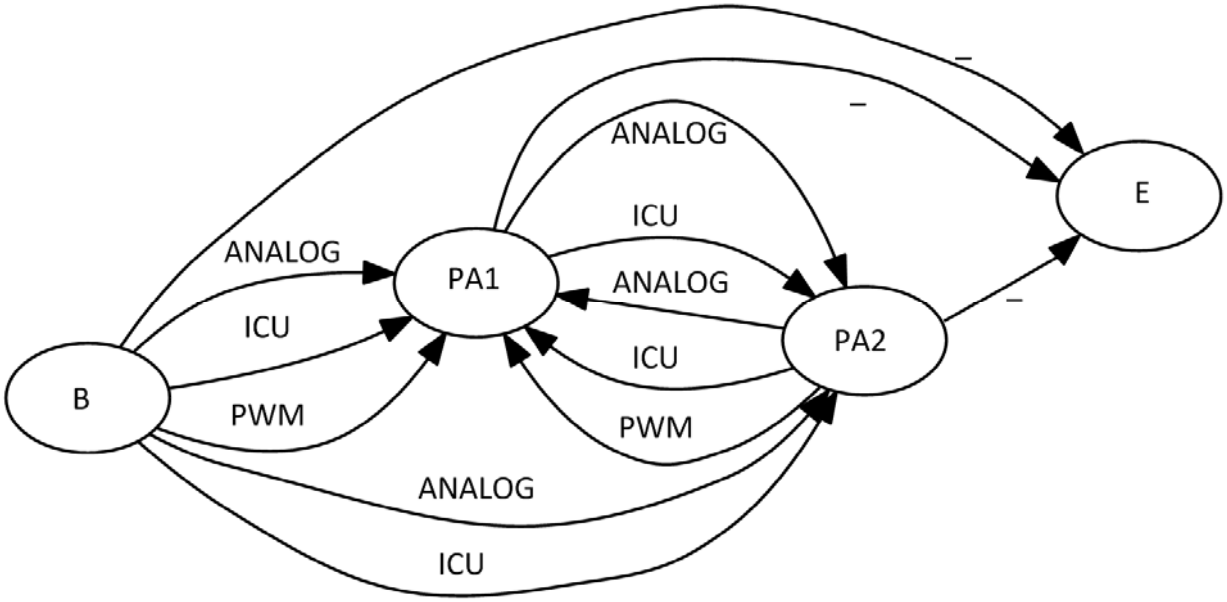


Schematic Hardware Architecture with assigned pinConnection

Problem Domain: Configuration Space

Pin	ADC1	I2C1	I2C2	I2C3	UART1	UART2	UART3	UART4	UART5	UART6	CAN	ICU1	PWM1	ICU2	PWM2	ICU3	PWM3	ICU4	PWM4	ICU5	PWM5	PWM6	ICU9	ICU10	ICU12
PA2	ADC1-IN2					UART2-TX									TIM2_CH3						TIM5_CH3		TIM9_CH1		
PA3	ADC1-IN3					UART2-RX									TIM2_CH4						TIM5_CH4		TIM9_CH2		
PA1	ADC1-IN1													TIM2_CH2						TIM5_CH2					
PB8											CAN1-RX								TIM4_CH3					TIM10_CH1	
PB11			I2C2-SDA				UART3-RX								TIM2_CH4										
PC6										UART6-TX							TIM3_CH3					TIM8_CH3			
PC9				I2C3-SDA													TIM3_CH4					TIM8_CH4			
PA8				I2C3-SCL								TIM1_CH1													
PB0	ADC1-IN8																TIM3_CH3								
PB1	ADC1-IN9																TIM3_CH4								
PB6		I2C1-SCL			UART1-TX																				
PB7					UART1-RX													TIM4_CH2							
PB9		I2C1-SDA									CAN1-TX														
PB10			I2C2-SCL				UART3-TX																		
PC10							UART3-TX	UART4-TX																	
PC11							UART3-RX	UART4-RX																	
PA4	ADC1-IN4																								
PA5	ADC1-IN5																								
PA6	ADC1-IN6																								
PA7	ADC1-IN7																								
PA9																									
PA10																									
PA15																									
PB4																									
PB5																									
PB14																									
PB15																									
PC0	ADC1-IN10																								
PC1	ADC1-IN11																								
PC2	ADC1-IN12																								
PC3	ADC1-IN13																								
PC4	ADC1-IN14																								
PC5	ADC1-IN15																								
PC7																									
PC12																									
PD2																									
PD5																									
PD6																									
PD8																									
PD9																									
PE5																									
PE6																									
PE9																									
PE11																									
PE13																									
PE14																									

Basic Representation Model



Visualization of the graph $G = \{N,E,A\}$

A concrete configuration is represented by a path P from n_B to n_E with $|P| < |N|$

Problem Domain: Complexity

$$C_{|M|=1}^{|N|} = \sum_{k=1}^{L} \binom{|N|}{k} = \sum_{k=1}^{L} \frac{|N|!}{(|N|-k)!k!}$$

$$K(n,m) = \begin{cases} 1 + \sum_{p=1}^{n} * K(p, m-1) & \text{if } m > 1, \\ 1 & \text{otherwise} \end{cases}$$

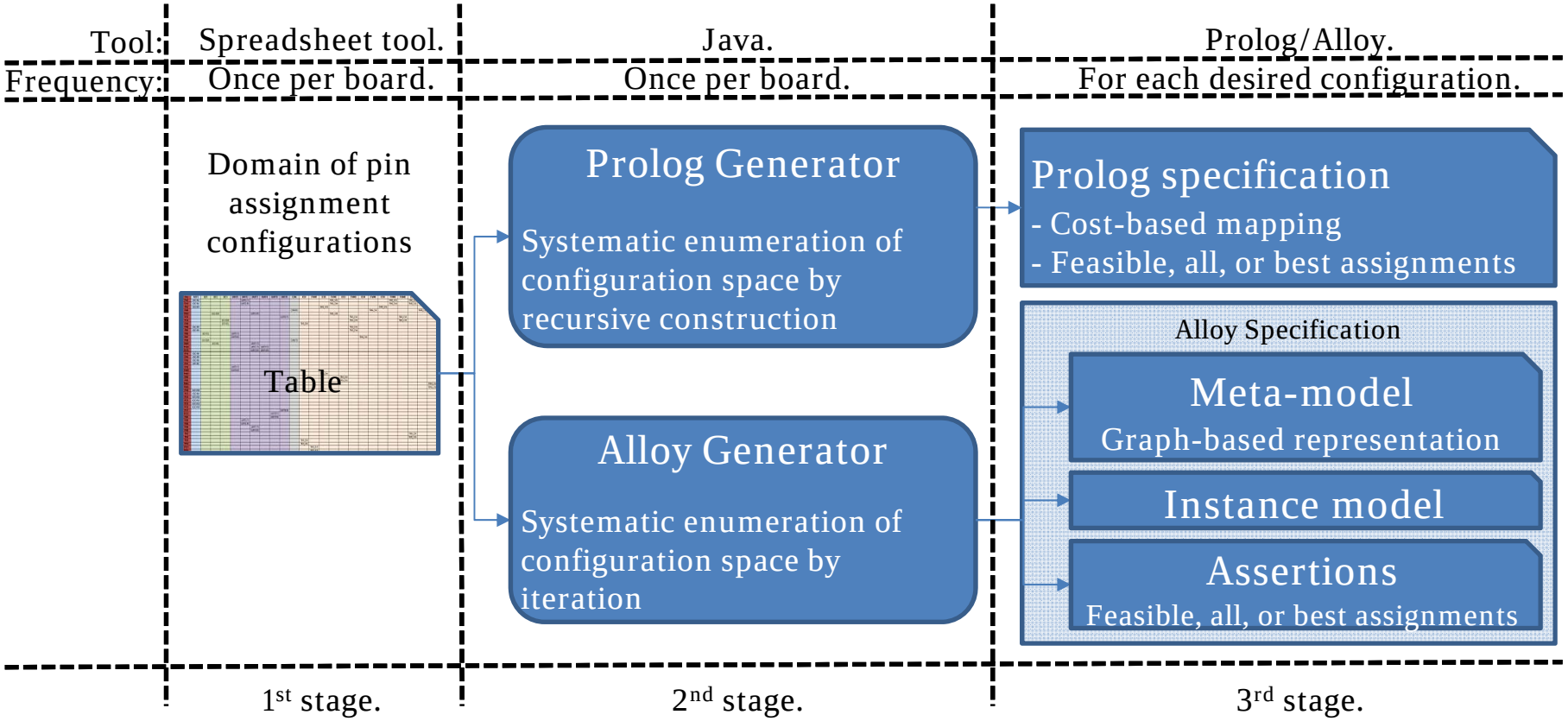
$$C_{|M|}^{|N|} = \sum_{k=1}^{L} \binom{|N|}{k} * K(k, |M|)$$

- ✓ Total 1,099,126,862,792 configuration possibilities (considering all pins)
- ✓ Total 14,689,111 configuration possibilities (considering pins with multiple usage)

Research Questions

- To determine a feasible, all possible, and the best configuration assignment:
  - How can Prolog be used to apply model checking on instances of the domain of possible pin assignment configurations?
  - How can Alloy be used to apply model checking on instances of the domain of possible pin assignment configurations?
- Which approach performs quicker compared to the other for the particular use cases?

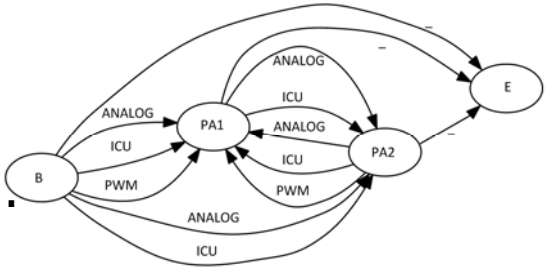
Overall Workflow



Prolog Approach

- Consists of two parts
  - Facts (generated)

```
config([analog,analog],[[pa1,pa2],7]).
```



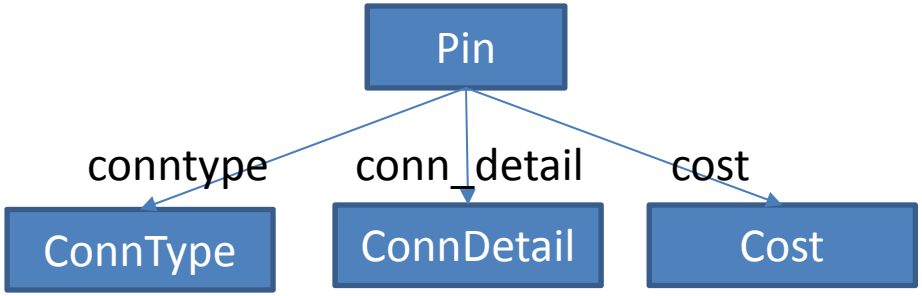
  - Inference

```
getConfig(RequiredConfiguration, Pair) :-
        msort(RequiredConfiguration, S),
        config(S, Pair).

allConfigs(RequiredConfiguration, Set) :-
    setof([Pins,Costs],
        getConfig(RequiredConfiguration,
        [Pins,Costs]), Set).

cheapestConfig(R, Pins, Costs) :-
    setof([Pins,Costs],
    getConfig(R, [Pins,Costs]), Set), Set = [_|_],
    minimal(Set, [Pins,Costs]).
```

Alloy Approach



Alloy meta-model

- Instance specification for a pin

```
one sig PA2 extends Pin {} {
conntype = ANALOG + SERIAL_TX + ICU + ICU
conn_detail = ADC1_IN2 + UART2_TX + TIM2_CH3 + TIM5_CH3
cost = 4}
```

- Generated negated assertion for the desired configuration “ANALOG,ANALOG”.

```
assert ANALOG_ANALOG {
all disj p1, p2:Pin |
not (
ANALOG in p1.conntype &&
ANALOG in p2.conntype &&
p1.cost.add[p2.cost] < = 3
)}
check ANALOG_ANALOG
```

Result

Prolog

Length	Costs for feasible assignment	Computation time for feasible configuration	Computation time for impossible configuration
1	3	0s	0s
2	7	0s	0s
3	11	0s	0s
4	13	0s	0.01s
5	15	0.03s	0.02s
6	17	0.11s	0.10s
7	19	0.29s	0.30s
8	21	0.78s	0.64s
9	23	1.06s	1.06s
10	26	2.47s	1.36s
		∅ =0.474s ± 0.79s	∅ =0.349s ± 0.50s

Prolog

Length	Number of all possible assignments	Costs for best assignment	Prolog computation time (all/best)
1	5	**2**	0s/0s
2	10	**4**	0s/0s
3	10	**7**	0s/0s
4	24	**9**	0.01s/0.01s
5	11	**13**	0.06s/0.03s
6	2	17	0.22s/0.11s
7	8	19	0.61s/0.30s
8	20	21	1.40s/0.64s
9	20	23	2.42s/1.08s
10	32	26	4.06s/1.38s
			$\varnothing_{all}$ = 0.878s ± 1.375s $\varnothing_{best}$ = 0.355s ± 0.51s

Alloy

Length	Costs for the first feasible assignment	Computation time for feasible configuration	Computation time for impossible configuration
1	3	0.53s	-
2	7	0.52s	0.52s
3	11	0.56s	0.53s
4	13	0.54s	0.53s
5	15	0.56s	0.53s
6	17	0.57s	0.64s
7	19	0.62s	0.62s
8	22	0.63s	0.56s
9	23	0.65s	0.67s
10	26	0.67s	0.68s
		∅ =0.58s ± 0.05s	∅ =0.59s ± 0.06s

Alloy

Length	Number of all possible assignments	Costs for best assignment	Alloy computation time (all/best)
1	5	**2**	0.07s/0.53s
2	20	**4**	0.24s/0.63s
3	60	**7**	0.59s/0.67s
4	480	**9**	1.57s/1.63s
5	840	**13**	2.27s/1.20s
6	720	17	2.16s/1.17s
7	2760	19	4.68s/1.09s
8	7320	**21**	10.43s/3.25s
9	7320	23	9.27s/2.88s
10	9960	26	14.12s/3.38s
			$\varnothing_{all}$ = 4.58s ± 5.02s $\varnothing_{best}$ = 1.64s ± 1.11s

Other Observation

Generated specification size

Prolog	Alloy
1.7GB	100KB

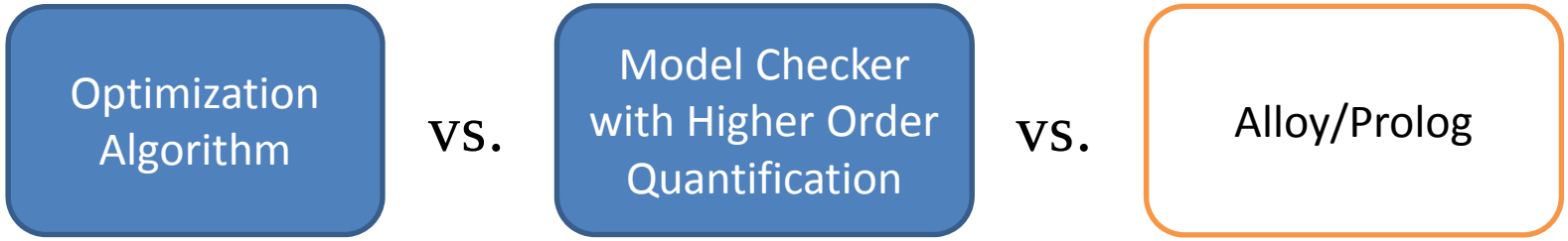
Time to generate specification

Prolog	Alloy
2102s (~35min)	Less than 1s

Time to load specification

Prolog	Alloy
347s (~6min)	Less than 1s

Future Work



- Extend the work with other available COTS interface boards (Odroid, Rapsberry, Arduino, etc.)

- How good are the tools wrt. longer configuration length and more pins?

Summary & Conclusion

- ✓ Prolog performs up to more than 3-times faster to find all possible and best solution
- ✓ Alloy performs up to more than 3-times faster to find feasible solution and reporting insolvable configurations
- ✓ Alloy does not directly support higher order quantification but workaround solution is possible
- ✓ With the considered total number of pins (16)
  and configuration length (10)
  both Prolog and Alloy offers practicable solution.

Thank You