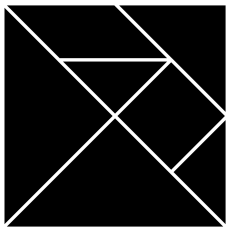


# A Study of non-Boolean Constraints in Variability Models of an Embedded Operating System

*FOSD 2011*

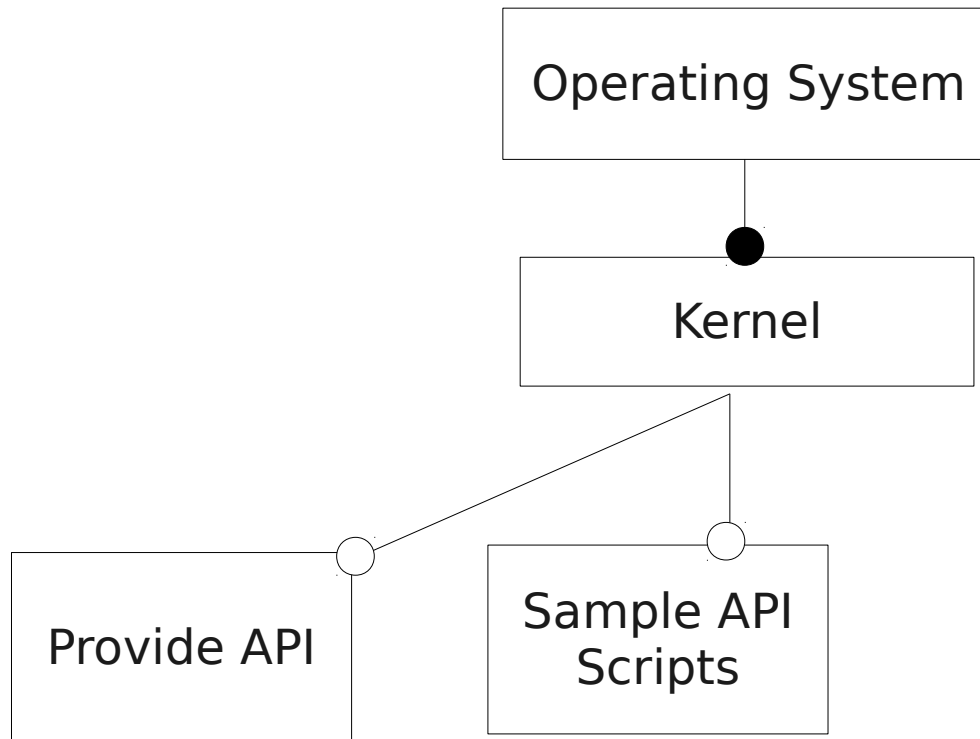


Leonardo Passos, **Marko Novakovic**, Yingfei Xiong,  
Krzysztof Czarnecki @ University of Waterloo  
Thorsten Berger @ University of Leipzig  
Andrzej Wasowski @ IT University of Copenhagen

# Contents

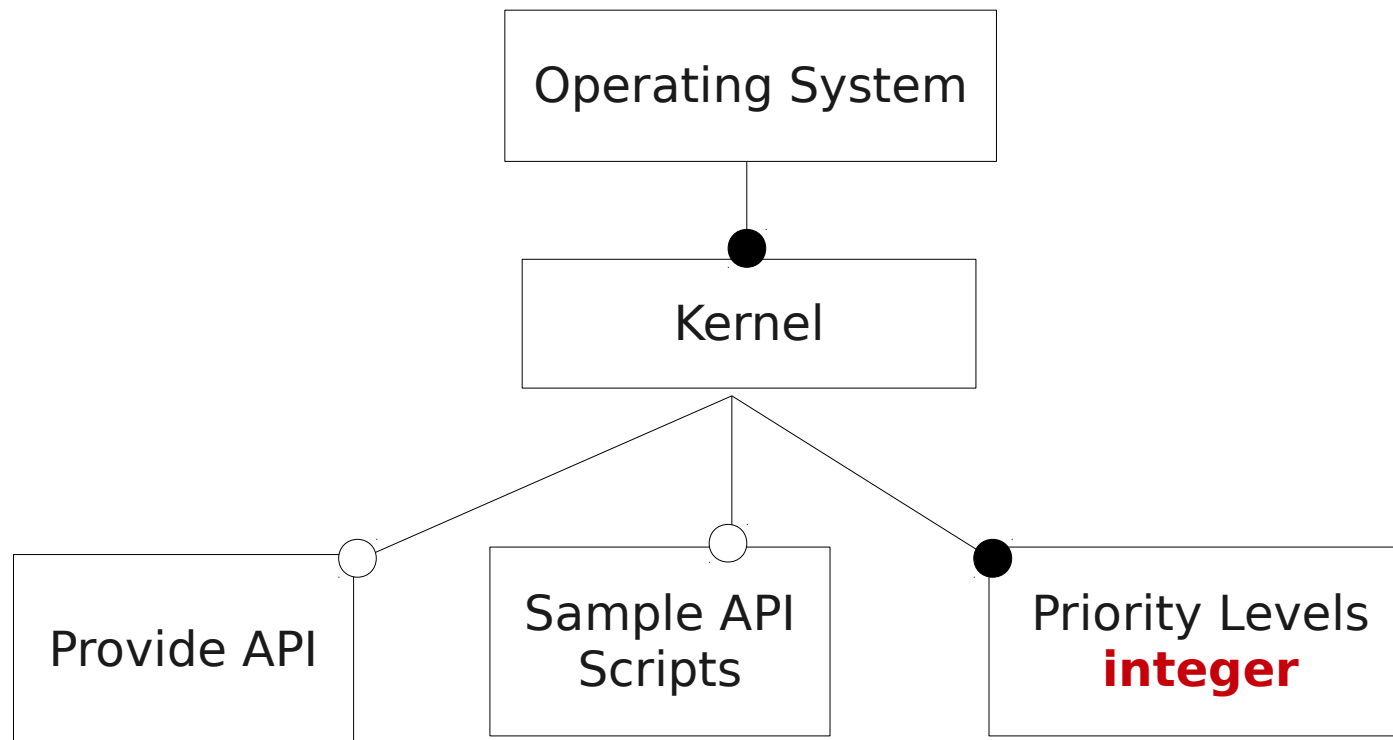
- Non-Boolean FMs
- Motivation
- eCos
- Results
  - Non-linear arithmetic constraints
- Conclusions

# Non-Boolean FMs



Sample API Scripts  $\Rightarrow$  Provide API

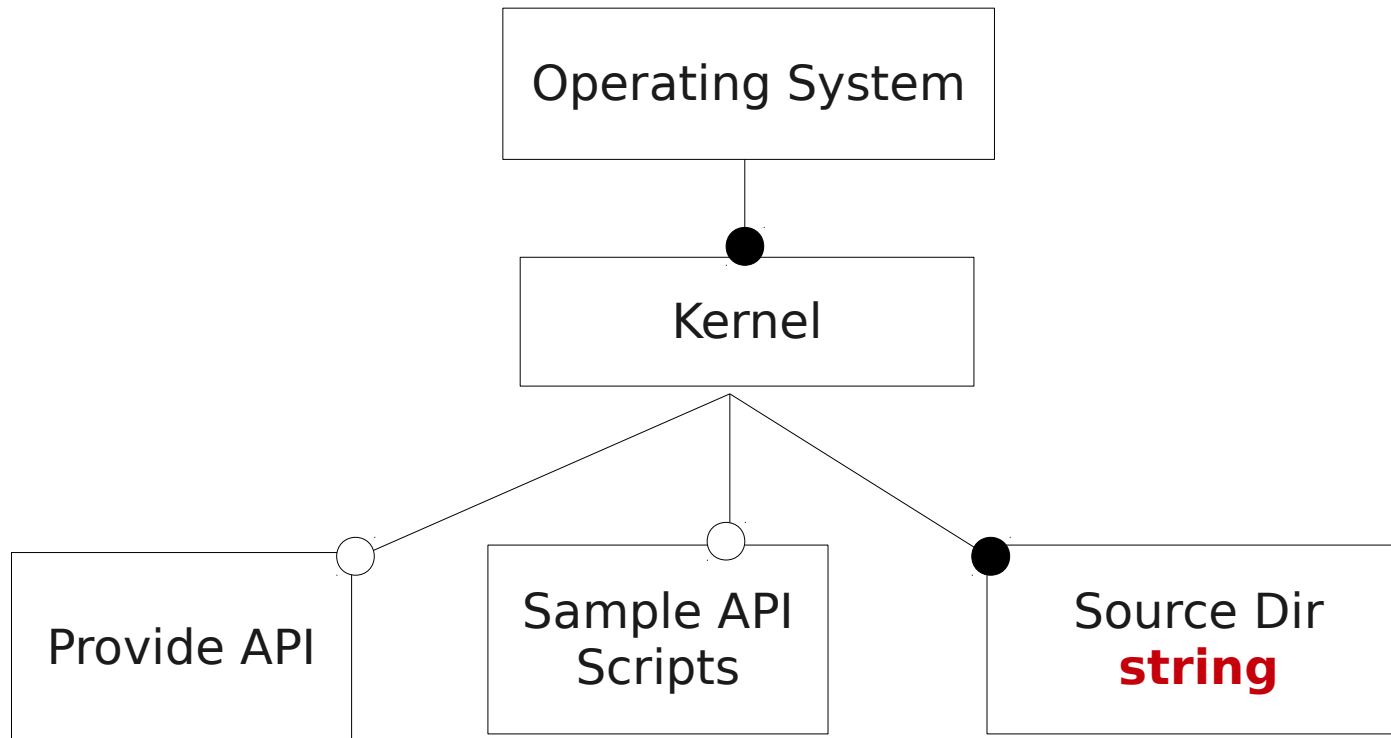
# Non-Boolean FMs



Sample API Scripts  $\Rightarrow$  Provide API

Priority Levels  $\geq 1$  && Priority Levels  $< 32$

# Non-Boolean FMs



Sample API Scripts  $\Rightarrow$  Provide API  
(Source Dir) . contains("src")

# Sample non-Boolean constraint

API\_SCRIPTS && LEVELS  $\leq$  32 &&

(BLOCK\_SIZE \* BLOCK\_COUNT + SWAP\_SIZE  $\leq$  MEM\_SIZE) &&

BASE\_LIB **contains** (LINUX ? “.so” : “.dll”) &&

SRC\_DIR **contains** (“src”)

$\Rightarrow$  ENABLE\_API

# Non-Boolean FMs

Contain constraints with:

- Arithmetic, Relational and String operations
- Integer, Float, String, Boolean operands

SAT checking is hard

- Boolean Constraints — NP Complete
- Integer, String and Float — undecidable in general

# Motivation

The Goal:

What constraints are used in practice?



# Motivation

The Goal:

What constraints are used in practice?

Why is that important?

# Motivation

We need efficient reasoning to:

- Better support configuration guidance
- Do model analyses – dead features detection
- List valid configurations

# Motivation

However:

- Constraints are hard to solve, potentially undecidable
- Can we use existing tools to reason over them?

# Motivation

## Benchmark for tool developers

- Add support for new constraints
- Optimize existing tools

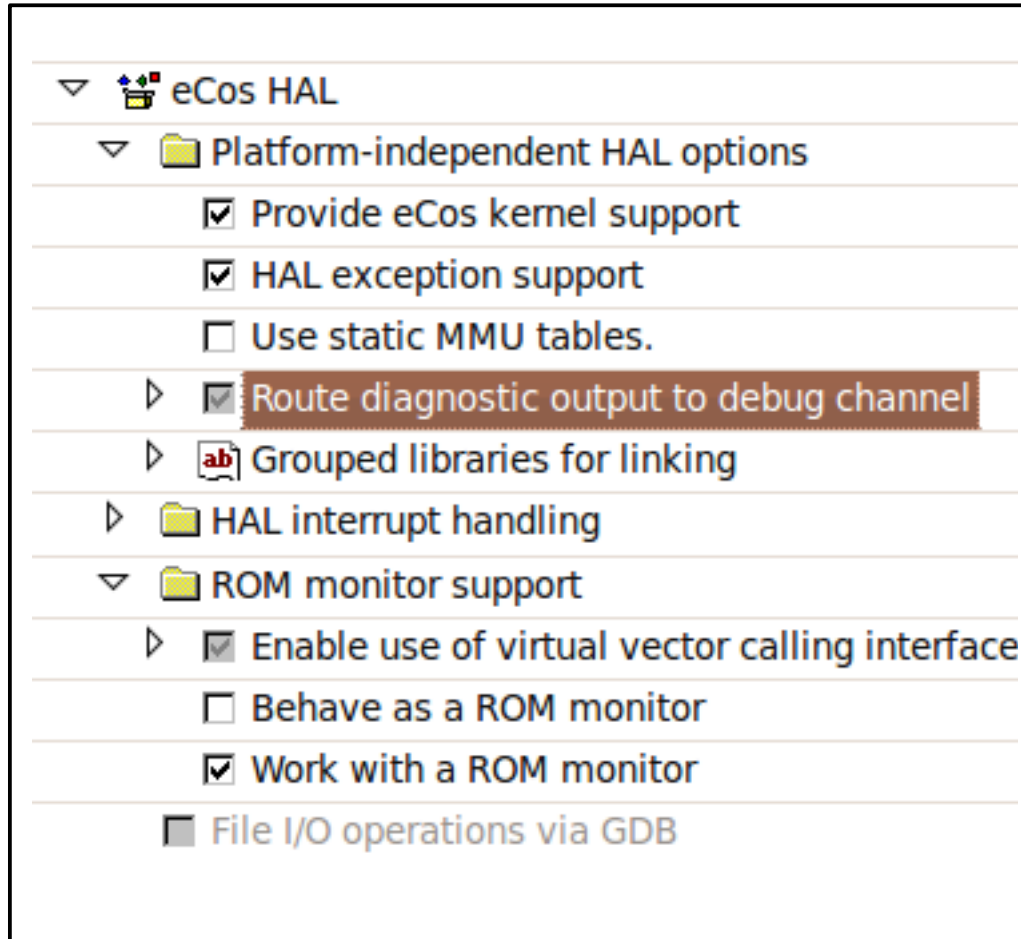
# Subject of the study

## *Embedded Configurable Operating System*

- Non-Boolean Feature Model
- Publicly Available



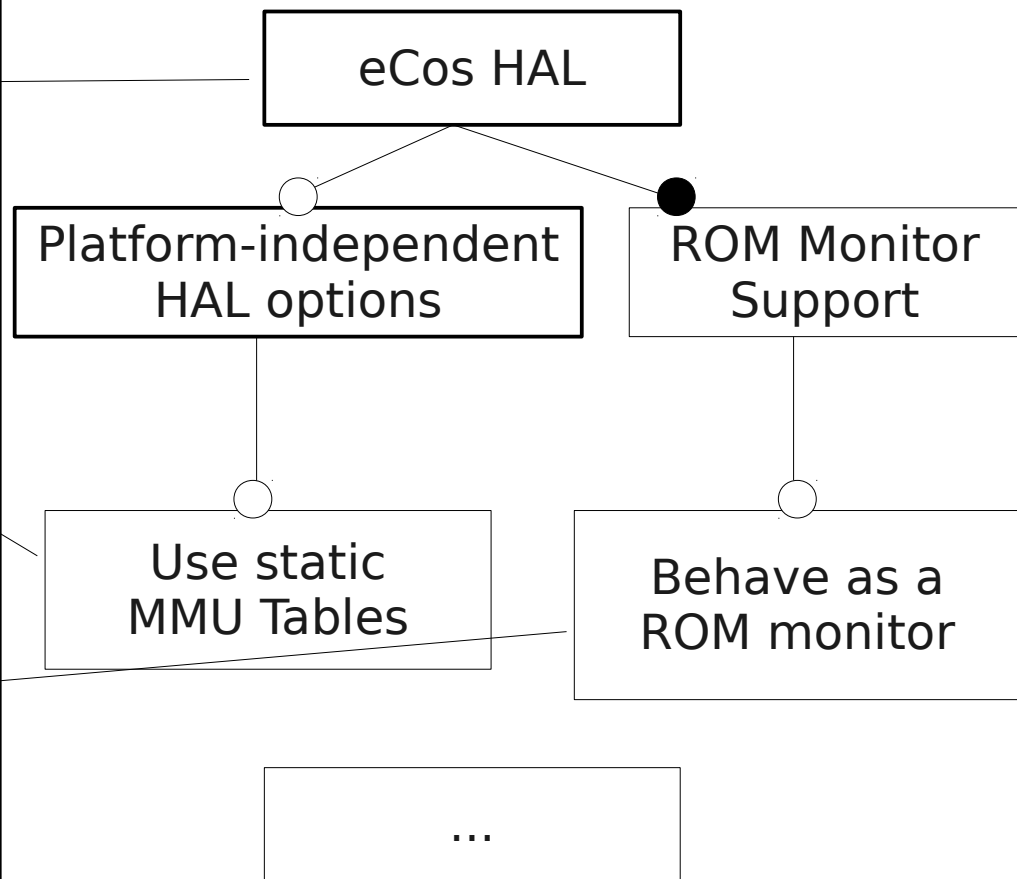
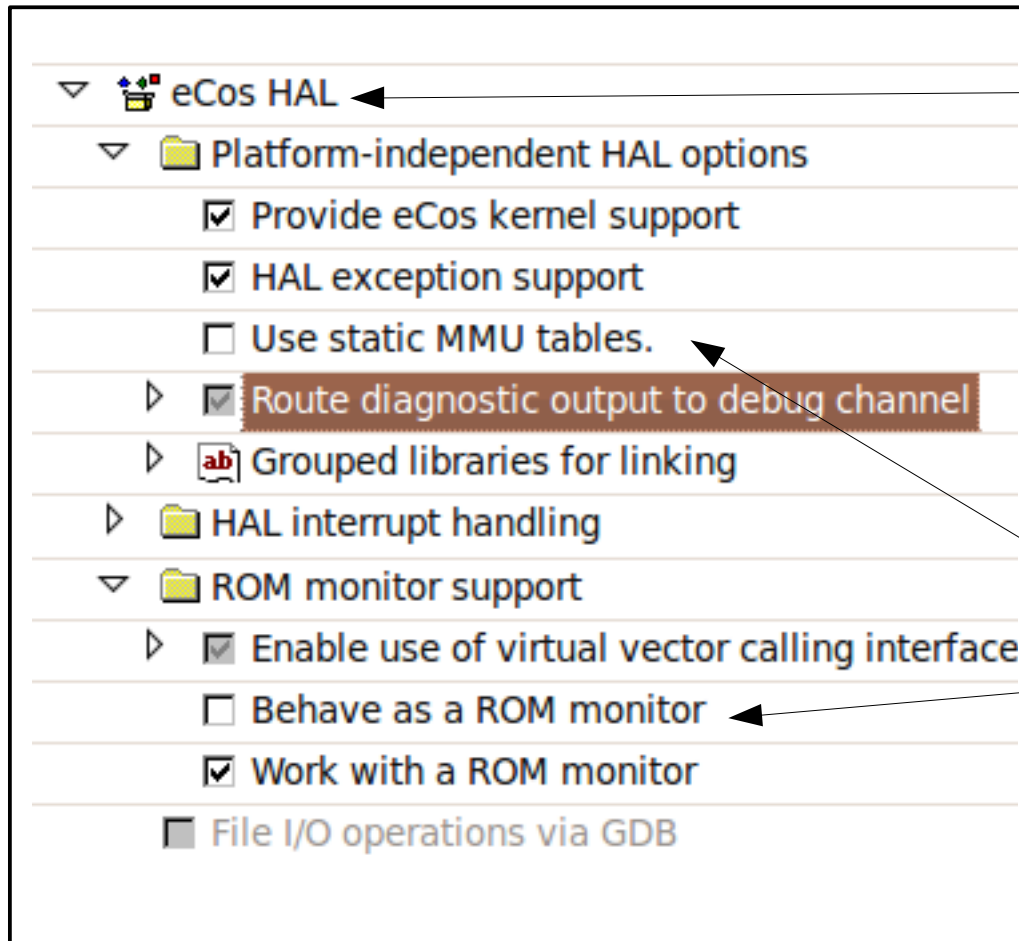
## 116 Architectures



Configuration done using the Configurator

## 116 Architectures

## Each is a Feature Model



Configuration done using the Configurator

# CDL

Domain-specific variability language provided by eCos

...

```
cdl_option CYGNUM_KERNEL_SCHED_BITMAP_SIZE {  
    display "Bitmap size"  
  
    requires CYGNUM_KERNEL_SCHED_PRIORITIES > 2  
  
    flavor data  
}
```

...



# CDL

Domain-specific variability language provided by eCos

...

```
cdl_option CYGNUM_KERNEL_SCHED_BITMAP_SIZE {  
  display "Bitmap size"
```

```
  requires CYGNUM_KERNEL_SCHED_PRIORITIES > 2
```

```
  flavor data
```

```
}
```

...

# CDL

Domain-specific variability language provided by eCos

...

```
cdl_option CYGNUM_KERNEL_SCHED_BITMAP_SIZE {  
  display "Bitmap size"
```

```
  requires CYGNUM_KERNEL_SCHED_PRIORITIES > 2
```

```
  flavor data
```

```
}
```

...

# Analyzing eCos

Different aspects for analyses.

# Analyzing eCos

Different aspects for analyses.

Syntactic

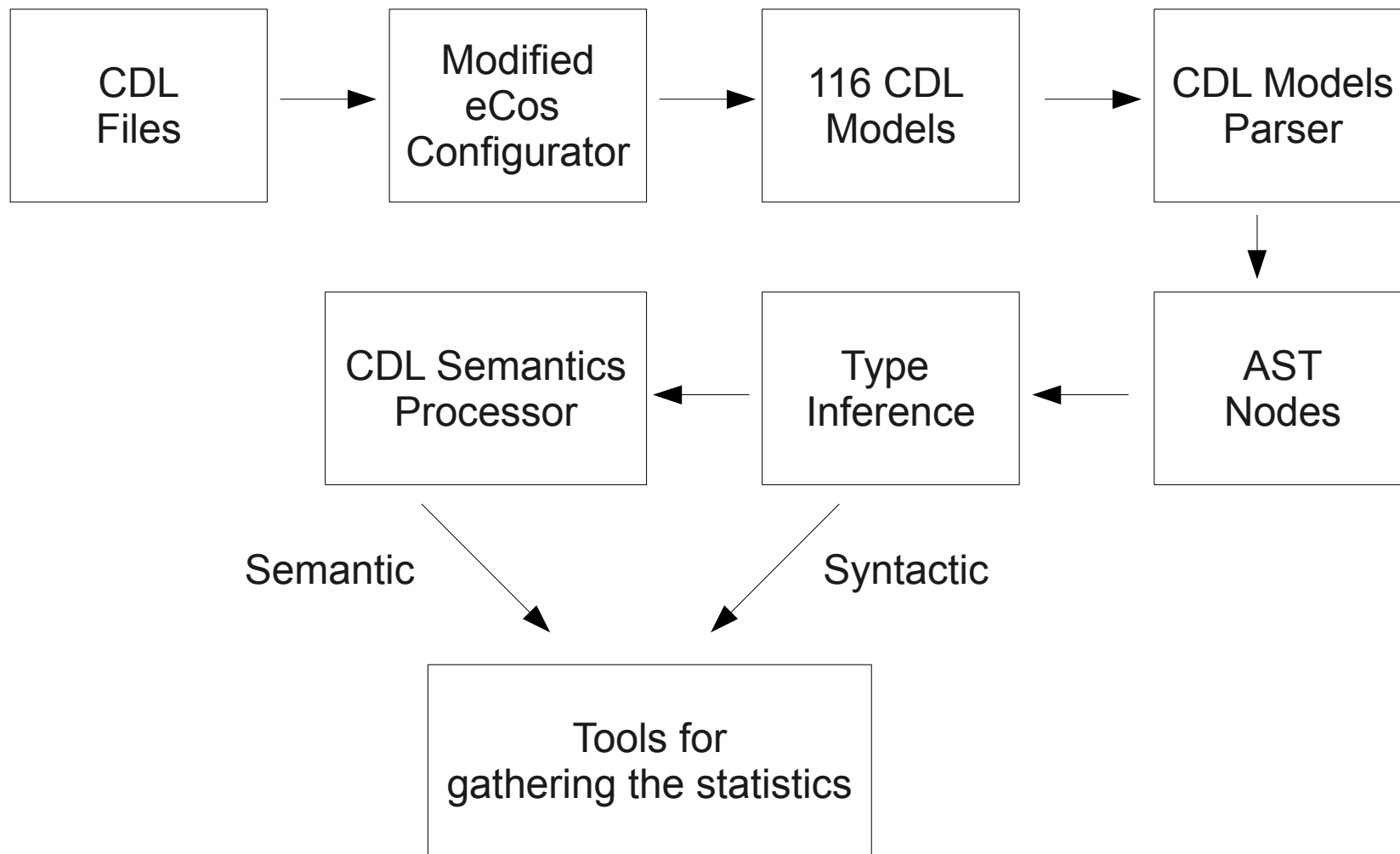
- **Models as created by eCos developers**

Semantic

- Configuration setting used by code generator
- **The behavior of the Configurator**
  - Richer semantics, for interactive support
  - E.g., is a feature active in the GUI or not

# Methodology

## The Toolchain



# Methodology

## Reverse engineering formal specification of CDL semantics

CDL Files

CDL Semantics Processor

Type Inference

AST Nodes

## Dynamic type inference

# The Results

Summary statistics (min, max, med)  
over 116 eCos models

# 1. Feature Types Proportions

eCos has 3 types of features

- Number (Integer and Float)
- String
- Boolean

Why?

- Many non-Boolean features can not be ignored



# 1. Feature Types Proportions

Total # of features:

1230 Median

1312 Maximum

1159 Minimum

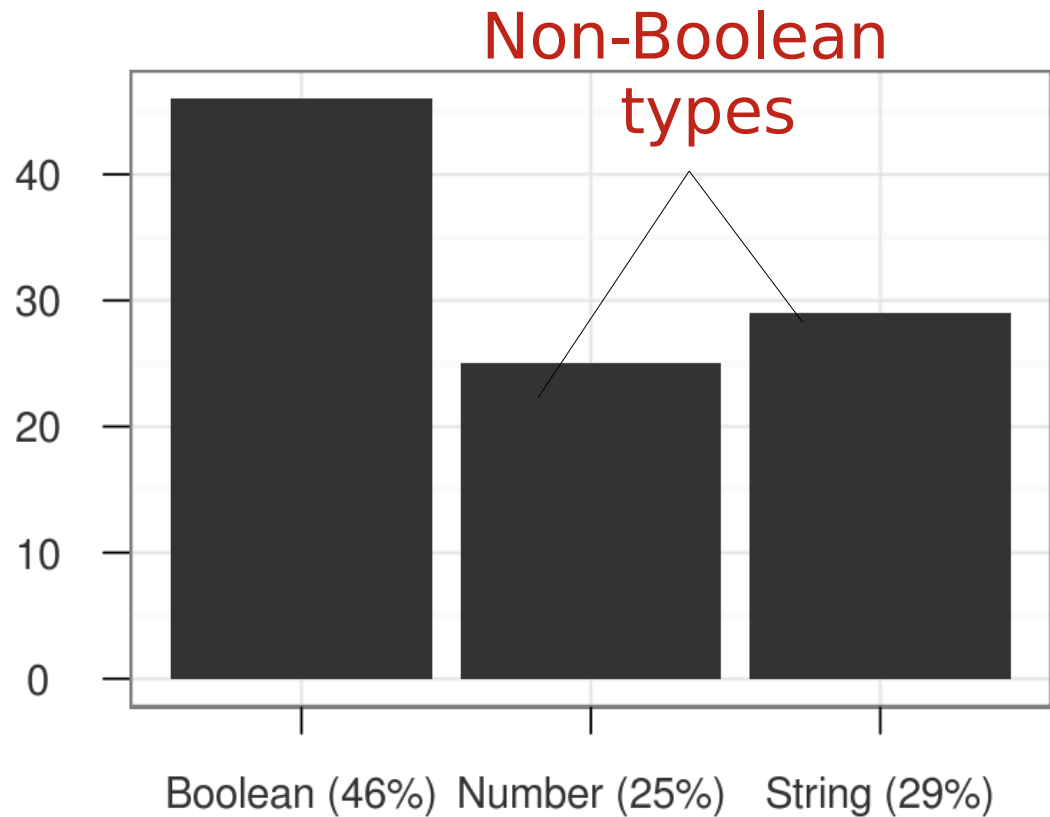


Figure: feature types - median value

## 2. Restriction on non-Boolean types

Static constraints effectively specifying types (sets of values)

- Ranges - 1 to 7
- Constants - "ROM"
- Enumerations - {1, 2, 3}
- Unrestricted - just **string** or **integer**

## 2. Restriction on non-Boolean types

### Advantages:

- Model simplification
- Shrinking the domain
- Replace constants occurrences with the value
- Enumerations are “easier” than integers

## 2. Restriction on non-Boolean types

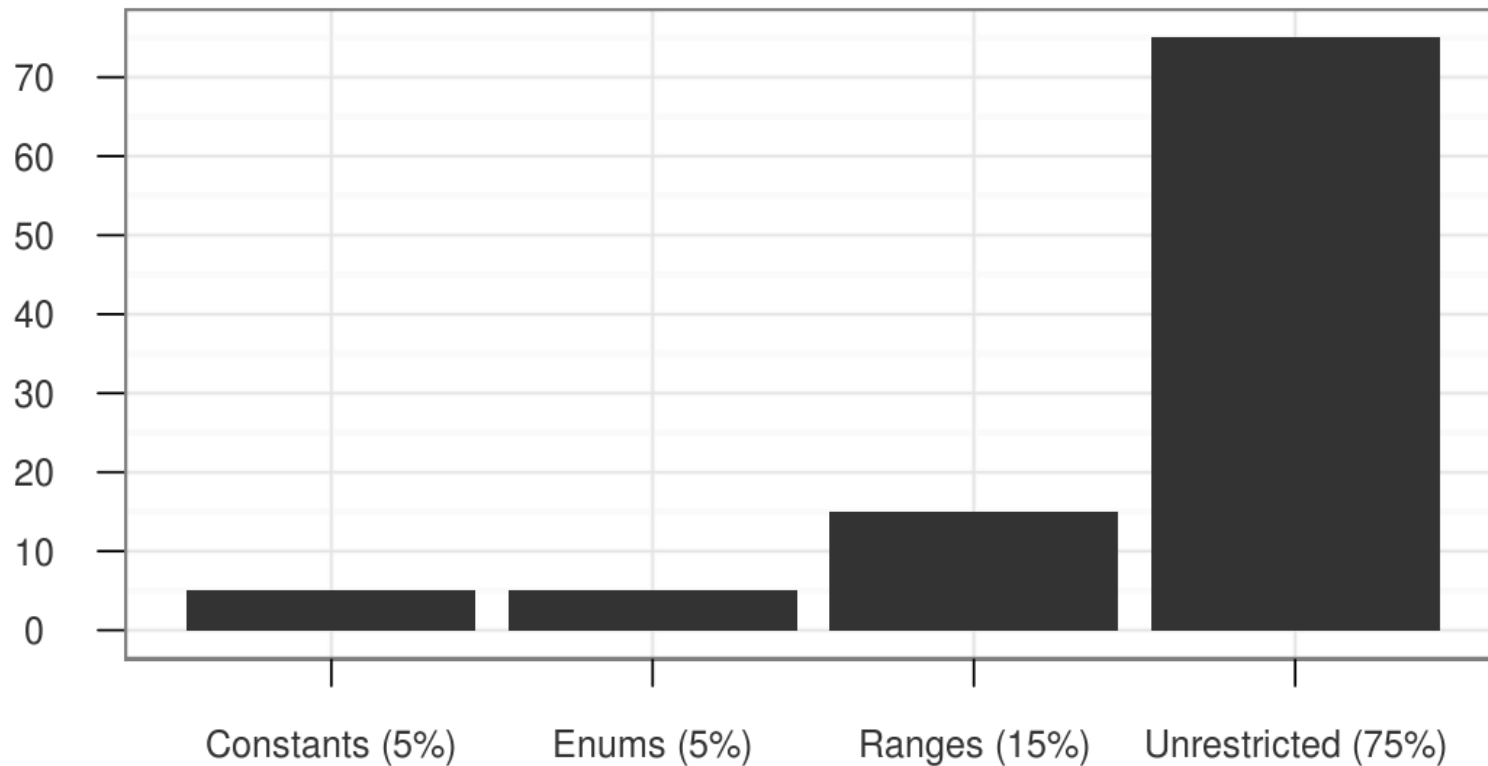


Figure: restrictions - median value

# 3. The Constraints (Syntactic level)

## Constraints classification:

- Purely Boolean
  - Boolean operators and features
  - $A \ \&\& \ B, A \ \|\ B$
- Purely non-Boolean
  - Non-Boolean operators and features
  - $A + 10 == C$
- Mixed
  - $B \ \&\& \ (A + 10 == C)$

# 3. The Constraints (Syntactic level)

We want to do efficient analysis over the constraints

- We want to better understand the hardness of the Real World constraints
- Purely Boolean – SAT solving

# 3. The Constraints (Syntactic level)

Number of constraints:

1015 Median  
1269 Maximum  
916 Minimum

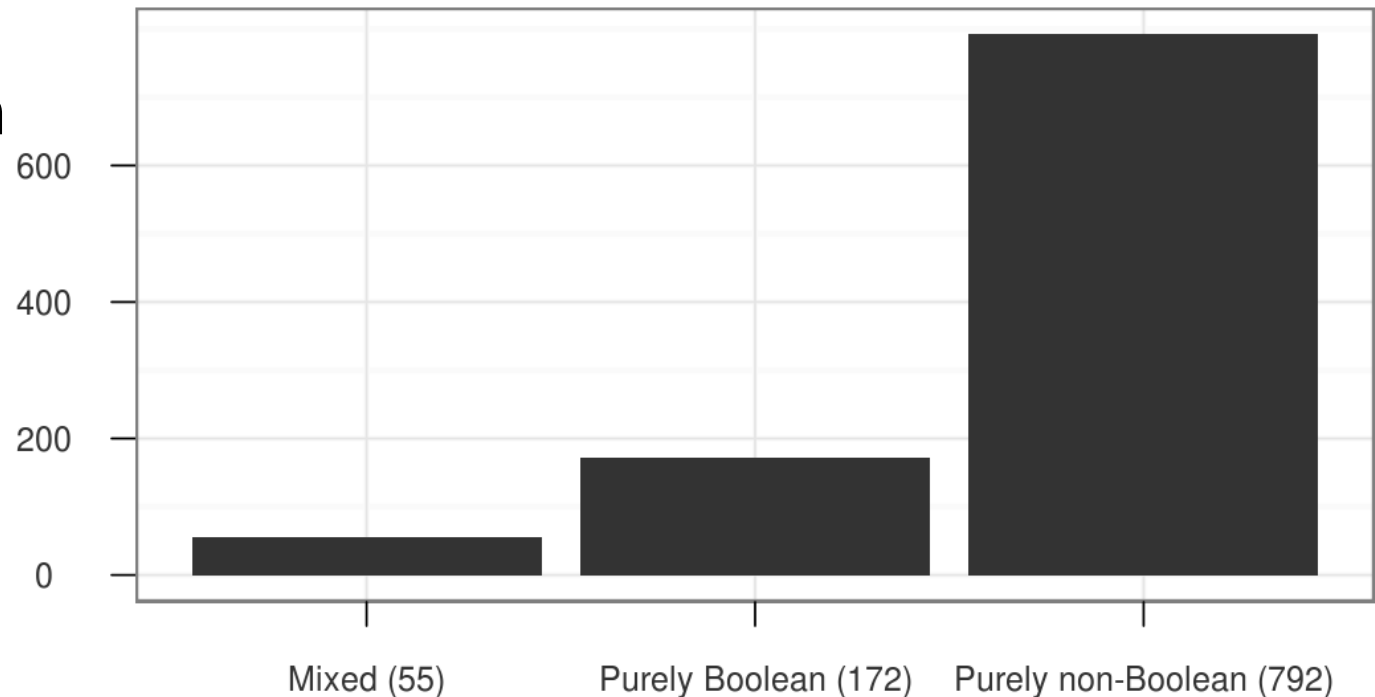


Figure: No. of constraints - median value

# 4. Semantic Constraints

## Capturing the configurator behavior

|  |         |              |                                    |
|--|---------|--------------|------------------------------------|
| Source-level debugging support   |         | Macro        | CYGDBG_HAL_DEBUG_GDB_BREAK_SUPPORT |
| <input type="checkbox"/> Include GDB stubs in HAL                                    |         | File         |                                    |
| <input type="checkbox"/> Include GDB external break support for stubs                |         | Enabled      | False                              |
| <input checked="" type="checkbox"/> Include GDB external break support when no stubs |         | DefaultValue | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| <input checked="" type="checkbox"/> Include GDB multi-threading debug support        |         | ActiveIf     | CYGINT_HAL_DEBUG_GDB_STUBS_BREAK   |
| <input type="checkbox"/> Number of times to retry sending a \$O packet               | 0       | Requires     | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| <input type="checkbox"/> Timeout period for GDB packets                              | 500     |              |                                    |
| <input type="checkbox"/> Location of CRC32 table                                     | RAM     |              |                                    |
| ROM monitor support  |         |              |                                    |
| <input type="checkbox"/> File I/O operations via GDB                                 |         |              |                                    |
| <input type="checkbox"/> Build Compiler sanity checking tests                        |         |              |                                    |
| <input type="checkbox"/> Common HAL tests  | tests/c |              |                                    |
| FUJITSU architecture   | v3_0    |              |                                    |

Figure: The configurator



# 4. Semantic Constraints

## Capturing the configurator behavior

The image shows a configurator interface with a tree view on the left and a property table on the right. The tree view is expanded to 'Source-level debugging support'. The option 'Include GDB external break support for stubs' is highlighted with a red circle and a brown background. The property table on the right lists the macro 'CYGDBG\_HAL\_DEBUG\_GDB\_BREAK\_SUPPORT' with its file, enabled status, default value, active if, and requires fields.

|  |         |              |                                    |
|--|---------|--------------|------------------------------------|
| Source-level debugging support   |         | Macro        | CYGDBG_HAL_DEBUG_GDB_BREAK_SUPPORT |
| <input type="checkbox"/> Include GDB stubs in HAL                                    |         | File         |                                    |
| <input checked="" type="checkbox"/> Include GDB external break support for stubs     |         | Enabled      | False                              |
| <input checked="" type="checkbox"/> Include GDB external break support when no stubs |         | DefaultValue | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| <input checked="" type="checkbox"/> Include GDB multi-threading debug support        |         | ActiveIf     | CYGINT_HAL_DEBUG_GDB_STUBS_BREAK   |
| <input checked="" type="checkbox"/> Number of times to retry sending a \$O packet    | 0       | Requires     | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| <input checked="" type="checkbox"/> Timeout period for GDB packets                   | 500     |              |                                    |
| <input checked="" type="checkbox"/> Location of CRC32 table                          | RAM     |              |                                    |
| ROM monitor support  |         |              |                                    |
| <input checked="" type="checkbox"/> File I/O operations via GDB                      |         |              |                                    |
| <input type="checkbox"/> Build Compiler sanity checking tests                        |         |              |                                    |
| <input checked="" type="checkbox"/> Common HAL tests                                 | tests/c |              |                                    |
| FUJITSU architecture   | v3_0    |              |                                    |

Figure: Enabling features

# 4. Semantic Constraints

## Capturing the configurator behavior

The image shows a configurator interface with a tree view on the left and a property table on the right. The tree view is expanded to 'Source-level debugging support'. The following options are visible:

- Include GDB stubs in HAL
- Include GDB external break support for stubs
- Include GDB external break support when no stubs
- Include GDB multi-threading debug support
- Number of times to retry sending a \$O packet: 0
- Timeout period for GDB packets: 500
- Location of CRC32 table: RAM

The right-hand table shows the following properties for the selected option:

|              |                                    |
|--------------|------------------------------------|
| Macro        | CYGDBG_HAL_DEBUG_GDB_BREAK_SUPPORT |
| File         |                                    |
| Enabled      | False                              |
| DefaultValue | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| ActiveIf     | CYGINT_HAL_DEBUG_GDB_STUBS_BREAK   |
| Requires     | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |

Figure: Providing the data

# 3. Semantic Constraints

## Capturing the configurator behavior

The image shows a configuration tool interface with two panes. The left pane displays a tree view of configuration options under 'Source-level debugging support'. The right pane shows the details for the selected option, 'Include GDB stubs in HAL'.

| Option   | Value   |
|--|---------|
| Source-level debugging support   |         |
| <input type="checkbox"/> Include GDB stubs in HAL                                    |         |
| <input type="checkbox"/> Include GDB external break support for stubs                |         |
| <input checked="" type="checkbox"/> Include GDB external break support when no stubs |         |
| <input checked="" type="checkbox"/> Include GDB multi-threading debug support        |         |
| <input type="checkbox"/> Number of times to retry sending a \$O packet               | 0       |
| <input type="checkbox"/> Timeout period for GDB packets                              | 500     |
| <input type="checkbox"/> Location of CRC32 table                                     | RAM     |
| ROM monitor support  |         |
| <input type="checkbox"/> File I/O operations via GDB                                 |         |
| <input type="checkbox"/> Build Compiler sanity checking tests                        |         |
| <input type="checkbox"/> Common HAL tests  | tests/c |
| FUJITSU architecture   | v3_0    |

|              |                                    |
|--------------|------------------------------------|
| Macro        | CYGDBG_HAL_DEBUG_GDB_BREAK_SUPPORT |
| File         |                                    |
| Enabled      | False                              |
| DefaultValue | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |
| ActiveIf     | CYGINT_HAL_DEBUG_GDB_STUBS_BREAK   |
| Requires     | CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS |

Red circles and arrows highlight the 'Include GDB stubs in HAL' option in the left pane and the 'Requires' field in the right pane, indicating a semantic constraint.

Figure: A constraint

# 4. Semantic Constraints

## Capturing the configurator behavior



Figure: Conflict

# 4. Semantic Constraints

Capturing the configurator behavior

We transform the model:

- Enable state variables - enabled\_var
- Data variables - data\_var
- Constraints mapping the conflicts

# 4. Semantic Constraints

## Semantic constraints classification:

- Purely Boolean
  - Enabled state variables
  - Boolean operators
- Purely non-Boolean
  - Data state variables
  - non-Boolean operators – relational, string, arithmetic
- Mixed

# 4. Semantic Constraints

Number of constraints:

616 Median

686 Maximum

593 Minimum

Median number of variables:

420 Data

521 Enabled

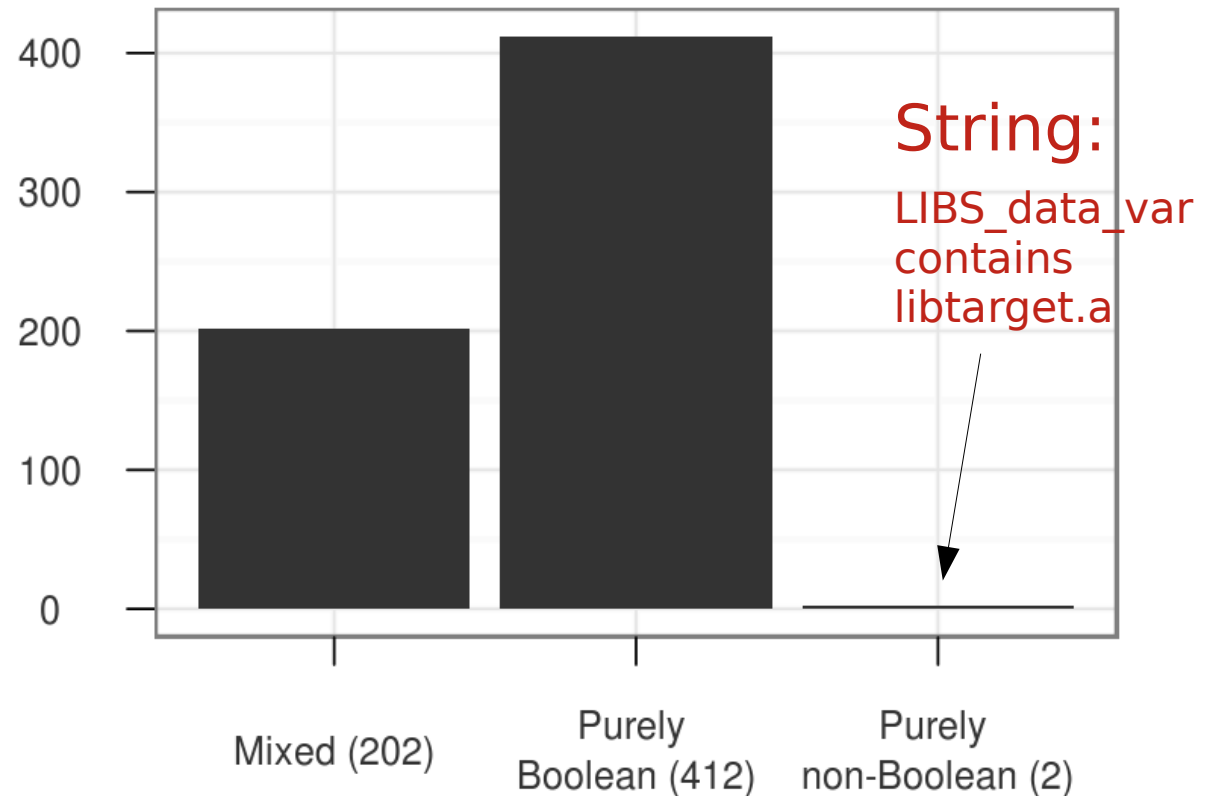


Figure: Number of occurrences  
median value

# 5. Semantic Expansion - Patterns

Sample eCos pattern:

```
(1 ≤  
(  
  ((  
    (RTC_NUMERATOR_data *  
      ((OSC_MAIN_data * PLL_MULTIPLIER_data) / PLL_DIVIDER_data) / 2)  
    )  
    / (TIMER_TC_enabled ? 32 : 16)  
  ) / RTC_DENOMINATOR_data) / 1000000000  
)  
)
```



# 5. Semantic Expansion - Patterns

Patterns:

$aXY \quad \begin{matrix} \geq \\ \leq \end{matrix} \quad b, \text{ max. occurrences} = 2$

$aXY / Z \quad \begin{matrix} \geq \\ \leq \end{matrix} \quad b, \text{ max. occurrences} = 2$

$aXY / PZ \quad \leq \quad b, \text{ max. occurrences} = 1$

$aXYZ/(\alpha+\beta)PQ \quad \begin{matrix} \geq \\ \leq \end{matrix} \quad b, \text{ max. occurrences} = 2$

# More details in the paper

- Boolean, number and string operator occurrence frequency at semantic and syntactic
- Explanation of the semantics
- All 116 models as Clafer models are available @ <http://gsd.uwaterloo.ca/FOSD11>

# Conclusions

- Studied 116 real-world non-Boolean FM
- ~50% of features are non-Boolean (numbers and strings)
- ~70% of constraints are non-Boolean
- Some constraints are complex (e.g. non-linear)
- Provided 116 models as a benchmark for tool builders
  
- Such non-Boolean models are likely to occur in embedded systems

## Future:

- Provide reasoning techniques that work on these constraints

Thank you!

Questions?