Formally Checking Large Data Sets in the Railways - engineering approach -

Lilian Burdy
Thierry Lecomte (speaker)

Michael Leuschel
Definitions

Data validation ≡
Automatic check of large data sets against properties

100,000+ raw data chunks

Expressed using B mathematical language

Are they
• Consistent ?
• Correct ?
• Safe ?

Model-checker (no human in the loop)

Data generation ≡
Data validation of partly instantiated large data sets

Is it possible to calculate missing data such as they are all
• Consistent ?
• Correct ?
• Safe ?
**Rationale**

Data are uploaded on embedded equipments for exploitation
Any error may lead to an accident

**Software modelling**

- **Target:** Cyclic, monolithic software
  - No IT, no OOP
  - Source code generated, mathematical demonstrations / scripts
- **B models** → **Event-B models**

**System modelling**

- **Target:** Systems
  - Source code generated, Animation, mathematical demonstrations / scripts

**Data validation**

- **Data**
  - CSV, Excel, Text
- **Rules**
  - B mathematical language XML
- **System Level Database**
- **Compliance**
  - Counter-examples
- **Data generation**
  - Data are uploaded on embedded equipments for exploitation
  - Any error may lead to an accident
Data sets ≡

Data describing the topology of the track

- **Addressing plan**: networked equipments, IP addresses
- **Scheme plan**
- **System Data**: 101 tables, around 50,000 data for Mexico L12
Application to railways

Raw data as inputs
- csv files, every csv column is a **constant** in the B model

- data are not preprocessed: everything is modeled in B mathematical language

- Supported types: BOOL, INT, STRING, seq(INT), seq(STRING).
Properties \equiv Relationships between the data

- Properties expressed with B mathematical language and decorated with substitution-like syntax
- Use of intermediate constructs to factorize development
- Simple specification and detection of counter-examples
Application to railways

RULE DB_GENERAL.3
COUNTEREXAMPLE

the name %1 is the name of an equipment of type ZC but is not in table ZC

ANY
name1, ind2

TYPE
STRING, INT

WHERE
ind2 : dom(ATC_Equipments_Cap!Name) &
ATC_Equipments_Cap!ATC_Equipment_Type(ind2) = "ZC" &
ATC_Equipments_Cap!Name(ind2)=name1

EXPECTED
#ind1.( ind1 : dom(ZCs_Cap!Name) & name1=ZCs_Cap!Name(ind1))

END

Rule name
Values to search for
Conditions to fulfill
If not fulfilled, counterexample is found and error message is displayed

A rule can be made of several sequential searches for counterexamples
**Intermediate construct**: Associate to each secondary detection device(sdd) and each consecutive points on the same track of the sdd the part of the track between the two points.

\[ SDD\_Point\_Normal\_Normal: \text{dom}(\text{Secondary\_Detection\_Devices\_Cap!ID}) \rightarrow (\text{dom}(\text{Points\_Cap!Name}) \times \text{dom}(\text{Points\_Cap!Name})) \rightarrow \text{dom}(\text{Tracks\_Cap!Name})*(\text{INT}\times\text{INT}) \]

\[ SDD\_Point\_Normal\_Normal = \%sdd.(sdd : \text{dom}(\text{Secondary\_Detection\_Devices\_Cap!ID}) \& \text{size}(\text{Secondary\_Detection\_Devices\_Cap!Point\_ID\_List}(sdd)) \geq 2 \]

| \%(point1,point2).(point1 : \text{SDD\_Point\_list}(sdd) \\
| & point2 : \text{SDD\_Point\_list}(sdd) \\
| & point1 /= point2 \\
| & \text{Points\_Cap!Track\_ID}(point1) = \text{Points\_Cap!Track\_ID}(point2) \\
| & \text{bool}(\text{TrackKpBegin}(\text{Tracks\_Cap!Name}~\text{Points\_Cap!Track\_ID}(point1))) \leq \text{TrackKpEnd}(\text{Tracks\_Cap!Name}~\text{Points\_Cap!Track\_ID}(point1))) \\
| = \text{bool}(\text{PointKpToe}(point1) \leq \text{PointKpToe}(point2)) \\
| & !\text{point3}.(\text{point3 : SDD\_Point\_list}(sdd) - \{\text{point1,point2}\} \\
| & & \& \text{Points\_Cap!Track\_ID}(point3) = \text{Points\_Cap!Track\_ID}(point1) \\
| | \Rightarrow \text{PointKpToe}(point3) /: \min(\{\text{PointKpToe}(point1),\text{PointKpToe}(point2)\}) .. \max(\{\text{PointKpToe}(point1),\text{PointKpToe}(point2)\})) \\
| | \Rightarrow (\text{min}(\{\text{PointKpToe}(point1),\text{PointKpToe}(point2)\})) \\
| | \Rightarrow \max(\{\text{PointKpToe}(point1),\text{PointKpToe}(point2)\})) \]
Rules to verify:
- 1,000 rules per project (generic / specific corpus)
- 450 rules formalized (rules added progressively as new projects are started)
- Target: 700 rules in 2013

Intermediate constructs:
- 150 (reused from one project to another)
Application to railways

Manual
30 days to verify 300 rules

Data validation process
few hours to verify 300 rules

Application to 20 projects for each major release
Modernization of High Speed Train Embedded Diagnosis System (EDS)

- (ancestor of) hierarchical grafcets
- PASCAL programs (with French keywords)
- Import old models
- IEC 61131 grafcets
- IEC 61131 Structured Text

Compiler AD-HOC

Binary code (EDS)

Given that input models are different

= Such as generated binaries are identical

Challenge

TO Develop a new compiler
IEC 61131-3

grafcets

FUNCTION FI001
169-
170
171
172
173
174
175
176
177
END_FUNCTION

IEC 61131-3
Structured Text

Binary code (EDS)

[0x001404] ADR_VL_VL_DR TRANSFERT 0x00d4 0x0000 0x82be
[0x00140c] ADR_VL_VL_DR TRANSFERT 0x82be 0x0000 0x8306
[0x001414] ADR_VL_VL_DR RMP_TABLE 0x0003 0x00d4 0x82f4
[0x00141c] ADR_VL_VL_DR TFR_TABLE 0x0003 0x8308 0x4004
[0x001424] ADR_VL_VL_DR MODULO 0x82be 0x002a 0x82ae
[0x00142c] ADR_VL_VL_DR ADDITION 0x82ae 0x0012 0x82ae
[0x001434] ADR_VL_VL_DR MULTIPLIE 0x82be 0x0017 0x82b0
[0x00143c] ADR_VL_VL_DR DIVISE 0x82b0 0x82be 0x82be
[0x001444] ADR_VL_VL_DR SOUSTRAC 0x82ae 0x82b0 0x82be
[0x00144c] ADR_VL_VL_DR SOUSTRAC 0x82be 0x0000 0x82c0
[0x001454] ADR_VL_VL_DR SOUSTRAC 0x82c0 0x0000 0x82c0
[0x00145c] RETOUR 0x00
Key idea:
Use data validation/generation process to check binary code against obsolete input models

~30,000 lines of code
162 grafcets
1000+ steps

(ancestor of) hierarchical grafcets

PASCAL programs (with French keywords)

Compiler AD-HOC

Binary code (EDS)

Safety Integrity Level 2

IEC 61131 grafcets

IEC 611131 Structured Text

New Compiler AD-HOC

Binary code (EDS)

800k bytes
~30,000 lines of code
162 grafcets
1000+ steps

Key idea:
Use data validation/generation process to check binary code against obsolete input models

Compiler Certification

[Contribution to]
Key idea:
Use data validation/generation process to check binary code against obsolete input models
Key idea:
Use data validation/generation process to check binary code against obsolete input models

~80 properties identified related to 1200 variables and code
• P01 No more dead code in the binary than in the input models
• P02 RAM memory space usage in binary file should comply with memory access in input models
• P03 Sub-grafcets called in the binary file should comply with sub-grafcets activated in input models

No symbol table available, so structure information should be recovered
→ property verification is ordered in order to reuse data previously generated

Compiler Certification

Stackless – intermediate results stored in specific memory area
• P03 Sub-grafcets called in the binary file should comply with sub-grafcets activated in input models
• P03 Sub-grafcets called in the binary file should comply with sub-grafcets activated in input models

List grafcat activations (old models)

Build B model of activations

\[ G7 = \{ \text{main}, G1, G2, G3, G4, \ldots \} \]
\[ \text{next: } G7 \leftrightarrow G7 \]
\[ \text{next} = \{ \ldots, G7 \rightarrow G11, \ldots \} \]

List grafcat activations (binary)

Build B model of activations

\[ \text{ADR} = \{0x01, 0x13, 0x15, \ldots \} \]
\[ \text{suiv: } \text{ADR} \leftrightarrow \text{ADR} \]
\[ \text{suiv} = \{ \ldots, 0x10 \rightarrow 0x15, \ldots \} \]

there exists a bijection \( \text{bij} \) that associates to a node of \( G7 \) a node of \( \text{ADR} \) such as children of both nodes match

\[ \text{bij: } G7 \leftrightarrow \text{ADR} \&!xx.(xx: G7 \rightarrow \text{bij}[\text{next}[\{xx\}]] = \text{suiv}[\text{bij}[\{xx\}]]) \]
Modelling completed in 2 days

Complete verification performed in 2 minutes:
- Models and binary match
- Some errors found like:
  - infinite loop (G13 activates G23, G23 activates G13)
  - dead code (elements declared but never used)
Data validation & data generation able to deal with industrial problems

- Data validation time divided by 10 at least
- Automation slightly improves the level of confidence
Conclusion & perspectives

Data validation & data generation able to deal with industrial problems
- Data validation time divided by 10 at least
- Automation slightly improves the level of confidence

Technology is mature
- Several R&D projects to assess and improve tools and methods
- Daily production on worldwide applications not restricted to B
  - Proprietary tools
  - Atelier B 4.1 integrates data validation projects
Conclusion & perspectives

Data validation & data generation able to deal with industrial problems
- Data validation time divided by 10 at least
- Automation slightly improves the level of confidence

Technology is mature
- Several R&D projects to assess and improve tools and methods
- Daily production on worldwide applications not restricted to B applications
  - Proprietary tools
  - Atelier B 4.1 integrates data validation projects

Certification environment to come

Checker 1
Data generated
Compliance
Counter-examples

Checker 2

Data
Formal properties

B models

Data

Checker 1

Checker 2

Pro B
Thank you for your attention

Lilian Burdy
Thierry Lecomte (speaker)
Michael Leuschel