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

	A	B	C	D	E	F	G	H	I
1	Name	ID	IP	Type	UpLink	DownLink	Length	GPS 1	GPS 2
2	Route_tx_001	243		R	Route_tx_005	Route_vx_002	345		
3	Route_vx_002	128		R	Route_vx_002	EndLine_000	128		
4	Switch_w_003	256	192.16.4.55	S	Route_vx_128	Route_tx_006	23		
5	Relay_s_004	12	192.16.4.10	Y				N 50.85 963	O 6.84 201
6	Route_tx_005	3		R	Route_tx_006	Route_vx_128	291		
7	Relay_s_001	55	192.16.4.125	Y					
8	Route_tx_006	22		R	EndLine_001	Route_vx_002	110		
9	Route_vx_128	127		R	Route_tx_006	Route_vx_002	145		
10	Switch_w_009	242	192.16.4.10	S	Route_vx_128	Route_tx_005	34		
11	EndLine_000	0		E		Route_vx_002	1		
12	EndLine_001	1		E	Route_vx_002		1		
13	Signal_xs_002	32	192.16.4.12	G	Route_vx_128		22		
14	Signal_xs_003	33	192.16.4.13	G	Route_tx_006		51		
15	Balise_b_001	301		B	Route_vx_128		0	N 50.85 933	O 6.84 508
16	Balise_b_002	302		B	Route_tx_005		0	N 50.86 123	O 6.84 550

Expressed using B mathematical language

Model-checker (no human in the loop)

Are they

- Consistent ?
- Correct ?
- Safe ?

Data generation ≡

Data validation of partly instantiated large data sets

	A	B	C	D	E	F	G	H	I
1	Name	ID	IP	Type	UpLink	DownLink	Length	GPS 1	GPS 2
2	Route_tx_001	243		R	Route_tx_005	Route_vx_002	345		
3	Route_vx_002	128		R	Route_vx_002	EndLine_000	128		
4	Switch_w_003	256	192.16.4.55	S	Route_vx_128	Route_tx_006	23		
5	Relay_s_004	12	192.16.4.10	Y				N 50.85 963	O 6.84 201
6	Route_tx_005								
7	Relay_s_001	55	192.16.4.125	Y					
8	Route_tx_006	22		R	EndLine_001	Route_vx_002	110		
9	Route_vx_128	127		R	Route_tx_006	Route_vx_002	145		
10	Switch_w_009	242	192.16.4.10	S	Route_vx_128	Route_tx_005	34		
11	EndLine_000	0		E		Route_vx_002	1		
12	EndLine_001	1		E	Route_vx_002		1		
13	Signal_xs_002	32	192.16.4.12	G	Route_vx_128		22		
14	Signal_xs_003	33	192.16.4.13	G	Route_tx_006		51		
15	Balise_b_001	301		B	Route_vx_128		0	N 50.85 933	O 6.84 508
16	Balise_b_002	302		B	Route_tx_005		0	N 50.86 123	O 6.84 550

Is it possible to calculate missing data such as they are all

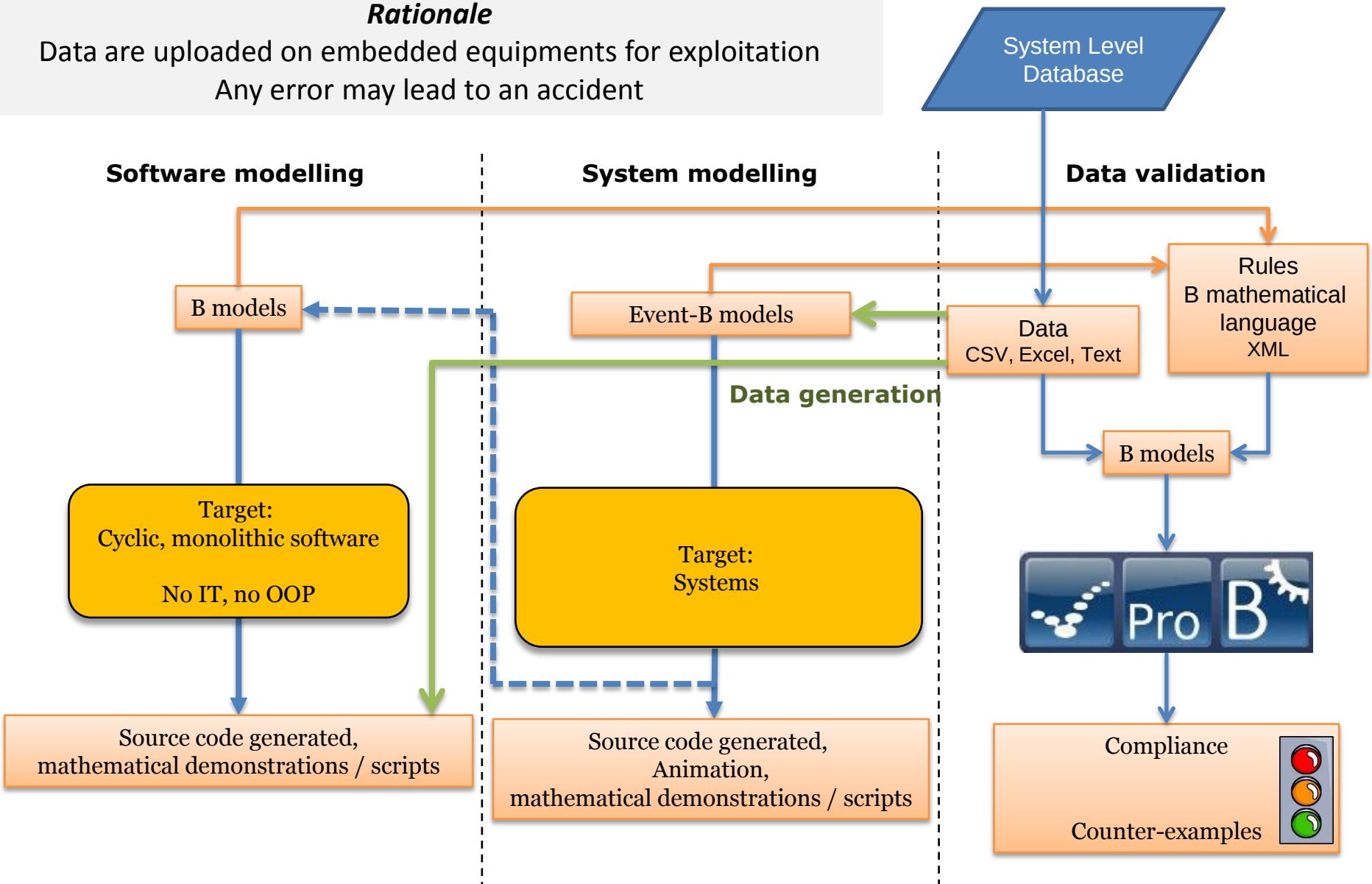
- Consistent ?
- Correct ?
- Safe ?

Application to railways

Rationale

Data are uploaded on embedded equipments for exploitation

Any error may lead to an accident

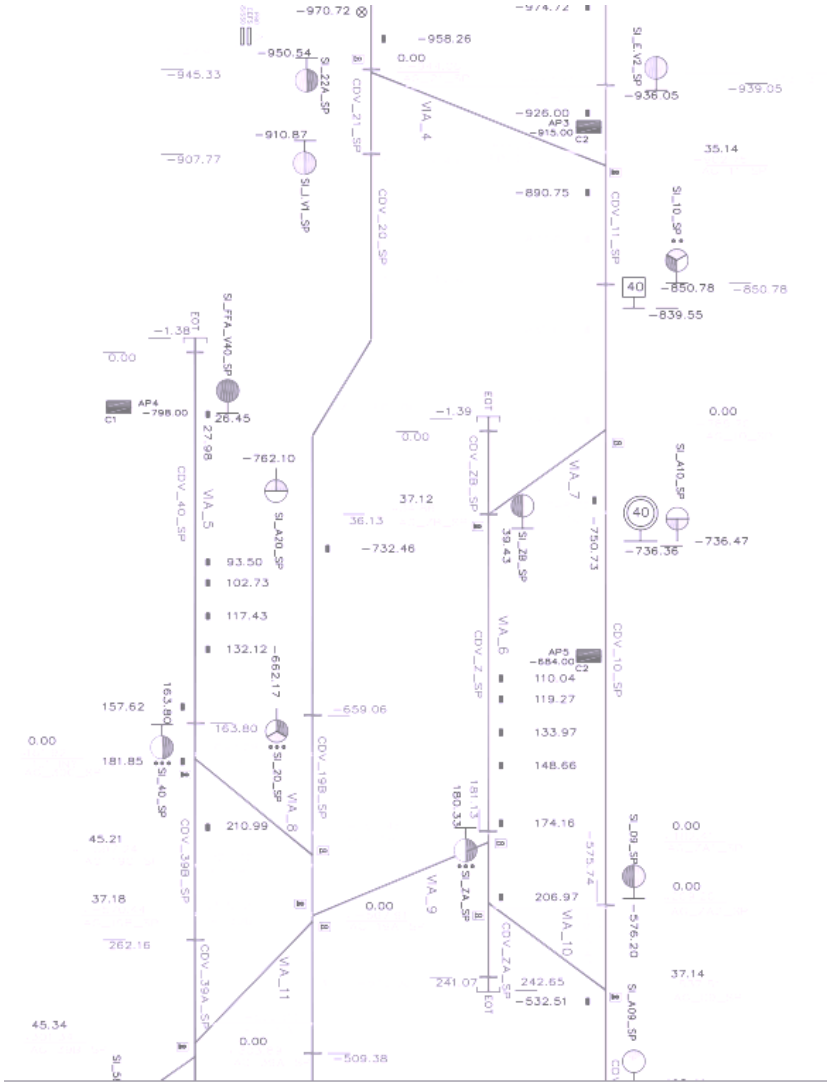


Application to railways

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

	Name	ID
2	Route_tx_001	243
3	Route_vx_002	128
4	Switch_w_003	256
5	Relay_s_004	12
6	Route_tx_005	3
7	Relay_s_001	55
8	Route_tx_006	22
9	Route_vx_128	127
10	Switch_w_009	242
11	EndLine_000	0
12	EndLine_001	1
13	Signal_xs_002	32
14	Signal_xs_003	33
15	Balise_b_001	301
16	Balise_b_002	302

Name = {0|->Route_tx_001, 1 |->Route_vx_002, 2 |-> Switch_w_003, ...}

ID = {0 |->243, 1|->128, 2|->256, ...}

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

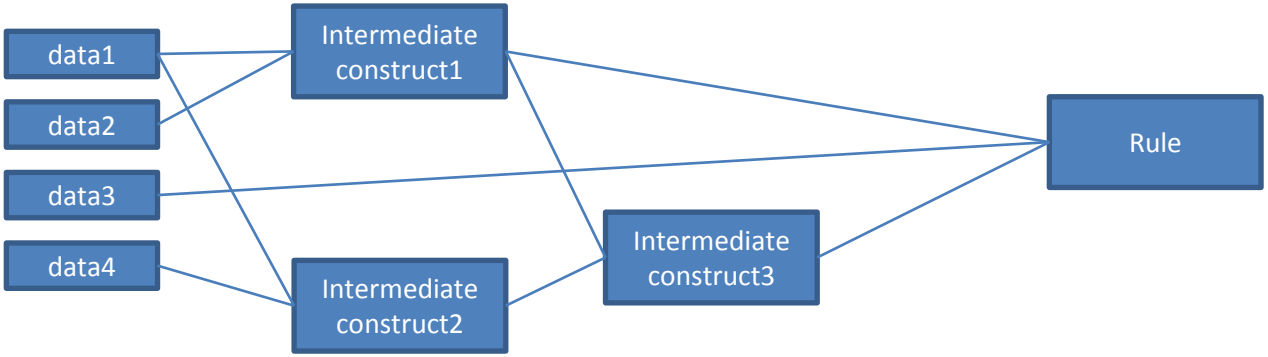
- Supported types: BOOL, INT , STRING, seq(INT), seq(STRING).

Application to railways

Properties ≡

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
```

Sheet name

Data name

Rule name

Values to search for

Conditions to fulfill

If not fulfilled, counterexample is found and error message is displayed

execution

RULE NAME	STATUS	COUNTEREXAMPLES
Rule_DB_General	KO	2

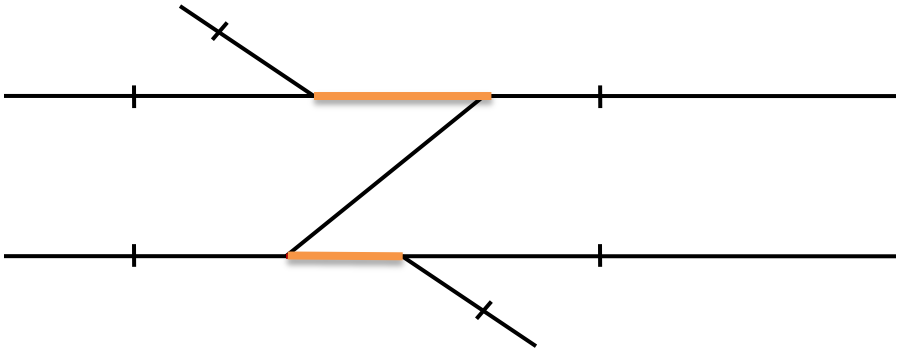
COUNTEREXAMPLE_0
the name ZC_A is the name of an equipment of type ZC but is not in table ZC

COUNTEREXAMPLE_1
the name ZC_AB is in table ZC but is not the name of an equipment of type ZC

A rule can be made of several sequential searches for counterexamples

Application to railways

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: dom(Secondary_Detection_Devices_Cap!ID) +-> (dom(Points_Cap!Name) * dom(Points_Cap!Name) +-> dom(Tracks_Cap!Name)*(INT*INT))

SDD_Point_Normal_Normal  =
%sdd.(sdd : dom(Secondary_Detection_Devices_Cap!ID) & size(Secondary_Detection_Devices_Cap!Point_ID_List(sdd)) >= 2
|
%(point1,point2).(point1 : SDD_Point_list(sdd)
      & point2 : SDD_Point_list(sdd)
      & point1 /= point2
      & Points_Cap!Track_ID(point1) = Points_Cap!Track_ID(point2)
      & bool(TrackKpBegin(Tracks_Cap!Name~(Points_Cap!Track_ID(point1))) <= TrackKpEnd(Tracks_Cap!Name~(Points_Cap!Track_ID(point1))))
       = bool(PointKpToe(point1) <= PointKpToe(point2))
      & !point3.(point3 : SDD_Point_list(sdd) - {point1,point2}
                  & Points_Cap!Track_ID(point3) = Points_Cap!Track_ID(point1)
                   =>  PointKpToe(point3) /: min({PointKpToe(point1),PointKpToe(point2)}) .. max({PointKpToe(point1),PointKpToe(point2)}))
 | Tracks_Cap!Name~(Points_Cap!Track_ID(point2))
  |-> (min({PointKpToe(point1),PointKpToe(point2)})
       |-> max({PointKpToe(point1),PointKpToe(point2)}))))
```

Application to railways

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

VS

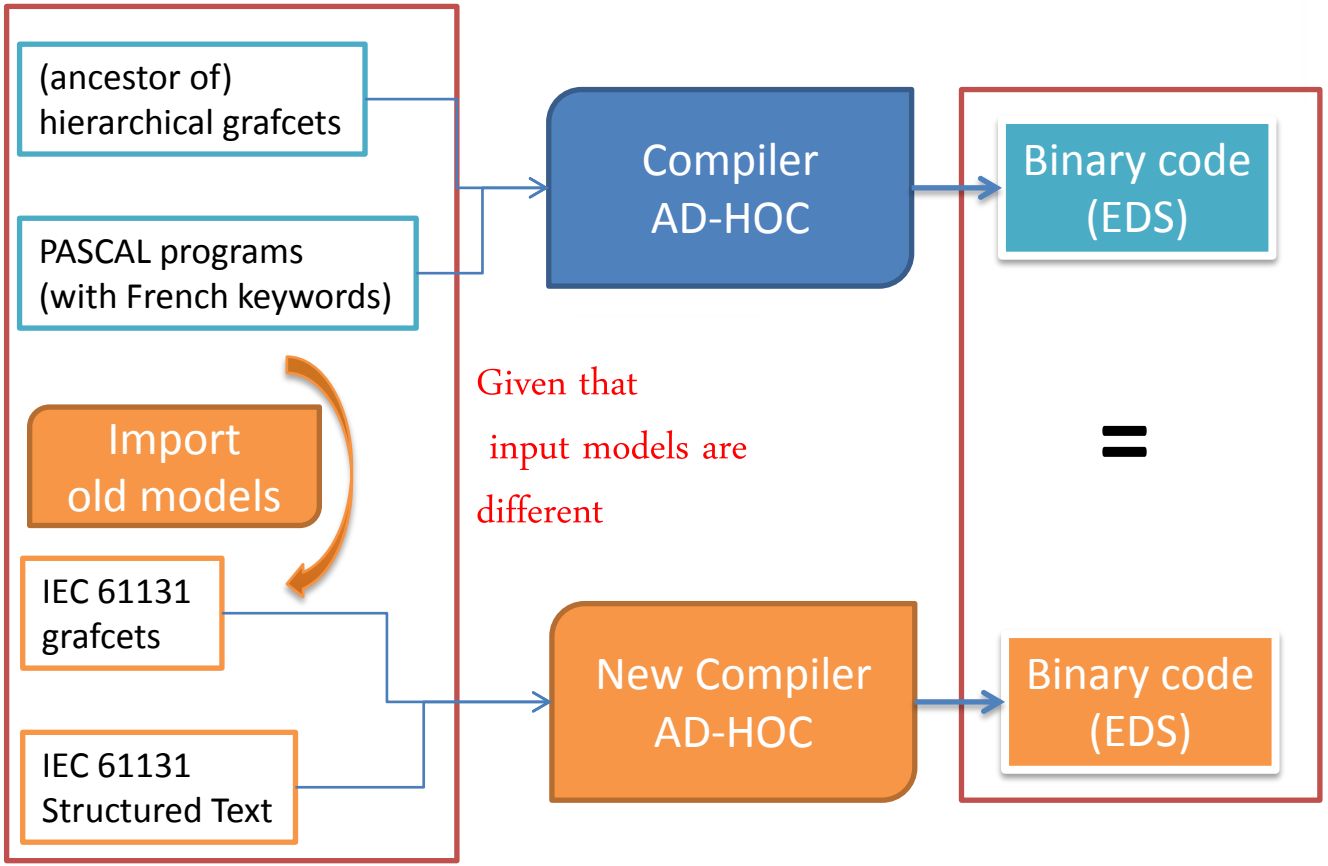
Data validation process
few hours to verify 300 rules

Application to 20 projects for each major release

Compiler Certification

[Contribution to]

Modernization of High Speed Train Embedded Diagnosis System (EDS)



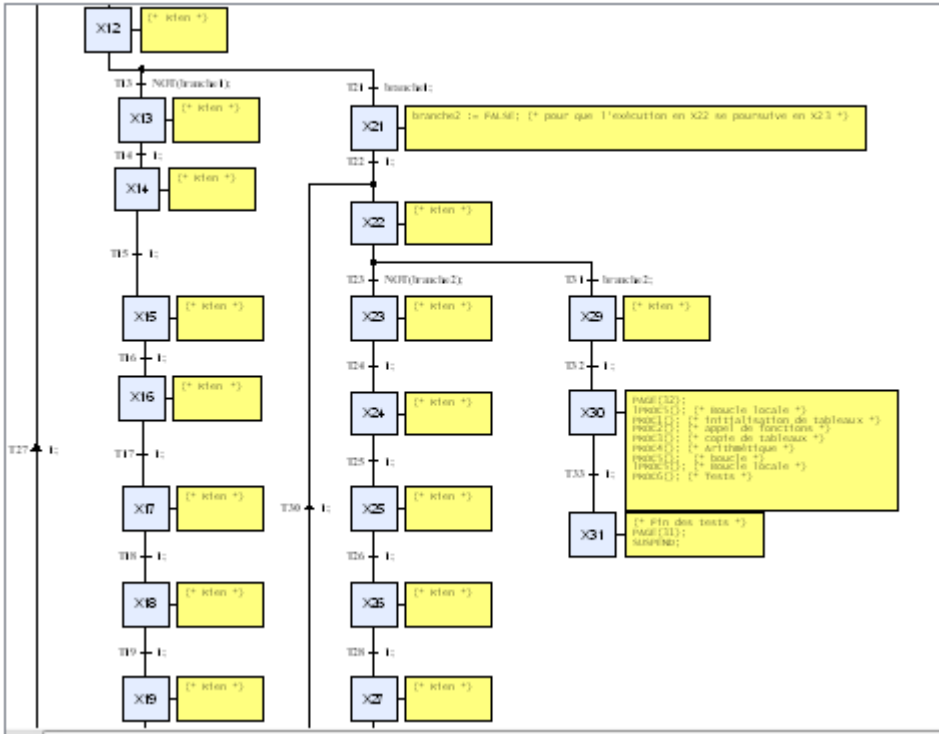
Given that input models are different

Such as generated binaries are identical

Compiler Certification

[Contribution to]

IEC 61131 -3
grafcets



IEC 61131 -3
Structured Text

```
FUNCTION FI001
    gI1 := gcI1;
    gtI1[10] := gI1;
    gtI1[1..3] := [ 3 (gcI1)];
    gtI1[1..3] := gtI1[11..13];
    gI1 := (gI1 MOD 42) + 18 - (gI1 * 23) / gI1;
    gI2 := -gI1;
    gI2 := -gI2;
END_FUNCTION
```

Binary code
(EDS)

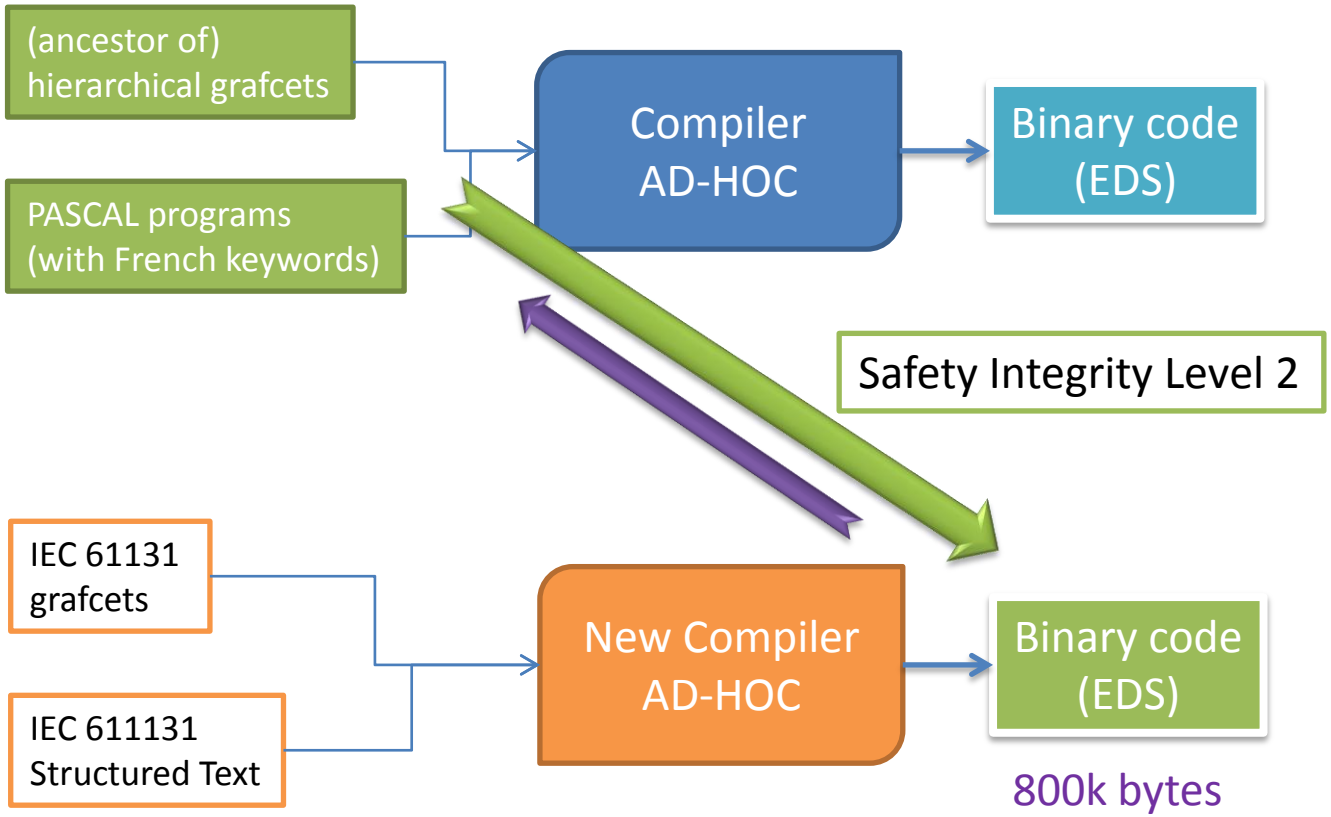
```
[0x001404] ADR_VL_VL_DR TRANSFERT 0x00d4 0x0000 0x82be
[0x00140c] ADR_DR_VL_DR TRANSFERT 0x82be 0x0000 0x8306
[0x001414] ADR_VL_VL_DR RMP_TABLE 0x0003 0x00d4 0x82f4
[0x00141c] ADR_VL_VL_DR TFR_TABLE 0x0003 0x8308 0x4004
[0x001424] ADR_DR_VL_DR MODULO 0x82be 0x002a 0x82ae
[0x00142c] ADR_DR_VL_DR ADDITION 0x82ae 0x0012 0x82ae
[0x001434] ADR_DR_VL_DR MULTIPLIE 0x82be 0x0017 0x82b0
[0x00143c] ADR_DR_DR_DR DIVISE 0x82b0 0x82be 0x82b0
[0x001444] ADR_DR_DR_DR SOUSTRAC 0x82ae 0x82b0 0x82be
[0x00144c] ADR_DR_VL_DR SOUSTRAC 0x82be 0x0000 0x82c0
[0x001454] ADR_DR_VL_DR SOUSTRAC 0x82c0 0x0000 0x82c0
[0x00145c] RETOUR 0x00
```

Compiler Certification

[Contribution to]

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

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

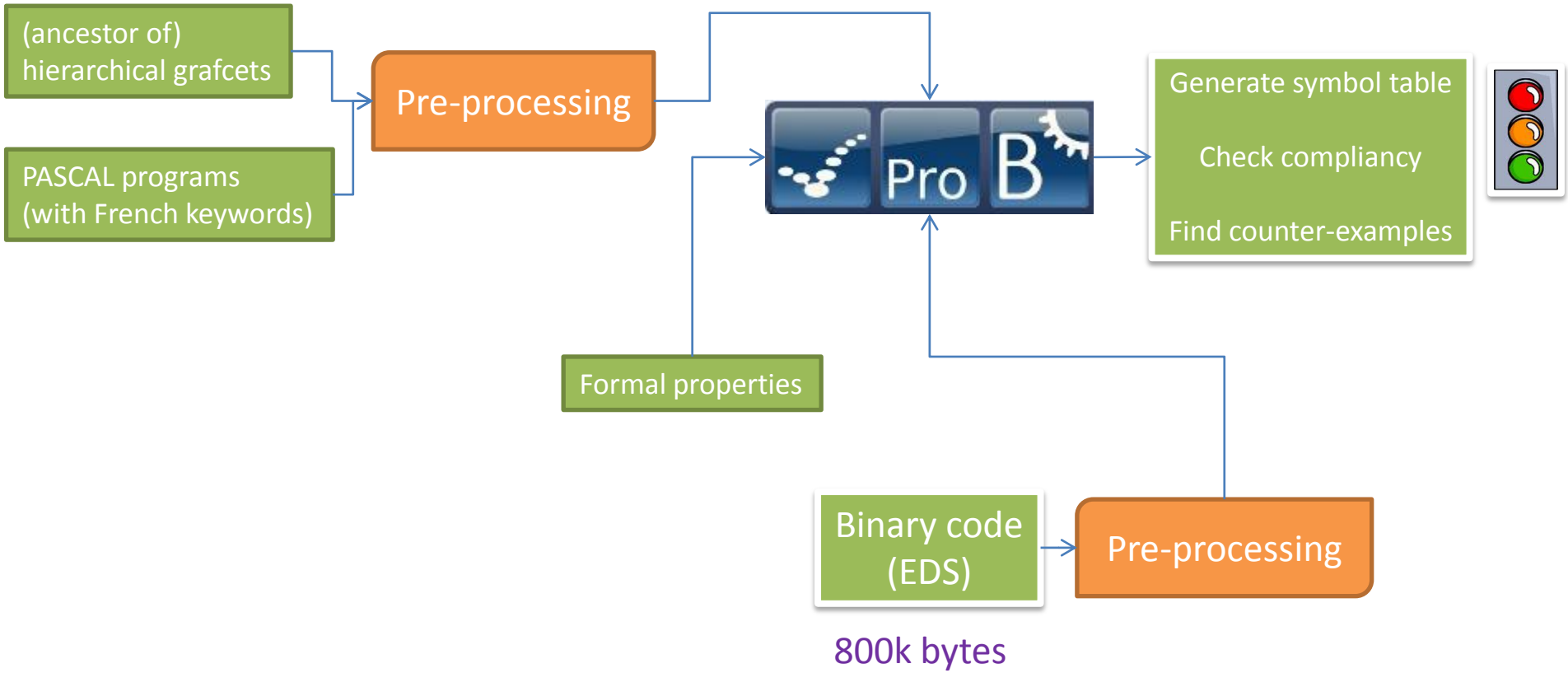


Compiler Certification

[Contribution to]

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

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



Compiler Certification

[Contribution to]

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

Formal properties

Compiler Certification

[Contribution to]

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 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

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

Compiler Certification

[Contribution to]

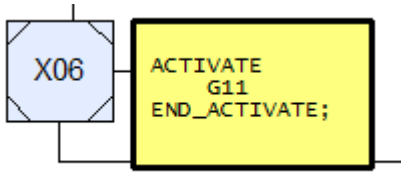
- P03 Sub-grafcets called in the binary file should comply with sub-grafcets activated in input models

Compiler Certification

[Contribution to]

- P03 Sub-grafcets called in the binary file should comply with sub-grafcets activated in input models

List grafcet activations (old models)



Build B model of activations

G7 = {main, G1, G2, G3, G4, …. }
next: G7 <-> G7
next = { …, G7 |-> G11, …}

List grafcet activations (binary)

```
[0x00198c] LANCE_GRAF 0x15
```

Build B model of activations

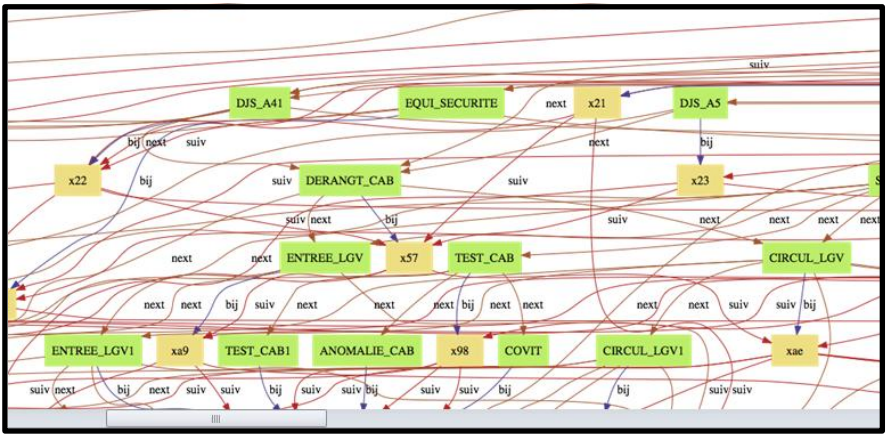
ADR = {0x01, 0x13, 0x15, …}
suiv: ADR <-> ADR
suiv = { … , 0x10 |-> 0x15, …}

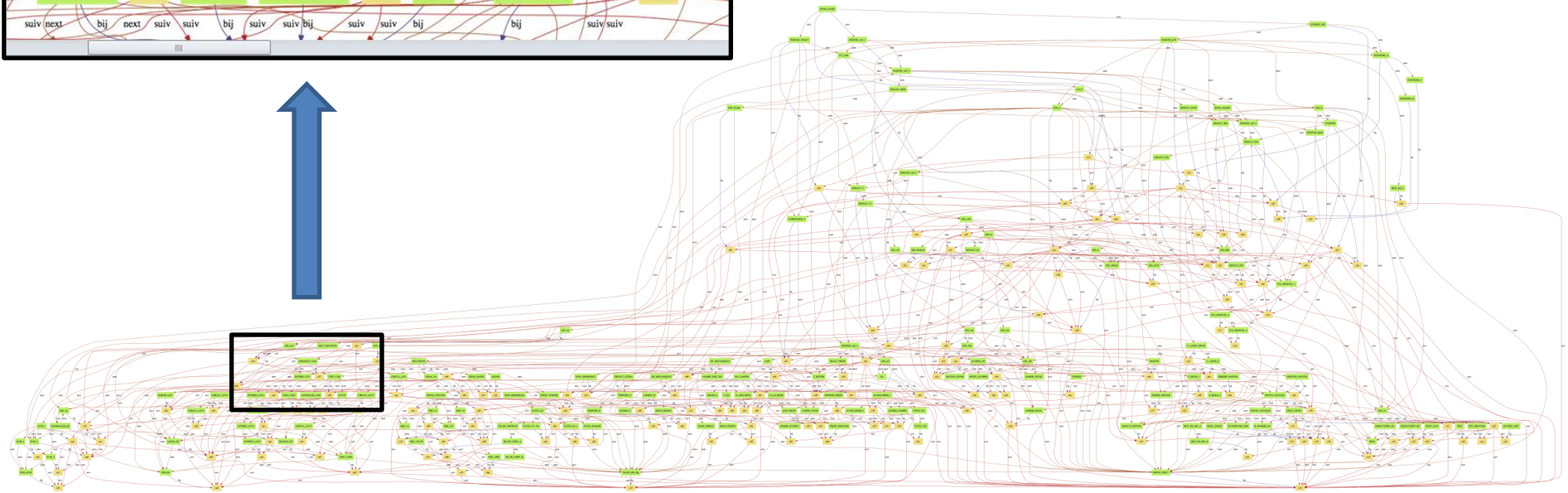
there exists a bijection bij that associates to a node of G7 a node of ADR such as children of both nodes match

bij: G7 >->> ADR &!xx.(xx: G7 => bij[next[{xx}]] = suiv[bij[{xx}]])

Compiler Certification

[Contribution to]





162! =
1229694218739449434110178928491750176572300599427169306620762521167814540117728965860988098467051531783599507442
9904709708273401807824365415928975695099566042246320538220924308010459938381430588227927174194100982189204709615
2931983263907734109259038720000000000000000000000000000000000000000000

Compiler Certification

[Contribution to]

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)

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

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

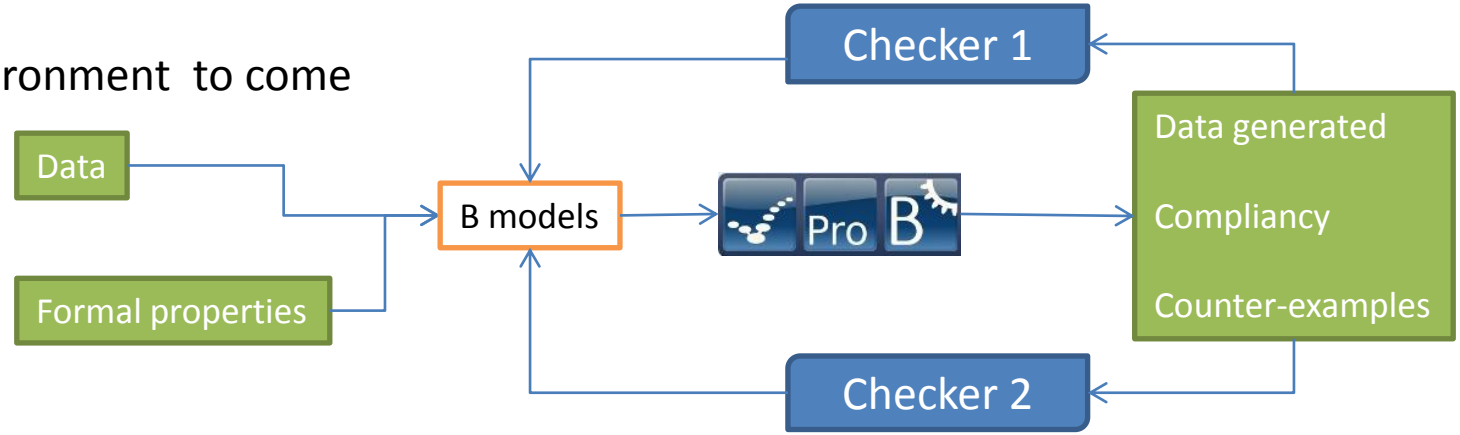
Data validation & data generation able to deal with industrial problems

- 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



Thank you for your attention

Lilian Burdy
Thierry Lecomte (speaker)

Michael Leuschel