

Testsuite

Still work in progress.

Latest work just testsuite cmd line.

Overview:

1. Start with overview of testing
2. Describe current possibilities in TestSuite
3. Discussion on where to apply and what to add

– Please interrupt with questions/remarks

Testing: when

1. Testing first design, using specs/ spec-based checks
2. New version/next wave questionnaire: compare with previous version(s)
3. Testing of postprocessing: dummy data to test further analysis
4. After the data has been collected

Testing when: initial design

Is the questionnaire according to specs ?

How are the specs defined: Word document, or more formal?

1. Are the question texts correct ?
2. Are all questions asked to the right respondent, e.g. not asking questions about driver's license to a 12-year-old
3. Under what conditions is a block asked?
4. IF $X = 1$ THEN, where X has not been asked yet?
5. ...other ?

Testing when: next version/wave ?

Two types of comparison:

1. Source code/Datamodel-based \Rightarrow WinDiff / Delta
2. What are the runtime effects of these changes? \Rightarrow TestSuite regression testing

Possible errors due to changes:

- a. - enum value: cannot enter IF testing against numeric value
 $X:(a,b,c) \rightarrow X:(a,b)$, IF $X = 3$ THEN
- b. + enum value: is there code that uses that new value?
- c. Effect on route, for example when IF condition changed
- d. ...

Testing when: post processing

- For a new questionnaire, postprocessing could be based on random records instead of real data records
- For manipula: given a .msux file, generate all (missing) input files

Testing when: after collection of data

- Questions not asked ? If so, why?
- Is there Blaise code never passed ?
- Time per question or block? Extremes?
- Given the data record, what was the route through the questionnaire ?
- Are the correct questions asked to a subgroup?
IF Age ≤ 30 : questions about current education

Available for testing

1. Inspect datamodel
 - Graph view in metaviewer or testsuite
 - SVG export, annotated, 1 block/file
2. Generate records
 - Sparse: try to pass all statements with as little records as possible; full datamodel or per block.
 - Random: generate N records, fill random answers, but within datamodel constraints
3. Comparison of 2 questionnaires
 - Windiff
 - Delta, maybe later also in TestSuite
 - Runtime effect, already in TestSuite
4. Jump to “ASK X” in DEP
5. New: Find at least one route / all routes to (ASK) statement X ~ record generator
6. New: Blaise debugger: only in prototype form , for now

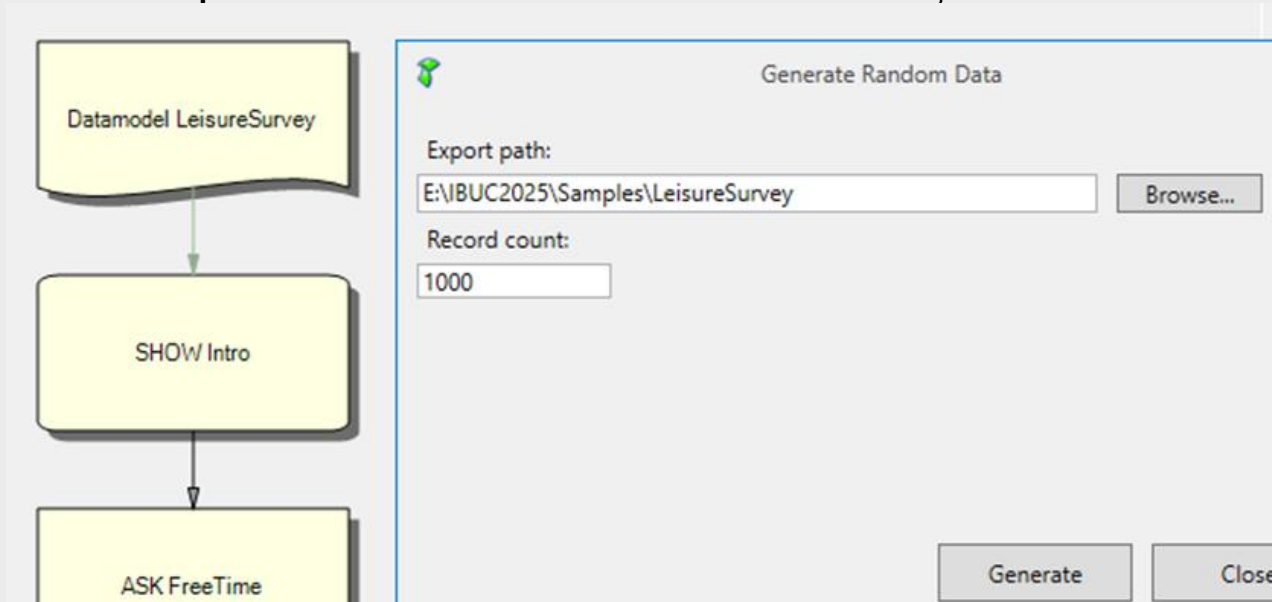
Available through:

1. ProgramAnalysis API

2. TestSuite command line, eg

```
testsuite.exe gen_random -f test.bmix -n 100
```

3. A few options are available in TestSuite UI, for now



1. Inspect datamodel: SVG export

TestSuite

Using cmd line (for now): export datamodel to SVG file

- One file with complete model
- Master model ending with blocks + files per block

2. Generate test data

- **Sparse**
 - Just enough records, aiming to achieve full coverage
 - Slow
- **Random**: generate N randomly filled data records
 - Walking through questionnaire using DataRecord.FirstField / NextField
 - Fast(er)

Both methods take into account:

- field values constraints and
- RULES constraints

Result:

- bdbx/bdix file
- .xml file, containing field values, routes followed and possible errors

2. CmdLine: generate records: sparse

Goal: minimal # records, but still full coverage

- Can detect possible errors for the current route, for example
 1. Using not-on-route fields inside question-text or condition
 2. Missing tests voor special answers ?
 3. Comparison of $X:1\dots3$ with the value 4
 4. IF $X = \text{empty}$, BUT X asked and empty is not allowed
 5. Numerical ($X:=Y/0$) or index ($X:=Y[-1]$) errors
- `--debug` : shows each generated route as an SVG
- Possible to include layout: ID in web page or look at layout expressions

Demo: GenerateSparseRecords

2. CmdLine: generate records: random

Generate a random value for each ASK encountered while walking through questionnaire using DataRecord.FirstField / NextField

- Also fills KEEP's
- Tries to overcome unresolved CHECK's,
- --debug: decision table + freqs per IF statement

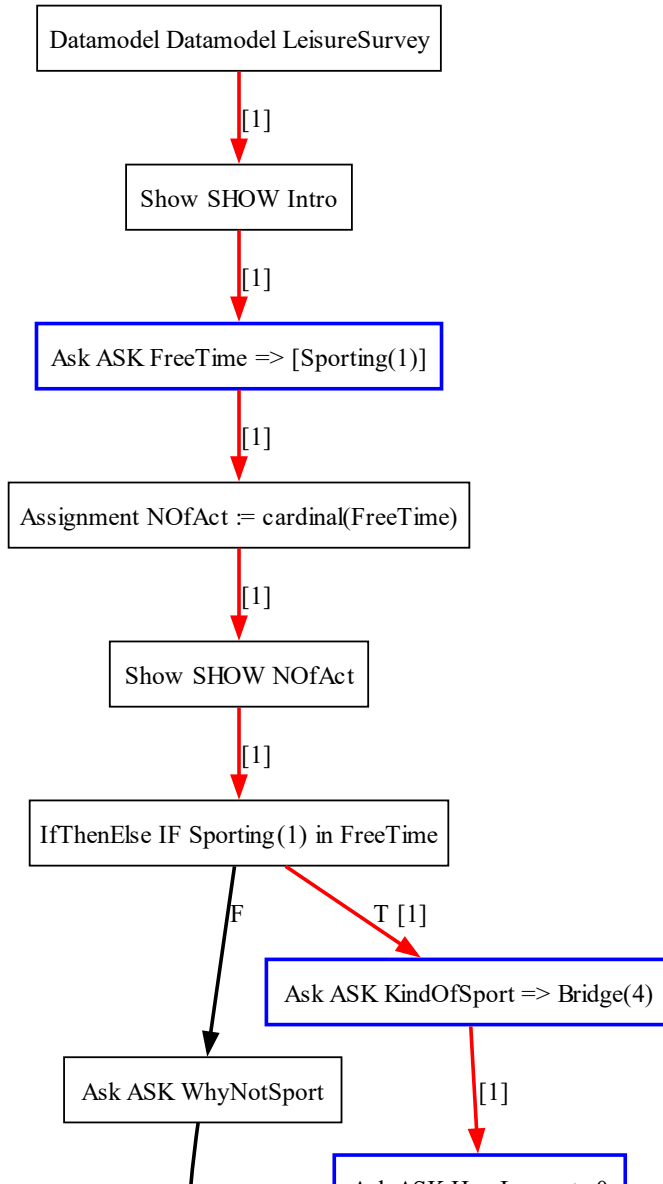
```
[11] IF x1(1) in Skema.BlokM.BSSC_W3  
    ==>
```

BSSC_W3	Result	Count
[2,6]	False	6
[1,5]	True	5

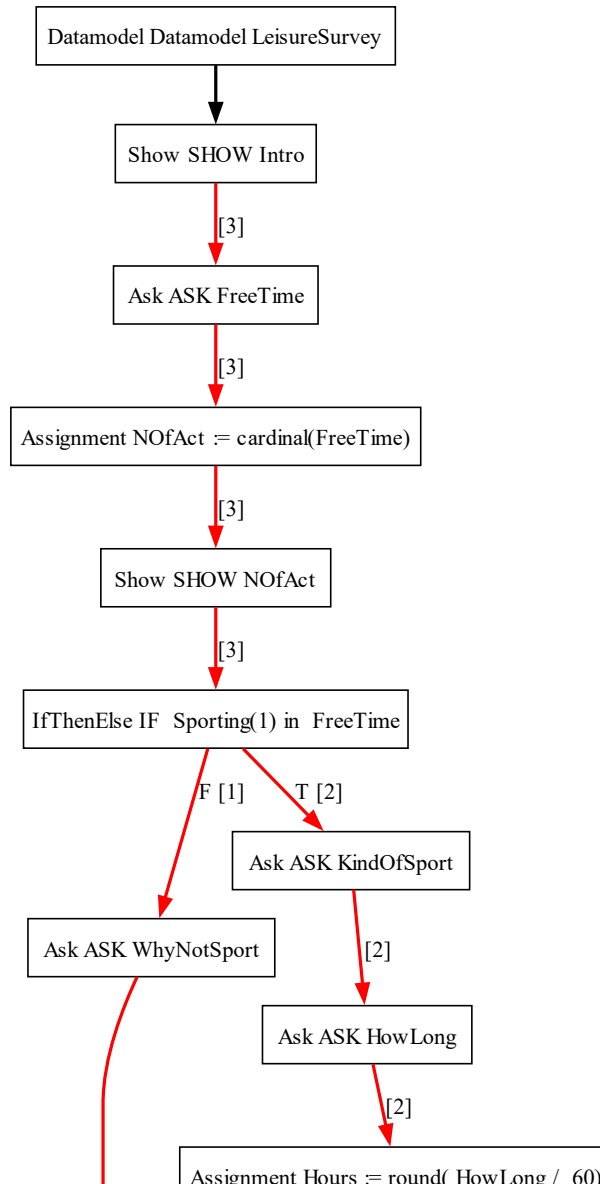
- `gen_random_direct` for datamodels with just fields and no rules: much faster

Demo : GenerateRandomRecord

2. Generate records: results



2. Generate records: results



2. Using generated test records

1. (Sparse): test for -possible- errors , on the route selected,
2. Regression testing: already in TestSuite.
3. Jump to ASK statement X; especially for retesting if an error was found on position X
4. Testing code that postprocesses actual surveydata, eg manipula
5. Debugger: when setting breakpoints, these statements must be on the route

3. Compare 2 datamodels

- **WinDiff** : comparison of two datamodel texts
→ possible, but difficult, for example in case of moved block
- **Delta**: better: comparison of two datamodel trees
→ can detect a small change, but is the effect?
- **TestSuite**: Comparison of actual route paths through the two datamodels
→ not so much the changes, but their effect on running the DEP; can detect changes if the datamodel does not change, but the blaise version or the run-time settings do.
Differences in field values: not yet.

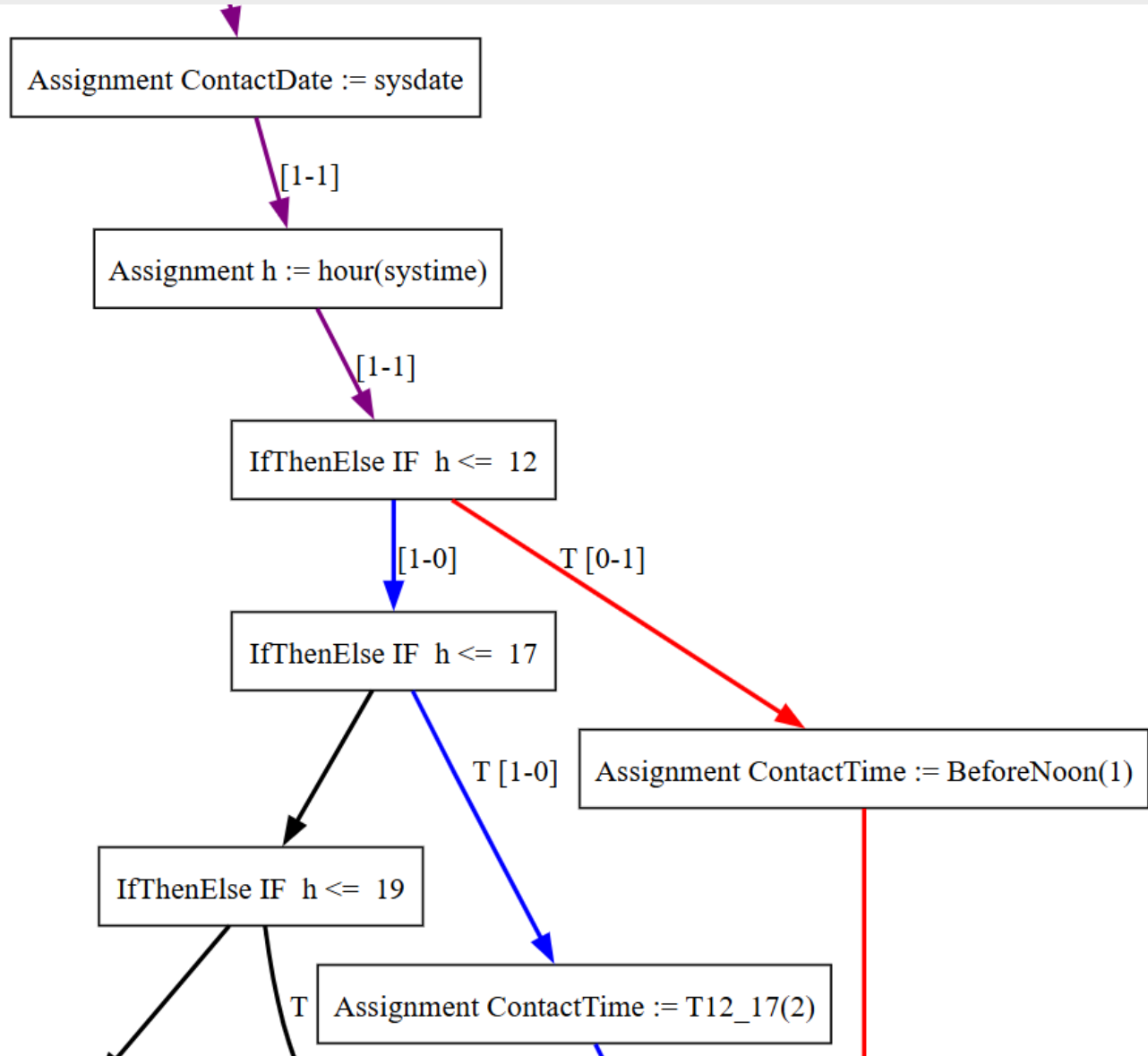
Demo: Delta

3. Compare 2 datamodels: TestSuite

1. Calculate sparse records for datamodel 1
→ route path + values for fields
2. For each record:
 1. Walk through datamodel 2 using `DataRecord.FirstField / NextField`
 2. At each ask: insert value from generated record
3. This may lead to :
 - Different route
 - Different field values (not in UI yet)

[Demo: TestSuite compare]

3. Compare 2 datamodels: using SVG



4. Jump to field X in the DEP

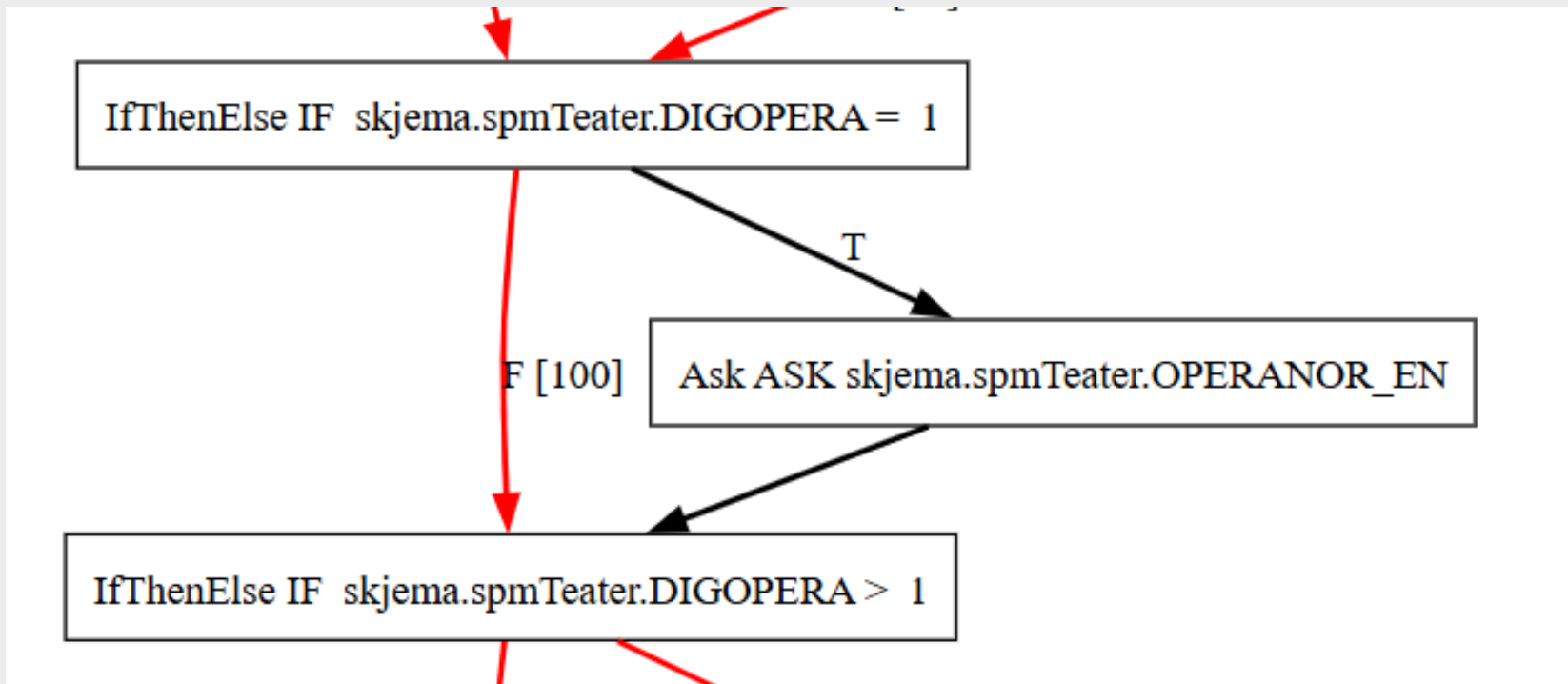
1. User selects field X
2. Select a generated record that contains the **ASK X** statement
3. Construct DEP command line; later also for web.
 - KeyValue:1234 (=Selected record)
 - Fields:a=22563,showtext='????'
 - RunMode:ThickClient
 - GotoField:X

[Demo]

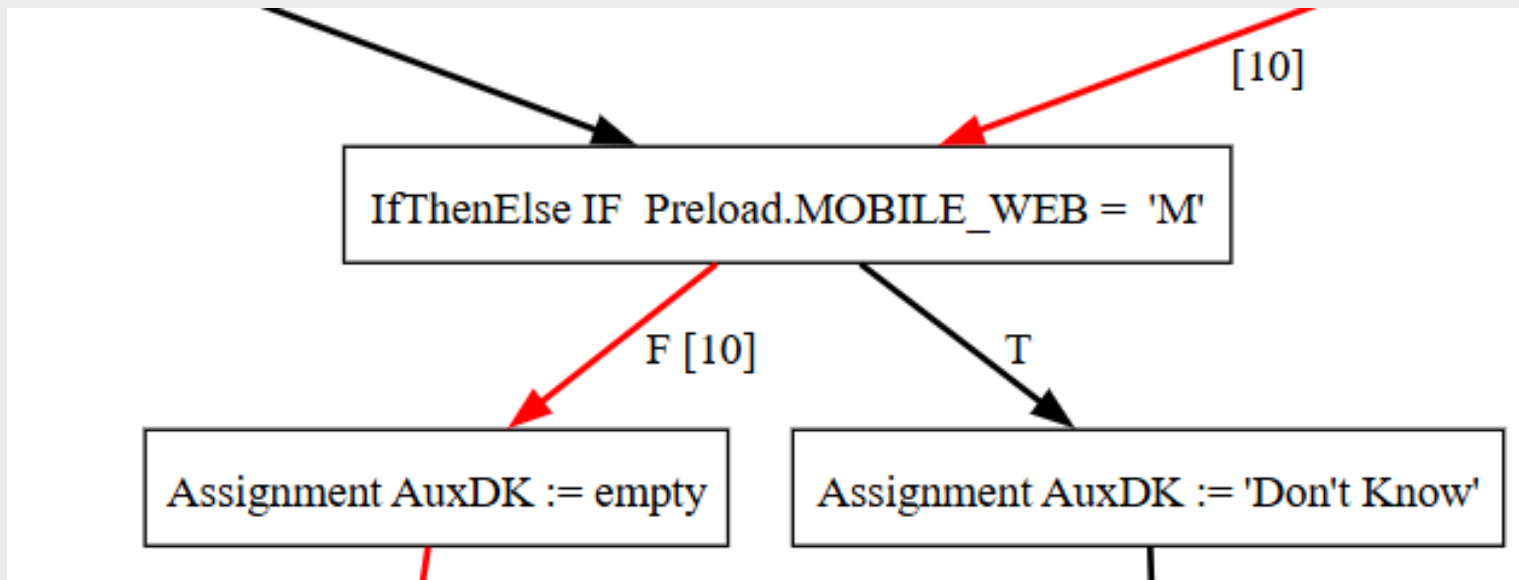
6. Why random may fail

INTEGER field

DIGOPERA:1..999

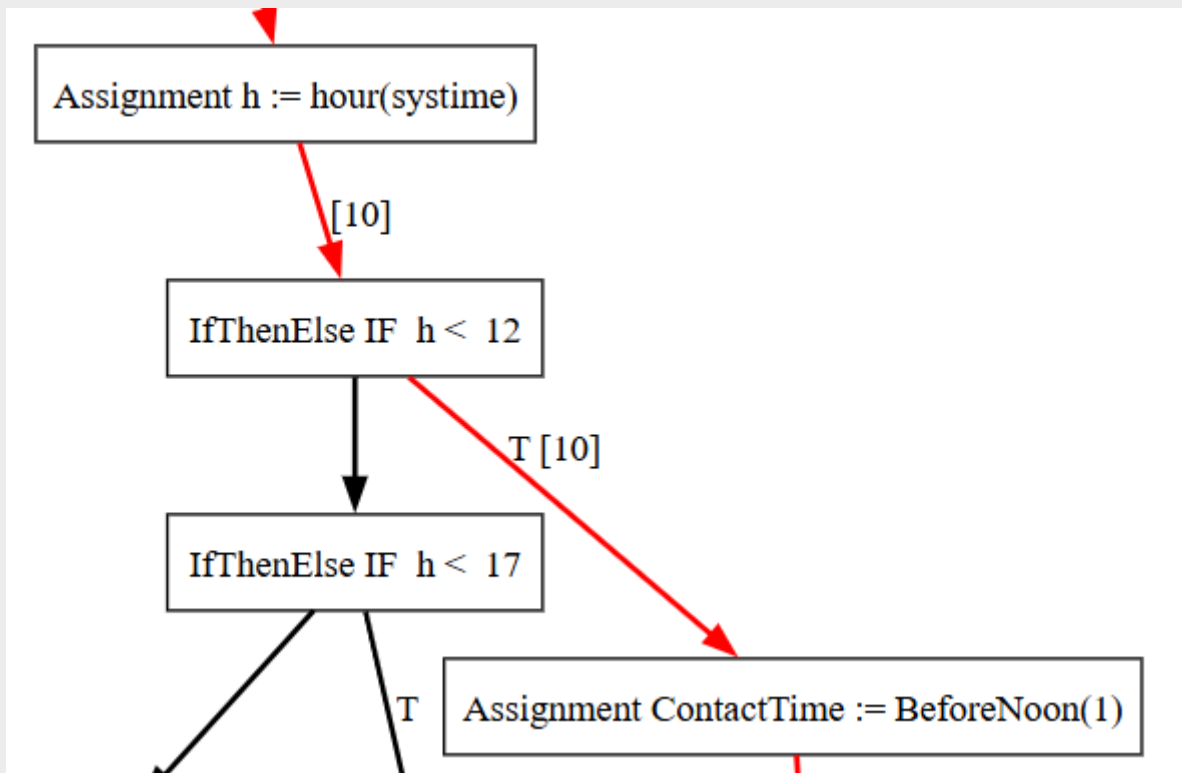


6. Why random may fail



6. Why random may fail

Time/Date dependence



2. Why sparse may fail

- Using #unvisited statements not always sufficient to guide the right path
- Few records:
 - IF X = enum1 THEN
 - ELSE IF X = enum2 THEN
 - Etc.
 - ==> Not enough passes, to cover all split-ups.
- Conditions to be evaluated during the algorithm become too complex

5. One/All routepaths to 1 statement

Find a route path to statement S

Driver: find a path to statements not covered in sparse/random generator

Main idea: find all relevant paths to S, for example $S = A4.ASK$

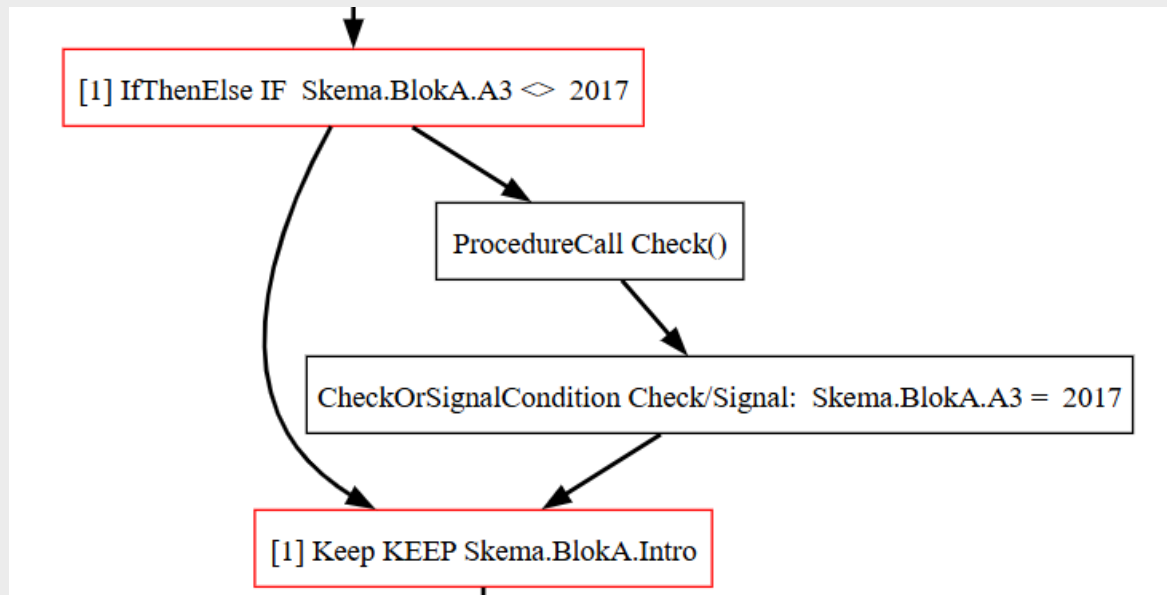
```
A3.ASK
...
if A3 <> 2017 then
    CHECK
    A3 = 2017 "...."
Endif
A4.ASK
```

Possibilities:

- $N = 0$: no paths found: question can never be asked
- $N \geq 1$:
 - we have at least 1 path that contains the question
 - Perhaps: we have all the paths leading to A4 → condition for asking this question (under investigation)

Demo: FindingRoutePath

Find 1 path



```
skema.bloka.a6 in [x7(7), x8(8)] and  
skema.bloka.a9 in [x1(1), x3(3)] and  
not
```

```
(  
    skema.bloka.a3 <= 2016 or skema.bloka.a3 >= 2018  
)
```

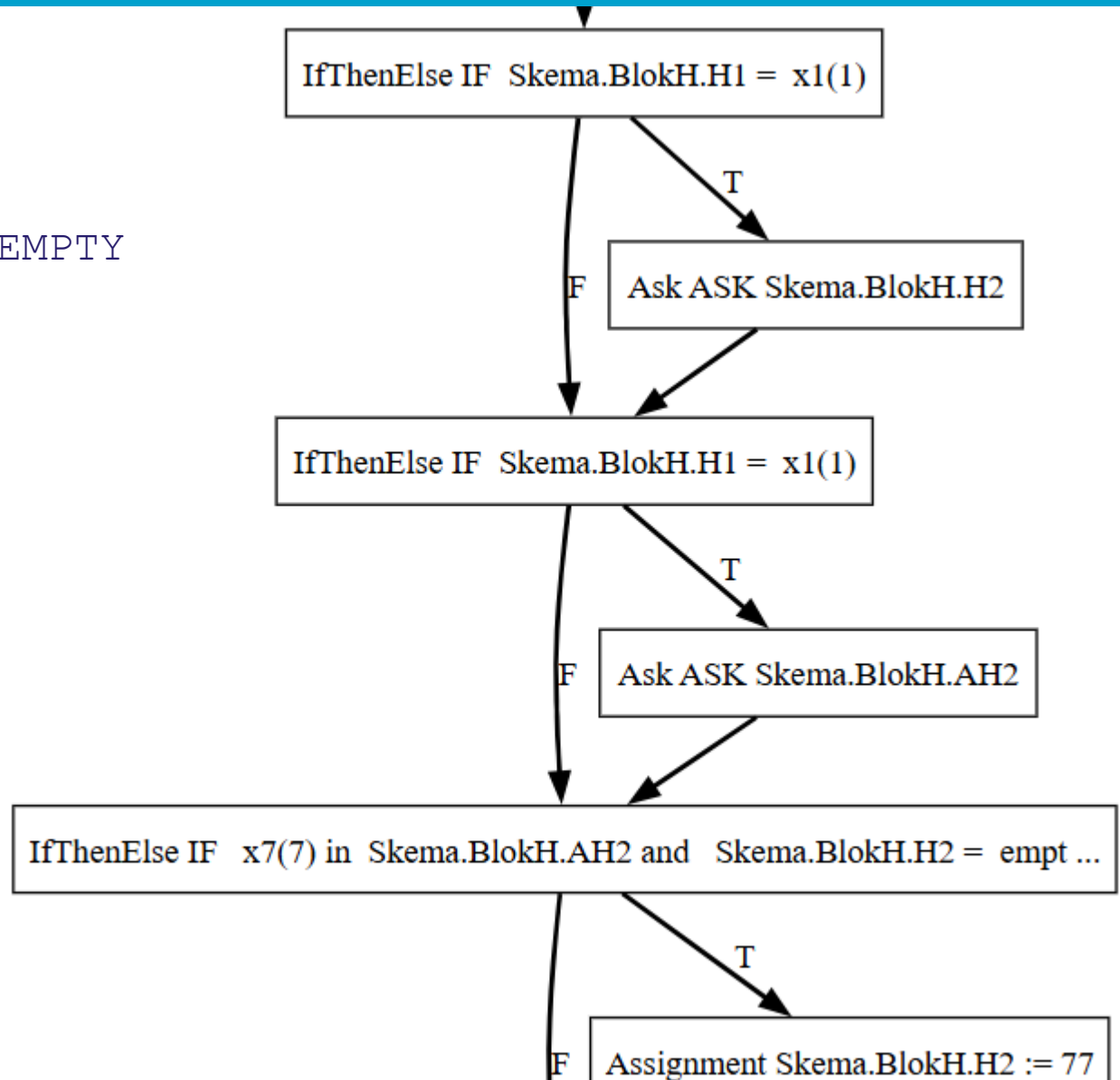
→

	a6		a9		a3	
Path0	x7(7)		x1(1)		2017	

Unreachable

H2: 0..99

AH2: SET OF ..., EMPTY



6. A Blaise debugger

The screenshot displays the Blaise debugger interface for a program named 'Form1'. The interface is divided into several panels:

- Program stack:** Shows the current execution context with 'Stack' expanded to show 'MAIN'.
- Statement List:** A table listing statements with their line numbers and types. A breakpoint is set at line 30.
- Variables Table:** A table showing the current state of variables, including their names, value types, and current values.

Statement List:

Type	Line	Statement
	17	Keep KEEP User
	18	IfThenElse IF WhenMade = empty
	19	IfThenElse THEN
	20	Assignment WhenMade.When := timetostr(system, 'yyyyMMdd,HH:mm:sszz')
	21	Assignment WhenMade.User := User
	22	Assignment ContactDate := sysdate
	23	Assignment h := hour(system)
	24	IfThenElse IF h < 12
	-1	IfThenElse THEN
	-1	Assignment ContactTime := BeforeNoon(1)
	25	IfThenElse ELSE(-IF)
	26	IfThenElse IF h < 17
	-1	IfThenElse THEN
	-1	Assignment ContactTime := T12_17(2)
	27	IfThenElse ELSE(-IF)
	28	IfThenElse IF h < 19
	29	IfThenElse THEN
B	30	Assignment ContactTime := T17_19(3)
	-1	IfThenElse ELSE(-IF)
	-1	IfThenElse IF h < 21
	-1	IfThenElse THEN
	-1	Assignment ContactTime := T19_21(4)
	-1	IfThenElse ELSE
	-1	Assignment ContactTime := Late(5)
	-1	IfThenElse IF AfterInterview = Yes(1)

Variables Table:

Name	Value Type	VarType	Value	Value Status	Value Statement
CaseEndDate_0	Date	Normal	NULL	Asked	
User	String	Normal	EXP: User_0	Assigned_ByAssignment	
User_0	String	Normal	NULL	Asked	
ContactDate	Date	Normal	EXP: sysdate	Assigned_ByAssignment @ As...	15.0.2
h	Integer	Normal	EXP: hour(system)	Assigned_ByAssignment @ As...	15.0.3

Breakpoint was hit: Assignment ContactTime := T17_19(3)

6. A Blaise debugger

The screenshot shows the Blaise debugger interface. On the left, the 'Stack' pane displays the current execution context: IfThenElse THEN, IfThenElse ELSE(-IF), IfThenElse ELSE(-IF), IfThenElse THEN, and MAIN. The main window displays a list of statements with columns for Type, Line, and Statement. Statement 30 is highlighted in yellow and marked with a blue 'B' in the left margin.

Type	Line	Statement
	17	Keep KEEP User
	18	IfThenElse IF WhenMade = empty
	19	IfThenElse THEN
	20	Assignment WhenMade.When := timetostr(system, 'yyyyMMdd,HH:mm:sszz')
	21	Assignment WhenMade.User := User
	22	Assignment ContactDate := sysdate
	23	Assignment h := hour(system)
	24	IfThenElse IF h < 12
-1		IfThenElse THEN
-1		Assignment ContactTime := BeforeNoon(1)
	25	IfThenElse ELSE(-IF)
	26	IfThenElse IF h < 17
-1		IfThenElse THEN
-1		Assignment ContactTime := T12_17(2)
	27	IfThenElse ELSE(-IF)
	28	IfThenElse IF h < 19
	29	IfThenElse THEN
B	30	Assignment ContactTime := T17_19(3)
-1		IfThenElse ELSE(-IF)
-1		IfThenElse IF h < 21
-1		IfThenElse THEN

Variables Current Statement Conditions Solution variables Statement details Claims

Name	Value Type	Var Type	Value	Value Status	Value Sta
CaseEndDate_0	Date	Normal	NULL	Asked	
User	String	Normal	EXP: User_0	Assigned_ByAssignment	
User_0	String	Normal	NULL	Asked	
ContactDate	Date	Normal	EXP: sysdate	Assigned_ByAssignment @ As...	15.0.2
h	Integer	Normal	EXP: hour(system)	Assigned_ByAssignment @ As...	15.0.3

Stack
Breakpc

Combined: TestSuite feature vs Phase

	Initial design	test postprocessing	updated version	post survey analysis
1. View graph	X			
1. Export svg/block	X		X	
2. sparse records	X	X	X	
3. compare/dynamic			X	
3. compare/static			X	
2. random records	X	X		
5. Find path to X	X		X	X
6. Debugger	X		X	X
analyse records				X
show routes				X

Questions + Discussion

Other uses for test-records?

- Defining target populations and see if they get the correct (block)questions?
- Manual testing + administration (like CAI Testing Tool (CTT))
- Random based on actual distributions of Age, Level of Education, ...?
- Additional options to influence the generators?
- ..??

Other functionalities for a test-suite ?

- Comparing data records?
- ..??