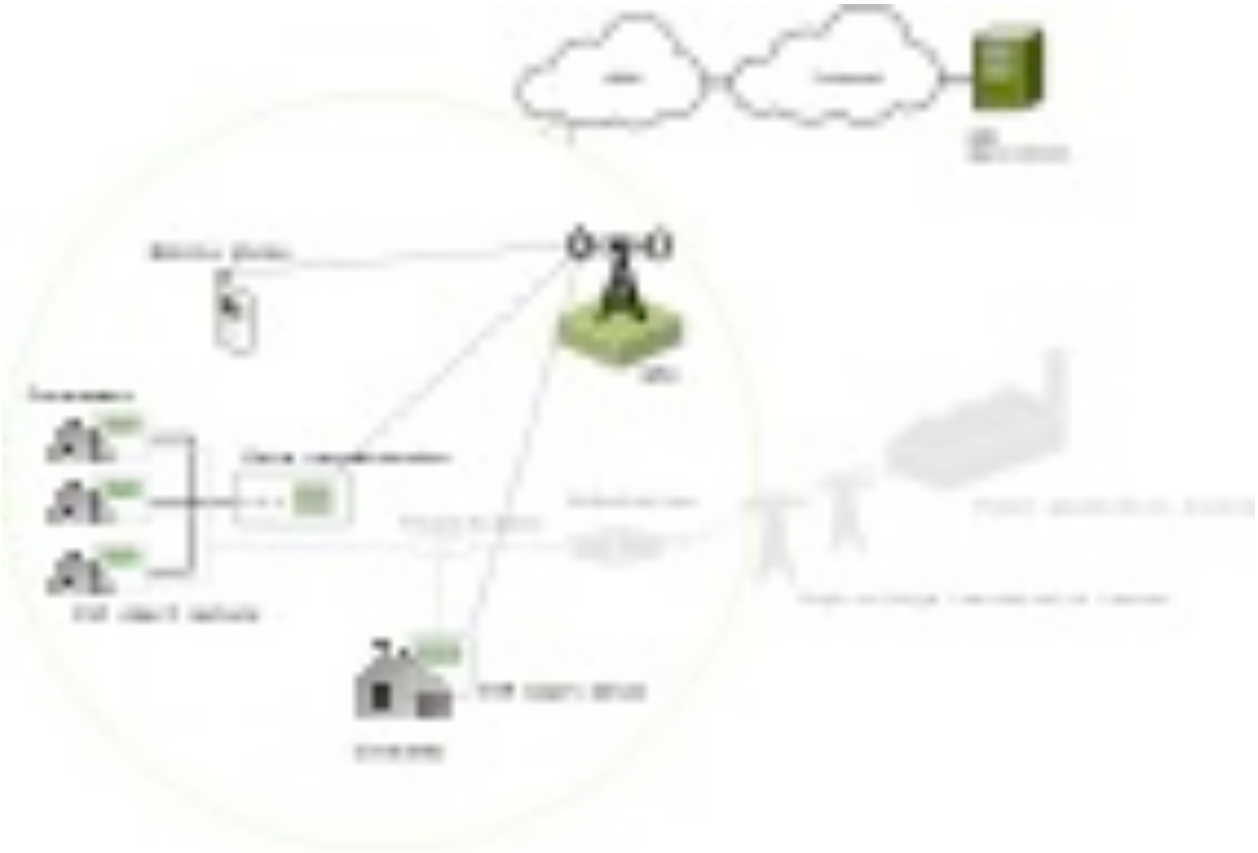
(Re-)configuration of Communication Networks in the Context of M2M Applications

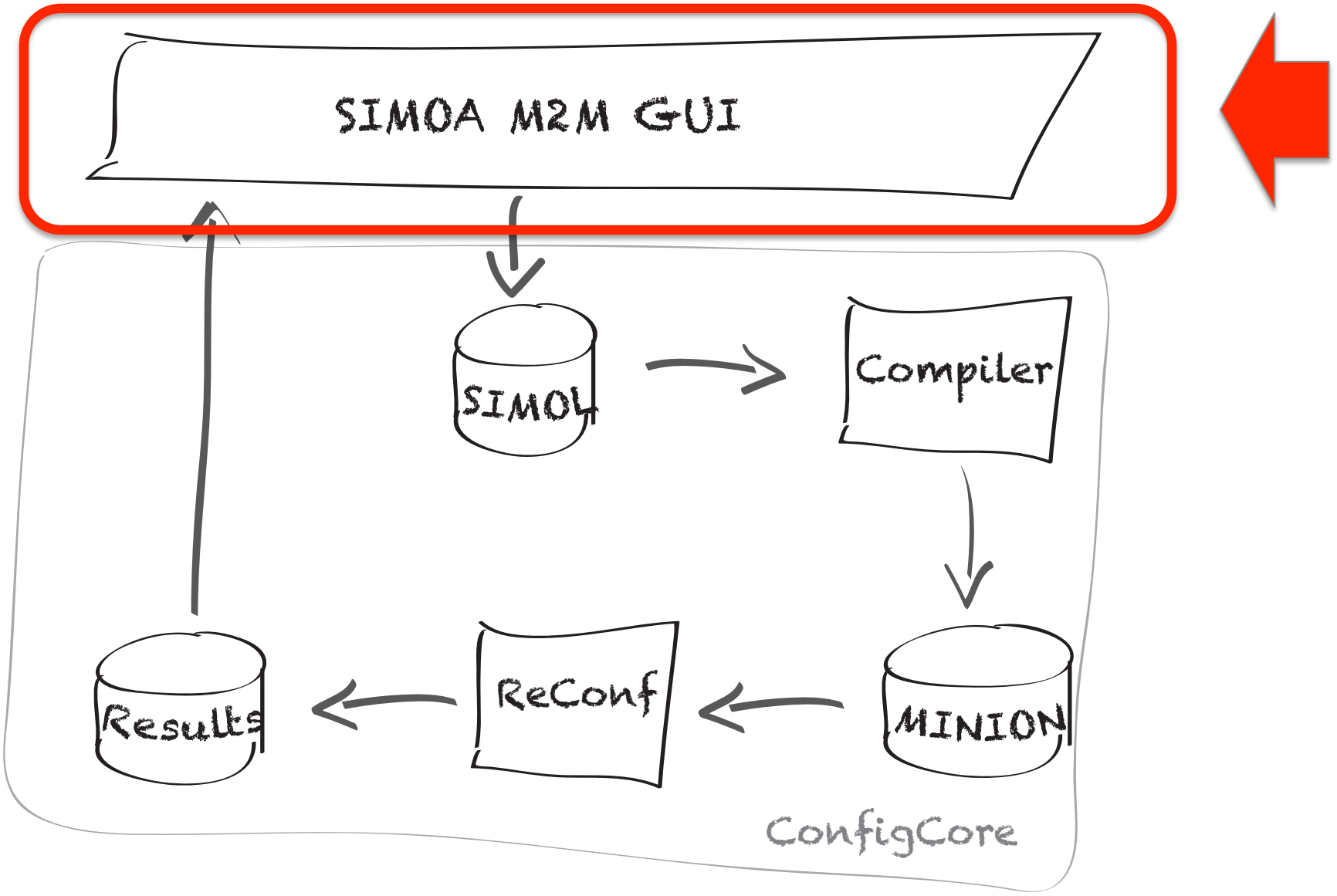
Iulia Nica and **Franz Wotawa**
Technische Universität Graz
Institute for Software Technology
8010 Graz, Inffeldgasse 16b/2, Austria
wotawa@ist.tugraz.at

Research partially funded within the BRIDGE research project Simulation and Configuration of Mobile networks with M2M Applications (SIMOA)

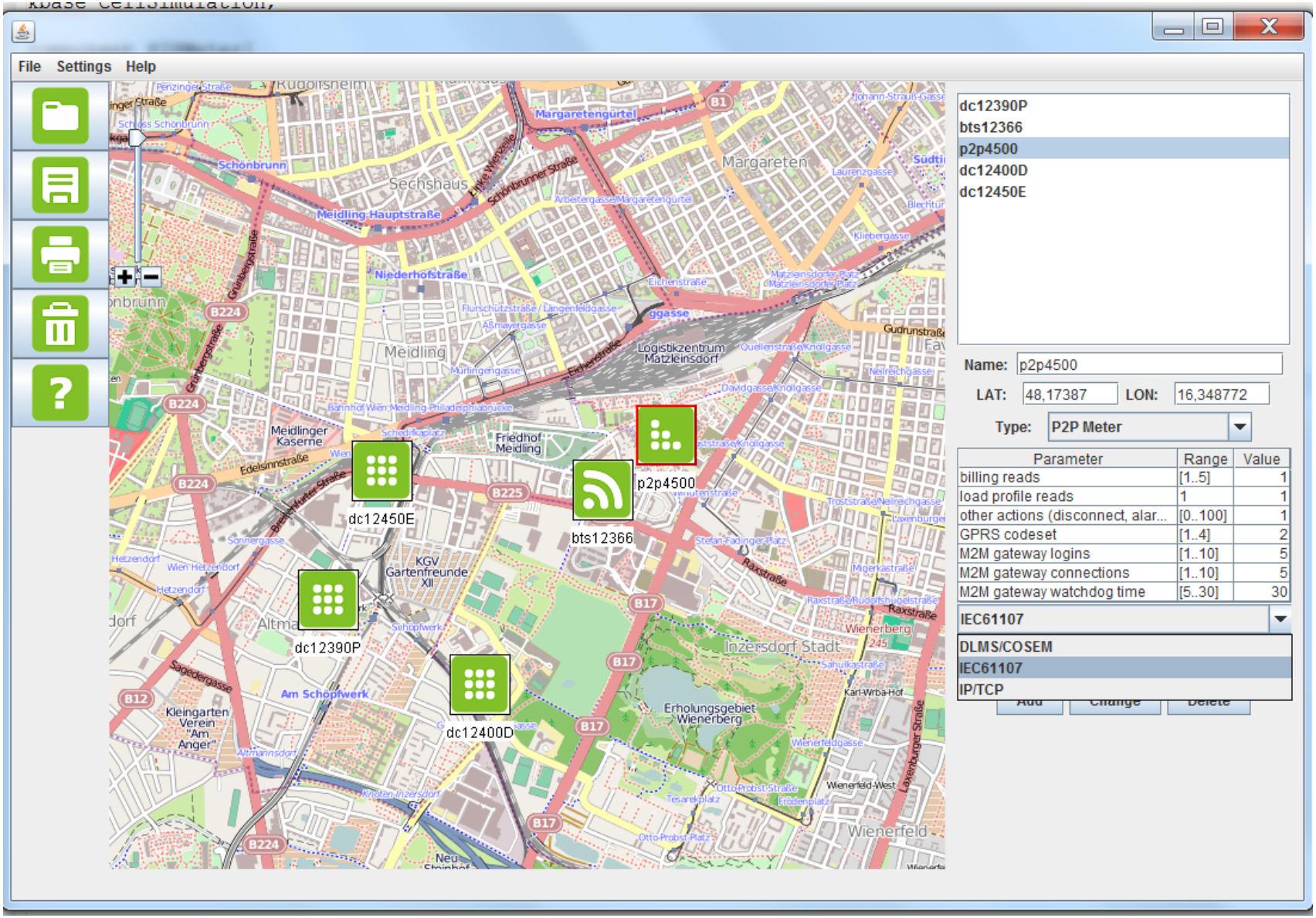
Motivation



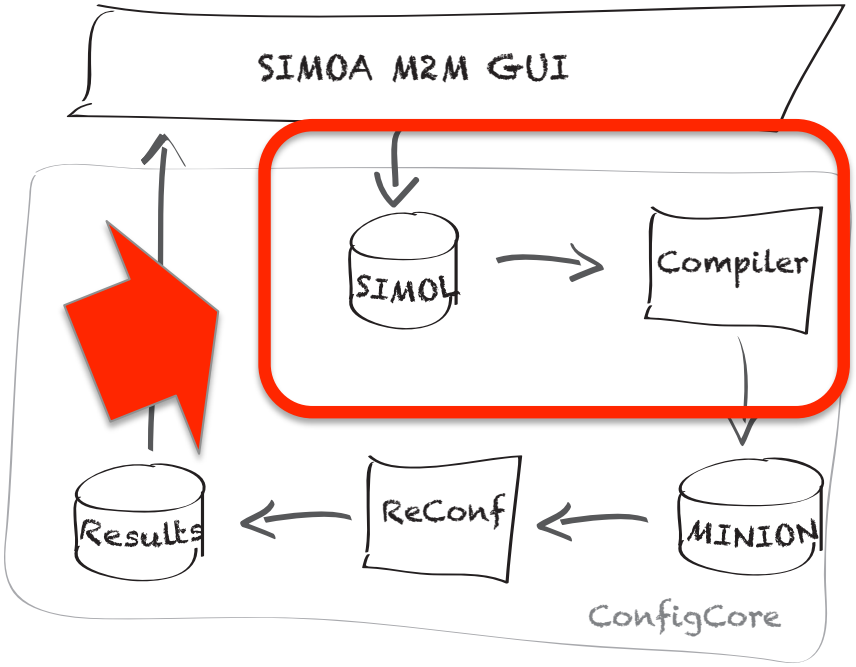
System architecture



SIMOA M2M GUI



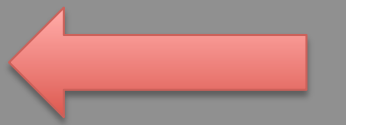
SIMOL



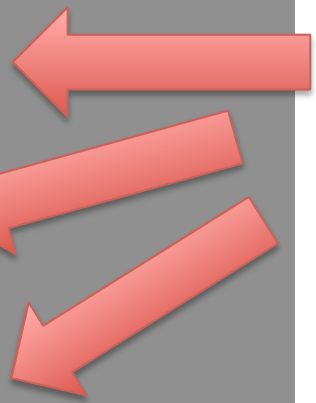
- General purpose modeling language
- Syntax close to Java
- OO
- bool, int, arrays
- Equation/constraint based semantics

```
kbase GPRSCell;
component P2PMeter {
    attribute int mdist,codeset,mRate;
    constraints {
        mdist = {1..3};
        codeset = {1..4};
    }
}
component FPC {
    attribute int value;
    constraints(default) {
        value = 1;
    }
    constraints(x1) {
        value = {2..4};
    }
    constraints(unknown) {

    }
}
```

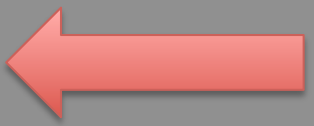


Constraints



Behavior modes

```
component BTS {
     attribute int fpc;
     constraints {
          FPC fpc1;
          fpc = fpc1.value;
     }
}
component Cell {
     attribute int neededR, realR;
     constraints {
          BTS b1;
          P2PMeter s[100];
          realR = sum([s], mRate)/P2PNo;
          realR >= neededR;
          ..
     }
     transition {
          forall ( P2PMeter ) {
               if (mdist = 1 and codeset = 2)
                    codeset.next = {2,3};
               if (mdist = 3 and codeset = 2)
                    codeset.next = {2,1};
          }
     }
}
```



new instance of FPC

sum constraint

state transition

conditional constraint

Semantics of SIMOL

```
kbase GPRSCell;
component P2PMeter {
    ....
    constraints {
        mdist = {1..3};
        codeset = {1..4};
    }
}

component Cell {
    ....
    P2PMeter s[100];
    ....
}
```

$$p2pmeter\_mdist \in \{1,2,3\}$$

$$p2pmeter\_codeset \in \{1,2,3,4\}$$

$$cell\_s\_1\_p2pmeter\_diss \in \{1,2,3\}$$

$$cell\_s\_2\_p2pmeter\_diss \in \{1,2,3\}$$

$$\vdots$$

$$cell\_s\_100\_p2pmeter\_diss \in \{1,2,3\}$$

$$\vdots$$

Semantics of SIMOL (II)

```
kbase GPRSCell;
component P2PMeter {
    ....
    constraints {
        mdist = {1..3};
        codeset = {1..4};
    }
}

component MyP2PMeter
extends P2PMeter {
    ....
    constraints {
        mdist = 1;
    }
}
```

$$p2pmeter\_mdist \in \{1,2,3\}$$

$$p2pmeter\_codeset \in \{1,2,3,4\}$$

$$myp2pmeter\_mdist \in \{1,2,3\}$$

$$myp2pmeter\_mdist = 1$$

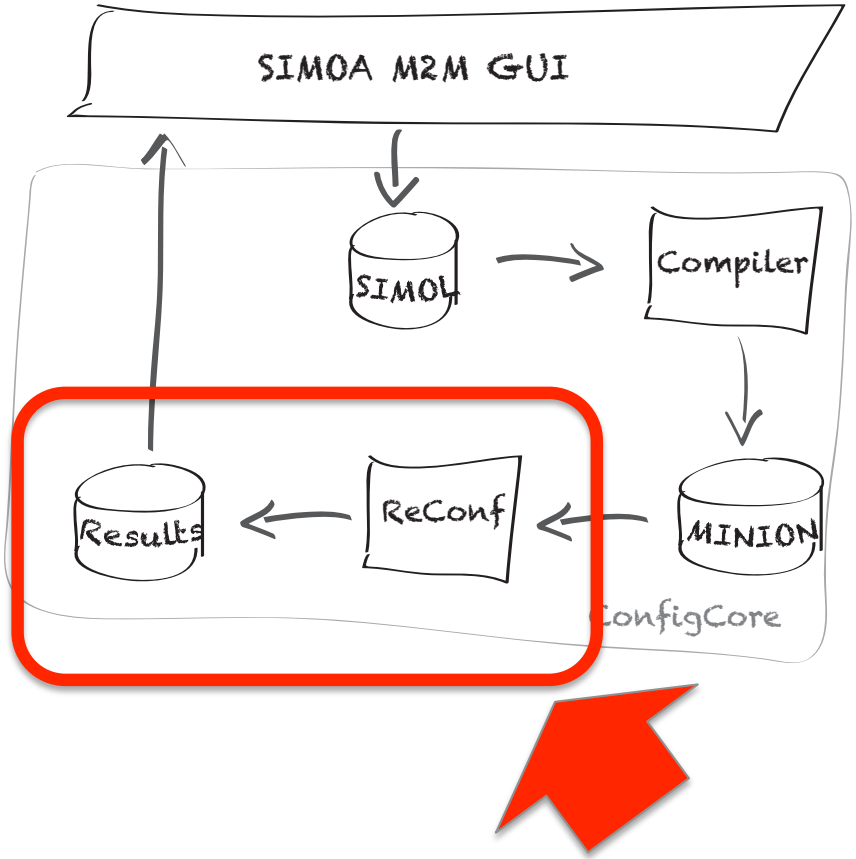
$$myp2pmeter\_codeset \in \{1,2,3,4\}$$

Summary SIMOL

- Syntax close to Java
- Semantics represented as set of equations/ constraints

- Implementation:
  - Mapping to MINION constraints

Reconfiguration engine



- Based on finding modes that allow fulfilling the requirements
- Use CSP solver (Minion)

Underlying idea

```
component FPC {
    attribute int value;
    constraints(default) {
        value = 1;
    }
    constraints(x1) {
        value = {2..4};
    }
    constraints(unknown) {

    }
}
```

```
Requirement: value = {4,5}
```

- The component modes can be changed by the reasoning engine
- Search for component modes such that there is no contradiction with requirements
- Reconfiguration as diagnosis

Formal definitions...

Definition 1 (Reconfiguration problem) *A reconfiguration problem can be defined as a tuple* $(KB, COMP, MODES)$, *where* $KB = SD \cup REQ$ *is the knowledge base comprising the model of the system* SD *and the requirements* REQ. $COMP$ *is a set of system*

Definition 2 (Mode assignment) *Given a set of components* $COMP$ *and a set of functional modes* $MODES$. *A mode assignment* M *is a function* $M : COMP \mapsto MODES$

Definition 3 (Reconfiguration) *Given a reconfiguration problem* $(KB, COMP, MODES)$. *A mode assignment* M *is a valid reconfiguration iff* $KB \cup \{M(c)|c \in COMP\}$ *is satisfiable.*

… up to minimality

Definition 4 (Number of changes) *Given a reconfiguration M for a reconfiguration problem $(KB, COMP, MODES)$. The number of changes (NOC) of M is equivalent to the number of modes in M deviating from the $default$ modes, i.e., $NOC(M) = |\{M(c)|c \in COMP \wedge M \neq default\}|$.*

We say that a reconfiguration M is optimal with respect to its NOC if it is minimal, i.e., there exists no other reconfiguration M' with $NOC(M') < NOC(M)$

Implementation

- Based on constraint solver MINION
- Algorithm close to model-based diagnosis

Algorithm 1 reconfig$(KB, COMP, MODES, n)$

Input: problem $(KB, COMP$ n

Output: All n cardinality n)

1: **for** $i = 0$ **to** n **do**
2: $CM = \{|\{M(c)|c \in COMP \wedge M \neq default\}| = i\} \cup KB$
3: $S = \mathcal{P}(\textbf{CSolver}(CM))$
4: **if** $S \neq \emptyset$ **then**
5: **return** S
6: **end if**
7: **end for**
8: **return** $\emptyset$

Search up to given cardina

Compute constraint model

If there is a solution of size *i*, return it.

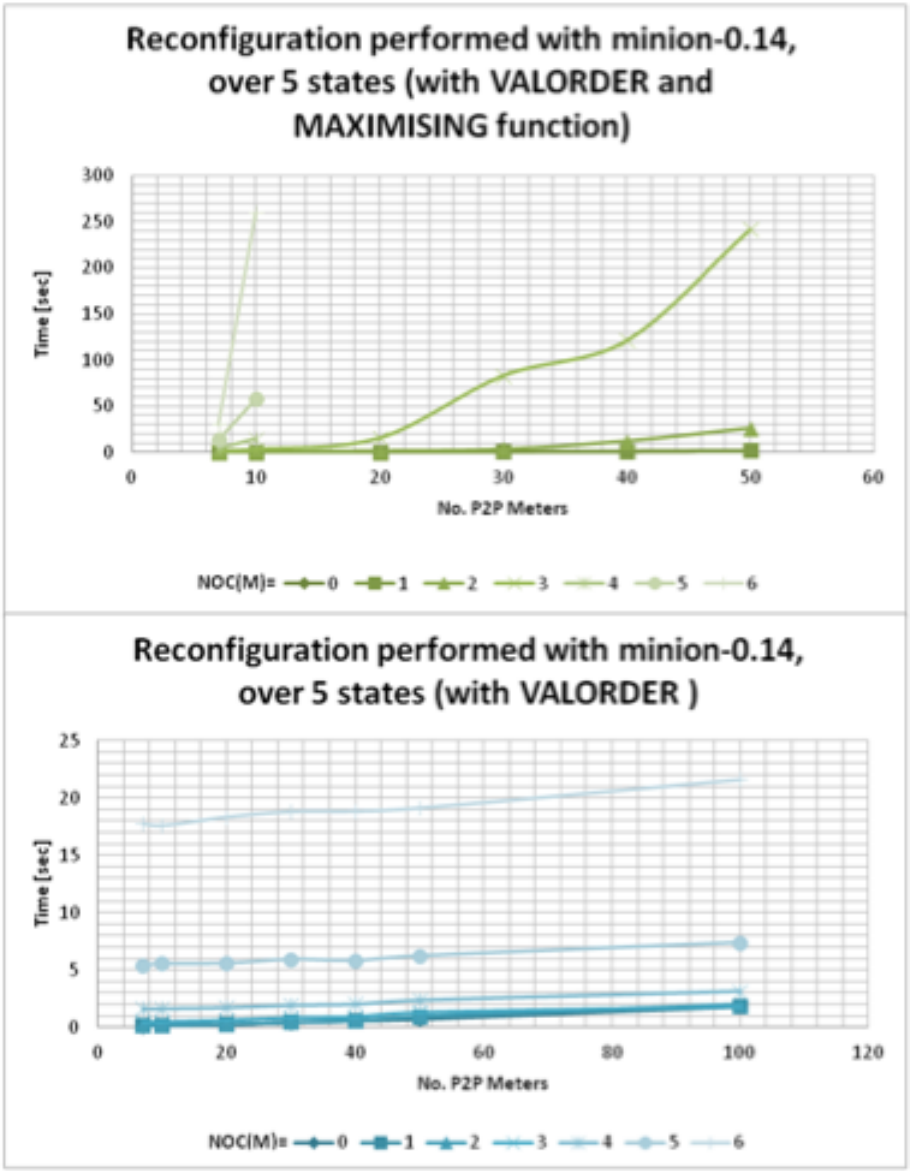
olver

SOME LESSONS LEARNED

Lesson 1

- Using the right reasoning engine in the **right way** is the key!

Coding is essential



Circuit	Single faults			Double faults			Triple faults		
	T_{avg} [s]	NoS_{min}	NoS_{max}	T_{avg} [s]	NoS_{min}	NoS_{max}	T_{avg} [s]	NoS_{min}	NoS_{max}
c17	0.0052	2	2	0.0104	1	1	0.0052	0	0
c432	0.2340	11	38	0.3432	0	72	0.5668	0	18
c499	0.3328	2	2	1.1856	28	28	15.8527	523	523
c880	0.7020	0	0	2.4284	0	0	26.4159	0	0
c1355	2.7612	0	5	86.5129	190	190	n.a.	n.a.	n.a.
c1908	9.2352	5	5	109.2785	0	0	n.a.	n.a.	n.a.
c2670	20.8417	1	1	21.1745	1	3	n.a.	n.a.	n.a.
c3540	37.2582	15	2	n.a.	0	n.a.	n.a.	0	n.a.
c5315	138.3620	1	68	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
c6288	38.2825	0	1	109.5279	0	109	n.a.	0	n.a.
c7552	184.6576	10	117	n.a.	0	n.a.	n.a.	n.a.	n.a.

Logic gates encoded using truth tables

Logic gates encoded using integer constraints directly

Circuit	Single faults			Double faults			Triple faults		
	T_{avg} [s]	NoS_{min}	NoS_{max}	T_{avg} [s]	NoS_{min}	NoS_{max}	T_{avg} [s]	NoS_{min}	NoS_{max}
c17	0.0104	2							0
c432	0.0364	11							18
c499	0.0468	2							523
c880	0.0624	0							0
c1355	0.1352	0							n.a.
c1908	0.3172	5							0
c2670	0.5252		1	0.5304	1	3	47.9858	1	18
c3540	0.4472		15	108.7845	0	274	0.234	0	n.a.
c5315	1.7628		68	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
c6288	0.2808	0	1	1.0348	0	109	n.a.	0	n.a.
c7552	2.2152	10	117	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.

More than 80 times faster!

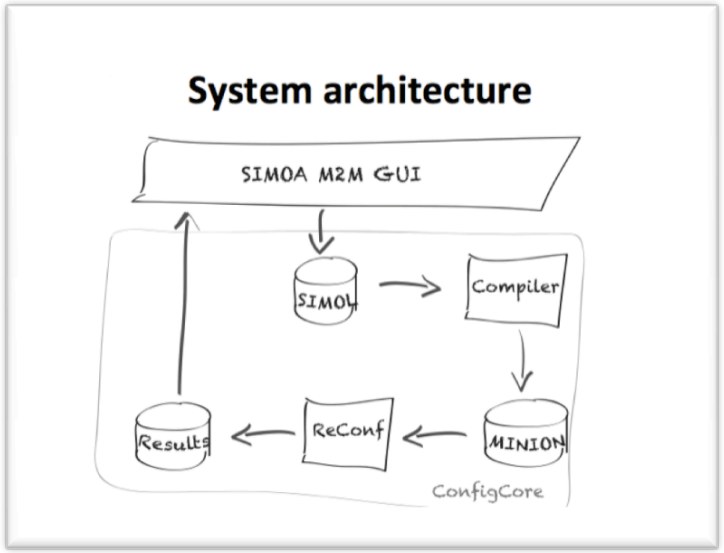
Lesson 2

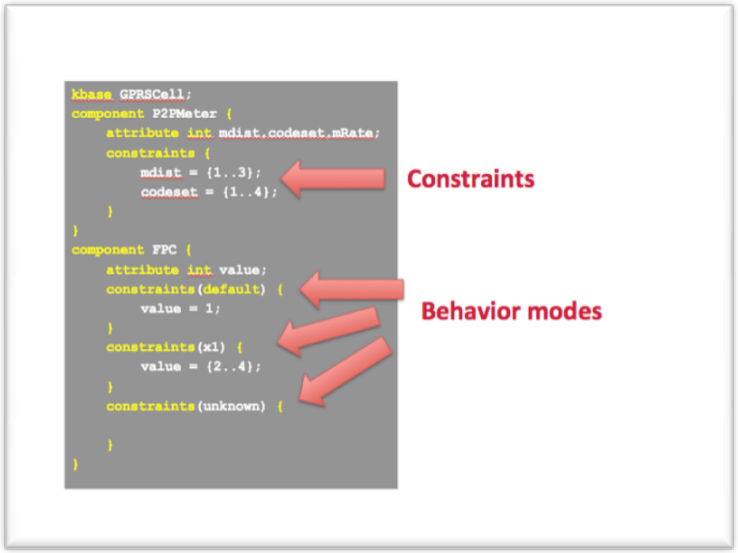
- Modeling is the key!
  - Modeling is not that easy
  - Modeling languages are hardly used
  - Training is essential (but requires time and effort on side of the industrial partners)

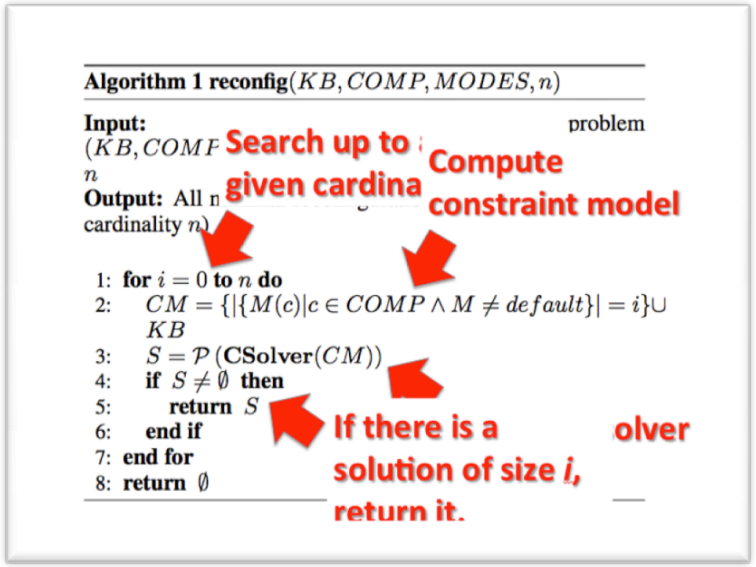
Lesson 3

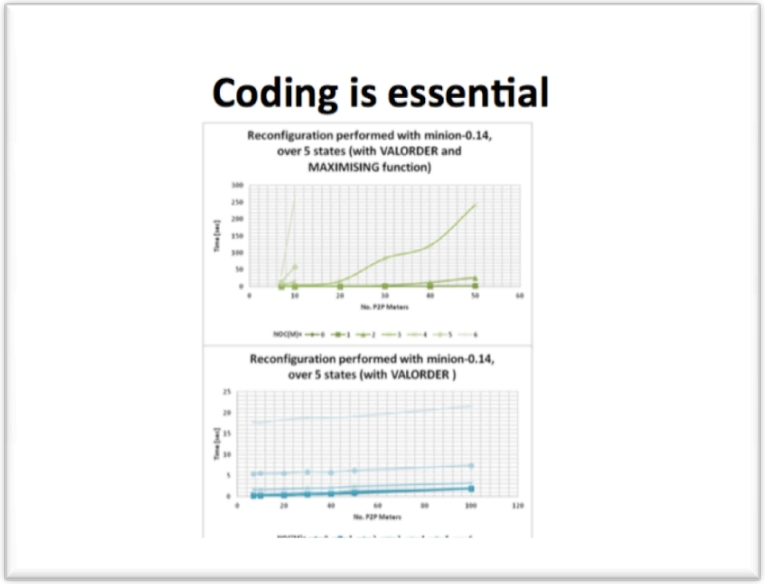
- Bringing research into (daily) practice is hard
  - Additional effort and money needed
  - Both is hard to get (even in case industry is happy with the obtained project results)

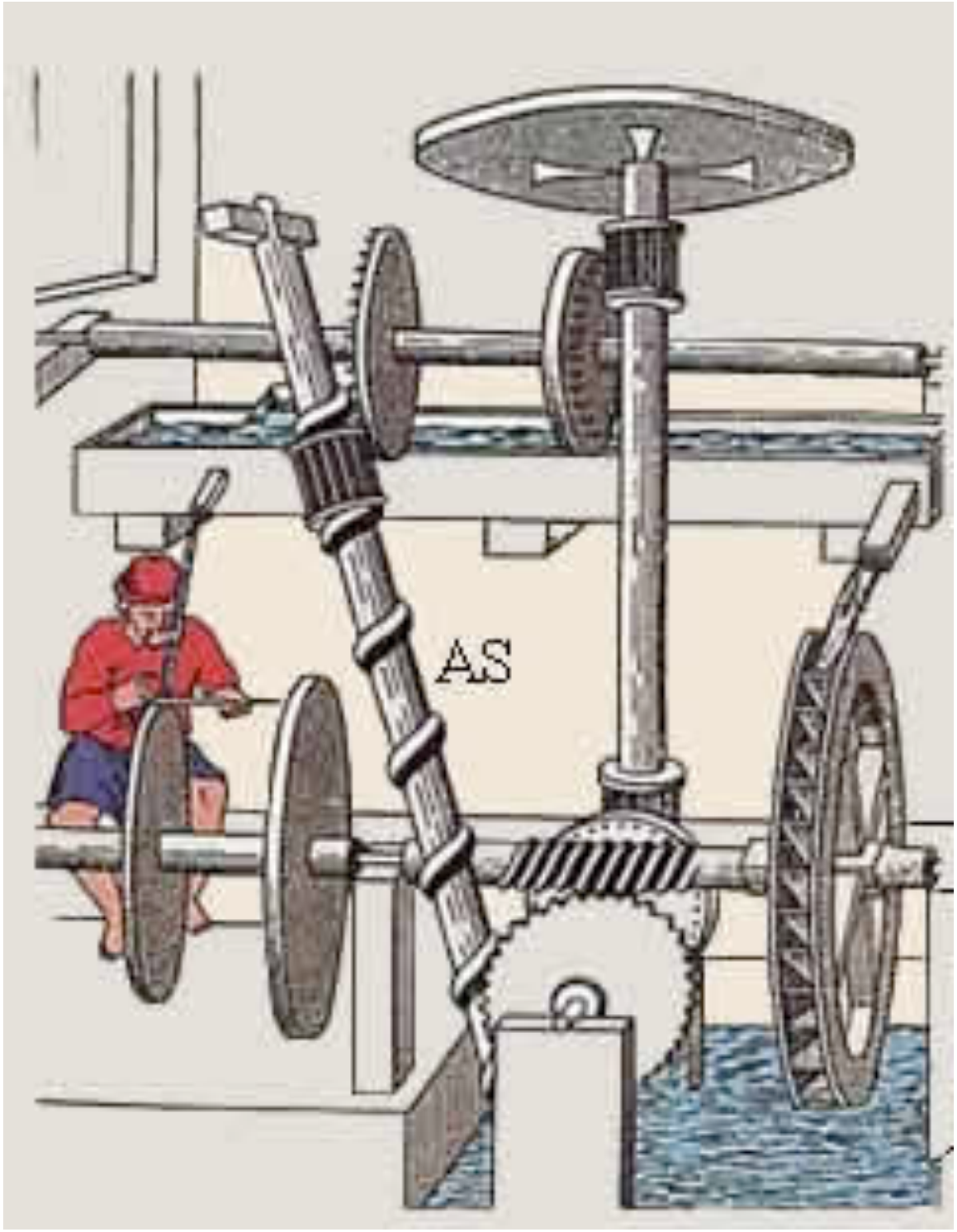
Conclusions











QUESTIONS?