Knowledge-based (re)configuration: an ASP approach

Kostyantyn Shchekotykhin

Alpen-Adria-Universität Klagenfurt

Acknowledgements

Many thanks to:

Markus Aschinger, Conrad Drescher, Andreas Falkner, Gerhard Friedrich, Georg Gottlob, Alois Haselböck, Peter Jevons, Anna Ryabokon, Gottfried Schenner, Herwig Schreiner, Erich Teppan, Evgenij Thorstensen (Projects “Reconcile” and “Constraint satisfaction for configuration”)

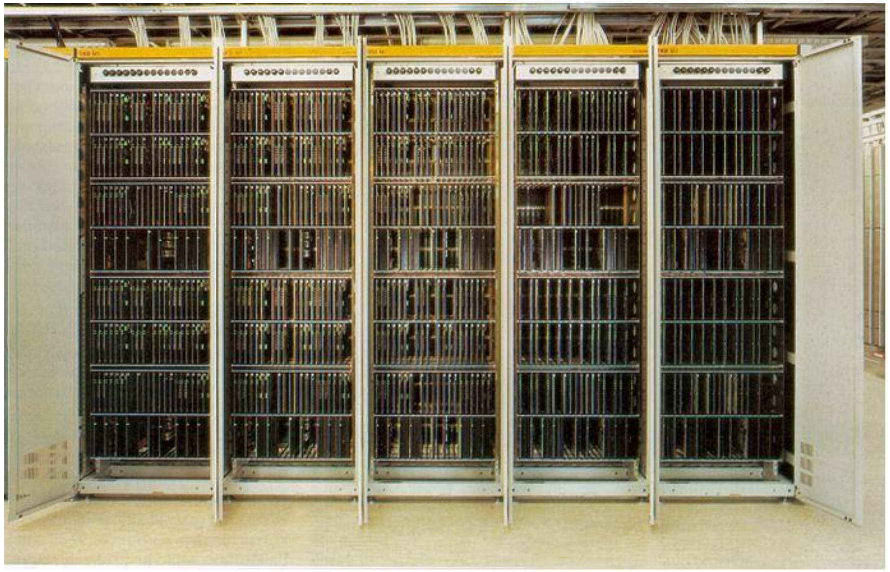
and many other researchers in the field of knowledge-based configuration who made this lecture possible

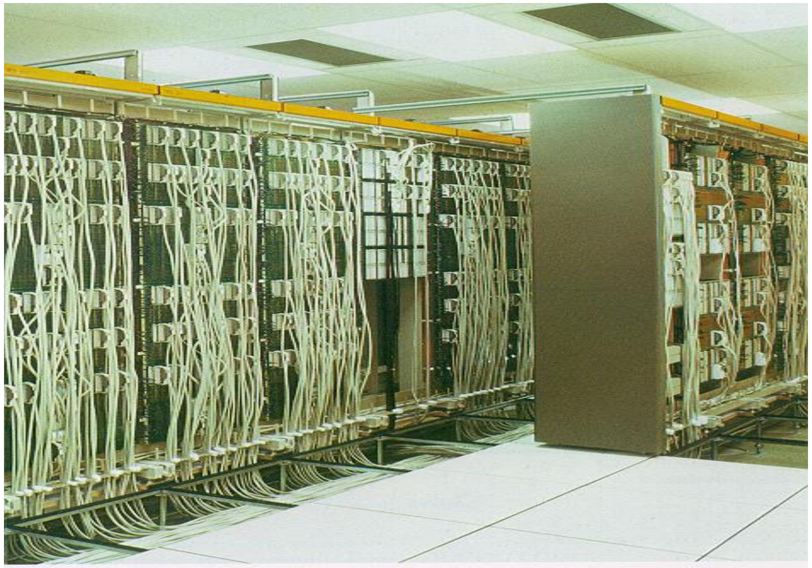
Outline

- Motivation
- Configuration problem
  - Problem definition
  - Example: House problem
- Configuration KRRs
  - A bit of history
  - Overview of modern KRRs
- Reconfiguration problem

Telecom switches (1997)

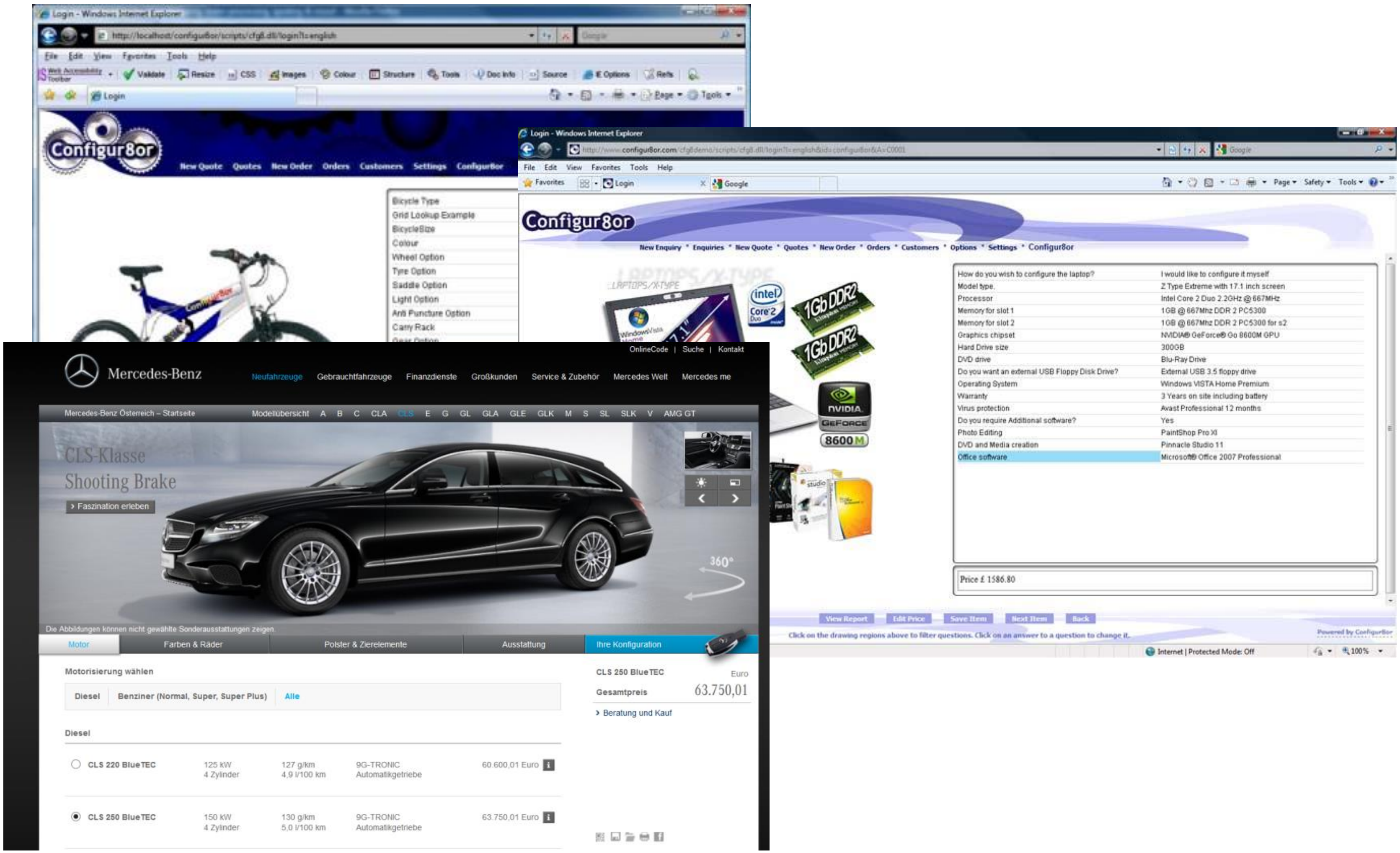
- 200 racks, 1000 frames, 30000 modules
  10000 cables
- Successful configuration system





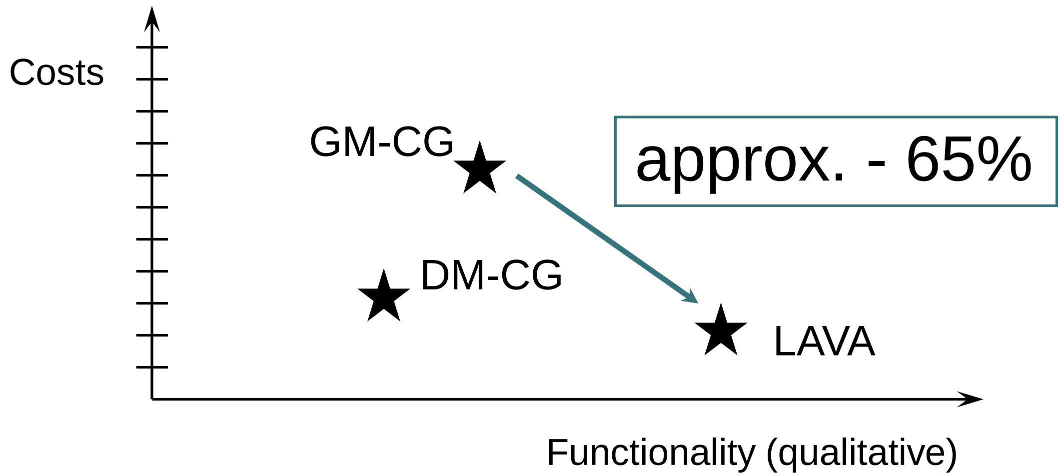
Server farms (@CERN)

Product customization



Knowledge-based configuration

Development costs for configurators:



Since 1997

Additional functionality:

- Interactive configuration
- Improved user interface
- Explanations
- Extension of existing configurations

...

DM- & GM-CG: standard procedural approach
LAVA: **"constraint based programming"** (generative, incomplete)

Requirements to a configurator

- Declarative language for problem description
  - Ideally: every programmer must be able to use it after some (moderate) training
- Computation of solutions in admissible time
- Optimization of found solutions (optionally)

Is there a KRR approach for satisfying the requirements?

Configuration problem

Configuration problem

Given

- Requirements (restricted FOL sentences)
  - Configuration or *general requirements* R_G
  - Customer or *instance requirements* R_I
- A set of predicated $R_S = \{p_1, \dots, p_n\}$ corresponding to a *solution schema*

Find a finite, subset minimal set of ground atoms, called *configuration*, $CONF = \pi_{R_S} M$

- M is a Herbrand model of $R_G \cup R_I$
- $\pi_{R_S} M$ is a projection of ground atoms M onto ground atoms over predicates R_S

Optimization problem

Given

- requirements R_G, R_I and solution schema R_S
- *cost function* $f: \boldsymbol{CONF} \mapsto \mathbb{N}$, where $\boldsymbol{CONF}$ denotes a set of all configurations
- *bound* $B \in \mathbb{N}$

Find a configuration $CONF$ such that

$$f(CONF) \leq B$$

Requirements overview I

Configuration requirements R_G

- Components catalog defines
  - Types (motherboard, cpu, fan, hdd, etc.)
  - Attributes (cpu frequency, motherboard format)
    - Ports (socket type of a motherboard component)
- Valid relations between the components
  - Cpu2Motherboard: for each cpu there exists a motherboard with a compatible socket

Encoded as linear Tuple Generation Dependencies (TGD) (see e.g. $Datalog^{\pm}$ [Calì et al., 2010])

Requirements overview II

Example of a linear TGD (LoCo [Aschinger et al, 2014])

$$\forall(id_1,\vec{x})\ C_1(id_1,\vec{x})$$
$$\rightarrow \exists_l^u(id_2)\ [C_2(id_2,\vec{y}) \wedge C_1 2C_2(id_1,id_2) \wedge \phi(id_1,id_2,\vec{x},\vec{y})]$$

- $C_i(id_j,\vec{x})$ denotes a component type with an identifier id_j and a vector of attributes $\vec{x}$
- $C_i 2C_j\left(id_i,id_j\right)$ denotes a relation between 2 components
- $\phi(id_1,id_2,\vec{x},\vec{y})$ is a formula expressing additional constraints
- $l \geq 0$ and $u > 0$ are lower and upper bounds

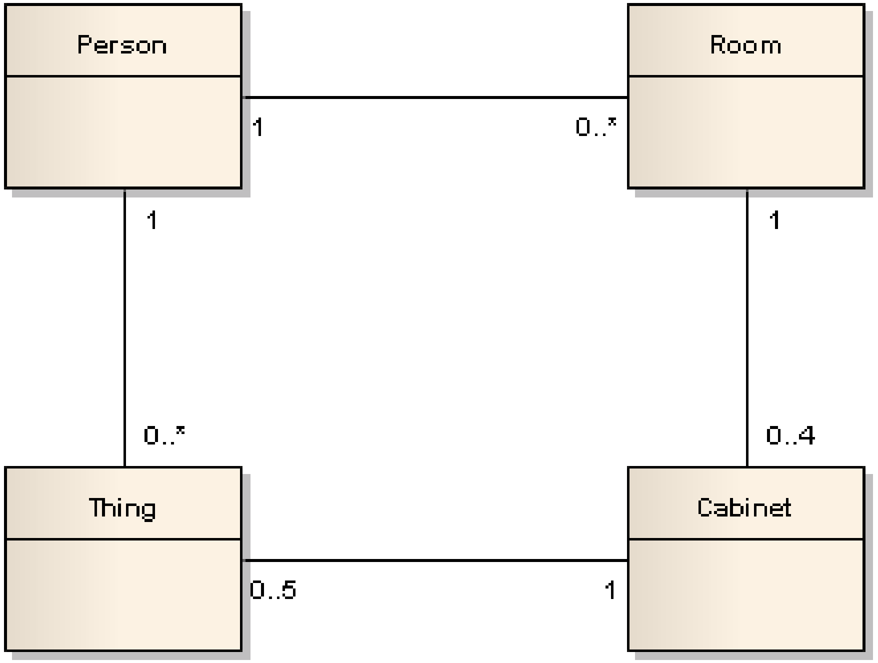
Requirements overview III

Customer requirements R_I

- Define available components
  - Facts of the form $C_j(id_i, \vec{x})$
- Specify input relations
  - Facts of the form $C_j2C_i\left(id_j, id_i\right)$
- Declare a solution schema
  - Set of predicates
- Provide preference criteria (optimization)

Configuration example (House)

things**s** (lamps) are related to **a** person (signal),
things**s** are stored in **a** cabinet (module),
cabinet**s** are placed in **a** room (frame)



1-n relations
+
additional constraints

(Siemens©)

House problem

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT | WIEN GRAZ

Customer requirements:

Declaration of available persons, things and ownership relations between them

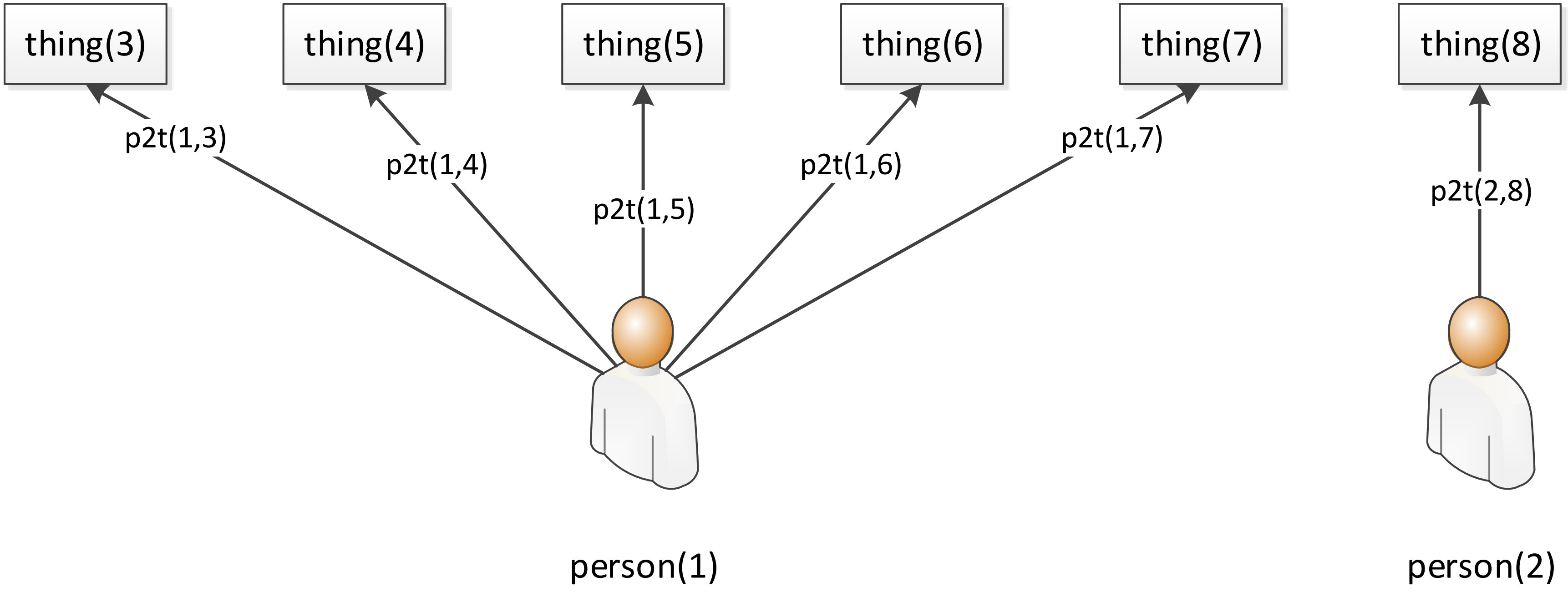
Configuration requirements:

1. each thing must be stored in exactly one cabinet
2. a cabinet can contain at most 5 things
3. every cabinet must be placed in exactly one room
4. a room can contain at most 4 cabinets
5. each room belongs to a person
6. and a room may only contain cabinets storing things of the owner of the room

Goal: store all things in a house such that the set of requirements is fulfilled

Sample customer requirements

- Person 1 owns things 3,4,5,6,7
- Person 2 owns thing 8



Predicates: `person/1, thing/1` and `p2t/2`

House encoding

- Customer requirements (facts)
- Configuration requirements
  - each thing must be stored in exactly one cabinet and a cabinet can contain at most 5 things

$$\forall id_T \; thing(id_T) \rightarrow \exists_1^1 id_C \; [cabinet(id_C) \land t2c(id_T, id_C)]$$
$$\forall id_C \; cabinet(id_C) \rightarrow \exists_0^5 id_T \; [thing(id_T) \land t2c(id_T, id_C)]$$

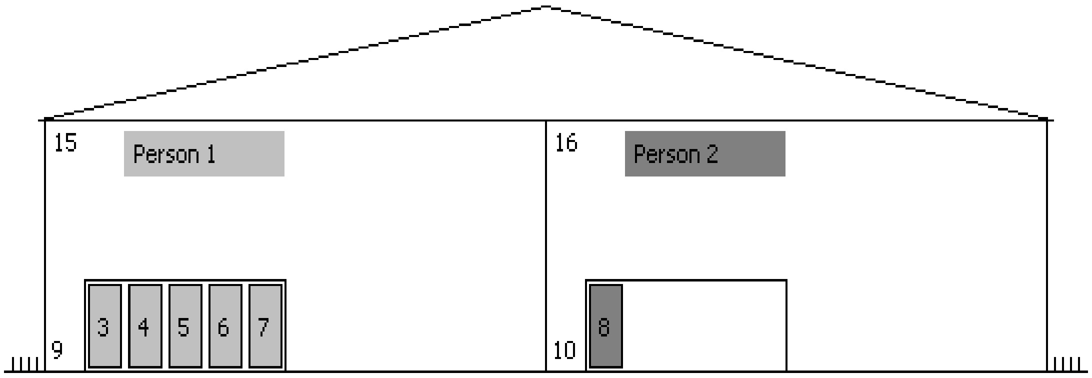
  - each room belongs to a person

$$\forall id_P, id_T, id_C, id_R \quad p2t(id_P, id_T) \land t2c(id_T, id_C) \land c2r(id_C, id_R) \rightarrow p2r(id_P, id_R)$$

Sample solution

- Solution schema: `cabinet/1, room/1, t2c/2, c2r/2` and `p2r/2`
- Solution:

```
cabinet(9). cabinet(10). room(15). room(16). ...
p2r(1,15). p2r(2,16). ...
c2r(9,15). c2r(10,16). ...
t2c(3,9).  t2c(8,10). ...
```



Configuration KRRs

KRRs overview

- Production rules (R1/XCON [McDermott, 1992], ILOG JRules, OpenRules)
- Case-based [Rahmer et al., 1996; Tseng et al., 2005]
- Model-based (Oracle, SAP, Siemens)
  - Constraint-based [Mittal et al 1989], Dynamic CSP [Mittal et al., 1990], Generative CSP [Stumptner 1997]
  - Specific languages: UML [Felfernig et al., 2000], Feature models [Dhungana et al., 2011], LoCo [Aschinger et al., 2014]
  - Description logics [McGuinness 2003]
  - SAT [Aschinger et al 2011] and ASP [Friedrich et al., 2011]
  - Hybrid approaches [Hotz et al., 2006]

CSPs

- De facto standard KRR for configuration Variables $V = \{v_1, \ldots, v_n\}$ and their domains $D = \{dom(v_1), \ldots, dom(v_n)\}$ represent components, attributes and relations
- Constraints define allowed compositions of components
  - TGDs are approximated by global cardinality constraints
- Supports optimization and heuristics
- Possible modeling language MiniZinc [Nethercote et al., 2007]

Generative CSP

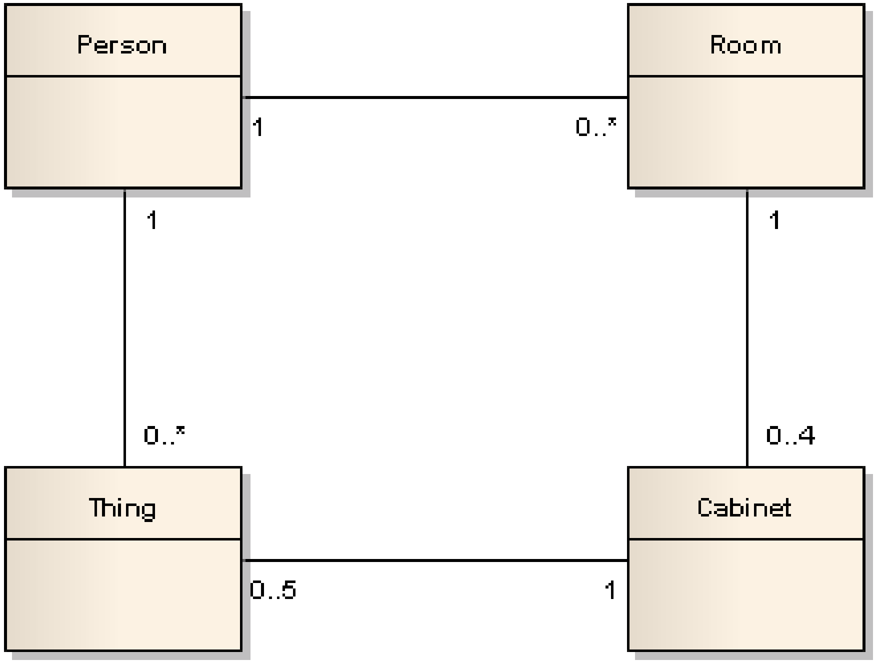
- Differentiate between component variables and other variables
- Component variables – classes
- Language is similar to OO programming

```
class Cpu
   attr socket: integer
class Mb
   attr socket: integer
assoc Cpu.board(1) - Mb.cpu(1)
constraint Mb.compatible :
        cpu.socket = cpu.board.socket
```

Object-oriented approaches

UML:

- Generic language (+)
- Constraints are hard to write (-)



(Siemens©)

Description logics

- Components are modeled as concepts (atoms over unary predicates)
- Roles (atoms over binary predicates) are used to model relations and attributes

$$Cpu \sqsubseteq \exists compatible.Mb$$
$$\forall X\; Cpu(X) \rightarrow \exists Y\; [compatible(X,Y) \wedge Mb(Y)]$$

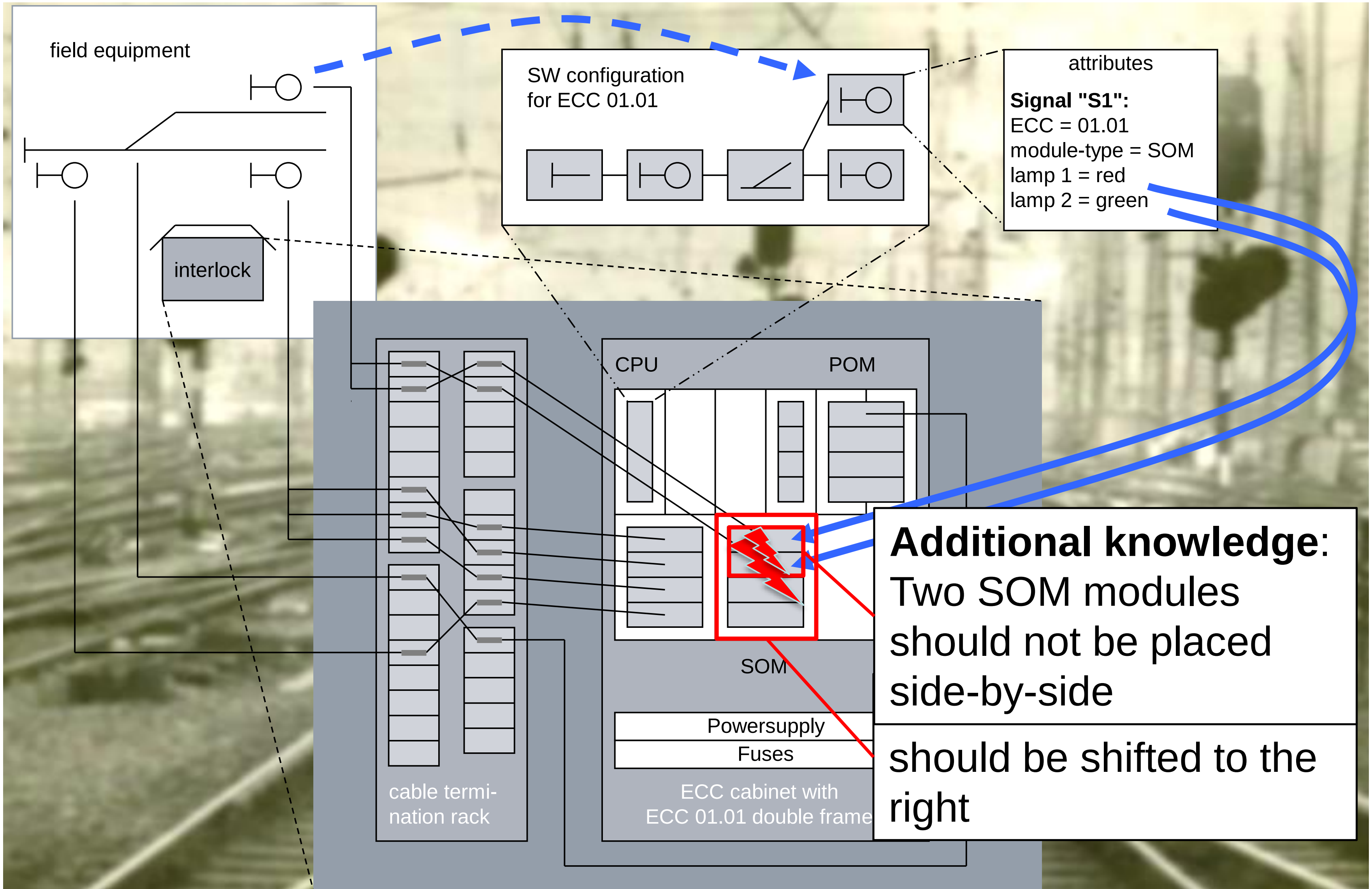
- Reasoning services support query answering
- Generation of a query (configuration) must be done by external tools, e.g. ASP

ASP

- Modeling of components catalog is the same as in FOL
- TGDs are approximated by either
  – disjunctive rules + cardinality constraints or
  – choice rules (cardinality atom in the head)
- Simple and easy to understand modeling language
- Supports optimization

Reconfiguration problem

Motivation



Additional knowledge:
Two SOM modules should not be placed side-by-side

should be shifted to the right

Sample configuration changes I

Additional customer requirements:

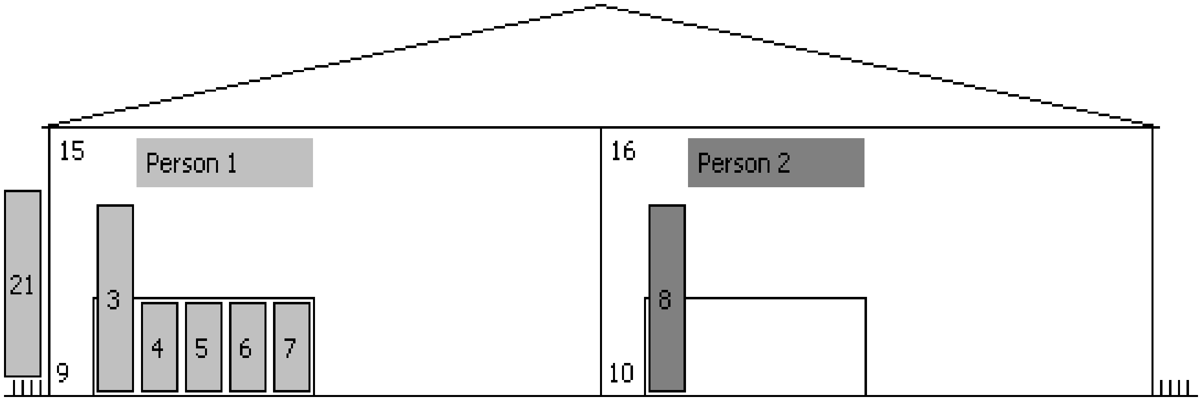
definitions of long and short things

```
thingLong(3). ... thingShort(7). thingLong(8).
thing(21). thingLong(21). p2t(1,21).
```

Sample configuration changes II

Additional configuration requirements:

- a cabinet is either small or high
- a long thing can only be put into a high cabinet
- a small cabinet occupies 1 and a high cabinet 2 of 4 cabinet places available in a room
- all legacy cabinets are small



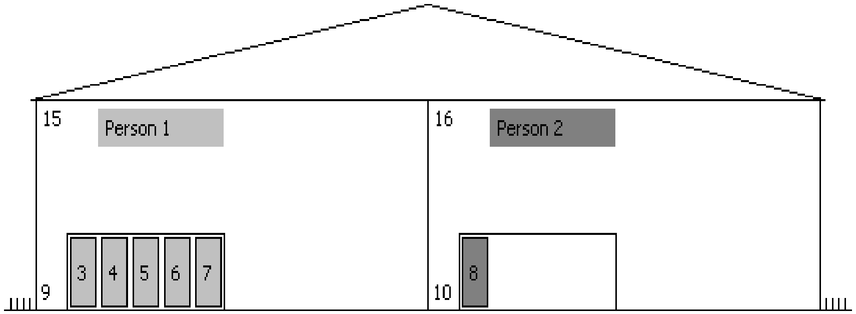
Inconsistent configuration

Transformation changes

Legacy configuration encoding (functional symbols):

```
legacyConfig(c2r(10,16)).
legacyConfig(cabinet(9)).
legacyConfig(cabinet(10)).
```

…



Transformation rules:

- Parts of the legacy configuration can either be reused or deleted
- Individuals of reused relation tuples should also be reused
- Relation tuples of deleted individuals should be excluded

Objective function

- Introduce costs for each action:
  - Creation costs: individuals and relation tuples absent in the legacy configuration
  - Reuse costs: individuals and relation tuples present in both reconfiguration and legacy configuration
  - Deletion costs: individuals and relation tuples of legacy configuration absent in the reconfiguration
- Optimization criterion: minimize sum of all costs

Reconfiguration problem

Given

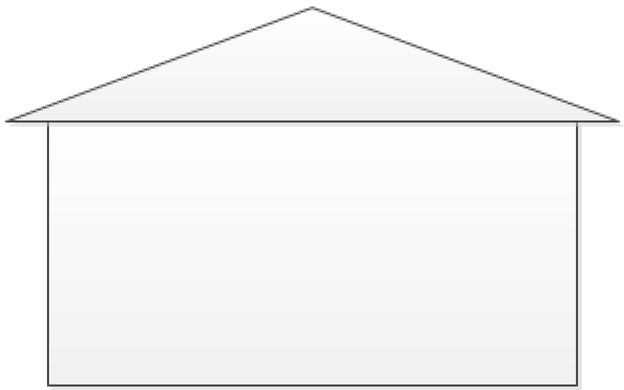
- *legacy configuration* as set of ground literals R_L,
- set of *requirements* $R_G \cup R_I$,
- set of *transformation rules* R_T
- set of predicates describing a *solution schema* R_S
- and an *objective function* f:$\boldsymbol{CONF} \mapsto \mathbb{N}$ and *bound* $B \in \mathbb{N}$

Find a finite, subset minimal set of ground atoms, reconfiguration, $RECONF = \pi_{R_S} M$ such that $f(RECONF) \leq B$

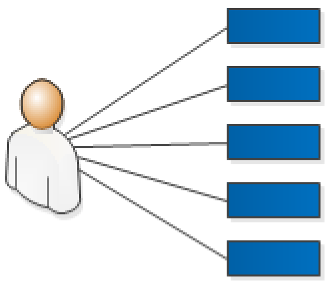
- M is a Herbrand model of $(R_G \cup R_I) \cup (R_L \cup R_T)$ and π is a projection operator

Test cases: Empty

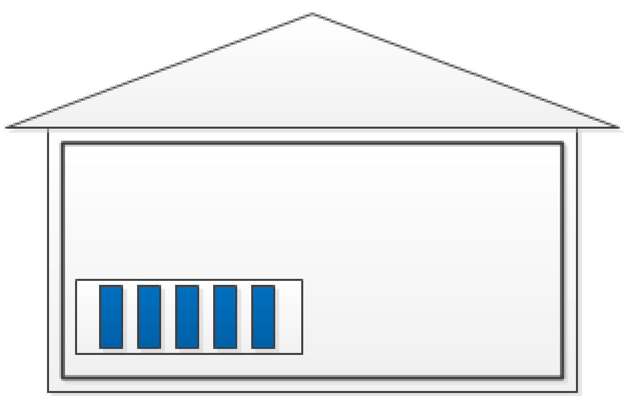
Legacy configuration



Customer requirements

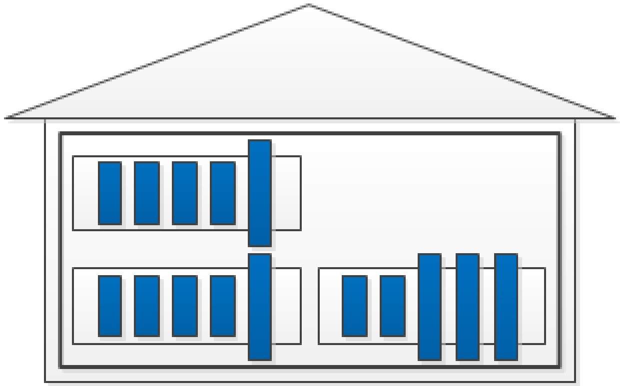


Possible solution

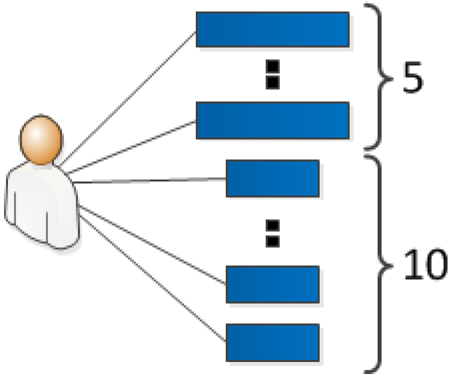


Test cases: Long

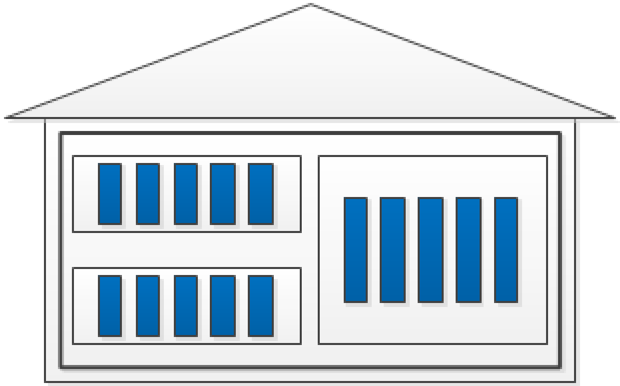
Legacy configuration



Customer requirements

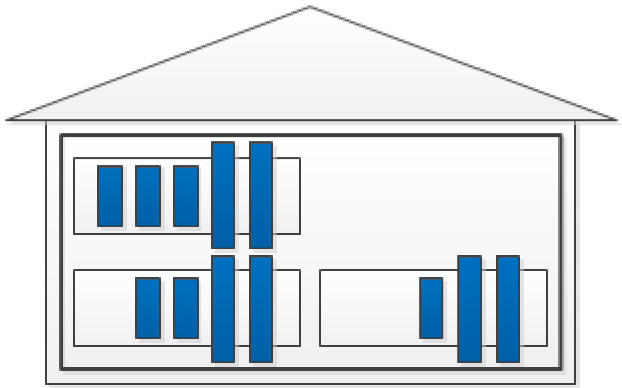


Possible solution

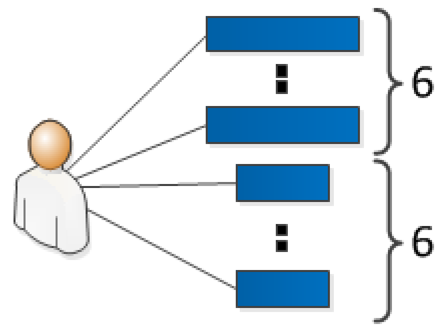


Test cases: New Room

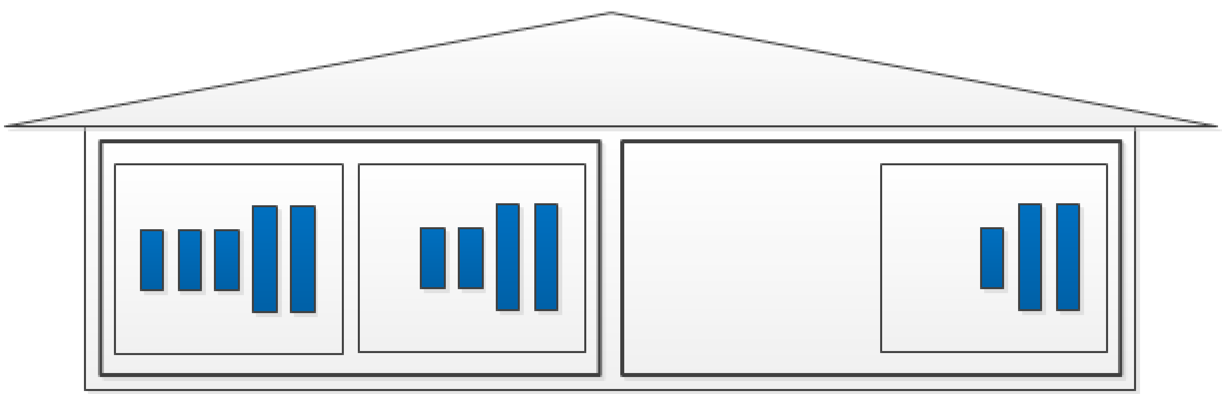
Legacy configuration



Customer requirements

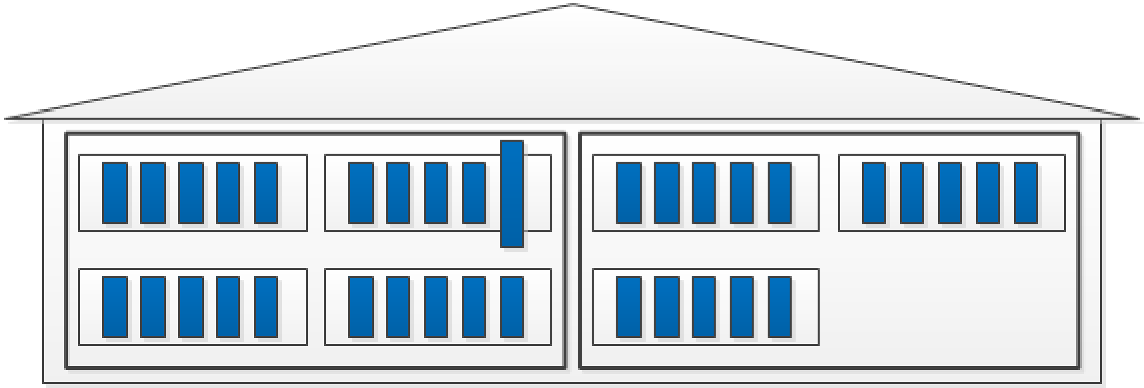


Possible solution

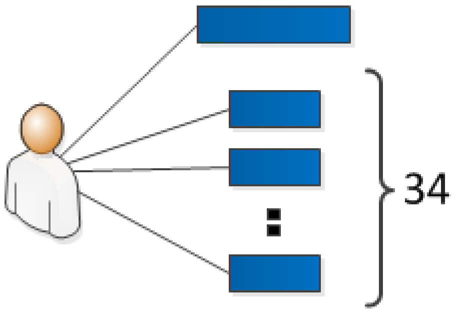


Test cases: Swap

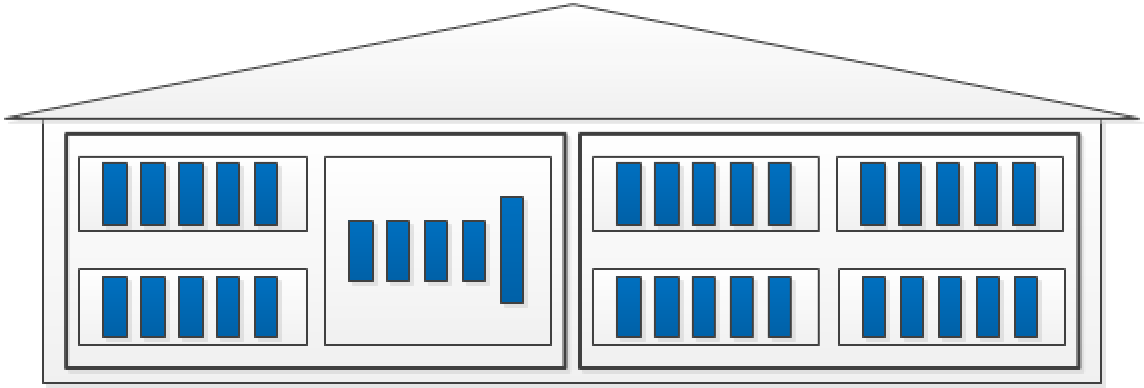
Legacy configuration



Customer requirements



Possible solution



Thank you for your attention!

Questions?