Constraint Programming for Decision Making in Self-Organizing, Adaptive Systems

Alexander Schiendorfer

Motivation

Institute for
Software & Systems
Engineering

Constraint programming

- Declarative paradigm to problem solving
- Specifying problems as *variables*, *domains*, and *constraints*
- A *versatile* tool for combinatorial and optimisation problems (resource allocation, job-shop scheduling etc)

Motivation

Institute for Software & Systems Engineering

Constraint programming

- Declarative paradigm to problem solving
- Specifying problems as *variables*, *domains*, and *constraints*
- A *versatile* tool for combinatorial and optimisation problems (resource allocation, job-shop scheduling etc)

Self-organizing, adaptive Systems:

- Claim to achieve higher robustness by local rules that lead to re-structuring
- *Fault-tolerance* by added redundancies
- Typically require a feedback loop along with reconfigurations (e.g., MAPE cycle, Observer-Controller, . . .) Ramirez and Cheng (2010)
- Intended system behaviour can be described in terms of first-order predicates over a system's variables
- $\rightarrow$ Use constraints from *specification* for the implementation of *reconfiguration*)

Two worlds

Institute for
Software & Systems
Engineering

Constraints for Specification and Analysis

- Derived from Requirements Engineering processes
- Formal description in, e.g., OCL
- Models available for formal analysis

Two worlds

Institute for
Software & Systems
Engineering

Constraints for Specification and Analysis

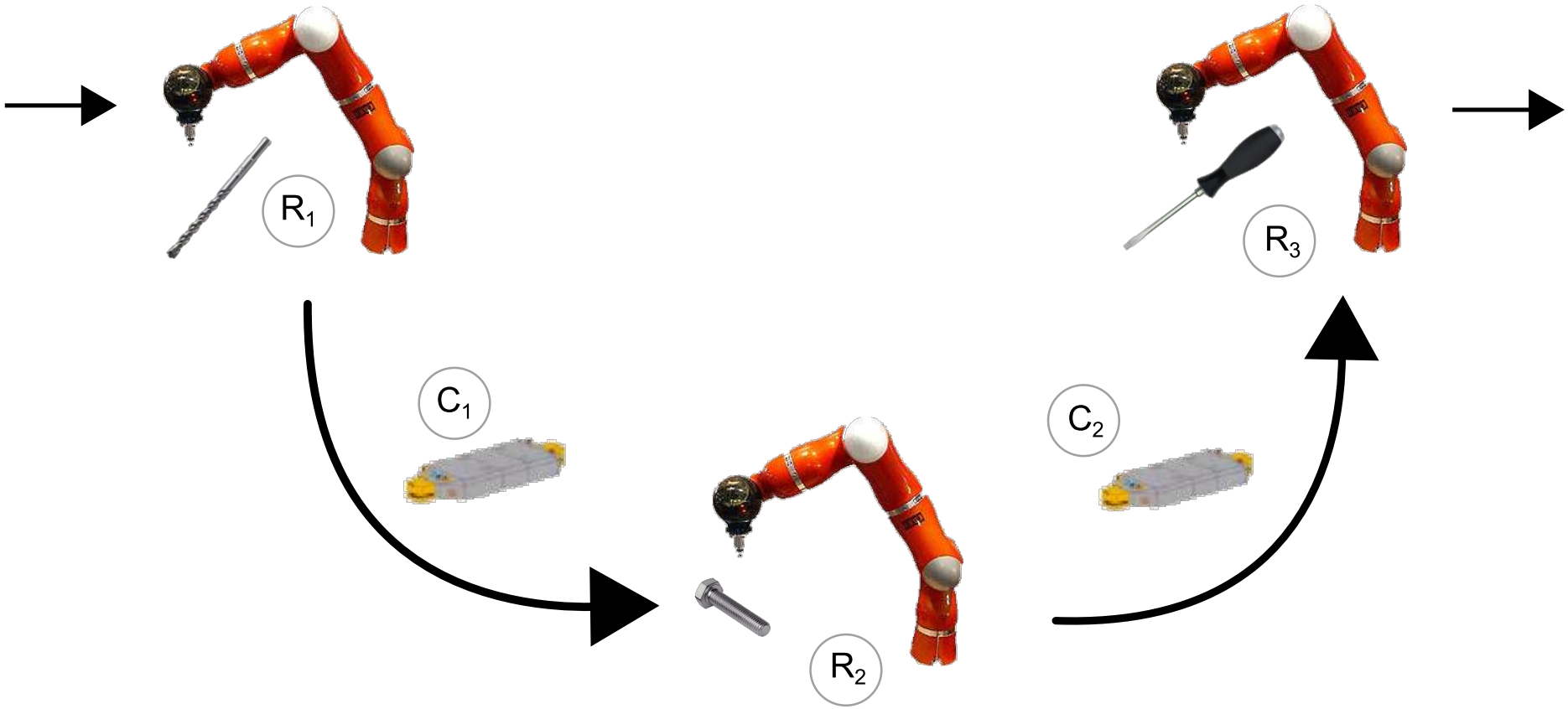
- Derived from Requirements Engineering processes
- Formal description in, e.g., OCL
- Models available for formal analysis

Constraints for Engineering and Optimization

- Decision making for reconfigurations
- Self-optimization using constraint models of agents that are combined automatically

A first example

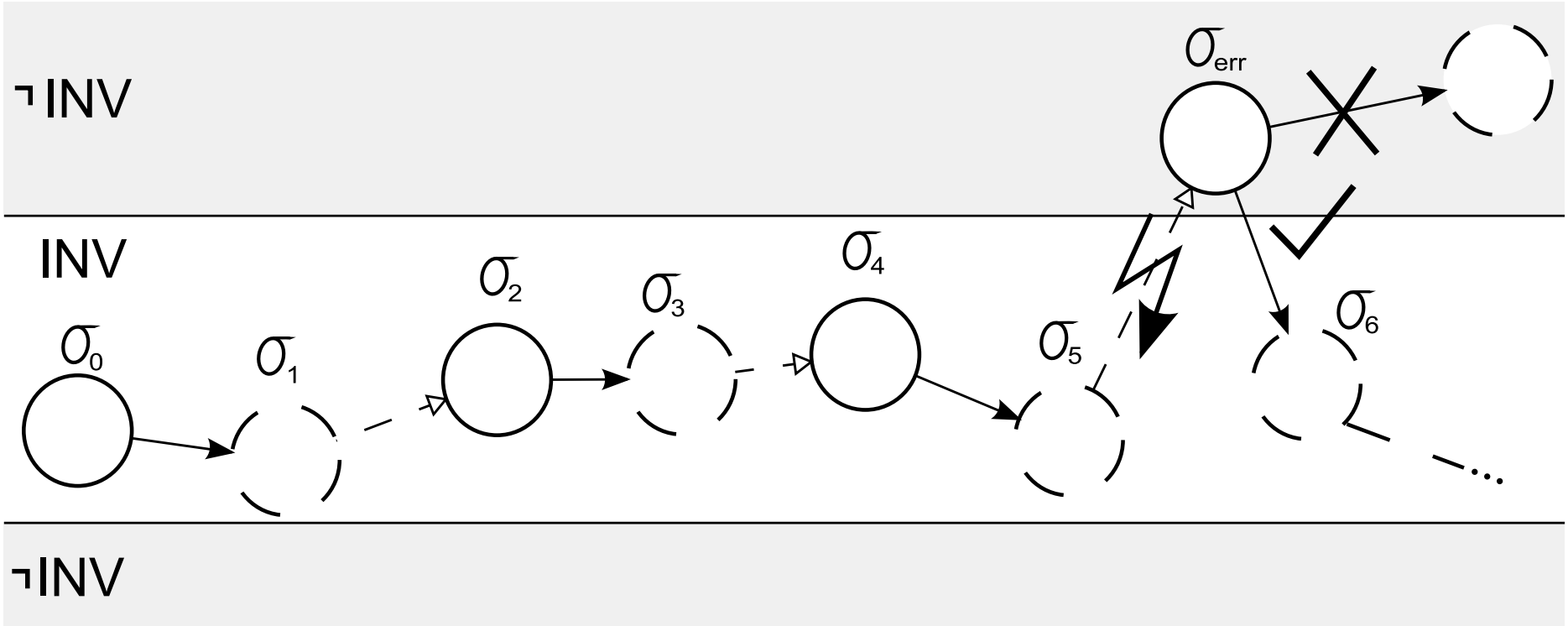
Institute for Software & Systems Engineering



A bit more formally

Institute for Software & Systems Engineering

The **R**estore-**I**nvariant-**A**pproach



Nafz et al. (2013)

General Idea

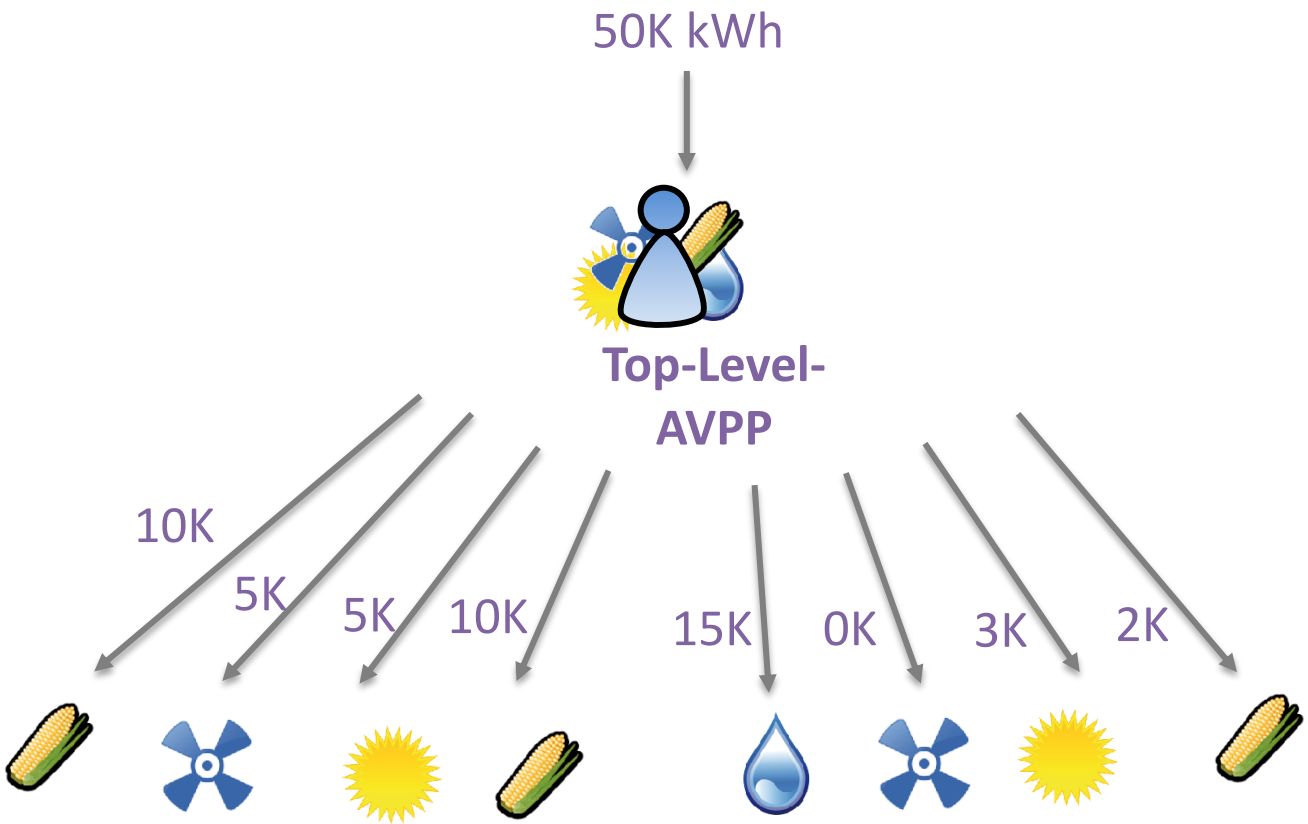
Institute for
Software & Systems
Engineering

1. Notice change in the system that violates the invariant
   1. Assume, a driller breaks $\rightarrow$ drill $\notin$ *availableRoles*
   2. Assume further, this robot is assigned to drill *assignedRole* $=$ drill
   3. e.g., *assignedRole* $\notin$ *availableRoles*
2. Search for new assignment using constraint programming
3. *Restore the invariant*
4. Provable correctness as long as the system is within the corridor (using, e.g., theorem provers such as KIV or model checking tools, hybrid automata, . . .)

A second example . . .

Institute for Software & Systems Engineering

Unit Commitment in Power Management



Time-dependent resource allocation

Institute for
Software & Systems
Engineering

Variation of general resource allocation

- Demand is given for a time frame $\mathcal{T}$
- Agents are subject to constraints such as minimal or maximal rates and *inertia*
- Therefore, *proactive* allocation necessary!
- Limited rates of change between consecutive time steps

Time-dependent resource allocation

Institute for
Software & Systems
Engineering

Variation of general resource allocation

- Demand is given for a time frame $\mathcal{T}$
- Agents are subject to constraints such as minimal or maximal rates and *inertia*
- Therefore, *proactive* allocation necessary!
- Limited rates of change between consecutive time steps

$$\begin{aligned}
\underset{\mathcal{P}}{\text{minimise}} \quad & \sum_{t \in \mathcal{T}} |\mathcal{D}_t - \sum_{a \in \mathcal{A}} \mathcal{P}_t^a| \\
\text{subject to} \quad & \exists [x, y] \in L^a : x \leq \mathcal{P}_t^a \leq y, \, \forall a \in \mathcal{A}, \, t \in \mathcal{T} \\
& \mathcal{P}_{t+1}^a \geq c_{\min}^a(\Sigma_t^a), \, \forall a \in \mathcal{A}, \, t \in \mathcal{T} \\
& \mathcal{P}_{t+1}^a \leq c_{\max}^a(\Sigma_t^a), \forall a \in \mathcal{A}, \, t \in \mathcal{T}
\end{aligned}$$

Time-dependent resource allocation

Institute for
Software & Systems
Engineering

Variation of general resource allocation

- Demand is given for a time frame $\mathcal{T}$
- Agents are subject to constraints such as minimal or maximal rates and *inertia*
- Therefore, *proactive* allocation necessary!
- Limited rates of change between consecutive time steps

$$\begin{aligned}
\underset{\mathcal{P}}{\text{minimise}} \quad & \sum_{t \in \mathcal{T}} |\mathcal{D}_t - \sum_{a \in \mathcal{A}} \mathcal{P}_t^a| \\
\text{subject to} \quad & \exists [x, y] \in L^a : x \leq \mathcal{P}_t^a \leq y, \, \forall a \in \mathcal{A}, \, t \in \mathcal{T} \\
& \mathcal{P}_{t+1}^a \geq c_{\min}^a(\Sigma_t^a), \, \forall a \in \mathcal{A}, \, t \in \mathcal{T} \\
& \mathcal{P}_{t+1}^a \leq c_{\max}^a(\Sigma_t^a), \forall a \in \mathcal{A}, \, t \in \mathcal{T}
\end{aligned}$$

For instance, if $\mathcal{P}_{t+1}^a \leq \mathcal{P}_t^a + 30$, then $\langle 50, 80, 110 \rangle$✓ but $\langle 50, 90, 125 \rangle$×

Roadmap

Institute for
Software & Systems
Engineering

Within this realm

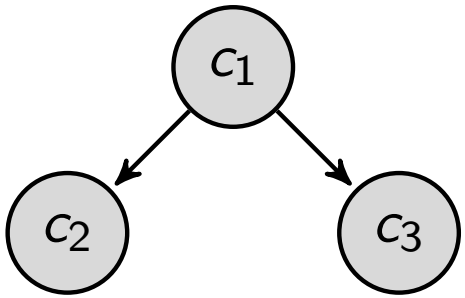
- A formalism to deal with over-constrained problems and specify *preferences*
- A framework to *synthesize* a set of constraint models into one optimization problem
- Algorithms to *abstract* a constraint model from a set of constraint models
- Learning-based to techniques to *build* models
- Tools to support the *elicitation* of preferences

Constraint Relationships – An example

Institute for Software & Systems Engineering

Schiendorfer et al. (2013); Knapp and Schiendorfer (2014)

- Variables: X, Y, Z
- Domains: $\{0, 1, 2\}$
- Constraints:
  - $c_1 : x + 1 = y$
  - $c_2 : z = y + 2$
  - $c_3 : x + y \leq 3$



- Not all three constraints can be satisfied simultaneously
- E.g. c_2 forces z to be 2 and y to be 0, conflicting with c_1
- We can choose between solutions satisfying $\{c_1, c_3\}$ or $\{c_2, c_3\}$
- How to settle this conflict?

Dealing with Heterogeneity

Institute for Software & Systems Engineering

Table : Different cold and hot start-up times in h for power plant types Jarass and Obermair (2012); Mayer et al. (2013)

Plant type	Cold start-up (down $>$ 48h)	Hot start-up (down $<$ 8h)
Black coal	4 – 5	2
Brown coal	6 – 8	2 – 4
Gas turbine	0.5	0.25
Photothermal	4 – 5	2

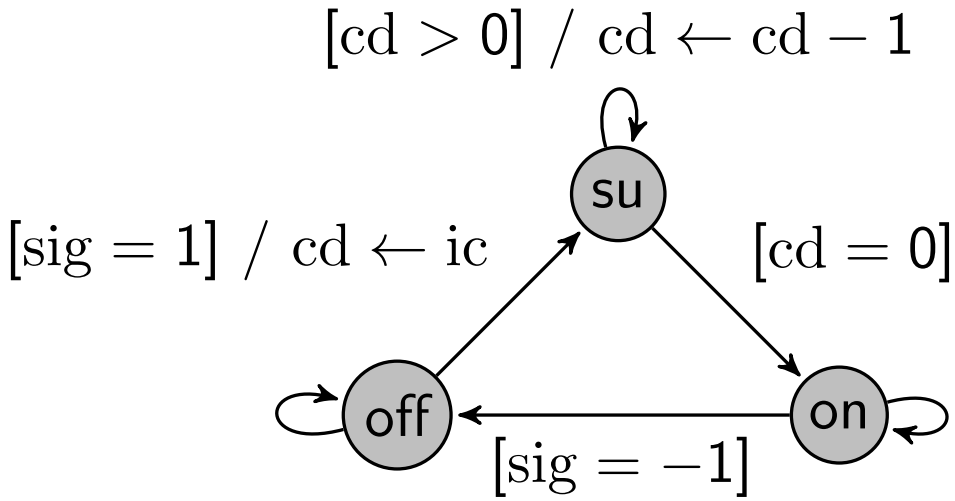
Dealing with Heterogeneity

Institute for Software & Systems Engineering

Table : Different cold and hot start-up times in h for power plant types Jarass and Obermair (2012); Mayer et al. (2013)

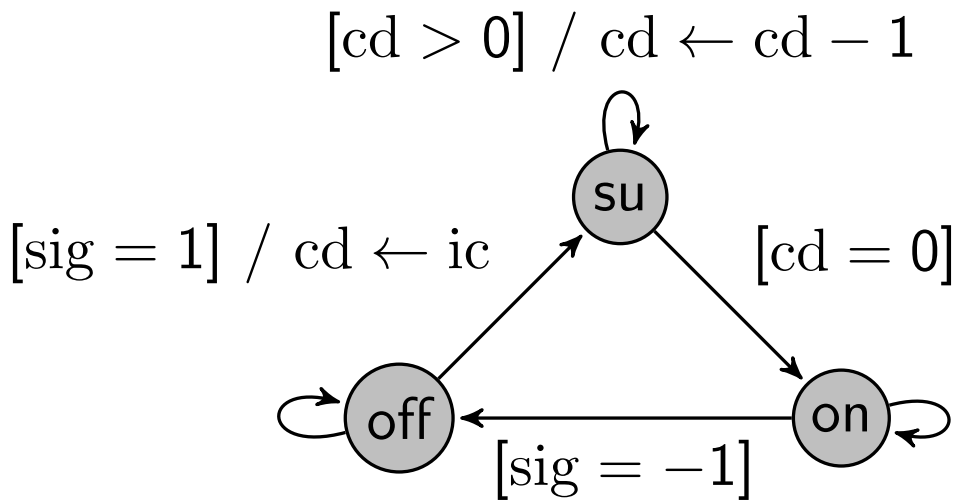
Plant type	Cold start-up (down $>$ 48h)	Hot start-up (down $<$ 8h)
Black coal	4 – 5	2
Brown coal	6 – 8	2 – 4
Gas turbine	0.5	0.25
Photothermal	4 – 5	2

Modelling variable start-up times



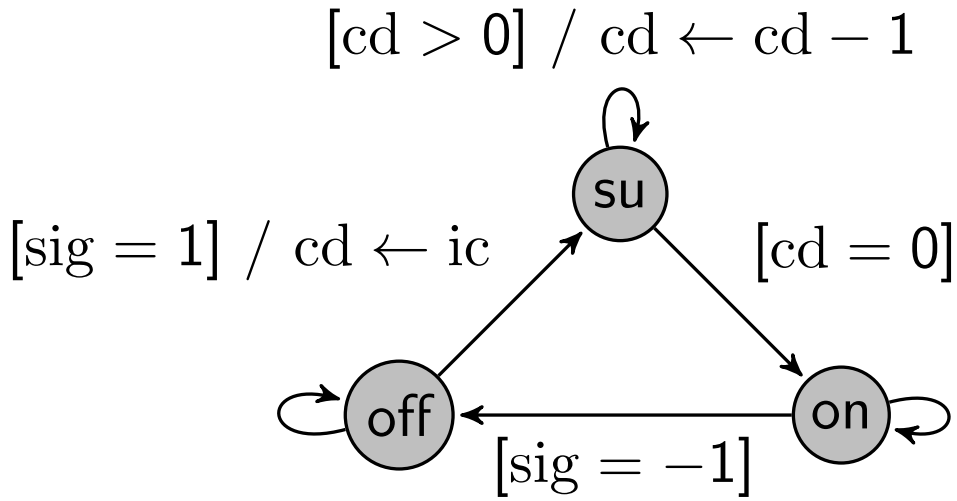
Modelling variable start-up times

Institute for
Software & Systems
Engineering



How to manage both kinds of plants (and others)?

Modelling variable start-up times

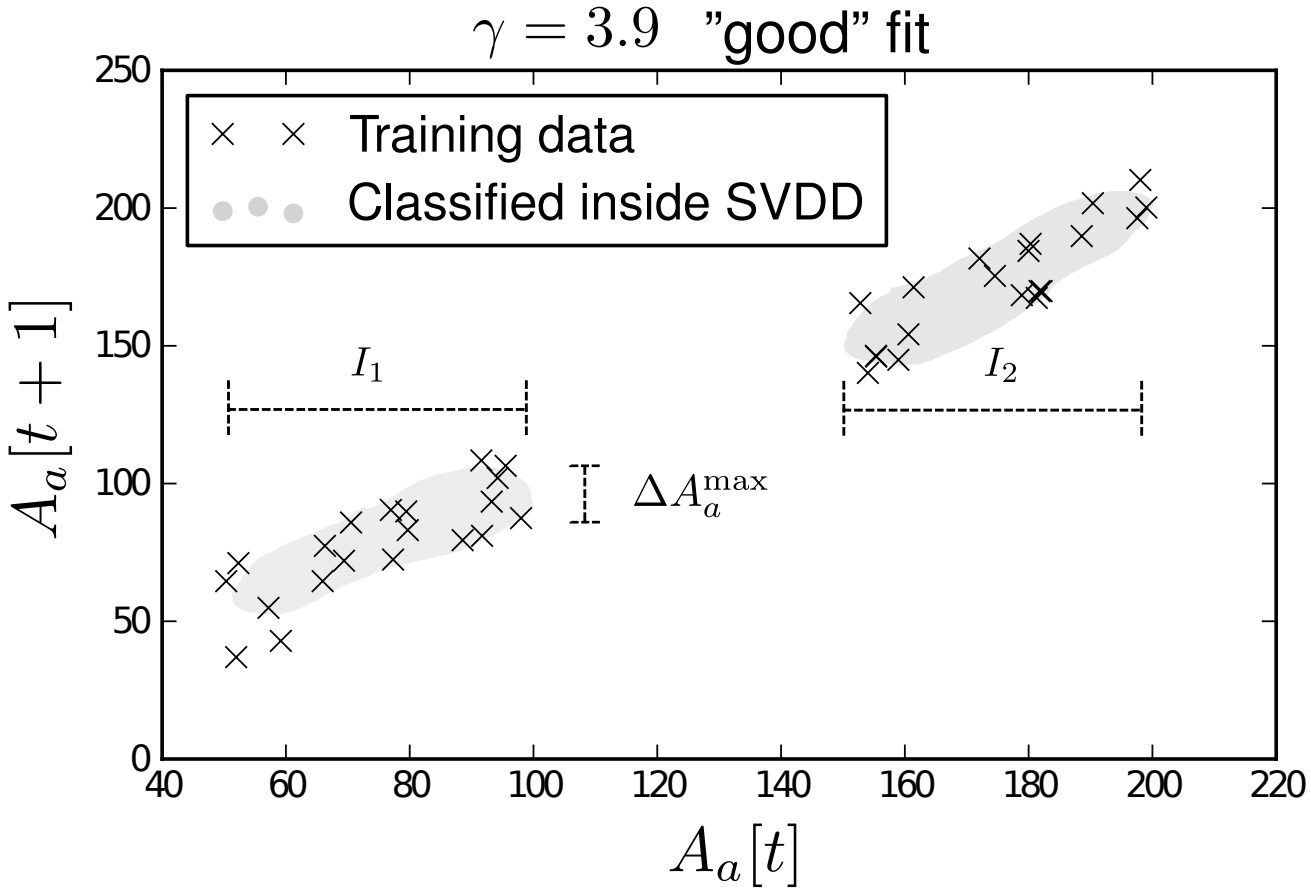


How to manage both kinds of plants (and others)?
→ Separation of shared or *public* interface variables (production) and *private* variables modelling internal aspects (e.g., for the current down-time, start-up duration at this particular down-time, etc.)
Schiendorfer et al. (2014)

Learning Models from Data

Institute for Software & Systems Engineering

First Approach with Support Vector Data Description



Preference Elicitation

Institute for
Software & Systems
Engineering

- Claim: Use simplicity of constraint relationships to offer a more intuitive way of expressing preferences to users
- Show users *solutions* (in terms of violated constraints) and let them state "I like solution A more than B".
- Find a constraint relationship that is consistent with a user's preference decisions
- $\rightarrow$ Tool: Abductive Logical Programming using CHR

Preference Elicitation

Institute for
Software & Systems
Engineering

- Claim: Use simplicity of constraint relationships to offer a more intuitive way of expressing preferences to users
- Show users *solutions* (in terms of violated constraints) and let them state "I like solution A more than B".
- Find a constraint relationship that is consistent with a user's preference decisions
- $\rightarrow$ Tool: Abductive Logical Programming using CHR

```
?- worsens([a,b], [b,c]), worsens([b], [a]).
crbetter(c,b)
crbetter(a,b)
crbetter(c,a)
```

Future work

Institute for Software & Systems Engineering

Theoretical

- Theoretical relationships of constraint relationships (and partial valuation structures) to other frameworks
- Abstraction algorithms for other application domains (e.g., energy consumers)
- Statistical model checking to obtain design-time guarantees

Practical

- Use automata-based models for simulation and engineering of constraint models
- Implementation of a reachability checker for hybrid automata
- Learning/abductive reasoning for preference elicitation
- Investigate alternative machine learning algorithms for model building
- Combine multiple paradigms: constraint-based models, automatas, machine learning models and optimization problem

References I

L. Jarass and G.M. Obermair. *Welchen Netzumbau erfordert die Energiewende?: Unter Berücksichtigung des Netzentwicklungsplans Strom 2012 (in German)*. MV-Wissenschaft. Monsenstein und Vannerdat, 2012. ISBN 9783869916415.

Alexander Knapp and Alexander Schiendorfer. Embedding Constraint Relationships into C-Semirings. Technical Report 2014-03, Institute for Software and Systems Engineering, University of Augsburg, 2014.

Johannes N. Mayer, Niklas Kreifels, and Bruno Burger. Kohleverstromung zu Zeiten niedriger Börsenstrompreise, August 2013.

References II

Florian Nafz, Jan-Philipp Steghöfer, Hella Seebach, and Wolfgang Reif. Formal modeling and verification of self-* systems based on observer/controller-architectures. In Javier Cámara, Rogério Lemos, Carlo Ghezzi, and Antónia Lopes, editors, *Assurances for Self-Adaptive Systems*, volume 7740 of *Lecture Notes in Computer Science*, pages 80–111. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36248-4. doi: 10.1007/978-3-642-36249-1_4. URL http://dx.doi.org/10.1007/978-3-642-36249-1_4.

Andres J. Ramirez and Betty H. C. Cheng. Design patterns for developing dynamically adaptive systems. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '10, pages 49–58, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-971-8. doi: 10.1145/1808984.1808990. URL http://doi.acm.org/10.1145/1808984.1808990.

References III

Alexander Schiendorfer, Jan-Philipp Steghöfer, Alexander Knapp, Florian Nafz, and Wolfgang Reif. Constraint Relationships for Soft Constraints. In Max Bramer and Miltos Petridis, editors, *Proc. 33rd SGAI Int. Conf. Innovative Techniques and Applications of Artificial Intelligence (AI'13)*, pages 241–255. Springer, 2013.

Alexander Schiendorfer, Jan-Philipp Steghöfer, and Wolfgang Reif. Synthesis and Abstraction of Constraint Models for Hierarchical Resource Allocation Problems. In *Proc. 6th Int. Conf. Agents and Artificial Intelligence (ICAART'14), Vol. 2*, pages 15–27. SciTePress, 2014.