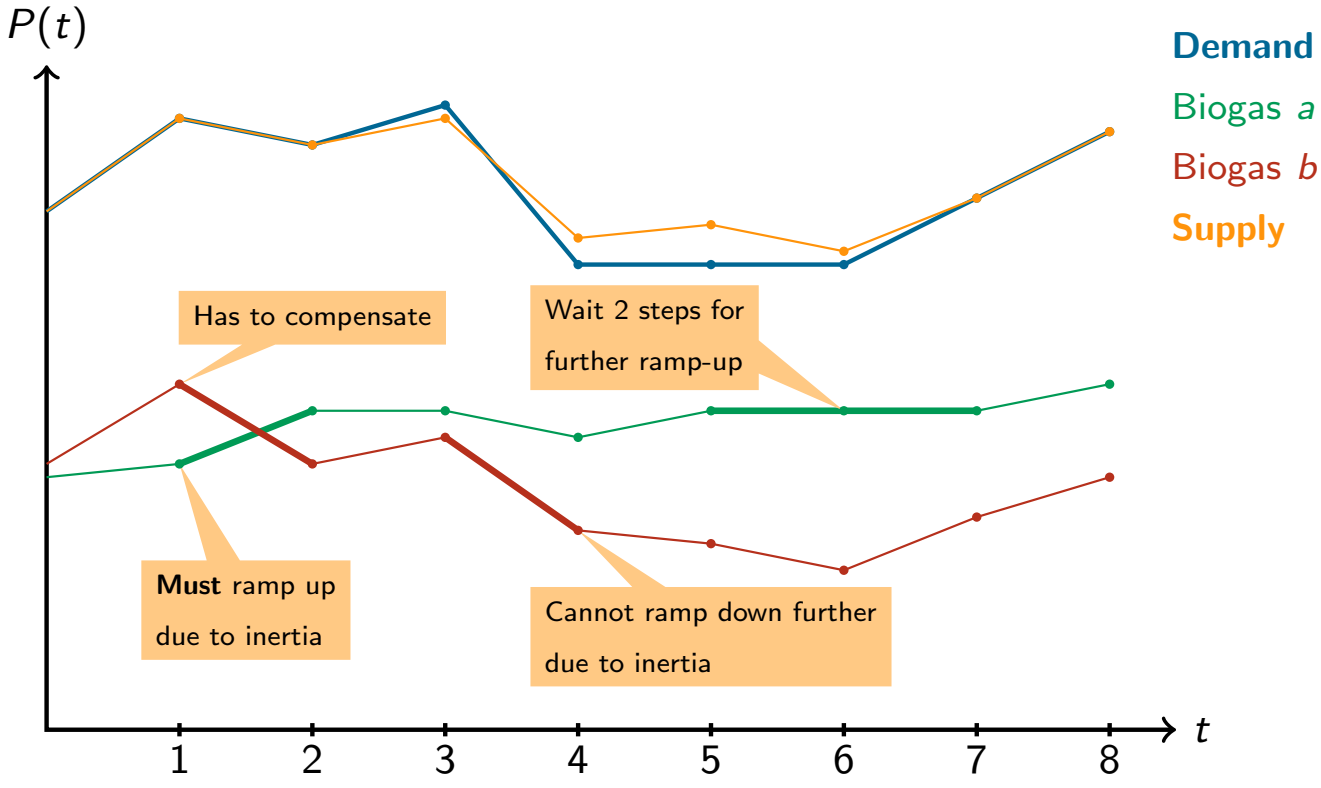
Soft and Hard Constraints in Large Self-Organizing Systems

Alexander Schiendorfer et al.

Institute for
Software & Systems
Engineering

Ziel: Stelle sicher, dass Erzeugung (Supply) und Verbrauch (Demand) in Balance sind.



Hierarchisches Energiemanagement

Regio-zentrale Fahrpläne

ICAART'14, SAOS'14

Marktbasiert

TAAS'15

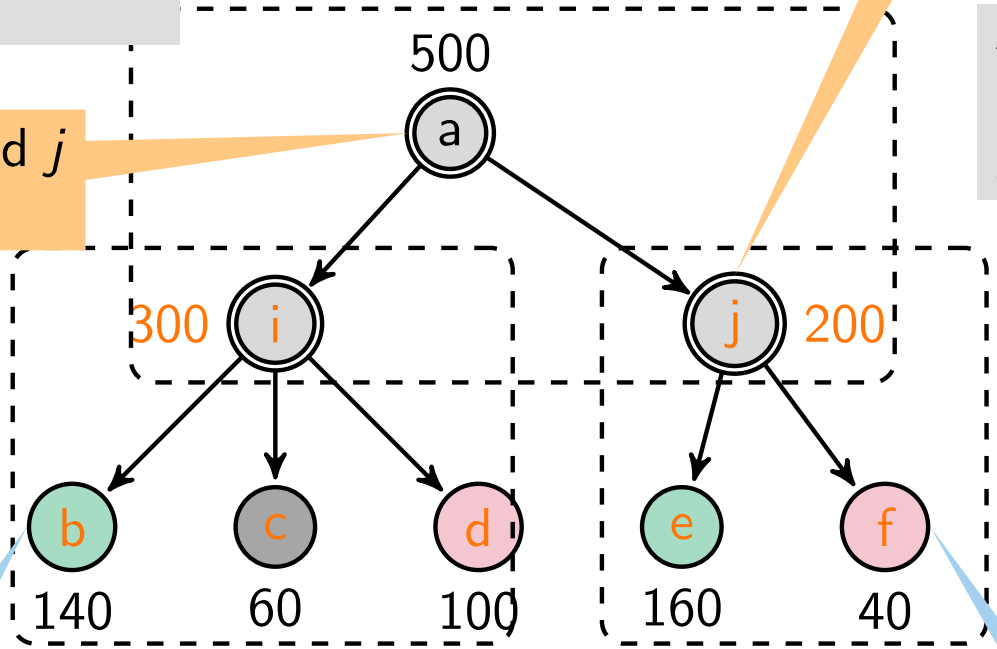
Wie kann ich *e* und *f* repräsentieren?

Abstraktion

ICAART'14, TCCI'15

SASO'15

Was sollen *i* und *j* beisteuern?



Supply Automata

SEN-MAS'14

TCCI'15

Wie vermeide ich meinen Speicher über 90% zu füllen?

Constraint Relationships / PVS

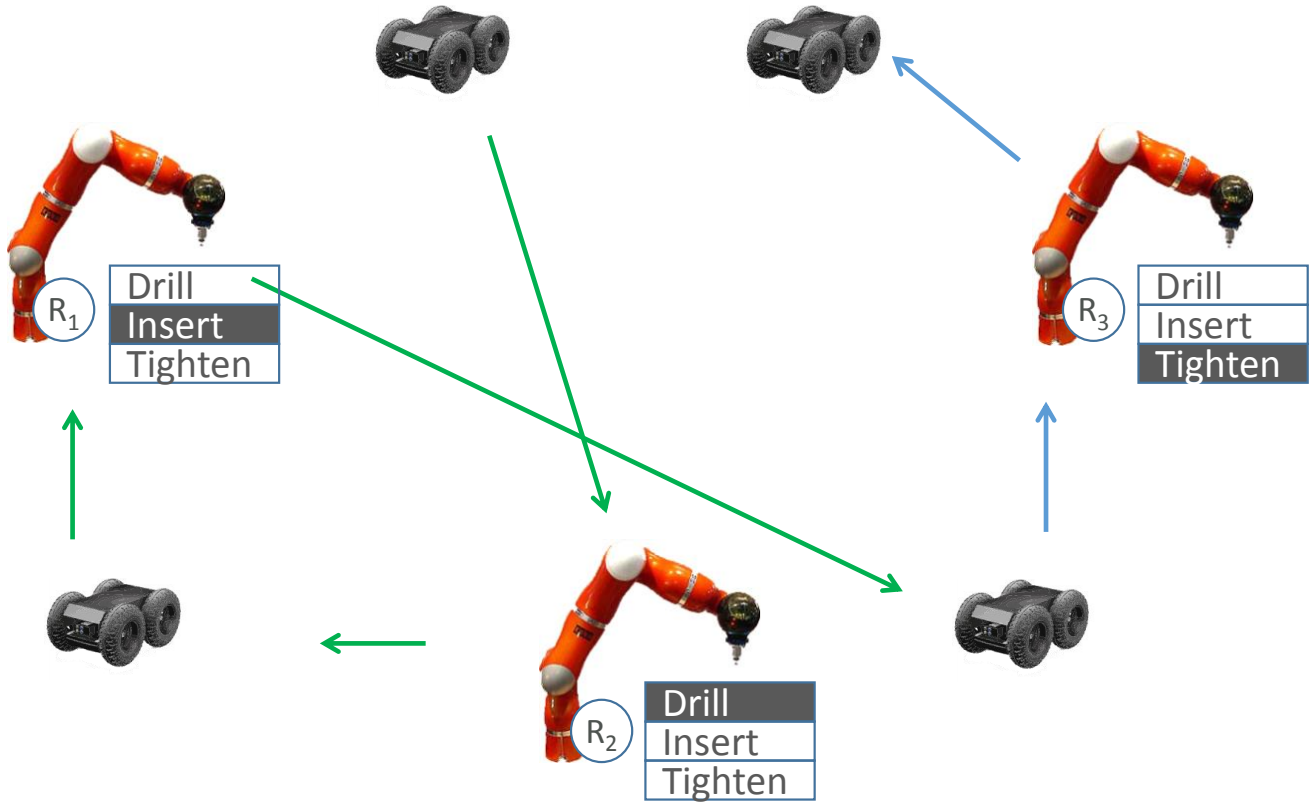
SGAI'13, ICTAI'14

Wirsing'15, Constraints'17

Wie beschreibe ich bevorzugte Abläufe?

Institute for Software & Systems Engineering

Ziel: Weise Tasks an Roboter zu, sodass ein korrekter Ressourcenfluss entsteht



(Seebach et al., 2010)

Constraint Satisfaction Probleme

Institute for Software & Systems Engineering

Ziel: Belege (endlich viele) Variablen aus X mit einem aus (endlich vielen) Werten aus D sodass alle Constraints C erfüllt werden.

Beispiel

- n Roboter, m Tasks
- Gebe jedem Roboter einen *unterschiedlichen* Task, stelle sicher, dass jeder Task belegt ist

```
% problem data
int: n; set of int: ROBOTS = 1..n;
int: m; set of int: TASKS = 1..m;

% decisions
array[ROBOTS] of var TASKS: allocation;

% goal
solve satisfy;

% have robots work on different tasks
constraint alldifferent(allocation);
constraint forall(t in TASKS) (exists(r in ROBOTS) (allocation[r] = t));
```

Constraint Optimization Probleme

Institute for Software & Systems Engineering

Ziel: Suche die beste Belegung, sodass eine **skalare** Zielfunktion $f : [X \to D] \to \mathbb{Z}$ minimiert (oder maximiert) wird.

Beispiel

- n (steuerbare) Supplier, m (steuerbare) Consumer decken Residuallast

```
% problem data
int: n; set of int: SUPPLIERS = 1..n;
array[SUPPLIERS] of int: costs;
int: m; set of int: CONSUMERS = 1..m;
int: residualLoad;

% decisions
array[SUPPLIERS] of var 0..100: supply;
array[CONSUMERS] of var 0..100: demand;

% goal
solve minimize sum(s in SUPPLIERS)(costs[s]*supply[s]);

% have robots work on different tasks
constraint sum(supply) - sum(demand) - residualLoad = 0;
```

Energie: Nicht nur Hard Constraints

Institute for Software & Systems Engineering

Harte Constraints aus Supply Automata:

$$\text{hardBounds} : \forall t \in T, a \in A : m[a][t] = \text{on} \rightarrow P_{\min} \leq S[a][t] \leq P_{\max}$$

Weiche Constraints anlagenspezifisch (z.B. Präferenz für 350 bis 390 KW):

$$\text{ecoSweet}_{\text{bio}} : \forall t \in T : m[\text{biogas}][t] = \text{on} \rightarrow 350 \leq S[\text{biogas}][t] \leq 390$$

oder Änderungsgeschwindigkeit

$$\text{inertia}_{\text{therm}} : \forall t \in T : |S[\text{biogas}][t] - S[\text{biogas}][t+1]| \leq 10$$

Soft Constraint Programming in MiniBrass

Institute for
Software & Systems
Engineering

Constraint Programming

- Deklarative Programmierung (ähnlich SQL, Prolog)
- Trennung von **Modell** und *Algorithmus*
- Geeignet für kombinatorische Probleme unter harten Bedingungen (Physik!)
- Modellierungssprache MiniZinc

Soft Constraint Programming

- Modellierung von **Präferenzen**
- Finde Lösungen, die *so gut wie möglich* sind
- Was bedeutet "gut"?
- Modellierungssprache MiniBrass

Warum MiniZinc?

Institute for
Software & Systems
Engineering

Rationale

Eine Sprache – viele Solver

Unterstützte Solver

- Gecode (CP)
- JaCoP (CP)
- Google Optimization Tools (CP)
- Choco (CP)
- G12 (CP/LP/MIP)

Welche Arten von Soft Constraints?

Institute for Software & Systems Engineering

Was machen wir nun mit $\text{inertia}_{\text{therm}}$ und $\text{ecoSweet}_{\text{bio}}$?

Max-CSP Erfülle so viele Constraints wie möglich (Freuder and Wallace, 1992)

Weighted CSP Minimiere die Summe der verletzten Constraints nach Gewicht (Shapiro and Haralick, 1981)

Fuzzy CSP Erfülle den (minimalen) Erfüllungsgrad (zwischen 0 und 1) über alle Soft Constraints (Ruttkay, 1994)

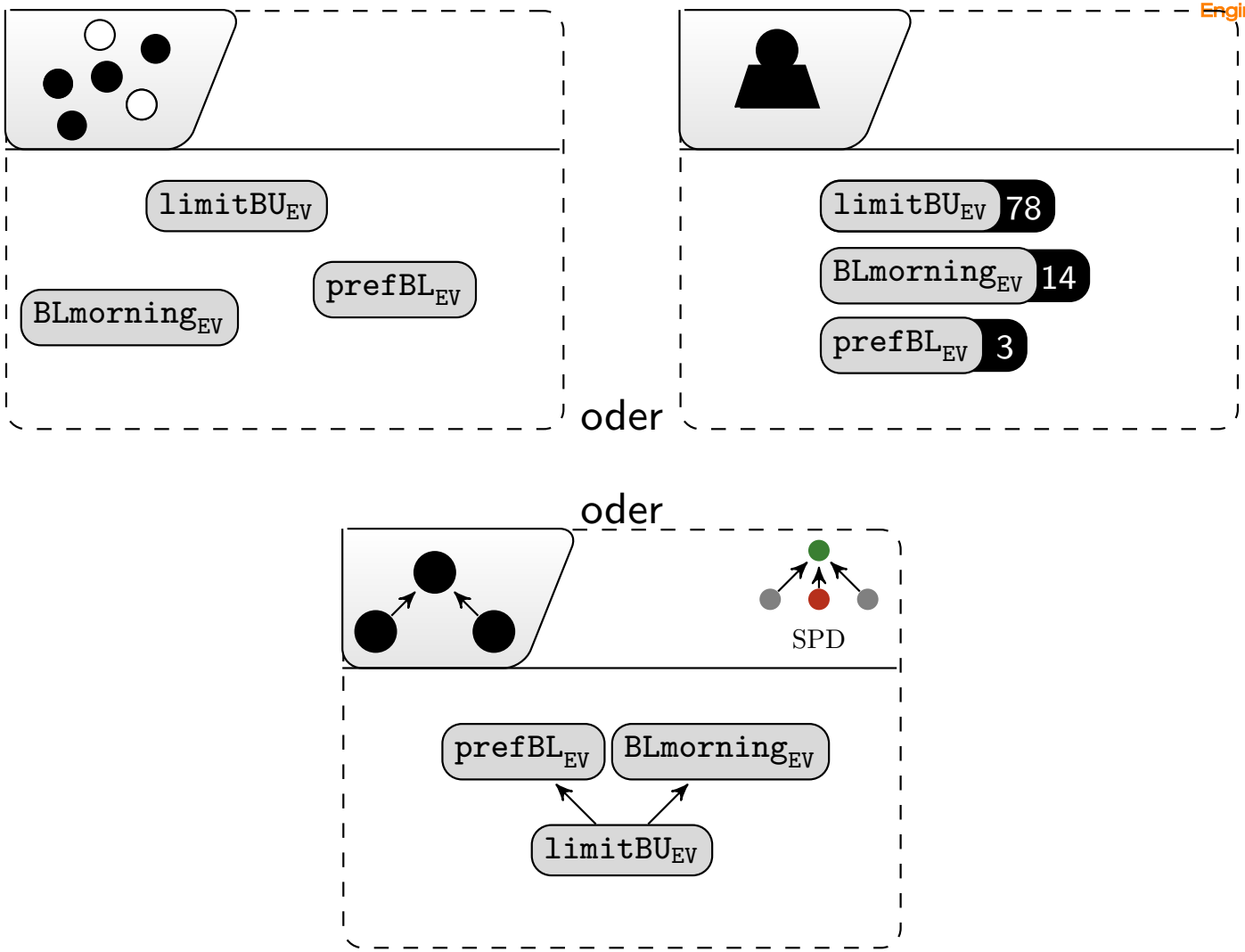
. . . und natürlich

Constraint Preferences (früher *Constraint Relationships*): Definiere partielle Wichtigkeitsordnung über Constraints; erhebe diese zu Mengen von verletzten oder erfüllten Constraints (Schiendorfer et al., 2013)

Zentrale Frage

→ Was sind die Gemeinsamkeiten? Was müssen wir "minimal" tun?

Beispiel: Max-CSP



Algebra zur Hilfe

Institute for Software & Systems Engineering

Wir benötigen ...

- Eine Menge M von Erfüllungsgraden, z.B. $[0.0, 1.0]$ oder $\{0, 1, \ldots k\}$ oder 2^{C_s}.
- Eine partielle Ordnung $\leq_M$ über M: $m \leq_M n$ drückt aus, dass m *schlechter* als n ist
- Eine Kombinationsoperation $\cdot_M$, um zwei Elemente aus M miteinander zu verrechnen
- Ein bestes Element ε_M, um *volle Zufriedenheit* auszudrücken

Gemeinsam nennen wir $(M, \cdot_M, \varepsilon_M, \leq_M)$ eine **partielle Bewertungsstruktur**.

(Gadducci et al., 2013; Schiendorfer et al., 2015)

PVS–Idee

Institute for
Software & Systems
Engineering

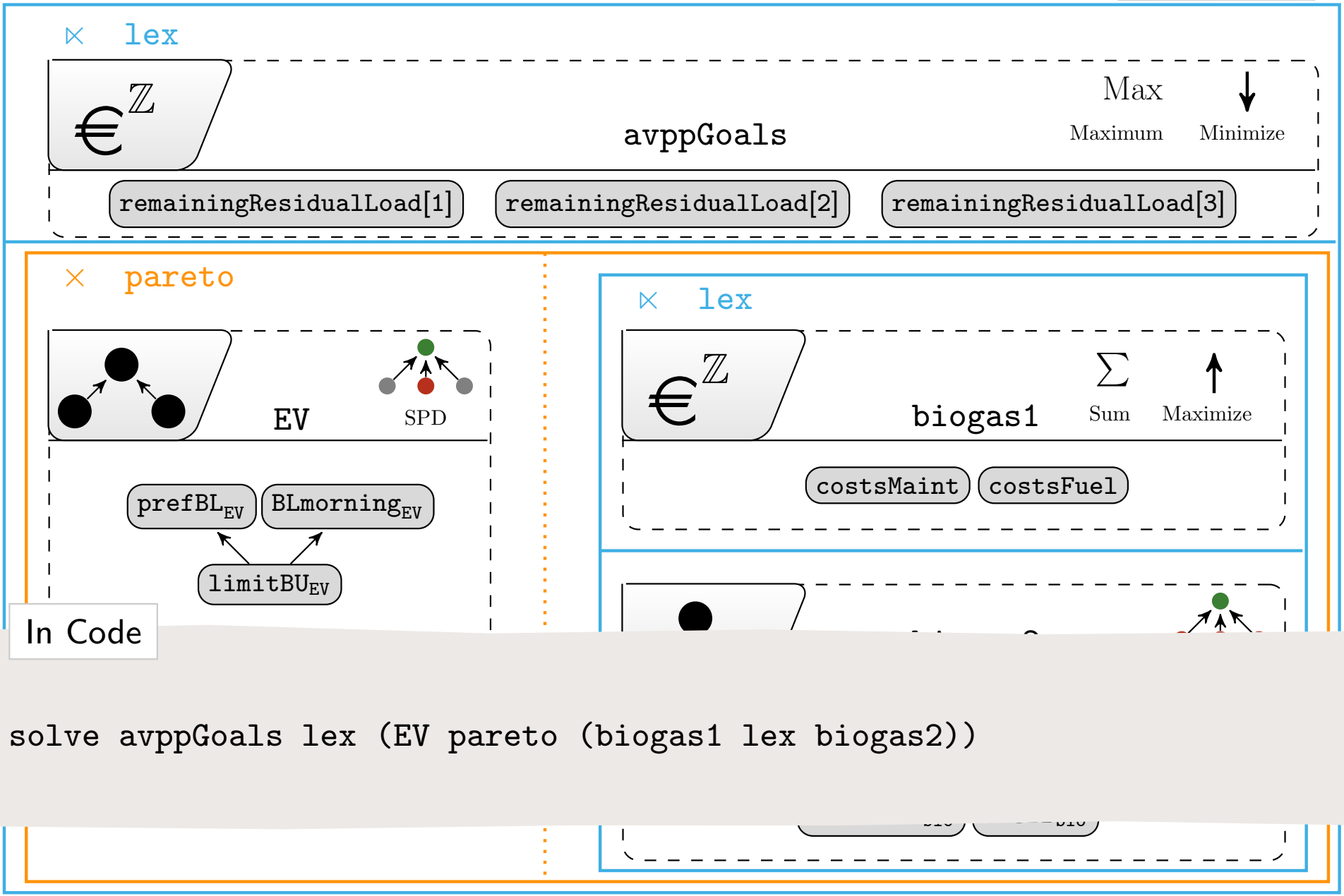
Konkrete PVS-Typen	M	$\cdot_M$	$\leq_M$	ε_M
Weighted CSP (WCSP)	$\mathbb{N}$	$+$	$\geq$	0
Cost Function Network (CFN)	$\{0, \ldots, k\}$	$+/\max$	$\geq$	0
Fuzzy CSP	$[0, 1]$	$\min$	$\leq$	1
Inclusion Max CSP	2^{C_s}	$\cup$	$\supseteq$	$\emptyset$
Constraint Preferences (CP)[1]	$\mathcal{M}^{\text{fin}}(C_s)$	$\uplus$	$\supseteq_{\text{SPD}}$	$\wr\wr$

Hauptidee

Implementiere Lösungsverfahren für Constraint-Probleme, die durch Bewertungsstrukturen geordnet sind. Instantiiere für konkrete Probleme.

[1] C_s is the set of soft constraints, $\supseteq_{\text{SPD}}$ is the SPD-ordering on sets.

Beispiel: Kombinationen (Schiendorfer et al., 2015)

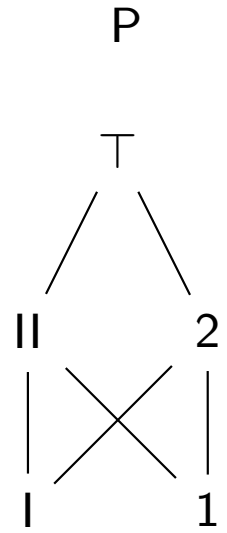


In Code

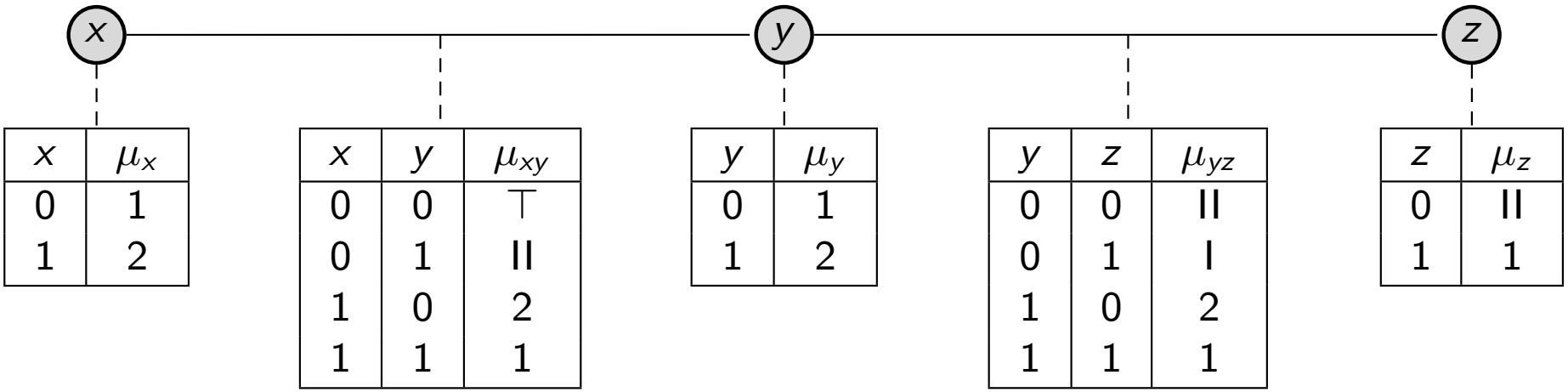
```
solve avppGoals lex (EV pareto (biogas1 lex biogas2))
```

Was müssen wir mindestens tun?

Institute for
Software & Systems
Engineering



SCSP



x	μ_x
0	1
1	2

x	y	μ_{xy}
0	0	⊤
0	1	II
1	0	2
1	1	1

y	μ_y
0	1
1	2

y	z	μ_{yz}
0	0	II
0	1	I
1	0	2
1	1	1

z	μ_z
0	II
1	1

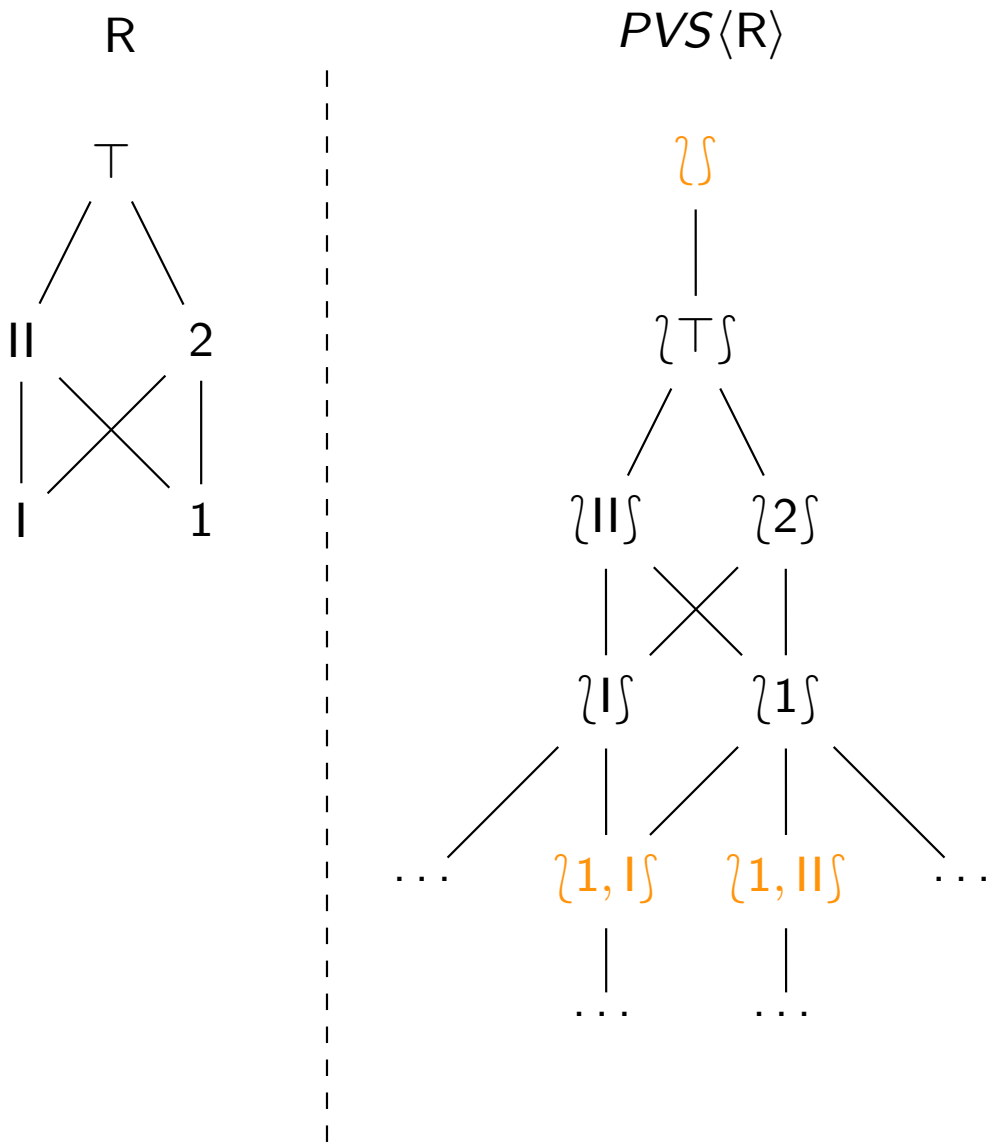
Von POs zu PVS: Skizze

Institute for
Software & Systems
Engineering

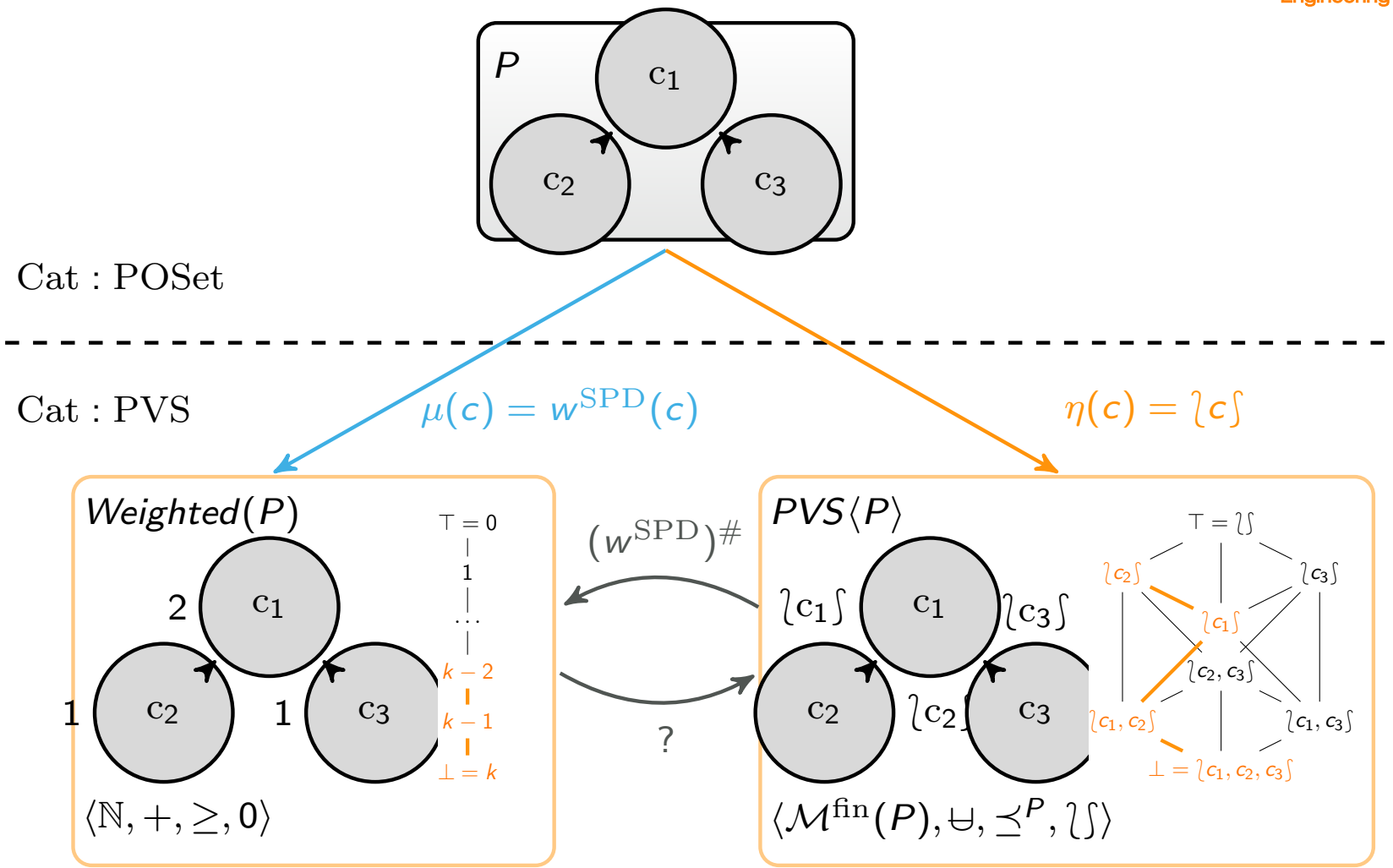
- Angenommen $(P, \leq_P)$
- Wir suchen eine PVS $(M, \cdot_M, \varepsilon_M, \leq_M)$
- Wie Multiplikation konstruieren?
  - Repräsentiere $p \in P$ durch "sich selbst": $p \in M$ ✓
  - Füge neue Elemente "$p \cdot_M q$" für alle Elemente $p, q \in M$ aus ✓
  - Stelle sicher dass Kommutativität und Assoziativität gelten, nicht aber *Idempotenz*: $(p \cdot_M q) \cdot_M r = q \cdot_M (r \cdot_M p)$
  - $\rightarrow$ Wähle *Multimengen* über P als Repräsentant der Produkte aus (Mengen *wären* idempotent: $p \cdot_M p \neq p$ soll aber gelten.
  - Neutrales Element $\wr\int$ für Multiplikation "geschenkt"
- Welche Ordnungsbeziehungen *müssen* gelten?
  - Wenn $p \leq_P q$ dann $\wr p \int \preceq^P \wr q \int$
  - $X \preceq^P \wr\int$, weil ε_M das Maximum in $\leq_M$ sein soll
  - Genannt: Smyth-Ordnung
- Beispiel: $\wr 1, \mathrm{I}, 2\int \preceq^P \wr 2, \mathrm{II}\int$

Was müssen wir mindestens tun?

Institute for
Software & Systems
Engineering



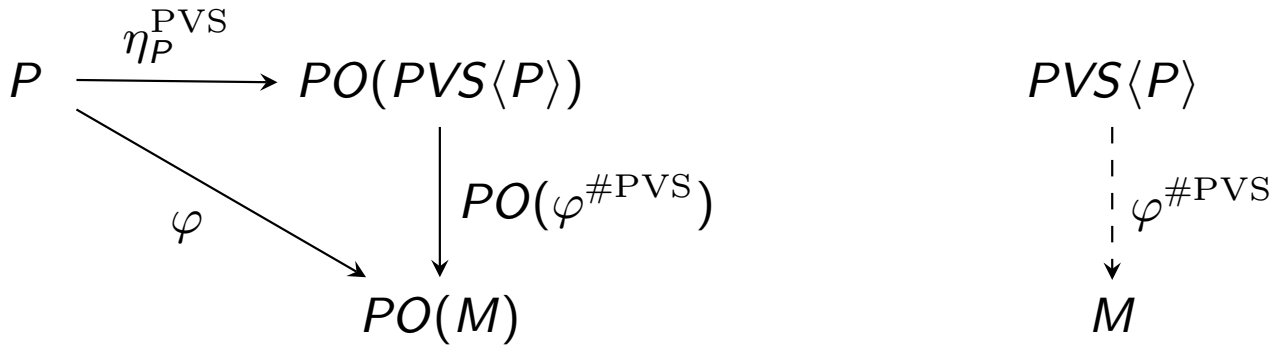
Looking for freedom . . .



Institute for
Software & Systems
Engineering

Lemma (PVS-Freiheit (Knapp and Schiendorfer, 2014))

$PVS\langle P\rangle$ is the free partial valuation structure over the partial order P.



Freie Konstruktionen

- no junk
- no confusion

In MiniBrass

```
type FreePVS = PVSType<mset[maxOccurrences] of 1..maxP> =
  params {
    array[int, 1..2] of 1..nScs: orderRelation;
    int: maxP;
    int: maxPerSc;
    int: maxOccurrences :: default('mbr.nScs * mbr.maxPerSc');
  } in
  instantiates with "../mbr_types/free-pvs-type.mzn" {
    times -> multiset_union;
    is_worse -> isSmythWorse;
    top -> {};
  };
```

```
PVS: fp = new FreePVS("fp") {
   soft-constraint c1: 'embed(x == 4, 3, 3)';
   soft-constraint c2: 'embed(x in {1,3,4}, 2, 3)';
   soft-constraint c3: 'embed(x <= 3, 1, 3)';

   orderRelation : '[| 2, 1 | 3, 1 |]';
   maxP: '3' ;
   maxPerSc : '2';
};

solve fp;
```

Institute for Software & Systems Engineering

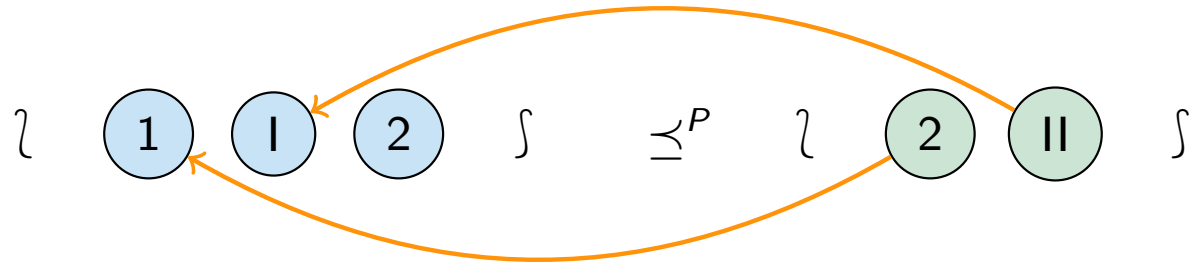
Die Smyth-Ordnung haben wir induktiv definiert:

$$p <_P q \Rightarrow T \uplus \wr p \int \prec^P T \uplus \wr q \int$$
$$T \supset U \Rightarrow T \prec^P U$$

Wie können wir sie nun für einen Constraint-Solver codieren?

Lemma (Witness für $\preceq^P$ (Schiendorfer et al., 2017))

$T \preceq^P U$ gilt genau dann wenn es eine injektive Abbildung $h : S(U) \to S(T)$ (genannt Witness-Funktion) mit $p \leq_P q$ wenn $h(j, q) = (k, p)$ für alle $(j, q) \in S(U)$ gibt.



Evaluations-Setup

Institute for
Software & Systems
Engineering

Evaluation-Probleme

Institute for
Software & Systems
Engineering

Originalprobleme aus der *MiniZinc-Benchmark-Library*; versehen um zusätzliche Soft Constraints als Constraint Preferences

Soft N-Queens N-Damen mit Zusatzbedingungen (z.B. 1 Dame in der Mitte)

Photo Placement Platziere Personen auf Foto neben gewünschten anderen (manche lieber als andere)

Talent Scheduling Plane Szenen und Drehtage; Szenen sollten nicht zu früh beginnen/spät enden; Manche Schauspiele gehen einander lieber aus dem Weg

On-call Rostering Belegungsplan für Bereitschaftsdienste unter arbeitsrechtlichen Bedingungen. Präferenzen für/gegen gewissen Daten bzw. Kooperation mit Kollegen

Multi-Skilled Project Scheduling Weise zeitgebundene Tasks an Agenten mit mehreren Capabilities zu (unter Einhaltung von Präzedenz) – ähnlich zu ODP/COMBO

Eckdaten

Institute for
Software & Systems
Engineering

Konfigurationen	16
Probleme	5
Instanzen	28
Solver	7
Versuchte Probleme	1793
Gelöste Probleme	1289 (71.9%)
Optimal gelöste Probleme	1250 (69.7%)

Kompatibilität

Institute for
Software & Systems
Engineering

	Gecode	JaCoP	OR-Tools	Choco	G12	Toulbar2
Free PVS	✓	✓	x	x	x	x
Constraint Preferences	✓	✓	x	x	x	x
Fuzzy CSP	✓	✓	(✓)	(✓)	(✓)	(✓)
Probabilistic CSP	✓	✓	(✓)	(✓)	(✓)	(✓)
Max CSP	✓	✓	✓	✓	✓	✓
Weighted CSP	✓	✓	✓	✓	✓	✓
Cost Function Networks	✓	✓	✓	✓	✓	✓

- ✓ . . . voll unterstützt
- (✓) . . . teils unterstützt durch Morphismen
- x . . . nicht unterstützt

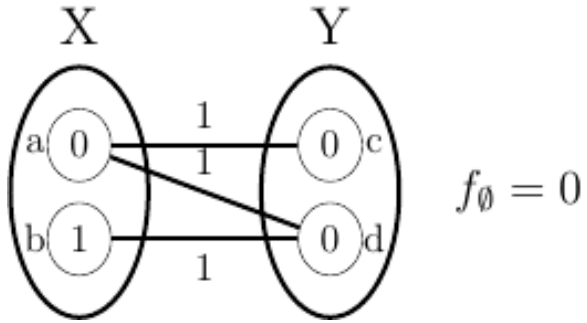
Solver-Vergleich auf Weighted CSP

Institute for Software & Systems Engineering

Klassische Solver

Optimisation Research Group NICTA

Soft Constraint Solver Solver




```
solve ToWeighted(constraintPrefs1);
```

Evaluationsfrage

Wie schnell und effektiv (hinsichtlich des Findens von Optima) können Weighted-Instanzen durch klassische Constraint Solver gegenüber dedizierten Soft Constraint Solvern gelöst werden?

Solver-Vergleich auf Weighted CSP I

Institute for Software & Systems Engineering

Solver	Time (secs)		# Wins	Objective		% Solved	% Optimal
MSPSP (8 instances)							
Gecode	0.32	(1.00)	8	2.50	(0.00)	100.00	100.00
G12	0.32	(1.01)	0	2.50	(0.00)	100.00	100.00
OR-Tools	0.33	(1.05)	0	2.50	(0.00)	100.00	100.00
JaCoP	0.52	(1.73)	0	2.50	(0.00)	100.00	100.00
Choco	0.70	(2.46)	0	2.50	(0.00)	100.00	100.00
Toulbar2	312.56	(1052.07)	0	29.13	(26.63)	0.00	0.00
On-Call Rostering (7 instances)							
Toulbar2	40.73	(1.44)	3	1.57	(0.00)	100.00	100.00
OR-Tools	275.23	(5.55)	2	3.71	(2.14)	100.00	57.14
Gecode	275.23	(5.54)	1	4.57	(3.00)	100.00	57.14
G12	276.36	(5.63)	1	5.57	(4.00)	100.00	57.14
JaCoP	276.63	(5.86)	0	5.14	(3.57)	100.00	57.14
Choco	276.72	(6.26)	0	5.14	(3.57)	100.00	57.14
Photo Placement (3 instances)							
Toulbar2	0.80	(1.11)	0	13.33	(0.00)	100.00	100.00
Choco	0.83	(1.21)	2	25.00	(11.67)	100.00	100.00
OR-Tools	1.49	(1.71)	1	13.33	(0.00)	100.00	100.00
JaCoP	3.18	(3.61)	0	13.33	(0.00)	100.00	100.00
Gecode	22.24	(21.62)	0	13.33	(0.00)	100.00	100.00
G12	27.40	(29.62)	0	13.33	(0.00)	100.00	100.00

Solver-Vergleich auf Weighted CSP II

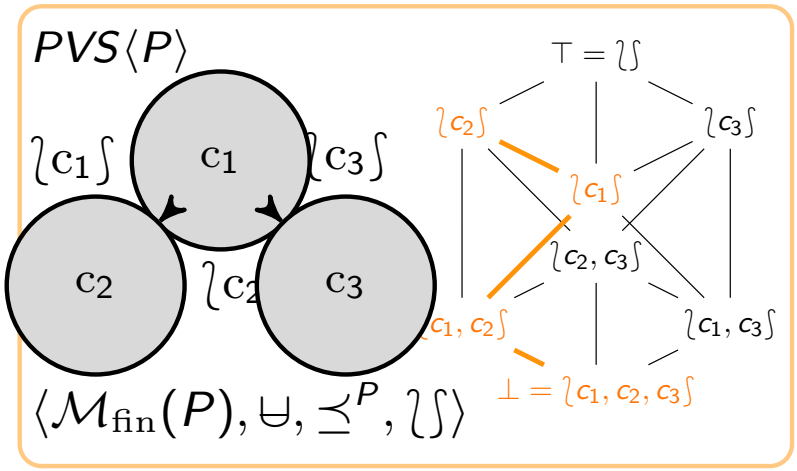
Institute for Software & Systems Engineering

Solver	Time (secs)		# Wins	Objective		% Solved	% Optimal
Soft N-Queens (3 instances)							
OR-Tools	0.03	(1.00)	3	0.33	(0.00)	100.00	100.00
Toulbar2	0.30	(10.43)	0	0.33	(0.00)	100.00	100.00
Choco	0.35	(12.54)	0	0.33	(0.00)	100.00	100.00
JaCoP	57.22	(1707.98)	0	0.33	(0.00)	100.00	100.00
Gecode	210.02	(6266.00)	0	1.67	(1.33)	100.00	66.67
G12	210.02	(6266.14)	0	1.67	(1.33)	100.00	66.67
Talent Scheduling (7 instances)							
OR-Tools	113.29	(1.01)	3	12.29	(0.00)	100.00	85.71
JaCoP	117.71	(1.84)	0	12.29	(0.00)	100.00	85.71
Choco	129.12	(3.27)	1	12.29	(0.00)	100.00	85.71
Toulbar2	158.27	(60.70)	0	28.43	(16.14)	28.57	28.57
Gecode	183.29	(4.70)	3	12.29	(0.00)	100.00	85.71
G12	194.91	(2.87)	0	12.29	(0.00)	100.00	85.71

Vergleich Constraint Preferences vs Weighted

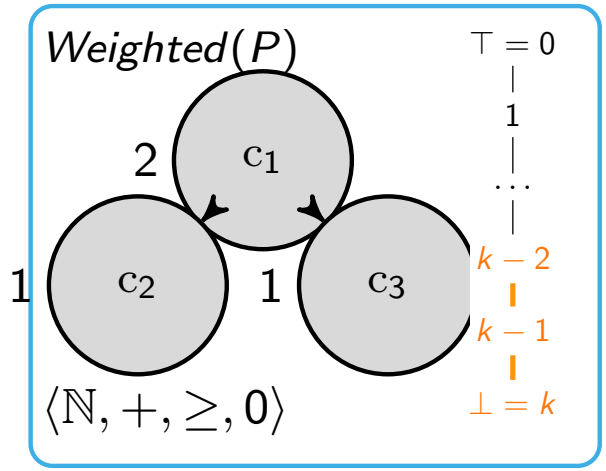
Institute for Software & Systems Engineering

Constraint Preferences




```
solve constraintPrefs1;
```

Weighted




```
solve ToWeighted(constraintPrefs1);
```

Evaluationsfrage

Ist es wesentlich teurer nach Smyth zu optimieren, anstatt ein gewichtetes Problem zu verwenden?

Vergleich Weighted vs Constraint Preferences

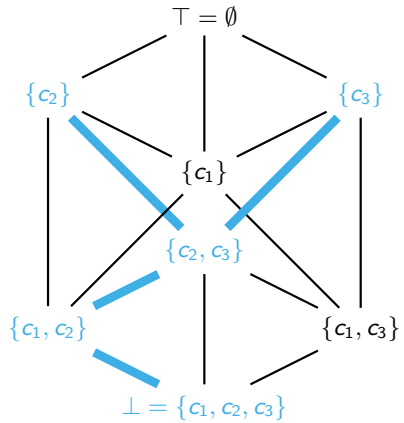
Institute for Software & Systems Engineering

Solver	Time Smyth	Time Weighted	Time Toulbar2	Obj. Weights	Obj. Smyth
MSPSP (6 instances)					
Gecode	12.74	**0.34**	–	2.67	5.50
Native Gecode	7.82	**0.26**	–	2.80	5.80
JaCoP	4.18	**0.45**	–	2.00	6.00
On-Call Rostering (5 instances)					
Gecode	220.46	**133.32**	14.52	3.20	7.20
Native Gecode	192.50	**133.32**	14.52	3.20	25.20
JaCoP	194.06	**135.28**	14.52	3.20	26.80
Photo Placement (3 instances)					
Gecode	6.69	**1.03**	0.68	13.00	13.00
Native Gecode	**9.96**	22.22	0.80	13.33	13.33
JaCoP	15.73	**3.18**	0.80	13.33	13.33
Soft N-Queens (3 instances)					
Gecode	**3.45**	210.02	0.30	1.67	2.00
Native Gecode	**3.49**	210.02	0.30	1.67	1.33
JaCoP	**3.94**	57.22	0.30	0.33	1.00
Talent Scheduling (6 instances)					
Gecode	**7.78**	158.94	–	12.50	14.25
Native Gecode	**13.50**	141.09	–	12.33	14.67
JaCoP	**15.63**	120.42	–	12.33	14.17

Nicht-Dominiert versus Strikte Verbesserung

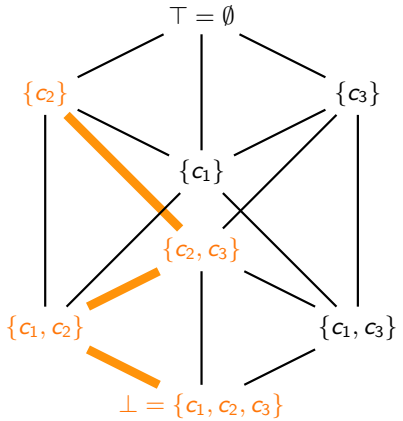
Institute for Software & Systems Engineering

Nicht-Dominiert




```
solve
search pvs_BAB_NonDom();
```

Strikte Verbesserung




```
solve
search pvs_BAB();
```

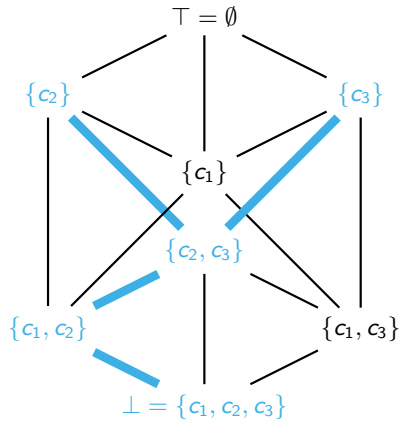
Evaluationsfrage

Wieviel Overhead erzeugt das Suchen aller (unvergleichbar) optimalen Lösungen gegenüber dem Suchein *einer* optimalen Lösung?

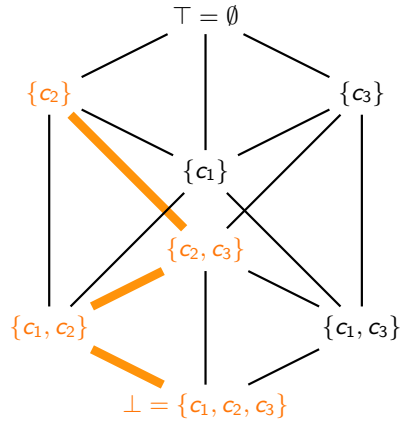
Nicht-Dominiert versus Strikte Verbesserung

Institute for Software & Systems Engineering

Nicht-Dominiert



Strikte Verbesserung



Problem	Time Non-Dominated BaB	Time Strict BaB	Absolute Overhead	Relative Overhead
MSPSP	**7.31**	8.89	−1.58	1.50
On-Call Rostering	329.44	**199.21**	130.23	1.82
Photo Placement	55.09	**7.51**	47.58	9.72
Soft N-Queens	**2.24**	3.65	−1.41	1.91
Talent Scheduling	33.44	**12.24**	21.21	2.30
Overall	102.00	**57.20**	44.80	2.97

Vergleich MIF vs. Standard

Institute for Software & Systems Engineering

```
type WeightedCsp = PVSType<bool, int> =
  params {[..] } in
  instantiates with "soft_constraints/mbr_types/weighted_type.mzn" { [...] }
 offers {
    heuristics -> getSearchHeuristicWeighted;
 };
```

Most-Important First

```
solve
:: pvsSearchHeuristic
search pvs_BAB();
```

Standard-Suche

```
solve
search pvs_BAB();
```

Evaluationsfrage

Kann die generische MIF-Suchheuristik die Suche nach Optima beschleunigen?

Vergleich MIF vs. Standard I

Institute for Software & Systems Engineering

Gruppiert nach Solvern.

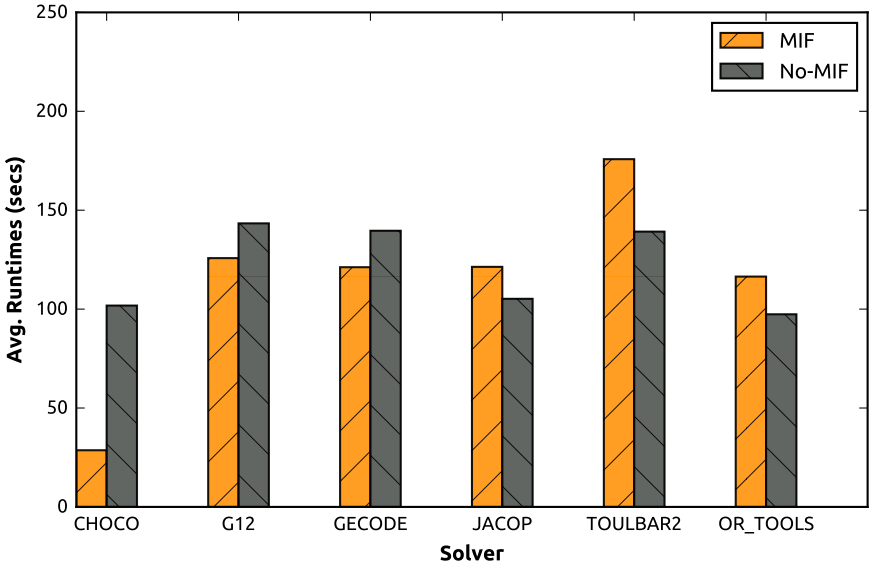
	Choco	G12	Gecode	JaCoP	Toulbar2	OR-Tools
Instances	28	28	28	28	28	28
Runtime difference	−73.14	−17.57	−18.42	16.15	36.63	19.05
Ratio MIF wins	0.64	0.32	0.29	0.46	0.57	0.32

Gruppiert nach Problemen.

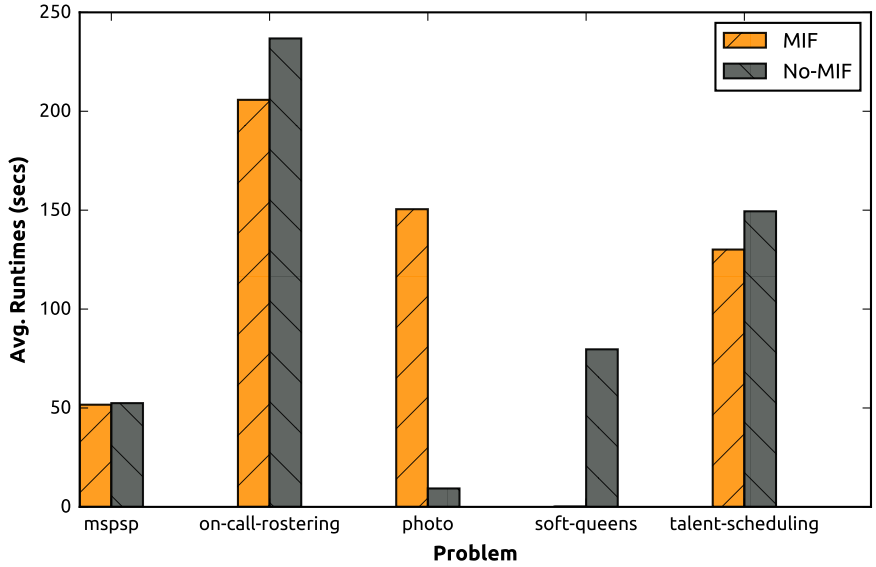
	MSPSP	On-Call Rostering	Photo Placement	Soft N-Queens	Talent Scheduling
Instances	48	42	18	18	42
Runtime difference	−0.80	−31.04	141.17	−79.51	−19.34
Ratio MIF wins	0.42	0.55	0.06	0.56	0.45

Vergleich MIF vs. Standard II

Institute for
Software & Systems
Engineering



(a) Runtimes grouped by solver for MIF on/off.



(b) Runtimes grouped by problem for MIF on/off.

References I

Freuder, E. C. and Wallace, R. J. (1992).
Partial Constraint Satisfaction.
Artif. Intell., 58(1–3):21–70.

Gadducci, F., Hölzl, M., Monreale, G., and Wirsing, M. (2013).
Soft constraints for lexicographic orders.
In Castro, F., Gelbukh, A., and González, M., editors, *Proc. 12th Mexican Int. Conf. Artificial Intelligence (MICAI'2013)*, Lect. Notes Comp. Sci. 8265, pages 68–79. Springer.

Knapp, A. and Schiendorfer, A. (2014).
Embedding Constraint Relationships into C-Semirings.
Technical Report 2014-03, Institute for Software and Systems Engineering, University of Augsburg.
`http://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/2684`.

References II

Ruttkay, Z. (1994).
Fuzzy constraint satisfaction.
In *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, pages 1263–1268. IEEE.

Schiendorfer, A., Knapp, A., Anders, G., and Reif, W. (2017).
Minibrass: Soft constraints for minizinc.
Constraints, 22(3):377–402.

Schiendorfer, A., Knapp, A., Steghöfer, J.-P., Anders, G., Siefert, F., and Reif, W. (2015).
Partial Valuation Structures for Qualitative Soft Constraints.
In Nicola, R. D. and Hennicker, R., editors, *Software, Services and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Emeritation*, Lect. Notes Comp. Sci. 8950. Springer.

References III

Schiendorfer, A., Steghöfer, J.-P., Knapp, A., Nafz, F., and Reif, W. (2013).
Constraint Relationships for Soft Constraints.
In Bramer, M. and Petridis, M., editors, *Proc. 33rd SGAI Int. Conf. Innovative Techniques and Applications of Artificial Intelligence (AI'13)*, pages 241–255. Springer.

Seebach, H., Nafz, F., Steghofer, J.-P., and Reif, W. (2010).
A software engineering guideline for self-organizing resource-flow systems.
In *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 194–203. IEEE.

Shapiro, L. G. and Haralick, R. M. (1981).
Structural descriptions and inexact matching.
IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 504–519.