TECHNISCHE UNIVERSITÄT DARMSTADT

Lineares System mit quadratischen Kosten

- Betrachtet wird folgender Spezialfall:

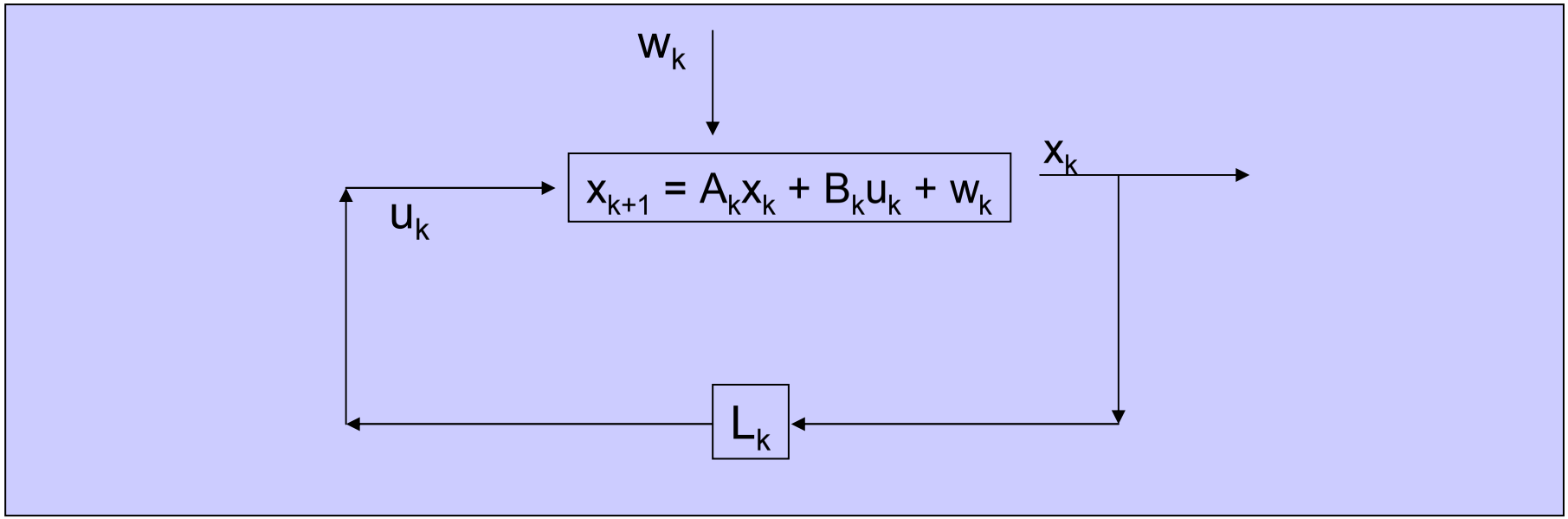
  $$x_{k+1} = A_k x_k + B_k u_k + w_k,\ k = 0,\ldots,N-1$$

  mit den quadratischen Kosten

$$\underset{\substack{w_k \\ k=0,\ldots,N-1}}{E} \left\{ x_N^T Q_N x_N + \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k) \right\}$$

- x_k und u_k sind Vektoren der Dimensionen n und m
- A_k, B_k, Q_k, R_k sind Matrizen mit passenden Dimensionen
- Q_k positiv semidefinit symmetrisch
- R_k positiv definit symmetrisch w_k seien unabhängig zufällig
- w_k habe Durchschnittswert 0

Einführung, Beispiele



Linearer Feedback-Controller für linear-quadratisches System

TECHNISCHE UNIVERSITÄT DARMSTADT

Lineares System mit quadratischen Kosten

→ Populäre Formulierung von Steuerungsproblemen.

1. die Kostenfunktion $J_k(x_k)$ ist quadratisch und die optimale Einstellungen der Kontrollvariablen u_k bekommt man durch Anwendung einer linearen Funktion auf den Zustand x_k.
2. Problem, geeignete Steuerung zu finden ist damit in P (quadratische Zeit) aber:
   Papadimitriou: R nonpositive-definite => Problem wird PSPACE-hard

   Herleitung zu 1.: Anwenden des DP-Algorithmus liefert:

$$J_N(x_N) = x_N^T Q_N x_N,$$
$$J_k(x_k) = \min_{u_k} \underset{w_k}{E}\left\{x_k^T Q_k x_k + u_k^T R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k)\right\}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Lineares System mit quadratischen Kosten

→ Einsetzen von hinten ergibt:

$$J_N(x_N) = x_N^T Q_N x_N,$$
$$J_k(x_k) = \min_{u_k} E_{w_k}\left\{x_k^T Q_k x_k + u_k^T R_k u_k + J_{k+1}(A_k x_k + B_k u_k + w_k)\right\}$$

→ insbesondere für k = N-1:

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1}} E_{w_{N-1}}\left\{\begin{array}{l} x_{N-1}^T Q_{N-1} x_{N-1} + u_{N-1}^T R_{N-1} u_{N-1} + \\ \left(A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + w_{N-1}\right)^T Q_N \left(A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + w_{N-1}\right) \end{array}\right\}$$

→ Durch Ausmultiplizieren des unteren Terms und Elimination des Terms $E\{w^T_{N-1} Q_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1})\}$, da $E\{w_{N-1}\}=0$, ergibt sich:

$$J_{N-1}(x_{N-1}) = x_{N-1}^T Q_{N-1} x_{N-1} + \min_{u_{N-1}}\left[\begin{array}{l} u_{N-1}^T R_{N-1} u_{N-1} + \\ u_{N-1}^T B_{N-1}^T Q_N B_{N-1} u_{N-1} + 2x_{N-1}^T A_{N-1}^T Q_N B_{N-1} u_{N-1} \end{array}\right]$$

$$+ x_{N-1}^T A_{N-1}^T Q_N A_{N-1} x_{N-1} A_{N-1} x_{N-1} + E\{w_{N-1}^T Q_N w_{N-1}\}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Lineares System mit quadratischen Kosten

→ Durch Bilden der totalen Ableitung nach u_{N-1}, und durch setzen der Ableitung zu Null ergibt sich:

$$(R_{N-1} + B_{N-1}^T Q_N B_{N-1}) u_{N-1} = - B_{N-1}^T Q_N A_{N-1} x_{N-1}$$

→ Die Matrix, die von links an u_{N-1} anmultipliziert wird, ist positive definit (und somit) invertierbar), weil R_{N-1} positiv definit ist und $B^T_{N-1} Q_N B_{N-1}$ positiv semidefinit ist. Minimierung des Kontrollvektors ergibt:

$$u^*_{N-1} = -(R_{N-1} + B_{N-1}^T Q_N B_{N-1})^{-1} B_{N-1}^T Q_N A_{N-1} x_{N-1}$$

→ Wenn wir diesen Ausdruck in J_{N-1} einsetzen, erhalten wir

$$J_{N-1}(x_{N-1}) = x_{N-1}^T K_{N-1} x_{N-1} + E\{w_{N-1}^T Q_N w_{N-1}\}, \; wobei$$

$$K_{N-1} = A_{N-1}^T (Q_N - Q_N B_{N-1} (B_{N-1}^T Q_N B_{N-1} + R_{N-1})^{-1} B_{N-1}^T Q_N) A_{N-1} + Q_{N-1}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Lineares System mit quadratischen Kosten

→ K_{N-1} ist positiv semidefinit und symmetrisch
- K_{N-1} wird aus symmetrischen Matrizen zusammengesetzt und
- $x^T K_{N-1} x = \min[x^T Q x + u^T R u + (A_{N-1}x+B_{N-1}u)^T Q_N (A_{N-1}x+B_{N-1}u)]$
  Q_{N-1}, R_{N-1} und Q_N sind pos. semidef. also ist die rechte Seite nicht negativ.
- also ist auch J_{N-1} positiv semidefinit

→ durch mehrfaches induktives Einsetzen erhält man

$$\mu^*_k(x_k) = L_k x_k$$
$$\text{mit } L_k = -(B^T{}_k K_{k+1} B_k + R_k)^{-1} B^T{}_k K_{k+1} A_k$$

→ und mit $K_N = Q_N$ und $K_k = A^T{}_k (K_{k+1} - K_{k+1} B_k (B^T{}_k K_{k+1} B_k + R_k)^{-1} B^T{}_k K_{k+1}) A_k + Q_k$

Diskrete Riccati Gleichung

→ und

$$J_0(x_0) = x_0^T K_0 x_0 + \sum_{k=0}^{N-1} E\{w_k^T K_{k+1} w_k\}$$

2-stufige Programme , Beispiel

Farmer – Problem

- 500 ha Land
- mindestens 200t Weizen und 240t Mais werden fürs Vieh benötigt.
- Verkaufspreis von Weizen/Mais = 170/150 Euro/t, für nicht benötigtes Getreide
- Einkaufspreis von Weizen/Mais = 238/210 Euro/t
- Verkaufspreis von Zuckerrüben: 36 Euro/t unter 6000t; 10 Euro ab mehr als 6000t (wegen Agrarregeln in der EU)
- Planzkosten Weizen/Mais/Zuckerrüben = 150/230/260 Euro/ha
- Ertrag in t/ha für Weizen/Mais/Zuckerrüben = 2.5/3/20

x_1 = Land für Weizen; x_2 = Land für Mais; x_3 = Land für Zuckerrüben;
w_1 = verkaufter Weizen in Tonnen; w_2 = verkaufter Mais in Tonnen; w_3 = verkaufte Zuckerüben zu gutem Preis; w_4 = verkaufte Zuckerüben zu schlechtem Preis;
y_1 = Tonnen gekaufter Weizen; y_2 = Tonnen gekaufter Mais

TECHNISCHE UNIVERSITÄT DARMSTADT

2-stufige Programme , Beispiel

$$\min 150x_1 + 230x_2 + 260x_3 + 238y_1 - 170w_1 + 210y_2 - 150w_2 - 36w_3 - 10w_4$$

s.t.

$$x_1+x_2+x_3 \leq 500$$
$$2.5x_1 + y_1 - w_1 \geq 200$$
$$3x_2 + y_2 - w_2 \geq 240$$
$$w_3+w_4 \leq 20x_3$$
$$w_3 \leq 6000$$
$$x_1,x_2,x_3,y_1,y_2,w_1,w_2,w_3,w_4 \geq 0$$

Stochastic Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

2-stufige Programme , Beispiel

Pflanzenart	Weizen	Mais	Zuckerrüben
Anzahl ha	120	80	300
Ertrag in Tonnen	300	240	6000
Verkauf in Tonnen	100	-	6000
Einkauf in Tonnen	-	-	-

Gewinn bei optimaler Lösung: 118.600 Euro

Unser Farmer ist verunsichert: Ertrag hängt doch sehr vom Wetter ab. Annahme, der Ertrag je ha erhöht / erniedrigt sich um 20%:

Pflanzenart	Weizen	Mais	Zuckerr
Anzahl ha	183.33	66.67	250
Ertrag in T.	550	240	6000
Verkauf in T.	350	-	6000
Einkauf in T.	-	-	-

max. Gewinn bei +20%: 167.667 Euro

Pflanzenart	Weizen	Mais	Zuckerr.
Anzahl ha	100	25	375
Ertrag in T.	200	60	6000
Verkauf in T.	-	-	6000
Einkauf in T.	-	180	-

max. Gewinn bei -20%: 59.950 Euro

Stochastic Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

2-stufige Programme , Beispiel

Unser Farmer würde gerne flexibel reagieren können. Entscheidungen für das Land (x_1,x_2,x_3) müssen sofort gefällt werden, aber die anderen Entscheidungen hängen von den Erträgen je ha ab.

→ Bilde 3 Szenarios mit index 1,2,3. Jedes Szenario habe Eintrittswahrscheinlichkeit 1/3.

$$\min\ 150x_1 + 230x_2 + 260x_3$$
$$- 1/3*(170w_{11}-238y_{11}+150w_{21}-210y_{21}+36w_{31}+10w_{41}) \quad \text{Szenario 1}$$
$$- 1/3*(170w_{12}-238y_{12}+150w_{22}-210y_{22}+36w_{32}+10w_{42}) \quad \text{Szenario 2}$$
$$- 1/3*(170w_{13}-238y_{13}+150w_{23}-210y_{23}+36w_{33}+10w_{43}) \quad \text{Szenario 3}$$
$$\text{s.t.}\ \ x_1 + x_2 + x_3 \le 500,\ 3x_1 + y_{11} - w_{11} \ge 200,\ 3.6x_2 + y_{21} - w_{21} \ge 240,$$
$$w_{31} + w_{41} \le 24x_3,\ w_{31} \le 6000,\ 2.5x_1 + y_{12} - w_{12} \ge 200$$
$$3x_2 + y_{22} - w_{22} \ge 240,\ w_{32} + w_{42} \le 20x_3,\ w_{32} \le 6000,$$
$$2x_1 + y_{13} - w_{13} \ge 200,\ 2.4x_2 + y_{23} - w_{23} \ge 240,\ w_{33} + w_{43} \le 16x_3,\ w_{33} \le 6000,$$
$$x, y, w \ge 0$$

2-stufige Programme, Beispiel

	Pflanzenart	Weizen	Mais	Zuckerrüben
1. Stufe	Anzahl ha	170	80	250
s=1 (+20%)	Ertrag in Tonnen	510	288	6000
	Verkauf in Tonnen	310	48	6000
	Einkauf in Tonnen	-	-	-
s=2	Ertrag in Tonnen	425	240	5000
	Verkauf in Tonnen	225	-	5000
	Einkauf in Tonnen	-	-	-
s=3 (-20%)	Ertrag in Tonnen	340	192	4000
	Verkauf in Tonnen	140	-	4000
	Einkauf in Tonnen	-	48	-

Erwarteter Gewinn bei optimaler Lösung: 108.390 Euro

Durchschnittlicher Gewinn der optimalen Einzellösungen: 115.406 Euro
Differenz = 7016 Euro ist „erwarteter Wert von perfekter Information“

Stochastic Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

mehrstufige Programme, formal

$$\min z = c^1 x^1 + E_{\xi^2}\left[\min c^2(\omega)x^2(\omega^2) + \ldots + E_{\xi^H}\left[\min c^H(\omega)x^H(\omega^H)\right]..\right]$$

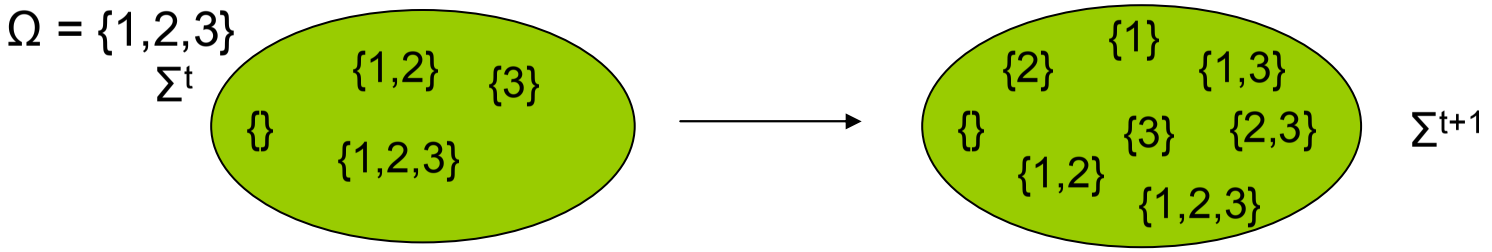
$$\begin{aligned} s.t. \quad & W^1 x^1 = h^1, \\ & T^1(\omega)x^1 + W^2 x^2(\omega^2) = h^2(\omega), \\ & \vdots \\ & T^{H-1}(\omega)x^{H-1}(\omega^{H-1}) + W^H x^H(\omega^H) = h^H(\omega), \\ & x^1 \geq 0; x^t(\omega^t) \geq 0, t = 2,\ldots,H \end{aligned}$$

Stochastic Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

mehrstufige Programme, formal

c_1 Vektor in $\mathbb{Q}^{n1}$, h_1 Vektor in $\mathbb{Q}^{m1}$, $\xi^t(\omega) = (c^t(\omega),h^t(\omega),T_1^{t-1}(\omega),...,T_{mt}^{t-1}(\omega))$
ist ein Zufallsvektor, definiert auf (Ω,Σ^t,P), für alle $t = 2,...,H$. Dabei ist $\Sigma^t \subseteq \Sigma^{t+1}$.
W^t ist eine feste Matrix und spiegelt den festen Recourse wider.



Entscheidungen x hängen von der Historie bis zum Zeitpunkt t ab,
die Historie bezeichnen wir mit ω^t

mehrstufige Programme, deterministisches Äquivalent, Version 1

als dynamisches Programm:

letzte Stufe

$$Q^H(x^{H-1}, \xi^H(\omega)) = \min c^H(\omega) x^H(\omega)$$
$$s.t. \quad W^H x^H(\omega) = h^H(\omega) - T^{H-1}(\omega) x^{H-1},$$
$$x^H(\omega) \geq 0 \quad und$$

Stufen 2,...H-1

$$Q^t(x^{t-1}, \xi^t(\omega)) = \min c^t(\omega) x^x(\omega) + E_\xi \{Q(x^t, \xi)\}$$
$$s.t. \quad W^t x^t(\omega) = h^t(\omega) - T^{t-1}(\omega) x^{t-1},$$
$$x^t(\omega) \geq 0.$$

1. Stufe

Der Wert, den wir suchen ist:

$$\min c^1 x^1(\omega) + E_\xi \{Q(x^1, \xi)\}$$
$$s.t. \quad W^1 x^1 = h^1, \quad x^1 \geq 0.$$

Mehrstufige Programme, deterministisches Äquivalent, Version 2

Als (gemischt ganzzahliges) lineares Programm:

Annahmen:

- klare **Stufung** von Entscheidungsvariablen und Zufallsereignissen

- Zufallsereignisse **unabhängig** von unseren Entscheidungen
  (Das Wetter wird im Normalfall morgen sonnig oder regnerisch sein,
  egal, welche Entscheidung wir in unserem Optimierungsproblem treffen.)

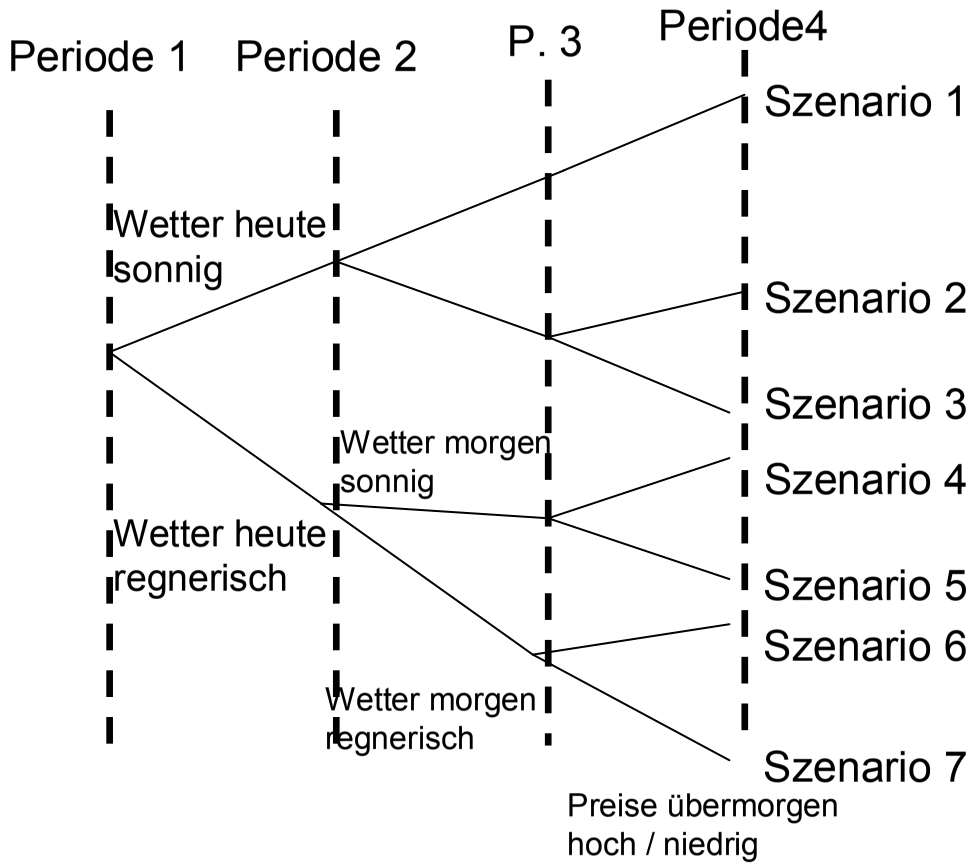
  → essentielle Annahmen für stochastische Programme
  → erlaubt die Aufspaltung von Zufallsprozess und Entscheidungsprozeß
  → führt zu **Szenariobäumen**

- **endliche Anzahl von möglichen Realisierungen** für zukünftigen Ausgängen

TECHNISCHE UNIVERSITÄT DARMSTADT

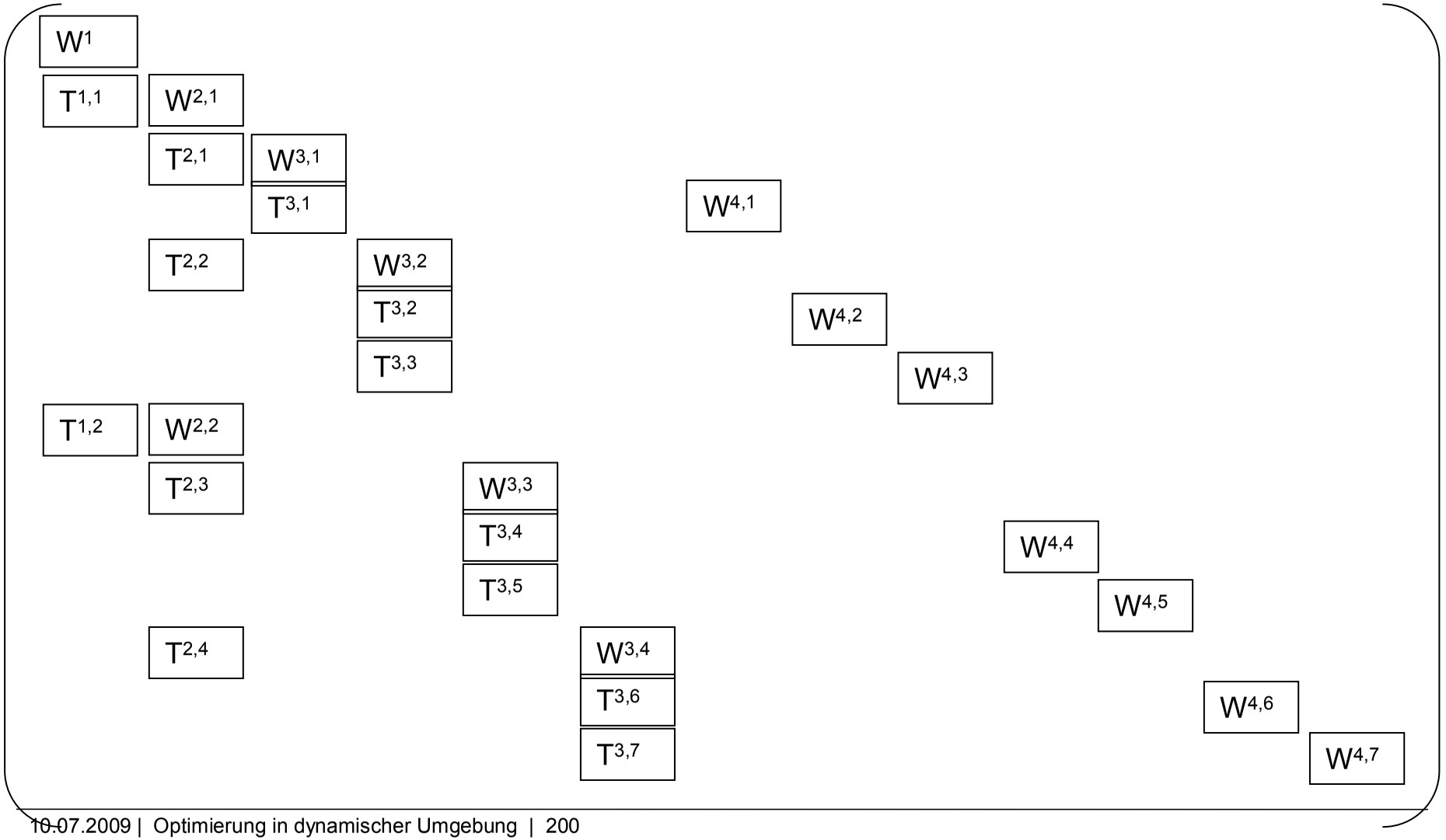
Mehrstufige Programme, deterministisches Äquivalent, Version 2

Szenariobaum:



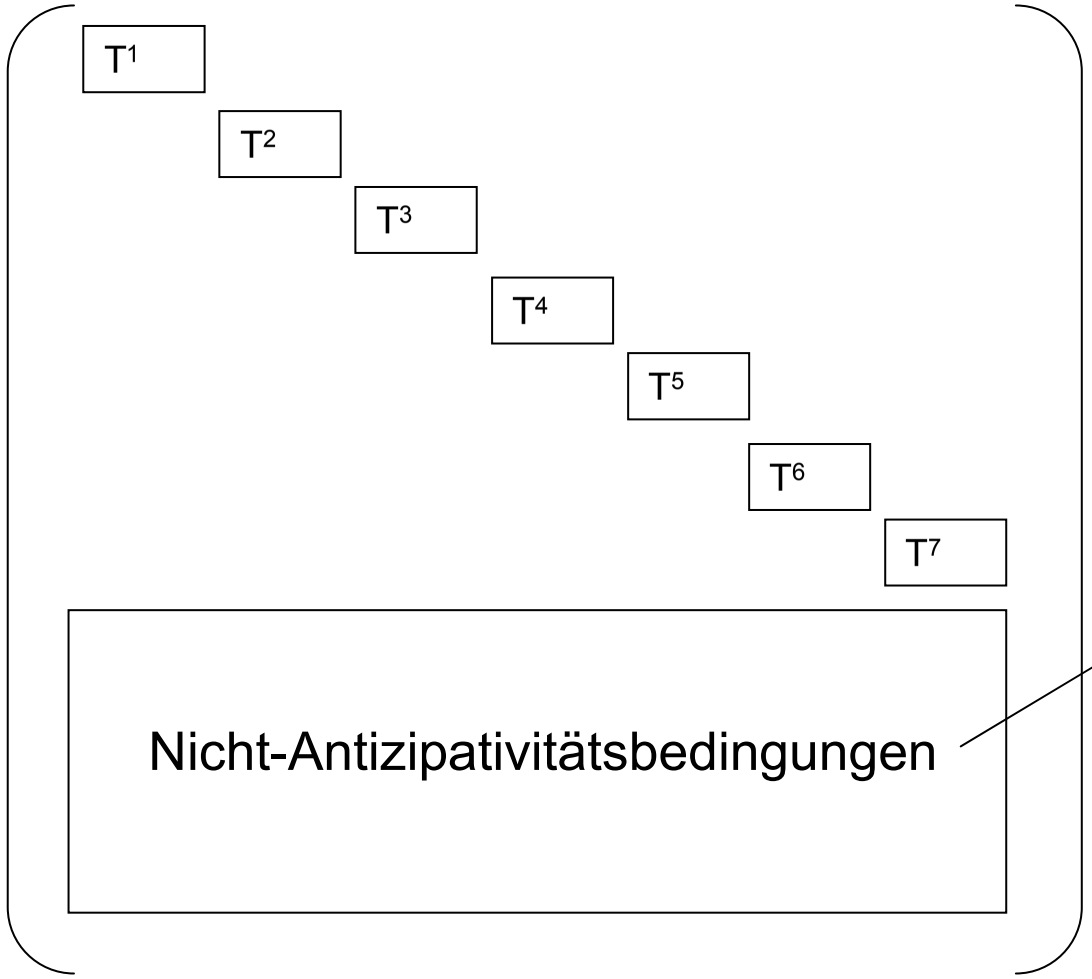
Stochastic Programming

Matrix



Stochastic Programming

Matrix, andere Darstellung



Vervielfachung von Variablen führt zu entkoppelten Szenarien

Einige Variablen müssen in allen (manchen) anderen Szenarien gleich sein.

Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

Eine NCL-“Maschine“ ist definiert über einen Graphen:
gegeben:

- ungerichteter Graph mit
  - nicht-negativen ganzzahligen ***Gewichten*** an den Kanten und
  - ganzzahligen ***minimum*** *in-flow constraints* an Knoten

Eine ***Konfiguration*** der Maschine ist eine Orientierung (Richtung) der Kanten, so dass die Summe der eingehenden Kantengewichte an jedem Knoten mindestens so groß ist, wie der minimum in-flow constraint für den jeweiligen Knoten.

Ein Zug von einer Konfiguration zu einer Anderen ist die Umkehrung einer Kante, so dass die Bedingungen an Konfigurationen eingehalten werden.

Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

Fragestellungen:

- Seien zwei Konfigurationen gegeben. Gibt es eine Folge von Zügen von A nach B?
- Gegeben seien 2 Kanten E_A und E_B und Orientierungen für diese. Gibt es Konfigurationen A und B, so dass E_A die gewünschte Richtung in A und E_B in B haben, und es eine Folge von Zügen von A nach B gibt?

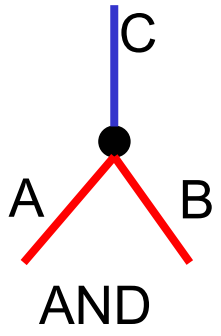
Nondeterministic Constraint Logic (NCL)

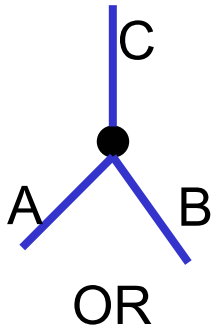
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

AND/OR Constraint Graphen (spezielle NCL-Graphen)

- Knoten mit minimum in-flow constraint 2 und Kanten-Eingangsgewichte 1, 1 und 2 verhalten sich in folgendem Sinne wie ein AND:
  Die Kante mit Gewicht 2 kann nur nach außen zeigen, wenn beide Kanten mit Gewicht 1 nach innen zeigen.
- Knoten mit minimum in-flow constraint 2 und Kanten-Eingangsgewichte 2, 2 und 2 verhalten sich in folgendem Sinne wie ein OR:
  Eine der Kanten kann nur nach außen zeigen, wenn eine der beiden anderen Kanten nach innen zeigt.





Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

Unterschiede NCL und klassische Logik

NCL	klassisch
* nicht-deterministisch	* deterministisch
* nicht a priori festgelegt, was Inputs, und was Outputs sind	* Inputs und Outputs statisch
* kein NOT	* Inverter essentiell

Nicht-Determinismus:
Wenn z.B. zwei Kanten in ein AND hineingehen, ist es der 3. Kante **erlaubt**, nach aussen gedreht zu werden. Es ist **nicht zwingend**. Ob es möglich ist, eine bestimmte Kante nach aussen zu drehen, ist nicht lokal ersichtlich.

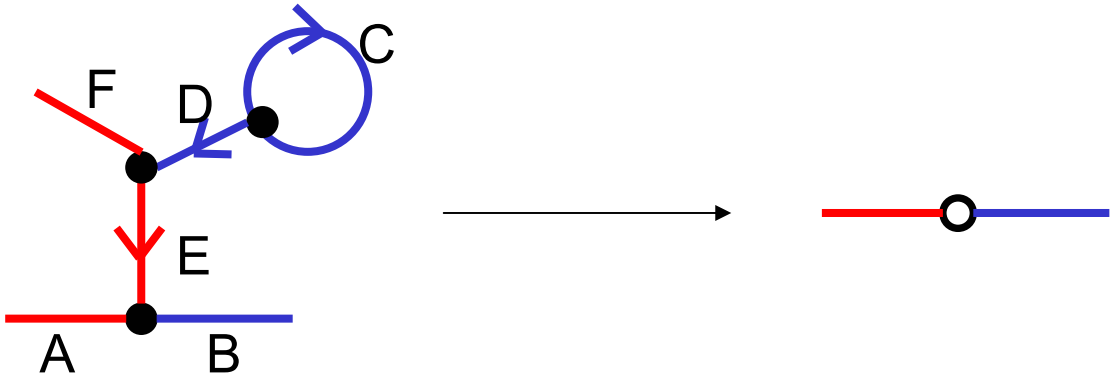
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

Blau-Rot-Adapter

Betrachtet man AND/OR-Graphen als Schaltkreise, möchten wir verschiedene Outputs mit Inputs verbinden. Bei verschiedenartigen Kanten (rot/blau) ist das nicht direkt möglich.



Es gibt eine nicht- verbundene Kante bei F. Etwas Überlegung besagt, solche Kanten treten in Paaren auf: Rote Kanten treten nur in OR-Elementen auf, d.h., F-E-A führt zu einer weiteren freien Kante X. Wir verbinden F mit X.

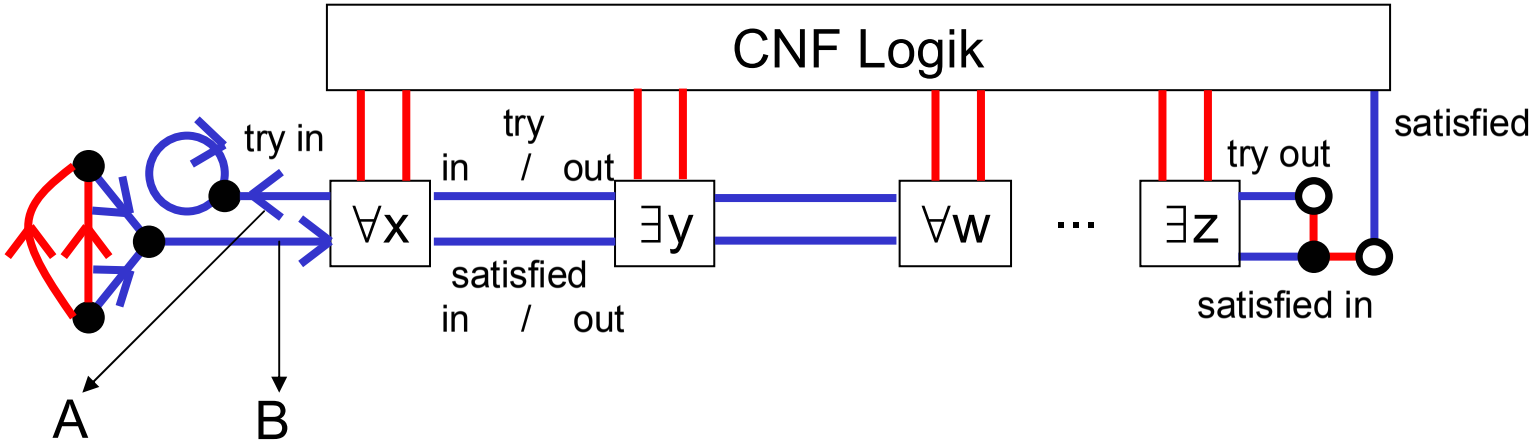
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness (zunächst: PSPACE-hard)

Überblick:



Kann man A umdrehen, so dass sich auch B umdrehen lässt?

Eine Möglichkeit, den Wahrheitsgehalt einer QBF zu finde geht so:
Gehe von außen nach innen durch die Quantifizierer. Bei Allvariaben: setze diese erst auf false, dann auf true und prüfe jeweils rekursiv, ob der Rest erfüllbar ist. Falls ja, return true, sonst false. Bei Existenzvariablen: return true, g.d.w. eine der Zuweisungen (true/false) erfolgreich ist.

Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness (zunächst: PSPACE-hard)

Überblick:

Quantifizierer-Schaltungen:
Wenn ein Quantifizierer **aktiviert** wird
- sind alle Variablen links (im Sinne des Bildes, vorige Folie) von ihm fixiert
- und die Variablen des Quantifizierers und rechts davon sind frei

Ein Quantifizierer kann sich dann und nur dann als **satisfied** melden, wenn die Formel von diesem Quantifizierer bis ganz nach rechts unter der Berücksichtigung der Quantoren erfüllbar ist.

Ein Quantifizierer wird aktiviert, indem seine *try-in*-Kante so gedreht wird, dass sie in den Quantifizierer hineinzeigt. Seine *try-out*-Kante kann nur vom Quantifizierer weg zeigen, wenn die *try-in*-Kante hineinzeigt, und seine Variablen fixiert sind.

Die Variablenzuweisung wird durch zwei herausgehende Kanten (x und $\overline{x}$) repräsentiert, von denen nur eine aus dem Quantifizierer hauszeigen kann.

Nondeterministic Constraint Logic (NCL)

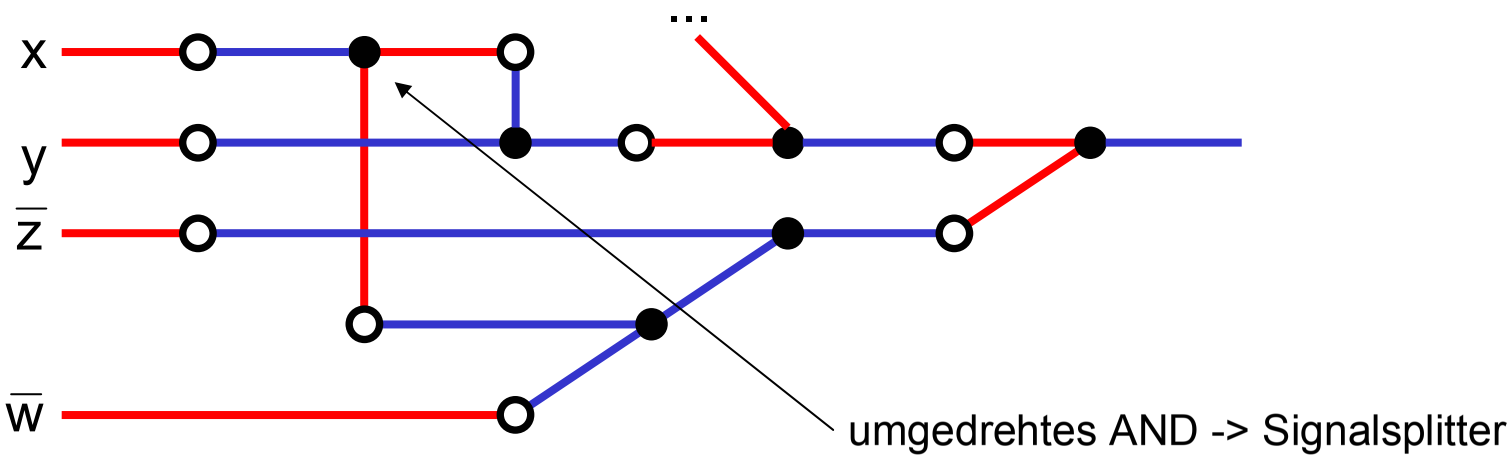
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness (zunächst: PSPACE-hard)

Details:

Kodierung der CNF-Formel
Die CNF Formel wird entsprechend herkömmlicher Logik verdrahtet.
Bsp:



$(x \vee y) \wedge \ldots \wedge (\overline{z} \vee x \vee \overline{w})$

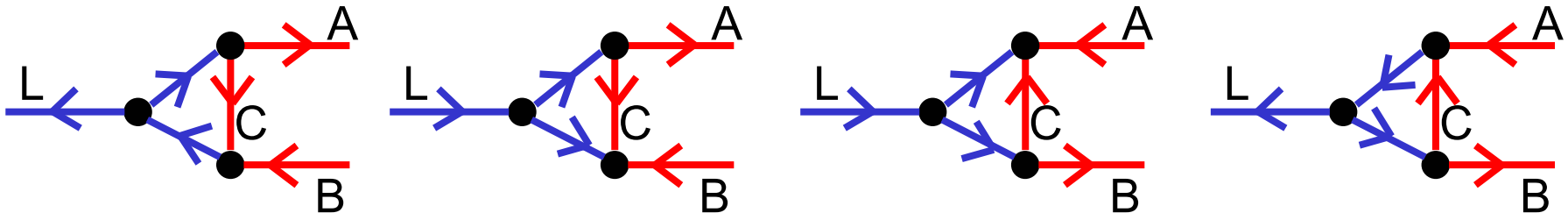
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness

Variablen fixieren („Bits fangen“)
Latches speichern Bits fest:



fixiert, A aktiv | gelöst, A aktiv | gelöst, B aktiv | fixiert, B aktiv

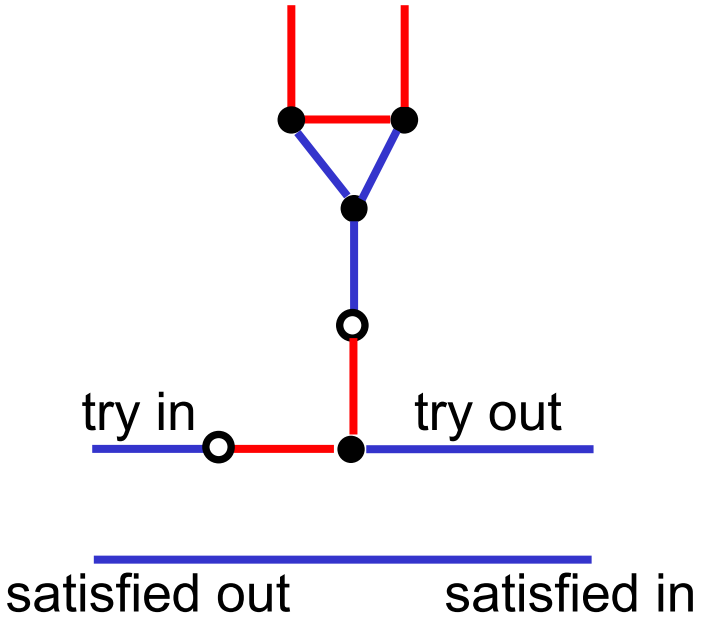
L n.links => eine der beiden anderen OR-Kanten nach links =>
B links (A egal) und **C nach unten**
L n.rechts => die Orientierung der beiden anderen OR-Kanten nach rechts ist möglich => **Umdrehen von C ist möglich**. Wurde C gedreht, kann man das Latch wieder festfrieren und L nach links zeigen lassen.

Nondeterministic Constraint Logic (NCL)

Graph-Formulierung

PSPACE-completeness

Existenzquantor



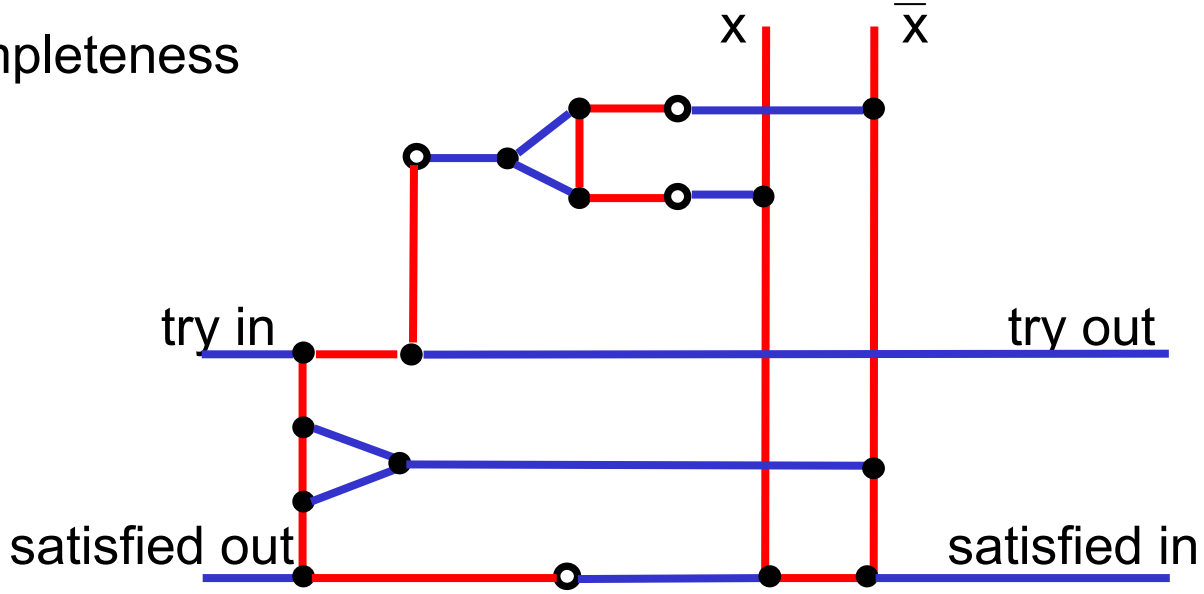
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness

Allquantor



Beh: Ein Allquantor Schaltkreis kann seine satisfied-out Kante nur nach rechts (gem. Bild) richten, wenn zuerst $\bar{x}$ nach oben zeigt und die satisfied-in Kante nach links zeigt, und später x noch oben und satisfied-in nochmal nach links zeigt.

Bew. s. Hearn and Demain: PSPACE-Completeness of Sliding-Block Puzzles and Other Problems through the Nondeterministic Constraint Logic Model of Computation

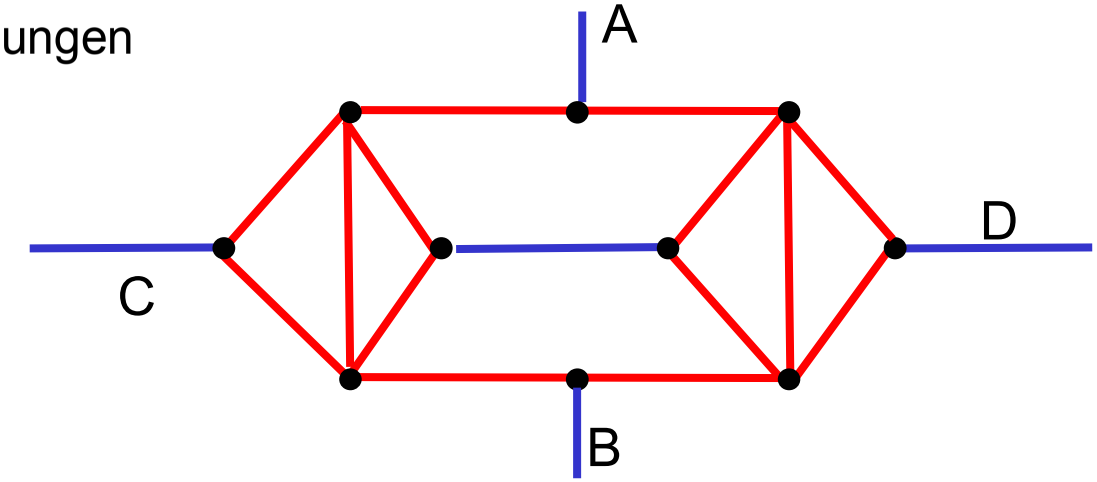
Alles zusammen: NCL ist PSPACE-schwer

Nondeterministic Constraint Logic (NCL)

Graph-Formulierung

PSPACE-completeness

Kreuzungen



Die bisherigen Graphen waren nicht planar. Mit Hilfe von Kreuzungen kann man die Graphen in planare Graphen umwandeln. In der Konstruktion oben kann von A und B nur einer nach aussen zeigen. Ebenso von C und D.

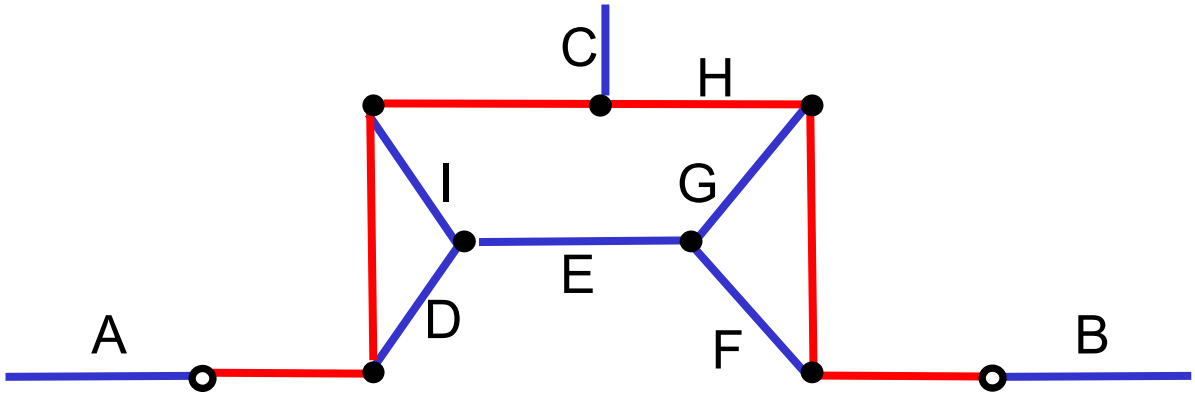
(Begründung s. Hearn and Demain: PSPACE-Completeness of Sliding-Block Puzzles and Other Problems through the Nondeterministic Constraint Logic Model of Computation)

Nondeterministic Constraint Logic (NCL)

Graph-Formulierung

PSPACE-completeness

Protected OR
Ein OR ist protected, wenn wegen globaler Bedingungen es nicht vorkommen kann, dass 2 der 3 Kanten in das OR hineinzeigen. Dann ist es nicht schlimm, wenn das OR für den Fall, dass doch 2 Kanten hineinzeigen, nicht richtig funktioniert und Fehler macht.
Man kann jedoch aus Protected ORs ein „echtes“ OR bauen:



A, B, und C verhalten sich wie ein OR; alle ORs innerhalb der Schaltung sind protected ORs

Nondeterministic Constraint Logic (NCL)

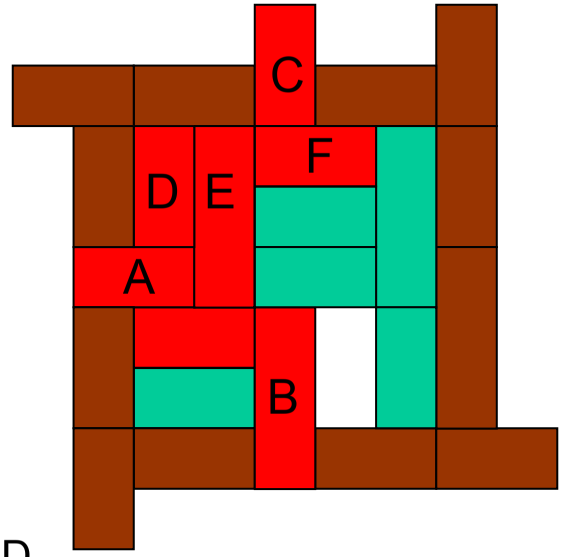
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

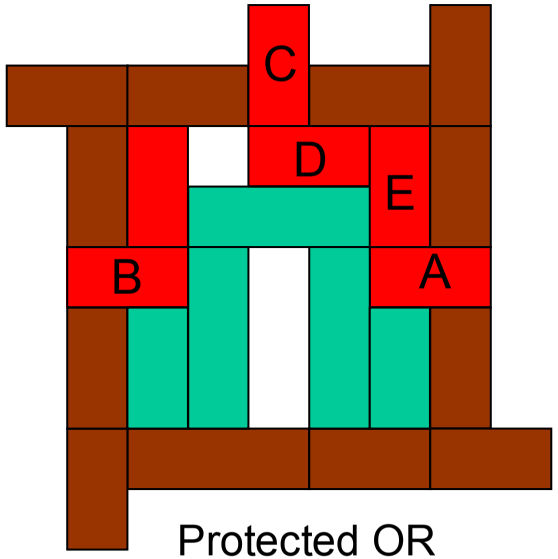
PSPACE-completeness

Rush Hour ist PSPACE schwer; AND und OR

Wie beschränken uns auf cars of 1 x 2 and 1 x 3 blocks.



AND
C darf nur runter, wenn B runter und A nach links geht



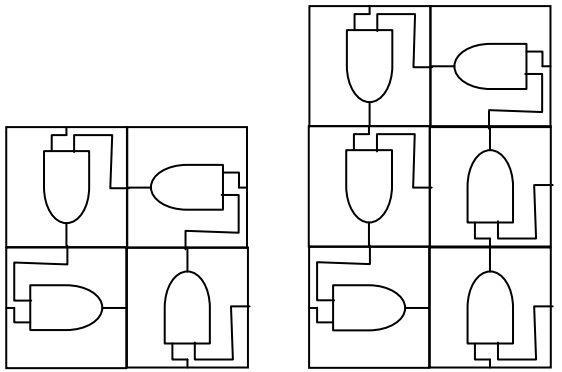
Protected OR
C darf runter, wenn B nach links oder A nach rechts geht

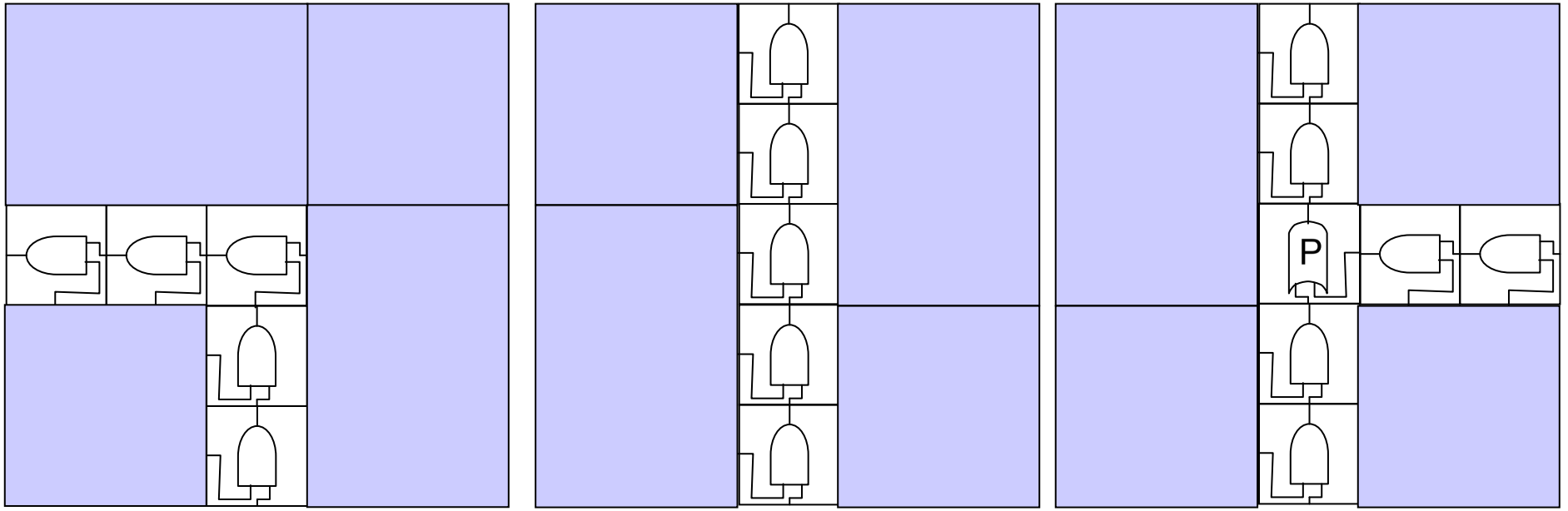
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness: Rush Hour ist
PSPACE schwer; Graphen





Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Satz von Savitch

NPSPACE = DPSPACE

Theorem: Für jede Funktion $s(n) \geq n$ gilt: $NSPACE_1(s(n)) \subseteq DSPACE_3(s^2(n))$
In Worten: Jede Berechnung einer t-Zeit / 1 Band Turingmaschine kann von einer 3-Band DTM in Platz $s^2(n)$ durchgeführt werden.
(Platzbedarf wird betrachtet ohne read-only-Eingabe)

Beweis:

- Betrachte NTM für $L \in NSPACE_1(s(n))$, die mit einer eindeutigen Konfiguration C_{akz} akzeptiert. D.h., es gibt nur einen akzeptierenden Endzustand und am Ende der Berechnung wird das Band gelöscht.
- Betrachte das Prädikat Erreicht-Konf(C,C‘,S,T): Dieses Prädikat ist wahr, wenn die S-Platz-NTM M ausgehend von C die Konfiguration C‘ innerhalb von T Schritten erreicht.
- **Lemma (noch zu zeigen): Erreicht-Konf(C,C‘,S,T) kann von einer 2-Band-DTM mit S·log(T) Platz entschieden werden.**
- Nun ist $T \leq 2^{O(s(n))}$ und damit ist $\log(T) = O(s(n)) \leq c \cdot s(n)$ für eine Konstante $c > 0$.
    - Sei C_{start} die Startkonfiguration
    - Das Prädikat Erreicht-Konf(C_{start},C_{akz},s(n),$2^{O(s(n))}$) entscheidet L.
- Dann kann eine 3-Band-DTM L in Platz $c \cdot s(n) \cdot s(n) = c \cdot s^2(n) = O(s^2(n))$ die Sprache L entscheiden.

Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Satz von Savitch

NPSPACE = DPSPACE

Lemma: Das Prädikat **Erreicht-Konf(C,C‘,S,T) kann von einer 2-Band- DTM mit S·log(T) Platz entschieden werden.**

Beweis: Betrachte folgende DTM M‘ auf Eingabe (C,C‘,S,T)

- falls T = 0 dann
  - akzeptiere falls C=C‘ und akzeptiere nicht falls C ungleich C‘.
- falls T = 1 dann
  - akzeptiere fals C‘ eine erlaubte Nachfolgekonfiguration von C ist, oder falls C=C‘. Sonst akzeptiere nicht.
- falls T > 1 dann
  - für alle Konfigurationen Z der Länge S
    - Berechne rekursiv r_1 = Erreicht-Konf(C,Z,S, $\lfloor T/2 \rfloor$)
    - Berechne rekursiv r_2 = Erreicht-Konf(Z,C‘,S, $\lceil T/2 \rceil$)
    - falls r_1 und r_2 gilt, halte und akzeptiere
  - akzeptiere nicht.

Platz: s(n)+1 in jeder Rekrsionstiefe bei Anzahl der Rekursionstiefen log(T).