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?

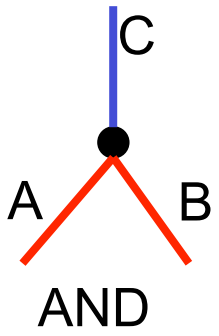
Nondeterministic Constraint Logic (NCL)

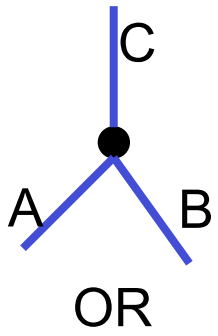
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

AND/OR Constraint Graphen (spezielle NCL-Graphen)

- Knoten mit minimum in-flow constraint 2 und Kanten-Eingangsgewichten 1, 1 und 2 verhalten sich in folgendem Sinne wie ein AND:
  Die Kante mit Gewicht 2 (blau) kann nur nach außen zeigen, wenn beide Kanten mit Gewicht 1 (rot) nach innen zeigen.
- Knoten mit minimum in-flow constraint 2 und Kanten-Eingangsgewichten 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.

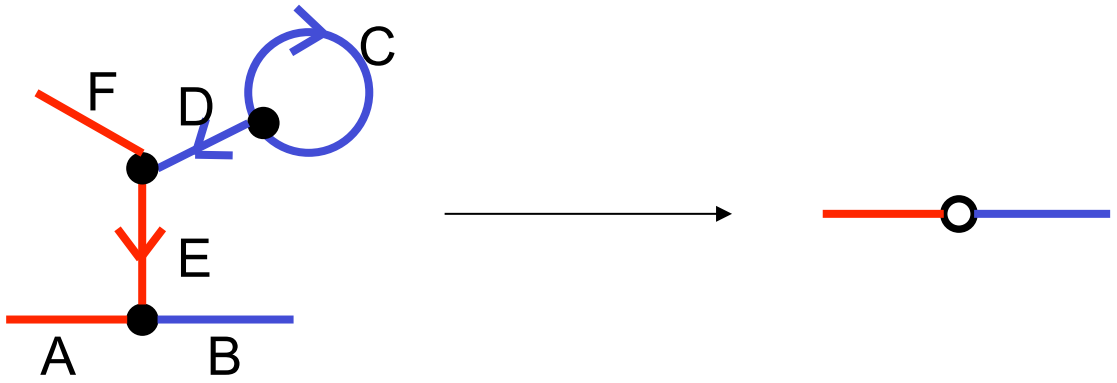
Nondeterministic Constraint Logic (NCL)

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.

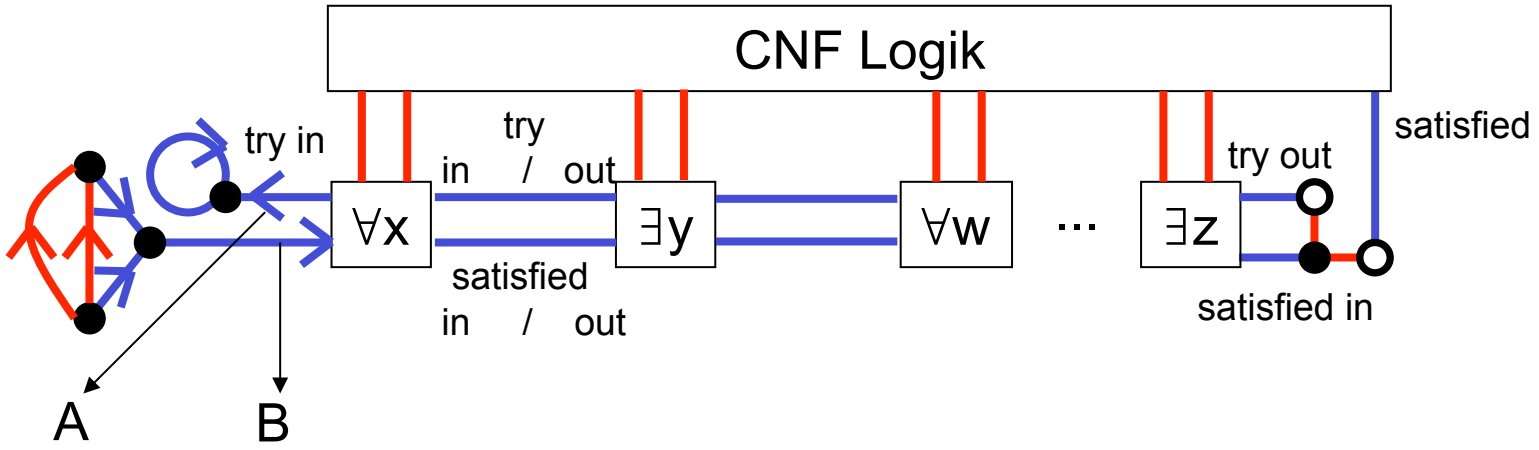
Nondeterministic Constraint Logic (NCL)

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)

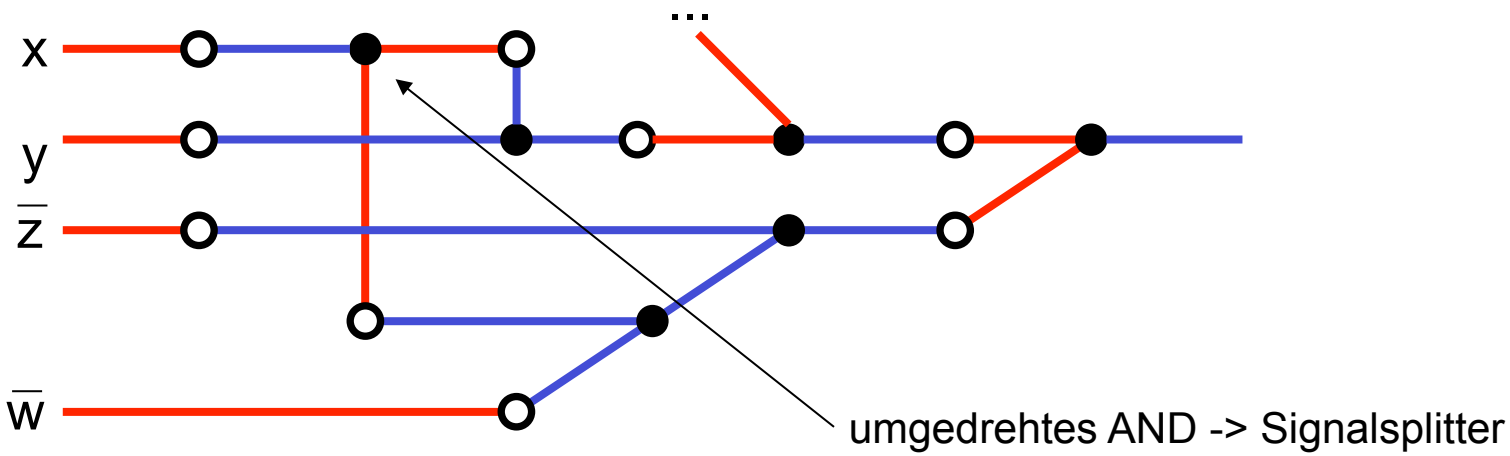
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness (zunächst: PSPACE-hard)

Details:

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



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

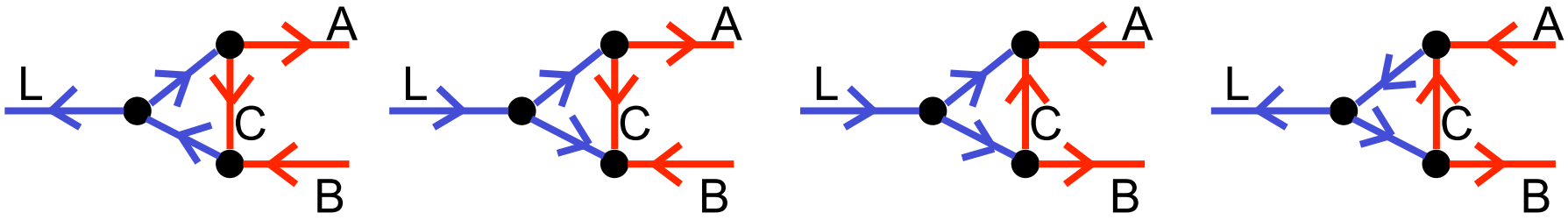
Nondeterministic Constraint Logic (NCL)

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.

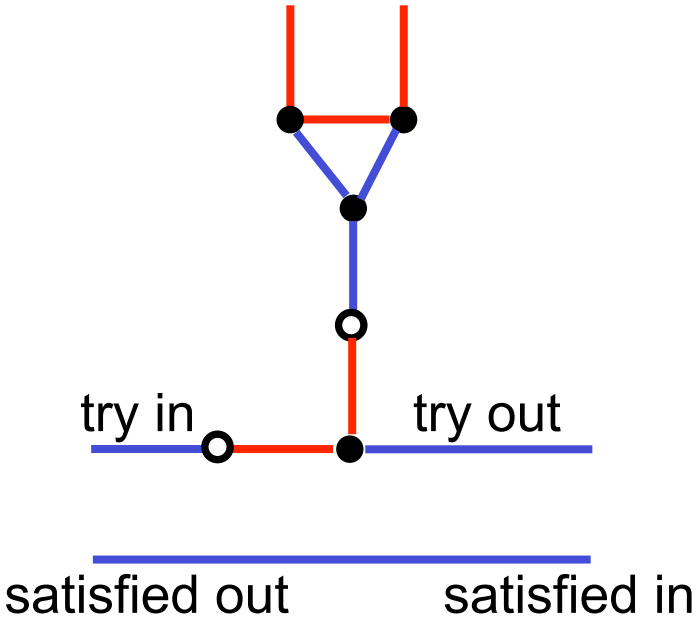
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

PSPACE-completeness

Existenzquantor



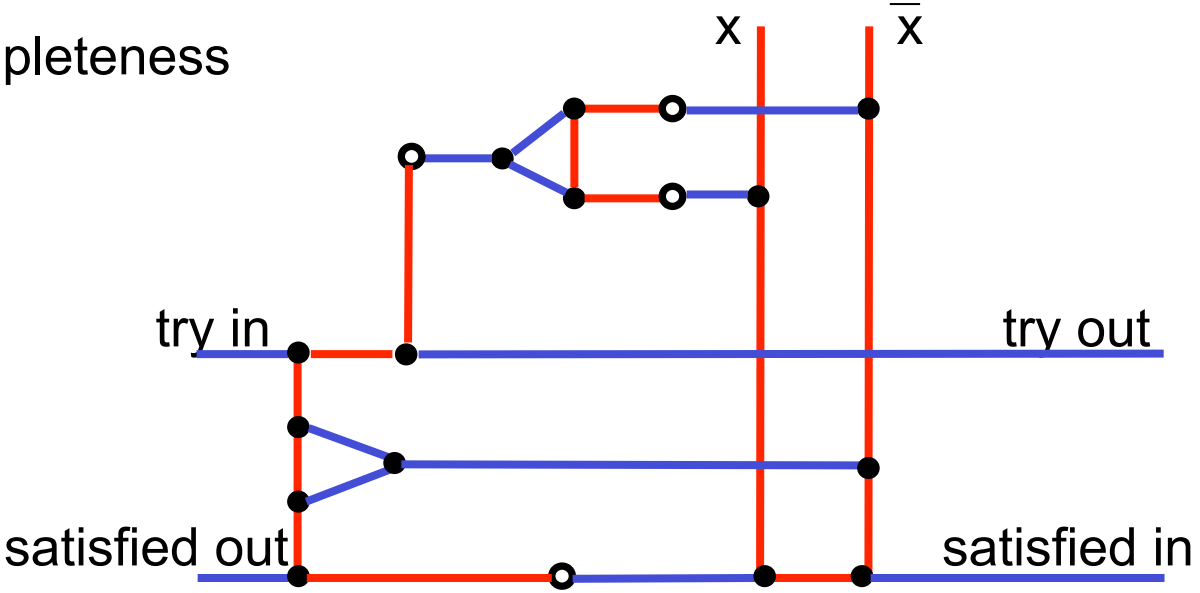
Nondeterministic Constraint Logic (NCL)

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

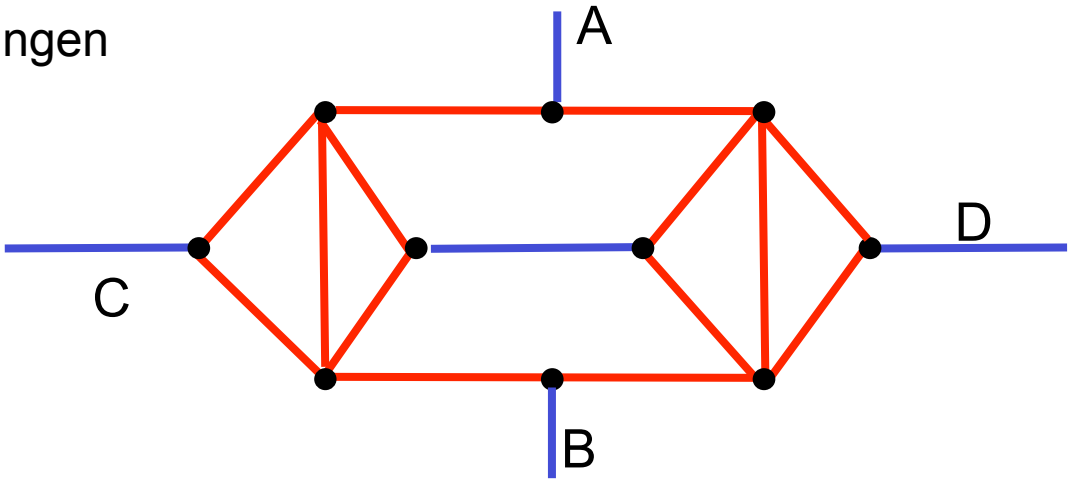
Alles zusammen: NCL ist PSPACE-schwer

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)

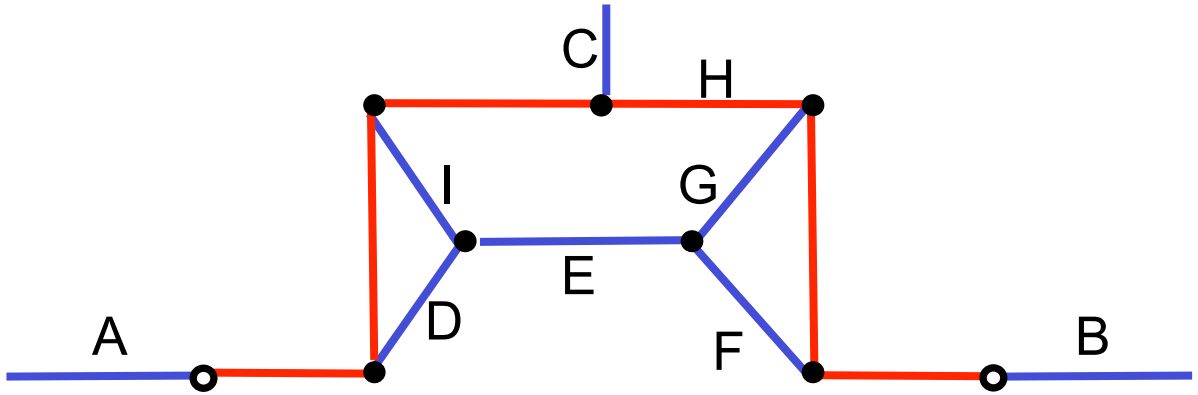
Nondeterministic Constraint Logic (NCL)

TECHNISCHE UNIVERSITÄT DARMSTADT

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)

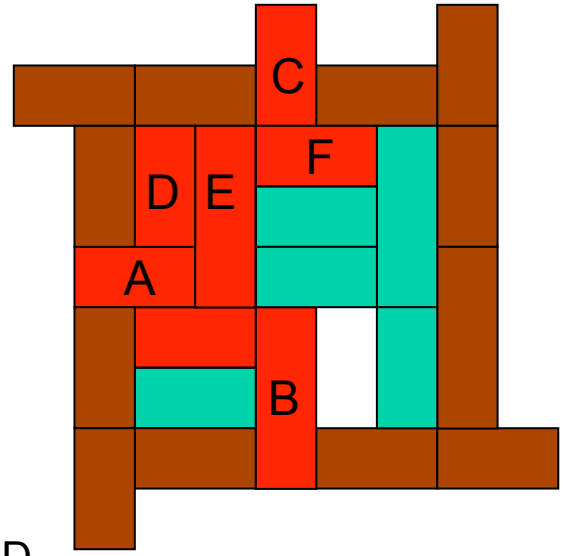
TECHNISCHE UNIVERSITÄT DARMSTADT

Graph-Formulierung

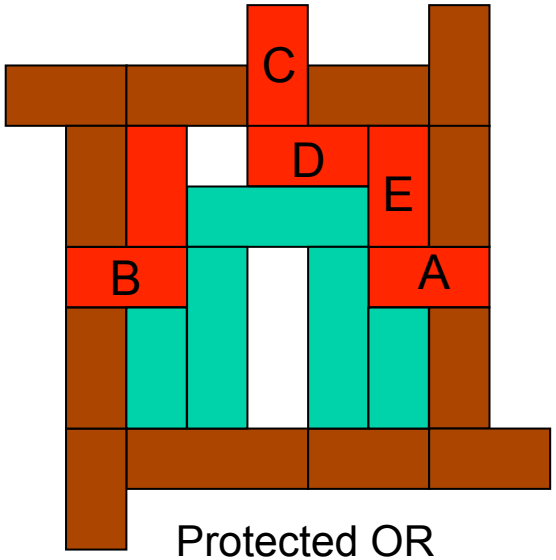
PSPACE-completeness

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



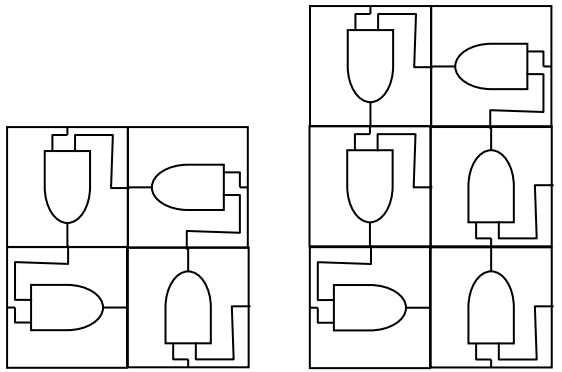
Protected OR
C darf runter, wenn B nach links oder A nach rechts geht

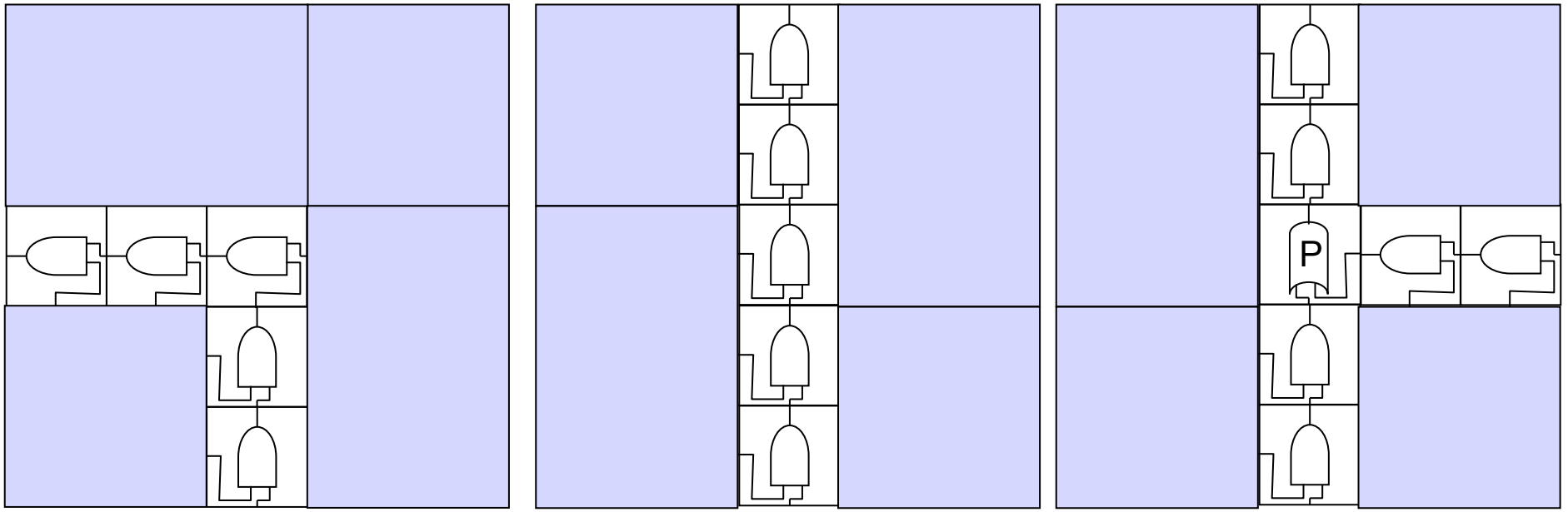
Nondeterministic Constraint Logic (NCL)

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: $NPSPACE_1(s(n)) \subseteq DPSPACE_3(s^2(n))$

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).