Rush Hour ist PSPACE-vollständig

Florian Thiemer

Freie Universität Berlin Fachbereich Informatik

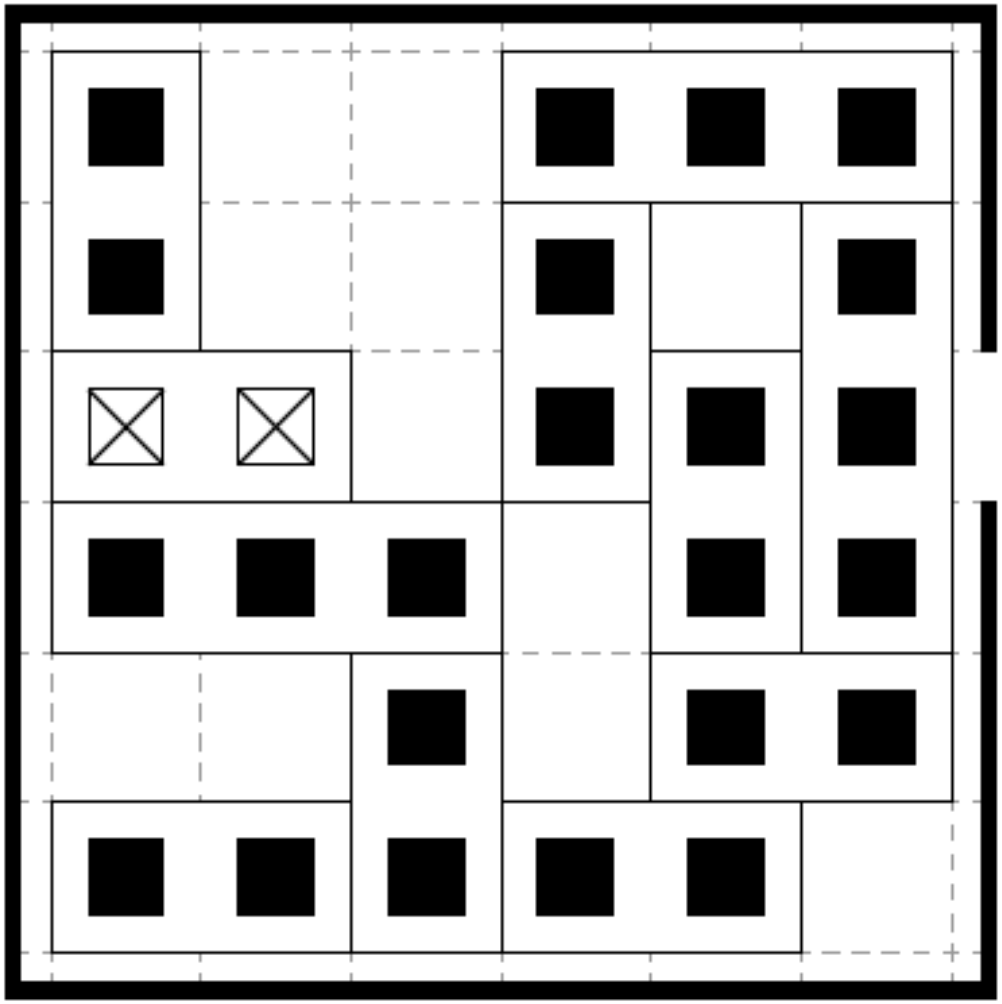
June 21, 2007

Inhalt

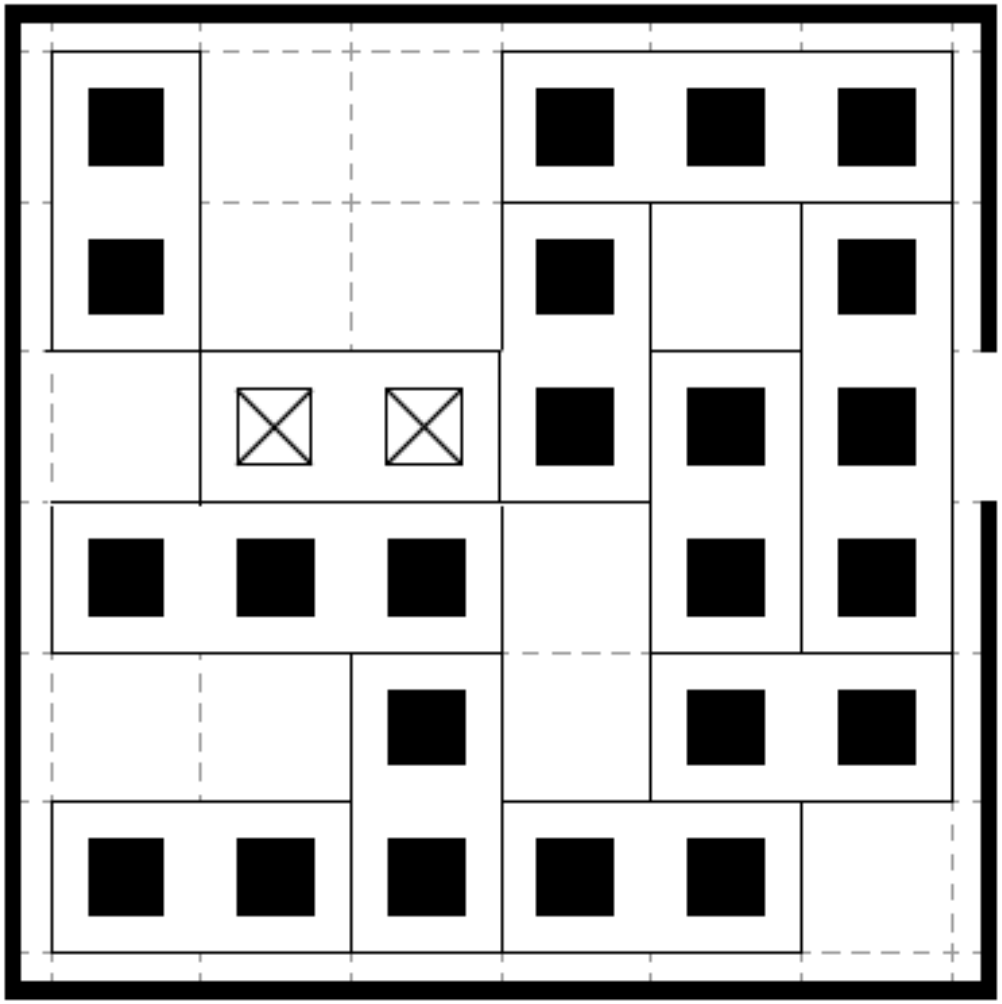
Rush Hour

- Wird auf einem quadratischem Spielfeld der Größe 6×6 gespielt
- Ziel des Spiels ist es die Autos so zu bewegen, dass ein Zielauto sich durch den Ausgang bewegen kann
- Ein Auto kann sich nur horizontal oder vertikal bewegen
- Bewegungsrichtung hängt von der Orientierung des Autos ab
- Die Orientierung des Autos kann sich nicht verändern
- Autos haben die Größe 2 oder 3

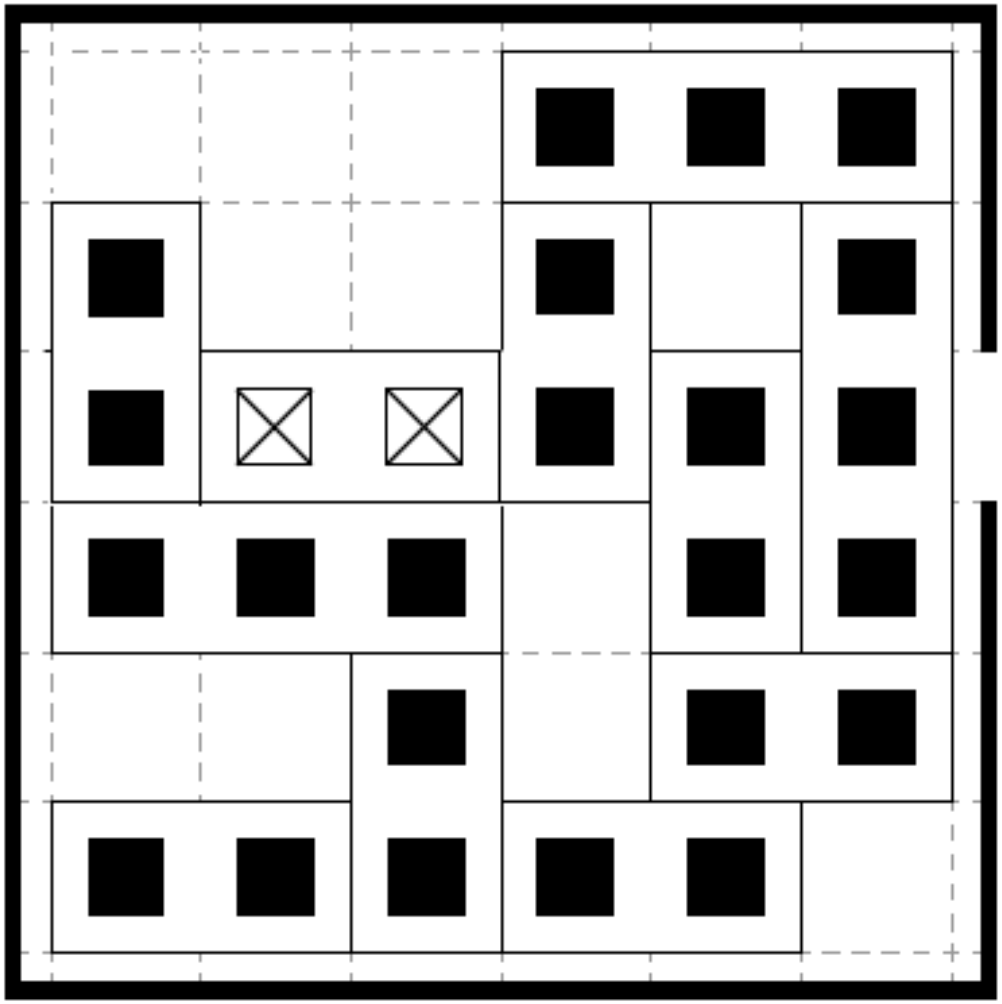
Beispiel



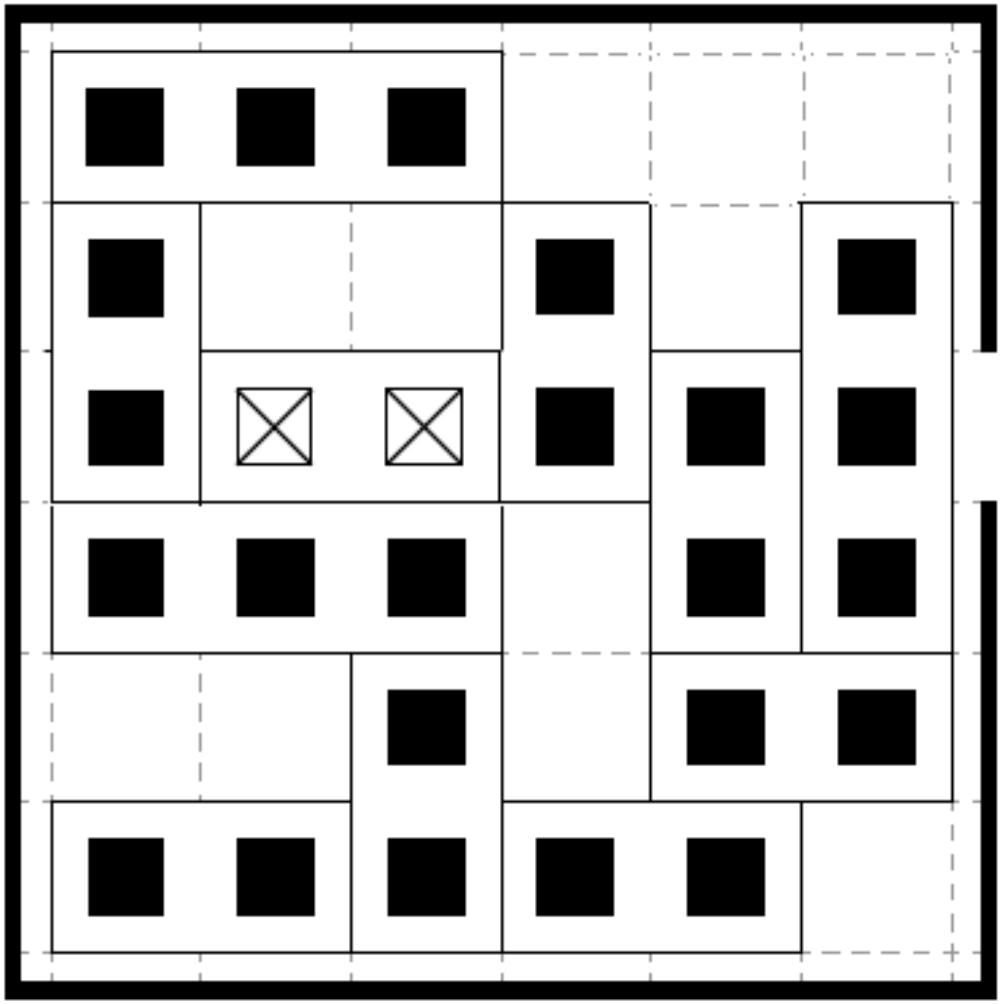
Beispiel



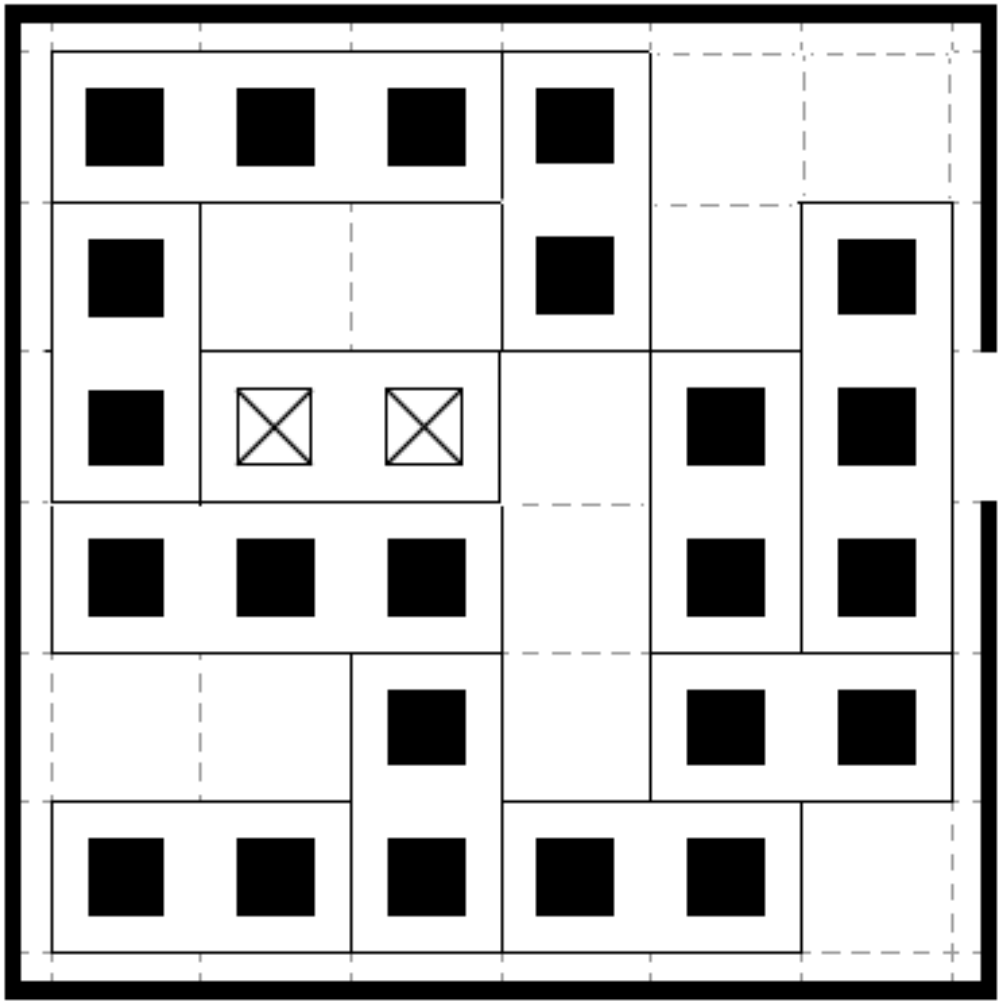
Beispiel



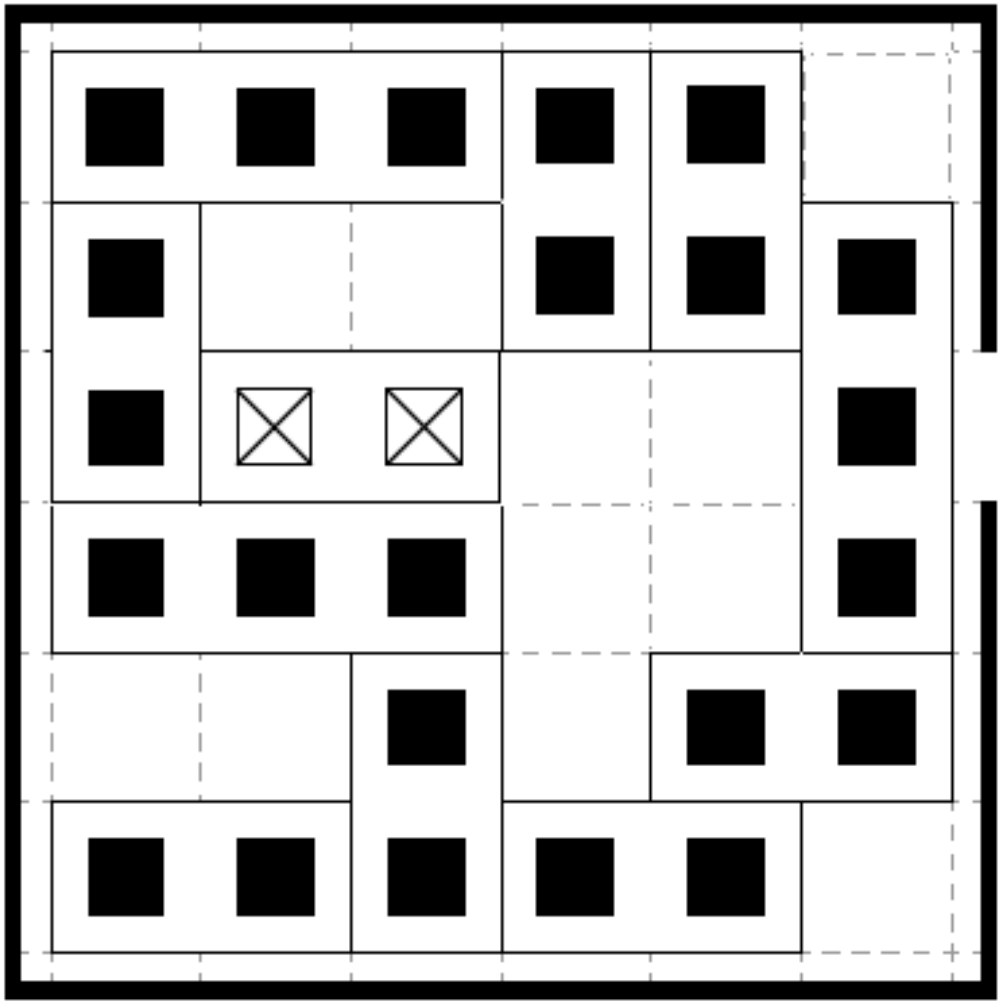
Beispiel



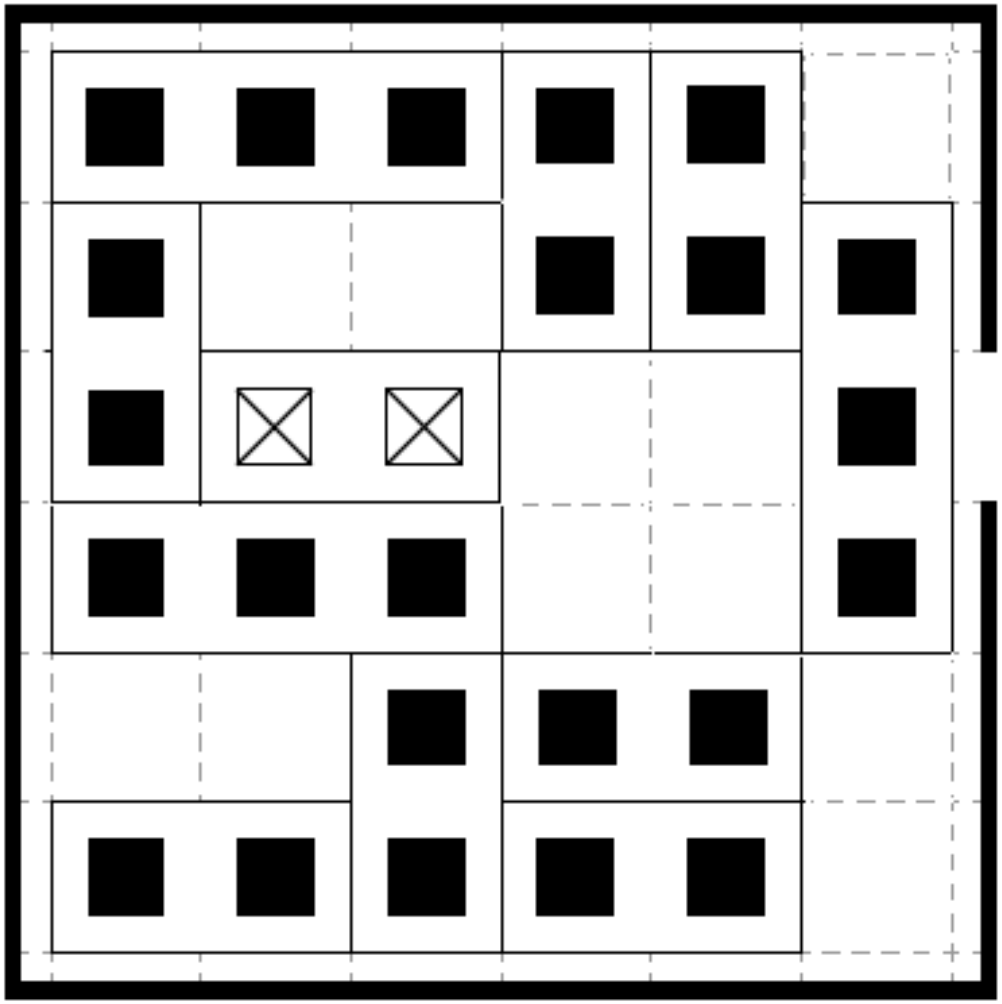
Beispiel



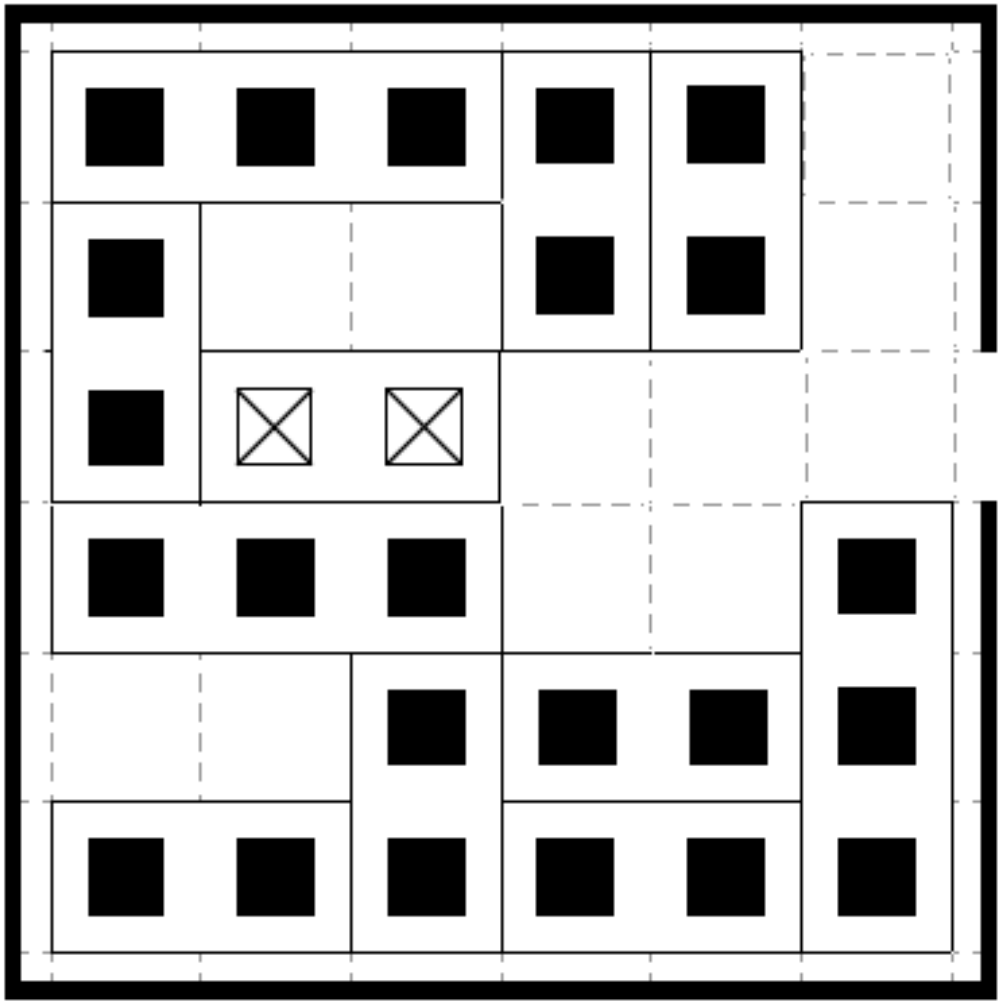
Beispiel



Beispiel



Beispiel



GRH

GRH ist eine Verallgemeinerung von Rush Hour mit den folgenden 2 Modifikationen:

- Das Spielfeld darf eine beliebige Höhe und Breite haben.
- Der Ausgang kann sich an einer beliebigen Position des Spielfeldrandes befinden

Logik mit Rush Hour

- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

Logik mit Rush Hour

- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

- Triggerlinien müssen sich überschneiden können

Logik mit Rush Hour

- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

- Triggerlinien müssen sich überschneiden können
- Triggerlinien müssen sich aufsplitten

Logik mit Rush Hour

- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

- Triggerlinien müssen sich überschneiden können
- Triggerlinien müssen sich aufsplitten
- konjunktive Zusammenführung von Triggerlinien

Logik mit Rush Hour

- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

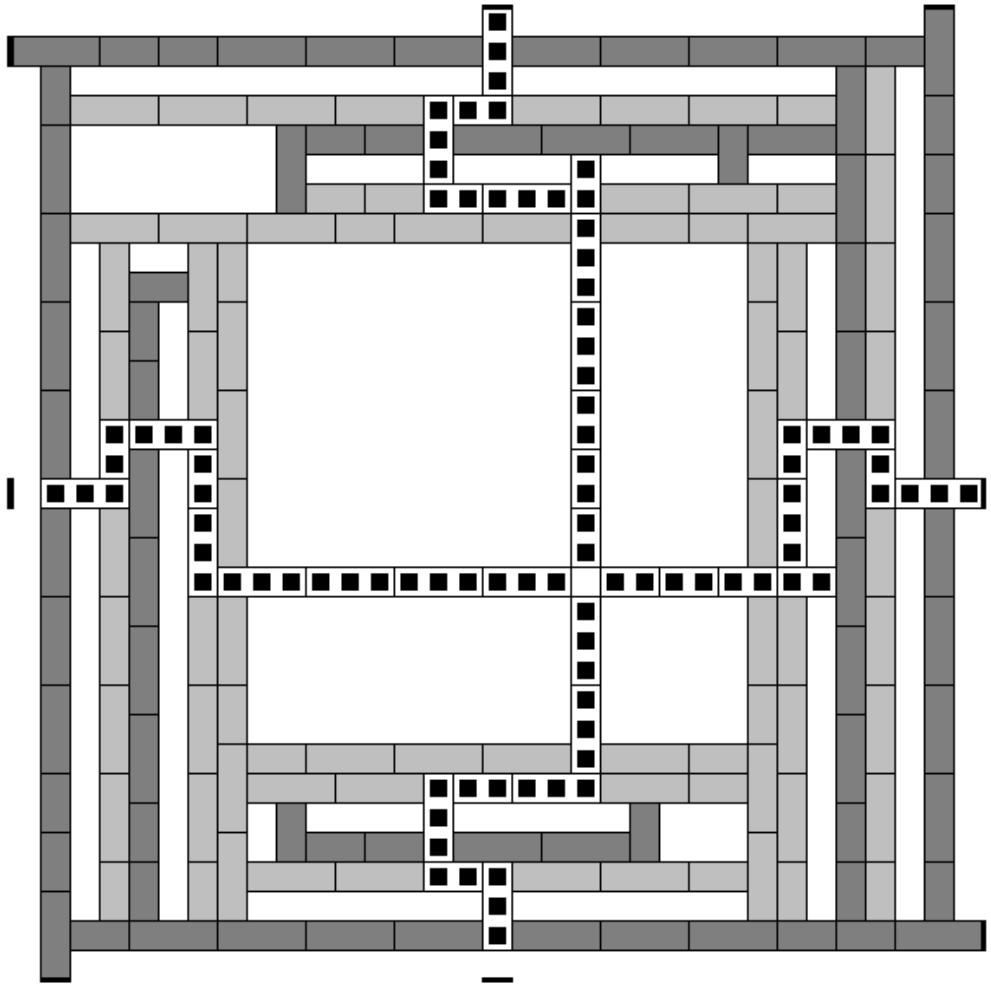
- Triggerlinien müssen sich überschneiden können
- Triggerlinien müssen sich aufsplitten
- konjunktive Zusammenführung von Triggerlinien
- disjunktive Zusammenführung von Triggerlinien

Logik mit Rush Hour

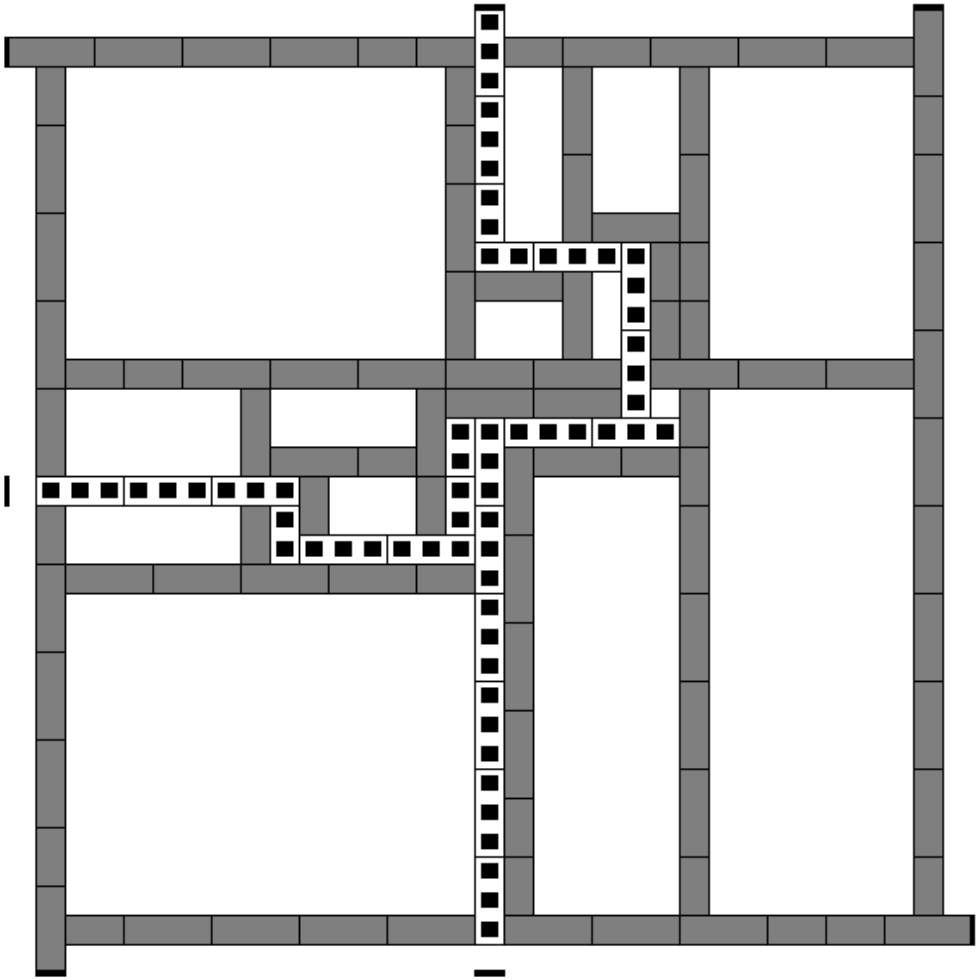
- Gibt es eine Kombination von Input-Autos, so dass ein Output-Auto sich auch bewegen kann ?
- Konstruktion von speziellen Baublöcken die es Triggerautos ermöglicht sich innerhalb des Blocks zu bewegen

- Triggerlinien müssen sich überschneiden können
- Triggerlinien müssen sich aufsplitten
- konjunktive Zusammenführung von Triggerlinien
- disjunktive Zusammenführung von Triggerlinien
- ??? Negation ???

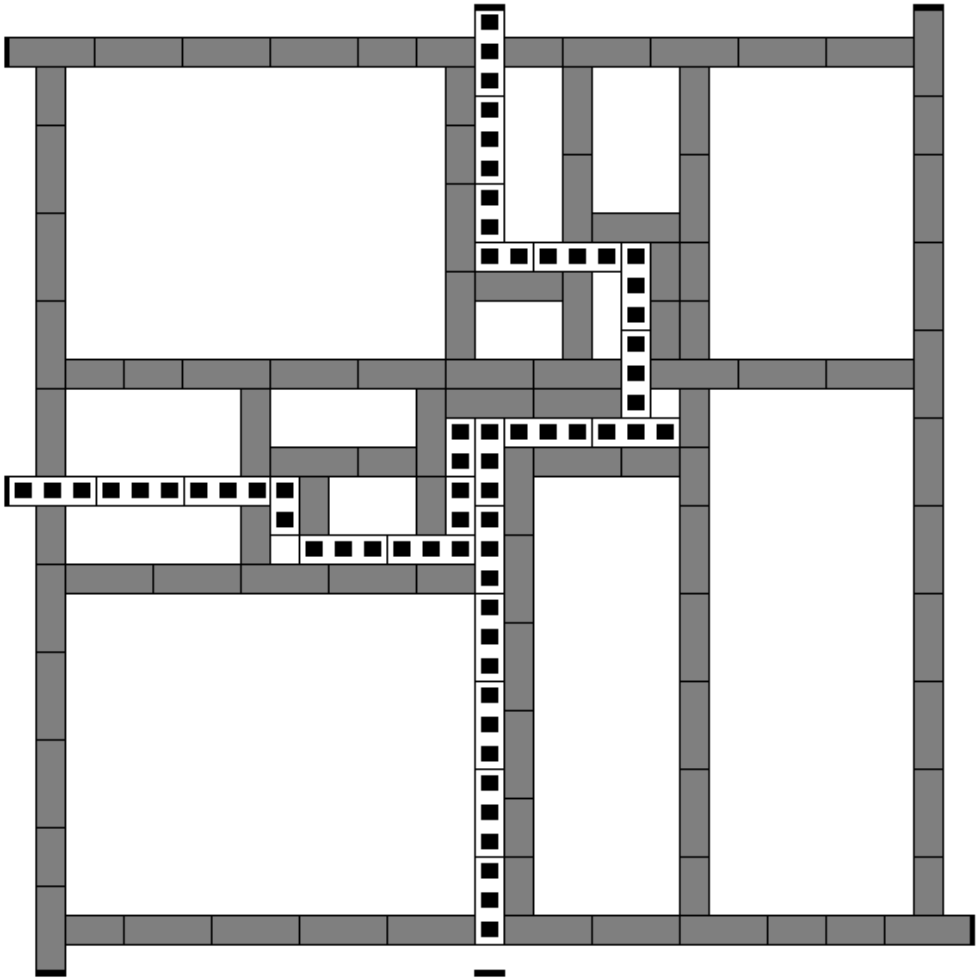
Überschneidung von Triggerlinien



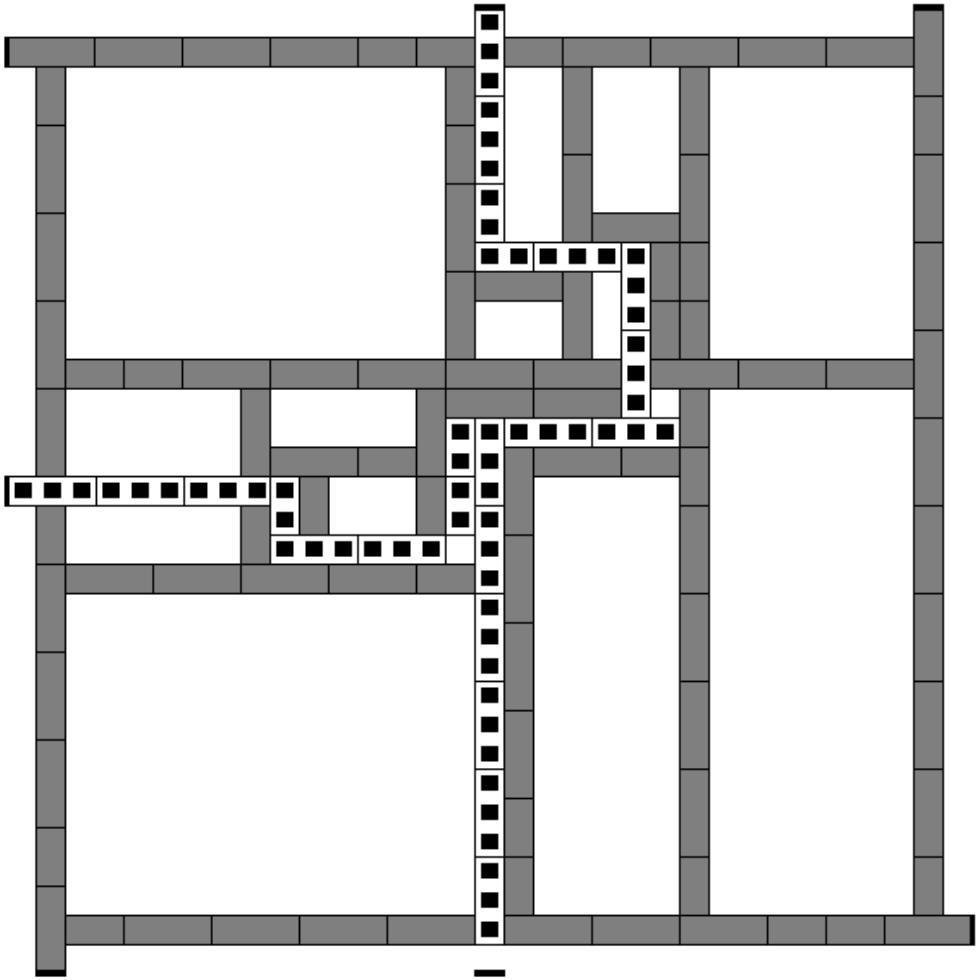
Konjunktiver Baublock



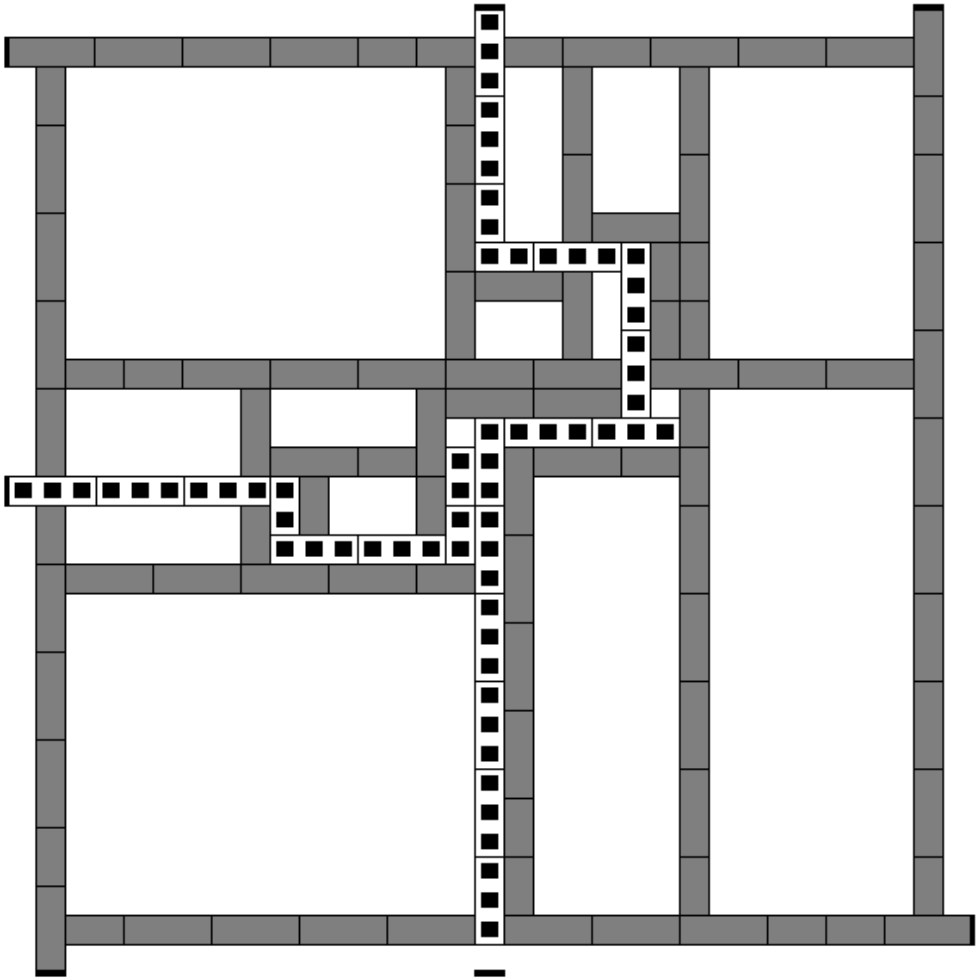
Konjunktiver Baublock



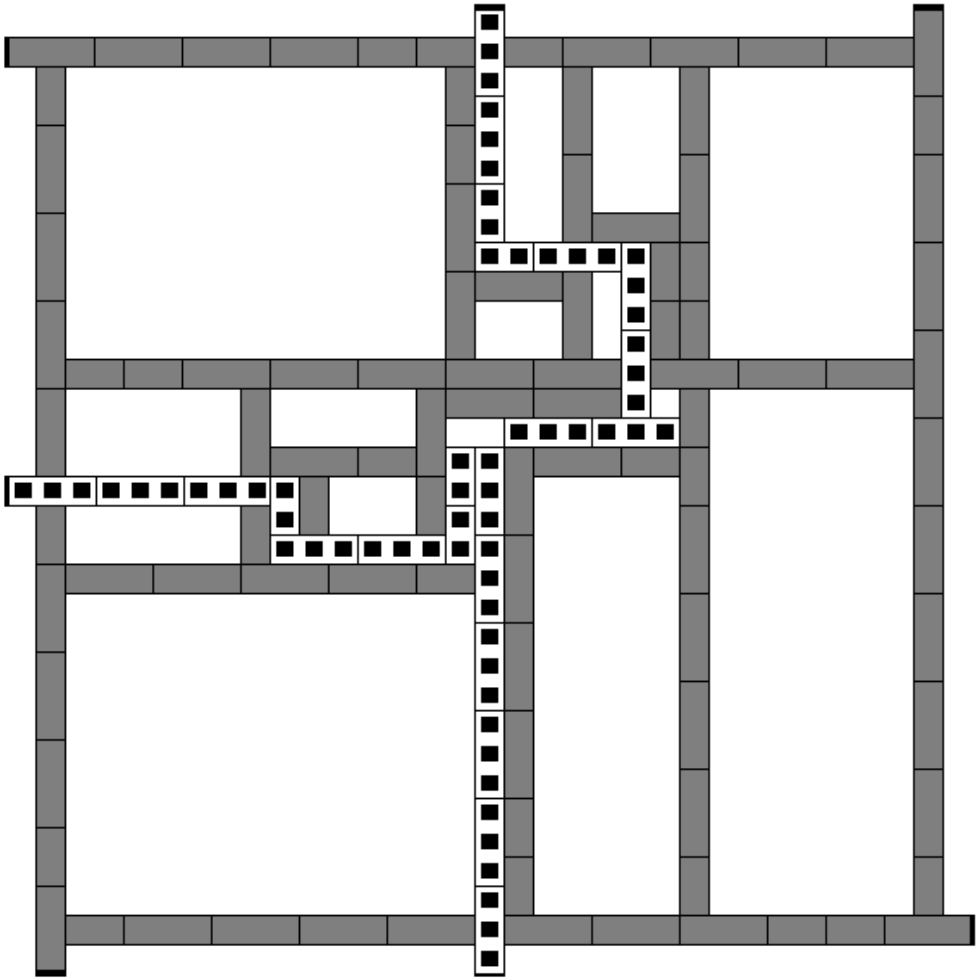
Konjunktiver Baublock



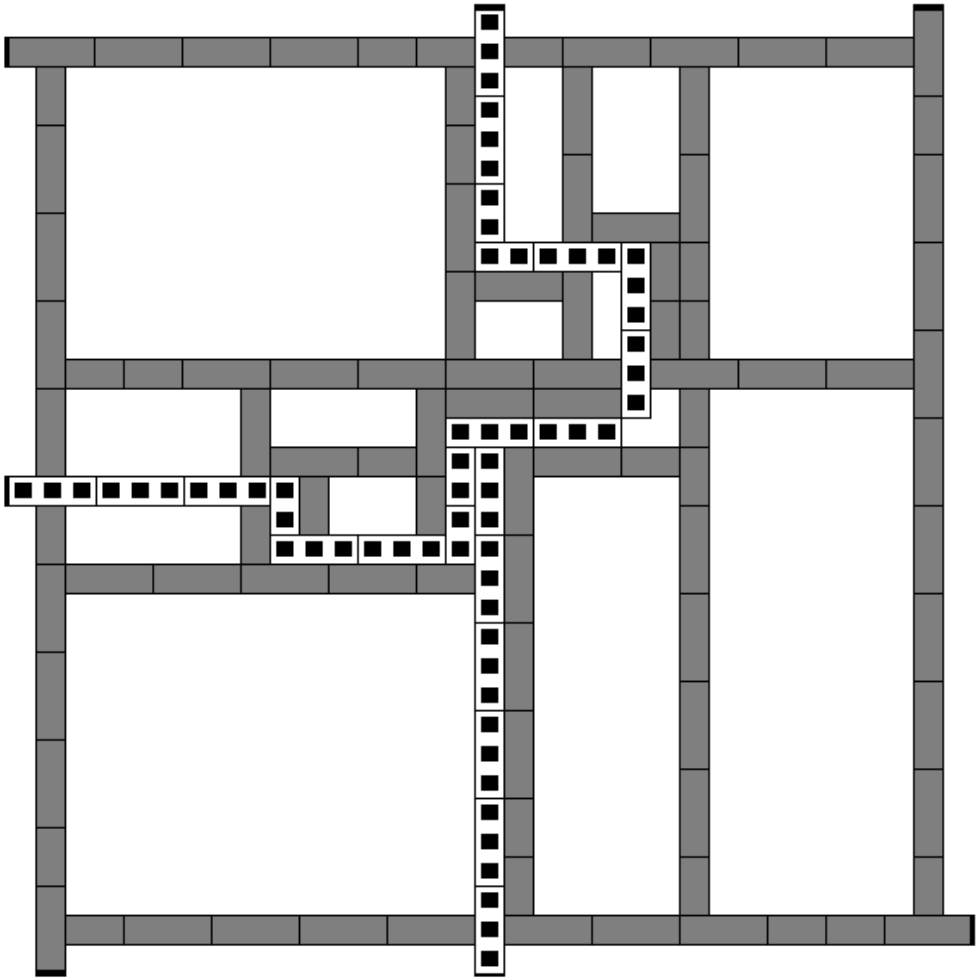
Konjunktiver Baublock



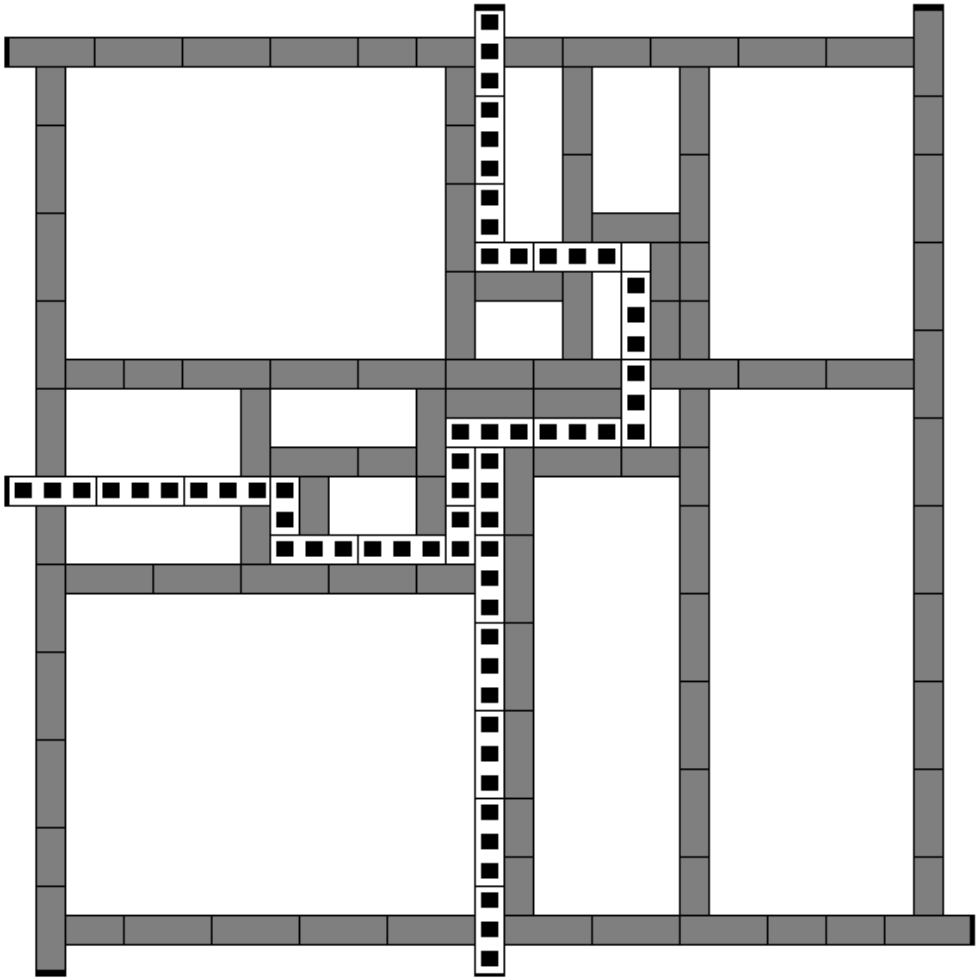
Konjunktiver Baublock



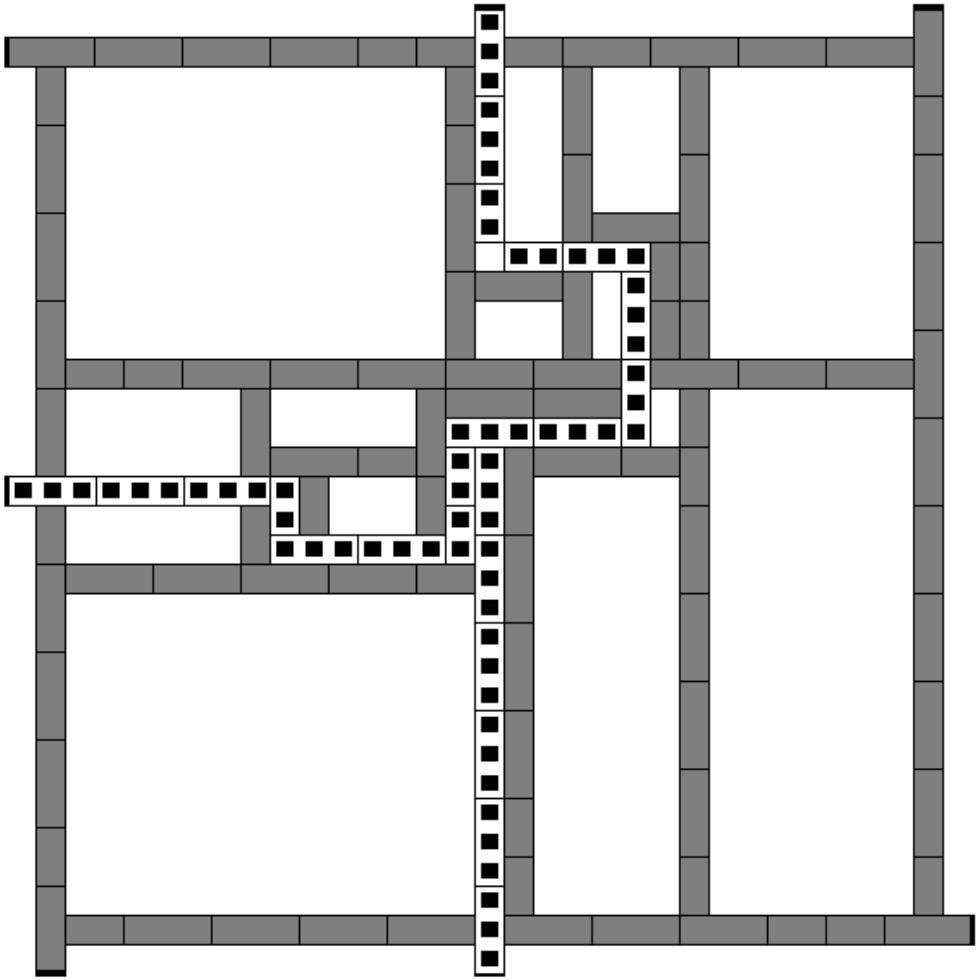
Konjunktiver Baublock



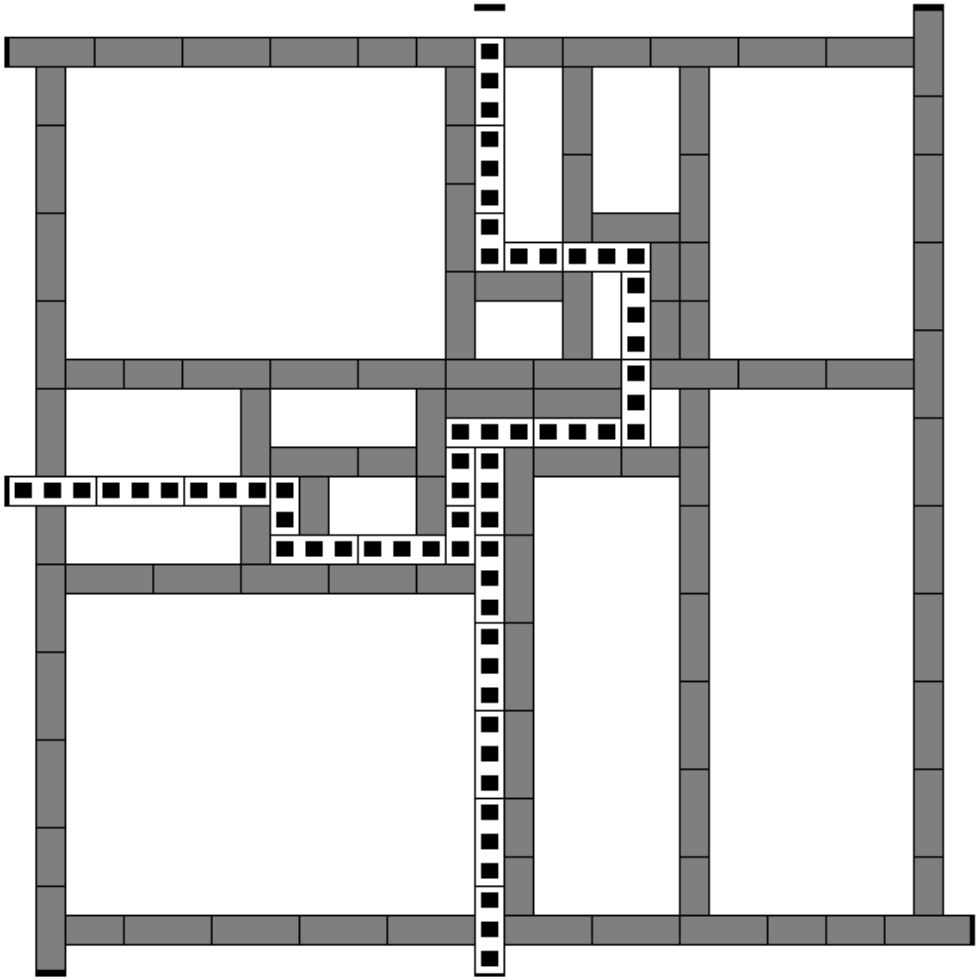
Konjunktiver Baublock



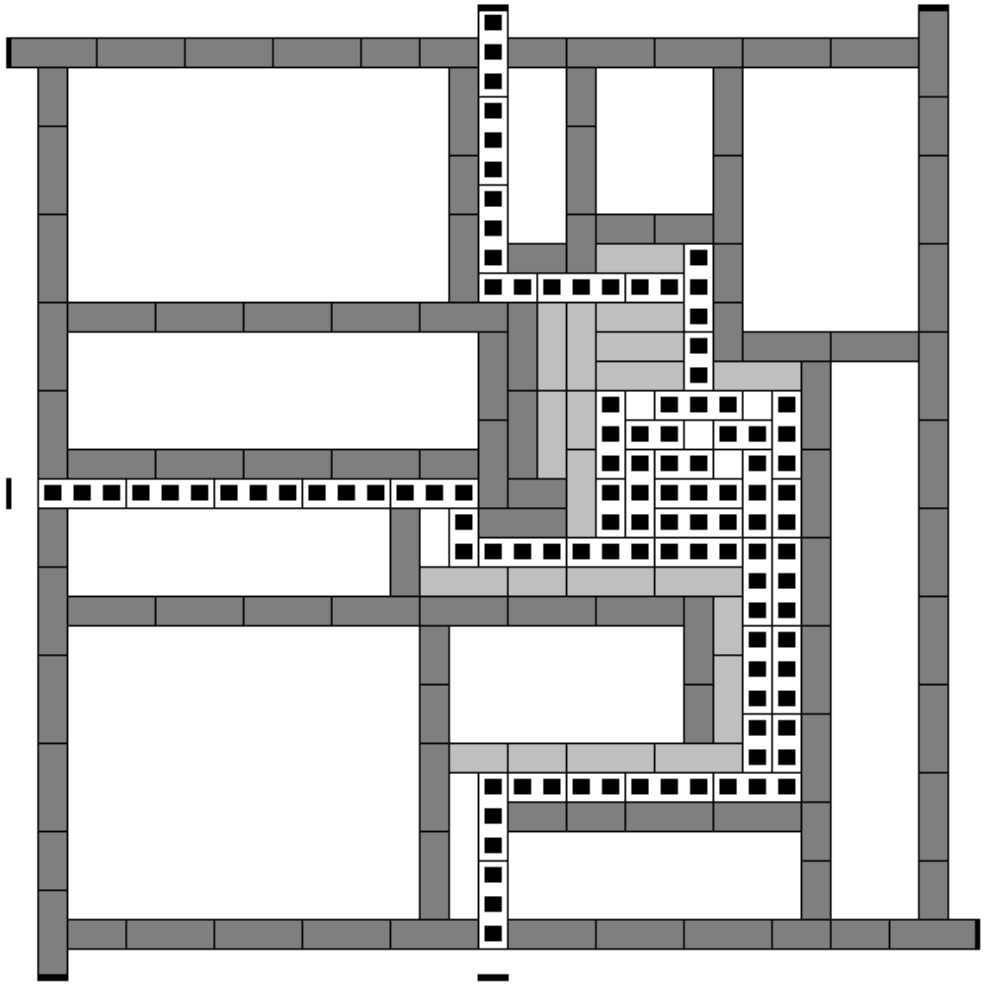
Konjunktiver Baublock



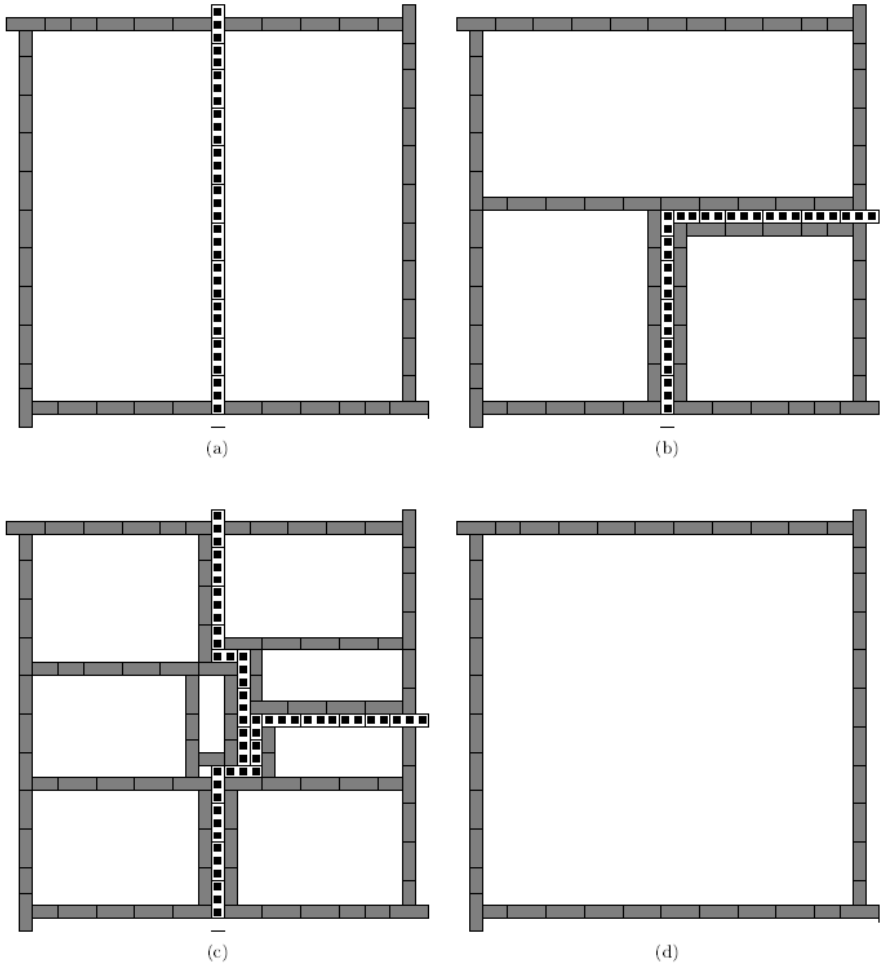
Konjunktiver Baublock



Disjunktiver Baublock



Weitere Baublöcke



Grundbausteine sind unzureichend

- Die Grundbausteine reichen nicht aus um Boolesche Logik darzustellen
- Angenommen die Zustände *offen* und *geschlossen* wären äquivalent mit *true* und *false* aus der Booleschen Logik
- *true* und *false* sollen also Ereignisse im Schaltkreis auslösen können
- Da *false* gleichbedeutend mit *geschlossen* ist, so würden sich die Autos auch weiterhin blockieren
- Damit wäre es unmöglich einen Invertierer zu bauen

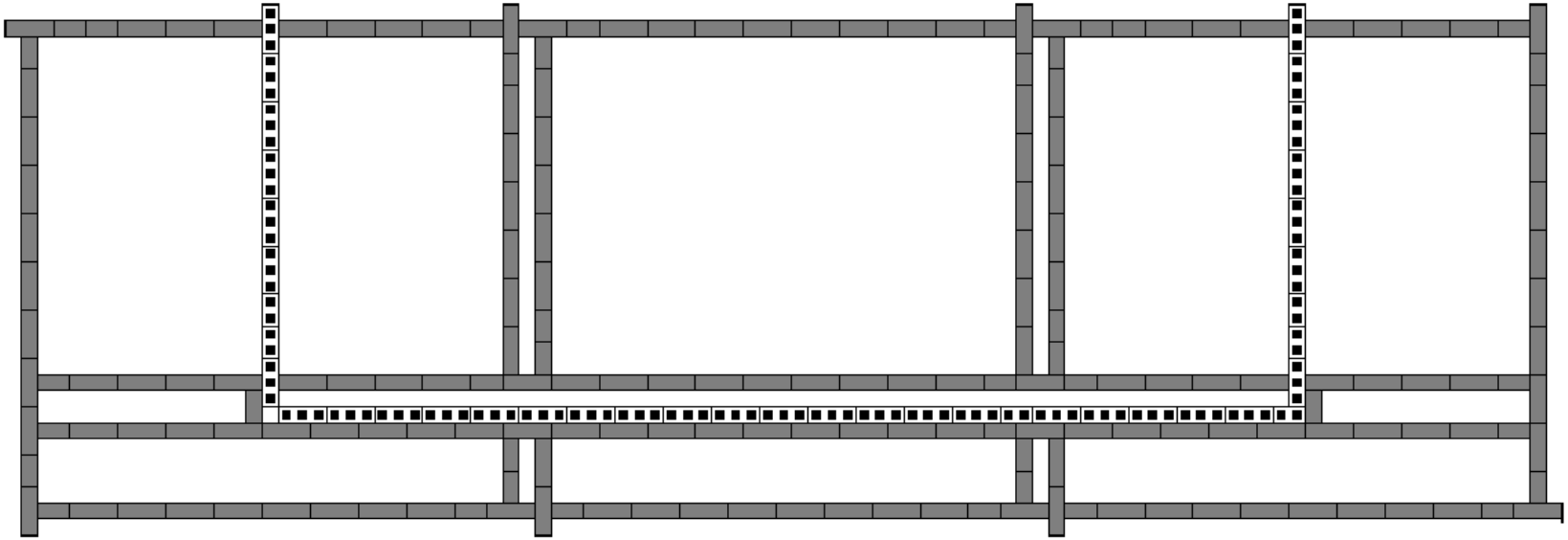
Darstellung von booleschen Werten

- Man stellt boolesche Werte durch 2 Triggerlinien dar
- Eine repräsentiert *true*, die andere *false*
- Beide Triggerlinien können Ereignisse in anderen Blöcken auslösen

Darstellung von booleschen Werten

- Man stellt boolesche Werte durch 2 Triggerlinien dar
- Eine repräsentiert *true*, die andere *false*
- Beide Triggerlinien können Ereignisse in anderen Blöcken auslösen
- Problem: Es gibt unsinnige Werte

Ein Switch



Boolesche Logik mit GRH

Wir definieren Boolesche Werte wie folgt:

$$\begin{aligned} 1 &= (A_T^+, A_F^-) \\ 0 &= (A_T^-, A_F^+) \\ - &= (A_T^-, A_F^-) \\ + &= (A_T^+, A_F^+) \end{aligned}$$

1 repräsentiert *true*, 0 repräsentiert *false*. - und + sind die beiden unsinnigen Werte. + Kann aufgrund der Konstruktion des Switches nie auftreten.

Boolesche Logik mit GRH

AND, OR und NOT mit GRH:

$$\begin{aligned} A \wedge B &= (A_T \mathbin{\overline{\wedge}} B_T, A_F \mathbin{\underline{\vee}} B_F) \\ A \vee B &= (A_T \mathbin{\underline{\vee}} B_T, A_F \mathbin{\overline{\wedge}} B_F) \\ \neg A &= (A_F, A_T) \end{aligned}$$

Damit:

$$\begin{aligned} - \wedge 0 &= 0 \\ - \wedge 1 &= - \\ - \vee 0 &= - \\ - \vee 1 &= 1 \end{aligned}$$

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch
- Alle Switches werden mit – Initialisiert

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch
- Alle Switches werden mit – Initialisiert
- Der Boolesche Ausdruck wird durch die GRH Schaltkreise mit NOT, AND und OR dargestellt

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch
- Alle Switches werden mit – Initialisiert
- Der Boolesche Ausdruck wird durch die GRH Schaltkreise mit NOT, AND und OR dargestellt
- Die NOT, AND und OR Schaltkreise werden nun durch die Grundbaublöcke verbunden

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch
- Alle Switches werden mit – Initialisiert
- Der Boolesche Ausdruck wird durch die GRH Schaltkreise mit NOT, AND und OR dargestellt
- Die NOT, AND und OR Schaltkreise werden nun durch die Grundbaublöcke verbunden
- Alle Ausgänge der verwendeten Blöcke werden mit – initialisiert

GRH ist NP-schwer

- Hierzu reduzieren wir SAT auf GRH
- Für jede Variable in einem Booleschen Ausdruck benutzen wir einen Switch
- Alle Switches werden mit – Initialisiert
- Der Boolesche Ausdruck wird durch die GRH Schaltkreise mit NOT, AND und OR dargestellt
- Die NOT, AND und OR Schaltkreise werden nun durch die Grundbaublöcke verbunden
- Alle Ausgänge der verwendeten Blöcke werden mit – initialisiert
- Umwandlung vom Booleschen Ausdruck in einen GRH-Schaltkreis also in polynomiell viel Platz und Zeit

GRH ist NP-schwer

- Das Zielauto wird mit dem Output des Schaltkreises blockiert

GRH ist NP-schwer

- Das Zielauto wird mit dem Output des Schaltkreises blockiert
- Der Output des Schaltkreises kann nur öffnen, genau dann wenn es eine Belegung der Booleschen Variablen gibt, die den Ausdruck erfüllen.

GRH ist NP-schwer

- Das Zielauto wird mit dem Output des Schaltkreises blockiert
- Der Output des Schaltkreises kann nur öffnen, genau dann wenn es eine Belegung der Booleschen Variablen gibt, die den Ausdruck erfüllen.
- GRH zu lösen, ermöglicht es uns also auch SAT zu lösen

GRH ist NP-schwer

- Das Zielauto wird mit dem Output des Schaltkreises blockiert
- Der Output des Schaltkreises kann nur öffnen, genau dann wenn es eine Belegung der Booleschen Variablen gibt, die den Ausdruck erfüllen.
- GRH zu lösen, ermöglicht es uns also auch SAT zu lösen
- GRH ist somit NP-schwer

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert
- Es gibt $2n$ mögliche Nachfolgekonfigurationen

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert
- Es gibt $2n$ mögliche Nachfolgekonfigurationen
- Wähle nichtdeterministisch eine der Möglichkeiten aus und aktualisiere Turingband

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert
- Es gibt $2n$ mögliche Nachfolgekonfigurationen
- Wähle nichtdeterministisch eine der Möglichkeiten aus und aktualisiere Turingband
- Solange wiederholen bis wir in Schleife kommen oder Zielauto herausfahren kann

GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert
- Es gibt $2n$ mögliche Nachfolgekonfigurationen
- Wähle nichtdeterministisch eine der Möglichkeiten aus und aktualisiere Turingband
- Solange wiederholen bis wir in Schleife kommen oder Zielauto herausfahren kann
- Wir brauchen also nur $O(n)$ viel Platz

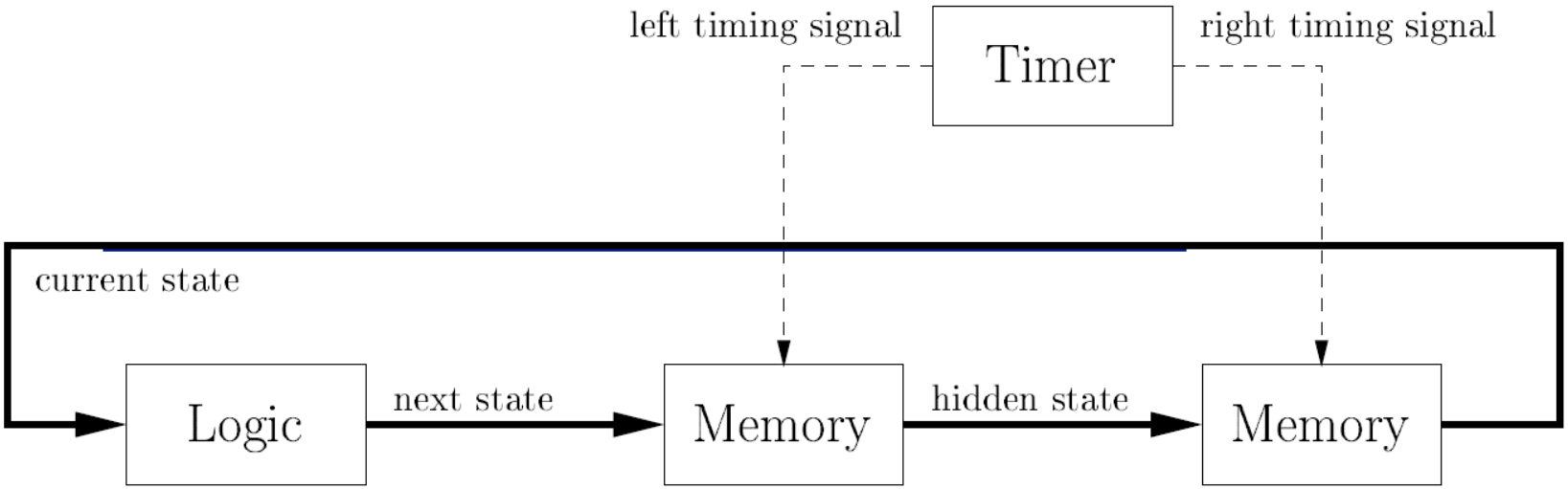
GRH $\in$ PSPACE

- Es reicht eine NTM mit polynomiell viel Platzbedarf anzugeben
- Startkonfiguration mit n Autos auf Turingband kodiert
- Es gibt $2n$ mögliche Nachfolgekonfigurationen
- Wähle nichtdeterministisch eine der Möglichkeiten aus und aktualisiere Turingband
- Solange wiederholen bis wir in Schleife kommen oder Zielauto herausfahren kann
- Wir brauchen also nur $O(n)$ viel Platz
- Also GRH $\in$ PSPACE

Grundidee des Beweises

- Wir zeigen, dass man jede TM $\in$ PSPACE mit einem GRH-Schaltkreis simmulieren kann
- Schaltkreis (Latch) der Zustände abspeichern kann erforderlich
- Konstruktion eines Latch $\Rightarrow$ Rückkopplung im GRH-Schaltkreis
- Taktgenerator (Timer) zum ansteuern der Speicherienheiten

Grundidee des Beweises



- Informationsfluß schrittweise
- Zwei Speicher nach dem master-slave Prinzip
- Aber nur so mächtig wie ein endlicher Automat

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

- Eine Speichereinheit hat 2 mögliche Vorgängerzustände

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

- Eine Speichereinheit hat 2 mögliche Vorgängerzustände
- Kann identisch oder verschieden vom aktuell zu speicherndem Wert sein

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

- Eine Speichereinheit hat 2 mögliche Vorgängerzustände
- Kann identisch oder verschieden vom aktuell zu speicherndem Wert sein
- Zwei verschiedene Autobewegungen die in den aktuellen Zustand geführt haben

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

- Eine Speichereinheit hat 2 mögliche Vorgängerzustände
- Kann identisch oder verschieden vom aktuell zu speicherndem Wert sein
- Zwei verschiedene Autobewegungen die in den aktuellen Zustand geführt haben
- Speicher kann jeden beliebigen Wert annehmen

Probleme mit GRH-Schaltkreisen

- Autos können zu einem beliebigen Zeitpunkt Bewegungen rückwärts ausführen
- Auswirkungen auf die Funktionsweise des Schaltkreises

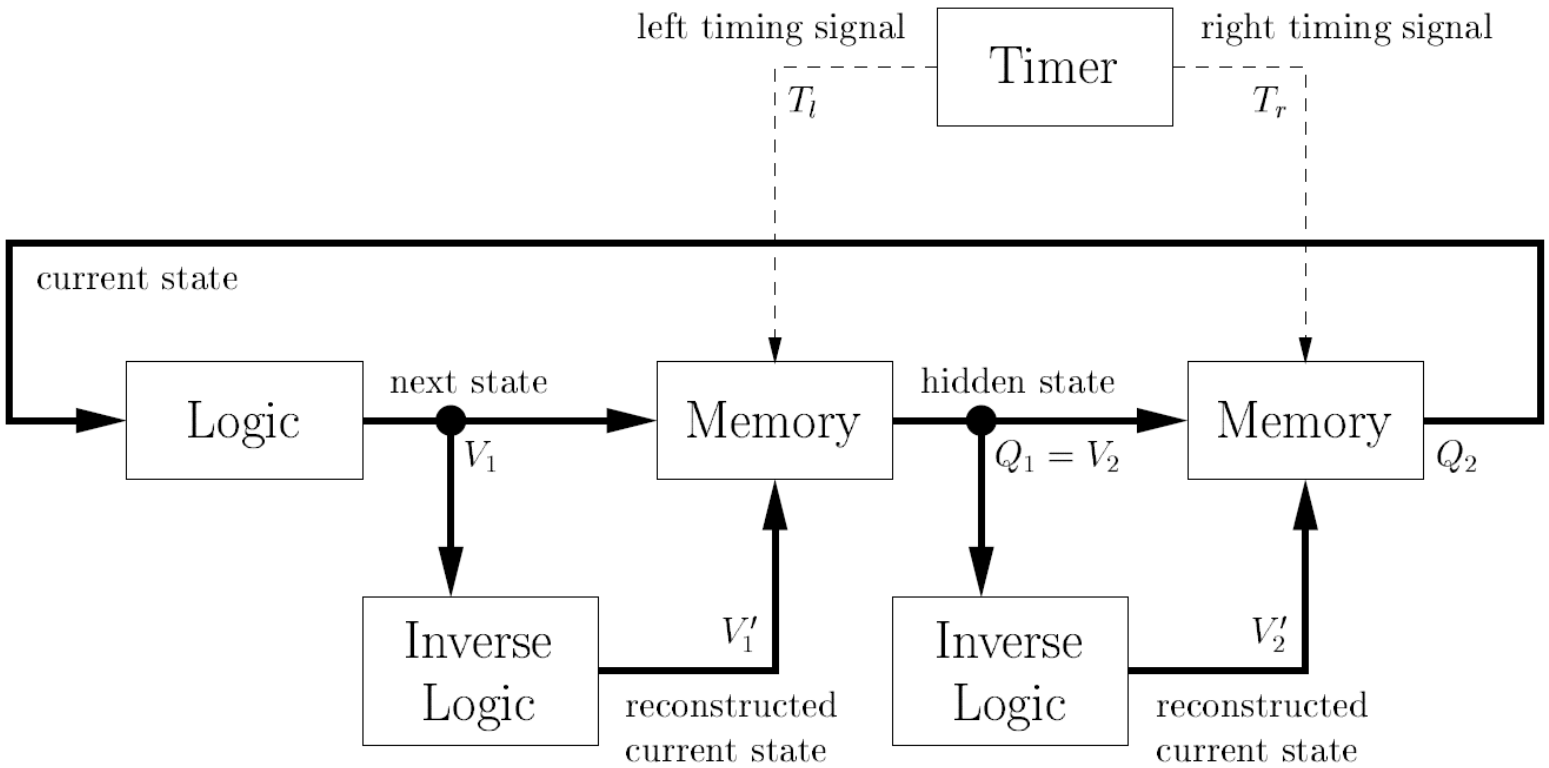
- Eine Speichereinheit hat 2 mögliche Vorgängerzustände
- Kann identisch oder verschieden vom aktuell zu speicherndem Wert sein
- Zwei verschiedene Autobewegungen die in den aktuellen Zustand geführt haben
- Speicher kann jeden beliebigen Wert annehmen

Das Problem ist also, wie bauen wir Speichereinheiten, so dass jeder Speicher nur einen legalen Vorgängerzustand hat?

Umkehrbare Logik

- Ein logischer Ausdruck ist umkehrbar, wenn seine Eingaben eindeutig aus seiner Ausgabe bestimmt werden kann
- Jeder nichtumkehrbare logische Ausdruck kann in einen umkehrbaren Ausdruck umgewandelt werden
- Dabei sind zusätzliche Ein- und Ausgänge erforderlich
- Jeder Schaltkreis kann also auch in einen äquivalenten Schaltkreis mit einem eindeutigen Inversen umgewandelt werden

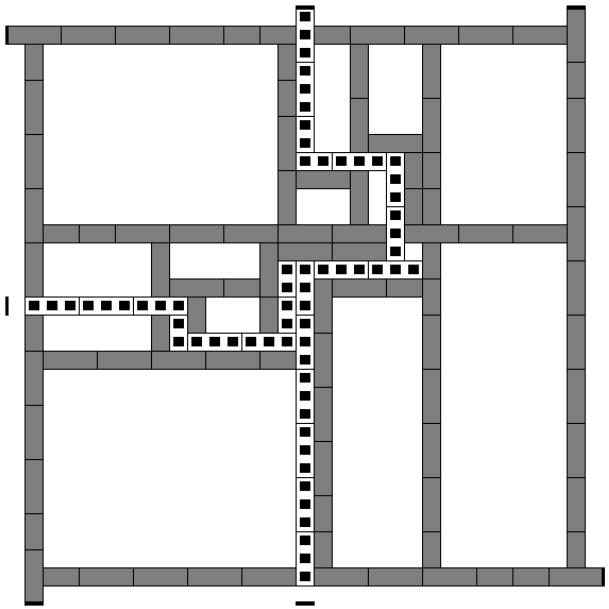
Recheneinheit mit Umkehrbarer Logik



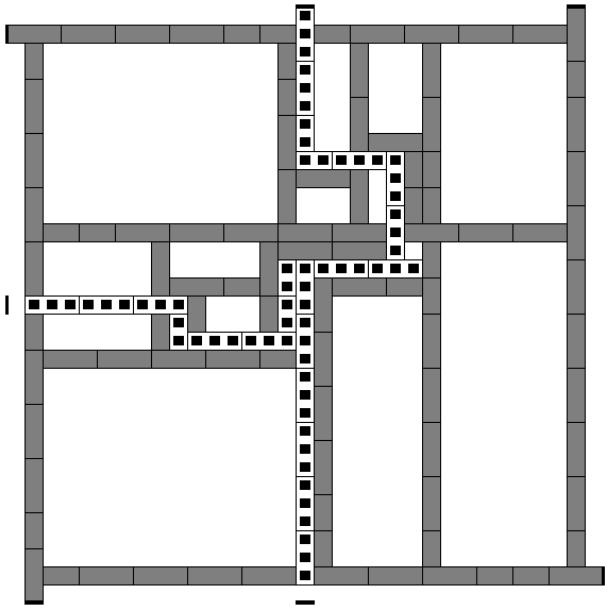
GRH Schaltkreise analysieren

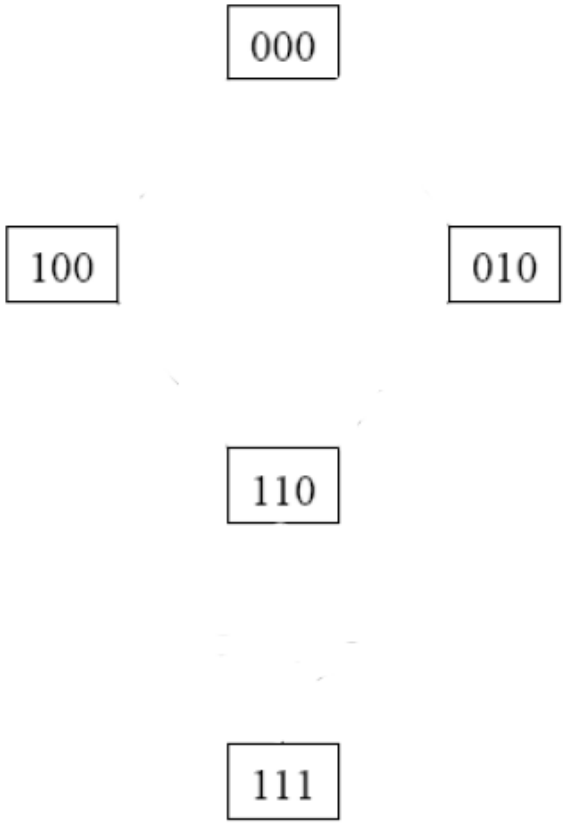
- Zum analysieren der Schaltkreise erzeugen wir uns einen Zustandübergangsgraphen
- Ein Knoten repräsentiert einen legalen Zustand
- Eine Kante repräsentiert einen legalen Zustandübergang

Zustandübergangsgraph des Konjunktiven Baublocks

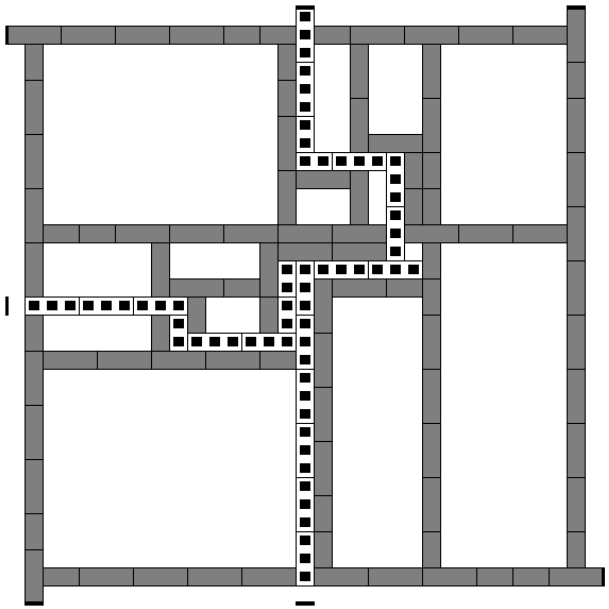


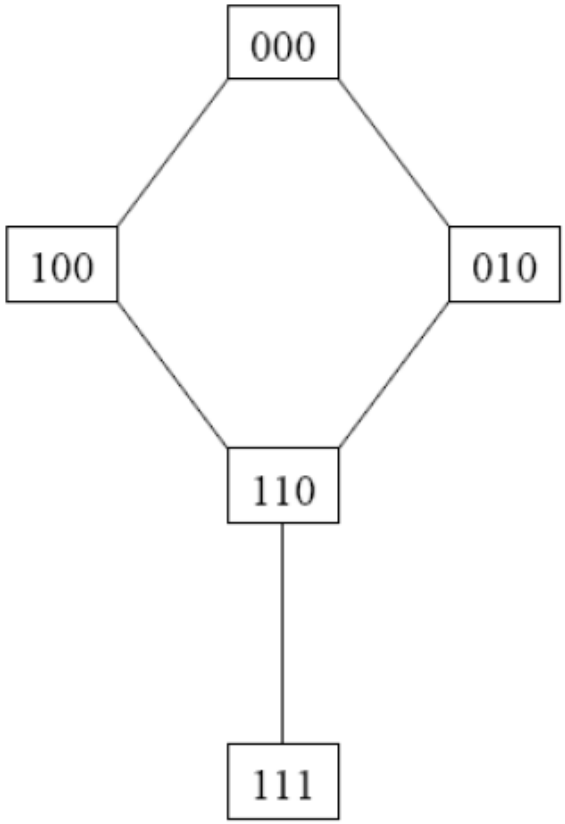
Zustandübergangsgraph des Konjunktiven Baublocks



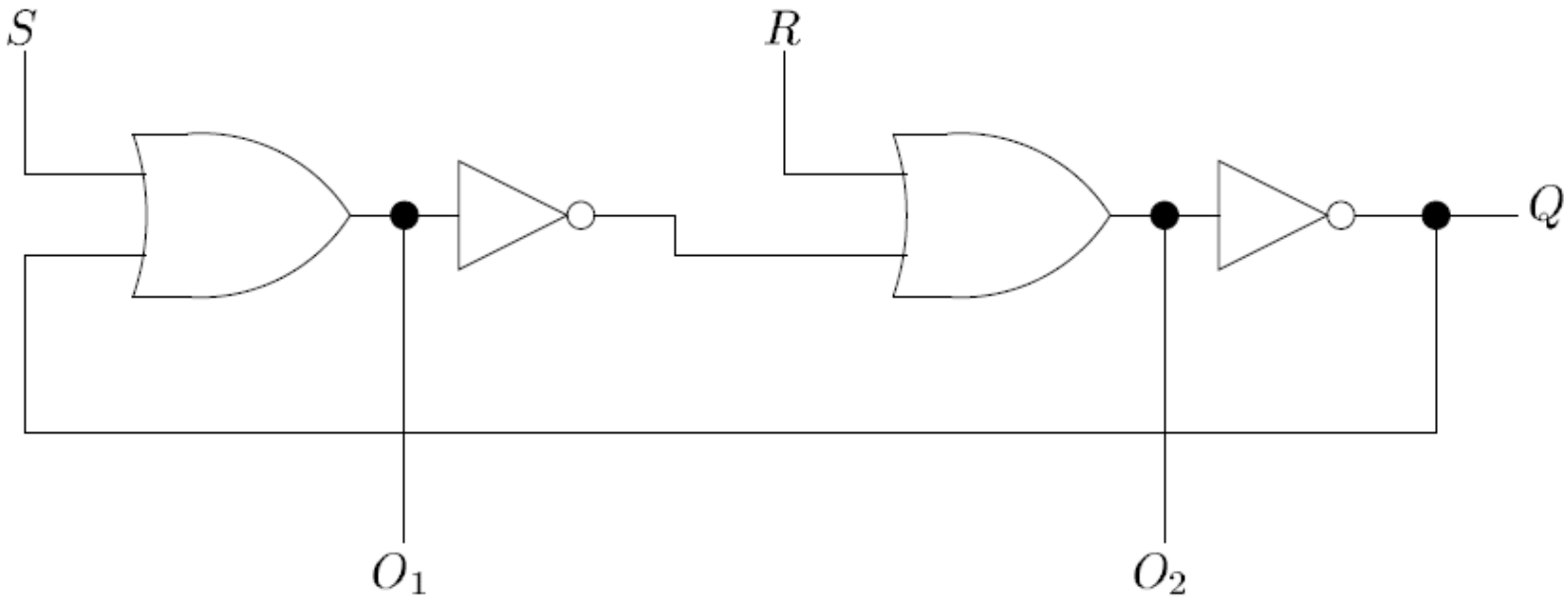


Zustandübergangsgraph des Konjunktiven Baublocks



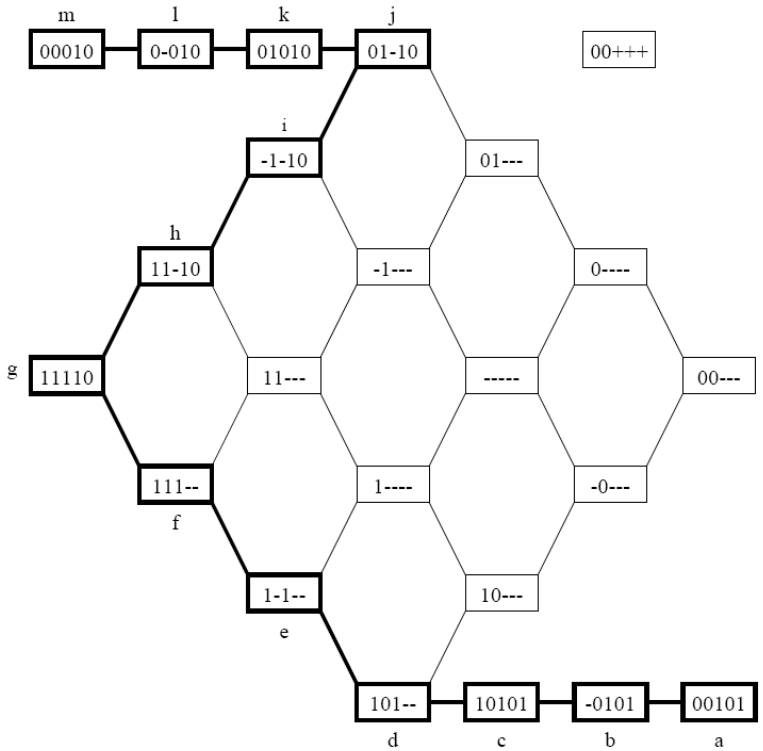


Simple Latch

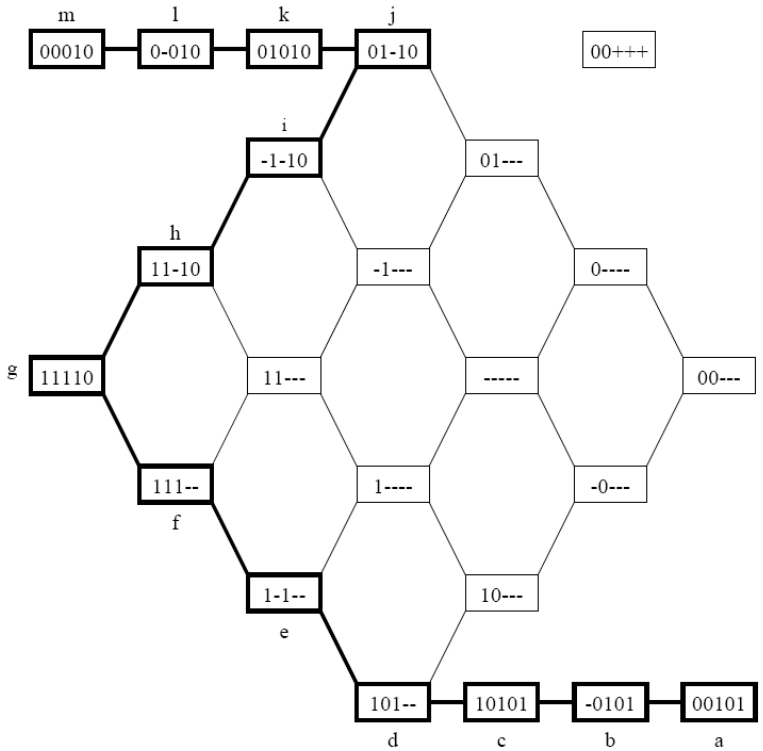


Funktionsweise aber verschieden eines klassischen Latch

Zustandübergangsgraph des Simple Latch

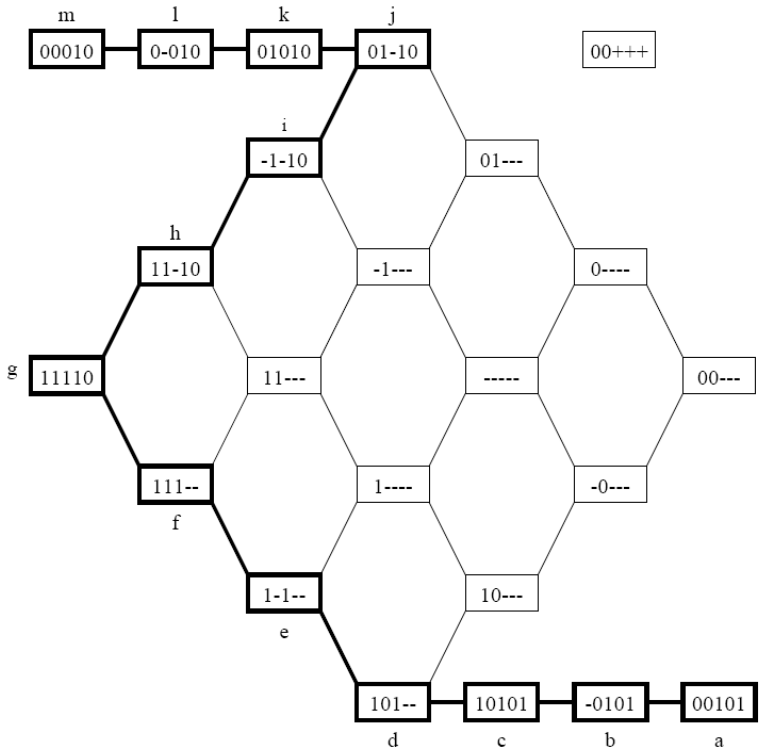


Zustandübergangsgraph des Simple Latch



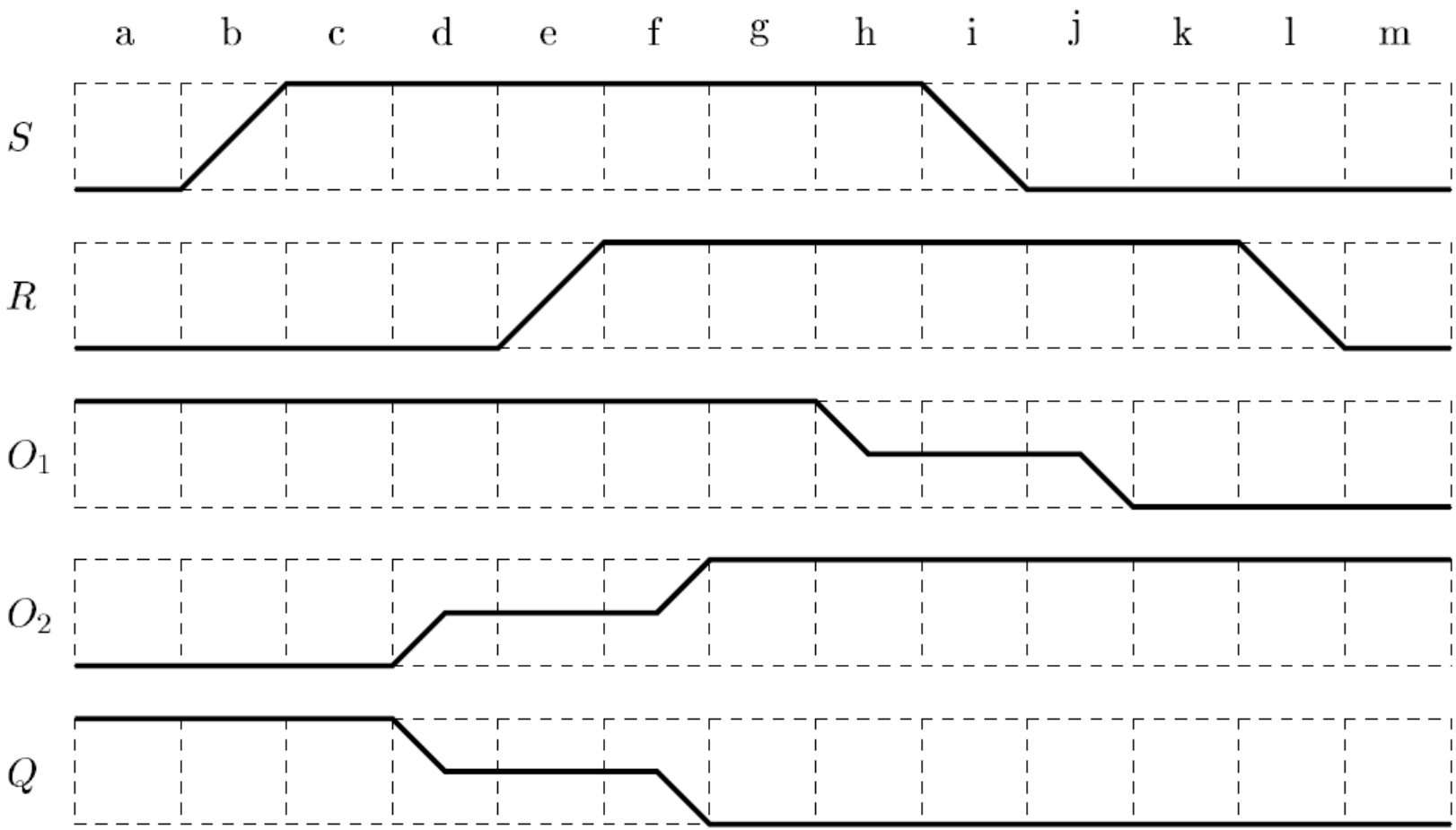
- Der Ausgang kann sich nicht ändern, wenn die Eingänge auf 0 gesetzt sind

Zustandübergangsgraph des Simple Latch



- Der Ausgang kann sich nicht ändern, wenn die Eingänge auf 0 gesetzt sind
- Belegt man nur einen Eingang, so kann man den Ausgang nie verändern

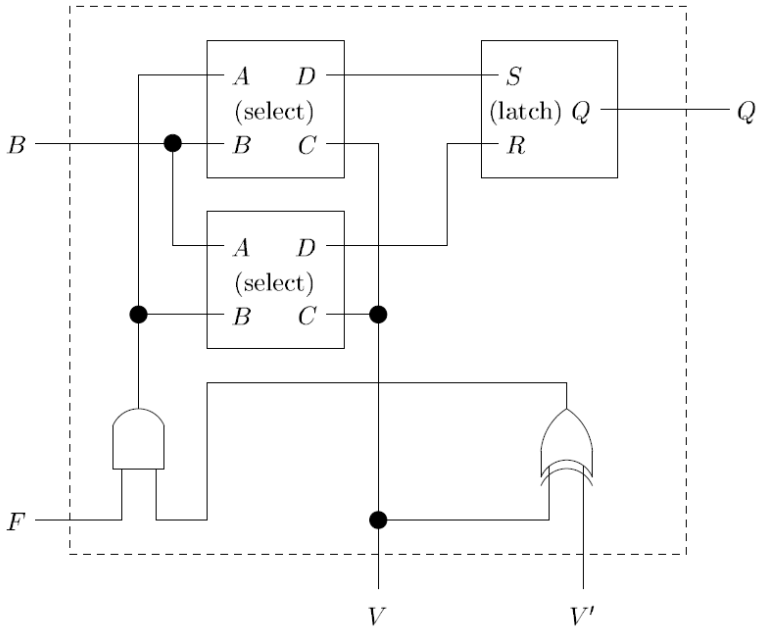
Simple Latch



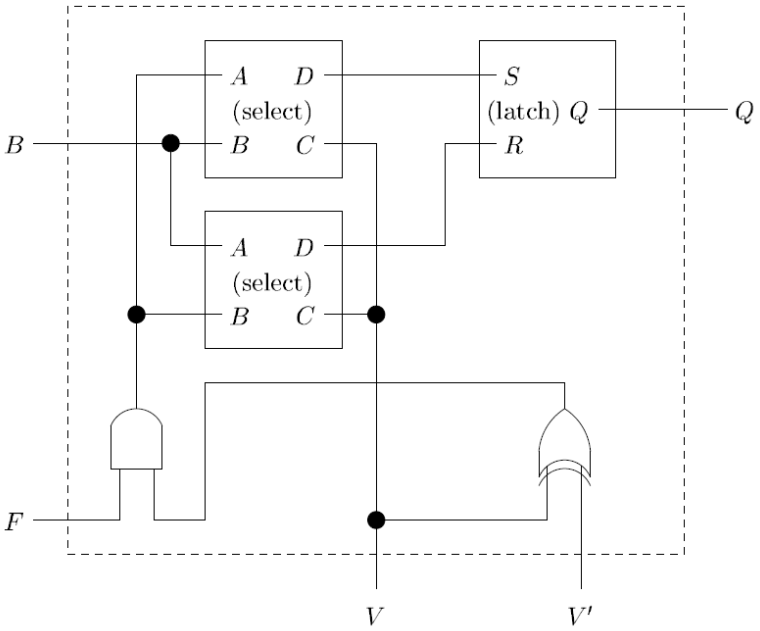
Simple Latch

- Simple Latch reicht für unsere Konstruktion noch nicht aus
- Latch kann seinen Zustand immer nur von 1 auf 0 ändern und umgekehrt
- Insbesondere nicht invertierbar in unserem Sinne
- Wir brauchen einen Timer der ein Puls-Paar erzeugt

Reversible Latch

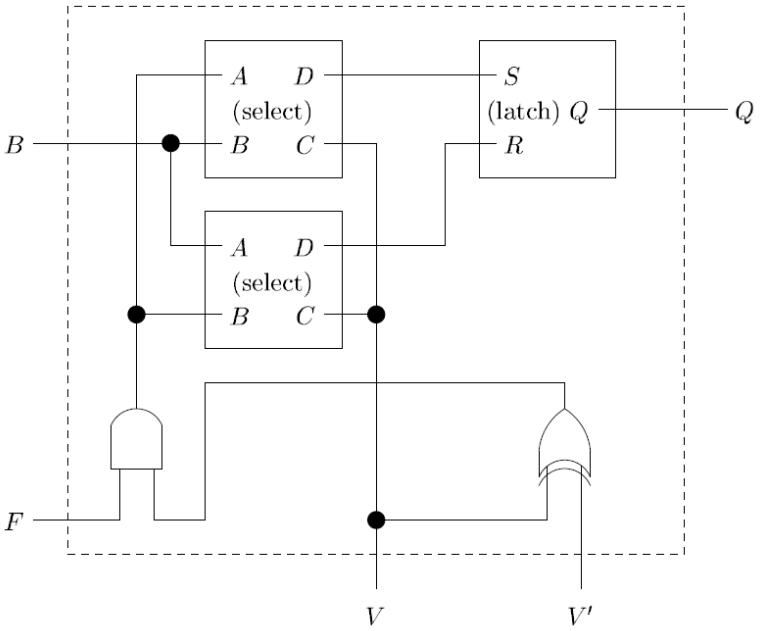


Reversible Latch



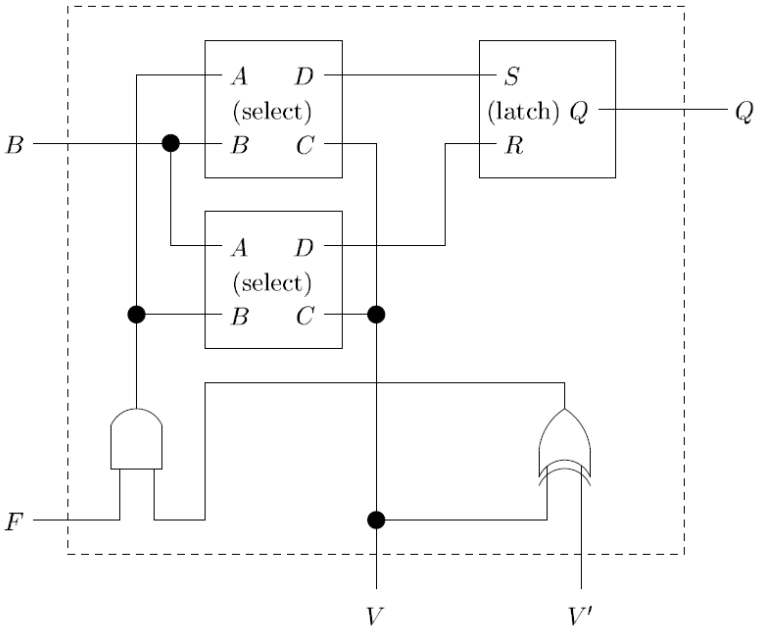
- Kommt der Front-Puls vor dem Back-Puls verhält der Latch sich wie normal
- Der neue Wert V bestimmt die Richtung des Pulspaares

Reversible Latch



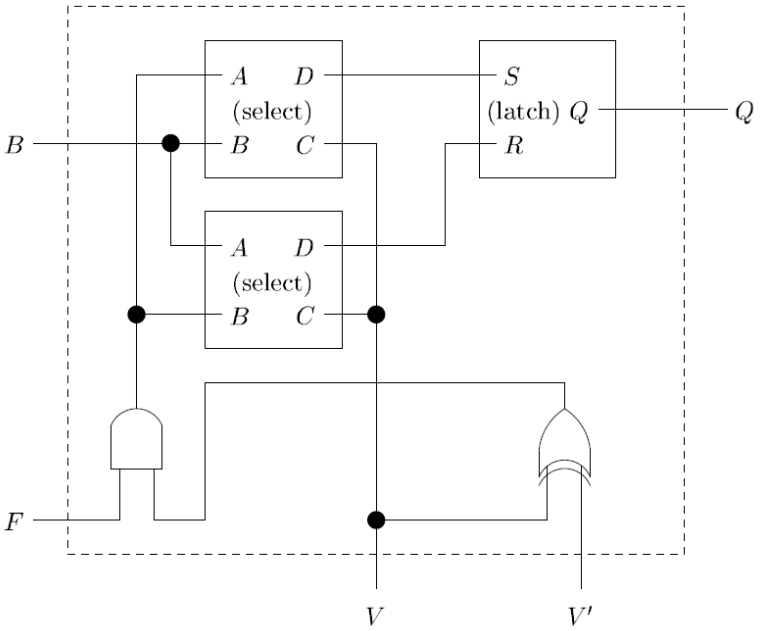
- Durch das XOR-Gate wird der Front-Puls unterdrückt, wenn vorheriger und neuer Wert identisch sind
- Der Latch bleibt dann im alten Zustand

Reversible Latch



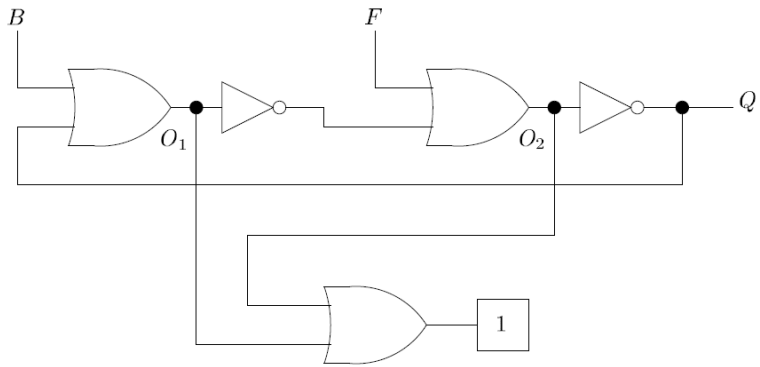
- Front-Puls wird nicht unterdrückt wenn V und V' verschieden sind
- Der Latch wechselt entsprechend den Zustand

Reversible Latch

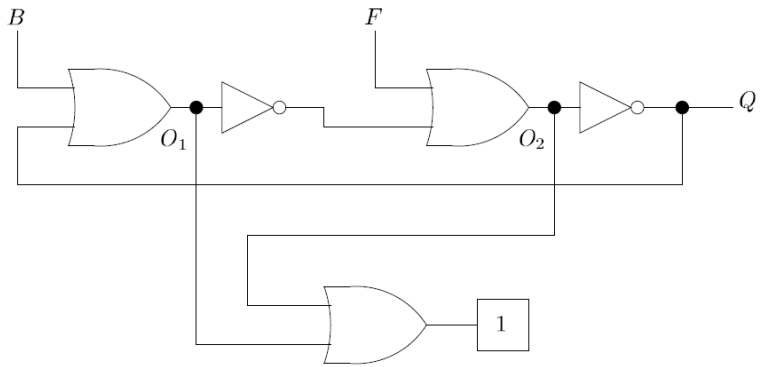


- V und V' sind nach der Durchführung unverändert
- Somit genügen Informationen vorhanden um den vorherigen Wert anzunehmen bei Rückwärtsbewegung

Einfacher Oszillator

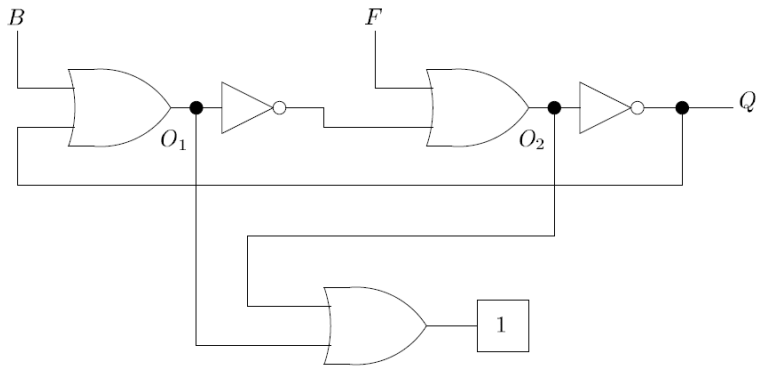


Einfacher Oszillator



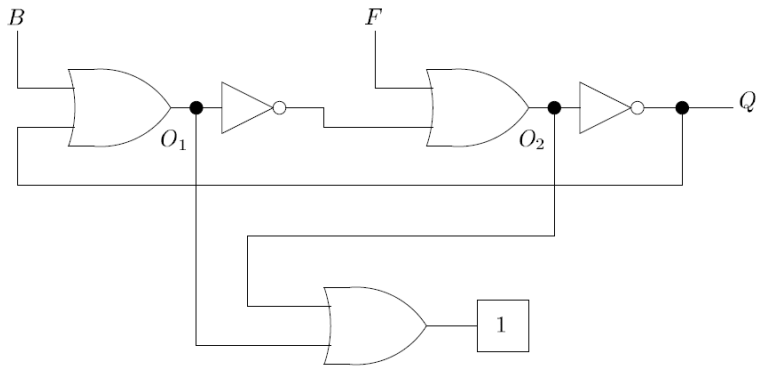
- Durch den festgelockten Oder-Schaltkreis auf 1 muss O_1 oder O_2 den Wert 1 haben
- O_1 und O_2 können nicht gleichzeitig im Zustand – sein

Einfacher Oszillator



- Nach dem Graphen des Simple Latch (2) muss der Oszillator einen Front und Back-Puls erzeugen wenn er seinen Ausgang ändern möchte

Einfacher Oszillator

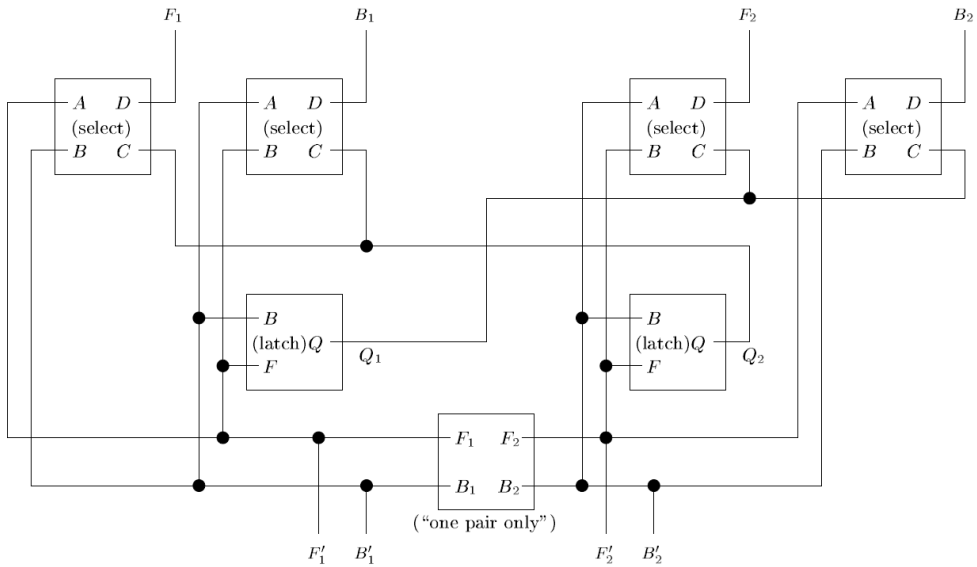


- Nach dem Graphen des Simple Latch (2) muss der Oszillator einen Front und Back-Puls erzeugen wenn er seinen Ausgang ändern möchte

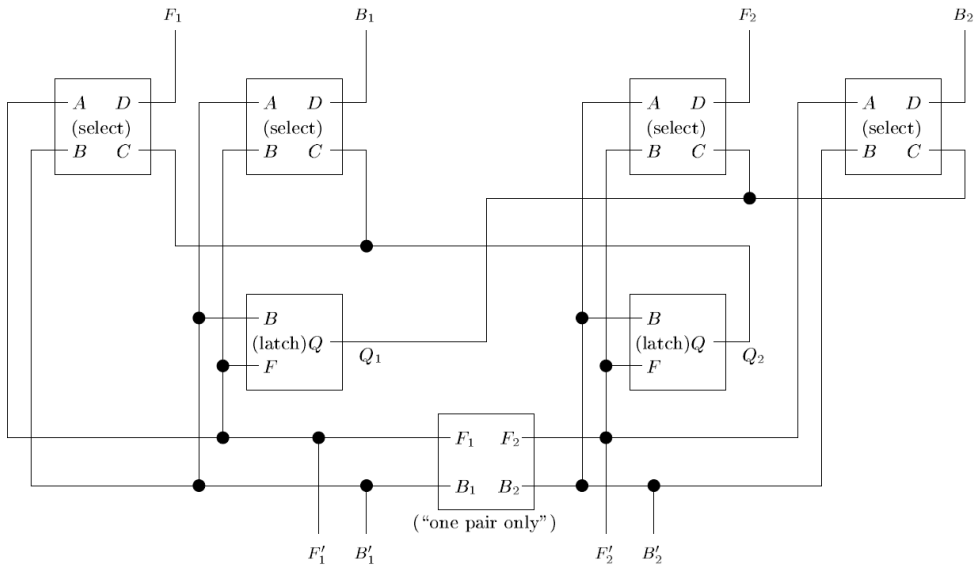
Problem

Haben wir einmal einen Front und Back-Puls erzeugt, so können wir anschließend das gleiche Puls-Paar aber nur in der verkehrten Reihenfolge erzeugen

Dual Timer

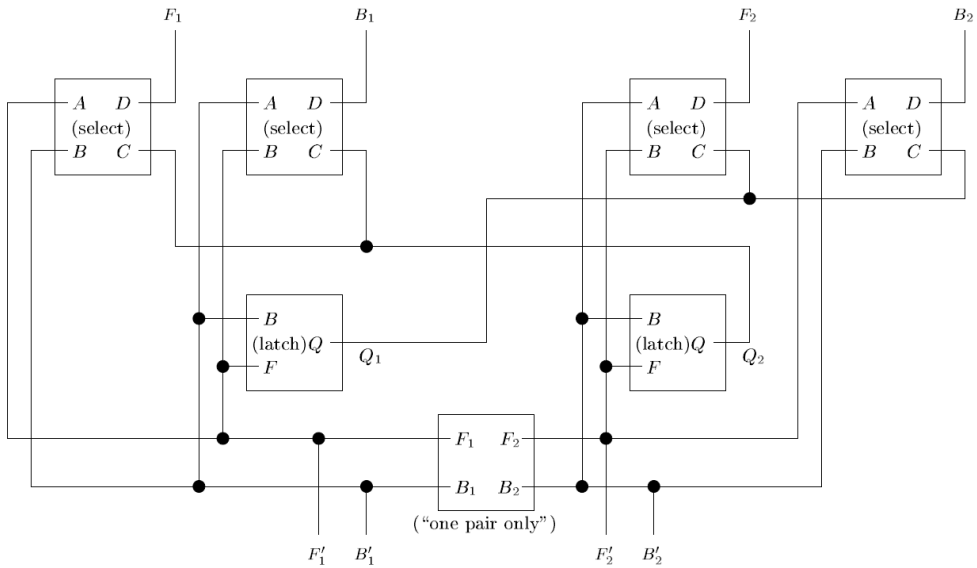

F_1
B_1
F_2
B_2
A D
(select)
B C
B
(latch)Q
F
Q_1
Q_2
F_1 F_2
B_1 B_2
("one pair only")
F_1'
B_1'
F_2'
B_2'

Dual Timer



- Mit One-Pair-Only Schaltkreis entweder F_1' und B_1' aktiv oder F_2' und B_2'
- O_1 und O_2 können sich also nicht gleichzeitig ändern
- 4 Fälle zu betrachten

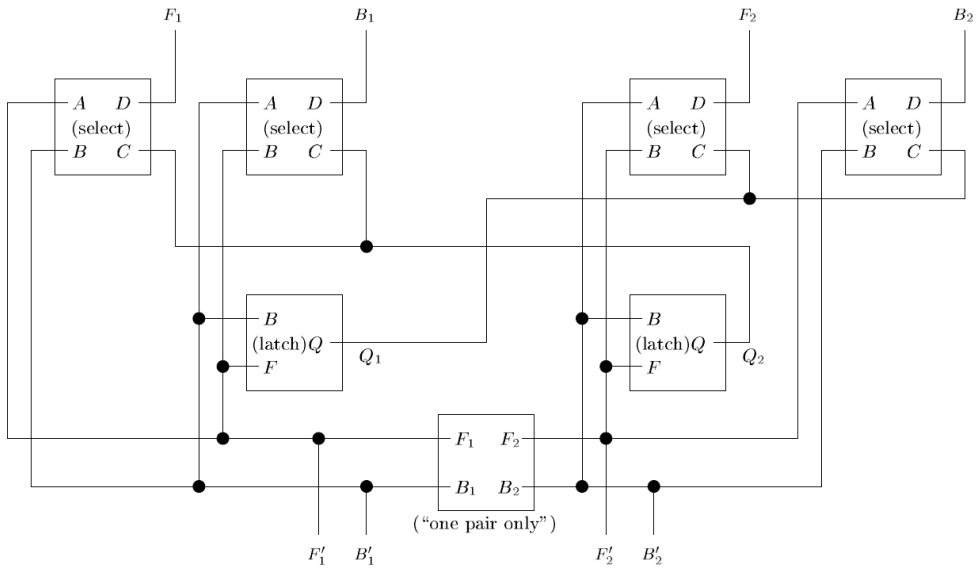
Dual Timer



$Q_1 : 0 \longrightarrow 1, Q_2 : 0 \longrightarrow 0$

- Nur F_1' und B_1' ändern ihre Werte
- Da Q_2 die ganze Zeit 0 ist, korrespondieren die Ausgänge F_1 und B_1 zu F_1' und B_1'.

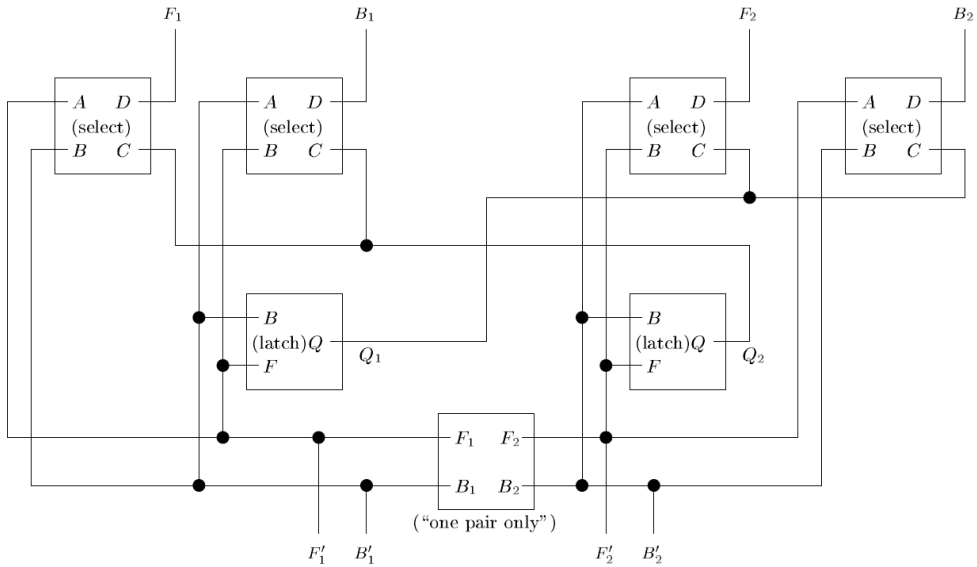
Dual Timer



$Q_1 : 1 \longrightarrow 1, Q_2 : 0 \longrightarrow 1$

- Nur F_2' und B_2' ändern ihre Werte
- Da Q_1 die ganze Zeit 1 ist, korrespondieren die Ausgänge F_2 und B_2 zu F_2' und B_2'.

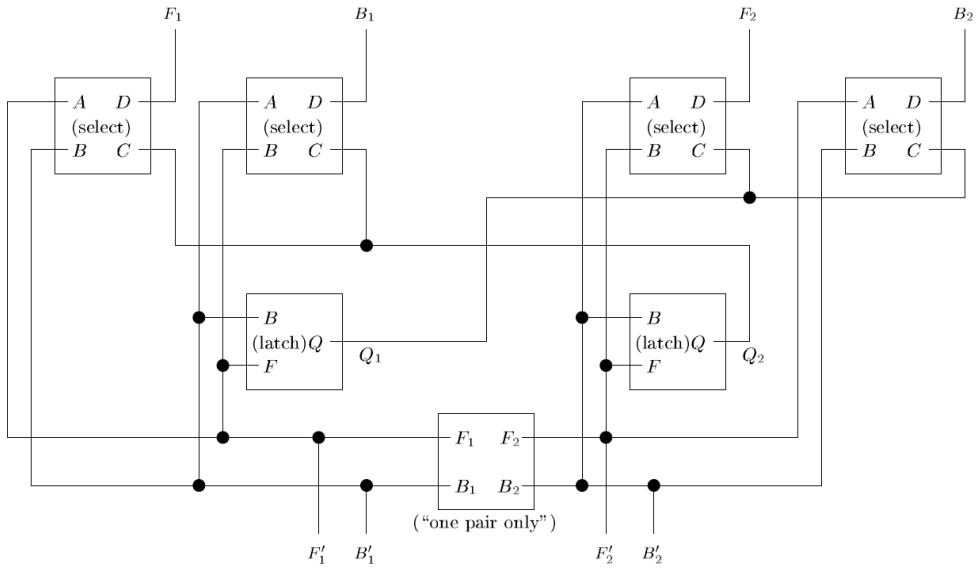
Dual Timer



$Q_1 : 1 \rightarrow 0, Q_2 : 1 \rightarrow 1$

- Eingänge bei Q_1 in umgekehrter Reihenfolge ausgegeben
- Da Q_2 die ganze Zeit 1 ist, korrespondieren die Ausgänge F_1 und B_1 zu B_1' und F_1'.

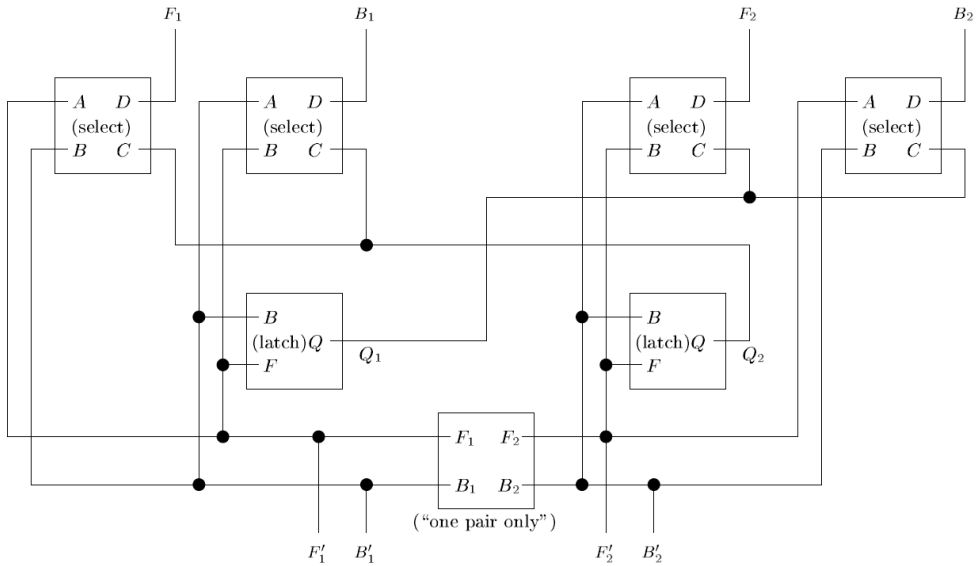
Dual Timer



$Q_1 : 0 \rightarrow 0, Q_2 : 1 \rightarrow 0$

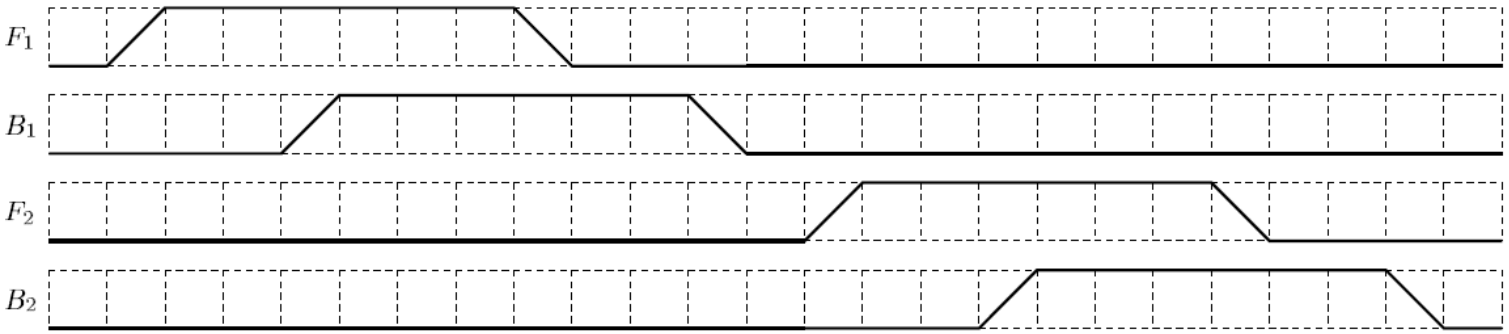
- Eingänge bei Q_2 in umgekehrter Reihenfolge ausgegeben
- Da Q_1 die ganze Zeit 0 ist, korrespondieren die Ausgänge F_2 und B_2 zu B_2' und F_2'.

Dual Timer



Nun sind wir in unserer Ausgangsposition und können wieder von vorne Anfangen

Dual Timer Graph



GRH ist PSPACE-schwer

- Es ist noch nicht klar, wieviel Aufwand man investieren muss, damit alle Speichereinheiten invertierbar sind
- Bislang können wir nur Turingmaschinen simmulieren, jeden Schritt eindeutig rückgängig machen können

GRH ist PSPACE-schwer

- Es ist noch nicht klar, wieviel Aufwand man investieren muss, damit alle Speichereinheiten invertierbar sind
- Bislang können wir nur Turingmaschinen simmulieren, jeden Schritt eindeutig rückgängig machen können

Wieviel Speicher brauchen wir um eine normale Turingmaschine auf einer rückkehrbaren Turingmaschine zu simmulieren?

GRH ist PSPACE-schwer

- Es ist noch nicht klar, wieviel Aufwand man investieren muss, damit alle Speichereinheiten invertierbar sind
- Bislang können wir nur Turingmaschinen simmulieren, jeden Schritt eindeutig rückgängig machen können

Wieviel Speicher brauchen wir um eine normale Turingmaschine auf einer rückkehrbaren Turingmaschine zu simmulieren?

C.H. Bennet hat gezeigt, dass man für jedes $\epsilon > 0$ eine TM mit Zeit- und Platzbedarf T und S auf einer rückkehrbaren TM mit $O(T^{1+\epsilon})$ und $O(S \log T)$ simmulieren kann

GRH ist PSPACE-schwer

- Man kann also TM auf einer rückkehrbaren TM mit polynomiell viel Speicherplatz simmulieren

GRH ist PSPACE-schwer

- Man kann also TM auf einer rückkehrbaren TM mit polynomiell viel Speicherplatz simmulieren
- Somit können wir diese rückkehrbare TM mit unserem GRH Schaltkreis simmulieren

GRH ist PSPACE-schwer

- Man kann also TM auf einer rückkehrbaren TM mit polynomiell viel Speicherplatz simmulieren
- Somit können wir diese rückkehrbare TM mit unserem GRH Schaltkreis simmulieren
- Dies geht für jede TM aus PSPACE

GRH ist PSPACE-schwer

- Man kann also TM auf einer rückkehrbaren TM mit polynomiell viel Speicherplatz simmulieren
- Somit können wir diese rückkehrbare TM mit unserem GRH Schaltkreis simmulieren
- Dies geht für jede TM aus PSPACE
- GRH ist also PSPACE-schwer und damit PSPACE vollständig

Quellen

Gary W. Flake, Eric B. Baum:
Rush Hour is PSPACE-complete, or Why you should generously tip parking lot attendants.
http://citeseer.ist.psu.edu/266206.html

Rush Hour
http://www.puzzles.com/products/RushHour/RushHourApp.htm

C. H. Bennet:
Time/Space trade-offs for reversible computation. *SIAM J. Computing*, 18(4):766-776, 1989.

Fragen?

Danke!