Algorithmen in Zellularautomaten

7. Synchronisation

Thomas Worsch

Fakultät für Informatik
Institut für Theoretische Informatik

Sommersemester 2020

Überblick

- Das klassische Firing Squad Synchronization Problem
- General an beliebiger Stelle
- Zweidimensionales FSSP
- Ausblick

Geschichte

- laut Waksman (1966):
  «The problem known as the ‹firing squad synchronization problem› was devised about the year 1957 by J. Myhill. It first appeared in print in a paper by E. F. Moore [...]»

  «The problem first arose in connection with causing all parts of a self-reproducing machine to be turned on simultaneously, and was first solved by J. McCarthy and M. Minsky.»

Geschichte

- laut Waksman (1966):
  «The problem known as the ‹firing squad synchronization problem›
  was devised about the year 1957 by J. Myhill.
  It first appeared in print in a paper by E. F. Moore [...]»

  «The problem first arose in connection with causing
  all parts of a self-reproducing machine to be turned on simultaneously,
  and was first solved by J. McCarthy and M. Minsky.»

- erste «zitierfähige» — jedenfalls zitierte — Lösung:
  Eiichi Goto:
  A minimal time solution of the firing squad problem.
  Dittoed course notes for Applied Mathematics 298,
  Harvard University, (1962), pp. 52-59.

Problem (Firing Squad Synchronisation Problem, FSSP)

gegeben
- $Q' = \{\#, g, \mathtt{s}, \mathtt{f}\}$ und $N = H_1^{(1)}$

gesucht
- Zellularautomat mit $Q \supseteq Q'$ und δ, so dass gilt:
  - $\{\#, \mathtt{s}\}$ ist passiv und # ist tot.
  - C überführt jede Konfiguration der Form $\#\mathtt{gss}\cdots\mathtt{s}\#$ in die Konfiguration $\#\mathtt{fff}\cdots\mathtt{f}\#$ mit gleichem Träger,
  - und zwar so, dass dabei in keiner der vorher auftretenden Konfigurationen der Zustand $\mathtt{f}$ vorkommt.

Problem (Firing Squad Synchronisation Problem, FSSP)

gegeben

- $Q' = \{\#, \mathrm{g}, \mathtt{s}, \mathtt{f}\}$ und $N = H_1^{(1)}$

gesucht

- Zellularautomat mit $Q \supseteq Q'$ und δ, so dass gilt:
  - $\{\#, \mathtt{s}\}$ ist passiv und $\#$ ist tot.
  - C überführt jede Konfiguration der Form $\#\mathtt{gss}\cdots\mathtt{s}\#$ in die Konfiguration $\#\mathtt{fff}\cdots\mathtt{f}\#$ mit gleichem Träger,
  - und zwar so, dass dabei in keiner der vorher auftretenden Konfigurationen der Zustand $\mathtt{f}$ vorkommt.

beachte

- Aktivität kann sich nur vom «General» g ausbreiten
- immer gleiches Q, gleiches δ unabhängig vom Träger

Algorithmus (Balzer)

Algorithmus (Balzer)

(1> >									
(2>	>								
(3>		>							
(	1>		>						
(	2>			>					
(	3>				>				
(		1>				>			
(		2>					>		
(		3>						>	
(			1>						<)
(			2>					<	)
(			3>				<		)
(				1>		<			)
(				2>	<				)
(				< <1)	(1> >				)

Algorithmus (Balzer)

(1> >									
(2>	>								
(3>		>							
(	1>		>						
(	2>			>					
(	3>				>				
(		1>				>			
(		2>					>		
(		3>						>	
(			1>						<)
(			2>					<	)
(			3>				<		)
(				1>		<			)
(				2>	<				)
(				< <1)	(1> >				)
(			<	<2)	(2>	>			)
(		<		<3)	(3>		>		)
(	<		<1	)	(	1>		>	)
(>			<2	)	(	2>			<)
(	>		<3	)	(	3>		<	)
(		«1)(1»		)	(		«1)(1»		)

Algorithmus (Balzer)

(1> >									
(2>	>								
(3>		>							
(	1>		>						
(	2>			>					
(	3>				>				
(		1>				>			
(		2>					>		
(		3>						>	
(			1>						<)
(			2>					<	)
(			3>				<		)
(				1>		<			)
(				2>	<				)
(				< <1)	(1> >				)
(			<	<2)	(2>	>			)
(		<		<3)	(3>		>		)
(	<		<1	)	(	1>		>	)
(>			<2	)	(	2>			<)
(	>		<3	)	(	3>		<	)
(		«1) (1»		)	(		«1) (1»		)
(	<	<2) (2>	>	)	(	<	<2) (2>	>	)
(>		<3) (3>		<)	(>		<3) (3>		<)
(	«1) (1»	) (	«1) (1»	)	(	«1) (1»	) (	«1) (1»	)

Algorithmus (Balzer)

(1> >	s	s	s	s	s	s	s	s	s
(2>	>	s	s	s	s	s	s	s	s
(3>		>	s	s	s	s	s	s	s
(	1>		>	s	s	s	s	s	s
(	2>			>	s	s	s	s	s
(	3>				>	s	s	s	s
(		1>				>	s	s	s
(		2>					>	s	s
(		3>						>	s
(			1>						<)
(			2>					<	)
(			3>				<		)
(				1>		<			)
(				2>	<				)
(				< <1)	(1> >				)
(			<	<2)	(2>	>			)
(		<		<3)	(3>		>		)
(	<		<1	)	(	1>		>	)
(>			<2	)	(	2>			<)
(	>		<3	)	(	3>		<	)
(		«1) (1»		)	(		«1) (1»		)
(	<	<2) (2>	>	)	(	<	<2) (2>	>	)
(>		<3) (3>		<)	(>		<3) (3>		<)
(	«1) (1»	) (	«1) (1»	)	(	«1) (1»	) (	«1) (1»	)
f	f	f	f	f	f	f	f	f	f

Zeitbedarf (von Balzers Algorithmus)

- für die Synchronisation von n Zellen «ungefähr»

$$t(n) \approx \frac{3}{2}n + t(\frac{n}{2}) \ .$$

- also

$$t(n) = 3n + \text{Terme niedrigerer Ordnung} \ .$$

- optimal?

Untere Zeitschranke für das FSSP (Waksman 1966)

Satz Es gibt keinen Zellularautomaten,
der das FSSP für alle n löst und
für mindestens eine Problemgröße $k \geq 2$
höchstens $2k - 3$ Schritte benötigt.

Beweis

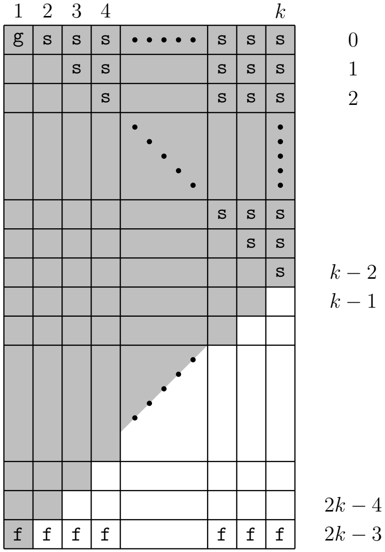
indirekt:

- gegeben: ein Zellularautomat C
- Annahme: C löst das FSSP für eine Problemgröße k in höchstens $2k - 3$ Schritten
- zeige: C löst *nicht* das FSSP für eine andere Problemgröße

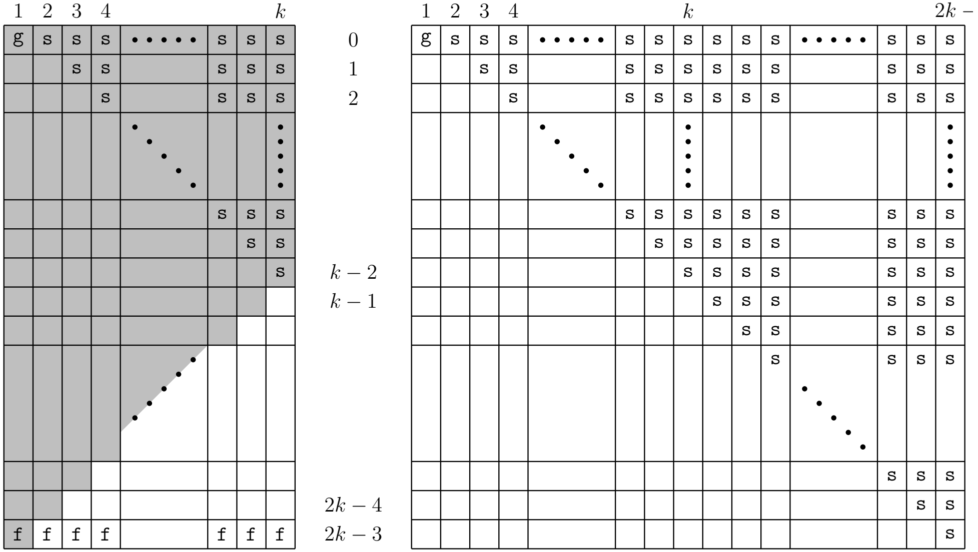
Beweis

1	2	3	4				k	
g	s	s	s	$\cdots\cdots$	s	s	s	0
		s	s		s	s	s	1
			s		s	s	s	2
				$\ddots$			$\vdots$	
					s	s	s	
						s	s	
							s	$k-2$
								$k-1$
								$2k-4$
f	f	f	f		f	f	f	$2k-3$

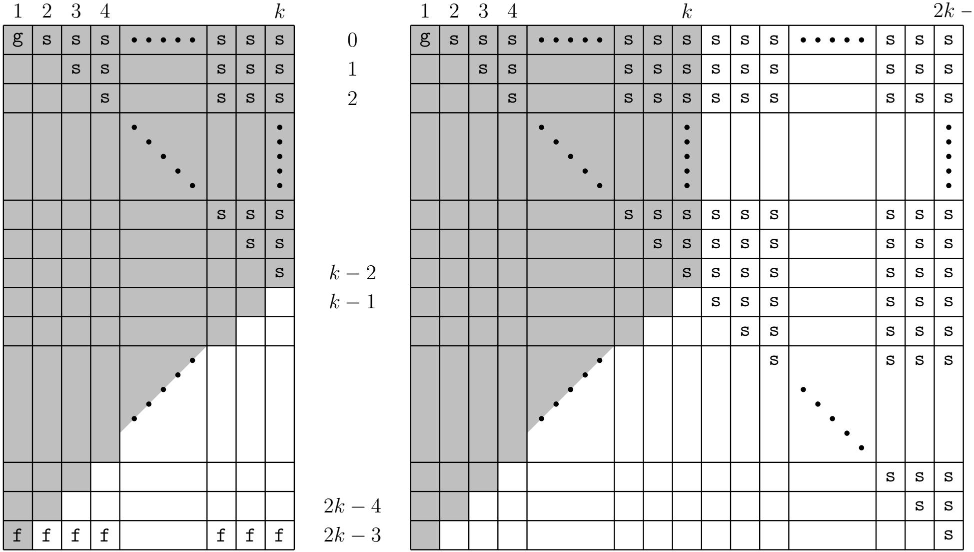
Beweis



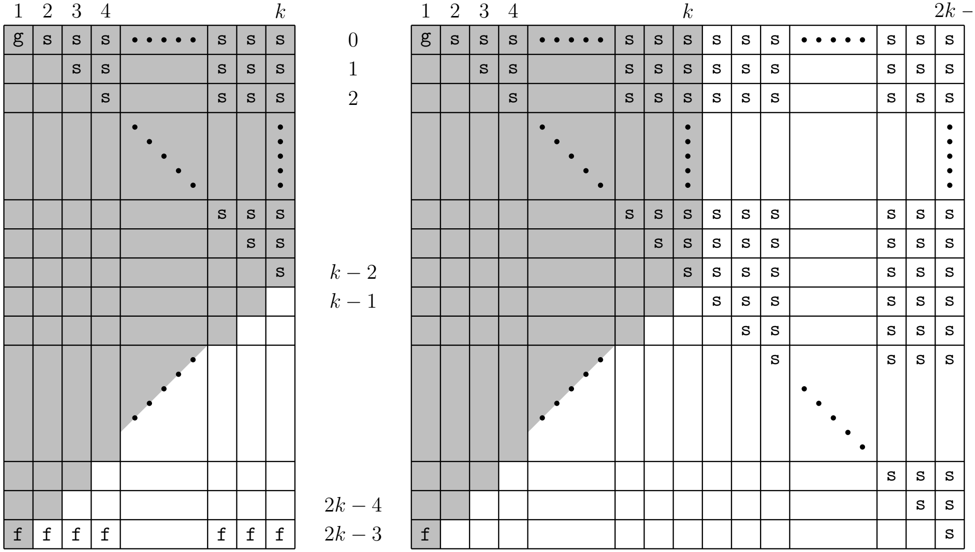
Beweis



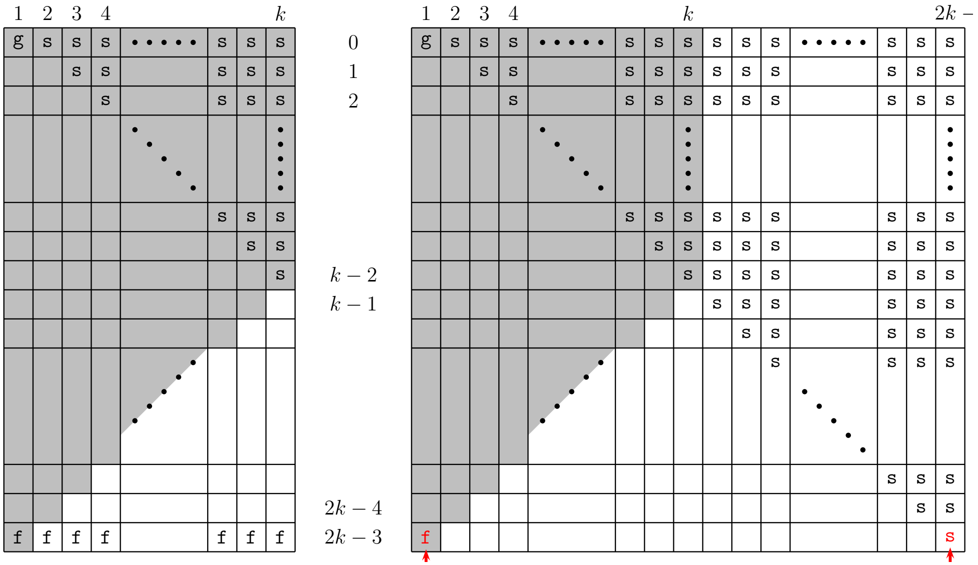
Beweis



Beweis



Beweis


1 2 3 4 k
g s s s • • • • • s s s
f f f f f f f
0
1
2
k − 2
k − 1
2k − 4
2k − 3
1 2 3 4 k 2k −
g s s s • • • • • s s s s s s • • • • • s s s
f
s

Zeitoptimale Algorithmen für das FSSP

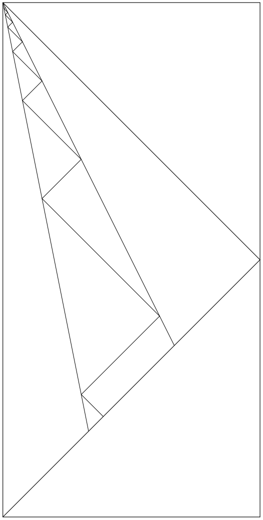
Satz (Goto, 1962)

- Es gibt einen Zellularautomaten, der das FSSP für alle $n \geq 2$ in genau $2n - 2$ Schritten löst.

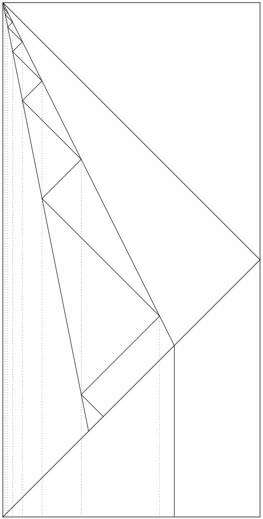
Satz (Mazoyer, 1987)

- Man kommt dabei mit nur 7 Zuständen aus, also außer #, g, `s`, `f` nur 3 weitere.

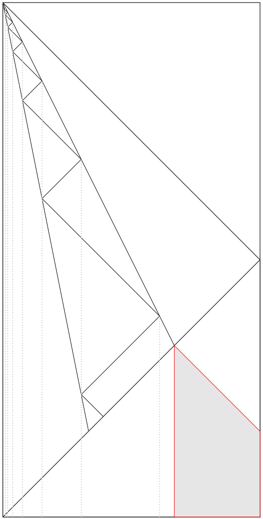
Algorithmus (Gerken)



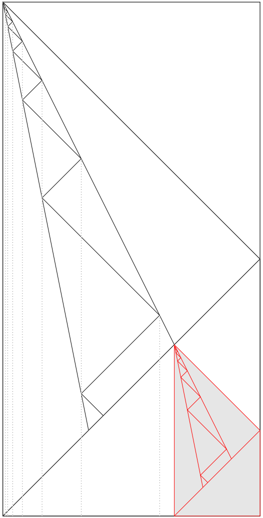
Algorithmus (Gerken)



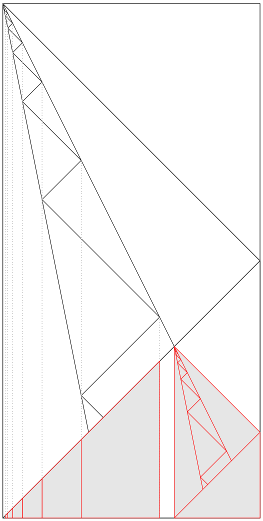
Algorithmus (Gerken)



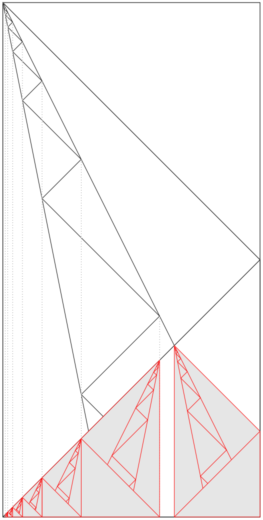
Algorithmus (Gerken)



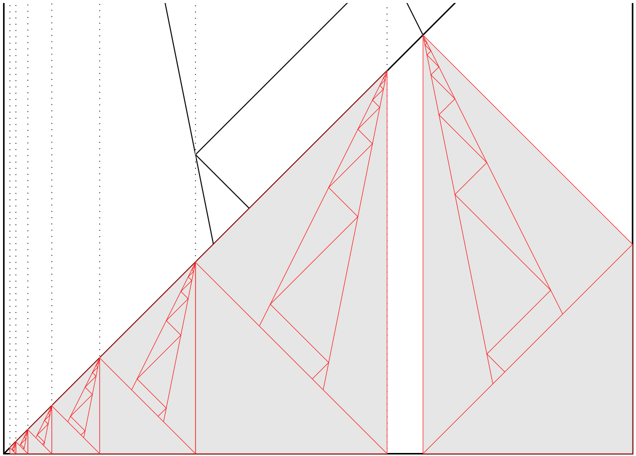
Algorithmus (Gerken)



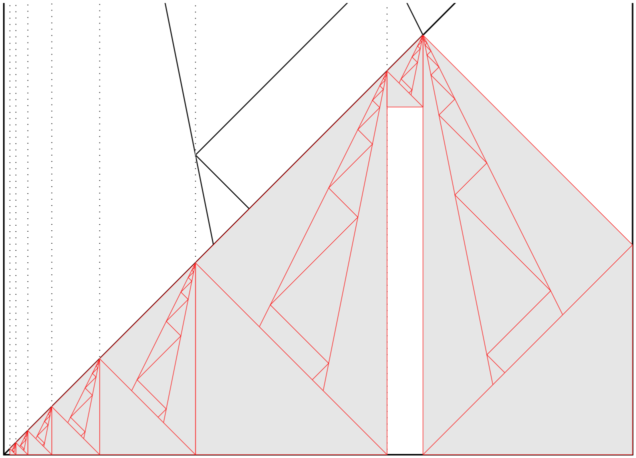
Algorithmus (Gerken)



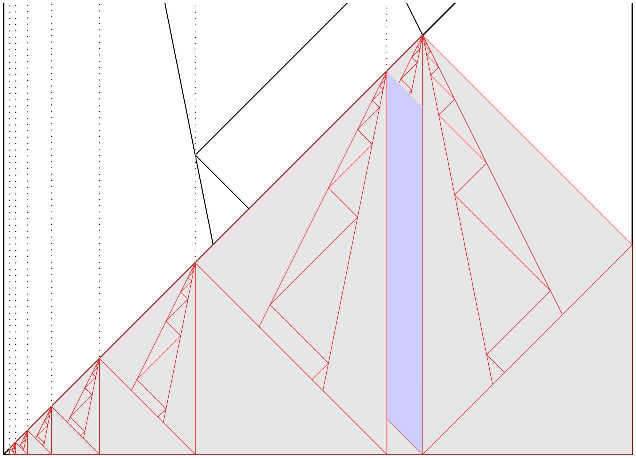
Algorithmus (Gerken)



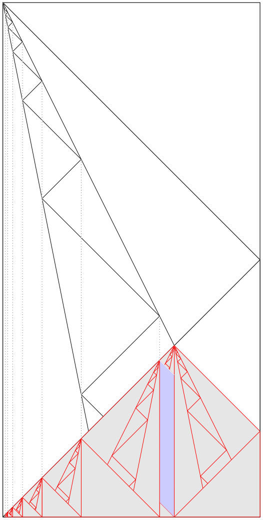
Algorithmus (Gerken)



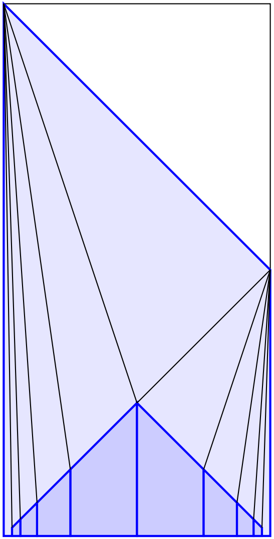
Algorithmus (Gerken)



Algorithmus (Gerken)



Algorithmus (Waksman)

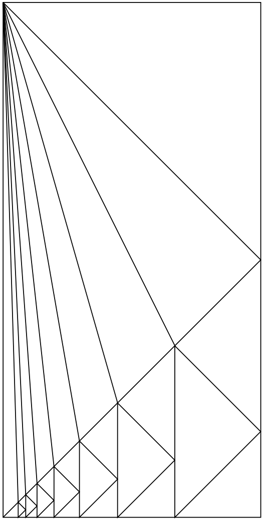


- unbeschränkt viele
  verschiedene Signalgeschwindigkeiten !
- mit endlich vielen Zuständen !?

Algorithmus (viele verschieden schnelle Signale)

>											
	<>										
1		<>									
1	<0		<>								
1	0	<		<>							
1	1		<		<>						
1	1	<0		<		<>					
1	<0	0	<		<		<>				
1	0	1		<		<		<>			
1	0	1	<0		<		<		<>		
1	0	<	0	<		<		<		<>	
1	1		1		<		<		<		<>
1	1		1	<0		<		<		<	
1	1		<	0	<		<		<		<
1	1	<0		1		<		<		<	
1	<0	0		1	<0		<		<		<
1	0	0		<	0	<		<		<	
1	0	0	<		1		<		<		<
1	0	1			1	<0		<		<	
1	0	1			<	0	<		<		<
1	0	1		<		1		<		<	
1	0	1	<0			1	<0		<		<
1	0	<	0			<	0	<		<	
1	1		0		<		1		<		<
1	1		0	<			1	<0		<	
1	1		1				<	0	<		<
1	1		1			<		1		<	
1	1		1		<			1	<0		<
1	1		1	<0				<	0	<	
1	1		<	0			<		1		<
1	1	<0		0		<			1	<0	
1	<0	0		0	<				<	0	<
1	0	0		1				<		1	
1	0	0		1			<			1	<0
1	0	0		1		<				<	0
1	0	0		1	<0				<		1

Algorithmus (Mazoyer)



dafür genügen 7 Zustände !

Algorithmus (Mazoyer)

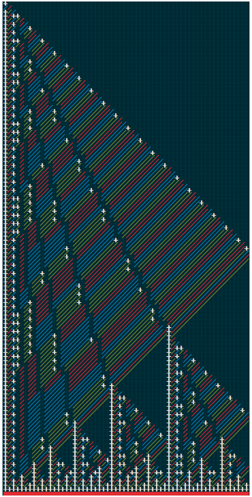


Bild: C. Wellenbrock (60 Zellen)

und nun … ? …

Verallgemeinerungen

Welche fallen Ihnen ein?

Überblick

- Das klassische Firing Squad Synchronization Problem
- General an beliebiger Stelle
- Zweidimensionales FSSP
- Ausblick

FSSP mit General an beliebiger Stelle

Problem

- gegeben: $Q' = \{\#, \mathrm{g}, \mathtt{s}, \mathtt{f}\}$ und $N = H_1^{(1)}$
- gesucht: Zellularautomat mit $Q \supseteq Q'$ und δ, so dass gilt:
  - $\{\#, \mathtt{s}\}$ ist passiv und # ist tot.
  - C überführt
    - jede Konfiguration der Form $\#\mathtt{ss}\cdots\mathtt{sgs}\cdots\mathtt{s}\#$
    - unabhängig von der Position von g
    - in die Konfiguration $\#\mathtt{fff}\cdots\mathtt{f}\#$
    - mit gleichem Träger,
  - und zwar so, dass dabei in keiner der vorher auftretenden Konfigurationen der Zustand $\mathtt{f}$ vorkommt.

FSSP mit General an beliebiger Stelle

Problem

- gegeben: $Q' = \{\#, \mathrm{g}, \mathtt{s}, \mathtt{f}\}$ und $N = H_1^{(1)}$
- gesucht: Zellularautomat mit $Q \supseteq Q'$ und δ, so dass gilt:
  - $\{\#, \mathtt{s}\}$ ist passiv und # ist tot.
  - C überführt
    - jede Konfiguration der Form $\#\mathtt{ss}\cdots\mathtt{sgs}\cdots\mathtt{s}\#$
    - unabhängig von der Position von g
    - in die Konfiguration $\#\mathtt{fff}\cdots\mathtt{f}\#$
    - mit gleichem Träger,
  - und zwar so, dass dabei in keiner der vorher auftretenden Konfigurationen der Zustand $\mathtt{f}$ vorkommt.

Untere Schranken?
Algorithmen?

Zeitbedarf für FSSP mit General an beliebiger Stelle

- es sei
  - n die Größe des Trägers und
  - k die Länge des kürzeren Abschnittes neben g

Satz

- Das FSSP mit dem General an beliebiger Stelle ist in $2n - 2 - k$ Schritten lösbar.

Satz

- Diese Zeit ist für $n \geq 2$ optimal.

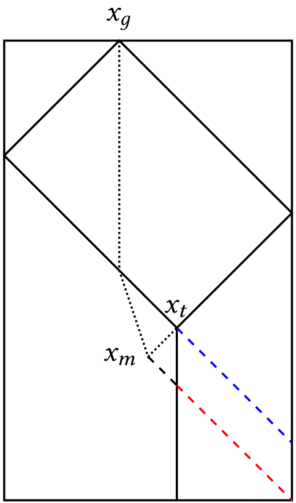
Beweisidee für untere Schranke

- analog zu vorhin:

  Jedes Ende muss «wissen», wie weit weg das andere ist.

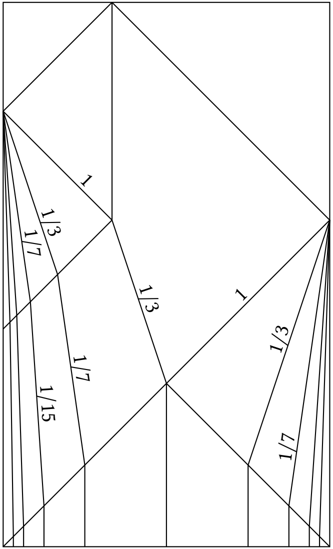
- *nicht* nur der General muss Bescheid wissen,
  deswegen *nicht* Zeit $2(n-k)-2$ ausreichend

Algorithmus (TW)

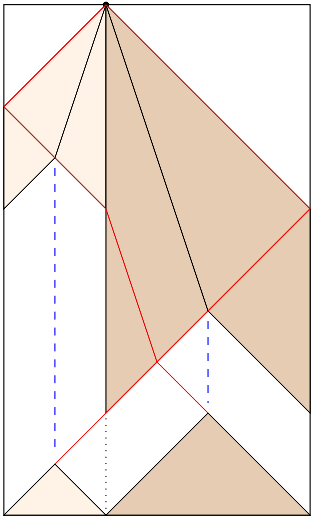


- Synchronisation startet an den Rändern
- blaues Signal friert Teilsynchronisation ein
- rotes Signal taut sie wieder auf

Algorithmus (Moore/Langdon 1968)



Algorithmus (S. Wacker)



Überblick

- Das klassische Firing Squad Synchronization Problem
- General an beliebiger Stelle
- Zweidimensionales FSSP
- Ausblick

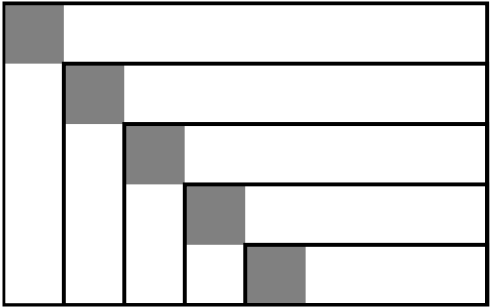
FSSP für Rechtecke

- analog zum Eindimensionalen
  - von Neumann Nachbarschaft (Radius 1)
- Anfangskonfiguration:
  Rechteck von s-Zellen mit
  einer g-Zelle im z.B. linken oberen Eck

g	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s

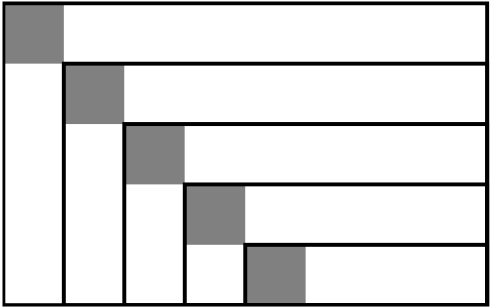
Algorithmus für das FSSP in Rechtecken mit General in der Ecke

Algorithmus für das FSSP in Rechtecken mit General in der Ecke



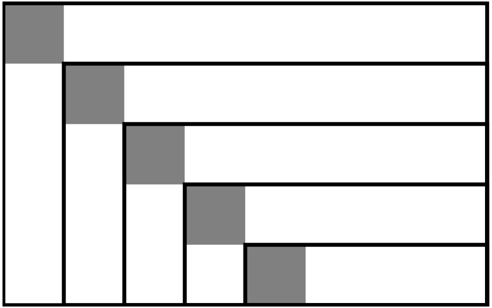
- synchronisiere jedes «L»
  wie eindimensional ℓ Zellen
- äußerstes «L»:
  - $\ell = n + m - 1$
  - $k = \min\{m, n\} - 1$

Algorithmus für das FSSP in Rechtecken mit General in der Ecke



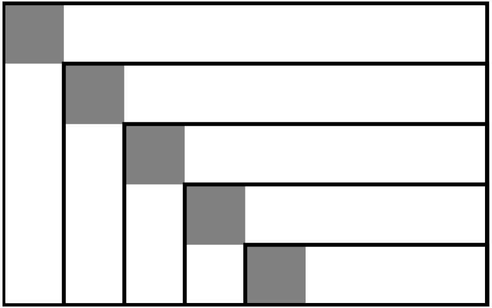
- synchronisiere jedes «L» wie eindimensional ℓ Zellen
- äußerstes «L»:
  - $\ell = n + m - 1$
  - $k = \min\{m, n\} - 1$
  - $2\ell - k - 2$ Schritte

Algorithmus für das FSSP in Rechtecken mit General in der Ecke



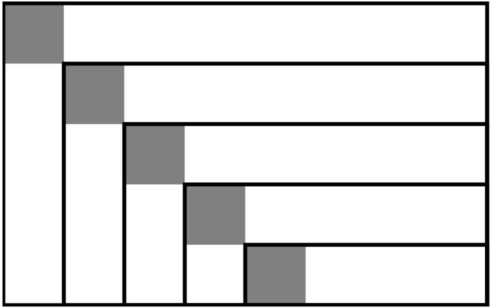
- synchronisiere jedes «L» wie eindimensional ℓ Zellen
- äußerstes «L»:
  - $\ell = n + m - 1$
  - $k = \min\{m, n\} - 1$
  - $2\ell - k - 2 = m + n + \max\{m, n\} - 3$ Schritte

Algorithmus für das FSSP in Rechtecken mit General in der Ecke



- synchronisiere jedes «L» wie eindimensional ℓ Zellen
- äußerstes «L»:
  - $\ell = n + m - 1$
  - $k = \min\{m, n\} - 1$
  - $2\ell - k - 2 = m + n + \max\{m, n\} - 3$ Schritte
- starte alle 3 Schritte Synchronisation des nächstkleineren «L»

Algorithmus für das FSSP in Rechtecken mit General in der Ecke



- synchronisiere jedes «L» wie eindimensional ℓ Zellen
- äußerstes «L»:
  - $\ell = n + m - 1$
  - $k = \min\{m, n\} - 1$
  - $2\ell - k - 2 = m + n + \max\{m, n\} - 3$ Schritte
- starte alle 3 Schritte Synchronisation des nächstkleineren «L»
- insgesamt $m+n+\max\{m, n\}-3$ Schritte
- optimal

Überblick

- Das klassische Firing Squad Synchronization Problem
- General an beliebiger Stelle
- Zweidimensionales FSSP
- Ausblick

Weitere Fragestellungen

- Verallgemeinerungen
  - d-dimensionale Quader mit dem General an beliebiger Stelle (Szwerinski, 1982)
  - mehrere Generäle (Diplomarbeit H. Schmid, 2003)
  - beliebige d-dimensionale zusammenhängende Muster
  - (langsam) wachsende Muster
- Spezialisierungen
  - Quadrate bzw. Würfel
  - kleinere untere Schranken als bei Rechtecken!
    warum?
- Modifikationen
  - Early-Bird-Problem (Rosenstiehl, Katona/Legendi, Vollmar)

Zusammenfassung

- Beim eindimensionalen FSSP ist für jeden der Fälle
  - ein General am linken Ende oder an beliebiger Stelle
  - mehrere Generäle an beliebigen Stellen

  zeitoptimale Lösungen bekannt.

- Im zweidimensionalen Fall ist das nur für die Fälle eines Generals bekannt.