Geosensornetze
WS 2013/2014

Hausarbeit von
Andre Lehnert und Marcell Salvage

4. Februar 2014

Inhaltsverzeichnis

1 Einführung

Mit Hilfe der NetLogo-Simulationsumgebung [3] wird eine dynamische Evakuierung von Gebäuden implementiert. Dazu werden die Grundrisse der Gebäude oder Etage in die Simulationsumgebung geladen. Diese dient als Grundlage für die Platzierung von Personen, Gefahrenevents und Notausgängen.

Auf der Flucht vor Gefahrenevents werden die Personen von mobilen Geräten unterstützt, die zur Warnung anderer Personen und zur Lokalisierung der Notausgänge dienen.

1.1 Aufgabenbeschreibung

Die Aufgabe besteht in der Umsetzung einer geeigneten Simulationsumgebung (siehe Kapitel 2). Auf deren Basis Algorithmen zur Lokalisierung und Bestimmung eines Fluchtweges zu den Notausgängen entwickelt werden (siehe Kapitel 3). Schließlich wird eine Evaluation der der Algorithmen in Punkto Effizienz und Zuverlässigkeit durchgeführt und Reflektiert (siehe Kapitel 4).

Personen werden in der NetLogo-Umgebung als Agenten realisiert, die sich nach dem Bewegungsmodell (siehe Abschnitt 2.2.2) innerhalb des Grundrisses bewegen. Die initiale Platzierung geschieht zufällig, analog zu der Platzierung der Gefahrenevents. Personen besitzen die Fähigkeit diese Gefahrenevents in ihrer Umgebung wahrzunehmen und als Gefahrensituation zu deuten. Der Personen versuchen daraufhin den besten Weg zu einem Notausgang zu finden und benachbarte Personen dabei über ihre mobilen Geräte zu warnen.

Als Gefahrensituationen (siehe Abschnitt 2.4) zählt eine gewisse Anzahl von Giftgasbomben mit eingebautem Zeitzünder, die je ein Gefahrenevent darstellen. Die freigesetzten Gasmengen sind regulierbar und breiten sich innerhalb des freien Raumes aus.

Zur Evakuierung der Personen aus dem Gefahrenevent werden Notausgänge (siehe Abschnitt 2.3) in dem Grundriss platziert, deren Position sich während der Simulation nicht ändert, sogenannte *anchor notes*.

Eine feste Position ist notwendig zur Realisierung der dezentralen Lokalisierungs-

algorithmen, die auf den mobilen Geräten der Personen aktiv sind und bei einer dynamischen Evakuierung assistieren. Durch eine lokal eingeschränkte Kommunikationsfähigkeit (siehe Abschnitt 2.2.3) werden Informationen über die Passierbarkeit der Notausgänge an die mobilen Geräte verteilt. Dies ermöglicht die sichere Evakuierung, falls beispielsweise das Giftgas einen Notausgang erreicht hat oder die Fluchtwege blockiert sind.

2 Simulationsumgebung

Das Kapitel der Simulationsumgebung befasst sich mit den Eingabemöglichkeiten zur Anpassung der Simulationsparameter, sowie der konkreten Umsetzung der simulierten Welt, mit der Logik für die Notausgänge, der Personen, der Gefahrensituationen und dem Kommunikationsmodell.

Die Ausgangsbasis für die verwendeten Modelle und Protokolle stammen aus den erstellten Übungen zur Vorlesung Gensensornetze, sie wurden jedoch auf die Anforderungen der Hausarbeit adaptiert.

2.1 Benutzerschnittstelle

In diesem Abschnitt werden die Interaktions- und Konfigurationsmöglichkeiten mit der Simulationsumgebung erläutert. Abbildung 2.1 stellt einen Ausschnitt des grafischen Interfaces von NetLogo da, anhand dessen die Erklärungen in den folgenden Abschnitten besser einzuordnen sind.

2.1.1 Benutzerevents

Zum Auslösen von Benutzerevents werden Buttons verwendet. Zu diesen zählen `setup`, `reset` und `go`, die hier kurz beschrieben werden.

setup-Button

Die Betätigung des `setup`-Buttons ruft eine Folge von Setup und Initialisierungsschritten auf. Zu Beginn wird die Simulationswelt erstellt. Dazu wird der Parameter `inputFile` (siehe Abschnitt 2.1.2) zum Einbinden einer Bild-Datei und das Mapping der Pixel-Farbwerte auf die passend skalierte NetLogo-Welt gemappt. Das Ergebnis ist der gewählte Grundriss, bei dem jedes Patch einen Farbwert erhalten hat.

Dieser Farbwert ist essentiell für den nachfolgenden Schritt in der `setup-patches`-Methode. Diese legt den Initialzustand jedes Patches fest (siehe Abschnitt 2.5).

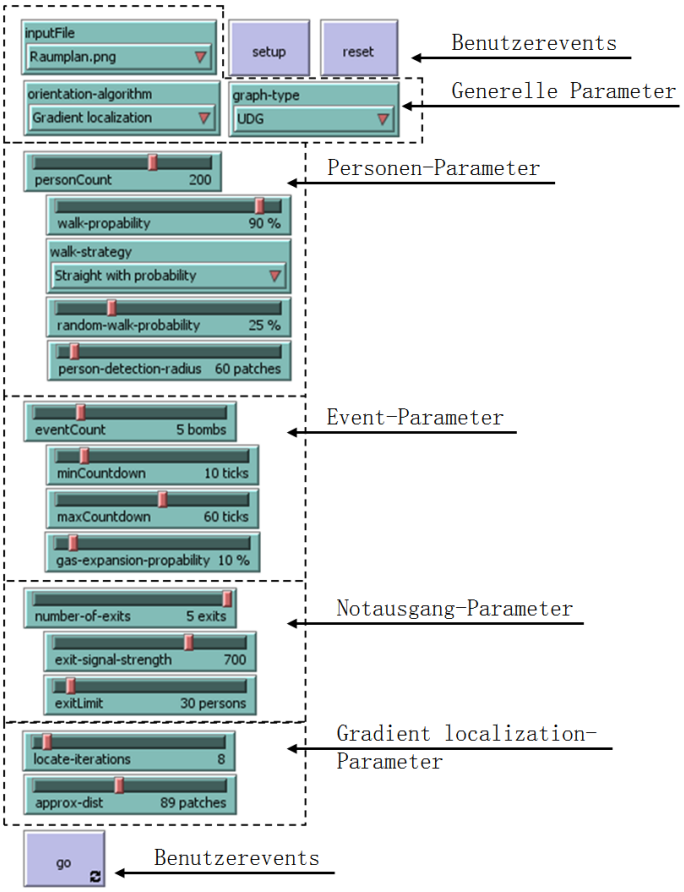

Abbildung 2.1: Grafische Oberfläche

Generell wird auf Grund des Grundrisses zwischen leerem Raum (weiß) und einer Wand unterschieden (schwarz).

Wände spielen auch bei der Initialisierung und Platzierung der Notausgänge eine wichtige Rolle, die mit der `setup-exits`-Methode realisiert sind. Entsprechend der Aufgabenstellung werden Notausgänge auf einem abstrakten Grundriss, ohne Wände, zufällig in der Welt platziert. Für alle anderen Grundrisse mit Wänden wurde auf statische vordefinierte Positionen für Notausgänge gesetzt.

Bei der Initialisierung wird zudem die lokale Konstante der maximalen Signalreichweite für den Orientierungsalgorithmus *cellular automaton* mit dem Parameter `exit-signal-strength` aus Abschnitt 2.1.5 gesetzt und alle Notausgänge in den Zustand *INIT* versetzt (vgl. Abschnitt 2.3).

Im Anschluss wird der *cellular automaton*-Algorithmus (siehe Abschnitt 3.2) ausgeführt, um die Patches mit Signalqualitätsinformationen auszustatten, damit die Personen den optimalen Fluchtweg bei einer Gefahrensituation nutzen.

Nachdem die Simulationswelt mit den statischen Elementen vorbereitet wurde, werden die Personen initialisiert und platziert. Dies geschieht mit der Methode `setup-persons`. Hier wird eine definierte Anzahl von Personen erstellt (siehe Abschnitt 2.1.3, die auf einem abstrakten Grundriss zufällig und auf allen anderen zufällig mit einer Wand-Detektion platziert werden. Die Personen befinden sich nun im Zustand *INIT* (vgl. Abschnitt 2.2.2).

Nach der Platzierung wird einmalig ein Kommunikationsgraph zwischen den Personen und den Notausgängen erstellt, damit der Nutzer ggf. den Graph-Typen oder die Kantenlänge anpassen kann (siehe Abschnitt 2.1.3).

Als letzter Schritt folgt die Initialisierung und Platzierung der Gefahrenevents mittels `setup-events`. Die analog zu den Personen auf dem abstrakten Grundriss zufällig und bei allen anderen Grundrissen mit Wand-Detektion platziert werden. Danach befinden sich alle Gefahrenevents im Zustand *INIT* (vgl. Abschnitt 2.4).

reset-Button

Mit dem `reset`-Button werden alle definierten Parameter des letzten Setups wiederhergestellt und die Personen auf ihre ursprüngliche Position zurückgesetzt. Ein fluten des Grundrisses ist nicht erforderlich und beschleunigt das durchführen von Messreihenreihen. Zudem bietet es die Möglichkeit den Kommunikationsgraphen der Personen auszublenden, dies ist unter anderem nützlich bei der Darstellung der initial approximierten Positionen der Personen.

go-Button

Schließlich kann die Simulation mit dem `go`-Button gestartet und pausiert werden, da hier die *forever*-Option aktiv ist. Ist diese deaktiviert, wird pro Betätigung des `go`-Buttons nur ein *Tick* durchgeführt.

2.1.2 Generelle Parameter

Der Parameter `input-file` erlaubt die Definition des Grundrisses der Simulationsumgebung. Während des Setups wird der Parameter zur Pfadauflösung für eine Bild-Datei verwendet. Diese wird mit dem Befehl `import-pcolors inputFile` auf die Simulationswelt gemappt.

`input-file` $\in \{$*Abstract.png*, *Abstract_static.png*, *Simple.png*, *Raumplan.png*$\}$

Der nächste generelle Parameter ist `orientation-algorithm`. Dieser dient der Auswahl eines Algorithmus zur Orientierung und Lokalisierung der Personen. Detaillierte Informationen sind im Kapitel 3 aufgeführt.

`orientation-algorithm` $\in \{$*Cellular automaton*, *Gradient localization*$\}$

Mit dem `graph-type`-Parameter hat der Nutzer die Wahl zwischen den Graphtypen, die in der Vorlesung vorgestellt wurden. Sofern *UDG* gewählt ist, wird der `person-detection-radius`-Parameter des folgenden Abschnitts für den Disk-Radius verwendet.

`graph-type` $\in \{Complete\ Graph, UDG, RNG, GG\}$

2.1.3 Personen–Parameter

`person-count` definiert die Anzahl der Personen in der Simulationsumgebung.

`person-count` $\in [1, 300]$

Mit dem `walk-probability`-Parameter wird die Wahrscheinlichkeit definiert, mit der Personen bei einem Tick einen Schritt machen. Bei 0% werden die Personen statisch an der gegenwärtigen Position fixiert. Es ist also möglich diesen Parameter während der Laufzeit zu verändern.

`walk-probability` $\in [0, 100]$

Der `walk-strategy`-Parameter erlaubt die Auswahl der *random walk*-Strategie (siehe Abschnitt 2.2.2).

`walk-strategy` $\in \{$*Complete random*, *Straight with collision detection*, *Straight with probability*$\}$

Analog zur `walk-probability` kann der Nutzer den `random-walk-probability`-Parameter zur Laufzeit anpassen und somit die Wahrscheinlichkeit der *random walk* Richtungsänderung bestimmen. Bei 100% wird jede Person nach jedem Tick eine Richtungsänderung vornehmen.

`random-walk-probability` $\in [0, 100]$

Der `person-detection-radius`-Parameter ist einer der entscheidendsten bei der Simulation. Hiermit wird der Radius definiert in dem eine Person eine Gefahrensituation wahrnehmen kann, sowie die Kommunikationsreichweite bei dem UDG-Graphen

bestimmt. Zudem wird damit der Abstand zu einem Notausgang bestimmt (siehe Kapitel 3.

`person-detection-radius` $\in [0, 700]$

2.1.4 Event–Parameter

Mit dem Parameter `event-count` wird die Anzahl der zu platzierenden Gefahrenevents festgelegt. Bei `event-count`$= 0$ wird es zu keiner Gefahrensituation kommen, sodass der random-walk getestet werden kann.

`event-count` $\in [0, 20]$

In der Aufgabenstellung wird ein zufälliges Auslösen von Gefahrensituationen gefordert, die beiden Parameter `min-countdown` und `max-countdown` bewerkstelligen dies. Jedes einzelne Gefahrenevent erhält zufällig einen initialen Countdown im Intervall [`min-countdown`, `max-countdown`]. Die obere bzw. untere Intervallgrenze der Parameter wird durch den jeweils anderen Parameter eingeschränkt.

`min-countdown` $\in [1, \textit{max-countdown}]$

`max-countdown` $\in [\textit{min-countdown}, 100]$

Der Parameter `gas-expansion-probability` legt für alle Gefahrenevents die Ausbreitungswahrscheinlichkeit und somit die Ausbreitungsgeschwindigkeit fest. Bei 0% wird nur genau ein Patch unter dem jeweiligen Gefahrenevent zu einer Bedrohung für die Personen. Personen können die Gefahrensituationen somit wahrnehmen, die Wahrscheinlichkeit das Personen sterben ist jedoch sehr gering.

`gas-expansion-probability` $\in [0, 100]$

2.1.5 Notausgang–Parameter

Der Parameter zur Einstellung der verfügbaren Notausgänge in der simulierten Welt, `number-of-exits` ist sehr bedeutsam bei der Lokalisierungsgenauigkeit und dem Fluchtverhalten der Personen.

`number-of-exits` $\in [1, 9]$

Zur dezentralen Orientierung der Personen und Schaffung eines optimalen Fluchtweges, wird auf den Zellulären Automaten zurückgegriffen und die Patches mit Informationen zur Signalstärke jedes Notausganges versehen.

Der Parameter `exit-signal-strength` bildet die maximale Signalstärke bzw. Reichweite der Notausgänge ab. Es ist möglich, dass nicht jedes Patch mit allen Signalinformationen versehen ist, da die Reichweite eines Notausganges zu gering war.

`exit-signal-strength` $\in [0, 1000]$

Die Aufgabenstellung fordert eine Limitierung Fluchtkapazitäten von Notausgängen. D.h. Notausgänge können maximal `exit-limit` Personen pro Tick evakuieren, sonst werden sie blockiert.

`exit-limit` $\in [1, 300]$

2.1.6 Gradient localization–Parameter

- -
TODO: Marcell Beschreibung der Parameter, Bedeutung
- -

`locate-iterations`

`locate-iterations` $\in [0, 100]$

`approx-dist`

`approx-dist` $\in [0, 200]$

2.2 Personen

2.2.1 Zustände

2.2.2 Bewegungsmodell

Dieser Abschnitt beschreibt zunächst das Bewegungsmodell der Personen ohne aktive Gefahrensituation.

2.2.3 Kommunikationsmodell

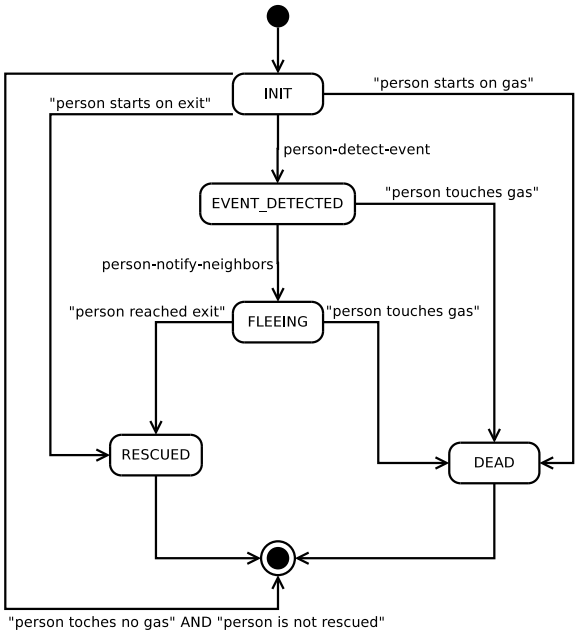

Abbildung 2.2: Zustandsdiagramm der Personen

Algorithmus 1 random-walk

```
nb ← one-of neighbors
while [patch-state] of nb = WALL do
  nb ← one-of neighbors
end while
face nb
forward 1
```

Protokoll 2 Warnung vor Gefahrensituationen

State Trans. Sys.: ⟨{INIT, EVENT_DETECTED, FLEEING, RESCUED, DEAD}⟩
Initialization: All notes in state INIT
Restrictions: Reliable communication; connected, bidirected communication graph $G = (V, E)$, neighborhoodfunction nbr: $V \to 2^V$
Local data:

INIT
Receiving(*event_detected*)
while *not event_detected* **do**
  random-walk
  create-graph(G) *; generate complete new graph*
end while
become EVENT_DETECTED

EVENT_DETECTED
broadcast(*event_detected*) *; broadcast event detection to linked neighbors*
become FLEEING

FLEEING
broadcast(*event_detected*) *; rebroadcast event detection to linked neighbors*
while *not person-reach-exit* **do**
  person-move-to-exit *; using orientation-algorithm*
  create-graph(G) *; generate complete new graph*
  if touching-gas **then**
    become DEAD
  end if
end while
become RESCUED

RESCUED
create-graph(G) *; generate complete new graph without note*

DEAD
create-graph(G) *; generate complete new graph without note*

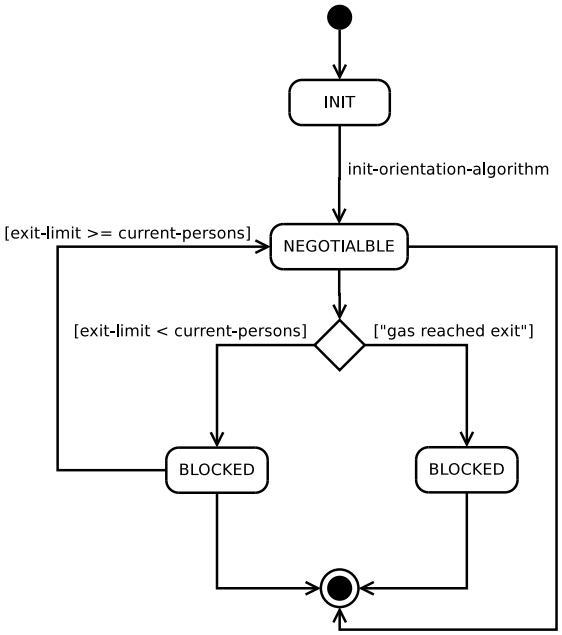

Abbildung 2.3: Zustandsdiagramm der Notausgänge

2.3 Notausgänge

2.3.1 Zustände

2.4 Gefahrensituationen

2.4.1 Zustände

2.5 Patches

- Initialisierung der Patches setup-patches - Zustandsfestlegung für alle Patches - white -¿ NONE - black -¿ WALL - rest -¿ WALL - Initialisierung der lokalen Daten `signal-noise` abhängig vom Zustand: leerer Raum -¿ -1, Wand -¿ sehr hohe Dämpfung 2.1.5

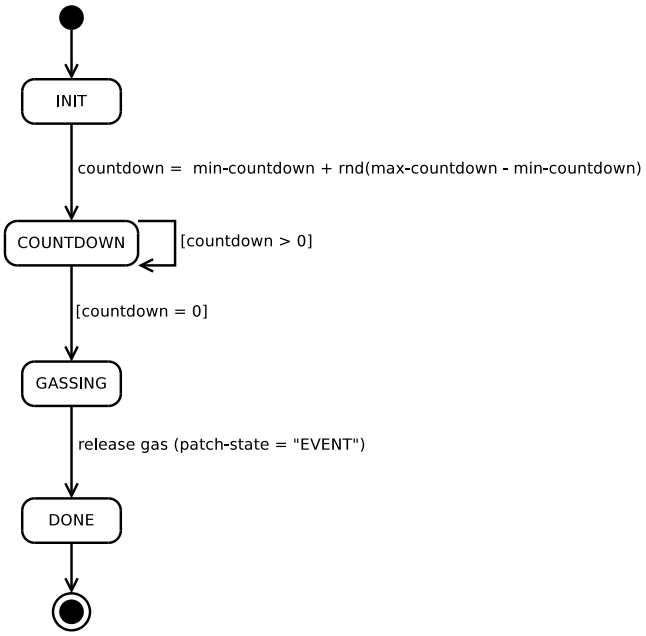

Abbildung 2.4: Zustandsdiagramm der Gefahrenevents

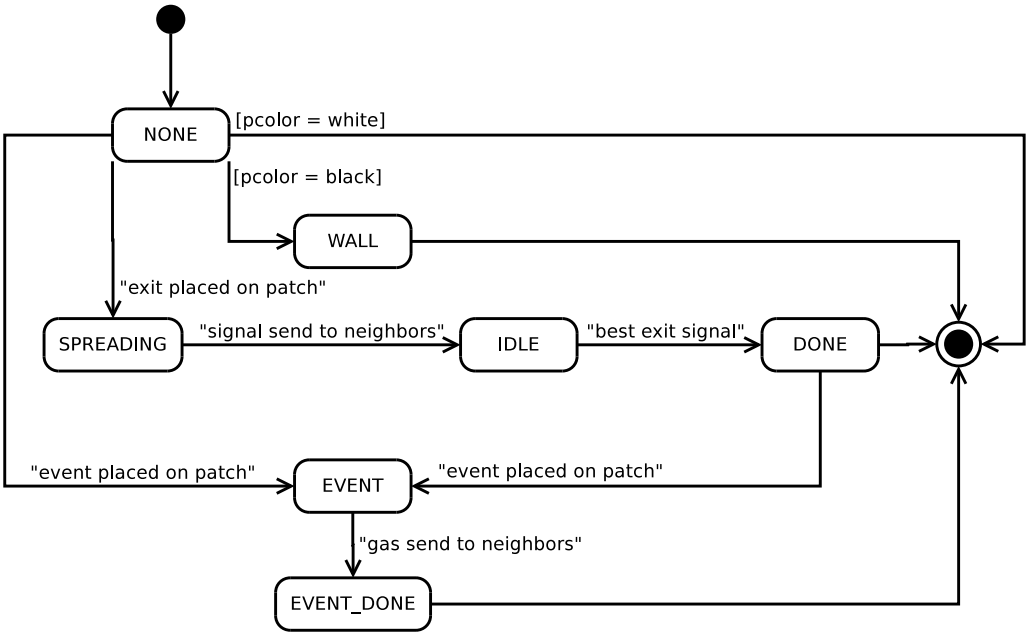

Abbildung 2.5: Zustandsdiagramm der Patches

Protokoll 3 Gefahrensituation

State Trans. Sys.: ⟨{INIT, COUNTDOWN, GASSING, DONE}⟩
Initialization: All notes in state INIT
Restrictions: All patches in state {NONE, DONE}
Local data: countdown ∈ $\mathbb{N}_{\geq 0}$

INIT
Spontaneously
countdown ← minCountdown + (random (maxCountdown - minCountdown))
become COUNTDOWN

COUNTDOWN
countdown ← countdown - 1
if *countdown* = 0 **then**
  become GASSING
end if

GASSING
ask patch-here [patch-state ← EVENT]
become DONE

DONE

2.5.1 Implizite Zustände

2.6 Ressourcen der Simulationsumgebung

Die Simulationsumgebung wird über die Datei `Evakuierung.nlogo` gestartet. Der Programmcode ist nach Funktion und *Breed*-Klasse unterteilt.

Gefahrensituationen

`event.nls`
`event-gassing.nls`

Die Quellcode-Datei `event.nls` beinhaltet den Lebenszyklus der Gefahrenevents und deren Zustandsübergangsprotokoll. Mittels `event-gassing.nls` wird die Ausbreitung des Giftgases implementiert, die größtenteils den Patch-Lebenszyklus manipuliert.

Lokalisierung

`locate.nls`

In dieser Datei wird der Algorithmus zur Lokalisierung aus dem Paper [2] implementiert.

Notausgänge

`exit.nls`
`exit-cellular-automaton.nls`
`exit-gsn.nls`

Die Quellcode-Datei `event.nls` steuert den Setup und den Lebenszyklus der Notausgänge. Mittels `exit-cellular-automaton.nls` wird der Orientierungsalgorithmus auf Basis des zellulären Automats realisiert. Letztlich definiert die Datei `exit-gsn.nls` das Kommunikationsmodell zur Übermittlung der Statusinformationen.

Personen

`person.nls`
`person-gsn.nls`
`person-linking.nls`

`person.nls` regelt den Setup der Personen und deren Lebenszyklus. `person-gsn.nls` umfasst den Quellcode für die Kommunikation zwischen Personen und mit der Datei `person-linking.nls` wird der Graph zwischen Personen erstellt.

Simulationswelt

`patch.nls`

Hier wird der Quellcode für den Lebenszyklus der *Patches* definiert.

3 Algorithmik

3.1 Gradienten Lokalisierung

3.2 Zellulärer Automat

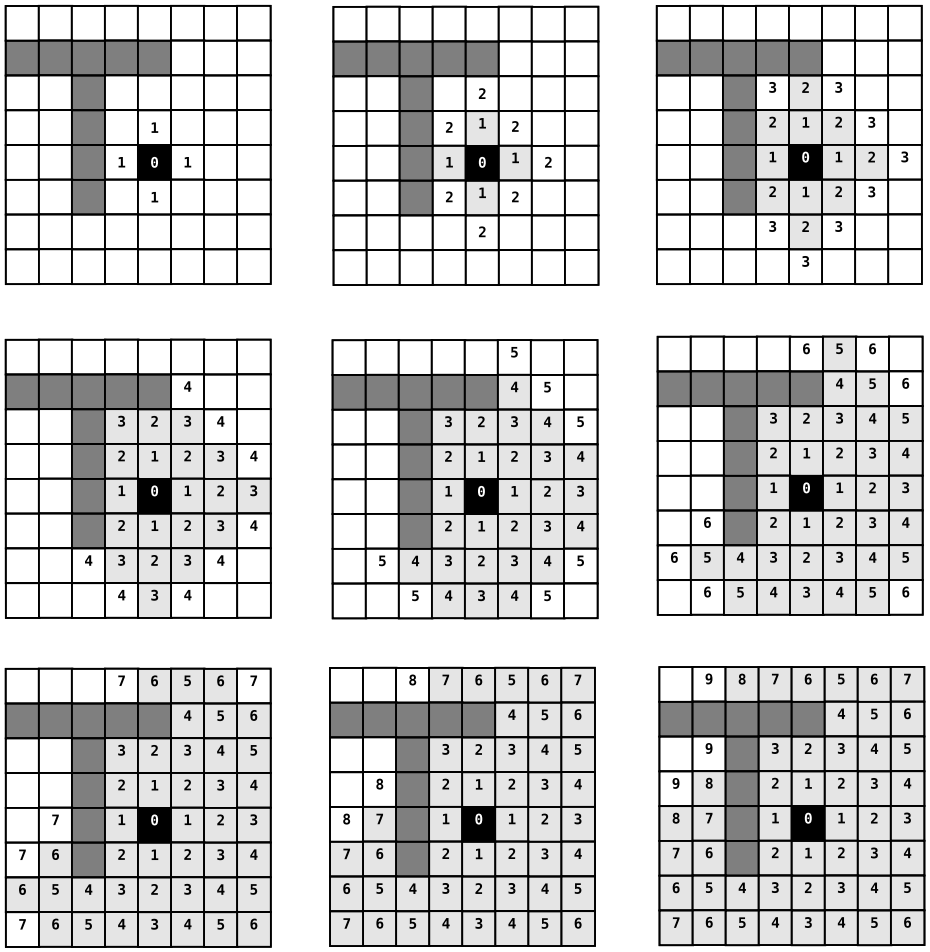

Abbildung 3.1: Fluten der Patches (Zellulärer Automat)

4 Evaluation

4.1 Effizienz

4.2 Fazit

Literaturverzeichnis

[1] Isaac Amundson and Xenofon D. Koutsoukos. *A Survey on Localization for Mobile Wireless Sensor Networks.* Department of Electrical Engineering and Computer Science, Vanderbilt University.

[2] Jonathan Bachrach, Radhika Nagpal, Michael Salib and Howard Shrobe. *Experimental Results for and Theoretical Analysis of a Self-Organizing Global Coordinate System for Ad Hoc Sensor Networks.* Telecommunication Systems, page 213–233. 2004.

[3] Uri Wilensky. *Netlogo.* Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. 1999. `http://ccl.northwestern.edu/netlogo/`, Stand: 26.01.2014.