

Abschlussvortrag zur Diplomarbeit
„Repräsentation und Simulation von Umgebungen
für Multiagentensysteme“

Stefan Tittel

Technische Universität Dortmund

25. November 2010

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

Definition: Agent (nach Wooldridge)

An **agent** is a computer system that is **situated** in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its design objectives.

Definition: Multiagentensystem (nach Krupansky)

A **multi-agent system (MAS)** is a computational system where **agents** cooperate or compete with others to achieve some individual or collective task.

Definition: Umgebung

Eine **Umgebung** stellt Agenten eine **virtuelle Welt** bereit. Der Zustand dieser Welt wird Agenten (ggf. nur ausschnittsweise) durch Nachrichten übermittelt. Agenten fordern Aktionen in der Welt durch den Versand von Nachrichten an die Umgebung an.

Definition: Agent (nach Wooldridge)

An **agent** is a computer system that is **situated** in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its design objectives.

Definition: Multiagentensystem (nach Krupansky)

A **multi-agent system (MAS)** is a computational system where **agents** cooperate or compete with others to achieve some individual or collective task.

Definition: Umgebung

Eine **Umgebung** stellt Agenten eine **virtuelle Welt** bereit. Der Zustand dieser Welt wird Agenten (ggf. nur ausschnittsweise) durch Nachrichten übermittelt. Agenten fordern Aktionen in der Welt durch den Versand von Nachrichten an die Umgebung an.

Definition: Agent (nach Wooldridge)

An **agent** is a computer system that is **situated** in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its design objectives.

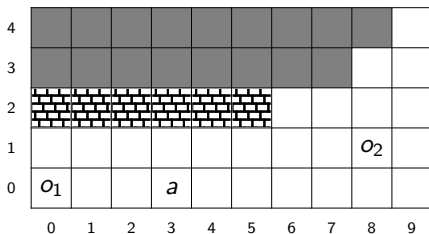
Definition: Multiagentensystem (nach Krupansky)

A **multi-agent system (MAS)** is a computational system where **agents** cooperate or compete with others to achieve some individual or collective task.

Definition: Umgebung

Eine **Umgebung** stellt Agenten eine **virtuelle Welt** bereit. Der Zustand dieser Welt wird Agenten (ggf. nur ausschnittsweise) durch Nachrichten übermittelt. Agenten fordern Aktionen in der Welt durch den Versand von Nachrichten an die Umgebung an.

Beispiel für eine Umgebung 1/2



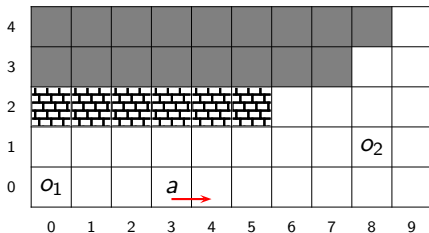
Nachricht: Umgebung → Agent *a*

- *a* in (3,0), *o*₁ in (0,0), *o*₂ in (8,1)
- Mauer in (0,2) ... (5,2)
- sichtbare Zellen: (0,0) ... (9,0), (0,1) ... (9,1), (0,2) ... (9,2), (8,3), (9,3), (9,4)

Nachricht: Agent *a* → Umgebung

Ich möchte nach Osten gehen.

Beispiel für eine Umgebung 1/2



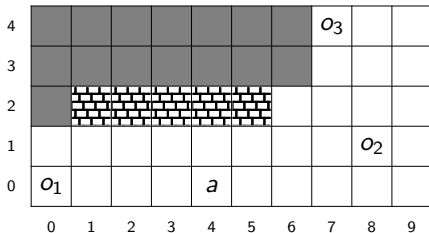
Nachricht: Umgebung \rightarrow Agent a

- a in $(3, 0)$, o_1 in $(0, 0)$, o_2 in $(8, 1)$
- Mauer in $(0, 2) \dots (5, 2)$
- sichtbare Zellen: $(0, 0) \dots (9, 0)$, $(0, 1) \dots (9, 1)$,
 $(0, 2) \dots (9, 2)$, $(8, 3)$, $(9, 3)$, $(9, 4)$

Nachricht: Agent $a \rightarrow$ Umgebung

Ich möchte nach Osten gehen.

Beispiel für eine Umgebung 2/2



Nachricht: Umgebung \rightarrow Agent a

- sichtbare Zellen: $(0,0) \dots (9,0)$, $(0,1) \dots (9,1)$, $(1,2) \dots (9,2)$, $(7,3) \dots (9,3)$, $(7,4) \dots (9,4)$
- a in $(4,0)$, o_1 in $(0,0)$, o_2 in $(8,1)$, o_3 in $(7,4)$
- Mauer in $(1,2) \dots (5,2)$

Ziel der Diplomarbeit: Konzeption und Implementierung eines flexiblen Frameworks zur Realisierung von Umgebungen für Multiagentensysteme

Vorgehen:

- 1 Eigenschaften und Funktionen zu unterstützender Umgebungen festlegen
- 2 Konzept zur Realisierung dieser Eigenschaften und Funktionen entwickeln
- 3 Implementierung durchführen

Ziel der Diplomarbeit: Konzeption und Implementierung eines flexiblen Frameworks zur Realisierung von Umgebungen für Multiagentensysteme

Vorgehen:

- 1 Eigenschaften und Funktionen zu unterstützender Umgebungen festlegen
- 2 Konzept zur Realisierung dieser Eigenschaften und Funktionen entwickeln
- 3 Implementierung durchführen

Ziel der Diplomarbeit: Konzeption und Implementierung eines flexiblen Frameworks zur Realisierung von Umgebungen für Multiagentensysteme

Vorgehen:

- 1 Eigenschaften und Funktionen zu unterstützender Umgebungen festlegen
- 2 Konzept zur Realisierung dieser Eigenschaften und Funktionen entwickeln
- 3 Implementierung durchführen

Ziel der Diplomarbeit: Konzeption und Implementierung eines flexiblen Frameworks zur Realisierung von Umgebungen für Multiagentensysteme

Vorgehen:

- 1 Eigenschaften und Funktionen zu unterstützender Umgebungen festlegen
- 2 Konzept zur Realisierung dieser Eigenschaften und Funktionen entwickeln
- 3 Implementierung durchführen

- 1 Einführung
- 2 Konzeptioneller Überblick**
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

- vollständig wahrnehmbar oder partiell wahrnehmbar
- deterministisch oder stochastisch
- episodisch oder sequentiell
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder stochastisch
- episodisch oder sequentiell
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder sequentiell
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder **sequentiell**
- statisch oder dynamisch
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- diskret oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- **diskret** oder kontinuierlich
- reaktiv oder proaktiv
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- **diskret** oder kontinuierlich
- reaktiv oder **proaktiv**
- ortsbezogen oder nicht ortsbezogen

- vollständig wahrnehmbar oder **partiell wahrnehmbar**
- deterministisch oder **stochastisch**
- episodisch oder **sequentiell**
- statisch oder **dynamisch**
- **diskret** oder kontinuierlich
- reaktiv oder **proaktiv**
- **ortsbezogen** oder nicht ortsbezogen

Festlegung der Eigenschaften und Funktionen 2/2

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

Festlegung der Eigenschaften und Funktionen 2/2

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

Festlegung der Eigenschaften und Funktionen 2/2

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

Festlegung der Eigenschaften und Funktionen 2/2

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

- zweidimensionales Grid als zentrale räumliche Komponente
- Einschränkung der Bewegungs- und Wahrnehmungsmöglichkeiten
 - abhängig von Hindernissen
 - abhängig von der Sichtweite des Agenten
- Nachrichtenvermittlung zwischen Agenten mit einschränkbarer Empfangbarkeit von Nachrichten
 - abhängig von Interferenzen
 - abhängig von Empfangsempfindlichkeit des Empfängers
 - abhängig von Sendestärke des Absenders
 - abhängig von Sichtverbindung zwischen Absender und Empfänger
- Interaktion mit Objekten
 - Aufnahme, Ablage, Verschiebung, Übergabe, Schachtelung
 - Interaktion mit komplexen Objekten vom Typ „Schließfach“
- Beschränkungen
 - Platz in Zellen beschränkbar
 - Platz im Inventar von Agenten und Objekten beschränkbar
- Zeitfortschritt und Ausführungsreihenfolge

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Es gibt folgende Arten von Hindernissen:

- **Mauer**: kein Agent oder Objekt darf sich in der Zelle befinden, sichtbeschränkend
- **Vorhang**: sichtbeschränkend
- **Nebel**: sichtbeschränkend (entsteht und verschwindet proaktiv)
- **Graben**: kein Agent oder Objekt darf sich in der Zelle befinden
- **Interferenz**: kommunikationsblockierend

Nur Interferenz kann mit anderen Hindernissen kombiniert werden.

Kapazitäten und Kapazitätsbedarfe

- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.

Kapazitäten und Kapazitätsbedarfe

- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.

Kapazitäten und Kapazitätsbedarfe

- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.

Kapazitäten und Kapazitätsbedarfe

- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.

Kapazitäten und Kapazitätsbedarfe

- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

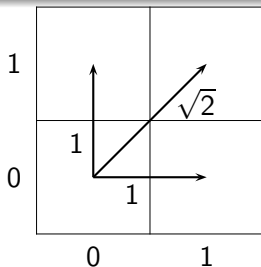
- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.

Kapazitäten und Kapazitätsbedarfe

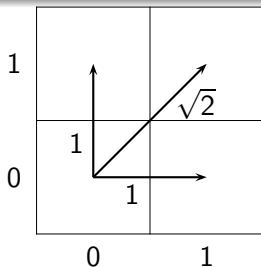
- Abstraktion von Begriffen wie Inventargröße oder Traglast durch den Begriff **Kapazität**
- Abstraktion von Begriffen wie Volumen oder Masse durch den Begriff **Kapazitätsbedarf**
- bei Zellen: Agent oder Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Agenten oder Objekts
- bei Agenten und Objekten: Objekt aufnehmbar, wenn freie Kapazität größer als Kapazitätsbedarf des Objekts
- freie Kapazität und ggf. Kapazitätsbedarf werden beim Aufnehmen, Ablegen oder Betreten aktualisiert

Beispiel

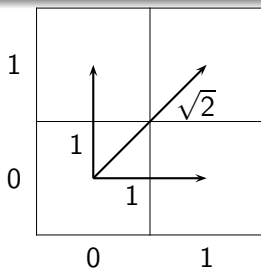
- freie Kapazität von Zelle c : 5,
- Kapazitätsbedarf von Agent a : 7
- Agent a kann Zelle c nicht betreten.



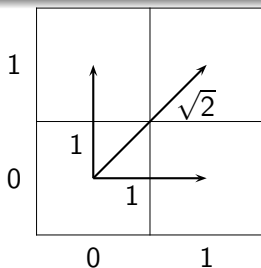
- bei Betrachtung des Grids als realweltliche Fläche:
 - horizontale oder vertikale Bewegung entspricht Schrittlänge 1
 - diagonale Bewegung entspricht Schrittlänge $\sqrt{2}$
- realweltliche Intuition: gleiche Schrittweite pro Zeitpunkt unabhängig von der Richtung
- Lösung: Aktionsanforderungen in diagonaler Richtung nur erfolgreich mit Wahrscheinlichkeit $\frac{1}{\sqrt{2}}$ (Diagonalkorrektur)
- damit Schrittweite diagonaler Aktionsanforderungen im Erwartungswert wie bei horizontalen und vertikalen Aktionsanforderungen



- bei Betrachtung des Grids als realweltliche Fläche:
 - horizontale oder vertikale Bewegung entspricht Schrittlänge 1
 - diagonale Bewegung entspricht Schrittlänge $\sqrt{2}$
- realweltliche Intuition: gleiche Schrittweite pro Zeitpunkt unabhängig von der Richtung
- Lösung: Aktionsanforderungen in diagonaler Richtung nur erfolgreich mit Wahrscheinlichkeit $\frac{1}{\sqrt{2}}$ (Diagonalkorrektur)
- damit Schrittweite diagonaler Aktionsanforderungen im Erwartungswert wie bei horizontalen und vertikalen Aktionsanforderungen



- bei Betrachtung des Grids als realweltliche Fläche:
 - horizontale oder vertikale Bewegung entspricht Schrittweite 1
 - diagonale Bewegung entspricht Schrittweite $\sqrt{2}$
- realweltliche Intuition: gleiche Schrittweite pro Zeitpunkt unabhängig von der Richtung
- Lösung: Aktionsanforderungen in diagonaler Richtung nur erfolgreich mit Wahrscheinlichkeit $\frac{1}{\sqrt{2}}$ (**Diagonalkorrektur**)
- damit Schrittweite diagonaler Aktionsanforderungen im Erwartungswert wie bei horizontalen und vertikalen Aktionsanforderungen



- bei Betrachtung des Grids als realweltliche Fläche:
 - horizontale oder vertikale Bewegung entspricht Schrittweite 1
 - diagonale Bewegung entspricht Schrittweite $\sqrt{2}$
- realweltliche Intuition: gleiche Schrittweite pro Zeitpunkt unabhängig von der Richtung
- Lösung: Aktionsanforderungen in diagonaler Richtung nur erfolgreich mit Wahrscheinlichkeit $\frac{1}{\sqrt{2}}$ (**Diagonalkorrektur**)
- damit Schrittweite diagonaler Aktionsanforderungen im Erwartungswert wie bei horizontalen und vertikalen Aktionsanforderungen

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung**
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch Zustandsüberführungsregeln
- Auslösung von Zustandsüberführungsregeln durch Einflüsse:
 - Aktionsanforderungen: Einflüsse, die Agenten auf die Umgebung ausüben
 - interne Einflüsse: von der Umgebung durch Einflusserzeugungsregeln selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - Auslöser
 - Vorbedingungen
 - Ausführungswahrscheinlichkeit
 - Nachbedingungen
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung interner Zwischenzustände
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch totale Ordnungen

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch **Zustandsüberführungsregeln**
- Auslösung von Zustandsüberführungsregeln durch **Einflüsse**:
 - Aktionsanforderungen: Einflüsse, die Agenten auf die Umgebung ausüben
 - interne Einflüsse: von der Umgebung durch Einflusserzeugungsregeln selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - Auslöser
 - Vorbedingungen
 - Ausführungswahrscheinlichkeit
 - Nachbedingungen
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung **interner Zwischenzustände**
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch **totale Ordnungen**

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch **Zustandsüberführungsregeln**
- Auslösung von Zustandsüberführungsregeln durch **Einflüsse**:
 - **Aktionsanforderungen**: Einflüsse, die Agenten auf die Umgebung ausüben
 - **interne Einflüsse**: von der Umgebung durch **Einflusserzeugungsregeln** selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - Auslöser
 - Vorbedingungen
 - Ausführungswahrscheinlichkeit
 - Nachbedingungen
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung **interner Zwischenzustände**
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch **totale Ordnungen**

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch **Zustandsüberführungsregeln**
- Auslösung von Zustandsüberführungsregeln durch **Einflüsse**:
 - Aktionsanforderungen: Einflüsse, die Agenten auf die Umgebung ausüben
 - interne Einflüsse: von der Umgebung durch Einflusserzeugungsregeln selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - **Auslöser**
 - **Vorbedingungen**
 - **Ausführungswahrscheinlichkeit**
 - **Nachbedingungen**
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung **interner Zwischenzustände**
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch **totale Ordnungen**

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch **Zustandsüberführungsregeln**
- Auslösung von Zustandsüberführungsregeln durch **Einflüsse**:
 - Aktionsanforderungen: Einflüsse, die Agenten auf die Umgebung ausüben
 - interne Einflüsse: von der Umgebung durch Einflusserzeugungsregeln selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - Auslöser
 - Vorbedingungen
 - Ausführungswahrscheinlichkeit
 - Nachbedingungen
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung **interner Zwischenzustände**
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch **totale Ordnungen**

Überblick: Vorgehen bei der Zustandsüberführung

- Bedingungen für Zeitfortschritt
- Überführung vom Zustand S_t nach S_{t+1} durch **Zustandsüberführungsregeln**
- Auslösung von Zustandsüberführungsregeln durch **Einflüsse**:
 - Aktionsanforderungen: Einflüsse, die Agenten auf die Umgebung ausüben
 - interne Einflüsse: von der Umgebung durch Einflusserzeugungsregeln selbst erzeugte Einflüsse zur Realisierung proaktiven Verhaltens
- Zustandsüberführungsregeln bestehen aus:
 - Auslöser
 - Vorbedingungen
 - Ausführungswahrscheinlichkeit
 - Nachbedingungen
- Behandlung mehrerer Einflüsse zu einem Zeitpunkt t durch Erzeugung **interner Zwischenzustände**
- Festlegung der Anwendungsreihenfolge von Einflüssen und Zustandsüberführungsregeln durch **totale Ordnungen**

- jede Aktionsanforderung löst Zeitfortschritt aus
 - einfach
 - keine konfligierenden Aktionsanforderungen
 - nicht geeignet für Wettbewerbssituationen
 - Agent kann Zeitfortschritt beeinflussen
 - Agenten kennen Realzeit bis zum Zeitfortschritt nicht
- Zeitfortschritt, wenn Aktionsanforderung von jedem Agenten vorliegt
 - Agent kann Zeitfortschritt blockieren (daher: zusätzlich Timeout)
 - konfligierende Aktionsanforderungen möglich
 - Agent kann Zeitfortschritt beeinflussen
 - Realzeit bis zum Zeitfortschritt nicht einheitlich
- fest getakteter Zeitfortschritt
 - konfligierende Aktionsanforderungen möglich
 - Zeitfortschritt unabhängig von Agenten
 - Agenten kennen Realzeit bis zum Zeitfortschritt
 - Auftreten unnötiger Wartezeiten

- jede Aktionsanforderung löst Zeitfortschritt aus
 - einfach
 - keine konfligierenden Aktionsanforderungen
 - nicht geeignet für Wettbewerbssituationen
 - Agent kann Zeitfortschritt beeinflussen
 - Agenten kennen Realzeit bis zum Zeitfortschritt nicht
- Zeitfortschritt, wenn Aktionsanforderung von jedem Agenten vorliegt
 - Agent kann Zeitfortschritt blockieren (daher: zusätzlich Timeout)
 - konfligierende Aktionsanforderungen möglich
 - Agent kann Zeitfortschritt beeinflussen
 - Realzeit bis zum Zeitfortschritt nicht einheitlich
- fest getakteter Zeitfortschritt
 - konfligierende Aktionsanforderungen möglich
 - Zeitfortschritt unabhängig von Agenten
 - Agenten kennen Realzeit bis zum Zeitfortschritt
 - Auftreten unnötiger Wartezeiten

- jede Aktionsanforderung löst Zeitfortschritt aus
 - einfach
 - keine konfligierenden Aktionsanforderungen
 - nicht geeignet für Wettbewerbssituationen
 - Agent kann Zeitfortschritt beeinflussen
 - Agenten kennen Realzeit bis zum Zeitfortschritt nicht
- Zeitfortschritt, wenn Aktionsanforderung von jedem Agenten vorliegt
 - Agent kann Zeitfortschritt blockieren (daher: zusätzlich Timeout)
 - konfligierende Aktionsanforderungen möglich
 - Agent kann Zeitfortschritt beeinflussen
 - Realzeit bis zum Zeitfortschritt nicht einheitlich
- fest getakteter Zeitfortschritt
 - konfligierende Aktionsanforderungen möglich
 - Zeitfortschritt unabhängig von Agenten
 - Agenten kennen Realzeit bis zum Zeitfortschritt
 - Auftreten unnötiger Wartezeiten

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

Definition

Ein Umgebungszustand S_t ist eine Menge von Fakten, die den Zustand der Umgebung im Zeitpunkt t repräsentieren.

Grundlegende Typen von Fakten:

- **Grid**(x, y): Koordinate (x, y) ist gültige Gridzelle
- **Agent**(a): a ist Agent
- **Object**(o): o ist Objekt
- **Contains**(z, o): Agent bzw. Objekt z beinhaltet Objekt o
- **Loc**(x, y, z): Agent oder Objekt z befindet sich in Gridzelle (x, y)
- **CapNeed**(z, c): Agent oder Objekt z besitzt Kapazitätsbedarf c

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

- **FreeCellCap**(x, y, c): Zelle (x, y) besitzt freie Kapazität c
- **FreeCap**(z, c): Agent oder Objekt z besitzt freie Kapazität c
- **MoveForce**(a, f): Agent a verfügt über **Verschiebekraft** f
- **Wall**(x, y): Gridzelle (x, y) besitzt Mauer
- **Trench**(x, y): Gridzelle (x, y) besitzt Graben
- **Curtain**(x, y): Gridzelle (x, y) besitzt Vorhang
- **Interference**(x, y): Gridzelle (x, y) besitzt Interferenz
- **AcceptReceive**(a_1, a_2, o): Agent a_1 akzeptiert Übergabe von Objekt o durch Agent a_2
- **benutzereigene Fakten** ohne Relevanz für die Semantik der Umgebung

Aufbau von Umgebungszuständen 3/3

1		<div style="border: 1px solid gray; padding: 5px; display: inline-block;">Objekt o_1 $c = 5; n = 3$</div>	<div style="border: 1px solid gray; padding: 5px; display: inline-block;">Objekt o_2 $c = 0; n = 1$</div>
	frei = 10	frei = 7	frei = 9
0			<div style="border: 2px solid gray; padding: 5px; display: inline-block;"><div style="border: 1px solid gray; padding: 5px; display: inline-block;">Agent a_1 $c = 1; n = 8; f = 5$</div><div style="border: 1px solid gray; padding: 5px; display: inline-block; margin-top: 5px;">Objekt o_3 $c = 2; n = 2$</div></div>
	frei = 10	frei = 10	frei = 2
	0	1	2

c : freie Kapazität, n : Kapazitätsbedarf, f : Verschiebekraft

Grundlegende Aktionsanforderungen 1/2

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

Grundlegende Aktionsanforderungen 1/2

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

Grundlegende Aktionsanforderungen 1/2

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

- **Move(a, d)**: Agent a möchte sich in Richtung d bewegen
- **Take(a, o)**: Agent a möchte Objekt o aus Zelle in Inventar aufnehmen
- **Release(a, o)**: Agent a möchte Objekt o aus Inventar in Zelle ablegen
- **MoveObject(a, o, d)**: Agent a möchte Objekt o in Richtung d bewegen
- **HandOver(a, a_1, o)**: Agent a möchte Objekt o aus Inventar an anderen Agenten a_1 übergeben
- **DeclareAccept(a, a_1, o)**: Agent a erklärt Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **RetractAccept(a, a_1, o)**: Agent a widerruft Bereitschaft, Objekt o von Agent a_1 zu empfangen
- **Load(a, o_1, o_2)**: Agent a möchte Objekt o_1 in Laderaum von Objekt o_2 legen

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

Move(tweety, e)

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkersch1“ aus der Zelle in sein Inventar aufnehmen

Take(tweety, schwarzwälderkersch1)

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

Load(tweety, apfel1, kiste1)

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

```
Move(tweety, e)
```

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkirsch1“ aus der Zelle in sein Inventar aufnehmen

```
Take(tweety, schwarzwälderkirsch1)
```

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

```
Load(tweety, apfel1, kiste1)
```

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

```
Move(tweety, e)
```

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkirsch1“ aus der Zelle in sein Inventar aufnehmen

```
Take(tweety, schwarzwälderkirsch1)
```

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

```
Load(tweety, apfel1, kiste1)
```

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

Move(tweety, e)

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkirsch1“ aus der Zelle in sein Inventar aufnehmen

Take(tweety, schwarzwälderkirsch1)

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

Load(tweety, apfel1, kiste1)

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

```
Move(tweety, e)
```

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkersch1“ aus der Zelle in sein Inventar aufnehmen

```
Take(tweety, schwarzwälderkersch1)
```

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

```
Load(tweety, apfel1, kiste1)
```

Grundlegende Aktionsanforderungen 2/2

- **UnloadToGrid**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in Zelle legen
- **UnloadToInventory**(a, o_1, o_2): Agent a möchte Objekt o_1 von Objekt o_2 entfernen und in sein Inventar aufnehmen
- **NoOp**(a): Agent a möchte keine Aktion ausführen

Beispiel: Agent „tweety“ möchte nach Osten gehen

Move(tweety, e)

Beispiel: Agent „tweety“ möchte Objekt „schwarzwälderkersch1“ aus der Zelle in sein Inventar aufnehmen

Take(tweety, schwarzwälderkersch1)

Beispiel: Agent „tweety“ möchte Objekt „apfel1“ dem Objekt „kiste1“ hinzufügen

Load(tweety, apfel1, kiste1)

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

- Umgebungszustand S_t
- Einflussmenge I_t , wobei jeder Einfluss in I_t mindestens eine Regel auslöst
- Menge der Zustandsüberführungsregeln R
- totale Ordnung π auf I_t , wobei $\pi_i(I_t)$ das i -te Element von I_t gemäß π angibt
- totale Ordnung ρ auf $R^* \subseteq R$, wobei $\rho_j(R^*)$ das j -te Element von R^* gemäß ρ angibt
- $R_k \subseteq R$ notiert die von einem $k \in I_t$ ausgelösten Regeln
- Ausführungsfunktion $\text{exec}(r, z, S)$ mit $r \in R$, $z \in I_t$ und Zustand S

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_{|I_t|-1}(I_t)}|-1}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|R_{\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|\pi_{|I_t|-1}(I_t)}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_{|I_t|-1}(I_t)}|-1}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|R_{\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_{|I_t|-1}(I_t)}|-1}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|R_{\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_{|I_t|-1}(I_t)}|-1}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|R_{\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Vorgehen zur Zustandsüberführung 2/3

$$\begin{aligned} S_t &\xrightarrow{\text{exec}(\rho_0(R_{\pi_0(I_t)}), \pi_0(I_t), S_t)} S_{t,0,1} \xrightarrow{\text{exec}(\rho_1(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,1})} S_{t,0,2} \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_0(I_t)}|-1}(R_{\pi_0(I_t)}), \pi_0(I_t), S_{t,0,|R_{\pi_0(I_t)}|-1})} S_{t,1,0} \dots \\ \dots &\xrightarrow{\text{exec}(\rho_{|R_{\pi_{|I_t|-1}(I_t)}|-1}(R_{\pi_{|I_t|-1}(I_t)}), \pi_{|I_t|-1}(I_t), S_{t,|I_t|-1,|R_{\pi_{|I_t|-1}(I_t)}|-1})} S_{t+1} \end{aligned}$$

- beginne mit Einfluss höchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - wende von diesem ausgelöste Regel mit zweithöchster Priorität an
 - ...
- fahre fort mit Einfluss zweithöchster Priorität
 - wende von diesem ausgelöste Regel mit höchster Priorität an
 - ...
- ...
- wurden alle Einflüsse und alle von den Einflüssen ausgelösten Regeln verarbeitet, erreiche S_{t+1}

Beispiel

$$\begin{aligned}
 I_t &= \{ \text{MoveObject}(\text{tweety}, \text{box}, n), \\
 &\quad \text{Move}(\text{geeko}, w) \} \\
 \pi_0(I_t) &= \text{Move}(\text{geeko}, w) \\
 \pi_1(I_t) &= \text{MoveObject}(\text{tweety}, \text{box}, n) \\
 R &= \{ \text{Regel 1 mit Auslöser } \text{MoveObject}(a, o, d), \\
 &\quad \text{Regel 2 mit Auslöser } \text{MoveObject}(a, o, d), \\
 &\quad \text{Regel 3 mit Auslöser } \text{Move}(a, d) \} \\
 \rho_0(R_{\pi_0(I_t)}) &= \text{Regel 3} \\
 \rho_0(R_{\pi_1(I_t)}) &= \text{Regel 2} \\
 \rho_1(R_{\pi_1(I_t)}) &= \text{Regel 1}
 \end{aligned}$$

$$\begin{aligned}
 S_t &\xrightarrow{\text{exec}(\text{Regel 3}, \text{Move}(\text{geeko}, w), S_t)} S_{t,1,0} \\
 &\xrightarrow{\text{exec}(\text{Regel 2}, \text{MoveObject}(\text{tweety}, \text{box}, n), S_{t,1,0})} S_{t,1,1} \\
 &\xrightarrow{\text{exec}(\text{Regel 1}, \text{MoveObject}(\text{tweety}, \text{box}, n), S_{t,1,1})} S_{t+1}
 \end{aligned}$$

- $xDir(d)$: gibt abhängig von Richtung d den Offset auf der x -Achse an, z. B. $xDir(w) = -1$ oder $xDir(n) = 0$
- $yDir(d)$: analog für y -Achse
- $dirProb(d)$: gibt abhängig von Richtung d die Ausführungswahrscheinlichkeit an ($\frac{1}{\sqrt{2}}$ für diagonale Richtungen, 1 sonst)
- $euklid(x_1, y_1, x_2, y_2)$: euklidischer Abstand zwischen (x_1, y_1) und (x_2, y_2)

- $xDir(d)$: gibt abhängig von Richtung d den Offset auf der x -Achse an, z. B. $xDir(w) = -1$ oder $xDir(n) = 0$
- $yDir(d)$: analog für y -Achse
- $dirProb(d)$: gibt abhängig von Richtung d die Ausführungswahrscheinlichkeit an ($\frac{1}{\sqrt{2}}$ für diagonale Richtungen, 1 sonst)
- $euklid(x_1, y_1, x_2, y_2)$: euklidischer Abstand zwischen (x_1, y_1) und (x_2, y_2)

- $xDir(d)$: gibt abhängig von Richtung d den Offset auf der x -Achse an, z. B. $xDir(w) = -1$ oder $xDir(n) = 0$
- $yDir(d)$: analog für y -Achse
- $dirProb(d)$: gibt abhängig von Richtung d die Ausführungswahrscheinlichkeit an ($\frac{1}{\sqrt{2}}$ für diagonale Richtungen, 1 sonst)
- $euklid(x_1, y_1, x_2, y_2)$: euklidischer Abstand zwischen (x_1, y_1) und (x_2, y_2)

- **xDir(d)**: gibt abhängig von Richtung d den Offset auf der x -Achse an, z. B. $xDir(w) = -1$ oder $xDir(n) = 0$
- **yDir(d)**: analog für y -Achse
- **dirProb(d)**: gibt abhängig von Richtung d die Ausführungswahrscheinlichkeit an ($\frac{1}{\sqrt{2}}$ für diagonale Richtungen, 1 sonst)
- **euklid(x_1, y_1, x_2, y_2)**: euklidischer Abstand zwischen (x_1, y_1) und (x_2, y_2)

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
----------------	----------

Pre: Vorbedingungen

Prob: Ausführungswahrscheinlichkeit

Post: Nachbedingungen

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
<i>Pre</i> : Vorbedingungen	
<i>Prob</i> : Ausführungswahrscheinlichkeit	
<i>Post</i> : Nachbedingungen	

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberföhrungsregeln 1/2

Name der Regel	Auslöser
----------------	----------

Pre: Vorbedingungen

Prob: Ausführungswahrscheinlichkeit

Post: Nachbedingungen

- Auslöser: Einflussstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
<i>Pre</i> : Vorbedingungen	
<i>Prob</i> : Ausführungswahrscheinlichkeit	
<i>Post</i> : Nachbedingungen	

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
----------------	----------

<i>Pre:</i>	Vorbedingungen
-------------	----------------

<i>Prob:</i>	Ausführungswahrscheinlichkeit
--------------	-------------------------------

<i>Post:</i>	Nachbedingungen
--------------	-----------------

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
----------------	----------

<i>Pre</i> :	Vorbedingungen
--------------	----------------

<i>Prob</i> :	Ausführungswahrscheinlichkeit
---------------	-------------------------------

<i>Post</i> :	Nachbedingungen
---------------	-----------------

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Aufbau von Zustandsüberführungsregeln 1/2

Name der Regel	Auslöser
<i>Pre</i> : Vorbedingungen	
<i>Prob</i> : Ausführungswahrscheinlichkeit	
<i>Post</i> : Nachbedingungen	

- Auslöser: Einflusstyp (offene Formel), freie Variablen werden durch Einfluss belegt; ist Einfluss Instanz des Auslösers, wird Regel ausgelöst
- *Pre* ist Menge zu erfüllender Formeln
- *Prob* ist Wahrscheinlichkeit für Anwendung der Nachbedingungen falls *Pre* erfüllt
- *Post* beschreibt Unterschied zwischen aktuellem Zustand und Folgezustand
- unterstützte Operatoren: \leq , $=$, $+$, $-$, \cdot
- $\neg F$ in *Pre* bedeutet: F darf nicht im aktuellen Zustand gelten
- $\neg F$ in *Post* bedeutet: F wird nicht Teil des Folgezustands

Beispiel für vorliegenden Einfluss $\text{Move}(\text{tweety}, w)$

VerySimpleMove	$\text{Move}(a, d)$
----------------	---------------------

<i>Pre:</i>	$\{\text{Loc}(x, y, a), \text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d))\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(x, y, a), \text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a)\}$
--------------	--

$$S_t = \{\text{Grid}(0, 0), \text{Grid}(0, 1), \text{Grid}(0, 2), \\ \text{Agent}(\text{tweety}), \text{Loc}(1, 0, \text{Tweety})\}$$

VerySimpleMove	$\text{Move}(\text{tweety}, w)$
----------------	---------------------------------

<i>Pre:</i>	$\{\text{Loc}(1, 0, \text{tweety}), \text{Grid}(0, 0)\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(1, 0, \text{tweety}), \text{Loc}(0, 0, \text{tweety})\}$
--------------	--

Beispiel für vorliegenden Einfluss Move(tweety, w)

VerySimpleMove	Move(a, d)
----------------	------------

<i>Pre:</i>	$\{\text{Loc}(x, y, a), \text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d))\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(x, y, a), \text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a)\}$
--------------	--

$$S_t = \{\text{Grid}(0, 0), \text{Grid}(0, 1), \text{Grid}(0, 2), \\ \text{Agent}(\text{tweety}), \text{Loc}(1, 0, \text{Tweety})\}$$

VerySimpleMove	Move(tweety,w)
----------------	----------------

<i>Pre:</i>	$\{\text{Loc}(1, 0, \text{tweety}), \text{Grid}(0, 0)\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(1, 0, \text{tweety}), \text{Loc}(0, 0, \text{tweety})\}$
--------------	--

Beispiel für vorliegenden Einfluss Move(tweety, w)

VerySimpleMove	Move(a, d)
----------------	------------

<i>Pre:</i>	$\{\text{Loc}(x, y, a), \text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d))\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(x, y, a), \text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a)\}$
--------------	--

$$S_t = \{\text{Grid}(0, 0), \text{Grid}(0, 1), \text{Grid}(0, 2), \\ \text{Agent}(\text{tweety}), \text{Loc}(1, 0, \text{Tweety})\}$$

VerySimpleMove	Move(tweety,w)
----------------	----------------

<i>Pre:</i>	$\{\text{Loc}(1, 0, \text{tweety}), \text{Grid}(0, 0)\}$
-------------	--

<i>Prob:</i>	1
--------------	---

<i>Post:</i>	$\{\neg\text{Loc}(1, 0, \text{tweety}), \text{Loc}(0, 0, \text{tweety})\}$
--------------	--

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{\text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg\text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg\text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2\}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{\neg\text{Loc}(x, y, a),$ $\neg\text{FreeCellCap}(x, y, c_1),$ $\neg\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n)\}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{\text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg\text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg\text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2\}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{\neg\text{Loc}(x, y, a),$ $\neg\text{FreeCellCap}(x, y, c_1),$ $\neg\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n)\}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

Zustandsüberführungsregel AgentMove

AgentMove	Move(a, d)
<i>Pre:</i>	$\{ \text{Loc}(x, y, a),$ $\text{Grid}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Wall}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\neg \text{Trench}(x + x\text{Dir}(d), y + y\text{Dir}(d)),$ $\text{FreeCellCap}(x, y, c_1),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{CapNeed}(a, n),$ $n \leq c_2 \}$
<i>Prob:</i>	$\text{dirProb}(d)$
<i>Post:</i>	$\{ \neg \text{Loc}(x, y, a),$ $\neg \text{FreeCellCap}(x, y, c_1),$ $\neg \text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2),$ $\text{Loc}(x + x\text{Dir}(d), y + y\text{Dir}(d), a),$ $\text{FreeCellCap}(x, y, c_1 + n),$ $\text{FreeCellCap}(x + x\text{Dir}(d), y + y\text{Dir}(d), c_2 - n) \}$

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

- Aufnahme von Objekten in das Inventar
- Ablage von Objekten aus dem Inventar
- Verschiebung von Objekten
- Zustimmung und Widerruf von Bereitschaftserklärungen zur Annahme von Objekten
- Übergabe von Objekten an andere Agenten
- Beladen eines Objekts mit einem anderen Objekt
- Entnahme eines Objekts aus einem anderen Objekt in das Inventar
- Entnahme eines Objekts aus einem anderen Objekt auf das Grid

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um \neg Locked(o)

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um \neg Locked(o)

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um \neg Locked(o)

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um $\neg\text{Locked}(o)$

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um $\neg\text{Locked}(o)$

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von *Pre* bestimmter Regeln um $\neg\text{Locked}(o)$

Schließfächer als komplexe Objekte

- **Schließfächer** können Objekte aufnehmen und mit **Passwort** verschlossen werden
- Entnahme von Objekten nur aus geöffnetem Schließfach möglich
- Öffnung eines verschlossenen Schließfachs nur mit korrektem Passwort
- neue Prädikate:
 - **SafeDepositBox(b)**: b ist Schließfach
 - **Password(b, p)**: Passwort p schützt Schließfach b
 - **Locked(b)**: Schließfach b ist verschlossen
- neue Aktionsanforderungen:
 - **Lock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p verschließen
 - **Unlock(a, b, p)**: Agent a möchte Schließfach b mit Passwort p öffnen
- neue Zustandsüberführungsregeln **LockDepositBox** und **UnlockDepositBox**
- Erweiterung von Pre bestimmter Regeln um $\neg\text{Locked}(o)$

- notwendig für proaktives Verhalten
- werden durch **Einflussserzeugungsregeln** erzeugt
- Anwendung von Einflussserzeugungsregeln bei jedem Zeitfortschritt
- Einflussserzeugungsregeln haben folgende Form:

Name der Regel

Pre: Vorbedingungen in S_t

Prob: Ausführungswahrscheinlichkeit

Post: erzeugte Einflüsse

- notwendig für proaktives Verhalten
- werden durch **Einflusserzeugungsregeln** erzeugt
- Anwendung von Einflusserzeugungsregeln bei jedem Zeitfortschritt
- Einflusserzeugungsregeln haben folgende Form:

Name der Regel

Pre: Vorbedingungen in S_t

Prob: Ausführungswahrscheinlichkeit

Post: erzeugte Einflüsse

- notwendig für proaktives Verhalten
- werden durch **Einflusserzeugungsregeln** erzeugt
- Anwendung von Einflusserzeugungsregeln bei jedem Zeitfortschritt
- Einflusserzeugungsregeln haben folgende Form:

Name der Regel

Pre: Vorbedingungen in S_t

Prob: Ausführungswahrscheinlichkeit

Post: erzeugte Einflüsse

- notwendig für proaktives Verhalten
- werden durch **Einflusserzeugungsregeln** erzeugt
- Anwendung von Einflusserzeugungsregeln bei jedem Zeitfortschritt
- Einflusserzeugungsregeln haben folgende Form:

Name der Regel

Pre: Vorbedingungen in S_t

Prob: Ausführungswahrscheinlichkeit

Post: erzeugte Einflüsse

Nebel als proaktives Verhalten

- neues Prädikat **Fog(x, y)**: Zelle (x, y) beinhaltet Nebel
- neue Einfluss erzeugungsregeln:

FogInfluence

Pre: {Grid(x, y)}

Prob: $p_{\text{createfog}}$

Post: MakeFog(x, y)

UnfogInfluence

Pre: {Grid(x, y)}

Prob: $p_{\text{removefog}}$

Post: RemoveFog(x, y)

- tatsächliche Wahrscheinlichkeiten abhängig von der Priorität der Einflüsse in π
- neue Zustandsüberführungsregeln: **CreateFog**, **UncreateFog**

Nebel als proaktives Verhalten

- neues Prädikat $\text{Fog}(x, y)$: Zelle (x, y) beinhaltet Nebel
- neue Einfluss erzeugungsregeln:

FogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{createfog}}$

Post: $\text{MakeFog}(x, y)$

UnfogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{removefog}}$

Post: $\text{RemoveFog}(x, y)$

- tatsächliche Wahrscheinlichkeiten abhängig von der Priorität der Einflüsse in π
- neue Zustandsüberführungsregeln: CreateFog , UncreateFog

Nebel als proaktives Verhalten

- neues Prädikat $\text{Fog}(x, y)$: Zelle (x, y) beinhaltet Nebel
- neue Einfluss erzeugungsregeln:

FogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{createfog}}$

Post: $\text{MakeFog}(x, y)$

UnfogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{removefog}}$

Post: $\text{RemoveFog}(x, y)$

- tatsächliche Wahrscheinlichkeiten abhängig von der Priorität der Einflüsse in π
- neue Zustandsüberföhrungsregeln: CreateFog , UncreateFog

Nebel als proaktives Verhalten

- neues Prädikat $\text{Fog}(x, y)$: Zelle (x, y) beinhaltet Nebel
- neue Einfluss erzeugungsregeln:

FogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{createfog}}$

Post: $\text{MakeFog}(x, y)$

UnfogInfluence

Pre: $\{\text{Grid}(x, y)\}$

Prob: $p_{\text{removefog}}$

Post: $\text{RemoveFog}(x, y)$

- tatsächliche Wahrscheinlichkeiten abhängig von der Priorität der Einflüsse in π
- neue Zustandsüberführungsregeln: CreateFog , UncreateFog

Nebel als proaktives Verhalten

- neues Prädikat **Fog**(x, y): Zelle (x, y) beinhaltet Nebel
- neue Einfluss erzeugungsregeln:

FogInfluence

Pre: {Grid(x, y)}

Prob: $p_{\text{createfog}}$

Post: MakeFog(x, y)

UnfogInfluence

Pre: {Grid(x, y)}

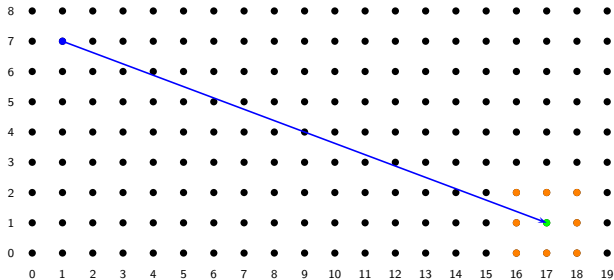
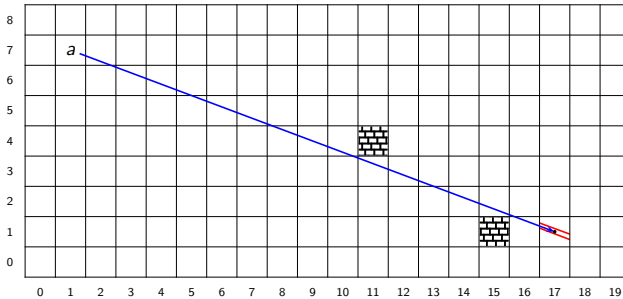
Prob: $p_{\text{removefog}}$

Post: RemoveFog(x, y)

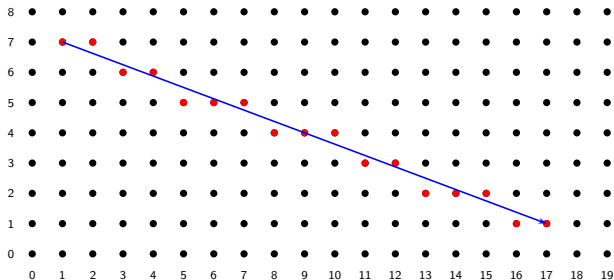
- tatsächliche Wahrscheinlichkeiten abhängig von der Priorität der Einflüsse in π
- neue Zustandsüberführungsregeln: **CreateFog**, **UncreateFog**

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung**
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

Sichtlinien 1/2

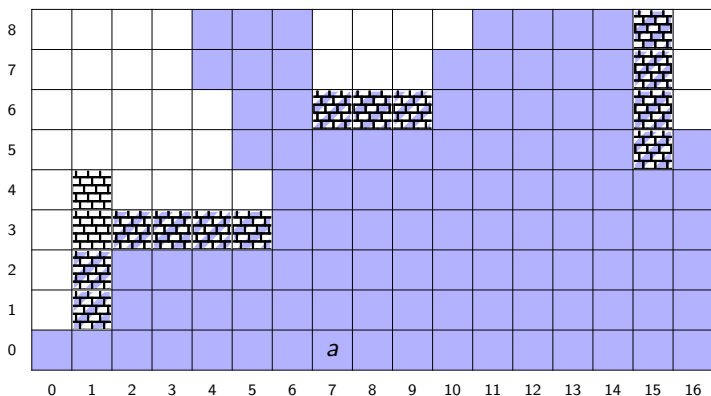


- Verwendung kontinuierlicher Sichtlinien problematisch
- stattdessen: Approximation durch **diskrete Sichtlinien**



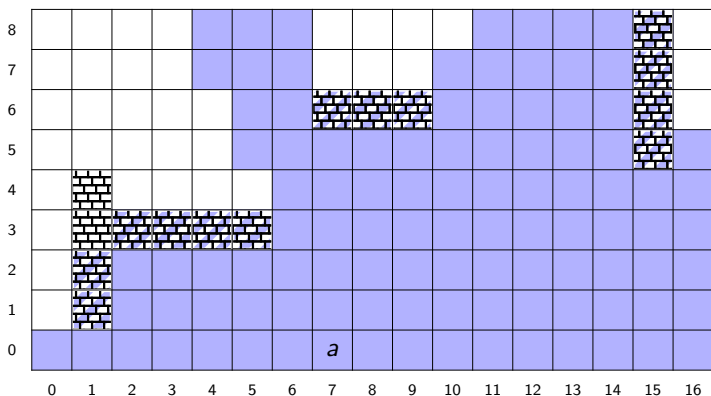
- $dsl(x_a, y_a, x_g, y_g)$: Menge aller Zellen auf der diskreten Sichtlinie von (x_a, y_a) nach (x_g, y_g)

Sichtbeschränkung durch Hindernisse



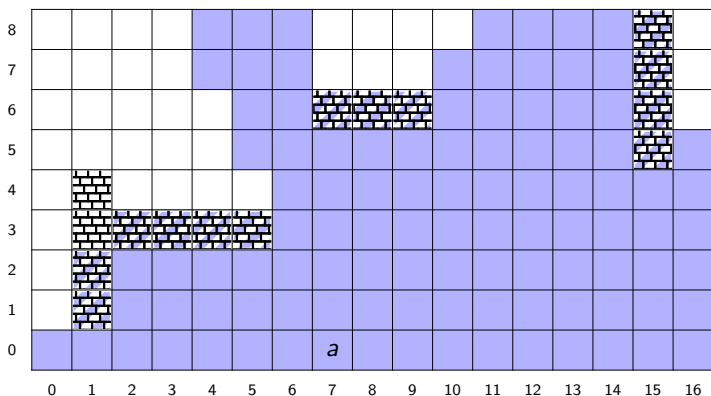
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) vollständig sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g)$
- Sichthindernis in Zelle (x_g, y_g) sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g) \setminus \{(x_g, y_g)\}$

Sichtbeschränkung durch Hindernisse



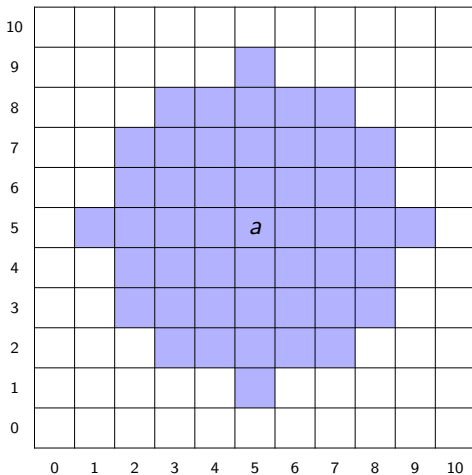
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) vollständig sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g)$
- Sichthindernis in Zelle (x_g, y_g) sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g) \setminus \{(x_g, y_g)\}$

Sichtbeschränkung durch Hindernisse



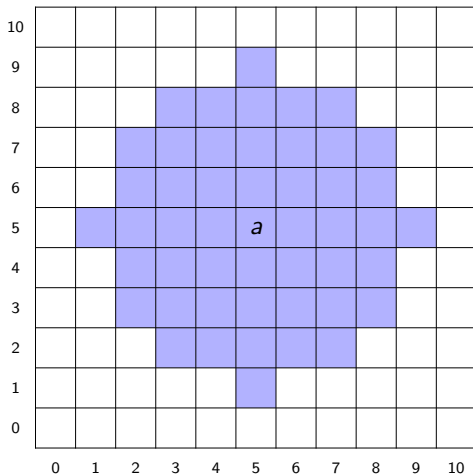
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) vollständig sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g)$
- Sichthindernis in Zelle (x_g, y_g) sichtbar, wenn kein Sichthindernis in $dsl(x_a, y_a, x_g, y_g) \setminus \{(x_g, y_g)\}$

Sichtbeschränkung durch Sichtweite



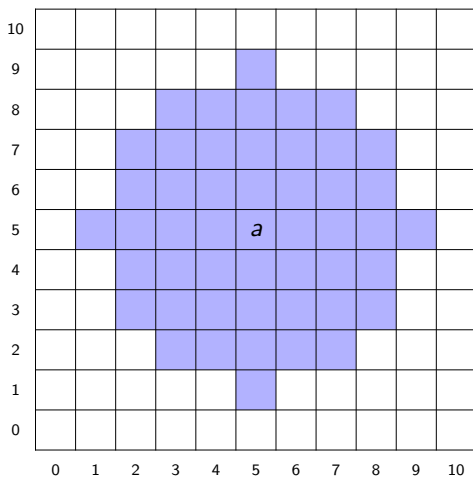
- **ViewRange(a, v):** Sichtweite des Agenten a ist v
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) sichtbar, wenn $\text{euklid}(x_a, y_a, x_g, y_g) \leq v$

Sichtbeschränkung durch Sichtweite



- **ViewRange(a, v)**: Sichtweite des Agenten a ist v
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) sichtbar, wenn $\text{euklid}(x_a, y_a, x_g, y_g) \leq v$

Sichtbeschränkung durch Sichtweite



- **ViewRange(a, v)**: Sichtweite des Agenten a ist v
- befinde sich Agent a in Zelle (x_a, y_a)
- Zelle (x_g, y_g) sichtbar, wenn $\text{euklid}(x_a, y_a, x_g, y_g) \leq v$

Bestimmung der Beobachtungsmenge

- sei $\text{Loc}(x_a, y_a, a) \in S_t$
- Menge $O_{S_t, a}^0$ der bzgl. Hindernissen und Sichtweite sichtbaren Agenten, Objekte und Hindernisse (ohne Zelle des Agenten)

aktuelle Zelle des Agenten immer sichtbar

$$O_{S_t, a}^1 = O_{S_t, a}^0 \cup (S_t \cap (\{\text{Grid}(x_a, y_a)\} \cup \{\text{Fog}(x_a, y_a)\} \\ \cup \{\text{Curtain}(x_a, y_a)\} \cup \{\text{Object}(o) \mid \text{Loc}(x_a, y_a, o) \in S_t\} \\ \cup \{\text{Agent}(z) \mid \text{Loc}(x_a, y_a, z) \in S_t\}))$$

Orte beobachtbarer Agenten und Objekte

$$O_{S_t, a}^2 = O_{S_t, a}^1 \cup \{\text{Loc}(x, y, z) \in S_t \mid \text{Object}(z) \in O_{S_t, a}^1 \\ \vee \text{Agent}(z) \in O_{S_t, a}^1\}$$

- $O_{S_t, a}^3 \dots O_{S_t, a}^8$: freie Kapazitäten und Kapazitätsbedarfe, Schließfächer, individuelle Objekttypen, eigenes Inventar, enthaltene Objekte, Enthaltenseinsbeziehungen

Bestimmung der Beobachtungsmenge

- sei $\text{Loc}(x_a, y_a, a) \in S_t$
- Menge $O_{S_t, a}^0$ der bzgl. Hindernissen und Sichtweite sichtbaren Agenten, Objekte und Hindernisse (ohne Zelle des Agenten)

aktuelle Zelle des Agenten immer sichtbar

$$O_{S_t, a}^1 = O_{S_t, a}^0 \cup (S_t \cap (\{\text{Grid}(x_a, y_a)\} \cup \{\text{Fog}(x_a, y_a)\} \\ \cup \{\text{Curtain}(x_a, y_a)\} \cup \{\text{Object}(o) \mid \text{Loc}(x_a, y_a, o) \in S_t\} \\ \cup \{\text{Agent}(z) \mid \text{Loc}(x_a, y_a, z) \in S_t\}))$$

Orte beobachtbarer Agenten und Objekte

$$O_{S_t, a}^2 = O_{S_t, a}^1 \cup \{\text{Loc}(x, y, z) \in S_t \mid \text{Object}(z) \in O_{S_t, a}^1 \\ \vee \text{Agent}(z) \in O_{S_t, a}^1\}$$

- $O_{S_t, a}^3 \dots O_{S_t, a}^8$: freie Kapazitäten und Kapazitätsbedarfe, Schließfächer, individuelle Objekttypen, eigenes Inventar, enthaltene Objekte, Enthaltenseinsbeziehungen

Bestimmung der Beobachtungsmenge

- sei $\text{Loc}(x_a, y_a, a) \in S_t$
- Menge $O_{S_t, a}^0$ der bzgl. Hindernissen und Sichtweite sichtbaren Agenten, Objekte und Hindernisse (ohne Zelle des Agenten)

aktuelle Zelle des Agenten immer sichtbar

$$\begin{aligned} O_{S_t, a}^1 &= O_{S_t, a}^0 \cup (S_t \cap (\{\text{Grid}(x_a, y_a)\} \cup \{\text{Fog}(x_a, y_a)\} \\ &\quad \cup \{\text{Curtain}(x_a, y_a)\} \cup \{\text{Object}(o) \mid \text{Loc}(x_a, y_a, o) \in S_t\} \\ &\quad \cup \{\text{Agent}(z) \mid \text{Loc}(x_a, y_a, z) \in S_t\})) \end{aligned}$$

Orte beobachtbarer Agenten und Objekte

$$\begin{aligned} O_{S_t, a}^2 &= O_{S_t, a}^1 \cup \{\text{Loc}(x, y, z) \in S_t \mid \text{Object}(z) \in O_{S_t, a}^1 \\ &\quad \vee \text{Agent}(z) \in O_{S_t, a}^1\} \end{aligned}$$

- $O_{S_t, a}^3 \dots O_{S_t, a}^8$: freie Kapazitäten und Kapazitätsbedarfe, Schließfächer, individuelle Objekttypen, eigenes Inventar, enthaltene Objekte, Enthaltenseinsbeziehungen

Bestimmung der Beobachtungsmenge

- sei $\text{Loc}(x_a, y_a, a) \in S_t$
- Menge $O_{S_t, a}^0$ der bzgl. Hindernissen und Sichtweite sichtbaren Agenten, Objekte und Hindernisse (ohne Zelle des Agenten)

aktuelle Zelle des Agenten immer sichtbar

$$\begin{aligned} O_{S_t, a}^1 &= O_{S_t, a}^0 \cup (S_t \cap (\{\text{Grid}(x_a, y_a)\} \cup \{\text{Fog}(x_a, y_a)\} \\ &\quad \cup \{\text{Curtain}(x_a, y_a)\} \cup \{\text{Object}(o) \mid \text{Loc}(x_a, y_a, o) \in S_t\} \\ &\quad \cup \{\text{Agent}(z) \mid \text{Loc}(x_a, y_a, z) \in S_t\})) \end{aligned}$$

Orte beobachtbarer Agenten und Objekte

$$\begin{aligned} O_{S_t, a}^2 &= O_{S_t, a}^1 \cup \{\text{Loc}(x, y, z) \in S_t \mid \text{Object}(z) \in O_{S_t, a}^1 \\ &\quad \vee \text{Agent}(z) \in O_{S_t, a}^1\} \end{aligned}$$

- $O_{S_t, a}^3 \dots O_{S_t, a}^8$: freie Kapazitäten und Kapazitätsbedarfe, Schließfächer, individuelle Objekttypen, eigenes Inventar, enthaltene Objekte, Enthaltenseinsbeziehungen

Bestimmung der Beobachtungsmenge

- sei $\text{Loc}(x_a, y_a, a) \in S_t$
- Menge $O_{S_t, a}^0$ der bzgl. Hindernissen und Sichtweite sichtbaren Agenten, Objekte und Hindernisse (ohne Zelle des Agenten)

aktuelle Zelle des Agenten immer sichtbar

$$\begin{aligned} O_{S_t, a}^1 &= O_{S_t, a}^0 \cup (S_t \cap (\{\text{Grid}(x_a, y_a)\} \cup \{\text{Fog}(x_a, y_a)\} \\ &\quad \cup \{\text{Curtain}(x_a, y_a)\} \cup \{\text{Object}(o) \mid \text{Loc}(x_a, y_a, o) \in S_t\} \\ &\quad \cup \{\text{Agent}(z) \mid \text{Loc}(x_a, y_a, z) \in S_t\})) \end{aligned}$$

Orte beobachtbarer Agenten und Objekte

$$\begin{aligned} O_{S_t, a}^2 &= O_{S_t, a}^1 \cup \{\text{Loc}(x, y, z) \in S_t \mid \text{Object}(z) \in O_{S_t, a}^1 \\ &\quad \vee \text{Agent}(z) \in O_{S_t, a}^1\} \end{aligned}$$

- $O_{S_t, a}^3 \dots O_{S_t, a}^8$: freie Kapazitäten und Kapazitätsbedarfe, Schließfächer, individuelle Objekttypen, eigenes Inventar, enthaltene Objekte, Enthaltenseinsbeziehungen

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation**
- 6 Implementierung
- 7 Einordnung und Ausblick

- Nachrichtenfaktum im Zustand für jeden Empfänger einer jeden Nachricht
- private Nachrichten
 - nur vom intendierten Empfänger empfangbar
 - Absender und Empfänger in derselben Zelle
- öffentliche Nachrichten
 - potenziell von allen Agenten empfangbar
 - Empfangbarkeit abhängig von:
 - Sendestärke des Absenders: $\text{SoundIntensity}(a, s)$
 - Empfangsempfindlichkeit des Empfängers: $\text{Hearing}(a, h)$
 - Interferenzen zwischen Absender und Empfänger
 - Absender dem Empfänger nicht immer bekannt

- Nachrichtenfaktum im Zustand für jeden Empfänger einer jeden Nachricht
- private Nachrichten
 - nur vom intendierten Empfänger empfangbar
 - Absender und Empfänger in derselben Zelle
- öffentliche Nachrichten
 - potenziell von allen Agenten empfangbar
 - Empfangbarkeit abhängig von:
 - Sendestärke des Absenders: $\text{SoundIntensity}(a, s)$
 - Empfangsempfindlichkeit des Empfängers: $\text{Hearing}(a, h)$
 - Interferenzen zwischen Absender und Empfänger
 - Absender dem Empfänger nicht immer bekannt

- Nachrichtenfaktum im Zustand für jeden Empfänger einer jeden Nachricht
- private Nachrichten
 - nur vom intendierten Empfänger empfangbar
 - Absender und Empfänger in derselben Zelle
- öffentliche Nachrichten
 - potenziell von allen Agenten empfangbar
 - Empfangbarkeit abhängig von:
 - Sendestärke des Absenders: $\text{SoundIntensity}(a, s)$
 - Empfangsempfindlichkeit des Empfängers: $\text{Hearing}(a, h)$
 - Interferenzen zwischen Absender und Empfänger
 - Absender dem Empfänger nicht immer bekannt

- neues Prädikat: $\text{PrivMsg}(a, a_1, m)$
- neue Aktionsanforderung $\text{SendPrivMsg}(a, a_1, m)$: Agent a möchte private Nachricht m an Agent a_1 versenden
- neue Zustandsüberführungsregel:

	CreatePrivMsg	SendPrivMsg(a, a_1, m)
<i>Pre:</i>	$\{\text{Loc}(x, y, a),$ $\text{Loc}(x, y, a_1),$ $\text{SoundIntensity}(a, s),$ $\text{Hearing}(a_1, h),$ $\neg\text{Interference}(x, y)\}$	
<i>Prob:</i>	1	
<i>Post:</i>	$\{\text{PrivMsg}(a, a_1, m)\}$	

Erzeugung privater Nachrichten

- neues Prädikat: $\text{PrivMsg}(a, a_1, m)$
- neue Aktionsanforderung $\text{SendPrivMsg}(a, a_1, m)$: Agent a möchte private Nachricht m an Agent a_1 versenden
- neue Zustandsüberführungsregel:

	CreatePrivMsg	SendPrivMsg(a, a_1, m)
<i>Pre:</i>	$\{\text{Loc}(x, y, a),$ $\text{Loc}(x, y, a_1),$ $\text{SoundIntensity}(a, s),$ $\text{Hearing}(a_1, h),$ $\neg\text{Interference}(x, y)\}$	
<i>Prob:</i>	1	
<i>Post:</i>	$\{\text{PrivMsg}(a, a_1, m)\}$	

- neues Prädikat: $\text{PrivMsg}(a, a_1, m)$
- neue Aktionsanforderung $\text{SendPrivMsg}(a, a_1, m)$: Agent a möchte private Nachricht m an Agent a_1 versenden
- neue Zustandsüberführungsregel:

	CreatePrivMsg	SendPrivMsg(a, a_1, m)
<i>Pre:</i>	$\{\text{Loc}(x, y, a),$ $\text{Loc}(x, y, a_1),$ $\text{SoundIntensity}(a, s),$ $\text{Hearing}(a_1, h),$ $\neg\text{Interference}(x, y)\}$	
<i>Prob:</i>	1	
<i>Post:</i>	$\{\text{PrivMsg}(a, a_1, m)\}$	

- sei $\{\text{SoundIntensity}(a, s), \text{Hearing}(a_1, h)\} \subset S_t$ und $\{\text{Loc}(x_1, y_1, a), \text{Loc}(x_2, y_2, a_1)\} \subset S_t$
- öffentliche Nachricht von Agent a von Agent a_1 empfangbar, genau dann wenn:
 - $h + s \geq \text{euklid}(x_1, y_1, x_2, y_2)$ und
 - $\text{dsl}(x_1, y_1, x_2, y_2)$ enthält keine Zelle mit Interferenz
- neue Prädikate:
 - falls Sichtverbindung, dann Absender erkennbar:
 $\text{PubMsg}(a, a_1, m)$
 - ohne Sichtverbindung Absender nicht erkennbar:
 $\text{AnonPubMsg}(a_1, m)$
- neue Aktionsanforderung $\text{SendPubMsg}(a, m)$: Agent a möchte öffentliche Nachricht m versenden
- neue Zustandsüberführungsregeln: CreatePubMsgVisi , $\text{CreatePubMsgInvisi}$

- sei $\{\text{SoundIntensity}(a, s), \text{Hearing}(a_1, h)\} \subset S_t$ und $\{\text{Loc}(x_1, y_1, a), \text{Loc}(x_2, y_2, a_1)\} \subset S_t$
- öffentliche Nachricht von Agent a von Agent a_1 empfangbar, genau dann wenn:
 - $h + s \geq \text{euklid}(x_1, y_1, x_2, y_2)$ und
 - $\text{dsl}(x_1, y_1, x_2, y_2)$ enthält keine Zelle mit Interferenz
- neue Prädikate:
 - falls Sichtverbindung, dann Absender erkennbar:
 $\text{PubMsg}(a, a_1, m)$
 - ohne Sichtverbindung Absender nicht erkennbar:
 $\text{AnonPubMsg}(a_1, m)$
- neue Aktionsanforderung $\text{SendPubMsg}(a, m)$: Agent a möchte öffentliche Nachricht m versenden
- neue Zustandsüberführungsregeln: CreatePubMsgVisi , $\text{CreatePubMsgInvisi}$

- sei $\{\text{SoundIntensity}(a, s), \text{Hearing}(a_1, h)\} \subset S_t$ und $\{\text{Loc}(x_1, y_1, a), \text{Loc}(x_2, y_2, a_1)\} \subset S_t$
- öffentliche Nachricht von Agent a von Agent a_1 empfangbar, genau dann wenn:
 - $h + s \geq \text{euklid}(x_1, y_1, x_2, y_2)$ und
 - $\text{dsl}(x_1, y_1, x_2, y_2)$ enthält keine Zelle mit Interferenz
- neue Prädikate:
 - falls Sichtverbindung, dann Absender erkennbar:
 $\text{PubMsg}(a, a_1, m)$
 - ohne Sichtverbindung Absender nicht erkennbar:
 $\text{AnonPubMsg}(a_1, m)$
- neue Aktionsanforderung $\text{SendPubMsg}(a, m)$: Agent a möchte öffentliche Nachricht m versenden
- neue Zustandsüberführungsregeln: CreatePubMsgVisi , $\text{CreatePubMsgInvisi}$

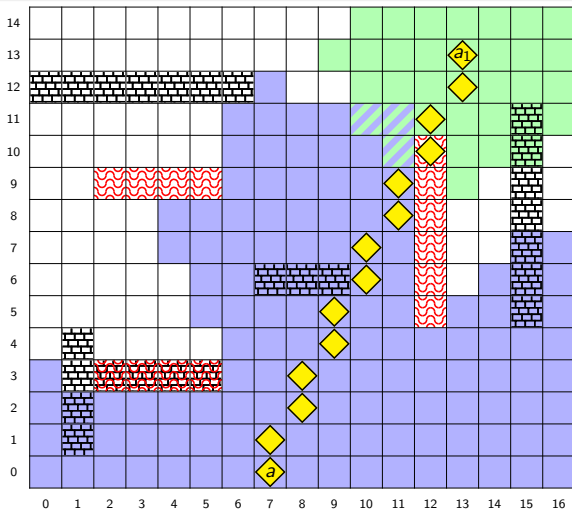
- sei $\{\text{SoundIntensity}(a, s), \text{Hearing}(a_1, h)\} \subset S_t$ und $\{\text{Loc}(x_1, y_1, a), \text{Loc}(x_2, y_2, a_1)\} \subset S_t$
- öffentliche Nachricht von Agent a von Agent a_1 empfangbar, genau dann wenn:
 - $h + s \geq \text{euklid}(x_1, y_1, x_2, y_2)$ und
 - $\text{dsl}(x_1, y_1, x_2, y_2)$ enthält keine Zelle mit Interferenz
- neue Prädikate:
 - falls Sichtverbindung, dann Absender erkennbar:
 $\text{PubMsg}(a, a_1, m)$
 - ohne Sichtverbindung Absender nicht erkennbar:
 $\text{AnonPubMsg}(a_1, m)$
- neue Aktionsanforderung $\text{SendPubMsg}(a, m)$: Agent a möchte öffentliche Nachricht m versenden
- neue Zustandsüberführungsregeln: CreatePubMsgVisi , $\text{CreatePubMsgInvisi}$


- sei $\{\text{SoundIntensity}(a, s), \text{Hearing}(a_1, h)\} \subset S_t$ und $\{\text{Loc}(x_1, y_1, a), \text{Loc}(x_2, y_2, a_1)\} \subset S_t$
- öffentliche Nachricht von Agent a von Agent a_1 empfangbar, genau dann wenn:
 - $h + s \geq \text{euklid}(x_1, y_1, x_2, y_2)$ und
 - $\text{dsl}(x_1, y_1, x_2, y_2)$ enthält keine Zelle mit Interferenz
- neue Prädikate:
 - falls Sichtverbindung, dann Absender erkennbar:
 $\text{PubMsg}(a, a_1, m)$
 - ohne Sichtverbindung Absender nicht erkennbar:
 $\text{AnonPubMsg}(a_1, m)$
- neue Aktionsanforderung $\text{SendPubMsg}(a, m)$: Agent a möchte öffentliche Nachricht m versenden
- neue Zustandsüberführungsregeln: CreatePubMsgVisi , $\text{CreatePubMsgInvisi}$


Endgültige Wahrnehmungsmenge


$$O_{S_t, a} = O_{S_t, a}^8 \cup \{\text{PrivMsg}(z, a, m) \in S_t\} \\ \cup \{\text{PubMsg}(z, a, m) \in S_t\} \cup \{\text{AnonPubMsg}(a, m) \in S_t\}$$

Beispiel: Erfolgreiche Kommunikation




 = Sendereichweite von a

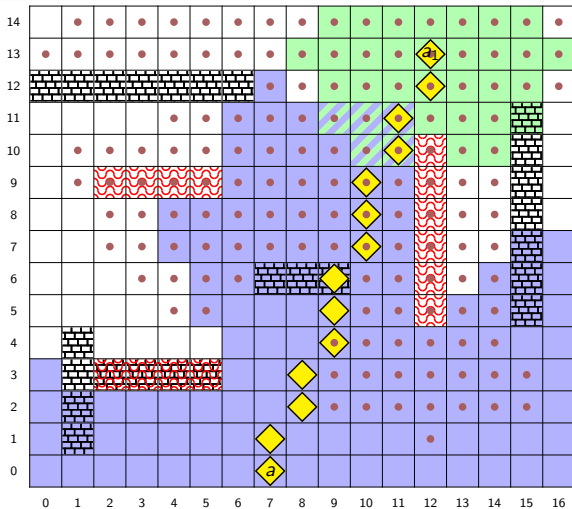
 = Empfangsreichweite von a_1

 = Sichtbeh. Hindernis

 = Kommunikationsbl. Hindernis

 = $dsl(7, 0, 13, 13)$

Beispiel: Erfolgreiche Kommunikation (anonym)



- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung**
- 7 Einordnung und Ausblick

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- plattformunabhängig (Java SE 6)
- bestehend aus:
 - Server
 - Agenten-Client
 - Beobachter-Client
 - API zur Verwendung durch Agenten
- Spezifikation von Umgebungen in XML-Sprache
- erweiterbar durch eigene Regeln und eigene komplexe Typen von Agenten und Objekten
- Kapselung beliebiger Nachrichteninhalte
- Kommunikation von Agenten und Beobachter-Clients mit dem Server in XML-Sprache über TCP/IP
- in Q1/2011 voraussichtlich verfügbar unter <http://www.tittel.net/gridworldsim/>

- 1 Einführung
- 2 Konzeptioneller Überblick
- 3 Zustand und Zustandsüberführung
- 4 Wahrnehmung
- 5 Kommunikation
- 6 Implementierung
- 7 Einordnung und Ausblick

- **Environment Description Language for Multi-Agent Simulation (ELMS)**
 - derzeit keine Implementierung verfügbar
 - unterstützt eigene komplexe Objekttypen direkt in der Spezifikation
 - Spezifikationsprache nicht für den menschlichen Gebrauch entwickelt
 - nicht mächtig genug für komplexe Funktionalität
 - nicht proaktiv
 - keine Nachrichtenvermittlung
 - unterstützt komplexe Spezifikation der Wahrnehmung
- **MASSim**
 - entwickelt für Wettbewerbssituationen
 - Semantik von Umgebungen nicht Teil von MASSim, sondern selbst in Java zu implementieren
 - dadurch sehr flexibel
 - Entwicklung eigener Umgebungen aber sehr aufwändig

- **Environment Description Language for Multi-Agent Simulation (ELMS)**
 - derzeit keine Implementierung verfügbar
 - unterstützt eigene komplexe Objekttypen direkt in der Spezifikation
 - Spezifikationsprache nicht für den menschlichen Gebrauch entwickelt
 - nicht mächtig genug für komplexe Funktionalität
 - nicht proaktiv
 - keine Nachrichtenvermittlung
 - unterstützt komplexe Spezifikation der Wahrnehmung
- **MASSim**
 - entwickelt für Wettbewerbssituationen
 - Semantik von Umgebungen nicht Teil von MASSim, sondern selbst in Java zu implementieren
 - dadurch sehr flexibel
 - Entwicklung eigener Umgebungen aber sehr aufwändig

- vorgestellter Formalismus zur Beschreibung der Semantik von Umgebungen hat sich als geeignet erwiesen
- entwickeltes Framework ohne echte Alternative
- dennoch nach wie vor große Diskrepanz zwischen realer und simulierter Welt
- mögliche Erweiterungen:
 - Zerstörbarkeit von Objekten und Agenten
 - komplexere Naturereignisse
 - dreidimensionales Grid
 - flexible Spezifikation der Wahrnehmung
 - Zusammenbauen von Objekten zu Objekten neuen Typs
 - logikbasierte Funktionalitätserweiterungen

- vorgestellter Formalismus zur Beschreibung der Semantik von Umgebungen hat sich als geeignet erwiesen
- entwickeltes Framework ohne echte Alternative
- dennoch nach wie vor große Diskrepanz zwischen realer und simulierter Welt
- mögliche Erweiterungen:
 - Zerstörbarkeit von Objekten und Agenten
 - komplexere Naturereignisse
 - dreidimensionales Grid
 - flexible Spezifikation der Wahrnehmung
 - Zusammenbauen von Objekten zu Objekten neuen Typs
 - logikbasierte Funktionalitätserweiterungen

- vorgestellter Formalismus zur Beschreibung der Semantik von Umgebungen hat sich als geeignet erwiesen
- entwickeltes Framework ohne echte Alternative
- dennoch nach wie vor große Diskrepanz zwischen realer und simulierter Welt
- mögliche Erweiterungen:
 - Zerstörbarkeit von Objekten und Agenten
 - komplexere Naturereignisse
 - dreidimensionales Grid
 - flexible Spezifikation der Wahrnehmung
 - Zusammenbauen von Objekten zu Objekten neuen Typs
 - logikbasierte Funktionalitätserweiterungen

- vorgestellter Formalismus zur Beschreibung der Semantik von Umgebungen hat sich als geeignet erwiesen
- entwickeltes Framework ohne echte Alternative
- dennoch nach wie vor große Diskrepanz zwischen realer und simulierter Welt
- mögliche Erweiterungen:
 - Zerstörbarkeit von Objekten und Agenten
 - komplexere Naturereignisse
 - dreidimensionales Grid
 - flexible Spezifikation der Wahrnehmung
 - Zusammenbauen von Objekten zu Objekten neuen Typs
 - logikbasierte Funktionalitätserweiterungen