Koordination mehrerer Agenten

Hans Moog

11. Juni 2008

1. Motivation
   - Rückblick
   - Probleme und Ziele
   - Beispiel: Coordinated Attack
2. Grundlagen
   - Formelle Zusammenfassung: Multi-Agenten-Systeme
   - Actions
   - Protokolle
   - Kontexte
3. Coordinated Attack
   - Problemstellung
   - Vorbetrachtung
   - Problemanalyse
   - Beweis

Multi-Agenten-Systeme

- Multi-Agenten-Systeme beinhalten trivialerweise n Agenten mit $n > 1$.

Multi-Agenten-Systeme

- Multi-Agenten-Systeme beinhalten trivialerweise n Agenten mit $n > 1$.
- Die Agenten haben dabei lokale, voneinander unabhängige Zustände, die zusammen den globalen Zustand des Systems formen.

Multi-Agenten-Systeme

- Multi-Agenten-Systeme beinhalten trivialerweise n Agenten mit $n > 1$.
- Die Agenten haben dabei lokale, voneinander unabhängige Zustände, die zusammen den globalen Zustand des Systems formen.
- Die Zustände der Agenten, und damit auch die globalen Zustände, können sich ändern.

Multi-Agenten-Systeme

- Multi-Agenten-Systeme beinhalten trivialerweise n Agenten mit $n > 1$.
- Die Agenten haben dabei lokale, voneinander unabhängige Zustände, die zusammen den globalen Zustand des Systems formen.
- Die Zustände der Agenten, und damit auch die globalen Zustände, können sich ändern.
- Die unterschiedlichen Permutationen der lokalen Zustände bilden dabei die Gesamtheit der möglichen globalen Zustände des Systems, zwischen denen die Zustandsänderungen möglich sind.

Multi-Agenten-Systeme

- Eine bestimmte durch Zustandsänderungen gebildete Folge von Zuständen bezeichnet man als Durchgang.

Multi-Agenten-Systeme

- Eine bestimmte durch Zustandsänderungen gebildete Folge von Zuständen bezeichnet man als Durchgang.
- Die nichtleere Menge aller Durchgänge über die Menge der globalen Zustände bilden dabei letztendlich das System.

Multi-Agenten-Systeme

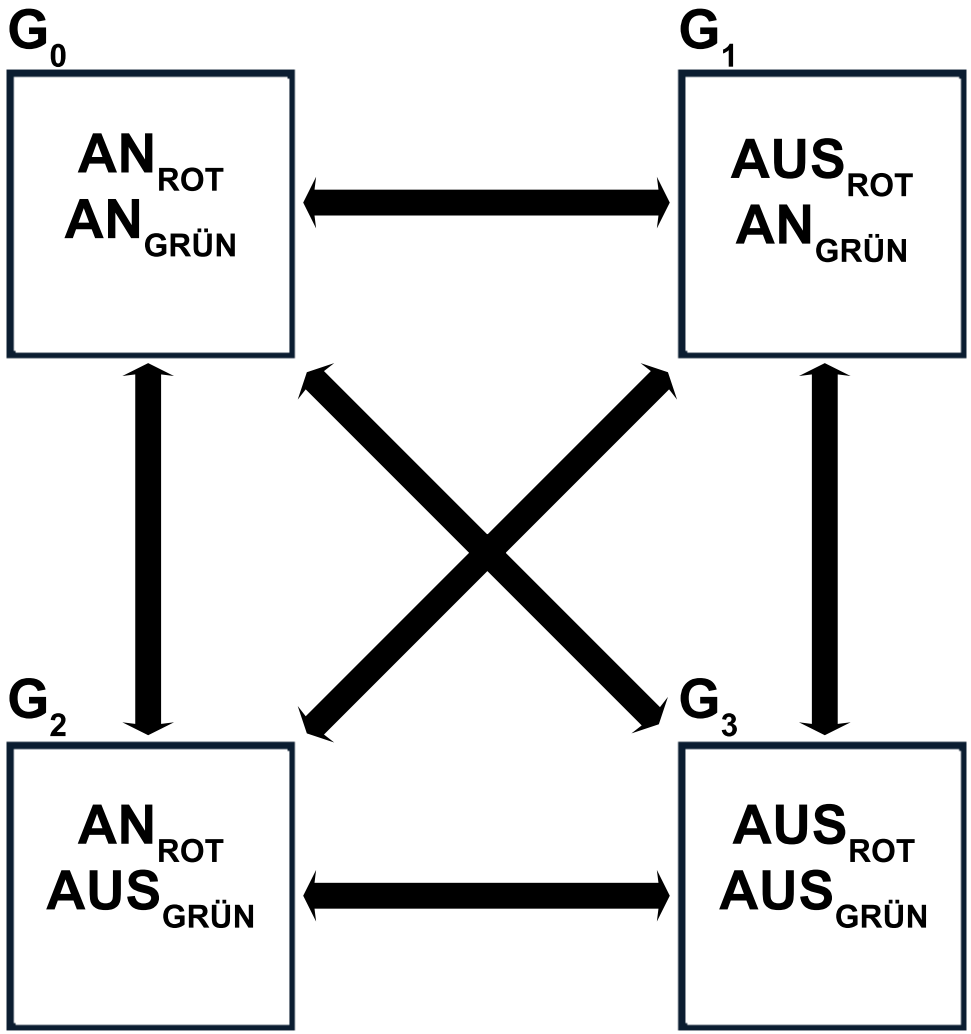
- Eine bestimmte durch Zustandsänderungen gebildete Folge von Zuständen bezeichnet man als Durchgang.
- Die nichtleere Menge aller Durchgänge über die Menge der globalen Zustände bilden dabei letztendlich das System.
- Ein „Punkt“ in diesem System lässt sich über einen Tupel (r,m) eindeutig identifizieren und bezeichnet Durchgang r zum Zeitpunkt m.

Beispiel: Fussgängerampel (Modell)

- Die rote und die grüne Lampe werden als eigenständige Agenten betrachtet, die entweder an oder aus sein können.

$$
\begin{aligned}
L_{ROT} &= \{AN, AUS\} \\
L_{GRÜN} &= \{AN, AUS\} \\
G &= L_{ROT} \times L_{GRÜN} \\
&= \{(AN_{ROT}, AN_{GRÜN}), (AUS_{ROT}, AN_{GRÜN}), \\
&\quad (AN_{ROT}, AUS_{GRÜN}), (AUS_{ROT}, AUS_{GRÜN})\}
\end{aligned}
$$

Beispiel: Fussgängerampel (Grafische Darstellung)



Fazit

- Mit der derzeitigen Definition von Multi-Agenten-Systemen, haben wir bereits ein recht gutes Werkzeug zur Modellierung der Agenten und ihrer Zustände zur Hand.

Fazit

- Mit der derzeitigen Definition von Multi-Agenten-Systemen, haben wir bereits ein recht gutes Werkzeug zur Modellierung der Agenten und ihrer Zustände zur Hand.
- Um wirkliche Probleme lösen und beschreiben zu können, fehlt jedoch noch eine Art Handlungsvorschrift für die Agenten.

Fazit

- Mit der derzeitigen Definition von Multi-Agenten-Systemen, haben wir bereits ein recht gutes Werkzeug zur Modellierung der Agenten und ihrer Zustände zur Hand.
- Um wirkliche Probleme lösen und beschreiben zu können, fehlt jedoch noch eine Art Handlungsvorschrift für die Agenten.
- Ausserdem ist unklar, wie die Zustandsübergänge überhaupt zu Stande kommen.

Fazit

- Mit der derzeitigen Definition von Multi-Agenten-Systemen, haben wir bereits ein recht gutes Werkzeug zur Modellierung der Agenten und ihrer Zustände zur Hand.
- Um wirkliche Probleme lösen und beschreiben zu können, fehlt jedoch noch eine Art Handlungsvorschrift für die Agenten.
- Ausserdem ist unklar, wie die Zustandsübergänge überhaupt zu Stande kommen.
- Daraus resultiert nur ein sehr eingeschränkter Nutzen des Modells.

Probleme und Ziele

- Gerade in der Informatik gibt es allerdings sehr viele Anwendungsfälle in denen viele Agenten in einem System miteinander kommunizieren (Netzwerke, Prozesse usw.).

Probleme und Ziele

- Gerade in der Informatik gibt es allerdings sehr viele Anwendungsfälle in denen viele Agenten in einem System miteinander kommunizieren (Netzwerke, Prozesse usw.).
- Daher ist der Bedarf nach einem Formalismus sehr groß, der es ermöglicht diese Systeme zu beschreiben und zu überprüfen.

Probleme und Ziele

- Gerade in der Informatik gibt es allerdings sehr viele Anwendungsfälle in denen viele Agenten in einem System miteinander kommunizieren (Netzwerke, Prozesse usw.).
- Daher ist der Bedarf nach einem Formalismus sehr groß, der es ermöglicht diese Systeme zu beschreiben und zu überprüfen.
- Um die angesprochenen praktische Probleme beschreiben zu können wollen wir auch das Verhalten der Agenten formalisieren und das Modell der Multi-Agenten-Systeme weiter konkretisieren.

Vogelperspektive

Beispiel: Coordinated Attack

- General A und B belagern mit ihrer Armee die Feinde im Tal.

Beispiel: Coordinated Attack

- General A und B belagern mit ihrer Armee die Feinde im Tal.
- Wenn sie alleine angreifen verlieren sie. Wenn sie gleichzeitig angreifen gewinnen sie.

Beispiel: Coordinated Attack

- General A und B belagern mit ihrer Armee die Feinde im Tal.
- Wenn sie alleine angreifen verlieren sie. Wenn sie gleichzeitig angreifen gewinnen sie.
- Sie können nur über Boten kommunizieren, die zwar aufgrund der relativ kurzen Distanz ohne Zeitverzögerung ankommen, aber vom Feind abgefangen werden können.

Beispiel: Coordinated Attack

- General A und B belagern mit ihrer Armee die Feinde im Tal.
- Wenn sie alleine angreifen verlieren sie. Wenn sie gleichzeitig angreifen gewinnen sie.
- Sie können nur über Boten kommunizieren, die zwar aufgrund der relativ kurzen Distanz ohne Zeitverzögerung ankommen, aber vom Feind abgefangen werden können.
- Ist es möglich, daß die beiden Generäle einen gemeinsamen Angriffszeitpunkt aushandeln? Wenn ja, wie? Wenn nein, warum?

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.
- L_e :=Menge der lokalen Zustände der Umgebung

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.
- L_e :=Menge der lokalen Zustände der Umgebung
- L_i :=Menge der lokalen Zustände des Agenten i

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.
- L_e :=Menge der lokalen Zustände der Umgebung
- L_i :=Menge der lokalen Zustände des Agenten i
- $G := L_e \times L_1 \times ... \times L_n$ (Menge der globalen Zustände)

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.
- L_e :=Menge der lokalen Zustände der Umgebung
- L_i :=Menge der lokalen Zustände des Agenten i
- $G := L_e \times L_1 \times ... \times L_n$ (Menge der globalen Zustände)
- Ein Durchgang ist eine Folge von globalen Zuständen.

Formelle Zusammenfassung: Multi-Agenten-Systeme

- Umgebung wird als Agent angesehen.
- L_e :=Menge der lokalen Zustände der Umgebung
- L_i :=Menge der lokalen Zustände des Agenten i
- $G := L_e \times L_1 \times ... \times L_n$ (Menge der globalen Zustände)
- Ein Durchgang ist eine Folge von globalen Zuständen.
- Das Paar (r ,m) heißt 'Punkt' in einem System und bezeichnet Durchgang r zum Zeitpunkt m.

Einführung

- Bisher wurde noch nicht betrachtet, woher die Durchgänge und Zustandsänderungen kommen.

Einführung

- Bisher wurde noch nicht betrachtet, woher die Durchgänge und Zustandsänderungen kommen.
- Wie man es inutitiv vermuten würde, sind sie ein Resultat von Aktionen, die von den Agenten oder der Umwelt ausgeführt werden.

Einführung

- Bisher wurde noch nicht betrachtet, woher die Durchgänge und Zustandsänderungen kommen.
- Wie man es inutitiv vermuten würde, sind sie ein Resultat von Aktionen, die von den Agenten oder der Umwelt ausgeführt werden.
- Dabei führen sie diese unabhängig voneinander aus. und halten sich an eine Art Protokoll.

Formalisierung

- Jeder Agent i besitzt dabei eine nichtleere Menge von Aktionen:

Formalisierung

- Jeder Agent i besitzt dabei eine nichtleere Menge von Aktionen:
- $ACT_i = \{a_1, ..., a_n\}$ mit $n > 0$

Formalisierung

- Jeder Agent i besitzt dabei eine nichtleere Menge von Aktionen:
- $ACT_i = \{a_1, ..., a_n\}$ mit $n > 0$
- Die Umwelt wird auch als Agent gesehen, und besitzt folgende nichtleere Aktionsmenge:

Formalisierung

- Jeder Agent i besitzt dabei eine nichtleere Menge von Aktionen:
- $ACT_i = \{a_1, ..., a_n\}$ mit $n > 0$
- Die Umwelt wird auch als Agent gesehen, und besitzt folgende nichtleere Aktionsmenge:
- $ACT_e = \{e_1, ..., e_n\}$ mit $n > 0$

Formalisierung

- Jeder Agent i besitzt dabei eine nichtleere Menge von Aktionen:
- $ACT_i = \{a_1, ..., a_n\}$ mit $n > 0$
- Die Umwelt wird auch als Agent gesehen, und besitzt folgende nichtleere Aktionsmenge:
- $ACT_e = \{e_1, ..., e_n\}$ mit $n > 0$
- In beiden Aktionsmengen wird eine spezielle Null-Aktion definiert, die keinen Effekt hat. Sie wird durch das Symbol 'Λ' ausgedrückt.

Probleme

- Zu wissen, welche Aktionen ausgeführt wurden, reicht in der Regel nicht aus um den globalen Zustand eines Systems zu bestimmen.

Probleme

- Zu wissen, welche Aktionen ausgeführt wurden, reicht in der Regel nicht aus um den globalen Zustand eines Systems zu bestimmen.
- Verschiedene Aktionen unterschiedlicher Agenten, die gleichzeitig ausgeführt werden, können miteinander interagieren.

Probleme

- Zu wissen, welche Aktionen ausgeführt wurden, reicht in der Regel nicht aus um den globalen Zustand eines Systems zu bestimmen.
- Verschiedene Aktionen unterschiedlicher Agenten, die gleichzeitig ausgeführt werden, können miteinander interagieren.
- Wenn zum Beispiel zwei Agenten gleichzeitig auf den gegenüberliegenden Seiten einer Tür ziehen, ist nicht eindeutig berechenbar was passieren wird, solange man die Berechnung nur in Isolation aus Sicht des Agenten durchführt.

Zusammengesetzte Aktionen

- Um mit interagierenden Agenten umgehen zu können, definieren wir sogenannte 'zusammengesetzte Aktionen'.

Zusammengesetzte Aktionen

- Um mit interagierenden Agenten umgehen zu können, definieren wir sogenannte 'zusammengesetzte Aktionen'.
- Eine 'zusammengesetzte Aktionen' ist ein Tupel der Form:

$$(a_e, a_1, ..., a_n)$$

wobei a_e eine Aktion der Umwelt und a_i eine Aktion von Agent i darstellt.

Zusammengesetzte Aktionen

- Jeder dieser zusammengesetzten Aktionen wird nun ein 'globaler Zustandstransformator' T zugewiesen, der eine Funktion darstellt, die globale Zustände auf andere gobale Zustände abbildet:

$$T : G \rightarrow G$$

und so eine Zustandsänderung ausführt.

Zusammengesetzte Aktionen

- Jeder dieser zusammengesetzten Aktionen wird nun ein 'globaler Zustandstransformator' T zugewiesen, der eine Funktion darstellt, die globale Zustände auf andere gobale Zustände abbildet:

$$T : G \rightarrow G$$

  und so eine Zustandsänderung ausführt.
- Das heisst: Wenn sich das System im Zustand s befindet und die Aktion $(a_e, a_1, ..., a_n)$ ausgeführt wird, befindet es sich anschließend im Zustand $T(s)$.

Zusammengesetzte Aktionen

- Um die zu den Aktionen gehörenden globalen Zustandstransformatoren zu erhalten, definieren wir zusätzlich eine sogenannte Transitionsfunktion τ, die eine zusammengesetzte Aktion auf ihren globalen Zustandstransformator abbildet.

$$\tau(a_e, a_1, ..., a_n)(s) = T(s)$$

Zusammengesetzte Aktionen

- Um die zu den Aktionen gehörenden globalen Zustandstransformatoren zu erhalten, definieren wir zusätzlich eine sogenannte Transitionsfunktion τ, die eine zusammengesetzte Aktion auf ihren globalen Zustandstransformator abbildet.

$$\tau(a_e, a_1, ..., a_n)(s) = T(s)$$

- Da nach unseren Definitionen verlangt wird, dass diese Formel für alle zusammengesetzten Aktionen $(a_e, a_1, ..., a_n)$ und alle globalen Zustände $s = (s_e, s_1, ..., s_n)$ definiert ist, in der Regel aber bestimmte Kombinationen nicht von Interesse sind oder nie auftauchen, können wir sie in diesen Fällen willkürlich definiert sein lassen.

Beispiel: Bit-Transmission Problem

- Erinnern wir uns an das Bit-Transmission Problem aus dem letzten Vortrag. Dort hat der Sender entweder ein Bit gesendet oder nichts getan:

Beispiel: Bit-Transmission Problem

- Erinnern wir uns an das Bit-Transmission Problem aus dem letzten Vortrag. Dort hat der Sender entweder ein Bit gesendet oder nichts getan:
- $ACT_S = \{sendbit, \Lambda\}$

Beispiel: Bit-Transmission Problem

- Erinnern wir uns an das Bit-Transmission Problem aus dem letzten Vortrag. Dort hat der Sender entweder ein Bit gesendet oder nichts getan:
- $ACT_S = \{sendbit, \Lambda\}$
- Analog verhielt sich der Empfänger in Bezug auf das Senden der Ack Message.

Beispiel: Bit-Transmission Problem

- Erinnern wir uns an das Bit-Transmission Problem aus dem letzten Vortrag. Dort hat der Sender entweder ein Bit gesendet oder nichts getan:
- $ACT_S = \{sendbit, \Lambda\}$
- Analog verhielt sich der Empfänger in Bezug auf das Senden der Ack Message.
- $ACT_R = \{sendack, \Lambda\}$

Beispiel: Bit-Transmission Problem

- Erinnern wir uns an das Bit-Transmission Problem aus dem letzten Vortrag. Dort hat der Sender entweder ein Bit gesendet oder nichts getan:
- $ACT_S = \{sendbit, \Lambda\}$
- Analog verhielt sich der Empfänger in Bezug auf das Senden der Ack Message.
- $ACT_R = \{sendack, \Lambda\}$
- Die Umgebung entscheidet nichtdeterministisch ob Nachrichten verloren gehen oder nicht und führt daher eine Aktion der Form (a, b) aus, wobei a entweder $delivers_S(current)$ oder Λ_S und b entweder $deliver_R(current)$ oder Λ_R ist.

Beispiel: Bit-Transmission Problem

- Die daraus resultierenden zusammengesetzten Aktionen kann man nun recht einfach ihren entsprechenden Zustandstransformationen zuordnen.

Beispiel: Bit-Transmission Problem

- Die daraus resultierenden zusammengesetzten Aktionen kann man nun recht einfach ihren entsprechenden Zustandstransformationen zuordnen.
- Das im Beispiel verwendete Modell ist nur eine willkürlich gewählte Variante. Man könnte das ganze natürlich auch anders modellieren.

Beispiel: Bit-Transmission Problem

- Die daraus resultierenden zusammengesetzten Aktionen kann man nun recht einfach ihren entsprechenden Zustandstransformationen zuordnen.
- Das im Beispiel verwendete Modell ist nur eine willkürlich gewählte Variante. Man könnte das ganze natürlich auch anders modellieren.
- Grundsätzlich versucht man aber immer ein Modell zu erstellen, welches so komplex wie nötig aber so einfach wie möglich ist.

Einführung

- Agenten wählen ihre auszuführenden Aktionen in der Regel anhand einer Art Handlungsvorschrift aus.

Einführung

- Agenten wählen ihre auszuführenden Aktionen in der Regel anhand einer Art Handlungsvorschrift aus.
- Dieses sogenannte Protokoll, stellt intuititv eine Beschreibung dar, welche Aktionen ein Agent, in Abhängigkeit seines lokalen Zustands, nutzen darf.

Einführung

- Agenten wählen ihre auszuführenden Aktionen in der Regel anhand einer Art Handlungsvorschrift aus.
- Dieses sogenannte Protokoll, stellt intuititv eine Beschreibung dar, welche Aktionen ein Agent, in Abhängigkeit seines lokalen Zustands, nutzen darf.
- Formell definiert man das Protokoll P_i für den Agenten i als Funktion von der Menge seiner lokalen Zustände L_i auf eine nichtleere Teilmenge seiner Aktionen ACT_i.

$$P_i : L_i \rightarrow SELACT_i \supseteq ACT_i$$

Protokoll

- Der Fakt, dass man auf eine Menge von Aktionen abbildet, deckt den möglichen Nichtdeterminismus des Protokolls ab.

Protokoll

- Der Fakt, dass man auf eine Menge von Aktionen abbildet, deckt den möglichen Nichtdeterminismus des Protokolls ab.
- Zwar wird zu jedem Zeitpunkt trotzdem nur eine Aktion ausgeführt, aber die Auswahl der Aktion erfolgt nichtdeterministisch.

Protokoll

- Der Fakt, dass man auf eine Menge von Aktionen abbildet, deckt den möglichen Nichtdeterminismus des Protokolls ab.
- Zwar wird zu jedem Zeitpunkt trotzdem nur eine Aktion ausgeführt, aber die Auswahl der Aktion erfolgt nichtdeterministisch.
- Ein Protokoll ist genau dann deterministisch wenn:

$$|P_i(s_i)| = 1$$

für alle Zustände $s_i \in L_i$.

Protokoll

- Analog zur Überlegung die Umgebung als Agent zu sehen, macht es Sinn für die Umgebung auch ein Protokoll zu definieren.

Protokoll

- Analog zur Überlegung die Umgebung als Agent zu sehen, macht es Sinn für die Umgebung auch ein Protokoll zu definieren.
- P_i ist bisher nur sehr allgemein definiert und wir schränken es daher auf die Menge der berechenbaren Funktionen ein. Wie man diese Berechenbarkeit formalisieren kann, würde hier aber den Rahmen sprengen.

Zusammengesetzte Protokolle

- Die Agenten führen ihre Protokolle nicht in Isolation aus, sondern bilden in Kombination mit den Protokollen der anderen Agenten die eigentliche Funktionalität des Systems.

Zusammengesetzte Protokolle

- Die Agenten führen ihre Protokolle nicht in Isolation aus, sondern bilden in Kombination mit den Protokollen der anderen Agenten die eigentliche Funktionalität des Systems.
- Daher definiert man ein zusammengesetztes Protokoll wie folgt:

$$P = (P_1, ..., P_n)$$

bestehend aus P_i, für jeden Agenten $i = 1..n$

Zusammengesetzte Protokolle

- Die Agenten führen ihre Protokolle nicht in Isolation aus, sondern bilden in Kombination mit den Protokollen der anderen Agenten die eigentliche Funktionalität des Systems.
- Daher definiert man ein zusammengesetztes Protokoll wie folgt:

$$P = (P_1, ..., P_n)$$

  bestehend aus P_i, für jeden Agenten $i = 1..n$
- Das Umgebungsprotokoll P_e wird bewusst nicht mit in die Definition aufgenommen, da das Umgebungsprotkoll oft nicht zur gewünschten Funktionalität beiträgt sondern diese eher stört und somit als Gegenspieler zum zusammengesetzten Protokoll gesehen werden kann.

Einführung

- Das zusammengesetzte Protokoll P und das Umgebungsprotokoll P_e beschreiben das Verhalten aller Teilnehmer des Systems.

Einführung

- Das zusammengesetzte Protokoll P und das Umgebungsprotokoll P_e beschreiben das Verhalten aller Teilnehmer des Systems.
- Daher eigenen sich diese Protokolle um das Verhalten eines kompletten Systems formell zu beschreiben bereits recht gut. Bei näherer Betrachtung zeigt sich aber, dass wir zur eindeutigen Beschreibung auch noch den Kontext kennen müssen in dem das Protokoll ausgeführt wird.

Einführung

- Das zusammengesetzte Protokoll P und das Umgebungsprotokoll P_e beschreiben das Verhalten aller Teilnehmer des Systems.
- Daher eigenen sich diese Protokolle um das Verhalten eines kompletten Systems formell zu beschreiben bereits recht gut. Bei näherer Betrachtung zeigt sich aber, dass wir zur eindeutigen Beschreibung auch noch den Kontext kennen müssen in dem das Protokoll ausgeführt wird.
- Der Kontext lässt sich durch folgenden Tupel beschreiben:

$$\gamma = (P_e, G_0, \tau, \Psi)$$

wobei P_e das Umgebungsprotokoll, G_0 die Menge der globalen Initialzustände, τ die Transitionsfunktion und Ψ eine Menge von Formeln darstellt, die zusätzliche Constraints für das System bereitstellen.

Interpretierte Kontexte

- Die von Ψ bereitgestellten Formeln sind dabei Formeln der Temporallogik.

Interpretierte Kontexte

- Die von Ψ bereitgestellten Formeln sind dabei Formeln der Temporallogik.
- Anmerkung: L_i und ACT_i sind implizit auch Teil des Kontexts. (über τ)

Interpretierte Kontexte

- Die von Ψ bereitgestellten Formeln sind dabei Formeln der Temporallogik.
- Anmerkung: L_i und ACT_i sind implizit auch Teil des Kontexts. (über τ)
- Der Kontext kann nun in Verbindung mit dem zusammengesetzten Protokoll dazu genutzt werden, das Verhalten eines Systems formal zu beschreiben.

Interpretierte Kontexte

- Die von Ψ bereitgestellten Formeln sind dabei Formeln der Temporallogik.
- Anmerkung: L_i und ACT_i sind implizit auch Teil des Kontexts. (über τ)
- Der Kontext kann nun in Verbindung mit dem zusammengesetzten Protokoll dazu genutzt werden, das Verhalten eines Systems formal zu beschreiben.
- In vielen Fällen hat man bei der Definition eines Kontexts bereits eine bestimmte Sammlung Φ von primitven Propositionen und eine entsprechende Interpretation π im Kopf. Dadurch kann man analog zu den interpretierten Systemen den Schritt zum interpretierten Kontext machen, den man als Tupel (γ, π) darstellt.

Interpretierte Kontexte

- In einem gut gewählten Kontext sind die einzelnen Komponenten orthogonal.

Interpretierte Kontexte

- In einem gut gewählten Kontext sind die einzelnen Komponenten orthogonal.
- Ein Durchgang r ist konsistent mit einem Protokoll P im Kontext γ wenn folgendes gilt:

Interpretierte Kontexte

- In einem gut gewählten Kontext sind die einzelnen Komponenten orthogonal.
- Ein Durchgang r ist konsistent mit einem Protokoll P im Kontext γ wenn folgendes gilt:
- $r(0) \in G_0$

Interpretierte Kontexte

- In einem gut gewählten Kontext sind die einzelnen Komponenten orthogonal.
- Ein Durchgang r ist konsistent mit einem Protokoll P im Kontext γ wenn folgendes gilt:
- $r(0) \in G_0$
- for all $m \geq 0$, if $r(m) = (s_e, s_1, ..., sn)$, dann existiert eine zusammengesetzte Aktion $(a_e, a_1, ..., an) \in P_e(S_e) \times P_1(S_1)...P_n(S_n)$, so dass $r(m+1) = \tau(a_e, a_1, ..., an)(r(m))$

Interpretierte Kontexte

- In einem gut gewählten Kontext sind die einzelnen Komponenten orthogonal.
- Ein Durchgang r ist konsistent mit einem Protokoll P im Kontext γ wenn folgendes gilt:
- $r(0) \in G_0$
- for all $m \geq 0$, if $r(m) = (s_e, s_1, ..., sn)$, dann existiert eine zusammengesetzte Aktion $(a_e, a_1, ..., an) \in P_e(S_e) \times P_1(S_1)...P_n(S_n)$, so dass $r(m+1) = \tau(a_e, a_1, ..., an)(r(m))$
- $r \in \Psi$

Interpretierte Kontexte

- Ein System R (bzw. ein interpretiertes System $I = (R, \pi)$) ist konstistent mit einem Protokoll P im Kontext γ (bzw. einem interpretierten Kontext (γ, π)) wenn jeder Durchgang $r \in R$ konsistent mit $P \in \gamma$ ist.

Interpretierte Kontexte

- Ein System R (bzw. ein interpretiertes System $I = (R, \pi)$) ist konstistent mit einem Protokoll P im Kontext γ (bzw. einem interpretierten Kontext (γ, π)) wenn jeder Durchgang $r \in R$ konsistent mit $P \in \gamma$ ist.
- Es gibt typischerweise viele Systeme, die mit einem Protokoll in einem Kontext konsistent sind.

Interpretierte Kontexte

- Ein System R (bzw. ein interpretiertes System $I = (R, \pi)$) ist konstistent mit einem Protokoll P im Kontext γ (bzw. einem interpretierten Kontext (γ, π)) wenn jeder Durchgang $r \in R$ konsistent mit $P \in \gamma$ ist.
- Es gibt typischerweise viele Systeme, die mit einem Protokoll in einem Kontext konsistent sind.
- In der Regel suchen wir aber das System, in dem alle möglichen Verhalten des Protokolls repräsentiert werden.

Interpretierte Kontexte

- Ein System R (bzw. ein interpretiertes System $I = (R, \pi)$) ist konstistent mit einem Protokoll P im Kontext γ (bzw. einem interpretierten Kontext (γ, π)) wenn jeder Durchgang $r \in R$ konsistent mit $P \in \gamma$ ist.
- Es gibt typischerweise viele Systeme, die mit einem Protokoll in einem Kontext konsistent sind.
- In der Regel suchen wir aber das System, in dem alle möglichen Verhalten des Protokolls repräsentiert werden.
- Ein solches System $(R^{rep}(P, \gamma))$ nennen wir Repräsentant von P. Analog nennt man das System $I^{rep}(P, \gamma, \pi) = (R^{rep}(P, \gamma), \pi)$ den interpretierten Repräsentanten von P im interpretierten Kontext (γ, π).

Fazit

- Man kann den Kontext als Beschreibung für eine Klasse von Systemen betrachten, an denen man interessiert ist.

Fazit

- Man kann den Kontext als Beschreibung für eine Klasse von Systemen betrachten, an denen man interessiert ist.
- Der Kontext beschreibt die Rahmenbedingungen in denen man ein Protokoll laufen lassen kann.

Fazit

- Man kann den Kontext als Beschreibung für eine Klasse von Systemen betrachten, an denen man interessiert ist.
- Der Kontext beschreibt die Rahmenbedingungen in denen man ein Protokoll laufen lassen kann.
- Durch das Ausführen unterschiedlicher Protokolle, kann man unterschiedliche Klassen von Systemen generieren, die alle die gleiche Charakteristik des unterliegenden Kontextes teilen.

Fazit

- Man kann den Kontext als Beschreibung für eine Klasse von Systemen betrachten, an denen man interessiert ist.
- Der Kontext beschreibt die Rahmenbedingungen in denen man ein Protokoll laufen lassen kann.
- Durch das Ausführen unterschiedlicher Protokolle, kann man unterschiedliche Klassen von Systemen generieren, die alle die gleiche Charakteristik des unterliegenden Kontextes teilen.
- Im folgenden Abschnitt werden wir ein Beispiel für eine Klasse von Systemen kennen lernen, die von einem entsprechenden Kontext definiert wird.

Informelle Betrachtung

Agreement

- Bevor wir mit der Analyse der Problemstellung beginnen sollten wir uns zunächst ein paar Gedanken über einige Grundbegriffe machen, denen wir höchstwahrscheinlich des öfteren begegnen werden.

Agreement

- Bevor wir mit der Analyse der Problemstellung beginnen sollten wir uns zunächst ein paar Gedanken über einige Grundbegriffe machen, denen wir höchstwahrscheinlich des öfteren begegnen werden.
- Da es bei dem angesprochenen Problem darum geht, dass beide Generäle einem gleichzeitigen Angriff zustimmen, ist zunächst zu klären was „Zustimmung“ überhaupt bedeutet.

Agreement

- Bevor wir mit der Analyse der Problemstellung beginnen sollten wir uns zunächst ein paar Gedanken über einige Grundbegriffe machen, denen wir höchstwahrscheinlich des öfteren begegnen werden.
- Da es bei dem angesprochenen Problem darum geht, dass beide Generäle einem gleichzeitigen Angriff zustimmen, ist zunächst zu klären was „Zustimmung“ überhaupt bedeutet.
- Der Einfachheit halber, heissen die Generäle ab sofort Alice und Bob.

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.
- *agree*(ψ) sei desweiteren eine Formel die immer dann wahr ist, wenn wenn Alice und Bob sich auf ψ geeinigt haben.

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.
- *agree*(ψ) sei desweiteren eine Formel die immer dann wahr ist, wenn wenn Alice und Bob sich auf ψ geeinigt haben.
- Es ist zu erwarten, dass sowohl Alice als auch Bob wissen, dass Sie sich geeinigt haben. (Jeder Teilnehmer der Abstimmung weiss, dass er zugestimmt hat.)

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.
- $agree(\psi)$ sei desweiteren eine Formel die immer dann wahr ist, wenn wenn Alice und Bob sich auf ψ geeinigt haben.
- Es ist zu erwarten, dass sowohl Alice als auch Bob wissen, dass Sie sich geeinigt haben. (Jeder Teilnehmer der Abstimmung weiss, dass er zugestimmt hat.)
- Daraus folgt: $agree(\psi) \Rightarrow E(agree(\psi))$ muss gelten.

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.
- $agree(\psi)$ sei desweiteren eine Formel die immer dann wahr ist, wenn wenn Alice und Bob sich auf ψ geeinigt haben.
- Es ist zu erwarten, dass sowohl Alice als auch Bob wissen, dass Sie sich geeinigt haben. (Jeder Teilnehmer der Abstimmung weiss, dass er zugestimmt hat.)
- Daraus folgt: $agree(\psi) \Rightarrow E(agree(\psi))$ muss gelten.
- Daraus folgt durch Induktionsregel für Common Knowledge: $agree(\psi) \Rightarrow C(agree(\psi))$ muss gelten.

Agreement

- Um die Betrachtungen noch weiter zu vereinfachen, repräsentieren wir das Statement, auf dass sich Alice und Bob einigen wollen durch die Formel ψ.
- $agree(\psi)$ sei desweiteren eine Formel die immer dann wahr ist, wenn wenn Alice und Bob sich auf ψ geeinigt haben.
- Es ist zu erwarten, dass sowohl Alice als auch Bob wissen, dass Sie sich geeinigt haben. (Jeder Teilnehmer der Abstimmung weiss, dass er zugestimmt hat.)
- Daraus folgt: $agree(\psi) \Rightarrow E(agree(\psi))$ muss gelten.
- Daraus folgt durch Induktionsregel für Common Knowledge: $agree(\psi) \Rightarrow C(agree(\psi))$ muss gelten.
- Daraus folgt: Zustimmung impliziert Common Knowledge.

Agreement

- Durch das Wissen um den Fakt, dass Zustimmung Common Knowledge benötigt, haben wir nun ein eindeutiges Werkzeug zur Hand um Probleme wie das Coordinated Attack Problem zu analysieren.

Agreement

- Durch das Wissen um den Fakt, dass Zustimmung Common Knowledge benötigt, haben wir nun ein eindeutiges Werkzeug zur Hand um Probleme wie das Coordinated Attack Problem zu analysieren.
- Wir definieren *delivered*, welches den Fakt repräsentiert, dass mindestens eine Nachricht uebermittelt wurde.

Agreement

- Durch das Wissen um den Fakt, dass Zustimmung Common Knowledge benötigt, haben wir nun ein eindeutiges Werkzeug zur Hand um Probleme wie das Coordinated Attack Problem zu analysieren.
- Wir definieren *delivered*, welches den Fakt repräsentiert, dass mindestens eine Nachricht uebermittelt wurde.
- Wenn Bob Alices Nachricht bekommt gilt trivialer Weise $K_b(\mathit{delivered})$. Nachdem Alice Bobs Bestätigung bekommt gilt $K_a K_b(\mathit{delivered})$. Nachdem Bob Alices Bestätigung auf seine Bestätigung erhalten hat gilt $K_b K_a K_b(\mathit{delivered})$ usw. ...

Agreement

- Durch das Wissen um den Fakt, dass Zustimmung Common Knowledge benötigt, haben wir nun ein eindeutiges Werkzeug zur Hand um Probleme wie das Coordinated Attack Problem zu analysieren.
- Wir definieren *delivered*, welches den Fakt repräsentiert, dass mindestens eine Nachricht uebermittelt wurde.
- Wenn Bob Alices Nachricht bekommt gilt trivialer Weise $K_b(\mathit{delivered})$. Nachdem Alice Bobs Bestätigung bekommt gilt $K_a K_b(\mathit{delivered})$. Nachdem Bob Alices Bestätigung auf seine Bestätigung erhalten hat gilt $K_b K_a K_b(\mathit{delivered})$ usw. ...
- Es stellt sich heraus, dass niemals Common Knowledge über die Nachrichtenübermittlung herrschen kann, egal wie oft man Nachrichten austauscht.

Agreement

- Es wird sich zeigen, dass in einem System mit „unsicherer“ Kommunikation generell niemals Common Knowledge über eine Nachrichtenübermittlung herrschen kann.

Agreement

- Es wird sich zeigen, dass in einem System mit „unsicherer“ Kommunikation generell niemals Common Knowledge über eine Nachrichtenübermittlung herrschen kann.
- Diese Erkenntnis stellt ein Problem für unsere Generäle dar, da ihre Koordination wie bereits Anfangs bemerkt, auf jeden Fall Common Knowledge benötigt.

Agreement

- Es wird sich zeigen, dass in einem System mit „unsicherer“ Kommunikation generell niemals Common Knowledge über eine Nachrichtenübermittlung herrschen kann.
- Diese Erkenntnis stellt ein Problem für unsere Generäle dar, da ihre Koordination wie bereits Anfangs bemerkt, auf jeden Fall Common Knowledge benötigt.
- Wir werden nun versuchen mit den Werkzeugen aus dem letzten Abschnitt zu beweisen, was wir jetzt durch qualifizierte Betrachtungen quasi informell hergeleitet haben (nämlich, dass es keine Lösung für das Coordinated Attack Problem in Systemen mit unsicherer Datenübertragung gibt).

Nachrichtenübermittlungskontext

- Der erste Schritt zum Beweisen unserer Behauptung liegt darin, eine Klasse von Kontexten zu definieren, in denen es Sinn macht über das Wissen der Agenten in Bezug auf Nachrichtenübermittlung zu sprechen.

Nachrichtenübermittlungskontext

- Der erste Schritt zum Beweisen unserer Behauptung liegt darin, eine Klasse von Kontexten zu definieren, in denen es Sinn macht über das Wissen der Agenten in Bezug auf Nachrichtenübermittlung zu sprechen.
- Wir machen dabei möglichst wenig Einschränkungen, um den Beweis so allgemeingültig zu halten wie möglich.

Nachrichtenübermittlungskontext

- Der erste Schritt zum Beweisen unserer Behauptung liegt darin, eine Klasse von Kontexten zu definieren, in denen es Sinn macht über das Wissen der Agenten in Bezug auf Nachrichtenübermittlung zu sprechen.
- Wir machen dabei möglichst wenig Einschränkungen, um den Beweis so allgemeingültig zu halten wie möglich.
- Daher schränken wir den Kontext nur dahingehend ein, dass ganz allgemein Nachrichtenübermittlungsaktionen stattfinden und die Umgebung alle Aktionen die ausgeführt werden aufzeichnet.

Nachrichtenübermittlungskontext

Formell nennen wir einen Kontext „Nachrichtenübermittlungskontext“ wenn er folgenden Anforderungen genügt:

- Die Umgebung und/oder einige der Agenten besitzen Aktionen die man als Nachrichtenübermittlungsaktionen bezeichnen kann (diese Aktionen resultieren im Ausliefern einer Nachricht an Agenten).

Nachrichtenübermittlungskontext

Formell nennen wir einen Kontext „Nachrichtenübermittlungskontext“ wenn er folgenden Anforderungen genügt:

- Die Umgebung und/oder einige der Agenten besitzen Aktionen die man als Nachrichtenübermittlungsaktionen bezeichnen kann (diese Aktionen resultieren im Ausliefern einer Nachricht an Agenten).
- γ ist ein aufzeichnender Kontext (der Zustand der Umgebung enthält alle zusammengesetzten Aktionen die bisher ausgeführt wurden und τ updated die Zustände entsprechend).

Nachrichtenübermittlungskontext

- Die Sprache enthält eine Proposition *delivered* mit der zuvor genannten Definition (da die Umgebung alle zusammengesetzten Aktionen enthält ist es einfach diesen Fakt ueber π zu erzwingen).

Nachrichtenübermittlungskontext

- Die Sprache enthält eine Proposition *delivered* mit der zuvor genannten Definition (da die Umgebung alle zusammengesetzten Aktionen enthält ist es einfach diesen Fakt ueber π zu erzwingen).
- Wir können diesen Kontext nun nutzen um eine Klasse von Systemen zu charakterisieren in denen wir über Nachrichtenübermittlung und das Wissen der Agenten darüber sprechen können.

Nachrichtenübermittlungskontext

- Die Sprache enthält eine Proposition *delivered* mit der zuvor genannten Definition (da die Umgebung alle zusammengesetzten Aktionen enthält ist es einfach diesen Fakt ueber π zu erzwingen).
- Wir können diesen Kontext nun nutzen um eine Klasse von Systemen zu charakterisieren in denen wir über Nachrichtenübermittlung und das Wissen der Agenten darüber sprechen können.
- Am Anfang eines Durchgangs ist *delivered* trivialerweise false, da zum Zeitpunkt 0 noch keine Nachrichten übermittelt wurden.

Unbounded Message Delivery

- Bevor wir mit dem eigentlichen formellen Beweis beginnen müssen wir zunächst definieren was mit einer „unsichere Nachrichtenübertragung gemeint ist. Intuitiv leuchtet es vielleicht ein, dass eine Nachrichtenübertragung immer dann als unsicher bezeichnet werden kann, wenn die Übermittlungszeit beliebig lange dauern kann oder die Nachrichten komplett verloren gehen (unendlich lange Verzögerung). Diese Art der Nachrichtenübermittlung heisst Unbounded Message Delivery (umd).

Unbounded Message Delivery

- Bevor wir mit dem eigentlichen formellen Beweis beginnen müssen wir zunächst definieren was mit einer „unsichere Nachrichtenübertragung gemeint ist. Intuitiv leuchtet es vielleicht ein, dass eine Nachrichtenübertragung immer dann als unsicher bezeichnet werden kann, wenn die Übermittlungszeit beliebig lange dauern kann oder die Nachrichten komplett verloren gehen (unendlich lange Verzögerung). Diese Art der Nachrichtenübermittlung heisst Unbounded Message Delivery (umd).
- Das Resultat ist, dass ein anderer Agent (außer der Empfänger) nur über weitere Nachrichten darüber informiert werden kann, dass der Empfänger eine Nachricht erhalten hat.

Unbounded Message Delivery

- Im Detail bedeutet das: Wenn das System R an „umd“ leidet, Agent i am Punkt (r, l) in R eine Nachricht erhält und kein weiterer Agent in Durchgang r zwischen Zeitpunkt l und m eine Nachricht von i erhält, dann halten alle anderen Agenten es für möglich, dass i noch garkeine Nachricht bekommen hat.

Unbounded Message Delivery

- Formalisiert heisst es wie folgt: R sei ein System, sodass für ein sorgfältig gewähltes π, das interpretierte System $I = (R, \pi)$ ein Nachrichtenübermittlungssystem ist. Gegeben Sei ein Durchgang $r \in R$; wir schreiben $d(r, m) = k$ wenn genau k Nachrichten in den ersten m Runden von r übermittelt wurden. $d(r, 0) = 0$ ist trivialerweise klar. Ein solches System R zeigt die Eigenschaften von „umd“ wenn wenn für alle Punkte (r, m) in R mit $d(r, m) > 0$ ein Agent i und ein Durchgang $r' \in R$ existiert, sodass für alle Agenten $j \neq i$ und Zeitpunkt $m' \leq m$ gilt, dass $r'_j(m') = r_j(m')$ und $d(r', m) < d(r, m)$.

Unbounded Message Delivery

- Mithilfe dieser Aussage und dem Wissen, dass unser Coordinated Attack Problem an umd leidet können wir nun zeigen, dass Common Knowledge über die Nachrichtenübermittlung in solch einem System nicht erworben werden kann und folgendes Theorem gilt:

Unbounded Message Delivery

- Mithilfe dieser Aussage und dem Wissen, dass unser Coordinated Attack Problem an umd leidet können wir nun zeigen, dass Common Knowledge über die Nachrichtenübermittlung in solch einem System nicht erworben werden kann und folgendes Theorem gilt:
- $I \models \neg C_G(\mathit{delivered})$

Unbounded Message Delivery

- Mithilfe dieser Aussage und dem Wissen, dass unser Coordinated Attack Problem an umd leidet können wir nun zeigen, dass Common Knowledge über die Nachrichtenübermittlung in solch einem System nicht erworben werden kann und folgendes Theorem gilt:
- $I \models \neg C_G(\mathit{delivered})$
- Für jeden Punkt (r, m) in I beweisen wir, dass $(I, r, m) \models \neg C_G(\mathit{delivered})$ durch Induktion über $d(r, m)$:

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.
- Sei $j \neq i$ ein Agent in G.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.
- Sei $j \neq i$ ein Agent in G.
- Die „umd“-Bedingung garantiert, dass $d(r', m) < d(r, m)$.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.
- Sei $j \neq i$ ein Agent in G.
- Die „umd“-Bedingung garantiert, dass $d(r', m) < d(r, m)$.
- Durch die Induktionsannahme haben wir $(I, r, m) \models \neg C_G(\mathit{delivered})$.

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.
- Sei $j \neq i$ ein Agent in G.
- Die „umd“-Bedingung garantiert, dass $d(r', m) < d(r, m)$.
- Durch die Induktionsannahme haben wir $(I, r, m) \models \neg C_G(\mathit{delivered})$.
- Da $r'_j(m) = r_j(m)$ ist (r', m) G-Erreichbar von (r, m).

Beweis

- Der Fall $d(r, m) = 0$ ist trivialerweise klar.
- Sei $d(r, m) = k + 1$ und nehmen wir an das Ziel ist wahr für alle Punkte (r', m') mit $d(r', m') \leq k$.
- Seien r' und i der Durchgang und der Agent entsprechend existent durch die „umd“-Bedingung.
- Sei $j \neq i$ ein Agent in G.
- Die „umd“-Bedingung garantiert, dass $d(r', m) < d(r, m)$.
- Durch die Induktionsannahme haben wir $(I, r, m) \models \neg C_G(\mathit{delivered})$.
- Da $r'_j(m) = r_j(m)$ ist (r', m) G-Erreichbar von (r, m).
- Daraus folgt: $(I, r, m) \models \neg C_G(\mathit{delivered})$.

Folgerung

- Daraus kann man folgern, dass in solchen Systemen tatsächlich kein Common Knowledge durch Kommunikation entstehen kann, woraus resultiert, dass es keine Lösung fuer das Coordinated Attack Problem gibt.