technische universität
dortmund

Bachelorarbeit

# Konzeption eines verteilten, multiagentenbasierten Steuerungssystems für Materialflusssysteme

Christian Hoppe
Oktober 2016

Gutachter:
Prof. Dr.-Ing. Markus Rabe
Prof. Dr. Heiko Krumm

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl IV: Rechnernetze und verteilte Systeme
http://ls4-www.cs.tu-dortmund.de

In Kooperation mit:
Fakultät für Maschinenbau
Fachgebiet IT in Produktion und Logistik

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Aktuelle Forschungstrends im Umfeld des *Internet der Dinge* und der *cyber-physischen Systeme* zielen darauf ab, steuernde Systeme in die zu steuernden Komponenten von Produktion und Logistik einzubetten. Dies ist eine Voraussetzung für eine Erhöhung des Automatisierungsgrads sowie weiterer Anwendungen im Rahmen der *Industrie 4.0*. Für Materialflusssysteme übernehmen jedoch klassischerweise dedizierte Materialflussrechner Steuerung und Überwachung sowie bei Stetigfördersystemen insbesondere auch das Routing der Transportgüter durch das System. Durch die Verlagerung der Steuerungslogik in das Stetigfördersystem selbst ließe sich auf diesen Materialflussrechner zu Gunsten einer Selbstorganisation und -steuerung des Stetigfördersystems verzichten. Damit wäre eine Grundvoraussetzung für eine bessere Integration von Stetigfördersystemen in den *Industrie 4.0*-Kontext von Produktion und Logistik gegeben. [BDD+13]

Im Rahmen der Dissertationsschrift *Verteilte Simulation und Emulation von Materialflusssystemen mit dezentraler Steuerung* von Herrn Dr. Damian Daniluk [Dan14] wurde gezeigt, dass bei bestimmten Klassen von Simulationsmodellen eine signifikante Beschleunigung bei der Simulation und Emulation von Materialflusssystemen mit dezentraler Steuerung möglich ist. Demnach kann ein Materialflusssystem als System einzelner Materialflussmodule (MFM) aufgefasst werden, welche zur Realisierung eines Routings des Transportguts durch Agenten repräsentiert werden.

Die wesentliche Forschungsfrage der Dissertationsschrift lag in der Ermittlung und Bewertung des mittels verteilter Simulation erzielbaren Geschwindigkeitsvorteils gegenüber der Ausführung solcher Materialflussmodelle auf einem Rechner. Zur Beantwortung dieser Forschungsfrage war aufgrund der großen Anzahl an Einflussgrößen eine Evaluation anhand einer empirischen Studie erforderlich. Dazu wurden konkrete dezentrale Steuerungsverfahren konzipiert und implementiert, die auf dem Multiagentenparadigma basieren. Diese waren so gewählt, dass sie in den empirischen Untersuchungen Schlussfolgerungen über ein möglichst breites Spektrum der denkbaren dezentralen Steuerungsverfahren ermöglichen und sind in diesem Sinne als Referenzverfahren zu verstehen. Daher hat der Author der vorliegenden Arbeit im Rahmen unterstützender Tätigkeiten am *Fraunhofer Institut für Materialfluss und Logistik (IML)* für die genannte Dissertationsschrift ein dezentrales, ver-

teilbares Steuerungssystem in agentenbasierter Form auf Basis des FIPA-konformen *Java Agent DEvelopment Framework (JADE)* implementiert.

Das entstandene Steuerungssystem ist in der Lage, analog zur Steuerungslogik im Simulator, mit Hilfe von Routingverfahren für paketvermittelnde Netze als Referenzverfahren, dort exemplarisch Link-State-Routing und Dynamic-Source-Routing, Transportgut durch prinzipiell beliebige Stetigfördersysteme zu leiten.

Dieses Steuerungssystem wurde im Rahmen von [Dan14] im Wesentlichen auf diejenigen Eigenschaften hin analysiert, die relevant für den im Rahmen der verteilten Simulation oder Emulation erzielbaren Speedup sind. Konzeption und Implementierung des Steuerungssystems als solches werden im Rahmen der Dissertation nicht detailliert beleuchtet.

Ein mögliches Verfahren zur Analyse verteilter Algorithmen ist die Analyse auf Safety und Liveness. Safety-Eigenschaften drücken dabei invariante Aussagen (informell Argumente der Art *keine schlechten Dinge geschehen*), Liveness-Eigenschaften hingegen die Wohlfundiertheit (informell Argumente der Art *gute Dinge werden (irgendwann) geschehen*) aus. [AS85, AS87] Motiviert durch die wachsende Anzahl autonom agierender Systeme wurden multiagentenbasierte Systeme bereits in verschiedenen Domänen mit Bezügen zur Logistik einer Safety-/Liveness-Analyse unterzogen. Dies reicht bis zu hoch komplexen Systemen, etwa mit dem Ziel, Sicherheitsgarantien über die Steuerung unbemannter Fluggeräte zu erhalten [CHF+16]. DSR als Referenzverfahren zum Routing spielt auch in mobilen Ad-Hoc-Netzwerken eine Rolle und ist in diesem Anwendungsfeld bereits mehrfach, u.A. auch sowohl Safety- als auch Liveness-Analysen unterzogen worden, wobei auch jene Eigenschaften nachgewiesen werden konnten, welche für das Routing in Materialflusssystemen relevant sind. [YZW06] Dabei steht allerdings auch die Beweglichkeit der Knoten im Vordergrund, die für starre Systeme wie Stetigfördersysteme keine Rolle spielt und sich verkomplizierend auf die Analyse auswirkt. Domänenübergreifend versucht die *Gaia*-Methode für agentenorienterten Entwurf und Analyse Safety- und Livenesseigenschaften über ein Rollen- und Rechtekonzept sicher zu stellen [WJK00], bietet aber bislang keine formale Semantik mit welcher diese Eigenschaften nachgewiesen werden können. In der Domäne der multiagentenbasierten Steuerungssysteme für Materialflusssysteme wurde bereits eine Überprüfung mittles model checking im Allgemeinen [TKSS13] und auf Safety- und Liveness [DFR08] als Ergänzung zu simulativen Techniken vorgeschlagen, jedoch nicht an konkreten Systemen oder Konzepten durchgeführt.

Die vorliegende Arbeit zielt darauf ab, diese Lücke zu schließen und die simulativen Studien in [Dan14] zu ergänzen. Die im Gegensatz zu einer experimentellen Untersuchung bislang scheinbar nicht berücksichtigte Analyse des asymptotischen Kommunikationsauf-

wands des Routings mittels DSR soll darüber hinaus ein potentielles Ausschlusskriterium behandeln. Ziel dieser Arbeit ist daher, zu zeigen, dass Stetigfördersysteme als Vertreter der Materialflusssysteme vorteilhaft mittels verteilter Agenten gesteuert werden können und insbesondere auch, dass sich dabei Routingverfahren mittels verteilter Agenten implementieren lassen. Es wird ein hypothetisches System betrachtet, in welchem die einzelnen Agenten von den jeweils zu steuernden Komponenten mittels eingebetteter Systeme selbst ausgeführt werden und kein dedizierter Materialflussrechner zum Einsatz kommt. Als zu untersuchendes Routingverfahren dient Dynamic-Source-Routing (DSR) aufgrund seiner kommunikationslastigen Verfahrensweise als zu untersuchendes Referenzverfahren.

Nach Erläuterung benötigter Grundbegriffe und Konzepte erfolgt der Eignungsnachweis anhand eines dazu formalisierten, funktionalen Modells auf Basis von Zustandstransitionssystemen (State-Transition-System, kurz: STS). Ein solches Modell bildet die Zustände des Systems und die Übergänge zwischen diesen Zuständen ab. Anhand einer Analyse der Übergänge in mögliche Folgezuständen ausgehend von einem momentanen Systemzustand lassen sich konkrete Aussagen bezüglich der Eigenschaften des Verfahrens treffen. Die sich an die Modellierung anschließende Safety-Analyse des Modells zeigt in diesem Kontext einerseits die Gültigkeit der gefundenen Routen. Eine folgende Liveness-Analyse untersucht die Reaktivität des Systems. In verteilten Systemen kommt im Gegensatz zu zentralisierten Systemen immer auch ein Kommunikationsaufwand hinzu, weshalb eine Analyse dessen Wachstums mit steigender Systemgröße erfolgt. Diese Analysen dienen der Bewertung des Steuerungssystems im Hinblick auf seine Eignung und werden abschließend vor diesem Hintergrund ausgewertet.

# Kapitel 2

# Verteilte Materialflusssteuerung

Für die Konzeption eines multiagentenbasierten Steuerungssystems für Materialflusssysteme müssen zuvor die konkrete Anwendungsdomäne sowie Anforderungen und Rahmenbedingungen geklärt werden. Gerade die Abgrenzung zwischen zentraler Steuerung durch einen dedizierten Materialflussrechner und dezentraler Steuerung durch eingebettete Agenten bedarf einer genauen Definition der Begrifflichkeiten.

## 2.1 Grundlagen der Materialflusssteuerung

Unter dem Begriff *Materialfluss* wird in der Logistik die Bewegung von physischen Gütern verstanden. Im Gegensatz zur *Lieferkette* wird damit lediglich Bewegung innerhalb festgesetzter Produktionsbereiche beschrieben. Dadurch wird der Materialflussbegriff zu einem Begriff der Intralogistik. Genauer nach [VDI73]: *Materialfluss ist die Verkettung aller Vorgänge beim Gewinnen, Be- und Verarbeiten, sowie bei der Verteilung von stofflichen Gütern innerhalb festgelegter Bereiche.* Im Folgenden werden darauf aufbauend die Grundbegriffe der Materialflusssteuerung erörtert.

### 2.1.1 Materialfluss, Materialflusssysteme und Materialflussrechner

Ein *Materialflusssystem (MFS)* stellt jenes intralogistische System dar, welches diese Güterbewegung leistet. Das Ziel eines MFS ist es, die geforderten Güter zum jeweils gewünschten Zeitpunkt und in der gewünschten Menge zum Ort des Bedarfs zu transportieren.

Die Sicherstellung dieser Anforderungen ist Aufgabe der *Materialflussrechnung* [Gro84], wobei diese in modernen Anlagen automatisiert durch einen *Materialflussrechner (MFR)* als Teil der Materialflusssteuerung geleistet wird.

Bei MFS kann grundsätzlich zwischen Stetig- und Unstetigförderern unterschieden werden. Ein klassisches Beispiel für Stetigförderer sind etwa Rollenbahnen, für Unstetigförderer hingegen Flurförderfahrzeuge wie etwa Gabelstapler. Dabei sind komplexe Stetigfördersysteme in praktisch jedem Warenverteilzentrum zu finden, wo sie den Transport der Güter

von Wareneingängen zu Warenausgängen, häufig mit einem dazwischen liegenden Lager, übernehmen. Ein weiteres alltägliches Beispiel ist der Gepäcktransport an Flughäfen, wo das Gepäck üblicherweise durch ein Stetigfördersystem von den verschiedenen Gepäckaufgabestationen zum jeweiligen Gate transportiert wird.

Das Routing der Güter ist also beim Transport von Stückgut durch derartige Stetigfördersysteme eine zentrale Aufgabe des Materialflussrechners. Aus diesem Grund werden sie im Folgenden als Repräsentanten für MFS gewählt. Als weitere Einschränkung wird auf die Betrachtung von Schüttgut verzichtet, da hier keine abgeschlossenen, atomaren Einheiten als Güter betrachtet werden, weshalb hier statt Routing eher Last- bzw. Massenverteilung stattfindet.

### 2.1.2 Zentrale vs. dezentrale Steuerung

Die Steuerung eines MFS ist, wie beschrieben, Aufgabe des Materialflussrechners. Dieser ist klassischerweise ein dediziertes Rechnersystem und als solches nicht Teil des MFS selbst. Die Steuerung in klassischen, zentralen Materialflusssteuerungen ist dabei nach [Büc00] hierarchisch organisiert und besteht aus

- einem Materialflussrechner, welcher höhere Aufgaben wie Auftragsdisposition, Routenplanung, etc. übernimmt, sowie
- speicherprogrammierbaren Steuerungen (SPS), welche die Steuerung und Überwachung der elektrischen Komponenten übernimmt.

Vor diesem Hintergrund kann klar zwischen konkreter Anlagensteuerung und Materialflussrechnung unterschieden werden. Im Rahmen dieser Arbeit bezieht sich die dezentrale Steuerung auf die Materialflussebene. Da das Steuerungssystem im Rahmen von [Dan14] zunächst mit dem Ziel der Emulation entstanden ist, fokussiert es sich auf die Routenplanung. Die Anlagensteuerung ist für konkrete Anlagen spezifisch und wird als solche im Rahmen der Ausarbeitung des Konzepts nicht detailliert beleuchtet werden. Ideen und Ansätze zur verteilten Anlagensteuerung finden sich beispielhaft u.A. auch in [Gün12].

## 2.2 Materialflussmodule und ihre Abbildung auf Agenten

Das in dieser Arbeit beschriebene Steuerungssystem baut, gemäß seines Ursprungs im Rahmen von [Dan14], auf dem auch dort durch den Begriff *Materialflussmodul (MFM)*

beschriebenen Konzept der Auffassung eines Materialflusssystems als Menge von Modulen auf. Diese dienen analog dazu auch in der vorliegenden Arbeit der Definition und Beschreibung eines Agenten und werden im Folgenden erläutert.

### 2.2.1 Elementtypen in Materialflussystemen

Nach [Gro84] kann für die Materialflussrechnung jedes MFS durch die verschiedenen Elementtypen *Objekte, Quellen, Senken, Verbindungselemente, Stationen und Speicher* beschrieben werden. Daran angelehnt wird die folgende Teilmenge dieser Elementtypen verwendet:

- *Objekte* - Güter und Transport(-hilfs)mittel
- *Quellen* - Eingangsschnittstellen, an denen Objekte in das MFS gelangen
- *Senken* - Ausgangsschnittstellen, an denen Objekte das MFS verlassen
- *Verbindungen* - Transportabschnitte, über die Objekte von Quellen zu Senken gelangen
- *Stationen* - Stellen im MFS, an denen sich Objekte unverändert für eine bestimmte Zeit ohne Bewegung aufhalten können

Diese dienen im Folgenden als Basisterminologie für die Beschreibung der einzelnen funktionalen Komponenten eines MFS.

### 2.2.2 Materialflussmodule

Ein MFM kann informell als *Fördertechnikabschnitt*, also als Teilabschnitt eines Stetigförderers verstanden werden. Diese Module bilden demnach auch in dieser Arbeit die *Infrastrukturelemente von Stetigfördersystemen* als *logistische Objekte des Materialflusssystems* [Dan14].

Folglich gelten für ein MFM auch in dieser Arbeit die dort beschriebenen Eigenschaften:

- *Ein MFM enthält eine eigene Steuerung, die unabhängig von den Steuerungen anderer MFM ist.*

- *Ein MFM kann mit den an den Eingängen und Ausgängen angeschlossenen Nachbar-MFM kommunizieren.*
- *Befährt ein Lastobjekt das MFM, so ist das MFM in der Lage, die Zielinformation des Lastobjektes auszulesen.*

Dabei wird mit einem *Lastobjekt* ein einzelnes, zu transportierendes Transportgut beschrieben.

Für eine formale Beschreibung eines MFM ist die zuvor beschriebene Klassifikation von Elementtypen nach [Gro84] hilfreich. Da im Rahmen dieser Arbeit der Fokus auf der Routenfindung im Steuerungssystem liegt, kann diese jedoch vereinfacht werden. Es wird deshalb folgende Definition der Elementtypen verwendet:

- Eine Quelle $S_Q$ beschreibt eine Stelle eines Stetigförderers, an welcher Transportgüter in diesen eingebracht werden.
- Eine Senke $S_S$ beschreibt eine Stelle eines Stetigförderers, an welcher Transportgüter aus diesem ausgebracht werden.
- Eine Station $S_T$ beschreibt eine Stelle eines Stetigförderers, welche dem Weitertransport eines Transportguts dient.

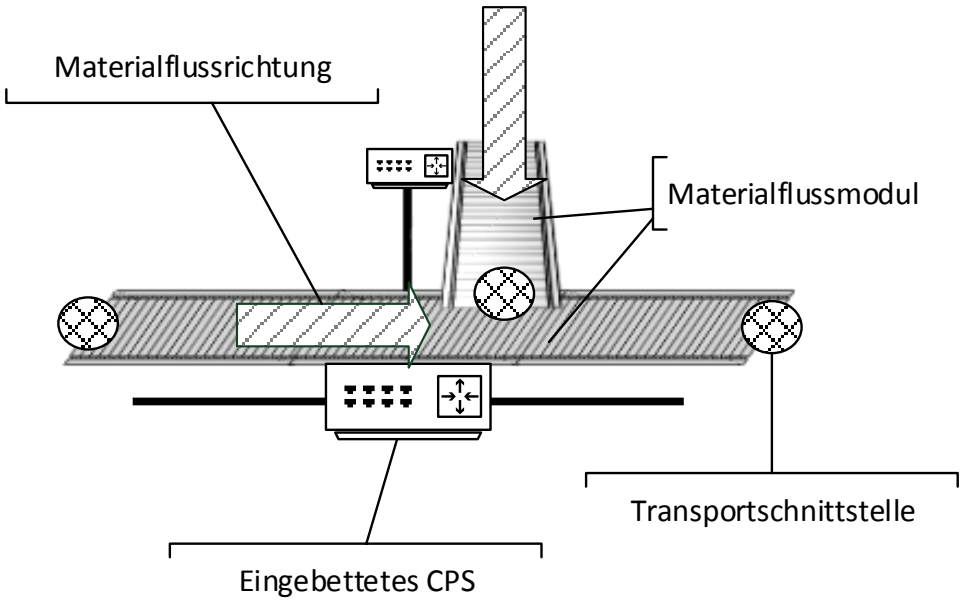
Im Besonderen bilden *Stationen* die Schnittstellen der MFM ab. Sie markieren folglich den Übergang zwischen den MFM.

Ein MFM kann also als beliebig lange Verkettung von Elementen aus $\{S_Q, S_S, S_T\}$ aufgefasst werden, wobei gilt:

- Am Anfang eines MFM befindet sich mindestens ein Element vom Typ $S_Q$ oder $S_T$
- Am Ende eines MFM befindet sich mindestens ein Element vom Typ $S_T$ oder $S_S$
- Ein Element vom Typ $S_T$ befindet sich genau dann an einer Position eines MFM, wenn dieses MFM an genau dieser Position eine durch ein Transportgut befahrbare eingehende oder ausgehende Schnittstelle zu einem benachbarten MFM besitzt.

Eine konkrete Anordnung dieser Elemente innerhalb des MFM wird im Folgenden als *innere Topologie* des MFM bezeichnet. Die Abbildung 2.1 zeigt zwei aneinander gefügte Materi-

**Abbildung 2.1:** Aufbau von Materialflussmodulen




alflussmodule, wobei das MFM im Vordergrund drei Transportschnittstellen zu anderen MFM aufweist.

### 2.2.3 Multiagentensysteme

Eine häufig verwendete Definition von Softwareagenten liefert [FG96]: *An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.* Diese Definition ist eine vielfach u.A. auch in [Dan14] genutzte und wird auch in dieser Arbeit als Referenz für den Agentenbegriff gebraucht werden. Dabei wird ein Agent hier jedoch losgelöst von konkreten, physischen Sensoren und Aktoren betrachtet und die Kontrolle dieser einer unterlagerten Anlagensteuerung überlassen (2.1.2). Die Agenten des in dieser Arbeit beschriebenen Steuerungssystems konzentrieren sich stattdessen auf die Routenfindung durch Kooperation und Kommunikation. Sie arbeiten dazu reaktiv, also nur durch spezifische Ereignisse zum Handeln aufgefordert und ohne Eigeninitiative, wobei eine Routinganfrage hier als Startereignis dient.

Ein System aus mehreren, miteinander interagierenden Agenten, welche eine Aufgabe lösen, wird Multiagentensystem genannt. Nach [Wei99]: *... multiagent systems, that is, systems*

*in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks.*

Für das vorliegende Steuerungssystem bedeutet dies, dass es sich aus mehreren homogenen Agenten zusammensetzt, welche gemeinsam die Routenfindung durch das gesteuerte Materialflusssystem leisten.

### 2.2.4 Abbildung der Materialflussmodule auf Agenten

Das vorliegende Steuerungssystemkonzept bildet Agenten auf Materialflussmodule ab. Dabei kann ein Materialflussmodul grundsätzlich mindestens eine, jedoch durchaus auch mehrere Transportschnittstellen aufweisen.

Anhand der Richtung des Fördergutstroms im Stetigförderer, lassen sich für die Transportschnittstellen vier Typen unterscheiden, um Grenzen von Fördertechnikabschnitten zu beschreiben:

- Einfache Verbindungen, bei denen ein Fördertechnikabschnitt mit einem Folgefördertechnikabschnitt verbunden ist.
- Verzweigungen, an welchen sich ein Eingangsfördergutstrom auf mehrere Ausgangsfördergutströme verteilt
- Zusammenführungen, an welchen mehrere Eingangsfördergutströme zu einem Ausgangsfördergutstrom zusammgefasst werden
- Kreuzungen, an welchen mehrere Eingangsfördergutströme auf mehrere Ausgangsfördergutströme verteilt werden

Eine Station $S_T$ wird dabei an genau solchen Stellen im Stetigfördersystem gefordert, an welchen eine Verzweigung, Zusammenführung oder Kreuzung vorliegt. Ferner gilt die Voraussetzung, dass für jedes Element vom Typ $S_T$ ein komplementäres Element vom Typ $S_T$ für jedes Nachbar-MFM vorliegt, so dass sich ein Stationenpaar ergibt, welches die Transportschnittstelle ausgangs- und eingangsseitig markiert.

Das vorliegende Steuerungssystem ordnet jedem einzelnen MFM einen eigenen dedizierten Agenten zu, welcher mittels eingebetteter Cyber-Physischen-Systeme durch das MFM selbst ausgeführt wird. Entsprechend repräsentiert dieser Agent auch alle Transportschnittstellen seines MFM. Über die Paarung der Stationen an den Transportschnittstellen sind

alle Nachbarn des Agenten eindeutig festgelegt. Ferner ist dem Agenten hieraus auch ersichtlich, an welchen Positionen sein MFM den Fördergutstrom erhält, oder weitertransportieren kann. Ein Agent kennt folglich die innere Topologie des MFM und kann diese für Routingentscheidungen und Kostenberechnungen, basierend auf den im MFM zurückzulegenden Wegstrecken, nutzen.

Die Gesamtheit der Materialflussmodule lässt sich nach [Dan14] durch einen Materialflussgraphen beschreiben. Dieser zeichnet sich dadurch aus, dass er den Fördergutstrom durch den Stetigförderer beschreibt und aufgrund der angenommenen Unidirektionalität von Stetigfördersystemen sowohl innerhalb der MFM als auch zwischen den MFM grundsätzlich gerichtet ist. Er enthält die MFM als Knoten und die Transportschnittstellen zwischen den MFM als, der Materialflussrichtung folgend gerichteten, Kanten.

Im Gegensatz dazu stellt sich der Kommunikationsgraph als ungerichteter Graph dar, der zwar wiederum die MFM als Knoten enthält, aber aufgrund der bidirektionalen Kommunikationverbindung zwischen den, den einzelnen MFM zugeordneten, Cyber-Physischen Systemen die gleichen Kanten wie der Materialflussgraph in ungerichteter Form enthält. Materialfluss- und Kommunikationsgraph sind grundsätzlich topologisch identisch.

Da die verschiedenen Graphen zur Beschreibung der Materialfluss- und Kommunikationsnetztopologie hier lediglich der Verdeutlichung des prinzipiellen Systemaufbaus dienen, sollen sie zwar nicht unerwähnt bleiben jedoch nicht weiter ausgeführt werden. Eine genaue Ausführung der verschiedenen Graphen und ihrer Zusammenhänge findet sich in [Dan14].

## 2.3 Dynamic Source Routing

Dynamic Source Routing (DSR) ist in [Joh94] sowie in [JM96] erstmalig beschrieben und stellt prinzipiell ein Routingverfahren für Datenpakete in drahtlosen Ad-hoc Netzwerken dar. Als solches zeichnet es sich durch seinen Verzicht auf statische Routingtabellen aus, welche andernfalls von den einzelnen Knoten eines Netzes vorgehalten werden müssten.

### 2.3.1 Routingverfahren

Beim Routing mittels DSR wird die Route vollständig vorab bestimmt. Da während der Durchführung der Route Störungen auftreten können, welche eine Änderung erforderlich machen könnten, muss diese währenddessen überwacht werden. Entsprechend lässt sich DSR grob in zwei Phasen unterteilen:

- *Route Discovery*: Die Bestimmung einer Route durch das Netz.
- *Route Maintenance*: Die Überwachung einer Route, während ihrer Durchführung.

Bei der *Route Discovery* wird dabei zunächst eine Anfrage vom Ausgangspunkt an alle benachbarten Knoten des Netzes versendet. Diese Routinganfrage enthält eine eindeutige Kennung (*Request ID*), das gewünschte Ziel und eine Liste aller Knoten (*Route Record*), welche diese Nachricht besucht hat und welche mit dem Startknoten initialisiert wird. Erhält ein Knoten eine solche Nachricht, lassen sich folgende Situationen unterscheiden:

- *Neue Anfrage an Nicht-Zielknoten* Der Knoten kennt weder die durch die Kennung bezeichnete Anfrage, noch ist er das Ziel.
- *Neue Anfrage an Zielknoten* Der Knoten ist das Ziel der Anfrage, kennt jedoch die durch die Kennung bezeichnete Anfrage noch nicht.
- *Folgeanfragen* Die Kennung der Anfrage ist dem Knoten bereits bekannt.

Bei einer neuen Anfrage an Nicht-Zielknoten wird die Nachricht an alle Nachbarn mit Ausnahme desjenigen, über welchen die Nachricht erhalten wurde weitergeleitet. Dabei merkt sich der Knoten die Kennung der Anfrage und fügt sich dem Route Record der Nachricht hinzu.

Bei einer neuen Anfrage an den Zielknoten stellt der Route Record der Nachricht eine vollständige Route vom Startknoten zum Zielknoten dar. Da die Kennung dem Knoten noch nicht bekannt war, ist diese Nachricht die erste mit vollständiger Route und somit auch die Kürzeste. Der Zielknoten beantwortet diese Nachricht unmittelbar mit einer *Route Reply* genannten Nachricht, welche den Route Record beinhaltet und rückwärts über den darin beschriebenen Weg zum Initiator gesendet wird.

Bei Folgeanfragen stellt die Nachricht eine Wiederholung dar, welche sich durch Kreise im Kommunikationnetz ergeben hat, oder sie beschreibt eine schlechtere Route, welche länger gebraucht hat um den Knoten zu erreichen. Sie wird deshalb verworfen.

Nach abgeschlossener Route Discovery wird die resultierende Route für den Versand des Datenpakets benötigt und zu diesem Zweck im Datenpaket mitgeführt.

Da im Rahmen von DSR jede Routenfindung mit einem Broadcast durch das gesamte vom Startknoten erreichbare Kommunikationsnetz einhergeht, stellt es ein relativ kommunikationslastiges Routingverfahren dar. Es wurde aus diesem Grund in [Dan14] bezüglich

Kommunikationslast in der Simulation als Worst-Case-Verfahren eingesetzt. Durch den Verzicht auf statische Routingtabellen sind bei Veränderungen im Netz keine Anpassungen am Routing nötig. Die Routingmöglichkeiten ergeben sich stattdessen automatisch aus der Verschaltung der Materialflussmodule untereinander.

Die *Route Maintenance* findet während des Transports eines Datenpakets durch das Netz statt und kann in drahtlosen Ad-hoc Kommunikationsnetzen durch den *data link level* geleistet werden, da hier *hop-by-hop-acknowledgements* genutzt werden können, um fehlgeschlagene Übertragung zu erkennen und eine erneute Übermittlung des verlorenen Datenpakets anzustoßen. Kann die Übertragungsstörung nicht behoben werden, wird eine *Route Error*-Nachricht an den ursprünglichen Absender übermittelt. [JM96]

### 2.3.2 Adaption für Materialflusssysteme

In [Dan14] werden wesentliche Unterschiede zwischen Kommunikationsnetzen und Materialflusssystemen festgestellt. Dort werden auch Änderungen beschrieben um DSR für Materialflusssysteme zu adaptieren. Das Steuerungssystem weist dementsprechend die dort beschriebenen Änderungen auf, welche hier kurz zusammengefasst werden:

#### Trennung zwischen Datennetz und Materialflussnetz

Informationen und Transportgut werden über unterschiedliche Netze aus unterschiedlichen Medien transportiert. Folglich gilt die Annahme, dass die zuerst beim Ziel eingetroffene Route Request-Nachricht die beste Route enthält in Materialflusssystemen nicht. Dies wird anschaulich, wenn eine lange Route über wenige Materialflussmodule mit einer kürzeren Route mit sehr vielen Materialflussmodulen verglichen wird: Da die Route Request-Nachricht über die lange Route nur wenige Zwischenstationen besucht, wird sie den Zielknoten voraussichtlich eher erreichen, als die Nachricht über die kürzere Route, welche mehr Zwischenstationen besuchen muss. Dies folgt aus dem vernachlässigbar kleinen Unterschied in der Transportzeit in datenübermittelnden Leitungen mit für Stetigfördersysteme vorstellbaren Längenunterschieden, wodurch die Anzahl der besuchten Knoten im Kommunikationsnetz der dominierende Faktor in der Übermittlungsdauer wird. Im Gegensatz dazu ist die Transportdauer im Materialflussnetz maßgeblich durch die Transportgeschwindigkeit und Länge der einzelnen Fördertechnikabschnitte beeinflusst.

Um dennoch kürzeste Routen zu finden, werden bei der Weiterleitung einer Route Request-Nachricht jeweils die Kosten für den Transport des Guts über das bei Benutzung dieser Route zu befahrende Stück des zugrundliegenden MFM hinzugefügt. Um diese Kosten zu

bestimmen, benötigt der Routingagent Informationen über die dazu zu befahrende Strecke und Geschwindigkeit. Wie der Agent diese Informationen erhält, wird im Rahmen dieser Arbeit offen gelassen. Denkbar wäre jedoch, diese Daten bei der Herstellung des MFM festzulegen, da sich die Länge nicht nachträglich ändert, oder aber aus der Anlagensteuerung zu erfragen.

Damit diese Alternativrouten den Zielagenten erreichen, werden in der Adaption für Materialflusssysteme Anfragen mit bekannter Kennung nicht verworfen, sondern ebenfalls weitergeleitet. Auch für den Zielagenten ergeben sich Änderungen. So kann die erste erhaltene Route nicht ohne weiteres für die Route Reply-Nachricht verwendet werden, da später eintreffende Routen günstigere Kosten aufweisen könnten. Um dennoch eine Terminierung der Route Discovery zu gewährleisten, wartet der Zielagent eine definierte Zeitspanne auf das Eintreffen weiterer, potentiell günstigerer Route Request-Nachrichten. Nach Ablauf der Zeit wird dann die bis dahin günstigste Route versendet. Damit ist jedoch das Auffinden der günstigsten Route nicht mehr garantiert und von der gewählten Wartezeit abhängig.

#### Eigene Topologie der MFM

Ein MFM hat im Gegensatz zu einem Knoten in Datennetzen selbst eine Topologie. So können mehrere Stationen auf einem MFM hintereinander liegen. Es kann deshalb erforderlich sein, ein MFM mehrfach zu durchlaufen um zum gewünschten Ziel zu kommen. Um Kreise zu vermeiden und unnötig versendete Nachrichten zu vermeiden, wird bei Nachrichten zu unbekannten Anfragen zusätzlich zur Kennung die aktuell angefallenen Kosten und die Station mit welcher der MFM verbunden ist, von welchem die Nachricht erhalten wurde, gespeichert. Weitere Route Request-Nachrichten zu bereits bekannten Anfragen werden schließlich nur dann weitergeleitet, wenn sie folgende Bedingungen erfüllt:

- Die Nachricht wurde von einem MFM erhalten, welcher durch eine in Förderrichtung früheren Station verbunden ist, oder
- die Kosten der neuen Nachricht geringer als die aktuell bekannten Kosten sind.

Die minimalen Kosten und die früheste Station werden mit Erhalt jeder Nachricht entsprechend laufend aktualisiert. Auf diese Weise ist sichergestellt, dass nur Route Request-Nachrichten weitergeleitet werden, die einen Zugewinn an Routinginformation darstellen.

### Duplikation und Volatilität von Datenpaketen

Im Gegensatz zu Transportgütern können Datenpakete beliebig vervielfältigt werden, was ein erneutes Versenden nach einem Fehler ermöglicht. (vgl. 2.3.1) Störfälle auf Materialflussebene werden durch die *Route Maintenance*-Phase von DSR erkannt und behandelt:

Zu dem Zeitpunkt an dem das Transportgut ein MFM erreicht, versendet dessen Agent eine *Route Maintenance Request*-Nachricht an den Agenten des nächsten auf der Route folgenden MFM. Ist dieses MFM bereit, das Transportgut zu übernehmen, sendet sein Agent eine positive *Route Maintenance Reply*-Nachricht. Andernfalls wird eine negative Antwort übersendet und der anfragende Agent stößt eine erneute Route Discovery an um ggf. eine Ausweichroute zu bestimmen. Da die Störfallbehandlung im Rahmen der Modellanalyse nicht weiter thematisiert wird, wird hier nicht näher darauf eingegangen.

Ein weiterer Faktor ist das Ausbleiben von Antworten im Fall, dass während der Route Discovery keine Route ermittelt werden konnte. Dieser ist in Materialflusssystemen, Störfälle außer acht gelassen, nur bei ungültigen Anfragen möglich. Allerdings ist auch in solchen Fällen eine Rückmeldung wünschenswert um eine Unterscheidung zwischen einer noch nicht abgeschlossenen Routenfindung und einer abgeschlossenen, aber negativ ausgefallenen Routenfindung treffen zu können. Aus diesem Grund beantwortet der Agent am Ausgangspunkt der Route Discovery nach Ablauf einer definierten Zeitspanne mit einem negativen Route Reply, wenn er bis dahin keinen Route Request erhalten hat. Dieses Timeout kann ebenfalls dazu verwendet werden, gesammelte Daten über einen Route Request zu „vergessen“: Da bekannt ist, dass der Ausgangsagent die Routinganfrage nach Ablauf des Timeouts negativ beantwortet, können auch alle anderen Agenten (deren Timeout später begonnen hat und aus diesem Grund auch später abläuft) die Routenfindung für beendet erklären und dessen Kennung sowie dazu gespeicherte Daten Zeiten sowie Stationen löschen.

### Lastverteilung

Um verschiedene, alternative Routen möglichst gleichmäßig Auslasten zu können, ist es wünschenswert, diese ebenfalls bei der Routenfindung zu berücksichtigen. Zu diesem Zweck wird die erwartete Nutzung eines MFM auf die im Rahmen der Route Discovery gemeldeten Kosten für den Transport über ein MFM aufgeschlagen. Dazu führt jeder Agent einen Lastzähler, der beim Rückversand der Route Reply-Nachricht, also am Ende der Route Discovery-Phase bei bereits ermittelter Route, jeweils erhöht wird. Auf diese Weise werden einzelne Teilabschnitte mit Benutzung teurer, wodurch prinzipiell ungünstigere Routen im

Vergleich zunehmend attraktiver werden. Umgekehrt wird beim anschließend erfolgenden Transport eines Guts der Lastzähler wieder dekrementiert, wodurch die Kosten für einen Teilabschnitt nach erfolgter Benutzung wieder sinken.

Da die Lastverteilung im Steuerungssystem optional und für die Folgende Modellierung nicht relevant ist, wird sie im Folgenden nicht näher erläutert.

### 2.3.3 Verhaltensbasierte Umsetzung

Das Steuerungssystem ist in Java umgesetzt. Dazu wurde das an den FIPA-Standard angelehnte Agentenframework JADE (Java Agent Development Environment) genutzt. Die im Folgenden genannten Eigenschaften von JADE finden sich u.A. in [BCG07].

JADE ermöglicht den Entwurf von Agenten mittels eines verhaltensbasierten Ansatzes bei dem ein Agent seine Aufgaben durch die Annahme einer oder mehrerer Verhaltensweisen (engl. *Behaviours*) ausführt. Entsprechend ist im Steuerungssystem das Routing in disjunkte Komponenten zerlegt, welche als jeweils eine Verhaltensweise, die ein Agent annehmen kann, umgesetzt sind:

Das *DSRDelayedReplyBehaviour* nimmt keine Nachrichten entgegen sondern startet für eine spezifische Routinganfrage ein Timeout, nach dessen Ablauf der Agent die beste bisher entdeckte Route in Form einer neuen Route Reply-Nachricht versendet. Anschließend entfernt der Agent diese Verhaltensweise wieder. Sie ist entsprechend nur im Zielagenten aktiv und dient der Beantwortung einer spezifischen Routinganfrage, nach Ablauf einer Warteperiode, in der bessere Routen eintreffen können.

Das *DSRSequenceTimeoutBehaviour* ist dazu beinahe identisch, versendet jedoch eine Negativnachricht; also eine Meldung, dass keine Route gefunden werden konnte. Sie ist dementsprechend nur im Startagenten aktiv und dient der Beantwortung einer spezifischen Routinganfrage im Fall, dass keine Route existiert.

Das *DSRRequestBehaviour* nimmt alle Route Request-Nachrichten entgegen und verarbeitet diese. Dabei werden zwei Fälle unterschieden:

1. Der Agent ist nicht der Zielagent, und leitet die Nachricht an seine Nachbarn weiter; weitere eintreffende Nachrichten zu dieser Routinganfrage jedoch nur, falls diese einen Zugewinn an Routinginformation darstellen ( 2.3.2).

2. Der Agent ist der Zielagent, und speichert die gefundene Route. Dann fügt er sich die Verhaltensweise *DSRDelayedReplyBehaviour* hinzu. Bei weiteren eintreffenden Nachrichten vergleicht er deren Route mit der gespeicherten und behält die jeweils günstigere.

Der Startagent fügt sich bei Erhalt einer neuen Route Request-Nachricht von Außen außerdem die Verhaltensweise *DSRSequenceTimeoutBehaviour* hinzu.

Das *DSRReplyBehaviour* ist analog zum *DSRRequestBehaviour* permanent aktiv. Dabei leitet der Agent jede erhaltene Route Reply-Nachricht anhand des Route Records an den nächsten Agenten weiter.

Um zu garantieren, dass eine bestimmte Nachricht von einer bestimmten Verhaltensweise behandelt wird, sind diese so implementiert, dass sie jeweils nur bestimmte Nachrichtentypen verarbeiten können. So sollen etwa Route Requests ausschließlich vom *DSRRequestBehaviour* verarbeitet werden und nicht fälschlicherweise vom *DSRReplyBehaviour*. JADE implementiert die *ACL* (Agent Communication Language) welche *Performatives* (deutsch etwa *Sprechakte*) definiert. Während der Routenfindung kommen dazu Nachrichten vom Typ *Request* für die Route Request-Nachrichten und Nachrichten vom Typ *Reply* für die Route Reply-Nachrichten zur Anwendung. Folglich behandelt *DSRRequestBehaviour* ausschließlich Nachrichten vom Typ *Request*, während *DSRReplyBehaviour* ausschließlich Nachrichten vom Typ *Reply* behandelt. Die Nachrichten werden von den Verhaltensweisen grundsätzlich in der Reihenfolge des Eintreffens (FIFO) abgearbeitet.

Die hier vorgestellten Verhaltensweisen ergeben in ihrer Gesamtheit die *Route Discovery*-Phase der Adaptierung von DSR für das Routing in Materialflusssystemen.

JADE führt ein Round-Robin-Scheduling der Verhaltensweisen durch. Ferner wird jede gestartete Verhaltensweise eines Agenten nach Erhalt jeder Nachricht einmal aktiviert. Verhaltensweisen eines einzelnen Agenten können nicht gleichzeitig zur Ausführung kommen, sodass Synchronisationsmechanismen auf den Datenstrukturen des Agenten entfallen können. Haben die jeweils durchgeführten Schritte der Verhaltensweisen, wie die hier beschriebenen, selbst endliche Laufzeit, ist einerseits sichergestellt, dass jede Verhaltensweise nach endlicher Zeit zur Ausführung kommt. Da jeweils nur eine einzelne Verhaltensweise als Konsument einer Nachricht in Frage kommt, ist darüber hinaus sichergestellt, dass ein Agent jede Nachricht in der dafür vorgesehenen Verhaltensweise verarbeitet. JADE führt die Agenten in sogenannten *Containern* aus. Diese dienen als Ausführungsplattform für einen oder mehrere Agenten und können zu einem Gesamtsystem verbunden werden.

Die Container sind dabei für die Agenten und den Nachrichtenversand zwischen diesen transparent, sodass ein Agent keine Kenntnis über den Container, in dem sich der Empfänger seiner Nachricht befindet, besitzen muss. Die Agenten können zur Laufzeit zwischen den Containern migriert werden, wobei dieser Vorgang für alle anderen Agenten ebenfalls transparent ist.

Das in dieser Arbeit vorgestellte Steuerungssystemkonzept sieht entsprechend, jeweils einen Container pro MFM vor, wobei jeder Container genau einen, dem MFM zugeordneten Agenten ausführt. Jedoch kann bei Störfällen die das Agentensystem, nicht jedoch das Materialflusssystem als solches betreffen, der gestörte Agent in einen Nachbarcontainer verschoben, oder dort neu erzeugt werden.

## 2.4 Transitionssysteme zur Modellierung

Transitionssysteme, auch Zustandstransitionssysteme, erlauben die Modellierung von diskreten, verteilten Systemen und ihrem Verhalten derart, dass eine formale Analyse ermöglicht wird. Sie gelten mittlerweile gemeinhin als Standardverfahren der Modellierung nebenläufiger Systeme. [BKL08] So wird auch in der vorliegenden Arbeit ein Zustandstransitionssystem zur Modellierung der Routenfindung genutzt.

Neben ihrer Anwendung zur Modellierung nebenläufiger Softwaresysteme eignen sich Transitionsysteme in anderen Domänen, so auch in der Intralogistik, zur Abbildung und Formalisierung: Etwa zur Beschreibung von Abläufen und Prozessen [SWS12] oder in UML-Form zur Spezifikation von Steuerungen [Bra05].

### 2.4.1 Definition

Ein Zustandstransitionssystem stellt sich als Graph dar, wobei die Knoten des Graphen den Zuständen (engl. *State*) und die Kanten den Zustandsübergängen (engl. *Transition*) entsprechen. Im Folgenden wird von *STS* für *State Transition System* bei einem Zustandstransitionssystem gesprochen.

Formal ergibt sich ein STS dabei zunächst als Tupel $(S, T)$ aus einer Menge Zuständen $S$ und einer Menge Zustandsübergängen (Transitionen) $T \subseteq S \times S$. Um bei einer Transition eine erwartete Eingabe, eine notwendige Bedingung, oder eine durchgeführte Aktion ausdrücken zu können, werden ferner Beschriftungen (engl. *Labels*) der Transitionen eingeführt, sodass sich ein wie im Folgenden genutztes STS abschließend als Tripel $(S, L, T)$ aus

einer Menge Zuständen $S$, einer Menge Beschriftungen $L$ und einer Menge Transitionen $T \subseteq S \times L \times S$ ergibt. Um den Initialzustand des betrachteten Systems zu markieren, kann eine Teilmenge $I \subset S$ als Menge der Startzustände definiert werden.

### 2.4.2 Safety, Liveness und Fairness

Die in der Modellanalyse anhand des modellierten Zustandstransitionssystems zu zeigenden Eigenschaften lassen sich in *Safety*- und *Liveness*-Eigenschaften unterscheiden. Dabei sind Safety-Eigenschaften solche Eigenschaften, aus denen die Aussage folgt, dass ein (unerwünschtes) Ereignis niemals Eintritt, wohingegen aus Liveness-Eigenschaften die Aussage folgt, dass ein erwünschtes Verhalten (irgendwann) eintreten wird. Diese Eigenschaften sind in [AS85] formal definiert.

Für den Nachweis dieser Eigenschaften wird in der Modellanalyse die Betrachtung der Fairness bei der Ausführung von Aktionen relevant, da in nebenläufigen Systemen nicht nur die durchgeführten Aktionen selbst, sondern auch eine Garantie über den Zeitpunkt zu welchem die Aktionen ausgeführt werden von Bedeutung ist. In 2.3.3 wurde bereits das Schedulingverhalten von JADE als verwendete Plattform erläutert. Von besonderem Interesse bei der Modellanalyse ist dabei jedoch die Frage, ob das Scheduling der Verhaltensweisen so gestaltet ist, dass ein Agent, wenn die Bedingungen für die Ausführung einer Verhaltensweise erfüllt sind, diese Ausführung nach endlicher Zeit vornimmt.

Ein solches Verhalten wird *Weak fairness* genannt und garantiert, dass ein Agent nicht unendlich lange in einem Zustand verweilt, obwohl er in einen Nachfolgezustand schalten sollte: *Weak fairness asserts that an operation must be executed if it remains possible to do so for a long enough time.* [Lam94] Kann *Weak fairness* vorausgesetzt werden, hängt das Schalten der Transitionen nur noch von den Schaltbedingungen ab, sodass sich folgende Aussage für das zu modellierende STS ableiten lässt:

$$\forall t \in T : \text{Schaltbedingung in Label } l \in L \text{ für } t \text{ erfüllt } \rightarrow t \text{ wird schalten}$$

# Kapitel 3

# Funktionale Modellierung des Steuerungssystems

Um das vorgestellte Steuerungssystem auf seine Eignung zu untersuchen, soll es auf die Erfüllung funktionaler Eigenschaften hin untersucht werden. Diese soll zunächst auf Basis einer Safety- und Liveness-Analyse erfolgen, sodass die Modellierung in einer dazu geeigneten Form erfolgen muss. Dazu wird im Folgenden ein STS beschrieben, welches als Grundlage für die in Kapitel 4 folgende Analyse dient.

## 3.1 Vorüberlegungen

Da sich das Steuerungssystem auf die Routenfindung konzentriert, stehen zunächst Fragen bezüglich der Gültigkeit der Route im Vordergrund. Kann gezeigt werden, dass die berechneten Routen gültig sind, lässt sich damit eine prinzipielle Eignung des beschriebenen Systems zu Steuerung von Materialflusssystemen belegen. Dazu muss die Modellierung Aussagen über die ausgetauschten Nachrichten ermöglichen, da Routinganfrage und -antwort jeweils über Nachrichten ausgetauscht werden.

Eine weitere Frage stellt sich nach der Reaktivität des Systems. Dabei sollen Aussagen über das Antwortverhalten gewonnen werden, wobei gezeigt werden soll, dass das System nach endlicher Zeit auf eine Anfrage antwortet. Dadurch kann ausgeschlossen werden, dass eine Anfrage unbeantwortet bleibt.

Entsprechend muss die Modellierung sowohl alle durchgeführten Aktionen während der Route Discovery-Phase beinhalten, als auch die ausgetauschten Nachrichten und das System als solches geeignet abbilden. Dabei Eine Konkretisierung der zu untersuchenden Eigenschaften folgt anschließend in der Analyse dieses Modells in Kapitel 4.

## 3.2 Modellierung

Wird eine Route Request-Nachricht mit neuer Route Request-ID (also ein zuvor unbekannter Route Request empfangen), startet die Routenfindung bezogen auf diese ID im Startzustand. Dies erlaubt es, die Abarbeitung genau eines Route-Requests (also alle Nachrichten und Vorgänge bezogen auf eine einzelne Route Request ID) zu modellieren. Dies ist hinreichend, um sowohl die Safety-Eigenschaften als auch Liveness zu zeigen, da diese sich ohnehin auf die Gültigkeit einzelner Routen, also auf den Ablauf einer einzelnen Routenfindung beziehen.

Zunächst stellt sich die Netzstruktur im Modell wie folgt dar:

- $n > 0$ Anzahl Agenten im Netz, jeder Agent hat einen eindeutigen Index $i : 0 < i \leq n$
- $agents[2..n]$ Array der Agenten, durch Index bezeichnet, mindestens 2 Agenten im System
- $neighbors[i] = \{j : i$ und $j$ haben eine bidirektionale Verbindung$\}$
- $inbound[i] = [k : i$ eingehender Materialfluss von $k$ an der Position von $k$ in $inbound]$
- $outbound[i] = [l : i$ ausgehender Materialfluss zu $l$ an der Position von $l$ in $outbound]$

Da für die Zustandsübergänge der konkrete Nachrichteninhalt von Bedeutung ist, folgt eine Modellierung relevanter Teile der auszutauschenden Nachrichten ($msg$):

- $FC \in \{DQ, DA, DM\}$ wobei $DQ$ ein DSR-Route Request, $DA$ ein DSR-Route-Reply und $DM$ eine DSR-Route-Maintenance-Nachricht darstellt.
- $SQ = INT\ x : x > 0$, Sequence Number (Route Request ID)
- $SX = INT\ x : x > 0$, Sender Agent Index, der Start der Route
- $RX = INT\ x : x > 0$, Receiving Agent Index
- $RY = INT\ x : x > 0$, Receiving Station Index
- $CX = INT\ x : x > 0$, Agent Index of Source Station
- $DX = INT\ x : x > 0$, Destination Agent Index
- $DY = INT\ x : x > 0$, Destination Station Index

- $RR = [x, \ldots, z] : x, z \in agents$, Route Record
- $RT = DOUBLE\ x : x > 0.0$, Route Record Travel Time

Um die Kosten für die Route berechnen zu können, wird eine Funktion benötigt, welche die Kosten für das befahren der Strecke von Eingangsstation zu Ausgangsstation eines Materialflussmoduls im Agenten bereitstellt. Der Einfachheit halber wird im Folgenden von einer statischen Tabelle ausgegangen, welche dem Paar $k, l : k \in inbound, l \in outbound$ einen numerischen Wert zuordnet. Diese Funktion wird als $cost(k, l)$ bezeichnet.

Zusätzlich hält jeder Agent zu einer Routenfindung gehörige Daten intern vor. Diese werden durch die Route Request ID ($SQ$-Feld der Nachricht) eindeutig zugeordnet und in *cache* vorgehalten. *cache* ist dabei eine Datenstruktur zum Vorhalten gelernter Daten zu einer einzelnen Routenfindung. Dabei ist die Struktur an die Nachrichteninhalte gekoppelt, sodass prinzipiell die selben Felder gespeichert werden können. Die Zustände des STS sind dann jeweils genau einer Route Request ID zugeordnet, sodass durch das STS die Bearbeitung einer einzelnen Routenfindung modelliert wird.

Aus dem Ablauf des Algorithmus ergibt sich zunächst die Zustandsmenge
$S = \{newRR, notDest, isDest, done\}$:

- *newRR*: Route-Request-ID unbekannt, starte Routenfindung
- *notDest*: Route-Request weitergeleitet, bessere Route Request-Nachrichten werden weitergeleitet, Route Replies zurück propagiert
- *isDest*: Warten auf bessere Requests, Beantwortung nach Ablauf des Timeouts *tstop*
- *done*: Routenfindung beendet, der Agent führt keine Verhaltensweise zu der spezifischen Route-Request-ID mehr aus

Um Versand und Erhalt der Nachrichten abbilden zu können, wird ferner mit *medium* das Transportmedium, etwa das zugrundliegende Rechnernetz abgebildet.

Das Modell der Systemstruktur und sein initialer Zustand ergibt sich zusammenfassend wie folgt mit einer lose an Pascal angelehnten Syntax:

**Listing 3.1:** Modellierung der Systemstruktur

```
constant tstop in 1..t
```

```
agents[]: array [2..n] of agent
          i: in 1..n
          cache: bag of record
                 RT: in double > 0
                 RY: in 1..k
                 RR[]: array [1..n] of agent
                 T: of time
          inbound: array [1..k] of agent
          outbound: array [1..l] of agent

medium[]: bag of msg
          FC: in {DQ, DA, DM}
          SX: in 1..n
          RX: in 1..n
          RY: in 1..k
          CX: in 1..n
          DX: in 1..n
          DY: in 1..l
          RR[]: array [1..n] of agent
          RT: in double > 0

state[]: array [2..n] of {newRR, notDest, isDest, done}

Init
    medium = {}
    state = [newRR,...,newRR]
```

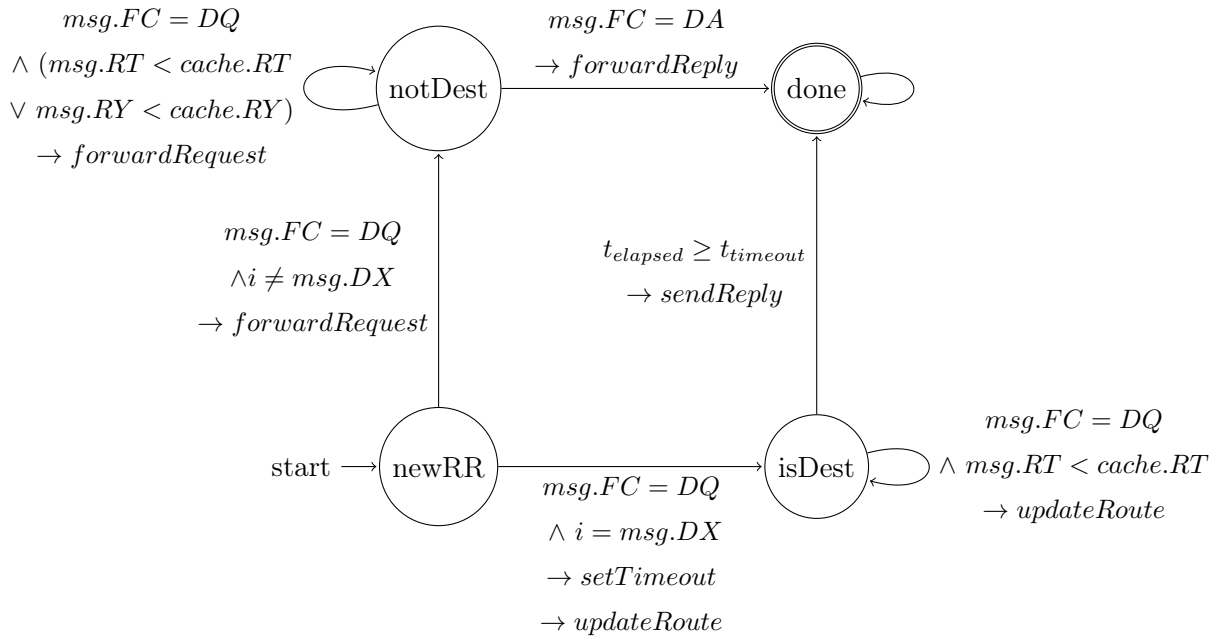
Die Menge der Übergänge $T$ zwischen den Zuständen bilden die verschiedenen Fallunterscheidungen beim Routing mit der DSR-Adaption ab:

- *newRR -> notDest* Neuen Route-Request erhalten: Weiterleiten
- *notDest -> notDest* Besseren Route-Request erhalten: Weiterleiten
- *notDest -> done* Route-Reply erhalten: Rückwärts Weiterleiten
- *newRR -> isDest* Route-Request an Zielknoten erhalten: Timeout starten
- *isDest -> isDest* Besseren Route-Request an Zielknoten erhalten und Timeout noch nicht abgelaufen: Route aktualisieren

- *isDest -> done* Timeout abgelaufen: Beantworte request

Die Übergänge zwischen diesen Zuständen sind im folgenden Diagramm in Abbildung 3.1 illustriert und formalisiert. Hierbei sind zunächst die für den Übergang geltenden Bedingungen und anschließend die durchgeführten Aktionen als Label-Menge $L$ aufgelistet.

**Abbildung 3.1:** Zustandsübergangsdiagramm der Routenfindung im Steuerungssystem




Der Übersichtlichkeit halber nicht explizit aufgeführt ist dabei das Verwerfen der Nachrichten. Dieses findet implizit immer dann statt, wenn in der Bedingung zum Schalten einer Transition zwar eine Nachricht erforderlich ist (also ein Datum aus $msg$) gelesen wird, die Bedingung aber unerfüllt bleibt, oder bei der ausgeführten Aktion keine Folgenachricht entsteht.

Beim Schalten der Transitionen werden folgende Aktionen durchgeführt:

*forwardRequest* speichert zunächst die minimale Dauer einer Route zu diesem Knoten.

$$set\ cache.RT\ to\ min(cache.RT, msg.RT)$$

Um sicherzustellen, dass eine Route falls nötig auch bei längerer Dauer als bereits bekannt weitergeleitet wird, wird die früheste Station über die eine Nachricht erhalten wird gespeichert

$$set\ cache.RY\ to\ min(cache.RY, msg.RY)$$

Der Nachrichtenversand erfolgt pro Nachbarknoten $j$ aus *outbound*. Dabei wird dem Route Record der Nachricht zunächst der ausführende Agent hinzugefügt

$$append\ i\ to\ msg.RR$$

in einer Kopie $msg'$ der Nachricht für jeden Nachbarn zu dem Materialfluss möglich ist werden jeweils die Kosten der Route um den Betrag erhöht, der bei Versand an den jeweiligen Nachbarn anfällt.

$$set\ msg'.RT\ to\ msg.RT + cost(msg.RR[msg.RR.length - 2], j)$$

Die Nachricht wird anschließend an alle Nachbarn, aus outbound mit einem Index größer als der Index des Agenten aus $msg.RY$, also allen von der Eingangsstation aus noch erreichbaren Nachbarn aus *outbound* versendet.

$$send\ msg'\ to\ j$$

*setTimeout* startet ein durch Konfiguration festgelegtes, endliches Zeitfenster, in dem der Zielknoten auf das Eintreffen weiterer Route Requests zur selben Sequenznummer warten kann.

$$start\ timeout$$

*updateRoute* speichert einen Route Request und dessen Kosten in *cache*. Dazu veranschlagt es die Kosten für das Befahren innerhalb des Zielknotens analog zur Vorgehensweise in *forwardRequest*:

$$set\ cache.RT = msg.RT + cost(msg.RR[msg.RR.length - 1], msg.DY)$$

$$set\ cache.RR = msg.RR + i$$

*sendReply* erzeugt zunächst eine Nachricht vom Typ Route Reply und legt darin die Route durch den besten gefundenen Route Record fest.

$$set\ msg.FC\ to\ DA$$

$$set\ msg.RT\ to\ cache.RT$$

$$set\ msg.RR\ to\ cache.RR$$

$$append\ i\ to\ msg.RR$$

Diese Nachricht wird nun dem Route Record entsprechend rückwärts an den dort jeweils aufgeführten Vorgänger versendet. *forwardReply* führt dieses Verhalten fort und versendet die Nachricht ebenfalls an den Knoten $p$, welcher unmittelbar vor ihm im Route Record enthalten ist.

$$send\ msg\ to\ p$$

Sollte kein Vorgänger mehr zu finden sein, ist der Startknoten erreicht und die Routenfindung ist beendet.

Die Zustandsübergänge und Veränderungen der Daten lässt sich dann wie folgt zusammenfassen:

**Listing 3.2:** Modellierung der Zustandsübergänge

```
newRR -> notDest:
    msg := removeFirst(medium, msg.FC = DQ)
    ^ state[i] = newRR ^ msg.FC = DQ ^ i != msg.DX
    ^ state'[i] = notDest ^ medium' = insert(medium,
       msg' to each j in i.outbound with j > msg.RY
            ^ msg'.RR = msg.RR ++ i ^ msg'.RT = msg.RT +
            cost(last(msg.RR.length), j)

notDest -> notDest:
    msg := removeFirst(medium, msg.FC = DQ)
    ^ state[i] = notDest ^ msg.FC = DQ ^ (msg.RT < i.cache.RT
                                    || msg.RY < i.cache.RY)
    ^ i.cache.RT' = min(i.cache.RT, msg.RT)
    ^ i.cache.RY' = min(i.cache.RY, msg.RY)
    ^ state'[i] = notDest ^ medium' = insert(medium,
       msg' to each j in i.outbound with j > msg.RY
            ^ msg'.RR = msg.RR ++ i ^ msg'.RT = msg.RT +
            cost(last(msg.RR), j)

newRR -> isDest:
    msg := removeFirst(medium, msg.FC = DQ)
    ^ state[i] = newRR ^ msg.FC = DQ ^ i = msg.DX
    ^ state'[i] = isDest
    ^ i.cache.T' = curTime() + tstop
    ^ i.cache.RR' = msg.RR ++ i
    ^ i.cache.RT' = msg.RT + cost(last(cache.RR), msg.DY)
```

```
isDest -> isDest:
    msg := removeFirst(medium, msg.FC = DQ)
    state[i] = isDest ^ msg.FC = DQ
    ^ msg.RT + cost(last(msg.RR), msg.DY) < i.cache.RT
    ^ state'[i] = isDest
    ^ i.cache.RR' = msg.RR ++ i
    ^ i.cache.RT' = msg.RT + cost(last(msg.RR), msg.DY)

isDest -> done:
    state[i] = isDest ^ i.cache.T >= curTime()
    ^ state'[i] = done
    ^ insert(medium, msg' to pre(cache.RR, i) :
        msg'.FC = DA, msg'.RR = cache.RR, msg'.RT = cache.RT)

notDest -> done:
    msg := removeFirst(medium, msg : msg.FC = DA)
    state[i] = notDest ^ msg.FC = DA
    ^ state'[i] = done
    ^ insert(medium, msg to pre(msg.RR, msg.SX))

done -> done:
    msg := removeFirst(medium, any)
```

Der Übersichtlichkeit halber sind auch in der obigen Definition einige Vorgänge stark vereinfacht und ihr Verhalten soll im Folgenden erläutert werden.

```
removeFirst(medium: array of msg, c : condition on msg)
```

Entnimmt die erste Nachricht aus *medium*, auf welche die Bedingung *c* zutrifft. Dies bildet die Zuordnung von Verhaltensweisen zu den Performatives in der Umsetzung des Steuerungssystems (vgl. 2.3.3) ab.

```
insert(medium: array of msg, m : msg)
```

Fügt die gegebene Nachricht $m$ dem *medium* hinzu.

```
cost(j: 1..n, k: 1..n)
```

Liefert die Kosten für den Transport eines Guts auf Agent $i$ von Eingangsagent $j$ nach Station $k$.

```
last(rr : array of agent)
```

Liefert den letzten Knoten im Route Record rr

```
predec(rr : array of agent, k: 1..n)
```

Liefert den Vorgänger von $i$ im Route Record $rr$ und nutzt ggf. die Eingangstation der Nachricht $k$ um die genaue Position im Route Record bei vorhandenen Kreisen zu bestimmen.

# Kapitel 4

# Modellanalyse

Das in Kapitel 3 beschriebene Modell kann nun auf Eigenschaften hin untersucht werden, die eine Aussage über die Eignung für einen praktischen Betrieb erlauben. Diese Analyse soll Aussagen über die Gültigkeit der Routen, Reaktivität des Systems und die Effizienz ermöglichen. Dazu wird das Modell zunächst sowohl auf Safety- als auch auf Livenesseigenschaften hin untersucht und abschließend das Wachstum des Kommunikationsaufwands betrachtet.

## 4.1 Safety-Eigenschaften

Für die Safety-Analyse von DSR werden solche Abläufe (also Routenfindungen) betrachtet, welche zu einer gültigen Route durch das System führen. Dabei werden folglich Routenfindungen nicht betrachtet, bei denen das Ziel vom Startknoten aus nicht erreichbar ist. Dazu sind einige Eigenschaften interessant, welche sich aus den formalen Anforderungen an einen Routing-Algorithmus ergeben. Diese drücken jene Eigenschaften aus, welche die Gültigkeit der Routen sicher stellen.

### 4.1.1 Erster Knoten im Route Record ist Startknoten

Diese Eigenschaft drückt aus, dass die gefundene Route tatsächlich beim Startknoten beginnt. Dabei ist der Beginn jedes Route Record gleich dem Startknoten. Dies kann durch folgende Bedingung ausgedrückt werden:

$$\forall\ msg : msg.RR \neq [\ ] \rightarrow msg.RR[0] = msg.SX$$

Die Garantie dieser Eigenschaft hängt dabei vornehmlich von zwei Bedingungen ab:

1. Der erste Knoten, der einen Route Request enthält ist der Startknoten der Route.
2. Der Startknoten der Route trägt sich in den Route Record ein.

Bedingung 1 ist durch die Systemarchitektur garantiert. Es wird im Folgenden o.B.d.A davon ausgegangen, dass das aufrufende System den initialen Route Request an den Startknoten der Route absetzt, wodurch diese Bedingung grundsätzlich erfüllt ist.

Bedingung 2 muss durch das Modell garantiert werden. Dazu ist die Veränderung des Route Record im Modell relevant. Dabei gelten folgende Aussagen welche sich an den Definitionen von *forwardRequest*, *updateRoute* und *sendReply* ablesen lassen:

- Dem Route Record wird grundsätzlich am Ende hinzugefügt, der bei Erhalt einer Nachricht bereits bestehende Teil des Route Record ist unveränderlich.
- Dem Route Record werden nur in *forwardRequest* und *updateRoute* neue Knoten hinzugefügt. *sendReply* konstruiert aus dem im Zielknoten gespeicherten und durch *updateRoute* ergänzten Route Record eine Antwort.

Da nun vorausgesetzt werden kann, dass ein bestehender Route Record nicht verändert, sondern ausschließlich ergänzt wird, ist lediglich zu klären, ob sich der Startknoten selbst im Route Record einträgt. Das ist einfach ersichtlich, da jeder Knoten bei der Routenfindung mit dem Zustand *newRR* beginnt. Dazu folgende Fallunterscheidung:

- Ein Knoten ist das Ziel: Dann wird er von Zustand *newRR* in *isDest* wechseln, wobei *updateRoute* ausgeführt wird, wodurch sich der Knoten im Route Record einträgt.
- Ein Knoten ist nicht das Ziel: Dann wird er von Zustand *newRR* in *notDest* wechseln, wobei *forwardRequest* ausgeführt wird, wodurch sich der Knoten im Route Record einträgt.

Andere Zustandsübergänge sind für diese Eigenschaft nicht weiter relevant, da diese nicht den Startzustand betreffen. Der Startknoten trägt sich also selbst im Route Record ein, weshalb dieser folglich mit dem Startknoten der Route beginnt.

### 4.1.2 Letzter Knoten im Route Record ist Zielknoten

Diese Eigenschaft drückt aus, dass die gefundene Route tatsächlich beim Zielknoten endet. Dabei ist das Ende jedes Route Record nach beendeter Routenfindung gleich dem Zielknoten. Dies kann durch folgende Bedingung ausgedrückt werden:

$$\forall\, msg : msg.FC = DA \rightarrow last(msg.RR) = msg.DX$$

Dabei kann genutzt werden, dass in 4.1.1 geklärt wurde, dass sich jeder Knoten bei Erhalt der ersten Nachricht in den Route Record einträgt. Ferner ist die Unveränderlichkeit eines bestehenden Route Records hilfreich und garantiert das unverfälschte Weiterleiten. Mit diesen Voraussetzungen verbleibt zu zeigen, dass nur der Zielknoten eine Antwortnachricht erzeugt.

Um eine Antwortnachricht zu erzeugen, muss ein Knoten sich im Zustand *isDest* befinden, welcher sich nur erreichen lässt, wenn der Knoten einen neuen Route Request erhält, wodurch er aus dem Startzustand *newRR* in den Zustand *isDest* wechselt. Bei diesem Wechsel und bei allen nachträglich eintreffenden alternativen Route Requests führt er *updateRoute* aus, wodurch der Route Record im Cache um diesen Zielknoten ergänzt wird. Der einzig verbleibende Zustandswechsel, welcher den Versand einer Nachricht zur Folge hat, ist der Wechsel von *isDest* zu *done*, wobei hier *sendReply* ausgeführt und damit die Antwortnachricht erzeugt und versendet wird. Da die Route in *cache* von *updateRoute* jeweils um den Zielknoten ergänzt wurde, enthält die in *sendReply* erzeugte Nachricht diesen als letztes Element.

### 4.1.3 Route Record enthält keine Kreise

Diese Eigenschaft fordert, dass die Knoten im Route-Record einer Route-Reply-Nachricht paarweise verschieden sind.

$$\forall i \in msg.RR : count(i, msg.RR) = 1$$

Üblicherweise führen Schleifen in Routing-Algorithmen die Gefahr der Nicht-terminierung ein. Es ist im Regelfall außerdem ineffizient, einen Knoten doppelt zu besuchen. Diese Eigenschaft ist für die vorliegende Adaption von DSR für Materialflusssysteme jedoch in dieser Form unzureichend, da ein Materialflussmodul (und wegen dessen Entsprechung zu einem einzelnen Agenten damit auch ein Knoten im Kommunikationsnetz) durchaus mehrfach durchlaufen werden kann, um sein Ziel zu erreichen. Dies kann sich aus der inneren Topologie der MFM ergeben: Wenn die einzige Ausgangsstation, über die ein Erreichen des Zielagenten möglich ist, topologisch (also in Materialflussrichtung) vor der Eingangsstation liegt, ein Kreis über einen anderen Agenten diese Ausgangsstation aber erreichbar macht, dann ist das Durchlaufen dieses Kreises notwendig.

**Abbildung 4.1:** Durch innere Topologie eines MFM notwendiger Kreisdurchlauf

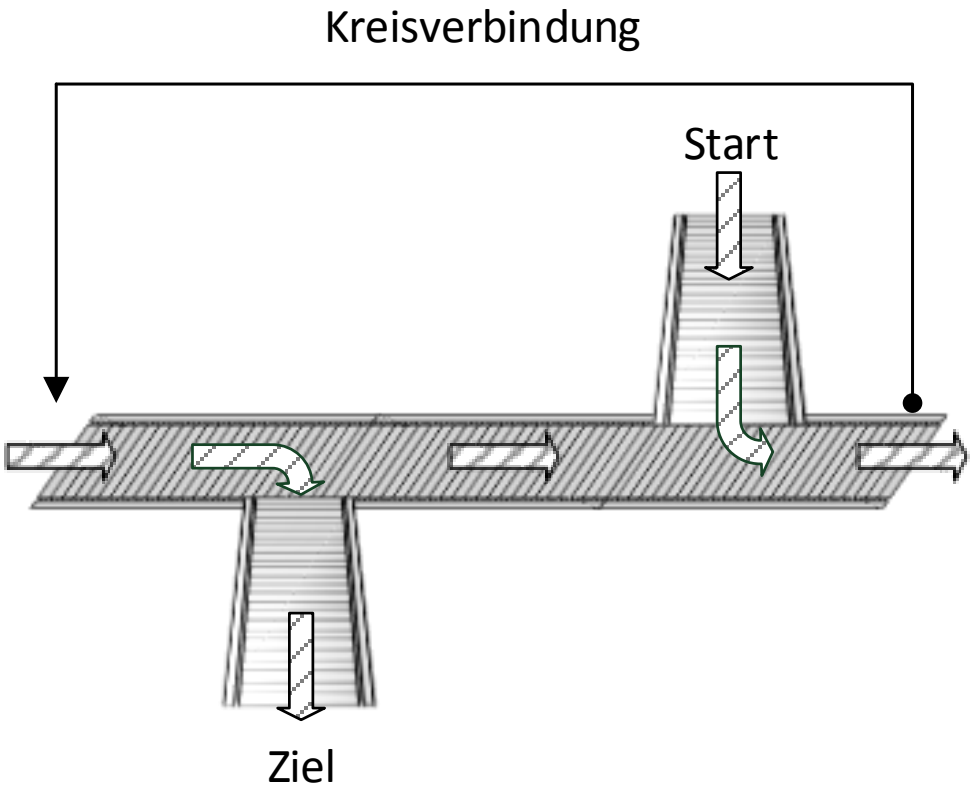



Abbildung 4.1 zeigt exemplarisch eine Situation, in der das Ziel vom Start nur dann erreichbar ist, wenn eine Kreisverbindung durchlaufen und damit ein MFM mehrfach durchlaufen wird.

Diese Kreise lassen sich allerdings in ihrer Anzahl beschränken: Da ein Agent eine endliche Anzahl an Eingangsstationen bedient, liegt ein unzulässiger Kreis genau dann vor, wenn ein einzelner Agent öfter im Route Record auftaucht, als er Eingangsstationen hat.

$$\forall i \in msg.RR : count(i, msg.RR) < i.inbound.length$$

Ob Kreise zugelassen sind, lässt sich im vorgestellten Steuerungssystem durch Konfiguration steuern, sodass beide Fälle abbildbar sind und Materialflusssysteme, welche Kreise zum Routing benötigen, gesteuert werden können. In der Modellierung wurden Kreise zugelassen, sodass ein Knoten maximal in der Anzahl seiner Eingangsstationen im Route Record vorkommen darf.

Entscheidend für die Verhinderung unkontrollierter Kreise bei der Routenfindung ist die Weiterleitung von Route Requests. Sobald ein Kreis vorliegt, ist die Nachricht zu verwerfen und nicht weiterzuleiten um einen endlosen Nachrichtenaustausch zu verhindern. Offensichtlich ist der erste weitergeleitete Route Request zunächst unkritisch und wird durch den Zustandswechsel eines Knoten *newRR* zu *notDest* ausgeführt. Jeder weitere eintreffende Route Request muss in *notDest* nun auf einen Zugewinn an Routinginformation geprüft werden.

Ein Zugewinn an Routinginformation liegt vor, wenn mindestens eine der folgenden Bedingungen zutreffen:

1. Die Kosten der Route sind geringer als die Kosten der bislang günstigsten Route: $msg.RT < cache.RT$

2. Der Knoten erhält die Nachricht über eine Station, welche noch vor der bislang frühesten Station in Materialflussrichtung liegt: $msg.RY < cache.RY$.

*forwardRequest* aktualisiert diese Daten in *cache* fortlaufend. Bedingung 1 garantiert, dass eine bessere Route als die bislang gefundenen weitergeleitet wird, während Bedingung 2 ermöglicht, auch teurere Routen weiterleiten zu können, wenn sie möglicherweise neue Pfade erschließen könnten. Dies ist notwendig, da je nach Station eine Route zwar günstig sein, aber nicht zum Ziel führen könnte, wohingegen eine möglicherweise teurere Route, welche jedoch über eine frühere Station eintrifft auch über eine frühere Ausgangsstation versendet werden und folglich eine neue Routingmöglichkeit erschließen kann.

Werden die Kosten einer Route außer Acht gelassen und als konstant angenommen, garantiert die Forderung nach einer echt kleineren neuen Eingangsstation für die Weiterleitung eines neuen Route Request bereits offensichtlich, dass maximal einmal pro Eingangsstation ein Route Request weitergeleitet werden kann, da dabei in *forwardRequest* jeweils die neue, früheste Eingangsstation gespeichert wird. Dies entspricht in abgewandelter Form genau der geforderten Bedingung.

Da die Kosten für die Routen jedoch nicht konstant sind, muss sichergestellt sein, dass diese mit wachsendem Route Record nicht kleiner werden können. Dies ist genau dann der Fall, wenn bei jeder Weiterleitung ein Wert $\geq 0$ zu $msg.RT$ addiert wird. Es wird deshalb angenommen, dass die Kosten jeder (Teil-)Route nicht negativ sein können, also:

$$\forall i \in [0, \ldots, n] \; \forall k \in \; inbound(i) \; \forall l \in outbound(i) \; cost(k, l) \geq 0$$

Damit können die Kosten für eine Route als mindestens konstant vorausgesetzt werden, wodurch Bedingung 2 beschränkend eingreift, sobald ein Route Request über eine bereits behandelte oder in Materialflussrichtung hintere Station eintrifft: Die Kosten sinken bei einem Kreisdurchlauf nicht, weshalb deren Anzahl auf die Anzahl der Eingangsstationen beschränkt bleibt.

## 4.2 Liveness

Die *Lebendigkeit (Liveness)* des Systems drückt grundsätzlich aus, ob garantiert werden kann, dass in endlicher Zeit eine Antwort erfolgt. Im Falle der Routenfindung, hier also der *Route Discovery*-Phase der DSR-Adaption, kann diese Antwort jedoch sowohl positiv, also mit berechneter Route, als auch negativ, also ohne berechnete Route ausfallen. Das Steuerungssystem garantiert eine Antwort vom Startagenten in endlicher Zeit durch Konfiguration eines Timeouts, nach dessen Ablauf eine Routenfindung für gescheitert erachtet und die Anfrage negativ beantwortet wird. Dieser Umstand ist jedoch nicht hinreichend für die Liveness des Systems. Stattdessen soll durch den Nachweis der Liveness gezeigt werden, dass nach endlicher Zeit eine positive Antwort auf eine Routinganfrage erfolgt. Dazu wird im Folgenden o.B.d.A der Fall einer durchführbaren Routinganfrage betrachtet, was bedeutet dass die betrachtete Routinganfrage so gestaltet ist, dass eine Route im Materiaflusssystem MFS vom Start zum angefragten Ziel existiert, sodass der Routingalgorithmus eine solche finden kann.

Die zu zeigende Liveness-Bedingung ergibt sich demnach als:

$$route(msg.CX, msg.DX) \in MFS \rightarrow agents[msg.SX]\ receives\ msg : msg.FC = DA$$

Um die Liveness der vorliegenden DSR-Adaption zu untersuchen, wird im Folgenden die Routenfindung in 2 Phasen getrennt. Diese ermöglicht eine vereinfachende Fallunterscheidung bei der Liveness-Betrachtung und verschärft die in 2.3 vorgestellte *Route Discovery*-Phase:

Während der *Request Propagation*-Phase leiten Agenten empfangene Route Request-Nachrichten an benachbarte, im Materialfluss nachgelagerte Agenten weiter, bis der Zielagent erreicht ist. Die Liveness-Bedingung dieser Phase ergibt sich zu:

$$route(msg.CX, msg.DX) \in MFS \rightarrow agents[msg.DX]\ receives\ msg : msg.FC = DQ$$

Während der *Reply Forwarding*-Phase leiten Agenten empfangene Route Reply-Nachrichten an den ihren jeweiligen Vorgänger im Route Record der Route-Reply-Nachricht weiter, bis

der Startagent erreicht ist. Die Liveness-Bedingung dieser Phase ergibt sich entsprechend zu:

$$\exists msg : msg.FC = DA \rightarrow agents[msg.SX]\ receives\ msg$$

### 4.2.1 Weak fairness

Das beschriebene Steuerungssystem und JADE als Plattform garantieren *Weak fairness* durch die Art des Schedulings der Verhaltensweisen. JADE aktiviert die Ausführung eines Agenten, sobald der Agent eine Nachricht erhält, oder ein vordefinierter Zeitpunkt erreicht ist. Aktiviert JADE einen Agenten, so aktiviert er damit auch grundsätzlich nacheinander in einem Round-Robin-Verfahren alle in diesem Agenten aktiven Verhaltensweisen. Diese prüfen einzeln, ob die Bedingungen für ihre weitere Ausführung erfüllt sind oder nicht und bearbeiten jeweils genau ein einzelnes Ereignis. Damit ist Weak fairness bereits erreicht, da garantiert ist, dass jede Verhaltensweise zur Ausführung kommen kann, wenn seine Bedingungen erfüllt sind und damit nur eine endliche Zeit zwischen Erfüllung der Schaltbedingung und dem tatsächlichen Schalten einer Transition liegt.

Folgendes Beispiel soll diesen Sachverhalt veranschaulichen: Ein Agent hat in seinem Nachrichtenpuffer eine große Anzahl an Route Request-Nachrichten zu bearbeiten und eine einzelne Route Reply-Nachricht, welche er weiterzuleiten hat. Ferner sei die Anzahl an Route Request-Nachrichten, die dieser Agent pro Zeiteinheit erhält, größer als die Rate an Route Request-Nachrichten, die dieser in einer Zeiteinheit verarbeiten kann.

Liegt keine Weak-Fairness vor, so ist es möglich, dass dieser Agent die vorliegende Route Reply-Nachricht unendlich lang vorhält und nicht weiterleitet, da er mit dem niemals kleiner werdenden Strom an Route Request-Nachrichten beschäftigt ist.

Das vorliegende Steuerungssystem bringt jedoch bei jedem Nachrichtenerhalt alle Verhaltensweisen nacheinander zur Ausführung, wobei diese jeweils nur genau eine einzelne Nachricht verarbeiten und jeder Verhaltensweise genau ein Nachrichtentypus zugewiesen ist. Folglich wird die Route Reply-Nachricht entweder sofort versendet, wenn diese Verhaltensweise zuerst aktiviert wird, oder aber unmittelbar nach der Verarbeitung genau einer einzelnen Route Request-Nachricht. Der zeitliche Abstand zwischen Eintritt eines Ereignis und Ausführung der daran geknüpften Aktion ist also zeitlich beschränkt und damit endlich.

### 4.2.2 Liveness der Request Propagation-Phase

Um die Liveness der Request Propagation-Phase zu untersuchen, wird der Versand der Route Request-Nachrichten vom Startagenten ausgehend zum Zielagenten betrachtet. Dabei ist nur jener Teil des Kommunikationsnetzes relevant, der durch Nachrichtenversand vom Startknoten aus erreichbar ist.

Eine Route Request-Nachricht wird genau dann nicht mehr weiter geleitet, wenn der Zielknoten erreicht ist, oder ein Agent keinen Nachfolger mehr hat, an den er eine Nachricht weiter leiten kann. Unter der Bedingung, dass eine Routinganfrage gültig ist, dass also ein Weg vom Start zum angefragten Ziel existiert, ist damit die Request Propagation-Phase erfolgreich beendet, da implizit auch der Zielagent erreicht worden ist.

Dieser systemweite Zustand lässt sich also durch folgende Eigenschaften charakterisieren:

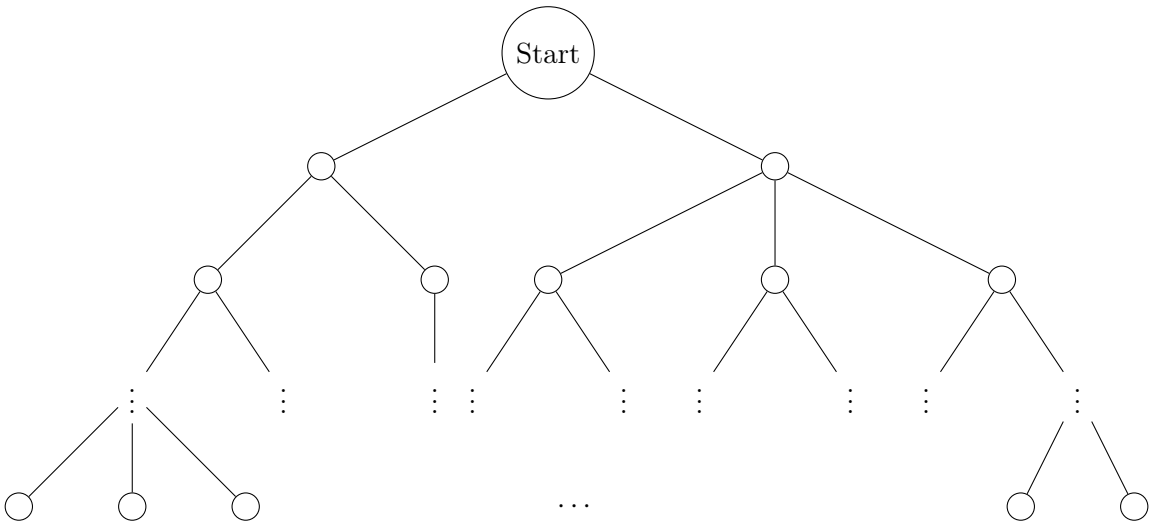
- Kein Agent befindet sich im Zustand $newRR$
- Es sind keine Route Request-Nachrichten zwischen den Agenten mehr auszutauschen

Die zu erfüllende Liveness-Eigenschaft der Request Propagation-Phase ergibt sich entsprechend aus diesen Bedingungen:

$$I \in T : \forall i \in [1 \ldots n] : state[i] = newRR$$
$$\Rightarrow$$
$$Q \in T : \forall i \in [1 \ldots n] : state[i] \neq newRR \wedge \ medium = \emptyset$$

Die Menge $T$ bezeichnet dabei die Menge möglicher Systemzustände. Dabei drückt $I$ den Initialzustand im System aus, in welchem jeder Agent sich im Startzustand der Routenfindung befindet, während $Q$ den durch die genannten Bedingungen definierten Endzustand des Systems definiert. Der Operator $Rightarrow$ wird in diesem Fall als *folgt nach endlicher Zeit* definiert.

**Abbildung 4.2:** Spannbaum im Kommunikationsnetz durch Zustandswechsel auslösende Route Request-Nachrichten




Ist dieser Spannbaum vollständig erkundet, sind alle Agenten aus dem Zustand $newRR$ in einen Nachfolgezustand gewechselt und einer der Blattknoten stellt den Zielknoten dar. Da der Spannbaum durch die initialen Nachrichten, welche den Zustandswechsel der Agenten aus $newRR$ auslösen, gebildet ist, ist mit der Erkundung des Spannbaums jedoch noch nicht das Ende der Request Propagation-Phase erreicht: Es können weiterhin Nachrichten ausgetauscht werden, da auch zwischen den Teilbäumen Kommunikationsverbindungen bestehen. Der Fortschritt im Spannbaum und das Ende seiner Erkundung in endlicher Zeit ist demnach noch nicht hinreichend für die Liveness.

Da der Spannbaum nur die Zustandsübergänge der Agenten, also die jeweils erste erhaltene Route Request-Nachricht eines Agenten abbildet, wird, um die Liveness-Eigenschaft zu zeigen, zunächst ein *Lattice* eingeführt. Dieser definiert eine Halbordnung über einen Vektorraum, womit im Folgenden der Fortschritt in der Request Propagation-Phase untersucht werden kann. Zunächst ergibt sich der Vektorraum $L$ durch Vektoren der Form $(a, r)$ wobei $a$ die Anzahl Agenten im Zustand $newRR$ und $r$ die Anzahl Route Request-Nachrichten im Medium widerspiegelt. Der Endzustand $Q$ entspricht folglich dem Vektor $(0, 0)$.

Ferner wird eine Halbordnung $P \prec P' : P, P' \in L$ definiert:

$$P < P' \Leftrightarrow \exists i : P_i < P'_i \wedge \forall j \neq i : P_j \leq P'_j$$

Ein Zustand im System ist im Sinne dieser Ordnung also kleiner, wenn der Vektor der ihn beschreibt, an wenigstens einer Stelle echt kleiner, jedoch an keiner weiteren echt größer ist. Dies dann der Fall, sobald ein weiterer Agent aus dem Startzustand in einen Nach-

folgezustand übergeht ohne die Nachrichtenanzahl im Medium zu erhöhen, oder wenn ein Agent eine Folgenachricht entgegen nimmt, aber verwirft. Die Ordnung entspricht also dem Vektorvergleich. Da der Endzustand das Minimum des Lattice bildet, liegt ein Fortschritt in Richtung des definierten Endzustands also dann vor, wenn sich sich der Lattice-Wert verringert. Dabei zeigt sich bereits jetzt, dass die Agentenkomponente $a$ jedes Vektors im Verlauf nicht wächst, da aus der Modellierung (3.1) ersichtlich ist, dass für einen Agenten keine Möglichkeit besteht, in den Startzustand zurückzukehren.

Damit die Livenessbedingung im Sinne der *endlichen Zeit* erfüllt ist sind ferner zwei Bedingungen gelten:

1. Es sind nur endlich viele Schritte nötig, um von $I$ zu $Q$ zu gelangen
2. Die nötigen Zwischenschritte schalten nach endlicher Zeit

Bedingung 1 ist erfüllt, wenn der Lattice ein definiertes, endliches Maximum besitzt und alle durchgeführten Einzelaktionen den Lattice-Wert jeweils verringern. Dabei können lediglich die folgenden Aktionen durchgeführt werden:

- *forwardRequest* entnimmt eine Nachricht und versendet an alle Nachbarn, dabei wird während der Request Propagation-Phase von $newRR$ in den Zustand $notDest$ geschaltet, oder innerhalb von $isDest$ eine Alternative weitergeleitet.
- *updateRoute* entnimmt eine Nachricht ohne eine Folgenachricht zu versenden. Diese Aktion tritt auf, wenn der Zielagent von $newRR$ in den Zustand $isDest$ schaltet, oder bessere Routen erhält.
- *drop* ist wie bisher nicht explizit modelliert, bildet im Folgenden aber den Erhalt einer Nachricht ohne nachfolgenden Versand ab. Dies ist während der Route Propagation-Phase insbesondere dann der Fall, wenn ein Agent von $newRR$ in $notDest$ schaltet, jedoch keine Nachfolger mehr hat, wenn also die Menge der Empfänger in $forwardRequest$ leer ist.

Anhand der Definitionen der Aktionen kann beobachtet werden, dass jede Aktion die Nachrichtenanzahl $r$ um 1 verringert, wohingegen aber nur bei einem Zustandswechsel auch die Agentenkomponente $a$ gleichermaßen jeweils um 1 verringert wird. Dadurch wird ebenfalls nochmals deutlich, weshalb statt des Spannbaums die kombinierte Betrachtung von Zuständen und Nachrichten im Lattice hilfreich ist: Wenn Aktionen existieren, die keinen Zustandswechsel der Agenten zur Folge haben, stellen diese keinen Fortschritt bezüglich der Anzahl der Agenten im Startzustand dar. Im vorliegenden Fall findet bedingt

$forwardRequest$ und *updateRoute* nur beim ersten Schalten einen Zustandswechsel und erzielt folglich nicht grundsätzlich einen Fortschritt im Spannbaum.

Allerdings stellt die Erhöhung der Nachrichtenkomponente $r$ bei Versand einer Nachricht ein Hindernis dar: Um zu zeigen, dass nur endlich viele Schritte bis zum Erreichen des Minimums $I = (0, 0)$ im Lattice erforderlich sind, muss jeder Schritt im Lattice einen Fortschritt in Richtung dieses Minimums darstellen, andernfalls könnte das System *stottern*, also z.B. zwischen Inkrement und Dekrement im Lattice hin und her springen.

Aus diesem Grund wird die Nachrichtenanzahlkomponente $r$ des Vektors leicht modifiziert: Statt die momentane Anzahl Nachrichten im System auszudrücken, drückt sie fortan die Anzahl noch nicht von einem Agenten aus dem Medium entnommener, also unbehandelter Nachrichten aus. Sie entspricht also der Anzahl der noch nicht verarbeiteten Nachrichten im System. Nun wird die Nachrichtenanzahl durch jede in der Request Propagation-Phase mögliche Aktion um 1 verringert und fortan nicht mehr erhöht. Ferner ist es einem beteiligten Agenten nicht möglich, im Startzustand $newRR$ zu verbleiben, wodurch sich die Anzahl Agenten mit dem jeweils ersten Schalten von $forwardRequest$ und *updateRoute* zusätzlich verringert.

Aus der Safety-Betrachtung in 4.1.3 ist klar, dass auch Kreise nicht dazu führen, dass Teilabschnitte unendlich oft durchlaufen werden. Daraus folgt, dass die mögliche Anzahl der Nachrichten endlich ist und der Lattice lässt sich mit dem Maximalwert $(n, m)$ im Startvektor $I$ definieren, wobei $m$ die Anzahl aller auszutauschenden Nachrichten ausdrückt.

Da, wie vorab in 4.2.1 gezeigt, *weak fairness* vorliegt, werden die einzelnen Aktionen in endlicher Zeit nach Erfüllung ihrer Bedingung ausgeführt. Es wird also in jedem Schritt der Lattice-Wert verringert, wobei jeder Schritt nach nur endlich langer Verzögerung durchgeführt wird.

Folglich ist spätestens nach einem theoretischen Maximum von $n+m$ endlich langen Schritten das Minimum des Lattice $I$ erreicht und somit sowohl der Zielagent über die Route informiert, als auch alle Nachrichten ausgetauscht, womit die Route Propagation-Phase beendet. Da sowohl die Anzahl Agenten $n$ als auch die Anzahl der zwischen diesen Agenten in der Route Propagation-Phase austauschbaren Nachrichten $m$ endlich sind, wird folglich das Ende der Route Propagation-Phase in endlicher Zeit erreicht.

### 4.2.3 Liveness der Reply Forwarding-Phase

Die Liveness-Eigenschaft der Reply Propagation-Phase ergibt sich bereits aus den Safety-Eigenschaften und der Weak fairness:

Da eine Route Reply-Nachricht vorliegt, muss der Zielagent erreicht worden sein:

$$\exists msg : msg.FC = DA \rightarrow state[msg.DX] = done$$

Für die Safety-Eigenschaften wurde ferner festgestellt, dass ein Agent sich bei der Weiterleitung einer Route Request-Nachricht dem Route Record der Nachricht am Ende hinzufügt. Da der Zielagent erreicht worden ist, folgt somit auch, dass der Route Record den Transportweg für die Route Reply-Nachricht darstellt, diese also beim Startagenten ankommen kann:

$$\exists msg : msg.FC = DA \rightarrow msg.RR[i-1] \in agents[msg.RR[i]].inbound$$

Mit Hilfe der Weak-Fairness folgt nun, dass die Route Reply-Nachricht den Startagenten auch erreichen und dort behandelt wird; dass also jeder Agent im Route Record, die Route Reply Nachricht in endlicher Zeit nach Erhalt weiterleiten wird. Somit ist garantiert, dass eine vom Zielagenten verschickte Route Reply-Nachricht nach endlicher Zeit beim Startagenten ankommt. Damit erfüllt die Reply Propagation-Phase die Liveness-Eigenschaft.

$$\exists msg : msg.FC = DA \rightarrow agents[msg.CX] receives msg$$

### 4.2.4 Liveness der Routenfindung

Aus der Liveness-Betrachtung der einzelnen Phasen folgt genau dann die Liveness der DSR-Adaption, wenn der Zielagent in endlicher Zeit nach Erhalt einer Route Request-Nachricht eine Route Reply-Nachricht versendet. Dies entspricht dem eventuellen Schalten der Transition von *isDest* zu *done* und folgt wiederum aus der Weak-Fairness, womit auch der Übergang zwischen den Phasen der Liveness-Anforderung genügt. Die Liveness-Eigenschaft der Route Discovery-Phase der beschriebenen DSR-Adaption

$$route(msg.CX, msg.DX) \in MFS \rightarrow agents[msg.CX] \ receives \ msg : msg.FC = DA$$

ist damit erfüllt.

## 4.3 Kommunikationswachstum

Das Kommunikationsverhalten des Steuerungssystems ist ein letztes, wichtiges Kriterium zur Beurteilung der praktischen Einsetzbarkeit des vorgestellten Konzepts: Wächst das

Nachrichtenaufkommen zu stark überproportional zur Systemgröße und -komplexität, so ist das ein möglicher Hinweis auf die Existenz einer technischen Grenze ab der die Leistung von eingebetteten CPS nicht mehr ausreicht, oder diese so dimensioniert sein müssen, dass die praktische Anwendung im Vergleich zu klassischen Materialflusssteuerungen unwirtschaftlich erscheint.

Um das Kommunikationverhalten zu analysieren, wird deshalb das asymptotische Wachstum der Anzahl auszutauschender Nachrichten bezogen auf Systemgröße und -komplexität betrachtet. Dazu wird die schon in der Liveness-Betrachtung eingeführte Fallunterscheidung zwischen Route Propagation-Phase und Reply Forwarding-Phase bemüht.

Sollte die Anzahl Nachrichten dabei exponentiell mit der Systemgröße wachsen, könnte das Steuerungssystem nur für vergleichsweise kleine Materialflusssysteme sinnvoll eingesetzt werden, was im Rahmen dieser Arbeit als ein Ausschlusskriterium für seine praktische Einsetzbarkeit erachtet würde.

### 4.3.1 Kommunikation während der Request Propagation

Grundsätzlich ähnelt die Request Propagation-Phase stark einer kürzesten-Wege-Suche mit Spannbaumerkundung. Allerdings wird dabei üblicherweise jeder Knoten nur einmal besucht. Ein Agent im Steuerungssystem erhält jedoch möglicherweise mehrfach Nachrichten, da sie bezogen auf seine innere Topologie in einer ungünstiger Reihenfolge eintreffen, oder sich jedesmal, bezogen auf die Kosten der Route, verbessern.

Um eine obere Abschätzung zu gewinnen, ist deshalb zunächst ein hypothetisches Szenario zu konstruieren:

1. Vom Startagenten lässt sich jeder andere Agent des Systems erreichen.

2. Jeder Agent kann von jedem Punkt seiner inneren Topologie aus den Zielagenten erreichen.

3. Die Route Request-Nachrichten erreichen jeden Agenten in umgekehrter Reihenfolge der Materialflussrichtung, sodass jeder Route Request über die nächste frühere Eingangsstation erreicht wird, als der vorherige.

4. Der Zielagent wird darüber hinaus in absteigender Reihenfolge der Kosten erreicht, sodass jede eintreffende, berechnete Route eine Verbesserung darstellt.

5. Alle Eingangsschnittstellen eines Agenten liegen in Materialflussrichtung vor seinen Ausgangsschnittstellen.

Das Maximum an Kommunikation stellt sich in diesem Szenario folglich am Zielagenten ein, da dieser alle Route Request-Nachrichten erhält. Wenn $n$ die Anzahl der Agenten im System, und $k$ und $l$ jeweils die maximale Anzahl an Eingangs- und Ausgangsstationen eines Agenten sind, so erhält der Zielagent mindestens $n$-Nachrichten, da jeder Agent erreicht wird (Bedingung 1) und selbst wiederum den Zielagenten erreichen kann (Bedingung 2). Damit ergibt sich eine erste obere Schranke für die Anzahl Nachrichten des Agenten $n$. Damit ist jedoch nur zunächst jeweils eine initiale Nachricht pro Agent berücksichtigt. Wegen Bedingung 3, 4 und 5 erhöht sich das Nachrichtenaufkommen durch die ungünstige Reihenfolge ihres Eintreffens für jeden Agenten jeweils um die Anzahl der Eingangs- und Ausgangsschnittstellen und die Anzahl an Nachrichten wächst auf von $n * k * l$.

Hier zeigt sich, dass das Szenario mit Schwächen behaftet ist: Kreise sind wegen Bedingung 5 ausgeschlossen und die Anzahl an Transportschnittstellen pro Agent kann stark schwanken, sodass eine weitergehende Abschätzung auf dieser Basis schwierig erscheint. Allerdings ist nun ersichtlich, dass die Anzahl der bei einer Routenfindung ausgetauschten Nachrichten nicht so sehr von der Anzahl Agenten im System, sondern vielmehr von der Anzahl der Transportschnittstellen abhängt.

Wenn das Stetigfördersystem aus $s$ logistischen Elementen vom Typ $S_Q, S_s$ oder $S_T$ (vgl. 2.2.2) besteht, so entspricht die Suche nach dem kürzesten Pfad im Materialflusssystem der Suche nach dem kürzesten Pfad in einem Graphen. Hierbei werden jedoch alle Pfade des Materialflussystems erkundet, wodurch die Request Propagation-Phase auf der Ebene der logistischen Elemente der Suche nach kürzesten Wegen in einem Graphen entspricht. Ein allgemein bekannter Algorithmus mit quadratischer Laufzeit für dieses Problem ist Dijkstra's Algorithmus, dessen Voraussetzung der nicht-negativen Kantengewichte hier entsprechend analog auch im Materialflusssystem gilt, da die Kosten auf den Strecken basieren, welche nicht negativ sein können. Folglich ergibt sich eine Abschätzung für die Request Propagation-Phase zu $(s^2)$.

Eine mögliche Komplikation besteht jedoch weiterhin darin, dass Agenten im Steuerungssystem mehrfach besucht werden müssen, da eine später eintreffende Nachricht geringere Kosten aufweist als die vorherige. Dies trifft auch dann zu, wenn das Kommunikationsaufkommen abhängig von den logistischen Elementen betrachtet wird, bleibt aber, da das jeweilige Minimum pro Agent gilt (vgl. 3.2), auf die Anzahl Agenten im System beschränkt. Folglich erhöht sich das Nachrichtenaufkommen weiter; jedoch statt auf $(s^3)$ voraussichtlich

nur um die Anzahl der Agenten des Steuerungssystems zu $\mathcal{O}(s^2 * n)$, wobei dies jedoch wiederum stark davon abhängig ist, wie viele logistische Elemente den MFM zugeordnet sind. Unter der Annahme, dass etwa die Hälfte aller nachträglichen Route Request-Nachrichten deshalb weitergeleitet werden, weil sie bessere Kosten aufweisen, als vorherige Nachrichten, reduziert sich diese Schranke zu $\mathcal{O}(s^2 * (n/2))$.

### 4.3.2 Kommunikation während des Reply Forwarding

Beim Weiterleiten einer Route Reply-Nachricht ist ein Agent zunächst nur ein einziges mal beteiligt. Das Kommunikationsaufkommen bezüglich eines einzelnen Agenten ist entsprechend:

$$\mathcal{O}(1)$$

Dies gilt jedoch nur, sofern der Route Record frei von Kreisen, die dort aufgeführten Agenten also paarweise verschieden sind. Andernfalls ist ein Agent mehrfach im Route Record enthalten. Folglich muss er die selbe Route-Reply-Nachricht mehrfach weiterleiten. Die Anzahl Kreise, an denen ein Agent beteiligt sein kann, ist jedoch wie in 2.3.2 beschrieben und als Safety-Eigenschaft in 4.1.3 nachgewiesen durch die Anzahl seiner Eingangsschnittstellen beschränkt. Folglich ergibt sich für einen Agenten die obere Schranke für die Anzahl an Route Reply-Nachrichten die er weiterleitet als:

$$\mathcal{O}(k)$$

Bezogen auf das Gesamtsystem ergibt sich unter der Annahme, dass alle $n$ Agenten an der Route beteiligt sind und eine maximale Anzahl Eingangsstationen $k$ haben, eine obere Schranke von

$$\mathcal{O}(n * k)$$

für die Reply Forwarding-Phase. Dies lässt sich alternativ auch durch die Anzahl MFM des längsten Pfads $p$ eines Eingangs zu einem Ausgang im MFS ausdrücken, weshalb sich alternativ die obere Schranke zu

$$\mathcal{O}(p)$$

ergibt.

# Kapitel 5

# Auswertung

## 5.1 Fokus von Modellierung und Analyse

Die Modellierung in 3 und die darauf aufbauenden Analysen in 4 blieben, dem Schwerpunkt dieser Arbeit entsprechend, auf den Routenfindungsaspekt des Steuerungssystem beschränkt. Folglich wurde die *Route Discovery*-Phase von DSR untersucht, wobei im Einzelnen ausgenutzt wurde, dass die bei der Umsetzung genutzten Technologien hilfreiche Eigenschaften zur Analyse garantieren.

Darüber hinaus wurde sowohl bei der Modellierung, als auch bei der Analyse jeweils eine einzelne Routenfindung betrachtet und von der parallelen Bearbeitung der Routinganfragen bewusst abstrahiert, da sich sowohl die betrachteten Safety-Eigenschaften, als auch die Liveness jeweils auf eine einzelne Routenfindung beziehen.

## 5.2 Safety Eigenschaften

Die Analyse auf Safety-Eigenschaften in 4.1 hatte zum Ziel, die Gültigkeit berechneter Routen zu bestimmen. Dabei wurde überprüft, ob gefundene Routen im Startknoten beginnen, im Zielknoten enden und darüber hinaus etwaige Kreise nicht unendlich oft durchlaufen. Sie setzen damit Kriterien fest, die für die Eignung zum praktischen Einsatz des Konzepts notwendig sind. Die Erfüllung dieser Eigenschaften stellt folglich eine Schlüsselanforderung an das Routing im Materialflusssystem und damit eine funktionale Anforderung an das Steuerungssystem dar.

Es konnte gezeigt werden, dass das Modell des Steuerungssystemkonzepts diese Anforderungen erfüllt; dass also gefundene Routen beim angefragten Startknoten beginnen, dass sie gleichermaßen beim Zielknoten enden und dass sie keine beliebig langen Kreise enthalten. Unter der gegebenen Voraussetzung, dass die Erreichbarkeit von Agenten im MFS der Erreichbarkeit von Zwischenstationen im MFS entspricht, ist dies ein starker Hinweis darauf, dass die berechneten Routen gültig sind.

Der Nachweis selbst gelang mit vergleichsweise geringem Aufwand und stellte sich in Teilen bereits aus der Modellierung ersichtlich dar.

## 5.3 Liveness

Die Analyse der Liveness in 4.2 hatte zum Ziel, das Antwortverhalten des Steuerungssystems bei der Routenfindung zu analysieren. Im Gegensatz zur Safety-Analyse wurden deshalb keine Eigenschaften der Route an sich, sondern solche über den eventuellen Versand einer Antwort untersucht. Es ist intuitiv ersichtlich, dass ein Steuerungssystem zum Routing nur dann sinnvoll einsetzbar ist, wenn auf die Frage nach einer Route auch eine Antwort folgen wird. Dementsprechend ist auch diese Eigenschaft ein notwendiges Kriterium für den praktischen Einsatz und somit eine funktionale Eigenschaft des Steuerungssystems.

Es konnte gezeigt werden, dass das Modell des Steuerungssystemkonzepts diese Anforderung erfüllt; dass also nur endlich viel Zeit zwischen der Anfrage einer Route und der Beantwortung der Anfrage liegt. Hierbei wurde der Fall gültiger Eingaben, also Anfragen zu tatsächlich erreichbaren Zielen, betrachtet. Da das Steuerungssystem für ungültige Routingabfragen, welche ohnehin unerwünscht sind und nicht den Regelfall darstellen, war dieser Fall nicht Gegenstand der Untersuchung.

Für den Nachweis der Liveness wurde die *Route Discovery*-Phase von DSR weiter aufgeteilt und eine Unterscheidung zwischen der *Request Propagation*-Phase, die sich durch das Fluten des Kommunikationsnetzes mit Route Request-Nachrichten auszeichnet, einerseits, sowie der *Reply Forwarding*-Phase, die sich durch den Rückversand der Routingantwort zum Startknoten auszeichnet, andererseits eingeführt. Die Liveness beider Phasen konnte getrennt voneinander gezeigt werden. Für die *Route Discovery*-Phase war dabei eine Einführung eines *Lattice* und der Nachweis des schrittweisen Fortschritts in diesem erforderlich, wohingegen für die *Reply Forwarding*-Phase, die Liveness leicht ersichtlich erschien. Auch für den Beginn der *Reply Forwarding*-Phase aus der *Request Propagation*-Phase heraus, also das Umschalten zwischen Routenfindung unter Weiterleitung der Antwort konnte Liveness gezeigt werden, wodurch die Route Discovery-Phase der Adaption von DSR im Steuerungssytemkonzept insgesamt die geforderte Liveness-Eigenschaft erfüllt.

## 5.4 Wachstum des Kommunikationsaufwands

Die Analyse des Wachstums des Kommunikationsaufwands in 4.3 hatte zum Ziel, zu prüfen, ob dieser mit wachsender Größe bzw. Komplexität des zu steuernden Materialflusssystems

in einem beherrschbaren Verhältnis steigt. Der Kommunikationsaufwand stellt hier also einen Ersatz zum Laufzeitverhalten von nicht-verteilten Algorithmen dar.

Das Wachstum des Kommunikationsaufwands mit steigender Materialflusssystemgröße und -Komplexität stellt dabei möglicherweise einen Hinweis dar, ob eine algorithmisch bedingte Schwelle existiert, ab der ein Einsatz in realen Bedingungen nicht, oder nicht wirtschaftlich, sinnvoll erscheint. Dementsprechend stellt es eine Kenngröße für Leistung und Effizienz dar und ist folglich eine nicht-funktionale Eigenschaft des Steuerungssystems

Um die Laufzeit zu überprüfen, wurde die während der Liveness-Analyse eingeführte Teilung der *Route Discovery* in eine *Request Propagation*- und eine *Reply Forwarding*-Phase aufgegriffen und beide Phasen getrennt voneinander untersucht. Für die *Reply Forwarding*-Phase wurde dabei schnell ersichtlich, dass diese in linearer Zeit mit der Länge der Route beendet ist und entsprechend asymptotisch durch die Anzahl Agenten (bei Kreisen zusätzlich auch ihrer Eingangsstationen) und damit durch die Anzahl der Materialflussmodule im System beschränkt ist.

Die Request Propagation-Phase entspricht prinzipiell dem Fluten (eng: Flooding) des gesamten, vom Startagenten aus erreichbaren Kommunikationsnetzes. Dabei können jedoch auch nachträglich eintreffende Nachrichten weiterversendet werden, wenn diese etwa längere Nachrichtenlaufzeiten, aber kürzere Kosten haben. Aus der in 2.3.2 beschriebenen Ungleichheit, besonders bezüglich der Kosten, zwischen Kommunikationsnetz und Materialflusssystem ist damit bereits ersichtlich, dass Annahmen der Art *die beste Route wird wahrscheinlich auch die erste übermittelte sein* keine Gültigkeit besitzen. Entsprechend konnte bereits erwartet werden, dass dieser Teil nicht ohne einschränkende Nebenbedingungen und Annahmen abgeschätzt werden kann.

Indem die Request Propagation-Phase auf Ebene der logistischen Elemente des Materialflusssystems der Suche kürzester Wege als Standardproblem der Informatik gegenübergestellt wurde, konnte ein erster Eindruck von der zu erwartenden Laufzeit erhalten werden. Wegen der ungewissen Nachrichtenreihenfolge weicht die Nachrichtenmenge davon allerdings nach oben ab und wird in einem Bereich zwischen quadratischer und kubischer Laufzeit liegen.

Diese Abschätzungen bleiben jedoch grobe Schätzungen und hängen stark von den Reihenfolgen ab, in denen Nachrichten einen Agenten erreichen und sind deshalb mit großer Unsicherheit behaftet. Sie bleiben folglich in ihrer Aussagekraft hinter den Erwartungen zurück. Es zeigt sich jedoch, dass der Kommunikationsaufwand der Request Propagation-Phase der Adaption von DSR deutlich stärker von der Anzahl der Transportschnittstellen

im Materialflusssystem als von der Anzahl Materialflussmodule abhängt. Eine Ergänzung der Adaption für Materialflusssysteme um die bislang nicht umgesetzten Optimierungen der ursprünglichen DSR-Beschreibung erscheinen vor diesem Hintergrund lohnenswert.

## 5.5 Eignung zum Routing in realen Materialflusssystemen

Da mit Hilfe der Safety-Eigenschaften die Gültigkeit der berechneten Routen und mit der Liveness des Systems auch ein Antwortverhalten mit endlicher Verzögerung garantiert werden konnte, wird das Steuerungssystemkonzept bezüglich seiner funktionalen Eigenschaften als geeignet erachtet, auch im praktischen Einsatz an realen MFS das Routing zu leisten.

Die Betrachtung des Kommunikationsaufwands wirft demgegenüber die Frage nach der Größe der noch praktikabel steuerbaren MFS auf, wobei das augenscheinlich polynomielle Wachstum hier noch kein Ausschlusskriterium darstellt. Das Wachstum der Nachrichtenmenge wird, auch wegen des verbleibenden Optimierungspotentials des Routingalgorithmus, noch als positives Ergebnis; gewertet; u.A. auch deshalb, weil diese Nachrichtenmenge, teilweise parallel, durch alle beteiligten Agenten bearbeitet wird und jeder Agent nur eine begrenzte Menge an Nachrichten zu bewältigen hat. Obwohl der Eindruck entstanden ist, dass noch weitere Analysen zu diesem Aspekt erforderlich sind, wird das beschriebene Steuerungssystemkonzept deshalb auch mit Bezug zur Effizienz als nicht-funktionaler Anforderung als geeignet erachtet.

# Kapitel 6

# Zusammenfassung

In dieser Arbeit wurde ein Konzept für die verteilte, multiagentenbasierte Steuerung von Materialflusssystemen beschrieben und analysiert. Dazu baute diese Arbeit auf Vorarbeiten des Authors auf, die dieser im Rahmen unterstützender Tätigkeiten für die Dissertationsschrift *Verteilte Simulation und Emulation von Materialflusssystemen mit dezentraler Steuerung* von Herrn Dr. Damian Daniluk [Dan14] geleistet hat. Motiviert durch aktuelle Forschungstrends im Umfeld des *Internet der Dinge* und der *cyber-physischen Systeme*, sollte dabei die Frage beantwortet werden, ob das dort erarbeitete und umgesetzte Steuerungssystemkonzept mit in das Fördersystem eingebetteten Rechnern abseits simulativer, empirischer Studien auch für den praktischen Einsatz in realen Materialflusssystemen geeignet ist.

Das während der unterstützenden Tätigkeiten entstandene Steuerungssystem und im Besonderen dessen Adaption von DSR für Materialflussysteme wurden dazu in einem ersten Schritt anhand des in [Dan14] definierten Begriffs des *Materialflussmoduls* näher erläutert und seine Verfahrensweise und auf Verhaltensweisen basierende Umsetzung beschrieben. Das Routingverfahren wurde anschließend mit Hilfe von Zustandstransitionssystemen modelliert und auf Safety- und Livenesseigenschaften hin untersucht, wobei sowohl Modellierung als auch Modellanalyse auf den Aspekt der Routenfindung beschränkt blieben und die folgende Routenüberwachung und folglich Störfallbehandlung und Lastverteilung bewusst nicht berücksichtigten. Es konnten dazu mit den drei Safety-Eigenschaften *Erster Knoten im Route Record ist Startknoten*, *Letzter Knoten im Route Record ist Zielknoten* und *Route Record enthält keine* (unendlich häufigen) *Kreise* grundlegende Anforderungen an die Routenfindung bezüglich Gültigkeit gefundener Routen garantiert und somit ein erstes Zwischenziel erreicht werden. In der folgenden Liveness-Analyse konnte gezeigt werden, dass auf eine Routinganfrage in endlicher Zeit eine Rountingantwort folgt, womit auch ein zweites Zwischenziel erreicht werden konnte und sich die prinzipielle Tauglichkeit des Steuerungssystems für das Routing in Materialflusssystemen zeigen ließ.

Gerade für die Liveness-Analyse zeigte sich dabei die Umsetzung der Adaption von DSR mit einem verhaltensbasierten Ansatz und JADE als verwendeter Plattform hilfreich, da der Nachweis der Weak-Fairness von dadurch garantierten Voraussetzungen erleichtert wurde.

Eine abschließende Analyse des Kommunikationsaufwands hatte zum Ziel, auszuschließen, dass dieser mit steigender Systemgröße derart unverhältnismäßig wächst, dass die Möglichkeit des realen Einsatzes gefährdet scheint. Diese Analyse blieb in ihrer Aussagekraft aufgrund von Unsicherheitsfaktoren beim Versand von Nachrichten hinter den Erwartungen zurück, wurde schließlich jedoch nicht als Hinweis auf ein Ausschlusskriterium gewertet.

Das in dieser Arbeit beschriebene, verteilte, multiagentenbasierte Steuerungssystemkonzept wird im Endergebnis entsprechend als geeignet erachtet, abseits simulativer Studien auch in realen Materialflussystemen das Routing der Güter zu leisten, womit das Ziel der Arbeit erreicht werden konnte.

Im Hinblick auf zukünftige Arbeiten stellt sich zunächst eine experimentelle Untersuchung des Laufzeit- und Kommunikationsverhaltens als lohnenswert heraus, da sich dadurch ein Hinweis auf eine geeignete Hardwareausstattung der Materialflussmodule und der Dimensionierung ihres Kommunikationsnetzes ergeben kann. Während die Analyse des Kommunikationsaufwands im Rahmen dieser Arbeit lediglich mit dem Ziel geführt wurde, ein mögliches Ausschlusskriterium für reale Einsatzszenarien zu untersuchen, könnte das Kommunikationsverhalten in zukünftigen Arbeiten auch mit dem Ziel analysiert werden, verschiedene alternative, reaktive Routingverfahren gegenüber zu stellen. Ferner enthielt die in dieser Arbeit untersuchte Adaption von DSR für Materialflusssysteme keine der bereits in der Erstbeschreibung von DSR enthaltenen Optimierungen wie etwa das Erlernen und Caching von Routen während der Ausführung. Diese könnten erheblichen Einfluss auf die Antwortzeiten des Systems und das Kommunikationsverhalten der Agenten haben, weshalb ein Vorstoß in diese Richtung lohnenswert erscheint. Der Fokus der Modellierung und Analyse im Rahmen dieser Arbeit lag auf der Routenfindung mittels DSR. Die bei DSR stattfindende Überwachung der Route bei ihrer Durchführung wurde hingegen nicht näher beleuchtet, womit Lastverteilung und Störfallbehandlung unberücksichtigt geblieben sind, was ebenfalls einen möglichen nächsten Schritt darstellt.

# Abbildungsverzeichnis

# Literaturverzeichnis


[AS85] Alpern, Bowen ; Schneider, Fred B.: Defining liveness. In: *Information processing letters* 21 (1985), Nr. 4, S. 181–185

[AS87] Alpern, Bowen ; Schneider, Fred B.: Recognizing safety and liveness. In: *Distributed computing* 2 (1987), Nr. 3, S. 117–126

[BCG07] Bellifemine, Fabio L. ; Caire, Giovanni ; Greenwood, Dominic: *Developing multi-agent systems with JADE*. Bd. 7. John Wiley & Sons, 2007

[BDD+13] Bauer, Klaus ; Diegner, Bernhard ; Diemer, Johannes ; Dorst, Wolfgang ; Ferber, Stefan ; Glatz, Rainer et a.: Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0 / acatech - Deutsche Akademie der Technikwissenschaften e.V. Version: April 2013. `http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Abschlussbericht_Industrie4.0_barrierefrei.pdf`. 2013. – Forschungsbericht. – Zuletzt geprüft: Juni 2016

[BKL08] Baier, Christel ; Katoen, Joost-Pieter ; Larsen, Kim G.: *Principles of model checking*. MIT press, 2008

[Bra05] Braatz, Arnulf: *Entwicklung einer Methode zur objektorientierten Spezifikation von Steuerungen*. Heimsheim : Jost-Jetter, 2005. – ISBN 3–936947–66–X

[Büc00] Büchter, H: Steuerungen für Stückgutfördersysteme: Entwicklungspotenziale und Trends. In: *F+H Fördern und Heben* 50 (2000), Nr. 3, S. 156–158

[CHF+16] Chen, Mo ; Hu, Qie ; Fisac, Jaime ; Akametalu, Kene ; Mackin, Casey ; Tomlin, Claire: Guaranteeing Safety and Liveness of Unmanned Aerial Vehicle Platoons on Air Highways. In: *To be submitted to The American Institute of Aeronautics and Astronautics Journal of Guidance, Control, and Dynamics* (2016), Februar. `https://arxiv.org/pdf/1602.08150v1`. – Zuletzt geprüft: Juni 2016

[Dan14] Daniluk, D.: *Verteilte Simulation und Emulation von Materialflusssystemen mit dezentraler Steuerung*. Verlag Praxiswissen, 2014 (Logistik für die Praxis). – ISBN 9783869750934

[DFR08] Düdder, Boris ; Follert, Guido ; Roidl, Moritz: Model Checking in multiagentengesteuerten Materialflusssystemen / TU Dortmund. Version: 2008.

http://www-seal.cs.tu-dortmund.de/seal/downloads/duedder/papers/MMB2008.pdf. 2008. – Forschungsbericht. – Zuletzt geprüft: Juni 2016

[FG96] Franklin, Stan ; Graesser, Art: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: *International Workshop on Agent Theories, Architectures, and Languages* Springer, 1996, S. 21–35

[Gro84] Grosseschallau, W: *Materialflussrechnung.* Berlin, Heidelberg, New York, London, Paris, Tokyo: Springer. zugleich Habilitationsschrift, 1984. – ISBN 3540130934

[Gün12] Günther, WA: Algorithmen und Kommunikationssysteme für die Zellurare Fördertechnik / Fraunhofer-Institut für Materialfluss und Logistik (IML), Lehrstuhl für Fördertechnik Materialfluss Logistik (fml) TU München. Version: 2012. http://www.fml.mw.tum.de/fml/images/Publikationen/Abschlussbericht_AiF_16166N.pdf. Lehrstuhl für Fördertechnik Materialfluß Logistik (fml) TU München, 2012. – Forschungsbericht. – Zuletzt geprüft: Juni 2016

[JM96] Johnson, David B. ; Maltz, David A.: Dynamic source routing in ad hoc wireless networks. In: *Mobile computing.* Springer, 1996, S. 153–181

[Joh94] Johnson, David B.: Routing in ad hoc networks of mobile hosts. In: *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on* IEEE, 1994, S. 158–163

[Lam94] Lamport, Leslie: The temporal logic of actions. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 16 (1994), Nr. 3, S. 872–923

[SWS12] Schmidt, Thorsten ; Wustmann, David ; Schmaler, Robert: Process Analysis for Material Flow Systems. (2012). http://www.mhi.org/downloads/learning/cicmhe/colloquium/2012/schmidt.pdf. – Zuletzt geprüft: September 2016

[TKSS13] Turek, Karsten ; Klotz, Thomas ; Schmidt, Thorsten ; Straube, Bernd: Ein Ansatz zur Verifikation von Materialflusssystemen durch Model Checking. In: *15. ASIM Fachtagung Simulation in Produktion und Logistik* (2013), Oktober. http://www.asim-fachtagung-spl.de/asim2013/papers/Proof_161_Turek.pdf. – Zuletzt geprüft: Juni 2016

[VDI73] VDI: *VDI 3300: Materialfluß-Untersuchungen.* Beuth, Berlin, 1973. – Zurückziehungsdatum: 02.07.2008. Zurückziehungsgrund: fehlende Aktualisierung.

[Wei99] Weiss, Gerhard: *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999

[WJK00] Wooldridge, Michael ; Jennings, Nicholas R. ; Kinny, David: The Gaia Methodology for Agent-Oriented Analysis and Design. In: *Autonomous Agents and Multi-Agent Systems* 3 (2000), September, Nr. 3, 285–312. `http://dx.doi.org/10.1023/A:1010071910869`. – DOI 10.1023/A:1010071910869. – ISSN 1387–2532

[YZW06] Yang, Huabing ; Zhang, Xingyuan ; Wang, Yuanyuan: A correctness proof of the dsr protocol. In: *International Conference on Mobile Ad-Hoc and Sensor Networks* Springer, 2006. – ISBN 978–3–540–49932–9, S. 72–83

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 5. Oktober 2016

Christian Hoppe