Regelbasierte Modellierung mit autonomen Transformationseinheiten*

Karsten Hölscher, Renate Klempien-Hinrichs,
Peter Knirsch, Hans-Jörg Kreowski, Sabine Kuske
hoelscher@informatik.uni-bremen.de, kh@biba.uni-bremen.de,
{knirsch,kreo,kuske}@informatik.uni-bremen.de

Abstract: In dieser Arbeit werden autonome Transformationseinheiten als Modellierungskonzept eingeführt. Es ist regelbasiert und erlaubt die Beschreibung von eigenständigen Komponenten und Prozessen, die innerhalb eines Gesamtsystems interagieren und dabei eine gemeinsame Umgebung transformieren, um ihre eigenen Ziele und das Gesamtziel zu erreichen. Jede Transformationseinheit besitzt eine autonome Kontrolle, die jeweils entscheidet, welche der möglichen eigenen Regeln als nächste angewendet wird. Da die Anwendbarkeit von Regeln vom Zustand der Umgebung abhängt, können die Transformationseinheiten sich über die Veränderung der Umgebung gegenseitig beeinflussen.

1 Einleitung

Datenverarbeitende Systeme bestehen heutzutage meist aus vielen Komponenten, die weit verteilt sein können, die gelegentlich oder häufig interagieren und miteinander kommunizieren, die sich teilweise ad hoc zu Netzen verbinden können, die auch nicht unbedingt ortsgebunden sind, sondern sich bewegen oder herumgetragen werden können. Solche Komponenten mögen eigenständig agieren und Prozesse duchführen, doch erst ihr Zusammenwirken macht das Gesamtsystem aus und führt zur Erledigung der gestellten Aufgaben. In dieser Arbeit schlagen wir das Konzept der autonomen Einheiten für die Modellierung solcher Systeme vor. Autonome Transformationseinheiten bilden eine Gemeinschaft in einer gemeinsamen Umgebung, in der sie ihre Aktivitäten entfalten und diese Umgebung dabei verändern können. Sie sind regelbasiert, wobei die Anwendung von Regeln gerade zu Veränderungen der Umgebung führt. Sie haben Ziele, die sie durch Anwendung der Regeln zu erreichen versuchen, und sie besitzen eine autonome Kontrolle, die ihnen in jeder Situation und zu jeder Zeit erlaubt, aus allen jeweils aktuell anwendbaren Regeln diejenige auszuwählen, die tatsächlich als nächste angewendet wird.

Die Motivation, autonome Einheiten als Modellierungskonzept einzuführen, stammt aus dem Sonderforschungsbereich 637 *Selbststeuerung logistischer Prozesse – Ein Paradig-*

*Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Sonderforschungsbereichs 637 “Selbststeuerung logistischer Prozesse” und von der Europäischen Union im Rahmen des RTN-Projekts SegraVis “Syntactic and Semantic Integration of Visual Modelling Techniques” unterstützt.

menwechsel und seine Grenzen (siehe z.B. [FHS04]), in dem interdisziplinär die Frage verfolgt wird, ob und unter welchen Umständen Selbststeuerung gegenüber der herkömmlichen Fremdsteuerung logistischer Prozesse Vorteile hinsichtlich Zeit, Kosten und Robustheit bringt. Die Leitidee bei autonomen Transformationseinheiten ist, die Möglichkeit zur Selbststeuerung in die Modellierung von Prozessen zu integrieren und damit insbesondere einen Rahmen zu schaffen, in dem Selbststeuerungsmechanismen semantisch fundiert untersucht und miteinander verglichen werden können. Die Ziele im Einzelnen sind:

1. Algorithmische und insbesondere logistische Prozesse sollen einheitlich und in großer Allgemeinheit beschrieben werden können, wobei vor allem auch auf eine formale Semantik Wert gelegt wird.

2. Das Spektrum der Anwendungen und der einbezogenen Methoden soll von klassischen Prozesskettenmodellen wie dem von Kuhn (siehe z.B. [Kuh02]) oder Scheer (siehe z.B. [Sch02]) und den bestens bekannten Petri-Netzen (siehe z.B. [Rei98]) bis hin zu Agentensystemen (siehe z.B. [Wei99]) und Schwarmintelligenz (siehe z.B. [WP98]) reichen.

3. Die Prozessdynamik soll sich auf der Regelbasiertheit autonomer Transformationseinheiten abstützen. Einzelne Prozess-, Transformations- und Berechnungsschritte entstehen durch Regelanwendungen, die lokale Veränderungen bewirken. Vorbild dabei sind sowohl grammatikalische Systeme aller Art (siehe z.B. [RS97]) und Termersetzungssysteme (siehe z.B. [BN98]) als auch die Gebiete Graphtransformation (siehe z.B. [Roz97, EKMR99, EEKR99]) und DNA-Computing (siehe z.B. [PRS98]). Die Regelbasiertheit soll sowohl die Ausführbarkeit der Semantik sicherstellen als auch Grundlage für Verifikationmethoden werden.

4. Der Aspekt der Selbststeuerung soll sich auf zwei Ebenen manifestieren. Zum einen besteht ein System aus einer Gemeinschaft autonomer Transformationseinheiten in einer gemeinsamen Umgebung. Auf dieser Ebene sind alle Transformationseinheiten gleichberechtigt und können jederzeit aktiv werden ohne Rücksicht auf andere, wobei die Aktivitäten allerdings vom aktuellen Zustand der Umgebung abhängen, da Regeln anwendbar sein müssen. Eine zusätzliche Steuerung findet nicht statt, die Transformationseinheiten sind also autonom. Zum anderen sind die autonomen Transformationseinheiten als Regelsysteme üblicherweise nichtdeterministisch, da mehrere Regeln oder eine Regel an mehreren Stellen anwendbar sein können. Die Selbststeuerung dient dann dazu, sich jeweils zwischen den Möglichkeiten zu entscheiden.

Autonome Transformationseinheiten werden im folgenden Abschnitt eingeführt. In den Abschnitten 3 bis 5 wird die Semantik von Systemen autonomer Transformationseinheiten in drei Stufen definiert. Den Anfang macht eine sequentielle Semantik, die sehr einfach ist, aber nur für Systeme angemessen ist, bei denen zu einer Zeit immer nur eine Aktivität möglich ist. Viele Algorithmen und sequentielle Prozesse, aber auch Brett- und Kartenspiele sind von dieser Art. Die nächste Stufe ist eine parallele Semantik, bei der verschiedene Aktivitäten gleichzeitig stattfinden können. Parallelableitungen in L-Systemen und einigen Graphtransformationsansätzen, das Schalten von Petri-Netzen sowie viele parallele Algorithmen und Prozesse lassen sich damit adäquat beschreiben. Während bei der parallelen Semantik die Parallelschritte nacheinander ablaufen, ist bei der nebenläufigen Semantik die zeitliche Beziehung zwischen verschiedenen Aktivitäten nicht festgelegt, es sei denn, sie müssen auf Grund kausaler Abhängigkeiten nacheinander stattfinden. Das neue Konzept der autonomen Transformationseinheiten wird durch zwei Beispiele illustriert. Zum einen werden Stellen/Transitionssysteme als Systeme autonomer Transformationseinheiten betrachtet, wobei jede Transition zu einer Transformationeinheit wird. Zum anderen wird ein Transportnetz modelliert, in dem Fahrzeuge Güter zu ihrem Bestimmungsort befördern, wobei sowohl die LKWs als auch die Pakete als autonome Transformationseinheiten modelliert werden. Die Arbeit wird mit einer kurzen Zusammenfassung und einem Ausblick abgeschlossen.

Es sei noch darauf hingewiesen, dass autonome Transformationen das Konzept der Transformationseinheiten verallgemeinern, wie er u.a. in [JKR05, Kus00, KK99, KKS97] untersucht wurde. Der wesentliche Unterschied ist, dass die bisherigen Abläufe von einer Haupteinheit gesteuert werden und keine Umgebungsänderungen stattfinden, die nicht von dieser Einheit kontrolliert werden. Erste Schritte in Richtung Verteiltheit bei Transformationseinheiten sind in [KK02] dokumentiert.

2 Autonome Transformationseinheiten

In diesem Abschnitt werden autonome Transformationseinheiten als ein neues Konzept zur Modellierung datenverarbeitender Systeme eingeführt, die sich aus unabhängigen, eigenständigen Komponenten zusammensetzen. Autonome Transformationseinheiten bilden eine Gemeinschaft in einer gemeinsamen, durch sie veränderbaren Umgebung.

Die Umgebungen werden der Einfachheit halber als Graphen vorausgesetzt, wobei diese aber sehr generisch verwendet werden. Sie umfassen alle möglichen Formen von Diagrammen. Je nach Bedarf können sie gerichtet oder ungerichtet sein. Durch geeignete Markierungen und Attribute sind sehr unterschiedliche Typen von Graphen möglich. Und da Graphen auch aus diversen Zusammenhangskomponenten und Teilgraphen bestehen können, sind Mengen, Multimengen und Listen von Graphen sowie anderweitig strukturierte Graphen als Umgebungen zulässig.

Jede autonome Transformationseinheit hat Ziele, verfügt über Regeln und besitzt eine Kontrollbedingung, mit der sie die Anwendung ihrer Regeln autonom steuert, um ihre Ziele zu erreichen. Regeln können auf Umgebungen angewendet werden und transformieren diese, so dass jede Regel semantisch eine binäre Relation auf Umgebungen definiert. Eine Kontrollbedingung besagt, welche Regelanwendungen auf die jeweils aktuelle Umgebung erlaubt sind. Semantisch beschreibt eine Kontrollbedingung ebenfalls eine binäre Relation auf Umgebungen. Ziele schließlich werden als Klassenausdrücke spezifiziert, die semantisch die Klasse von Umgebungen beschreiben, bei denen das Ziel erreicht ist.

Alle verfügbaren Umgebungen, Regeln, Kontrollbedingungen und Klassenausdrücke bilden einen Transformationsansatz, der mit den Regeln, Kontrollbedingungen und Klassenausdrücken insbesondere die syntaktischen Ingredienzien für autonome Transformationseinheiten zur Verfügung stellt. Außerdem werden Klassenausdrücke auch verwendet, um die initialen Umgebungen einer Systemmodellierung und die Gesamtziele zu definieren.

Ein *Transformationsansatz* $\mathcal{A} = (\mathcal{G}, \mathcal{R}, \mathcal{X}, \mathcal{C})$ besteht aus einer Klasse $\mathcal{G}$ von Graphen, die *Umgebungen* genannt werden, einer Klasse $\mathcal{R}$ von *Regeln*, einer Klasse $\mathcal{X}$ von *Klassenausdrücken* und einer Klasse $\mathcal{C}$ von *Kontrollbedingungen*. Jede Regel $r \in \mathcal{R}$ spezifiziert eine binäre Semantikrelation $\mathit{SEM}(r) \subseteq \mathcal{G} \times \mathcal{G}$. Jedes Paar $(G, H) \in \mathit{SEM}(r)$ ist eine *Regelanwendung* von r, wird auch *direkte Ableitung* genannt und durch $G \underset{r}{\Rightarrow} H$ bezeichnet. Jeder Klassenausdruck $X \in \mathcal{X}$ spezifiziert eine Menge von Umgebungen $\mathit{SEM}(X) \subseteq \mathcal{G}$ als Semantik. Jede Kontrollbedingung $C \in \mathcal{C}$ spezifiziert als Semantik eine binäre Relation $\mathit{SEM}(C) \subseteq \mathcal{G} \times \mathcal{G}$ von Umgebungen.

Ein *System autonomer Transformationseinheiten* $S = (A, I, T)$ besteht aus einer Menge autonomer Einheiten A, einem Klassenausdruck I, der die *initialen Umgebungen*, und einem Klassenausdruck T, der die *terminalen Umgebungen* bzw. *Gesamtziele* beschreibt.

Eine *autonome Transformationseinheit* ist ein Konstrukt $aut = (goal, rules, control)$, wobei $goal \in \mathcal{X}$ das *Ziel*, $rules \subseteq \mathcal{R}$ die *Regelmenge* und $control \in \mathcal{C}$ die *Kontrolle* ist.

Beispiel Stellen/Transitions-Systeme

Stellen/Transitions-Systeme sind eine häufig verwendete Art von Petri-Netzen, die sich als System autonomer Transformationseinheiten deuten lassen. Umgebungen können in diesem Fall die Stellen/Transitions-Netze mit Markierungen sein. Regeln sind die Transitionen, ihr Schalten definiert Regelanwendungen als Markierungsänderungen in der üblichen Weise. Als Klassenausdrücke kann man einzelne Markierungen nehmen, die sich selbst als Semantik beschreiben. Als Klassenausdruck wird außerdem der Standardausdruck *all* benötigt, der immer alle Umgebungen zulässt, und als einzige Kontrollbedingung wird die Standardbedingung *free* verwendet, die die Allrelation auf Umgebungen beschreibt und keinerlei Einschränkungen mit sich bringt.

Betrachtet man nun eine einzelne Transition t als autonome Transformationseinheit $aut(t) = (all, \{t\}, free)$, so wird ein S/T-Netz N mit Transitionsmenge T und initialer Markierung m_0 als System autonomer Transformationseinheiten $S(N, m_0) = (\{aut(t) \mid t \in T\}, m_0, all)$ modelliert.

Beispiel Transportnetz

Zur Illustration wird ein Beispiel aus der Transportlogistik skizziert, das allerdings sehr vereinfacht ist, um den Rahmen dieser Arbeit nicht zu sprengen.

Ein Transportnetz ist ein Graph, dessen Knoten die Standorte repräsentieren, an denen Pakete auf- und abgeladen werden können. Die Kanten repräsentieren Verbindungen dieser Standorte und sind mit den Zeiten gewichtet, die ein Abfahren dieser Strecken beansprucht. Die Abbildung 1 zeigt einen kleinen Ausschnitt mit den Städten Dortmund, Bremen, Hamburg und Hannover als mögliche Standorte. *LKWs* und *Pakete*, von denen es beliebig viele geben kann, werden als autonome Transformationseinheiten modelliert und nutzen dieses Transportnetz als Umgebung. Ausprägungen dieser Transformationseinheiten werden im Transportnetz als Knoten repräsentiert, die mit jeweils eindeutigen Kennziffern versehen sind. Die Abbildung zeigt ein Transportnetz mit zwei LKWs (1,2)

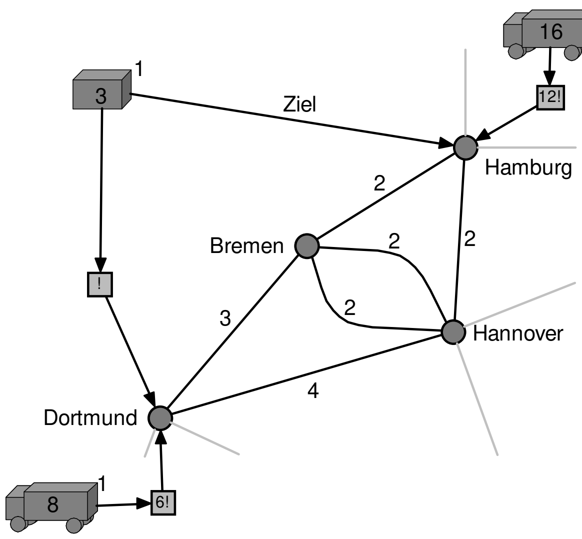

Abbildung 1: Ein Transportnetz durch einen Graphen repräsentiert

und einem Paket (1). Die LKW-Knoten sind mit einer Zahl markiert, die der Gesamtzeit entspricht, die der zugehörige LKW unterwegs sein darf. In unserem Beispiel darf also der LKW 1 acht Stunden fahren, während für LKW 2 die doppelte Zeit zulässig ist (weil er z.B. mit zwei Fahrern ausgestattet ist). Beide LKWs sind mit einem rechteckigen Tourknoten verbunden, der eine Zahl und ein Ausrufungszeichen enthält. Die Zahl entspricht der Kapazität des LKW, in unserem Beispiel ist dies das Zuladungsgewicht in Tonnen. LKW 1 darf also sechs Tonnen und LKW 2 zwölf Tonnen Ladung transportieren. Das Ausrufungszeichen markiert den Knoten als den aktuellen Tourknoten. Ein Paketknoten enthält als Markierung das Gewicht des Paketes. Auch sie ist mit einem rechteckigen Tourknoten verbunden, der wiederum mit dem Standort verbunden ist, an dem das Paket zum Versand bereitsteht. Der Standort an den das Paket versendet werden soll, wird durch eine mit "Ziel" markierte Kante spezifiziert. Wie bei der LKW-Einheit bezeichnet das Ausrufungszeichen im Tourknoten den aktuellen Tourknoten.

Die Transformationseinheit *LKW* besitzt zunächst eine Regel zur Planung der LKW-Tour. In Abbildung 2 ist diese Regel dargestellt. Durch die Anwendung dieser Regel erweitert ein LKW seine Tour, indem er die Fahrt zu einem benachbarten Standort in seine Tour einplant. Hierbei darf die Zeit zum Befahren der Strecke zwischen den Standorten nicht größer sein als die Zeit, die dem LKW noch zum Fahren bleibt.

Eine Paketeinheit besitzt eine dazu analoge Tourplanungsregel. Darüber hinaus besitzt sie jedoch noch eine zweite Regel, die weitere Tourabschnitte verhindert, falls das Ziel bereits erreicht ist. Aus Platzgründen sind diese Regeln hier nicht grafisch dargestellt. Mit den Tourplanungsregeln können beliebig viele Ausprägungen von LKWs und Paketen ihre jeweiligen Touren völlig unabhängig voneinander planen.

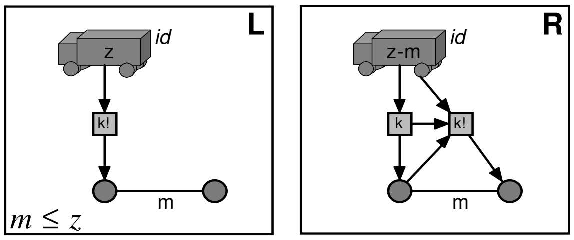

Abbildung 2: Planung einer LKW-Tour

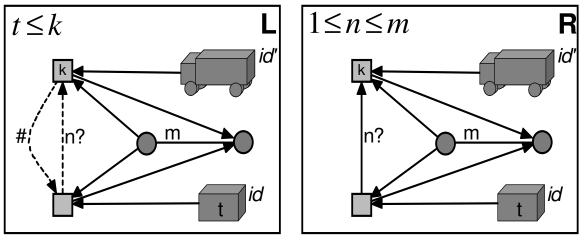

Abbildung 3: Paket-Angebot

Damit ein Paket überhaupt von einem Standort zum nächsten gelangen kann, besitzt eine Paketeinheit eine Regel, die es erlaubt, ein Angebot an einen vorbeifahrenden LKW abzugeben. Diese Regel ist in Abbildung 3 dargestellt. Falls ein LKW auf einem Tourabschnitt des Paketes fährt und die Kapazität des LKW auf diesem Tourabschnitt ausreicht, kann das Paket ein Angebot auf Mitnahme abgeben. Die gepunkteten Kanten zwischen den beiden Tourknoten in der linken Regelseite stellen negative Anwendungsbedingungen dar. Sie bedeuten, dass die Regel nicht angewendet werden kann, falls eine dieser Kanten zwischen den Knoten existieren.

Die LKW Einheit besitzt zwei weitere Regeln, die über den Umgang mit einem Paketangebot entscheiden. Die erste Regel (Abbildung 4) spezifiziert die Ablehnung des Angebots, angezeigt durch eine mit # markierte Kante. In diesem Fall kann das Paket kein weiteres Angebot abgeben, da die negative Anwendungsbedingung das Vorhandensein einer solchen Kante für die Abgabe eines Angebots verbietet. Die zweite Regel der LKW-Einheit beschreibt die Annahme des Angebots und damit die Mitnahme des Paketes für den betrachteten Tourabschnitt.

In den folgenden Abschnitten wird die Semantik von Systemen autonomer Transformationseinheiten in drei Varianten definiert. Die einfachste ist sequentiell und damit nur für Systeme geeignet, in denen zu einem Zeitpunkt nur ein Ereignis eintreten oder eine Aktion

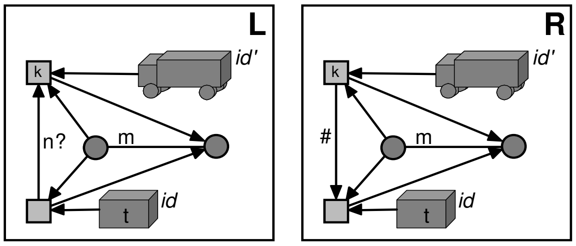

Abbildung 4: Angebot ablehnen

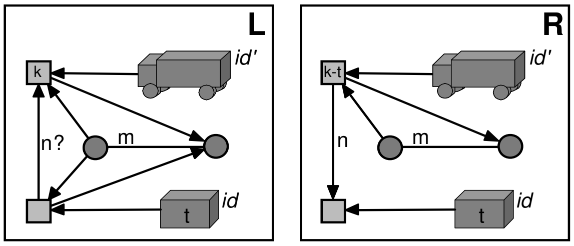

Abbildung 5: Paket mitnehmen

ausgeführt werden kann. Bei der zweiten Semantik können beliebig viele Aktivitäten parallel laufen. Die dritte schließlich berücksichtigt echte Nebenläufigkeit. Nur kausal voneinander abhängige Aktionen, wenn beispielsweise eine benötigt, was die andere herstellt, finden nacheinander statt. Die zeitliche Reihenfolge sonstiger Aktionen liegt nicht fest.

3 Sequentielle Ablaufsemantik

Da Regeln durch Anwendung direkte Ableitungsschritte erlauben, erhält man durch sequentielle Komposition eine erste einfache Ablaufsemantik für Systeme autonomer Transformationseinheiten. Es werden sowohl endliche als auch unendliche Abläufe betrachtet.

Sei $S = (A, I, T)$ ein System autonomer Transformationseinheiten. Dann ist ein *endlicher sequentieller Ablauf*, der auch *endliche Ableitung* oder *Berechnung* genannt wird, gegeben durch $(G_i)_{i \in [n]}$ mit $[n] = \{0, \ldots, n\}$ für $n \in \mathbb{N}$, wobei für jedes $i = 1, \ldots, n$ folgendes gelten muss:

> Es gibt eine autonome Transformationseinheit $aut_i = (goal_i, rules_i, control_i)$ und eine Regel $r_i \in rules_i$ mit $G_{i-1} \underset{r_i}{\Rightarrow} G_i$ und $(G_{i-1}, G_i) \in SEM(control_i)$.

Entsprechend ist ein *unendlicher sequentieller Ablauf* durch eine Folge $(G_i)_{i\in\mathbb{N}}$ gegeben, wobei für $i \in \mathbb{N}$ dasselbe wie im endlichen Fall gilt.

Abläufe sind also beliebige sequentielle Kompositionen von Regelanwendungen autonomer Transformationseinheiten des gegebenen Systems, wobei die jeweilige Regelanwendung durch die Kontrolle der aktuell agierenden Einheit erlaubt ist. Die Menge aller sequentiellen Abläufe wird mit $\mathit{SEQ}(A)$ bezeichnet. Entsprechend enthält $\mathit{SEQ}(A, I)$ alle Abläufe, die mit einer initialen Umgebung beginnen, und $\mathit{SEQ}(A, I, T) = \mathit{SEQ}(S)$ alle endlichen Abläufe, die zusätzlich in einer terminalen Umgebung enden.

Im letzteren Fall kann man statt der Ablaufsemantik auch eine Input-Output-Relation zuordnen, die die Systemberechnung ohne Zwischenschritte beschreibt: Für ein Paar gilt $(G, H) \in \mathit{REL}_{\mathit{SEQ}}(S)$, falls $(G_i)_{i\in[n]} \in \mathit{SEQ}(S)$ existiert mit $G = G_0$ und $H = G_n$. Aber auch für beliebige Abläufe macht die Zielspezifikation Sinn, denn man kann bei Abläufen $(G_i)_{i\in N}$ ja auch zwischendurch prüfen, ob T gerade erreicht ist: $G_{i_0} \in \mathit{SEM}(T)$?

Analog kann eine sequentielle Semantik für eine einzelne autonome Transformationseinheit $aut = (goal, rules, control)$ definiert werden, wenn man bedenkt, dass neben den Regelanwendungen der betrachteten Einheit auch andere Umgebungsänderungen – nämlich durch andere Einheiten – möglich sind.

Sei $\mathit{CHANGE} \subseteq \mathcal{G} \times \mathcal{G}$ eine binäre Relation auf Umgebungen, die alle Änderungen ausdrückt, die nicht durch aut verursacht werden. Sei $N = [n] = \{0, \dots, n\}$ für ein $n \in \mathbb{N}$ oder $N = \mathbb{N}$. Dann ist ein sequentieller aut-Ablauf eine Folge $(G_i)_{i\in N}$ derart, dass für alle $i \geq 1$ entweder $(G_{i-1}, G_i) \in \mathit{CHANGE}$ oder für ein $r \in rules$ $G_{i-1} \underset{r}{\Rightarrow} G_i$ und $(G_{i-1}, G_i) \in \mathit{SEM}(control)$. Die Menge der sequentiellen aut-Abläufe wird mit $\mathit{SEQ}_{\mathit{CHANGE}}(aut)$ bezeichnet.

Zwischen den sequentiellen Abläufen $\mathit{SEQ}(A)$ für eine Menge A autonomer Transformationseinheiten und eines ihrer Mitglieder $aut \in A$ gibt es einen engen Zusammenhang:

$$\mathit{SEQ}(A) = \mathit{SEQ}_{\mathit{SEQ}(A-\{aut\})}(aut).$$

Jeder sequentielle Ablauf ist also ein aut-Ablauf für jede autonome Transformationseinheit aut aus A und umgekehrt, wenn die sonstigen Änderungen der Umgebung gerade die sequentiellen Abläufe der anderen autonomen Transformationseinheiten sind.

Beispiel Stellen/Transitions-Systeme

Betrachtet man das System autonomer Transformationseinheiten $S(N, m_0)$, das zu einem S/T-System korrespondiert, dann ist eine Regelanwendung gerade das Schalten einer Transition, so dass sequentielle Abläufe gerade den Schaltfolgen des S/T-Systems entsprechen.

Beispiel Transportnetz

Die Tourplanung von Paket- und LKW-Einheiten können als sequentielle Abläufe in einem Transportnetz betrachtet werden. In Abbildung 6 werden Anfang und Ende eines Ablaufs für eine Tour des Pakets 1 gezeigt, welches von Dortmund aus über Bremen nach Hamburg transportiert werden möchte, während LKW 1 einen Tourabschnitt von Dortmund nach Hannover plant und LKW 2 von Hamburg nach Hannover.

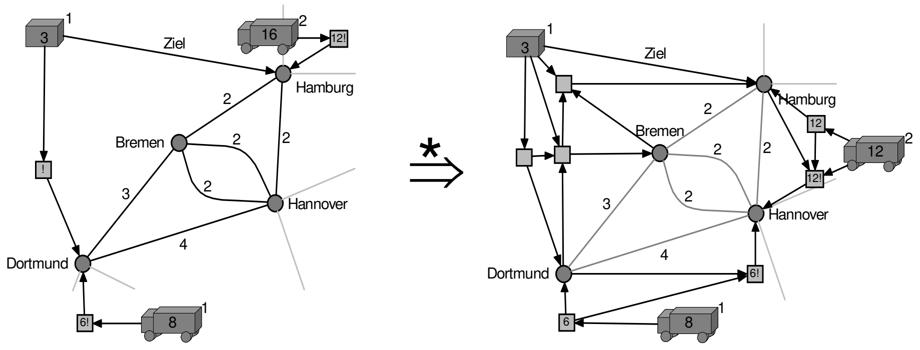

Abbildung 6: Ableitung

4 Parallele Semantik

Bei eigenständig agierenden Transformationseinheiten in einer gemeinsamen Umgebung ist es in vielen Fällen unrealistisch, dass immer nur eine Regelanwendung zu einem Zeitpunkt geschehen kann. Die tatsächlichen Abläufe in vielen Datenverarbeitungssituationen werden wesentlich besser widergespiegelt, wenn in verschiedenen Teilen der Umgebung Aktivitäten zeitgleich passieren und wenn Vorgänge, die sich gegenseitig nicht beeinflussen, parallel eintreten können.

Um solche parallelen Abläufe formal erfassen zu können, wird vorausgesetzt, dass nicht nur einzelne Regeln, sondern Multimengen von Regeln – also mehrere Regeln und einzelne davon mehrfach – auf Umgebungen anwendbar sein können. Sei zu diesem Zweck $\mathcal{A} = (\mathcal{G}, \mathcal{R}, \mathcal{X}, \mathcal{C})$ ein *Transformationsansatz mit Parallelität*, d.h. für jede endliche Multimenge R über $\mathcal{R}$ gibt es eine binäre Semantikrelation $SEM(R) \subseteq \mathcal{G} \times \mathcal{G}$. Statt $(G, H) \in SEM(R)$ kann auch in diesem Fall $G \underset{R}{\Rightarrow} H$ geschrieben und von einer direkten Parallelableitung gesprochen werden.

Parallele Abläufe lassen sich dann ähnlich wie sequentielle Abläufe definieren, wenn man dort jede vorkommende Anwendung einer Einzelregel durch die Anwendung einer Multimenge von Regeln ersetzt. Allerdings muss die Definition für das Einhalten der Kontrollbedingungen angepasst werden, denn es wäre unrealistisch anzunehmen, dass die Kontrolle einer Transformationseinheit auch steuert, welche Regeln anderer Transformationseinheiten parallel anwendbar sind. Entsprechend erhält man dann auch die korrespondierenden Mengen paralleler Abläufe $PAR(A), PAR(A, I)$ und $PAR(A, I, T) = PAR(S)$ und eine Input-Output-Relation $REL_{PAR}(S)$.

Beispiel Stellen/Transitions-Systeme

Definiert man für Multimengen von Transitionen eines S/T-Systems das parallele Schalten in der üblichen Weise, erhält man einen Transformationsansatz mit Parallelität. Für das System $S(N, m_0)$ entsprechen dann die parallelen Abläufe genau den Schaltfolgen von Multimengen von Transitionen.

Beispiel Transportnetz

Die Tourplanung von Paket- und LKW-Einheiten können aber auch als parallele Abläufe in einem Transportnetz betrachtet werden. Der in Abbildung 6 angedeutete Ablauf kann auch aus einer Sequenz der zwei Tourplanungsschritte des Pakets und beliebig dazu paralleler Tourplanung der beiden LKWs entstehen.

Im allgemeinen können sequentielle und parallele Abläufe sehr unterschiedliche Ergebnisse produzieren. Bei zellulären Automaten, bei denen alle vernetzten endlichen Automaten parallel schalten in Abhängigkeit von den aktuellen Zuständen seiner Nachbarn, würde das Schalten einiger den Kontext für die anderen verändern, so dass ihr späteres Schalten zu anderen Konfigurationen führte. Bei vielen anderen Methoden und Ansätzen wie Petri-Netze, Termersetzung und Graphtransformation verkleinert Parallelität die Schrittzahl, ändert aber nichts am Endergebnis, weil das parallele Ausführen von Aktivitäten auch in beliebiger Reihenfolge mit demselben Resultat durchgeführt werden kann. Dieses Phänomen wird *echte Nebenläufigkeit* genannt. Im Kontext von Transformationsansätzen mit Parallelität muss dafür folgendes gelten:

> Sei $R = R' + R''$ die Summe zweier Multimengen von Regeln und $G \underset{R}{\Rightarrow} X$ ein paralleler Ableitungsschritt. Dann existieren parallele Ableitungsschritte $G \underset{R'}{\Rightarrow} H$ und $H \underset{R''}{\Rightarrow} X$ für eine geeignete Umgebung H.

Da jede endliche Multimenge die Summe einzelner Elemente ist und die Summe von Multimengen kommutativ ist, folgt aus der echten Nebenläufigkeit, dass jeder Parallelschritt mit demselben Ergebnis in beliebiger Reihenfolge von Anwendungen der zugehörigen Einzelregeln sequentialisiert werden kann. Parallele Abläufe und ihre Sequentialisierungen werden (aus nebenläufiger Sicht) *äquivalent* genannt. Betrachtet man eine Äquivalenzklasse eines parallelen Ablaufs, d.h. alle Abläufe, die untereinander äquivalent sind, so lässt sich von je zwei Regelanwendungen nur noch dann die zeitliche Reihenfolge ermitteln, wenn die eine auf die andere warten muss. Sonst kommen sie sowohl parallel als auch in beliebiger Reihenfolge vor.

Für die Abläufe einer Menge autonomer Transformationseinheiten A gilt, dass $SEQ(A) \subseteq PAR(A)$ und dass es zu jedem Ablauf $s \in PAR(A)$ einen äquivalenten Ablauf $\bar{s} \in SEQ(A)$ gibt. Für ein System autonomer Transformationseinheiten S folgt daraus insbesondere

$$REL_{SEQ}(S) = REL_{PAR}(S).$$

5 Nebenläufige Semantik

Wie die sequentielle, so ist auch die parallele Ablaufsemantik nicht für alle Anwendungssituationen angemessen, denn von eigenständig und unabhängig voneinander agierenden Komponenten kann man nicht erwarten, dass sie immer gleichzeitig mit einem gemeinsamen Anfang und einem gemeinsamen Ende aktiv werden, wie es bei Parallelschritten der Fall ist. Wenn solche Komponenten weit voneinander entfernt sind oder gerade an völlig verschiedenen Aufgaben arbeiten, bei denen sie sich nicht in die Quere kommen, ist Gleichzeitigkeit vielleicht sogar gar nicht feststellbar, macht aber auf jeden Fall keinen Sinn. Ein zeitliches Nacheinander ist in solchen nebenläufigen und verteilten Systemen nur dann sicher gegeben, wenn eine Aktivität eine andere voraussetzt. Drückt man solche kausale Abhängigkeit durch eine gerichtete Kante zwischen den Aktivitäten aus, entsteht bei einem nebenläufigen Ablauf ein azyklischer Graph von Aktivitäten. Das führt zu dem Konzept nebenläufiger Abläufe in Systemen autonomer Transformationseinheiten. Dieselbe Idee liegt dem Prozessbegriff von Petri-Netzen zugrunde.

Sei $S = (A, I, T)$ ein System autonomer Transformationseinheiten über einem Transformationsansatz $\mathcal{A} = (\mathcal{G}, \mathcal{R}, \mathcal{X}, \mathcal{C})$ mit Parallelität. Dann besteht ein *nebenläufiger Ablauf* aus einer Startumgebung G_0 und einem azyklischen gerichteten Graphen $run = (V, E, lab)$ mit einer Knotenmenge V, einer Kantenmenge $E \subseteq V \times V$ und einer Knotenmarkierung $lab : V \to \mathcal{R}$, die den Knoten Regeln zuordnet. Für G_0 und run muss außerdem folgendes gelten:

1. Jeder Knoten von run ist durch einen Weg erreichbar von einem initialen Knoten aus, d.h. von einem Knoten ohne hineingehende Kanten.
2. Jedes vollständige Anfangsstück von run, d.h. jeder Teilgraph, der die initialen Knoten enthält und mit jedem Knoten alle Wege von initialen Knoten zu diesem, ist endlich oder enthält einen unendlichen Weg.
3. Für jedes vollständige Anfangsstück existiert ein paralleler Ablauf $(G_i)_{i \in N}$ für $N = [n]$, $n \in \mathbb{N}$ oder $N = \mathbb{N}$ mit einer bijektiven Beziehung zwischen den Knoten des Teilgraphen und den angewendeten Regeln, wobei letztere jeweils die Markierung des zugehörigen Knotens ist. Diese Beziehung bewahrt die kausale Abhängigkeit, d.h. die Regel, die die Quelle einer Kante des Teilgraphen markiert, wird in einem früheren Ableitungsschritt angewendet als die Regel, die das Ziel dieser Kante markiert.

Die erste Bedingung besagt, dass run keinen unendlichen Weg ohne Anfang enthalten kann, weil es sonst einen Weg gäbe, der durch keinen Ablauf widergespiegelt würde. Die zweite Bedingung bedeutet, dass immer nur endlich viele Knoten kausal voneinander unabhängig sind. Die dritte Bedingung garantiert die prinzipielle Ausführbarkeit von nebenläufigen Abläufen.

Beispiel Stellen/Transitions-Systeme

Zumindest für den Spezialfall der Bedingungs/Ereignis-Netze gibt es mit den Okkurrenznetzen einen anerkannten Prozessbegriff, der mit dem hier gegebenen große Ähnlichkeiten aufweist. Ersetzt man in einem Okkurrenznetz die Wege der Länge 2, die über eine Bedingung laufen, durch eine direkte Kante, entsteht ein nebenläufiger Ablauf im obigen Sinne.

Beispiel Transportnetz

Für das Beispiel der Transportnetze gilt, dass die Tourplanungen sowohl der LKWs als auch der Pakete nebenläufig geschehen können. Ebenso können verschiedene Paket-LKW-

Paare nebenläufig Verhandlungen führen.

Eine ausführliche Darstellung des Zusammenhangs zwischen parallelen und nebenläufigen Abläufen sprengt den Rahmen dieser Arbeit und muss deshalb auf eine Folgearbeit verschoben werden. Aber es sei darauf hingewiesen, dass im Falle echter Nebenläufigkeit eine enge Beziehung zwischen nebenläufigen Abläufen und kanonischen Ableitungen besteht, wie sie in [Kre78] für Graphtransformation auf der Basis des Doppelpushout-Ansatzes betrachtet wurden. Solche kanonischen Ableitungen repräsentieren die Äquivalenzklassen paralleler Ableitungen in eindeutiger Weise durch maximale Parallelität mit frühst möglicher Regelanwendung.

6 Konklusion

In dieser Arbeit haben wir mit dem System autonomer Transformationseinheiten ein neues regelbasiertes Konzept für die Modellierung von datenverarbeitenden Systemen vorgestellt, die sich aus verschiedenen Komponenten und Prozessen zusammensetzen und bei denen diese Komponenten und Prozesse verteilt sind, eigenständig arbeiten, aber auch kommunizieren und interagieren können. Mit den sequentiellen, parallelen und nebenläufigen Abläufen ist die operationelle Semantik solcher Systeme in drei Stufen eingeführt worden, wobei sequentielle und parallele Abläufe alle Systemaktivitäten in eine feste zeitliche Reihenfolge bringen, während nebenläufige Abläufe nur kausale Abhängigkeiten in der Abfolge fixieren. Der Ansatz verwendet Graphen und Transformationsregeln auf Graphen, so dass eine Art visuelle Modellierung möglich ist, wie das Beispiel der Transportnetze illustriert. Gleichzeitig ist der Ansatz so flexibel und allgemein gehalten, dass sich andere Modellierungsansätze einbetten lassen und auf diese Weise semantische Vergleiche zwischen verschiedenen Modellierungsmethoden möglich werden. Das Beispiel der Petri-Netze gibt einen ersten Hinweis in diese Richtung. In zukünftigen Untersuchungen muss das substanziiert werden. Ohnehin mussten in dieser ersten Darstellung autonomer Transformationseinheiten noch viele Fragen unbeantwortet und viele Aspekte unberücksichtigt bleiben. Dazu gehören unter anderem folgende Punkte:

1. Die sequentiellen und parallelen Abläufe bestehen aus Anwendung einzelner Regeln oder Multimengen von Regeln, so dass diese Stufe der Semantik zu markierten Transitionssystemen korrespondieren, wie sie häufig zur semantischen Fundierung von Kommunikation und Verteiltheit herangezogen werden. Dieser Zusammenhang muss näher untersucht werden.

2. Wie am Ende von Abschnitt 5 angedeutet, gibt es eine enge Beziehung zwischen nebenläufigen Abläufen und kanonischen Ableitungen, die spezielle parallele Abläufe sind. Eine genauere Erforschung dieser Beziehung wäre interessant.

3. Ein wesentlicher Aspekt herkömmlicher Transformationseinheiten ist die Strukturierung, die dadurch erreicht wird, dass Transformationseinheiten andere importieren und mit Teilaufgaben beauftragen können. Die autonomen Transformationsein-

heiten sind der Einfachheit halber erst einmal ohne zusätzliche Strukturierung eingeführt. Es macht aber auch für autonome Transformationseinheiten Sinn, sich Hilfe zu verschaffen und große Einheiten in kleinere aufzuteilen, so dass zukünftige Überlegungen auch strukturierte autonome Transformationseinheiten vorsehen sollten.

4. Die Hauptaufgabe zukünftiger Forschung im Hinblick auf autonome Transformationseinheiten wird aber die Kontrolle betreffen. Zum einen sind spezifische Kontrollmechanismen zu untersuchen, die für Autonomie und Selbststeuerung besonders geeignet sind. Dazu werden voraussichtlich insbesondere Konzepte gehören, die Teile der Umgebung evaluieren und vielleicht sogar abhängig vom Ziel der Einheit die jeweils nächstbeste Aktion wählen. Zum anderen wird die Kontrolle, die der Einfachheit halber nur für Einzelschritte definiert ist, für längere Abläufe eingeführt werden müssen, wie es bei den nichtautonomen Transformationseinheit bereits der Fall ist.

5. Die Signifikanz und Brauchbarkeit autonomer Transformationseinheiten als Modellierungsansatz werden sich vor allem auch durch ein Spektrum von Fallstudien erweisen, die von Spielen über Agentensysteme und Kolonien künstlicher Ameisen bis zu konventionellen Ansätzen zur Prozessmodellierung reichen.

Alle Interessierten mögen sich eingeladen fühlen, an der Klärung der offenen Fragen mitzuwirken.

Literatur

[BN98] Franz Baader und Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[EEKR99] Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski und Grzegorz Rozenberg, Hrsg. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools*. World Scientific, 1999.

[EKMR99] Hartmut Ehrig, Hans-Jörg Kreowski, Ugo Montanari und Grzegorz Rozenberg, Hrsg. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 3: Concurrency, Parallelism, and Distribution*. World Scientific, 1999.

[FHS04] Michael Freitag, Otthein Herzog und Bernd Scholz-Reiter. Selbststeuerung logistischer Prozesse – Ein Paradigmenwechsel und seine Grenzen. *Industrie Management*, 20(1):23–27, 2004.

[JKR05] Dirk Janssens, Hans-Jörg Kreowski und Grzegorz Rozenberg. Main Concepts of Networks of Transformation Units with Interlinking Semantics. In Hans-Jörg Kreowski, Ugo Montanari, Fernado Orejas, Grzegorz Rozenberg und Gabriele Taentzer, Hrsg., *Formal Methods in Software and System Modeling*, Lecture Notes in Computer Science Vol. 3393, Seiten 325–342. Springer Verlag, 2005.

[KK99] Hans-Jörg Kreowski und Sabine Kuske. Graph Transformation Units with Interleaving Semantics. *Formal Aspects of Computing*, 11(6):690–723, 1999.

[KK02] Peter Knirsch und Sabine Kuske. Distributed Graph Transformation Units. In Andrea Corradini, Hartmut Ehrig, Hans-Jörg Kreowski und Grzegorz Rozenberg, Hrsg., *Proc. First International Conference on Graph Transformation (ICGT)*, Lecture Notes in Computer Science Vol. 2505, Seiten 207–222, 2002.

[KKS97] Hans-Jörg Kreowski, Sabine Kuske und Andy Schürr. Nested graph transformation units. *International Journal on Software Engineering and Knowledge Engineering*, 7(4):479–502, 1997.

[Kre78] Hans-Jörg Kreowski. *Manipulationen von Graphmanipulationen*. Dissertation, Univ. Berlin, 1978. Dissertation, Fachbereich Informatik.

[Kuh02] Axel Kuhn. Prozessketten – ein Modell für die Logistik. In Hans-Peter Wiendahl, Hrsg., *Erfolgsfaktor Logistikqualität*, Seiten 58–72. Springer Verlag, 2002.

[Kus00] Sabine Kuske. *Transformation Units—A Structuring Principle for Graph Transformation Systems*. Dissertation, University of Bremen, 2000.

[PRS98] Gheorghe Păun, Grzegorz Rozenberg und Arto Salomaa. *DNA Computing — New Computing Paradigms*. Springer Verlag, 1998.

[Rei98] Wolfgang Reisig. *Elements of Distributed Algorithms – Modeling and Analysis with Petri Nets*. Springer Verlag, 1998.

[Roz97] Grzegorz Rozenberg, Hrsg. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*. World Scientific, Singapore, 1997.

[RS97] Grzegorz Rozenberg und Arto Salomaa, Hrsg. *Handbook of Formal Languages, Vol. 1–3*. Springer Verlag, 1997.

[Sch02] August-Wilhelm Scheer. *Vom Geschäftsprozeß zum Anwendungssystem*. Springer Verlag, 2002.

[Wei99] Gerhard Weiss, Hrsg. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[WP98] T. White und B. Pagurek. Towards Multi-Swarm Problem Solving in Networks. In Y. Demazeau, Hrsg., *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, Seiten 333–340. IEEE Press, 1998.