Autonome Transformationseinheiten zur regelbasierten Modellierung vernetzter logistischer Prozesse

Karsten Hölscher, SFB 637 „Selbststeuerung logistischer Prozesse“
Dr.-Ing. Renate Klempien-Hinrichs, Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft
Dr.-Ing. Peter Knirsch, SFB 637 „Selbststeuerung logistischer Prozesse“
Prof. Dr.-Ing. Hans-Jörg Kreowski, Professur Theoretische Informatik
Dr.-Ing. Sabine Kuske, SFB 637 „Selbststeuerung logistischer Prozesse“
Universität Bremen*

1. Einleitung

Datenverarbeitende Systeme bestehen heutzutage meist aus vielen Komponenten, die logisch und physikalisch 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 durchführen, doch erst ihre Vernetzung und ihr Zusammenwirken machen das Gesamtsystem aus und führen zur Erledigung der gestellten Aufgaben. Durch diesen Aufbau heutiger datenverarbeitender Systeme entsteht aber auch eine problematische Komplexität, die von der Heterogenität, der Verteiltheit und der nicht trivialen Kommunikation und Interaktion der Komponenten herrührt. Dieser Komplexität zu begegnen ist eine Aufgabe, der man sich beim Entwurf, der Analyse und der Weiterentwicklung von Systemen stellen muss und die nicht allein durch die verbesserte Rechenleistung heutiger Computer kompensiert werden kann. Eigenständig handelnde, verteilte Komponenten stellen aber nicht nur ein Problem dar, sie werden auch als Chance gesehen und als neues Paradigma, wie es sich in der wachsenden Popularität von Softwareagenten zeigt.

Als grundlegend für den Entwurf, die Analyse und die Weiterentwicklung von Systemen erweist sich eine geeignete Modellierung. Viele der aktuellen Modellierungsmethoden, wie die UML (siehe [OMG]), haben Schwächen im Bereich der semantischen Fundierung, wie zum Beispiel in [Kus01] gezeigt.

Wir schlagen daher in dieser Arbeit das Konzept der autonomen Einheiten für die semantisch fundierte 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 werden soll.

2. Selbststeuerung als neues Paradigma

Die Motivation, autonome Einheiten als Modellierungskonzept einzuführen, stammt aus dem Sonderforschungsbereich 637 „Selbststeuerung logistischer Prozesse – Ein Paradigmenwechsel und seine Grenzen“ (siehe z.B. [FHS04]), in dem interdisziplinär

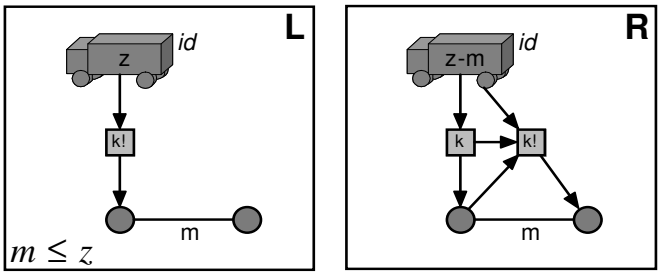
* Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Sonderforschungsbereichs 637 “Selbststeuerung logistischer Prozesse” und von der Europäischen Union im Rahmen des RTN Projekts SegraVis “Syntactic and Semantic Integration of Visual Modelling Techniques” unterstützt.

E. Müller and B. Spanner-Ulmer (Eds.): Vernetzt planen und produzieren (VPP 2006), pp. 113-118, 2006

die Frage verfolgt wird, ob und unter welchen Umständen Selbststeuerung gegenüber der herkömmlichen Fremdsteuerung logistischer Prozesse Vorteile hinsichtlich Zeit, Kosten und Robustheit bringt. Kann man nicht, um die Planungs- und Berechnungskomplexität zum Beispiel in großen Transport- und Produktionsnetzen in den Griff zu bekommen, von der zentralen Steuerung abrücken und den einzelnen logistischen Objekten das Vermögen mitgeben, in gewissem Rahmen Entscheidungen selbstständig zu treffen? Inwieweit lässt sich die Effizienz und die Robustheit solcher Systeme durch das Vorhanden sein von Selbststeuerung steigern?
Die Leitidee bei der Modellierung mit autonomen Transformationseinheiten ist, die Möglichkeit zur Selbststeuerung in die Modellierung von vernetzten logistischen Prozessen zu integrieren und damit insbesondere einen Rahmen zu schaffen, in dem Selbststeuerungsmechanismen semantisch fundiert untersucht und miteinander verglichen werden können.

3. Modellieren mit Graphen und Graphtransformation

Graphische Repräsentationen werden häufig genutzt, um Ideen, Konzepte und deren Abhängigkeiten zu verdeutlichen. Eine besondere Rolle spielt dabei der Datentyp Graph: ein aus Knoten und Kanten bestehendes Konstrukt. Stellt ein Graph also nun eine gegebene Situation in einem System dar, so gestattet es die Graph-transformation (siehe [Roz97]) dieser statischen Sicht eine dynamische hinzuzufügen, indem der Graph über so genannte (Graph-)Regeln regelbasiert geändert wird. Um die Dynamik in einem gegebenen System sinnvoll modellieren zu können, d.h. mögliche Folgezustände eines Systems einzugrenzen, bedarf es neben der allgemeinen Regelanwendung noch einiger weiterer Kontrollmechanismen, die in der autonomen Transformationseinheit zusammengefasst sind. Es sei noch darauf hingewiesen, dass autonome Transformationseinheiten das Konzept der Trans-formationseinheiten verallgemeinern, wie es u.a. in [KK99] untersucht wurde. Autonome Transformationseinheiten stellen einen generischen Ansatz dar, um Regelmengen zu strukturieren und Prozesse zu modellieren. Formal sind die Konzepte, die man dafür braucht, in dem *Transformationsansatz* $\mathcal{A} = (\mathcal{G}, \mathcal{R}, \mathcal{X}, \mathcal{C})$ zusammengefasst. Er 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 dabei eine binäre Semantikrelation SEM(r) $\subseteq \mathcal{G} \times \mathcal{G}$, d.h. SEM($r$) gibt an, welche Graphen mittels r zu welchen Graphen transformiert werden können. Die folgende Grafik zeigt ein Beispiel für eine solche Regel. Beim Anwenden dieser Regel wird ein Vorkommen des linken Graphen durch eines des rechten ersetzt, wobei Knoten die sowohl in der linken als auch in der rechten Regelseite vorkommen erhalten bleiben. Knoten die nur in der linken Seite vorkommen werden gelöscht, Knoten, die ausschließlich in der rechten Seiten vorkommen werden hinzugefügt. Jedes Paar (G, H) $\in$ SEM(r) ist eine *Regelanwendung* von r, wird *direkte Ableitung* genannt und durch $G \Rightarrow_r H$ bezeichnet. Jeder Klassenausdruck $X \in \mathcal{X}$ spezifiziert eine Menge von Umgebungen SEM(X) $\subseteq \mathcal{G}$ als Semantik.



Jede Kontrollbedingung $C \in \mathcal{C}$ spezifiziert als Semantik eine binäre Relation auf Umgebungen SEM(C) $\subseteq \mathcal{G} \times \mathcal{G}$.
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 *Kontrollkomponente* ist.
Ein *System autonomer Transformationseinheiten S = (A, I, T)* besteht aus einer Menge autonomer Einheiten *A*, Klassenausdrücken *I* und *T* für die *initialen* und die *terminalen Umgebungen,* die die *Gesamtziele* beschreiben.
Die intuitive Idee für die Definition einer nebenläufigen Semantik eines solchen Systems von autonomen Transformationseinheiten ist, dass alle autonomen Transformationseinheiten ihre lokalen Regeln *rules* zwar nebenläufig *aber* in einer Reihenfolge gemäß ihrer Kontrollkomponenten auf einen gemeinsamen Umgebungsgraph anwenden.
Für den Fall, dass die autonomen Transformationseinheiten sequentiell arbeiten definiert man: Ein *endlicher sequentieller Ablauf*, der auch *endliche Ableitung* oder *Berechnung* genannt wird, ist gegeben durch $(G_i)_i \in [n]$ mit $[n] = \{0,..., n\}$ für $n \in \mathbb{N}$, wobei für jedes $i = 1,..., n$ folgendes gelten muss: Es gibt eine autonome Transformationseinheit $aut_i = (goal_i, rules_i, control_i)$ und eine Regel $r_i \in rules_i$ für die gilt: $G_{i-1} \Rightarrow r_i\ G_i$ und $(G_{i-1}, G_i) \in$ SEM($control_i$).
Der wesentliche Unterschied zu den bekannten Transformationseinheiten 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 Parallelität, Nebenläufigkeit und Verteiltheit bei Transformationseinheiten sind in [KK02, JKR05] dokumentiert.

4. Anwendungsszenarien

Das neue Konzept der autonomen Transformationseinheiten wird hier nur kurz durch zwei Beispiele illustriert. Zum einen werden Stellen/Transitionssysteme oder kurz S/T-Systeme (siehe [Rei98] für verteilte Systeme) als Systeme autonomer Transformationseinheiten betrachtet, wobei jede Transition zu einer eigenen autonomen Transformationseinheit 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. Weitere Arbeiten zu autonomen Transformationseinheiten sind in [Kreo05] und in [KKK06] zu finden.

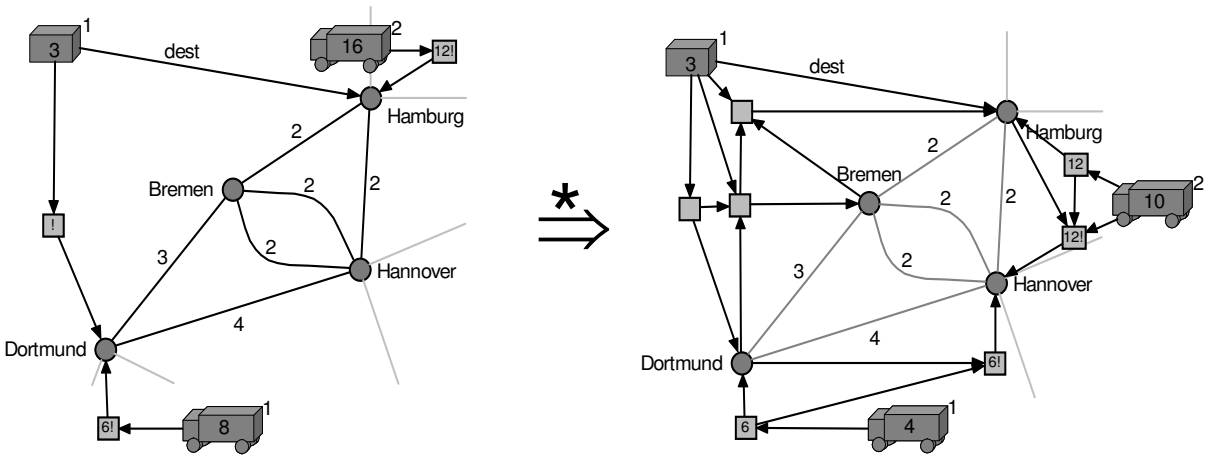
S/T-Systeme

S/T-Systeme sind eine häufig verwendete Art von Petri-Netzen. Sie bestehen im einfachsten Fall aus einer Menge von Stellen, die Marken aufnehmen können, einer Menge von Transitionen, gerichteten Kanten zwischen Stellen und Transitionen oder Transitionen und Stellen und einer initialen Markierung der Stellen. Transitionen, die für die Dynamik im System zuständig sind, lassen sich als System autonomer Transformationseinheiten deuten. Umgebungen können in diesem Fall die S/T-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 dann 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.

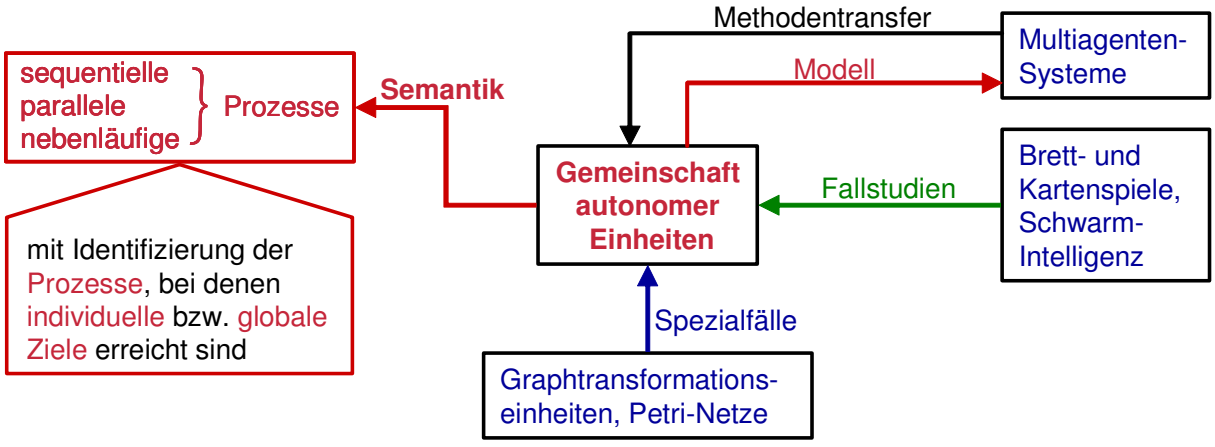
Transportnetz

Das Pickup-and-Delivery-Problem spielt in der Logistik eine bedeutende Rolle. Eine Anzahl von Transportern müssen Güter an verschiedenen Quellen abholen und an definierten Senken abliefern. Dabei sind einige Restriktionen wie Zeitfenster, verfügbarer Laderaum usw. zu beachten. Es hat sich u.a. in [KKK02] herausgestellt, dass dieses Problem sehr gut mit graphtransformatorischen Methoden zu modellieren ist. Auf der linken, unten stehenden Grafik ist die Ausgangssituation des Problems als Graph visualisiert:



Es ist ein Kartenausschnitt dargestellt mit den Städten Dortmund, Bremen, Hannover und Hamburg als Knoten und den benötigten Fahrzeiten als Kantenmarkierungen. Der LKW 1 mit freier Fahrzeit 8 befindet sich in Tour 6 in Dortmund, ein anderer LKW 2 mit freier Fahrzeit 16 in Hamburg. Das Stückgut 1 mit Gewicht 3 befindet sich in Dortmund und möchte nach Hamburg. Es ist noch in keiner Tour vorgesehen. Nun können die LKW unter Berücksichtigung ihrer maximalen Fahrzeit Touren planen. Dazu verwenden sie die weiter oben abgebildete Graphregel. Auch die weiter autonome Transformationseinheit des Stückguts hat Regeln und eine Kontrollkomponente, die es gestatten, eine Tour „autonom“ zu planen. Koinzidieren Wegstrecken von LKW und Gut, kann das Gut transportiert werden. Ein mögliches Ergebnis von wiederholten Regelanwendungen der Tourenplanung ist rechts in der Grafik abgebildet.

5. Zusammenfassung und Ausblick



Die oben abgebildete Grafik visualisiert die Ergebnisse und das Vorgehen der Untersuchung von autonomen Transformationseinheiten zur Modellierung von selbststeuernden Prozessen: Das bekannte Konzept der Transformationseinheiten wurde zu autonomen Transformationseinheiten erweitert und mit einer sequentiellen, einer parallelen und einer nebenläufigen Semantikdefinition versehen. Erste Fallstudien und Simulationen belegen die Angemessenheit der Modellierung sowie die Vorteile des gewählten visuellen Ansatzes. Vergleicht man autonome Transformationseinheiten mit Petri-Netzen oder Multiagentensystemen, zeigt sich der generelle Charakter dieser formalen Modellierungsmethode.

Literatur

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

[JKR05] Dirk Janssens, Hans-Jörg Kreowski, and Grzegorz Rozenberg. Main concepts of networks of transformation units with interlinking semantics. In Hans-Jörg Kreowski, Ugo Montanari, Fernado Orejas, Grzegorz Rozenberg, and Gabriele Taentzer, editors, Formal Methods in Software and System Modeling, Lecture Notes in Computer Science Vol. 3393, pages 325–342. Springer Verlag, 2005.

[KKK06] Karsten Hölscher, Peter Knirsch, Hans-Jörg Kreowski: Modelling Transport Networks by Means of Autonomous Units. In H.-D. Haasis, H. Kopfer, J. Schönberger, editors, *Operations Research Proceedings 2005*, pages 399-404. Springer, 2006.

[KKK02] Renate Klempien-Hinrichs, Peter Knirsch, Sabine Kuske: Modeling the Pickup-and-Delivery Problem with Structured Graph Transformation. In Hans-Jörg Kreowski, Peter Knirsch, editors, Proc. APPLIGRAPH Workshop on Applied Graph Transformation (Satellite Event of ETAPS 2002), pages 119-130. 2002.

[KK02] Peter Knirsch and Sabine Kuske. Distributed graph transformation units. In Andrea Corradini, Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, Proc. First International Conference on Graph Transformation (ICGT), Lecture Notes in Computer Science Vol. 2505, pages 207–222, 2002.

[KK99] Hans-Jörg Kreowski and Sabine Kuske. Graph transformation units with interleaving semantics. Formal Aspects of Computing, 11(6):690–723, 1999.

[Kus01] Sabine Kuske: A Formal Semantics of UML State Machines Based on Structured Graph Transformation. In Martin Gogolla, Cris Kobryn, editors, UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, volume 2185 of Lecture Notes in Computer Science, pages 241-256. 2001.

[Kreo05] Hans-Jörg Kreowski: Autonomous Units to Model Cooperating Logistic Processes: Basic Features. In K.S. Palwar et al., editor, *Proceedings of the 10th International Symposium on Logistics (ISL 2005), Lisbon*, pages 377-380. 2005.

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

[OMG] OMG: Unified Modeling Language – Resource Page. http://www.uml.org

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