Modelling Transport Networks by Means of Autonomous Units*

Karsten Hölscher, Peter Knirsch, and Hans-Jörg Kreowski

University of Bremen, Department of Mathematics and Computer Science
(hoelscher,knirsch,kreo)@informatik.uni-bremen.de

Summary. The concept of autonomous units to model distributed logistic processes and their interactions in a transport network is introduced. Autonomous units provide a general approach with rigorous semantics that allow the visual modelling of logistic processes in the transport domain in a systematic and structured way. Differing from existing models it especially incorporates the specification of autonomous or self-controlled behaviour of the participating actors. It means that the respective actions are not always predefined but allow for autonomous choice. By example in this paper a negotiation based approach is introduced. Due to this approach being formal and well-defined it supports testing and verification of required properties of the modelled systems at the level of specification.

1 Introduction

Transport logistics deals with the problems of how to transport load from one place to another while minding a set of constraints. Time frames for the delivery have to be kept in mind. The fleet size is restricted and so are the drivers capacities. In general not only the feasibility of the transport is of importance but also economic constraints. It is a well-known result from graph theory and complexity theory that those scheduling problems are hard to solve. The travelling sales person problem [LK73], for instance, is NP-complete although it seems to be quite simple compared to realistic scenarios. Having small scheduling problems and an idealised environment, exact solution can be computed in time. But if schedules become large, the runtime of such exact algorithms increases dramatically and make them practically not applicable. As a result, it is most likely that there is no efficient, exact algorithm computing such tours

* Research partially supported by the EC Research Training Network SegraVis (Syntactic and Semantic Integration of Visual Modeling Techniques) and by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 *Autonomous Cooperating Logistic Processes — A Paradigm Shift and its Limitations.*

in a reasonable time. But concerning the transport logistics it is the everyday business of a carrier to schedule trucks that pickup and deliver loads, and return to a depot afterwards. Heuristics that compute good solutions instead of optimal are the way out of this dilema. In [SS95] an introduction to the pickup and delivery problem can be found.

Today the structural and dynamic complexity of transport networks is increasing. The demands for transports are hardly predictable. If demand changes occur many plans are invalidated and the scheduling has to start again. Central planning is a bottleneck in the decentralised global world. A new challenge is to pass autonomy to the actors that have capabilities to adapt to changes at runtime. This is the scope of the Collaborative Research Centre CRC 637 *Autonomous Cooperating Logistic Processes – A Paradigm Shift and its Limitations*.

In this work a methodology is sketched to formally model transport networks by means of autonomous units that allow for autonomous adaptations. Although formal modelling seems to be extra work load in business it has many advantages. Using formal models one can specify all processes that are valid in a certain transport network where processes are regarded as sequences of operations. From all valid processes the best can be chosen. A model additionally facilitates the understanding of the processes especially if it has a visual representation. A model allows fast adaptations and algorithms can easily be derived thereof.

The concept of autonomous units generalises graph transformation units as studied in Kreowski, Kuske, and Schürr [KK99, KKS97] to structure large rule-based systems. It is a rule-based instantiation of the idea of agents and agent systems (see, e.g., [WJ94]) as first introduced in Knirsch and Kreowski in [KK00]. Graph transformation (see, e.g., [Roz97]) is a well-suited rule based mechanism to change graphs in a well-defined way.

An autonomous unit may represent any active component of a logistic system. In the particular context of transport logistics, it represents a vehicle, load, or even an RFID tag. Autonomous units have access to a common environment, in which they may cooperate or compete. Depending on the application domain such an environment can consist of all relevant places, e.g. cities, ports, stations, airports, etc., and relations between them, e.g. roads, railways, waterways, and communication channels. Additionally, in the pickup and delivery scenario the loads and available vehicles are part of the environment.

The autonomous units define the operational capabilities of the components. They run in a potentially non-deterministic way. In general they have a choice when performing the next action. Each unit controls itself autonomously to cut down this non-determinism. It is not controlled from outside. How the choice of the next action is done depends on the type of autonomy specified in the autonomous unit.

2 Autonomous Units

An *autonomous unit* is defined as $unit = (g, U, P, c)$ where g is a *goal* (formulated in a proper logic or language), U is a set of identifiers naming *used autonomous units* (that are imported in order to use their capabilities), P is a set of rules specifying the operational capabilities, and c is some control condition, restricting the possible orders of actions.

The goal g describes what the unit is trying to achieve or what is meant to become true. An autonomous unit acts in a specific environment, which it may change by choosing a suitable rule from P and applying it. Via these changes a unit may act directly towards the given goal or it may communicate with other units (on which actions it might depend). It may also use other units and their functionalities from U. The unit is considered autonomous in the sense that the next action is selected non-deterministically and not controlled from outside the unit. The simplest form would be to let the unit randomly decide on the next action. A more sophisticated form of autonomy can be achieved by using the control condition. The control condition may be very restricting and thus eliminating the non-determinism completely. It may as well be less restrictive to leave some room for non-deterministic choice. A typical kind of control conditions forces the order of rule applications or of calls of helping units.

3 Basic Modelling of Transport Networks

In the context of this work the environment of autonomous units is a transport network, consisting of places like depots or airports connected by different relations like roads or railways. Such a network of places and relations is naturally visualised as graphs with nodes representing places and edges representing relations. For this reason the rules of autonomous units are here graph transformation rules. For the pickup and delivery scenario more rules are specified in [KKK02].

Loads have to be picked up at certain places and delivered to other ones. In this very basic network example, the only mode of transport are trucks. Autonomous units are assigned to each of the trucks and loads, respectively, modelling their capabilities and autonomous behaviour.

In a first step a truck unit plans its tour for the day. How this is done is not nearer specified here. Graph transformation rules select non-deterministically (and currently regardless of a concrete goal) place nodes as part of the tour and mark them by inserting special tour nodes. This is done by all truck units that are used in the model. Figure 1 shows an excerpt of a transport network with a tour of one truck and one load. The tour of the truck (represented by the truck-shaped node) is planned to start in `Dortmund` and lead to `Hamburg` via `Bremen`. The tour is visualized by square nodes that are connected (by dotted lines) to the place nodes (depicted as circles) and the truck. The direction of the edges connecting the places with the tour node determines the direction of this tour section. Analogously the direction of the edges connecting the places with the load (represented by the rectangular node) determine the pickup and the delivery place. In the example the load has to be picked up in `Hanover` and delivered to `Hamburg`. The solid, undirected edges represent the roads between the places.

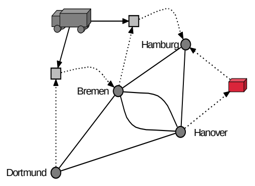

Fig. 1. Transport Network Graph

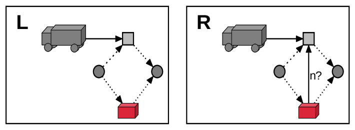

Fig. 2. Rule for a package offer

If a situation is found, where source and target of a tour section coincide with source and target places of a load, a negotiation for transport may commence. It is initiated by a package unit which contains a rule as shown in Figure 2. A graph transformation rule can be applied if the situation specified in its left-hand side `L` can be found in the environment. The counterparts of items that are only present in `L` but not in the right-hand side `R` are deleted from the environment. Items that are only present in `R` are added to the environment. Thus the application of the rule in Figure 2 yields a new edge connecting the load with the tour node of a truck. The edge is labelled with `n`?, meaning that the load offers the truck a price `n` for being transported.

The truck unit in turn has a rule that reacts to this new situation. It checks for an incoming `n`? edge, and may either accept or reject the offer. This depends on the internal structure of the truck unit. It could e.g. be possible that two packages make a price offer for transport for that tour section, so that the truck may choose the higher offer. In our first basic approach the truck unit decides non-deterministically and regardless of its goal whether to accept or reject an offer. The corresponding rule is depicted in Figure 3. It shows two right-hand sides `R1` and `R2` for the left-hand side `L`. This is a shorthand notation for two rules with the same left-hand side. In both cases `L` specifies a situation where a truck received an offer from a package for a tour section. In `R1` the offer is accepted by flipping the edge and relabelling it with `n`!. In `R2` the offer is rejected by deleting the offer edge. The truck units may also reschedule their tours. This involves deleting those tour nodes, that are not necessary to transport all packages with accepted offers. More formally, the control condition of a truck unit is specified in the following way: *plan_tour***;(accept_offer|reject_offer|reschedule_tour)!*

This means that the unit applies the rule *plan_tour* arbitrarily often followed by a choice of three rules. Here offers may be accepted or rejected as explained above or the tour be rescheduled. This choice is iterated as long as

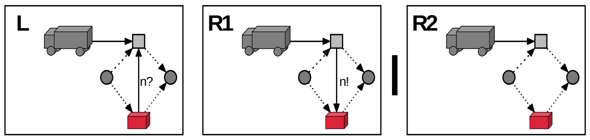

Fig. 3. Rule for accepting or rejecting an offer

possible, i.e. until no offers and also no empty tour sections remain in the environment. The control condition of a package unit is specified as: *make_offer!*

This means that the unit applies the rule *make_offer* as long as possible, i.e. until the corresponding load is scheduled to be transported from the pickup place to the delivery point.

4 Semantics of Autonomous Units

The semantics of the example transport network is given as environment transformations which are composed of rule applications and actions of used units obeyeing the control condition. This operational semantics provides a description of a simulation of the modelled processes and their cooperation.

Let ENV be the set of environments, and *unit* be one autonomous unit of a given system *Net* of autonomous units. Furthermore let $CHANGE(unit) \subseteq ENV \times ENV$ be a binary relation of environments describing the changes in the environment that can occur in addition to the changes *unit* can perform while acting autonomously. Then a **computation** of *unit* is defined as a sequence of environments $E_1, \ldots, E_k$, such that (E_i, E_{i+1}) for $i = 1, \ldots, k-1$ is obtained by applying a rule of *unit* or by calling a used unit according to the control condition. In order to account for changes not performed by *unit*, it may also be from *CHANGE(unit)*. Such a computation yields the input/output pair (E_1, E_k). The set of all these pairs is called the **semantic relation** *SEM(unit)* of one *unit*. Analogously a **computation** of *Net* is a sequence $(E_1, \ldots, E_k)$ if it is a computation of every unit of *Net*. This sequence describes the interaction of all the units with each other during a single run of the system, yielding the input/output pair (E_1, E_k) of a system run. The set *SEM(Net)* of all these pairs is called the **semantic relation** of *Net*.

This semantics definition induces a proof schema that allows to verify properties of the semantic relations by induction on the length of computations. This may be used to prove that the goals of a unit are reached (or not reached).

5 Conclusion

In this work we have presented the basic ideas and features of autonomous units and explained by example their application for modelling autonomous processes in a basic transport network. The model we presented so far is simple, since e.g. the trucks plan random tours and randomly accept or reject offers. Additional information like the distance of the regarded tour section or the size and weight of the package, the loading time, priorities, intelligent choice using reasoning and the like may get involved in the decisions. They may also depend on the goal that the truck unit tries to achieve. Detailed case studies are needed in the near future to illustrate and investigate these concepts. Furthermore the idea of autonomous units as a method for modelling should be compared to other modelling approaches that are currently employed in logistic scenarios, like e.g. Petri Nets, UML, or business process models. Currently the operational character of the autonomous units is purely sequential. It should be investigated how parallelism and concurrency can be incorporated into this approach to better reflect the real world, where all the units act simultaneously.

References

[KK99] H.-J. Kreowski and S. Kuske. Graph transformation units and modules. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools*, pages 607–638. World Scientific, Singapore, 1999.

[KK00] P. Knirsch and H.-J. Kreowski. A note on modeling agent systems by graph transformation. In M. Nagl, A. Schürr, and M. Münch, editors, *AGTIVE'99 International Workshop on Applications of Graph Transformation with Industrial Relevance*, volume 1779 of *Lecture Notes in Computer Science*, pages 79–86, Berlin, 2000. Springer-Verlag.

[KKK02] R. Klempien-Hinrichs, P. Knirsch, and S. Kuske. Modeling the pickup-and-delivery problem with structured graph transformation. In H.-J. Kreowski and P. Knirsch, editors, *Proc. Applied Graph Transformation (AGT'02)*, 2002. 119–130.

[KKS97] H.-J. Kreowski, S. Kuske, and A. Schürr. Nested graph transformation units. *International Journal on Software Engineering and Knowledge Engineering*, 7(4):479–502, 1997.

[LK73] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21(9):498–516, 1973.

[Roz97] G. Rozenberg, editor. *Handbook on Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations.* World Scientific, Singapore, 1997.

[SS95] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[WJ94] M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In M. J. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 1–39. Springer, Berlin, 1994.