Diagrammatic Specification of Mobile Real-Time Systems

Sven Linker[*]

Carl von Ossietzky University of Oldenburg
sven.linker@informatik.uni-oldenburg.de

Abstract. Behavior of spatio-temporal systems depends on real-time as well as spatial aspects. More and more safety-critical systems fall into this domain and thus raise the urge for formal specification and verification methods for this type of systems. For this purpose, we develop a diagrammatic language of *Shape Diagrams* that concentrates on the critical concepts and is usable by both engineers and scientists. We present two syntaxes, an abstract one based on hypergraphs and graph transformation systems that constitutes the abstract structure, and a concrete one given in terms of conventions for drawing diagrammatic pictures.

Key words: diagrammatic specification, spatio-temporal systems, formal reasoning, graph transformation

In the last decades, the use of *real-time systems*, i.e., systems which are required to react to given inputs within a certain time bound, has dramatically spread, especially in safety-critical areas. *Mobile* real-time systems additionally have to respect spatial constraints and relations to ensure safe behaviour. Examples of such systems would be cars organizing themselves automatically as platoons, aircraft controlling devices, or, on a lesser scale of criticality, automated and autonomous vacuum cleaners. Due to the complexity of these systems, methods for formally verifying their correct behaviour are highly desired. However, mathematical formalisms allowing for proofs of the correctness of mobile real-time systems with respect to time bounds as well as both qualitative and quantitative spatial constraints are sparse. To the best of our knowledge, the only formalism capturing spatial and temporal aspects in a uniform manner is *Shape Calculus* [1], a multi-dimensional logic interpreted on models based on polyhedra.

Diagrams are an often used engineering method to enhance communication between engineers during development. Hence a diagrammatic language suited for the specification of mobile real-time systems is desired. To bridge the gap between engineering tasks and the formal verification of correct behaviour, such a language should be equipped with formal semantics.

For this purpose, we propose the language of *Shape Diagrams* (see Fig. 1). In the following, we briefly describe a concrete and hint at the definition of

[*] This work is supported by the German Research Foundation (DFG), grant GRK 1076/1

an abstract syntax for this language. We establish the *concrete syntax* informally by conventions. Shape Diagrams consist of a stack of *layers*, which denote successive points in time. Layers are depicted by rectangles parallely projected onto the drawing plane. The interior of a layer describes a spatial situation, where objects are abstracted to labelled rectangles called *shapes*. To reduce the complexity of Shape Diagrams, we do not allow for arbitrarily shaped objects. For non-rectangular objects, safe bounding boxes have to be used. To represent restrictions on the durations between layers as well as the distances between objects and the borders of layers, arrows annotated with real-valued intervals are employed. Note that arrows constraining distances may connect shapes across layers. In such a case, auxillary lines have to be drawn to resolve ambiguities. We support an *assumption/commitment-style* reasoning, i.e., to express that under certain assumptions, a system is required to fulfill the commitments, we employ shading. E.g. the diagram in Fig. 1 asserts, that if two cars drive one after the other at a distance of 60m, they are required to build a platoon within 10 to 30s, i.e., the rear car follows the car in front at a distance between 2 and 3m.

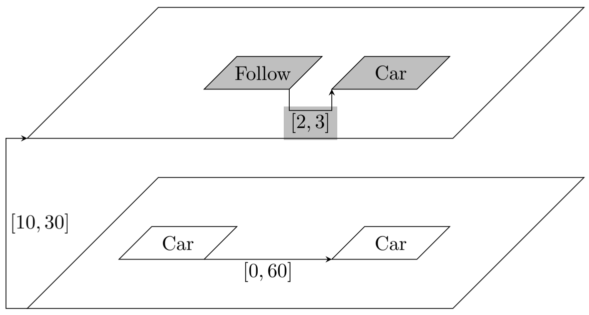

Fig. 1. A Shape Diagram

The *abstract syntax* of Shape Diagrams has to represent the following entities and properties: the diagrammatic elements (layers, shapes and arrows), the attachment relations of arrows, the positions of shapes in relation to each other and the order of layers. To reflect these different aspects in the abstract syntax, we use the notion of *hypergraphs*, i.e. graphs where an edge may connect an arbitrary number of nodes via its so-called *tentacles*. We employ typed hyperedges, i.e. the type of an edge determines the number and names of its tentacles.

The part of an abstract syntax graph representing the lower layer of Fig. 1 is depicted in Fig. 2. Following Minas [2], each diagrammatic element is represented by a hyperedge. The different elements are distinguished by hyperedge types, e.g. we employ the types shape and tarrow for shapes and constraints on durations, respectively. The vertices of the graphs denote *attachment areas* of the elements, e.g. each tarrow edge visits two vertices with the tentacles s and t. The layer edges visiting these nodes are representing the source and target of the arrow. The description of the relative position of a shape is more difficult. We use an approach developed by Guesgen [3] and Nabil et al. [4] generalizing interval relationships to more than one dimension. The idea is based on projecting the objects onto the axes, thus obtaining an interval on each axis for each shape. Then the relations of two intervals in each dimension can be stored indepedently, by hyperedges representing the interval relations visiting attachment nodes of shape edges. In Fig. 2, the edges labelled = and < denote the relative positions of the shapes in Fig. 1.

For a formal definition of the abstract syntax, we employ a *graph transformation system*. Such a system consists of an axiom, i.e. a hypergraph, and transformation rules to obtain new graphs by repeatedly replacing parts of the axiom resp. the resulting graphs, similar to textual grammars. The rules differ strongly in their complexity. For example, the rule for the creation of layers is context-free, i.e. it only replaces a single hyperedge without taking other incident edges into account. For the creation of the spatial arrows, i.e. hyperedges of type sarrow, the existence of at least one shape or layer is necessary. Hence these rules are only applicable in a certain embedding context. The most complex rules create the relative spatial positions. They make use of both embedding contexts and application conditions [5], because on the one hand at least two shapes have to be present for edges defining relative positions, but on the other hand exactly one relation between two shapes has to be created for a complete syntax graph.

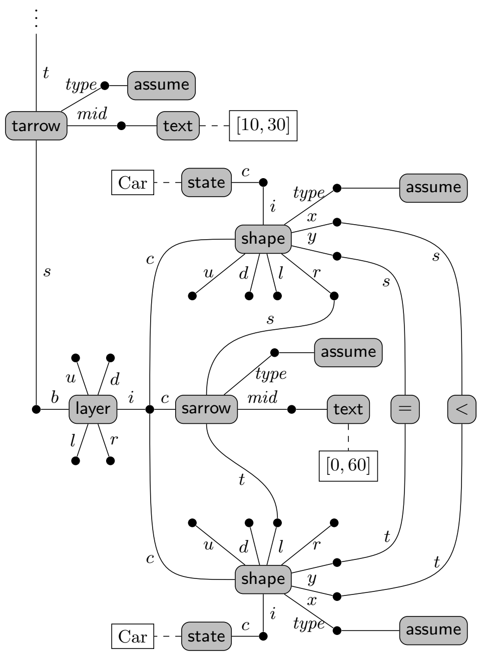

Fig. 2. Part of the Abstract Syntax Graph of Fig. 1

References

1. Schäfer, A.: Specification and Verification of Mobile Real-Time Systems. PhD thesis, University of Oldenburg (2006)
2. Minas, M.: Hypergraphs as a uniform diagram representation model. In: TAGT'98, London, UK, Springer (2000) 281–295
3. Guesgen, H.W.: Spatial reasoning based on Allen's temporal logic. Technical Report TR-89-049, Int'l. Comp. Science Inst., Berkeley (1989)
4. Nabil, M., Shepherd, J., Ngu, A.H.H.: 2D projection interval relationships: A symbolic representation of spatial relationships. In: Proc. 4th SSD, Springer (1995) 292–309
5. Habel, A., Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. MSCS **19** (2009) 245–296