Journal of Visual Languages and Computing 38 (2017) 84–96

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Resource-aware policies☆

Paolo Bottoni[a,*], Andrew Fish[b,**], Alexander Heußner[c,**], Francesco Parisi Presicce[a]

[a] Dipartimento di Informatica, "Sapienza" Università di Roma, Italy
[b] School of Computing, Engineering and Mathematics, University of Brighton, UK
[c] Otto-Friedrich-Universität Bamberg, Germany

ARTICLE INFO

Keywords:
Annotations
Resources
Modelling spider diagrams
Synchronisation
Conformance
Policies

ABSTRACT

In previous papers, we proposed an extension of Spider Diagrams to object-oriented modelling, called Modelling Spider Diagrams (MSDs), as a visual notation for specifying admissible states of instances of types, and for verifying the conformance of configurations of instances with such specifications. Based on this formalisation, we developed a notion of transformation of MSDs, modelling admissible evolutions of configurations. In the original version of MSD, individual instances evolve independently, but in reality evolutions often occur in the context of available resources, so transformations must be extended to take this into account. In this paper we provide an abstract syntax for MSDs, in terms of typed attributed graphs, and a semantics for the specification of policies based on notions from the theory of graph transformations, and we associate with them a notion of resources. We also introduce a synchronisation mechanism, based on annotation of instances with resources, so that the transformations required by a policy occur with respect to available resources. In particular, resources can be atomically produced or consumed or can change their state consistently with the evolution of the spiders subject to the policy.

1. Introduction

In the context of requirements engineering, the expression and understanding of policy specifications by various, possibly non-technical, stakeholders, may be favoured by the adoption of diagrammatic, intuitive representations. In [1] we presented a framework for expressing *temporal* policies that restrict admissible evolutions of the state of instances of types, based on an underlying extension of Spider Diagrams for representing types or their instances. In this framework, in contrast to classical Spider Diagrams, curves represent admissible states, and additional temporal information is provided by suitable annotations of the graphical elements. A temporal policy then specifies over which periods an instance of a given type can be in some given state. However, these extensions may not be sufficient in practice, if the evolution of an instance, hence the viability of a policy, depends essentially on the (un-)availability of some resource.

We use, as running examples, a number of scenarios derived from actual parking policies, along the lines of the examples that we have used as testbed for our definition of policies [1,2] and which permit a number of variations.

Running Example. The Fiumicino airport in Rome has recently put in place a policy to constrain the time during which a private car collecting passengers can stay in the arrival area. The car's registration plate is photographed when entering the area. If the car does not leave within 15 min, a fine will be issued to the owner. To avoid this penalty, the owner can: (a) park in a special 'free parking' zone, where the car is allowed to park for 15 min. (2) leave the restricted area, or (3) enter a 'toll parking' zone, where the car can stay indefinitely (actually, no longer than 24 h). When entering a parking zone, the owner must collect a ticket while the plate is photographed again, and the car is recognised to have left the area when the ticket associated with that plate is discharged at the exit. If the car stays in the free parking zone for more than 15 min, a fee must be paid, and registered on the ticket, otherwise the car will be subject to a fine when it is recognised as leaving the arrival area. In either case, if the duration of the stay exceeds the time that has been paid for, a fine will be issued. No car will

☆ We are happy to contribute to an homage to S.-K.Chang celebrating his achievements in the foundation and conduction of the Journal of Visual Languages and Computing, for many years together with the late Stefano Levialdi, and in the establishing the whole area of research of Visual Languages. This work brings together two areas of research, diagrammatic reasoning and graph transformations, which, although never directly investigated by S.-K., fit very well with his vision that "cooperative and interdisciplinary research can lead to a better understanding of the visual communication process for developing an effective methodology to design the next generation of visual languages" (from the preface to the groundbreaking 1986 volume on Visual Languages curated by him). Thanks, S.-K., for starting the field!

* Corresponding author.
** Corresponding authors.
E-mail addresses: bottoni@di.uniroma1.it (P. Bottoni), Andrew.Fish@brighton.ac.uk (A. Fish), alexander.heussner@uni-bamberg.de (A. Heußner).

http://dx.doi.org/10.1016/j.jvlc.2016.10.004
Received 1 December 2015; Received in revised form 27 May 2016; Accepted 19 October 2016
Available online 20 October 2016
1045-926X/

be admitted to any parking zone which does not offer available parking spaces, so that the possibility for a car to comply with the policy by staying in a parking place is ultimately subject to the availability of the space. Buses can only load and discharge in suitable spaces, while taxis have to enter a reserved area. Payment can be performed in different ways, e.g. cash, credit card, company subscriptions, or through a mobile application.

We do not consider here the timing aspect, as it has been extensively treated in [1], but we focus on two important features of policy modelling:

(1) A policy (implicitly or explicitly) defines *admissible sequences* of operations or states. For example, a car having enjoyed a free parking period must either leave the area or buy some time before leaving the area anyway. A car cannot enjoy a free parking period after paying a ticket unless it exits and re-enters the controlled area.
(2) In order to comply with a policy, *resources* of some kind must be available. For example if the driver wants to buy some parking time he or she must have some paying instrument available, and the airport system must provide suitable devices to collect the payment, e.g. parking meters, or connection with the application database. On the parking side, a car which cannot find a parking space, of any type, must necessarily leave the controlled area. The *state* of these resources change either as a consequence of an action executed by some car following the policy, e.g. a parking place is occupied or abandoned by a car, or for independent reasons, e.g. a set of parking places is excluded from usage due to roadworks or security reasons on special occasions.*Running Example* (*Contd.*). Parking regulations in the City of Rome distinguish between three types of public parking places alongside streets: completely free places, short-term free parking places, and toll parking places where parking time must be paid in advance. Short-term parking places cannot be continually occupied by the same car for more than three hours, and no extra time can be bought for these places, but the car can leave the place and reoccupy it, placing an indication of the time at which the new occupation starts. Compliance to this policy is not controlled by automatic means but inspectors can check it at any time (looking for cars which have overstayed their free or paid period). In order for this policy not to be repelled by courts, as appellants can lament the lack of free spaces, the city council must guarantee that in each neighbourhood a certain ratio of (completely or short-term) free to toll places is respected.

We use this example as an indication of the fact that policies may have conditions of *validity*, out of which they cannot be enforced. These conditions may refer to availability of resources, as well as to specific *events* which activate them or not. We also remark that policies apply to well-defined *types* of elements, e.g. ambulances, police, or firefighter cars are not subject to parking restrictions, and that individual instances start being subjected to a policy only when some *trigger event* occurs, e.g. a car enters the Fiumicino controlled area.

Running Example (*contd.*). Garages of commercial malls offer free parking to all customers, but customers may have to provide evidence of a purchase at the mall (upon leaving) to qualify for the use of the free parking. Internally, the mall can reserve spaces closer to the shopping area for pregnant women or families with children, as well as for disabled people, all states which can be easily demonstrated. Multi-storey car parks can reserve different areas for long vehicles and for rental cars, and they do not admit methane-propelled cars in covered zones. In all these cases, availability of a resource is constrained by some property of the car (or of its owner), or some combination of properties. Other similar cases include private car-parks which can only be entered by customers who possess a parking permit or membership card.

The above are cases where access conditions may differ for elements with specific properties (which can be assessed before or after using the resource). Conformance to a policy may then depend on such properties and on synchronisation of the changes in state dictated by the policy with changes in the availability of multiple resources at the same time.

The formal model for *resource-aware* policies presented here extends our previous policy framework along two directions: On the one hand, we provide a more general notion of policy in Section 2, of which temporal policies are a special case. On the other hand, after introducing our notion of resources in Section 3, we consider, in Section 4, how (un-)availability of resources affects conformance to a policy. Discussion of related work is postponed until Section 5, after which Section 6 concludes the paper.

Three recent lines of research concur in this work: (1) the extension of Spider Diagrams to the world of OO modelling, in particular through the definition of policies; (2) the definition of an abstract representation of Spider Diagrams in terms of Spider Graphs [3], from which we derive a formal notion of *conformance* of an instance to a policy; (3) the notion of annotation as a flexible way for connecting different domains. In addition, we introduce a generic notion of resource and consequently enrich the notion of policy. We use annotations to relate elements of the domain for which the policy is defined with resources needed for being conformant to the policy, as expressed by global constraints. We define a process of synchronisation between elements and resources in transformations which ensures conformity to the policy.

In particular, we adopt a simplified version of the notion of Spider Graphs, which we have developed to set the logical formalism of Spider Diagrams within the framework of attributed typed graphs and we provide a number of constructs for expressing and reasoning on policies. We introduce an original notion of synchronisation of elements subject to a policy with the needed resources, via the annotation mechanisms presented in [4]. In the resulting setting, systems, modelled as Spider Graphs, can evolve according to a policy under constraints represented by resource availability, while considering resource evolution only in the context of system evolution. Annotations are used both in the representation of constraints and in the construction of synchronised rules, modelling the concurrent evolution of systems and resources. System evolution is modelled through transformation rules ensuring conformity with a policy. Rules can be derived from the policy specification following a procedure described in [2].

The idea of constraining transformations on resources presented in Section 4 is general and could be used in any domain in which policies determine admissible transitions. We point the reader to [3] for detailed motivations for the use of Modelling Spider Diagrams, and on the introduction of Spider Graphs.

2. Formal setting: modelling spider diagrams, graphs and policies

Graphs and typed graphs. Following [5], a graph is a tuple (V, E, s, t), with V and E finite sets of *nodes* and *edges*, and functions $s: E \to V$, $t: E \to V$ mapping an edge to its source and target. A *graph morphism* $m: G \to H$ is given by a pair of functions $m_V: V_G \to V_H$ and $m_E: E_G \to E_H$ preserving sources and targets of edges. Morphism composition is denoted by $\bigcirc$, where $m_1 \bigcirc m_2$ indicates that m_1 is applied to the result of the application of m_2.

A *graph transformation rule* is a span of graph morphisms $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$, and is applied following the Double Pushout (DPO) Approach. Fig. 1 (left) shows a DPO direct derivation diagram. Square (1) is a pushout (i.e. G is the union of L and D through their common elements in K), modelling the deletion of the elements of L not in K, while pushout (2) adds to D the new elements, i.e. those present in R but not in K, to obtain H as the result of the rule