Journal of Computer and System Sciences 86 (2017) 82–107

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Process-centric views of data-driven business artifacts[☆]

Adrien Koutsos[a], Victor Vianu[b,c,*]

[a] ENS Cachan, France
[b] UC San Diego, United States
[c] INRIA-Saclay, France

A R T I C L E   I N F O

Article history:
Received 16 October 2015
Received in revised form 24 June 2016
Accepted 21 November 2016
Available online 16 January 2017

Keywords:
Workflows
Data-aware
Process-centric
Views

A B S T R A C T

Declarative, data-aware workflow models are becoming increasingly pervasive. While these have numerous benefits, classical process-centric specifications retain certain advantages. Workflow designers are used to development tools such as BPMN or UML diagrams, that focus on control flow. Views describing valid sequences of tasks are also useful to provide stakeholders with high-level descriptions of the workflow, stripped of the accompanying data. In this paper we study the problem of recovering process-centric views from declarative, data-aware workflow specifications in a variant of IBM's business artifact model. We focus on the simplest process-centric views, specified by finite-state transition systems, describing regular languages. The results characterize when process-centric views of artifact systems are regular, using both linear and branching-time semantics. We also study the impact of data dependencies on regularity of the views. As a side effect, we obtain several new results on verification of business artifacts, including a decidability result for branching-time properties.

1. Introduction

Data-driven workflows have become ubiquitous in a variety of application domains, including business, government, science, health-care, social networks [42], crowdsourcing [6,7], etc. Such workflows resulted from an evolution away from the traditional process-centric approach towards data-awareness. A notable exponent of this class is the *business artifact model* pioneered in [36,29], deployed by IBM in professional services. Business artifacts (or simply "artifacts") model key business-relevant entities, which are updated by a set of services that implement business process tasks. This modeling approach has been successfully deployed in practice [4,3,9,15,44]. In particular, the Guard-Stage-Milestone (GSM) approach [13,26] to artifact specification uses declarative services with pre- and post-conditions, concurrency, and hierarchy. The OMG standard for Case Management Model and Notation (CMMN) [11], announced last year, draws key foundational elements from GSM [32].

Declarative, high-level specification tools such as GSM allow to generate the application code from the high-level specification. This not only allows fast prototyping and improves programmer productivity but, as a side effect, provides new opportunities for automatic verification. Indeed, the high-level specification is a natural target for verification, as it addresses the most likely source of errors (the application's specification, as opposed to the less likely errors in the automatic

☆ A preliminary version of this article appeared in the *Intl. Conf. on Database Theory* (ICDT) 2015 [28].
* Corresponding author.
E-mail addresses: adrien.koutsos@ens-cachan.fr (A. Koutsos), vianu@cs.ucsd.edu (V. Vianu).

http://dx.doi.org/10.1016/j.jcss.2016.11.012
0022-0000/© 2017 Elsevier Inc. All rights reserved.

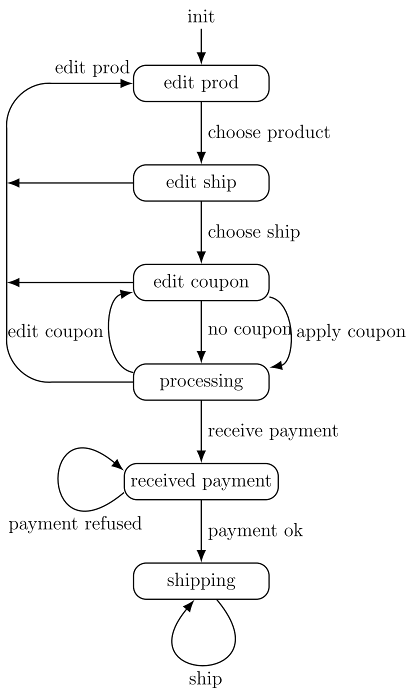

Fig. 1. A linear-time process-centric view.

generator's implementation). This has spawned an entire line of research seeking to trace the boundary of decidability of properties of such systems, expressed in variants of temporal logic (see [19]).

While declarative specifications of workflows have many benefits, they also come with some drawbacks. Workflow designers are used to traditional process-centric development tools, such as BPMN (Business Process Model and Notation), workflow nets, and UML activity diagrams, that focus on control flow while under-specifying or ignoring the underlying data. Such process-centric views of workflows are also useful to provide stakeholders with customized descriptions of the workflow, tailored to their role in the organization. For example, an executive may only require a high-level summary of the business process. Descriptions of the workflows as sequences of tasks stripped of data are intuitive and often sufficient for many users. Thus, recovering such specifications for business artifacts in an intuitive, familiar process-centric language is often desirable. However, although they differ in the specific contructs, all common process-centric languages are finite-state, and can only specify regular languages. We are therefore interested in understanding when business artifacts can be faithfully represented by regular languages of sequences of tasks.

Specifically, in this paper we study views of business artifact runs consisting of the sequences of services applied in each run.[1] The views come in two flavors, depending on whether we are interested in linear runs alone, or in the more informative branching-time runs. We call these the linear-time, resp. branching-time (service) views of the artifact system.

Example 1. To illustrate linear-time service views of declarative workflows, we consider a variant of the running example of [12], specified in Section 2.1. In the example, the customer can choose a product, a shipment method and apply a coupon to the order. The order is filled in a sequential manner as is customary on e-commerce web-sites. After the order is filled, the system awaits for the customer to submit a payment. If the payment matches the amount owed, the system proceeds to shipping the product. At any time before submitting a valid payment, the customer may edit the order (select a different product) an unbounded number of times. The linear-time service view of this workflow consists of the sequences of service names that occur in all infinite runs of the system, and is specified by the finite-state transition system shown in Fig. 1 (we discuss the use of infinite runs in Section 2). A more informative view, that captures the services applied in branching-time runs of the system, is shown in Fig. 2. Intuitively, the branching-time view captures *precisely* which services may be applied *at each point* in a run. To understand the difference with linear-time views, consider the states labeled *edit coupon* in Figs. 1 and 2. In the linear-time view specification, there is only one such state, in which two actions can be taken: *no coupon* and *apply coupon*. However, the two actions are not *always* applicable whenever the state *edit coupon* is reached. If no product has an applicable coupon, the only action possible is *no coupon*. If for all products and shipping method there is some applicable coupon, then both *no coupon* and *apply coupon* are applicable. Finally, if some products have applicable coupons

[1] In various formalizations of business artifacts, tasks are called *services*.