Information Sciences 421 (2017) 1–14

Contents lists available at ScienceDirect

Information Sciences

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

Modeling and control of flexible context-dependent manufacturing systems☆

André Lucas Silva, Richardson Ribeiro, Marcelo Teixeira*

Federal University of Technology-Paraná, Pato Branco, Brazil

ARTICLE INFO

Article history:
Received 7 September 2016
Revised 20 August 2017
Accepted 26 August 2017
Available online 31 August 2017

Keywords:
Formal modeling
Context recognition
Automated programming
Factory automation
Flexible manufacturing systems

ABSTRACT

In emerging Manufacturing Systems (MSs), flexibility is a key issue. It is related to the ability for a MS to recognize the context and switch its workflow accordingly. Although the literature has provided automated options to model and control MSs, programming context-dependent controllers remains challenging. This is an event-based construction that integrates a large and intricate combination of events and states in order to make the controller flexible, i.e., include context-sensitiveness strategies subject to switching at runtime. Without self-adaptation, each system configuration may require an entire control solution to be recalculated, which implies redesigning the whole modeling and implementation structures. This paper shows that a system model can nevertheless be enriched with elements collected from the context, which optimizes the design of formula-based constraints that can then be integrated to control frameworks for synthesis and code generation. The result is a controller that recognizes the context and takes control decisions accordingly. Examples are provided to illustrate the approach.

1. Introduction

A *Manufacturing System* (MS) is an industrial production process that transforms material into products by integrating people, equipments and technology [1]. When it is context sensitive, i.e., it detects and reacts to changes occurred in the factory floor, starting to behave differently, then it is said to be *flexible*, being called *Flexible Manufacturing System* (FMS) [2]. Nowadays, FMSs represent an opportunity for shifting from fixed to customized production [3–5]. When associated to computational technology, e.g., *web, intelligence, Big Data, IoT, Cloud*, etc., they lead to modern advanced methods for industry of the future [6–9].

In manufacturing automation, the use coordinated, intelligent and flexible work-units has becoming increasingly decisive and they can help, for example, to autonomously move production and materials, perform tasks that are dangerous for humans, increase performance, efficiency, customize production, etc. However, for a flexible work-unit to be really useful in manufacturing, its controller is required to make “smart” responses to dynamic environments. It is expected that factory components interact with each other and with the environment in a concurrent manner, sharing resources and behaving in a maximally permissive and non blocking way.

☆ Modeling and Control of Context-Dependent Manufacturing Systems

* Corresponding author.
E-mail addresses: andsil@alunos.utfpr.edu.br (A. Lucas Silva), richardsonr@utfpr.edu.br (R. Ribeiro), marceloteixeira@utfpr.edu.br (M. Teixeira).

http://dx.doi.org/10.1016/j.ins.2017.08.084
0020-0255/

In conjunction, these features make it hard to obtain controllers that ensure safety properties in manufacturing. In fact, the programming task depends on so many factors, such as the number of components, environment size, concurrency, parallelism, behavioral shifting, etc., so that traditional paradigms for software development are usually unsuitable. The literature provides some alternatives, such as the use of concurrent and dynamic programming [10,11], interfaces [12], general-purpose control languages [13,14], computational intelligence [15,16], etc. Another alternative is to adopt high level model-driven strategies in order to express system behaviors and requirements [17–19]. In this case, automated operations can be processed in order to calculate sub behaviors holding properties of interest.

When the process is event-driven, i.e., when the objective is to define operational sequences of machines based on events that spontaneously occur in the plant, then state-machine is a natural choice for modeling. A formal approach that supports the automated synthesis of controllers for *Discrete Event-based Systems* (DESs) is the *Supervisory Control Theory* (SCT) [20]. Mathematically grounded on *Finite-state Automata* (FA) formalism [21], the SCT facilitates capturing the system semantic through high-level models and leads automatically to the optimal controller code.

In spite of its practical relevance and formal background, the SCT faces severe limitations to solve real, modern, industrial-scale processes, as its modeling framework is essentially static and non adaptive. In fact, although FA can quite efficiently describe event-driven behaviors, they do not directly include, for example, context-sensitiveness. If a controller is not context-sensitive, an entire control solution may have to be recalculated whenever the system configuration shifts, as every context may impose a particular control law. As a consequence, the whole modeling and implementation structures also change, which is time consuming and can be infeasible for systems comprising real-time constraints.

An alternative that has recently attracted attention in the literature suggests to enrich the SCT with *Extended Finite-state Automata* (EFA), which combine FA to a variable structure [22,23]. Besides providing a user-friendly framework, EFA also support efficient computational processing by means of abstraction techniques [24], modular control [25], etc. However, from the best of our knowledge, EFA have not yet been exploited in modeling and control of flexible context-dependent MSs.

In this paper, EFA are used to enrich a MS model with elements collected from the context updated by the system. A set of variables is associated to the modeling structure in order to store context elements. Formulas are combined to the system model transitions in order to update variables whenever a given target event occurs. Then, variable values are used to express context-switching specifications that lead to flexibly coordinate factory floor components. The enriched system model and specifications can be integrated to the SCT framework for synthesis [24]. The product is a controller that recognizes the context and takes control decisions accordingly. A methodology that guides the EFA modeling is provided and MSs examples illustrate the approach.

The manuscript is structured as follows. Section 2 introduces some preliminaries on DES control, which is exemplified in Section 2.2 in the context of multiple robots coordination; Section 3 presents the modeling with EFA and its integration to the SCT framework; An example is presented in Section 4 and some conclusions and perspectives are discussed in Section 6.

2. Preliminaries

Discrete Event Systems (DESs) refer to a family of systems characterizing a number of industrial applications, such as manufacturing, communication, robotic, etc. DESs have in common the fact that their transitions are not guided by the time, but by *events* that occur irregularly, progressing the systems in many different and unpredictable manners [21]. Thus, it is conceivable that modeling DESs is also different from modeling time-dependent systems.

Languages are formalisms that can be used to describe discrete-event behaviors [21]. Their basic structure are *events*, which are taken from a finite *alphabet* Σ, where Σ^* denotes the set of all (finite) *strings* constructed by taking events from Σ, including the *empty string* ε. A subset $L \subseteq \Sigma^*$ is called a *language*. The *prefix-closure* of L is $\overline{L} = \{ s \in \Sigma^* \mid st \in L \text{ for some } t \in \Sigma^* \}$, which includes all prefix of all strings in L. In practice, when L intends to describe a DES behavior, then a string $s \in L$ can be associated to a (completed or not) task or, in other words, an operational sequence possible in the system, and t identifies the next-step after s. Any behavior outside L is undesired and, therefore, prevented in the system.

A language L it is said to be *regular* if, and only if, it can be recognized by a finite state automaton [26]. In industry, regularity is of interest, as it delimits a class of languages that are suitable for computational processing, i.e., they can be represented by automata occupying finite memory when stored in a computer [21].

Finite State Automata (FA) define a simple, intuitive, and powerful framework to formally generate languages. A FA can be formally defined as a 5-tuple $G = \langle \Sigma, Q, q^\circ, Q^\omega, \mapsto \rangle$, where: Σ is the alphabet of events; Q is the set of states; $q^\circ \in Q$ is the initial state; $Q^\omega \subseteq Q$ is the subset of marked states (complete tasks); $\rightarrow \subseteq Q \times \Sigma \times Q$ is the state transition relation.

For two any states $q_1, q_2 \in Q$, $q_1 \xrightarrow{\sigma} q_2$ denotes a transition from the state q_1 to q_2 with the event $\sigma \in \Sigma$, and $G \xrightarrow{s} q$ denotes that a string s is possible in G. Two languages can be defined from G: $L(G) = \{s \in \Sigma^* \mid G \xrightarrow{s} q \in Q\}$; $L^\omega(G) = \{s \in \Sigma^* \mid G \xrightarrow{s} q \in Q^\omega\}$. $L(G)$ is called the *generated language* and it includes all strings possible in G, while $L^\omega(G)$ is the *marked language*, i.e., the set of strings leading to marked states, in practice associated to complete tasks.

Two FA, $A = \langle \Sigma_A, Q_A, q_A^\circ, Q_A^\omega, \mapsto_A \rangle$ and $B = \langle \Sigma_B, Q_B, q_B^\circ, Q_B^\omega, \mapsto_B \rangle$, can be synchronously composed as $A \parallel B = \langle \Sigma_A \cup \Sigma_B, Q_A \times Q_B, (q_A^\circ, q_B^\circ), Q_A^\omega \times Q_B^\omega, \rightarrow \rangle$, in which:

- $(q_A, q_B) \xrightarrow{\sigma} (q'_A, q'_B)$, if $\sigma \in \Sigma_A \cap \Sigma_B$;