Science of Computer Programming 94 (2014) 109–129

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

ASM, controller synthesis, and complete refinement

Richard Banach [a,*,1], Huibiao Zhu [b,2], Wen Su [c,3], Xiaofeng Wu [b]

[a] School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK
[b] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, 3663 Zhongshan Road North, Shanghai 200062, PR China
[c] School of Computer Engineering and Science, Shanghai University, Shanghai, PR China

HIGHLIGHTS

- Decomposing controller and plant from a unified model.
- ASM scheduling policy.
- Continuous ASM.
- Chopsticks case study.

ARTICLE INFO

Article history:
Received 18 January 2013
Received in revised form 23 April 2014
Accepted 24 April 2014
Available online 6 May 2014

Keywords:
ASM
Controller
Complete refinement
Continuous ASM
Chopsticks

ABSTRACT

While many systems are naturally viewed as the interaction between a controller subsystem and a controlled, or plant subsystem, they are often most easily initially understood and designed monolithically, simply as a collection of variables that represent various aspects of the system, which interact in the most self-evident way. A practical implementation needs to separate controller from plant though. We study the problem of when a monolithic ASM system can be split into controller and plant subsystems along syntactic lines derived from variables' natural affiliations. We give restrictions that enable the split to be carried out cleanly, and we give conditions that ensure that the resulting pair of controller and plant subsystems have the same behaviours as the original design. We relate this phenomenon to the concept of *complete* refinement in ASM. Making this strategy work effectively, usually requires a nontrivial domain theory, into which a number of properties which are neither the sole possession of the controller subsystem nor of the plant subsystem must be placed. We argue that these properties are latent in the original monolithic model. We illustrate the theory with a case study concerning eating with chopsticks. This leads to an extension of controller synthesis for continuous ASM systems, which are briefly covered. The chopsticks case study is then extended into the continuous sphere.

* Corresponding author.
E-mail addresses: banach@cs.man.ac.uk (R. Banach), hbzhu@sei.ecnu.edu.cn (H. Zhu), wsu@shu.edu.cn (W. Su), xfwu@sei.ecnu.edu.cn (X. Wu).

[1] A large portion of the work reported in this paper was done while the first author was a visiting researcher at the Shanghai Key Laboratory of Trustworthy Computing at East China Normal University. The support of ECNU is gratefully acknowledged.

[2] Huibiao Zhu is supported by National High Technology Research and Development Program of China (No. 2012AA011205), National Natural Science Foundation of China (No. 61361136002 and No. 61321064), Shanghai Knowledge Service Platform Project (No. ZF1213) and Shanghai Minhang Talent Project.

[3] Wen Su was supported in part by the Open Project of the Shanghai Key Laboratory of Trustworthy Computing (No. 07dz22304201303).

http://dx.doi.org/10.1016/j.scico.2014.04.013
0167-6423/

1. Introduction

Today, when one considers the ubiquity of embedded controllers, which take on the digital role in the interaction of a digital and an external system, it becomes clear that many systems are naturally viewed as the interaction between a controller subsystem and a controlled, or plant subsystem. Regarding the high level design of such systems, the fact that the ultimate design needs to be split into controller and plant subsystems is evident from the outset. However, it is often easier in the earlier stages of design to ignore that fact, and to focus exclusively on the overall system goals. This means postponing for the time being the issue of how the solution arrived at is to be organised into the two subsystems. Such a *monolithic* approach means that there is simply less to worry about in the earlier stages of design, when there is the most uncertainty concerning the most critical aspects of the problem. This allows the bulk of this early design activity to focus on the overall goals rather than lower level technical detail.

However, a practical implementation needs to separate the controller from the plant, since it is the controller which behaves according to a human-created digital design, and the plant behaves according to patterns determined by the laws of nature. In this paper we study the problem of when a monolithic ASM system design, embodying this dual controller/plant nature, can be split into separate controller and plant subsystems. This is to be done along generic syntactic lines derived from the most natural associations of the system variables to one or other (controller or plant) subsystem. The approach generalises a specific case study in which this task arose and where it was tackled rather informally [2]. We find that the success of the generic approach to such a goal requires that the monolithic design satisfies some simple criteria *ab initio*. As well as studying the problem from an abstract viewpoint, we present some examples.

In more detail, the rest of the paper is as follows. Section 2 describes the controller synthesis problem in abstract terms, focusing on the specific way that controller and plant are to be separated. A sufficient condition for the success of the desired controller/plant separation is formulated and proved. The undecidability of controller synthesis is also discussed in Section 2.1 by reduction to the Halting Problem. In Section 3 we consider a straightforward computable approximation to the controller synthesis problem, and argue that it is adequate for practical purposes. Section 4 discusses the role of the domain theory in the formulation of the controller synthesis problem — in many cases, the rules governing the behaviour of the system overall, can be viewed as belonging neither entirely to the controller subsystem nor entirely to the plant subsystem. Section 5 relates the preceding material to the ASM concept of *complete refinement*. When the controller synthesis problem is resolved successfully, each version of the overall system description refines the other. Section 6 introduces an example based on the idea of picking up food with chopsticks, viewed as a control problem. Section 7 extrapolates the preceding ideas to the case of continuous ASM, in which smoothly changing (as well as discretely changing) behaviours are admitted. Section 8 extends the discussion of the chopsticks case study by taking on board the continuous notions. In Section 9, we loosen the tight synchronisation between controller and plant, evident in the account so far, to create a slightly more liberal framework for the continuous case. Section 10 concludes.

2. The controller synthesis problem

We consider a generic ASM system consisting of basic ASM rules using straightforward single variable locations and a simple element of nondeterminism. Following [6], for our purposes, such a rule can be written as:

$$\begin{aligned}&\textsc{Op}(pars) =\\ &\textbf{if } guard(xs, pars) \textbf{ then choose } xs' \textbf{ with } rel\big(xs', xs, pars\big) \textbf{ do } xs := xs'\end{aligned} \tag{1}$$

In (1), *pars* are the input parameters (as needed) and *xs* are the variables modified by the rule. The rule's guard is *guard*, and *rel* represents the relationship that is to hold between the parameters, the before-values of the variables *xs*, and their after-values referred to as xs', when the rule fires. As usual, in a single step of a run of the system, all rules which are enabled (i.e. whose guards are true) fire simultaneously, provided that the totality of updates defined thereby is consistent, else the run aborts.

In this paper we are interested in control applications, and we envisage the design done in a monolithic way at the outset, addressing system-wide design goals before plunging into the details of subsystem design. Thus the design may start by being expressed using system-wide variables. However, by a process of gradual refinement, the collection of variables will eventually end up such that each variable can be identified as belonging to either the controller-subsystem-to-be, or the plant-subsystem-to-be. Despite this prospective partition of the variables though, a typical legacy of the top-down design process will be that many, or even all, of the rules of the system description will still involve variables of both kinds.

The controller synthesis problem is the problem of taking such a collection of rules (call it *Sys*), and separating it into one set of rules for the controller (call it *Con*) and another set for the plant (call it *Pla*), such that each subsystem of rules reads only the variables accessible to it, and each modifies only the variables that it owns. Moreover, this is to be done in such a way that the combination of the rules in *Con* and *Pla* generates the same behaviour (i.e. the same set of runs) as the original ruleset *Sys*.

Note that in [6], the importance of distinguishing *controlled* functions from *monitored* ones is firmly stressed, in one sense solving the controller synthesis problem right at the outset, since the distinction already separates the controller from the plant. Our perspective is different however, since it permits this aspect to be postponed for an initial portion of the