On the Difficulties of Concurrent-System Design, Illustrated with a $2 \times 2$ Switch Case Study

Edgar G. Daylight$^1$ and Sandeep K. Shukla$^2$

$^1$ a.k.a. Karel Van Oudheusden, Institute of Logic, Language, and Computation, University of Amsterdam, The Netherlands
egdaylight@yahoo.com

$^2$ Department of Electrical & Computer Engineering, Virginia Tech., USA
shukla@vt.edu

Abstract. While various specification languages for concurrent-system design exist today, it is often not clear which specification language is more suitable than another for a particular case study. To address this problem, we study four different specification languages for the same $2 \times 2$ Switch case study: TLA$^+$, Bluespec, Statecharts, and the Algebra of Communicating Processes (ACP). By slightly altering the design intent of the Switch, we obtain more complicated behaviors of the Switch. For each design intent, we investigate how each specification, in each of the specification languages, captures the corresponding behavior. By using three different criteria, we judge each specification and specification language. For our case study, however, all four specification languages perform poorly in at least two criteria! Hence, this paper illustrates, on a seemingly simple case study, some of the prevailing difficulties of concurrent-system design.

Keywords: formal specification languages, local reasoning, adaptability, non-functional requirements.

1 Introduction

Many papers on concurrent-system design introduce a specification language, backed up by multiple case studies. This paper, in contrast, studies various specification languages for a particular case study, i.e. a $2 \times 2$ Switch. In fact, we will examine three different behaviors of the Switch by means of the same specification language, and do this for four different languages: TLA$^+$, Bluespec, Statecharts, and the Algebra of Communicating Processes (ACP).

To compare specifications, we introduce three criteria in Sections 1.1–1.3. The importance of each criterion depends primarily on the designer’s purpose to use the specification language.
1.1 Local Reasoning

In order to introduce our first criterion, we present the following line of thought. The complexity encountered when specifying a concurrent system is proportional to the degree of global spatial and temporal reasoning that is required on behalf of the designer. Global spatial reasoning is synonymous for reasoning across relatively many spatial elements of the system under investigation (i.e., the Switch). For example, a designer who reasons across four different buffers of the Switch applies more global spatial reasoning than a designer who only has to reason across a maximum of two buffers while specifying the Switch’s behavior. Global temporal reasoning is synonymous for reasoning across relatively many states of the system. A designer who reasons across five consecutive states of the Switch’s underlying Finite State Machine applies more global temporal reasoning than a designer who only has to reason across a maximum of two consecutive states.

Global (spatial and temporal) reasoning may depend on either the case study, the chosen specification language, or both. An important remark is, that, if global reasoning for the Switch is not influenced by the chosen specification language, then there is little to gain from our subsequent discussions in terms of global reasoning. Our analysis, however, will show that the chosen specification language does in fact matter. For instance, global reasoning about a TLA+ specification generally differs from that of an ACP specification, even though the amount of global reasoning can be the same in both cases.

Our first criterion, therefore, is the local (as opposed to global) reasoning that is required by the designer in order to specify the Switch’s behavior.

Local Reasoning as an Ideal. It should be noted, however, that since each system is built from localized components that work together by means of some form of communication, we view local reasoning more as an ideal than as a realistic attribute of a specification language. We also stress that it is a subjective matter as to whether a specification language should avoid global reasoning as much as possible or not.

The ideal of local reasoning is best illustrated by means of a Kahn Process Network (KPN). In his 1974 paper [7], Kahn showed that if (i) each component process in a KPN is monotonic and continuous, and if (ii) the communications between such processes are infinitely buffered, then (iii) the entire system is deterministic and deadlock-free. Proving (i) and (ii) only requires local reasoning, while the result (iii) is a global property of the system under investigation. However, when confronted with a realistic (i.e. implementable) system, property (ii) does not hold. This, in turn, results in global reasoning when proving correctness claims, such as deadlock-freedom.

1.2 Adaptability

A second criterion is adaptability to variations in design intent. A specification language is adaptable for the Switch case study if it is capable of coping well with variations in the design intent of the Switch. For instance, consider two specifications in the same language of a simple $2 \times 2$ Switch and a more complicated