Approximating the Number of Execution Paths in Simulink Models

Jochen Quante

Robert Bosch GmbH, Corporate Research
Renningen, Germany

jochen.quante@de.bosch.com

Abstract

Simulink[1] is in widespread use for developing control applications, for example in the automotive domain. Simulink models often are the main artifacts that developers work on. The most common artifact is a block diagram. Code is directly generated from these models. Therefore, the models are also subject to maintenance and thus to the well-known effects of software ageing. In order to monitor and control maintainability of such models, it is necessary to measure them [3]. One important maintainability metric that has not been addressed for block diagrams so far is the number of execution paths, as this determines the number of test cases that are needed to execute each possible path through the software at least once. The number of control branches (cyclomatic complexity) is not enough, as the number of paths can vary vastly depending on their arrangement [2]. Therefore, an approximation of the number of execution paths for Simulink models is needed.

1 Block Diagrams

A block diagram is basically a directed hierarchical graph with ports, where connections denote data flows and nodes represent different operations on the data. For example, an *addition* node with two incoming data flows adds these two values and outputs their sum. A block diagram may also contain *subsystem* blocks, where each subsystem contains another block diagram, which allows modeling larger hierarchical systems. Control flow can basically occur in two forms: a) *triggered subsystems*, i. e., subsystems that are only executed when (or while) a trigger condition is true, and b) *switches*, which switch between multiple incoming data flows. Figure 1 shows an example for triggered subsystems, and Figures 2 and 3 show the use of switches.

2 NPATH Metric

The number of paths is potentially infinite as soon as loops are involved. Therefore, Nejmeh [2] defined the NPATH metric as an approximation of the number of *acyclic* execution paths. This means that each loop is only counted once. Despite this simplification, NPATH still tends to grow exponentially with function size. Nejmeh's metric is defined on the abstract syntax tree (AST). It basically sums up the number of paths of different branches (if-else, while, for) and multiplies the number of paths for sequential statements.

[1] https://www.mathworks.com/products/simulink.html

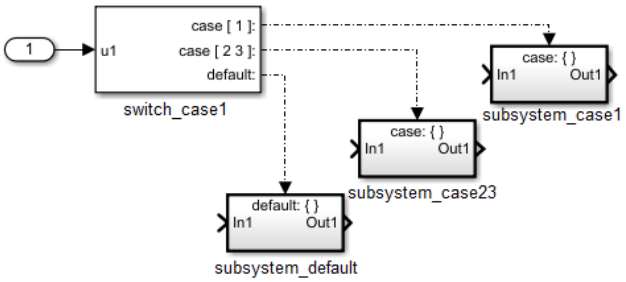

Figure 1: Switch-case with three triggered subsystems

With respect to triggered subsystems, we can easily create an AST-like representation that contains the basic control flow: A triggered subsystem becomes a child of an *if* or *switch-case* node from the parent diagram. Normal (unconditional) subsystems are handled like function calls. Nejmeh's metric can then be directly applied to these constructs. However, this is not true for switches, which need a special handling.

3 Data Flow Switches

Data flow switches also affect control flow, as only the data that comes in through the active inport needs to be calculated. An execution path in a block diagram with switches is thus characterized by the subset of the model that is active in a given scenario, based on the current switch settings. Each such subset corresponds to one execution path.

Control dependencies between switches and to other elements are not clear from the model and need to be determined first. In the following, we present an algorithm for calculating an approximation of the number of paths that are induced by switches.

Definitions: Let (V, E) be the data flow graph for a given block diagram, with vertices V and edges E. Let $S \subseteq V$ be the nodes of type Switch or MultiportSwitch. For $v \in S$, let v_c be the switch condition and $v_1, \ldots, v_n$ the different data inports of v (where n is the number of data inports of v).

Backward slicing from inputs: For $v \in S$ and $i \in \{1, \ldots, n\}$, let b_{v_i} be the backward slice on (V, E)

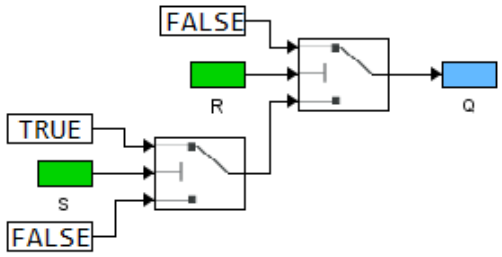

Figure 2: This flipflop has three paths, as the two switches are nested.

for inport v_i. Merge all slices for same condition and input port: $d_{v_c,i} := \bigcup_{v \in S} b_{v_i}$ with $i \in \{1, \ldots, n\}$.

Control dependent subset: For each switch condition v_c, calculate the intersection and union of all slices. As the nodes in the intersection are active in all cases, they are not control dependent on v_c. The relevant nodes thus are: $R_{v_c} := \bigcup_{i=1}^{n} d_{v_c,i} \setminus \bigcap_{i=1}^{n} d_{v_c,i}$.

Formal concept analysis: Create a binary relation $B \subseteq V \times V$ that contains all pairs (v_c, v_d) with v_c a condition node and $v_d \in R_{v_c}$. Perform formal concept analysis [1] on that formal context, which results in a concept lattice that contains all subset relations. In particular, it tells us which switches are control dependent on which other switches, and which are not.

Concept lattice evaluation: The number of switch-induced paths in the block diagram can be calculated from the lattice. We start traversal of the lattice from the bottom element $\perp$ (i. e., no condition nodes), which corresponds to the top level in terms of control dependency, and then visit its superconcepts. In case an object appears newly in the concept's extent, we have a nested switch, which means that the corresponding path counts have to be added. Otherwise, we have independent switches, i. e., a sequence, which means the path counts have to be multiplied.

$$cnt(x) = \begin{cases} inp(x) + \sum_{p \in sup(x)} (cnt(p) - 1) & \text{if } dext(x) \neq \emptyset \\ \prod_{p \in sup(x)} cnt(p) & \text{if } dext(x) = \emptyset \\ & \wedge\ sup(x) \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

NPATH $= cnt(\perp)$, with $inp(x)$ the number of inports of concept x's corresponding switch, $sup(x)$ the set of superconcepts of concept x, and $dext(x)$ the delta extent of x (i. e., the set of objects for which x is the smallest concept that contains them).

4 Examples

Figures 2 and 3 show two examples for diagrams with switches. In Figure 2, the two switches are nested in a way that one data input to the switch comes from the other switch. Therefore, we have nested switches and thus only three paths. In Figure 3, we have an

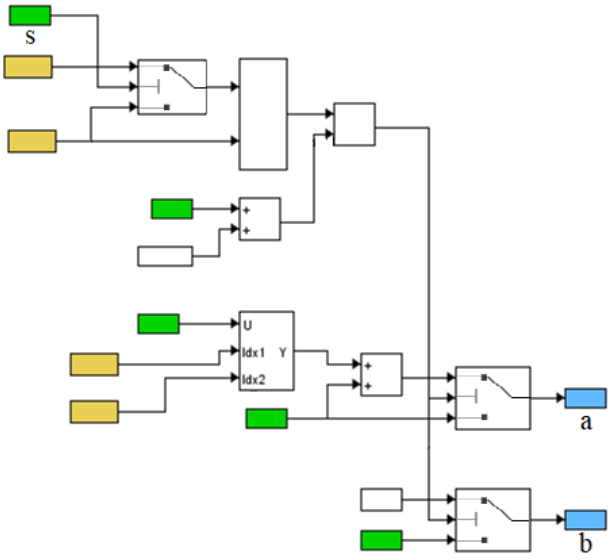

Figure 3: Example with three switches and four paths.

example of two switches that use the same condition. They are thus counted as two paths. The third switch is not control dependent on the other two switches. Therefore, we have the sequential case, which means multiplication of these two paths with the other two paths, which results in four paths.

5 Overall Path Metric

In order to get the overall NPATH metric for a given Simulink model, we construct the corresponding AST for the control-relevant aspects (triggered subsystems) and annotate the switch-induced local path count to the diagram nodes. Then, we apply Nejmeh's approach to aggregate the path counts according to the kind of construct. This results in an approximation of the number of non-cyclic execution paths through the model.

6 Conclusion

We have introduced a method for approximating the execution path count in block diagrams. The method can be used for complexity and testability assessment. It adds another potentially relevant dimension to block diagram metrics that has not been covered in the literature so far.

References

[1] B. Ganter and R. Wille. *Formal Concept Analysis – Mathematical Foundations*. Springer, 1999.

[2] B. A. Nejmeh. NPATH: A measure of execution path complexity and its applications. *Comm. of the ACM*, 31(2):188–200, 1988.

[3] M. Olszewska, Y. Dajsuren, H. Altinger, A. Serebrenik, M. Waldén, and M. G. J. van den Brand. Tailoring complexity metrics for Simulink models. Proc. of 10th European Conf. on Software Architecture – Workshops, pages 5:1–5:7, 2016.