# **Paper Review 1:** Overview and Performance Evaluation of Supervisory Controller Synthesis with Eclipse ESCET v4.0

Ivo Melse, s1088677

March 3, 2025

## 1 Objective

A supervisory controller controls the behavior of a cyber-physical system in order to ensure correct and safe behavior. It accomplishes this by making observations of the unsupervised system, which is referred to as the ”plant”, and interacting with it accordingly. In Figure 1, an overview is shown of how a supervisor could interact with the plant. The supervisory controller prevents the human operator from making operational errors that would result in unsafe situations.

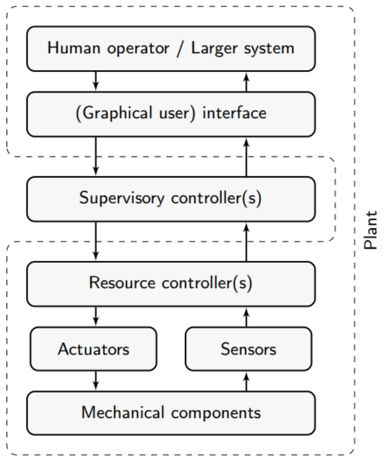
Human operator / Larger system
(Graphical user) interface
Supervisory controller(s)
Resource controller(s)
Actuators
Sensors
Mechanical components
Plant

Figure 1: Overview of the possible role of a supervisor within a bigger system

Synthesis-based engineering (SBE) is a method of constructing supervisory controllers, which is aimed at automating the engineering process as much as possible. Unlike in a more traditional model-based approach in which a requirement model is implemented by engineers and verified, the engineers only need to specify a formal model of the requirements and plants. Then a program is executed which generates a correct-by-construction supervisory controller based on these.

The Eclipse Supervisory Control Engineering Toolkit (ESCET) is an open-source project that provides an Eclipse-based toolkit for synthesis-based engineering. In particular, it uses the CIF language for modeling requirements and plants. It has been in development since 2020.

Briefly speaking, ESCET's supervisory synthesis works as follows: CIF code of the requirements and plants is specified as the input. Then, through a series of conversion steps, the CIF is converted to a single Extended Finite Automaton (EFA). The representation of this automaton is *symbolic*, which means that the states and transitions are not explicitly instantiated. Instead, predicates with underlying Boolean Decision Diagrams (BDDs) are used to represent the model. The main part of the algorithm then operates on this symbolic EFA by repeating a number of altering operations, until a fixed point is reached. The result is a model of a supervisory controller that is correct, safe and non-blocking.[1] This follows from the correctness of the algorithm. Finally, code that implements the resulting EFA in the engineering context can be retrieved by either code generation or manual coding.

The main goal of the paper is to provide an overview of CIF and ESCET and starting point for future researchers interested in improving it. This is implemented in four parts:

1. Describe CIF's symbolic synthesis algorithm in detail.
2. Describe a set of 23 benchmark models for synthesis-based engineering.
3. Describe performance improvements of symbolic synthesis between ESCET versions 0.8 and 4.0.
4. Discuss the possibilities of non-monolithic and multi-level synthesis.

## 2 Proposal

The paper does not propose many new ideas, although it does describe implementations of existing ideas, in particular in the section about performance improvements. This coincides with the goal of the paper.

[1]Here non-blocking means that the system is required to be able to reach a certain set of states. For example, non-blockingness prevents supervisory synthesis from generating a controller of a drawbridge that never allows it to close. This supervisor would be completely safe but useless.

## 3 Evidence

### 3.1 Performance improvements

Between ESCET version 0.8 and version 4.0, a number of performance improvements were made. The authors claim that ESCET version 4.0 is faster than version 0.8, and provide empirical evidence to support this in the form of measurements. The researchers present 23 benchmarks for SBE, some of which are academic and some of which are from industry. Each of these benchmarks consists of a model of the plant and requirements. For each benchmark, they provide the *reduction factor*, the factor by which the performance improved in version 0.8 as compared to version 4.0.

The reduction factors of the 23 benchmarks are presented in a table, and in a graph which is also shown in Figure 2. The performance is measured in terms of time and memory. Time is measured in the number of BDD operations, and memory is measured in the maximum number of BDD nodes. This way of measuring was chosen to ensure platform-independence of the results.

The reduction factors are far from uniform across the benchmarks. The best improvement was benchmark number 4, with a 4293.6 in terms of time and 518.3 in terms of memory. The worst improvement is benchmark 22 with a 2.1 in terms of time and 0.5 in terms of memory (ESCET 4.0 actually used twice the memory that 0.8 used). The researchers examine benchmark 22 in more detail and show that the performance of version 4.0 increases again when using settings that are optimal for this particular benchmark.

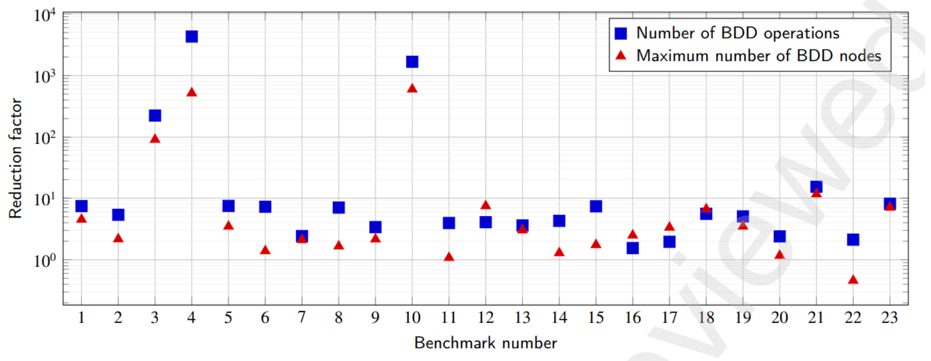

Figure 2: Reduction factors of the 23 benchmarks

In order to give an indication of the performance of ESCET v4.0 in practice, the researchers also provide the execution time in seconds of the benchmarks in ESCET version 4.0 on a particular computer.

The researchers conclude the section by considering some threats to the validity of their results, namely selection bias (because they used benchmarks that were developed in the academic community of which they are a part themselves), attrition bias (because the test cases were set up in a way that would allow them to finish on version 0.8 in reasonable time) and sampling bias (because the benchmarks that they used might not be representative for real world applications).

In general, I believe that the researchers provide valid evidence to substantiate their claim that ESCET v4.0 is faster than ESCET v0.8. They also paid proper attention to factors that could invalidate the results.

### 3.2 Algorithm correctness

In the third section, a detailed rundown of ESCETs symbolic synthesis algorithm is provided. Oddly enough, the authors do not provide a correctness argument, and do not even mention the correctness of this algorithm, even though their entire toolkit depends on the correctness of supervisory controller synthesis. They should have mentioned a previous paper by Ouedraogo *et al.* in 2012 explicitly. This paper proves the correctness. They do cite this paper, but they should have explicitly used its result as the argument for the correctness of CIF's supervisory controller synthesis.

## 4 Shoulders of giants

First of all, the paper depends on early work on automata theory and control theory during the by figures such as Church, Turing, Neumann, Kleene, etc.

More specifically, the paper uses supervisory control theory (SCT), also known as the Ramadge–Wonham framework. Ramadge and Wonham were the authors that originally considered supervisory control synthesis in the 1970s and 80s and developed some of the first synthesis algorithms. The authors refer to their work in the introduction.

As already mentioned, the supervisory controller synthesis algorithm in ESCET is an implementation of the algorithm that was proposed by Ouedraogo *et al.* in 2012. This work is (very briefly) mentioned in section 3.

In the section about the improvements that were made to ESECT version 4.0 relative to version 0.8, the authors cite relevant work in supervisory controller synthesis. For example, Fei *et al.* (2014), Lousberg *et al.* (2020). Both of these introduce some optimizations that were later implemented in ESCET v4.0.

Finally, the original ESCET paper is mentioned. The main author of the current paper is also an author of this previous paper.

## 5 Impact

In terms of citations and influence in academics, it is hard to assess the impact of the paper at the current time because it has not been peer-reviewed or published yet.

The paper will not be impactful in the sense of introducing new ideas. The paper is really more of a *progress update* for ESCET. I doubt that the paper will be cited a lot beyond the small academic community around ESCET. This does not mean that it is a bad paper. To the contrary, this paper will serve as a starting point for further improvements, as the authors intended.

Apart from academics, SBE and ESCET are already helping engineers to build better cyber-phisical systems. In a previous paper on ESCET, the authors refer to many successful case studies, and mention that Rijkswaterstaat is seriously considering adapting SBE.

## 6 Writing

The writing of the paper is of good quality. A pleasant mix of longer and shorter sentences is used. The paper uses a simple vocabulary (except for technical terms), and explains concepts in a clear and structured way. In section 3, an example is effectively used to explain the linearization and plantification of EFAs.

It would have been helpful to provide another example in section 3.8 where the main loop of the supervisory synthesis algorithm is explained. I found this section quite hard to follow.

## 7 Unresolved issues

The researchers mention that despite the improvement in performance, scalability might still be a problem for the practical usage of synthesis-based engineering. They point out that non-monolithic synthesis should be considered to remedy this. Non-monolithic controller synthesis does not generate a single supervisor like in the algorithm discussed in the paper. Instead, it creates multiple supervisors which are each responsible for the correct behavior of a part of the system. Their combined behavior then ensures the correct behavior of the entire system.

One instance of non-monolithic synthesis is called multi-level synthesis. In this technique, plants and requirements are automatically put into hierarchical groups. ESCET v4.0 already supports multi-level synthesis, but this support is still limited; only 11 out of the 23 benchmarks are supported. The researchers show some preliminary performance results for the supported benchmarks. From these, it appears that multi-level synthesis in ESCET has potential to improve perfromance further.

## 8 Verdict

Overal, I think the paper is of good quality. I expect it to be accepted in a peer-reviewed journal. I do think that the authors should do a small revision to mention the correctness of the algorithm, and provide an example in section 4.