Universiteit Antwerpen

Translating statecharts and class diagrams to DEVS

Sam Pieters

Principal Adviser: Hans Vangheluwe
Assistant Adviser: Joeri Exelmans

Dissertation Submitted in May 2024 to the
Department of Mathematics and Computer Science
of the Faculty of Sciences, University of Antwerp,
in Partial Fulfillment of the Requirements
for the Degree of Master of Science.

Contents

List of Figures

List of Tables

Nederlandstalige Samenvatting

Nederlandse samenvatting komt hier.

Acknowledgements

Write Acknowledgements here

Abstract

This master's thesis explores the translation of Synchronous Concurrent Constraint Datalog (SCCD) models into the PyDEVS (Python-based Discrete Event System Specification) framework, investigating its implications and applications. SCCD, a formalism for modeling concurrent systems, offers powerful expressive capabilities for describing complex interactions. PyDEVS, on the other hand, provides a versatile platform for discrete event simulation.

The primary objective of this research is to bridge the gap between these two simulation paradigms, facilitating the integration of SCCD models into the PyDEVS ecosystem. This translation enables the utilization of SCCD models within PyDEVS for simulation and analysis, thereby extending the reach of SCCD-based approaches to a broader audience of researchers and practitioners.

The thesis first presents an in-depth analysis of SCCD and PyDEVS, highlighting their respective features, strengths, and weaknesses. Subsequently, it proposes a systematic methodology for translating SCCD models into PyDEVS specifications, addressing challenges such as semantic differences, modeling paradigms, and computational efficiency.

To demonstrate the utility of this translation approach, the thesis showcases several case studies where SCCD models are successfully translated into PyDEVS and simulated using the PyDEVS framework. These case studies encompass diverse application domains, including concurrent systems, distributed algorithms, and communication protocols.

Furthermore, the thesis discusses the practical implications and benefits of integrating SCCD into PyDEVS, such as enhanced scalability, interoperability with existing simulation tools, and the ability to leverage PyDEVS's rich ecosystem of libraries and tools for simulation analysis.

Overall, this research contributes to advancing the field of discrete event simulation by facilitating the seamless integration of SCCD models into the

PyDEVS framework, opening up new avenues for research and applications in diverse domains requiring concurrent system modeling and analysis.

CHAPTER 1

Introduction

In today's complex and interconnected world, the modeling and simulation of concurrent systems play a pivotal role in understanding and analyzing various phenomena, ranging from distributed algorithms to communication protocols. Synchronous Concurrent Constraint Datalog (SCCD) has emerged as a powerful formalism for expressing and reasoning about concurrent systems, offering expressive capabilities to capture intricate interactions among components. On the other hand, the PyDEVS (Python-based Discrete Event System Specification) framework provides a versatile platform for discrete event simulation, facilitating the modeling and analysis of dynamic systems.

However, despite their individual strengths, SCCD and PyDEVS represent distinct simulation paradigms with differences in modeling semantics, computational approaches, and tooling ecosystems. This dichotomy poses challenges for researchers and practitioners seeking to leverage the advantages of both SCCD and PyDEVS for modeling and simulating concurrent systems.

This master's thesis aims to bridge this gap by exploring the translation of SCCD models into the PyDEVS framework, thereby facilitating the integration of SCCD-based approaches into the PyDEVS ecosystem. By enabling the simulation of SCCD models using PyDEVS, this research seeks to extend the reach of SCCD-based methodologies to a broader audience while harnessing the capabilities of PyDEVS for efficient and scalable simulation.

In this introduction, we provide an overview of SCCD and PyDEVS, highlighting their respective features, strengths, and applications. We also outline the motivation behind this research, discussing the potential benefits of integrating SCCD into PyDEVS and the challenges inherent in reconciling the differences between these simulation paradigms. Furthermore, we present an

outline of the thesis, detailing the structure and organization of subsequent chapters, which include an analysis of SCCD and PyDEVS, a methodology for SCCD-to-PyDEVS translation, case studies demonstrating the translation approach, and discussions on the implications and future directions of this research.

Overall, this thesis aims to contribute to the advancement of discrete event simulation by bridging simulation paradigms and facilitating the seamless integration of SCCD models into the PyDEVS framework, thereby fostering innovation and exploration in concurrent system modeling and analysis.

1.1 Definitions

1. **Synchronous Concurrent Constraint Datalog (SCCD):** A formalism for modeling concurrent systems, combining elements of constraint programming and database query languages to express constraints and concurrent actions in a synchronized manner.

2. **PyDEVS:** Python-based Discrete Event System Specification, a framework for discrete event simulation that provides a platform for modeling and analyzing dynamic systems using the DEVS (Discrete Event System Specification) formalism.

3. **Formalism:** A precise mathematical or computational framework used to represent and analyze systems, providing rules and syntax for expressing system behaviors.

4. **Discrete Event Simulation:** A simulation technique used to model systems where events occur at distinct points in time, with changes in system state triggered by these events.

5. **Translation:** The process of converting models or specifications from one formalism or representation to another while preserving essential characteristics and semantics.

6. **Modeling Paradigm:** A set of principles, concepts, and methodologies used for constructing models to represent real-world systems or phenomena.

7. **Semantic Differences:** Variations in meaning or interpretation between different formalisms or modeling approaches, which may require adjustments or transformations during the translation process.

8. **Computational Efficiency:** The ability of a simulation framework or algorithm to execute simulations with minimal computational resources, such as time and memory.

9. **Statechart:**

10. **SCXML:**

CHAPTER 2

Background

2.1 SCCD

2.1.1 Statechart

2.1.2 SCXML

2.2 PyDEVS

2.2.1 DEVS Formalism

CHAPTER 3

Methodology

The methodology employed in this thesis outlines the systematic approach taken to translate System of Communicating Concurrent Deterministic Finite-State Machines (SCCD) into Discrete Event System Specification (DEVS). This section provides an overview of the strategies, tools, and processes utilized to achieve the translation objectives.

The translation of SCCD to DEVS involves converting models represented in one formalism into another while preserving their essential behavioral characteristics. Such translation is critical for interoperability, model verification, and simulation interoperability between systems designed using different modeling paradigms.

In this chapter, we present the overarching methodology for translating SCCD to DEVS, detailing the steps involved, challenges encountered, and the rationale behind the chosen approaches. Additionally, we discuss alternative approaches considered during the development process, providing insights into why certain strategies were pursued while others were discarded.

The methodology encompasses various stages, including initial approach exploration, tool selection, implementation workflow, validation, and verification. Each stage plays a crucial role in ensuring the accuracy and effectiveness of the translation process.

Furthermore, the methodology emphasizes the importance of evaluating the translated DEVS models against predefined criteria to assess their fidelity to the original SCCD specifications. By adhering to a systematic methodology, this research aims to contribute to the advancement of model translation techniques and facilitate seamless integration between different modeling formalisms.

This chapter serves as a guide to the methodology adopted in this thesis, providing readers with a comprehensive understanding of the processes involved in translating SCCD to DEVS.

3.1 Translation Tools

The translation of System of Communicating Concurrent Deterministic Finite-State Machines (SCCD) to Discrete Event System Specification (DEVS) involves the use of specialized tools and techniques to facilitate the conversion process. In this section, we discuss the evolution of the translation tools employed in this research, from initial parsing methods to the adoption of a visitor pattern for seamless translation.

Initially, the translation process began with a straightforward approach of parsing the SCCD models represented in XML format line by line. While this method provided basic functionality, it quickly became apparent that it was not efficient or robust enough to handle the complexities of SCCD models and their translation to DEVS.

Upon further investigation, it was discovered that the SCCD compiler (TODO: reference) already implemented a visitor pattern for translating XML files into Python and Java representations. Leveraging this existing pattern proved to be a significant breakthrough in the translation process. By utilizing the visitor pattern, which allows for the traversal of complex data structures without modifying the structure itself, we were able to streamline the translation process and improve its efficiency and accuracy.

The visitor pattern provided a structured and modular approach to translating SCCD models to DEVS, enabling clear separation between the translation logic and the underlying model representation. This decoupling of concerns allowed for easier maintenance, extensibility, and reuse of the translation code.

An additional advantage of adopting the visitor pattern for XML to DEVS translation was its compatibility with the existing compiler infrastructure of SCCD. The modular nature of the visitor pattern facilitated seamless integration of the DEVS translation functionality into the existing compiler framework of SCCD. This integration allowed for the incorporation of DEVS translation as a standard feature within the SCCD compiler, providing users with a comprehensive toolset for model development and analysis.

By adopting the visitor pattern for XML to DEVS translation, we were able to overcome many of the limitations encountered with the previous parsing methods. The visitor pattern facilitated a more systematic and reliable approach to translating SCCD models to DEVS, ultimately enhancing the

quality and fidelity of the translated models.

In summary, the evolution of translation tools from basic XML parsing to the adoption of the visitor pattern reflects the iterative nature of the research process, wherein insights gained from experimentation and exploration lead to the refinement and improvement of translation techniques.

3.2 Considered Translation Approaches

3.2.1 First approach: Implementing everything into one AtomicDEVS

3.2.2 Second approach: Setting every component to AtomicDEVS

The first approach was to set every possible component (Controller, ObjectManager and classes) to an AtomicDEVS model. This proved to be impossible with the standard atomicDEVS because in SCCD, statecharts can be created at runtime. The classic AtomicDEVS does not allow to create AtomicDEVS objects at runtime and thus mapping statecharts to a corresponding AtomicDEVS is impossible.

CHAPTER 4

Implementation

Blablabla

4.1 Definitions

Some introductory section

CHAPTER 5

Examples

Blablabla

5.1 Definitions

Some introductory section

CHAPTER 6

Conclusions

Example of todo: TODO: Write conclusions here.

Bibliography

Serge Demeyer, Stéphane Ducasse, and Oscar Nierstrasz. *Object-Oriented Reengineering Patterns*. Square Bracket Associates, 2008.

Appendices

APPENDIX A

An appendix

An Appendix is just like another chapter.