Example Thesis updated by Tosin Adewumi

Johan E. Carlson

Dept. of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden

Supervisors:

Name of your supervisor(s)

To my surprise...

Abstract

This is an abstract, imported from the file *abstract.tex*.

Changes

Enhancements in this template (from the original by Johan E. Carlson) include:

1. Update of the SRT full dept. name on the 1st page in the file cseethesis_example.
2. A change to defaultbibliographystyle{**plain**} in the cseethesis_example file instead of the old, which gives blank in-text citation.
3. Inclusion in cseethesis_example of makeglossaries, the example: newacronym{nlp}{NLP}{Natural Language Processing} for handling acronyms and the closing printglossary at the end of the file.
4. Example usage of the acronym package in the paper1 file: Natural Language Processing (NLP) is the full version of this short version NLP and the long version Natural Language Processing.
5. Addition of **newcommand{\makepapertobesubmitted}** in cseethesis file for manuscripts yet to be submitted.
6. Figure included on the front page in the file cseethesis_example since the old logo alternative is tricky.

Contents

Acknowledgments

The creation of this template has taken several years, and the shape it is in now would not have been possible with out the patient testers out there. To mention just a few who found and reported bugs, and occasionally even provided bug fixes: Gustav Johansson, Sara Sandberg, Yvonne Aitomäki, Fredrik Hägglund, Jesper Martinsson, Patrik Pääjärvi, and Martin Sehlstedt. To all of those I forgot to mention, please accept my apologies.

Luleå, June 2009
Johan E. Carlson

Improvements by Tosin Adewumi, October 9, 2021.

Part I

Chapter 1

Thesis Introduction

"This report, by its very length, defends itself against the risk of being read."

Winston Churchill

1.1 General information

Current version of the `cseethesis` document class is: 3.1.
Last modification: October 9, 2021

As of version 3.0, the template is no longer backwards compatible.

1.1.1 About the document class

This document class was originally created in 2002 when I was working on my own PhD thesis. Since then, many people used it, found bugs (and occasionally even corrected them), and suggested improvements.

The style is tailor-made for the typical types of theses that we write at the department, i.e. an introductory part followed by a collection of published or submitted research papers.

The template supports the use of both LaTeX and pdfLaTeX. If you use the command `\includegraphics` to import your figure and you supply the filename with its extension (e.g. .eps, .pdf), compilation should be possible with either one. For this to work, both .eps and .pdf versions of all figures must be available.

The template is totally free to use, modify and distribute, as long as reference to the original author is kept and as long as all files remain in the package. Modified versions can only be distributed if it is clearly mentioned in the document class that modifications have been made and by whom.

The whole package comes AS IS. I will correct bugs every now and then, but other than that, don't expect any support whatsoever.

1.1.2 About this document

This document, as well as the actual LaTeX code for it, makes up the documentation on how to use the document class.

Only this chapter contains any readable information. Chapters 2 and 3 are only included as examples of some of the features of the template. The text is nonsense, but the corresponding LaTeX code may be of some use. The same goes for the appended papers, which are only there as examples of a few options of the document class.

Read this chapter carefully. If you have comments on what else should be in here in order to simplify the use of the document class, let me know.

1.2 Chapters

1.2.1 Defining chapters

In order to add flexibility to the template, a new command, called `\makechapter` is provided. This command takes three mandatory and one optional argument, as

```
\makechapter[optional quote]{page header}{toc entry}{Chapter title}
```

The reason this is solved like this is to allow for shorter page headers if the chapter name is very long. Also, if the actual chapter heading needs to be manually split in several lines (if the automatic splitting does not look so good), the table of contents (toc) entry might have to be defined differently. Note that normally, the last three arguments can be the same.

The use of an optional quote as an introduction to the chapter is demonstrated in this chapter. It can just as well be left out, which is demonstrated in this document (see the code).

1.2.2 Importing chapter contents

The sub-documents containing the chapters should start directly, i.e. they must not contain any \begin{document} or \end{document} tags.

See this file, *chapter1.tex* for details.

1.3 How to append papers

Papers are included using the `\input` command, just as with chapters. You have to typeset paper title, authors, and abstract manually. See the example papers accompanying this document for an example.

To make the separator sheet preceding each paper, use one of the following commands:

- `\makepaper` – Published paper.
- `\makepaperaccepted` – Accepted, not yet published paper.

- \makepapersubmitted – Submitted, not yet accepted paper.
- \makepapertobesubmitted – Not yet submitted paper.

See code for this example document for examples on how to use.

1.4 Cross-references

All labels throughout the thesis have to be unique. If the same equation or figure shows up twice e.g. a figure used both in the introduction and one of the included papers), it has to be given different labels. Referring to labels is done as usual.

A simple trick to make sure this is the case and that will also help you keep track of all labels you used is to use the following naming convention:

- ch1:fig:labelname, ch1:tab:labelname, ch1:eq:labelname, etc. all denote figures, tables and equations in Chapter 1.
- paperA:fig:labelname, paperA:tab:labelname, paperA:eq:labelname, etc. all denote figures, tables and equations in Paper A.

For existing text, e.g. papers, this is easily achieved by a simple search-and-replace operation on the string \label{. Any text editor will do that for you!

1.5 Appendices

It is possible to have any number of appendices for each chapter. Simply type \appendix before the first appendix, which is then a normal section, but numbered differently.

To add appendices to papers, use the \paperappendix command

1.6 Including bibliography lists

In a thesis one might want several separate bibliographies. For example, one for the first part, and then separate bibliographies for each of the included papers.

This is solved using the bibunits package together with a slight work-around in this template. For the first part of the thesis, there is only one bibliography list, typeset like a chapter (see this example document). In the papers, the bibliography lists are typeset as un-numbered sections. See this file cseethesis_example.tex and paper1.tex for examples how to place the bibliographies.

Note that the command

- \makebib is used in Part I, to typeset the reference list in the thesis introduction.
- \putbib is used in the papers in Part II.

1.7 How to compile your project

Finally, you probably like to know how to build your project to a final PDF or PostScript file. Start by verifying that you can compile this document. This is how it goes:

1. Run LaTeX (or pdfLaTex) once.
2. Then run BibTeX on all the bu¡i¿ files.
3. Run LaTeX twice more, to build the final DVI document (or pdfLaTeX if you want a PDF file).

The above steps are easily collected in a script or batch file. See the files `make.bat` and `compilebibunits.bat` for examples.

1.8 Revision history

The template has evolved during several years and the exact revisions are not clear to anyone. Starting from version 1.6, however, the changes are more well-documented. This example document will always support only the latest release of the template. Below is a list of the revisions made to the document class (and when applicable, the example document):

- Version 3.1, September 1, 2010
  - Fixed bug related to appendix numbering.
  - Fixed page numbering of "Part" pages.
  - Added a comment at top of cseethesis_example.tex, for improved compatibility with some editors.
- Version 3.0, June 7, 2009
  - Fixed bug related to page headers in chapters containing no subsections.
  - Removed the EU class option and replaced with a logo argument to the preamble. See the code of this document for an example.
  - The template is **no longer compatible with previous versions**.
  - Removed the definition of boldface Greek letters from the document class, since this is not the proper place for that.
- Version 2.5, March 5 2009
  - Renamed the template *cseethesis*. It is a continuation of the project initially called *eisthesis*, but since its use has spread I decided to change the name.
  - Various minor bug fixes.

  - Update of example document, making examples of additions and revisions in recent versions of the template.

- Version 2.36: Added "Part" to the table of contents.
- Version 2.35:
  - Fixed a bug regarding the page headers in the "appended papers part".
  - Fixed a bug causing the section numbering to be wrong in a chapter succeeding a chapter containing appendices.
  - Added a chapter 3 in this example document, illustrating how to handle page headers for chapters without any sections. See the code at the top of `chapter3.tex`.
- Version 2.3:
  - Added the commands `\appendix` and `\paperappendix`, see section 1.5.
  - The bibliography list is now typeset similar to a chapter in Part I, and as an un-numbered section in Part II. This required the use of separate commands for including the lists (see Sec. 1.6)
  - Typesetting fixes for the table of contents page. As a consequence, the package `titletoc` is now required.
  - Minor other code cleanup and bug fixes.
- Version 2.25:
  - Fixed a minor bug that used to generate a warning message regarding font shapes in the page headers.
- Version 2.2:
  - pdf and eps class options removed. The document class compiles with either pdfLaTeX or LaTeX.
  - The reference lists in papers are now typeset in the same way as in Part I of the thesis.
  - Some minor adjustments of page header heights.
- Version 2.1:
  - Cross-references to chapters now work like they should. See the main document of this example.
  - Major bug fixes to BibTeX reference lists, table of contents generation etc.
  - The only change the user has to do is to use the new `\makebib` instead of bibunits' `\putbib`.

- Version 2.0:
  - Support for EU logotype on main page. The files `eu1_f_eng.pdf` and `eu1_f_eng.eps` must be placed in the same directory as the document class.
  - Support for pdf class options.
  - Update of the \`makechapter` command. It now requires three arguments. See main document for example.
  - Support for BibTeX, using the `bibunits.sty` package.
- Version 1.6: Various bug fixes to figure spacing etc.

Chapter 2

Title appearing on the chapter start page

2.1 First section of the second chapter

This is the text of the second chapter. []

2A This is an appendix section

Text of the appendix

2A.1 Subsection 1

Yet some text, and an equation

$$\text{abs}\left(e^{j\pi}\right) = ? \tag{2A.1}$$

2A.2 Subsection 2

And then some...

2B This is another appendix section

This section concludes the appendix.

Chapter 3

Nonsense chapter, here only to verify that some issues in previous versions are really resolved!

This is a chapter with no sections, only here in order to test the document template. Please ignore the rest of this. dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd

fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd

oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdso-

hew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd

dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd

dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo dkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfodkhgfdsh fdsjhgiepy kdslkfds ewiuyfe fkjfdsohew dskjhfd fduew dsk di e sd lkjd dslkfd oiew sao cdkåwq acslkgå sdspo dsjdspe dkfo

Chapter 4

Nonsense chapter, here only to verify that some issues in previous versions are really resolved!

References

Part II

Paper A

The Title of the Papers in the Thesis are Automatically Split In Several Lines if Necessary

Authors:
John Doe and Jane Doe

Reformatted version of paper originally published in:
Example Thesis, Internal Report, Luleå University of Technology, 2002.

The Tile of the Paper

John Doe and Jane Doe

Abstract

Abstract text of the paper...

1 Introduction

The text of this article is imported from the file *paper1.tex*. The title and abstract part above are typeset manually (see file for code template). [**?**]

Natural Language Processing (NLP) is the full version of this short version NLP and the long version Natural Language Processing.

Be sure NOT to have any \begin{document} or \end{document} tags in the imported files.

Some references here too, just to show the use of bibunits .

A First appendix of paper A

Some appendix text.

A.1 A subsection of the appendix

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t}\, dt. \tag{A.1}$$

A.2 Another subsection of the appendix

Test subsubsection

B Another appendix

Some text in the second appendix

Paper B

Cyber-physical automation systems modelling with IEC 61499 for their formal verification

Authors:
Midhun Xavier, Sandeep Patil, Valeriy Vyatkin

Reformatted version of paper accepted for publication in:
Proceedings of the IEEE International Conference on Industrial Informatics (INDIN), 2021.

1 Introduction

Distributed industrial automation systems pose a significant challenge for their efficient verification and validation due to their heterogeneous structure, use of wireless communication and decentralised logic. The inherent inter twinning of computational and communication processes with complex physical dynamics has called for the term cyber-physical systems (CPS) [13] to emphasize the challenges and the need for new development approaches.

The IEC 61499 architecture [2] is getting increasingly recognised as a powerful mechanism for engineering such systems. It has been proven also as an efficient way of modeling CPS in automation [7].

The challenge of IEC 61499 verification has been well-recognized from the early stages of the standard's development and evaluation [16]. Closed-loop modelling has been proposed for the most comprehensive verification [18], which implies the need for modelling the plant.

In quite many works, the plant modelling [4] was done in the same formalism, which was used eventually to represent the model for the model-checker. Graphical modelling languages of finite-state machines and Petri nets [3] were used in particular, and the models were prepared using the corresponding graphical editors. However, the IEC 61499 itself provides a graphical engineering interface and supports programming in terms of state machines. Therefore, a problem-oriented notation could be proposed to take advantage of the existing tools and avoid using additional ones in the process of modelling. This paper proposes such an approach by introducing a tool chain.

The paper is structured as follows: Section 2 discusses the related work and problem statement. Section 3 and 4 illustrates an example and simulation model in detail. Section 5 describes the discrete-state modelling approach including the implementation of non-deterministic transition in smv. Section 6 gives an overview of the fb2smv tool's functionalities and features. Section 7 presents the results and analysis of the work. Finally, Section 8 concludes the paper and outlines future goals.

2 Related works and problem statement

Christensen suggested a model-driven development approach [5] for distributed automation systems that is based on the use of the model-view-control object-oriented design pattern. The approach supports several development stages, from simulation in the loop to the deployment [14]. In [18] that approach was extended to also include formal verification into the verification and validation of function block systems.

The suggested framework is heavily based on the closed-loop architecture of the model, where the plant part is explicitly represented in the overall system model. This architecture allows for easy integration with a simulation model for the virtual commissioning purposes, which can be seamlessly converted to the deployment configuration. In addition, the closed-loop configuration could be transformed to a structurally similar formal model, appropriate for more exhaustive verification by means of model-checking. The

concept of an integrated environment VEDA presented in [17] had already supported closed-loop verification of IEC 61499 function block systems. While the controller parts of the closed-loop models were automatically translated into the corresponding formal model in a Petri-net like language, the model of the plant parts had to be developed manually in the same formalism. This made the process of model creation quite difficult, not allowing for systematic use of the tool by control engineers. The emergence of the automatic model generator fb2smv [1] that is capable of creating SMV models from IEC 61499 function block systems, promises increased potential of formal verification on account of using the industry grade model-checkers of the NuSMV [6] and NuXMV family. However, the problem of creating the model of plant in SMV remains to be the limiting factor [15] for industrial application of the corresponding verification tool-chain.

In this paper, we attempt to overcome this hurdle by proposing a CPS modelling method that is entirely based on IEC 61499. By means of the same modelling language, we represent both simulation models of the plant, which are equivalent to hybrid automata, and the models for formal verification, which are equivalent to discrete-state automata with non-determinism. The latter model can be derived from the former by applying a sequence of transformation steps.

Both kinds of plant models, implemented as IEC 61499 function blocks can be included in the multi-closed-loop model connected to the real controllers. This opens the opportunity of checking the distributed control logic of CPS in simulation and model-checking.

3 Illustrative example

The modelling method and tool-chain used for CPS verification are illustrated in this paper using a laboratory-scale distributed automation system "Drilling station", described in the next subsection. In this case study, we selected and created a formal model for the same system. Implementation of formal models of real systems as well as verification is done with the help of the tool chain.

Figure 1: Drilling station system.

The drilling station system in Fig. 1 is composed of several mechatronic components, among which, in this study we selected only the Drill and rotating Table. It is assumed that the mechatronic components are smart, i.e. they are equipped with their own control devices, implementing their basic operations, which are as follows.

The Drill moves in an upward or downward direction. Whenever a workpiece is detected by the sensor under the drill, it moves downward and starts drilling. Once it completes drilling, it moves upwards and rests at the home position.

The Table rotates from one fixed position to another. The cycle is completed when it rotates six times. When a workpiece is placed in the loading positions, the table rotates to bring it under the drill.

The control logic of each mechatronic component is implemented as a function block which follows the IEC 61499 standard. The function block diagram shown in Fig. 2 consists of the two function blocks orchestrated to work together by means of event and data connections between each other and sources and sinks of sensor inputs and actuator outputs (function blocks on the left hand side and right hand side respectively). A close-up on the interacting controllers is presented in Fig. 3.

It is assumed that the smart mechatronic components are delivered by their vendors together with the software components for implementing their control logic. They are integrated to the drilling station in a way, assuming that the control of internal operations in each mechatronic component is implemented by its predefined function block, and the integrator tries to minimise its software development effort by reusing the software components received from the vendors. This approach requires exhaustive testing of the orchestrated system on compliance with functional and non-functional requirements.

Hence, we will use both simulation in the loop with the plant model and formal verification in closed loop for more exhaustive exploration of the state space.

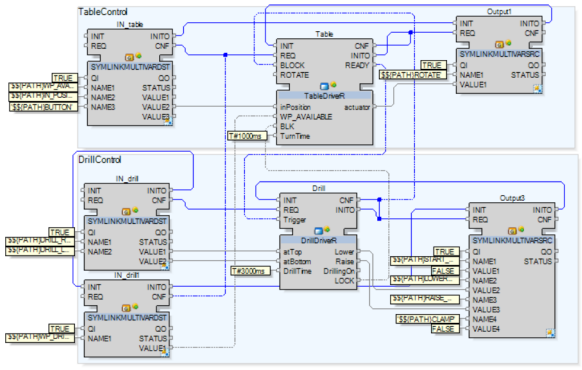

Figure 2: Function block representation of the distributed automation of drilling station.

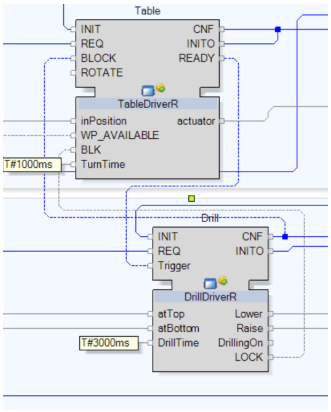

Figure 3: Close-up on the decentralised controllers' interaction.

4 Simulation Model

The simulation-in-the-loop environment is shown in Fig. 4. The original function blocks containing the autonomous control logic of drill and table are connected with the function blocks implementing simulation models of the drill and table respectively. The controllers are also connected with each other in exactly the same way as in the real configuration in Fig. 3. For example, the function block TableMod11 of type TableModTop represents a model of the table that is driven by one control signal `fwd`. When this signal receives the value `TRUE`, the simulated table starts continuous rotation clockwise. The rotation is stopped if `fwd` resets to `FALSE`.

The model outputs the Boolean value `FixPos` which becomes `TRUE` when the table comes to one of the six fixed positions. To keep the table in the fixed position, the controller has to stop motion by resetting the control signal `fwd` to `FALSE`. Besides, the model produces the `WP_DRILL` signal, which is the reading from the sensor indicating the presence of a workpiece under the drill.

The simulation environment reproduces the working behavior of the real plant which helps to visually identify the behavior of the system before deploying the distributed control to the real hardware. Besides, the errors identified during the formal verification can be represented in simulation.

The simulation models, used in this configuration, have continuous dynamics, which is the time-domain implementation of hybrid state machine as discussed in [18]. The core part of the plant simulation function block is based on the hybrid automaton model of the process, as illustrated in Fig. 5. The state machine has three states corresponding to the static position of the moving object, such as the vertical axis of the drill, or the rotating table. These are `stHOME`, `stEND` and `stSTOP`. There are also two dynamic states, when the coordinate of the object is changing: `dMOVETO` and `dRETURN`.

When the state machine is in the one of the dynamic states (say, `dMOVETO`), it emits the START output event which invokes the external `E_CYCLE` FB, which starts emitting periodic events, activating the function block with the state machine. The state machine remains in the `dMOVETO` state until the position reaches the end position, i.e. Pos=DIST.

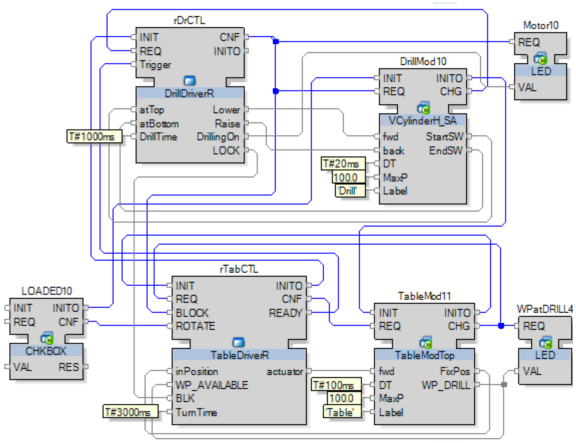

Figure 4: The Function Block representation of the simulation-in-the-loop configuration.

Until then the loopback transition condition is true, so the state-machine remains in the `dMOVETO` state. Every time the loopback transition is executed, the event CHG is emitted which also invokes the external "Integrator" FB, which recalculates the new value of the process variable based on the current value. The duration of the time interval between recalculations is determined by the `DT` parameter of the `E_CYCLE` and speed of the motion.

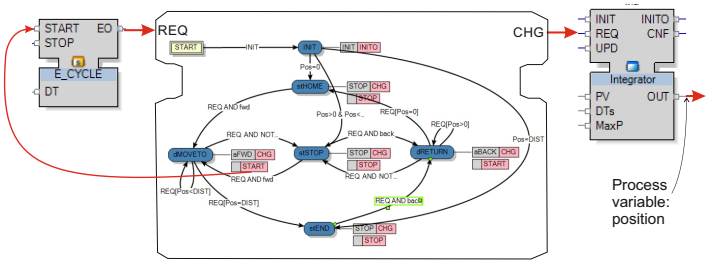

Figure 5: Computational implementation of a hybrid automaton in function block.

Given the ever changing process values, the evolution of the model can be visually displayed using internal or external means. The development reported in this paper was done using the NxtStudio of NxtControl, which offers a proprietary visualisation technology called `CAT`. The plant model blocks were implemented as the `CATs`, therefore the model behaviour was implemented internally, within the same development environment. The interactive system visualisation by means of CATs is shown in Fig. 6.

5 Discrete-state Modelling Approach

The discrete-state model of the system is created in IEC 61499 based on the simulation model described in the previous section.

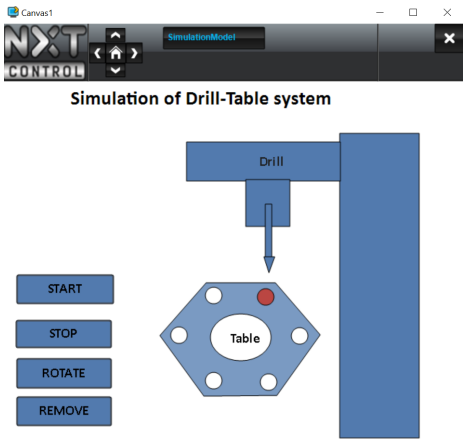

Figure 6: The visual representation of the simulation process.

The discrete-state equivalent of the simulation configuration is shown in Fig. 7. Here the function blocks simulating the drill and table are substituted by their analogs operating in the discrete state domain, instead of modelling the continuous process parameters, such as the drill's and table's numeric position. The model can be then simulated in the IEC 61499 IDE with values of function block inputs and outputs displayed and modified interactively. It can be translated to the SMV model using the fb2smv tool and exposed to the formal verification by model-checking.

5.1 Notation for Plant Modelling

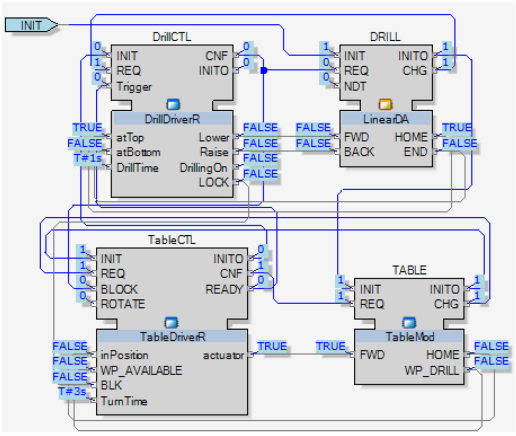

Figure 7: Discrete state function block model of the Drilling station.

In the current example, the drill model is represented by an instance of a basic discrete motion model LinearDA. Its state-machine implementation is shown in Fig. 8. The

execution semantics of the state-machine follows the rules of IEC 61499, i.e. the function block is activated by an input event, and the state machine evolution is following the rules for execution control chart (ECC) of basic function blocks.

Similarly to the hybrid state-machine in Fig. 5, the discrete state model of the drill specifies three static and two dynamic states, but it does not model the position as a numeric value. The drill moves from `stHOME` to `stEND` state via a motion state called `ddMOVETO`. The state transition occurs from stHOME to ddMOVETO whenever FWD signal is TRUE. It is remarkable to note the NDT event input of the LinearDA function block, which remained unassigned in the application in Fig.7. The NDT is reserved in the proposed modelling notation for Non-Deterministic Transition. Whenever the formal model generator encounters NDT in the state machine, it will interpret it accordingly. For example, the SMV modelling of NDT will be described in section 5.3

In terms of our model, the use of NDT in the transition from the motion state `ddMOVETO` to the static state `stEND` models the unknown duration of the motion from one state to another. Plant may fail whenever the `FWD` and `BACK` signals become `TRUE` simultaneously, so we introduced an ERROR state which helps to identify whether the controller creates this scenario.

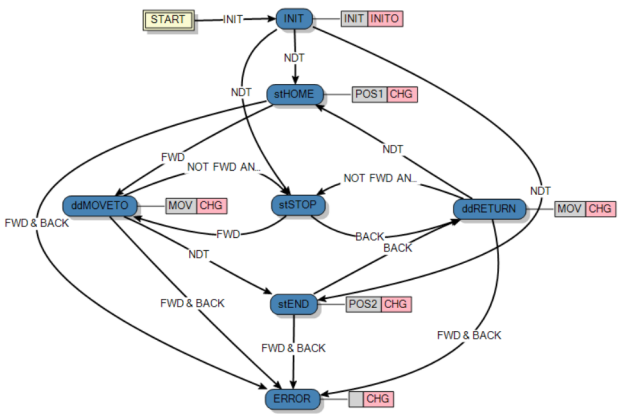

Figure 8: Discrete state linear motion process model with NDT.

5.2 Non Deterministic Transitions in controllers

Non-deterministic transition can be also helpful for simplification of controller models containing timers. For example, in our case study, the controllers were developed as state machines with timeouts, therefore they are implemented in composite function blocks. In the drill, drilling process needs to be done for several durations, which is achieved with the help of The `E_DELAY` function block. The composite function block consists of a real controller and `E_DELAY` function block as shown in Fig. 9(a).

However, formal modelling of the timers in SMV is computationally hard. It can be

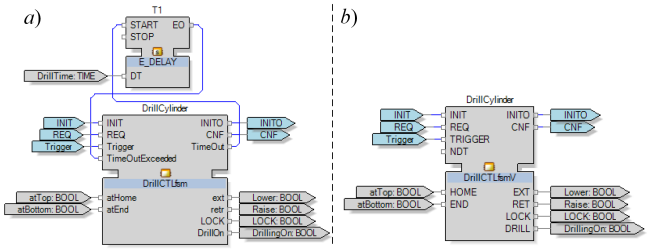

Figure 9: a) The real drill controller with external timeout. b) The interface of modified drill Controller with non-deterministic transition input.

avoided if the concrete delay duration was substituted by non-deterministic transitions with the help of NDT signal. Therefore, the controllers can be modified this way in order to reduce complexity of model-checking. Therefore, we removed the timeout `E_DELAY` substituted the corresponding input of the function block and added NDT input. The execution control chart of the real drill controller and the modified drill controller are shown in Fig. 10.

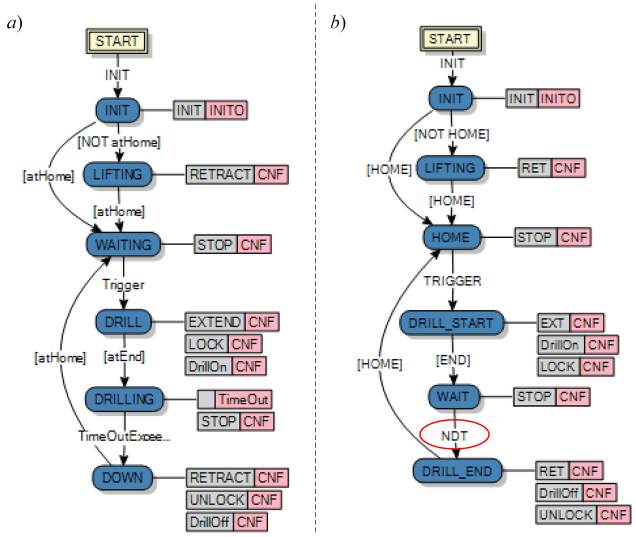

Figure 10: a) The ECC of the real drill controller; b) The ECC of the modified drill controller with a non-deterministic transition modelling the time delay.

5.3 Modelling of non-determinism in SMV

SMV provides a way to accomplish non-deterministic choice by providing a set of values to the signal. The first statement is used to declare the variable NDT as a Boolean type and the second statement is used to initialize the NDT variable to either TRUE or FALSE value.

```
1 | VAR NDT:= boolean;
2 | init(NDT):= { TRUE, FALSE };
```

In every transition we are giving a provision to choose either TRUE or FALSE. This makes the NDT variable unpredictable in each transition.

```
3 | next(NDT):={ TRUE, FALSE };
```

Implementing non-determinism in every transition can be limited by introducing conditions in the next statement. If it is not required for the NDT variable to choose values in every transition then we can design like below:

```
4 | next(NDT):= case
5 |      Condition: { TRUE, FALSE };
6 |      TRUE : NDT;
7 | Esac;
```

6 fb2smv tool

The **fb2smv** tool [1] is a model generator for generating SMV models of function block systems in IEC 61499. It is a part of formal verification tool-chain, that includes the model checker NuSMV and the tool for counterexample analysis in terms of the original FB system.

The tool implements the formal model of IEC 61499 as per the modelling method present in [8]. In order to construct `SMV` code, the **fb2smv** tool uses Abstract State Machine (ASM) [12] as an intermediate model. The tool takes IEC 61499 function blocks expressed in XML format as input and generates a formal model with the help of ASM semantics. According to [8], the structure of `SMV` code consists of the declaration part and the rules part, which are the ASM rules.

The tool converts basic and composite function blocks and also includes more additional features like, limiting the boundaries of variables to reduce the state space, changing execution order of FBs, deciding the input event priority by changing its order etc. The proposed non-deterministic transitions notation has been added to the tool as a result of this work.

7 Results and Analysis

In this paper, we demonstrated on a case study example, the use of IEC 61499 language for design and formal verification of cyber-physical automation systems. First, we designed the control logic of each mechatronic component in a drilling station and then implemented it as a function block in IEC 61499 standard. The simulation environment of the drilling station is developed with the help of the same controllers which are used in the real configuration. We reproduced the same plant behavior in the simulation model using the function blocks. In order to create a formal model of the system, we

transformed existing function blocks of the plant models and controllers by adding non-deterministic transitions, using the proposed NDT notation. Using fb2smv we converted the functional blocks to SMV code, which was verified using NuSMV on a machine with Intel(R) core(TM) i7-10510U CPU @1.80GHz 2.30GHz with 32GB RAM. The simulate feature of NuSMV was used to check the formal model's working behavior. Simulating the SMV code in interactive mode gives us the provision to go through each state by giving appropriate sensor inputs. We can try all possible input combinations to verify whether the system behaves as we expected. We can also randomly simulate the traces or we can manually go through each state by selecting different input combinations.

The main drawback of simulation is that the number of possible behaviors can be too large or even infinite. Simulation can show the presence of bugs, not their absence.

More comprehensive verification can also be done by model-checking various properties, i.e. checking whether a requirement is true or not in all possible execution traces of control logic. This will allow us to test critical scenarios where there can be failures in some combination of inputs. The formal model of the system was verified with help of CTL [11] specifications. While testing the CTL specifications in NuSMV, we found the following statement is false so it is possible that the table can rotate while the drilling process is going on.

```
-- specification  G !(DRILL_TABLE_CFB3_inst.DrillCTL_RET = TRUE &
   DRILL_TABLE_CFB3_inst.ActuatorGen_EO = TRUE)
```

While executing the above specification, the NuSmv gave a counterexample that contradicts this statement. The counterexample generation for the above specification took 26000 seconds to complete. The counterexample helps to identify the error in the controller's design. We tested the same logic in the simulation model, the table rotated from its current position to another while drilling the workpiece. The real system also exhibits the same issue.

To fix this issue we need to analyze the counterexample provided by the NuSMV, but identifying the variables changed in each iteration is difficult. The Nutrac tool provides a better way to understand each state. The Nutrac tool converts the counterexample to a CSV file. This CSV file's columns represent states and rows represent input/output events or data input/output variables. We analyzed the CSV file and identified the issue. The issue was present in the execution control chart of the table's controller. It is required to add the BLOCK signal and it should be checked before moving from DRILLED state to REMOVED state. We verified the CTL specification again and this time NuSMV gave the TRUE result. The modified controller is tested in the real object, as well as in a simulation system and it was behaving as we expected i.e table was not able to rotate while drilling is going on.

In order to identify whether the controller produces the `FWD` and `BACK` signals true at the same time, the following specification is used.

```
-- specification  G !(DRILL_TABLE_CFB3_inst.DRILL.Q_smv = ERROR_ecc)
```

After executing the above specification, the NuSMV produced the output as TRUE. The ECC of the drill model never goes to ERROR_ecc state i.e. the controller never

produces `FWD=TRUE` and `BACK=TRUE` condition simultaneously.

This paper [17] proposes a software tool called VEDA (verification environment for distributed application), which is used for closed loop modelling and verification of distributed control systems in intelligent manufacturing. Manually developed plants and automatically generated controllers are combined for model-based verification in IEC 61499 standard but it requires higher integration effort. The paper [9] [10] describes a closed loop system with a simulation model used to autogenerate SMV but the resulting complexity was prohibitive. In this paper, we propose a methodology to create a closed-loop model staying within IEC 61499 standard which produces lower complexity of model-checking than [9] [10] and reasonable engineering effort that is less than in [17].

8 Conclusion and Future Work

In this paper, we proposed a tool chain which helps to verify and analyze the function blocks implemented in IEC 61499 standard. This tool chain can be used for continuous development and evaluation of distributed control systems. With the help of this tool chain, it is possible to test the system quickly and efficiently. The accurate implementation formal model is necessary to identify all possible flaws in the system. The existing functionalities of fb2smv along with the non-deterministic transitions in function blocks help to provide a similar formal model of the system. Previously, the counterexample analysis was complicated but now the Nutrac tool solves this issue by giving a better representation of counterexample in CSV format. The developers working on complex system design can use this tool chain for continuous development and testing.

The non-deterministic transitions in function blocks can be extended by introducing NDT as a variable to the FBs. This NDT variable can be any of any IEC 61499 data type but the NDT variable should be able to choose one value from set values. For example, if we introduce NDT as a variable of integer data type and its values limited 0 to 5 then it should be able to randomly select one value from 0 to 5. In order to introduce more randomness to our formal model, it's better to implement NDT as a variable instead of using NDT as an event signal. An interesting field to explore is if we can generate the specification as well as the plant model automatically then existing manual interventions can be avoided. The tool chain which identifies all possible errors and fixes them automatically could be the next step in the future.

The models used in this paper are not based on time so the timing problems due to different set of time scales of controller and plant cannot be identified by this approach. The extension of notation to timed automata can be added for future work.

9 Acknowledgements

This work was sponsored, in part, by the H2020 project 1-SWARM co-funded by the European Commission (grant agreement: 871743).

References

[1] fb2smv model generator.

[2] IEC 61499-1: Function Blocks Part 1: Architecture, 2012.

[3] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE transactions on software engineering*, 17(3):259, 1991.

[4] Igor Buzhinsky and Valeriy Vyatkin. Plant model inference for closed-loop verification of control systems: Initial explorations. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 736–739. IEEE, 2016.

[5] James H Christensen. Design patterns for systems engineering with IEC 61499. *Verteilte Automatisierung-Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung*, page 63–71, 2000.

[6] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, and Marco Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[7] Wenbin Dai, Cheng Pang, Valeriy Vyatkin, James H Christensen, and Xinping Guan. Discrete-event-based deterministic execution semantics with timestamps for industrial cyber-physical systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(3):851–862, 2017.

[8] Dmitrii Drozdov, Victor Dubinin, Sandeep Patil, and Valeriy Vyatkin. A formal model of IEC 61499-based industrial automation architecture supporting time-aware computations. *IEEE Open Journal of the Industrial Electronics Society*, 2:169–183, 2021.

[9] Dmitrii Drozdov, Victor Dubinin, Sandeep Patil, and Valeriy Vyatkin. A formal model of IEC 61499-based industrial automation architecture supporting time-aware computations. *IEEE Open Journal of the Industrial Electronics Society*, 2:169–183, 2021.

[10] Dmitrii Drozdov, Sandeep Patil, Victor Dubinin, and Valeriy Vyatkin. Towards formal verification for cyber-physically agnostic software: A case study. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 5509–5514, 2017.

[11] E Allen Emerson and Joseph Y Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of computer and system sciences*, 30(1):1–24, 1985.

[12] Yuri Gurevich and E Börger. Evolving algebras 1993: Lipari guide. *Evolving Algebras*, 40, 1995.

[13] Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2017.

[14] Sandeep Patil, Dmitrii Drozdov, and Valeriy Vyatkin. Adapting software design patterns to develop reusable iec 61499 function block applications. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 725–732. IEEE, 2018.

[15] Roopak Sinha, Sandeep Patil, Luis Gomes, and Valeriy Vyatkin. A survey of static formal methods for building dependable industrial automation systems. *IEEE Transactions on Industrial Informatics*, 15(7):3772–3783, 2019.

[16] Valeriy Vyatkin and H-M Hanisch. A modeling approach for verification of IEC 1499 function blocks using net condition/event systems. In *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA'99 (Cat. No. 99TH8467)*, volume 1, pages 261–270. IEEE, 1999.

[17] Valeriy Vyatkin and Hans-Michael Hanisch. Verification of distributed control systems in intelligent manufacturing. *Journal of Intelligent Manufacturing*, 14(1):123–136, 2003.

[18] Valeriy Vyatkin, Hans-Michael Hanisch, Cheng Pang, and Chia-Han Yang. Closed-loop modeling in future automation system engineering and validation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):17–28, 2008.

Paper C

Yet Another Sub-Optimal Estimator of Sinusoids in Noise

Authors:
Dr. C

Reformatted version of paper submitted to:
Example Thesis, Internal Report, Luleå University of Technology, 2009.

Yet Another Sub-Optimal Estimator of Sinusoids in Noise

Dr. C

Abstract

Abstract text of the paper...

1 Introduction

The text of this article is imported from the file *paper3.tex*. The title and abstract part above are typeset manually (see file for code template).

Be sure NOT to have any \begin{document} or \end{document} tags in the imported files.

Some references here too, just to show the use of bibunits .

Paper D

An example of a yet-to-be-submitted paper

Authors:
Dr. C

Manuscript to be submitted.

papers/paper3.tex