Appendix of the paper
"A Systematic Literature Review on Counterexample Explanation"

Arut Prakash Kaleeswaran[a,b], Arne Nordmann[a],
Thomas Vogel[b], Lars Grunske[b]

[a]*Bosch Corporate Sector Research,* Renningen, Germany
{*ArutPrakash.Kaleeswaran*|*Arne.Nordmann*}*@de.bosch.com*
[b]*Humboldt-Universität zu Berlin,* Berlin, Germany
{*thomas.vogel*|*lars.grunske*}*@informatik.hu-berlin.de*

1. Overview

This appendix contains a bibliography of all primary studies that we have analyzed in our systematic literature review, provides an overview of the data items we extracted from these studies (Section 2) and tabulates the primary studies for each data item we extracted (Section 3). For the latter, the data items are grouped by research questions.

List of Tables

2. Data Items

The following table lists and describes the data items that we extracted from the primary studies, and further maps them to the research questions.

Table 1: Extracted data items.

ID	Item	RQs	Description
F1	Different types of counterexample representations	RQ1	The way counterexample is represented for improving the interpretation. Different types of counterexample explanations are graphical, textual, tabular, and trace representation.
F2	Statements on representing a counterexample	RQ1	Qualitative statements that describe the uniqueness or advantages of the representation.
F3	Different types of processed counterexamples	RQ2	The way counterexample is processed and generates either a modified trace of counterexample or an additional trace in addition to the counterexample for improving the interpretation. Different types of procssed counterexamples are minimized counterexample, witness and counterexample, and multiple counterexample.
F4	Statements on processed counterexample	RQ2	Qualitative statements that describe the uniqueness or advantages of the processed counterexample.
F5	Categorizing minimized counterexample studies based on specifications	RQ2	The minimized counterexample studies are categorized based on the kind of specification and verification model. Categories are qualitative, real-time, and probabilistic.
F6	Methods used to minimize a counterexample	RQ2	The methods used to minimize a counterexample are collected and clustered. Methods found are search, translation and abstraction, and comparison with correct system behavior.
F7	Additional information to enrich the counterexample explanation	RQ3	In addition to the counterexample explanation, more information is provided to improve the interpretation.
F8	Statements on processed counterexample	RQ3	Qualitative statements that describe the uniqueness or advantages of the additional information.
F9	Input domain (system and requirement specification)	RQ4	Collects the different design models used as input domain. If no design model is found, we consider the verification model as the input domain.

Table 1 continued from previous page

ID	Item	RQs	Explanation
F10	Output domain (counterexample explanation)	RQ4	Collects the different output domains used to explain the counterexample. If the final output is a trace, then we didn't consider for this research question.
F11	Relations between input and output domains	RQ4	We categorize the relation of counterexample explanation to the input domain as whether the counterexample represented is either the same or different from the user-provided input domain. Further, we identify whether the counterexample is represented as simulation in the given input or reference/mapping to the given input domain.
F12	Temporal logic	RQ5	Collects the different temporal logics used for counterexample explanation.
F13	Property types	RQ5	Collects the different specification properties used for counterexample explanation.
F14	Verification tools	RQ6	Collects the different verification tools used for counterexample explanation.
F15	Frameworks	RQ6	Collects the different frameworks used for counterexample explanation.
F16	Statements on frameworks or model checkers	RQ6	Qualitative statements that describe the uniqueness or advantages of the framework or model checker.
F17	Application domain	RQ7	Collects the different application domains used for counterexample explanation.
F18	Application types	RQ7	Collects the different applications (e.g, industrial, realworld) used for counterexample explanation.
F19	Evaluation aspects	RQ7	Collects the different evaluation aspects used for counterexample explanation.
F20	Evaluation methods	RQ7	Collects the different evaluation methods used for counterexample explanation.

3. List of primary studies

This section complements Section 5 of the systematic literature review that describes the results of the review. Particularly, it tabulates the primary studies for the categories we used for the quantitative data items. The data items are grouped by the seven research questions of our study.

3.1. RQ1: How are counterexamples explained and what are the effects of this explanation on counterexample interpretation?

Table 2: Different types of counterexample representations (F1).

Types of counterexample representations	**Primary studies**	**Count**
Graphical Representation	[1–54]	54
Trace	[55–89]	35
Textual Representation	[90–106]	17
Graphical Representation and Tabular View	[107–111]	5
Tabular View	[112–116]	5
Total count	**116**	

3.2. RQ2: How are counterexamples processed and what effect does it have on interpreting the counterexample?

Table 3: Different types of processed counterexample (F3).

Processed counterexample	**Primary studies**	**Count**
Minimized counterexample	[2, 7, 28–30, 39, 43, 45, 51, 54–56, 58, 59, 61–66, 68–70, 72–76, 79–81, 83–88, 95]	38
Witness and Counterexample	[11, 21, 40, 57, 67, 71, 77, 89, 105]	9
Multiple Counterexample	[38, 46, 53, 60, 78, 102, 104]	7
N/A	[1, 3–6, 8–10, 12–20, 22–27, 31–37, 41, 42, 44, 47–50, 52, 82, 90–94, 96–101, 103, 106–116]	62
Total count	**116**	

Table 4: Categorizing minimized counterexample studies based on specifications (F5).

Categories	**Primary studies**	**Count**
Qualitative	[2, 28–30, 45, 55, 56, 58, 59, 62–66, 68, 69, 73, 74, 79, 80, 83–88, 95]	27
Probabilistic	[39, 43, 51, 54, 61, 70, 72, 75, 76, 81]	10
Real-time	[7]	1
N/A	[1, 3–6, 8–27, 31–38, 40–42, 44, 46–50, 52, 53, 57, 60, 67, 71, 77, 78, 82, 89–94, 96–116]	78
Total count	**116**	

Table 5: Methods used to minimize a counterexample (F6).

Methods	**Primary studies**	**Count**
Search	[2, 39, 51, 55, 56, 61–66, 69, 74, 87]	14
Translation and abstraction	[7, 70, 72, 75, 95]	5
Comparison with correct system behavior	[29, 30, 86, 88]	4
A specific or adapt an existing algorithm	[28, 43, 45, 54, 58, 59, 68, 73, 76, 79–81, 83–85]	15
N/A	[1, 3–6, 8–27, 31–38, 40–42, 44, 46–50, 52, 53, 57, 60, 67, 71, 77, 78, 82, 89–94, 96–116]	78
Total count	**116**	

3.3. RQ3: What kind of additional information is used to enrich the counterexample explanation?

Table 6: Additional information to enrich the counterexample explanation (F7).

Categories	**Primary studies**	**Count**
Additional information along with graphical representation	[1, 14, 52]	3
Additional information along with textual representation	[90–92, 96, 112]	5
N/A	[2–13, 15–51, 53–89, 93–95, 97–111, 113–116]	108
Total count	**116**	

3.4. RQ4: For which input domains are the counterexample explanation approaches available? What is the influence of the input domain on the explanation?

Table 7: Input domain (F9).

Input domain	**Primary studies**	**Count**
System model (State machine/Kripke/MDP/DTMC/LTS)	[5, 6, 10–12, 19, 24, 29–32, 34, 38–40, 42, 43, 45–52, 54, 93, 97, 100, 104, 108, 110, 113, 115, 116]	35
Programming Language	[2, 21, 25, 26, 28, 41, 44, 99, 101–103, 105, 106, 109, 114]	15
Function Block Diagram	[1, 8, 13, 16, 17, 36, 37]	7
Structured English Language	[91, 92, 96, 98]	4
Component diagram	[3, 7, 18, 111]	4
Graph	[22]	1
Others	[4, 9, 14, 15, 20, 23, 27, 33, 35, 53, 90, 94, 95, 107, 112]	15
N/A (Trace representation)	[55–89]	35
Total count	**116**	

Table 8: Output domain (F10).

Output domain	**Primary studies**	**Count**
Programming Language	[101–103, 105, 106]	5
Function Block Diagram	[1, 13, 16, 17, 36, 37]	6
Structured English Language	[91, 92, 96–98]	5
Component Diagram	[7, 18]	2
Graph	[21, 22, 29–32, 39, 41–46]	13
Fault tree	[34, 47–51, 54]	7
Others	[2–6, 8–12, 15, 19, 20, 24–28, 33, 35, 38, 40, 52, 53, 90, 93, 99, 100, 104, 107–116]	43
N/A (Trace representation)	[14, 23, 55–89, 94, 95]	35
Total count	**116**	

Table 9: Relations between input and output domains (F11).

Category	**Sub-category**	**Primary studies**	**Count**
Independent	Simulation	[12, 15, 25]	3
	Reference/Mapping	-	-
	Traceability	[3, 6, 8, 22, 28, 38, 39, 45–48, 50, 51, 54, 90–92, 95–98, 100, 112]	23
	No Reference/Mapping	[2, 5, 10, 11, 19–21, 24, 26, 29–35, 40–44, 49, 52, 53, 93, 99, 104, 108–110, 113–116]	34
Same	Simulation	[1, 13, 16–18, 36, 37, 107, 111]	9
	Reference/Mapping	[4, 7, 9, 14, 23, 27, 94, 101–103, 105, 106]	12
	Traceability	-	-
	No Reference/Mapping	-	-
N/A (Trace representation)		[14, 23, 55–89, 94, 95]	35
Total count			**116**

3.5. *RQ5: What are the different temporal logics used to express system specifications and what type of properties are covered in counterexample explanation approaches?*

Table 10: Temporal logic (F12).

Temporal logic	**Primary studies**	**Count**
LTL	[1, 2, 6, 17–19, 24–27, 36, 37, 48, 52, 53, 55, 56, 58, 62, 63, 65, 67, 71, 74, 78, 83, 88, 100, 105–107]	31
CTL	[8, 11, 20, 38, 40, 85, 87, 109, 110, 115]	10
PCTL	[39, 75, 76, 81, 93]	5
CSL	[47, 49, 50, 54]	4
LTL, CTL	[13, 113, 114]	3
μ-calculus	[22, 41]	2
ACTL	[60, 80]	2
CL	[82, 95]	2
PSL	[10, 84]	2
Others	[3, 7, 9, 42, 45, 46, 57, 61, 91, 98]	10
Unknown/not specific	[4, 5, 12, 14–16, 21, 23, 28–35, 43, 44, 51, 59, 64, 66, 68–70, 72, 73, 77, 79, 86, 89, 90, 92, 94, 96, 97, 99, 101–104, 108, 111, 112, 116]	45
Total count	**116**	

Table 11: Property types (F13).

Property types	**Primary studies**	**Count**
Safety, Liveness	[1, 2, 4, 7, 8, 19, 20, 30, 37, 41, 44, 48, 52, 53, 57, 58, 62, 74, 89]	19
Safety	[17, 21, 28, 29, 38, 61, 63, 67, 68, 73, 77, 102, 104, 105]	14
Liveness	[10, 26, 83, 86]	4
Unknown/not specific	[3, 5, 6, 9, 11–16, 18, 22–25, 27, 31–36, 39, 40, 42, 43, 45–47, 49–51, 54–56, 59, 60, 64–66, 69–72, 75, 76, 78–82, 84, 85, 87, 88, 90–101, 103, 106–116]	79
Total count	**116**	

3.6. RQ6: Which verification tools and frameworks are developed and used to explain counterexamples, and how do they effect the counterexample explanation?

Table 12: Verification tools (F14).

Verification tool	**Primary studies**	**Count**
NuSMV/SMV/nuXmv	[1, 3, 13, 15–18, 27, 36, 37, 40, 45, 46, 58, 60, 67, 73, 78, 80, 84, 88, 107, 108, 110–116]	31
PRISM	[43, 47, 49–51, 54, 61, 70, 72, 75, 76, 81, 93, 97]	15
SPIN	[6, 15, 19, 49, 51, 55, 56, 62, 64, 74, 104]	12
Maude	[2, 12, 24–26]	5
ACL2	[90–92, 96]	4
VIS	[11, 20, 59, 109]	4
Others	[10, 14, 21, 22, 31–34, 38, 42, 48, 54, 57, 63, 72, 76, 79, 85, 89, 95, 99–103, 105, 106, 114, 116]	24
Unknown/not specific	[4, 23, 28–30, 39, 41, 44, 52, 53, 65, 66, 69, 71, 77, 82, 83, 86, 87, 94, 98]	21
Total count	**116**	

Table 13: Frameworks (F15).

Framework	Reference	URL	Count
DiPro	[43, 54, 81, 93]	`http://www.uni-konstanz.de/soft/dipro/download.php`	4
MODCHK	[1, 13, 17, 37]	`https://github.com/igor-buzhinsky/nusmv_counterexample_visualizer`	4
ASSERT	[90–92, 96]	-	4
AutoFocus3	[3, 18, 111]	`https://www.fortiss.org/veroeffentlichungen/software/autofocus-3`	3
SpinCause	[48, 49, 51]	`http://www.uni-konstanz.de/soft//tools/spincause/`	3
KEGVis	[38, 40, 42]	`http://www.drawsvg.org/`	3
CLEAR	[29, 30]	`https://github.com/gbarbon/clear/`	2
A2G2V	[31, 32]	-	2
IFADIS	[108, 115]	-	2
STANCE	[14, 23]	-	2
FaultCAT, CX2FT	[47, 50]	-	2
SMART	[85, 87]	-	2
FRET	[33]	`https://github.com/NASA-SW-VnV/fret`	1
RailComplete	[98]	`https://www.railcomplete.com/en/downloads/`	1
IBM RoseRT	[10]	`https://www.ibm.com/docs/en/rtr`	1
Ivy	[5]	`https://www.cs.tau.ac.il/~odedp/ivy/`	1
COMICS	[39]	`https://www-i2.informatik.rwth-aachen.de/i2/comics/`	1
DSValidator	[99]	`https://ssvlab.github.io/dsverifier/dsvalidator/index.html`	1
FASTEN	[107]	`https://sites.google.com/site/fastenroot/`	1
PLCverif	[114]	`https://readthedocs.web.cern.ch/display/ICKB/PLCverif/`	1
PyNuSMV	[46]	`https://pypi.org/project/pynusmv/`	1
VIS	[109]	`https://ptolemy.berkeley.edu/projects/embedded/research/vis/`	1

AMASE	[4]	https://github.com/afrl-rq/OpenAMASE/wiki/About-AMASE	1
Arcade.PLC	[28]	https://arcade.embedded.rwth-aachen.de/doku.php?id=arcade.plc	1
[Mc]SQUARE	[77]	https://arcade.embedded.rwth-aachen.de/	1
RuleBase PE	[52]	http://www.research.ibm.com/haifa/Workshops/rulebase2010/index.shtml	1
MechatronicUML	[7]	http://www.mechatronicuml.org/en/index.html	1
ELARVA	[82]	http://www.cs.um.edu.mt/svrg/Tools/ELARVAplus/	1
NuSeen	[113]	http://nuseen.sourceforge.net/	1
SpinRCP	[19]	http://lms.uni-mb.si/spinrcp/	1
FLAVERS/Ada	[69]	http://laserweb.cs.umass.edu/verification-examples/chiron/original/2a2e/source/ada_flavers/index.html	1
GraphML	[21]	http://graphml.graphdrawing.org/	1
OERITTE	[36]	https://github.com/ShakeAnApple/cxbacktracker/	1
AnaCon	[95]	-	1
Pseudomerge	[22]	-	1
EOFM	[100]	-	1
ProofProd	[71]	-	1
Evidence Explorer	[41]	-	1
Alfi	[78]	-	1
Theseus	[15]	-	1
MACEMC	[44]	-	1
QuantUM	[34]		1
ATL	[35]		1

Unknown/not specific	[2, 6, 8, 9, 11, 12, 16, 20, 24–27, 45, 53, 55–68, 70, 72–76, 79, 80, 83, 84, 86, 88, 89, 94, 97, 101–106, 110, 112, 116]	-	52
Total			116

3.7. RQ7: How are counterexample explanation approaches evaluated?

Table 14: Application domain (F17).

Application domain	**Primary studies**	**Count**
Protocol	[2, 5, 25, 26, 32, 39, 44, 62, 70, 72, 74–76, 85]	14
Hardware	[11, 15, 28, 36, 43, 50, 53, 73, 79, 81, 93, 104, 110, 114]	14
Automotive	[20, 23, 34, 35, 42, 47, 51, 54, 78, 88]	10
Robotics	[4, 8, 18, 45, 97, 99]	6
Avionics	[89–92, 96]	5
Nuclear	[1, 13, 17, 37]	4
Railway	[94, 98, 112]	3
Others	[6, 7, 9, 12, 14, 27, 41, 48, 49, 63, 95, 105, 111]	13
Unknown/not specific	[10, 21, 22, 29–31, 52, 55, 57–59, 61, 64, 65, 67–69, 77, 80, 82, 84, 86, 87, 101, 102, 106–108]	28
N/A	[3, 16, 19, 24, 33, 38, 40, 46, 56, 60, 66, 71, 83, 100, 103, 109, 113, 115, 116]	19
Total count	**116**	

Table 15: Application types (F18).

Application type	**Primary studies**	**Count**
Reference to non-industrial application	[5, 22, 25, 26, 29, 30, 32, 39, 41, 43, 52, 57, 58, 62, 70, 72, 75, 76, 79, 81, 85–87, 90, 93, 99]	26
Reference to industrial application	[2, 8, 10–12, 21, 31, 36, 42, 44, 55, 73, 74, 77, 82, 101, 102]	17
Example application	[2, 8, 10–12, 21, 31, 36, 38, 42, 44, 55, 73, 74, 77, 82, 100–102]	17
Industrial application	[1, 6, 13–15, 17, 37, 45, 53, 91, 92, 96, 98, 111, 112]	15
Non-Industrial application	[4, 9, 18, 94, 95, 97]	6
Reference to industrial and non-industrial applications	[48, 49, 61, 78, 104]	5
Industrial and non-industrial applications	[50]	1
Unknown/not specific	[59, 64, 65, 67–69, 80, 84, 107, 108]	10
N/A	[3, 16, 19, 24, 33, 38, 40, 46, 56, 60, 66, 71, 83, 100, 103, 109, 113, 115, 116]	19
Total count	**116**	

Table 16: Evaluation aspects (F19).

Evaluation aspect	**Primary studies**	**Count**
Efficiency, Performance	[10, 11, 20, 28, 35, 39, 41, 42, 44, 45, 47–51, 55, 58, 59, 62, 67–69, 72–77, 80, 84, 85, 94, 99, 105]	34
Effectiveness	[1, 2, 4–9, 12–15, 17, 18, 22, 23, 25–27, 31, 32, 36, 54, 63, 81, 82, 86, 90–93, 95, 96, 101, 106–108, 110–112, 114]	41
Efficiency, Performance, and Scalability	[34, 52, 57, 61, 64, 65, 70, 78, 79, 89, 97, 102]	12
Efficiency, Performance, and Efficiency	[21, 29, 30, 37, 43, 53, 88, 98, 104]	9
Scalability and Effectiveness	[87]	1
N/A	[3, 16, 19, 24, 33, 38, 40, 46, 56, 60, 66, 71, 83, 100, 103, 109, 113, 115, 116]	19
Total count	**116**	

Table 17: Evaluation methods (F20).

Evaluation method	**Primary studies**	**Count**
Use-Case(s)	[1, 2, 4–10, 12–15, 17, 18, 20–23, 25–28, 31, 32, 34–37, 41, 43–45, 47–50, 52–54, 57, 58, 61, 63, 72, 75, 77–79, 81, 82, 86, 88–96, 98, 99, 101, 102, 104–106, 111, 114]	70
Benchmark and Use-Case(s)	[39, 42, 51, 62, 70, 73, 74, 76, 85, 87, 97]	11
Benchmark	[11, 55, 59, 64, 65, 67–69, 80, 84]	10
User-Study and Use-Case(s)	[29, 30, 110, 112]	4
User-Study	[107, 108]	2
N/A	[3, 16, 19, 24, 33, 38, 40, 46, 56, 60, 66, 71, 83, 100, 103, 109, 113, 115, 116]	19
Total count	**116**	

References

[1] A. Pakonen, I. Buzhinsky, V. Vyatkin, Counterexample visualization and explanation for function block diagrams, in: INDIN, 2018, pp. 747–753.

[2] T. T. T. Nguyen, K. Ogata, A way to comprehend counterexamples generated by the maude LTL model checker, in: SATE, 2017, pp. 53–62.

[3] A. Campetelli, F. Hölzl, P. Neubeck, User-friendly model checking integration in model-based development, in: 24th International Conference on Computer Applications in Industry and Engineering, 2011.

[4] L. Feng, L. R. Humphrey, I. Lee, U. Topcu, Human-interpretable diagnostic information for robotic planning systems, in: IROS, 2016, pp. 1673–1680.

[5] O. Padon, K. L. McMillan, A. Panda, M. Sagiv, S. Shoham, Ivy: safety verification by interactive generalization, in: PLDI, 2016, pp. 614–630.

[6] Z. Zheng, J. Tian, T. Zhao, Refining operation guidelines with model-checking-aided FRAM to improve manufacturing processes: a case study for aeroengine blade forging, Cognition, Technology & Work 18 (4) (2016) 777–791.

[7] C. Gerking, W. Schäfer, S. Dziwok, C. Heinzemann, Domain-specific model checking for cyber-physical systems, in: MoDeVVa@MoDELS, 2015, pp. 18–27.

[8] S. Patil, V. Vyatkin, C. Pang, Counterexample-guided simulation framework for formal verification of flexible automation systems, in: INDIN, 2015, pp. 1192–1197.

[9] V. Ciancia, S. Gilmore, G. Grilletti, D. Latella, M. Loreti, M. Massink, Spatio-temporal model checking of vehicular movement in public transport systems, STTT 20 (3) (2018) 289–311.

[10] J. Elamkulam, Z. Glazberg, I. Rabinovitz, G. Kowlali, S. C. Gupta, S. Kohli, S. Dattathrani, C. P. Macia, Detecting design flaws in UML state charts for embedded software, in: HVC, 2006, pp. 109–121.

[11] H. Jin, K. Ravi, F. Somenzi, Fate and free will in error traces, STTT 6 (2) (2004) 102–116.

[12] T. T. T. Nguyen, K. Ogata, Graphical animations of state machines, in: DASC/PiCom/DataCom/CyberSciTech, 2017, pp. 604–611.

[13] A. Pakonen, T. Tahvonen, M. Hartikainen, M. Pihlanko, Practical applications of model checking in the finnish nuclear industry, in: NPIC & HMIT, American Nuclear Society ANS, 2017, pp. 1342–1352.

[14] T. Bochot, P. Virelizier, H. Waeselynck, V. Wiels, Paths to property violation: A structural approach for analyzing counter-examples, in: HASE, 2010, pp. 74–83.

[15] H. Goldsby, B. H. C. Cheng, S. Konrad, S. Kamdoum, A visualization framework for the modeling and formal analysis of high assurance systems, in: MoDELS, 2006, pp. 707–721.

[16] A. Pakonen, T. Matasniemi, J. Lahtinen, T. Karhela, A toolset for model checking of PLC software, in: ETFA, 2013, pp. 1–6.

[17] A. Pakonen, K. Björkman, Model checking as a protective method against spurious actuation of industrial control systems, in: ESREL, CRC Press, 2017, pp. 3189–3196.

[18] S. Kanav, V. Aravantinos, Modular transformation from AF3 to nuxmv, in: MODELS, 2017, pp. 300–306.

[19] Z. Brezocnik, B. Vlaovic, A. Vreze, SpinRCP: the eclipse rich client platform integrated development environment for the spin model checker, in: SPIN, 2014, pp. 125–128.

[20] I. Schinz, T. Toben, C. Mrugalla, B. Westphal, The rhapsody UML verification environment, in: SEFM, 2004, pp. 174–183.

[21] D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, A. Stahlbauer, Witness validation and stepwise testification across software verifiers, 2015, pp. 721–733.

[22] M. Sassolas, M. Chechik, S. Uchitel, Exploring inconsistencies between modal transition systems, Software and System Modeling 10 (1) (2011) 117–142.

[23] K. C. Castillos, H. Waeselynck, V. Wiels, Show me new counterexamples: A path-based approach, in: ICST, 2015, pp. 1–10.

[24] M. T. Aung, T. T. T. Nguyen, K. Ogata, Analysis of two flawed versions of A mutual exclusion protocol with maude and SMGA, in: ICSCA, 2018, pp. 194–198.

[25] T. T. T. Nguyen, K. Ogata, Graphically perceiving characteristics of the MCS lock and model checking them, in: SOFL+MSVL, 2017, pp. 3–23.

[26] Y. Phyo, K. Ogata, Analysis of some variants of the anderson array-based queuing mutual exclusion protocol with model checking and graphical animations, in: DSA, IEEE, 2018, pp. 126–135.

[27] F. U. Muram, H. Tran, U. Zdun, Counterexample analysis for supporting containment checking of business process models, in: BPM, 2015, pp. 515–528.

[28] S. Biallas, N. Friedrich, H. Simon, S. Kowalewski, Automatic error cause localization of faulty plc programs, IFAC-PapersOnLine 48 (7) (2015) 79–84.

[29] G. Barbon, V. Leroy, G. Salaun, Debugging of behavioural models using counterexample analysis, IEEE Transactions on Software Engineering (2019) 1–14.

[30] G. Barbon, V. Leroy, G. Salaün, Debugging of behavioural models with CLEAR, in: TACAS, 2019, pp. 386–392.

[31] A. T. Al Ghazo, M. Ibrahim, H. Ren, R. Kumar, A2g2v: Automated attack graph generator and visualizer, in: Proceedings of the 1st ACM MobiHoc Workshop on Mobile IoT Sensing, Security, and Privacy, Mobile IoT SSP'18, Association for Computing Machinery, New York, NY, USA, 2018.

[32] A. T. Al Ghazo, M. Ibrahim, H. Ren, R. Kumar, A2g2v: Automatic attack graph generation and visualization and its applications to computer and scada networks, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019).

[33] A. Mavridou, H. Bourbouh, P. Garoche, D. Giannakopoulou, T. Pressburger, J. Schumann, Bridging the gap between requirements and simulink model analysis, in: REFSQ, 2020.

[34] M. Kölbl, S. Leue, An efficient algorithm for computing causal trace sets in causality checking, in: ATVA, 2019, pp. 171–186.

[35] M. Kölbl, S. Leue, H. Singh, From sysml to model checkers via model transformation, in: SPIN, 2018, pp. 255–274.

[36] P. Ovsiannikova, I. Buzhinsky, A. Pakonen, V. Vyatkin, Visual counterexample explanation for model checking with oeritte, CoRR abs/2012.15097 (2020). `arXiv:2012.15097`.

[37] A. Pakonen, I. Buzhinsky, K. Björkman, Model checking reveals design issues leading to spurious actuation of nuclear instrumentation and control systems, Reliab. Eng. Syst. Saf. 205 (2021) 107237.

[38] M. Chechik, A. Gurfinkel, A framework for counterexample generation and exploration, STTT 9 (5-6) (2007) 429–445.

[39] N. Jansen, E. Ábrahám, M. Volk, R. Wimmer, J. Katoen, B. Becker, The COMICS tool - computing minimal counterexamples for dtmcs, in: ATVA, Vol. 7561 of LNCS, Springer, 2012, pp. 349–353.

[40] A. Gurfinkel, M. Chechik, Multi-valued model checking via classical model checking, in: CONCUR, 2003, pp. 263–277.

[41] Y. Dong, C. R. Ramakrishnan, S. A. Smolka, Model checking and evidence exploration, in: ECBS, 2003, pp. 214–223.

[42] A. Gurfinkel, M. Chechik, B. Devereux, Temporal logic query checking: A tool for model exploration, IEEE Trans. Software Eng. 29 (10) (2003) 898–914.

[43] H. Aljazzar, S. Leue, Debugging of dependability models using interactive visualization of counterexamples, in: QEST, 2008, pp. 189–198.

[44] C. E. Killian, J. W. Anderson, R. Jhala, A. Vahdat, Life, death, and the critical transition: Finding liveness bugs in systems code (awarded best paper), in: NSDI, 2007.

[45] F. Weitl, S. Nakajima, Incremental construction of counterexamples in model checking web documents, in: WWV, 2010, pp. 34–50.

[46] S. Busard, C. Pecheur, Producing explanations for rich logics, in: FM, 2018, pp. 129–146.

[47] M. Kuntz, F. Leitner-Fischer, S. Leue, From probabilistic counterexamples via causality to fault trees, in: SAFECOMP, 2011, pp. 71–84.

[48] F. Leitner-Fischer, S. Leue, Causality checking for complex system models, in: VMCAI, 2013, pp. 248–267.

[49] F. Leitner-Fischer, S. Leue, On the synergy of probabilistic causality computation and causality checking, in: SPIN, 2013, pp. 246–263.

[50] F. Leitner-Fischer, S. Leue, Probabilistic fault tree synthesis using causality computation, IJCCBS 4 (2) (2013) 119–143.

[51] F. Leitner-Fischer, S. Leue, Spincause: a tool for causality checking, in: SPIN, 2014, pp. 117–120.

[52] I. Beer, S. Ben-David, H. Chockler, A. Orni, R. J. Trefler, Explaining counterexamples using causality, Formal Methods in System Design 40 (1) (2012) 20–40.

[53] F. Copty, A. Irron, O. Weissberg, N. P. Kropp, G. Kamhi, Efficient debugging in a formal verification environment, STTT 4 (3) (2003) 335–348.

[54] H. Aljazzar, F. Leitner-Fischer, S. Leue, D. Simeonov, Dipro - A tool for probabilistic counterexample generation, in: SPIN, 2011, pp. 183–187.

[55] P. Gastin, P. Moro, M. Zeitoun, Minimization of counterexamples in SPIN, in: SPIN, 2004, pp. 92–108.

[56] P. Gastin, P. Moro, Minimal counterexample generation for SPIN, in: SPIN, 2007, pp. 24–38.

[57] T. Kumazawa, T. Tamai, Counterexample-based error localization of behavior models, in: NFM, 2011, pp. 222–236.

[58] V. Schuppan, A. Biere, Shortest counterexamples for symbolic model checking of LTL with past, in: TACAS, 2005, pp. 493–509.

[59] K. Ravi, F. Somenzi, Minimal assignments for bounded model checking, in: TACAS, 2004, pp. 31–45.

[60] E. M. Clarke, S. Jha, Y. Lu, H. Veith, Tree-like counterexamples in model checking, in: LICS, 2002, pp. 19–29.

[61] H. Aljazzar, S. Leue, Directed explicit state-space search in the generation of counterexamples for stochastic model checking, IEEE Trans. Software Eng. 36 (1) (2010) 37–60.

[62] S. Edelkamp, S. Leue, A. Lluch-Lafuente, Directed explicit-state model checking in the validation of communication protocols, STTT 5 (2-3) (2004) 247–267.

[63] A. Groce, W. Visser, What went wrong: Explaining counterexamples, in: SPIN, 2003, pp. 121–135.

[64] H. Hansen, J. Geldenhuys, Cheap and small counterexamples, in: SEFM, 2008, pp. 53–62.

[65] H. Hansen, A. Kervinen, Minimal counterexamples in o(n log n) memory and o(nˆ2) time, in: ACSD, 2006, pp. 133–142.

[66] N. Kumar, V. Kumar, M. Viswanathan, On the complexity of error explanation, in: VMCAI, 2005, pp. 448–464.

[67] S. Shen, Y. Qin, S. Li, Localizing errors in counterexample with iteratively witness searching, in: ATVA, 2004, pp. 456–469.

[68] S. Shen, Y. Qin, S. Li, A fast counterexample minimization approach with refutation analysis and incremental SAT, in: ASP-DAC, 2005, pp. 451–454.

[69] J. Tan, G. S. Avrunin, L. A. Clarke, S. Zilberstein, S. Leue, Heuristic-guided counterexample search in FLAVERS, in: SIGSOFT, 2004, pp. 201–210.

[70] R. Wimmer, N. Jansen, E. Ábrahám, B. Becker, J. Katoen, Minimal critical subsystems for discrete-time markov models, in: TACAS, 2012, pp. 299–314.

[71] D. A. Peled, A. Pnueli, L. D. Zuck, From falsification to verification, in: FST TCS, 2001, pp. 292–304.

[72] R. Wimmer, N. Jansen, E. Ábrahám, J. Katoen, B. Becker, Minimal counterexamples for linear-time probabilistic verification, Theor. Comput. Sci. 549 (2014) 61–100.

[73] S. Shen, Y. Qin, S. Li, A faster counterexample minimization algorithm based on refutation analysis, in: DATE, 2005, pp. 672–677.

[74] S. Edelkamp, A. Lluch-Lafuente, S. Leue, Directed explicit model checking with HSF-SPIN, in: SPIN, 2001, pp. 57–79.

[75] N. Jansen, E. Ábrahám, J. Katelaan, R. Wimmer, J. Katoen, B. Becker, Hierarchical counterexamples for discrete-time markov chains, in: ATVA, 2011, pp. 443–452.

[76] E. Ábrahám, N. Jansen, R. Wimmer, J. Katoen, B. Becker, DTMC model checking by SCC reduction, in: QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, 2010, pp. 37–46.

[77] J. Brauer, A. Simon, Inferring definite counterexamples through under-approximation, in: NFM, 2012, pp. 54–69.

[78] A. L. J. Dominguez, N. A. Day, Generating multiple diverse counterexamples for an efsm, Technical Report CS-2013–06, University of Waterloo (2013).

[79] K. Chang, V. Bertacco, I. L. Markov, Simulation-based bug trace minimization with bmc-based refinement, IEEE Trans. on CAD of Integrated Circuits and Systems 26 (1) (2007) 152–165.

[80] S. Shen, Y. Qin, S. Li, Counterexample minimization for actl, in: CHARME, Vol. 5, 2005, pp. 393–397.

[81] H. Debbi, M. Bourahla, Generating diagnoses for probabilistic model checking using causality, CIT 21 (1) (2013) 13–23.

[82] C. Colombo, A. Francalanza, I. Grima, Simplifying contract-violating traces, in: FLACOS, 2012, pp. 11–20.

[83] V. Schuppan, A. Biere, Liveness checking as safety checking for infinite state spaces, Electr. Notes Theor. Comput. Sci. 149 (1) (2006) 79–96.

[84] K. Heljanko, T. A. Junttila, M. Keinänen, M. Lange, T. Latvala, Bounded model checking for weak alternating büchi automata, in: CAV, 2006, pp. 95–108.

[85] Y. Zhao, X. Jin, G. Ciardo, A symbolic algorithm for shortest EG witness generation, in: TASE, 2011, pp. 68–75.

[86] G. Barbon, V. Leroy, G. Salaün, Counterexample simplification for liveness property violation, in: SEFM, 2018, pp. 173–188.

[87] C. Jiang, G. Ciardo, Improving sat-based bounded model checking for existential CTL through path reuse, in: LPAR, 2018, pp. 471–487.

[88] A. P. Kaleeswaran, A. Nordmann, T. Vogel, L. Grunske, Counterexample interpretation for contract-based design, in: IMBSA, 2020, pp. 99–114.

[89] W. E. Kholy, M. El-Menshawy, J. Bentahar, M. Elqortobi, A. Laarej, R. Dssouli, Model checking intelligent avionics systems for test cases generation using multi-agent systems, Expert Syst. Appl. 156 (2020) 113458.

[90] A. W. Crapo, A. Moitra, Using OWL ontologies as a domain-specific language for capturing requirements for formal analysis and test case generation, in: ICSC, 2019, pp. 361–366.

[91] A. W. Crapo, A. Moitra, C. McMillan, D. Russell, Requirements capture and analysis in ASSERT(TM), in: RE, 2017, pp. 283–291.

[92] A. Moitra, K. Siu, A. W. Crapo, M. Durling, M. Li, P. Manolios, M. Meiners, C. McMillan, Automating requirements analysis and test case generation, Requir. Eng. 24 (3) (2019) 341–364.

[93] H. Debbi, Diagnosis of probabilistic models using causality and regression, in: VECoS, 2014, pp. 34–44.

[94] B. Luteberget, J. J. Camilleri, C. Johansen, G. Schneider, Participatory verification of railway infrastructure by representing regulations in railcnl, in: SEFM, 2017, pp. 87–103.

[95] K. Angelov, J. J. Camilleri, G. Schneider, A framework for conflict analysis of normative texts written in controlled natural language, J. Log. Algebr. Program. 82 (5-7) (2013) 216–240.

[96] A. Moitra, K. Siu, A. W. Crapo, H. R. Chamarthi, M. Durling, M. Li, H. Yu, P. Manolios, M. Meiners, Towards development of complete and conflict-free requirements, in: RE, 2018, pp. 286–296.

[97] L. Feng, M. Ghasemi, K. Chang, U. Topcu, Counterexamples for robotic planning explained in structured language, CoRR abs/1803.08966 (2018). arXiv:1803.08966.

[98] B. Luteberget, C. Johansen, Efficient verification of railway infrastructure designs against standard regulations, Formal Methods in System Design 52 (1) (2018) 1–32.

[99] L. C. Chaves, I. Bessa, L. C. Cordeiro, D. Kroening, Dsvalidator: An automated counterexample reproducibility tool for digital systems, in: HSCC, 2018, pp. 253–258.

[100] M. L. Bolton, E. J. Bass, Using task analytic models to visualize model checker counterexamples, in: IEEE International Conference on Systems, Man and Cybernetics, 2010, pp. 2069–2074.

[101] A. Groce, D. Kroening, F. Lerda, Understanding counterexamples with explain, in: CAV, 2004, pp. 453–456.

[102] T. Ball, M. Naik, S. K. Rajamani, From symptom to cause: localizing errors in counterexample traces, in: SIGPLAN-SIGACT, 2003, pp. 97–105.

[103] E. M. Clarke, D. Kroening, F. Lerda, A tool for checking ANSI-C programs, in: TACAS, 2004, pp. 168–176.

[104] S. Leue, M. T. Befrouei, Counterexample explanation by anomaly detection, in: SPIN, 2012, pp. 24–42.

[105] A. Groce, S. Chaki, D. Kroening, O. Strichman, Error explanation with distance metrics, STTT 8 (3) (2006) 229–247.

[106] F. Pu, Y. Zhang, Localizing program errors via slicing and reasoning, in: HASE, 2008, pp. 187–196.

[107] D. Ratiu, M. Gario, H. Schoenhaar, FASTEN: an open extensible framework to experiment with formal specification approaches: using language engineering to develop a multi-paradigm specification environment for nusmv, in: FormaliSE@ICSE, IEEE / ACM, 2019, pp. 41–50.

[108] K. Loer, M. D. Harrison, An integrated framework for the analysis of dependable interactive systems (IFADIS): its tool support and evaluation, Autom. Softw. Eng. 13 (4) (2006) 469–496.

[109] S. Jeong, J. Yoo, S. D. Cha, VIS analyzer: A visual assistant for VIS verification and analysis, in: ISORC, 2010, pp. 250–254.

[110] J. C. Campos, M. D. Harrison, Interaction engineering using the IVY tool, in: EICS, 2009, pp. 35–44.

[111] A. Campetelli, M. Junker, B. Böhm, M. Davidich, V. Koutsoumpas, X. Zhu, J. C. Wehrstedt, A model-based approach to formal verification in early development phases: A desalination plant case study, in: Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering, 2015, pp. 91–100.

[112] L. van den Berg, P. A. Strooper, W. Johnston, An automated approach for the interpretation of counter-examples, Electr. Notes Theor. Comput. Sci. 174 (4) (2007) 19–35.

[113] P. Arcaini, A. Gargantini, E. Riccobene, Nuseen: A tool framework for the nusmv model checker, in: ICST, 2017, pp. 476–483.

[114] D. Darvas, E. Blanco Vinuela, B. Fernández Adiego, Plcverif: A tool to verify plc programs based on model checking techniques (2015).

[115] K. Loer, M. D. Harrison, Towards usable and relevant model checking techniques for the analysis of dependable interactive systems, in: ASE, 2002, pp. 223–226.

[116] D. Ratiu, B. Schätz, M. Völter, B. Kolb, Language engineering as an enabler for incrementally defined formal analyses, in: FormSERA, 2012, pp. 9–15.

Acronyms

ACL2 A Computational Logic for Applicative Common Lisp.

AMASE Aerospace Multi-agent Simulation Environment.

ASSERT Analysis of Semantic Specifications and Efficient generation of Requirements-based Tests.

BDD Binary Decision Diagrams.

BFL Brute Force Lifting.

BFS Breath-First Search.

BPMN Business Process Model and Notation.

Butramin BUg TRAce MINimization.

CAD Computer-aided Design.

CBD Contract-Based Design.

CBMC C Bounded Model Checker.

CL Contract Language.

CLAN Contract Language ANalyser.

CNL Controlled/Constrained Natural Language.

COMICS Computing Minimal Counterexamples.

CSL Continuous Stochastic Logic.

CTL Computation Tree Logic.

CTMC Continuous-Time Markov Chain.

DFS Depth-First Search.

DiPro Directed Probabilistic Counterexample Generation Tool.

DSL Domain-Specific Language.

DTMC Discrete-Time Markov Chain.

FASTEN FormAl SpecificaTion ENvironment.

FMEA Failure Mode and Effect Analysis.

FTA Fault Tree Analysis.

GF Grammatical Framework.

GraphML Graph Markup Language.

GUI Graphical User Interface.

HAZOP Hazard and Operability.

KEGVis Kounterexample generator and visualizer.

LTL Linear Temporal Logic.

LTS Labelled Transition System.

MDP Markov Decision Processes.

MILP Mixed Integer Linear Programming.

MPS Meta Programming System.

MRMC Markov Reward Model Checker.

MSC Message Sequence Chart.

NuSMV New Symbolic Model Verifier.

PCTL Probabilistic Computation Tree Logic.

PLC Programmable Logic Controller.

PRISM Probabilistic Symbolic Model Checker.

PROMELA Process or Protocol Meta Language.

PSL Property Specification Language.

RAE Requirements Analysis Engine.

RTCTL Real Time Computation Tree Logic.

SAT Satisfiability.

SMT Satisfiability Modulo Theories.

SMV Symbolic Model Verifier.

SPIN Simple PROMELA Interpreter.

STANCE Structural Analysis of Counter-Examples.

SysML Systems Modeling Language.

TCTL Timed Computational Tree Logic.

UAV Unmanned Aerial Vehicle.

UML Unified Modeling Language.

VIS Verification Interacting with Synthesis.

XBF eXtended Best-First.

XChek Multi-valued Model-Checker.