List of Publications

The publications that are found for each research question are listed based on their categories described in Section 5 of SLR.

Types of Counterexample Representation	**Publications**
Graphical Representation	[1–55]
Trace	[56–91]
Textual Representation	[92–107]
Graphical Representation and Tabular View	[108–112]
Tabular View	[113–117]
Total Count	**116**

Table 1: Counterexample representations.

Category	**Publications**
Minimized counterexample	[2, 7, 29–31, 40, 44, 46, 52, 55–57, 59, 60, 62–67, 69–71, 73–77, 81–83, 85–90, 96]
Witness and Counterexample	[12, 22, 41, 58, 68, 72, 79, 91, 106]
Multiple Counterexample	[39, 47, 54, 61, 80, 103, 105]
Total Count	**54**

Table 2: Counterexample processing

Categories	**Publications**
Additional information along with graphical representation	[1, 15, 53]
Additional information along with textual representation	[92–94, 97, 113]
Total Count	**8**

Table 3: Enriching counterexamples with additional information.

Input domain	**Publications**
System model (State machine/Kripke/MDP/DTMC/LTS)	[5, 6, 10–13, 20, 25, 30–33, 35, 39–41, 43, 44, 46–53, 55, 98, 101, 105, 109, 111, 114, 116, 117]
Programming Language	[2, 22, 26, 27, 29, 42, 45, 100, 102–104, 106, 107, 110, 115]
Function Block Diagram	[1, 8, 14, 17, 18, 37, 38]
Component diagram	[3, 7, 19, 112]
Structured English Language	[93, 94, 97, 99]
SCADE/Simulink Model	[15, 24, 34]
CNL	[95, 96]
Others	[4, 9, 16, 21, 23, 28, 36, 54, 92, 108, 113]
Total	**81**

Table 4: Publications for Input domain.

Output Domain	**Publications**
Graph	[22, 23, 30–33, 40, 42–47]
State Machine	[2, 5, 6, 13, 25–27, 29]
Fault tree	[35, 48–52, 55]
Tabular view	[113–117]
Programming Language	[102–104, 106, 107]
Structured English Language	[93, 94, 97–99]
Function Block Diagram	[1, 14, 17, 18, 37, 38]
Trace Simulation of signal	[8, 12, 34, 53, 54]
Textual Representation	[10, 100, 101]
Others	[3, 4, 7, 9, 11, 16, 19–21, 28, 36, 39, 41, 92, 105, 108–112]
Total	**81**

Table 5: Publications for Output domain.

Category	Sub-Category	Publications
Independent	Reference/Mapping	-
	Simulation	[13, 16, 26]
	Traceability	[3, 6, 8, 23, 29, 39, 40, 46–49, 51, 52, 55, 92–94, 96–99, 101, 113]
	No Reference/Mapping	[2, 5, 10–12, 20–22, 25, 27, 30–36, 41–45, 50, 53, 54, 100, 105, 109–111, 114–117]
Same	Reference/Mapping	[4, 7, 9, 15, 24, 28, 95, 102–104, 106, 107]
	Simulation	[1, 14, 17–19, 37, 38, 108, 112]
	Traceability	-
	No Reference/Mapping	-
Total		**81**

Table 6: Relations between input domains and output domains.

Specification	Publications
LTL	[1, 2, 6, 18–20, 25–28, 37, 38, 49, 53, 54, 56, 57, 59, 63, 64, 66, 68, 72, 75, 80, 85, 90, 101, 106–108]
CTL	[8, 12, 21, 39, 41, 87, 89, 110, 111, 116]
PCTL	[10, 40, 76, 77, 83]
CSL	[48, 50, 51, 55]
LTL, CTL	[14, 114, 115]
μ-calculus	[23, 42]
ACTL	[61, 82]
CL	[84, 96]
PSL	[11, 86]
Others	[3, 7, 9, 43, 46, 47, 58, 62, 93, 99]
Total	**71**

Table 7: Types of specification.

Property	Publications
Safety, Liveness	[1, 2, 4, 7, 8, 20, 21, 31, 38, 42, 45, 49, 53, 54, 58, 59, 63, 75, 91]
Safety	[18, 22, 29, 30, 39, 62, 64, 68, 69, 74, 79, 103, 105, 106]
Liveness	[11, 27, 85, 88]
Total	**37**

Table 8: Types of property specification.

Verification Tool	**Publication**
NuSMV/SMV/nuXmv	[1, 3, 14, 16–19, 28, 37, 38, 41, 46, 47, 59, 61, 68, 74, 80, 82, 86, 90, 108, 109, 111–117]
PRISM	[10, 44, 48, 50–52, 55, 62, 71, 73, 76, 77, 83, 98]
SPIN	[6, 16, 20, 50, 52, 56, 57, 63, 65, 75, 105]
Maude	[2, 13, 25–27]
ACL2	[92–94, 97]
VIS	[12, 21, 60, 110]
Others	[11, 15, 22, 23, 32–35, 39, 43, 49, 55, 58, 64, 73, 77, 81, 87, 91, 96, 100–104, 106, 107, 115, 117]
Total	**100**

Table 9: Verification tools.

Framework	Publications	URL
DiPro	[10, 44, 55, 83]	http://www.uni-konstanz.de/soft/dipro/download.php
AutoFocus3	[3, 19, 112]	https://www.fortiss.org/veroeffentlichungen/software/autofocus-3
MODCHK	[1, 14, 18, 38]	https://github.com/igor-buzhinsky/nusmv_counterexample_visualizer
SpinCause	[49, 50, 52]	http://www.uni-konstanz.de/soft//tools/spincause/
KEGVis	[39, 41, 43]	http://www.drawsvg.org/
CLEAR	[30, 31]	https://github.com/gbarbon/clear/
FRET	[34]	https://github.com/NASA-SW-VnV/fret
RailComplete	[99]	https://www.railcomplete.com/en/downloads/
IBM RoseRT	[11]	https://www.ibm.com/support/pages/ibm-rational-rose-realtime-7001-ifix001
Ivy	[5]	https://www.cs.tau.ac.il/~odedp/ivy/
COMICS	[40]	https://www-i2.informatik.rwth-aachen.de/i2/comics/
DSValidator	[100]	https://ssvlab.github.io/dsverifier/dsvalidator/index.html
FASTEN	[108]	https://sites.google.com/site/fastenroot/
PLCverif	[115]	https://readthedocs.web.cern.ch/display/ICKB/PLCverif/
PyNuSMV	[47]	https://pypi.org/project/pynusmv/
VIS	[110]	https://ptolemy.berkeley.edu/projects/embedded/research/vis/
AMASE	[4]	https://github.com/afrl-rq/OpenAMASE/wiki/About-AMASE
Arcade.PLC	[29]	https://arcade.embedded.rwth-aachen.de/doku.php?id=arcade.plc
[Mc]SQUARE	[79]	https://arcade.embedded.rwth-aachen.de/
RuleBase PE	[53]	http://www.research.ibm.com/haifa/Workshops/rulebase2010/index.shtml
MechatronicUML	[7]	http://www.mechatronicuml.org/en/index.html
ELARVA	[84]	http://www.cs.um.edu.mt/svrg/Tools/ELARVAplus/
NuSeen	[114]	http://nuseen.sourceforge.net/
SpinRCP	[20]	http://lms.uni-mb.si/spinrcp/

Table 10 continued from previous page

Framework	**Publications**	**URL**
FLAVERS/Ada	[70]	`http://laserweb.cs.umass.edu/verification-examples/chiron/original/2a2e/source/ada_flavers/index.html`
GraphML	[22]	`http://graphml.graphdrawing.org/`
OERITTE	[37]	`https://github.com/ShakeAnApple/cxbacktracker/`
ASSERT	[92–94, 97]	-
A2G2V	[32, 33]	-
IFADIS	[109, 116]	-
STANCE	[15, 24]	-
FaultCAT,CX2FT	[48, 51]	-
AnaCon	[96]	-
Pseudo-merge	[23]	-
EOFM	[101]	-
ProofProd	[72]	-
Evidence Explorer	[42]	-
SMART	[87, 89]	-
Alfi	[80]	-
Theseus	[16]	-
MACEMC	[45]	-
QuantUM	[35]	
ATL	[36]	
Total	**62**	

Table 10: Counterexample explanation frameworks.

ID	Item	RQs	Explanation
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.

Table 15 continued from previous page

ID	Item	RQs	Explanation
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)	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.
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	Counterexample representation relates to the input domain	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	RQ5	Collects the different specification properties used for counterexample explanation.
F14	Frameworks	RQ6	Collects the different frameworks used for counterexample explanation.
F15	Model Checker	RQ6	Collects the different model checkers 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.

Table 15 continued from previous page

ID	Item	RQs	Explanation
F17	Application Domain	RQ7	Collects the different application domains used for counterexample explanation.
F18	Applications	RQ7	Collects the different applications (e.g, industrial, realworld) used for counterexample explanation.
F19	Evaluation method	RQ7	Collects the different evaluation methods used for counterexample explanation.
F20	Evaluation Aspects	RQ7	Collects the different evaluation aspects used for counterexample explanation.

Table 15: Extracted data items.

Application Domain	**Publications**
Protocol	[2, 5, 26, 27, 33, 40, 45, 63, 71, 73, 75–77, 87]
Hardware	[10, 12, 16, 29, 37, 44, 51, 54, 74, 81, 83, 105, 111, 115]
Automotive	[21, 24, 35, 36, 43, 48, 52, 55, 80, 90]
Robotics	[4, 8, 19, 46, 98, 100]
Avionics	[91–94, 97]
Nuclear	[1, 14, 18, 38]
Railway	[95, 99, 113]
Others	[6, 7, 9, 13, 15, 28, 42, 49, 50, 64, 96, 106, 112]
Total	**69**

Table 11: Publications for application domain.

Use-Case	**Publications**
Reference to non-industrial use-case	[5, 10, 23, 26, 27, 30, 31, 33, 40, 42, 44, 49, 50, 53, 58, 59, 62, 63, 71, 73, 76, 77, 80, 81, 83, 87–89, 92, 100, 105]
Reference to industrial use-case	[7, 21, 24, 28, 29, 35, 36, 48–50, 52, 55, 62, 64, 80, 90, 91, 105–107, 111, 115]
Example Use-Case	[2, 8, 11–13, 22, 32, 37, 39, 43, 45, 56, 74, 75, 79, 84, 101–103]
Non-Industrial Use-Case	[4, 9, 19, 51, 95, 96, 98]
Industrial Use-Case	[1, 6, 14–16, 18, 38, 46, 51, 54, 93, 94, 97, 99, 112, 113]
Total	**89**

Table 12: Publications for use-case.

Evaluation Aspects	**Publications**
Efficiency, Performance	[11, 12, 21, 22, 29–31, 35, 36, 38, 40, 42–46, 48–54, 56, 58–60, 62, 63, 65, 66, 68–71, 73–77, 79–82, 86, 87, 90, 91, 95, 98–100, 103, 105, 106]
Effectiveness	[1, 2, 4–10, 13–16, 18, 19, 22–24, 26–28, 30–33, 37, 38, 44, 54, 55, 64, 83, 84, 88–90, 92–94, 96, 97, 99, 102, 105, 107–109, 111–113, 115]
Scalability	[35, 53, 58, 62, 65, 66, 71, 80, 81, 89, 91, 98, 103]
Total	**97**

Table 13: Publications for evaluation aspects.

Evaluation Method	**Publications**
Use-Case(s)	[1, 2, 4–11, 13–16, 18, 19, 21–24, 26–33, 35–38, 40, 42–46, 48–55, 58, 59, 62–64, 71, 73–77, 79–81, 83, 84, 87–100, 102, 103, 105–107, 111–113, 115]
Benchmark	[12, 40, 43, 52, 56, 60, 63, 65, 66, 68–71, 74, 75, 77, 82, 86, 87, 89, 98]
User-Study	[30, 31, 108, 109, 111, 113]
Total	**97**

Table 14: Publications for evaluation methods.

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

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

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

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

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

[25] 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.

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

[27] 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.

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

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

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

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

[32] 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.

[33] 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).

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

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

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

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

[38] 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.

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

[40] 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.

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

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

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

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

[45] 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.

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

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

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

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

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

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

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

[53] 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.

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

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

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

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

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

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

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

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

[62] 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.

[63] 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.

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

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

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

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

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

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

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

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

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

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

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

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

[76] 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.

[77] 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.

[78] A. Cimatti, M. Roveri, V. Schuppan, A. Tchaltsev, Diagnostic information for realizability, in: VMCAI, 2008, pp. 52–67.

[79] J. Brauer, A. Simon, Inferring definite counterexamples through underapproximation, in: NFM, 2012, pp. 54–69.

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

[81] 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.

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

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

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

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

[86] 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.

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

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

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

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

[91] 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.

[92] 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.

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

[94] 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.

[95] 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.

[96] 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.

[97] 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.

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

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

[100] 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.

[101] 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.

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

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

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

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

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

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

[108] 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.

[109] 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.

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

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

[112] 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.

[113] 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.

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

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

[116] 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.

[117] 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.