Table 1: Simulink Fault Patterns and mutation operators used in BERTiMuS

Fault ID	Fault Patterns	Corresponding mutation operators	References
1	Incorrect signal data types in math equations	Changing number type from double or float to int, changing single to double. Changing unit of a numbers, e.g., changing milliseconds unit into seconds unit for signal data. Changing MATLAB 'fixdt(0,8,3)' data type with MATLAB 'fixdt(0,3,8)' data type. Changing arrays to other arrays.	[6]
2	Incorrect 'GoTo' block or 'From' block	Changing the tag value of a 'GoTo' block or the tag value of a 'From' block.	[6]
3	False 'Saturate on integer overflow' in math operations blocks	Changing the property between 'on' and 'off'.	[6]
4	Incorrect constant values	Changing gain/constant block value, such as changing the constant value from 0.5 to 50. The type of gain/constant value can be scalar and array.	[6], [3], [2], [5], [1]
5	Incorrect Simulink blocks	Changing arithmetic operators, e.g., replacing '-' with '+' or '*' or '/'. Changing relation operators, e.g., replacing '$\leq$' with '$\geq$' or '=' or '$\neq$'. Changing logical operator, e.g., replacing 'AND' with 'OR' or 'NOR'. Changing inputs value of Sum block, e.g., replacing inputs value '++' with '+-' or '–'.	[6], [3], [2], [5], [1]
6	False initial conditions and sample time	Changing the initial value and sampletime in 'Integration' and 'Unit Delay' blocks.	[6], [1]
7	Incorrect transition conditions in Stateflow models	Modifying relation and logical operators, such as 'NOT' to 'AND', '¡' to '¿'. Changing keyword 'before' to 'after'.	[6], [3], [2], [5]
8	Incorrect variable names in Stateflow models	Changing variable names, such as changing 'send(LO)' to 'send(RI)', changing 'Gear=1' to 'Gear=2'.	[6], [5], [1]
9	Incorrect actions in Stateflow models	Changing arithmetic operators, modifying constants. Changing state duration, e.g., changing 'after(10 sec)' to 'after(8 sec)', changing logical operator '&&' to '___'.	[6], [5], [1]
10	Changing keywords in Stateflow models	Changing transition-related keywords, e.g., changing 'before' to 'after' or 'at'. Changing state-related keywords, e.g., changing 'entry' to 'exit' or 'on'. Changing event-related keywords, e.g., changing 'send' to 'broadcast'. Changing time-related keywords, e.g., changing 'every' to 'after'.	[6], [1]
11	Expand expressions in Stateflow models	Introducing negate operators, e.g., replacing '(x==y)' with '!(x==y)'. Adding arithmetic operations, e.g. , replacing 'x' with 'x+1', 'x-1' or 'x¿1'. Adding condition statement, e.g. , replacing 'a==1' with 'a==1 —— b==3'.	[3], [4]

References

[1] Nguyen Thanh Binh et al. Mutation operators for simulink models. In *2012 Fourth International Conference on Knowledge and Systems Engi-*

neering, pages 54–59. IEEE, 2012.

[2] Nannan He, Philipp Rümmer, and Daniel Kroening. Test-case generation for embedded simulink via formal concept analysis. In *Proceedings of the 48th Design Automation Conference*, pages 224–229, 2011.

[3] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, 37(5):649–678, 2010.

[4] Ahmed Khanfir, Renzo Degiovanni, Mike Papadakis, and Yves Le Traon. Efficient mutation testing via pre-trained language models. *arXiv preprint arXiv:2301.03543*, 2023.

[5] Khuat Thanh Le Thi My Hanh and Nguyen Thanh Binh Tung. Mutation-based test data generation for simulink models using genetic algorithm and simulated annealing. *International Journal of Computer and Information Technology*, 3(04):763–771, 2014.

[6] Reza Matinnejad, Shiva Nejati, Lionel C Briand, and Thomas Bruckmann. Test generation and test prioritization for simulink models with dynamic behavior. *IEEE Transactions on Software Engineering*, 45(9):919–944, 2018.