National Technical University of Athens
School of Electrical and Computer Engineering
Division of Software Engineering

Efficient Implementation of Conformance Testing Techniques in Active Automata Learning

Evaluation of variations on popular conformance testing algorithms

---

Diploma Thesis

of

ANASTASIOS STEFANOS ANAGNOSTOU

Supervisor: Konstantinos Sagonas
Assoc. Professor NTUA

Athens, May 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Software Engineering

Efficient Implementation of Conformance Testing Techniques in Active Automata Learning

Evaluation of variations on popular conformance testing algorithms

---

Diploma Thesis

of

ANASTASIOS STEFANOS ANAGNOSTOU

Supervisor: Konstantinos Sagonas
Assoc. Professor NTUA

Approved by the examination committee on 16 May 2025.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Konstantinos Sagonas	Dimitris Fotakis	Aristeidis Pagourtzis
Assoc. Professor NTUA	Professor NTUA	Professor NTUA

Athens, May 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Software Engineering

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Αναστάσιος Στέφανος
Αναγνώστου, Graduate of
School of Electrical and
Computer Engineering,
National Technical
University of Athens

16 June 2025

Περίληψη

Η παρούσα εργασία προτείνει αλγορίθμους συμμόρφωσης στο πλαίσιο της Ενεργής Εκμάθησης Αυτομάτων, οι οποίοι αξιοποιούν γνώσεις από προηγούμενους γύρους εκμάθησης για τη βελτίωση της αποδοτικότητας στην αναζήτηση αντιπαραδειγμάτων. Η έμφαση δίνεται στην αξιολόγηση του αντίκτυπου της στοχοθέτησης των νεοεκμαθημένων καταστάσεων μιας υπόθεσης. Τρεις τροποποιήσεις υπαρχόντων αλγορίθμων—της μεθόδου W, της μεθόδου Wp και της μεθόδου Random Wp—δοκιμάστηκαν παράλληλα με μία νέα προσέγγιση, τη μέθοδο Stochastic State Coverage. Τα πειραματικά αποτελέσματα δείχνουν ότι η τροποποίηση της μεθόδου Wp υπερέχει της αρχικής της εκδοχής. Αντίθετα, οι τροποποιήσεις της μεθόδου W και της μεθόδου Random Wp, καθώς και η νεοπροταθείσα μέθοδος, έδωσαν μικτά αποτελέσματα, χωρίς σαφή συνολική βελτίωση σε όλα τα σενάρια. Η μελέτη αυτή δείχνει ότι η στοχοθέτηση των νέων καταστάσεων μιας υπόθεσης μπορεί να βελτιώσει την απόδοση των δοκιμών συμμόρφωσης. Μελλοντική εργασία μπορεί να εστιάσει στην διερεύνηση των συνθηκών υπό τις οποίες επιτυγχάνονται καλύτερα αποτελέσματα, καθώς και στην ανάπτυξη νέων αλγορίθμων που ακολουθούν παρόμοια προσέγγιση.

Λέξεις Κλειδιά

Εκμάθηση Μοντέλων, Ενεργή Εκμάθηση Αυτομάτων, Έλεγχος Λογισμικού, Έλεγχος Πρωτοκόλλων, Έλεγχος Συμμόρφωσης

Abstract

This thesis proposes conformance testing algorithms, in the context of Active Automata Learning, that leverage knowledge from previous learning rounds to improve the efficiency of counterexample search. The focus is on evaluating the impact of targeting the newly learned states of a hypothesis. Three modifications of existing algorithms—the W Method, the Wp Method, and the Random Wp Method—were tested alongside a novel approach, the Stochastic State Coverage Method. Experimental results show that the modification of the Wp Method outperforms its original version. In contrast, the modifications of the W Method and the Random Wp Method, as well as the newly proposed method, yielded mixed results, with no clear overall improvement across all scenarios. This study shows that targeting the new states of a hypothesis may improve conformance testing performance. Future work can focus on exploring the circumstances under which better results can be achieved, as well as on the development of new algorithms that follow a similar approach.

Keywords

Model Learning, Active Automata Learning, Software Testing, Protocol Testing, Conformance Testing

Στην οικογένειά μου, τους φίλους μου, τους καθηγητές μου και σε όσους πίστεψαν σε εμένα.

Acknowledgements

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Κωστή Σαγώνα, για την καθοδήγησή του και γιατί είναι ο λόγος που ήρθα σε επαφή με το αντίκειμενο της διπλωματικής μου. Απευθύνω ένα ιδιαίτερο ευχαριστώ στους καθηγητές που υπερέβησαν τον ρόλο τους και δεν μου έδειξαν μόνο πως να δουλεύω σωστά, αλλά αποτέλεσαν πρότυπα με τον χαρακτήρα τους. Σε ευχαριστώ Κώστα, Μηνά, Αντρέα, Νατάσα. Τέλος, είμαι ευγνώμων στην οικογένειά μου για την σταθερότητα και την αγάπη της, όπως και στην κοπέλα μου Νικολέτα που στέκεται δίπλα μου. Δεν μπορώ να ελπίζω για καλύτερο σύντροφο. Ευχαριστώ.

Αθήνα, Ιούνιος 2025

Αναστάσιος Στέφανος Αναγνώστου

Contents

List of Figures

List of Tables

Εκτενής Ελληνική Περίληψη

Το κεφάλαιο αυτό παρουσιάζει συνοπτικά στα Ελληνικά το σύνολο της διπλωματικής εργασίας.

Εισαγωγή

Η εκμάθηση μοντέλων (Model Learning) είναι ο τομέας της επιστήμης υπολογιστών που ασχολείται με την κατασκευή μοντέλων συστημάτων μέσω της παρατήρησης της συμπεριφοράς τους. Ειδικότερα, η ενεργή εκμάθηση αυτομάτων (Active Automata Learning) αντιμετωπίζει το υπό μάθηση σύστημα (SUL) ως ένα μαύρο κουτί και του θέτει ερωτήματα με ακολουθίες εισόδου, καθοδηγώντας τη διαδικασία μάθησης. Ο τελικός στόχος της εκμάθησης ενεργών αυτομάτων είναι η κατασκευή μιας μηχανής πεπερασμένων καταστάσεων (FSM) που είναι ισοδύναμη με το SUL. Διαφορετικοί αλγόριθμοι μπορούν να προταθούν ανάλογα με τον τύπο της FSM που επιλέγεται για το τελικό μοντέλο. Η παρούσα διατριβή ασχολείται με τους αλγορίθμους που κατασκευάζουν μηχανές Mealy.

Οι αλγόριθμοι ενεργής εκμάθησης αυτομάτων λειτουργούν με βάση το πλαίσιο του ελάχιστα επαρκούς δασκάλου (Minimally Adequate Teacher), που προτάθηκε το 1987 από την Angluin [4]. Το MAT είναι ένα μαντείο που μπορεί να απαντήσει σε ερωτήματα ισοδυναμίας, δηλαδή αν ένα δεδομένο αυτόματο είναι ισοδύναμο με το SUL. Αν το αυτόματο είναι ισοδύναμο με το το SUL, ο δάσκαλος απαντάει 'ναι', ενώ αν δεν είναι, παρέχει κάποια ακολουθία εισόδου που διαφοροποιε το συγκεκριμένο αυτόματο από το SUL, που ονομάζεται αντιπαράδειγμα.

Με αυτόν τον τρόπο, η διαδικασία εκμάθησης μπορεί να χωριστεί σε δύο διακριτά μέρη: το μέρος της κατασκευής υποθέσεων , το οποίο χειρίζεται ο αλγόριθμος εκμάθησης, και το μέρος της επικύρωσης υποθέσεων (ερώτημα ισοδυναμίας), το οποίο χειρίζεται ο δάσκαλος. Στην πράξη, δεν υπάρχει ένας δάσκαλος που να έχει τέλεια γνώση του SUL, οπότε ο δάσκαλος προσεγγίζεται με μεθόδους ελέγχου συμμόρφωσης (Conformance Testing). Οι μέθοδοι ελέγχου συμμόρφωσης προσπαθούν να αποδείξουν την ισοδυναμία της υπόθεσης με το SUL κατασκευάζοντας μια σουίτα δοκιμών, η οποία είναι ένα σύνολο εισόδων για το σύστημα, και ρωτώντας τόσο την υπόθεση όσο και το SUL με κάθε περίπτωση δοκιμής του της σουίτας δοκιμών. Αν δεν βρεθεί καμία ασυμφωνία μεταξύ των δύο συστημάτων, τότε θεωρούνται ισοδύναμα, ενώ αν βρεθεί μία ασυμφωνία, τότε η περίπτωση δοκιμής που την προκάλεσε αποτελεί το αντιπαράδειγμα.

Η έρευνα για την εκμάθηση μοντέλων επικεντρώνεται, μεταξύ άλλων, στην ελαχιστοποίηση του συνολικού αριθμού ερωτημάτων προς το SUL, τόσο κατά τη διάρκεια του μέρους κατασκευής υποθέσεων όσο και κατά τη διάρκεια του μέρους επικύρωσης υποθέσεων. Ωστόσο, έχει αποδειχθεί [13, 1] ότι ο αριθμός των ερωτημάτων κατά την επικύρωση της υπόθεσης είναι ο κυρίαρχος παράγοντας στο του συνολικού αριθμού των ερωτημάτων που αποστέλλονται στο SUL. Ως εκ τούτου, το ενδιαφέρον έχει μετατοπιστεί στη βελτιστοποίηση των αλγορίθμων ελέγχου συμμόρφωσης.

Μια συνέπεια του πλαισίου MAT είναι ότι όλα τα μαθησιακά πειράματα λειτουργούν σε κύκλους κατασκευής υποθέσεων και επικύρωσης υποθέσεων. Όπως θα δείξουμε στα επόμενα κεφάλαια, οι περισσότεροι αλγόριθμοι ελέγχου συμμόρφωσης αντιμετωπίζουν κάθε υπόθεση ως ανεξάρτητο αντικείμενο και δεν επαναχρησιμοποιούν πληροφορίες από προηγούμενους γύρους μάθησης. Η παρούσα εργασία διερευνά πώς η τροποποίηση ορισμένων αυτών των δημοφιλών αλγορίθμων ώστε να ενσωματώνουν πληροφορίες από διάφορους γύρους επηρεάζει την αποδοτικότητα των ερωτημάτων τους, ιδίως όσον αφορά τον αριθμό των ερω-

τημάτων που αποστέλλονται στο SUL.

Θεωρητικό Υπόβαθρο / Επικύρωση Υπόθεσης

Ο σκοπός αυτής της ενότητας είναι να παρουσιαστούν οι διάφορες μέθοδοι που έχουν χρησιμοποιηθεί για την υλοποίηση της επικύρωσης υπόθεσης (equivalence query) στο πλαίσιο των αλγορίθμων ενεργής εκμάθησης αυτομάτων, καθώς και να αναδειχθούν ορισμένα χαρακτηριστικά και διαφορές στις προσεγγίσεις τους.

Προαπαιτούμενα

Μία μηχανή Mealy είναι μία εξάδα $M = (S, S_0, I, O, \delta, \lambda)$, όπου S είναι το σύνολο των καταστάσεων, S_0 είναι η αρχική κατάσταση, I είναι το αλφάβητο εισόδου, O είναι το αλφάβητο εξόδου, $\delta : S \times I \to S$ είναι η συνάρτηση μετάβασης και $\lambda : S \times I \to O$ είναι η συνάρτηση εξόδου. Η συνάρτηση μετάβασης αντιστοιχίζει μία κατάσταση και ένα σύμβολο εισόδου στην επόμενη κατάσταση, ενώ η συνάρτηση εξόδου αντιστοιχίζει μία κατάσταση και ένα σύμβολο εισόδου στο αντίστοιχο σύμβολο εξόδου. Το σύνολο όλων των δυνατών ακολουθιών εισόδου συμβολίζεται με I^* και το σύνολο όλων των δυνατών ακολουθιών εξόδου με O^*.

Τυχαίος Έλεγχος

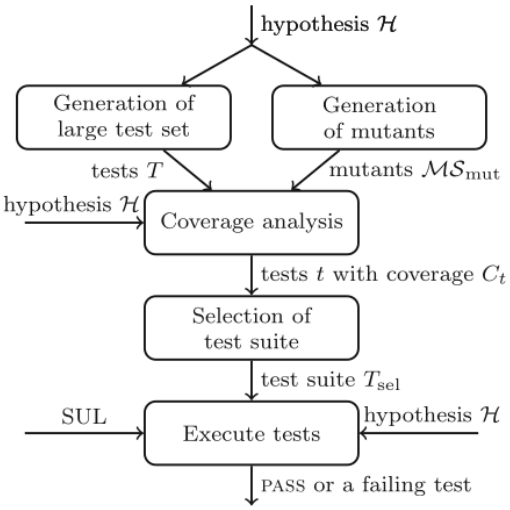
Ο τυχαίος έλεγχος περιλαμβάνει τη δειγματοληψία τυχαίων ακολουθιών από το I^*, όπου I είναι το αλφάβητο εισόδου, και τη χρήση αυτών ως περιπτώσεις ελέγχου. Περιλαμβάνει τις *τυχαίες διαδρομές* (random walks), όπου το μήκος των ακολουθιών εισόδου ακολουθεί γεωμετρική κατανομή, και τις *τυχαίες λέξεις* (random words), όπου το μήκος των ακολουθιών εισόδου είναι ομοιόμορφα κατανεμημένο σε κάποιο προκαθορισμένο εύρος.

Κάλυψη Μεταβάσεων / Επιλογή με Μεταλλάξεις

Η μέθοδος Transition Coverage [2] λειτουργεί δημιουργώντας ένα μεγάλο σύνολο ελέγχων ξεκινώντας με μία τυχαία διαδρομή μέσω του υποδείγματος και επιλέγοντας επαναληπτικά μία μετάβαση του μοντέλου την οποία προσπαθεί να προσεγγίσει από την τρέχουσα κατάσταση. Τέλος, εκτελείται ακόμη μία τυχαία διαδρομή. Αφού παραχθεί το σύνολο ελέγχων, ένα υποσύνολο των περιπτώσεων ελέγχου επιλέγεται άπληστα, με βάση το ποσοστό κάλυψης μεταβάσεων που προσφέρει η κάθε περίπτωση.

Οι συγγραφείς γενικεύουν τη μέθοδο, παρατηρώντας ότι η κάλυψη μεταβάσεων δεν είναι το μόνο είδος κάλυψης που μπορεί να οριστεί και ότι μπορούν να χρησιμοποιηθούν διαφορετικά κριτήρια κάλυψης με χρήση *τελεστών μετάλλαξης* (mutation operators). Οι τελεστές μετάλλαξης δέχονται ως είσοδο μία μηχανή Mealy και παράγουν ένα σύνολο μεταλλαγμένων μοντέλων. Οι περιπτώσεις ελέγχου που επιλέγονται για το τελικό σύνολο είναι αυτές που προσφέρουν τη μέγιστη κάλυψη, δηλαδή αυτές που διακρίνουν τα περισσότερα μεταλλαγμένα μοντέλα.

Είναι εύκολο να επαληθευθεί ότι οι τελεστές μετάλλαξης είναι πιο γενικοί από την κάλυψη μεταβάσεων. Μπορεί να οριστεί τελεστής μετάλλαξης που αλλάζει την έξοδο μίας συγκεκριμένης μετάβασης. Η εκτέλεση αυτής της μετάβασης μπορεί να εντοπίσει το πολύ έναν μεταλλαγμένο μοντέλο. Άρα, η επιλογή περιπτώσεων ελέγχου βάσει αυτού του τελεστή μετάλλαξης είναι ισοδύναμη με την επιλογή βάσει κάλυψης μεταβάσεων.



Σχήμα 1: *Διάγραμμα ροής της επιλογής βασισμένης σε μεταλλάξεις [2].*

Ένα πλεονέκτημα αυτής της μεθόδου είναι ότι μπορεί να χρησιμοποιήσει γνώση του πεδίου, δηλαδή συγκεκριμένους τελεστές μετάλλαξης, για να εντοπίσει αντιπαραδείγματα πιο αποδοτικά.

Η Μέθοδος W

Η μέθοδος W [5] χρησιμοποιεί δύο σύνολα: το σύνολο W, που ονομάζεται *σύνολο χαρακτηρισμού*, δηλαδή ένα σύνολο ακολουθιών εισόδου που διακρίνουν μεταξύ κάθε ζεύγους καταστάσεων του SUL, και το σύνολο P, το οποίο είναι ένα σύνολο κάλυψης μεταβάσεων του S. Υποθέτει ένα ανώτατο όριο m στον αριθμό των καταστάσεων n του SUL και ορίζει το σύνολο $I[m-n] = \bigcup_0^{m-n} I$, όπου I είναι το αλφάβητο εισόδου. Το σύνολο ελέγχου που κατασκευάζει η μέθοδος είναι το σύνολο ακολουθιών εισόδου $P \cdot Z = P \cdot I[m-n] \cdot W$, όπου $\cdot$ είναι ο τελεστής συνένωσης.

Ουσιαστικά, οι είσοδοι που ανήκουν στο σύνολο κάλυψης μεταβάσεων P χρησιμοποιούνται ώστε κάθε ακμή να διανυθεί τουλάχιστον μία φορά, και, για την κατάσταση στην οποία οδηγεί κάθε ακμή, χρησιμοποιούνται οι είσοδοι του συνόλου Z ώστε να διακρίνονται οι καταστάσεις.

Η μέθοδος W και οι παραλλαγές της αποδεικνύουν ισοδυναμία έως το όριο m. Αυτό σημαίνει ότι, αν το υπόδειγμα δεν αποτύχει στο σύνολο ελέγχου, τότε είτε είναι ισοδύναμο με την SUL, είτε το SUL έχει αυστηρά περισσότερες από m καταστάσεις.

Η μέθοδος W είναι αντιπροσωπευτικό παράδειγμα των αλγορίθμων τύπου *Πρόσβαση-Βήμα-Ταυτοποίηση* (Access-Step-Identify). Πρόκειται για αλγόριθμους δοκιμών συμμόρφωσης των οποίων η λειτουργία μπορεί να διαιρεθεί σε τρία μέρη: το μέρος *πρόσβασης*, όπου επιτυγχάνεται πρόσβαση σε μία συγκεκριμένη κατάσταση, το μέρος *βήματος*, όπου εισάγεται μία ακολουθία εισόδου για μετάβαση σε νέα κατάσταση, και το μέρος *ταυτοποίησης*, όπου εισάγεται μία διακριτική ακολουθία για την αναγνώριση της κατάστασης. Έχει παρατηρηθεί ότι ο μόνος τρόπος βελτίωσης των αλγορίθμων αυτής της κατηγορίας είναι η βελτίωση του μέρους *ταυτοποίησης*, που βασίζεται στις διακριτικές ακολουθίες που χρησιμοποιούνται [7].

Η Μερική Μέθοδος W (Μέθοδος Wp)

Η μερική μέθοδος W (Wp-method) [16] είναι μία βελτίωση της μεθόδου W. Αντί να χρησιμοποιεί το χονδρικό σύνολο χαρακτηρισμού W, η μέθοδος Wp χρησιμοποιεί λεπτομερέστερα σύνολα ταυτοποίησης W_i για κάθε κατάσταση s_i του SUL. Ξεκινά με την επαλήθευση ότι οι καταστάσεις του SUL είναι ταυτοποιήσιμες στο υπόδειγμα, ενώ παράλληλα κατασκευάζει τα σύνολα W_i. Αυτά μπορούν να χρησιμοποιηθούν μεταγενέστερα για πιο αποδοτική επαλήθευση της ισοδυναμίας των μεταβάσεων του υποδείγματος με αυτές του SUL.

Υπάρχει και μία *τυχαία* παραλλαγή αυτής της μεθόδου [15], όπου εκτελείται ένα τυχαίο υποσύνολο του συνόλου ελέγχου. Έχει παρατηρηθεί ότι βρίσκει αντιπαραδείγματα ταχύτερα [3] [13], αλλά δεν αποδεικνύει ισοδυναμία όπως οι ντετερμινιστικές μέθοδοι.

Εναρμονισμένα Αναγνωριστικά Καταστάσεων (HSI)

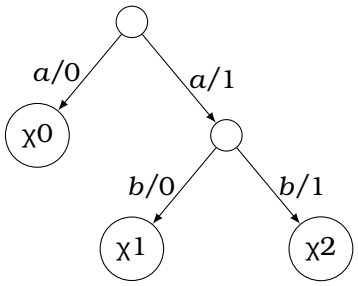
Η μέθοδος HSI [20], όπως και οι μέθοδοι W και Wp, χρησιμοποιεί την έννοια του συνόλου χαρακτηρισμού αλλά αναγνωρίζει ότι δεν είναι απαραίτητη η χρήση ολόκληρου του συνόλου. Αντίθετα, χρησιμοποιεί υποσύνολα αυτού που ονομάζονται *εναρμονισμένα σύνολα ταυτοποίησης καταστάσεων* (harmonized state identification sets). Αυτά τα σύνολα έχουν την ιδιότητα ότι, αν υπάρχει διακριτική ακολουθία για δύο καταστάσεις, τότε αυτή είναι πρόθεμα και των δύο συνόλων. Ένα πλεονέκτημα της μεθόδου είναι ότι εφαρμόζεται και σε μερικώς ορισμένες μη ντετερμινιστικές μηχανές και τείνει να παράγει μικρότερα σύνολα ελέγχου σε σχέση με τις μεθόδους W και Wp.

Προκαθορισμένες Διακριτικές Ακολουθίες (PDS)

Οι προκαθορισμένες διακριτικές ακολουθίες (Preset Distinguishing Sequences) είναι ακολουθίες εισόδου που μπορούν να χρησιμοποιηθούν για την ταυτοποίηση της τρέχουσας κατάστασης του συστήματος από όλες τις υπόλοιπες. Ουσιαστικά, το πιο ελάχιστο δυνατό σύνολο χαρακτηρισμού W αποτελείται από μία μόνο προκαθορισμένη διακριτική ακολουθία, εφόσον αυτή η ακολουθία και μόνο μπορεί να διακρίνει κάθε κατάσταση από κάθε άλλη. Δυστυχώς, μία τέτοια ακολουθία δεν υπάρχει σε κάθε FSM και, ως εκ τούτου, τεχνικές που βασίζονται αποκλειστικά στην ύπαρξη PDS δεν μπορούν να χρησιμοποιηθούν καθολικά.

Προσαρμοστικές Διακριτικές Ακολουθίες (ADS)

Μία προσαρμοστική διακριτική ακολουθία (Adaptive Distinguishing Sequence) [19] είναι ένα δέντρο αποφάσεων με στόχο την ταυτοποίηση της τρέχουσας κατάστασης του συστήματος. Δομικά, είναι ένα δέντρο του οποίου οι κόμβοι είναι είσοδοι που αντιστοιχούν σε σύνολα πιθανών καταστάσεων και οι ακμές είναι έξοδοι που παρατηρούνται ως αποκρίσεις στην είσοδο. Διαφέρει από τις προκαθορισμένες διακριτικές ακολουθίες στο ότι παρατηρεί την έξοδο του εκάστοτε ερωτήματος και στη συνέχεια αποφασίζει ποιο θα είναι το επόμενο σύμβολο της ακολουθίας, αντί να είναι η ακολουθία προκαθορισμένη από την αρχή. Το πλεονέκτημά της είναι το ότι μία FSM που δεν παραδέχεται PDS, μπορεί να παραδέχεται ADS. Ωστόσο, ακόμη και μία ADS ενδέχεται να μην υπάρχει για ορισμένες FSM, και άρα δεν μπορεί να χρησιμοποιηθεί καθολικά.



Σχήμα 2: *Απλή ADS που δείχνει πώς, με είσοδο 'a' και έξοδο '0', αναγνωρίζεται η κατάσταση q_0, ενώ με εισόδους 'ab' και εξόδους '10' και '11' αντίστοιχα, αναγνωρίζονται οι καταστάσεις q_1 και q_2.*

Για λόγους σαφήνειας, πρέπει να αναφερθεί ότι οι ADS έχουν επίσης χρησιμοποιηθεί και στο μέρος εκμάθησης των αλγορίθμων AAL [12].

Υβριδική Προσαρμοστική Διακριτική Ακολουθία (H-ADS)

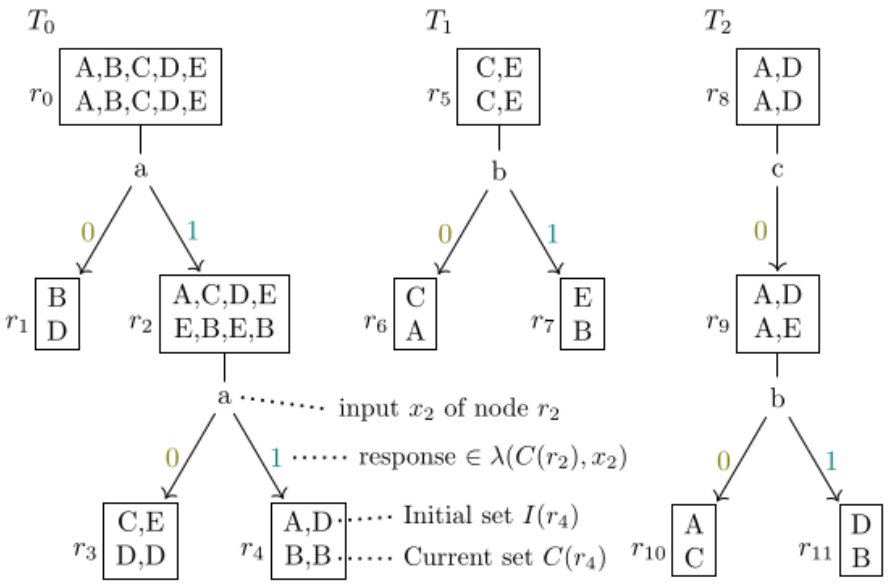
Οι υβριδικές προσαρμοστικές διακριτικές ακολουθίες (Hybrid ADS ή H-ADS) [24] βελτιώνουν τις ADS προσφέροντας καθολική εφαρμοσιμότητα. Βρίσκεται μία ελλιπής ADS[1], της οποίας τα φύλλα μπορεί να είναι σύνολα καταστάσεων αντί για μοναδικές καταστάσεις. Αυτή η ακολουθία εμπλουτίζεται στη συνέχεια με διακριτικές ακολουθίες ανά ζεύγος καταστάσεων, προκειμένου να διακριθούν οι πιθανοί υποψήφιοι καταστάσεων.

Αξιοσημείωτο είναι ότι οι συγγραφείς κάνουν επίσης χρήση αυτού που αποκαλούν «υποαλφάβητα» (subalphabets). Στην προσπάθεια να βρουν πιο εύκολα παράξενα αντιπαραδείγματα, δημιουργούν πρώτα περιπτώσεις ελέγχου χρησιμοποιώντας το υποαλφάβητο που προκύπτει από το αντιπαράδειγμα του προηγούμενου υποδείγματος και έπειτα περιπτώσεις με το πλήρες αλφάβητο. Πρόκειται για την πρώτη εμφάνιση μετάδοσης γνώσης από προηγούμενο προς το τρέχον στάδιο επικύρωσης υπόθεσης.

Ελλιπείς Προσαρμοστικές Διακριτικές Ακολουθίες (I-ADS)

Οι ελλιπείς προσαρμοστικές διακριτικές ακολουθίες (Incomplete Adaptive Distinguishing Sequences ή I-ADS) [14] στοχεύουν επίσης στη διόρθωση των περιορισμών των ADS. Οι I-ADS είναι ένα σύνολο από ADS που μπορούν να χρησιμοποιηθούν όταν δεν υπάρχει ADS για το δεδομένο σύστημα. Ουσιαστικά, οι I-ADS έχουν την ίδια δομή με μία ADS, αλλά τα φύλλα τους δεν απαιτείται να είναι μοναδικές καταστάσεις· μπορεί να είναι και σύνολα καταστάσεων. Έτσι, όταν κατά τη διάρκεια της διαδικασίας ταυτοποίησης συναντάται τέτοιο φύλλο, στέλνεται reset στο σύστημα και μία ακολουθία πρόσβασης που οδηγεί το σύστημα σε ένα σύνολο πιθανών αρχικών καταστάσεων που μπορούν πλέον να διακριθούν. Με αυτό τον τρόπο, πολλές I-ADS μπορούν να χρησιμοποιηθούν από κοινού ως μία ADS, αλλά απαιτούν reset και επιπλέον ακολουθίες πρόσβασης.

[1] να μην συγχέεται με την τεχνική IADS που έχει το ίδιο όνομα.



Σχήμα 3: *Τρεις διαφορετικές I-ADS που συνδυαστικά σχηματίζουν ένα πλήρες διακριτικό σύνολο για μία υποθετική αυτόματη μηχανή με πέντε καταστάσεις, από τους Soucha and Bogdanov [27].*

Μέθοδος H

Η μέθοδος H [7] [8] διαφέρει από τις άλλες μεθόδους αποφεύγοντας τη χρήση σταθερών διακριτικών ακολουθιών. Οι συγγραφείς αξιοποιούν το γεγονός ότι κάθε m-πλήρες σύνολο ελέγχου περιέχει διακριτικές ακολουθίες για κάθε μετάβαση του συστήματος. Υποστηρίζουν ότι κάθε πλήρες σύνολο ελέγχου TS περιέχει το σύνολο $V \cdot \text{Pref}(X^{m-n+1})$, όπου V είναι ένα κλειστό ως προς τα προθέματα σύνολο κάλυψης καταστάσεων της προδιαγραφής και X είναι το αλφάβητο εισόδου. Αν η υπόθεση μπορεί να διακριθεί από την προδιαγραφή μέσω μιας ακολουθίας εισόδου μεγαλύτερης από το επιτρεπόμενο μήκος του συνόλου ελέγχου, τότε το σύνολο ελέγχου περιέχει σίγουρα τουλάχιστον δύο ακολουθίες που είναι προθέματα αυτής, οδηγούν την υπόθεση στην ίδια κατάσταση t, αλλά οδηγούν την προδιαγραφή σε δύο διαφορετικές καταστάσεις s_1, s_2. Συνεπώς, αν υπάρχει μία ακολουθία που διακρίνει το s_1 από το s_2, τότε διακρίνει και το t από τα s_1, s_2. Με βάση αυτή τη διαπίστωση, προτείνουν έναν αλγόριθμο που κατασκευάζει ένα m-πλήρες σύνολο ελέγχου με τέτοιου είδους διακριτικές ακολουθίες, του οποίου οι περιπτώσεις ελέγχου τείνουν να είναι μικρότερες από εκείνες της παραδοσιακής μεθόδου W.

Μέθοδος SPY

Η μέθοδος SPY [23] αποσκοπεί στη μείωση των διακλαδώσεων μέσα στο σύνολο ελέγχου. Η διακλάδωση συμβαίνει όταν διαφορετικές περιπτώσεις ελέγχου μοιράζονται ένα κοινό πρόθεμα. Όταν συμβαίνει αυτό, το σύνολο ελέγχου περιέχει κάποια πλεονάζοντα δεδομένα, καθώς η κοινή ακολουθία εισόδου πρέπει να επαναχρησιμοποιηθεί πολλές φορές. Οι συγγραφείς παρατηρούν ότι, εφόσον έχουν ήδη συντομευτεί οι ακολουθίες μεταφοράς και έχουν χρησιμοποιηθεί οι πιο σύντομες διακριτικές ακολουθίες, ο μόνος τρόπος περαιτέρω μείωσης

του μήκους του συνόλου ελέγχου είναι η μείωση των διακλαδώσεων. Δηλαδή, η μείωση του αριθμού επαναχρησιμοποιήσεων κοινών προθεμάτων. Καθιερώνουν τις συνθήκες που πρέπει να πληρούνται για να επιτευχθεί αυτή η μείωση χωρίς να χαθεί η m-πληρότητα και προτείνουν μία μέθοδο παραγωγής συνόλων ελέγχου που συνδυάζει μικρά υποκλαδιά σε μεγαλύτερες ακολουθίες, με αποτέλεσμα μικρότερα σύνολα ελέγχου.

Μέθοδος SPY-H

Η μέθοδος SPY-H [26] επιχειρεί να συνδυάσει τα πλεονεκτήματα των μεθόδων SPY και H, ώστε να παραχθούν ακόμη μικρότερα σύνολα ελέγχου. Στοχεύει να ικανοποιεί τις ίδιες συνθήκες που εξασφαλίζουν m-πληρότητα όπως η μέθοδος SPY, αλλά επαληθεύει την προσπελάσιμη κατάσταση μέσω διακριτικών ακολουθιών, όπως αυτές που χρησιμοποιούνται από τη μέθοδο H.

Μοντέλο Συνδυασμού Ειδικών (Mixture of Experts)

Η μέθοδος Mixture of Experts [18] επιχειρεί να επιταχύνει την αναζήτηση αντιπαραδειγμάτων, ξεκινώντας από μικρότερα σύνολα ελέγχου που είναι πιο πιθανό να τα περιέχουν, τα οποία προκύπτουν από προηγούμενους γύρους ερωτημάτων ισοδυναμίας. Αυτά τα μικρότερα σύνολα ελέγχου παράγονται από 'ειδικούς' (experts), δηλαδή συναρτήσεις που δέχονται ως είσοδο μία λέξη και μία μηχανή Mealy και επιστρέφουν σύνολα υποαλφαβήτων. Η μέθοδος μπορεί να ενσωματωθεί σε οποιονδήποτε αλγόριθμο τύπου Access-Step-Identify (όπως αυτοί που περιγράφηκαν παραπάνω). Χρησιμοποιεί έναν συνδυασμό αυτών των ειδικών και έναν αλγόριθμο ενισχυτικής μάθησης για να προσαρμόζει δυναμικά την πιθανότητα να αναζητηθεί αντιπαράδειγμα στα υποαλφάβητα κάθε ειδικού σε κάθε δοκιμή. Οι πιθανότητες αυτές μεταφέρονται και στους επόμενους γύρους ερωτημάτων ισοδυναμίας, καθιστώντας τη —κατά την άποψη του συγγραφέα— την πρώτη μέθοδο που αξιοποιεί συστηματικά πληροφορία από πολλαπλά ερωτήματα ισοδυναμίας.

AALpy Μαντεία Ισοδυναμίας

Η AALpy [21] είναι μια βιβλιοθήκη για την ενεργή εκμάθηση αυτομάτων σε Python, η οποία υλοποιεί διάφορους αλγορίθμους μαντείων ισοδυναμίας. Το σύνολο των υλοποιημένων αλγορίθμων περιλαμβάνει τόσο μεθόδους όπως αυτές που περιγράφηκαν παραπάνω και έχουν αξιολογηθεί σε ξεχωριστές δημοσιεύσεις, όσο και ευρετικές μεθόδους που δεν έχουν αναλυθεί εκτενώς. Οι συγγραφείς παρέχουν συνοπτικές περιγραφές για αυτές, αλλά δεν τις αξιολογούν πειραματικά.

Μαντείο Εστίασης Μεταβάσεων (Transition Focus Oracle)

Το μαντείο Transition Focus βασίζεται σε τυχαίο περίπατο και παραμετροποιείται με μια παράμετρο ϵ, έτσι ώστε κάθε είσοδος να οδηγεί στην ίδια κατάσταση με πιθανότητα ϵ και σε νέα κατάσταση με πιθανότητα $1 - \epsilon$.

Μαντείο Probably Approximately Correct (PAC)

Το μαντείο PAC βασίζεται σε τυχαίες λέξεις. Παραμετροποιείται με τις παραμέτρους ϵ και δ, οι οποίες καθορίζουν τον αριθμό των δοκιμών κατά τη διάρκεια της διαδικασίας επικύρωσης, έτσι ώστε να εγγυάται ότι το μοντέλο που έχει μαθευτεί είναι μια ϵ-προσέγγιση του SUL με πιθανότητα τουλάχιστον $1 - \delta$.

Μαντείο Με Βάση Λανθάνουσα Μνήμη (Cache Based Oracle)

Το μαντείο Cache Based αξιοποιεί την αποθήκευση ερωτημάτων (query caching) που υλοποιεί η AALpy για SULs, με σκοπό την αποφυγή διπλών ερωτημάτων για πρόθεματα με ήδη παρατηρημένες εκτελέσεις. Τα αποτελέσματα των ερωτημάτων αποθηκεύονται σε μια δενδρική δομή, η οποία ενημερώνεται κατά την κατασκευή της υπόθεσης και κατά τον έλεγχο ισοδυναμίας. Κάθε δοκιμή που πραγματοποιεί το μαντείο είναι μια διαδρομή από τη ρίζα του δέντρου έως ένα από τα φύλλα, με ένα τυχαίο περπάτημα συγκεκριμένου μήκους να προστίθεται στο τέλος.

Μαντείο Καλύψεως Μεταβάσεων K-Βαθμού (K-Way Transition Coverage Oracle)

Το μαντείο K-Way Transition Coverage είναι παρόμοιο με το μαντείο καλύψεως μεταβάσεων που περιγράφηκε παραπάνω. Δημιουργεί έναν αριθμό τυχαίων διαδρομών και επιλέγει άπληστα ένα υποσύνολό τους έτσι ώστε να μεγιστοποιηθεί η κάλυψη μεταβάσεων k-βαθμού. Η παράμετρος k αναφέρεται στον αριθμό βημάτων μεταξύ της αρχής και του τέλους μιας μετάβασης.

Μαντείο Καλύψεως Καταστάσεων K-Βαθμού (K-Way State Coverage Oracle)

Το μαντείο K-Way State Coverage εκτελεί μία δοκιμή για κάθε k-συνδυασμό ή k-παραλλαγή καταστάσεων και εξερευνά την περιοχή γύρω από την τελική κατάσταση μέσω ενός τυχαίου περιπάτου.

Μαντείο Προθεμάτων Καταστάσεων (State Prefix Oracle)

Το μαντείο State Prefix προσπελαύνει κάθε κατάσταση μέσω της ακολουθίας πρόσβασης της και εξερευνά την περιοχή γύρω της εκκινώντας έναν αριθμό τυχαίων περιπάτων με μήκος εντός προκαθορισμένου εύρους. Μπορεί επίσης να παραμετροποιηθεί έτσι ώστε να εξερευνά πρώτα τις νέες καταστάσεις της υπόθεσης, καθιστώντας αυτή τη μέθοδο τη δεύτερη που έχει παρατηρηθεί να αξιοποιεί πληροφορία από προηγούμενους γύρους εκμάθησης.

Συζήτηση

Οι περισσότερες μέθοδοι λειτουργούν ξεχωριστά για κάθε υπόθεση και προσπαθούν να μειώσουν το μέγεθος της παραγόμενης δοκιμαστικής σουίτας πειραματιζόμενες με διαφορετικούς τρόπους εύρεσης και χρήσης αναγνωριστικών καταστάσεων και διακριτικών ακολουθιών.

Μια μικρή εξαίρεση σε αυτή την παρατήρηση είναι η μέθοδος H-ADS, η οποία αξιοποιεί επίσης το αντιπαράδειγμα της προηγούμενης υπόθεσης για να αναζητήσει πιο αποδοτικά ένα νέο. Ωστόσο, η τεχνική αυτή φαίνεται να έχει χρησιμοποιηθεί περισσότερο ως πρόχειρη λύση παρά ως αναπόσπαστο μέρος της μεθόδου.

Μια άλλη εξαίρεση είναι το μαντείο προθεμάτων καταστάσεων της AALpy, το οποίο μπορεί να ρυθμιστεί ώστε να εξερευνά πρώτα τις νέες καταστάσεις της υπόθεσης.

Τέλος, υπάρχει η προσέγγιση του Mixture of Experts, η οποία φαίνεται να εστιάζει ακόμη περισσότερο στην ιδέα αξιοποίησης πληροφορίας από προηγούμενους γύρους μαντειών ισοδυναμίας. Είναι επίσης η μόνη προσέγγιση που καλύπτει όλα τα κριτήρια, καθώς ενσωματώνει όλα τα πλεονεκτήματα των αλγορίθμων Access-Step-Identify ενώ ταυτόχρονα αξιοποιεί ιδέες από άλλες μεθόδους.

Είναι προφανές ότι δεν έχει δοθεί ιδιαίτερη έμφαση στην ανάπτυξη μηχανισμών μνήμης μεταξύ διαδοχικών υποθέσεων, οι οποίοι θα μπορούσαν ενδεχομένως να καταστήσουν την αναζήτηση αντιπαραδειγμάτων πιο αποδοτική.

Αξιολόγηση Μεθόδων Υλοποίησης Ερωτημάτων Ισοδυναμίας

Εισαγωγή

Λόγω της απουσίας ενός αληθινού μαντείου ισοδυναμίας στη διαδικασία ενεργής εκμάθησης αυτομάτων, τα ερωτήματα ισοδυναμίας προσεγγίζονται μέσω τεχνικών ελέγχου συμμόρφωσης. Δεδομένου ότι ο χρόνος εκτέλεσης των δοκιμών κυριαρχεί στη διαδικασία εκμάθησης, είναι ευρέως αποδεκτό ότι ο αριθμός και το μήκος των δοκιμών πρέπει να ελαχιστοποιούνται [3]. Για το λόγο αυτό, οι περισσότερες μέθοδοι ελέγχου συμμόρφωσης εστιάζουν στην παραγωγή μικρών σουιτών δοκιμών, τόσο σε πλήθος όσο και σε μεμονωμένο μήκος ακολουθιών.

Εμπειρική Αξιολόγηση

Εκτός από τη θεωρητική ανάλυση πολυπλοκότητας των μεθόδων ελέγχου συμμόρφωσης, είναι κρίσιμο να αξιολογείται η απόδοσή τους εμπειρικά. Τέτοιες αξιολογήσεις πραγματοποιούνται συνήθως είτε σε πραγματικά συστήματα (π.χ. πρωτόκολλα επικοινωνίας ή ελεγκτές υλικού) είτε σε προσομοιωμένα συστήματα, χρησιμοποιώντας είτε μοντέλα που έχουν ήδη μαθευτεί είτε χειρονακτικά κατασκευασμένα μοντέλα ως υποκατάστατα των πραγματικών υλοποιήσεων. Αυτό επιτρέπει ελεγχόμενα πειράματα και αναπαραγωγιμότητα.

Παρότι η αλγοριθμική πολυπλοκότητα παρέχει ένδειξη της αναμενόμενης ή χειρότερης απόδοσης μιας μεθόδου, συνήθως ως προς τον αριθμό καταστάσεων n, δεν αποτυπώνει πλήρως τη δομική ή συμπεριφορική πολυπλοκότητα του μοντέλου. Επομένως, τα μοντέλα πρέπει να χαρακτηριστούν με περισσότερα από το μέγεθός τους. Μια πιο λεπτομερής προσέγγιση ποσοτικοποίησης της δυσκολίας ενός μοντέλου είναι το *hardness score* [3], που ορίζεται ως:

$$\text{hardness score} = \frac{\text{learn-hardness}}{\text{test-chance}} \tag{1}$$

όπου το *learn-hardness* ορίζεται ως:

$$\begin{aligned} \text{learn-hardness} &= \left(k \cdot n^2 + n \cdot \log m\right) \cdot (n + m) \\ k &= \text{μήκος του αλφαβήτου εισόδου} \\ n &= \text{αριθμός καταστάσεων} \\ m &= |p^{max}| + |w^{max}|, \quad \text{προσέγγιση του μεγαλύτερου αντιπαραδείγματος} \end{aligned} \tag{2}$$

και το *test-chance* δίνεται από:

$$\text{test-chance} = \left(|p^{\max}| \cdot k^{|w^{\max}|}\right)^{-1} \tag{3}$$

όπου:

$$\begin{aligned} p^{\max} &= \arg\max_{p} |p|, \ p \in \text{Prefixes}, \quad \text{μεγαλύτερη ακολουθία πρόσβασης} \\ w^{\max} &= \arg\max_{w} |w|, \ w \in W, \quad \text{μεγαλύτερη διακριτική κατάληξη} \end{aligned} \tag{4}$$

Επομένως, ο συνδυασμός του δείκτη δυσκολίας (hardness score) ενός μοντέλου και των μετρικών απόδοσης μιας μεθόδου ελέγχου συμμόρφωσης παρέχει μια πιο σφαιρική κατανόηση της απόδοσης.

Μετρικές Αξιολόγησης

Για να εξασφαλιστεί δίκαιη και ανεξάρτητη σύγκριση, οι περισσότερες μελέτες αναφέρουν αφηρημένες μετρικές που δεν βασίζονται στον πραγματικό χρόνο εκτέλεσης. Συνήθεις μετρικές περιλαμβάνουν:

- το συνολικό πλήθος εκτελεσμένων δοκιμών,
- το συνολικό πλήθος συμβόλων εισόδου σε όλες τις δοκιμές,
- τον αριθμό επαναρυθμίσεων (resets) που εκδόθηκαν προς το σύστημα υπό εκμάθηση, ιδιαίτερα όταν οι επαναρυθμίσεις είναι δαπανηρές ή περιορισμένες [12].

Εκτός από αυτές τις βασικές μετρικές κόστους, κάποιες μελέτες προτείνουν επιπλέον δείκτες. Για παράδειγμα, ο δείκτης *Αεραγε Περσενταγε οφ Φαυλτς Δετεστεδ* (ΑΠΦΔ) [13] αντικατοπτρίζει πόσο γρήγορα εντοπίζονται σφάλματα κατά τη διάρκεια του ελέγχου. Ορίζεται ως:

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{n} \text{TC}_i \cdot \Delta H_i}{\text{TC} \cdot n} + \frac{1}{2 \cdot \text{TC}} \tag{5}$$

όπου

$$\begin{aligned} \Delta H_i &= \text{νέες καταστάσεις που ανακαλύφθηκαν μεταξύ των γύρων εκμάθησης } i-1 \text{ και } i \\ \text{TC} &= \text{συνολικός αριθμός συμβόλων και επαναρυθμίσεων σε όλη την εκμάθηση και τον έλεγχο} \\ \text{TC}_i &= \text{αριθμός συμβόλων και επαναρυθμίσεων μέχρι τον γύρο i} \end{aligned} \tag{6}$$

Άλλες προτάσεις, όπως οι δείκτες s_1 και s_2 [3], δεν είναι απόλυτοι δείκτες απόδοσης, αλλά χρησιμοποιούνται για σύγκριση μεταξύ αλγορίθμων σε ένα σύνολο πειραμάτων. Ο δείκτης s_1 είναι το άθροισμα των μέσων βημάτων των ερωτημάτων συμμετοχής που τέθηκαν κατά τα ερωτήματα ισοδυναμίας σε όλα τα πειράματα εκμάθησης:

$$s_1(\text{comb}) = \sum_{m \in \text{Models}} \text{meanSteps}(\text{comb}, m) \tag{7}$$

όπου:

$$\begin{aligned} \text{comb} &= \text{συνδυασμός μαθητευόμενου - μαντείου} \\ \text{Models} &= \text{το σύνολο μοντέλων υπό εκμάθηση} \\ \text{meanSteps} &= \text{μέσος αριθμός βημάτων ερωτημάτων συμμετοχής κατά τον έλεγχο} \end{aligned}$$

Αυτός ο δείκτης μπορεί να χρησιμοποιηθεί για σύγκριση της απαιτούμενης \μάθησης" (εκμάθησης) από ένα μαντείο ισοδυναμίας για ένα σύνολο μοντέλων. Ωστόσο, δεν παρέχει

πλήρη εικόνα της απόδοσης των μαντείων, καθώς επηρεάζεται από ιδιάζοντα μοντέλα και δεν αποτυπώνει καλά την απόδοση για καθένα μοντέλο ξεχωριστά. Για παράδειγμα, αν ένα μαντείο έχει πολύ κακή απόδοση σε ένα μεγάλο μοντέλο που απαιτεί πολλές ερωτήσεις, η επίδραση στο s_1 μπορεί να είναι πιο σημαντική από τις βελτιώσεις σε πολλά μικρότερα μοντέλα. Γι' αυτό, ο δείκτης s_1 συμπληρώνεται από τον δείκτη s_2.

Ο δείκτης s_2 είναι το άθροισμα των μέσων βημάτων των ερωτήσεων συμμετοχής κατά τα ερωτήματα ισοδυναμίας, διαιρεμένο με τον μέγιστο από αυτούς τους μέσους όρους σε όλα τα πειράματα:

$$s_2(\text{comb}) = \sum_{m \in \text{Models}} \frac{\text{meanSteps}(\text{comb}, m)}{\max_{a \in \text{Combos}} \text{meanSteps}(a, m)} \tag{8}$$

όπου:

$$\text{Combos} = \text{το σύνολο συνδυασμών μαθητευόμενου – μαντείου}$$

Καθώς κάθε όρος διαιρείται με τον αριθμό ερωτήσεων του χειρότερου εκτελεστή, αυτός ο δείκτης μετρά πόσο κοντά βρίσκεται κάθε μοντέλο στον χειρότερο και δεν επηρεάζεται από τάξεις μεγέθους διαφορετικών απαιτήσεων ερωτήσεων για τη μάθηση ενός μοντέλου. Χρησιμοποιείται ως συμπλήρωμα στον δείκτη s_1 για να δείξει πόσο καλά αποδίδει ένα μαντείο ισοδυναμίας σε σχέση με άλλα για ένα συγκεκριμένο σύνολο μοντέλων.

Επιπλέον, είναι σύνηθες να εισάγονται ποινές σε αυτές τις μετρικές για πειράματα που δεν ολοκληρώνονται επιτυχώς ή υπερβαίνουν τα όρια πόρων. Στην περίπτωση των δεικτών s_1 και s_2, αυτό γίνεται προσθέτοντας 1 στον δείκτη s_2 για κάθε αποτυχημένο πείραμα του αξιολογούμενου αλγορίθμου, δεδομένου ότι αυτό είναι το μέγιστο δυνατό κάθε όρου στο άθροισμα.

Διαχείριση Μη ντετερμινισμού

Όταν αποτιμώνται μη ντετερμινιστικές μέθοδοι ελέγχου συμμόρφωσης — όπως εκείνες που βασίζονται σε τυχαία εξερεύνηση ή παραγωγή εισόδων — είναι απαραίτητο να επαναλαμβάνονται πειράματα αρκετές φορές ώστε να ληφθεί υπόψη η διακύμανση. Για παράδειγμα, οι Klees et al. [17] προτείνουν την εκτέλεση κάθε διαμόρφωσης 30 φορές και την αναφορά μέσων αποτελεσμάτων.

Παραμετροποίηση

Σημαντικό για benchmarks που συγκρίνουν διάφορες μεθόδους ελέγχου συμμόρφωσης είναι η δίκαιη και συνεπής ρύθμισή τους. Για παράδειγμα, κατά τη σύγκριση των στρατηγικών Random Walk και Random Word, πρέπει να ρυθμιστούν οι παράμετροι τους έτσι ώστε το αναμενόμενο μήκος των δοκιμών να είναι κατά προσέγγιση ίσο. Ομοίως, άλλες παράμετροι — όπως βάθος αναζήτησης, αναμενόμενο μήκος δοκιμής ή πιθανότητα επαναρύθμισης — πρέπει να ελεγχθούν ώστε οι παρατηρούμενες διαφορές να προέρχονται από τους ίδιους τους αλγορίθμους και όχι από διαφορετική παραμετροποίηση.

Πειραματική Διάταξη

Αρχικά, όλα τα πειράματα εκτελέστηκαν με τη χρήση της βιβλιοθήκης AALpy [21] και όλα τα μαντεία υλοποιήθηκαν επεκτείνοντας την κλάση βάσης μαντείου της βιβλιοθήκης. Οι περισσότερες μέθοδοι που αξιολογήθηκαν υλοποιήθηκαν ως παραλλαγές δημοφιλών μεθόδων, όπως η μέθοδος W, η μέθοδος Wp και η Random Wp μέθοδος. Ωστόσο, η βιβλιοθήκη AALpy στερούσε υλοποίηση των μεθόδων Wp και Random Wp. Ως αποτέλεσμα, η μέθοδος Wp υλοποιήθηκε από το μηδέν και συγχωνεύθηκε στο έργο, με παράλληλη βελτιστοποίηση του κώδικα για τη μέθοδο W. Επιπλέον, η υλοποίηση της LearnLib για την Random Wp Method χρησιμοποιήθηκε ως αναφορά και μεταφέρθηκε στην AALpy.

Για την αξιολόγηση των αλγορίθμων, μάθαμε μηχανές Mealy που προέρχονται από προηγούμενα πειράματα εκμάθησης μοντέλων από υλοποιήσεις διαφόρων πρωτοκόλλων: TLS [6, 25], MQTT [28], TCP [10] και DTLS [11, 9]. Όλα τα πειράματα εκμάθησης πραγματοποιήθηκαν με τον ίδιο λεαρνερ: τον λεαρνερ L^* με επεξεργασία αντιπαραδειγμάτων Ριεστ–Σχαπιρε [22]. Χρησιμοποιήθηκαν επαρκείς πόροι εκμάθησης, ώστε η συντριπτική πλειοψηφία των πειραμάτων να ολοκληρώνεται επιτυχώς για τουλάχιστον ένα μαντείο. Τα πειράματα εκμάθησης που αποτυγχάνουν επιβαρύνονται με ποινή.

Κατά το ερώτημα ισοδυναμίας κάθε γύρου εκμάθησης, μετρώνται οι συνολικές μεμβερσηιπ χυεριες που το μαντείο ισοδυναμίας στέλνει στο SUL. Τα πειράματα με μη ντετερμινιστικούς αλγορίθμους συμμόρφωσης επαναλήφθηκαν 30 φορές, όπως προτείνει η Klees et al. [17]. Εκτός από συνηθισμένες στατιστικές (μέση τιμή, τυπική απόκλιση), χρησιμοποιήθηκαν οι δείκτες s_1 και s_2 της Aichernig, Tappler, and Wallner [3], ελαφρώς τροποποιημένοι. Επειδή ο λεαρνερ ήταν κοινός και τα σηαραστεριζατιον σετς κατασκευάζονταν με την ίδια μέθοδο, μετρήθηκε η μέση τιμή μεμβερσηιπ χυεριες, αντί βημάτων.

Ο δείκτης s_1 ορίζεται ως το άθροισμα των μέσων μεμβερσηιπ χυεριες που έγιναν κατά τα ερωτήματα ισοδυναμίας σε όλα τα πειράματα:

$$s_1(\text{alg}) = \sum_{m \in \text{Models}} \text{avg\_MQ}(\text{alg}, m) \tag{9}$$

Ο δείκτης s_2 είναι το άθροισμα των μέσων μεμβερσηιπ χυεριες διαιρεμένο με τη μέγιστη από αυτές στις συγκρίσιμες μεθόδους:

$$s_2(\text{alg}) = \sum_{m \in \text{Models}} \frac{\text{avg\_MQ}(\text{alg}, m)}{\max_{a \in \text{Algs}} \text{av\_MX}(a, m)} \tag{10}$$

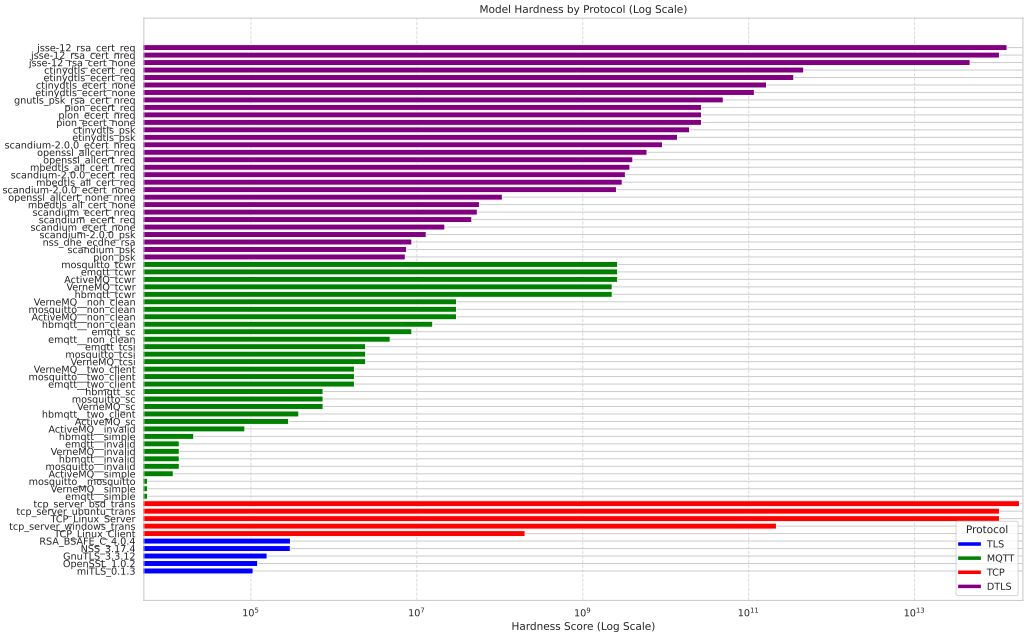
Οι μέσοι όροι υπολογίστηκαν μόνο από επιτυχημένα πειράματα. Έτσι s_1 και s_2 απεικονίζουν την απόδοση των μαντείων σε επιτυχείς εκτελέσεις. Για τις αποτυγχασμένες, χρησιμοποιήθηκε ποινικοποιημένη έκδοση του s_2: προστίθεται ο μέσος αριθμός αποτυχιών στον σορε. Για ντετερμινιστικούς αλγορίθμους, η ποινή είναι πάντα 1 ανά αποτυχία· για μη ντετερμινιστικούς, η ποινή διαιρείται δια του αριθμού των δοκιμών.

Τέλος, υπολογίστηκε μια έκδοση του s_1 αγνοώντας τις χυεριες του τελευταίου γύρου εκμάθησης. Ο τελευταίος γύρος πάντα αποτυγχάνει να βρει αντιπαράδειγμα και εξαντλεί τη σουίτα ή τον προϋπολογισμό εκμάθησης. Εφόσον το ζητούμενο είναι η ταχύτητα εντοπισμού αντιπαραδειγμάτων, η παράλειψη αυτού του γύρου δίνει πιο σαφή εικόνα απόδοσης.

Μοντέλα του Benchmark

Το hardness score των Aichernig, Tappler, and Wallner [3] χρησιμοποιήθηκε ως δείκτης δυσκολίας των μοντέλων.

Στο Σχήμα 4 φαίνεται το hardness score όλων των μοντέλων benchmark, με χρωματική διάκριση ανά πρωτόκολλο. Στη συλλογή αυτή, τα μοντέλα TLS φαίνεται να είναι εύκολα, ενώ τα TCP είναι κυρίως δύσκολα. Τα πρωτόκολλα MQTT και DTLS παρουσιάζουν διαβαθμίσεις δυσκολίας, με τα μοντέλα DTLS γενικά πιο δύσκολα.



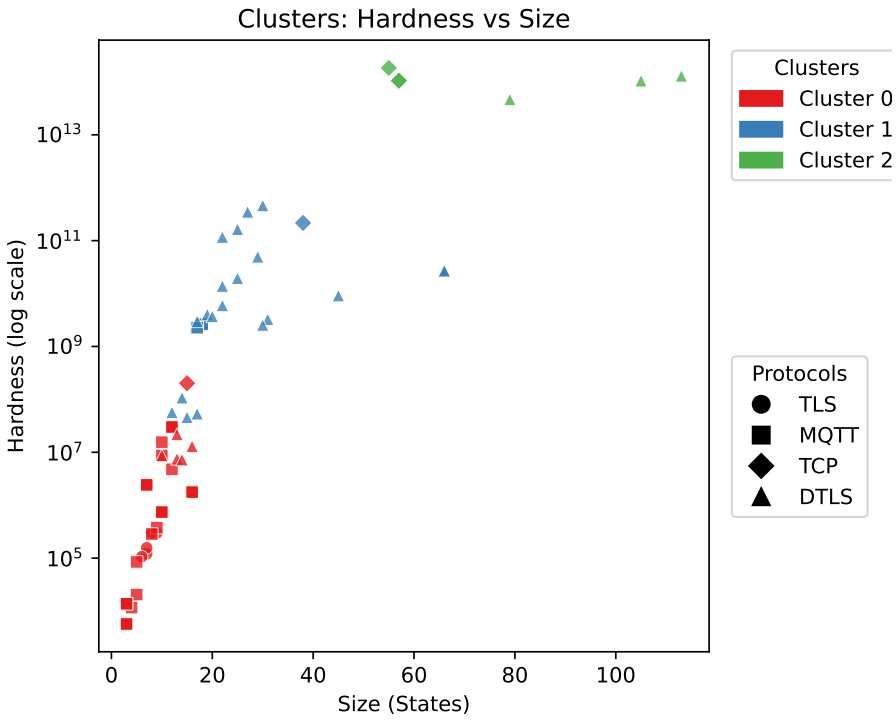
Σχήμα 4: *Barplot του hardness score για όλα τα μοντέλα στο σύνολο benchmark*

Επιπλέον, πραγματοποιήθηκε ανάλυση συσταδοποίησης λαμβάνοντας υπόψη το hardness score, τον αριθμό καταστάσεων, το μέγεθος του αλφαβήτου εισόδου, το μεγαλύτερο πρόθεμα και την μεγαλύτερη κατάληξη. Στο Σχήμα 5 φαίνεται η σχέση μεταξύ hardness score και αριθμού καταστάσεων, με χρώματα ανά συστάδα και σήμανση ανά πρωτόκολλο. Τα μοντέλα χωρίζονται σε τρεις συστάδες, συνοψισμένες στον πίνακα 4.1 με τις μεσαίες τιμές κάθε χαρακτηριστικού.

Ομάδα 0 Τα ευκολότερα μοντέλα με έως 20 καταστάσεις. Αποτελείται κυρίως από MQTT και TLS, με μερικά DTLS.

Ομάδα 1 Μέτρια δύσκολα μοντέλα με 20–60 καταστάσεις (ένας εξ αυτών κοντά στα 80). Περιλαμβάνει κυρίως DTLS, και δύο εξαιρέσεις από TCP και MQTT.

Ομάδα 2 Τα δυσκολότερα μοντέλα, με 60–120 καταστάσεις. Περιλαμβάνει τα πιο δύσκολα μοντέλα των TCP και DTLS.



Σχήμα 5: *Ανάλυση clustering των μοντέλων*

Γενικά, το μέγεθος του αλφαβήτου δεν διαφέρει σημαντικά μεταξύ των συστάδων. το hardness score και το μέγεθος καθορίζουν την κατάταξη – τα μικρότερα μοντέλα είναι ευκολότερα, τα μεγαλύτερα δυσκολότερα. Το μήκος προθέματος ακολουθεί την ίδια τάση, διότι τα μεγαλύτερα μοντέλα φυσιολογικά απαιτούν μεγαλύτερες ακολουθίες πρόσβασης. Τέλος, τα εύκολα μοντέλα έχουν πολύ κοντές καταλήξεις, ενώ οι καταλήξεις μεγαλώνουν με την αύξηση του μεγέθους του μοντέλου. Αυτές οι τάσεις συμφωνούν με τον ορισμό του hardness score.

Αλγόριθμοι

Για να εκτιμηθεί η επίδραση της προτεραιοποίησης νέων καταστάσεων στην έναρξη του ερωτήματος ισοδυναμίας, χρησιμοποιήθηκαν μερικά καθιερωμένα μαντεία ισοδυναμίας ως βάση σύγκρισης. Υλοποιήθηκαν παραλλαγές τους και μετρήθηκε η απόδοσή τους χρησιμοποιώντας τους παραπάνω δείκτες.

Stochastic State Coverage

Το μαντείο Stochastic State Coverage είναι παραλλαγή του μαντείου State Coverage της AALpy. Βασική ιδέα είναι να ξεκινούν τυχαίες διαδρομές μέχρι μήκος max από κάθε κατάσταση. Στην αρχική έκδοση, οι καταστάσεις επιλέγονται τυχαία. Το παραλλαγμένο μαντείο ταξινομεί τις καταστάσεις σε ομάδες ηλικίας βάσει του γύρου εκμάθησης στον οποίο προστέθηκαν και ορίζει κατανομή πιθανοτήτων πάνω στις ομάδες. Η κατάσταση για τη διαδρομή επιλέγεται με δειγματοληψία πρώτα της κατανομής και έπειτα ομοιόμορφα μέσα από την

επιλεγμένη ομάδα. Έτσι, οι πιθανότητες πρόσβασης σε ηλικιακές ομάδες προσαρμόζονται κατά βούληση μέσω της κατανομής.

Για παράδειγμα, αν θέλουμε η n-οστή ηλικιακή ομάδα να έχει πιθανότητα $n \cdot p$, με $p \in [0, 1]$, τότε:

$$\begin{aligned} p_1 &= p \\ p_{n+1} &= p_n + p \end{aligned} \tag{11}$$

Λύνοντας για p:

$$\begin{aligned} \sum_{n=1}^{N} n \cdot p &= 1 \\ p &= \frac{1}{\sum_{n=1}^{N} n} \\ p &= \frac{2}{N(N+1)} \end{aligned} \tag{12}$$

και έτσι η πιθανότητα της ομάδας n είναι $n \cdot p$.

Με την ίδια λογική δοκιμάστηκαν τρεις κατανομές:

- Γραμμική: $n \cdot p$, με $p = \frac{2}{N(N+1)}$
- Τετραγωνική: $n^2 \cdot p$, με $p = \frac{6}{N(N+1)(2N+1)}$
- Εκθετική: $2^n \cdot p$, με $p = \frac{1}{2^N - 1}$

Προϋπολογισμός Εκμάθησης

Για να εκτιμηθεί ο απαιτούμενος προϋπολογισμός εκμάθησης , εκτελέστηκαν πειράματα με το τυχαίο state coverage μαντείο, ξεκινώντας από μικρό προϋπολογισμό και αυξάνοντάς τον διπλασιαστικά μέχρι κάθε μοντέλο από το ίδιο πρωτόκολλο να μαθευτεί με εμπιστοσύνη (30 συνεχείς φορές). Όταν βρέθηκε άνω όριο, έγινε διχοτόμος αναζήτηση με τα ίδια κριτήρια.

Με αυτόν τον τρόπο βρέθηκαν τα εξής:

- Τα μοντέλα TLS μάθευτηκαν χωρίς καθόλου ερωτήματα συμμετοχής, καθώς η αρχική υπόθεση ήταν σωστή.
- Τα μοντέλα MQTT μάθευτηκαν με το πολύ 10000 ερωτήματα συμμετοχής σε κάθε γύρο.
- Τα TCP (σχετικά λίγα, μόνο 5) παρουσίασαν διακύμανση, ορισμένα απαιτούσαν ≤ 50000 ερωτήματα, άλλα έως 2200000.
- Τα DTLS επίσης παρουσίασαν διακύμανση: κάποια έως 500000 ερωτήματα, άλλα έως 70000.

Reversed W / Wp Method

Η Reversed W / Wp Method είναι απλή παραλλαγή των αντίστοιχων μεθόδων, με στόχο να διερευνηθεί αν υπάρχει όφελος στην προτεραιότητα πρόσβασης νέων καταστάσεων. Η

σουίτα W παράγεται ως το καρτεσιανό γινόμενο του prefix set, των συμβολοσειρών μέχρι συγκεκριμένο μήκος και του characterization set. Η Wp δημιουργεί δύο διαφορετικές σουίτες, αλλά η ίδια λογική εφαρμόζεται στην πρώτη. Προγραμματιστικά, το prefix set κατασκευάζεται με την προσθήκη καταστάσεων με σειρά παλιά→νέα. Η Reversed W Method το φτιάχνει αντίστροφα, νέα→παλιά. Μπορεί παραμετροποιηθεί με *diff_depth*, που ορίζει πόσα group να αντιστραφούν. Για παράδειγμα, *diff_depth* = 3 βάζει τις 3 τελευταίες ομάδες πρώτες. Οι δοκιμασμένες παραλλαγές είχαν *diff_depth* = $1, 2, 3, 6, \text{full}$ και ονομάστηκαν Reversed1...Reversed6 και Reversed. Η παράμετρος exploration τέθηκε σε 2, για να εξασφαλιστεί 2-πληρότητα.

Στοχαστική τυχαία Wp Method

Η Random Wp Method είναι παραλλαγή της Wp Method. Επιλέγει τυχαίο πρόθεμα για να προσπελάσει κατάσταση, μετά εκτελεί τυχαίο περπάτημα από αυτή, και τέλος επιλέγει ακολουθία διάκρισης είτε από το τοπικό σύνολο που διακρίνει την κατάσταση είτε από το καθολικό σύνολο.

Η Stochastic Random Wp Method προσθέτει στο προηγούμενο τη λογική ηλικιακής προτεραιότητας: οι νέες καταστάσεις έχουν μεγαλύτερη πιθανότητα επιλογής, διαχωρίζοντας τις σε ηλικιακές ομάδες. Δοκιμάστηκαν οι ίδιες παραλλαγές με την Stochastic State Coverage.

Αποτελέσματα

Ο Πίνακας 5.8 συνοψίζει τα αποτελέσματα των παραλλαγών του Stochastic State Coverage, παρουσιάζοντας τα περισσότερα και λιγότερα επιπλέον ερωτήματα, καθώς και τα λιγότερα και περισσότερα αποθηκευμένα ερωτήματα σε σχέση με την παραλλαγή “Random”. Το σύμβολο ■ δηλώνει ότι η μετρική δεν είναι εφαρμόσιμη. Για παράδειγμα, οι παραλλαγές του μαντείου Stochastic State Coverage στα πειράματα με το πρωτόκολλο MQTT δεν προκάλεσαν επιπλέον ερωτήματα, επομένως χρησιμοποιείται το σύμβολο ■ επειδή η μετρική δεν υπολογίστηκε.

Τα αποτελέσματα είναι μικτά. Ορισμένες ομάδες μοντέλων, όπως τα μοντέλα του πρωτοκόλλου MQTT και τα μοντέλα DTLS με μεγέθη 60–120, φαίνεται να επωφελούνται από τις παραλλαγές του Stochastic State Coverage, ενώ άλλες, όπως τα μοντέλα TCP και DTLS με μεγέθη 10–50, όχι.

Ο Πίνακας 5.9 συνοψίζει τα αποτελέσματα των παραλλαγών της μεθόδου WMethod, παρουσιάζοντας τα περισσότερα και λιγότερα επιπλέον ερωτήματα, καθώς και τα λιγότερα και περισσότερα αποθηκευμένα ερωτήματα σε σχέση με την παραλλαγή “Normal”. Η απόδοση όλων των παραλλαγών φαίνεται παρόμοια, με μικρές αποκλίσεις. Στις περισσότερες ομάδες μοντέλων, η παραλλαγή “Normal” υπερτερεί σε σχέση με τις παραλλαγές “Reversed”, με εξαίρεση το πρωτόκολλο MQTT, όπου οι παραλλαγές “Reversed” έχουν καλύτερη απόδοση.

Ο Πίνακας 5.10 συνοψίζει τα αποτελέσματα των παραλλαγών της μεθόδου WpMethod, παρουσιάζοντας τα περισσότερα και λιγότερα επιπλέον ερωτήματα, καθώς και τα λιγότερα και περισσότερα αποθηκευμένα ερωτήματα σε σχέση με την παραλλαγή “Normal”. Αν και η απόδοση των παραλλαγών είναι παρόμοια, οι παραλλαγές “Reversed” υπερτερούν στην ομάδα των μοντέλων MQTT και στα μοντέλα DTLS με μεγέθη 60–80, χωρίς να υστερούν σημαντικά στις υπόλοιπες ομάδες.

Ο Πίνακας 5.17 συνοψίζει τα αποτελέσματα των παραλλαγών της μεθόδου Random WpMethod, παρουσιάζοντας τα περισσότερα και λιγότερα επιπλέον ερωτήματα, καθώς και τα λιγότερα και περισσότερα αποθηκευμένα ερωτήματα σε σχέση με την παραλλαγή “Normal”. Οι περισσότερες παραλλαγές έχουν σαφώς χειρότερη απόδοση σε σχέση με την παραλλαγή “Normal”. Τα πειράματα με το MQTT είναι τα μόνα που αποδίδουν σταθερά καλύτερα από την παραλλαγή “Normal”.

Τα αποτελέσματα των πειραμάτων δείχνουν ότι η εστίαση στα νέα καταστάσεις μιας υπόθεσης κατά την αναζήτηση αντιπαραδειγμάτων μπορεί να οδηγήσει σε καλύτερη απόδοση σε ορισμένες περιπτώσεις, εάν γίνεται με μέτρο, αλλά μπορεί επίσης να οδηγήσει σε χειρότερη απόδοση εάν γίνεται υπερβολικά επιθετικά. Απαιτούνται πιο εξελιγμένες προσεγγίσεις για την ιδέα αυτή, και είναι απαραίτητη η επιθεώρηση των μοντέλων ώστε να κατανοηθεί γιατί οι αλγόριθμοι αποδίδουν όπως αποδίδουν.

Μελλοντική Δουλειά

Η μελλοντική δουλειά θα μπορούσε να ακολουθήσει πολλές κατευθύνσεις. Η μέθοδος Stochastic State Coverage θα μπορούσε να δοκιμαστεί με μια ευρύτερη ποικιλία συναρτήσεων πιθανότητας , ή ακόμη και με μια συνάρτηση πιθανότητας που μαθαίνεται κατά τη διάρκεια της μάθησης, παρομοίως με την προσέγγιση Mixture of Experts [18].

Οι υλοποιήσεις της μεθόδου W/Wp και των παραλλαγών της εξαρτώνται πολύ από τον τρόπο με τον οποίο κατασκευάζονται τα τοπικά και τα καθολικά σύνολα χαρακτηρισμού. Θα είχε ενδιαφέρον να δούμε πώς αποδίδουν οι αλγόριθμοι και οι παραλλαγές τους με διάφορα σύνολα χαρακτηρισμού.

Μια άλλη ιδέα θα ήταν να δοκιμάζονται νεότερες καταστάσεις, αλλά να γίνεται αυτό υπό κάποια συνθήκη ανάλογα με τις ιδιότητες του γράφου υποθέσεων. Για παράδειγμα, διαπιστώθηκε κατά τη διάρκεια των πειραμάτων ότι εάν μια κατάσταση sink έμπαινε στην υπόθεση, τότε παρέμενε για το υπόλοιπο του πειράματος εκμάθησης. Επομένως, θα είχε νόημα να προσπαθήσουμε να παραλείψουμε τα ερωτήματα που στοχεύουν σε αυτήν.

Τα ίδια συγκριτικά αποτελέσματα θα μπορούσαν επίσης να εκτελεστούν με διαφορετικούς αλγορίθμους μάθησης από τον L^* ή με διαφορετικές μεθόδους επεξεργασίας αντιπαραδειγμάτων. Επιπλέον, θα μπορούσαν να προταθούν μαντεία ισοδυναμίας τα οποία δεν είναι πλήρως αποσυνδεδεμένα από τον αλγόριθμο μάθησης αλλά τον γνωρίζουν και χρησιμοποιούν τα χαρακτηριστικά του για να βρίσκουν αντιπαραδείγματα γρηγορότερα.

Τέλος, στο πνεύμα της διάδοσης της γνώσης από προηγούμενους γύρους μάθησης σε μελλοντικούς, θα ήταν ενδιαφέρον να αποδειχθούν, ή να θεωρηθούν αποδεδειγμένες μετά από κάποιους γύρους, ιδιότητες του γράφου υπόθεσης και να χρησιμοποιηθούν για να καθοδηγήσουν τη διαδικασία μάθησης.

Chapter 1

Introduction

Model learning [29] is the field of computer science that deals with constructing models of systems by observing their behavior. Active Automata Learning (AAL) in particular treats the System Under Learning (SUL) as a black box and queries it with input sequences, guiding the learning process. The final goal of active automata learning is to construct a finite state machine (FSM) that is equivalent to the SUL. Different algorithms have been proposed depending on the type of FSM that is chosen for the final model. This work is concerned with algorithms that construct Mealy machines.

Active automata learning algorithms operate on the Minimally Adequate Teacher (MAT) framework, proposed by Angluin [4]. The MAT is an oracle that can answer equivalence queries, that is, whether a given automaton is equivalent to the SUL. If the automaton is equivalent to the SUL, the teacher is supposed to answer "yes", whereas if it isn't, it is supposed to provide some input sequence that differentiates that automaton from the SUL, called the counterexample.

That way, the learning process can be divided into two distinct parts: the hypothesis construction part, which is handled by the learning algorithm, and the hypothesis validation part (equivalence query), which is handled by the teacher. In practice, there doesn't exist a teacher that has perfect knowledge of the SUL, so the teacher is approximated by conformance testing methods.

Conformance testing methods attempt to prove equivalence of the hypothesis to the SUL by constructing a test suite, which is a set of inputs for the system, and querying both the hypothesis and the SUL with each test case of the test suite. If no discrepancy is found between the two systems, then they are considered equivalent, whereas if one discrepancy is found, then the test case that caused it is the counterexample.

Model learning research focuses, amongst other things, on minimizing the total number of queries to the SUL, both during the hypothesis construction part as well as during the hypothesis validation part. However, it has been shown [13, 1] that the number of queries during hypothesis validation is the dominant factor in the total number of queries sent to the SUL. Therefore, focus has shifted into optimizing the conformance testing algorithms.

An implication of the MAT framework is that all learning experiments operate in rounds of hypothesis construction and hypothesis validation. As will be shown in the coming chapters, most conformance testing algorithms treat each hypothesis as an independent

object and do not reuse information from previous learning rounds. This thesis investigates how modifying some of these popular algorithms to incorporate information across rounds affects their query efficiency, particularly in terms of the number of queries sent to the SUL.

Overview The next chapter presents existing conformance testing methods used for validating hypotheses in active automata learning, compares these methods with respect to their key characteristics, and introduces our own proposal for improving them. In Chapter 3, we turn our attention to the topic of benchmarking, highlighting the motivations behind it as well as the challenges that it poses. Chapter 4) details the experimental setup, describing the models used in the learning process, the conformance testing methods –—both existing and newly proposed—– that were evaluated, and the metrics employed for benchmarking. Finally, in Chapter 5, we present and analyze the experimental results, organizing them by conformance testing method and model group. The thesis concludes with a chapter discussing potential directions for future work.

Chapter 2

Hypothesis Validation

The aim of this section is to showcase the various methods that have been employed for implementing the hypothesis validation (equivalence query) part of active learning algorithms and to highlight some of their characteristics and differences in the approaches.

2.1 Preliminaries

A Mealy machine is a tuple $M = (S, S_0, I, O, \delta, \lambda)$, where S is the set of states, S_0 is the initial state, I is the input alphabet, O is the output alphabet, $\delta : S \times I \to S$ is the transition function and $\lambda : S \times I \to O$ is the output function. The transition function maps a state and an input symbol to the next state, while the output function maps a state and an input symbol to the output symbol. The set of all possible input sequences is denoted by I^* and the set of all possible output sequences is denoted by O^*.

2.2 Random Testing

Random testing involves sampling random sequences from I^*, where I is the input alphabet, and using them as test cases. Random testing includes *random walks*, where the length of the input sequences is geometrically distributed and *random words*, where the length of the input sequences is uniformy distributed in some predefined range.

2.3 Transition Coverage / Mutation-Based Selection

Transition Coverage [2] conformance testing works by generating a large test suite by starting with a random walk through the hypothesis model and repeatedly choosing a transition of the model and attempting to reach it from the current state. Finally, another random walk is executed. Having generated the test suite, a subset of the total test cases is chosen greedily, depending on the transition coverage that each test case provides.

The authors of this method actually generalize it, by noticing that transition coverage is not the only kind of coverage that can be defined and that different coverage criteria can be used by using *mutation operators*. Mutation operators take a Mealy machine as input and produce a set of mutants as output. The test cases that are chosen for the final

test suite are those that achieve the highest coverage; that is, those that distinguish the most mutants.

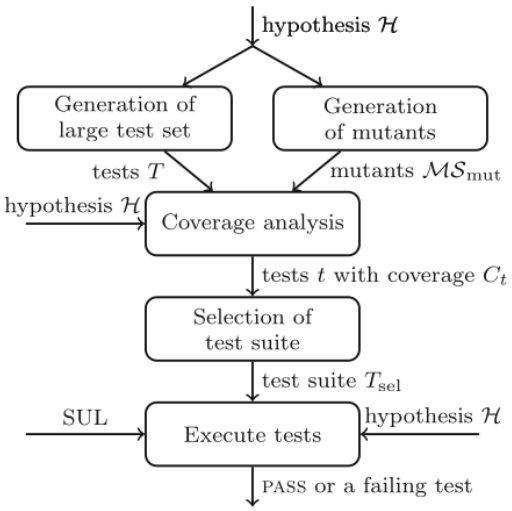


Figure 2.1: *Flow diagram of mutation-based selection [2].*

It is easy to verify that mutation operators are more general than transition coverage. A mutation operator can be defined that alters the output of a specific transition. Executing a transition can identify at most one mutant. Therefore, selecting test cases based on this mutation operator is equivalent to selecting based on transition coverage.

A merit of this method is that it can use domain specific knowledge, that is, specific mutation operators, to search for counterexamples more efficiently.

2.4 The W-method

The W-method [5] utilizes two sets: the W-set, called a *characterization set*, that is a set of input sequences that distinguish between any two pairs of states in the SUL , and the P-set, that is a transition cover set of S. It assumes an upper bound m on the number of states n in the SUL and defines the set $I[m-n] = \bigcup_0^{m-n} I$, where I is the input alphabet. The test suite that the method constructs is the set of input sequences $P \cdot Z = P \cdot I[m-n] \cdot W$, where $\cdot$ is the concatenation operator.

In essence, the inputs that belong to the transition cover set P are used so that every edge is taken at least once, and, for the state where each edge leads, the inputs that belong to the set Z are used to distinguish between the states.

The W-method and its variants prove equivalence up to the bound m. This means that, if the hypothesis does not fail the test suite, then either it is equivalent to the SUL, or the SUL has strictly more than m states.

The W-method is a prime example of what are called *Access-Step-Identify* algorithms. These are conformance testing algorithms whose operation may be divided into three parts: the *access* part, where a target state is accessed, the *step* part, where some input sequence is sent in order to reach a new state, and the *identify* part, where a distinguish-

ing sequence is sent in order to identify the current state. It has been observed that the only way to improve the algorithms that fall under this category is to improve the *identify* part, which is based on the distinguishing sequences that are used [7].

2.5 The partial W-method (Wp-method)

The partial W-method (Wp-method) [16] is an improvement to the W-method. Instead of utilizing the coarse characterization set W, the Wp-method makes use of finer identification sets W_i for every state s_i in the SUL. It starts by checking that the states of the SUL are identifiable in the hypothesis, at the same time constructing the identification sets W_i. These can later be used to more efficiently check for the equivalence of the transitions of the hypothesis with the transitions of the SUL.

There is also a *randomized* variant of this method [15], where a random subset of the test suite is executed. This has been observed to be faster at finding counterexamples [3] [13], but it does not prove equivalence as the deterministic methods do.

2.6 Harmonized State Identifiers (HSI)

The HSI method [20], like the W and Wp methods, uses the notion of the characterization set but also finds that it is not necessary to use the whole set. Instead, it uses subsets of this set called *harmonized state identification sets*. These sets have the property that, if a separating sequence for two states exists, then this sequence is a prefix of both of the harmonized state identification sets of the two states. The merits of this method is that it is applicable to partially defined non-deterministic state machines and it tends to produce smaller test suites than the W and Wp-methods.

2.7 Preset Distinguishing Sequences (PDS)

Preset distinguishing sequences are input sequences that can be used to identify the current state of the system from all others. Essentially, the most minimal characterization set W possible consists of one preset distinguishing sequence, since this sequence alone can differentiate every state from every other state. Unfortunately, this sequence may not exist for every FSM and, as a result, techniques solely based on the existence of PDS cannot be universally used.

2.8 Adaptive Distinguishing Sequences (ADS)

An adaptive distinguishing sequence [19] is a decision tree whose aim is to identify the current state of the system. Structurally, it is a tree whose nodes are inputs that correspond to a set of possible states and whose edges are outputs observed as responses to the input . It differs from preset distinguishing sequences in that it observes the output of the current query and correspondingly decides what the next symbol of the sequence will be, instead of the sequence being determined from the beginning. Its merit is the fact

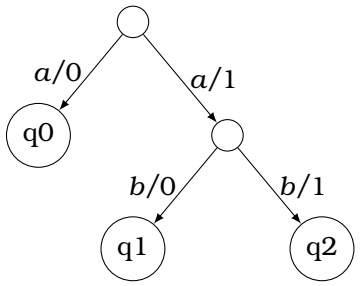


Figure 2.2: *Simple ADS showing how, with input "a" and output "0", the state q_0 is identified, whereas with inputs "ab" and outputs "10" and "11" respectively, the states q_1 and q_2 are identified.*

that, an FSM that does not admit a PDS may admit an ADS. However, an ADS still may not exist for some FSMs, deeming it not universally usable.

For the sake of clarity, it should be mentioned that ADSs have also been used in the learning part of AAL algorithms [12].

2.9 Hybrid ADS (H-ADS)

Hybrid adaptive distinguishing sequences (H-ADS) [24] improve on ADSs by being universally usable. An incomplete ADS is found[1], whose leaves may also be sets of states instead of singleton sets. This sequence is later augmented with pairwise separating sequences in order to distinguish between the potential states.

It is noteworthy that the authors also make use of what they name "subalphabets". In an effort to find peculiar counterexamples faster, they generate test cases using the subalphabet given by the previous hypothesis' counterexample first, and then generate test cases using the whole alphabet. This is the first instance of knowledge sharing between previous and current hypothesis validation.

2.10 Incomplete Adaptive Distinguishing Sequences (I-ADS)

Incomplete adaptive distinguishing sequences [14] also aim to rectify the shortcomings of ADS. I-ADSs are a set of ADSs that can be used when an ADS does not exist for the given system. Essentially, I-ADSs have the same structure as an ADS, but the leaf nodes are not required to be singleton states; they may also be sets of states. So, when, in the process of distinguishing a state, such a leaf node is encountered, a reset is sent to the system and an access sequence such that the system is led to a set of possible initial states that can be distinguished. In that way, multiple I-ADSs can be utilized as an ADS but with resets and extra access sequences required.

[1] not to be confused with the technique 2.10 that shares the same name.

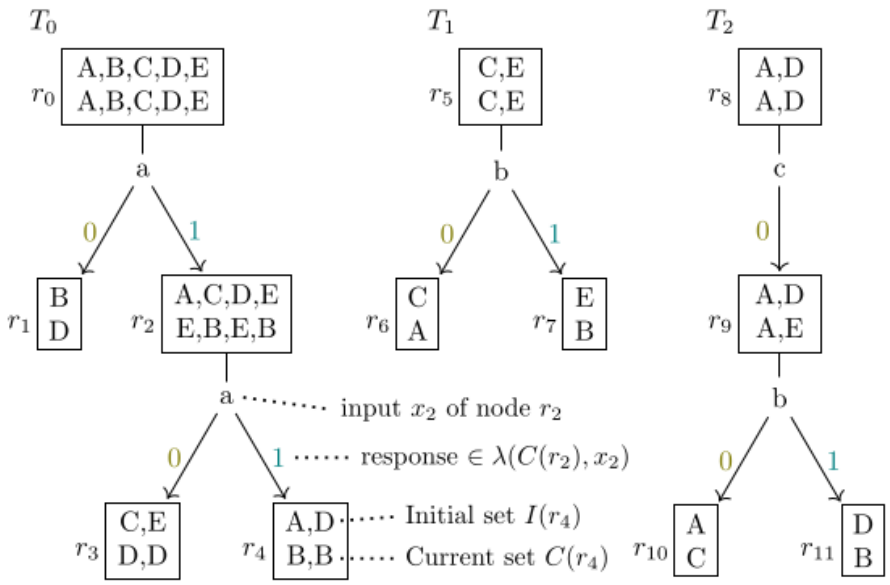


Figure 2.3: *Three different I-ADS that combined make up a fully distinguishing set of a hypothetical automaton with five states, by Soucha and Bogdanov [27]*

2.11 H

The H-method [7] [8] differentiates itself from other methods by avoiding fixed distinguishing sequences. The authors leverage the fact that every m-complete test suite has a distinguishing sequence for each transition of the system. They argue that every complete test suite *TS* contains the set $VPref(X^{m-n+1})$, where V is a prefix-closed state cover of the specification and X is the input alphabet, and that, if the hypothesis can be distinguished from the specification by an input sequence that is longer than what the test suite allows, then the test suite surely contains at least two sequences that are prefixes of the former, lead the hypothesis to the same state t but lead the specification to two different states s_1, s_2. As a result, if a sequence distinguishes s_1 from s_2, then it distinguishes t from s_1, s_2. Based on this insight, they propose an algorithm that constructs an m-complete test suite with this kind of distinguishing sequences, whose test cases tend to be shorter than those of the traditional W method.

2.12 SPY

The SPY method [23] attempts to reduce test branching. Test branching occurs when different test cases in the test suit share a common prefix. When this happens, the test suite contains some redundancy, in the sense that the input that consists the common prefix has to be reused multiple times. The authors of the method realize that, the only way to further reduce the length of a test suite, if the transfer sequences are already shortened and the shortest distinguishing sequences are used, is to reduce the number of branches in the test suite. That is, reduce the number of times a common prefix is used. They establish the conditions which must be satisfied to reduce test branches while

also retaining m-completeness and develop a test suite generation method that combines short branches into longer test sequences and finally results in shorter test suites.

2.13 SPY-H

The SPY-H method [26] attempts to combine the merits of the SPY method and the H method in order to construct even smaller test suites. It aims to satisfy the same conditions that guarantee m-completeness as the SPY method, but it verifies the reached state by using distinguishing sequences like those used by the H method.

2.14 Mixture of Experts

The Mixture of Experts method [18] attempts to speed up the search for counterexamples by first searching in smaller test suites that are more likely to contain them, drawn from previous rounds of equivalence queries. These smaller test suites are generated by "experts", functions that take a word and a Mealy machine as input and generate sets of subalphabets. The method is embeddable to any Access-Step-Identify algorithm (like the ones described above). It employs a mixture of these experts and uses a reinforcement learning algorithm to dynamically adjust the probability of searching for counterexamples in each expert's subalphabets with each test case. These probabilities are carried on to next rounds of equivalence queries, making it the first method, to the author's knowledge, to systematically make use of cross-equivalence query information.

2.15 AALpy Equivalence Oracles

AALpy [21] is an active automata learning python library that implements a number of equivalence query algorithms. The set of implemented algorithms consists both of methods like the ones described previously, which have been evaluated in separate publications, as well as heuristic methods that haven't been extensively analyzed. The authors provide summaries for them in but they do not benchmark them.

2.15.1 Transition Focus Oracle

The Transition Focus oracle is a random walk based oracle that is parametrized by a parameter ϵ, so that each input loops to the same state with probability ϵ and leads to a new state with probability $1 - \epsilon$.

2.15.2 Probably Approximately Correct (PAC) Oracle

The PAC oracle is a random word based oracle. It is parametrized by parameters ϵ and δ, which define the number of test cases during conformance testing, such that it guarantees that the learned model is an ϵ-approximation of the SUL with a probability of at least $1 - \delta$.

2.15.3 Cache Based Oracle

The Cache Based oracle utilizes the query caching that is AALpy implements for SULs, so as to avoid duplicate queries for prefixes with observed traces. The membership query results are encoded in a tree structure which is updated during hypothesis construction and equivalence checking. Each test case that the oracle poses is a path from the tree root to one of the leaves, with a random walk of specific lenght appended at the end.

2.15.4 K-Way Transition Coverage Oracle

The K-Way Transition Coverage oracle is similar to the Transition Coverage oracle discussed above. It generates a number of random walks and greedily selects a subset of them so that the k-way transition coverage is optimized. The parameter k refers to the number of steps between the start and the end of a transition.

2.15.5 K-Way State Coverage Oracle

The K-Way State Coverage oracle runs a test case for every k-combination or k-permutation of states and explores the surroundings of the final state via a random walk.

2.15.6 State Prefix Oracle

The State Prefix oracle accesses each state using its access sequence and explores its surroundings by initiating a number of random walks withing a certain length range. It can also be parametrized so that it first explores the new states of the hypothesis, making this method the second one that has been observed to utilize information from previous learning rounds.

2.16 Discussion

The table 2.1 summarizes the above methods based on some of their most notable characteristics. The columns of the table are named after the characteristic and the rows are named after each method. Checkmarks and black boxes are used to denote whether the method has the characteristic or not, respectively.

The plethora of blackboxes in the “Cross-EQ” column shows that most methods work on a per-hypothesis basis and attempt to reduce the length of the generated test suite by experimenting with different ways of searching for and using state identifiers and distinguishing sequences.

One slight exception to this observation is the H-ADS method that also makes use of the previous hypothesis’ counterexample to search for a new one more efficiently, although this technique seems to have been employed as a hack rather than an integral part of the method.

Another exception is AALpy’s State Prefix oracle, which can be set to first explore the hypothesis’ new states.

Method	Universal	ASI	M-complete	Domain Specific Knowledge	Subalphabets	Cross-EQ
Random Walks	✓	■	■	■	■	■
Random Words	✓	■	■	■	■	■
Transition Coverage	✓	■	■	■	■	■
Mutation Selection	✓	■	■	✓	■	■
W-Method	✓	✓	✓	■	■	■
Wp-Method	✓	✓	✓	■	■	■
PDS	■	✓	✓	■	■	■
ADS	■	✓	✓	■	■	■
H-ADS	✓	✓	✓	■	✓	■
I-ADS	✓	✓	✓	■	■	■
H	✓	✓	✓	■	■	■
SPY	✓	✓	✓	■	■	■
SPY-H	✓	✓	✓	■	■	■
Mixture of Experts	✓	✓	✓	✓	✓	✓
Transition Focus	✓	■	■	■	■	■
PAC	✓	■	■	■	■	■
Cache Based	✓	■	■	■	■	■
K-Way Transition Coverage	✓	■	■	■	■	■
K-Way State Coverage	✓	■	■	■	■	■
State Prefix	✓	✓	■	■	■	✓

■	absent
✓	present

Table 2.1: *Summary of methods and their characteristics.*

Finally, there is the Mixture of Experts approach, which seems to focus even more on the idea of using information drawn from previous rounds of equivalence queries. It is also the only approach to tick every box, given that it has all merits of Access-Step-Identify algorithms and also uses ideas from other methods.

It is evident that not much work has been done on employing some kind of memory mechanism between hypotheses that could potentially make the search for counterexamples more efficient.

Chapter 3

Benchmarking Equivalence Query Methods

3.1 Introduction

Due to the lack of a true equivalence oracle in the process of active automata learning, equivalence queries are approximated using conformance testing techniques. Since the execution time of test cases dominates the learning process, it is widely acknowledged that the number and length of test cases used for conformance testing should be minimized [3]. For this reason, most conformance testing algorithms focus on generating small test suites, both in terms of the number of test cases and their individual lengths.

3.2 Empirical Evaluation

Beyond analyzing the theoretical complexity of conformance testing methods, it is crucial to evaluate their performance empirically. Such evaluations are typically carried out either on real-world systems (e.g., communication protocols or hardware controllers) or on simulated systems, using previously learned or manually constructed models as stand-ins for actual implementations. This allows for controlled experimentation and reproducibility.

While algorithmic time-complexity provides insight into the expected or worst-case performance of a method, usually with respect to the number of states n in the system, it does not fully capture the structural or behavioral complexity of the model. As a result, models must be characterized using more than just their size. A more nuanced approach to quantifying model difficulty is the *hardness score* [3], which is defined as:

$$\text{hardness score} = \frac{\text{learn-hardness}}{\text{test-chance}} \tag{3.1}$$

where the *learn-hardness* is:

$$\begin{aligned} \text{learn-hardness} &= \left(k \cdot n^2 + n \cdot \log m\right) \cdot (n + m) \\ k &= \text{length of the input alphabet} \\ n &= \text{number of states} \\ m &= |p^{max}| + |w^{max}|, \qquad \text{Approximation of the longest counterexample} \end{aligned} \tag{3.2}$$

and the *test-chance* is given by:

$$\text{test-chance} = \left(|p^{\max}| \cdot k^{|w^{\max}|}\right)^{-1} \tag{3.3}$$

where:

$$\begin{aligned} p^{\max} &= \text{argmax}_p |p| \quad &&: p \in \text{Prefixes}, && \text{Longest access sequence} \\ w^{\max} &= \text{argmax}_w |w| \quad &&: w \in W, && \text{Longest distinguishing suffix} \end{aligned} \tag{3.4}$$

As such, inspecting the hardness score of a model alongside the performance metrics of a conformance testing algorithm provides a more comprehensive understanding of the algorithm's performance.

3.3 Evaluation Metrics

To ensure fair and system-independent comparisons, most benchmarks report abstract metrics that do not rely on wall-clock execution time. Commonly used metrics include:

- the total number of test cases executed,
- the total number of input symbols across all test cases, and
- the number of resets issued to the system under learning, particularly when resets are expensive or limited [12].

In addition to these basic cost metrics, some studies propose their own measures. For instance, the *Average Percentage of Faults Detected* (APFD) [13] reflects how quickly faults are found during testing. It is defined as:

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{n} \text{TC}_i \cdot \Delta H_i}{\text{TC} \cdot n} + \frac{1}{2 \cdot \text{TC}} \tag{3.5}$$

where

$$\begin{aligned} \Delta H_i &= \text{number of new states discovered between learning rounds } i-1 \text{ and } i \\ \text{TC} &= \text{total number of symbols and resets executed in learning and testing} \\ \text{TC}_i &= \text{number of symbols and resets executed up until round } i \end{aligned} \tag{3.6}$$

Other proposals, such as the s_1 and s_2 scores [3], are not absolute measures of performance, but are used as a metric to compare algorithms to one another with respect to a set of experiments. The s_1 score is the sum of the average membership queries' steps posed during equivalence queries, over all of the learning experiments.

$$s_1(\text{comb}) = \sum_{m \in \text{Models}} \text{meanSteps}(\text{comb}, m) \tag{3.7}$$

where:

comb = the learner - tester combination

Models = the set of models under learning

meanSteps = the average number of membership queries' steps during testing

This score can be used to compare the learning budget that an equivalence oracle requires to learn a set of models. However, it doesn't provide a comprehensive view of the performance of the oracles, because it is sensitive to outliers and does not capture well the performance of the oracles on each model. For example, if an oracle has worst performance when learning a large model that requires a lot of queries to be learned, the impact of this bad performance on the s_1 score can be greater than the impact of performing better when learning a lot of smaller models. That's why the s_1 score is complemented by the s_2 score.

The s_2 score is the sum of the average membership queries posed during equivalence queries divided by the maximum of those averages recorded in all of the learning experiments.

$$s_2(\text{comb}) = \sum_{m \in \text{Models}} \frac{\text{meanSteps}(\text{comb}, m)}{\max_{a \in \text{Combos}} \text{meanSteps}(a, m)} \tag{3.8}$$

where:

Combos = the set of learner - tester combinations

Because each term of the sum is divided by the number of queries of the worst performer, this score effectively measures how close each model is to the worst performer and is not affected by the differences in order of magnitute in queries needed to learn a model. It can be used as a complement to the s_1 score to show how well an equivalence oracle performs in comparison to other equivalence oracles when learning a specific set of models.

Furthermore, it is common to introduce penalties in these metrics for experiments that fail to complete successfully or that exceed resource limits. In the case of the s_1 and s_2 scores, this is done by adding 1 to the s_2 score for each failed experiment of the algorithm being evaluated, given that this is the maximum possible value of each term in the sum.

3.4 Handling Non-Determinism

When benchmarking non-deterministic conformance testing algorithms—such as those relying on randomized exploration or input generation—it is necessary to repeat experiments multiple times to account for variability. Klees et al. [17], for example, suggest running each configuration 30 times and reporting average outcomes.

3.5 Parameterization

Benchmarks that aim to compare the performance of different conformance testing algorithms must take care to configure them in a fair and consistent manner. For example, when comparing “Random Walk” with “Random Word” strategies, one must tune their respective parameters so that the expected length of generated test cases is approximately equal. Similarly, other parameters—such as the depth of search, the expected length of test cases or the reset probability—should be controlled to ensure that any performance differences observed are attributable to the algorithms themselves rather than to differing configuration.

Chapter 4

Experimental Setup

This chapter covers the measures that are were used for the benchmarks, the models that were learned during the learning experiments and the equivalence oracles that were put to the test.

4.1 Experimental Setup

To begin with, all experiments were conducted using the AALpy library [21] and all of the oracles were implemented by extending the base oracle class of the library. Most of the methods tested were implemented as variations of popular methods like the W Method, the Wp Method and the Random Wp Method. However, the AALpy library lacked implementation of the Wp Method and the Random Wp Method. As a result, the Wp method was implemented from scratch and merged into the project, taking the opportunity to optimize the code for the W Method as well. Moreover, the LearnLib implementation of the Random Wp Method was used as a reference and ported to AALpy.

In order to evaluate the algorithms, Mealy automata from a variety of protocols were learned, specifically TLS [6, 25], MQTT [28], TCP [10], and DTLS [11, 9]. All of the learning experiments were conducted with the same learner, specifically the L^* learner with Rivest-Schapire counterexample [22] processing. Enough learning resources were used, such that the vast majority of learning experiment ends successfully for at least one oracle. The learning experiments that fail were accordingly penalized.

During the equivalence query of each learning round, the total membership queries that are posed from the conformace testing algorithm to the SUL are measured. The learning experiments of stochastic conformance testing algorithms were repeated 30 times, as proposed by Klees et al. [17]. Aside from common statistical metrics, like the expected value and the standard deviation, the s_1 and s_2 scores, proposed by Aichernig, Tappler, and Wallner [3], were used, slightly modified. Because the learner is the same for all experiments and the characterizing sets are constructed with the same method for all conformance testing algorithms, the mean number of steps was not computed but rather the average number of membership queries.

The s_1 score is the sum of the average membership queries posed during equivalence queries, over all of the learning experiments.

$$s_1(\text{alg}) = \sum_{m \in \text{Models}} \text{avg\_MQ}(\text{alg}, m) \tag{4.1}$$

The s_2 score is the sum of the average membership queries posed during equivalence queries divided by the maximum of those averages recorded in all of the learning experiments.

$$s_2(\text{alg}) = \sum_{m \in \text{Models}} \frac{\text{avg\_MQ}(\text{alg}, m)}{\max_{a \in \text{Algs}} \text{avg\_MQ}(a, m)} \tag{4.2}$$

The averages required for the computation of the scores were computed over the successful experiments. As such, the s_1 and s_2 scores reflect the performance of the oracles on successful learning experiments. In order to account for failures, a penalized version of the s_2 score was used, where the average number of failures of each algorithm is added to its score. For deterministic algorithms, the penalty is always 1 for each failure, whereas for non-deterministic algorithms, it is the number of failures divided by the number of trials.

Finally, a variant of the s_1 score was calculated ignoring the queries of the last learning round. The last learning round, by its nature, fails to find a counterexample, because no counterexample exists, and therefore exhausts the test suite or learning budget of the oracle. The only way to reduce the number of queries in the last round is to construct a smaller test suite or limit the learning budget. Given that this thesis is concerned with the faster discovery of counterexamples when one exists, excluding the last learning round queries will provide better insight to the performance of the oracles.

4.2 Benchmark Models

The hardness score proposed by Aichernig, Tappler, and Wallner [3] and presented in equation 3.1 was used as a measure of how difficult the models are to learn.

Figure 4.1 shows the hardness score of all of the models in the benchmark set, colored by protocol. In this set of benchmark models, the TLS protocol seems to contain easy models, while the TCP protocol mostly hard models. The MQTT and DTLS protocols contain models of varying difficulty, but the DTLS models seem to be harder in general.

Moreover, a cluster analysis was done on the models, taking into consideration their hardness score, number of states, length of input alphabet, longest prefix and longest suffix. Figure 4.2 shows the hardness scores of models in relation to the number of states, colored by cluster and marked by protocol. The models fall into three clusters, summarized also in table 4.1, where the median of each feature is contained.

Cluster 0 Cluster 0 is the set of the easiest models to learn that have no more than 20 states. This set of models comprises mainly the MQTT and TLS protocol, but also contains some DTLS models.

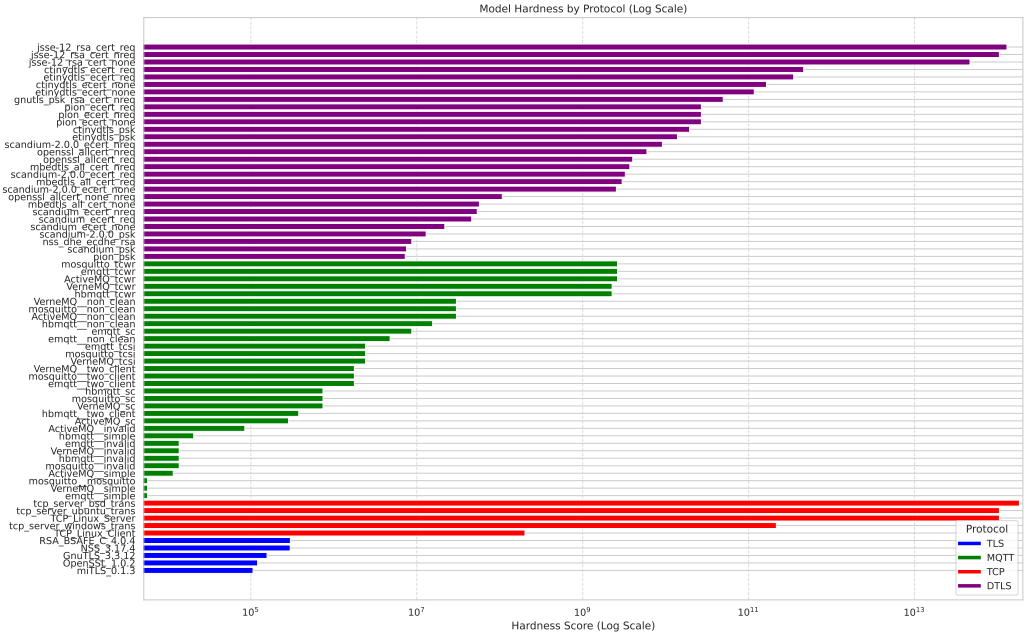


Figure 4.1: *Barplot showing the hardness score of all of the models in the benchmark set*

Table 4.1: *Cluster Summary*

Cluster	Hardness	Sizes	Inputs	Longest Prefix	Longest Suffix
0	7.413247e+05	9	9	4	1
1	3.860247e+09	22	10	7	3
2	1.054145e+14	68	11	10	6

Cluster 1 Cluster 1 is the set of modestly hard models to learn that have size between 20 and 60 states, with an outlier having close to 80. This set of models comprises mainly DTLS models and two outliers, one from the TCP and one from the MQTT protocol.

Cluster 2 Cluster 2 is the set of the hardest models to learn that have size between 60 and 120 states. This set of models contains the hardest models of the TCP and DTLS protocols.

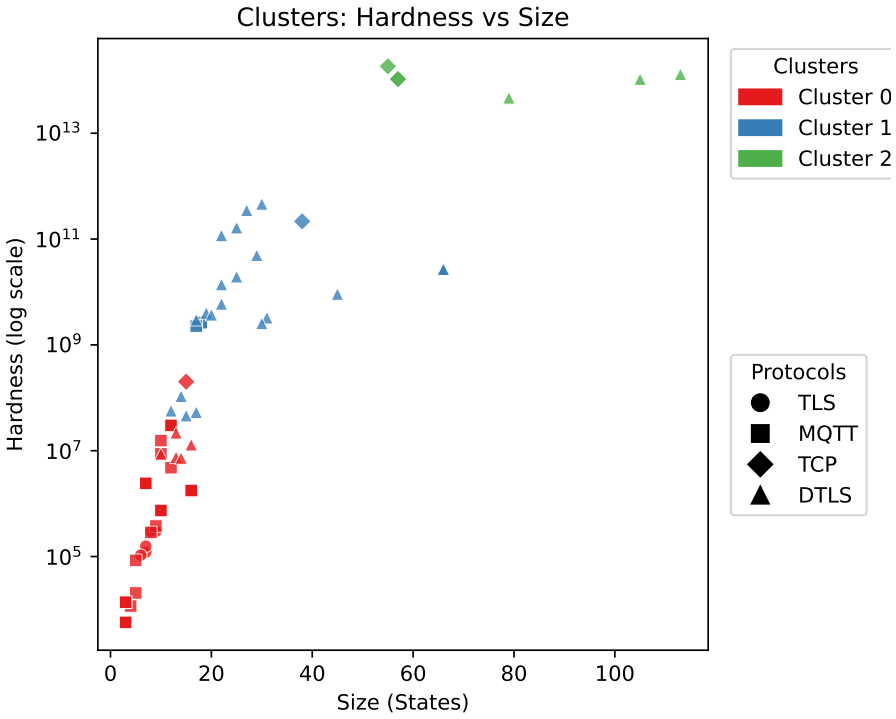


Figure 4.2: *Cluster analysis of the models*

In general, it seems that the alphabet size doesn't vary much in the clusters. The hardness score and size seem to determine the cluster, with the smallest models tending to be easier and the largest models harder. The prefix length follows the same trend, but this can be explained by the fact that larger models naturally have longer access sequences for their state. Finally, the easy models have extremely short suffixes and the suffixes seem to grow the larger a model is. These trends agree with the definition of the hardness score given in equation 3.1.

4.3 Algorithms

In order to measure the impact of starting the equivalence query by checking the new states first, some established equivalence oracles were used as baselines. Variations of those oracles were implemented and have their performance measured against them, using the scores mentioned before.

4.3.1 Stochastic State Coverage

The Stochastic State Coverage oracle is a modification to AALpy's State Coverage oracle. The basic idea of the algorithm is to start a number of random walks up to a maximum length from every state in the automaton. In the original version of the algorithm, the states are accessed randomly. The Stochastic State Coverage oracle splits the states into age groups, depending on the round of learning that they were added in, and to define a probability distribution function over these age groups. The state for the random walk is chosen by sampling the probability distribution function and then sampling uniformly the states from the age group that was chosen. That way, the priority that is given to each age group can be tuned by the probability distribution function.

For example, suppose that one desires to tune the oracle such that the i-th age group of states has probability $i \cdot p$ of being accessed, for some $p \in [0, 1]$. This can also be defined recursively as:

$$\begin{aligned} p_1 &= p \\ p_{n+1} &= p_n + p \end{aligned} \tag{4.3}$$

Then, one can solve for the probability p:

$$\begin{aligned} \sum_{n=1}^{N} n \cdot p &= 1 \\ p &= \frac{1}{\sum_{n=1}^{N} n} \\ p &= \frac{2}{N \cdot (N+1)} \end{aligned} \tag{4.4}$$

and the probability for the n-th age group is $n \cdot p$.

Based on this idea, three different versions of the Stochastic State Coverage oracle were tested, each with a different probability distribution function. The probability distribution functions that were tested are:

- Linear, where the probability of the n-th age group is $n \cdot p$, for $p = \frac{2}{N \cdot (N+1)}$
- Quadratic, where the probability of the n-th age group is $n^2 \cdot p$, for $p = \frac{6}{N \cdot (N+1) \cdot (2N+1)}$
- Exponential, where the probability of the n-th age group is $2^n \cdot p$, for $p = \frac{1}{2^N - 1}$

Learning Budget

An estimate of the required learning budget, in terms of membership queries during equivalence checking, was acquired. This was done by performing some learning experiments using the uniformly random state coverage oracle, starting from a low learning budget and doubling it until each model in the given protocol could be learned with confidence; that is 30 times in a row. Once an upper bound was found, a binary search between the upper bound and the upper bound halved was performed with the same criteria.

This way, it was found that the TLS models could be learned with no learning budget at all, since the initial hypothesis that was constructed was also correct. The MQTT models could all be learned with no more than 10000 membership queries during each round of equivalence checking. The TCP models, despite there being only 5 of them, had more variance; some of them could be learned with no more than 50000 queries whereas other required up to 2200000 queries. Finally, the DTLS models, being great in number, also showed some variance, some of them requiring up to 500000 queries, whereas others no more than 70000.

4.3.2 Reversed W / Wp Method

The Reversed W / Wp Method is a simple modification of the corresponsing methods, with the goal of providing a hint of whether there is any gain in checking the new states of a hypothesis first. The test suite that the W Method constructs is the product of the prefix set, the set of strings up to a certain length and the characterization set. The test suite that the Wp method constructs is a bit different, being comprised by two test suites that are constructed differently, but the same logic applies to its first test suite. From a programming perspective, the prefix set is constructed using the states in the order that they were added in the hypothesis automaton, old to new. The Reversed W Method constructs the prefix set in the opposite order, new to old. It can be further parameterized such that only part of the test suite is constructed from new to old, by the *diff_depth* parameter. So, for example, a *diff_depth* parameter of 3 means that the 3 latest age groups of states were put first, while the rest of the test suite retains its order. The variants that are were put to the test use parameters $diff\_depth = 1, 2, 3, 6, full$ and are respectively named Reversed1..Reversed6 and Reversed. The exploration parameter for the oracles was set to 2, so they guarantee 2-completeness.

4.3.3 Stochastic Random Wp Method

The Random Wp Method is a variant of the Wp Method. It randomly selects a prefix from the prefix set in order to access some state. Afterwards, a random walk is executed from the state and, finally, a characterizing sequence is randomly chosen either from the local set of characterizing sequences; that is the set of sequences that can distinguish the specific state that was reached after the random walk, or from the global set of characterizing sequences.

The Stochastic Random Wp Method is a variant of the Random Wp Method that, similar to the Stochastic State Coverage method, assigns higher probability of being sampled to new states of the hypothesis, by splitting them into age groups. The same variants with the Stochastic State Coverage algorithm were put to the test.

Chapter 5

Experimental Results

This chapter presents the results of the experiments that were conducted. The results are presented first by method and its variations and then by protocol.

5.1 Stochastic State Coverage Variants

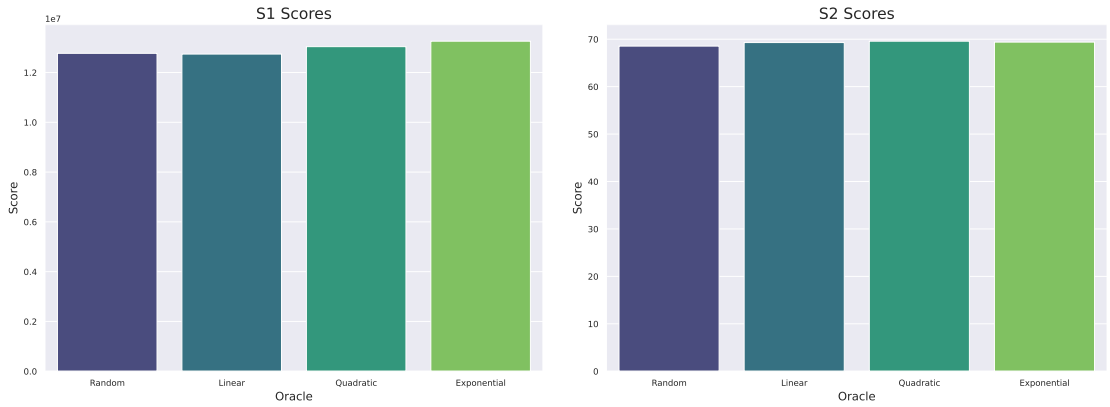


Figure 5.1: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions.*

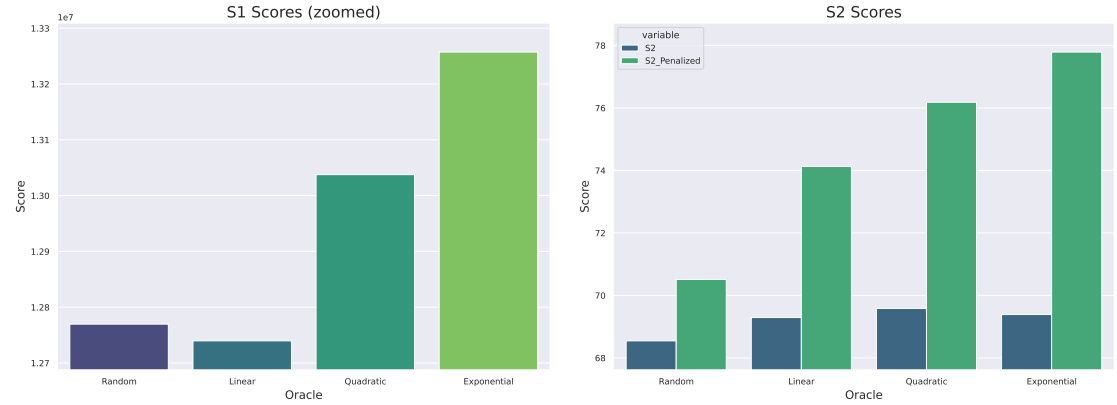


Figure 5.2: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions, zoomed in.*

The s_1 and s_2 scores of the Stochastic State Coverage oracle with the different probability distribution functions are presented in Fig. 5.1.

The scores are very similar for all of the probability distribution functions and it is not easy to make a distinction between them. The differences become more evident after zooming in, as shown in Fig. 5.2.

All of the scores seem to favour the “Random” variant, which suggests that, out of the four variants, it is the preferred one to choose when learning a wide variety of models. Specifically, the s_1 score shows that the “Random” variant is expected to perform the least number of queries and the penalized s_2 score shows that it is also the most resilient to failures. Nonetheless, the not penalized s_2 scores remain close for all variants. Thoroughly inspecting the results, reveals more information.

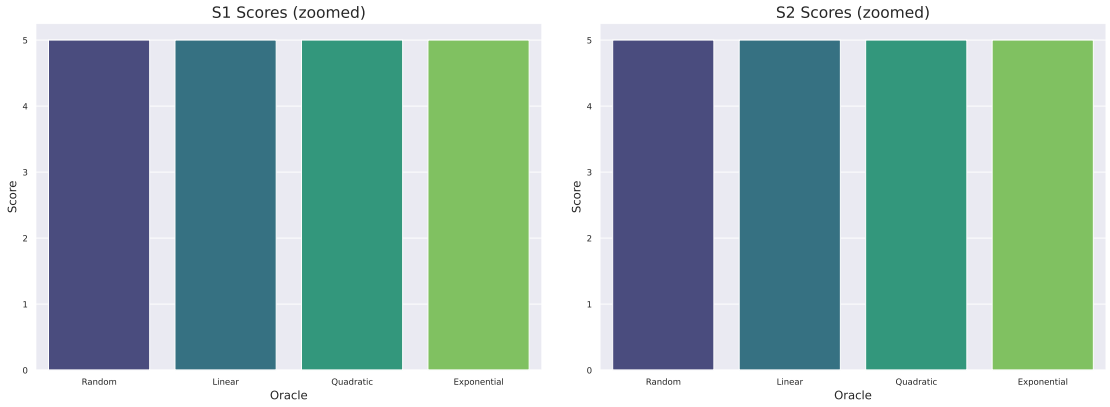


Figure 5.3: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions for TLS, zoomed in.*

TLS The TLS-specific results in Fig. 5.3 show nothing interesting because the models are so small that they are learned in the first round. Therefore, all conformance testing algorithms perform exactly the same. For the same reason, they do not actually affect the overall results. The penalized score is not shown since all of the models were learned in the first round successfully.

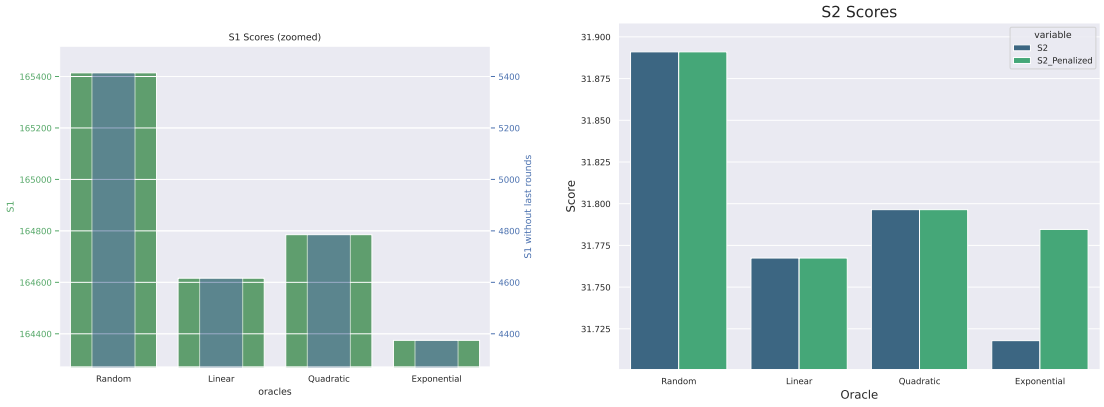


Figure 5.4: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions for MQTT, zoomed in.*

MQTT The MQTT-specific results in Fig. 5.4 show that the "Linear" and "Exponential" variant performs the best, followed by the "Quadratic" variant. The same trend shows both in the s_1 as well as the s_2 scores. These scores indicate that the variants perform the least amount of queries in general and are less likely to be the worst performers. While the percentage of saved queries during the whole learning process is of order:

$$\Delta Q = \frac{165400 - 164400}{165400} \Rightarrow \boxed{\Delta Q = \frac{1000}{165400} = 0.6\%} \tag{5.1}$$

The percentage of saved queries that could actually be saved is:

$$\Delta Q = \frac{5400 - 4400}{5400} \Rightarrow \boxed{\Delta Q = \frac{1000}{5400} = 18.5\%} \tag{5.2}$$

Table 5.1: *The number of failures for the MQTT models for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
hbmqtt__two_client_will_retain	17	0	0	0	2

Table 5.2: *MQTT models' statistics regarding number of states.*

Statistic	Value
Min	3
Median	10
Max	18

The penalized s_2 score for the "Exponential" variant deems it slightly worse than the "Linear" one. While this variant seems to be more prone to failing, it seems to have failed only twice out of 30 trials for 32 models, which isn't a lot. Table 5.1 shows the model that caused it to fail. Cross referencing this table with Table 5.2 shows that the model is among the largest ones.

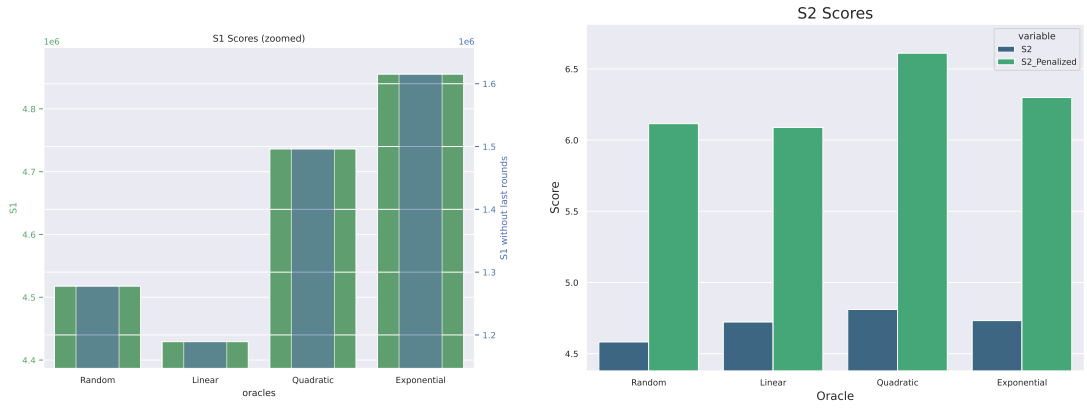


Figure 5.5: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions for TCP, zoomed in.*

TCP The TCP-specific results in Fig. 5.5 show that the "Linear" variant has the best s_1 score, while the "Random" variant has the best s_2 score, however being close to the rest of the variants. Given how the "Linear" variant has the best s_1 score, it seems to be the best performer overall.

The percentage of queries saved with respect to the "Random" variant is around:

$$\Delta Q = \frac{12.7 \times 10^5 - 11.8 \times 10^5}{12.7 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{0.9}{12.7} = 7\%} \tag{5.3}$$

Table 5.3: *The number of failures for the TCP models for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
tcp_server_windows_trans	38	10	5	15	13
TCP_Linux_Client	15	1	7	13	6
tcp_server_ubuntu_trans	57	12	12	11	13
TCP_Linux_Server	57	14	8	8	7
tcp_server_bsd_trans	55	9	9	7	8

whereas the percentage of extra queries of the worst case, that is the "Exponential" variant, with respect to the "Random" variant is around:

$$\Delta Q = \frac{16.1 \times 10^5 - 12.7 \times 10^5}{12.7 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{3.4}{12.7} = 26.7\%} \tag{5.4}$$

All of the variants seem to suffer from a considerable amount of failures, as shown in the penalized s_2 score. The details of the failures are shown in Table 5.3.

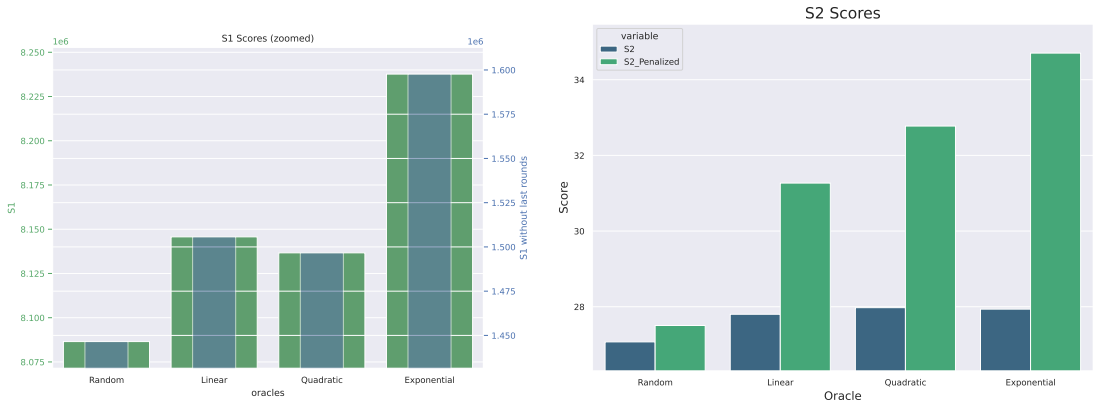


Figure 5.6: *The s_1 and s_2 scores of the Stochastic State Coverage oracle with different probability distribution functions for DTLS, zoomed in.*

DTLS Finally, the DTLS-specific results favour the "Random" variant over the other ones. The penalized s_2 score shows that the "Quadratic" and "Exponential" variants fail more often than the "Linear" and the "Random". The least and greatest amount of extra queries with respect to the "Random" variant is:

$$\Delta Q = \frac{1.5 \times 10^6 - 1.435 \times 10^6}{1.435 \times 10^6} \Rightarrow \boxed{\Delta Q = 4.5\%} \tag{5.5}$$

$$\Delta Q = \frac{1.6 \times 10^6 - 1.435 \times 10^6}{1.435 \times 10^6} \Rightarrow \boxed{\Delta Q = 11.4\%} \tag{5.6}$$

However, since the DTLS models that were used for the experiments were a lot and varied in size; that is, the number of states, they were also inspected size-wise.

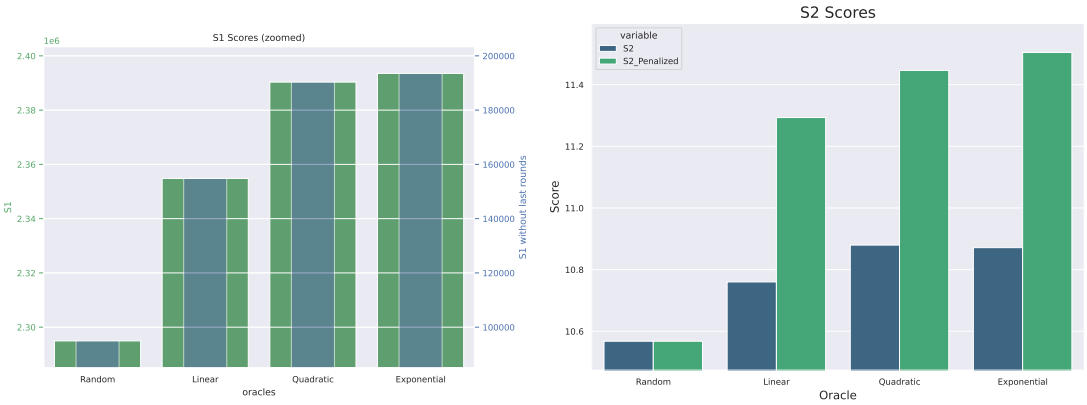


Figure 5.7: *Stochastic State Coverage scores for DTLS models of sizes 10 - 20.*

Sizes 10 - 20 The results for sizes 10 - 20 are shown in Fig. 5.7. The "Random" variant remains the top performer in both scores. Notably, it is the only variant that does not fail at all. Among the other variants, the "Linear" one performs the best. The least and greatest increase in queries with respect to the "Random" variant is:

$$\Delta Q = \frac{15.5 \times 10^4 - 9.5 \times 10^4}{9.5 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{6}{9.5} = 63.1\%} \tag{5.7}$$

$$\Delta Q = \frac{19.5 \times 10^4 - 9.5 \times 10^4}{9.5 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{10}{9.5} = 105.2\%} \tag{5.8}$$

Table 5.4: *The number of failures for the DTLS models of size 10-20 for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
scandium_latest_psk	13	0	0	0	0
scandium_latest_ecdhe_cert_nreq	17	0	0	0	0
openssl-1.1.1b_all_cert_req	19	0	15	16	19
scandium_latest_ecdhe_cert_none	13	0	0	0	0
pion_psk	14	0	0	0	0
scandium-2.0.0_psk	16	0	0	0	0
scandium_latest_ecdhe_cert_req	15	0	0	0	0
openssl-1.1.1b_all_cert_none_nreq	14	0	1	1	0
mbedtls_all_cert_req	17	0	0	0	0
nss-3.6.7_dhe_ecdhe_rsa	10	0	0	0	0
mbedtls_all_cert_none	12	0	0	0	0

Inspecting the failures in Table 5.4 , it can be noticed that there is an outlier model that causes the vast majority of the failures of the non "Random" variants.

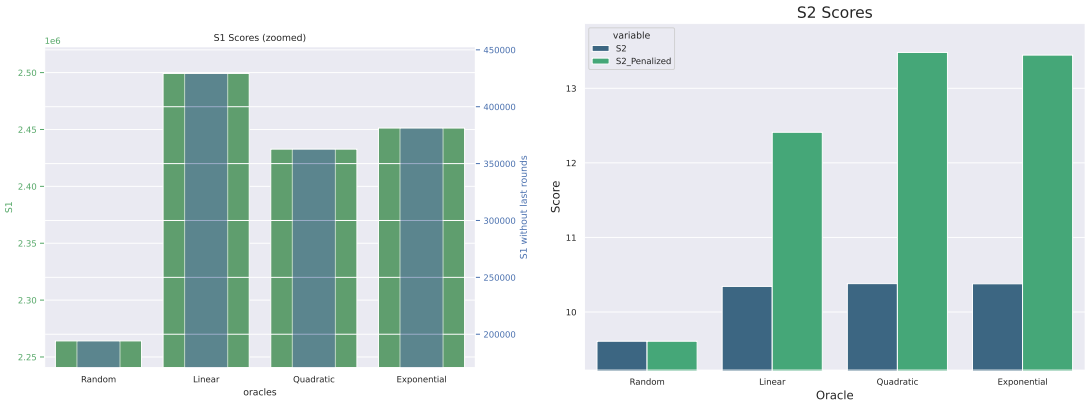


Figure 5.8: *Stochastic State Coverage scores for DTLS models of sizes 20 - 30.*

Sizes 20 - 30 The results for sizes 20 - 30 are shown in Fig. 5.8. Similar to the previous results, the "Random" variant remains the top performer in both scores and counts no failures. Among the other variants, the "Quadratic" one has the lowest s_1 score, while the "Linear" one has the lowest penalized s_2 score.

$$\Delta Q = \frac{36 \times 10^4 - 19 \times 10^4}{19 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{17}{19} = 89.4\%} \tag{5.9}$$

$$\Delta Q = \frac{42.5 \times 10^4 - 19 \times 10^4}{19 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{23.5}{19} = 123.6\%} \tag{5.10}$$

Table 5.5: *The number of failures for the DTLS models of size 20-30 for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
ctinydtls_psk	25	0	0	0	2
gnutls-3.5.19_psk_rsa_cert_nreq	29	0	0	0	2
etinydtls_psk	22	0	4	13	15
openssl-1.1.1b_all_cert_nreq	22	0	26	29	27
etinydtls_ecdhe_cert_none	22	0	12	15	16
etinydtls_ecdhe_cert_req	27	0	11	14	13
mbedtls_all_cert_nreq	20	0	0	0	0
ctinydtls_ecdhe_cert_none	25	0	0	3	2

Inspecting the failures in Table 5.5, it can be seen that, while there is no singular outlier, there is a subset of models that disproportionately cause the failures of the non "Random" variants.

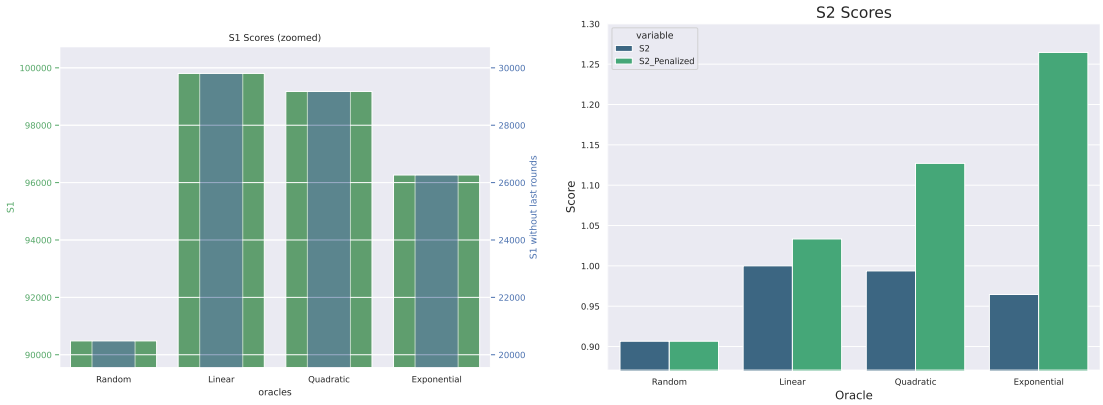


Figure 5.9: *Stochastic State Coverage scores for DTLS models of sizes 40 - 50.*

Sizes 40 - 50 The results for sizes 40 - 50 are shown in Fig. 5.9. This group contained only one model of size 45, therefore the results are not very informative. However, it is the case that all non "Random" variations perform better than the "Random" variation, be it by a small margin, and are just as successful.

$$\Delta Q = \frac{2.6 \times 10^4 - 2 \times 10^4}{2 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{26 - 20}{20} = 30\%} \tag{5.11}$$

$$\Delta Q = \frac{3 \times 10^4 - 2 \times 10^4}{2 \times 10^4} \Rightarrow \boxed{\Delta Q = 50\%} \tag{5.12}$$

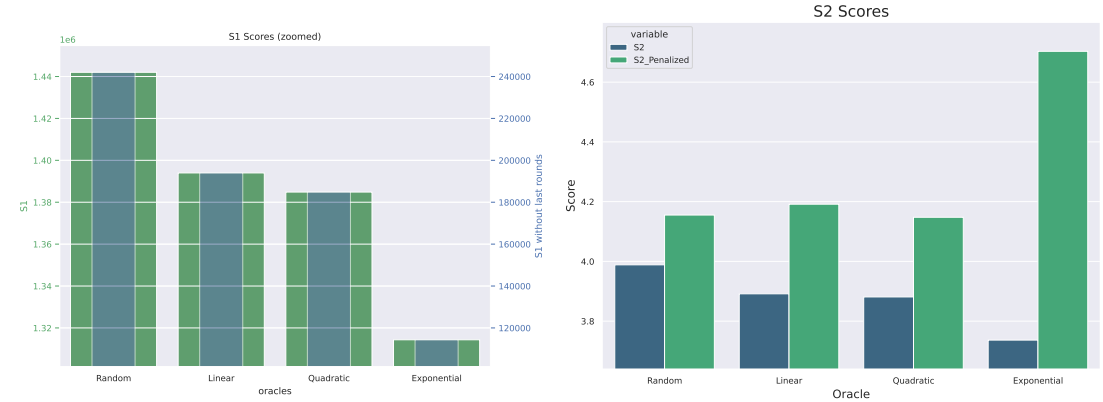


Figure 5.10: *Stochastic State Coverage scores for DTLS models of sizes 60 - 80.*

Sizes 60 - 80 The results for sizes 60 - 80 are shown in Fig. 5.10. In this group, the non "Random" variants perform better than the "Random" one, with the top performer being the "Exponential" variant, followed by the "Quadratic" and the "Linear" one. The penalized s_2 score shows that the "Exponential" variant is the least resilient to failures. The percentage of queries saved with respect to the "Random" variant is:

$$\Delta Q = \frac{240000 - 115000}{240000} \Rightarrow \boxed{\Delta Q = \frac{240 - 115}{240} = 52\%} \tag{5.13}$$

$$\Delta Q = \frac{240000 - 195000}{240000} \Rightarrow \boxed{\Delta Q = \frac{240 - 195}{240} = 18.7\%} \tag{5.14}$$

Table 5.6: *The number of failures for the DTLS models of size 60-80 for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
pion_ecdhe_cert_req	66	0	0	0	0
pion_ecdhe_cert_nreq	66	0	0	0	0
pion_ecdhe_cert_none	66	0	0	0	0
jsse-12_rsa_cert_none	79	5	9	8	29

Inspecting the failures in Table 5.6 shows that again there is an outlier model that causes the vast majority of the failures of all of the models, especially of the "Exponential" one.

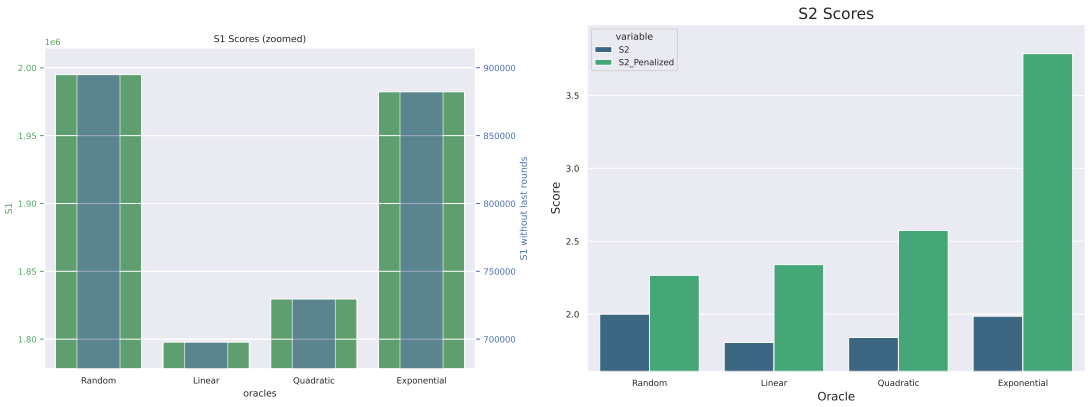


Figure 5.11: *Stochastic State Coverage scores for DTLS models of sizes 100 - 120.*

Sizes 100 - 120 The results for sizes 100 - 120 are shown in Fig. 5.11. The "Random" variant scores the highest s_1 score and s_2 score, but the lowest penalized s_2 score, meaning it is the least likely to fail. The "Linear" variant is the top performer, scoring the lowest s_1 score and the lowest s_2 score. All of the non "Random" variants perform better than the "Random" but they all tend to fail more, especially the "Exponential" variant.

$$\Delta Q = \frac{9 \times 10^5 - 8.75 \times 10^5}{9 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{9 - 8.75}{9} = 2.7\%} \tag{5.15}$$

$$\Delta Q = \frac{9 \times 10^5 - 7 \times 10^5}{9 \times 10^5} \Rightarrow \boxed{\Delta Q = 22\%} \tag{5.16}$$

Table 5.7: *The number of failures for the DTLS models of size 100-120 for the Stochastic State Coverage oracle with different probability distribution functions.*

Model	Size	Random	Linear	Square	Exp.
jsse-12_rsa_cert_req	113	1	7	7	24
jsse-12_rsa_cert_nreq	105	7	9	15	30

The details of the failures are shown in Table 5.7. The least and greatest percentage of queries saved with respect to the "Random" variant is:

Table 5.8: *Stochastic State Coverage variants performance summary*

Protocol	Greatest Extra	Least Extra	Least Saved	Greatest Saved
TLS	0 %	0 %	0 %	0 %
MQTT	■	■	0.6 %	18.5 %
TCP	26.7 %	■	■	7 %
DTLS	11.4 %	4.5 %	■	■
DTLS 10 20	105.2 %	63.1 %	■	■
DTLS 20 30	123.6 %	89.4 %	■	■
DTLS 40 50	50 %	30 %	■	■
DTLS 60 80	■	■	18.7 %	52 %
DTLS 100 120	■	■	2.7 %	22 %

Table 5.8 summarizes the results of the Stochastic State Coverage variants, showcasing the greatest and least extra queries, as well as the least and greatest saved queries with respect to the "Random" variant. The ■ symbol indicates that the metric is not applicable. For example, the variants of the Stochastic State Coverage oracle in the MQTT protocol experiments did not cause any extra queries, therefore the ■ symbol is used because the metric was not computed.

The resuls are mixed. Some groups of models, like the MQTT protocol models as well as the DTLS models of sizes 60-120, seem to benefit from the Stochastic State Coverage variants, while others, like the TCP and the DTLS models of sizes 10-50, do not.

5.2 WMethod Variants

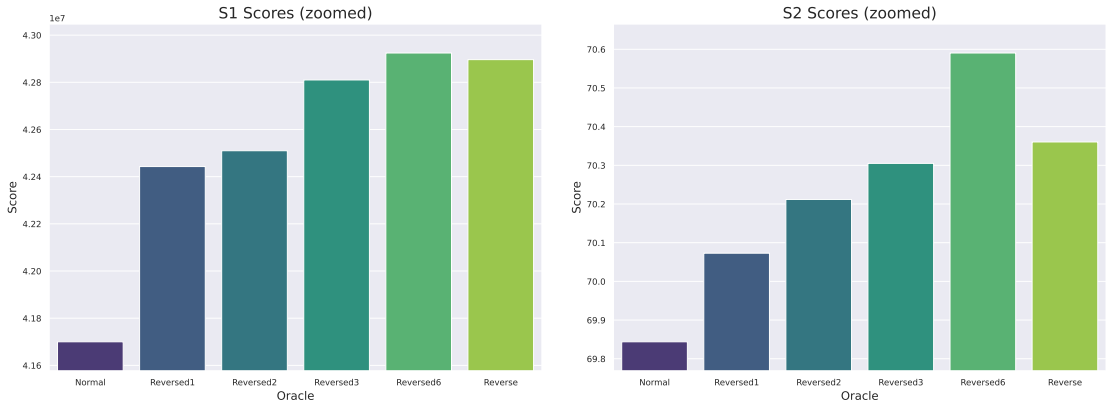


Figure 5.12: *The s_1 and s_2 scores of the W Method and of the Reversed W Method, zoomed in.*

The s_1 and s_2 scores for the WMethod variants are presented in below in Fig. 5.12. It seems that, overall, the "Normal" variant of the W method performs better. We inspect the protocol-specific results to gain more information.

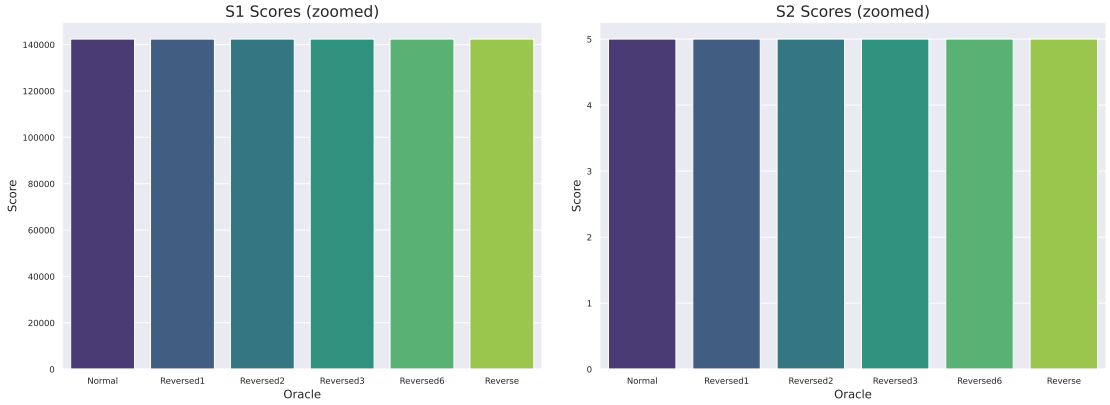


Figure 5.13: *The s_1 and s_2 scores of the W Method and of the Reversed W Method for TLS, zoomed in.*

TLS Similar to the state coverage methods, Fig. 5.13 show that the TLS protocols are again learned in just one round and therefore the difference conformance testing algorithms all perform the same.

MQTT The MQTT-specific results in Fig. 5.14 seem to favour the "Reverse" variant of the W method, as both the s_1 and s_2 scores are lower, though by a margin. Specificaly, the variant with parameter $diff\_depth = 2, 3, 6$, which means that the prefixes of the first $2, 3, 6$ age group come first respectively, performs the best. This is consistent with the fact that all of the stochastic state coverage variants outperformed the uniformly random one in the MQTT protocol.

$$\Delta Q = \frac{53000 - 52000}{52000} \Rightarrow \boxed{\Delta Q = \frac{1}{52} = 1.9\%} \tag{5.17}$$

$$\Delta Q = \frac{52000 - 48500}{52000} \Rightarrow \boxed{\Delta Q = \frac{520 - 485}{520} = 6.7\%} \tag{5.18}$$

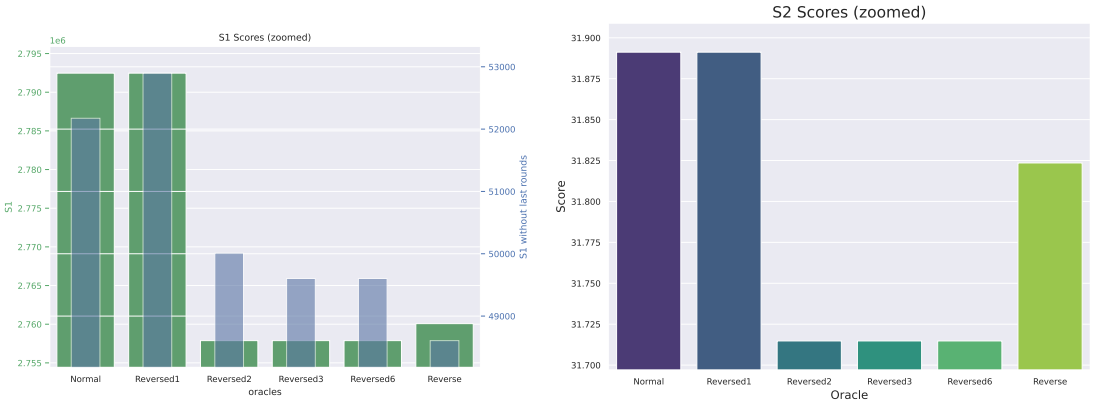


Figure 5.14: *The s_1 and s_2 scores of the W Method and of the Reversed W Method for MQTT, zoomed in.*

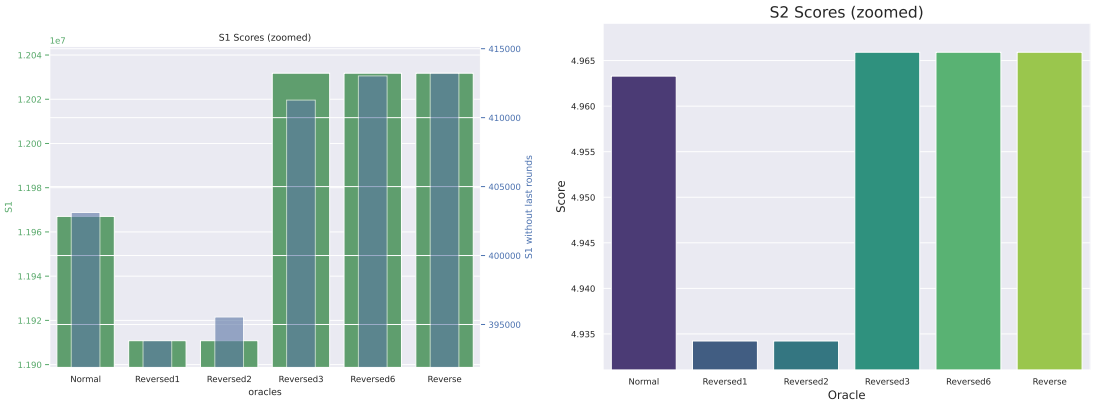


Figure 5.15: *The s_1 and s_2 scores of the W Method and of the Reversed W Method for TCP, zoomed in.*

TCP The TCP-specific results in Fig. 5.15 favour the "Reversed1" and "Reversed2" variants, which have the lowest s_1 and s_2 scores. It is interesting to note that, the variants give too much priority to the newer states end up exhibiting worse performance. The greatest savings and losses of queries, by the "Reversed1" and "Reversed" variants, respectively are:

$$\Delta Q = \frac{402500 - 395000}{402500} \Rightarrow \boxed{\Delta Q = \frac{4025 - 3950}{4025} = 1.8\%} \tag{5.19}$$

$$\Delta Q = \frac{412500 - 402500}{402500} \Rightarrow \boxed{\Delta Q = \frac{100}{4025} = 2.48\%} \tag{5.20}$$

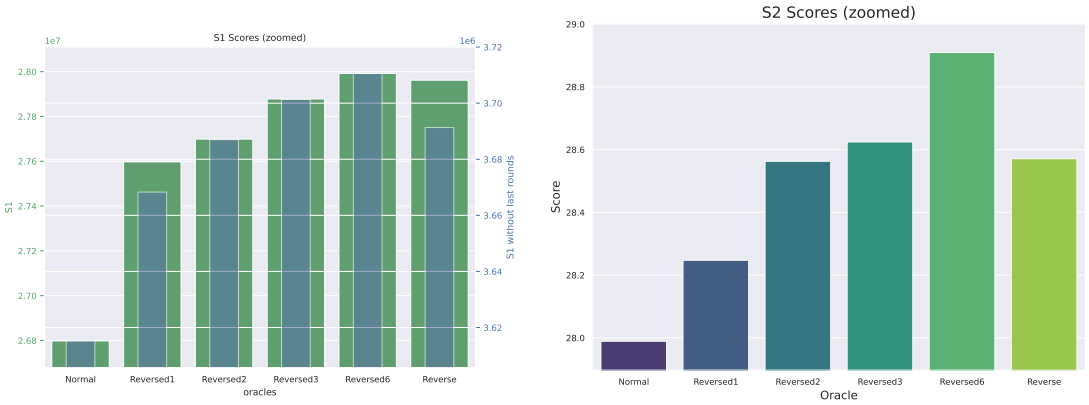


Figure 5.16: *The s_1 and s_2 scores of the W Method and of the Reversed W Method for DTLS, zoomed in.*

DTLS Finally, the DTLS-specific results in Fig. 5.16 favor the "Normal" variant of the W Method, as it performs better in both scores. Due to the high variety of sizes in the DTLS models, we present more specific results grouped by size.

$$\Delta Q = \frac{3.71 \times 10^6 - 3.62 \times 10^6}{3.62 \times 10^6} \Rightarrow \boxed{\Delta Q = \frac{0.09}{3.62} = 2.48\%} \tag{5.21}$$

$$\Delta Q = \frac{3.67 \times 10^6 - 3.62 \times 10^6}{3.62 \times 10^6} \Rightarrow \boxed{\Delta Q = \frac{0.05}{3.62} = 1.38\%} \tag{5.22}$$

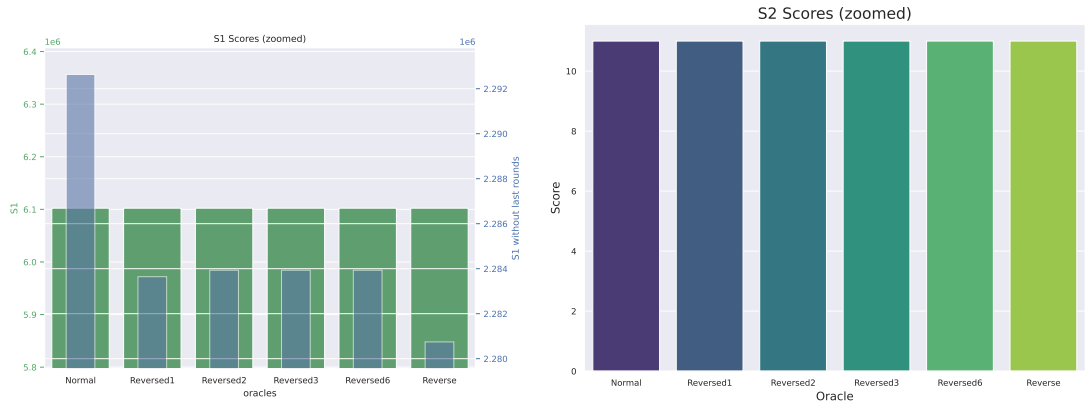


Figure 5.17: *W Method scores for DTLS models of sizes 10 - 20.*

Sizes 10 - 20 The results for sizes 10 - 20 are shown in Fig. 5.17. All of the variants perform the same. This may be due to how the counterexamples are distributed in the test suite. It is interesting to note that, inspecting the s_1 scores without the last round queries shows that the performance of the different variants during the learning process is not the same, but rather the "Reversed" use less queries, creating a difference that is made up at the end, either because of a larger final test suite or because of more learning rounds.

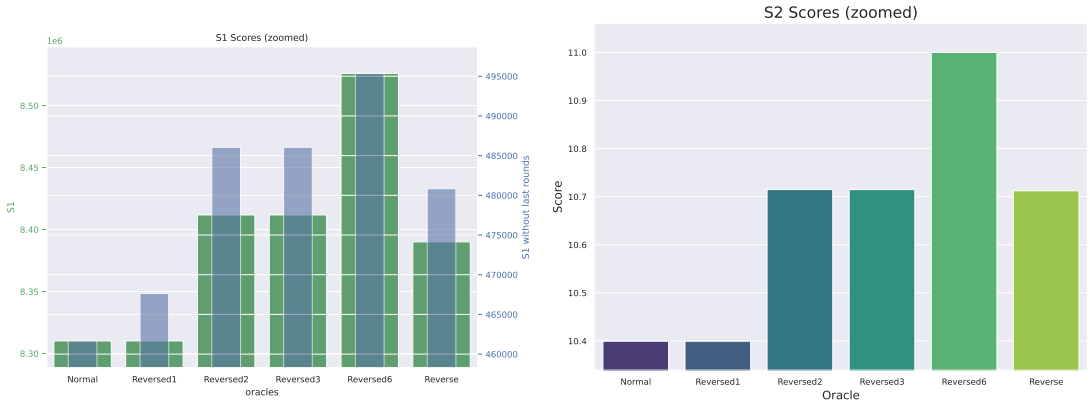


Figure 5.18: *W Method scores for DTLS models of sizes 20 - 30.*

Sizes 20 - 30 The results for sizes 20 - 30 are shown in Fig. 5.18. The "Normal" variant outperforms the "Reversed" variants, with the exception of the variant with parameter $diff\_depth = 1$. The s_1 score ommiting the last round shows that the "Normal" variant tends to find counterexamples faster than all other variants. The least and greatest amount of extra queries with respect to the "Normal"

$$\Delta Q = \frac{467.5 \times 10^3 - 462.5 \times 10^3}{467.5 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{5}{467.5} = 1.06\%} \tag{5.23}$$

$$\Delta Q = \frac{495 \times 10^3 - 462.5 \times 10^3}{495 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{32.5}{495} = 6.56\%} \tag{5.24}$$

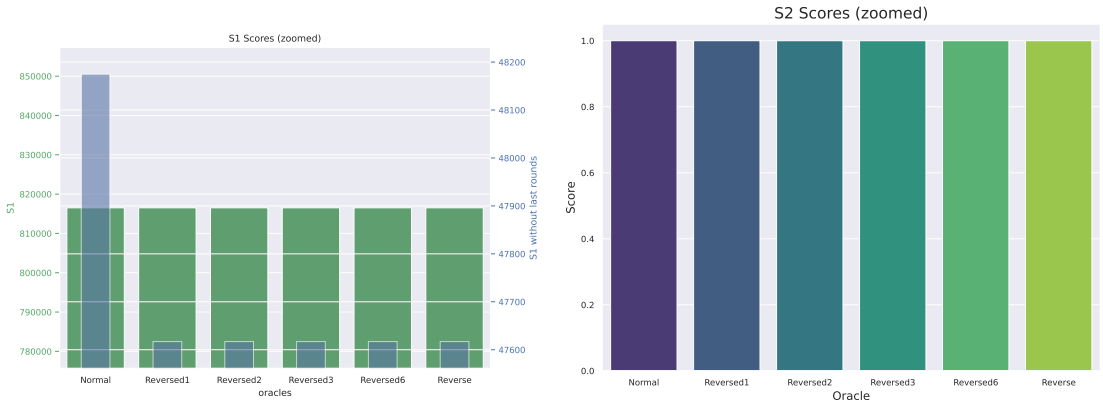


Figure 5.19: *W Method scores for DTLS models of sizes 40 - 50.*

Sizes 40 - 50 The results for sizes 40 - 50 are shown in Fig. 5.19. This group contains only one model so the results are not very informative. Be that as it may, all of the variants perform the same overall, with the "Reversed" variants requiring less queries in the intermediate rounds.

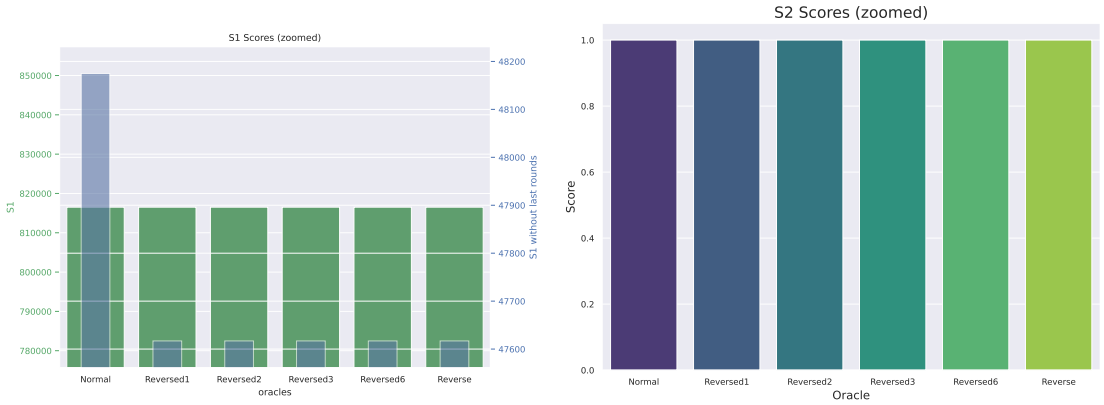


Figure 5.20: *W Method scores for DTLS models of sizes 60 - 80.*

Sizes 60 - 80 The results for sizes 60 - 80 are shown in Fig. 5.20 and are identical to the results for sizes 40 - 50. The "Reversed" variants require less queries in the intermediate rounds.

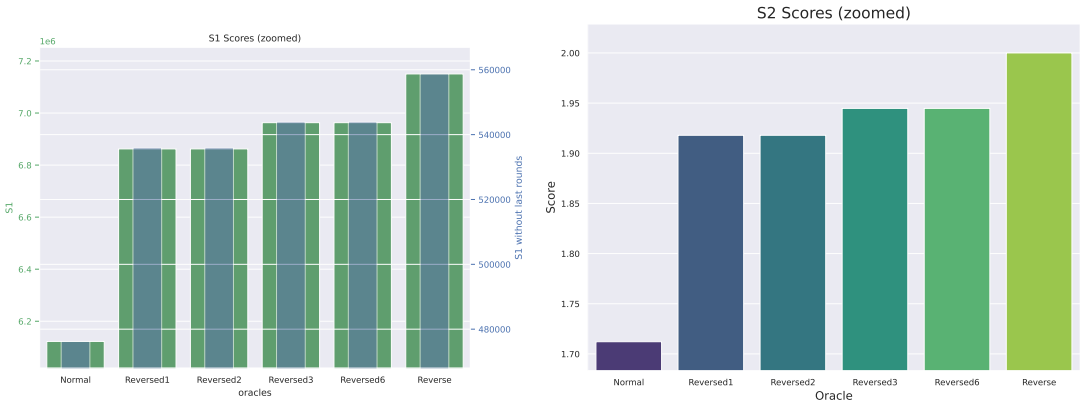


Figure 5.21: *W Method scores for DTLS models of sizes 100 - 120.*

Sizes 100 - 120 The results for sizes 100 - 120 are shown in Fig. 5.21. The "Normal" variant outperforms all of the "Reversed" variants. The least and greatest amount of extra queries with respect to the "Normal" variant is:

$$\Delta Q = \frac{560 \times 10^3 - 475 \times 10^3}{560 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{85}{560} = 15.17\%} \tag{5.25}$$

$$\Delta Q = \frac{535 \times 10^3 - 475 \times 10^3}{535 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{60}{535} = 11.21\%} \tag{5.26}$$

Table 5.9: *WMethod variants performance summary*

Protocol	Greatest Extra	Least Extra	Least Saved	Greatest Saved
TLS	0 %	0 %	0 %	0 %
MQTT	■	■	1.9 %	6.7 %
TCP	2.48 %	■	■	1.8 %
DTLS	2.48 %	1.38 %	■	■
DTLS 10 20	0 %	0 %	0 %	0 %
DTLS 20 30	6.56 %	1.06 %	■	■
DTLS 40 50	0 %	0 %	0 %	0 %
DTLS 60 80	0 %	0 %	0 %	0 %
DTLS 100 120	15.17 %	11.21 %	■	■

Table 5.9 summarizes the results of the WMethod variants, showcasing the greatest and least extra queries, as well as the least and greatest saved queries with respect to the "Normal" variant. The performance of all of the variants seems to be similar, with small deviations. In most groups of models, the "Normal" variant outperforms the "Reversed" variants, with the exception of the MQTT protocol, where the "Reversed" variants perform better.

5.3 WpMethod Variants

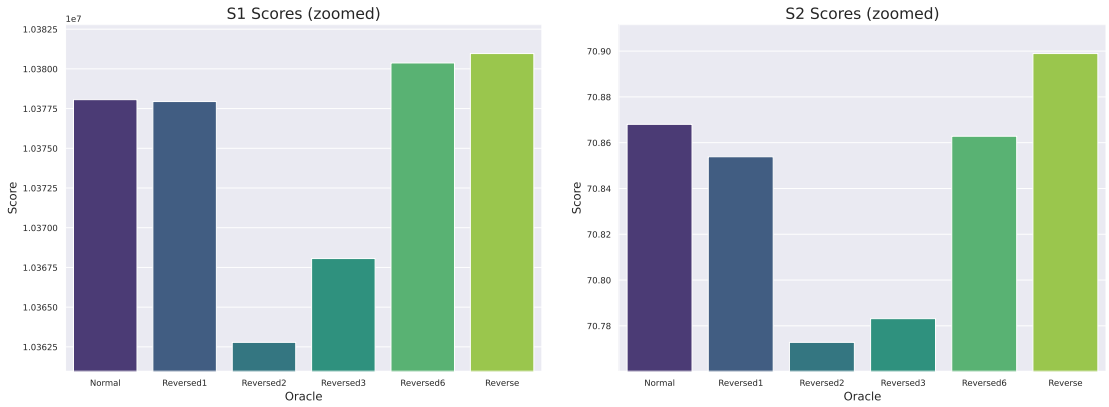


Figure 5.22: *The s_1 and s_2 scores of the Wp Method variants zoomed in.*

The results for the WpMethod variants are shown in Fig. 5.22. The best overall method is the "Reversed" variant with parameter $diff\_depth = 2$, scoring the lowest s_1 and s_2 scores. Inspecting the protocol-specific results will provide better insight into the performance of the variants.

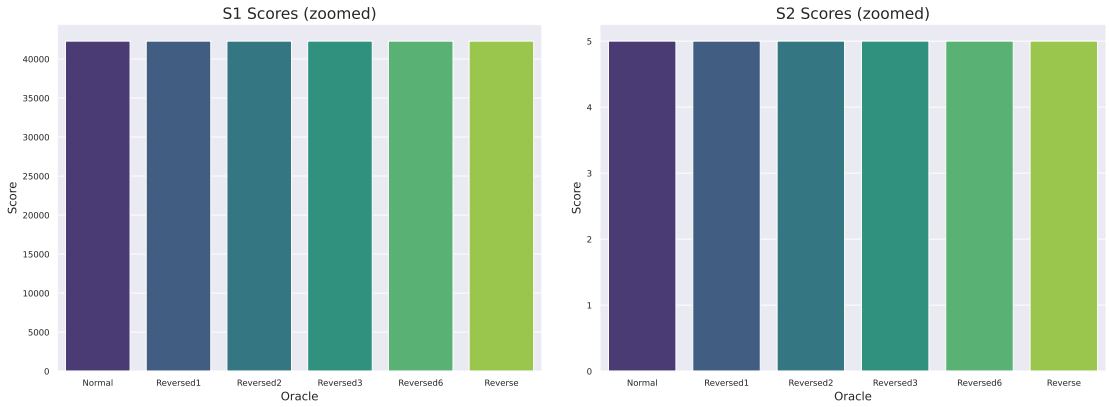


Figure 5.23: *The s_1 and s_2 scores of the Wp Method variants for TLS.*

TLS The TLS-specific Fig. 5.23 shows the same scores for every method.

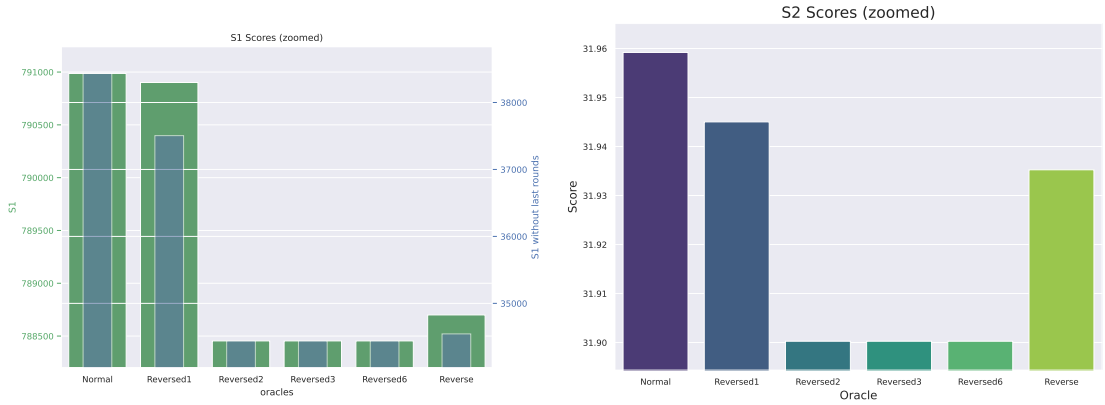


Figure 5.24: *The s_1 and s_2 scores of the Wp Method variants for MQTT.*

MQTT The MQTT-specific Fig. 5.24 shows that the "Reversed" variant scores better, with the parameter *diff_depth* $= 2, 3, 6$ being the best performers. This agrees with the corresponding "WMethod" results shown in Fig. 5.14. The least and greatest percentage of saved queries is:

$$\Delta Q = \frac{38.5 \times 10^3 - 37.5 \times 10^3}{38.5 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{1}{37.5} = 2.66\%} \tag{5.27}$$

$$\Delta Q = \frac{38.5 \times 10^3 - 34.5 \times 10^3}{38.5 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{4}{38.5} = 10.38\%} \tag{5.28}$$

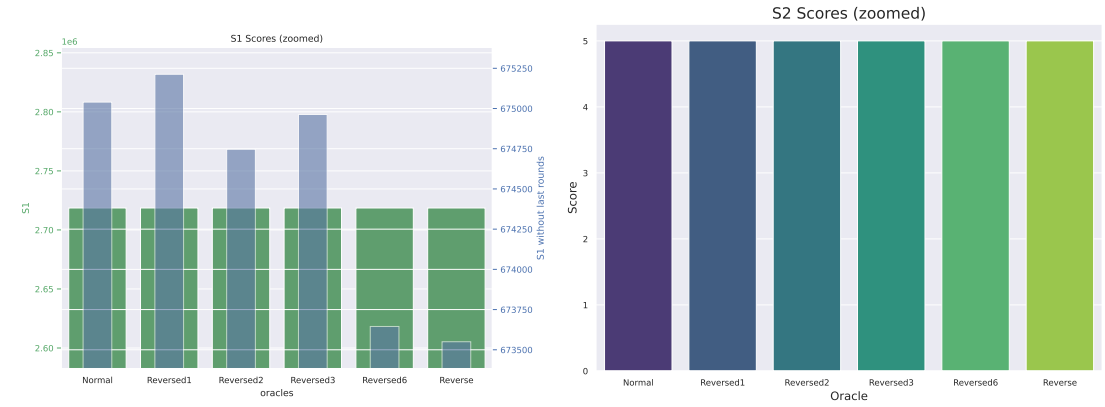


Figure 5.25: *The s_1 and s_2 scores of the Wp Method variants for TCP.*

TCP The TCP-specific Fig. 5.25 shows the exact same performance for all methods. As far as the intermediate rounds are concerned, the "Reversed" variants find counterexamples faster than the "Normal" one, with the exception of the one with parameter *diff_depth* $= 1$.

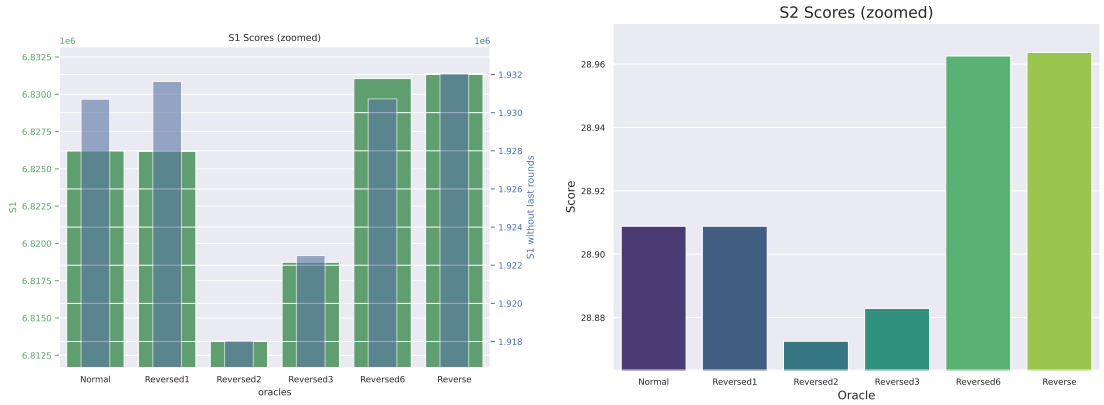


Figure 5.26: *The s_1 and s_2 scores of the Wp Method variants for DTLS.*

DTLS The DTLS-specific Fig. 5.26 shows some variants of the "Reversed" method to be the worst performing, specifically those with parameter *diff_depth* $= 6, full$, the variants with parameters *diff_depth* $= 2, 3$ to be the top performers and finally the variant with paramter *diff_depth* $= 1$ to perform as well as the "Normal" variant. The greatest percentage of extra queries and of queries saved is:

$$\Delta Q = \frac{1.932 \times 10^6 - 1.931 \times 10^3}{1.932 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{0.001}{1.932} = 0.05\%} \tag{5.29}$$

$$\Delta Q = \frac{1.931 \times 10^6 - 1.918 \times 10^3}{1.931 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{0.013}{1.931} = 0.67\%} \tag{5.30}$$

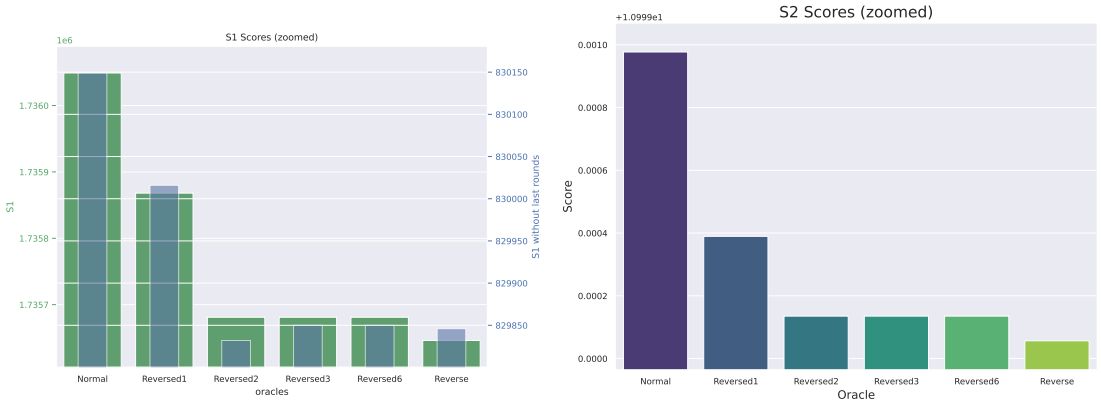


Figure 5.27: *Wp Method scores for DTLS models of sizes 10 - 20.*

Sizes 10 - 20 The results for sizes 10 - 20 are shown in Fig. 5.27. All of the "Reversed" variants outperform the "Normal" variant and they get progressively better as the *diff_depth* parameter increases. The least and greatest percentage of saved queries is:

$$\Delta Q = \frac{830150 - 830025}{830150} \Rightarrow \boxed{\Delta Q = \frac{125}{830150} = 0.01\%} \tag{5.31}$$

$$\Delta Q = \frac{830150 - 829825}{830150} \Rightarrow \boxed{\Delta Q = \frac{325}{830150} = 0.03\%} \tag{5.32}$$

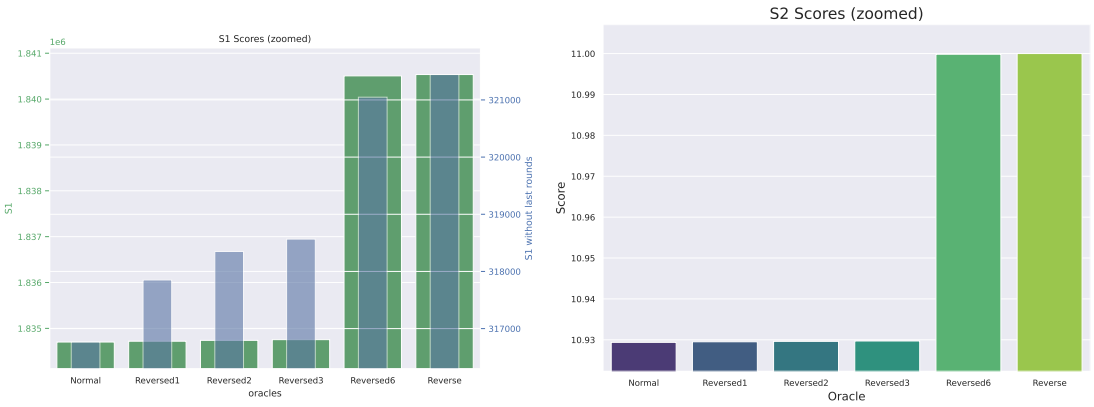


Figure 5.28: *Wp Method scores for DTLS models of sizes 20 - 30.*

Sizes 20 - 30 The results for sizes 20 - 30 are shown in Fig. 5.28. In this case, the top performer is the "Normal" variant and "Reversed" variants perform at best just as well and at worst worse than the "Normal" variant, specifically the variants with parameter *diff_depth* = 6,*full*. The least and greatest percentage of extra queries is:

$$\Delta Q = \frac{318 \times 10^3 - 316.5 \times 10^3}{316.5 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{1.5}{316.5} = 0.47\%} \tag{5.33}$$

$$\Delta Q = \frac{321.5 \times 10^3 - 316.5 \times 10^3}{316.5 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{5}{316.5} = 1.57\%} \tag{5.34}$$

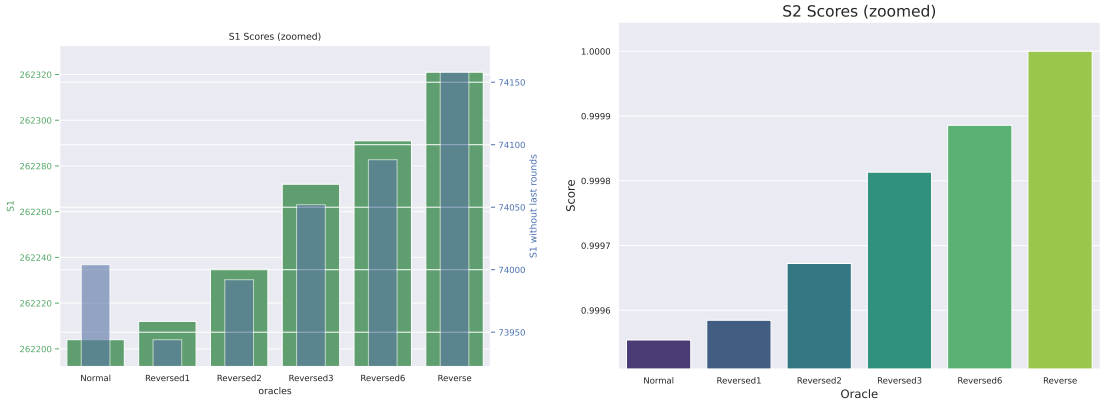


Figure 5.29: *Wp Method scores for DTLS models of sizes 40-50.*

Sizes 40 - 50 The results for sizes 40–50 are shown in Fig. 5.29. The performance of the variants for this group, constisting of only one model, gets progressively worse as the *diff_depth* parameter increases, with the top performer being the "Normal" variant. The least and greatest percentage of extra queries is.

$$\Delta Q = \frac{74 \times 10^3 - 73.95 \times 10^3}{74 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{0.05}{74} = 0.06\%} \tag{5.35}$$

$$\Delta Q = \frac{74.15 \times 10^3 - 74 \times 10^3}{74 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{0.15}{74} = 0.2\%} \tag{5.36}$$

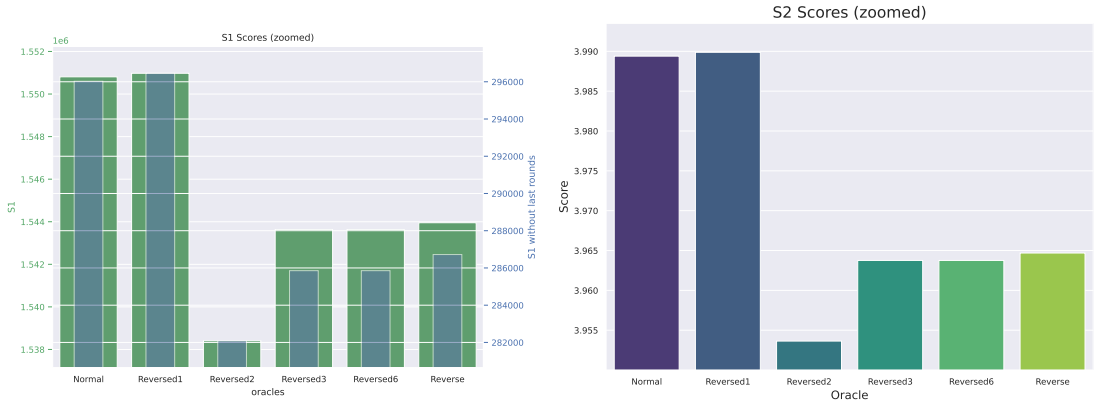


Figure 5.30: *Wp Method scores for DTLS models of sizes 60-80.*

Sizes 60–80 The results for sizes 60–80 are shown in Fig. 5.30. The top performers of this group is the "Reversed" variant with parameter $diff\_depth = 2$, followed by the variants with parameters $diff\_depth = 3, 6, full$. The worst performer is the variant with parameter $diff\_depth = 1$, slightly outperformed by the "Normal" variant. The least and greatest percentage of saved queries is:

$$\Delta Q = \frac{296.25 \times 10^3 - 296 \times 10^3}{296 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{0.25}{296} = 0.08\%} \tag{5.37}$$

$$\Delta Q = \frac{296 \times 10^3 - 282 \times 10^3}{296 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{14}{282} = 4.96\%} \tag{5.38}$$

Sizes 100–120 The results for sizes 100–120 are shown in Fig. 5.31. Here, all of the variants perform just as well, with the exception of the "Reversed" variants with parameter $diff\_depth = 6, full$, which are the worst performers. The greatest percentage of extra queries is:

$$\Delta Q = \frac{420 \times 10^3 - 414 \times 10^3}{414 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{6}{414} = 1.44\%} \tag{5.39}$$

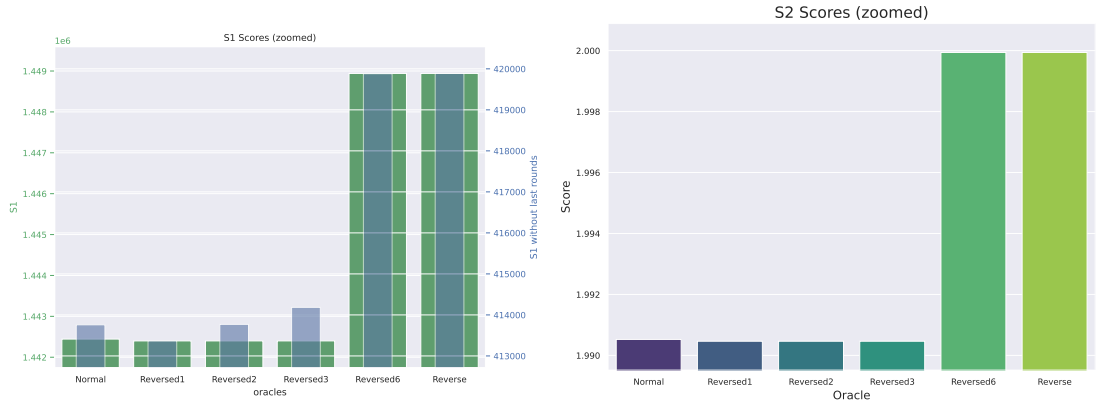


Figure 5.31: *Wp Method scores for DTLS models of sizes 100 - 120.*

Table 5.10: *WpMethod variants performance summary*

Protocol	Greatest Extra	Least Extra	Least Saved	Greatest Saved
TLS	0 %	0 %	0 %	0 %
MQTT	■	■	2.66 %	10.38 %
TCP	0 %	0 %	0 %	0 %
DTLS	0.05 %	■	■	0.67 %
DTLS 10 20	■	■	0.01 %	0.03 %
DTLS 20 30	1.57 %	0.47 %	■	■
DTLS 40 50	0.2 %	0.06 %	■	■
DTLS 60 80	■	■	0.08 %	4.96 %
DTLS 100 120	1.44 %	■	■	■

Table 5.10 summarizes the results of the WpMethod variants, showcasing the greatest and least extra queries, as well as the least and greatest saved queries with respect to the “Normal” variant. Even though the performance of the variants is similar, the “Reversed” variants outperform the “Normal” variant in the group of MQTT models and DTLS models of sizes 60-80, while not performing significantly worse in the other groups.

5.4 Random WpMethod Variants

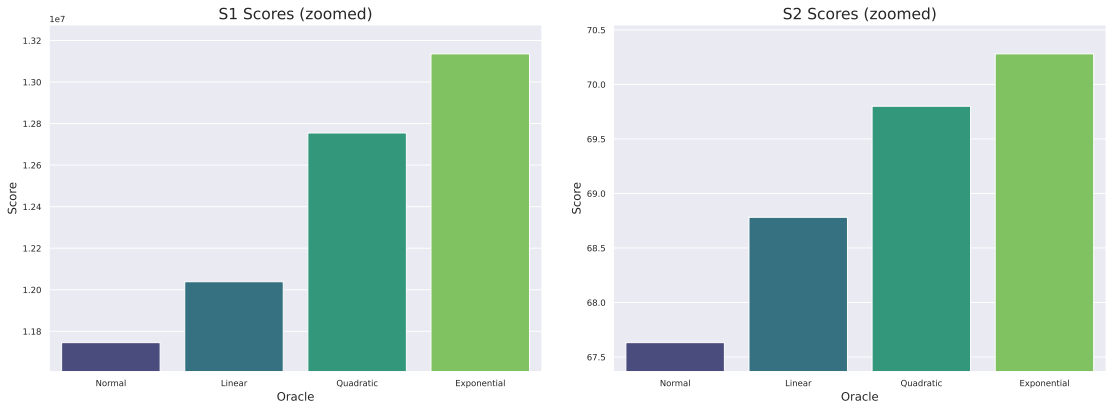


Figure 5.32: *The s_1 and s_2 scores of the Random Wp Method variants.*

The results for the Random WpMethod variants are shown in Fig. 5.32. The best overall method is the "Normal" variant, scoring the lowest s_1 and s_2 scores. Inspecting the protocol-specific results will provide better insight into the performance of the variants.

TLS The TLS-specific Fig. 5.33 shows the same scores for every method.

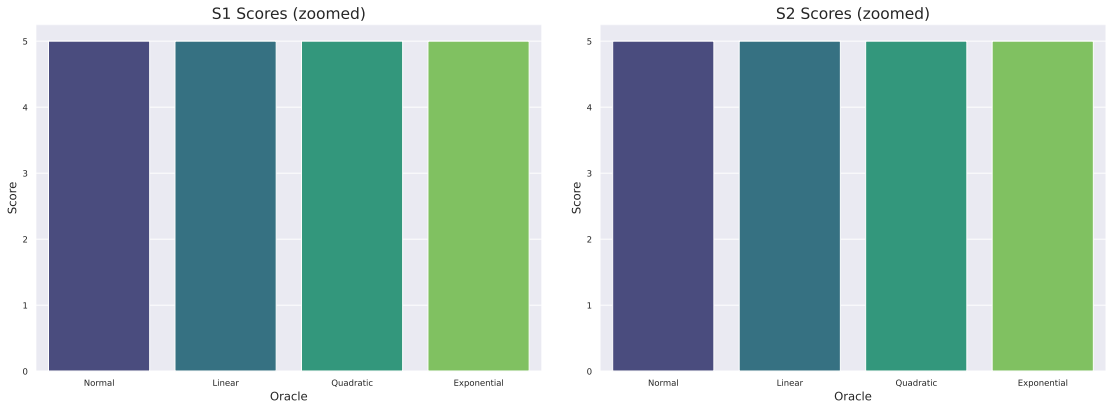


Figure 5.33: *The s_1 and s_2 scores of the Random Wp Method variants for TLS.*

MQTT The MQTT-specific Fig. 5.34 show that all variants that check the new states first offer better performance than the uniformly "Normal" variant, even when accounting for failed learning experiments. The "Exponential" one is the top performer in both scores, while the "Quadratic" performs almost identically to the "Linear". The least and greatest percentage of saved queries is:

$$\Delta Q = \frac{9000 - 7500}{9000} \Rightarrow \boxed{\Delta Q = \frac{1500}{9000} = 16.6\%} \tag{5.40}$$

$$\Delta Q = \frac{9000 - 6750}{9000} \Rightarrow \boxed{\Delta Q = \frac{2250}{9000} = 25\%} \tag{5.41}$$

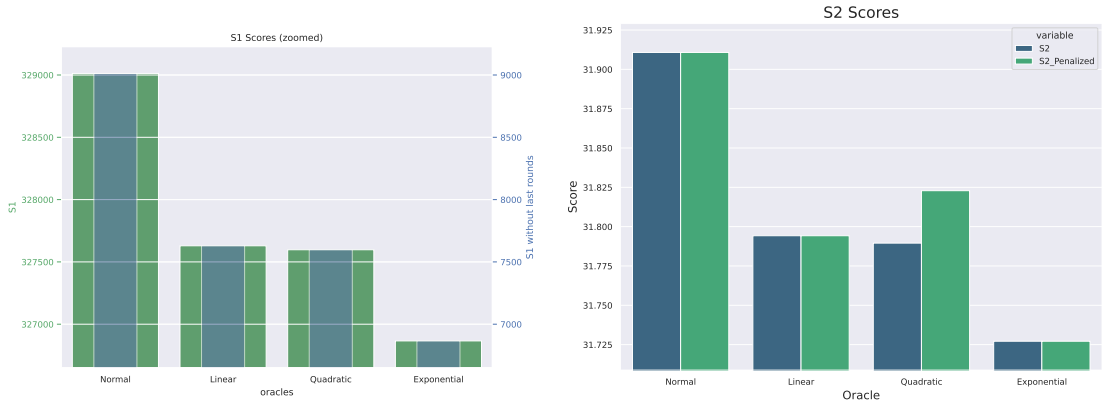


Figure 5.34: *The s_1 and s_2 scores of the Random Wp Method variants for MQTT.*

Table 5.11: *The number of failures for the MQTT models for the Stochastic Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
hbmqtt__two_client_will_retain	17	0	0	1	0

The slightly higher penalized s_2 score of the "Quadratic" variant is due to one failed experiment, which was caused by the model shown in Table 5.11, the same model that caused the "Exponential" variant of Stochastic State Coverage to fail, as noted in table 5.1.

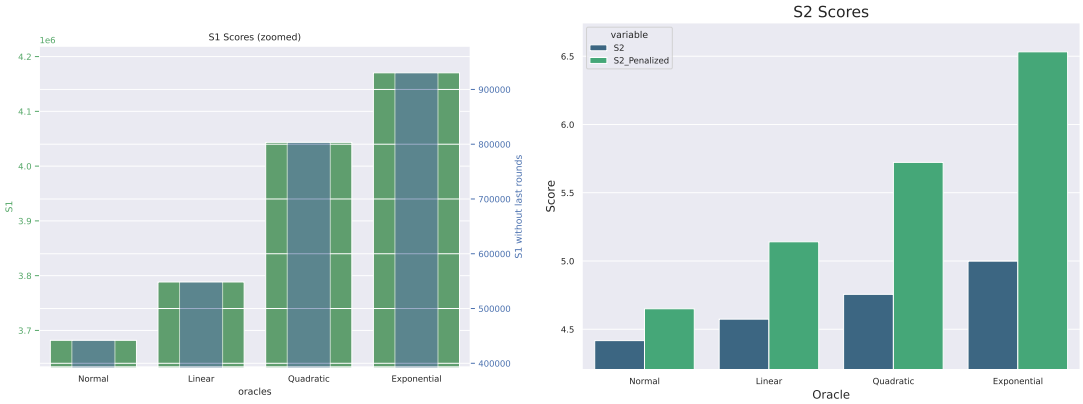


Figure 5.35: *The s_1 and s_2 scores of the Random Wp Method variants for TCP.*

TCP The TCP-specific Fig. 5.35 shows that the "Normal" variant outperforms all of the other variants while at the same time having the smaller failure rate. The least and greatest percentage of extra queries is:

$$\Delta Q = \frac{5.5 \times 10^5 - 4.5 \times 10^5}{4.5 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{1.5}{4.5} = 33.3\%} \tag{5.42}$$

$$\Delta Q = \frac{9.25 \times 10^5 - 4.5 \times 10^5}{4.5 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{4.75}{4.5} = 105.5\%} \tag{5.43}$$

Table 5.12: *The number of failures for the TCP models for the Stochastic Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
tcp_server_windows_trans	38	5	7	11	16
TCP_Linux_Client	15	2	9	17	10
tcp_server_ubuntu_trans	57	0	1	0	7
TCP_Linux_Server	57	0	0	1	8
tcp_server_bsd_trans	55	0	0	0	5

Table 5.12 shows the failed experiments.

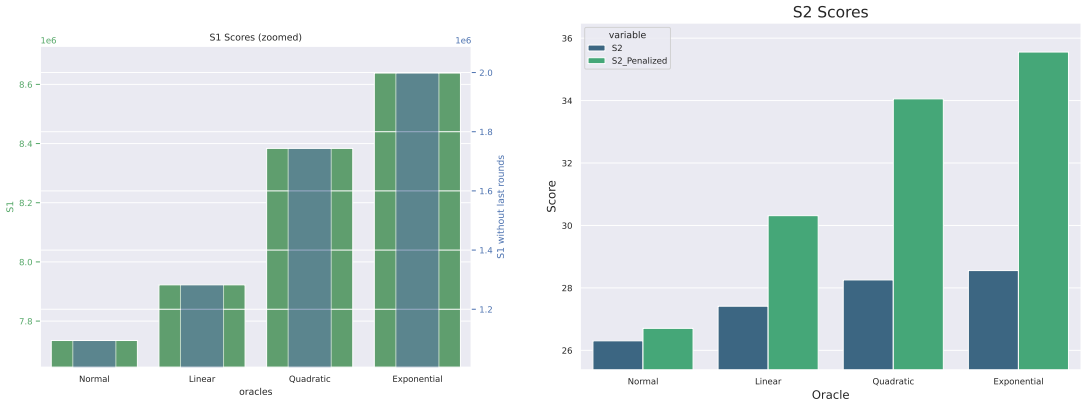


Figure 5.36: *The s_1 and s_2 scores of the Random Wp Method variants for DTLS.*

DTLS The DTLS-specific Fig. 5.36 again shows the "Normal" variant outperforming the rest and being less error prone. The least and greatest percentage of extra queries is:

$$\Delta Q = \frac{1.3 \times 10^6 - 1.1 \times 10^6}{1.1 \times 10^6} \Rightarrow \boxed{\Delta Q = \frac{0.2}{1.1} = 18.18\%} \tag{5.44}$$

$$\Delta Q = \frac{2 \times 10^6 - 1.1 \times 10^6}{1.1 \times 10^6} \Rightarrow \boxed{\Delta Q = \frac{0.9}{1.1} = 81.81\%} \tag{5.45}$$

However, due to the variety of model sizes in the set of DTLS models that were tested, it is worth examining them closer, grouped by size.

Sizes 10 - 20 The results for sizes 10 - 20 are shown in Fig. 5.37. The "Normal" variant outperforms the other variants while having the lowest chance of failing. The least and greatest percentage of extra queries is:

$$\Delta Q = \frac{1.75 \times 10^5 - 1.3 \times 10^5}{1.3 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{0.45}{1.3} = 34.61\%} \tag{5.46}$$

$$\Delta Q = \frac{1.95 \times 10^5 - 1.3 \times 10^5}{1.3 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{0.65}{1.3} = 50\%} \tag{5.47}$$

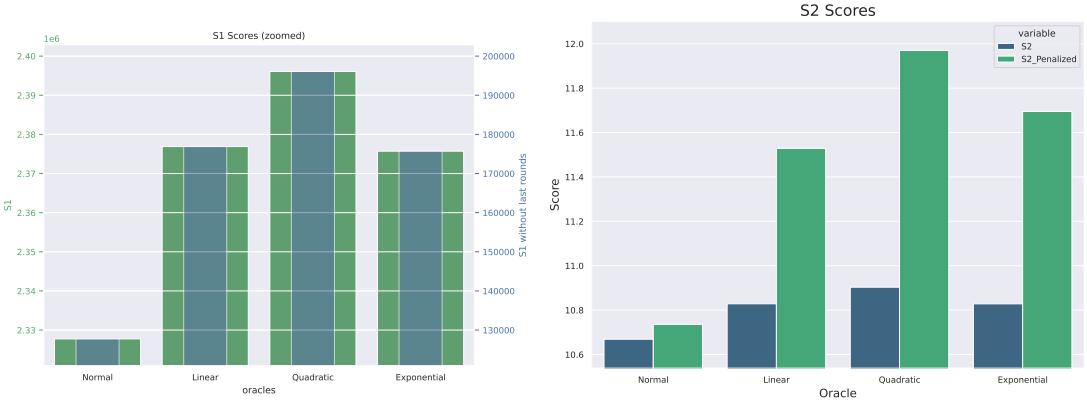


Figure 5.37: *Random Wp Method scores for DTLS models of sizes 10-20.*

Table 5.13: *The number of failures for the DTLS models of size 10-20 for the Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
scandium_latest_psk	13	0	0	0	0
scandium_latest_ecdhe_cert_nreq	17	0	0	0	0
openssl-1.1.1b_all_cert_req	19	2	21	25	26
scandium_latest_ecdhe_cert_none	13	0	0	0	0
pion_psk	14	0	0	0	0
scandium-2.0.0_psk	16	0	0	0	0
scandium_latest_ecdhe_cert_req	15	0	0	0	0
openssl-1.1.1b_all_cert_none_nreq	14	0	0	7	0
mbedtls_all_cert_req	17	0	0	0	0
nss-3.6.7_dhe_ecdhe_rsa	10	0	0	0	0
mbedtls_all_cert_none	12	0	0	0	0

Table 5.13 shows the failures of the variants for each model. It is evident that there is an outlier that causes the vast majority of the failures of the non “Normal” variants and it is the same outlier as in the case of table 5.4.

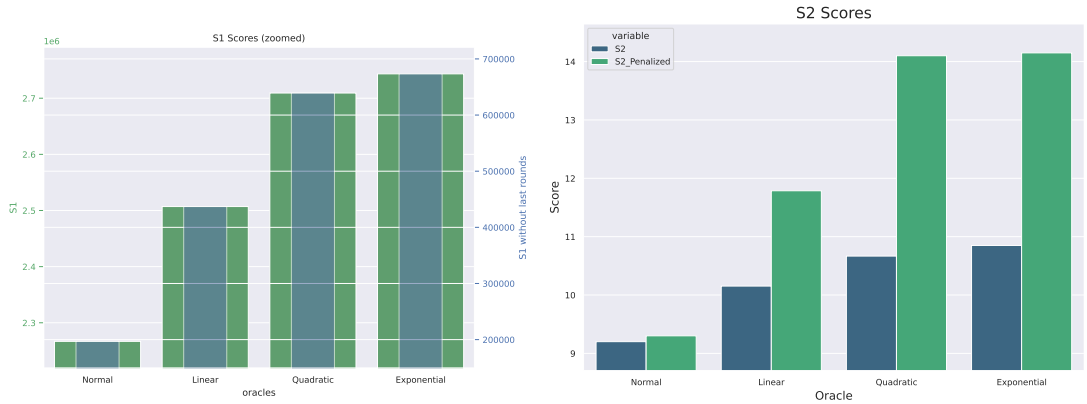


Figure 5.38: *Random Wp Method scores for DTLS models of sizes 20-30.*

Sizes 20–30 The results for sizes 20–30 are shown in Fig. 5.38. Similar to the previous results, the "Normal" variation is the top perfomer and the least error prone. The least and greatest percentage of extra queries is:

$$\Delta Q = \frac{4.25 \times 10^5 - 2 \times 10^5}{2 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{2.25}{2} = 125\%} \tag{5.48}$$

$$\Delta Q = \frac{6.75 \times 10^5 - 2 \times 10^5}{2 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{4.75}{2} = 237.5\%} \tag{5.49}$$

Table 5.14: *The number of failures for the DTLS models of size 20-30 for the Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
ctinydtls_psk	25	0	0	2	1
gnutls-3.5.19_psk_rsa_cert_nreq	29	0	0	0	3
etinydtls_psk	22	0	0	11	6
openssl-1.1.1b_all_cert_nreq	22	2	27	30	29
etinydtls_ecdhe_cert_none	22	0	3	18	14
etinydtls_ecdhe_cert_req	27	0	4	20	15
mbedtls_all_cert_nreq	20	0	0	0	0
ctinydtls_ecdhe_cert_none	25	0	1	1	4

Table 5.14 shows the failures of the variants for each model. Despite there not being one particular outlier, there is again a small subset of models that cause the vast majority of the failures of the non "Normal" variants.

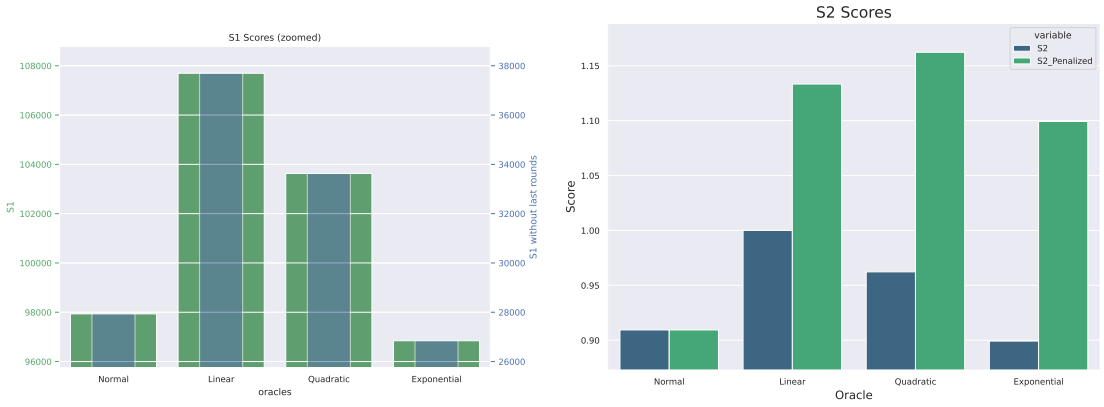


Figure 5.39: *Random Wp Method scores for DTLS models of sizes 40-50.*

Sizes 40–50 The results for sizes 40–50 are shown in Fig. 5.39. This group contained only one model of size 45, therefore the results are not very informative. However, it is the case that the "Exponential" variant performs the least number of queries, albeit by a small margin. Be that as it may, it is the most error prone variant, while the "Normal" variation performed slightly more queries while being successful at all experiments. The greatest percentage of saved and extra queries is:

$$\Delta Q = \frac{28 \times 10^3 - 27 \times 10^3}{28 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{1}{28} = 3.57\%} \tag{5.50}$$

$$\Delta Q = \frac{37.5 \times 10^3 - 28 \times 10^3}{28 \times 10^3} \Rightarrow \boxed{\Delta Q = \frac{9.5}{28} = 33.9\%} \tag{5.51}$$

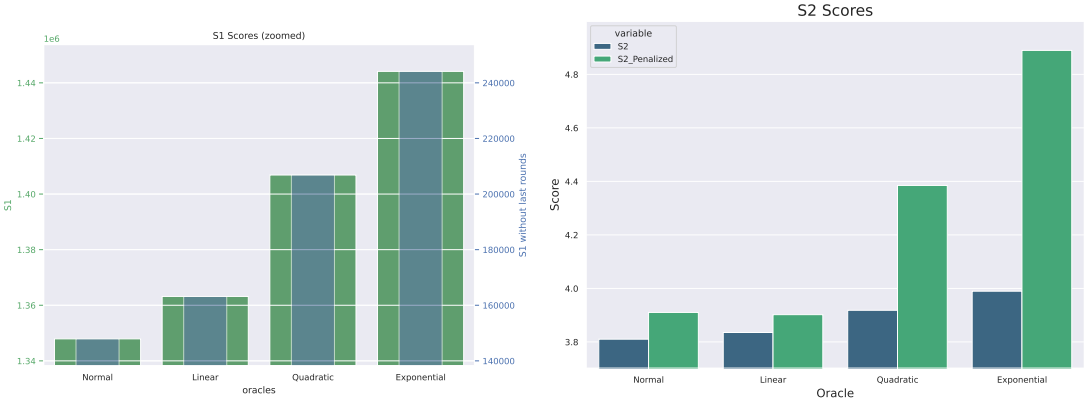


Figure 5.40: *Random Wp Method scores for DTLS models of sizes 60-80.*

Sizes 60–80 The results for sizes 60–80 are shown in Fig. 5.40. The "Normal" variant performs the least amount of queries and is almost as error prone than the "Linear" variant that performs slightly more queries. The least and greatest percentage extra queries is:

$$\Delta Q = \frac{16 \times 10^4 - 15 \times 10^4}{15 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{1}{15} = 6.66\%} \tag{5.52}$$

$$\Delta Q = \frac{24 \times 10^4 - 15 \times 10^4}{15 \times 10^4} \Rightarrow \boxed{\Delta Q = \frac{9}{15} = 60\%} \tag{5.53}$$

Table 5.15: *The number of failures for the DTLS models of size 60-80 for the Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
pion_ecdhe_cert_req	66	0	0	0	0
pion_ecdhe_cert_nreq	66	0	0	0	0
pion_ecdhe_cert_none	66	0	0	0	0
jsse-12_rsa_cert_none	79	3	2	14	27

Table 5.15 shows the failures of the variants for each model. All of the failures are caused by one model, "jsse-12_rsa_cert_none", which also causes the "Exponential" variant to fail almost every time.

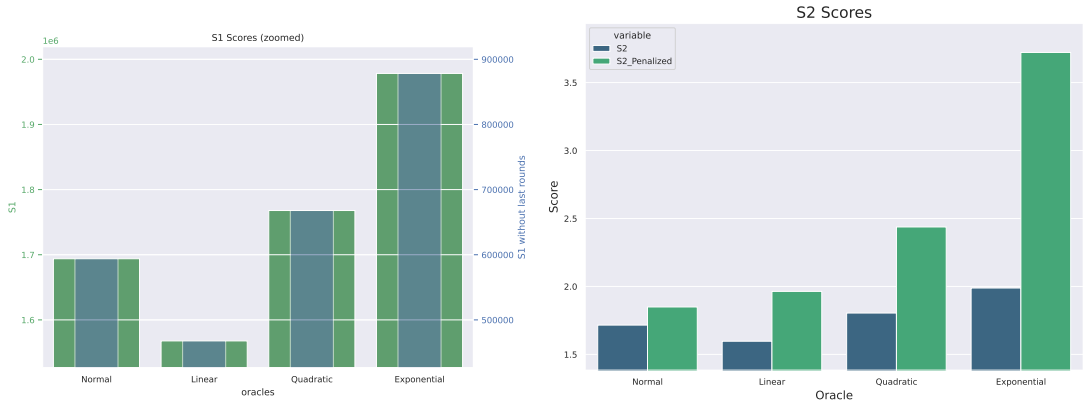


Figure 5.41: *Random Wp Method scores for DTLS models of sizes 100-120.*

Sizes 100–120 The results for sizes 100–120 are shown in Fig. 5.41. The "Linear" variant is the top performer in both scores but is more error prone than the "Random" variant as the Table 5.16 shows. The greatest percentage of saved and extra queries is:

$$\Delta Q = \frac{6 \times 10^5 - 4.5 \times 10^5}{6 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{1.5}{6} = 25\%} \tag{5.54}$$

$$\Delta Q = \frac{8.75 \times 10^5 - 6 \times 10^5}{6 \times 10^5} \Rightarrow \boxed{\Delta Q = \frac{2.75}{6} = 45.83\%} \tag{5.55}$$

Table 5.16: *The number of failures for the DTLS models of size 100-120 for the Random Wp Method variants.*

Model	Size	Random	Linear	Square	Exp.
jsse-12_rsa_cert_req	113	1	8	12	29
jsse-12_rsa_cert_nreq	105	3	3	7	23

Table 5.17: *Random WpMethod variants performance summary*

Protocol	Greatest Extra	Least Extra	Least Saved	Greatest Saved
TLS	0 %	0 %	0 %	0 %
MQTT	■	■	16.6 %	25%
TCP	105.5 %	33.3 %	■	■
DTLS	81.81 %	18.18 %	■	■
DTLS 10 20	50 %	34.61 %	■	■
DTLS 20 30	237.5 %	125 %	■	■
DTLS 40 50	33.9 %	■	■	3.57 %
DTLS 60 80	60 %	6.66 %	■	■
DTLS 100 120	45.83 %	■	■	25 %

Table 5.17 summarizes the results of the Random WpMethod variants, showcasing the greatest and least extra queries, as well as the least and greatest saved queries with respect to the "Normal" variant. Most of the variants perform worse than the "Normal" one

by a large margin. The MQTT experiments are the only ones that consistently outperform the “Normal” variant.

5.5 Summary

The results of the experiments show that focusing on the new states of a hypothesis during counterexample search can lead to better performance in some cases, if it is done moderately, but it can also lead to worse performance, if it is done too aggressively. More sophisticated approaches to the idea are needed and model inspection to determine why the algorithms perform the way they do is necessary.

Chapter 6

Future Work

Future work could follow many directions. The Stochastic State Coverage oracle could be put to the test with a wider variaty of probability functions, or even with a probability function that is learned during learning, similar in spirit as in the Mixture of Experts approach [18].

The implementations of the W / Wp Method and its variants depend a lot on how the local and global characterizing sets are constructed. It would be of interest to see how the algorithms and their variants perform with different characterizing sets.

Another idea would be to test newer states but to do so under some condition depending on the properties of the hypothesis graph. For example, it was found during experiments that if a sink state made its way into the hypothesis, then it remained a sink state for the rest of the learning experiment. Therefore, it would make sense to try skipping queries that target the sink state.

The same benchmarks could also be run with more learning algorithms than just L^* or with different counterexample processing methods. Moreover, equivalence oracles could be proposed that are not completely decoupled from the learning algorithm but are aware of it and utilize its characteristics to find counterexamples faster.

Finally, in the spirit of propagating knowledge from previous learning rounds to future ones, it would be interesting to prove, or consider proven after some rounds, properties of the hypothesis graph and use them to guide the learning process.

Bibliography

[1] Fides Aarts et al. “Improving active Mealy machine learning for protocol conformance testing”. In: *Machine learning* 96 (2014), pp. 189–224.

[2] Bernhard K Aichernig and Martin Tappler. “Efficient active automata learning via mutation testing”. In: *Journal of Automated Reasoning* 63 (2019), pp. 1103–1134.

[3] Bernhard K Aichernig, Martin Tappler, and Felix Wallner. “Benchmarking Combinations of Learning and Testing Algorithms for Automata Learning”. In: *Formal Aspects of Computing* 36.1 (2024), pp. 1–37.

[4] Dana Angluin. “Learning regular sets from queries and counterexamples”. In: *Information and computation* 75.2 (1987), pp. 87–106.

[5] Tsun S. Chow. “Testing software design modeled by finite-state machines”. In: *IEEE transactions on software engineering* 3 (1978), pp. 178–187.

[6] Joeri de Ruiter and Erik Poll. “Protocol State Fuzzing of TLS Implementations”. In: *24th USENIX Security Symposium*. Washington, D.C., USA: USENIX Association, Aug. 2015, pp. 193–206. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter.

[7] Margarita Dorofeeva and Irina Koufareva. “Novel modification of the W-method”. In: *Bulletin of the Novosibirsk Computing Center. Series: Computer Science* 2002 (2002), pp. 69–80.

[8] R Dorofeeva, K El-Fakih, and N Yevtushenko. “An improved FSM-based conformance testing method”. In: *Proc. of the IFIP 25th International Conference on Formal Methods for Networked and Distributed Systems*, pp. 204–218.

[9] Paul Fiterau-Brostean et al. “DTLS-Fuzzer: A DTLS Protocol State Fuzzer”. In: *15th IEEE Conference on Software Testing, Verification and Validation*. ICST 2022. Valencia, Spain: IEEE, Apr. 2022, pp. 456–458. DOI: 10.1109/ICST53961.2022.00051. URL: https://doi.org/10.1109/ICST53961.2022.00051.

[10] Paul Fiterău-Broștean, Ramon Janssen, and Frits W. Vaandrager. “Combining Model Learning and Model Checking to Analyze TCP Implementations”. In: *Computer Aided Verification - 28th International Conference, CAV 2016, Proceedings, Part II*. Vol. 9780. LNCS. Springer, 2016, pp. 454–471. DOI: 10.1007/978-3-319-41540-6_25. URL: https://doi.org/10.1007/978-3-319-41540-6%5C_25.

[11] Paul Fiterău-Broștean et al. “Analysis of DTLS Implementations Using Protocol State Fuzzing”. In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2523–2540. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/fiterau-brostean.

[12] Markus Theo Frohme. “Active automata learning with adaptive distinguishing sequences”. In: *arXiv preprint arXiv:1902.01139* (2019).

[13] Bharat Garhewal and Carlos Diego N Damasceno. “An experimental evaluation of conformance testing techniques in active automata learning”. In: *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2023, pp. 217–227.

[14] Robert M Hierons and Uraz Cengiz Türker. “Incomplete distinguishing sequences for finite state machines”. In: *The Computer Journal* 58.11 (2015), pp. 3089–3113.

[15] Malte Isberner, Falk Howar, and Bernhard Steffen. “The open-source learnLib: a framework for active automata learning”. In: *Computer Aided Verification: 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I 27*. Springer. 2015, pp. 487–495.

[16] Fujiwara Bochmann Khendek et al. “Test selection based on finite state models”. In: *IEEE Transactions on software engineering* 17.591-603 (1991), pp. 10–1109.

[17] George Klees et al. “Evaluating fuzz testing”. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 2123–2138.

[18] Loes Kruger, Sebastian Junges, and Jurriaan Rot. “Small test suites for active automata learning”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2024, pp. 109–129.

[19] David Lee and Mihalis Yannakakis. “Testing finite-state machines: State identification and verification”. In: *IEEE Transactions on computers* 43.3 (1994), pp. 306–320.

[20] Gang Luo, Alexandre Petrenko, and Gregor v Bochmann. “Selecting test sequences for partially-specified nondeterministic finite state machines”. In: *Protocol Test Systems: 7th workshop 7th IFIP WG 6.1 international workshop on protocol text systems*. Springer. 1995, pp. 95–110.

[21] Edi Muškardin et al. “AALpy: an active automata learning library”. In: *Innovations in Systems and Software Engineering* 18.3 (2022), pp. 417–426.

[22] Ronald L Rivest and Robert E Schapire. “Inference of finite automata using homing sequences”. In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. 1989, pp. 411–420.

[23] Adenilso Simao, Alexandre Petrenko, and Nina Yevtushenko. “On reducing test length for FSMs with extra states”. In: *Software testing, verification and reliability* 22.6 (2012), pp. 435–454.

[24] Wouter Smeenk et al. “Applying automata learning to embedded control software”. In: *Formal Methods and Software Engineering: 17th International Conference on Formal Engineering Methods, ICFEM 2015, Paris, France, November 3-5, 2015, Proceedings 17*. Springer. 2015, pp. 67–83.

[25] Juraj Somorovsky. “Systematic Fuzzing and Testing of TLS Libraries”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. Vienna, Austria: ACM, 2016, pp. 1492–1504. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978411. URL: http://doi.acm.org/10.1145/2976749.2978411.

[26] Michal Soucha and Kirill Bogdanov. “SPYH-method: an improvement in testing of finite-state machines”. In: *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE. 2018, pp. 194–203.

[27] Michal Soucha and Kirill Bogdanov. “State identification sequences from the splitting tree”. In: *Information and Software Technology* 123 (2020), p. 106297.

[28] Martin Tappler, Bernhard K. Aichernig, and Roderick Bloem. “Model-Based Testing IoT Communication via Active Automata Learning”. In: *Software Testing, Verification and Validation, (ICST) 2017 IEEE International Conference on*. Tokyo, Japan: IEEE Computer Society, Mar. 2017, pp. 276–287. DOI: 10.1109/ICST.2017.32. URL: https://doi.org/10.1109/ICST.2017.32.

[29] Frits W. Vaandrager. “Model learning”. In: *Commun. ACM* 60.2 (2017), pp. 86–95. DOI: 10.1145/2967606. URL: https://doi.org/10.1145/2967606.

List of Abbreviations

AAL	Active Automata Learning
ADS	Adaptive Distinguishing Sequences
APFD	Average Percentage of Faults Detected
ASI	Access-Step-Identify
EQ	Equivalence Query
FSM	Finite State Machine
H-ADS	Hybrid Adaptive Distinguishing Sequences
HSI	Harmonized State Identifiers
I-ADS	Incomplete Adaptive Distinguishing Sequences
MAT	Minimal Adequate Teacher
MQ	Membership Query
PAC	Probably Approximately Correct
PDS	Preset Distinguishing Sequences
SUL	System Under Learning