System verification with Model Checking

Λεωνίδας Αβδελάς | ΑΜ: 03113182

Εισαγωγή - Πρόβλημα

Σε εφαρμογές όπως safety-critical embedded systems [3], όπως ένας βηματοδότης, ένα λάθος στον σχεδιασμό είτε του software είτε του hardware, μπορεί να κοστίσει πολύ ακριβά για τον σχεδιαστή, τόσο σε οικονομικά μεγέθη, αλλά ακόμα και σε ανθρώπινες ζωές. Για να αποφευχθούν τέτοια λάθη, η βιομηχανία, αλλά και πολλοί επιστήμονες προσπαθούν να εντοπίζουν τα σφάλματα νωρίς στην διαδικασία σχεδιασμού, ακόμα και πριν την δημιουργία του συστήματος, καθώς αυτό εξασφαλίζει μικρότερα κόστη, οικονομικά και χρονικά.

Η μέθοδος που χρησιμοποιείται στην βιομηχανία περισσότερο αυτή τη στιγμή είναι αυτή του simulation και testing, δοκιμάζοντας διάφορες καταστάσεις που μπορεί να βρεθεί το σύστημα. Η μέθοδος της προσομοίωσης δοκιμάζει διάφορες εισόδους σε μια αφαιρετική μορφή του μοντέλου ή του συστήματος και η μέθοδος του testing δοκιμάζει διάφορες εισόδους στο πραγματικό σύστημα και παρατηρεί τις εξόδους. Δυστυχώς όταν το σύστημα φτάνει ένα στάδιο ωριμότητας ή όταν το μέγεθος του είναι πολύ μεγάλο, η χρήση αυτών των μεθόδων γίνεται όλο και λιγότερο αποδοτική, ειδικά για τον έλεγχο όλων των πιθανών καταστάσεων, αφού ότι οι καταστάσεις στις οποίες μπορεί να βρεθεί το σύστημα είναι πάρα πολλές και οι μέθοδοι δεν προσφέρουν κάποια ακριβή ένδειξη πόσα σφάλματα παραμένουν στο σύστημα.

Έτσι ανέκυψε η ανάγκη για να χρησιμοποιηθεί μια μέθοδος που θα αποδείκνυε μαθηματικά την ορθότητα του συστήματος. Ένας τύπος μεθόδων που κερδίζουν έδαφος τα τελευταία χρόνια είναι οι formal methods, οι οποίες είναι ένα σύνολο τεχνικών με βάση τα μαθηματικά για την μοντελοποίηση, ανάλυση και επαλήθευσή των συστημάτων [2].

Αυτές οι μέθοδοι εκτελούν εξαντλητική ανάλυση όλων των συμπεριφορών που μπορεί να έχει το σύστημα,

Αρχικά αυτοί οι έλεγχοι γίνονταν με το χέρι για τα πολύ ςριτιςαλ συστήματα, αλλά με την πάροδο του χρόνου, δημιουργήθηκαν σοφτωαρε που θα έκαναν την ίδια δουλεία.

Οι formal methods χρησιμοποιούνται κυρίως στο βήμα το σχεδιασμού του συ-

στήματος, αλλά και σε μετέπειτα φάσεις. Κάποιες από τις πιο επιτυχημένες formal methods είναι το model checking, το abstract interpretation, το equivalence checking και το verification by deduction. Όλες αυτές οι μέθοδοι αρχικά αναπτύχθηκαν από την ακαδημαϊκή κοινότητα και τα τελευταία χρόνια αρχίζουν και χρησιμοποιούνται όλο και περισσότερο στην βιομηχανία, ειδικά σε συστήματα που η κρισιμότητα τους και το κόστος αποτυχίας είναι μεγάλο. Σε αυτήν την εργασία θα ασχοληθούμε με την μέθοδο του **model checking**, αλλά θα εξετάσουμε συνοπτικά πώς δουλεύουν και οι άλλες μέθοδοι.

Abstract Interpretation

Σε αυτή την μέθοδο, στόχος είναι ο υπολογισμός ιναριαντς, συνθηκών που θα ισχύουν κάθε φορά που θα λειτουργεί το σύστημα, ανεξάρτητα της εισόδου του. Για παράδειγμα, θα μπορούσε σε ένα πρόγραμμα, η ανάλυση με αυτή την μέθοδο να καταλήγει στο συμπέρασμα ότι η τιμή μιας μεταβλητής είναι πάντα 5.

Model Checking

Σε αυτή τη μέθοδο, ο χρήστης παρέχει ένα μοντέλο (ή ένα σύστημα) και τον προσδιορισμό λειτουργίας του, καθώς και τα δεδομένων εισόδου και η μέθοδος αποφαίνεται αν μπορεί να υπάρξει κάποιο πιθανό λάθος ή γίνεται επιτυχημένος έλεγχος λειτουργίας.

Equivalence checking

Σε αυτή την μέθοδο, δύο μοντέλα συγκρίνονται μεταξύ τους για να βρεθεί πόσο όμοια συμπεριφέρονται κάτω από διάφορες συνθήκες.

Verification by Deduction

Σε αυτή την μέθοδο, η ιδιότητα του συστήματος είτε αποδεικνύεται με κάποιας μορφής απόδειξη ή αποδεικνύεται ότι η ιδιότητα δεν ισχύει. Σε αυτή την μέθοδο ο χρήστης πρέπει να παρέχει ιναριαντς σε κάποια σημεία της λειτουργίας του συστήματος. Καθώς η απόδειξη μιας ιδιότητας μπορεί να πάρει πολύ καιρό, συνήθως χρησιμοποιείται μόνο στις πιο ςριτιςαλ ιδιότητες των συστημάτων και μπορεί να χρησιμοποιηθεί και για συστήματα που έχουν άπειρες καταστάσεις.

Model Checking

Το υπόλοιπο της εργασίας βασίζεται στο [1].

Το model checking χρησιμοποιείται για συστήματα που έχουν πεπερασμένο αριθμό καταστάσεων, με αποτέλεσμα να μπορεί η αναζήτηση να γίνει πλήρως αυτοματοποιημένα σε αντίθεση με το verification by deduction. Η διαδικασία θα ελέγξει

όλες τις καταστάσεις και θα καταλήξει αν ο προσδιορισμός που τέθηκε αρχικά ισχύει ή όχι. Το model checking έχει δύο βασικά πλεονεκτήματα έναντι των άλλων μεθόδων:

1. Είναι πλήρως αυτόματο και δεν χρειάζεται επίβλεψη ή γνώση λογικής και απόδειξης θεωρημάτων.
2. Όταν ο σχεδιασμός αποτύχει η διαδικασία του model checking επιστρέφει ένα αντιπαράδειγμα που είναι εξαιρετικά χρήσιμο στην διαδικασία εύρεσης σφαλμάτων. Το πρόβλημα που αντιμετωπίζει το model checking είναι όταν ο χώρος τον πιθανών καταστάσεων επεκτείνεται σε πολύ μεγάλο βαθμό λόγω πολλών υποσυστημάτων που δρουν ταυτόχρονα. Τότε, ο χρόνος που χρειάζεται η μέθοδος αυξάνεται εκθετικά και, ανάλογα το μέγεθος, μπορεί να χρειάζεται υπερβολικά πολύ χρόνο για να τερματίσει.

Ο ταυτοχρονισμός υποσυστημάτων είναι ένα από τα μεγαλύτερα προβλήματα που αντιμετωπίζει η μέθοδος και είναι ένα πρόβλημα που θα μας απασχολήσει αρκετά και σε αυτή την εργασία.

Διαδικασία model checking

Θα αναφέρουμε περιληπτικά τα βήματα που χρειάζονται να γίνουν κατά την διαδικασία του model checking πριν τα εξετάσουμε πιο διεξοδικά.

Modelling: Το πρώτο βήμα είναι η μετατροπή του συστήματος μας σε ένα μοντέλο που μπορεί να χρησιμοποιηθεί από το εργαλείο μας. Σε αυτό το σημείο μπορούμε να αφαιρέσουμε πληροφορίες που είναι άχρηστες ή μη-σχετικές με την διαδικασία που εξετάζουμε.

Specification: Πέρα από το μοντέλο, σαν είσοδο στο εργαλείο προσθέτουμε και τις ιδιότητες που θέλουμε το μοντέλο μας να πληροί. Συνήθως αυτές έχουν κάποια χρονική λογική, για να ελέγξουμε την συμπεριφορά του συστήματος σε μια περίοδο χρόνου. Ένα πρόβλημα που αντιμετωπίζεται εδώ είναι το πρόβλημα της πληρότητας. Μέσω του model checking μπορούμε να επαληθεύσουμε ότι το μοντέλο μας πληροί τα specifications, αλλά είναι αδύνατο να εξακριβώσουμε ότι το specification καλύπτει όλες τις ιδιότητες που θα έπρεπε να πληροί το σύστημα.

Verification: Είναι η διαδικασία ελέγχου του μοντέλου με βάση το specification. Ιδανικά η διαδικασία θα είναι πλήρως αυτόματη, αλλά συνήθως χρειάζεται η συμβολή του χρήστη, ειδικά στην ανάλυση των αποτελεσμάτων της διαδικασίας.

Modelling

Στην εργασία θα ασχοληθούμε με reactive συστήματα. Αυτά τα συστήματα αλληλοεπιδρούν συχνά με το περιβάλλον και συνήθως δεν τερματίζουν. Για να μπορέσουμε να τα μοντελοποιήσουμε, θα πρέπει να καταγράψουμε τις καταστάσεις (states) του και πώς αυτές αλλάζουν όταν εκτελούνται κάποιες ενέργειες. Έτσι ένα ζευγάρι καταστάσεων (πριν και μετά) ονομάζεται μετάβαση (transition) του

συστήματος και ο συνολικός υπολογισμός μπορεί να οριστεί με βάση τις μεταβάσεις. Υπολογισμός (computation) ονομάζεται ένα άπειρο σύνολο από καταστάσεις, όπου κάθε κατάσταση προέρχεται από την προηγούμενη με βάση κάποια μετάβαση.

Για να απεικονίσουμε αυτές τις μεταβάσεις, θα χρησιμοποιήσουμε ένα γράφο μεταβάσεων που ονομάζεται **Kripke structure**. Έστω AP ένα σύνολο από ατομικούς όρους (οι ιδιότητες που θέλουμε να έχει το σύστημα μας σε διάφορες καταστάσεις). Ως δομή Kripke ορίζουμε την τετράδα M= (S, S_0, R, L), όπου:

1. S είναι ένα πεπερασμένο σύνολο από καταστάσεις.
2. $S_0 \subseteq S$ είναι το σύνολο των αρχικών καταστάσεων.
3. $R \subseteq S \times S$ είναι οι σχέσεις μεταβάσεων. Οι σχέσεις αυτές πρέπει να είναι ολικές, δηλαδή για κάθε κατάσταση $s \in S$, υπάρχει μια κατάσταση $s' \in S$, έτσι ώστε $\left(s, s'\right) \in R$.
4. $L \cdot S \rightarrow 2^{AP}$ μια συνάρτηση που αντιστοιχεί κάθε κατάσταση με τις ατομικές προτάσεις που ισχύουν σε αυτή.

Κάποιες φορές οι αρχικές καταστάσεις δεν θα μας ενδιαφέρουν, οπότε και θα τις παραλείπουμε από τον ορισμό. Ένα μονοπάτι στην δομή M από την κατάσταση s είναι μια άπειρη ακολουθία καταστάσεων $\pi = s_0 s_1 s_2 \ldots$ τέτοια ώστε $s_0 = s$ και υπάρχει το $R\left(s_i, s_{i+1}\right), \forall i \leq 0$.

Για να περιγράψουμε ταυτόχρονα συστήματα, συστήματα δηλαδή στα οποία παραπάνω από μια αλλαγή συμβαίνει ταυτόχρονα, θα χρησιμοποιήσουμε κατηγορηματική λογική. Έστω $V = u_1, \ldots u_n$ το σύνολο των μεταβλητών του συστήματος. Υποθέτουμε ότι οι μεταβλητές στο V εκτείνονται πάνω στο πεπερασμένο σύνολο D, το οποίο κάποιες φορές καλείται το *πεδίο ορισμού ή σύμπαν* της ερμηνείας. Μια εκτίμηση για το V είναι μια συνάρτηση που συσχετίζει μια τιμή στο D με κάθε μεταβλητή u στο V.

Μια κατάσταση σε ένα ταυτόχρονο σύστημα μπορεί να περιγραφεί δίνοντας τιμές σε όλα τα στοιχεία του V. Με άλλα λόγια, μια κατάσταση, είναι μια εκτίμηση $s.V \rightarrow D$ για το σύνολο μεταβλητών του V. Αν γνωρίζουμε την εκτίμηση, μπορούμε να γράψουμε ένα τύπο για την εκτίμηση αυτή. Για παράδειγμα, αν έχουμε $V = u_1, u_2, u_3$ και την εκτίμηση $\langle u_1 \leftarrow 2, u_2 \leftarrow 3, u_3 \leftarrow 5\rangle$, καταλήγουμε στον τύπο $(u_1 = 2) \wedge (u_2 = 3) \wedge (u_3 = 5)$. Έτσι για παράδειγμα, οι αρχικές καταστάσεις του συστήματος μπορούν να περιγραφούν από τον τύπο S_0 πάνω στις μεταβλητές του V.

Πέρα από τις καταστάσεις, πρέπει να μπορούμε να περιγράψουμε και σύνολα μεταβάσεων μεταξύ καταστάσεων. Θα επεικτείνουμε την παραπάνω ιδέα. Έστω έχουμε δύο σύνολα μεταβλητών συστήματος V και V'. Οι μεταβλητές στο V θα είναι οι μεταβλητές της τωρινής κατάστασης και οι μεταβλητές στο V' θα είναι της επόμενης κατάστασης. Κάθε μεταβλητή u στο V θα έχει μια αντίστοιχη μεταβλητή στο V', την οποία την ορίζουμε ως u'. Μια εκτίμηση των μεταβλητών στο V και το V' μπορεί να θεωρηθεί αν ορίσουμε ένα διατεταγμένο ζεύγος καταστάσεων ή

μετάβαση. Χρησιμοποιώντας τύπους όπως πριν και ορίζουμε ως το σύνολο των ζεύγων καταστάσεων ως σχέση μετάβασης (transition relation). Αν το R είναι μια σχέση μετάβασης, τότε γράφουμε $R(V, V')$ για να το περιγράψουμε.

Για να γράψουμε specifications που περιγράφουν ιδιότητες των σύγχρονων συστημάτων, πρέπει πρώτα να ορίσουμε ένα σύνολο ατομικών προθέσεων AP. Οι ατομικές προθέσεις συνήθως θα έχουν την μορφή $u = d$, όπου $u \in V$ και $d \in D$. Μια πρόθεση $u = d$ θα είναι αληθής σε μια κατάσταση s αν $s(u) = d$. Όταν η u είναι μια μεταβλητή στο πεδίο ορισμού True, False, δεν χρειάζεται να συμπεριλάβουμε και το $u =$ **True** και το $u =$ False στο AP. Θα γράφουμε u για να δείξουμε ότι $s(u) =$ Τρυε και $\neg u$ για να δείξουμε ότι $s(u) =$ False.

Τώρα μπορούμε να εξάγουμε μία δομή Kripke $M = (S, S_0, R, L)$ από τους τύπους κατηγορηματικής λογικής S_0 και R για να αναπαραστήσουν το ταυτόχρονο σύστημα.

- Το σύνολο των καταστάσεων S είναι το σύνολο όλων των εκτιμήσεων στο V
- Το σύνολο των αρχικών καταστάεων S_0 είναι το σύνολο όλων των εκτιμήσεων s_0 στο V που ικανοποιούν τον τύπο S_0
- Έστω s και s' δύο καταστάσεις, τότε το $R(s, s')$ ισχύει αν το R εκτιμάται σε True όταν κάθε $u \in V$ ανατίθεται σε τιμή $s(u)$ και κάθε $u' \in V'$ ανατείθεται στην τιμή $s'(u)$.
- Η συνάρτηση ετικέτας $LS \to 2^{AP}$ ορίζεται έτσι ώστε $L(s)$ είναι το υποσύνολο όλων των ατομικών όρων που είναι αληθείς στο s. Αν το u είναι μια boolean μεταβλητή, τότε το $u \in L(s)$ μας δείχνει ότι $s(u) =$ True και $u \notin L(s)$ ότι s(u) = False.

Επειδή απαιτούμε ότι η μεταβατική σχέση μιας δομής Kripke είναι πάντα ολική, πρέπει να επεκτείνουμε την σχέση R, αν κάποιες καταστάσεις δεν έχουν διάδοχο. Σε αυτή την περίπτωση τροποποιούμε το R, ώστε να ισχύει το $R(s, s)$.

Παράδειγμα

Έστω ένα απλό σύστημα με μεταβλητές x και y με τιμές στο $D = 0, 1$. Έτσι η εκτίμηση των μεταβλητών x και y είναι ένα ζεύγος $(d_1, d_2) \in D \times D$, όπου d_1 είναι η τιμή για το x και d_2 η τιμή για το y. Το σύστημα έχει μια μετάβαση

$$x = (x + y) \mod 2$$

και ξεκινάει από την κατάσταση στην οποία $x = 1, y = 1$. Το σύστημα περιγράφεται από δύο τύπους κατηγορηματικης λογικής. Το σύνολο των αρχικών καταστάσεων είναι το $S_0(x, y) = x = 1 \wedge y = 1$ και η μετάβαση είναι $R(x, y, x', y') = x' = (x+y) \mod 2 \wedge y' = y$. Η δομή Kripke που εξάγεται από αυτούς τους τύπους είναι:

- $S = D \times D$
- $S_0 = (1, 1)$

- $R = ((1,1),(0,1)),((0,1),(1,1)),((1,0),(1,0)),((0,0),(0,0))$
- $L((1,1)) = x = 1, y = 1$, $L((0,1)) = x = 0, y = 1$, $L((1,0)) = x = 1, y = 0$, $L((0,0)) = x = 0, y = 0$

Το μοναδικό μονοπάτη σε αυτή την δομή Kripke που ξεκινά από αρχική κατάσταση είναι το $(1,1)(0,1)(1,1)(0,1)\ldots$. Αυτό το μονοπάτι είναι ο μόνος υπολογισμός του συστήματος.

Ένα ακόμα πράγμα που πρέπει να προσέξουμε είναι η λεπτομέρεια των μεταβάσεων. Αν είναι πολύ τραχειές coarse, τότε η δομή μας μπορεί να μην περιλαμβάνει κάποιες καταστάσεις που είναι παρατηρήσημες. Ως αποτέλεσμα ο αλγόριθμος μας μπορεί να αποτύχει στο να βρεί σημαντικά λάθη. Αν από την άλλη είναι πολύ λεπτές fine, τότε το πρόβλημα είναι ότι μπορεί να δημιουργηθούν καταστάσεις που δεν υπάρχουν στην πραγματικότητα και να υπάρξουν σφάλματα που δεν θα συμβούν ποτέ στην πραγματικότητα.

Ταυτόχρονα Συστήματα

Ένα ταυτόχρονο σύστημα αποτελείται από ένα σύνολο από εξαρτήματα που εκτελούνται ταυτόχρονα. Κανονικά αυτά τα εξαρτήματα έχουν κάποιον τρόπο επικοινωνίας μεταξύ τους. Θεωρούμε δύο τρόπους εκτέλεσης: Ασύγχρονο, οπού ένα εξάρτημα κάνει βήμα κάθε φορά και σύχρονο, όπου όλα τα εξαρτήματα κάνουν ένα βήμα κάθε φορά. Ακόμα θα ξεχωρίσουμε τρείς τρόπους επικοινωνίας. Τα εξαρτήματα είτε επικοινωνούν με την αλλαγή κάποιας κοινής μεταβλητής, είτε ανταλάσσοντας μηνυνήματα ή με κάποιο προτόκολλο χειραψίας.

Χρονικές Λογικές

Η λογική CTL* (Computation Tree Logic *) χρησιμποιήται για να περιγράψει σειρές από μεταβάσεις μεταξύ καταστάσεων σε ένα reactive σύστημα. Ο χρόνος δεν αναφέρεται ρητά, αντίθετα ένας τύπος μπορεί να ορίζει ότι τελικά θα φτάσουμε κάποια κατάσταση ή δεν φτάνουμε ποτέ σε μια σφαλματική κατάσταση. Για να περιγράψουμε αυτά τα πράγματα, χρησιμοποιούμε χρονικούς τελεστές. Αυτοί μπορούν να συνδιαστούν και με boolean τελεστές ή να ενφωλιαστούν αυθαίρετα.

Ου τύποι της CTL* περιγράφουν υπολογιστικά δέντρα. Το δέντρο δημιουργήται ορίζοντας ως μια κατάσταση σε μια δομή Kripke ως αρχική και ξετυλίγοντας την δομή σε ένα άπειρο δέντρο με την αρχική κατάσταση ως ρίζα. Το δέντρο δείχνει όλους τους πιθανούς υπολογισμούς που ξεκινάνε από την ρίζα.

Οι τύποι δημιουργώνται χρησμοποιόντας ποσοδείκτες μονοπατίου και χρονικούς τελεστές.

Οι ποσοδείκτες μονοπατιού χρησιμοποιούνται για να περιγράψουν την δομή των διακλαδόσεων του δέντρου. Υπάρχουν δύο τέτοιοι ποσοδείκτες. Ο **A** (¨για όλα τα υπολογιστικά μονοπάτια¨) και ο **E** (¨για κάποιο υπολογιστικό μονοπάτι¨). Αυτοί χρησιμοποιούνται σε κάποια κατάσταση για να δείξουμε ότι όλα ή κάποια από τα μονοπάτια που ξεκινάνε σε αυτή την κατάσταση έχουν κάποια ιδιότητα.

Οι χρονικοί τελεστές περιγράφουν ιδιότητες ενός μονοπατιού στο δέντρο. Υπάρχουν πέντε βασικοί τελεστές:

- **X** (¨την επόμενη φορά¨) απαιτεί ότι μια ιδιότητα ισχύει στην δεύτερη κατάσταση του μονοπατιού.
- **F** (¨τελικά' ή ¨στο μέλλον¨) σημαίνει ότι η ιδιότητα θα ισχύει σε κάποια κατάσταση στο μονοπάτι.
- **G** (¨πάντα¨) σημαίνει ότι η ιδιότητα ισχύει σε κάθε κατάσταση στο μονοπατι.
- Το **U** (¨μέχρι¨) συνδιάζει δύο ιδιότητες. Ισχύει, αν υπάρχει μια κατάσταση στο μονοπάτι στην οποία ισχύει η δεύτερη ιδιότητα και σε κάθε προηγούμενη από αυτή κατάσταση, ισχύει η πρώτη ιδιότητα.
- **R** (¨απελευθέρωσε¨) είναι το λογικό ανάποδο του **U**. Απαιτεί ότι η δεύτερη ιδιότητα ισχύει κατα μήκος του μονοπατιού μέχρι την πρώτη κατάσταση που ισχύει η πρώτη ιδιότητα. Η διαφορά είναι ότι δεν υπάρχει κάποια εγγύηση ότι θα ισχύει κάποια στιγμή η πρώτη ιδιότητα.

Υπάρχουν δύο είδη τύπων στην $CTL*$: οι τύποι κατάστασης (που ισχύουν σε μια συγκεκριμένη κατάσταση) και οι τύποι μονοπατιού (που ισχύουν κατα μήκος ενός συγκεκριμένου μονοπατιού).

Έστω AP το σύνολο των ονομάτων των ατομικών προθέσεων. Το συντακτικό των τύπων κατάστασης δίνεται από τους παρακάτω κανόνες:

- Αν $p \in AP$, τότε το p είναι τύπος κατάστασης.
- Αν f και g είναι τύποι κατάστασης, τότε $\neg f$, $f \vee g$, $f \wedge g$ είναι τύποι κατάστασης.
- Αν f είναι τύπος μονοπατιού, τότε $\mathbf{E}f$ και $\mathbf{A}f$ είναι τύποι κατάστασης.

Χρειάζονται δύο ακόμη κανόνες για να περιγραφούν οι τύποι μονοπατιού.

- Αν f είναι τύπος κατάστασης, τότε το f είναι και τύπος μονοπατιού.
- Αν f και g είναι τύποι μονοπατιού, τότε $\neg f$, $f \vee g$, $f \vee g$, $\mathbf{X}f$, $\mathbf{F}f$, $\mathbf{G}f$, $f\mathbf{U}g$, $f\mathbf{R}g$ είναι τύποι μονοπατιού.

Η CTL* είναι το σύνολο των τύπων καταστάσεων που δημιουργείται από τους παραπάνω κανόνες.

Τώρα θα ορίσουμε την σημασιολογία του CTL* με βάση την δομή Kripke. Όπως έχει αναφερθεί, η δομή Kripke M είναι μια τριπλέτα $\langle S, R, L\rangle$, όπου S είναι το σύνολο των καταστάσεων, $R \subseteq S \times S$ είναι η σχέση των μεταβάσεων, η οποία πρέπει να είναι ολική και $L : S \to 2^{AP}$ είναι η συνάρτηση που κάνει αντιστοιχία κάθε κατάσταση με ένα σύνολο ατομικών προτάσεων που είναι αληθείς στην κατάσταση αυτή. Ένα μονοπάτι M είναι μια άπειρη ακολουθία καταστάσεων, $\pi = s_0 s_1 \ldots$, τέτοι ώστε για κάθε $i \geq 0, (s_i, s_{i+1}) \in R$. Χρησιμοποιούμε το π^i για να ορίσουμε την κατάληξη του π που ξεκινάει από το s_i.

Αν το f είναι ένας τύπος κατάστασης, τότε η σημειογραφία $M, s \models f$ συμβολίζει ότι το f ισχύει στην κατάσταση s στην δομή Kripke M. Παρόμοια, αν το f είναι ένας τύπος μονοπατιού, τότε η σημειογραφία $M, s \models f$ συμβολίζει ότι ο f ισχύει κατά μήκος του μονοπατίου π στην δομή Kripke M. Όταν η δομή M είναι ξεκάθαρη από τα συμφραζόμενα, την παραλείπουμε. Η σχέση $\models$ ορίζεται επαγωγικά όπως φαίνεται παρακάτω (θεωρώντας ότι f_1 και f_2 είναι τύποι κατάστασης και τα g_1 και g_2 είναι τύποι μονοπατιού):

1. $M, s \models p \leftrightarrow p \in L(s)$.
2. $M, s \models \neg f_1 \leftrightarrow M, s \not\models f_1$.
3. $M, s \models f_1 \vee f_2 \leftrightarrow M, s \models f_1$ ή $M, s \models f_2$.
4. $M, s \models f_1 \wedge f_2 \leftrightarrow M, s \models f_1$ και $M, s \models f_2$.
5. $M, s \models \mathbf{E}g_1 \leftrightarrow$ υπάρχει μονοπάτι π που ξεκινά από το s, τέτοιο ώστε $M, \pi \models g_1$.
6. $M, s \models \mathbf{A}g_1 \leftrightarrow$ για κάθε μονοπάτι π που ξεκινά από το s, ισχύει $M, \pi \models g_1$.
7. $M, \pi \models f_1 \leftrightarrow$ η s είναι η πρώτη κατάσταση του π και $M, s \models f_1$.
8. $M, \pi \models \neg g_1 \leftrightarrow M, \pi \not\models g_1$.
9. $M, \pi \models g_1 \vee g_2 \leftrightarrow M, \pi \models g_1$ ή $M, \pi \models g_2$.
10. $M, \pi \models g_1 \wedge g_2 \leftrightarrow M, \pi \models g_1$ και $M, \pi \models g_2$.
11. $M, \pi \models \mathbf{X}g_1 \leftrightarrow M, p^1 \models g_1$.
12. $M, \pi \models \mathbf{F}g_1 \leftrightarrow$ υπάρχει $k \geq 0$ τέτοι ώστε $M, \pi^k \models g_1$.
13. $M, \pi \models \mathbf{G}g_1 \leftrightarrow$ για κάθε $i \geq 0, M, p^i \models g_1$.
14. $M, \pi \models g_1 \mathbf{U} g_2 \leftrightarrow$ υπάρχει ένα $k \geq 0$ τέτοιο ώστε $M, \pi^k \models g_2$ και για κάθε $0 \leq j \leq k, M, \pi^j \models g_1$.
15. $M, \pi \models g_1 \mathbf{R} g_2 \leftrightarrow$ για κάθε $j \geq 0$, αν για κάθε $i \leq jM, \pi^i \not\models g_1$, τότε $M, \pi^j \models g_2$.

Είναι εύκολο να δούμε ότι οι τελεστές $\vee, \neg, \mathbf{X}, \mathbf{U}$ και $\mathbf{E}$ είναι αρκετοί για να εκφράσουμε κάθε τύπο CTL*.

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $f\mathbf{R}g \equiv \neg(\neg f \mathbf{U} \neg g)$
- $\mathbf{F}f \equiv \text{True}\mathbf{U}f$
- $\mathbf{G}f \equiv \neg\mathbf{F}\neg f$
- $\mathbf{A}(f) \equiv \neg\mathbf{E}(\neg f)$

Η λογική που θα χρησιμοποιήσουμε για τον αλγόριθμο μας είναι μια υπολογικής της CTL* και είναι η CTL (Computation Tree Logic). Η CTL είναι ένα υποσύνολο της CTL*, όπου κάθε ένας από τους χρονικούς τελεστές $\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}, \mathbf{R}$ πρέπει να

έχει ακριβώς πριν του έναν ποσοδείκτη μονοπατιού. Πιο συγκεκριμένα η CTL είναι το υποσύνολο της CTL* που παίρνουμε περιορίζοντας το συντακτικό των τύπων του μονοπατιού χρησιμοποιώντας τον παρακάτω κανόνα:

- Αν f και g είναι τύποι κατάστασης, τότε $\mathbf{X}f, \mathbf{F}f, \mathbf{G}f, f\mathbf{U}g, f\mathbf{R}g$ είναι τύποι μονοπατιών.

Υπάρχουν 10 βασικοί τελεστές στην CTL

- **AX** και **EX**
- **AF** και **EF**
- **AG** και **EG**
- **AU** και **EU**
- **AR** και **ER**

Κάθε ένας από αυτούς τους τελεστές μπορεί να εκφραστεί ως συνδιασμός των τελεστών $\mathbf{EX}, \mathbf{EG}, \mathbf{EU}$

- $\mathbf{AX}f = \neg\mathbf{EX}(\neg f)$
- $\mathbf{EF}f = \mathbf{E}[\text{True}\mathbf{U}f]$
- $\mathbf{AG}f = \neg\mathbf{EF}(\neg f)$
- $\mathbf{AF}f = \neg\mathbf{EG}(\neg f)$
- $\mathbf{A}[f\mathbf{U}g] \equiv \neg\mathbf{E}[\neg g\mathbf{U}(\neg f \wedge \neg g)] \wedge \neg\mathbf{EG}\neg g$
- $\mathbf{A}[f\mathbf{R}g] \equiv \neg\mathbf{E}[\neg f\mathbf{U}\neg g]$
- $\mathbf{E}[f\mathbf{R}g] \equiv \neg\mathbf{A}[\neg f\mathbf{U}\neg g]$

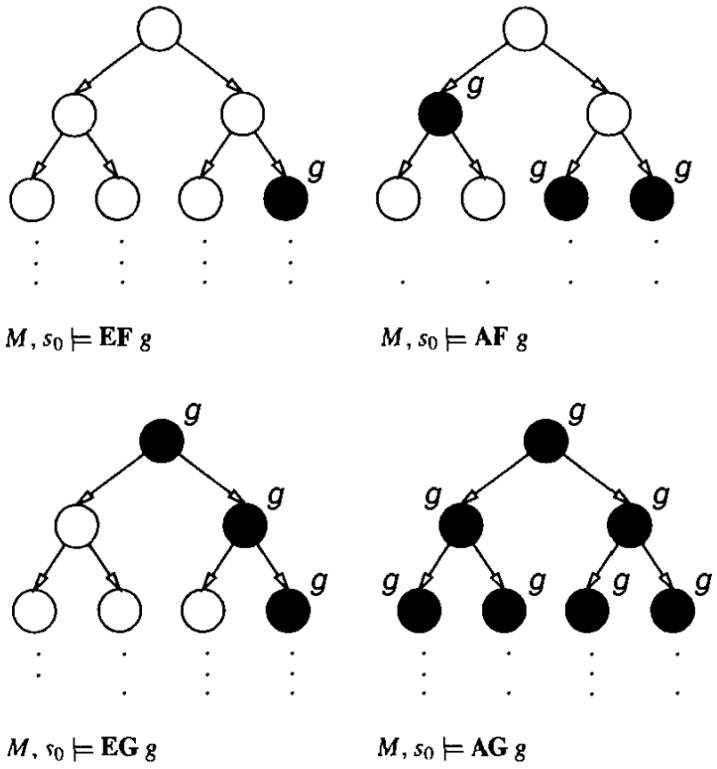
Οι τέσσερις τελεστές που χρησιμοποιούνται περισσότερο φαίνονται στο Σχήμα 1.

Κάποι συνήθεις CTL τύποι που μπορεί να προκύψουν κατά την επαλήθευση ενός πεπερασμένου σύγχρονου προγράμματος είναι παρακάτω:

- **EF**(Start $\wedge \neg$Ready): Είναι δυνατόν να πάμε σε μια κατάσταση που το Start ισχύει, αλλά το Ready δεν ισχύει.
- **AF**(Req $\rightarrow$ **AF**Ack): Αν υπάρξει κάποιο Request, τότε θα γίνει κάποια στιγμή Aknowledge
- **AG**(**AF**DeviceEnabled): Η πρόθεση DeviceEnabled ισχύει απείρως συχνά σε κάθε υπολογιστικό μονοπάτι.
- **AG**(**EF**Restart): Από κάθε κατάσταση είναι δυνατό να βρεθούμε στην κατάσταση Restart

Αλγόριθμος Model Checking

Το πρόβλημα είναι: Δεδομένης μιας δομής Kripke $M = (S, R, L)$ που αναπαριστά ένα σύχρονο σύστημα πεπερασμένων καταστάσεων και ένα τύπο χρονικής λογικής



Σχήμα 1: Βασικοί τελεστές της CTL

f, ο οποίος περιγράφει κάποιο επιθυμητό specification, βρες το σύνολο όλων των καταστάσεων στο S που ικανοποιούν το f:

$$s \in S | M, s \models f$$

Συνήθως, κάποιες καταστάσεις του συστήματος θεωρούνται αρχικές καταστάσεις. Το σύστημα ικανοποιεί το δωσμένο specification δεδομένου ότι όλες οι αρχικές καταστάσεις είναι στο σύνολο αυτό.

Ο αλγόριθμος θα λειτουργήσει αντιστοιχόντας σε κάθε κατάσταση s το σύνολο $\text{labels}(s)$ των υποτύπων του f που είναι αληθείς στο s. Αρχικά το $\text{labels}(s)$ είναι απλά $L(s)$. Ο αλγόριθμος περνάει από μια σειρά από βήματα. Κατά την διάρκεια του βήματος i, γίνετια επεξεργασία σε υποτύπους με $i-1$ ενφωλιασμένους τελεστές. Όταν γίνει η επεξεργασία ενός υποτύπου, προστίθεται στην αντιστοίχηση της κάθε κατάστασης στην οποία είναι αληθής. Όταν ο αλγόρθμος τερματίσει, θα έχουμε ότι $M, s \models f$ ανν $f \in \text{labels}(s)$.

Όπως αναφέραμε και παραπάνω, υπάρχουν έξι περιπτώσεις που πρέπει να καλύπτει ο αλγόριθμός μας (και μια που είναι η περίπτωση της ατομικής πρότασης), αφού από αυτές τις έξι $(\neg f_1, f_1 \vee f_2, \mathbf{EX} f_1, \mathbf{E}[f_1 \mathbf{U} f_2], \mathbf{EG} f_1$ μπορούμε να καλύψουμε όλες τις υπόλοιπες.

- Για τύπους που είναι ατομικές προτάσεις f, απλώς κάνουμε αντιστοίχηση όλες τις καταστάσεις για τις οποίες η f είναι αληθής.
- Για τύπους της μορφής $\neg f_1$, κάνουμε αντιστοίχηση τις καταστάσεις που δεν αντιστοιχούν στην f_1.
- Για το $f_1 \vee f_2$, κάνουμε αντιστοίχηση κάθε κατάσταση που είναι αντιστοιχισμένη με την f_1 ή την f_2.
- Για $\mathbf{EX} f_1$ κάνουμε αντιστοίχηση κάθε κατάσταση που έχει διάδοχο ο οποίος είναι αντιστοιχισμένος με την f_1.
- Για να διαχειριστούμε τύπους της μορφής $g = \mathbf{E}[f_1 \mathbf{U} f_2]$ θα βρούμε πρώτα όλες τις καταστάσεις που είναι αντιστοιχισμένες με την f_2. Μετά μετακινούμαστε προς τα πίσω χρησιμοποιώντας το αντίστροφο της σχέσης μετάβασης R και βρίσκουμε όλες τις καταστάσεις που μπορούν να προσεγγιστούν από ένα μονοπάτι όπου κάθε κατάσταση είναι αντιστοιχισμένη με το f_1. Όλες αυτές οι καταστάσεις πρέπει να αντιστοιχηστούν με g.
- Για την περίπτωση $g = \mathbf{EG} f_1$, πρέπει να αποσυνθέσουμε τον γράφο σε μη-τετριμμένες ισχυρες συνδεδεμένες συνιστώσες. Μια ισχυρά συνδεδεμένη συνιστώσα C είναι ένας μέγιστος υπογράφος, τέτοις ώστε κάθε κόμβος στον C προσεγγίζεται από κάθε άλλον κόμβο στο C κατά μήκος ενός κατευθυνόμενου μονοπατιού το οποίο περιέχεται εξόλοκλήρου μεσα στο C. Ο C είναι μη-τετριμμένος ανν είτε έχει πάνω από ένα κόμβο ή περιέχει ένα κόμβο με αυτό-βρόχο. Έστω M', το οποίο παίρνουμε από το M διαγράφοντας από το S όλες τις καταστάσεις στις οποίες ο f_1 δεν ισχύει και περιορίζοντας το R και το L αντίστοιχα. Έτσι $M' = (S', R', L')$ όπου $S' = s \in S | M, s \models f_1$,

$R' = R|_{S' \times S'}$ και $L' = L|_{S'}$. Το R' μπορεί να μην είναι ολικό σε αυτή την περίπτωση. Οι καταστάσεις που δεν έχουν εξερχόμενες μεταβάσεις μπορούν να διαγραφούν, αλλά αυτό δεν επηρεάζει την ορθότητα του αλγορίθμου. Ο αλγόριθμος βασίζεται στην παρατήρηση ότι: $M, s \models \mathbf{EG} f_1$ ανν ισχύουν οι παρακάτω δύο συνθήκες:

1. $s \in S'$
2. Υπάρχει μονοπάτι στο M' που οδηγεί από το s σε κάποιο κόμβο t σε μια μη-τετριμένη, ισχύρα συνδεδεμένη συνιστώσα C του γράφου (S', R').

Με βάση αυτό και την δομή M' που περιγράψαμε πριν, χωρίζουμε τον γράφο (S', R') σε ισχυρά συνδεδεμένες συνιστώσες χρησιμοποιώντας τον αλγόριθμο του Tarjan. Μετά βρίσκουμε τις καταστάσεις που ανήκουν σε αυτές τις μη-τετριμμένες συνιστώσες. Έπειτα, δουλεύομε προς τα πίσω χρησιμοποιώντας το αντίστροφο της R' και βρίσκουμε όλες τις καταστάσεις που μπορούν να προσεγγιστούν από ένα μονοπάτι όπου η κάθε κατάσταση είναι αντιστοιχισμένη με το f_1.

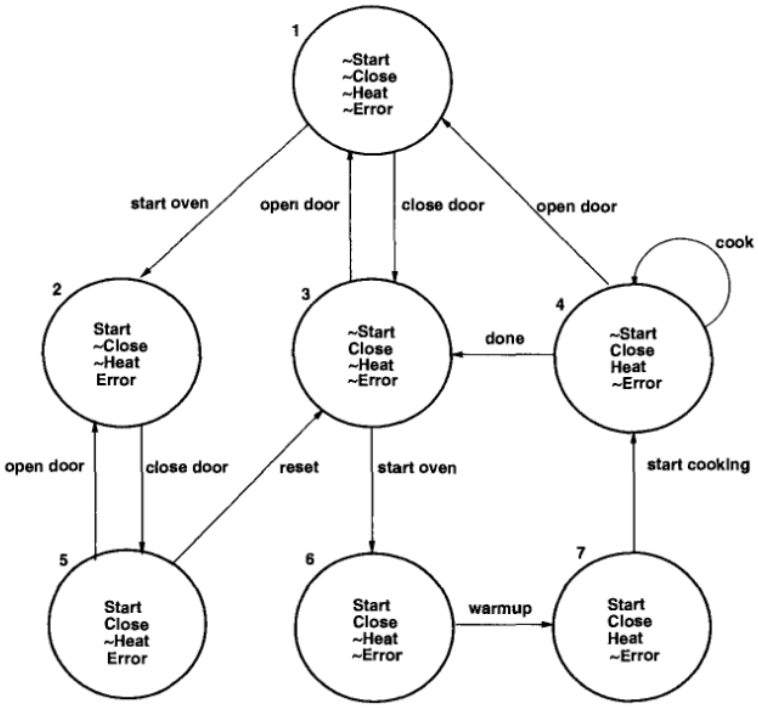
Για να διαχειριστούμε κάθε CTL τύπο f, εφαρμόζουμε διαδοχικά τον αλγόριθμο αντιστοίχησης στους υποτύπους του f, ξεκινώντας με το μικρότερο και πιο βαθεία εμφωλιασμένο και συνεχίζουμε προς τα έξω για να συμπεριλάβουμε όλο το f. Με αυτό τον τρόπο εγγυόμαστε ότι όποτε επεξεργαζόμαστε έναν υποτύπο του f, όλοι οι υποτύποι αυτού έχουν ήδη επεξεργαστεί.

Θα παρουσιάσουμε τον αλγόριθμο σε ένα μικρό παράδειγμα που περιγράφει την λειτουργία του φούρνου μικροκυμάτων. Στο Σχήμα 2, μπορούμε να δούμε την δομή Kripke για τον φούρνο. Για πληρότητα, σε κάθε κατάσταση φαίνονται οι ατομικές προτάσεις που ισχύουν σε αυτή, καθώς και οι αρνήσεις των προτάσεων που δεν είναι αληθής σε αυτή. Οι περιγραφές στις ακμές μας δείχνουν τις πράξεις που προκαλούν μεταβάσεις και δεν είναι μέρος της δομής.

Ελέγχουμε τον τύπο $\mathbf{AG}(\text{Start} \rightarrow \mathbf{AF}\text{Heat})$, το οποίο είναι ισοδύναμο με την φόρμουλα $\neg\mathbf{EF}(\text{Start} \wedge \mathbf{EG}\neg\text{Heat})$. (Χρησιμοποιήσαμε το $\mathbf{EF}f$ ως συντομογραφία του $\mathbf{E}[\text{true}\mathbf{U}f]$). Ξεκινάμε υπολογίζοντας το σύνολο καταστάσεων που ικανοποιούν τους ατομικούς τύπους και συνεχίζουμε με πιο περίπλοκους τύπους. Έστω $S(g)$ το σύνολο όλων των καταστάσεων που αντιστοιχίζονται με τον ύποτυπο g.

$$S(\text{Start}) = 2, 5, 6, 7$$
$$S(\neg\text{Heat}) = 1, 2, 3, 5, 6$$

Για να υπολογίσουμε το $S(\mathbf{EG}\neg\text{Heat})$ πρώτα βρίσκουμε το σύνολο των μη-πεπερασμένων ισχυρά συνδεδεμένων συνιστοσών στο $S' = S(\neg\text{Heat})SCC = 1, 2, 3, 5$. Συνεχίζουμε θέτοντας T, το σύνολο όλων των καταστάσεων που θα έπρεπε να αντιστοιχιστούν με το $\mathbf{EG}\neg\text{Heat}$ να είναι η ένωση όλων των στοιχεών του SCC. Έτσι αρχικά $T = 1, 2, 3, 5$. Καμία άλλη κατάσταση στο S' δεν μπορεί να προσεγγίσει μια κατάσταση στο T κατά μήκος ενός μονοπατιού στο S'. Άρα



Σχήμα 2: Παράδειγμα φούρνου μικροκυμάτων

ο υπολογισμός τερματίζει με $S(\mathbf{EG}\neg\text{Heat}) = 1, 2, 3, 5$. Τέλος υπολογίζουμε το $S(\text{Start} \wedge \mathbf{EG}\neg\text{Heat}) = 2, 5$.

Όταν υπολογίζουμε το $S(\mathbf{EF}(\text{Start}\wedge\mathbf{EG}\neg\text{Heat}))$, ξεκινάμε θέτοντας $T = S(\text{Start}\wedge \mathbf{EG}\neg\text{Heat})$. Μετά χρησιμοποιούμε το αντίστροφο του πίνακα μεταβάσεων για να αντιστοιχίσουμε όλες τις καταστάσεις με τον τύπο στον οποίο αντιστοιχούν. Παίρνουμε:

$$S(\mathbf{EF}(\text{Start} \wedge \mathbf{EG}\neg\text{Heat})) = 1, 2, 3, 4, 5, 6, 7$$

Έτσι υπολγίζουμε ότι $S(\neg\mathbf{EF}(\text{Start} \wedge \mathbf{EG}\neg\text{Heat})) = 0$. Αφού η αρχική κατάσταση δεν περιέχεται στο σύνολο, καταλήγουμε ότι το σύστημα δεν ικανοποιεί το specification.

References

[1] Edmund M. Clarke Jr., Orna Grumberg, and Doron Peleg. *Model Checking*. 1999.

[2] Jörg Kreiker et al. «Modeling, Analysis, and Verification – The Formal Manifesto 2010». In: 2010, pp. 21–41.

[3] Colin Walls. *Safety-Critical Systems: The basics*. 2016. URL: https://www.embedded.com/safety-critical-systems-the-basics/ (visited on 09/30/2019).