Σχεδιασμός Πρωτοκόλλων

Χειμερινό εξάμηνο

Περιεχόμενα μαθήματος

- Μηχανές πεπερασμένων καταστάσεων
- Μηχανές Turing
- Μηχανές πεπερασμένων καταστάσεων που επικοινωνούν
- Ελαχιστοποίηση μηχανών
- Σύνθεση μηχανών
- Επέκταση μηχανών

Οι βασικοί κανόνες για το σωστό σχεδιασμό πρωτοκόλλων

SIMPLICITY — THE CASE FOR LIGHT-WEIGHT PROTOCOLS

«A well-structured protocol can be built from a small number of well-designed and well-understood pieces. To understand the working of the protocol it should suffice to understand the working of the pieces from which it is constructed and the way in which they interact.»

MODULARITY — A HIERARCHY OF FUNCTIONS

«A protocol that performs a complex function can be built from smaller pieces that interact in a well-defined and simple way. Each smaller piece is a light-weight protocol that can be separately developed, verified, implemented, and maintained. Orthogonal functions are not mixed; they are designed as independent entities. Error control and flow control, for instance, are orthogonal functions.»

Οι βασικοί κανόνες για το σωστό σχεδιασμό πρωτοκόλλων

WELL-FORMED PROTOCOLS

«A *well-formed protocol is not over-specified, that is, it does not contain any unreachable* or unexecutable code. A well-formed protocol is also not *under-specified or* incomplete.»

«A well-formed protocol is *bounded: it cannot overflow known system limits, such as* the limited capacity of message queues.»

«A well-formed protocol is *self-stabilizing. If a transient error arbitrarily changes the* protocol state, a self-stabilizing protocol always returns to a desirable state within a finite number of transitions, and resumes normal operation.»

«A well-formed protocol, finally, is *self-adapting. It can adapt, for instance, the rate at* which data are sent to the rate at which the data links can transfer them, and to the rate at which the receiver can consume them.»

Οι βασικοί κανόνες για το σωστό σχεδιασμό πρωτοκόλλων

ROBUSTNESS

«It is not difficult to design protocols that work under normal circumstances. It is the unexpected that challenges them. It means that the protocol must be prepared to deal appropriately with every feasible action and with every possible sequence of actions under all possible conditions. The protocol should make only minimal assumptions about its environment to avoid dependencies on particular features that could change. A robust design automatically scales up with new technology, without requiring fundamental changes. The best form of robustness, then, is not *over-design by* adding functionality for anticipated new conditions, but *minimal design by removing* non-essential assumptions that could prevent adaption to unanticipated conditions.»

Οι βασικοί κανόνες για το σωστό σχεδιασμό πρωτοκόλλων

CONSISTENCY

There are some standard and dreaded ways in which protocols can fail. We list three of the more important ones.

Deadlocks — states in which no further protocol execution is possible, for instance because all protocol processes are waiting for conditions that can never be fulfilled.

Livelocks — execution sequences that can be repeated indefinitely often without ever making effective progress.

Improper terminations — the completion of a protocol execution without satisfying the proper termination conditions.

Σχεδιασμός

- Ένα πρωτόκολλο μπορεί να περιγραφεί σαν μια μηχανή καταστάσεων
- **Κατάσταση:** συμβολίζει τις ενέργειες που μπορεί να εκτελέσει μια διεργασία, ποια γεγονότα περιμένει να συμβούν και πώς θα αντιδράσει σε αυτά τα γεγονότα
- Υποθέσεις που κάνει για τις άλλες διεργασίες
- Η φορμαλιστική περιγραφή των μηχανών παίζει σημαντικό ρόλο στην επαλήθευση, στη σύνθεση και στον έλεγχο συμμόρφωσης

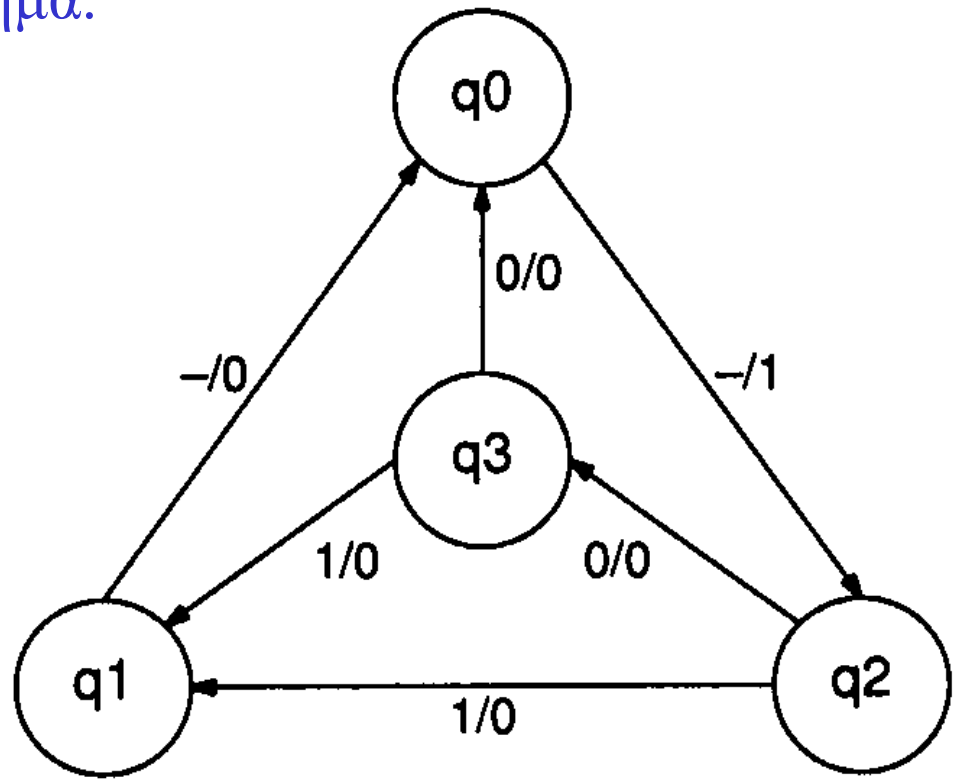
Μη φορμαλιστική περιγραφή

Πίνακας 8.1 — Mealy

Συνθήκη		Αποτέλεσμα	
Τρέχουσα Κατάσταση	In	Out	Επόμενη Κατάσταση
q0	–	1	q2
q1	–	0	q0
q2	0	0	q3
q2	1	0	q1
q3	0	0	q0
q3	1	0	q1

Διάγραμμα μετάβασης καταστάσεων

- Αν δεν μπορεί να εκτελεστεί κάποιος κανόνας μετάβασης είμαστε σε end-state.
- Η ύπαρξη κάποιων end-states μπορεί να σημαίνει πρόβλημα.



— State Transition Diagram

Αιτιοκρατικές Πεπερασμένες μηχανές καταστάσεων

- Deterministic finite state machine (also known as deterministic finite state automaton (DFSA) or deterministic finite automaton (DFA)) is a finite state machine where for each pair of state and input symbol there is one and only one transition to a next state.

Μη αιτιοκρατικές μηχανές πεπερασμένων καταστάσεων

Παράδειγμα μη αιτιοκρατικής μηχανής

Τρέχουσα Κατάσταση	In	Out	Επόμενη Κατάσταση
q1	–	0	q0
q1	0	0	q3

Μηχανές Turing

- Ο προηγούμενος ορισμός μιας μηχανής πεπερασμένων καταστάσεων είναι και ο απλούστερος.
- Παραλλαγές του βασικού μοντέλου διαφέρουν ως προς τον τρόπο που ορίζεται το περιβάλλον της μηχανής και επομένως διαφέρουν στον ορισμό των συνθηκών και των αποτελεσμάτων των κανόνων μετάβασης.
- Για τα πραγματικά συστήματα πεπερασμένων καταστάσεων, το περιβάλλον πρέπει να είναι επίσης μια «πεπερασμένη κατάσταση» (π.χ. οριζόμενο ως μια άλλη μηχανή πεπερασμένων καταστάσεων).
- Εάν δεν τηρείται αυτή η απαίτηση, λαμβάνουμε το καλά γνωστό μοντέλο του Θεωρητικού Υπολογιστή Turing (Turing Machine model).

Μηχανές Turing

Περιγραφή Μηχανής Turing

Συνθήκη		Αποτέλεσμα	
Τρέχουσα Κατάσταση	In	Out / Μετακίνηση	Επόμενη Κατάσταση
q0	0	1/L	q1
q0	1	1/R	q2
q1	0	1/R	q0
q1	1	1/L	–
q2	0	1/R	q1
q2	1	1/L	q3
q3	–	–	–

q3=end state

Επικοινωνούσες μηχανές πεπερασμένων καταστάσεων

- Τι θα γίνει αν συμπέσουν τα σήματα εισόδου με τα σήματα εξόδου
- Τι είναι σήμα?
- Τα σήματα μπορούν να πάρουν ένα πεπερασμένο αριθμό τιμών
- Η αλλαγή στην τιμή μπορεί να πραγματοποιηθεί σε συγκεκριμένες χρονικές στιγμές
- Αλγόριθμος 2 βημάτων: κοιτάμε το σήμα εισόδου και την κατάσταση & αλλαγή κατάστασης και ενημέρωση του σήματος εξόδου
- Τα σήματα έχουν καταστάσεις

Επικοινωνούσες μηχανές πεπερασμένων καταστάσεων

- Με τα παραπάνω ποια είναι η συμπεριφορά της μηχανής;

Πίνακας 8.1 — Mealy

Συνθήκη		Αποτέλεσμα	
Τρέχουσα Κατάσταση	In	Out	Επόμενη Κατάσταση
q0	–	1	q2
q1	–	0	q0
q2	0	0	q3
q2	1	0	q1
q3	0	0	q0
q3	1	0	q1

Επικοινωνούσες μηχανές πεπερασμένων καταστάσεων

- Συγχρονισμός μόνο στα βήματα εκτέλεσης του αλγορίθμου
- Μπορούμε να φτιάξουμε αρκετά πολύπλοκα συστήματα
- Ασύγχρονη επικοινωνία: Οι μηχανές συνδέονται με FIFO ουρές (εισόδου και εξόδου)
- Συγχρονισμός βάσει της κατάστασης
- Αν η ουρά είναι άδεια και θέλουμε να πάρουμε ένα μήνυμα ή είναι γεμάτη και θέλουμε να βάλουμε ένα μήνυμα δεν μπορούμε
- Η μετάβαση από κατάσταση σε κατάσταση γίνεται σε βήματα που δεν μπορούν να διακοπούν (atomic steps)

Alternating Bit Protocol

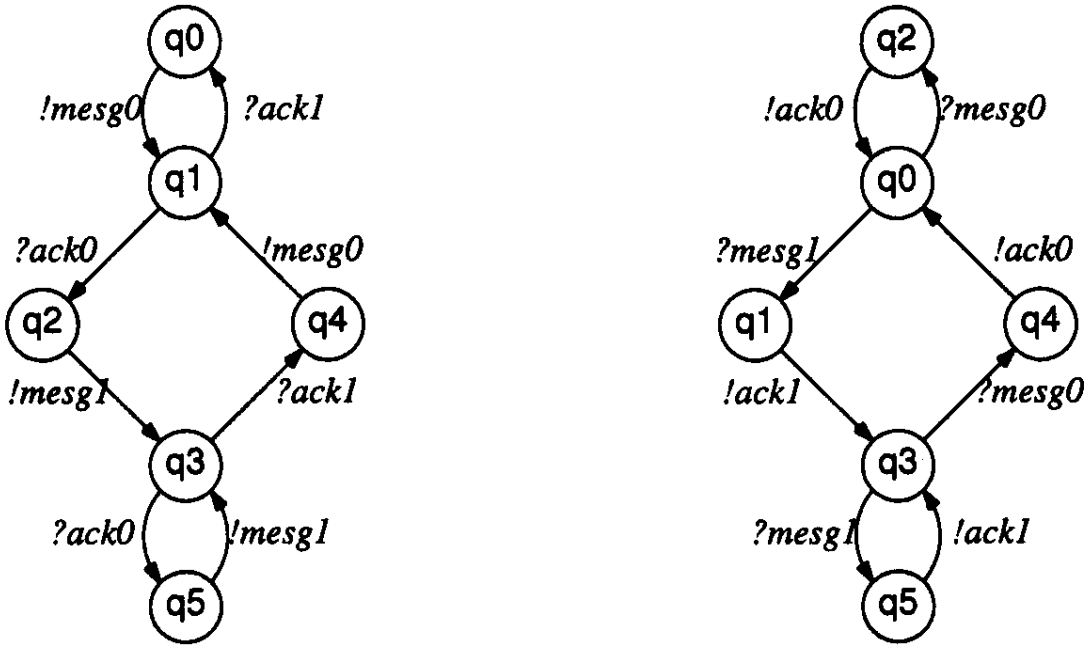
Αποστολέας (Sender)

State	In	Out	Next State
q0	–	mesg0	q1
q1	ack1	–	q0
q1	ack0	–	q2
q2	–	mesg1	q3
q3	ack0	–	q2
q3	ack1	–	q0

Παραλήπτης (Receiver)

State	In	Out	Next State
q0	mesg1	–	q1
q0	mesg0	–	q2
q1	–	ack1	q3
q2	–	ack0	q0
q3	mesg0	–	q4
q3	mesg1	–	q5
q4	–	ack0	q0
q5	–	ack1	q3

Alternating Bit Protocol



State Transition Diagrams

Δεν υπάρχουν παράμετροι…
υπονοούνται από το όνομα του μηνύματος

Alternating Bit Protocol

- Τι λείπει από την προηγούμενη προδιαγραφή?

Κατάσταση	Input	Output	Next State
q1	-	mesg0	-
q3	-	mesg1	-

Μας ενδιαφέρει η δυνατότητα να συμβεί κάτι και όχι η πιθανότητα

Σύγχρονη επικοινωνία

- Ένα σήμα πρέπει να επιλεγεί από δύο μηχανές ταυτόχρονα

Χρήστης (User)

State	In	Out	Next State
q0	P	–	q1
q1	–	V	q0

Εξυπηρετητής (Server)

State	In	Out	Next State
q0	–	P	q1
q1	V	–	q0

- Τι θα γίνει αν έχουμε 2 χρήστες και ένα server?
- Η σύγχρονη επικοινωνία είναι υποπερίπτωση της ασύγχρονης για ουρές μεγέθους 0

Φορμαλιστικός ορισμός μηχανών

- Μία Επικοινωνούσα Μηχανή Πεπερασμένων καταστάσεων είναι ένα δαίμονας που:
  - δέχεται σύμβολα εισόδου
  - γεννάει σήματα εξόδου
  - Αλλάζει την εσωτερική του κατάσταση βάσει κάποιου προδιαγεγραμμένου πλάνου
  - επικοινωνεί μέσω συνδεδεμένων ουρών αναμονής FIFO, που απεικονίζουν την έξοδο μιας μηχανής πάνω στην είσοδο της άλλης μηχανής

Φορμαλιστικός ορισμός μηχανών

- *Η έννοια της ουράς αναμονής (queue):*

  Μια *ουρά αναμονής μηνυμάτων (message queue)* είναι ένα τριπλό «σύνολο» (triple) (S, N, C), όπου:

  - *S* είναι ένα πεπερασμένο σύνολο (finite set) που αποκαλείται ως το λεξιλόγιο της ουράς αναμονής (queue vocabulary),
  - *N* είναι ένας ακέραιος που ορίζει τον αριθμό των θυρίδων που βρίσκονται εντός της ουράς αναμονής (number of slots in the queue), και
  - *C* είναι τα περιεχόμενα της ουράς αναμονής (queue contents), ένα διατεταγμένο σύνολο στοιχείων από το S.

- Τα στοιχεία των *S* και *C* αποκαλούνται μηνύματα (messages).
- Σε αυτά αποδίδονται ονόματα με μοναδικό τρόπο, αν και είναι μη οριζόμενα αφηρημένα αντικείμενα.
- Εάν ορίζονται περισσότερες από μια ουρές αναμονής, απαιτούμε το να είναι ασύνδετα τα λεξιλόγια ουράς αναμονής.

Φορμαλιστικός ορισμός μηχανών

Ας είναι *M* το σύνολο όλων των ουρών αναμονής μηνυμάτων (messages queues). Χρησιμοποιείται ένας άνω δείκτης , $1 \le m \le |M|$, για την αναγνώριση μιας μεμονωμένης ουράς αναμονής, και χρησιμοποιείται ένας δείκτης $1 \le n \le N$, για την αναγνώρι ση μιας θυρίδας εντός της ουράς αναμονής.

Τότε, C_n^m είναι το n-στό μήνυμα εντός της m-στής ουράς αναμονής.

Ένα λεξιλόγιο του συστήματος, *V*, μπορεί να ορίζεται ως η ένωση όλων των λεξιλογίων ουρών αναμονής, συν ένα μηδενικό στοιχείο (null element) το οποίο υποδηλώνεται με το σύμβολο ε. Δοθέντος του συνόλου των ουρών αναμονής M, που απαριθμούνται από 1 έως $|M|$, το λεξιλόγιο του συστήματος, *V*, ορίζεται ως

$$V = \bigcup_{1}^{|M|} S^m \bigcup \varepsilon$$

Φορμαλιστικός ορισμός μηχανών

- **Ορισμός μιας επικοινωνούσας μηχανής πεπερασμένων καταστάσεων-(1)**

Μια **επικοινωνούσα μηχανή πεπερασμένων καταστάσεων (communicating finite state machine) είναι μια πλειάδα (tuple)** ***(Q, q0 , M, T),*** όπου

- ***Q*** είναι ένα πεπερασμένο, μη άδειο σύνολο καταστάσεων,
- ***q0*** είναι ένα στοιχείο του Q, η *αρχική κατάσταση (initial state),*
- ***M*** είναι ένα σύνολο όλων των ουρών αναμονής μηνυμάτων, όπως ορίζεται παραπάνω, και
- ***T*** είναι μια σχέση μετάβασης κατάστασης (state transition relation).

- **Η σχέση *T* δέχεται δύο ορίσματα, T(q, a), όπου**
- ***q* είναι η τρέχουσα κατάσταση και *a* είναι μια δράση (action).**

- **Επιτρέπονται μόνο τρεις τύποι δράσεων:**
- **είσοδοι (inputs),**
- **έξοδοι (outputs), και**
- **μια μηδενική δράση (null action) ε.**

Λειτουργία των μηχανών

1. Ρύθμιση όλων των μηχανών στην αρχική τους κατάσταση, και αρχικοθέτηση όλων των ουρών αναμονής μηνυμάτων σε άδειο.

$$\forall(i)\ 1 \leq i \leq P \longrightarrow q^i = q^i_0$$

$$\forall(i)\ 1 \leq i \leq |M| \longrightarrow C^j = \emptyset$$

2. Επιλογή μιας αυθαίρετης μηχανής j και ενός αυθαίρετου κανόνα μετάβασης T^j με $T^j(q^j, a) \neq \emptyset$ και που να είναι εκτελέσιμο και εκτέλεση αυτού.

3. Εάν δεν απομένουν καθόλου εκτελέσιμοι κανόνες μετάβασης, τερματίζει ο αλγόριθμος.

Η διαδρομή μπορεί να είναι μια ατέρμονη οδός, ή μπορεί να είναι η μηδενική διαδρομή.

Λειτουργία των μηχανών

Ας είναι $1 \le d(a) \le |M|$ η αριθμαναμονής προορισμού μιας δράσης a, και ας είναι $n(a)$ το μήκος της ουράς από την οποία επιλέγεται ή λαμβάνεται, $n(a) \in S^{d(a)}$.

Επιπλέον, το N^i παριστάνει τον αριθμό των θυρίδων σε μια αριθμαναμονής μηνύματος i.

Σε ένα ασύγχρονο σύστημα, πχ., *μπορούν να χρησιμοποιούνται οι ακόλουθες τρεις κανόνες για τον προσδιορισμό του ενός εκτελέσιμου*:

$$a = \varepsilon \qquad (1)$$

ή

$$\text{Το } a \text{ είναι μια εξόδου και } n(a) = C_1^{d(a)} \qquad (2)$$

ή

$$\text{Το } a \text{ είναι μια εξόδου και } \left| C^{d(a)} \right| < N^{d(a)} \qquad (3)$$

Ο αλγόριθμος FSMEXECUTION δεν υπόκειται σε αναγκαστικά σε περιορισμό

Ισοδυναμία μηχανών

- Δύο μηχανές λέγεται ότι είναι *ισοδύναμες (equivalent)* εάν αυτές μπορούν να δημιουργούν την ίδια ακολουθία συμβόλων εξόδου, όταν σε αυτές προσφέρεται η ίδια ακολουθία συμβόλων εισόδου.

- *Η λέξη-κλειδί, εδώ, είναι η λέξη «μπορούν».*
- Οι μηχανές που θεωρούμε, μπορούν να πραγματοποιούν μη αιτιοκρατικές επιλογές (nondeterministic choices) μεταξύ κανόνων μετάβασης εάν είναι εκτελέσιμες, *ταυτόχρονα*, περισσότερες από μια επιλογές.

- Αυτή η έννοια της μη αιτιοκρατίας (nondeterminism) σημαίνει ότι δύο ίσες μηχανές (equal machines) *μπορούν* να συμπεριφέρονται διαφορετικά όταν προσφέρονται σε αυτές τα ίδια σύμβολα εισόδου.

- Ο κανόνας για ισοδυναμία (equivalence) έγκειται στο ότι οι μηχανές πρέπει να έχουν ισοδύναμες επιλογές ώστε να βρίσκονται σε ισοδύναμες καταστάσεις.

Ισοδυναμία μηχανών

- Οι *καταστάσεις (states)* σε μια απλή-μεμονωμένη μηχανή λέγονται ότι είναι ισοδύναμες εάν η μηχανή μπορεί
- να ξεκινάει σε οποιαδήποτε από αυτές τις καταστάσεις και
- να μπορεί να δημιουργεί το ίδιο σύνολο δυνατών ακολουθιών των εξόδων όταν της προσφέρεται οποιαδήποτε δοθείσα ακολουθία δοκιμής (test sequence) των εισόδων.

- *Σε κάθε περίπτωση, ο ορισμός μιας ενδεδειγμένης σχέσης ισοδυναμίας (equivalence relation) για καταστάσεις, πρέπει να επιλέγεται προσεκτικά.*

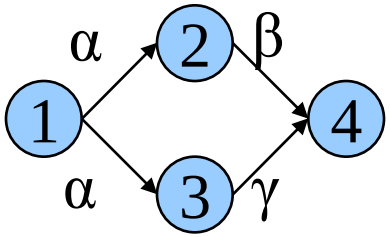
Ισοδυναμία μηχανών

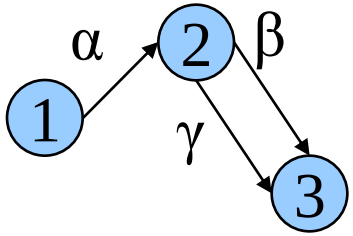
- if (receive a) then non deterministically (receive b) or (receive c)
- if (receive a) then (receive b)

  OR non deterministically

  if (receive a) then (receive c)
  - Τι γίνεται στην περίπτωση λήψης a και στη συνέχεια b στις δύο μηχανές;

Ισοδύναμες;





Promela model checker Spin

Downloading Spin:

- http://spinroot.com/spin/Man/README.html

Ελαχιστοποίηση καταστάσεων

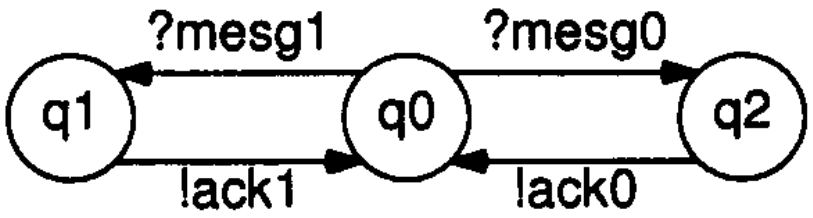
Receiver-II

Condition		Effect	
Current State	In	Out	Next State
q0	mesg1	–	q1
q0	mesg0	–	q2
q1	–	ack1	q0
q2	–	ack0	q0

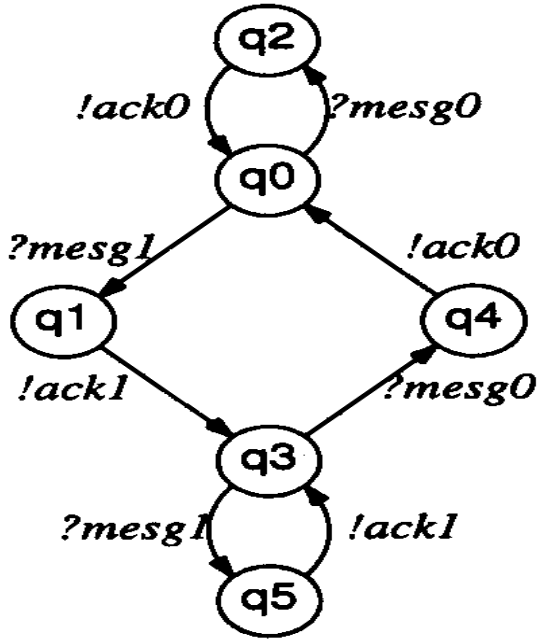
Παραλήπτης (Receiver)

State	In	Out	Next State
q0	mesg1	–	q1
q0	mesg0	–	q2
q1	–	ack1	q3
q2	–	ack0	q0
q3	mesg0	–	q4
q3	mesg1	–	q5
q4	–	ack0	q0
q5	–	ack1	q3

Ελαχιστοποίηση καταστάσεων



State Transition Diagram



Receiver

ns

Ελαχιστοποίηση καταστάσεων

1. Ορισμός ενός πίνακα (array) *E* των τιμών Μπουλ $|Q|\times|Q|$. Αρχικά κάθε στοιχείο $E[i,j]$ του πίνακα τιμοδοτείται στην τιμή αληθής της επόμενης συνθήκης, για όλες τις δράσεις *α*:

$$T(i,a) \neq \varnothing \Leftrightarrow T(j,a) \neq \varnothing$$

Οι δύο καταστάσεις δεν είναι ισοδύναμες εκτός εάν οι αντίστοιχες σχέσεις μετάβασης κατάστασης ορίζονται για τις ίδιες δράσεις.

2. Εάν η υπό θεώρηση μηχανή περιέχει μόνο αιτιοκρατικές επιλογές, το *T* ορίζει μια μοναδική διάδοχο κατάσταση για όλα τα λήμματα *αληθής (true)* του πίνακα *E*. Μεταβάλλετε τότε την τιμή όλων εκείνων των λημμάτων $E[i,j]$ στην τιμή

$$\forall(a), \quad E[T(i,a),T(j,a)]$$

Αυτό σημαίνει ότι οι καταστάσεις δεν είναι ισοδύναμες εκτός εάν οι διάδοχές τους καταστάσεις (successors states) είναι επίσης ισοδύναμες. Όταν τα *T(i,α)* και *T(j,α)* μπορούν να έχουν περισσότερα από ένα στοιχεία, η σχέση είναι πιο πολύπλοκη. Η τιμή του $E[i,j]$ τώρα τιμοδοτείται σε *ψευδής (false)* εάν οποιαδήποτε από τις δύο ακόλουθες συνθήκες είναι *ψευδής* για οποιαδήποτε δράση *α*.

$$\forall(p), \quad p \in T(i,a) \quad \rightarrow \quad \exists(q), q \in T(j,a) \quad \kappa\alpha\iota \quad E[p,q]$$

$$\forall(p), \quad p \in T(j,a) \quad \rightarrow \quad \exists(q), q \in T(i,a) \quad \kappa\alpha\iota \quad E[q,p]$$

Αυτό σημαίνει ότι οι καταστάσεις *i* και *j* δεν είναι ισοδύναμες εκτός για κάθε δυνατή διάδοχο κατάσταση *p* της κατάστασης *i* υπάρχει τουλάχιστον μια ισοδύναμη διάδοχος κατάσταση *q* της κατάστασης *j*, και αντίστροφα.

3. Επανάληψη του βήματος 2 μέχρις ότου ο αριθμός των λημμάτων *ψευδής (false)* στον πίνακα *E* δεν μπορεί πλέον να αυξάνεται.

Ελαχιστοποίηση καταστάσεων

Παραλήπτης (Receiver)

State	In	Out	Next State
q0	mesg1	–	q1
q0	mesg0	–	q2
q1	–	ack1	q3
q2	–	ack0	q0
q3	mesg0	–	q4
q3	mesg1	–	q5
q4	–	ack0	q0
q5	–	ack1	q3

Equivalence

	q0	q1	q2	q3	q4	q5
q0	1	0	0	1	0	0
q1	0	1	0	0	0	1
q2	0	0	1	0	1	0
q3	1	0	0	1	0	0
q4	0	0	1	0	1	0
q5	0	1	0	0	0	1

Δοκιμές Συμμόρφωσης

- **Η διαδικασία για ισοδυναμία δοκιμών (testing equivalence) των καταστάσεων μπορεί να εφαρμόζεται για τον προσδιορισμό της ισοδυναμίας των δύο μηχανών.**
- **Τότε το πρόβλημα συνίσταται στον προσδιορισμό του ότι κάθε κατάσταση στη μια μηχανή έχει μια ισοδύναμη κατάσταση στην άλλη μηχανή, και αντίστροφα.**
- **Φυσικά, οι μηχανές δεν χρειάζεται να είναι «ίσες» για να είναι «ισοδύναμες».**

Δοκιμές Συμμόρφωσης

- Ας υποτεθεί ότι έχουμε μια επίσημη προδιαγραφή πρωτοκόλλου, σε μορφή μηχανής πεπερασμένων καταστάσεων, και μια υλοποίηση αυτής της προδιαγραφής.

- Οι δύο μηχανές πρέπει να είναι ισοδύναμες, δηλαδή η υλοποίηση η οποία αντιμετωπίζεται ως ένα «μαύρο κουτί», θα πρέπει να αποκρίνεται σε σήματα εισόδου ακριβώς όπως θα έπρεπε να αποκρίνεται η μηχανή αναφοράς.

- *Εντούτοις δεν μπορούμε να γνωρίζουμε με βεβαιότητα οτιδήποτε σχετικά με την αληθινή εσωτερική δομή της υλοποίησης.*

Σύνθεση μηχανών

Γενικά:

Με την σύνθεση δύο ξεχωριστών μηχανών πεπερασμένων καταστάσεων σε μια μεμονωμένη μηχανή, μπορεί να μειωθεί η πολυπλοκότητα των επαληθεύσεων που βασίζεται πάνω σε μοντέλα μηχανών πεπερασμένων καταστάσεων.

Προσέγγιση:

Σύνθεση μηχανών

1. Ορισμός του συνόλου του γινομένου (product set) αυτών των δύο συνόλων των καταστάσεων των δύο μηχανών καταστάσεων. Η πρώτη μηχανή έχει $|Q^1|$ καταστάσεις και η δεύτερη μηχανή έχει $|Q^2|$ καταστάσεις, επομένως το σύνολο του γινομένου περιέχει $|Q^1| \times |Q^2|$ καταστάσεις. Ονοματίζουμε αρχικά τις καταστάσεις της νέας μηχανής μέσω συναλύσωσης των ονομάτων καταστάσεων των αρχικών μηχανών, με μία καθορισμένη σειρά. **Αυτό ορίζει το σύνολο Q της συνδυασμένης μηχανής.** Η αρχική κατάσταση $q0$ της νέας μηχανής είναι ο συνδυασμός $q_0^1 q_0^2$ των αρχικών καταστάσεων των δύο αρχικών μηχανών.

2. Το σύνολο των ουρών αναμονής των μηνυμάτων, *M*, της συνδυασμένης μηχανής είναι η ένωση των συνόλων των ουρών αναμονής των ξεχωριστών μηχανών $M^1 \bigcup M^2$.

Τα δύο αρχικά σύνολα δεν χρειάζεται να είναι ασύνδετα. Το λεξιλόγιο, *V*, της νέας μηχανής είναι το συνδυασμένο λεξιλόγιο των *M*1 και *M*2, και το σύνολο των δράσεων *a* είναι η ένωση όλων των δράσεων που μπορούν να εκτελούν οι ξεχωριστές μηχανές.

3. Για κάθε κατάσταση $q^1 q^2$ εντός του Q, ορίζεται η σχέση μετάβασης *T* για κάθε δράση *a*, ως η μη αποκρατική επιλογή των αντίστοιχων σχέσεων των *M*1 και *M*2 ξεχωριστά, όταν τοποθετούνται στις ξεχωριστές καταστάσεις *q*1 και *q*2. Αυτό μπορεί να γραφεί:

$$\forall(q^1 q^2), \quad \forall(a) \quad \rightarrow \quad T(q^1 q^2, a) = T^1(q^1, a) \bigcup T^2(q^2, a)$$

Σύνθεση μηχανών

- Η συνδυασμένη μηχανή μπορεί να ελαχιστοποιείται χρησιμοποιώντας τον Αλγόριθμο 2 (FSM Minimization).

- Ο Αλγόριθμος-3 FSM Composition (Σύνθεση FSM) μπορεί εύκολα να προσαρμόζεται, για να συνδυάζει περισσότερες από δύο μηχανές.

- Η μεγαλύτερη αξία της τεχνικής σύνθεσης από το τελευταίο τμήμα είναι ότι μας επιτρέπει να απλοποιήσουμε πολύπλοκες συμπεριφορές. (Π.χ. σε επαληθεύσεις πρωτοκόλλου, θα μπορούσαμε με βεβαιότητα να έχουμε το πλεονέκτημα μιας μεθόδου που μας επιτρέπει να πραγματοποιούμε σύνθεση δύο μηχανών σε μια μηχανή).

- Μια μέθοδος θα μπορούσε να είναι το να προβούμε σε σύνθεση δύο μηχανών χρησιμοποιώντας τον Αλγόριθμο FSM Composition, μέσω της απομάκρυνσης όλων των εσωτερικών αλληλεπιδράσεών τους (δηλαδή της αρχικής διεπαφής μεταξύ των δύο μηχανών), και ελαχιστοποιώντας την προκύπτουσα μηχανή.

Αλλαγές στο μοντέλο FSM

- Ορίζουμε ένα επιπλέον primitive πουορίζεται περίπου οπως μια ουρά: τη μεταβλητή variable. Οι μεταβλητές έχουν συμβολικά ονόματα και αναπαριστούν αφηρημένα αντικείμενα – π.χ. integer values. Η διαφορά με την ουρά είναι οτι η μεταβλητή μπορεί να έχει μόνο μια τιμή κάθε φορά από το πεδίο τιμών της. Μόνο η τελευταία τιμή που τέθηκε η μεταβλητή μπορεί να ανακληθεί.
- Χρήση των queues για τη μεταφρά integer τιμών αντί αφηρημένων ατνικειμένων
- Εισαγωγή πεδίου αριθμητικών και λογικών τελεστών - operators για τη διαχείριση των περιεχομένων των variables.

Επέκταση των μηχανών

Μια μεταβλητή με ένα πεπερασμένο εύρος μπορεί να υπόκειται σε προσομοίωση με τετριμμένο τρόπο, *μέσω μιας μηχανής πεπερασμένων καταστάσεων.*

Finite State Variable

Current State	In	Out	Next State
q0	s0	–	–
q0	s1	–	q1
q0	s2	–	q2
q0	rv	–	r0
r0	–	0	q0
q1	s0	–	q0
q1	s1	–	–
q1	s2	–	q2
q1	rv	–	r1
r1	–	1	q1
q2	s0	–	q0
q2	s1	–	q1
q2	s2	–	–
q2	rv	–	r2
r2	–	2	q2

Επέκταση των μηχανών

Input and output actions can now be generalized as well. We will define I/O actions as finite, ordered sets of values. The values can be expressions on variables from *A,* or simply constants. By definition the first value from such an ordered set defines the destination queue for the I/O, within the range 1*..ïMï. The remaining values define a* data structure that is appended to, or retrieved from, the queue when the I/O action is performed.

Extended Finite State Machine

Current State	Action	Next State
q0	In?x,y	q1
q1	x>y	q2
q1	x<y	q3
q1	x=y	q4
q2	x=x-y	q1
q3	y=y-x	q1
q4	Out!x	q5
q5	–	–