Εισαγωγή

Στην ενότητα αυτή θα μελετηθούν τα εξής θέματα:

Ο ρόλος της ανάλυσης και επαλήθευσης συστημάτων

Τεχνικές ανάλυσης συστημάτων

Στόχοι του μαθήματος

Διδασκαλία

Διαλέξεις:	Τετάρτη, 1500 – 1800
Φροντιστήριο:	Τετάρτη, 1400 – 1500
Εργαστήριο:	Παρασκευή, 1330 – 1500

Ιστοσελίδα μαθήματος: http://www.cs.ucy.ac.cy/~annap/epl664/

Περιγραφή

Το μάθημα θα μελετήσει:

- Πρότυπα μοντελοποίησης συστημάτων:
  - Συστήματα μεταβάσεων
  - Άλγεβρες Διεργασιών
  - Αυτόματα
- Τεχνικές ανάλυσης συστημάτων:
  - Μοντελοέλεγχος (model-checking)
  - Ισοδυναμίες (simulations)
- Εργαλεία που επιτρέπουν την αυτοματοποιημένη ανάλυση συστημάτων:
  - SPIN
  - UPPAAL

Παράλληλα συστήματα, κατανεμημένα συστήματα, πρωτόκολλα επικοινωνίας, κλπ

Στόχοι του μαθήματος

- Εξοικείωση με διάφορες τεχνικές επαλήθευσης συστημάτων
- Αναγνώριση δυνατοτήτων και περιορισμών της κάθε μιας από αυτές
- Εξοικείωση με εργαλεία επαλήθευσης συστημάτων
- Διαδικασία: Επιλογή εργαλείου, μοντελοποίηση, ανάλυση, εύρεση λαθών
- Ειδικά θέματα: Χρόνος, πιθανότητα, ασφάλεια

Αξιολόγηση

5 σειρές ασκήσεων	$5 \times 5 =$ 25%
Ενδιάμεση Εξέταση	25%
Τελική εξέταση	50%

Βιβλιογραφία

- D. Peled, *Software Reliability Methods*. Springer-Verlag, 2001.
- Christel Baier, Joost-Pieter Katoen, *Principles of Model Checking*. MIT Press, May 2008
- M. Huth and A. Ryan. *Logic in Computer Science: Modeling and Reasoning about Concurrent Systems.* Cambridge University Press, 2nd edition, 2004.
- Επιλεγμένα άρθρα βιβλιογραφίας.

Γιατί ανάλυση συστημάτων;

Ανάγκη για ανάλυση και επαλήθευση συστημάτων.

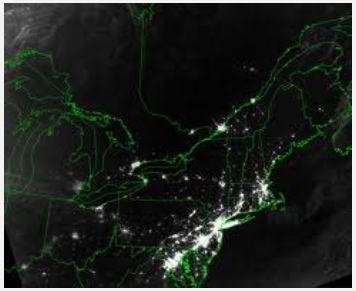
- Η κοινωνία της πληροφορίας είναι γεγονός. Υπολογιστές και προγράμματα έχουν εξαπλωθεί σε πολλαπλούς τομείς της ζωής μας:
  - ενθυλακωμένα συστήματα (embedded systems)
  - αυτόνομα συστήματα
  - e-banking και e-shopping
  - μεταφορικά μέσα
  - ιατρική
  - …
- Η αξιοπιστία υλικού και λογισμικού είναι κύριας σημασίας
- Λάθη μπορεί να αποβούν όχι μόνο δαπανηρά (FDIV στο Pentium-II – 475 εκατομμύρια USD) αλλά και μοιραία (Therac-25)

"It is fair to state, that in this digital era correct systems for information processing are more valuable than gold."

Αξιοπιστία λογισμικού και επιπτώσεις

New Year Mobile Bug, 2011

Goce gravity satellite failure, 2010



Blackout, 2003

Ariane-5 crash, 1996

Atlanta airport, 2008

Επαλήθευση Συστημάτων

Επαλήθευση συστημάτων αφορά στον έλεγχο κατά πόσο ένα σύστημα ικανοποιεί τις απαιτήσεις που υπάρχουν από αυτό.

Διαδεδομένες τεχνικές επαλήθευσης λογισμικού

- Εξέταση κώδικα από τρίτο
  - στατική τεχνική
  - αναγνωρίζει από 31% μέχρι και το 90% των λαθών
  - "δύσκολα" λάθη, π.χ. αλγοριθμικά λάθη ή λάθη που προκύπτουν από παραλληλισμό στο πρόγραμμα, δεν εντοπίζονται εύκολα
- Testing
  - δυναμική τεχνική όπου ο κώδικας εκτελείται

30-50% του κόστους διεκπεραίωσης ενός λογισμικού προγράμματος αφιερώνεται στο testing.

Ο χρόνος και η προσπάθεια που ξοδεύεται στον έλεγχο ενός συστήματος είναι συχνά μεγαλύτερος από αυτόν της κατασκευής του.

Τυπικές Μέθοδοι – Formal Methods

- Το ΕΠΛ664 θα μελετήσει την θεωρία και πρακτική της Επαλήθευση και Ανάλυσης Συστημάτων βασισμένη σε *τυπικές μεθόδους* (formal methods).

- Τι είναι οι τυπικές μέθοδοι;
  - *"εφαρμοσμένα μαθηματικά" για τη μοντελοποίηση και ανάλυση υπολογιστικών συστημάτων*

Παρά του ότι βασίζονται σε μαθηματικά μπορούν να τύχουν χρήσης από οποιοδήποτε!

- Προσφέρουν
  - τη δυνατότητα της επαλήθευσης συστημάτων από τη φάση σχεδιασμού
  - αποδοτικές και αυστηρές μεθόδους ελέγχου συστημάτων (πιο μεγάλη κάλυψη και ψηλότερη εγγύηση ορθότητας)
  - μείωση του χρόνου/κόστους που απαιτείται για επαλήθευση

Τυπικές Μέθοδοι – Formal Methods

- Συστήνονται για δημιουργία λογισμικού κριτικής ασφάλειας από οργανισμούς όπως τους
  - ESA (European Space Agency)
  - FAA (Federal Aviation Authority)
  - NASA

"Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers."

NASA

Τυπικές μέθοδοι ανάλυσης συστημάτων

- Στόχος: η απόδειξη της ορθότητας ενός συστήματος με μαθηματική ακρίβεια
- Χρήση τυπικών τεχνικών σε συνδυασμό με εργαλεία
- Διαφορετικές προσεγγίσεις:
  - *παραγωγικές μέθοδοι* (deductive methods)
  - *τυπικός έλεγχος πειραμάτων* (formal testing)
  - *τεχνικές ελέγχου μοντέλου* (model-checking)

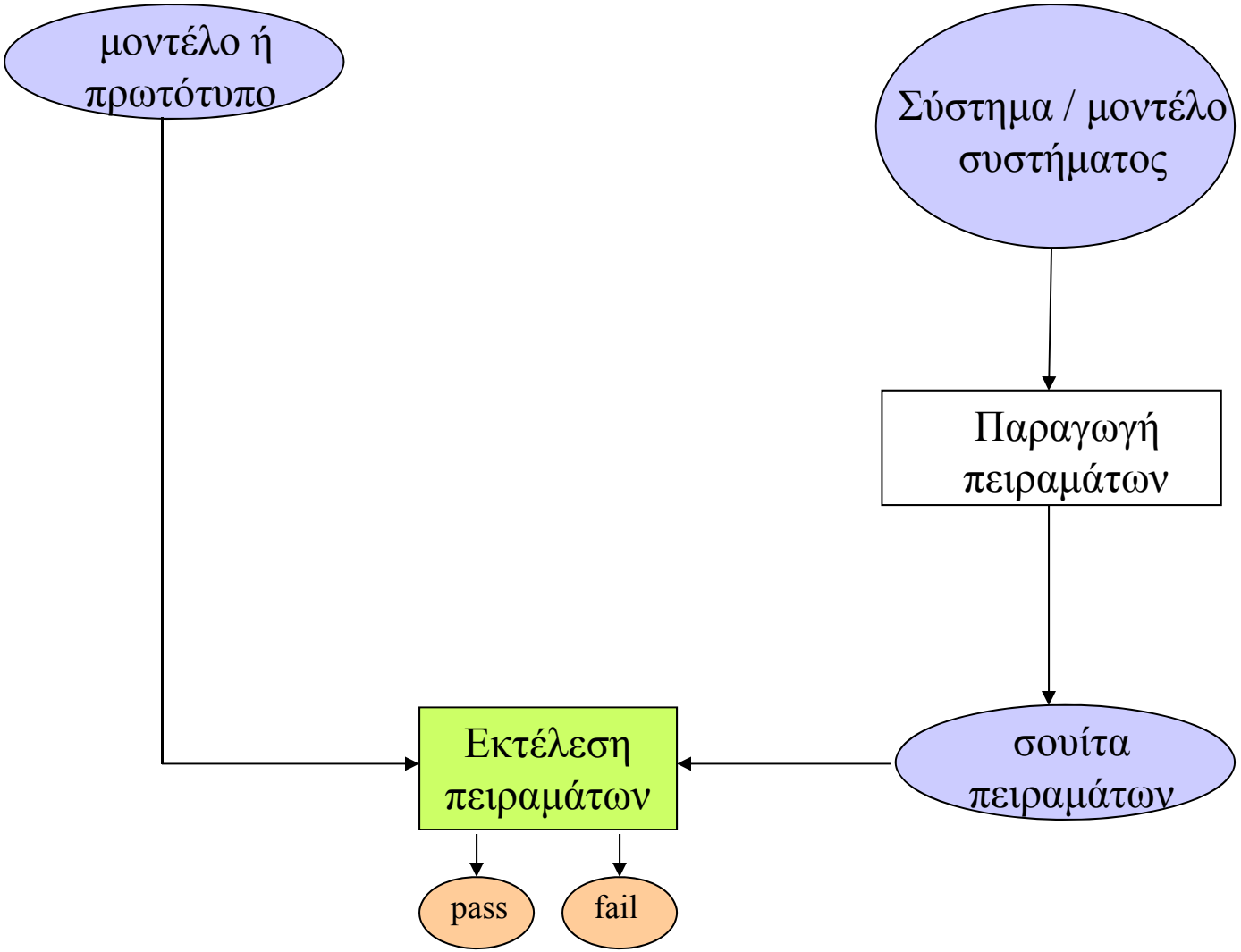
Προσομοιώσεις και πειράματα

- Βασική διαδικασία:
  - Δημιούργησε ένα μοντέλο (προσομοιώσεις) ή μια υλοποίηση (πειράματα) και προσομοίωσε τη συμπεριφορά που παράγεται με κάποια δεδομένα εισόδου. Παρατήρησε κατά πόσο η αντίδραση του συστήματος είναι η επιθυμητή.

- Μειονεκτήματα
  - ο αριθμός των πιθανών συμπεριφορών είναι συχνά τεράστιος ή/και μη-πεπερασμένος
  - κάποια από τις συμπεριφορές που δεν ελέγχθηκαν πιθανόν να περιέχει το μοιραίο λάθος

- Οι προσομοιώσεις και τα πειράματα μπορούν να δείξουν την ύπαρξη λαθών αλλά όχι την απουσία τους

Testing


μοντέλο ή πρωτότυπο
Σύστημα / μοντέλο συστήματος
Παραγωγή πειραμάτων
Εκτέλεση πειραμάτων
σουίτα πειραμάτων
pass
fail

Testing

- Είναι χρήσιμη τεχνική όταν
  - Είναι δύσκολο να κτιστεί μοντέλο του συστήματος
  - Υπομέρη του συστήματος (π.χ. συσκευές) δεν μπορούν να μοντελοποιηθούν
  - Ο κώδικας δεν είναι διαθέσιμος

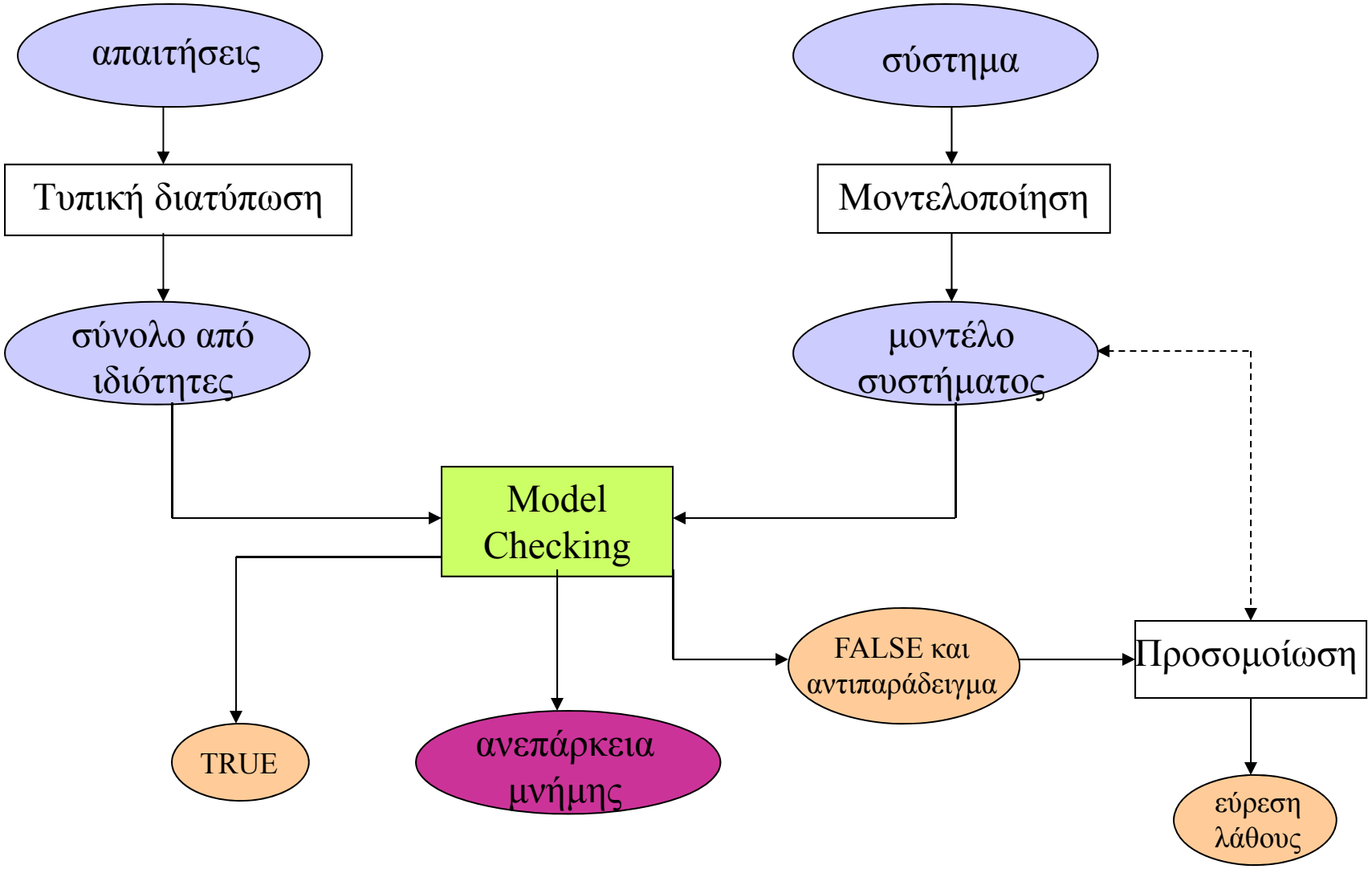
- Συμπληρωματική μέθοδος προς τον μοντέλο-έλεγχο αφού ελέγχει το ίδιο το σύστημα και όχι κάποιο μοντέλο αυτού.

Model-checking

- Επαναστατική μέθοδος για την αυτοματοποιημένη επαλήθευση συστημάτων
- Ελέγχει κατά πόσο ένα μοντέλο ικανοποιεί μια *τροπική* ιδιότητα π.χ.
  - Το σύστημα δίνει πάντα το σωστό αποτέλεσμα;
  - Το σύστημα περιέχει αδιέξοδα (deadlock); Για παράδειγμα όταν δύο παράλληλα προγράμματα περιμένουν το ένα το άλλο, σταματώντας με αυτό τον τρόπο ολόκληρο το σύστημα.
  - Μπορεί να εμφανιστεί αδιέξοδο μέσα σε μια ώρα από την εκκίνηση της εκτέλεσης του συστήματος;
  - Η απάντηση δίνεται πάντα μέσα σε 8 λεπτά;
- Βασίζεται σε συστηματικό έλεγχο του συνόλου καταστάσεων ενός συστήματος.

Ο μοντέλο-έλεγχος απαιτεί μια ακριβή και ξεκάθαρη δήλωση των ιδιοτήτων που θα ελεγχθούν. Αυτό γίνεται συνήθως σε μια χρονική λογική (temporal logic).

Model-checking



Πλεονεκτήματα του μοντέλο-ελέγχου

- Εφαρμόζεται σε πολλά συστήματα (υλικό, λογισμικό, πρωτόκολλα,…)
- Ύπαρξη εργαλείων
- Σε περίπτωση μη ικανοποίησης μας δίνει αντιπαράδειγμα
- Μαθηματικά αποδεδειγμένη
- Κοιτάζει όλες τις δυνατές εκτελέσεις
- Το ενδιαφέρον της βιομηχανίας για αυτό συνεχώς αυξάνεται

Περιορισμοί του μοντέλο-ελέγχου

- Ασχολείται κυρίως με εφαρμογές με πολύπλοκη ροή και όχι επεξεργασία δεδομένων
- Τα αποτελέσματα βασίζονται στην ποιότητα του μοντέλου
- Η πολυπλοκότητα (state-space explosion problem) μπορεί να αποτελέσει εμπόδιο
- Η δημιουργία κατάλληλων μοντέλων απαιτεί πείρα.

And the Turing award goes to…

- Milner, 1991: for the developments of logical formalisms in computer science (LCF, the mechanization of Scott's Logic of Computable Functions, probably the first theoretically based yet practical tool for machine assisted proof construction).

Robin Milner
(1934-2010)

- Pnueli, 1996: For seminal work introducing temporal logic into computing science and for outstanding contributions to program and systems verification.

Amir Pnueli
(1941-2009)

- Clarke, Emerson, Sifakis, 2007: For [their roles] in developing model checking into a highly effective verification technology, widely adopted in the hardware and software industries.

E. Allen Emerson (1954-)

Edmund M. Clarke (1945-)

Ιωσήφ Σιφάκης (1946 -)

Παραδείγματα Εφαρμογών

- Ασφάλεια: Το πρωτόκολλο Needham Schroeder
  - Εύρεση λάθους 17 χρόνια μετά από τη δημιουργία και διαδεδομένη χρήση του
- Λογισμικό σε διαστημικούς πυραύλους
  - Το Mars Pathfinder της NASA
- Λογισμικό σε αυτόνομα συστήματα
  - Boeing helicopter
- Συστήματα συγκοινωνιών
  - Μοντέλο τραίνων με 10^{476} καταστάσεις
- Εργαλεία μοντελο-ελέγχου για τις C, Java, C++

Τυπικό πρόβλημα παρεμβολής

process Inc = **while** *true* **do if** x < 200 **then** x = x + 1 **od**

process Dec = **while** *true* **do if** x > 0 **then** x = x – 1 **od**

process Reset = **while** *true* **do if** x = 200 **then** x = 0 **od**

Είναι πάντα το x ανάμεσα στο 0 και το 200;

Παράδειγμα μοντέλο-ελέγχου

```
int x = 0;

proctype Inc(){
   do :: true -> if :: (x < 200) -> x = x + 1 fi od
}
proctype Dec(){
   do :: true -> if :: (x > 0) -> x = x – 1  fi od
}
proctype Reset(){
   do :: true -> if :: (x = 200) -> x = 0 fi od
}

init{
   atomic{ run Inc(); run Dec(); run Reset()}
}
```

Παράδειγμα μοντέλο-ελέγχου

Για να ελέγξουμε τις τιμές του x εισάγουμε μία διεργασία που ελέγχει ότι $0 \leq x \leq 200$.

```
proctype Check(){
 assert (x >= 0 && x <= 200)
}

init{
   atomic{ run Inc(); run Dec(); run Reset();
           run Check()}
}
```

Με αυτό το δεδομένο το εργαλείο μας δίνει μήνυμα λάθους:

```
pan: assertion violated (x >= 0 && x <= 200)
(at  depth 1802)
Depth reached 3598, errors 1, 12609 states stored.
```

Το αντιπαράδειγμα

```
...........
606: proc 1 (Inc)   line  9 (state 2) [(x<200)]
607: proc 1 (Inc)   line  9 (state 3) [(x=x+1)]
608: proc 1 (Inc)   line  9 (state 1) [(1)]
609: proc 3 (Reset) line 13 (state 2) [(x==200)]
610: proc 3 (Reset) line 13 (state 3) [(x = 0)]
611: proc 3 (Reset) line 13 (state 1) [(1)]
612: proc 2 (Dec)   line  5 (state 3) [(x=x-1)]
613: proc 2 (Dec)   line  5 (state 1) [(1)]
spin: line 17 Error: assertion violated
```

Διόρθωση του λάθους

```
int x = 0;

proctype Inc(){
   do :: true -> atomic{if :: (x < 200) -> x=x+1 fi} od
}
proctype Dec(){
   do :: true -> atomic{ if :: (x > 0) -> x=x–1  fi} od
}
proctype Reset(){
   do :: true -> atomic{if :: (x = 200) -> x = 0 fi} od
}

init{
   atomic{ run Inc(); run Dec(); run Reset()}
}
```