ΕΠΑΛΗΘΕΥΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ Ι

- Η τυπική επαλήθευση βάση μοντέλου είναι κατάλληλη για συστήματα επικοινωνούντων διεργασιών (π.χ. κατανεμημένα συστήματα) όπου το βασικό πρόβλημα είναι ο έλεγχος αλλά γενικά δεν υπάρχουν σύνθετα δεδομένα. Βασικά στηριζόμαστε στην υπόθεση ότι τα μοντέλα που προκύπτουν από μία αφαιρετική θεώρηση των συστημάτων που περιγράφουν είναι ***πεπερασμένων καταστάσεων***.
- Αυτή όμως η υπόθεση δεν ισχύει για (σειριακά) εκτελέσιμα προγράμματα που μπορεί να διαχειρίζονται μη τετριμμένα δεδομένα και μπορεί π.χ. να έχουν μεταβλητές τύπου integer, λίστας, δέντρου οπότε ουσιαστικά μιλάμε για μηχανές με ***άπειρο χώρο καταστάσεων***.

ΕΠΑΛΗΘΕΥΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ ΙΙ

- Οι τεχνικές επαλήθευσης που χρησιμοποιούνται για προγράμματα δεν κάνουν εξαντλητικό έλεγχο κάθε κατάστασης του συστήματος αφού αυτό θα ήταν ανέφικτο μια και μιλάμε για άπειρα πολλές καταστάσεις.
- Αντί γι΄ αυτό συνήθως αποβλέπουμε στην κατασκευή μιας απόδειξης για την ιδιότητα που ενδιαφέρει χρησιμοποιώντας μία θεωρία αποδεικτικής που παρακάμπτει την απαίτηση να ελεγχθούν άπειρα πολλά μοντέλα ενός κάποιου συνόλου τύπων κατηγορηματικού λογισμού.

ΕΠΑΛΗΘΕΥΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ ΙΙΙ

- Παρά το γεγονός ότι πολλά από τα βήματα που απαιτούνται για να επαληθευτεί ότι ένα πρόγραμμα ικανοποιεί κάποια προδιαγραφή είναι μηχανιστικά (και άρα αυτοματοποιούνται) δεν υπάρχει εγγύηση ότι αυτό μπορεί να γίνει για όλες τις περιπτώσεις αλγοριθμικά. Αν και υπάρχουν προσεγγίσεις που βοηθούν τον προγραμματιστή να φέρει σε πέρας μία τέτοια διαδικασία, είναι βέβαιο ότι μιλάμε για μία ημιαυτόματη διαδικασία.
- ***Παράδειγμα τεχνικής επαλήθευσης προγραμμάτων***: Hoare triples
- ***Παράδειγμα προγραμματιστικών βοηθημάτων***: Java Modeling Language ή Object Constraint Language (C++)

ΕΠΑΛΗΘΕΥΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ IV

Για ποιο λόγο χρειαζόμαστε την επαλήθευση προγραμμάτων;

- Η τυπική προδιαγραφή ενός προγράμματος αποτελεί σημαντικό κομμάτι της τεκμηρίωσής του. Η λογική δομή της τυπικής περιγραφής εκφράζεται από έναν τύπο σε μία κατάλληλη γλώσσα λογικής, που βασικά είναι το αντικείμενο μιας απόδειξης.
- Η επαλήθευση προγραμμάτων ως προς τυπικές προδιαγραφές μειώνει το χρόνο συντήρησης και απλοποιεί την ανάπτυξη λογισμικού, γιατί απομακρύνει τα περισσότερα λάθη στη φάση της σχεδίασης.
- Λογισμικό που έχει προδιαγραφεί τυπικά και επαληθευθεί είναι πιο εύκολο να το επαναχρησιμοποιήσουμε γιατί έχουμε μία ξεκάθαρη εικόνα για το τι αυτό κάνει.
- Στα συστήματα κρίσιμης ασφάλειας η τυπική προδιαγραφή και επαλήθευση είναι απαρέγκλιτη προϋπόθεση.

ΕΠΑΛΗΘΕΥΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ V

Διαδικασία κατασκευής λογισμικού με τυπική προδιαγραφή και επαλήθευση

- Μετατροπή μιας μη τυπικής περιγραφής *D* του πεδίου του προβλήματος σε έναν «ισοδύναμο» τύπο ϕ_D σε κάποια συμβολική λογική.
- Κατασκευή προγράμματος *P* που υλοποιεί τον ϕ_D μέσα σε ένα προγραμματιστικό περιβάλλον.
- Απόδειξη ότι το πρόγραμμα *P* ικανοποιεί τον τύπο ϕ_D.

ΤΡΙΠΛΕΤΕΣ HOARE I

- Ένα πρόγραμμα διατυπωμένο στη σύνταξη κάποιας γλώσσας όταν εκτελείται φτάνει τη μηχανή σε μία κάποια ‘κατάσταση’ από την οποία μετά από υπολογισμούς μπορεί να τερματίζει. Αν αυτό συμβαίνει τότε το αποτέλεσμα είναι μία βασικά διαφορετική κατάσταση.
- Η κατάσταση της μηχανής μπορεί να αναπαρασταθεί από ένα διάνυσμα τιμών που αντιπροσωπεύουν όλες τις μεταβλητές του προγράμματος (αν αυτό δεν έχει υποπρογράμματα).
- Τι σύνταξη θα χρησιμοποιήσουμε για τον τύπο ϕ_D, δηλ. την τυπική προδιαγραφή του προγράμματος; Θα πρέπει να μπορούμε να μιλάμε για τις μεταβλητές που αναφέρονται στην κατάσταση μετά την εκτέλεση του προγράμματος χρησιμοποιώντας τελεστές όπως =, < κλπ.

ΤΡΙΠΛΕΤΕΣ HOARE II

ΠΑΡΑΔΕΙΓΜΑ

«Υπολόγισε έναν αριθμό y
που το τετράφωνό του είναι μικρότερο του x»

ΠΡΟΔΙΑΓΡΑΦΗ: $y\ y < x$

ΕΡΩΤΗΜΑ: «Τι συμβαίνει αν το x είναι αρνητικός;»

ΑΝΑΘΕΩΡΗΜΕΝΗ ΠΕΡΙΓΡΑΦΗ: «Αν η είσοδος x είναι θετικός, τότε υπολόγισε έναν αριθμό y που το τετράγωνό του είναι μικρότερο από x»

ΣΥΜΠΕΡΑΣΜΑ: Πρέπει όχι απλά να μπορούμε να μιλάμε για την κατάσταση μετά από την εκτέλεση του προγράμματος αλλά και για την κατάσταση πριν την εκτέλεσή του.

ΤΡΙΠΛΕΤΕΣ HOARE III

- Δηλαδή, ο ***ισχυρισμός*** (assertion) που κάνουμε θα έχει τη μορφή τριπλέτας

$$(\phi)\ \boldsymbol{P}\ (\psi)$$

που σημαίνει ότι αν το πρόγραμμα $\boldsymbol{P}$ εκτελεστεί σε μία κατάσταση που ικανοποιεί τον τύπο ϕ, τότε η κατάσταση μετά από την εκτέλεση ικανοποιεί τον τύπο ψ.

- Η τυπική προδιαγραφή λοιπόν του προγράμματος $\boldsymbol{P}$ παίρνει τελικά την μορφή

$$(x > 0)\ \boldsymbol{P}\ (y\ y < x)$$

που σημαίνει ότι αν το πρόγραμμα $\boldsymbol{P}$ βρίσκεται σε κατάσταση για την οποία ισχύει ότι $x > 0$, τότε στην κατάσταση τη στιγμή του αποτελέσματος θα ισχύει $y\ y < x$. Δεν μας λέει τι θα γίνει αν $x \leq 0$.

ΤΡΙΠΛΕΤΕΣ HOARE IV

- Στην τριπλέτα Hoare

$$(\phi)\ P\ (\psi)$$

ο τύπος ϕ λέγεται ***προαπαιτούμενο*** (precondition) του P και ο τύπος ψ λέγεται ***συνέπεια*** (postcondition) του P.

- Συχνά δεν θέλουμε να βάλουμε κάποιους περιορισμούς ως προαπαιτούμενο δηλ. υπονοούμε ότι άσχετα με την κατάσταση εκκίνησης του προγράμματος θέλουμε ως συνέπεια να ικανοποιεί στην κατάσταση του αποτελέσματος τον τύπο ψ. Σε αυτή την περίπτωση στη θέση του προαπαιτούμενου χρησιμοποιούμε το σύμβολο $\top$ που όπως είδαμε στον προτασιακό λογισμό είναι πάντα **true**.

ΤΡΙΠΛΕΤΕΣ HOARE V

- **ΠΡΟΣΟΧΗ**: Η τριπλέτα Hoare δεν προδιαγράφει ένα συγκεκριμένο πρόγραμμα ***P*** ή μία μοναδική συμπεριφορά. Έτσι το πρόγραμμα

```
y = 0;
```

και το πρόγραμμα

```
y = 0;
while (y*y < x) {
        y = y + 1;
        }
y = y - 1;
```

ικανοποιούν αμφότερα την προδιαγραφή τριπλέτας που θέσαμε, ***παρόλο που έχουν ως αποτέλεσμα διαφορετικές τιμές***.

ΤΡΙΠΛΕΤΕΣ HOARE VI

- Στη συνέχεια μας ενδιαφέρει να αναπτύξουμε την έννοια της απόδειξης που θα επιτρέπει να αποδείξουμε ότι ένα πρόγραμμα P ικανοποιεί την τυπική προδιαγραφή δοθέντος του προαπαιτούμενο ϕ και της συνέπειας ψ.

- Στον προτασιακό λογισμό που είδαμε, αλλά και στον κατηγορηματικό λογισμό που δεν είδαμε μπορούμε να κάνουμε τέτοιες αποδείξεις αναλύοντας τη λογική δομή του τύπου που θέλουμε να αποδείξουμε.

  Αν π.χ. θέλουμε να αποδείξουμε ότι $\phi \rightarrow \psi$ τότε υποθέτουμε ότι ισχύει ο ϕ και αν αποδείξουμε ότι ισχύει ο ψ, τότε μπορούμε να εφαρμόσουμε τον αποδεικτικό κανόνα $\rightarrow$ i και έτσι να ολοκληρώσουμε την απόδειξη.

ΤΡΙΠΛΕΤΕΣ HOARE VII

- Για τις τριπλέτες Hoare ο αποδεικτικός λογισμός που θα φτιάξουμε πρέπει να μπορεί να χειρίζεται δύο διαφορετικά πράγματα: λογικούς τύπους σε σχέση με τον κώδικα ενός προγράμματος ***P***.

- Θα πρέπει ο αποδεικτικός λογισμός επίσης να είναι ***συνθετικός*** σε σχέση με τη δομή του προγράμματος ***P***, δηλ. να μπορούμε από απλούστερες προδιαγραφές να αποδεικνύουμε πιο σύνθετες. Έτσι θα είναι εφαρμόσιμος σε μεγάλα προγράμματα που π.χ. περιέχουν υποπρογράμματα ή και τμήματα προγραμμάτων (modules).

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ Ι

- ***Μερική Ορθότητα (partial correctness)***

  Λέμε ότι η τριπλέτα $(\phi)\ P\ (\psi)$ ικανοποιείται με σημασία μερικής ορθότητας αν για όλες οι καταστάσεις που ικανοποιούν τον τύπο ϕ η κατάσταση που προκύπτει από την εκτέλεση του P ικανοποιεί τη συνέπεια ψ, με την προϋπόθεση ότι το P τερματίζει. Αυτό το γράφουμε ως

  $$\models_{\text{par}} (\phi)\ P\ (\psi)$$

  Η σχέση $\models_{\text{par}}$ λέμε ότι είναι η σχέση ικανοποίησης για μερική ορθότητα.

  ΕΊΝΑΙ ΟΧΙ ΙΔΙΑΙΤΕΡΑ ΙΣΧΥΡΗ ΕΓΓΥΗΣΗ-ΠΑΡΑΔΕΙΓΜΑ:

  ```
  while true { x = 0; }
  ```

  Ικανοποιεί με μερική ορθότητα κάθε πιθανή προδιαγραφή αφού η μερική ορθότητα αναφέρεται στην αλήθεια ισχυρισμών μόνο για την περίπτωση που το πρόγραμμα τερματίζει.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ ΙΙ

- ***Ολική Ορθότητα (total correctness)***

  Λέμε ότι η τριπλέτα $(\phi)\ P\ (\psi)$ ικανοποιείται με σημασία ολικής ορθότητας αν για όλες οι καταστάσεις που ικανοποιούν τον τύπο ϕ το πρόγραμμα P τερματίζει και η κατάσταση που προκύπτει από την εκτέλεση του P ικανοποιεί τη συνέπεια ψ. Αυτό το γράφουμε ως

  $$\models_{\text{tot}} (\phi)\ P\ (\psi)$$

  Η σχέση $\models_{\text{tot}}$ λέμε ότι είναι η σχέση ικανοποίησης για ολική ορθότητα.

ΠΑΡΑΔΕΙΓΜΑ:

Πρόγραμμα που κάνει ατέρμονο loop δεν ικανοποιεί καμία προδιαγραφή με σημασία ολικής ορθότητας.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ ΙΙΙ

- Η απόδειξη της ολικής ορθότητας συνήθως γίνεται σε δύο βήματα: αποδεικνύεται πρώτα η μερική ορθότητα και στη συνέχεια αποδεικνύεται ότι το πρόγραμμα τερματίζει.

ΠΑΡΑΔΕΙΓΜΑ: πρόγραμμα `succ`

```
a = x + 1;
if (a - 1 == 0) {
        y = 1;
} else {
        y = a;
}
```

Το πρόγραμμα `succ` ικανοποιεί την προδιαγραφή

$$(\top)\ \texttt{succ}\ (\texttt{y} = (\texttt{x} + 1))$$

Οι αποδεικτικοί κανόνες πρέπει να είναι δυνατό να αποδείξουν τον παραπάνω ισχυρισμό. Θα μπορούσε το πρόγραμμα να είναι απλούστερο.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ IV

ΠΑΡΑΔΕΙΓΜΑ: πρόγραμμα `fac1`

```
y = 1;
z = 0;
while (z != x) {
        z = z + 1;
        y = y * z;
}
```

Αυτό το πρόγραμμα τερματίζει μόνο αν το `x` είναι αρχικά μη αρνητικός (γιατί;).

Θα μπορούσε να αποδειχθεί ότι

$$\models_{\text{tot}} (x \geq 0)\ \text{fac1}\ (y = x!)$$

Οι αποδεικτικοί κανόνες πρέπει να είναι δυνατό να αποδείξουν τον παραπάνω ισχυρισμό. Παρόλα αυτά, δεν πρέπει να ισχύει ότι

$$\models_{\text{tot}} (\top)\ \text{fac1}\ (y = x!)$$

Με σημασία μερικής ορθότητας ισχύουν και τα δύο.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ V

- Αν απλά ο λογισμός μπορεί να αποδείξει τριπλέτες της μορφής $(\phi)\ P\ (\psi)$ τότε αν αυτό γίνεται για σημασία μερικής ορθότητας το γράφουμε

$$\vdash_{\text{par}} (\phi)\ P\ (\psi)$$

ενώ αν γίνεται με σημασία ολικής ορθότητας τότε το γράφουμε

$$\vdash_{\text{tot}} (\phi)\ P\ (\psi)$$

- Δηλ. το $\models_{\text{par}} (\phi)\ P\ (\psi)$ σημαίνει ότι το P ***είναι*** μερικώς ορθό ενώ το $\vdash_{\text{par}} (\phi)\ P\ (\psi)$ υπονοεί ότι ***μπορεί ο λογισμός να αποδείξει*** ότι είναι μερικώς ορθό.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ VI

- Ένας αποδεικτικός λογισμός λέμε ότι είναι ***καλά ορισμένος (sound)*** αν σε όλες τις περιπτώσεις που είναι δυνατό να αποδειχθεί ένας ισχυρισμός, αυτός ο ισχυρισμός είναι πράγματι αληθής. Για τη μερική ορθότητα αυτό το γράφουμε ως εξής:

$$\vdash_{\text{par}} (\phi)\ \boldsymbol{P}\ (\psi) \text{ συνεπάγεται ότι } \models_{\text{par}} (\phi)\ \boldsymbol{P}\ (\psi)$$

για όλους τους τύπους ϕ, ψ και προγράμματα $\boldsymbol{P}$.

Για την ολική ορθότητα αυτό το γράφουμε ως εξής:

$$\vdash_{\text{tot}} (\phi)\ \boldsymbol{P}\ (\psi) \text{ συνεπάγεται ότι } \models_{\text{tot}} (\phi)\ \boldsymbol{P}\ (\psi)$$

για όλους τους τύπους ϕ, ψ και προγράμματα $\boldsymbol{P}$.

ΜΕΡΙΚΗ ΚΑΙ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ VII

- Ένας αποδεικτικός λογισμός λέμε ότι είναι ***πλήρης (complete)*** αν σε όλες τις περιπτώσεις μπορεί να αποδειχθεί ένας ισχυρισμός που είναι πράγματι αληθής. Για τη μερική ορθότητα αυτό το γράφουμε ως εξής:

  $$\models_{\text{par}} (\phi)\ P\ (\psi) \text{ συνεπάγεται ότι } \vdash_{\text{par}} (\phi)\ P\ (\psi)$$

  για όλους τους τύπους ϕ, ψ και προγράμματα P.

- Γενικά όπως σε κάθε λογισμό έτσι και σε έναν λογισμό επαλήθευσης προγραμμάτων η απόδειξη του καλώς ορισμένου (soundness) γίνεται για κάθε αποδεικτικό κανόνα ανεξάρτητα από τους άλλους. Αντίθετα, η απόδειξη της πληρότητας (completeness) είναι πιο σύνθετη αφού εξαρτάται από το πώς οι αποδεικτικοί κανόνες του λογισμού αλληλεπιδρούν μεταξύ τους.

ΜΕΤΑΒΛΗΤΕΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ ΛΟΓΙΣΜΟΥ Ι

ΠΑΡΑΔΕΙΓΜΑ: πρόγραμμα `fac2`

```
y = 1;
while (x != 0) {
        y = y * x;
        x = x - 1;
}
```

Αυτό το πρόγραμμα «καταναλώνει» την είσοδο `x`. Υπολογίζει σωστά το παραγοντικό της `x` και αποθηκεύει την τιμή στην `y` και θα θέλαμε αυτό να το γράψουμε σε μια τριπλέτα Hoare:

$$(x \geq 0)\ \texttt{fac2}\ (y = x!)$$

Αυτός όμως ο συμβολισμός δε βολεύει γιατί αν το πρόγραμμα τερματίζει τότε η `x` θα είναι `0` και η `y` θα είναι το παραγοντικό της αρχικής τιμής της `x`.

Πρέπει με κάποιο τρόπο να «θυμόμαστε» την αρχική τιμή της `x`.

ΜΕΤΑΒΛΗΤΕΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ ΛΟΓΙΣΜΟΥ ΙΙ

ΠΑΡΑΔΕΙΓΜΑ: πρόγραμμα `fac2` (συνέχεια)

Πρέπει με κάποιο τρόπο να «θυμόμαστε» την αρχική τιμή της `x`. Αυτό γίνεται με τη χρήση ***μεταβλητών εμβέλειας λογισμού***. Έτσι, η τριπλέτα Hoare του παραδείγματος γράφεται

$$(\texttt{x} = x_0 \wedge \texttt{x} \geq \texttt{0})\ \texttt{fac2}\ (\texttt{y} = x_0!)$$

και διαβάζεται ως: «***για όλους τους ακεραίους*** x_0 αν η `x` είναι ίση με x_0, $\texttt{x} \geq \texttt{0}$ και εκτελείται το πρόγραμμα έτσι ώστε να τερματίζει, τότε η κατάσταση του αποτελέσματος θα ικανοποιεί την $\texttt{y} = x_0!$»

- Οι μεταβλητές εμβέλειας λογισμού δεν εμφανίζονται πουθενά μέσα στο πρόγραμμα. Η κατάσταση στην οποία βρίσκεται το σύστημα υπονοεί μία συγκεκριμένη τιμή για κάθε μεταβλητή του προγράμματος, αλλά όχι για τις μεταβλητές λογισμού. Οι μεταβλητές αυτές παίζουν ρόλο ανάλογο με αυτές που συνήθως χρησιμοποιούμε στον κατηγορηματικό λογισμό με ποσοδείκτες π.χ. $\forall$`i` και $\exists$`e`.

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ Ι

- Οι κανόνες του αποδεικτικού λογισμού μας επιτρέπουν να περάσουμε από απλούς ισχυρισμούς της μορφής

  $$(\phi)\ P\ (\psi)$$

  σε πιο σύνθετους. Ο κανόνας της ανάθεσης τιμής δεν έχει προϋποθέσεις, που σημαίνει ότι μπορούμε να φτιάξουμε τριπλέτα από το τίποτα προκειμένου να εκπληρώσουμε το σκοπό της απόδειξης.

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ ΙΙ

$$\frac{(\phi)\, C_1\, (\eta) \quad (\eta)\, C_2\, (\psi)}{(\phi)\, C_1\, ; C_2\, (\psi)}$$ **σύνθεση (c)**

$$\frac{}{(\psi[\mathrm{E}/x])\, x = \mathrm{E}\, (\psi)}$$ **ανάθεση (a)**

$$\frac{(\phi \wedge B)\, C_1\, (\psi) \quad (\phi \wedge \neg B)\, C_2\, (\psi)}{(\phi)\ \text{if}\ B\ \{C_1\}\ \text{else}\ \{C_2\}\ (\psi)}$$ **εντολή *if* (i)**

$$\frac{(\psi \wedge B)\, C\, (\psi)}{(\psi)\ \texttt{while}\ \ B\ \{C\}\ (\psi \wedge \neg\ B)}$$ **μερικό while (w)**

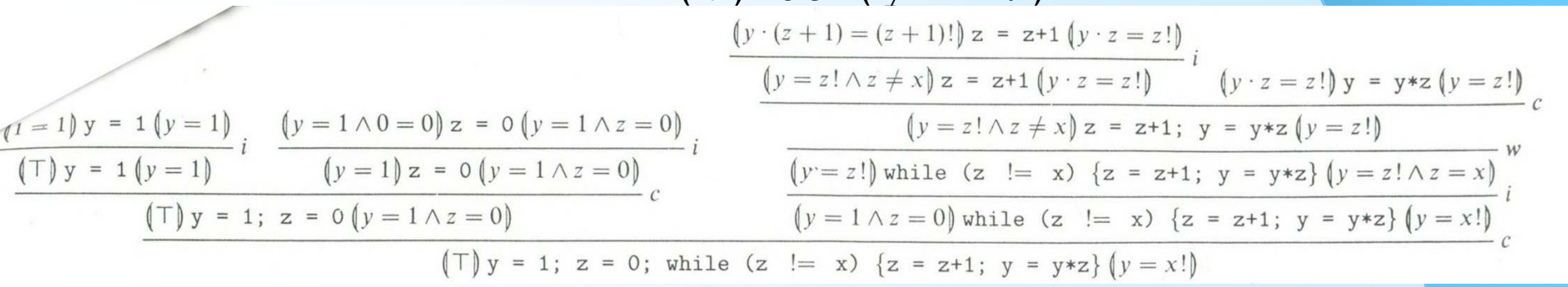
$$\frac{\vdash\ \phi' \rightarrow \phi \quad (\phi)\ \ C\ \ (\psi) \quad \vdash\ \ \psi \rightarrow \psi'}{(\phi')\, C\ \ (\psi')}$$ **συνεπαγωγή (i)**

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ ΙΙΙ

ΠΑΡΑΔΕΙΓΜΑ: πρόγραμμα `fac1`

```
y = 1;
z = 0;
while (z != x) {
        z = z + 1;
        y = y * z;
}
```

$(\top)$ `fac1` $(y = x!)$



απόδειξη σε μορφή δέντρου

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ IV

Έστω ότι ένα πρόγραμμα P αποτελείται από τις εντολές C_1; C_2; . . .; C_{n-1}; C_n όπου κάθε μία από τις C_i είναι μία από τις εντολές ανάθεσης, `if` ή `while` και καμία δεν είναι σύνθεση άλλων απλούστερων εντολών. Έστω ότι θέλουμε να αποδείξουμε:

$$\vdash_{\text{par}} (\phi_0)\ \boldsymbol{P}\ (\phi_n)$$

Τότε μπορούμε να διασπάσουμε το πρόβλημα σε μικρότερα προβλήματα αν προσπαθήσουμε να βρούμε τύπους $\phi_j (0 < j < n)$ τέτοιους ώστε οι $\vdash_{\text{par}} (\phi_i)\ \boldsymbol{C}_{i+1}\ (\phi_{i+1})$ να ισχύουν για όλα τα $i = 0, 1, \ldots, n-1$. Έτσι η απόδειξή μας παίρνει τελικά τη μορφή

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ V

(ϕ_0)

C_1;

(ϕ_1) αιτιολόγηση

C_2;

.

.

(ϕ_{n-1}) αιτιολόγηση

C_n;

(ϕ_n) αιτιολόγηση

απόδειξη σε μορφή «ταμπλό»: είναι πιο βολικό αρχίζοντας από την ϕ_n να προσπαθούμε μέσω της C_n να βρούμε την ϕ_{n-1} κ.λ.π. δηλ. να κινούμαστε από το τέλος προς την αρχή

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ VI

- Για τη διαδικασία υπολογισμού της $\boldsymbol{\phi_i}$ από την C_{i+1} και την $\boldsymbol{\phi_{i+1}}$ λέμε ότι ***υπολογίζουμε το ασθενέστερο προαπαιτούμενο*** (weakest precondition) της C_{i+1} δοθείσης της συνέπειας $\boldsymbol{\phi_n}$.

  Αυτό σημαίνει ότι ψάχνουμε για τη λογικά ασθενέστερη συνθήκη που η αλήθεια της στην αρχή της εκτέλεσης της C_{i+1} είναι αρκετή για να εγγυηθεί την $\boldsymbol{\phi_{i+1}}$. Ένας τύπος $\boldsymbol{\phi}$ λέμε ότι είναι ασθενέστερος από τον τύπο $\boldsymbol{\psi}$, αν ο $\boldsymbol{\phi}$ συνεπάγεται από τον τύπο $\boldsymbol{\psi}$ στον κατηγορηματικό λογισμό.

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ VII

- Αξίωμα ανάθεσης σε απόδειξη «ταμπλό»:

  $(\psi[E/x])$

  $x = E$

  (ψ) ***ανάθεση***

- Συνεπαγωγή σε απόδειξη «ταμπλό»:

  μας επιτρέπει να γράψουμε έναν τύπο ϕ_2 ακριβώς κάτω από έναν άλλο τύπο ϕ_1 - χωρίς να παρεμβάλλεται κώδικας - αν ο ϕ_2 συνεπάγεται από τον ϕ_1

  Ένας τύπος συνεπάγεται από έναν άλλο τύπο αν αυτό αποδεικνύεται στον κατηγορηματικό λογισμό (στην επαλήθευση προγραμμάτων η απόδειξη παραλείπεται).

  Επίσης χρησιμοποιείται όταν τελικά εμφανίζεται το ασθενέστερο προαπαιτούμενο.

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ VIII

ΠΑΡΑΔΕΙΓΜΑ

Θέλουμε να αποδείξουμε ότι

$$\vdash_{par} (y = 5)\ \texttt{x = y + 1}\ (x = 6)$$

$(y = 5)$	
$(y + 1 = 6)$	συνεπαγωγή
`x = y + 1`	
$(x = 6)$	ανάθεση

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ IX

ΠΑΡΑΔΕΙΓΜΑ

Θέλουμε να αποδείξουμε ότι $\vdash_{par} (\tau)\ \boldsymbol{P}\ (u = x + y)$

όπου ***P***

```
z = x;
z = z + y;
u = z;
```

(τ)	
$(x + y = x + y)$	συνεπαγωγή
`z = x;`	
$(z + y = x + y)$	ανάθεση
`z = z + y;`	
$(z = x + y)$	ανάθεση
`u = z;`	
$(u = x + y)$	ανάθεση

Η τ ικανοποιείται σε όλες τις καταστάσεις αλλά το ίδιο ισχύει με την $x + y = x + y$.

Άρα ισχύει η $\tau \rightarrow x + y = x + y$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ Χ

- Εντολή if σε απόδειξη «ταμπλό»:

  έστω ότι θέλουμε δοθείσης της ψ να υπολογίσουμε την ασθενέστερη ϕ που είναι τέτοια ώστε

  $$(\ \phi\ )\ \texttt{if}\ (\texttt{B})\ \{\texttt{C}_1\}\ \texttt{else}\ \{\texttt{C}_2\}\ (\psi)$$

  Η ϕ υπολογίζεται ως εξής:

  1. Κινούμαστε από την ψ προς τα πάνω μέσω της $\texttt{C}_1$ που μπορεί βέβαια να είναι όχι μία εντολή αλλά ακολουθία εντολών. Έστω ότι το αποτέλεσμα είναι ο τύπος ϕ_1.
  2. Κάνουμε το ίδιο από την ψ προς τα πάνω μέσω της $\texttt{C}_2$ και έστω ότι το αποτέλεσμα είναι ο τύπος ϕ_2.
  3. Θέτουμε τον τύπο ϕ να είναι $(\texttt{B} \rightarrow \phi_1) \wedge (\neg\texttt{B} \rightarrow \phi_2)$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XI

ΠΑΡΑΔΕΙΓΜΑ

Έστω το πρόγραμμα `succ` με τον κώδικα

```
a = x + 1;
if (a - 1 ==0) {
    y = 1;
} else {
    y = a;
}
```

και έστω ότι θέλουμε να αποδείξουμε $\vdash_{par} (\top)$ `succ` $(y = x + 1)$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XII

(τ)	
$((x+1-1=0 \rightarrow 1 = x + 1) \wedge (\neg(x+1-1=0) \rightarrow x+1 = x + 1))$	
`a= x + 1;`	
$((a-1=0 \rightarrow 1 = x + 1) \wedge (\neg(a-1=0) \rightarrow a = x + 1))$	ανάθεση
`if (a - 1 ==0)`	
$(1 = x + 1)$	εντολή if
`y = 1;`	
$(y = x + 1)$	ανάθεση
`} else {`	
$(a = x + 1)$	εντολή if
`y = a;`	
$(y = x + 1)$	ανάθεση
`}`	
$(y = x + 1)$	εντολή if

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XIII

- Το βήμα που προτείνεται για τον κανόνα `if` σε απόδειξη «ταμπλό» παραπέμπει σε ένα διαφορετικό αποδεικτικό κανόνα από αυτόν που παρουσιάστηκε που όμως μπορεί να αποδειχθεί ότι προκύπτει με χρήση των κανόνων που εισάγαμε.

$$\frac{(\phi_1)\, C_1\, (\psi) \quad (\phi_2)\, C_2\, (\psi)}{\big((B \to \phi_1) \wedge (\neg B \to \phi_2)\big) \text{if } B \,\{C_1\} \text{ else } \{C_2\}\, (\psi)}$$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XIV

- Εντολή while σε απόδειξη «ταμπλό»:

$$\frac{(\eta \wedge B)\, C\, (\eta)}{(\eta)\ \text{while}\ \ B\, \{C\}\, (\eta \wedge \neg B)}$$

Από τον αποδεικτικό κανόνα φαίνεται ότι ο τύπος η επιλέγεται έτσι ώστε να είναι *αμετάβλητο* από το σώμα εντολών C της `while`. Αυτό σημαίνει ότι αν η B είναι αληθής και η η είναι αληθής πριν από την εκτέλεση της C και αν το σώμα εντολών C τερματίζει, τότε η η είναι επίσης αληθής στο τέλος της εκτέλεσης.

- Η ερμηνεία αυτή είναι συμβατή με την έννοια της μερικής ορθότητας.

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XV

- Παρόλα αυτά, στη μορφή που έχει ο αποδεικτικός κανόνας μας επιτρέπει να αποδεικνύουμε μόνο τριπλέτες όπου η συνέπεια είναι η ίδια με το προαπαιτούμενο μαζί βέβαια με το $\neg B$, δηλαδή τριπλέτες της μορφής

$$(\ \eta\ )\ \texttt{while}\ (\texttt{B})\ \{\texttt{C}\}\ (\eta \wedge \neg B\ )$$

- Τι γίνεται αν θέλουμε να αποδείξουμε τριπλέτες της μορφής

$$(\ \varphi\ )\ \texttt{while}\ (\texttt{B})\ \{\texttt{C}\}\ (\psi\ )$$

όπου οι τύποι φ και ψ δε σχετίζονται μεταξύ τους;

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XVI

- Η απάντηση είναι ότι πρέπει να βρούμε έναν κατάλληλο τύπο η τέτοιο ώστε να ισχύουν οι

  $\vdash \varphi \rightarrow \eta$

  $\vdash \eta \wedge \neg B \rightarrow \psi$

  $(\ \eta\ )$ `while (B) {C}` $(\eta \wedge \neg B\ )$

- Αν ισχύουν αυτά τότε μπορούμε με τον κανόνα της συνεπαγωγής να αποδείξουμε το

  $(\ \varphi\ )$ `while (B) {C}` $(\psi\ )$

- Το πιο κρίσιμο βήμα της απόδειξης είναι η επιλογή του ***αμετάβλητου*** (invariant).

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XVII

- **Ορισμός**: Ένα ***αμετάβλητο*** (invariant) σε μία εντολή `while (B) {C}` είναι ένας τύπος η τέτοιος ώστε

$$\models_{par} (\eta \wedge B)\ C\ (\eta)$$

δηλαδή αν ο τύπος η και η B είναι αληθείς σε μία κατάσταση στην οποία εκτελείται το `C` και τερματίζει, τότε η η είναι επίσης αληθής στην κατάσταση που προκύπτει ως αποτέλεσμα.

- Ένα χρήσιμο αμετάβλητο συνήθως εκφράζει μία σχέση μεταξύ των μεταβλητών του σώματος εντολών C, μία σχέση που διατηρείται από το loop ακόμη και αν οι τιμές των μεταβλητών αλλάξουν. Ένας τρόπος για να ανακαλύπτουμε αμετάβλητο είναι να δημιουργήσουμε ένα ίχνος εκτέλεσης του προγράμματος (trace).

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XVIII

ΠΑΡΑΔΕΙΓΜΑ

Έστω το πρόγραμμα `fac1` με τον κώδικα

```
y = 1;
z = 0;
while (z != x) {
      z = z + 1;
      y = y * z;
}
```

επανάληψη	z	y	B
0	0	1	true
1	1	1	true
2	2	2	true
3	3	6	true
4	4	4	true
5	5	120	true
6	6	720	false

και έστω ότι το παραπάνω πρόγραμμα εκτελείται για `x = 6`.

Στο παράδειγμα αυτό το αμετάβλητο εντοπίζεται εύκολα:

`y = z!`

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XIX

ΠΑΡΑΔΕΙΓΜΑ (συνέχεια)

Το αμετάβλητο που εντοπίστηκε επίσης διαθέτει τις ιδιότητες που επιθυμούμε:

- είναι αρκετά ασθενές ώστε τελικά συνεπάγεται από το προαπαιτούμενο της εντολής while που είναι $y = 1 \wedge z = 0$ (θυμίζουμε ότι 0! =1)
- είναι και αρκετά ισχυρό έτσι ώστε μαζί με την άρνηση της συνθήκης του while να συνεπάγεται τη συνέπεια

$$y = x!$$

Δηλαδή ισχύουν οι συνεπαγωγές

$\vdash (y = 1 \wedge z = 0) \rightarrow (y = z!)$

και $\vdash (y = z! \wedge x = z) \rightarrow (y = x!)$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ ΧΧ

- Εντολή while σε απόδειξη «ταμπλό»:
  1. Εντοπίζουμε έναν τύπο η που ελπίζουμε να είναι ένα κατάλληλο αμετάβλητο.
  2. Αποδεικνύουμε ότι $\vdash \eta \wedge \neg B \rightarrow \psi$ όπου B είναι η λογική συνθήκη της εντολής while. Αυτό δείχνει ότι η η είναι αρκετά ισχυρή ώστε να συνεπάγεται της συνέπεια.
  3. Κινούμαστε με τον τύπο η μέσα από το σώμα του loop C εφαρμόζοντας τους κανόνες που πρέπει ανάλογα με τη μορφή του C. Έστω ότι ο τύπος που προκύπτει είναι ο η'.
  4. Αποδεικνύουμε ότι $\eta \wedge B \rightarrow \eta'$, που ουσιαστικά αποδεικνύει ότι το η είναι αμετάβλητο.
  5. Τέλος, γράφουμε την η πάνω από την εντολή while και βέβαια είναι ακόμη καλύτερα αν μπορούμε να αποδείξουμε $\vdash \varphi \rightarrow \eta$

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΜΕΡΙΚΗ ΟΡΘΟΤΗΤΑ XXI

ΠΑΡΑΔΕΙΓΜΑ (συνέχεια)

(τ)	
(1 = 0 !)	συνεπαγωγή
`y = 1;`	
(y = 0 !)	ανάθεση
`z = 0;`	
(y = z !)	ανάθεση
`while (z != x) {`	
(y = z ! ∧ z ≠ x)	υπόθεση αμετάβλητου & συνθήκη
(y * (z + 1) = (z + 1)!)	συνεπαγωγή
`z = z + 1;`	
(y * z = z !)	ανάθεση
y = y * z;	
(y = z !)	ανάθεση
}	
(y = z ! ∧ ¬(z ≠ x))	μερικό while
(y = x !)	συνεπαγωγή

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ Ι

- Για την ολική ορθότητα πρέπει
  - να κάνουμε την απόδειξη για τη μερική ορθότητα
  - να αποδείξουμε ότι η δοθείσα εντολή while τερματίζει
- Όλοι οι αποδεικτικοί κανόνες είναι ίδιοι εκτός από το while
- Στο while η απόδειξη τερματισμού προϋποθέτει τον εντοπισμό μιας ακέραιης έκφρασης που η τιμή της μειώνεται κάθε φορά που εκτελείται το σώμα του loop μία φορά αλλά ποτέ δε γίνεται αρνητική. Αν μπορεί να βρεθεί μία τέτοια έκφραση αυτή λέγεται ***μεταβλητό μέρος*** και ουσιαστικό η ύπαρξή της αποδεικνύει ότι το loop τερματίζει.
- Για το πρόγραμμα `fac1` ένα κατάλληλο μεταβλητό μέρος είναι το

```
x - z
```

ΑΠΟΔΕΙΚΤΙΚΟΣ ΛΟΓΙΣΜΟΣ ΓΙΑ ΟΛΙΚΗ ΟΡΘΟΤΗΤΑ ΙΙ

- Αποδεικτικός κανόνας ολικού while

$$\frac{(\eta \wedge B \wedge 0 \leq E = E_0)\, C\, (\eta \wedge 0 \leq E < E_0)}{(\eta \wedge 0 \leq E)\ \text{while}\ B\, \{C\}\, (\eta \wedge \neg B)}$$