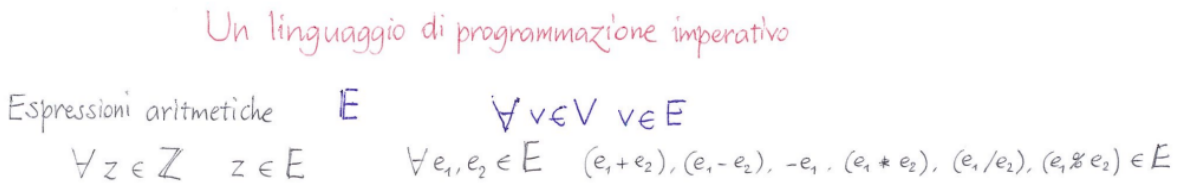
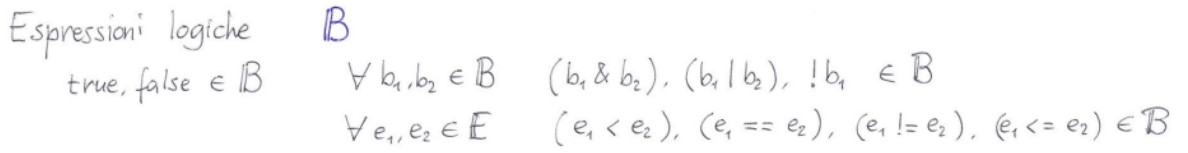
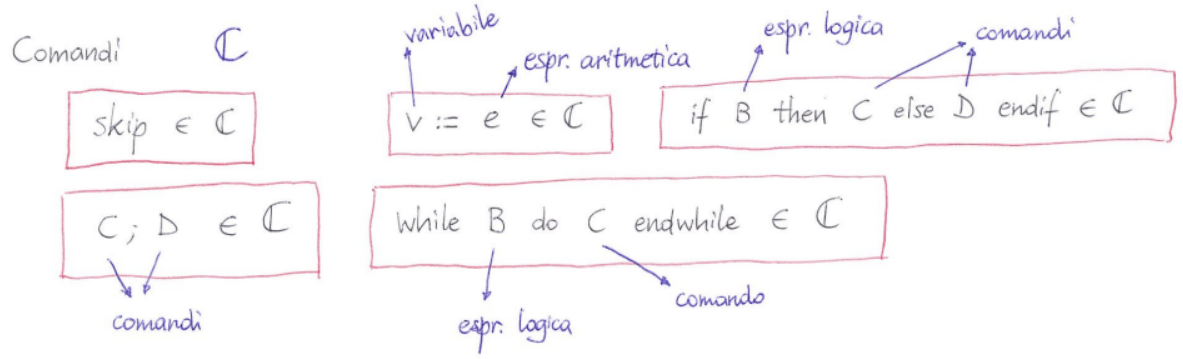
# Lezione 14 23/11/2023




*E* è l'insieme delle espressioni aritmetiche. *V* è l'insieme delle variabili. Una variabile da sola corrisponde ad un'espressione aritmetica che ha come valore il contenuto della variabile. Noi useremo solo valori interi.
Fissato un numero intero, questo da solo può formare un'espressione aritmetica. Definiamo espressioni più complicate, prendendo due espressioni e combinandole con operazioni aritmetiche. La sintassi richiede che le espressioni siano contenute da parentesi di modo da mostrare l'ordine di precedenza.




Definiamo le espressioni logiche booleane con la lettera *B*. Abbiamo le costanti logiche vero e falso, e abbiamo i connettivi logici che combinano due espressioni logiche.




Introduciamo il termine comando per indicare un programma completo o un frammento di programma.

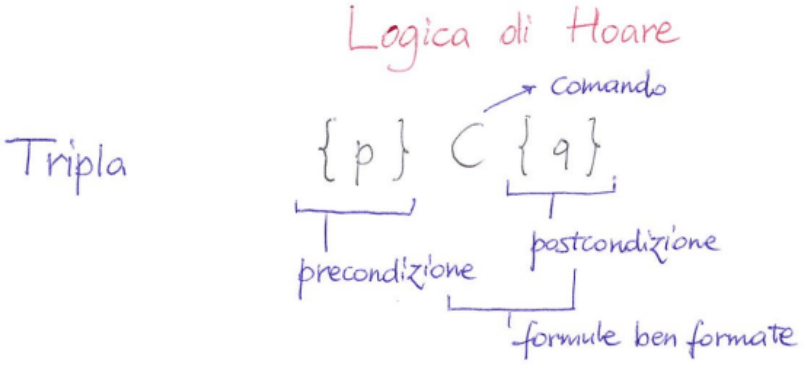
Skip è l'istruzione che non fa nulla, avanza semplicemente il contatore del programma.

Un assegnamento è formato da un'identificatore di variabile, poi := come in pascal, e sulla destra un'espressione aritmetica.
Abbiamo un'istruzione di scelta, con l'uso del *then*, *else* e *endif*. Non c'è la possibilità di omettere l'else.

Il punto e virgola è un'operatore di sequenza, di due comandi.

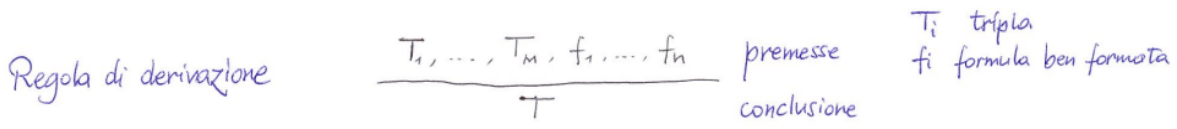
L'unica istruzione iterativa di questo linguaggio è il while, con l'uso del *do* e dell'*endwhile*.

---




Una tripla è fatta da una precondizione, un comando e una postcondizione. La tripla è vera se eseguendo C a partire da uno stato di memoria dove è vera la precondizione *p*, l'esecuzione termina in un tempo finito e nello stato finale vale la postcondizione. Quando vorremo dimostrare la correttezza di un programma, dovremo innanzitutto scrivere una tripla che descrive il nostro criterio di correttezza.

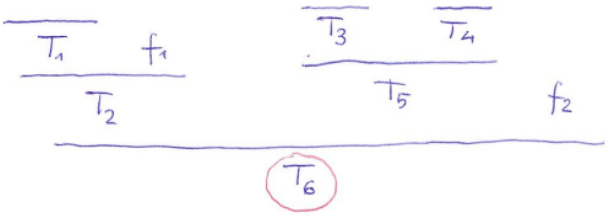
In p scriviamo quello che supponiamo sia vero all'esecuzione del programma.




Sopra alla linea orizzontale ci sono le premesse, che possono essere triple o formule ben formate della logica proposizionale. Sotto alla riga abbiamo la conclusione.

Possiamo quindi mostrare una dimostrazione in questo modo:

Dimostrazione




# Logica di Hoare – regole di derivazione

1) $\dfrac{}{\{p\}\ skip\ \{p\}}$ skip

Se il comando corrisponde alla singola istruzione skip, allora è valida qualunque tripla dove la pre condizione e la post condizione coincidano.

2) $\dfrac{\{p\}\ C\ \{p'\}\quad \{p'\}\ D\ \{q\}}{\{p\}\ C;D\ \{q\}}$ sequenza, composizione

Questa regola ha 2 premesse. La post condizione di una delle due triple coincide con la pre condizione dell'altra.

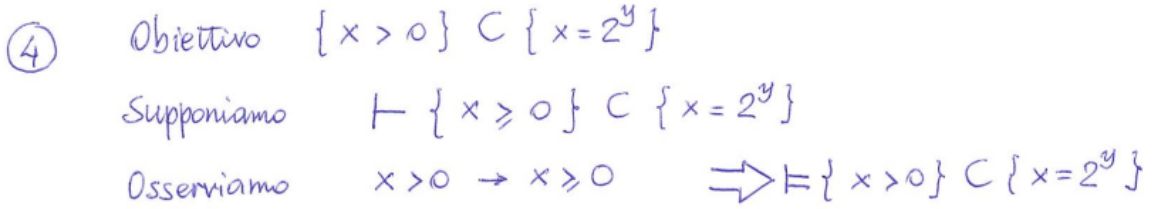
3) $\dfrac{\{p \wedge B\}\ C\ \{q\}\quad \{p \wedge \neg B\}\ D\ \{q\}}{\{p\}\ \text{if } B \text{ then } C \text{ else } D \text{ endif } \{q\}}$ scelta

Avendo una condizione di scelta (quella sotto tra pre e post), se eseguiamo le istruzioni sopra partendo dallo stato p allora l'output sarà q.

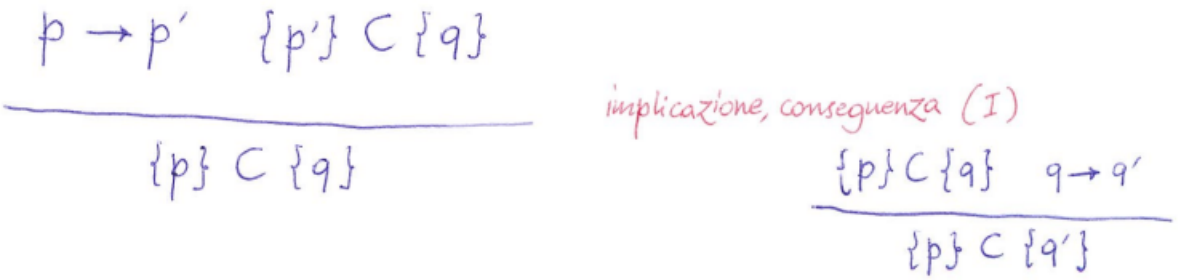
Nello stato iniziale vale p e B, e vogliamo dimostrare che se eseguiamo C allora arriviamo allo stato finale q. Quindi ci disinteressiamo di D.

Analogamente, se supponiamo che nello stato iniziale non vale B allora eseguiremo la seconda tripla raggiungendo q. Quindi se possiamo derivare le due triple separatamente, allora possiamo avere anche quella completa (sotto).

Quindi le triple che compongono le due premesse sono più semplici rispetto alla composizione finale.

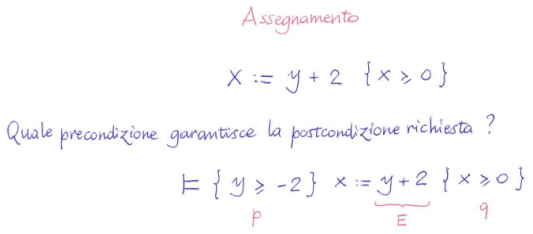



Supponiamo di avere un comando C, e voglio dimostrare che avendo x>0 ed eseguendo C mi ritroverò con x = 2^y in memoria.
Supponiamo che applicando le regole di derivazione a C, arriviamo a derivare (simbolo della T storta che significa derivabile). Se vale la relazione (la dimostro), allora anche la tripla è valida. Però io ho derivato che se x≥0 allora eseguendo C raggiungo lo stato dove x=2^y, ma io invece volevo dimostrarlo partendo da x>0. Possiamo tradurre quest'osservazione in una regola. Dico che x>0 implica x≥0 (qualunque valore vero per il primo è vero anche per il secondo). Quindi se quest'osservazione è corretta, allora dico che vale (freccia grande) la tripla con x>0.




Questa regola di implicazione ha una regola speculare che ragiona sulla post condizione (quella a destra).

---




Assegniamo a x il valore di y+2.

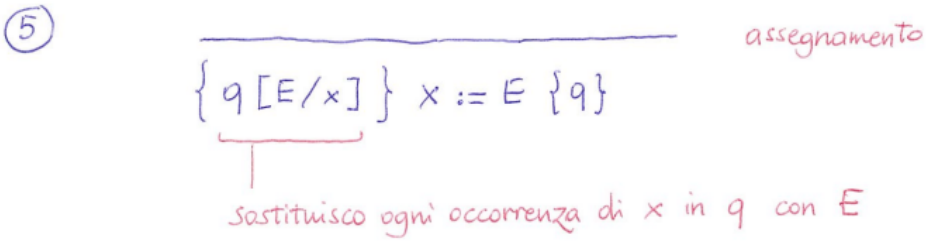
Se vogliamo che la post condizione sia x≥0, cosa dobbiamo supporre nella pre condizione?

Dobbiamo supporre che, dato che y non viene modificata, che y+2 ≥0, di conseguenza y≥-2.

In questo caso particolare possiamo anche osservare che la precondizione che abbiamo ricavato è anche una precondizione necessaria. Per ogni valore iniziale di y che non soddisfa questa formula, otterrei una tripla non valida.

Nella post condizione abbiamo sostituito a x l'espressione che viene assegnata a x. Torneremo a ragionare su questo in modo più approfondito.

Possiamo generalizzare il ragionamento di modo da ottenere una regola:




Abbiamo una variabile a sinistra (x) e un'espressione (E) a destra. Indico che come pre condizione riscriviamo q, sostituendo ogni occorrenza della variabile x con l'intera espressione E.

Nell'esempio precedente abbiamo fatto esattamente questo, e in più spostando il 2 dall'altra parte dell'espressione.

---

## While

Fino ad ora ci siamo disinteressati della terminazione, ma ora è necessario considerarla.

Distingueremo tra correttezza parziale (se abbiamo derivato una certa tripla, con l'ipotesi che il programma termini) e correttezza totale (dopo aver dimostrato la correttezza parziale, dimostreremo che l'esecuzione del programma termini in tempo finito). Oggi ci occupiamo della correttezza parziale.

Una formula che è vera al termine di una generica operazione se era vera prima si chiama variabile di ciclo. Quando parliamo di invariante, questo è quello che intendiamo. All'esame può chiedere cos'è l'invariante di ciclo. Bisogna dire che è una formula che SE è vera all'inizio dell'iterazione, è vera anche alla fine. Può capitare che temporaneamente, durante l'esecuzione, diventi falsa. Ma deve ritornare vera alla fine.

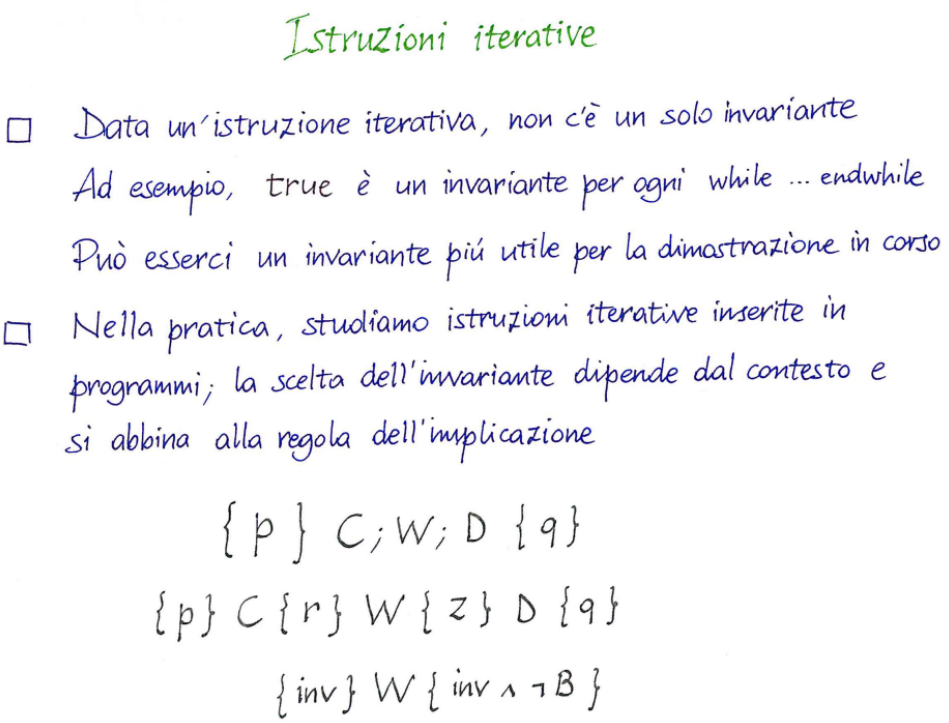

Vogliamo ricavare la tripla che ha questa forma: una precondizione, un'istruzione iterativa, e una post condizione dove troviamo di nuovo la pre condizione e la condizione B falsa (perchè termini l'esecuzione del ciclo).

Quando possiamo legittimamente derivare questa tripla? La premessa dice che se eseguiamo C partendo da uno stato dove vale l'invariante e la condizione di ciclo, allora al termine dell'esecuzione del corpo dell'esecuzione vale di nuovo l'invariante. Noi consideriamo una singola iterazione e ragioniamo su questa.

La premessa ci dice che inv è l'invariante. Siccome stiamo ragionando sulla correttezza parziale, stiamo supponendo che l'esecuzione terminerà, ovvero ad un certo punto B diventerà falsa. Questo implica che il comando C verrà ripetuto un certo numero di volte finito. Indipendentemente dal numero di ripetizioni, la premessa ci dice che alla fine la formula inv sarà ancora vera.

Nella precondizione della conclusione non compare la condizione di ciclo. Quindi questa regola funziona anche quando B è falsa nello stato iniziale. In quel caso C non verrebbe mai eseguito.

La formula sopra alla riga è la definizione di un'invariante di ciclo.




In nero c'è un esempio, che può capitare all'esame.

Un caso comune è quello il programma contiene l'istruzione iterativa incastrata tra comandi (C e D senza cicli). Applicando la regola di derivazione per la sequenza, possiamo supporre che la nostra dimostrazione si costruita concatenando diverse triple: una che parte dalla precondizione p, considera il comando C e arriva alla post condizione r, che diventa la pre condizione di W, applicando W calcoliamo la post condizione z che diventa pre condizione del comando D che ha z come pre condizione e q come post condizione.

Nella dimostrazione complessiva dovremo cercare di determinare un'invariante legato a r e z. Se possiamo dimostrare che r implica l'invariante, allora possiamo sostituire.

# Esercizi

su carta