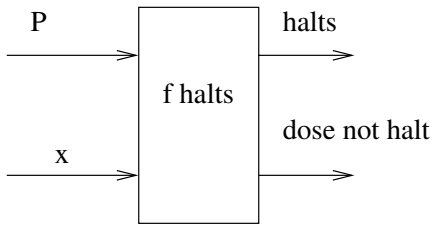
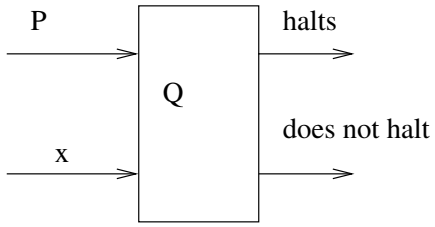
Complementi al libro *Programming Languages Concepts* per il corso di Informatica 3

1 Capitolo 2: 2.1.3 Dimostrazione dell'indecidibilità dell'halt

f_{halt},si può rappresentare in questo modo



Supponi che esista un programma Q che computa f_{halt} (termina sempre e produce "halt" o "does not halt"):

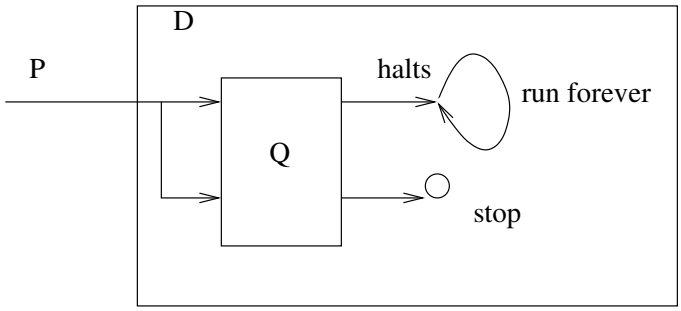


Puoi pensare ad un programma come una classe Java con un metodo static run che esegue il programma e restituisce una string. Ad esempio pensa a Q come una classe Java con un metodo run che prende due stringhe e restituisce una stringa.

```
class Q {
...
public static String run(String program, String input){
   ... // analizza program e input
   // return "halts" se program.run(input) halts,
   // altrimenti return "does not halt"
  }
}
```

Adesso costruiamo a partire da Q un programma D, che prende un programma P ed è definito così:

D(P) = if Q(P, P) = halts then run forever else halt.



Puoi pensare a D come la seguente classe Java:

```
class D {
...
public static void run(String program){
  // richiama Q.run
  if (Q.run(program,program).equals("halts")){
    // run forever
     while(true){}
    } else return;
  }
}
```

Ora metti P = D e ottieni una contraddizione.

In termini di programmi Java, prendi la stringa che da il programma D, mettila in una variabile Dstr chiama D.run(Dstr).

Se quando chiamo Q.run(D,D) ottenessi "halts", vorrebbe dire che Q rietiene che il programma D con input D si ferma, ma è sbagliato perchè poi dopo l'if ho un while(true). Se chiamando Q.run(D,D) ottenessi ("does not halt") anche questo è sbagliato, perchè poi mi fermo. Q non può esistere.

Nota

questo non vuol dire che non esista un programma che può in molti casi analizzare un programma e dire con esattezza se si ferma oppure no. NON posso scrivere un programma che analizza qualsiasi programma e che funzioni sempre .

2 Esercizio 2.2 halt0

Può esistere un programma che computi la funzione Halt0 ? la prima dimostrazione (come suggerito dal libro) si ottiene riducendo il problema dell'Halt0 al problema dell'Halt. Basta costruire un programma Q che dato un programma P e un input x, costruisce un nuovo programma P0 che è uguale a P tranne che sostituisce il valore di x ogni volta che x è usato. Si utilizza tale Q e halt0 per costruire Halt.

La seconda dimostrazione:

Se Halt0 esiste, posso costruire un programma D che chiama Halt0 con se stesso

3 Capitolo 2: Macchine di Turing

Example: Palindromes over binary alphabet

Ecco un esempio di macchina di Turing che risconosce stringhe binarie palindrome.

Input alphabet: 0,1 Tape alphabet: 0,1,b

States	State Interpretation
i	stato iniziale
p0	cerca 0 all'estremità destra
p1	cerca 1 all'estremità destra
q0	trovata l'estremità destra: controlla che sia 0
q1	trovata l'estremità destra: controlla che sia 1
r	ritorna all'estremità sinistra
t	stringa palindroma
f	stringa non palindroma

Transitions

δ(i,b) = (t,b,R) accept if tape is blank
δ(i,0) = (p0,b ,R) delete 0 at LHS; look for 0 at RHS
δ(i,1) = (p1,b,R) delete 1 at LHS; look for 1 at RHS
δ(p0,0) = (p0,0,R) move to RHS
δ(p0,1) = (p0,1,R)
δ(p1,0) = (p1,0,R)
δ(p1,1) = (p1,1,R)
δ(p0,b) = (q0,b,L) found RHS; now check whether 0 or 1
δ(p1,b) = (q1,b,L)
δ(q0,0) = (r ,b,L) check RHS=0; delete it
δ(q1,1) = (r,b ,L) check RHS=1; delete it
δ(q0,b) = (t,b,R) accept if all tape is blank
δ(q1,b) = (t,b,R)
δ(r,0) = (r,0,L) return to LHS
δ(r,1) = (r,1,L)
δ(r,b) = (i,b,R) found LHS; reurn to state i
... mancano transizioni allo stato f

3.1 Esercizi su macchina di Turing

- Unary increment - this function takes a string of 1's and adds another: 111 →1111 ?

- Unary decrement - this function takes a string of 1's and removes one: 111→11
- Binary double - this function doubles a binary string: 111→1110
- Binary increment - add one to a binary number: 101→110

soluzione stati
i: stato iniziale ad estremità sinistra - cerca estremità destra
q: estremità destra: controlla che sia 0 o 1: se è 0 trasforma in 1, ok. Se è uno trasforma in 0 e vai in r
r: scorri numero da dex a sin sommando 1.
f: finito
Transizioni:
i ,1 -> i,1,R i ,0 -> i,0,R
i ,b -> q,b,L
q,0 -> f, 1,R
q,1 -> r,0,L
r,1 -> r,0,L r,0 -> f,1,R

r e q possono essere uniti

- Unary double - double the length of a unary string 111→111111
- Binary addition

4 Capitolo 6

Type Safe C:

- CCured: http://manju.cs.berkeley.edu/ccured/, CCured is a source-to-source translator for C. It analyzes the C program to determine the smallest number of run-time checks that must be inserted in the program to prevent all memory safety violations.
- Cyclone: http://www.research.att.com/projects/cyclone/, Cyclone is a programming language based on C that is safe, meaning that it rules out programs that have buffer overflows, dangling pointers, format string attacks, and so on.

5 Capitolo 10: 10.2.1

NOTA: equals in JAVA