Il problema della terminazione

Sappiamo già, grazie ad un ragionamento basato sul concetto di cardinalità, che esistono infiniti problemi per i quali non possiamo scrivere programmi in grado di risolverli. La nostra speranza a questo punto è che questi "problemi senza soluzione" siano problemi poco interessanti, cioè senza rilevanza pratica. Mostriamo ora che le cose non stanno così, cioè che esiste almeno un problema insolubile per il quale sarebbe invece estremamente utile trovare una soluzione. Il problema in questione è quello di stabilire se l'esecuzione di un dato programma in C termina oppure entra in un qualche loop infinto che ne impedisce la terminazione. Per enunciare in maniera più chiara questo risultato è necessario introdurre una opportuna notazione.

Sia S^{01} l'insieme delle funzioni scritte in C che prendono come input una stringa di caratteri e restituiscono 0 o 1. Un esempio semplice di funzione in S^{01} è la funzione che restituisce la classe di congruenza modulo 2 della lunghezza della stringa fornita in input. Data $f \in S^{01}$ e una stringa s scriviamo $f(s)$ per indicare l'output della funzione con input s. Osserviamo però che $f(s)$ può non essere definito in quanto è possibile che per certi input una data funzione non termini (cioè entri in un loop infinito). Osserviamo che le funzioni scritte in C sono anch'esse delle stringhe di caratteri; ne segue che se $f, g \in S^{01}$ è possibile calcolare $f(g)$, $g(f)$, o anche $f(f)$. Ad esempio $f(f)$ corrisponde a far eseguire la funzione f dandogli come input la stringa che contiene il codice della funzione f.

È possibile dimostrare che in C è possibile scrivere una funzione `int interpreta(char s[], char t[])` tale che, date due stringhe f e t, `interpreta(`f`,`t`)` restituisce 0 se f non è una definizione legale di una funzione in S^{01}, e restituisce $f(t)$ altrimenti. Se la funzione f con input t non termina allora è ammesso che `interpreta(`f`,`t`)` non termini. In altre parole, se $f \in S^{01}$, la procedura `interpreta(`f`,`t`)` emula il comportamento di f con input t.

Dimostriamo ora che in C non può esistere un procedura `int termina(char s[], char t[])` tale che date due stringhe f, t `termina(`f`,`t`)` resituisce 1 se $f \in S^{01}$ e $f(t)$ termina, e restituisce 0 altrimenti (cioè se f non è una descrizione di una funzione in S^{01} o se $f(t)$ non termina). Notiamo che si richiede che `termina` termini sempre. Supponiamo per assurdo che esista una tale funzione `termina`. Allora potremmo definire la seguente funzione

```
int pippo(char f[]) {
  if(termina(f,f)==1)
    return 1-interpreta(f,f);
  return 1;
}
```

È immediato verificare che `pippo` appartiene a S^{01} e che `pippo` termina per un qualsiasi input: dato che `termina` termina sempre, l'unico punto in cui `pippo` potrebbe bloccarsi è durante l'esecuzione di `interpreta`. Ma la chiamata di `interpreta` viene eseguita solo nel caso in cui `termina` ci assicura che tale chiamata non si bloccherà.

Sia t la stringa contentente la definizione di `pippo` (cioè $t =$ `int pippo(char f[])...`). Vediamo cosa succede se proviamo a calcolare `pippo(`t`)`. Dato che `pippo` appartiene a S^{01} e termina per ogni input, l'`if` nella seconda riga risulterà vero e il valore restituito da `pippo(`t`)` sarà quindi `1-interpreta(`t`,`t`)`. Ma, per definizione, `interpreta(`t`,`t`)` è esattamente il valore `pippo(`t`)`, quindi il valore restituito da `pippo(`t`)` dovrebbe coincidere con `1-pippo(`t`)` che è assurdo.

È naturale chiedersi se il risultato appena visto vale per tutti i linguaggi di programmazione. Le cose non stanno così: immaginiamo un linguaggio di programmazione (chiamiamolo D) che sia identico al C tranne che non esistono i cicli e non è possibile usare sottoprogrammi (funzioni). Dato che ogni istruzione di un programma in D viene eseguita al più una volta, i programmi scritti nel linguaggio D terminano sempre quindi è evidente che è possibile scrivere un programma in D che dice se un certo programma in D termina.

Riguardando la dimostrazione appena fatta è immediato rendersi conto che quello che abbiamo dimostrato è che in un *qualsiasi* linguaggio di programmazione non è possibile avere contemporaneamente sia la funzione `interpreta` che la funzione `termina`. Nel linguaggio C possiamo scrivere la funzione `interpreta` e di conseguenza non possiamo avere la funzione `termina`. Viceversa, nel linguaggio D possiamo scrivere `termina` e non è invece possibile scrivere la funzione `interpreta`.