Sull'equivalenza di domande positive e negative con risorse computazionali limitate

ALBERTO PETTOROSSI *

1. INTRODUZIONE

I linguaggi naturali sono imprecisi e possono determinare ambiguità nei concetti che si desidera definire e/o comunicare attraverso di essi. Se nella costruzione di teorie scientifiche si usano i linguaggi naturali, a causa di questa imprecisione, si corre il rischio di definire teorie inconsistenti. Tali teorie non consentono l'enunciazione di risultati che abbiano validità scientifica e, in particolare, nelle scienze matematiche e nelle scienze fisiche non permettono la formulazione di teoremi o leggi attendibili. Per evitare questo grave pericolo, spesso in tali scienze, come anche in altre scienze, si fa uso di *linguaggi formalizzati*.

I linguaggi formalizzati che ciascuna scienza usa, non sono fissi, anzi evolvono, così come evolvono le metodologie proprie di ciascuna scienza. Fortunatamente, allo stato attuale dello sviluppo delle discipline scientifiche, i linguaggi formalizzati permettono una determinazione sufficientemente precisa dei concetti utilizzati in tali discipline.

In questa nota prenderemo in considerazione il linguaggio della *logica formale* che è alla base delle scienze matematiche e fisiche. Nell'ambito della logica formale desideriamo studiare il problema della negazione e della sua potenza espressiva. Questo problema riveste un interesse specifico quando si vogliano studiare concetti formali delle teorie scientifiche e si vogliano determinare, con un processo deduttivo, le proprietà che tali concetti posseggono oppure le proprietà che essi non posseggono.

Un tale approccio alla determinazione delle proprietà degli concetti formali è usato in alcune teorie della conoscenza ed è caratterizzato dal fatto che ogni concetto viene individuato da una coppia di approssimazioni: (i) *un'approssimazione per difetto*, che caratterizza le proprietà possedute dai cosiddetti *esempi positivi* del concetto, e (ii) *un'approssimazione per eccesso*, la cui negazione caratterizza le proprietà possedute dai cosiddetti *esempi negativi* del concetto[1].

Affronteremo il problema della negazione nella logica formale da un particolare punto di vista che possiamo chiamare il punto di vista *computazionale*. Non considereremo, pertanto, il punto di vista *linguistico* o quello *filosofico*. Prenderemo in considerazione la situazione in cui le risorse computazionali disponibili per rispondere a determinate domande o risolvere problemi scientifici siano soggette a limitazioni di spazio di memoria o di tempo di calcolo. Tale è la situazione in cui si debbano risolvere, attraver-

* Informatica Teorica, Università degli Studi di Roma Tor Vergata. pettorossi@info.uniroma2.it

[1] Per maggiori dettagli su questo punto cf. A. Skowron, *Rough Sets*, in G. Jacovitti - A Pettorossi - R. Consolo - V. Senni (Edd.), *Information: Science and Technology for the New Century*, Quaderni Sefir 7, Lateran University Press, Rome 2007, pp. 113-161. Cf. anche le indicazioni bibliografiche ivi contenute.

so deduzioni logiche, problemi relativi a teorie matematiche o fisiche di una certa complessità. Infatti, in tal caso accade che le deduzioni, molto spesso effettuate con l'ausilio di sistemi automatici di calcolo, possono risentire dei limiti di risorse disponibili in termini di spazio di memoria o tempo di calcolo.

2. PROBLEMI E LA LORO NEGAZIONE

Nella nostra indagine sulla negazione, studieremo la questione se la domanda (A): «L'oggetto x, appartenente all'insieme U, ha la proprietà $P(x)$?» con risposte possibili «sì» e «no», sia equivalente o meno alla domanda (B): «L'oggetto x, appartenente all'insieme U, non ha la proprietà $P(x)$?» con risposte possibili «no» e «sì», rispettivamente. Nel seguito diremo che la domanda (B) è associata al problema negato del problema a cui è associata la domanda (A). Scriveremo la domanda (B) anche così: «L'oggetto x, appartenente all'insieme U, ha la proprietà $\neg P(x)$?» ove il simbolo "¬" denota la negazione. Per esempio, se $P(x)$ indica che x è un numero naturale pari, allora $\neg P(x)$ indica che x è un numero naturale che non è pari, cioè, è un numero naturale dispari.

Si noti che affinché il problema e il problema negato abbiano un senso ben determinato, occorre che sia specificato per ogni proprietà in esame l'insieme U a cui il generico oggetto x si suppone appartenere. Tale insieme U è chiamato *insieme universo*.

Può sembrare, a prima vista, che le domande (A) e (B) siano equivalenti e, cioè, che dare la risposta alla domanda (A) richieda le stessa quantità di risorse computazionali (spazio di memoria o tempo di calcolo) richieste per dare la risposta alla domanda (B). Vedremo, invece, che ciò non è vero e che, in generale, tali domande richiedono differenti risorse di calcolo e che tale differenza dipende dalla cosiddetta difficoltà computazionale della proprietà $P(x)$.

Per fissare le idee il lettore pensi alle seguenti tre proprietà $P1(x)$, $P2(x)$ e $P3(x)$.

Proprietà $P1(x)$:

> x è una stringa di caratteri ognuno dei quali è '0' o '1' tale che, in base 2, denota un numero divisibile per 3. L'insieme universo per la proprietà $P1$ è quello di tutte le stringhe finite di '0' e '1'.

Per esempio, $P1(10010)$ vale perché 10010 denota in binario il numero 18 e 18 è divisibile per 3. $P1(1101)$ non vale (e $\neg P1(1101)$ vale) perché 1101 denota in binario il numero 13 e 13 non è divisibile per 3.

Proprietà $P2(x)$:

> (i) x è una stringa di i (≥ 0) copie di 'a' seguita da j (≥ 0) copie di 'b' seguita da k (≥ 0) copie di 'c' tale che *non* vale che $i = j = k$, oppure (ii) x è una stringa in cui possono occorrere copie di 'a', 'b' e 'c' (non necessariamente in quest'ordine e non necessariamente tutte presenti) in cui una 'a' occorre a destra di una 'b' oppure una 'a' occorre a destra di una 'c' oppure una 'b' occorre a destra di una 'c'. L'insieme universo per la proprietà $P2$ è quello di tutte le stringhe finite di 'a', 'b', e 'c'.
>
> Per esempio, $P2(aabbbbc)$ e $P2(aaccbb)$ valgono e $P2(aabbcc)$ non vale

(e $\neg P2(aabbcc)$ vale) perché in *aabbbbc* il numero delle '*a*' è diverso dal numero delle '*b*', in *aaccbb* c'è una '*b*' a destra di una '*c*' e in *aabbcc* ci sono due copie di '*a*', due copie di '*b*' e due copie di '*c*'.

Proprietà $P3(x)$:

> x è una stringa di caratteri del tipo $P \$ n$, tale che: (i) P è una stringa che denota un programma Java, chiamiamolo P_J, (ii) n è una stringa di caratteri ognuno dei quali è '0' o '1' tale che, in base 2, denota il numero intero N e (iii) il programma P_J termina quando viene eseguito con il numero N in input. (Il carattere \$, che si assume non occorrere in P, è utilizzato in $P \$ n$ per separare le due sottostringhe P e n.) L'insieme universo per la proprietà $P3$ è quello di tutte le stringhe finite fatte da stringhe di caratteri che si usano nel linguaggio di programmazione Java, seguite da '\$', seguite da stringhe finite di '0' e '1'.

Ora consideriamo il caso di una *macchina di calcolo M* che sia in grado di risolvere il problema di decidere se una proprietà $P(x)$ valga o no per una data stringa x, avendo a disposizione una certa quantità di spazio di memoria e tempo di calcolo. (Non possiamo qui definire formalmente la nozione di macchina di calcolo, ma gli esempi che presenteremo in seguito potranno dare un'idea sufficientemente precisa di tale nozione.)

Si può dare il caso che la macchina *M*, con le stesse risorse di spazio di memoria e tempo di calcolo, *non* sia in grado di risolvere il problema negato, cioè di decidere se la proprietà $\neg P(x)$ valga o no per una data stringa x. In questo senso diremo che la risoluzione di un problema e del problema negato *non* sono computazionalmente equivalenti.

In generale, si ha che la soluzione di un problema e del corrispondente problema negato possono richiedere quantità differenti di risorse di spazio e di tempo. E tale differenza può essere così rilevante da trasformare un problema che è *solubile* (per cui, cioè, esiste una macchina che lo risolve) in uno che è *insolubile* (per cui, cioè, non esiste nessuna macchina che lo risolve). A questo proposito si veda anche quanto detto da A. Montanari[2]. In particolare, per la proprietà $P3(x)$ si dà il caso che, pur esistendo una macchina di calcolo che verifica se per una data stringa x, la proprietà $P3(x)$ valga o non valga, tuttavia *non* esiste nessuna macchina di calcolo che sia in grado di verificare se la proprietà $\neg P3(x)$ valga o non valga, indipendentemente dalle quantità di spazio di memoria e tempo di calcolo che fossero messe a disposizione per tale macchina.

3. Gerarchia di problemi e la loro difficoltà computazionale

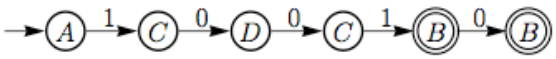
Facendo riferimento alle tre proprietà $P1(x)$, $P2(x)$ e $P3(x)$ introdotte nella sezione precedente, consideriamo il problema, chiamiamolo $A1$, di verificare se la proprietà $P1(x)$ valga o non valga per una data stringa x dell'insieme universo per $P1$. Per risolvere tale problema possiamo far uso di un automa finito, cioè, di una macchina di calcolo che, a partire da uno stato iniziale, transisce di stato prendendo in considerazione la stringa x un carattere alla volta. Tale automa, il cui numero di stati è finito (da cui il nome di automa finito), deve memorizzare tutta l'informazione di cui ha bisogno

[2] *Per un dizionario filosofico dell'informatica*, in questo volume.

in uno dei suoi stati. Esso non ha a disposizione nessun altro dispositivo per memorizzare informazione.

In Figura 1 mostriamo un automa finito che accetta le stringhe binarie che denotano i numeri divisibili per 3 senza zeri iniziali. In tale figura e nelle seguenti, lo stato iniziale è denotato da una freccia entrante, gli stati accettanti (quelli in cui la risposta è «sì») hanno due circonferenze di bordo e gli stati non accettanti (quelli in cui la risposta è «no») hanno una sola circonferenza di bordo.

L'automa di Figura 1 accetta la stringa 10010, che denota in binario il numero 18 divisibile per 3, perché la stringa 10010, considerandone un carattere alla volta da sinistra verso destra, fa transire dallo stato iniziale *A* allo stato accettante *B* attraverso le seguenti transizioni:



Osservando il comportamento dell'automa durante la computazione, si vede che il tempo di calcolo necessario, misurato come numero di transizioni di stato effettuate, è uguale alla lunghezza della stringa di input e lo spazio di memoria necessario è collegato al numero delle transizioni di stato dell'automa finito (che è uguale al numero degli stati, che nel nostro caso è 4, moltiplicato per il numero dei possibili caratteri della stringa di input, che nel nostro caso è 2, perché i caratteri della stringa di input possono essere '0' o '1').

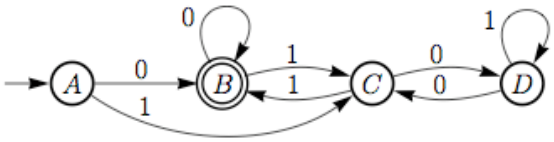


Figura 1: Un automa finito che accetta le stringhe binarie, senza zeri iniziali, che denotano i numeri *divisibili* per 3.

Nel caso della proprietà $P1(x)$ il problema negato, cioè quello di verificare se $\neg P1(x)$ vale per una data stringa x dell'insieme universo per $P1$, è anche esso risolubile da un automa finito. Tale automa è raffigurato in Figura 2 ed è derivato da quello di Figura 1 avendo scambiato gli stati di risposta «sì» (con due circonferenze di bordo) con quelli di risposta «no» (con una sola circonferenza di bordo).

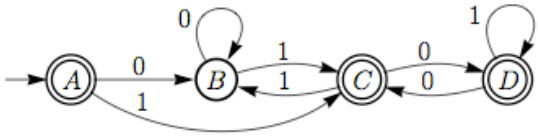


Figura 2: Un automa finito che accetta le stringhe binarie, senza zeri iniziali, che denotano i numeri *non divisibili* per 3.

Consideriamo ora il problema, chiamiamolo *A*2, di verificare se la proprietà $P2(x)$ valga o non valga per una data stringa x dell'insieme universo per $P2$. Per risolvere tale problema è necessario far uso di un automa finito che abbia a disposizione un

dispositivo ausiliario per memorizzare informazioni. Chiameremo 'pila' questo dispositivo ausiliario. Si può dimostrare che nessun automa finito senza un dispositivo ausiliario per memorizzare informazioni (o valori), può risolvere il problema $A2$.

La pila è un dispositivo in cui si può fare una sequenza di operazioni, ognuna delle quali è: (i) domandare se la pila è vuota oppure no (cioè, se non ha nessun valore oppure ha almeno un valore inserito), o (ii) inserire un valore, o (iii) togliere l'ultimo valore inserito (e non ancora tolto).

Consideriamo, per esempio, la pila in cui sono stati inseriti, nell'ordine, prima il valore 3, poi il valore 2, poi un'altra volta il valore 2 e infine il valore 4. Essa viene denotata così [4, 2, 2, 3] (si noti che, per così dire, l'immissione degli elementi è avvenuta da sinistra). Se da tale pila togliamo un valore, abbiamo l'ultimo valore inserito, cioè 4, e la pila risultante sarà [2, 2, 3]. Poi, se in quest'ultima pila inseriamo il valore 6 otteniamo la lista [6, 2, 2, 3]. Se da questa pila togliamo uno alla volta i suoi 4 valori, otteniamo la pila vuota che denotiamo con [ ].

Ora, nel caso del problema $A2$, il problema negato non può essere risolto da nessun automa finito dotato di una pila. Si dimostra, infatti, (ed è un classico risultato della Teoria della Computazione) che l'insieme delle stringhe dell'insieme universo per $P2$ e del tipo $a^i b^i c^i$ ove i è un intero maggiore di 0, non può essere riconosciuto di nessun automa finito con una pila[3].

Consideriamo, infine, il problema, chiamiamolo $A3$, di verificare se la proprietà $P3(x)$ valga o non valga per una data stringa x dell'insieme universo per $P3$.

Si dimostra che il problema $A3$ può essere risolto da un automa finito che abbia a disposizione due pile. Come nel caso del problema $A2$, il negato del problema $A3$ ha una difficoltà computazionale più alta di quella del corrispondente problema non negato. Infatti si dimostra che nessun automa finito dotato di due pile riesce a risolvere il negato del problema $A3$.

Nota 1. Invece di due pile si può far uso di due dispositivi chiamati contatori, senza diminuire la capacità computazionale della macchina di calcolo risultante. Un contatore è una particolare pila su cui si può fare una sequenza di operazioni, ognuna delle quali è: (i) domandare se il contatore è vuoto oppure no, o (ii) inserire un '1', o (iii) estrarre un '1' se il contatore non è vuoto. Il termine contatore deriva dal fatto che, essendo possibile memorizzare in esso una o più copie dell'unico carattere '1', la configurazione di un contatore memorizza in effetti un numero intero non negativo che ricorda quante operazioni di inserimento di un '1' sono state fatte in più rispetto a quelle di estrazione di un '1'.

C'è una differenza importante tra la coppia di problemi $<A2$, negato del problema $A2>$ e la coppia di problemi $<A3$, negato del problema $A3>$. Per entrambe le coppie di problemi la difficoltà computazionale del problema negato è più alta di quella del problema non negato. Tuttavia il negato del problema $A2$ è, come si dice, *solubile*, cioè esiste un automa finito con due pile che lo risolve, mentre il negato del problema $A3$ non è solubile, cioè non esiste nessun automa finito con due pile che è in grado di risolverlo.

Di più, si ha che, come abbiamo accennato sopra, la soluzione del negato del problema $A3$ non si può ottenere neppure aumentando senza limiti le risorse di spazio di

[3] Cf. J.E. Hopcroft – J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Boston 1979.

memoria e quelle di tempo di calcolo, e neppure usando altri tipi di automi (quali gli automi "non deterministici" o quelli quantistici o biologici) o usando dispositivi per la memorizzazione dell'informazione più potenti delle pile come, per esempio, code o alberi[4].

Si noti anche un'altra interessante proprietà del problema $A3$: un qualsiasi automa finito con due pile che risolve il problema $A3$, non riesce a garantirne la soluzione *in tempo finito*, nel senso che, esiste una stringa $P \$ n$ dell'insieme universo per la proprietà $P3$ (ed è una stringa per cui il programma P_J, corrispondente alla sequenza P di caratteri, non termina quando ha in ingresso N) che data in input all'automa che risolve il problema $A3$, fa sì che la computazione di tale automa non termini e che in un tempo finito non si possa sapere se la risposta al problema sarà «sì» oppure «no». E questo vale comunque si aumentino, come abbiamo detto sopra, le risorse di spazio di memoria e quelle di tempo di calcolo.

Nota 2. I risultati che abbiamo illustrato in questa sezione sono ben noti nella Teoria degli Automi e della Calcolabilità[5] e fanno parte della cultura di base nell'ambito dell'Informatica Teorica e della Teoria della Computazione. Pensiamo che tali risultati possono essere di interesse anche in altri ambiti, quali quelli più strettamente connessi alla filosofia del linguaggio e forse anche alla riflessione filosofica in generale. Pensiamo anche che, senza voler fare estrapolazioni indebite al di fuori della Teoria della Computazione, spesso occorra, anche in ambito filosofico, tener presente le risorse di calcolo degli agenti (automatici o no) che fanno deduzioni o cooperano alla definizione di teorie o alla costruzione di modelli della realtà. E queste risorse sono di fondamentale importanza per la determinazione delle capacità degli agenti di calcolo nella soluzione di problemi scientifici.

4. CONSIDERAZIONI FINALI

Abbiamo visto che le difficoltà di soluzione di un problema e del problema negato non sono, in generale, le stesse e abbiamo visto anche che tali difficoltà dipendono dalla natura del problema in esame.

In particolare, abbiamo visto che il problema $A1$ e il suo negato hanno la stessa difficoltà computazionale ed entrambi possono essere risolti da un automa finito.

Il problema $A2$ e il suo negato non hanno la stessa difficoltà computazionale e, mentre il problema $A2$ può essere risolto da un automa finito dotato di una pila, il negato del problema $A2$ non può essere risolto da un automa finito dotato di una pila, ma può essere risolto da un automa finito dotato di due o più pile.

Il problema $A3$ e il suo negato non hanno la stessa difficoltà computazionale e, mentre il problema $A3$ può essere risolto da un automa finito dotato di due pile, il negato del problema $A3$ non può essere risolto da nessun automa finito, anche se dotato di due pile. Anzi, il negato del problema $A3$ non può essere risolto da nessun automa dotato di un qualsiasi altro tipo o numero di dispositivi ausiliari per la memorizzazione

[4] Cf. A. Aho - J. E. Hopcroft - J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Boston 1983.

[5] Per le nozioni di Teoria della Computabilità non definite in questo articolo si rimanda a J.E. Hopcroft - J.D. Ullman, *Introduction to Automata Theory*, cit.

di informazioni, che abbia la stessa potenza della cosiddetta *Macchina di Turing*[6]. Non entriamo qui nel dettaglio tecnico, diciamo solo che ogni dispositivo automatico di calcolo che possa ragionevolmente essere considerato tale, ha la stessa potenza di risoluzione di problemi che ha la macchina di Turing (il lettore esperto riconoscerà in questa affermazione la cosiddetta *ipotesi di Church*[7]).

Possibilità dello sviluppo tecnologico. I risultati che brevemente abbiamo illustrato in questa nota non sono dipendenti dallo sviluppo della tecnologia: essi sono limiti intrinseci, relativi al rapporto tra il concetto di computazione (nella teoria della calcolabilità di Turing) e quello di negazione in logica (nella teoria del calcolo dei predicati del primo ordine[8]). Tali risultati sono da considerarsi di una validità stringente come quella dei teoremi dell'aritmetica, cioè come, per esempio, il teorema che asserisce che 2+3 è uguale a 5.

Potenza dei calcolatori. Vogliamo fare una nota finale sulla potenza di calcolo dei calcolatori che noi usiamo e che sono tanta parte della nostra vita e del nostro lavoro di tutti i giorni. Può sembrare strano, ma la loro potenza è quella degli automi finiti senza dispositivi ausiliari (quali pile o altro), anche se tali automi finiti hanno disponibilità molto elevate di spazio di memoria e tempo di calcolo. Questo è dovuto al fatto che tali calcolatori sono costituiti da un numero finito di componenti elementari, ognuno dei quali può trovarsi in un numero finito di stati. Nonostante questa limitazione, i calcolatori che usiamo sono per noi di grande aiuto per la soluzione di molti problemi e dimostrano un'elevata potenza di calcolo. Ciò è dovuto al fatto che in pratica non si debbono risolvere tutte le istanze di un dato problema, ma solo alcune istanze in cui la dimensione dell'input è relativamente piccola (per esempio si vuole risolvere il problema $A2$ solo nel caso in cui le stringhe di ingresso non siano più lunghe di un certo fissato numero di caratteri).

Applicazione dei risultati sulla negazione ad altri ambiti scientifici. I risultati sulla negazione che abbiamo illustrato, possono avere una certa rilevanza anche in ambiti scientifici non strettamente inerenti all'informatica o alla matematica o alla fisica. In particolare credo che questi risultati vadano tenuti presente anche nel campo della filosofia del linguaggio e quando si propongono problemi e si formulano domande le cui risposte, calcolabili o non calcolabili in modo automatico, debbano essere ricercate, avendo a disposizione risorse computazionali di spazio di memoria o di tempo di calcolo che sono in qualche modo limitate. E ciò accade molto spesso in pratica.

---

[6] J.E. Hopcroft - J.D. Ullman, *Introduction to Automata Theory*, cit., sez. 7.6 («Church's Hypothesis»), pp. 166-167.
[7] *Ibid*., p. 176.
[8] E. Mendelson, *Introduction to Mathematical Logic*, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, California, Usa, 1987.