Per un vocabolario filosofico dell'informatica

PARTE GENERALE

Angelo Montanari

Dipartimento di Matematica e Informatica

Università di Udine

Sommario. Obiettivo del presente lavoro è mettere a fuoco sia declinazioni particolari di termini che appartengono da sempre al vocabolario filosofico, quali intelligenza, conoscenza, rappresentazione e ontologia, ma anche linguaggio, tempo, infinito e (non)determinismo, sia termini caratteristici dell'informatica, quali, ad esempio, algoritmo, modello di calcolo, complessità e trattabilità, sicuramente dotati di una valenza "filosofica". Dopo aver esplorato preliminarmente il territorio dell'intelligenza artificiale e i suoi dintorni, ci concentreremo su alcuni dei vocaboli caratteristici dell'informatica. A partire da questi, mostreremo come alcuni termini tradizionali del vocabolario filosofico, quali linguaggio, determinismo, tempo e infinito, vengano (re)interpretati in modo abbastanza originale in ambito informatico.

Introduzione

In questo lavoro, metteremo a fuoco

- **declinazioni particolari** di termini da sempre appartenenti al vocabolario filosofico, quali intelligenza (artificiale), conoscenza, rappresentazione, ontologia, linguaggio, (non)determinismo, tempo, infinito
- **termini caratteristici** dell'informatica "filosoficamente" rilevanti, quali algoritmo, decidibilità, modello di calcolo, complessità, trattabilita

Informatica e filosofia

Obiettivo del corso è stimolare una **riflessione multidisciplinare** sul tema proposto per **colmare i numerosi vuoti** della riflessione sulla valenza filosofica dell'informatica

Esistono buoni libri sull'informatica di natura **divulgativa**

D. Harel, Computers Ltd. What they really can't do, 2000 (tr. it. Computer a responsabilità limitata. Dove le macchine non riescono ad arrivare, Einaudi, Torino 2002)

e buoni libri di **storia** dell'informatica

M. Davis, The Universal Computer. The Road from Leibniz to Turing, 2000 (tr. it. Il Calcolatore Universale. Da Leibniz a Turing, Adelphi, Milano 2003)

mancano libri che analizzino le **implicazioni filosofiche** dei risultati di più di mezzo secolo di ricerca in informatica

Statuto epistemologico dell'informatica

Non sorprende che, nonostante la pervasività degli strumenti informatici, i contributi "filosofici" fondamentali dell'informatica non facciano ancora parte della cultura condivisa

Non sorprende neppure che lo "**statuto epistemologico**" dell'informatica sia, in una certa misura, controverso

C'è chi riconduce l'informatica nell'alveo generale della matematica e chi ritiene che la dimensione tecnologica sia parte essenziale del proprium dell'informatica

Informatica e matematica

Il legame tra informatica e matematica è evidenziato con chiarezza da Davis nella sua storia del calcolatore universale

Nella sua ricostruzione, Davis pone Leibniz all'origine di una storia che ha per protagonisti Boole, Frege, Cantor, Hilbert, Gödel e Turing ed è, quindi, tutta interna alla storia della (logica) matematica (non è un caso che tutti i lavori fondamentali di Turing, von Neumann, Church, Post e Kleene sui modelli di calcolo degli anni '30 e '40 siano apparsi sulle maggiori riviste matematiche del tempo)

Ciò porta significativi elementi a supporto della posizione di chi considera l'**informatica** (**teorica**) parte della **matematica**

Informatica e ingegneria

Ciononostante, è del tutto evidente che i calcolatori, così come oggi li conosciamo, non esisterebbero senza alcuni risultati fondamentali della fisica e senza la disponibilità di alcuni essenziali **dispositivi tecnologici**

Il problema di stabilire quale ruolo giochino gli aspetti tecnologici nella definizione dello statuto epistemologico dell'informatica si pone, dunque, in modo del tutto naturale

È un ruolo estrinseco o intrinseco?

Informatica, matematica e ingegneria

Davis, nell'epilogo del suo libro, riconosce come la quasi totalità dei ricercatori che col loro lavoro hanno contribuito in maniera importante alla nascita del calcolatore universale, con la sola eccezione di Turing, non avessero intravisto questo esito della loro opera

Nemmeno si può dire che l'avventura intellettuale che ha portato alla nascita del calcolatore universale, che si è dispiegata su più secoli, sia anche all'origine dei progressi tecnologici che hanno reso possibile la realizzazione concreta dei computer

Pur nella diversità dei loro ruoli, va, quindi, riconosciuto ad entrambe le componenti dell'informatica, quella matematica e quella tecnologica, un ruolo determinante

Struttura della presentazione

- Intelligenza artificiale e dintorni
- Le parole dell'informatica
- Parole note, nuovi significati
- Conclusioni

Intelligenza artificiale e dintorni

L'intelligenza artificiale è uno dei temi sui quali più si è sviluppata la **riflessione "filosofica"** all'interno dell'informatica

J. McCarthy and P.J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, Volume 4: 463–502, 1969

P.J. Hayes, The naïve physics manifesto, In: *Expert Systems in the Micro-Electronic Age*, D. Michie (Ed.), Edinburgh University Press, 1978

M. Minsky, *The Society of Mind*, 1985 (tr. it. La Società della Mente, Adelphi, 1989)

Le ragioni e i rischi di un **linguaggio antropomorfo** (intelligenza, conoscenza, memoria, ragionamento, apprendimento)

Intelligenza (artificiale): il test di Turing

Che **cos'è l'intelligenza** e come possiamo stabilire se un sistema artificiale (una macchina) è intelligente?

Una macchina può essere definita intelligente se riesce a convincere una persona che il suo comportamento, dal punto di vista intellettuale, non è diverso da quello di un essere umano medio

Il **test di Turing** (meno banale di quanto possa superficialmente apparire)

- **Astrazione** da tutti gli elementi di contorno (la conformazione dei soggetti, le loro capacità fisiche)
- Interpretazione operativa dell'intelligenza che si manifesta attraverso la **comunicazione linguistica** (stretto legame tra intelligenza e capacità linguistiche)

Intelligenza algoritmica

L'intelligenza artificiale propone un'**interpretazione operativa** (**operazionale**) dell'intelligenza (e non potrebbe essere altrimenti), che definisce la portata e i limiti delle ricerche svolte in tale campo

Intelligenza come capacità di risolvere problemi (**problem solving**), di pianificare e mettere in atto insiemi di azioni finalizzati al raggiungimento di un particolare obiettivo

Obiettivo dell'intelligenza artificiale: **riprodurre i comportamenti** intelligenti della persona umana, ossia quei comportamenti la cui esecuzione si ritiene richieda intelligenza, che producono un **effetto rilevabile** sul mondo circostante (movimento, comunicazione, etc.)

Punti di contatto molto forti col concetto di algoritmo: intelligenza come capacità di elaborare algoritmi (**intelligenza algoritmica**)

Simulazione vs. emulazione

A partire dagli anni '70 si è discusso estesamente del rapporto tra l'**intelligenza umana** e l'**intelligenza artificiale**

- paradigma della **simulazione** (l'intelligenza artificiale fornisce prestazioni analoghe a quelle dell'intelligenza umana tentando di riprodurre i meccanismi alla base del suo funzionamento)
- paradigma dell'**emulazione** (prestazioni analoghe possono essere ottenute con meccanismi del tutto arbitrari)

Si è imposto il paradigma dell'emulazione (il caso della **linguistica computazionale**)?

(Rappresentazione della) conoscenza

La disponibilità di una rappresentazione interna della conoscenza di senso comune (**commonsense knowledge**) sul mondo è ritenuta condizione essenziale per il funzionamento dei sistemi artificiali di ragionamento automatico (si vuole rappresentare il modo in cui le persone vedono il mondo, non il mondo come visto dalla fisica, Hayes)

Per risultare utile una tale rappresentazione deve essere **ragionevolmente completa** in termini di copertura delle modalità di comprensione del mondo da parte delle persone

Molto spesso ci si è ristretti a porzioni del mondo limitate ed in una certa misura artificiali (il **mondo dei blocchi**)

Linguaggi di rappresentazione della conoscenza

Quale **linguaggio** per la rappresentazione della conoscenza?

- **reti semantiche**
- **logica** (proposizionale, del primo ordine, modale)
- varianti operazionali di linguaggi logici (ad esempio, il linguaggio di **programmazione logica** GOLOG, alGOl in LOGic, usato in robotica cognitiva)

Ontologia (applicata)

Il termine ontologia è entrato prepotentemente nel vocabolario dell'informatica sotto la spinta convergente dei

- filosofi che si interessano di **ontologia applicata**
- ricercatori che utilizzano metodi e strumenti dell'**intelligenza artificiale** per organizzare e gestire in modo più sistematico le conoscenze disponibili

Esempio. Sfruttare conoscenze di natura semantica nelle operazioni di ricerca su internet tradizionalmente guidate da criteri di natura sintattica (area del **semantic web**)

Ontologia applicata all'informatica - 1

Obiettivo: fornire uno **schema categoriale** sufficientemente potente e generale da inquadrare in modo sistematico le entità di un determinato dominio applicativo

Assunzione fondamentale: per operare in modo efficace nel mondo reale, un sistema artificiale deve disporre non solo di solide capacità di ragionamento, ma anche della **capacità di rappresentare** il mondo in modo adeguato

L'ontologia fornisce alcune **nozioni di base** (oggetto, evento, proprietà, cambiamento, relazioni fra classi e sottoclassi, relazioni spazio-temporali tra le parti e il tutto) a fondamento del buon comportamento del sistema

Ontologia applicata all'informatica - 2

A partire dalle nozioni di base, a seconda del dominio di interesse, verranno **selezionati e classificati** i tipi di entità rilevanti per lo specifico vocabolario ontologico (la classificazione delle entità non sempre è ovvia)

L'ontologia in informatica non è confinata nell'ambito dell'**intelligenza artificiale**

Esempio. Nell'area delle basi di dati lo sviluppo di un'ontologia condivisa è visto come uno strumento per l'**integrazione** in un unico sistema (distribuito) di **basi di dati eterogenee** che fanno riferimento a sorgenti diverse

Ontologia applicata e filosofia

Il filosofo Ferraris vede nell'ontologia applicata un possibile (nuovo) strumento per rifondare un discorso sulla realtà in cui viviamo:

"l'ontologia applicata muove da un'**assunzione realistica**: il mondo esiste indipendentemente da ciò che ne pensiamo e sappiamo, e ha delle proprietà oggettive che appartengono ad esso, e non a noi. E questo vale non solo per la natura, ma anche per buona parte del mondo umano, che come tale vale solo in quanto ha la caratteristica di esistere indipendentemente dalle credenze dei singoli"

Le parole dell'informatica

- algoritmi (teoria degli)
- decidibilità/indecidibilità
- modelli di calcolo
- complessità (teoria della)
- trattabilità

Algoritmi (teoria degli) - 1

Il concetto di algoritmo non è di per sè un contributo originale dell'informatica. Gli algoritmi sono noti da sempre non solo in matematica (l'algoritmo per il calcolo del massimo comun divisore di due numeri interi positivi fu inventato da Euclide nel quarto secolo a.C.), ma anche in numerose altre discipline (architettura)

Ciò che è nuovo è il ruolo centrale che essi assumono nell'informatica (**teoria degli algoritmi**, o algoritmica)

Un **algoritmo** è una descrizione finita, non ambigua, di una sequenza di passi che consente di risolvere un determinato problema

In generale un problema può essere risolto da vari algoritmi (ad esempio, il problema dell'ordinamento)

Algoritmi (teoria degli) - 2

Caratteristiche essenziali di un algoritmo:

- **uniformità** (un algoritmo si applica a tutte le istanze di un dato problema, potenzialmente infinite, e non ad una singola istanza, e la sua formulazione non dipende dalla singola istanza, ma è la stessa per ogni istanza)
- **effettività** (la raggiungibilità della soluzione voluta in un tempo finito o, equivalentemente, in un numero finito di passi)

Algoritmi e programmi

Un **programma** è una formulazione di un algoritmo in uno specifico linguaggio (linguaggio di programmazione) che può essere letto e interpretato da un computer

Osservazione. La formulazione di un algoritmo dipende dal soggetto (macchina) che lo deve eseguire: un algoritmo deve essere formulato utilizzando i comandi (istruzioni) che il soggetto (macchina) è in grado di comprendere ed eseguire

Tale dipendenza è all'origine della **gerarchia di linguaggi** di programmazione, che spaziano dai linguaggi di programmazione di alto livello, che utilizzano istruzioni di controllo complesse, ai linguaggi macchina, che utilizzano istruzioni direttamente eseguibili dalla macchina, e di algoritmi/programmi (**compilatori**, **interpreti**) che consentono di passare dagli uni agli altri

Decidibilità e indecidibilità

La risposta alla domanda (ingenua): "*esiste un algoritmo per ogni problema?*" è negativa

L'esistenza di problemi per i quali non esistono algoritmi si può mostrare facendo vedere che esistono funzioni non computabili (funzioni per le quali non esiste un algoritmo in grado di computarle). Esistono, infatti, più funzioni che nomi per designarle e algoritmi per computarle. L'argomento può essere formalizzato usando il metodo della diagonale di Cantor

Una domanda meno ingenua è: "*che tipo di problemi intendono risolvere gli algoritmi (ossia quali sono i problemi algoritmici) e quali fra questi problemi sono effettivamente in grado di risolvere?*"

Esistono varie categorie di problemi algoritmici (numerici, di ordinamento, di ricerca su grafi, di ottimizzazione)

I problemi di decisione

Un ruolo particolare è ricoperto dei cosiddetti **problemi decisionali**, il cui algoritmo deve decidere se una data proprietà è vera o falsa e fornisce, quindi, una risposta di tipo sì/no.

Un esempio di problema decisionale è il problema di stabilire se un dato intero positivo n sia o meno primo.

Un problema algoritmico privo di soluzione è detto problema non computabile; nel caso dei problemi decisionali, un problema privo di soluzione è detto **indecidibile**.

Problemi indecidibili - 1

Esistono importanti **problemi** decisionali **indecidibili** nell'ambito della (logica) **matematica**

Esempio 1 [Gödel]. Il **teorema di incompletezza di Gödel** dimostra che all'interno di ogni sistema formale contenente l'aritmetica esistono proposizioni che il sistema non riesce a decidere (non riesce, cioè, a dare una dimostrazione né di esse né della loro negazione)

Inoltre, fra le proposizioni che un sistema formale contenente l'aritmetica non riesce a dimostrare c'è anche quella che, in termini numerici, esprime la non-contraddittorietà del sistema (fallimento del programma hilbertiano che voleva dimostrare la non-contraddittorietà dell'intera teoria formale dei numeri sfruttando la sola aritmetica finitistica)

Problemi indecidibili - 2

Esempio 2 [Turing-Church]. L'**Entscheidungsproblem** (*problema della decisione*), posto sempre da Hilbert, era il problema di trovare un algoritmo che, data una formula della logica del primo ordine, fosse in grado di determinare se essa era o meno valida

L'indecidibilità dell'Entscheidungsproblem fu dimostrata da Turing facendo ricorso alla cosiddetta macchina di Turing e, contemporaneamente, da Church attraverso il lambda calcolo

Osservazione. I risultati "negativi" di Turing e Church hanno avuto conseguenze importanti e imprevedibili: il modello proposto da Church è alla base del paradigma della programmazione funzionale, mentre il modello di Turing si riflette nel paradigma della programmazione imperativa

Problemi indecidibili - 3

Esistono importanti **problemi** decisionali **indecidibili** in **informatica**.

Esempio 3 [Turing]. Il **problema della terminazione**, ossia il problema di stabilire, ricevuti in ingresso un qualsiasi programma e un suo possibile input, se l'esecuzione di tale programma sullo specifico input termina o meno, è indecidibile.

Esempio 4 [Rice]. Il **teorema di Rice** mostra che non esiste un algoritmo in grado di stabilire se un dato programma gode di una qualsiasi proprietà non banale (una proprietà non banale dei programmi è una proprietà che è soddisfatta da alcuni programmi, ma non da tutti, e che non dipende dallo specifico linguaggio di programmazione utilizzato, ma dagli algoritmi in sè).

Problemi indecidibili - 4

Fra i problemi indecidibili ve ne sono alcuni molto semplici. E' questo il caso di molti problemi di ricoprimento (**tiling problem**).

Un problema di ricoprimento richiede di stabilire, ricevuto in ingresso un insieme finito di tipi diversi di piastrelle colorate (quadrati di dimensioni unitarie divisi in quattro parti dalle diagonali, ognuna delle quali colorata in un certo modo), se sia possibile o meno coprire una qualunque porzione finita del piano con piastrelle dei tipi dati, rispettando alcune semplici condizioni sui colori delle piastrelle adiacenti (ad esempio, stesso colore).

E' molto comune mostrare l'indecidibilità di un problema **riducendo** ad esso una delle varianti indecidibili del problema del ricoprimento.

Livelli di indecidibilità

E' possibile definire una **relazione di equivalenza** sull'insieme dei problemi indecidibili in termini di riducibilità: due problemi indecidibili si dicono equivalenti se l'algoritmo immaginario in grado di decidere l'uno (**oracolo**) potrebbe essere utilizzato per risolvere l'altro, e viceversa.

Esempio. Il problema della terminazione e il problema del ricoprimento sono equivalenti.

Non tutti i problemi indecidibili sono equivalenti: ve ne sono alcuni **più indecidibili** di altri. Vi sono cioé situazioni in cui la riduzione precedente vale in una direzione, ma non nell'altra.

E' possibile definire una gerarchia infinita di problemi sempre più indecidibili (**livelli di indecidibilità**).

Modelli di calcolo

Un **modello di calcolo**, o di computazione, è una macchina in grado di eseguire algoritmi. Il modello di calcolo di riferimento in informatica è un modello molto semplice detto **macchina di Turing**

Quali problemi algoritmici possono essere risolti con una macchina di Turing? La **tesi di Church** afferma che le macchine di Turing sono in grado di risolvere tutti i problemi algoritmici effettivamente risolubili. E' una tesi, non un teorema (la nozione di risolubilità effettiva non è formalizzata), ma da tutti accettata

Tutti i modelli di calcolo proposti per catturare la nozione di computabilità effettiva a partire dagli anni '30, dal lambda calcolo di Church alle regole di produzione di Post, alla classe delle funzioni ricorsive di Kleene, sono fra loro equivalenti ed equivalenti alla macchina di Turing

Il calcolatore universale

Turing mostrò anche come creare una singola macchina di Turing (**macchina di Turing universale**) in grado di fare tutto ciò che può essere fatto da una qualsiasi macchina di Turing

Una tale macchina di Turing riceve in ingresso una macchina di Turing MT e un ingresso per essa e si comporta esattamente come si comporterebbe la macchina MT su tale input

E' il modello matematico del **calcolatore universale**.

MT e il suo input sono **trattati come dati** dalla macchina di Turing universale che si comporta come un interprete degli attuali linguaggi di programmazione

Nuovi modelli di calcolo

Negli ultimi anni hanno acquistato un certo rilievo nuovi modelli di calcolo, incluse la computazione quantistica (**quantum computing**) e la computazione molecolare (**DNA computing**)

La tesi di Church rimane valida.

Ad esempio, un computer quantistico è in grado di emulare ogni computazione effettuata con uno dei modelli di calcolo classici, ma vale anche il viceversa: un modello di calcolo classico è in grado di simulare qualsiasi computazione quantistica

Complessità (teoria della)

La distinzione tra problemi decidibili e indecidibili non è l'unica distinzione di interesse in informatica

Ci sono problemi computabili/decidibili la cui soluzione risulta troppo costosa dal punto di vista delle risorse di tempo di calcolo e/o di spazio di memoria necessarie ad un algoritmo per risolverli

La classificazione dei problemi sulla base dell'ammontare di risorse (**tempo** e/o **spazio**) richiesto per la loro soluzione da parte di un qualche strumento di calcolo (ad esempio, una macchina di Turing) prende il nome di (teoria della) **complessità computazionale**

Alcuni elementi fondamentali

- In generale, tempo e spazio necessari dipendono (sono funzione di) dalla dimensione dell'input: **analisi asintotica** (al crescere della dimensione dell'input)
- A parità di dimensione, la complessità può variare al variare dell'input; di norma, **analisi del caso peggiore** (non è l'unica possibile, né sempre la più ragionevole)
- Complessità degli algoritmi e **complessità dei problemi**
- Limiti superiori e inferiori alla complessità dei problemi (ad esempio, il problema della calcolo del massimo di un insieme di elementi distinti): se coincidono, **soluzioni ottimali**

Trattabilità e intrattabilità

Il tempo si misura calcolando il numero di azioni elementari effettuate durante l'esecuzione di un programma espresso in funzione della dimensione dell'input

A seconda della struttura di tale funzione, si parla di tempo logaritmico, di tempo polinomiale, di tempo esponenziale nella dimensione dell'input

Distinguiamo tra algoritmi (buoni) che richiedono un tempo polinomiale, o inferiore, e algoritmi (cattivi) che richiedono un tempo esponenziale, o superiore (trascuriamo ciò che sta nel mezzo)

Un problema per il quale esiste una soluzione algoritmica buona è detto **trattabile**; un problema che ammette solo soluzioni algoritmiche cattive è detto **intrattabile**

Complessità e modelli di calcolo

Secondo la tesi di Church la computabilità/decidibilità di un problema **non dipende** dal modello di calcolo di riferimento

E' possibile mostrare che vale un risultato analogo per quanto riguarda la complessità (**trattabilità**) di un problema: i modelli di calcolo sono correlati polinomialmente (un problema che può essere risolto in un dato modello, può essere risolto in qualsiasi altro modello e la differenza rispetto al tempo di computazione può essere espressa con una **funzione polinomiale**)

Ciò non sembra necessariamente valere per la computazione quantistica..

Problemi fortemente intrattabili - 1

Non tutti i problemi che ammettono solo soluzioni algoritmiche cattive sono uguali.

Vi sono, ad esempio, problemi che hanno una complessità doppiamente esponenziale (è questo il caso dell'aritmetica di Presburger).

In generale, un problema si dice **elementarmente decidibile** se può essere risolto da un algoritmo che, su un input di dimensione n, esegue un numero di operazioni superiormente limitato da una funzione $2^{2^{\cdot^{\cdot^{2^n}}}}$, con un numero prefissato di esponenti.

Problemi fortemente intrattabili - 2

Esistono problemi decidibili che **non** sono **elementarmente decidibili**.

E' questo il caso del problema della (in)soddisfacibilità delle formule della logica monadica al second'ordine di un successore WS1S (lo stesso vale per il suo frammento al prim'ordine).

In analogia con quanto fatto con i problemi indecidibili, è possibile organizzare gerarchicamente i problemi decidibili sulla base della loro complessità (**gerarchie di classi di complessità**)

I problemi $\mathcal{NP}$-completi

Problemi $\mathcal{NP}$. Vi è una classe di problemi decidibili che possiedono limiti inferiori polinomiali (lineari o al più quadratici), per i quali sono note soluzioni esponenziali, ma non polinomiali

$\mathcal{NP}$-completezza. I problemi in tale classe sono fra loro equivalenti (nel senso della riducibilità)

Tali problemi sono caratterizzati dal fatto che **verificare** se una possibile soluzione è veramente tale richiede un tempo polinomiale ("È più facile criticare che fare" Vardi)

Inoltre, è possibile mostrare che essi possono essere risolti in tempo polinomiale da un **algoritmo non deterministico** (in ogni punto di scelta tale algoritmo fa la scelta giusta, se una tale scelta esiste)

$\mathcal{P}$ vs. $\mathcal{NP}$

Se indichiamo con $\mathcal{P}$ la classe dei problemi la cui soluzione si può **ottenere** in tempo polinomiale, ovviamente

$$\mathcal{P} \subseteq \mathcal{NP}$$

Stabilire se

$$\mathcal{P} = \mathcal{NP}$$

è un **problema aperto**

Parole note, nuovi significati

- linguaggio
- determinismo/non determinismo
- tempo
- infinito

Linguaggio

Linguaggio naturale vs. **linguaggi formali**

Tre punti di vista alternativi sui linguaggi formali:

- linguaggi **generati** da una grammatica (linguaggio = insieme delle parole generate dalla grammatica)
- linguaggi **accettati** da una macchina/automa (linguaggio = insieme delle stringhe accettate dall'automa)
- linguaggi **definiti** da una formula (linguaggio = insieme dei modelli della formula)

La gerarchia di Chomsky

Quattro livelli di **grammatiche** (approccio generativo):

- le grammatiche di tipo 0, che includono tutte le grammatiche formali e consentono di generare tutti i linguaggi riconosciuti da macchine di Turing (l'interpretazione delle coppie di input/output come parole consente l'identificazione dei problemi risolti da una macchina di Turing coi linguaggi riconosciuti da una macchina di Turing);
- le grammatiche sensibili al contesto, che consentono di generare tutti e soli i linguaggi riconosciuti da macchine di Turing non-deterministiche limitate linearmente;
- le grammatiche libere dal contesto, che consentono di generare tutti e soli i linguaggi riconosciuti da automi a pila non-deterministici;
- le grammatiche regolari, che consentono di generare tutti e soli i linguaggi riconosciuti da automi a stati finiti.

Linguaggi formali e macchine

Il contributo più originale dell'informatica rispetto al linguaggio risiede nello stretto legame che essa stabilisce tra i linguaggi (formali) e le macchine (astratte) / automi

interpretazione operazionale dei linguaggi: un **linguaggio è** (un insieme di oggetti, parole, ma anche alberi o grafi, riconosciuto da) **una macchina**

Determinismo

Il termine determinismo è uno dei termini più controversi in ambito filosofico.

Ad esso sono strettamente collegate questioni filosofiche classiche quali, ad esempio, quelle circa il legame tra predeterminazione e libero arbitrio e tra caso e necessità.

Più debole è, invece, il legame con un principio fondamentale della meccanica quantistica, quale il principio di indeterminazione di Heisenberg (che, però, interviene nella computazione quantistica). Tale principio afferma l'impossibilità di conoscere simultaneamente posizione e quantità di moto di un dato oggetto con precisione arbitraria ed è legato essenzialmente alle problematiche della misura.

Determinismo e sistemi aperti

Il (non)determinismo occupa un posto privilegiato in informatica.

In generale, il nondeterminismo in informatica può essere visto come uno strumento per astrarre parti sconosciute o complesse dei sistemi.

Da sistemi chiusi a **sistemi aperti**.

Nondeterminismo per modellare eventi/decisioni che non sono sotto il controllo del sistema (dipendono dall'ambiente con cui il sistema interagisce)

Determinismo e non determinismo

Macchine / automi deterministici e non deterministici (equivalenza / non equivalenza)

Esempi di automi deterministici (che riconoscono linguaggi di parole finite o di parole infinite) e non deterministici (che riconoscono linguaggi di parole infinite)

Algoritmi deterministici e non deterministici (le istruzioni di scelta non deterministica)

Determinismo/non determinismo e teoria della complessità

Tempo (discreto)

Uno dei principali contributi dell'informatica alla riflessione sul tempo è stato quello di promuovere il modello del **tempo discreto** quale rappresentazione astratta più naturale della nozione di computazione

In informatica la nozione di tempo è, infatti, strettamente collegata all'evoluzione passo dopo passo (step-by-step) di un sistema (artificiale)

Esempi. I passi di esecuzione di un algoritmo, la sequenza di stati attraversati da un sistema discreto e la successione di azioni eseguita da un agente per raggiungere un determinato obiettivo

Automi temporizzati e ibridi

La ricerca sui modelli di tempo discreti non esaurisce ovviamente l'interesse della comunità informatica per la tematica del tempo

Ad esempio, per modellare le proprietà e i vincoli temporali che caratterizzano l'interazione dei sistemi (artificiali) con l'ambiente in cui si trovano ad operare, i modelli del tempo discreto risultano spesso insufficienti

Per superate tali limitazioni sono state proposte varie estensioni e soluzioni alternative che spaziano da versioni real-time dei modelli del tempo discreto (**automi temporizzati**) fino a modelli ibridi che integrano tempo discreto e tempo continuo (**automi ibridi**)

Nuove questioni in agenda

L'informatica ha modificato l'agenda dei logici che si occupano del tempo

Negli anni '70 le questioni erano quelle dell'espressività, della completezza assiomatica e della decidibilità

Oggi si sono aggiunte le questioni circa la determinazione della complessità computazionale del problema della soddisfacibilità, lo sviluppo di un buon sistema di prova (proof system) e l'esistenza di algoritmi efficienti di model checking

Infinito - 1

Dall'infinito come **problema**..

Nei programmi sin qui considerati, che a fronte di un dato input devono produrre un determinato output, l'infinito si presenta come un problema: dobbiamo garantire che il programma produca l'output atteso in un tempo finito, ossia che il programma sia terminante

all'infinito come **risorsa**..

Vi è un'altra classe di programmi/sistemi di fondamentale importanza, detti **programmi/sistemi reattivi** (ad esempio, i sistemi operativi), la cui funzione è quella di mantenere nel tempo una certa modalità di interazione con l'ambiente in cui operano e che sono inerentemente non terminanti (**computazioni infinite**)

Infinito - 2

Come specificare e verificare in modo automatico le proprietà attese dei programmi/sistemi reattivi (proprietà di sicurezza, vitalità e precedenza)? il ruolo della **logica temporale**

Sistemi a stati finiti e **sistemi a stati infiniti**

Caratterizzazione finita di oggetti infiniti (computazioni infinite, spazio degli stati infinito)

Il ruolo dell'**astrazione** (di nuovo un termine con valenza filosofica che già abbiamo incontrato, ad esempio nella gerarchia dei linguaggi di programmazione)

Alcuni testi - 1

A. Montanari, *Per un vocabolario filosofico dell'informatica*, in: Istanze Epistemologiche e Ontologiche delle Scienze Informatiche e Biologiche, a cura di G. Cicchese, A. Pettorossi, S. Crespi Reghizzi, V. Senni, Citta Nuova, 2011, pp. 82-106.

A. Montanari, *Riduzionismo e non in intelligenza artificiale*, in: La differenza umana. Riduzionismo e antiumanesimo, a cura di L. Grion, Anthropologica, Annuario di Studi Filosofici del Centro Studi Veneto Jacques Maritain, 2009, pp. 113-128.

A. Montanari, *Alcune questioni di tecnoetica dal punto di vista di un informatico*, Teoria, Volume XXVII/2007/2, Edizioni ETS, Pisa, 2007, pp. 57-72.

Alcuni testi - 1

M. Davis, *The Universal Computer. The Road from Leibniz to Turing*, 2000 (tr. it. Il Calcolatore Universale. Da Leibniz a Turing, Adelphi, 2003).

D. Harel, *Computers Ltd. What they really can't do*, 2000 (tr. it. Computer a responsabilità limitata. Dove le macchine non riescono ad arrivare, Einaudi, 2002).

M. Minsky, *The Society of Mind*, 1985 (tr. it. La Società della Mente, Adelphi, 1989).