APhEx 23, 2021 (ed. Vera Tripodi)
Ricevuto il: 25/06/2020
Accettato il: 05/12/2020
Redattori: Claudio Calosi & Pierluigi Graziani

APhEx
PORTALE ITALIANO DI FILOSOFIA ANALITICA
GIORNALE DI FILOSOFIA
NETWORK
N°23, 2021

L e t t u r e c r i t i c h e

Giuseppe Primiero, **On the Foundations of Computing**, Oxford University Press, Oxford, 2019, pp. 320.

Alberto Termine

1. Introduzione

L'informatica al giorno d'oggi è senz'altro una delle discipline più importanti con un notevole impatto sulla ricerca scientifica, sullo sviluppo tecnologico e nella vita quotidiana di ognuno. A ben vedere più che come una disciplina essa si presenta ormai come un insieme eterogeno di discipline che spaziano da ricerche più teoriche, per lo più inserite nell'ambiente accademico, ad un ampio ventaglio di settori applicativi che intersecano vari aspetti della ricerca scientifica, dell'industria e del business. Questa eterogeneità si riflette nella mancanza di una definizione univoca del termine *Informatica* e, conseguentemente, nella mancanza di una chiara caratterizzazione di quali debbano essere considerati i suoi *Fondamenti*.

Tradizionalmente, nell'ambiente accademico, il nome "Fondamenti dell'Informatica" è utilizzato in riferimento a quell'insieme di programmi di ricerca che comprende: la *Teoria dei Linguaggi Formali,* la *Teoria della Calcolabilità*, la *Teoria della Complessità Computazionale* e la *Teoria degli Algoritmi* (Brookshear 2006). In particolare, ci si riferisce ad essi come ai "fondamenti dell'informatica" in virtù del ruolo epistemologico chiave che nozioni quali *linguaggio formale*, *computabilità*, *algoritmo* e *complessità computazionale* giocano in pressoché tutti i settori dell'informatica contemporanea. Ritenere tuttavia che essi rappresentino in toto i pilastri su cui è costruito l'edificio dell'informatica è oltremodo riduttivo: significherebbe ridurre l'informatica a una disciplina meramente logico-matematica, una sorta di surrogato delle scienze esatte che si occupa di elaborare strumenti per la formalizzazione di problemi computazionalmente trattabili. Chiaramente l'informatica è molto più di questo, in essa vi sono apporti fondamentali provenienti dall'ingegneria elettronica e dalle scienze sperimentali; apporti che non possono certo essere relegati in secondo piano. Ciò nonostante, buona parte della letteratura storico-filosofica "tradizionale" inerente ai fondamenti dell'informatica si è concentrata quasi esclusivamente sui suoi aspetti logico-matematici e sulle relazioni che essa intrattiene con la logica simbolica, l'algebra e la filosofia del linguaggio (Turner & Angius 2017).

Solo negli ultimi tempi in ambito filosofico sono emersi lavori in grado di fornire una trattazione che tenga conto tanto dell'apporto delle scienze esatte quanto di quello dell'ingegneria e delle scienze sperimentali. Uno di questi è sicuramente la monografia di Matti Tedre (2014) "The Science of Computing" che individua tre diverse caratterizzazioni epistemologico-metodologiche dell'informatica: la caratterizzazione matematica, che svolge un ruolo predominante nel periodo che va dagli albori della disciplina agli anni '70; quella ingegneristica, che emerge negli anni '40 con la nascita dei primi calcolatori e fiorisce negli anni '70 grazie al diffondersi dell'ingegneria del software; infine quella scientifico-sperimentale, la quale acquisisce un ruolo fondamentale a partire dagli anni '80 per via dell'ampia diffusione dei modelli computazionali come strumento cardine nell'indagine scientifica, portando via via l'informatica ad assimilarsi alle altre scienze sperimentali (Tedre 2014).

Questa suddivisione storico-epistemologica dei fondamenti dell'informatica in *Fondamenti Matematici*, *Fondamenti Ingegneristici* e *Fondamenti Sperimentali* è alla base anche dell'opera di Primiero e ne costituisce il filo conduttore (Primiero 2019, Introduzione). A differenza del

lavoro di Tedre, la monografia di Primiero risulta più completa, offrendo al lettore anche un'ampia e dettagliata giustificazione filosofica del perché i fondamenti dell'informatica possano essere suddivisi secondo una tale struttura tripartita. Una giustificazione che è possibile grazie all'inserimento della trattazione in un preciso quadro ontologico ed epistemologico dei processi computazionali. Una delle proposte più interessanti avanzate da Primiero, infatti, è senz'altro l'idea di un'ontologia a più livelli dei processi computazionali che ricorda quella proposta da David Marr negli anni '80 per i processi cognitivi (Marr 1985). Secondo tale ontologia, dal livello più 'astratto', quello della *specifica formale* di un algoritmo, si arriva fino al livello più 'concreto', ossia quello dell'implementazione fisica di un programma tramite circuiti elettrici e porte logiche (Primiero 2019). A ciascun livello vengono attribuiti uno statuto ontologico e uno epistemologico ben determinati, così come sono ampiamente dettagliate le relazioni intercorrenti fra essi (Primiero 2019). Proprio tale "*ontologia multilivello*" costituisce lo strumento filosofico con cui rendere conto della tripartizione dei fondamenti dell'informatica. Alle scienze esatte spetterebbe infatti il compito di studiare i processi computazionali al livello della specifica formale e degli algoritmi, l'ingegneria si occuperebbe di costruire programmi e calcolatori in grado di implementare questi ultimi, e infine il metodo sperimentale costituirebbe lo strumento cardine per verificare che il processo computazionale implementato produca effettivamente, dato le opportune risorse, gli output desiderati. Proprio i processi computazionali, identificati come l'oggetto fondamentale di studio dell'informatica, non solo possono essere descritti e compresi 'a più livelli', ma lo studio di ciascuno di questi livelli può e dev'essere attuato con strumenti metodologici e secondo valori epistemici differenti. Nessuno livello, inoltre, risulta più fondamentale rispetto agli altri, né dal punto di vista epistemologico né da quello ontologico. Ne consegue che uno studio rigoroso dei processi computazionali non possa esimersi dal considerarne tanto gli aspetti matematici quanto quegli ingegneristici e sperimentali. Proprio queste tesi costituiscono il cuore della trattazione svolta da Primiero e mirano a dare una risposta innovativa ai decennali dibattiti che hanno visto contrapposti i sostenitori dell'informatica quale scienza esatta, strettamente imparentata con la logica matematica, ai sostenitori dell'informatica come disciplina ingegneristicamente orientata, votata alla produzione di artefatti al pari delle altre branche dell'ingegneria (Eden 2007).

Inoltre, rispetto a molti altri lavori recenti nel settore (Tedre 2014 e Turner 2018) un pregio dell'opera di Primiero è senz'altro la grande

attenzione ai dettagli tecnici. L'autore dedica ampio spazio alla trattazione tanto degli aspetti matematici quanto di quelli ingegneristici che ritiene entrambi fondamentali per una comprensione approfondita delle tematiche oggetto del dibattito filosofico. In questo senso, l'opera raggiunge a pieno il suo obbiettivo di costituire tanto una proposta filosofica innovativa nell'ambito del dibattito sui fondamenti dell'informatica quanto un'introduzione tecnica utile per gli studiosi di formazione filosofica che intendano addentrarsi nei dettagli tecnici (Primiero 2019a, Introduzione). Non da ultimo, è da sottolineare quanto dettagli tecnici e discussione filosofica si accompagnino ad un rigoroso inquadramento storico delle tematiche trattate: inquadramento che risulta fondamentale per capire tanto le modalità con cui l'informatica si è andata strutturando quale scienza al contempo matematica, ingegneristica e sperimentale, quanto il perché siano nati determinati dibattiti in seno ai suoi fondamenti.

2. Suddivisione Generale dell'Opera

Nei paragrafi che seguono analizzo più nel dettaglio i contenuti dell'opera. Prima di passare ad una analisi dettagliata, vorrei però fornire una panoramica della suddivisione generale del manoscritto. Come anticipato, Primiero parte dal presupposto che i fondamenti dell'informatica si possano suddividere in fondamenti matematici, ingegneristici e sperimentali: la suddivisione complessiva dell'opera rispecchia questo assunto. La Parte I è infatti dedicata ad una trattazione generale della Teoria della Computabilità, dello studio matematico degli algoritmi e del programma di ricerca noto come *Formal Verification*. La parte II, invece, partendo da una analisi della storia, dei metodi e delle nozioni fondamentali dell'ingegneria informatica affronta alcuni dei dibattiti più noti della filosofia dell'informatica contemporanea: tra questi quello sulla natura degli artefatti computazionali, quello sulla nozione di implementazione, quello sul rapporto tra algoritmi e programmi e infine quello inerente lo statuto epistemologico dell'informatica quale disciplina ingegneristica votata alla costruzione di software e calcolatori. La parte III, infine, approfondisce il tema dell'informatica vista come scienza sperimentale: essa tratta alcune nozioni fondamentali, come quelle di *Modello computazionale, Esperimento computazionale* e *Validità Sperimentale*, concludendo con un'analisi delle analogie e delle differenze sussistenti tra il metodo sperimentale così come esso viene inteso in informatica rispetto a come esso è concepito nelle altre scienze sperimentali.

Si noti che la suddivisione proposta da Primiero ha anche un preciso significato storico: gli albori dell'informatica, infatti, si intrecciano con lo sviluppo della Teoria della Computabilità all'interno del dibattito sui fondamenti della matematica sorto agli inizi del secolo scorso, cui ha fatto seguito negli anni '40 lo sviluppo dei primi calcolatori digitali, la nascita della *Software Engineering* e infine lo sviluppo di strumenti di indagine sperimentale quali i modelli computazionali e le simulazioni al calcolatore.

Le tre parti sopra elencate risultano a loro volta suddivise in capitoli, ognuno dei quali costituisce un'unità tematica a sé stante e può essere letto anche indipendentemente dal resto del volume. Ciò lascia al lettore la libertà di decidere quali e quanti capitoli leggere in base ai propri scopi e interessi: a tal proposito nella prefazione l'autore fornisce alcuni possibili itinerari di lettura basati sulla seguente suddivisione (Primiero 2019a, Prefazione): i capitoli da 1 a 4, da 7 a 8 e da 12 a 13 sono prevalentemente di taglio tecnico, fornendo una trattazione rispettivamente della Teoria della Computabilità, delle architetture dei calcolatori e delle varie tipologie di modelli computazionali e simulazioni esistenti. I capitoli 5, 9, 10 e 14 hanno carattere più storico-filosofico; infine i capitoli 6, 11 e 15 si concentrano su di un dibattito molto specifico, quello inerente il concetto di *Validità Computazionale*, concetto cui l'autore conferisce un ruolo epistemologico estremamente rilevante e che, seppur articolato nei rispettivi concetti di *Formal Computational Validity*, *Physical Computational Validity* ed *Experimental Computational Validity*, è presente in modo trasversale in tutte le varie declinazioni dei fondamenti dell'informatica.

3. Parte I

I primi capitoli sono dedicati a una discussione dettagliata del cosiddetto *Entscheidungsproblem* (Problema della Decisione) e del ruolo storico-epistemologico che esso ha giocato nella nascita dell'informatica. Il capitolo 2 offre una panoramica dei tre principali programmi fondazionali in Filosofia della Matematica dell'inizio del Novecento: il programma logicista di Frege, quello intuizionista di Brower e quello formalista di Hilbert. Inoltre, esso introduce alcune nozioni logiche e insiemistiche fondamentali e presenta il famoso *Paradosso di Russell*. Il capitolo 3 ha principalmente lo scopo di introdurre il lettore al problema della decisione tramite la presentazione di alcune nozioni tecniche fondamentali quali: induzione matematica, calcolabilità di una funzione, enumerabilità e decidibilità di un insieme. Il capitolo 4 presenta due delle più note

formulazioni rigorose della nozione di calcolabilità effettiva proposte durante gli anni '30 del secolo scorso: la teoria della ricorsività di Kleene (Kleene 1935) e la lambda-definibilità di Church (Church 1932). L'autore enuncia e dimostra inoltre alcuni teoremi fondamentali della teoria delle funzioni ricorsive, primo fra tutti il *Teorema di Forma Normale di Kleene* (Primiero 2019a, 37). Il capitolo 5 è interamente dedicato alla definizione di calcolabilità effettiva proposta da A.M.Turing e nota in letteratura come *T-Computabilità*. In esso viene presentata l'idea di *Macchina di Turing* e sottolineata la portata filosofica che ebbe l'intuizione di Turing di definire la calcolabilità effettiva per mezzo di una modellizzazione matematicamente rigorosa di un *processo di calcolo puramente meccanico*, ovvero tale per cui ogni passo successivo del calcolo dipende univocamente dal passo precedente e da un insieme fissato di regole di manipolazione sintattica (Turing 1936 e Lolli 2016).

A partire dal capitolo 6 la trattazione si sposta sul versante del dibattito filosofico. In particolare, il capitolo analizza uno dei dibattiti fondamentali nella filosofia dell'informatica contemporanea, quello inerente alla natura degli algoritmi (Turner & Angius 2017 e Turner 2018). L'autore prende anzitutto le mosse dalla constatazione che a tutt'oggi manca una definizione chiara e univoca di che cosa sia un algoritmo (Primiero 2019a, 69); prova quindi a fornirne una intuitiva nei termini di un insieme auto-contenuto di istruzioni per ottenere un determinato obbiettivo da applicare "passo-dopo-passo" (Primiero 2019a, 69). Partendo da tale definizione intuitiva, egli discute quindi varie versioni più rigorose fino ad arrivare a quella di: «algoritmo come specifica formale di una procedura ricorsiva» (Primiero 2019, 73). Si tratta di una definizione largamente accettata nel dibattito tra coloro che guardano agli algoritmi come controparti formali dei programmi (Turner 2018). L'autore analizza i limiti di questa definizione e, in particolare, l'incapacità di essa di rendere conto dei vincoli dati dall'implementazione di un dato algoritmo sotto forma di un programma per un dato calcolatore (Primiero 2019a, 74). Sposta quindi il focus su di un'altra definizione, quella di: «algoritmo come *Implementable Abstract State Machine*»
(Primiero 2019a, 75). Sfruttando la ben nota connessione fra algoritmi e macchine astratte (Bajer & Katoen 2008), con questa definizione egli riesce a catturare alcune differenze fondamentali, in termini di costo computazionale, date dal risolvere un problema computazionalmente trattabile tramite un calcolo sequenziale piuttosto che con altre procedure di calcolo (Primiero 2019a, 74). Si tratta di differenze che non possono essere

colte da una semplice specifica formale ma che risultano evidenti dal momento che si identifica un algoritmo con una data *Abstract State Machine*.

Il Capitolo 7 discute più in generale della natura matematica dell'informatica esaminando le relazioni tra quest'ultima e le scienze esatte, prima fra tutte la logica. Esso prende le mosse dalla dimostrazione dell'esistenza di una relazione di corrispondenza tra *calcoli logici* e *programmi per calcolatori*; relazione conosciuta nel dibattito come *Corrispondenza di Curry-Howard-De Bruijn* e che costituisce il fondamento del programma di ricerca della *Formal Verification* (Turner & Angius 2017). L'assunto di partenza della Formal Verification, difatti, riguarda proprio la possibilità di verificare la validità di un dato programma *P* dimostrando la validità della sequenza logica che ad esso corrisponde (Primiero 2019a). Alla Formal Verification l'autore dedica gran parte del capitolo, presentandone i vari approcci: quello *proof-teoretico*, con un accenno alla logica di Floyd-Hoare (Hoare 1969), quello *semantico*, con una breve discussione della semantica denotazionale di Scott e Strachey e, infine, la cosiddetta *Resource-based Analysis* (Reynolds 2002). Quest'ultimo, in particolare, è considerato dall'autore il più promettente ed efficacie tra gli approcci recenti (Primiero 2019a, 97). Dopo l'ampia panoramica tecnica, nella seconda parte del capitolo egli si concentra sul lungo dibattito epistemologico sorto tra sostenitori, come Dijkrsta (1974) e Hoare (1969), e detrattori, come De Millo, Lipton, Perlis (1979) e Fetzer (1988, 1991), della Formal Verification. Primiero discute in particolare due argomenti classici contro l'utilità della Formal Verification: quello presentato da De Millo, Lipton e Perlis (1979) e quello discusso da Fretzer (1988). Pur riconoscendo l'impossibilità di sostenere posizioni estreme come quelle di Hoare (1953, 135); Primiero pare schierarsi con i sostenitori della Formal Verification, sottolineando il successo che alcune tecniche sviluppate in seno ad esso, come il *model-checking*, riscuotono a tutt'oggi (Primiero 2019, 105).

Nella parte finale del capitolo, Primiero discute quindi la nozione di *Formal Computational Validity,* la prima delle tre differenti modalità con cui la nozione generale di *Validità Computazionale* è declinata nel corso dell'opera.

Si tratta della modalità inerente i livelli più astratti dell'ontologia dei processi computazionali: in quanto tale essa è perciò definita sulla base alle proprietà formali che questi ultimi esibiscono: in particolare, l'autore

definisce *valido* un algoritmo se e solo se esso soddisfa la sua specifica formale[1] (Primiero 2019, 113-114).

4. Parte II

Il capitolo 8 inaugura la seconda parte dell'opera e introduce il lettore ai fondamentali dell'ingegneria dei calcolatori digitali. Partendo dalla presentazione dei circuiti di Shannon (Shannon 1938, 471-95), Primiero fornisce un'ampia panoramica storica dell'hardware dei primi calcolatori elettronici: egli spazia dai primi sistemi di *storing* dei dati, ai *transistors*, alle tecniche di miniaturizzazione dei circuiti, per finire con una presentazione dettagliata dell'Architettura di Von Neumann (Von Neumann 1945) e di come questa abbia costituito la base per la progettazione dell'ENIAC, il primo calcolatore elettronico digitale della storia (Primiero 2019a, 130-6). Il capitolo 9 discute le tendenze storiche che hanno caratterizzato l'evoluzione della tecnologia dei calcolatori digitali dai primi esemplari degli anni Quaranta fino alla cosiddetta quarta generazione di calcolatori, emersa negli anni Novanta del secolo scorso (Primiero 2019a, 143-57). In particolare, l'autore dedica ampio spazio all'enunciazione e alla discussione del significato storico-filosofico della cosiddetta *Legge di Moore* (Moore 1975, 11-13): quest'ultima non sarebbe altro che una congettura che mette in relazione il numero delle componenti elettroniche per calcolatore con il costo unitario di ciascuna componente. Secondo Primiero, essa però cattura una reale tendenza fondamentale che caratterizza l'evoluzione dell'hardware dei calcolatori digitali: si tratta della tensione costante tra l'aumento delle componenti elettroniche impiegate per la costruzione di ciascun calcolatore digitale e i tentativi di miniaturizzazione progressiva delle medesime. Tensione che, secondo l'autore, emerge dal profondo legame tra informatica e mondo del business creatosi a partire dagli anni Cinquanta del secolo scorso (Primiero 2019a, 156-61). Tale legame porterebbe al sorgere di una serie di esigenze:

1. L'esigenza di disporre di calcolatori digitali in un numero crescente di settori dell'industria e del business;

[1] La definizione fornita da Primiero è più complessa di come è qui presentata: in particolare essa si serve delle nozioni fondamentali di *Simulazione* e *Bisimulazione*, per una panoramica più completa si veda: Primiero 2019, 113-114.

2. L'esigenza di disporre di calcolatori digitali sempre più potenti e veloci;
3. L'esigenza di abbattere i costi di produzione e diffusione dei calcolatori digitali.

Assieme esse genererebbero una dinamica evolutiva singolare: in un primo momento i costruttori aumenterebbero il numero di componenti per calcolatore, determinando un abbattimento dei costi di produzione; ben presto però l'esigenza di avere calcolatori sempre più potenti porterebbe i costruttori a dover aumentare sempre più il numero delle componenti avvicinandosi alla soglia critica oltre la quale il costo per componente torna ad aumentare in modo esponenziale. Per far fronte a questo problema i costruttori investirebbero quindi ingenti risorse nella miniaturizzazione delle componenti: la ricerca scientifica e lo sviluppo tecnologico frutto di questi investimenti permetterebbero infine di spostare, da una generazione di calcolatori all'altra, la soglia critica più avanti.

La trattazione svolta in questo capitolo rappresenta senz'altro un contributo innovativo, soprattutto in quanto sottolinea il ruolo fondamentale dell'industria e del business nel determinare il decorso evolutivo dell'informatica e, in particolare, il suo progressivo trasformarsi in disciplina ingegneristica legata al mondo dell'industria e del business. Nella letteratura di storia dell'informatica sono senz'altro presenti molte opere (Ceruzzi 2003) che analizzano il rapporto tra informatica e mondo dell'industria: il lavoro di Primiero tuttavia è tra i pochi ad analizzare tale rapporto da un punto di vista prettamente epistemologico, mettendo in risalto il ruolo svolto nel ridefinire lo statuto dell'informatica da disciplina matematica a ramo dell'ingegneria.

Il capitolo 10 è dedicato alla discussione della nozione di *physical computation* come "controparte ingegneristica" della nozione matematica di *formal computation*. Esso si apre con un'esposizione sintetica delle interpretazioni del rapporto tra *formal* e *physical computation*: concezione causale, concezione sintattica, concezione semantica, concezione linguistica e concezione minimalista (Primiero 2019a, 172 e Piccinini 2015). Primiero esamina pregi e difetti di ciascuna di esse, soffermandosi in particolare su quella minimalista, molto in voga nel dibattito contemporaneo (Piccinini 2015 e Primiero 2019a, 173). L'autore propone quindi la sua originale definizione di *physical computation* quale:

> processo algoritmico di elaborazione dell'informazione sensibile al livello di astrazione a cui un determinato sistema fisico S è descritto… (Primiero 2019a, 175).

Si tratta di una definizione alquanto innovativa nel panorama del dibattito contemporaneo, soprattutto perché inserita in un ben preciso quadro ontologico dei processi computazionali già discusso dall'autore in alcuni precedenti lavori (Fresco & Primiero 2013). Al cuore di questo quadro ontologico vi è la proposta di un'ontologia "multilivello" secondo la quale ogni livello inferiore della gerarchia risulta legato a quello immediatamente superiore da una precisa relazione di *istanziazione* (Fresco & Primiero 2013 e Primiero 2019a, 174-89). Come detto, si tratta di un approccio per molti versi innovativo rispetto a gran parte del dibattito 'classico' in filosofia dell'informatica che assume tendenzialmente un'ontologia "dualista" dei processi computazionali (Turner & Angius 2017) basata su due livelli fondamentali: quello *matematico-formale* e quello *fisico*. Un'eccezione è rappresentata dai cosiddetti indirizzi minimalisti (Piccinini 2015) che tuttavia, sposando una concezione riduzionista (Piccinini 2015), sono solite conferire priorità epistemologica allo studio degli artefatti computazionali reali piuttosto che a quello delle macchine astratte (Piccinini 2015). Al contrario l'ontologia proposta da Primiero pare in grado di superare sia il dualismo che il riduzionismo delle concezioni minimaliste facendo uso di cinque livelli ontologici distinti (Primiero 2019a,174-6):

1. livello dell'intenzione,
2. livello dell'algoritmo,
3. livello del programma,
4. livello del linguaggio-macchina,
5. livello dei circuiti elettrici.

A ciascuno di essi, inoltre, viene associata un'appropriata definizione di *processo di elaborazione dell'informazione*: al livello dell'intenzione, ad esempio, un processo di elaborazione dell'informazione verrà definito come un'attività di *problem-solving*, a livello dell'algoritmo come l'esecuzione di una determinata serie di istruzioni e, infine, al livello dei circuiti elettrici come un insieme di processi fisici che agiscono sulle proprietà elettromagnetiche usate per codificare l'informazione (Primiero 2019a, 174-5).

Tale quadro ontologico viene ripreso nel capitolo 11 per affrontare e risolvere il cosiddetto *Problema dell'Implementazione*, uno dei problemi più noti e discussi nella filosofia dell'informatica contemporanea (Turner &

Angius 2017). La relazione di implementazione è considerata infatti la relazione fondamentali che connette la descrizione matematico-formale di una computazione con il suo realizzatore fisico: si è soliti perciò infatti dire in informatica che un *programma* implementa una certa *Abstract State Machine*, oppure che un *circuito elettrico* implementa un *programma*, o, più semplicemente, che un certo *hardware* implementa un certo *software* e che un certo *software* implementa una certa *funzione*. Cosa significhi che A *implementa* B, tuttavia, è da sempre oggetto di dibattito (Rapaport 2005). Una delle principali proposte presenti nella letteratura contemporanea, che si rifà all'ontologia dualista sopramenzionata, identifica la relazione di implementazione con un tipo particolare di *relazione semantica* (Piccinini 2015): essa non sarebbe altro che una relazione di soddisfacibilità tra un dato costrutto sintattico, ovvero la descrizione logico-formale di un programma, e un modello che soddisfa tale costrutto, ovvero l'artefatto computazionale, per il tramite di un medium: il programma (Rapaport 2005). Un approccio differente a quello semantico, e molto in voga al giorno d'oggi, è invece il cosiddetto *mapping account* (Piccini 2015 e Primiero 2019a, 192): quest'ultimo definisce la relazione di implementazione come una particolare corrispondenza biunivoca tra gli stati, le proprietà e le relazioni di un algoritmo e i corrispettivi stati, proprietà e relazioni di un artefatto computazionale. Tale funzione di corrispondenza, a sua volta, assume diverse connotazioni e può essere intesa in senso riduzionista, come esprimente *identità metafisica* e/o *semantica*, oppure in senso non riduzionista, come esprimente un tipo particolare di relazione biunivoca quale *sopravvenienza* o *emergenza* (Piccinini 2015). Come per la nozione di *physical computation*, anche per quella di *implementazione* Primiero elabora una proposta originale e alternativa a quelle note in letteratura (Primiero 2019a, 194): egli definisce infatti la relazione di implementazione come:

> …relazione di istanziazione tra coppie ordinate composte da un costrutto epistemologico E e un dominio ontologico O di un dato artefatto computazionale… (Primiero 2019a, 194).

Secondo tale definizione è perciò possibile affermare, ad esempio, che un'occorrenza della coppia E/O:= < *Compito, Algoritmo* > implementa un'occorrenza della coppia E/O:= < *Problema, Intenzione* > se e solo se < *Problema, Intenzione* > è *istanziata* da < *Compito, Algoritmo* > (Primiero 2019a, 194). Si tratta di una strategia risolutiva senz'altro originale ma che, almeno nella sua formulazione più grezza, sembra andare incontro a un

problema: in che termini definire la relazione di istanziazione? Da quanto si evince nel testo suppongo che l'autore la assuma in qualche modo come primitiva, ovvero come una sorta di relazione metafisica fondamentale in grado di ordinare e tenere assieme i livelli della gerarchia ontologica precedentemente definita (Primiero 2019a, 174). Se le cose stanno così sorge però spontanea la seguente domanda: perché non assumere direttamente come primitiva la relazione di *implementazione*? perché complicare il quadro ontologico con due relazioni che paiono dopotutto equivalenti sul piano metafisico?

Il capitolo 12, che chiude la Parte II, discute infine lo statuto epistemico dell'informatica quale branca dell'ingegneria soffermandosi, in particolare, sull'emergere in primo piano della *Software Engineering* e del ruolo del *programmatore* (Primiero 2019a, 199). Nel corso del capitolo l'autore analizza le modalità con cui, nel corso della seconda metà del Novecento, la caratterizzazione dell'informatica quale disciplina ingegneristica abbia affiancato quella dell'informatica quale scienza matematica, portando infine a una concezione dell'informatica al contempo sia come *scienza degli algoritmi* e come *scienza dei calcolatori digitali* (Primiero 2019, 206). A seguito di questo cambio di paradigma elementi essenziali dell'ingegneria dei calcolatori e della sua metodologia non potevano più essere trascurati dal dibattito filosofico: valori come l'*usabilità* e l'*efficienza* sono perciò entrati di diritto nell'epistemologia e nella metodologia della disciplina. Ciò porta alla necessità di elaborare un quadro epistemologico nuovo e più complesso, in grado di rendere conto dei complessi rapporti tra aspetti matematici e ingegneristici della disciplina. Sulla scia della sua proposta ontologica, Primiero propone quindi di definire l'informatica quale:

> studio sistematico delle ontologie e dell'epistemologia delle strutture informazionali. (Primiero 2019a, 211)

dove con *struttura informazionale* egli intende l'abbinamento: *contenuto informazionale* - *serie di meccanismi* atti a trattare tale contenuto definito relativamente ad un dato livello di astrazione (Primiero 2019a, 211). La lunga analisi svolta nel corso del capitolo consente infine all'autore di approdare a una nuova nozione di validità computazionale definita come *Physical Computational Validity*. Quest'ultima estende la precedente nozione di *Formal Computational Validity* di modo da considerare non solo i vincoli formali, ma anche i vincoli ingegneristici che concorrono a definire

la validità di un processo computazionale fisicamente realizzato (Primiero 2019a, 212)[2].

5. Parte III

L'ultima parte del libro è infine incentrata sugli aspetti sperimentali dell'informatica. In essa vengono trattate tematiche di particolare rilevanza non solo per la filosofia dell'informatica, ma più in generale per tutta la filosofia della scienza contemporanea: sia in virtù dell'uso sempre più preponderante di concetti, strumenti e metodologie proprie dell'informatica all'interno delle scienze sperimentali, sia in virtù del ruolo che strumenti quali gli *esperimenti computazionali*, i *modelli computazionali* e le *simulazioni al calcolatore* rivestono nella scienza al giorno d'oggi (Magnani & Bertolotti 2017). Nel particolare: il capitolo 13 discute la metodologia dell'informatica sperimentale confrontandola sia con le concezioni classiche della metodologica scientifica, prime fra tutte il falsificazionismo popperiano (Primiero 2019a, 220), che con i modelli tradizionali della spiegazione scientifica, primo fra tutti il *modello nomologico-deduttivo* (Hempel & Oppenheim 1943) (Boniolo & Vidali 2003). Nel corso del capitolo l'autore discute le nozioni base di *Ipotesi* ed *Esperimento Computazionale*, quali rispettivamente strumento teoretico e strumento operativo fondamentale dell'informatica intesa come scienza sperimentale. Per entrambi ne delinea inoltre una tassonomia sintetica (Primiero 2019a, 222-8).

Il capitolo 14 si concentra invece sul dibattito inerente la natura dei *modelli computazionali* e delle *simulazioni al calcolatore*. Il capitolo è in gran parte dedicato ad elencare le varie tipologie di *modello computazionale* e di *simulazione* presenti in letterature e solo in conclusione introduce brevemente il dibattito sul ruolo epistemologico delle simulazioni, dibattito che riprende e discute più nel dettaglio nel capitolo seguente. Il capitolo 15 costituisce il cuore filosofico della parte III e, come detto, approfondisce il dibattito inerente il rapporto tra simulazioni computazionali ed esprimenti scientifici standard, dibattito che sta al centro della concezione dell'informatica come scienza sperimentale. Le simulazioni al calcolatore, dopotutto, non sono altro che la controparte 'computazionale' degli

[2] Anche in questo caso la definizione fornita da Primiero è molto più complessa e articolata di come qui viene riassunta. Per una trattazione più esauriente rimando il lettore interessato direttamente a Primiero 2019a, 212.

esperimenti in scienza sperimentale così come i *modelli computazionali* non sono altro che la controparte dei modelli matematici. I sostenitori dello statuto dell'informatica quale scienza sperimentale, perciò, tipicamente assumono l'esistenza di una qualche forma di analogia tra: *simulazioni* ed *esperimenti*, *modelli computazionali* e *modelli matematici* e, infine, *fenomeni target*, ovvero gli oggetti di studio della scienza sperimentale e *artefatti computazionali*, ovvero gli oggetti di studio dell'informatica. Che tipo di analogie sussistano tra queste nozioni è tuttavia oggetto di dibattito e proprio su di esso si focalizza l'autore nel corso del capitolo. La prima tesi ad essere esaminata è quella dell'*identità*, ovvero: le simulazioni sono *identiche* agli (sono istanze di) esperimenti scientifici. Se tale tesi fosse corretta, allora le simulazioni potrebbero essere usate senza alcun problema in sostituzione dei normali esperimenti non computazionali, il che implicherebbe, secondo l'autore, l'esistenza di strette relazioni di identità tra il *fenomeno target*, il modello matematico elaborato per esso, la traduzione di quest'ultimo in un modello computazionale e la sua implementazione in un artefatto computazionale. Come l'autore fa notare, si tratta di una conseguenza alquanto problematica (Primiero 2019a, 250), motivo per cui nel resto del capitolo l'autore discute tre possibili alternative ad essa. Queste, a loro volta, si rifanno ad altrettante concezioni della relazione sussistente tra i *fenomeni target* e gli artefatti computazionali sui quali vengono implementate le simulazioni di tali fenomeni. Vengono prese in considerazione tre opzioni:

1. Isomorfismo: esiste una corrispondenza uno-a-uno tra elementi/proprietà del fenomeno target ed elementi/proprietà formali nell'artefatto computazionale;
2. Analogia: ogni proprietà/elemento nell'artefatto computazionale è una versione semplificata di un elemento/proprietà del fenomeno target;
3. Similarità: L'artefatto contiene controparti di elementi/proprietà del fenomeno target legate ad esse da una relazione di *somiglianza*.

Tutte e tre queste concezioni della relazione tra fenomeni target e artefatti vengono analizzate nel dettaglio per determinare che ripercussioni abbiano sull'idea che le simulazioni al calcolatore possano essere intese come attività analoghe agli esperimenti scientifici non computazionali. Tuttavia, rileva l'autore, l'analisi delle modalità con cui è possibile definire la relazione tra artefatti e fenomeni target non sembra condurci ad una risposta definitiva: conseguentemente, egli propone di spostare il focus sulla

relazione tra *modelli matematici* e *modelli computazionali*, entità delle quali è disponibile in letteratura una caratterizzazione formale più rigorosamente definita. L'autore presenta e argomenta a favore di una tesi nota come *Simulazionismo*, la quale identifica la relazione tra modelli matematici e modelli computazionali in una relazione di *simulazione* (Primiero 2019a, 250). Egli distingue quindi tra *Simulazionismo Debole*, ovvero la tesi secondo cui il modello computazionale *simula* il modello matematico ma non viceversa, e *Simulazionismo forte*, ovvero la tesi secondo cui tra i due modelli sussisterebbe una relazione di *Bi-simulazione* (Primiero 2019a, 250). Introduce infine la nozione di *Simulazione approssimata* che utilizza per rendere conto dei casi in cui un modello computazionale è in grado di simulare solo parzialmente e con una certa dose di errore, il modello matematico corrispondente. L'introduzione della nozione di simulazione approssimata consente inoltre all'autore di rendere conto della dinamica di correzione costante a cui sono sottoposti i modelli computazionali nella pratica sperimentale (Primiero 2019a, 250).

Nonostante la tesi simulazionista sostenuta sia sicuramente interessante e in linea con il dibattito filosofico sul tema, essa, come tutte le posizioni filosofiche, presta il fianco ad alcune critiche cui l'autore non sembra dedicare eccessivo approfondimento ma sulle quali mi vorrei brevemente soffermare. Anzitutto, egli sembra presupporre che sia possibile, quantomeno in linea di principio, stabilire l'esistenza di un *isomorfismo* tra il fenomeno target, o parti di esso, e il modello formale che lo descrive. Ciò, a mio avviso, è del tutto implausibile: per stabilire il sussistere di una relazione di isomorfismo, e in generale di una qualsiasi relazione di tipo algebrico, tra un modello e il fenomeno che esso intende rappresentare occorrerebbe disporre già di una descrizione matematica di quest'ultimo, ma tale descrizione è nient'altro che un modello a sua volta. Come stabilire quindi l'esistenza di un isomorfismo tra quest'ulteriore modello e il fenomeno target? Occorrerebbe un terzo modello, e così via all'infinito. Non si tratta dell'unico problema: la definizione di *Simulazionismo Debole* fornita da Primiero implica la possibilità di caratterizzare ogni generico modello matematico come un insieme di stati e transizioni tra essi ma non è affatto scontato che ciò sia possibile, anzi, sono numerosi gli esempi di modelli che non sono traducibili in sistemi a stati e transizioni (Magnani & Bertolotti 2017).

Al di là di queste problematiche puntuali, l'intera parte III dell'opera, pur rigorosa nei dettagli formali e nelle definizioni, risulta un po' troppo distante dalla pratica scientifica reale, soprattutto per lo scarso uso di esempi

e la completa mancanza di *case-studies*. Alcuni esempi concreti di *modelli computazionali* in uso nell'informatica contemporanea e/o la discussione di uno o più *case-studies* avrebbe senz'altro reso la trattazione più attraente per un pubblico di scienziati sperimentali e/o filosofi della scienza.

L'opera si chiude infine con il capitolo 16 che completa la lunga trattazione della nozione di validità computazionale trasversale a tutte le tre parti principali dell'opera. Nel corso del capitolo, facendo tesoro di quanto esposto nel precedente, l'autore introduce e discute una serie di nozioni quali: *Weak* e *Strong Verification*, *Weak* e *Strong Validation* etc. (Primiero 2019a, 257) giungendo infine alla formulazione del criterio di *Valid Experimental Computation* così definito:

> Un artefatto *t* implementa un processo computazionale sperimentale P valido se e solo se:
> *1.* Il modello matematico-formale MM di P raggiunge i suoi scopi ed è robusto,
> *2.* Esiste un modello computazionale CM che simula fortemente (strongly simulates) MM,
> *3.* L'implementazione di CM in t è utile (usable), funzionale (reliable) e garantisce una computazione fisica valida. >> (Primiero 2019a, 269).

Il criterio è formulato in modo da sintetizzare i requisiti già presenti nei precedenti e pertanto costituire il punto di arrivo della ricerca di una definizione rigorosa per il concetto intuitivo di validità computazionale, ricerca che costituisce uno dei fili portanti dell'intera opera.

6. Considerazioni Generali

In conclusione, vorrei ribadire e approfondire alcune considerazioni generali sull'intera opera. Come detto, il volume di Primiero ha senz'altro il pregio di offrire una trattazione sistematica dei fondamenti dell'informatica che sia al contempo filosofica, storica e tecnica: ciò lo rende per molti versi un unicum nella letteratura filosofica contemporanea. Infatti, nonostante altre opere, come la sopracitata monografia di Tedre (2014) o l'articolo "*Three Paradigms of Computer Science*" di Amnon H. Eden (2007), propongano una tripartizione dei fondamenti dell'informatica analoga a quella delineata da Primiero, essi non forniscono una trattazione in grado di coprire al contempo tanto gli aspetti storico-filosofici quanto quelli tecnici al livello di approfondimento offerto da *On The Foundations of Computing*. Il lavoro di Tedre, per esempio, risulta molto dettagliato dal punto di vista storico-filosofico ma non discute quasi per nulla gli aspetti tecnici e, per tale

motivo, difficilmente potrebbe adempiere allo scopo di rappresentare un'utile introduzione tecnica dell'informatica ad uso di un pubblico filosoficamente-orientato, scopo che invece l'opera di Primiero si prefigge e raggiunge a pieno. Un altro pregio del volume di Primiero è senz'altro la presenza di esercizi proposti al lettore alla fine di ogni capitolo: essendo l'opera pensata anche come manuale introduttivo, la presenza di un eserciziario rappresenta per il lettore un utilissimo strumento di autovalutazione: anche da questo punto di vista il volume risulta innovativo rispetto a gran parte della manualistica canonica in filosofia dell'informatica. Ultimo pregio che mi preme menzionare è la presenza di un *abstract* all'inizio di ogni capitolo che ne riassume brevemente i contenuti. Come ho già precedentemente sottolineato, una dell'intenzioni dell'autore è rendere i vari capitoli dell'opera delle unità tematiche auto-sussistenti di modo che il lettore possa scegliere cosa e quanto leggere: la presenza di un abstract all'inizio di ogni capitolo non solo agevola il lettore premettendogli di orientarsi rispetto ai contenuti, ma lo aiuta anche a farsi un'idea di quali e quante siano le problematiche discusse all'interno dei fondamenti dell'informatica senza doversi necessariamente addentrare in una lettura approfondita di ciascun capitolo.

Al di là dei molti pregi, sono però presenti anche alcune criticità che è mio dovere sottolineare. La prima riguarda la trattazione della *Tesi di Church-Turing* svolta nella parte I: il contesto e i dettagli formali sono esposti con grande chiarezza e rigore ma viene tralasciato quasi completamente il lungo e complesso dibattito filosofico sorto attorno allo statuto epistemologico e semantico di tale tesi (Copeland 2017). Si tratta di un dibattito molto importante che ha coinvolto alcuni tra i più noti logici e filosofi del linguaggio del Novecento, da Stephen Kleene (1952) a Saul Kripke (2013) passando per lo stesso Turing (1935), e che meriterebbe, a mio avviso, una maggior attenzione. A questo proposito occorre puntualizzare che un veloce cenno al dibattito sullo statuto della Tesi di Church-Turing è presente nel capitolo 6, ma si tratta di un'esposizione molto breve e da una prospettiva alquanto sconnessa dal dibattito canonico.

Più nel complesso, un difetto dell'opera è il suo eccessivo sbilanciamento a favore della trattazione formale unita alla povertà di esempi. La cura con cui Primiero definisce in modo rigoroso le varie nozioni che via via si incontrano è apprezzabile e in linea con la formazione logica dell'autore: tuttavia essa può rendere a tratti difficoltosa la lettura, soprattutto per il lettore con scarsa familiarità con la materia. La povertà di esempi, inoltre, rischia di affaticare ulteriormente il lettore. Ciò è vero in

particolar modo per la parte III, dove la trattazione dei concetti di *ipotesi, modello* ed *esperimento computazionale* fa un uso pressoché nullo di esempi concreti: questo non aiuta il lettore a farsi un'idea chiara di come queste nozioni vengano adoperate nella pratica sperimentale. In generale, volendo l'opera di Primiero rappresentare anche un contributo di filosofia della scienza, un'attenzione maggiore a come gli stessi informatici adoperano, nel concreto, i concetti e le nozioni discusse sarebbe senz'altro utile.

In conclusione, si può affermare che l'opera di Primiero soddisfi senza ombra di dubbio gli obbiettivi che l'autore si propone di voler raggiungere. Essa costituisce un'utilissima introduzione al tema dei fondamenti dell'informatica e una trattazione ricca di proposte innovative per i "frequentatori" del dibattito filosofico sui fondamenti dell'informatica. Tuttavia, in vista di una futura seconda edizione, un maggior bilanciamento tra trattazione formale ed esempi concreti, tanto a scopo illustrativo quanto a supporto delle tesi in essa contenute, la renderebbe senz'altro un'opera migliore di quanto già non sia.

Bibliografia

Bajer C., Katoen J.P. 2008, *Principles of Model Checking*, Cambridge (MA), The MIT Press.

Boniolo G., Vidali P. 2003, *Introduzione alla Filosofia della Scienza*, Milano, Bruno Mondadori Editore.

Brookshear G., 2006, *Computer Science: An Overview* (9th Edition), London, Pearson.

Ceruzzi P. 2003, *A History of Modern Computing*, Cambridge (MA), The MIT Press.

Copeland J.B., 2017, «The Church-Turing Thesis», *Standford Encyclopedia of Philosophy*. On-line: https://plato.stanford.edu/entries/church-turing/.

Church A., 1932, «A Set of Postulates for the Foundation of Logic», *Annals of Mathematics* (Second Series), 33, pp. 346-366.

DeMillo R. A., Lipton R. J., Perlis A. J., 1979, «Social processes and proofs of theorems and programs», *Communications of the ACM*, 22(5), pp. 271-280.

Dijkstra E. W., 1974, «Programming as a discipline of mathematical nature», *American Mathematical Monthly*, 81(6), pp. 608-612.

Dijkstra E.W., 1975, «Correctness concerns and, among other things, why they are resented», in *Proceedings of the 1975 International Conference on Reliable Software,* ACM SIGPLAN Notices, Los Angeles, California U.S.A, pp. 546-50.

Eden A.H., 2007, «Three Paradigms of Computer Science», *Mind and Machines*, 17, 2, pp. 135-67. On-line: https://doi.org/10.1007/s11023-007-9060-8.

Fresco N., Primiero G., 2013, «Miscomputation», *Philosophy and Technology*, 26, 3, pp. 253-72.

Fetzer J. H., 1988, «Program verification: The very idea». *Communications of the ACM*, 31(9), pp. 1048-1063.

Fetzer J. H., 1991, «Philosophical aspects of program verification», *Minds & Machines*, 1(2), pp. 197-216.

Haigh T., 2010, «Dijkstra's crisis: The end of Algol and beginning of so ware engineering», draft for discussion in SOFT-EU Project Meeting, September.On-line: http://www.tomandmaria.com/Tom/Writing/DijkstrasCrisis_LeidenDRAFT.pdf.

Hempel C.G., Oppenheim P., 1948, «Studies in the Logic of Explanation», *Philosophy of Science*, 15, pp. 135-75.

Hoare C.A.R., 1969, «An axiomatic basis for computer programming». *Communications of the ACM*, 12, 10, pp. 576-80.

Kleene S. 1935, «A Theory of Positive Integers in Formal Logic», *American Journal of Mathematics*, 57, pp. 153-244.

Kleene S. 1952, *Introduction to Metamathematics*, Amsterdam, North-Holland Editor.

Kripke S.A., 2013, «The Church-Turing 'Thesis' as a Special Corollary of Gödel's Completeness Theorem», in Copeland, Posy, and Shagrir 2013, pp. 77-104.

Lolli G., 2016, *Tavoli, Sedie e Boccali di Birra*, Milano, Raffaello Cortina Editore.

Magnani L., Bertolotti T., 2017, *Handbook of Model-Based Science*, New-York, Springer.

Marr D., 1982, *Vision*, Cambridge (MA), The MIT-Press.

Moore G.E., 1975, «Progress in digital integrated electronics», *IEDM Tech. Digest*, pp. 11-13.

Odifreddi P., 1992, *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. North-Holland, Amsterdam-London-New York-Tokyo.

Ousterhout J., 2019, *A Philosophy of Software Design*, Palo Alto (CA), Yaknyam Press.

Piccinini G., 2015, *Physical Computation: A Mechanistic Account*, Oxford, Oxford University Press.

Primiero G., 2019a, *On The Foundations of Computing*, Oxford, Oxford University Press.

Primiero G., 2019b, «Design, Malfunction, Validity: Three More Tasks for the Philosophy of Computing», *Philosophy & Technology*. On-line: https://doi.org/10.1007/s13347-019-00367-6.

Rapaport W.J., 2005, «Implementation is semantic interpretation: Further thoughts». *Journal of Experimental and Theoretical Artificial Intelligence*, 17, 4, pp. 385-417.

Reynolds J.C., 2002, «Separation Logic: a logic for shared mutable data structures», in *Proceedings of the 17h annual IEEE Symposium on Logic in Computer Science*, 55-74.

Shannon C.E., 1937, «A symbolic analysis of relay and switching circuits». *Transactions American Institute of Electrical Engineers*, 57:471-95.

Tedre M., 2014, *The Science of Computing*, London, CRC Press.

Turner R., Angius N., 2017, «The Philosophy of Computer Science», *Standford Encyclopedia of Philosophy*. On-line: https://plato.stanford.edu/entries/computer-science/.

Turner R., 2018, *Computational Artifact: Towards a Philosophy of Computer Science*, Berlin, Springer.

Turing A.M., 1936, «On Computable Numbers, with an Application to the Eintscheidungsproblem», *Proceedings of the London Mathematical Society* (Series 2), 42 (1936–37), pp. 230–265.

Von Neumann J., 1945, «First draft of a report on the EDVAC», *Contract*, 670: pp. 1-34