Come ti sconfiggo l'entropia informazionale: automi cellulari e computazione reversibile

Francesco Berto, Gabriele Rossi e Jacopo Tagliabue[1]

Abstract

Lo sviluppo delle nanotecnologie ha portato la *computer science* a dedicare sempre più attenzione alla realizzabilità fisica dei propri modelli computazionali. L'idea stessa di macchina di Turing intende esprimere, in forma idealizzata, i requisiti a cui qualsiasi processo di calcolo è soggetto. Non tutti i vincoli *fisici* della computazione sono però catturati dai modelli tradizionali. In particolare, non lo è il vincolo della *reversibilità*. Le leggi dinamiche della fisica sono reversibili a livello microscopico: distinti stati iniziali nell'evoluzione di un sistema microfisico portano sempre a distinti stati finali. Di conseguenza, chiunque voglia simulare procedure di computazione radicate nella natura delle entità di base che compongono il nostro universo fisico, dovrebbe ricorrere a regole dinamiche reversibili. Inoltre, la computazione irreversibile è dispendiosa: cancellare informazione genera calore e richiede energia. Quando un bit di informazione viene cancellato, in base alle leggi della termodinamica il bit viene semplicemente rilasciato nell'ambiente circostante aumentandone l'entropia.

Presentiamo qui un modello di "mondo fisico digitale", basato su una semplice intuizione di fondo: la natura ultima dell'universo è *discreta* e *finita*, ed esiste un forte isomorfismo fra mondo fisico e informazionale. Il modello è un automa cellulare dalla dinamica perfettamente reversibile, in grado di conservare la totalità dell'informazione presente fin dall'inizio dell'universo, senza generare entropia. Il nostro universo fisico-informazionale realizza gli operatori logici e gli altri mattoni costitutivi della computazione universale. Di conseguenza, nel nostro mondo digitale possono svilupparsi ed esistere *computer*: macchine di Turing universali, capaci di calcolare (se si accetta la Tesi di Turing) tutto ciò che è calcolabile. Il modello fornisce quindi indicazioni importanti per la realizzazione di sistemi computazionali ad alte prestazioni: sistemi che utilizzano le risorse offerte dal mondo fisico nel modo più efficiente possibile – ad esempio, mostrando come sia possibile costruire circuiti logici a dissipazione di energia interna virtualmente nulla!

> Se il più efficiente supercomputer lavora tutto il giorno per computare un problema di simulazione meteorologica, qual è la minima quantità di energia che deve essere dissipata secondo le leggi della fisica? La risposta è in realtà molto semplice da calcolare, perché non ha niente a che fare con la quantità di computazioni. La risposta è sempre uguale a zero.
>
> Edward Fredkin, *A Physicist's Model of Computation*

1. Automi cellulari: una "fisica del pensiero"

Gli automi cellulari (AC) sono rappresentazioni matematiche di *sistemi complessi* – ossia, di sistemi dinamici costituiti da una pluralità di parti che interagiscono in modo non lineare.

Vi sono diverse ragioni per studiare gli AC: anzitutto, consentono di rappresentare e simulare lo sviluppo di una grande varietà di strutture dinamiche – dalle reti neurali, agli

[1] Laboratori di Intelligenza Artificiale applicata iLabs srl, via Pattari, 6, 20122 Milano, Web site: www.ilabs.it E-mail: info@ilabs.it Tel. +39 02 45 47 63 05 Fax +39 02 72 00 44 95.

algoritmi genetici che traducono in forma trattabile computazionalmente l'evoluzione della vita, a certe reazioni chimiche, fino a sistemi socio-economici e militari complessi. In secondo luogo, sono utili modelli concettuali per studiare in forma estremamente "pura" lo sviluppo di *pattern*: una caratteristica saliente degli AC consiste nel fatto che fenomeni *complessi* emergono, ad alto livello, come risultato della cooperazione fra individui semplici che, a basso livello, agiscono e interagiscono sulla base di regole elementari. Probabilmente *il* sistema complesso per eccellenza è costituito dalla più interessante porzione di mondo fisico a noi nota: il cervello umano, in cui $\sim 10^{10}$ neuroni semplici variamente interconnessi, ed operanti secondo semplici regole biochimiche, producono quell'insieme di capacità cognitive che raduniamo sotto il termine "intelligenza".

Ma vi è un terzo motivo per occuparsi di AC: vi possiamo fondare un'*ontologia fisica* computazionale – un modello computazionalmente trattabile della stessa realtà fisica che ci circonda. Uno dei motivi per cui gli odierni computer faticano a realizzare ciò che noi chiamiamo "intelligenza" è che non catturano se non in forma molto rozza certi aspetti – in particolare, quelli "non logici" – tipici dell'intelligenza umana. Ma i normali computer a volte sono inefficaci anche nel loro campo specifico, ossia nel far computazioni: pensiamo, ad esempio, alla difficolà di fattorizzare numeri interi molto grandi (un'attività essenziale per la sicurezza in Internet, la crittografia, etc.). Al contrario, nanotecnologie e ricerca nelle teorie non-standard della computabilità ci mostrano che possiamo trovare computazione dove meno ce l'aspetteremmo, nei meandri più piccoli della realtà: al livello biologico, nel modo in cui DNA, RNA e proteine stipano e trasmettono le informazioni genetiche; a livello chimico-fisico, nella distribuzione e nel movimento dei gas perfetti; e persino nella realtà subatomica descritta dalla teoria quantistica. Il computer quantistico è ancora un'idea, ma un'idea non irrealizzabile. Ricercatori di Harvard e Princeton studiano la possibilità di costruire computer biologici costituiti effettivamente da DNA, RNA e catene proteiche, e capaci di computare operatori booleani nelle cellule organiche. Altre forme di computazione sfruttano le proprietà chimico-fisiche di materiali specifici, come il carbonio.

Queste forme di computazione hanno alcune caratteristiche in comune: il fatto di procedere in parallelo; la capacità di realizzare in un unico supporto sia la *memorizzazione* che il *trasferimento* dei segnali; e la possibilità di ridurre computazioni che dal punto di vista dei normali computer sarebbero esponenziali a problemi di difficoltà polinomiale. Quel che ci interessa ora è il quadro fisico-informazionale che questa situazione suggerisce: forse *l'universo* di cui facciamo parte può davvero essere visto come un computer!

L'intuizione alla base della ricerca sviluppata nei nostri laboratori, edi cui questo articolo presenta un esempio, è che vi sia un isomorfismo così forte fra realtà fisica e realtà informazionale, che l'universo di cui facciamo parte possa essere plausibilmente visto come un computer, la cui attività fondamentale ad ogni istante t consiste nel computare il proprio stato all'istante successivo $t+1$ (e nel farlo, come vedremo, *localmente* e digitalmente). In questa prospettiva ontologica, "l'informazione gioca il ruolo di una variabile primitiva che informa il corso dello sviluppo dell'universo" (Ilachinski [2001]: 605), il che vuol dire che la nozione di *informazione* va inclusa come un primitivo fisico, accanto a nozioni quali *spazio* e *tempo*![2]

Il marchio di fabbrica degli automi cellulari è la loro natura *discreta*, oltre che deterministica. Nel modello da noi sviluppato abbiamo incorporato una ulteriore assunzione di *finitezza*: nessuna parte della teoria dovrà far uso – per dirlo in termini filosofici – della

[2] Per qualche riferimento illustre, cf. il *computing universe* di Konrad Zuse [1982], e il monumentale Wolfram [2002].

nozione di infinito in atto. La nostra strategia è conforme a quello che Edward Fredkin ha chiamato il Principio di Natura Finita:

> Natura Finita è l'ipotesi che a livello ultimo ogni quantità fisica, con l'inclusione dello spazio e del tempo, si rivelerà essere discreta e finita; che la quantità di informazione in qualsiasi volume, per quanto piccolo, dello spaziotempo, sarà finita ed uguale a una di un piccolo numero di possibilità. […] Natura Finita implica che il sostrato fondamentale della realtà fisica operi in maniera simile al funzionamento di certi computer specializzati, chiamati automi cellulari. […] Natura Finita non si limita a suggerire che gli aspetti informazionali della fisica sono importanti; sottolinea che gli aspetti informazionali sono *tutto* ciò che conta in fisica al livello più microscopico. (Fredkin [1993])

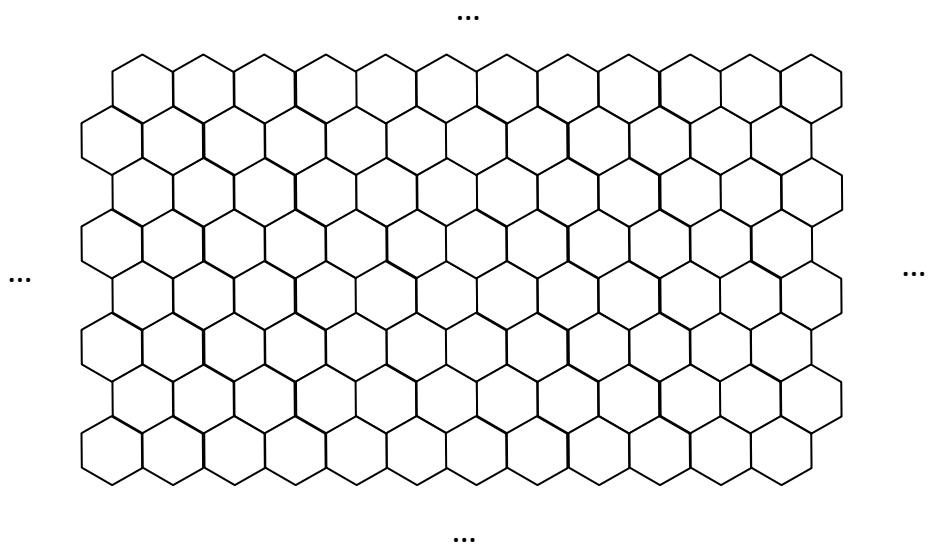
2. Un universo digitale

Cominciamo indicando le caratteristiche proprie di ogni automa cellulare, spiegando come queste vengono declinate nel modello di "universo digitale" (sia U) da noi sviluppato (segnaliamo che nel nostro laboratorio di ricerca abbiamo sviluppato un semplice software, il quale realizza il modello di universo che andiamo a descrivere):

(1) Spazio-tempo discreto: un AC consiste in un reticolo di *celle elementari*. Questo reticolo può essere mono-, bi-, tri-, o multi-dimensionale. Il modello da noi scelto è a due dimensioni – una soluzione che consente insieme di modellare reali fenomeni fisici, preservando una struttura intuitivamente facile da capire.
Una singola cella può essere pensata come l'unità elementare di spazio. Inoltre, in conformità al nostro assunto finitistico, questo spazio deve essere finito: esiste al massimo un numero finito N, per quanto grande, di celle, ossia di unità spaziali minime.[3] Supponiamo che anche il tempo sia discreto e formato da unità minime: gli istanti t_0, t_1, …

(2) Omogeneità del reticolo: tutte le celle sono equivalenti dal punto di vista morfologico. In due dimensioni, ci sono solo tre figure regolari che possono suddividere lo spazio in modo uniforme, occupandolo per intero: triangolo equilatero, quadrato, ed esagono. La più parte degli scienziati che studiano AC scelgono il quadrato (si pensi ad esempio allo storico gioco *Life* di Conway). Nel nostro laboratorio abbiamo effettuato numerose simulazioni e scelto, infine, l'esagono. Il nostro universo avrà dunque un aspetto del genere:

[3] Anche se gli AC studiati in letteratura sono perlopiù edificati su spazi infiniti, la restrizione finitistica entra comunque in gioco quando vengono modellati al computer, stante la natura necessariamente finita della memoria di qualsiasi calcolatore.



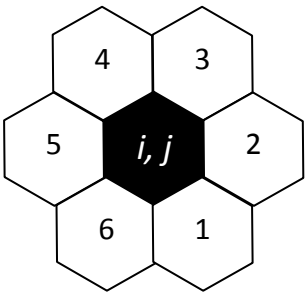
Una struttura a esagoni presenta molti vantaggi topologici (ad esempio: la distanza fra celle è approssimabile come un raggio).[4] Comunque, i risultati computazionali ottenibili con l'AC da noi realizzato possono essere trasferiti tali e quali anche su un classico reticolo a quadrati. Una volta fissata una base dello spazio, ogni cella è univocamente individuata da una coppia di numeri interi, diciamo, <*i*, *j*>: le *coordinate* della sua posizione nello spazio.

Se l'universo U è finito, quali sono i suoi "bordi"? E non ci sarà comunque qualcosa al di là di ogni (presunto) limite dell'universo? Questo antico paradosso a carico dei finitisti riguarda anche chi assume modelli con un'infinità di celle, visto che, si diceva, ogni simulazione di un AC al computer può implementare al massimo un numero finito di celle. Spesso i bordi che racchiudono un universo a *N* celle sono arbitrariamente stipulati come istanzianti un unico stato designato, cosicché non sono affetti dalla computazione che ha luogo all'interno dell'universo. Qualsiasi sistemazione standard, comunque, andrà bene – per i nostri scopi, ciò fa poca differenza.

(3) Stati discreti: ad ogni istante di tempo, ogni cella si trova in uno e uno solo di un insieme finito (diciamo, Σ) di stati discreti (diciamo che *k* sia il numero di questi stati). Indicheremo con l'espressione "$\sigma_{i, j, t}$" lo stato della cella <*i*, *j*> al tempo *t*.

(4) Interazioni locali: ogni cella interagisce soltanto con le sei celle confinanti – il suo *neighbourhood*, per usare la terminologia anglosassone – che possiamo numerare convenzionalmente da 1 a 6 in senso antiorario in questo modo:

[4] Come il corrispettivo tridimensionale dei modelli a celle quadratiche è il cubo, così il corrispettivo tridimensionale di un modello a esagoni è il *dodecaedro rombico*, il quale conserva nelle tre dimensioni molti dei vantaggi mereotopologici dell'esagono.



Indicheremo con "[*i*, *j*]" il *neighbourhood* della cella <*i*, *j*>.

(5) Regola dinamica: ad ogni nuovo istante, ogni cella aggiorna il proprio stato rispetto all'istante immediatamente precedente sulla base di un'unica *regola* (indichiamola con ϕ), tenendo conto degli stati delle celle nel suo *neighbourhood* (tecnicamente, $\phi: \Sigma \rightarrow \Sigma$ è una funzione deterministica da stati a stati). L'idea è che lo stato di una cella all'istante di tempo t+1 sia interamente determinato (in un modo che vedremo fra poco) dagli stati all'istante immediatamente precedente delle sue "vicine immediate", ossia del suo *neighbourhood*, in base alla regola ϕ: $\sigma_{i, j, t+1} = \phi(\sigma_{[i, j], t})$.

Il nostro universo funziona in modo *omogeneo*, non solo nel senso che le celle sono morfologicamente equivalenti, ma anche nel senso che sono tutte governate dall'unica regola ϕ. Esistono automi cellulari non omogenei, ossia in cui celle diverse possono essere governate da regole diverse che determinano le rispettive transizioni di stato. La possibilità di implementare regole molteplici facilita l'ottenimento di molti risultati computazionali (ad esempio: realizzare una macchina di Turing universale nell'AC in questione: cf. Sipper [2004]). Riteniamo però che una sistemazione omogenea con un'unica regola abbia un'eleganza superiore, soprattutto quando (come vedremo sotto) consente di ottenere gli stessi risultati di una non omogenea.

Inoltre, il nostro universo funziona in modo *sincrono*: tutti i punti del reticolo aggiornano il proprio stato simultaneamente. Questa è la sistemazione di gran lunga prevalente negli studi sugli AC; anche se c'è chi ha esplorato l'opzione di universi asincroni (cf. Ingerson e Buvel [1984]), l'opzione sincrona, dato un tempo assoluto, è coerente con la nostra assunzione finitista e "discretista" di fondo. Se infatti celle diverse aggiornano il proprio stato in tempi diversi ma commensurabili, allora l'asincronia è, in certo modo, soltanto apparente. Ma se i tempi sono fra loro incommensurabili, il loro rapporto è un numero irrazionale, il che introduce un elemento incompatibile con la nostra assunzione che tutto, ai livelli ultimi e fondamentali nel nostro universo, sia discreto e numerabile.

Il nostro modello di universo ha dunque tre caratteristiche concettuali tipiche di ogni AC:

(1) **L'informazione è localmente** (a livello di singola cella) **e globalmente** (a livello dell'universo nel suo complesso) ***finita***. Ogni porzione di spazio incorpora e processa una quantità finita di informazione; e lo stesso vale per l'universo nel suo complesso. Chiamiamo *configurazione* del nostro universo U al tempo t lo stato complessivo di U a quel tempo.[5]

[5] Possiamo indicare una configurazione con una notazione vettoriale: la configurazione dell'universo al tempo t, $\bar{\sigma}_t$, non è altro che la sequenza ordinata degli stati di ciascuna cella dell'universo a t, ossia $\bar{\sigma}_t = <\sigma_{0, 0, t}, \sigma_{0, 1, t}, \sigma_{1, 0, t}, \sigma_{1, 1, t}, \sigma_{1, 2, t}, \dots, \sigma_{i, j, t}, \dots, \sigma_{m, n, t}> \in \Gamma$, dove Γ è l'insieme delle configurazioni (*phase space*).

Siccome ogni cella può assumere un numero finito di stati $|\Sigma| = k$, ed esiste un numero finito N di celle nell'universo, esisteranno al massimo k^N configurazioni possibili. Di conseguenza, l'intera evoluzione dell'universo U è rappresentata da un *grafo* di transizione G_Φ finito, che è il grafo della regola di transizione *globale* $\Phi: \Gamma \rightarrow \Gamma$: la funzione indotta dalla nostra regola locale ϕ, che mappa configurazioni su

(2) **L'informazione è processata *localmente*:** nel nostro universo non si dà azione a distanza. Le eventuali "azioni a distanza" (inclusi i misteri della fisica quantistica, come la non-località implicata dal famoso esperimento mentale di Einstein, Podolski e Rosen [1935]) sono da considerarsi come approssimazioni, perché le transizioni di stato sono locali nel tempo e nello spazio: ogni cella interagisce solo con le sue vicine, e il suo stato all'istante *t* dipende solo da quello che succede (nel *neighbourhood*) all'istante immediatamente precedente.

(3) **L'informazione è processata *in parallelo***: quel che accade al livello ultimo della realtà è che un grande numero di punti dello spazio processa i propri stati secondo le connessioni con il "vicinato".

Il nostro universo condivide queste tre caratteristiche, si diceva, con molti altri sistemi di automi cellulari. Ma ne ha anche una quarta, che lo rende davvero molto *speciale* rispetto alla maggior parte dei sistemi noti (ad esempio, *Life* di Conway):

(4) **L'informazione è *interamente conservata***. Ciò è dovuto al fatto che la (unica) regola in base a cui il nostro intero universo si evolve è *reversibile* in senso forte. Per spiegare l'importanza di questo punto, cominceremo spiegando cosa vuol dire che una regola è reversibile, distinguendo due sensi di reversibilità – un senso *debole* e uno *forte*. Formuleremo quindi la nostra regola ϕ; e avendo stabilito che la nostra regola è fortemente reversibile, spiegheremo perché questo fatto ha un'importanza fondamentale.

3. Reversibilità, debole e forte

La grande maggioranza delle regole che governano la dinamica degli AC noti sono *irreversibili*. La (ir)reversibilità delle regole per AC è del tutto analoga a quella per un qualsiasi algoritmo: una regola locale ρ è detta irreversibile quando diversi input possono dar luogo allo stesso output.[6] Quando ciò accade – e accade nella grande maggioranza dei casi: la regola di *Life* di Conway non è invertibile; ma anche le più semplici funzioni logiche booleane che troviamo anche nei motori di ricerca su internet, come AND e OR, non lo sono – non è in generale possibile, guardando solo alla configurazione di un automa al tempo t+1, stabilire quale fosse la sua configurazione a t.

Invece, una regola ρ: A $\rightarrow$ B per un AC si dice (*debolmente*) reversibile quando ha una regola *inversa*,[7] che trasforma gli output della prima nei rispettivi input. Se riusciamo a trovare, data una regola, la sua inversa, possiamo far scorrere un AC "al contrario" e recuperare le sue configurazioni precedenti.

---

configurazioni: $\bar{\sigma}_{t+1} = \Phi(\bar{\sigma}_t)$. La finitezza del nostro universo, e in particolare del grafo G_Φ, implica che, qualsiasi sia la sua configurazione iniziale, la sua evoluzione dovrà esibire un ciclo periodico dopo al massimo k^N iterazioni della regola che lo governa.

[6] E corrispondentemente una regola globale P (ossia una mappa da configurazioni a configurazioni) è detta irreversibile quando diverse configurazioni del sistema a t possono dar luogo a una stessa configurazione a t+1.

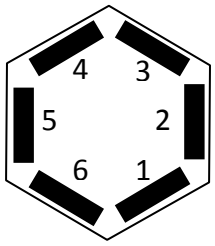
[7] Ossia c'è una ρ^{-1}: B $\rightarrow$ A. Tecnicamente, ρ^{-1} è l'inversa di ρ se e solo se $\rho^{-1} \circ \rho = \mathrm{Id}_A$: A $\rightarrow$ A (e $\rho \circ \rho^{-1} = \mathrm{Id}_B$: B $\rightarrow$ B), dove Id è la funzione identità, e $\circ$ è l'operatore di composizione; dunque $(\rho^{-1} \circ \rho)(\sigma_{i,j,t}) = \rho^{-1}(\rho(\sigma_{i,j,t})) = \sigma_{i,j,t}$. Idem per la regola globale P: $\Gamma \rightarrow \Gamma$, ossia la mappa da configurazioni a configurazioni dell'AC indotta dalla regola locale corrispondente. L'invertibilità di P assicura che ciascuno stato complessivo $\bar{\sigma} \in \Gamma$ dell'universo ha esattamente un predecessore $\bar{\sigma}^{-1} \in \Gamma$, tale che $P(\bar{\sigma}^{-1}) = \bar{\sigma}$. Le sole regole reversibili, dunque, sono quelle che corrispondono a funzioni biiettive (fatto salvo, naturalmente, il fatto che le funzioni iniettive sono invertibili sulla loro immagine, che è un sottoinsieme del rispettivo codominio).

Una regola invece si dice *fortemente* reversibile (o *time-reversal invariant*: cf. Ilachinski [2001]: 95, 370) quando è l'inversa di *se stessa*. Che questa sia una proprietà più forte della precedente, è ovvio: quando una regola è fortemente reversibile, la sequenza di stati passati dell'AC può essere recuperata semplicemente *invertendo il tempo*, e lasciando che l'AC computi i propri stati secondo la propria regola. La stessa configurazione iniziale dell'universo può essere recuperata mediante un'evoluzione al contrario basata sulla stessa regola che lo governava quando il tempo scorreva "in avanti": come se si facessero scorrere al contrario i fotogrammi di un film.

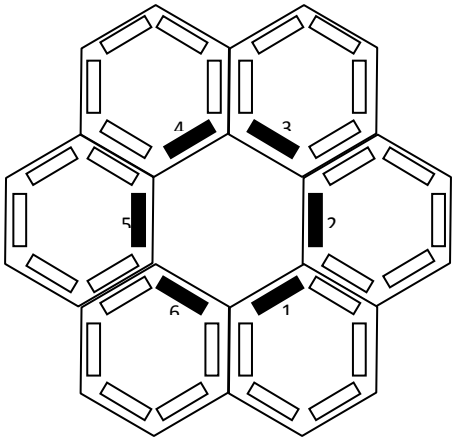
4. Super-regola

Perché in generale la grande maggioranza delle regole dinamiche che governano gli AC non sia reversibile, è chiaro: l'insieme degli stati delle celle che costituiscono il *neighbourhood* di una cella a un tempo *t* non può essere in generale ricostruito soltanto guardando solo allo stato di quella cella a *t*+1. Per ottenere una regola locale che dia luogo a un AC globalmente reversibile occorre dunque procedere con estrema cautela.

Nel nostro modello abbiamo cominciato con l'assumere che lo stato di ogni punto del reticolo sia non un singolo bit, ma una *sestupla* di bit, fisicamente corrispondenti ai sei lati della cella:



Ogni lato può assumere il valore 1 (diciamo, "attivo") o 0 (diciamo, "inattivo").[8] Piuttosto originale è il modo in cui definiamo il *neighbourhood* per la nostra cella. Stabiliamo che ciascuna cella assuma come parte dell'input rilevante per l'applicazione della nostra regola dinamica ϕ soltanto il valore del lato *confinante* di ciascuna delle celle contigue. Possiamo allora numerare questi lati allo stesso modo:

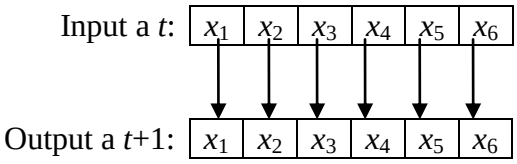


[8] Lo stato della cella $<i, j>$ al tempo *t* è dunque $\sigma_{i, j, t} = <x_1, x_2, x_3, x_4, x_5, x_6>$, dove x_1 corrisponde al lato n.1, x_2 al lato n. 2, etc., e ciascuno degli *x* ha valore $v \in V = \{1, 0\}$. Poiché $|V| = 2$, ciascuna cella avrà dunque $2^6 = 64$ stati istanziabili (ossia, $|\Sigma| = 64$)

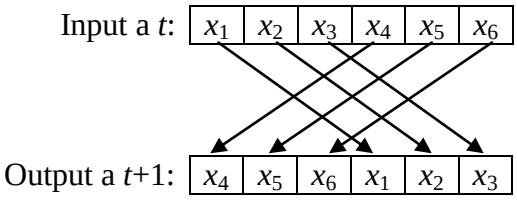
La nostra regola è così una funzione da singole sestuple a singole sestuple, ossia ϕ: $\{1, 0\}^6 \rightarrow \{1, 0\}^6$. Dal punto di vista intuitivo, possiamo pensare alla dinamica in gioco come a una situazione in cui ciascuna cella "legge" o "percepisce" l'output presentatole dalle celle contigue nel rispettivo lato confinante, ed "agisce" esponendo alle vicine il proprio output, "pensato", ossia computato, secondo la regola ϕ. E la regola è, espressa in termini matematici, la seguente:

$$\sigma_{i, j, t+1} = \begin{cases} \text{(a) } \mathit{Perm}(\sigma_{[i, j], t}), \text{ se } \sum_{v[i, j],t} (\text{mod } 2) = 1 \\ \text{(b) } \mathit{Id}(\sigma_{[i, j], t}), \text{ se } \sum_{v[i, j],t} (\text{mod } 2) = 0 \end{cases}$$

Dove "$\sum_{v[i, j],t}$ (mod 2)" indica la sommatoria modulo 2 dei valori $v \in \{1, 0\}$ di ciascun membro della sestupla in input al tempo t. *Id* è la funzione identità definita su Σ, ossia la funzione che manda ciascuna sestupla $\sigma \in \Sigma$ in se stessa: $\mathit{Id}(<x_1, x_2, x_3, x_4, x_5, x_6>) = <x_1, x_2, x_3, x_4, x_5, x_6>$, ovvero:



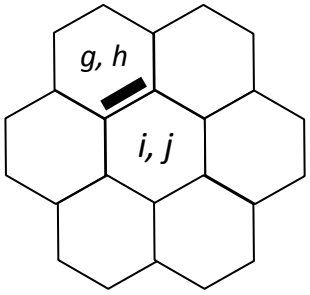
Perm è una permutazione definita su Σ, ossia una funzione *Perm*: $\Sigma \rightarrow \Sigma$ che permuta gli elementi di ciascuna sestupla scambiando i primi tre con gli ultimi tre e viceversa: $\mathit{Perm}(<x_1, x_2, x_3, x_4, x_5, x_6>) = <x_4, x_5, x_6, x_1, x_2, x_3>$, ovvero:



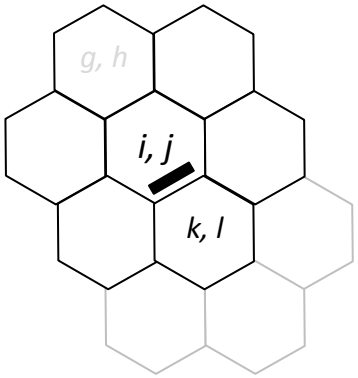
Traducendo dal matematico all'italiano corrente, ciò che la nostra regola garantisce è che:

(a) Se il numero di 1, o di "bit attivi", della sestupla in entrata in una qualsiasi cella è dispari (quindi la loro somma modulo 2 = 1), la cella darà come output la permutazione corrispondente. Come risultato, il segnale in arrivo da una cella contigua verrà trasmesso dal lato opposto. Ad esempio, stabiliamo di colorare un lato di ogni cella di nero se quel lato è "attivo", lasciandolo invece in bianco se è "inattivo". Prendiamo quindi il caso più semplice:

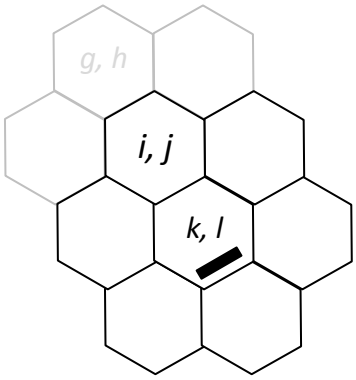
quello di un singolo bit 1, "percepito" da una cella <*i*, *j*> nella cella confinante in alto a sinistra (sia <*g*, *h*>) al tempo *t*, mentre tutti gli altri lati sono "zeri" o "inattivi":



La cella "percepisce" dalle celle del neighbourhood un messaggio ben preciso: "singolo bit attivo in arrivo da in alto a sinistra". E la nostra cella "esporrà" alle vicine di casa questa configurazione:[9]



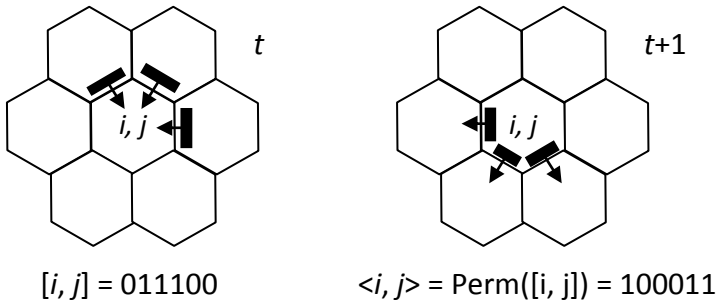
Supponiamo ora che la cella evidenziata come <*k*, *l*> si trovi nella stessa situazione in cui si trovava <*i*, *j*> all'istante precedente, ossia abbia come unico bit "attivo" del suo neighbourhood [*k*, *l*] il bit "espostole" a *t*+1 da <*i*, *j*>. Allora, a *t*+2 avremo:[10]



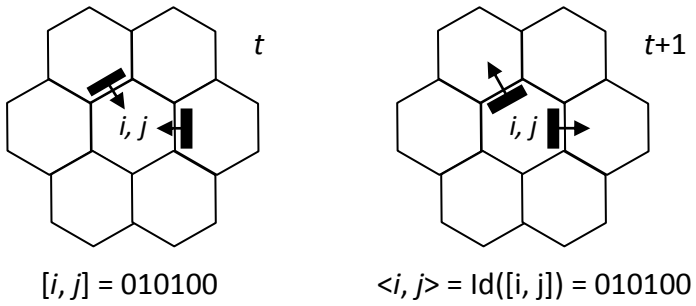
[9] Infatti, [*i*, *j*] = <0, 0, 0, 1, 0, 0> (o, per brevità: 000100). E l'output corrispondente a *t*+1, in base alla nostra regola, è *Perm*(<0, 0, 0, 1, 0, 0>) = <1, 0, 0, 0, 0, 0> (o, per brevità: *Perm*(000100) = 100000), a *t*+1.

[10] Infatti, [*k*, *l*] = 000100, e anche <*k*, *l*> applicherà *Perm*.

L'effetto complessivo sarà una sorta di "moto rettilineo uniforme" del bit attivo da in alto a sinistra verso il basso a destra. In generale, lo stesso accadrà ogni volta che l'input darà un numero dispari di bit attivi. Ad esempio, ecco tre bit attivi in arrivo dalle direzioni 2, 3 e 4:



(b) Quel che accade invece quando la somma dei valori dei membri delle sestuple è pari (e quindi si applica *Id*), è un "rimbalzo" dei segnali che vengono rispediti nelle direzioni di provenienza. Ad esempio, ecco due bit attivi in arrivo dalle direzioni 2 e 4:



5. Proprietà della super-regola

La nostra regola è dunque concettualmente semplicissima: dice che se la somma (modulo 2) dei membri dell'input è pari ossia = 0, allora i segnali possono proseguire con "moto rettilineo uniforme"; mentre se la somma (modulo 2) è dispari = 1, allora i segnali devono "tornare da dove sono venuti". Eppure, questa semplice regola dà luogo a risultati piuttosto interessanti.

Anzitutto, la nostra regola garantisce una *conservazione totale del numero di bit* "attivi" e "non attivi" presenti nell'universo: visto che sia *Perm* che (banalmente) *Id* sono permutazioni, qualunque sia la sestupla presentata come input, l'output conserverà sempre lo stesso numero di 1 e di 0. Oltre ad essere conservativa, la nostra regola è chiaramente *reversibile*: infatti, non solo il numero di bit "attivi" e "inattivi" è da essa conservato, ma ogni input è mappato sia da *Id* che da *Perm* su un output distinto. Infine, la regola è anche *fortemente* reversibile, perché coincide con la sua inversa, su cui è mappata dalla trasformazione che inverte il tempo.[11]

[11] Naturalmente, *Id* è fortemente reversibile; ma anche *Perm* lo è: data una qualsiasi sestupla <*a*, *b*, *c*, *d*, *e*, *f*> come input, non solo essa verrà mappata su un output distinto <*d*, *e*, *f*, *a*, *b*, *c*>; ma anche, la regola per ritornare dall'output <*d*, *e*, *f*, *a*, *b*, *c*> al suo (unico) input è daccapo *Perm*(<*d*, *e*, *f*, *a*, *b*, *c*>) = <*a*, *b*, *c*, *d*, *e*, *f*>. Che la nostra regola ϕ sia (fortemente) reversibile vuol dire che nel grafo globale corrispondente, G_{Φ}, non

6. Reversibilità e fisica dell'informazione

Quel che accade inevitabilmente in una dinamica *non* reversibile è un certo grado di quella che in gergo si chiama "entropia informazionale": l'applicazione di una regola dinamica non reversibile può cancellare una certa quantità di informazione riguardante il passato del sistema. Questo è inevitabile, ad esempio, quando un certo gate logico ha più input che output: un AND produce uno 0 in uscita; quali erano i bit in entrata? 1 e 0, o 0 e 1, o due zeri?

Fino agli anni '70 l'interesse per gli AC dotati di regole reversibili fu piuttosto scarso, perché si riteneva che tali regole non fossero in grado di supportare la computazione universale – ossia la capacità, propria di una macchina di Turing universale (MTU) di valutare qualsiasi funzione computabile. Ora, anche se la conservazione di informazione implicata dalla reversibilità è una bella cosa, senza universalità non succede gran che di interessante dal punto di vista informatico. Ma nel 1973 Charles Bennett mostrò che la computazione universale non ha bisogno di essere necessariamente realizzata mediante gate logici irreversibili, e nel 1976 Toffoli produsse i primi AC reversibili capaci di computazione universale, accendendo un interesse per questo genere di modelli che prosegue ancora oggi.

Vi sono almeno due ragioni – ampiamente evidenziate in letteratura – per cui qualsiasi computazione irreversibile è insoddisfacente dal punto di vista teorico. La prima, più tecnica, è che la computazione irreversibile è dispendiosa: fin dai tempi di von Neumann si sa che cancellare un bit di informazione è un dispendio energetico (almeno $\sim 3 \cdot 10^{-21}$ joules a temperatura ambiente): la perdita di informazione ha un costo termodinamico, che va pagato in termini di dispersione in energia "non computazionale". Un dispendio, dunque, che non è dovuto a inefficienza nel *design* dei circuiti logici o nei materiali utilizzati, ma ha una radice più fondamentale: è una conseguenza diretta dell'esistenza di computazioni logiche irreversibili. Man mano che la tecnologia avanza, il problema di questo dispendio energetico, e l'esigenza di costruire computer reversibili a dissipazione energetica virtualmente nulla, si faranno sempre più pressanti.

La seconda ragione è che qualsiasi computazione irreversibile può rispecchiare solo una realtà fisica macroscopica, molto lontana dal livello ontologico ultimo. La ragione di ciò è che le leggi dinamiche della fisica sono *reversibili* a livello microscopico e deterministico: il che vuol dire che distinti stati iniziali nell'evoluzione di un sistema microfisico portano sempre a distinti stati finali; e ciò vale sia per la formulazione classica delle leggi, che per quella in termini di meccanica quantistica.

La conseguenza di questa situazione è chiara: chiunque voglia tentare di simulare attraverso un AC procedure di computazione *realmente* radicate nella natura delle entità di base che compongono il nostro universo fisico, dovrebbe ricorrere a regole dinamiche reversibili, a meno di dover abbandonare quel forte isomorfismo fra realtà fisica e computazionale di cui si diceva sopra. Viceversa, nella misura in cui questo isomorfismo è preservato nel nostro modello, questo si candida a fornire indicazioni importanti per la realizzazione di sistemi computazionali ad alte prestazioni: sistemi che utilizzano le risorse offerte dal mondo fisico nel modo più efficiente.

---

compare alcun cosiddetto *garden-of-eden*, ossia alcuna configurazione del nostro universo che possa figurare solo come configurazione *iniziale*, ma mai essere prodotta dall'evoluzione dell'universo. Infatti i risultati combinati di Moore e Myhill [1963] mostrano che esistono configurazioni che non hanno predecessori se e solo se ve ne sono alcune che ne hanno più di uno, il che è escluso dalla reversibilità.

7. Computazione universale

Nonostante la sua semplicità, la nostra regola ha qualcosa che la rende davvero notevole: essa infatti fa sì che nel nostro universo possano esistere *computer*! Consente cioè l'esistenza di macchine di Turing universali, in grado di computare (se accettiamo la Tesi di Turing) tutto ciò che è computabile. Quando una regola per un AC consente di fare ciò, si dice che è *computation-universal*, capace di computazione universale. La nostra regola ϕ è dunque *computation-universal*.

Naturalmente, occorre provarlo. In letteratura vi sono due modi fondamentali per mostrare che una certa regola per AC è capace di computazione universale: uno consiste nel ridurre matematicamente la regola a un'altra di cui è già noto che è computation-universal. Un altro, più "costruttivo", consiste essenzialmente nel mostrare in modo diretto che tutti i "mattoni di base" di un computer convenzionale o di una MTU possono essere emulati da pattern generati dalla regola nel suo universo d'azione. Questa è la strategia adottata nella dimostrazione originaria fornita da Berkelamp, Conway e Guy [1982] del fatto che *Life* è capace di computazione universale, ed è stata seguita anche da noi per la nostra super-regola ϕ.

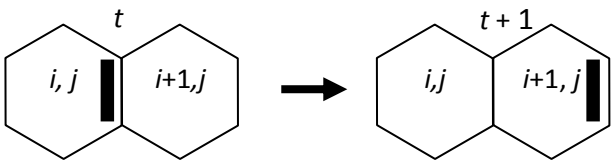
Perché un AC possa essere computation-universal, in generale, occorre che esso possa ospitare configurazioni delle celle che lo compongono, tali da poter essere viste come *pattern* stabili nel tempo: pacchetti localizzati di informazioni trasportabili attraverso il reticolo. Più precisamente, i "primitivi" della computazione consistono nella possibilità di (1) *trasmettere* e (2) *conservare* segnali, che richiedono una memoria e canali di trasmissione per i bit (attraverso cui questi possano scorrere o essere ri-direzionati, inviati in molteplice copia in direzioni diverse, etc.), nonché (3) la possibilità di *processare* i segnali, la quale richiede un insieme di gate logici (come AND, OR e NOT) che sia funzionalmente completo. Una volta dimostrato che un AC è in grado di contenere tutti questi mattoni di base, costruire di fatto circuiti corrispondenti a quelli di un computer convenzionale è semplicemente una questione di routine, magari noiosa da completare, ma concettualmente secondaria.

Anzitutto, come hanno mostrato Edward Fredkin e Tommaso Toffoli nel loro classico *Conservative Logic* [1982], i punti (1) e (2) summenzionati sono realizzati concettualmente da un unico primitivo in qualsiasi AC reversibile: "da un punto di vista relativistico, non c'è distinzione intrinsica fra *conservazione* e *trasmissione* di segnali. Occorre dunque cercare un singolo primitivo di conservazione-trasmissione in grado di svolgere indifferentemente ambo le funzioni" (Fredkin e Toffoli [1982]: 224; cf. anche Margolus [1984]).

Ad esempio, consideriamo un segnale *s* che connette due luoghi dello spaziotempo, L_1 e L_2. Se L_1 e L_2 sono separati spazialmente, si dirà che *s* è stato trasmesso dall'uno all'altro in un certo intervallo di tempo. Se però L_1 e L_2 sono colocalizzati, diremo che *s* è stato conservato in quel(l' unico) luogo $L_1 \equiv L_2$: ad esempio (cf. Fredkin e Toffoli [1982]: 226), *s* può essere un messaggio che spedisco alla mia segretaria in forma di e-mail (trasmissione), oppure *s* può essere lasciato sulla mia scrivania in forma di nota perché la segretaria lo raccolga il giorno dopo (conservazione). "Dunque, è chiaro che i termini 'conservazione' e 'trasmissione' descrivono dal punto di vista di quadri di riferimento diversi un unico processo fisico" (Ibid).

Ora, nel nostro universo *ogni singola cella esagonale svolge la doppia funzione di conservazione-trasmissione di segnali*, in quanto è "animata" dalla nostra regola. Sappiamo infatti che, quando il numero di bit attivi percepiti dalla cella in input è dispari (e in particolare, quando tale numero è uguale a uno), la singola cella fa traslare il segnale alla cella accanto con "moto rettilineo uniforme". Ogni cella funziona dunque come una *unità-circuito*:

il ruolo di una unità-circuito è quello di spostare il bit di informazione da un punto al punto contiguo nell'intervallo di un'unità di tempo – ad esempio:



Com'è chiaro, si possono dunque ottenere circuiti di lunghezza arbitraria semplicemente come successioni di unità-circuito, ossia considerando sequenze di celle contigue l'una all'altra. Un circuito rettilineo di lunghezza *n* è un percorso dove un segnale qualsiasi, come una sequenza di bit, può scorrere, e i cui estremi, poiché il segnale si sposta complessivamente di una cella esattamente per ogni unità di tempo, sono separati anche da *n* unità di tempo.

Naturalmente, una memoria finita (l'unica possibile, stante gli stretti requisiti "finitari" da noi seguiti nella definizione del nostro universo) è facile da implementare, utilizzando semplicemente sequenze codificate di bit che, attraverso opportune "deviazioni" indotte mediante scontri precisi, si muovono in circolo; oppure, che formano configurazioni stabili o periodiche. Ma vi è anche un senso molto più forte e radicale, in cui si può dire che il nostro universo ha "memoria": stante che la sua regola è *time-reversal invariant*, infatti, l'universo *mantiene memoria perfetta* della propria configurazione iniziale per tutto il tempo della propria evoluzione, visto che nessun bit d'informazione viene creato o distrutto, e che qualsiasi configurazione può sempre essere recuperata per via della reversibilità.

La parte complicata del nostro lavoro consiste nel mostrare che il nostro universo è in grado di processare le proprie informazioni sulla base di un insieme funzionalmente completo di gate logici. È ben noto che in qualsiasi AC reversibile la computazione non può che essere un routing condizionale di segnali. Detto alla buona: un AC reversibile deve rispettare un requisito di *conservatività* per cui "i segnali sono trattati come oggetti inalterabili che possono essere spostati durante una computazione, ma mai creati o distrutti" (Fredkin e Toffoli [1982]: 227). È ancora più noto che una logica con AND, NOT e FAN-OUT o *copia* di segnali forma un insieme universale di primitivi logici; ma, come già osservato, l'ordinario AND non è reversibile e, intuitivamente, un FAN-OUT, duplicando un segnale, sembra implicare una *creatio ex nihilo* di informazione che contraddice il requisito di conservatività.

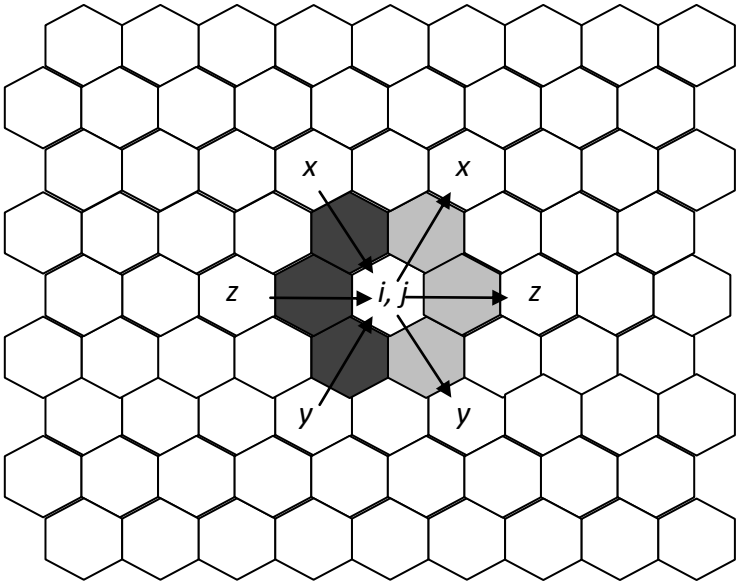
Eppure, questi problemi sono risolti nel nostro universo in modo sorprendentemente brillante. L'idea fondamentale è tanto semplice quanto elegante: *ogni singola cella <i, j> del nostro universo può essere vista come un gate logico* – anzi, precisamente come un gate logico *universale* – proprio in quanto è un luogo di transizione (trasmissione o collisione) di bit e sequenze di bit di informazione.

Anzitutto, sappiamo che un gate logico conservativo deve avere tanti input quanti output. Ora, è facile vedere che gate logici con 2 input e 2 output non possono darci la ricercata universalità.[12] Occorre dunque aumentare il numero delle linee di input-output. Esistono in letteratura gate logici universali e reversibili, come il Fredkin gate (che ha una tavola di verità con 3 input e 3 output), ma non sono semplici da implementare in un AC con i requisiti di "plausibilità fisica" che noi cerchiamo. Anche noi assumiamo come base il numero 3, e trattiamo la singola cella qualsiasi del nostro reticolo come un gate che prende

[12] In questo caso, abbiamo infatti $4^4 = 256$ possibili tavole di verità, e solo 24 sono reversibili. È stato dimostrato da Margolus [1988] che tutte le 24 tavole reversibili possono essere ridotte a composizioni dello XOR booleano, e lo XOR da solo non è universale.

input da tre dei suoi sei lati, generando un output da altri tre. Questo vuol dire intuitivamente che, quando vorremo simulare operazioni come la negazione, NOT o la congiunzione, AND, alcuni input e output saranno "pleonastici".[13]

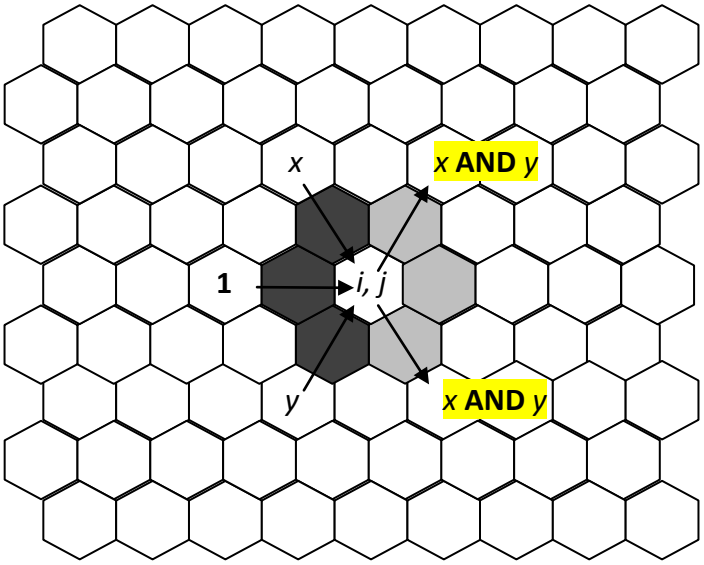
Consideriamo allora la singola cella <*i*, *j*>; supponiamo che prenda come input i segnali in arrivo dalle direzioni *x*, *y* e *z* qui segnate (corrispondenti alle direzioni 4, 5 e 6 nella numerazione convenzionale vista sopra) in grigio scuro, e restituisca come output i segnali che escono dai tre lati opposti, segnati in grigio chiaro:



È facile mostrare che i primitivi della computazione, quali la negazione NOT, la congiunzione logica AND, il FAN-OUT di segnali (oltre a ritardi di segnale, deviazioni e rimbalzi di ogni tipo) possono essere realizzati sulla base di questo semplice schema. Abbiamo presentato una prova dettagliata di questi fatti in un volume in corso di pubblicazione (Berto, Rossi e Tagliabue [2010]). Per non tediare il lettore con troppi dettagli tecnici, ci limiteremo a fornire un esempio, trattando il caso della congiunzione. Ogni cella del nostro universo può implementare una congiunzione logica, ossia un AND, reversibile: basta considerare *z* come una "linea di controllo", che viene attivata fissandone il valore in $z = 1$ (ossia facendo sopraggiungere in direzione *z* una sequenza di bit "attivi" o 1). In questo modo, considerando i bit in arrivo nelle direzioni *x* e *y*, la cella produce sia nella direzione di uscita *x* che nella direzione di uscita *y* un output che è la congiunzione di *x* e *y*: l'output di *x* (e di *y*) = 1 se e solo se l'input di $x = 1$ e quello di $y = 1$, altrimenti l'output di *x* (e di *y*) = 0:

z	*x*	*y*	*Output x e Output y*: *x* AND *y*
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	0

[13] Tecnicamente: data una funzione *f* con un certo numero di input-output, si può ottenere una funzione *g* "immersa" in *f*, ossia con un numero di input-output minore, assegnando valori prespecificati *costanti* a certe linee di input, e disinteressandoci di alcune linee di output (*garbage bits*, come li si chiama in letteratura). Si può allora dire che *f* simula *g* mediante costanti e *garbage*.



Le configurazioni per ottenere NOT e FAN-OUT sono del tutto analoghe.

8. Conclusioni

La nostra regola, supportando i primitivi della computazione di un qualsiasi computer digitale, è capace di computazione universale; se identifichiamo la configurazione iniziale del nostro AC con i dati d'inizio dell'universo, l'evoluzione di questo è quella prescritta dalla nostra regola ϕ (con la corrispondente regola globale indotta Φ): il programma con cui esso elabora quei dati. E sappiamo che, grazie a ϕ, questo universo nella sua evoluzione può produrre ed ospitare computer, ossia macchine di Turing universali, e dunque implementare qualsiasi algoritmo finito e valutare qualsiasi funzione computabile.

Inoltre, tutti gli operatori logici realizzabili dalle nostre celle sono conservativi (ugual numero di 1 e di 0 in entrata e in uscita) e fortemente reversibili: se re-introduciamo gli output di uno qualsiasi di questi gate logici come suoi input e facciamo funzionare il gate al contrario, il risultato saranno gli input originari. Se teniamo nota di tutti i risultati intermedi della computazione e facciamo scorrere l'algoritmo al contrario quando abbiamo finito, torneremo là dove abbiamo cominciato, senza usare energia o generare calore. E nel frattempo, avremo calcolato il risultato che cercavamo!

Infine, notiamo che, se nel nostro universo possono esistere MTU, esso è *imprevedibile* in un senso preciso: per il Teorema della Fermata, notoriamente, non esiste un algoritmo generale che possa predire se una MTU, dato un certo input, si arresterà dopo *n* passi fornendoci il suo output. E lo stesso vale per l'evoluzione del nostro universo, in quanto può implementare computer universali: non esiste una scorciatoia computazionale per predire gli esiti della sua evoluzione. L'emulazione più efficiente possibile dell'evoluzione del nostro universo è *questa stessa* evoluzione. Stante che la nostra regola è *computation-universal*, oltre un certo livello di complessità dei pattern, l'unica strada per sapere quali configurazioni saranno prodotte dall'universo consiste nel far scorrere il software che lo implementa a partire da una configurazione iniziale, e… Nello stare a guardare – nel mettere in atto, insomma, un esercizio di matematica sperimentale. Se le nostre congetture sull'isomorfismo fra realtà e informazione sono giuste (in particolare: gli operatori logici possono essere – come direbbe

un filosofo – "ramseyficati" mediante la loro tavola di verità; e il mondo fisico fornisce realizzatori per quei ruoli informazionali), questo si rivelerà anche essere un esercizio di *fisica* sperimentale!

Riferimenti

Bennett C. [1973], "Logical Reversibility of Computation", *IBM Jour. Res. Dev. 6*, pp. 525-32.

Berkelamp E.R., Conway J.H., Guy R.K. [1982], *Winning Ways for Your Mathematical Plays*, Academic Press.

Berto F., Rossi G. e Tagliabue J. [2010], *La matematica dei modelli di riferimento*, in uscita.

Einstein A., Podolski B. e Rosen N. [1935], "Can Quantum Mechanical Description of Physical Reality Be Considered Complete?", *Phys. Rev. 37*, p. 777.

Fredkin E. [1982],

Fredkin E. [1993], "A New Cosmogony", in PhysComp92, Los Alamitos, CA: Computer Society Press, pp. 116-121.

Fredkin E. e Toffoli T. [1982], "Conservative Logic", *International Journal of Theoretical Physics 21*, pp. 219-54.

Ilachinski A. [2001], *Cellular Automata. A Discrete Universe*, Singapore: World Scientific.

Ingerson T.E. e Buvel R.L. [1984], "Structure in Asynchronous Cellular Automata", *Physica 10D*, pp. 59-68.

Margolus N. [1984], "Physics-Like Models of Computation", *Physica 10D*, pp. 81-95.

Margolus N. [1988], *Physics and Computation*, Ph.D. Thesis, Tech. Rep. MIT.

Myhill J. [1963], "The Converse of Moore's Garden of Eden Theorem", *Proc. Am. Math. Soc. 14*, pp. 685-6.

Neumann J. von [1948],

Sipper M. [2004], *Evolution of Parallel Cellular Machines. The Cellular Programming Approach*, Springer.

Toffoli T. [1977], "Computation and Construction Universality of Reversible Cellular Automata", *Jour. Comp. Sys. Sci. 15*, p. 213.

Toffoli T. e Margolus N. [1990], "Invertible Cellular Automata: a Review", *Physica D45*, pp. 229-52.

Wolfram S. [2002], *A New Kind of Science*, Wolfram media.

Zuse K. [1982], "The Computing Universe", *Int. Jour. Of Theo. Phy. 21*, pp; 589-600.