Sommario

La finalità di questa trattazione è quella di esporre le nozioni fondamentali della teoria degli automi cellulari, con una particolare attenzione per quanto riguarda gli aspetti legati al loro potere espressivo. Partirò presentando alcuni dei risultati noti di questa teoria. Si parlerà anche brevemente del rapporto fra automi cellulari e mondo naturale, sia per quanto riguarda la simulazione, sia in relazione alle possibili implementazioni fisiche. Successivamente analizzerò le difficoltà relative ad una buona definizione di universalità computazionale per questo modello di calcolo e fornirò alcune tracce di dimostrazione per automi cellulari noti.

Indice

1 Introduzione **2**
1.1 Cos'è un automa cellulare 2
1.2 Il gioco della vita di Conway 3

2 Basi **5**
2.1 Definizione formale 5
2.2 Automi cellulari elementari 7
2.3 Termini e nozioni di base 9
2.4 Configurazioni finite e periodiche 12

3 Alcune proprietà **16**
3.1 Iniettività e suriettività 16
3.2 Automi cellulari come sistemi dinamici 19
3.3 Invertibilità e leggi di conservazione 23

4 Universalità **30**
4.1 Definire l'universalità computazionale 30
4.2 Universalità diretta 32
4.3 Caso bidimensionale 34
4.4 Caso unidimensionale 40
4.5 Universalità spontanea 43
4.6 Caso invertibile 49

5 Conclusioni **53**

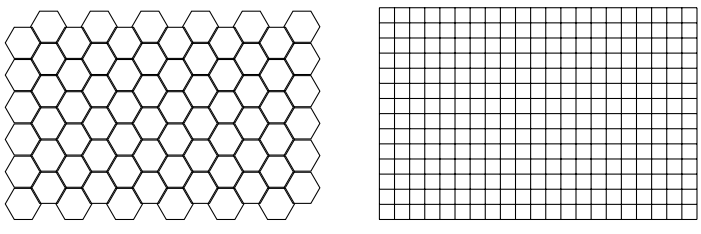
Capitolo 1

Introduzione

1.1 Cos'è un automa cellulare

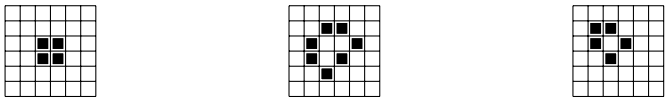
Gli automi cellulari sono fra i più vecchi modelli di calcolo ispirati dal mondo fisico. I primi studi furono effettuati da Von Neumann negli anni quaranta ed erano motivati da interessi nel campo della biologia. L'intento era quello di definire *sistemi artificiali* capaci di simulare il comportamento di *sistemi naturali*. In particolare la ricerca era orientata a modelli capaci di replicarsi che fossero computazionalmente universali. Si desiderava studiare modelli di calcolo che avessero la proprietà, ispirata probabilmente dal cervello umano, di non distinguere la componente di calcolo da quella di memoria. Le interazioni in un automa cellulare sono *locali*, *deterministiche* e *sincrone*; il modello è *massicciamente parallelo*, *omogeneo* e *discreto* sia nello spazio, sia nel tempo. Il legame degli automi cellulari con il mondo fisico e la loro semplicità sono alla base del loro successo nel campo della simulazione di fenomeni naturali.

Informalmente possiamo dire che un automa cellulare è definito su di una griglia regolare infinita di qualunque dimensione in cui la forma delle celle sia sempre la stessa (vedi figura 1.1). Ogni punto della griglia viene chiamato cel-



(a) (b)

Figura 1.1: Esempi di griglie bidimensionali: griglia con cellule esagonali (a) e griglia con cellule quadrate (b).



(a) (b) (c)

Figura 1.2: Esempi di *nature morte*: un blocco (a), una foglia (b) e una barca (c).

(a) (b) (c) (d)

Figura 1.3: Due esempi di *oscillatori*: un blinker (a,b) e un fungo (c,d).

lula. In ogni istante ogni cellula della griglia si trova in uno stato fra i finiti stati possibili. A intervalli di tempo discreti tutte le cellule contemporaneamente aggiornano il proprio stato, basandosi su quello di un numero finito di altre cellule dette *vicini*. Solitamente il vicinato di ogni cellula contiene tutte quelle che hanno distanza minore di un certo valore. La regola applicata da ogni cellula per determinare il suo prossimo stato è la stessa. L'evoluzione globale di un automa cellulare è l'evoluzione della configurazione complessiva della rete di cellule. Ogni configurazione viene talvolta detta *generazione*, se messa in relazione all'evoluzione dello stato complessivo della rete di cellule o di un pattern.

Come si può immaginare è possibile definire automi cellulari molto diversi fra loro, variando il numero di dimensioni della griglia, l'insieme degli stati o altro. Non ci meraviglia particolarmente che gli automi cellulari di tipo più complesso siano computazionalmente equivalenti alle macchine di Turing. Al contrario, un risultato molto sorprendente è che anche automi cellulari incredibilmente semplici si siano rivelati computazionalmente universali.

1.2 Il gioco della vita di Conway

Presentiamo come esempio un automa cellulare molto famoso, ideato da Conway [1], che prende il nome di *game-of-life*. La griglia è bidimensionale e le cellule sono quadrate. Una cellula può essere viva (rappresentata con un quadrato nero ■) o morta (rappresentata con un quadrato bianco □). Ogni cellula modifica il

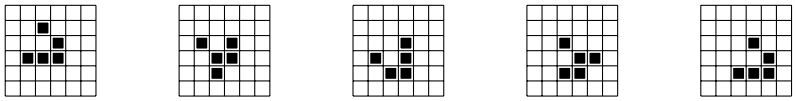


Figura 1.4: Esempio di *nave spaziale*: un aliante (*glider*) che si sposta.

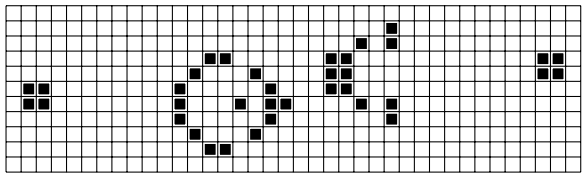


Figura 1.5: Esempio di *cannone*, il più piccolo conosciuto: cannone di Gosper.

proprio stato, in funzione dello stato delle otto cellule che ha attorno (i vicini), in accordo alle seguenti regole:

- una cellula viva muore se il numero di vicini vivi è inferiore a due (morte per isolamento);
- una cellula viva muore se il numero di vicini vivi è maggiore di tre (morte per sovraffollamento);
- una cellula morta diventa viva se il numero di vicini vivi è esattamente tre (nascita)[1];
- in ogni altro caso una cellula conserva il proprio stato.

Possiamo notare che le regole sono piuttosto semplici, ma il comportamento delle configurazioni, sorprendentemente, è tutt'altro che banale. In effetti Conway ha dimostrato che è indecidibile stabilire se un dato pattern porterà prima o poi ad una configurazione in cui tutte le cellule sono morte.

È stata studiata una classificazione, per riferirsi alle varie classi di pattern che si possono osservare in una configurazione di questo automa cellulare.

Natura morta: si tratta di pattern che rimangono immutati nel tempo, punti fissi della regola di aggiornamento (figura 1.2).

Oscillatore: strutture temporalmente periodiche, non rimangono immutati, ma dopo un numero finito di aggiornamenti si ripresentano in modo identico e nello stesso punto della griglia (figura 1.3).

Nave spaziale: pattern che dopo un numero finito di generazioni si ripetono identiche, ma traslate nello spazio (figura 1.4).

Cannone: un pattern che come un oscillatore ritorna nello stato iniziale dopo un numero finito di generazioni, ma che inoltre emette navi spaziali (figura 1.5).

Si noti che molti dei pattern osservati non sono stati disegnati ad hoc, ma sono emersi da configurazioni iniziali casuali. Durante l'evoluzione di un pattern infatti gli oggetti collidono fra loro, talvolta distruggendosi, talvolta creando nuovi pattern.

[1] Sì, ogni cellula ha tre genitori in questo gioco.

Capitolo 2

Basi

2.1 Definizione formale

Quella che presentiamo adesso è la definizione formale di automa cellulare alla quale faremo riferimento nel resto della trattazione.

Definizione 1. *(Automa cellulare). Si dice automa cellulare una quadrupla*

$$A = (d, S, N, f)$$

dove:

- $d \in \mathbb{Z}_+$ *è la dimensione dello spazio* $\mathbb{Z}^d$ *su cui è definito l'automa cellulare*[1]
- S *è un insieme finito di stati*
- $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$ *è una tupla di* m *elementi distinti di* $\mathbb{Z}^d$ *(detta* indice di vicinato*)*
- $f : S^m \longrightarrow S$ *è la* regola di aggiornamento locale *che definisce per ogni* configurazione *del* vicinato*, quale stato la* cellula *assumerà nella prossima generazione*

Talvolta la definizione di automa cellulare forza ogni cellula ad appartenere al proprio vicinato [4].

Lo spazio $\mathbb{Z}^d$ viene detto *spazio cellulare* e i suoi elementi sono detti *cellule*. Una *configurazione* c di un automa cellulare è una funzione che associa uno stato ad ogni cellula.

$$c : \mathbb{Z}^d \longrightarrow S$$

[1] L'assunzione che lo spazio sia $\mathbb{Z}^d$ e che quindi le cellule siano quadrate non riduce il potere espressivo del sistema [22].

Lo stato della cellula $\vec{n} \in \mathbb{Z}^d$ nella configurazione c è indicato allora come $c(\vec{n})$. Si noti che l'insieme delle possibili configurazioni è non numerabile. L'*indice di vicinato* definisce la posizione relativa del vicinato di una cellula. Il *vicinato* di una cellula $\vec{n} \in \mathbb{Z}^d$ è una ennupla contenente le cellule $(\vec{n}+\vec{n}_1, \vec{n}+\vec{n}_2, \ldots, \vec{n}+\vec{n}_m)$. Risulta abbastanza naturale indicare il vicinato di una cellula e l'indice di vicinato di un automa cellulare come insiemi di cellule o di posizioni relative anziché come ennuple. Una volta definito un ordine sugli elementi di $\mathbb{Z}^d$ è facile osservare che le due rappresentazioni sono equivalenti e per questo ci sentiremo liberi di passare dall'una all'altra a seconda di quale risulti più comoda. La *configurazione successiva* di un automa cellulare la cui configurazione attuale è $c \in S^{\mathbb{Z}^d}$ è una configurazione c' tale che

$$\forall \vec{n} \in \mathbb{Z}^d . c'(\vec{n}) = f((c(\vec{n}+\vec{n}_1), c(\vec{n}+\vec{n}_2), \ldots, c(\vec{n}+\vec{n}_m)))$$

È evidente che ad ogni configurazione di un automa cellulare corrisponde univocamente una configurazione successiva. Pertanto possiamo definire la *funzione di transizione globale* $G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ di un automa cellulare come l'unica funzione dall'insieme $S^{\mathbb{Z}^d}$ in sé stesso che verifica

$$\forall c \in S^{\mathbb{Z}^d} . \forall \vec{n} \in \mathbb{Z}^d . G(c)(\vec{n}) = f((c(\vec{n}+\vec{n}_1), c(\vec{n}+\vec{n}_2), \ldots, c(\vec{n}+\vec{n}_m)))$$

Talvolta per esplicitare il suo rapporto con l'automa cellulare A, la sua funzione di transizione globale verrà indicata come $G[A]$.

Generalmente identificheremo la funzione di transizione globale di un automa cellulare con l'automa stesso. Questo è corretto solo parzialmente, in quanto a più automi cellulari (quadruple) corrisponde la stessa funzione. Si pensi ad esempio ad un'estensione dell'indice di vicinato che viene ignorata dalla regola di aggiornamento locale: il comportamento dell'automa sarebbe sostanzialmente lo stesso di quello originale e quindi anche la funzione di transizione globale.

Perché la funzione di transizione globale di due automi cellulari sia uguale questi devono sicuramente avere stessa dimensione d e insieme di stati S, ma possono differire per quanto riguarda l'indice di vicinato. Un elemento di $S^{\mathbb{Z}^d}$ che appartiene all'indice di vicinato di un automa cellulare viene detto *vicino fittizio* se non influenza mai il valore calcolato dalla *regola di aggiornamento locale*. È possibile dimostrare che se due automi cellulari hanno la stessa funzione di transizione globale allora i loro indici di vicinato sono uguali a meno di comprendere vicini fittizi. Da questo si può dedurre che due automi cellulari che hanno stessa funzione di transizione globale e non possiedono vicini fittizi hanno identica regola di aggiornamento locale. Vale quindi il seguente teorema

[14].

Teorema 1. *Dati due automi cellulari A e B, se $G[A] = G[B]$ allora o vale $A = B$ oppure l'indice di vicinato di A o di B contiene un vicino fittizio. Equivalentemente possiamo dire che se né A né B possiedono vicini fittizi allora $A = B \iff G[A] = G[B]$.*

In virtù di questo risultato ci riferiamo alla funzione di transizione globale di un automa cellulare A come *funzione dell'automa cellulare* A e consideriamo *equivalenti* due automi cellulari ($A \equiv B$) se la loro *funzione di transizione globale* è identica ($G[A] = G[B]$).

Osserviamo inoltre che se G e H sono due funzioni di automi cellulari allora lo è anche la loro composizione $G \circ H$.

2.2 Automi cellulari elementari

Presentiamo ora una classe di automi cellulari particolarmente semplici, nota con il nome di *automi cellulari elementari*, che ci saranno utili nel corso della trattazione. Gli *automi cellulari elementari* sono in un certo senso gli automi cellulari più semplici che possiamo immaginare (o quasi): hanno dimensione uno e ogni cellula può trovarsi in uno di due stati, inoltre il vicinato di ogni cellula contiene sé stessa e le due cellule adiacenti.

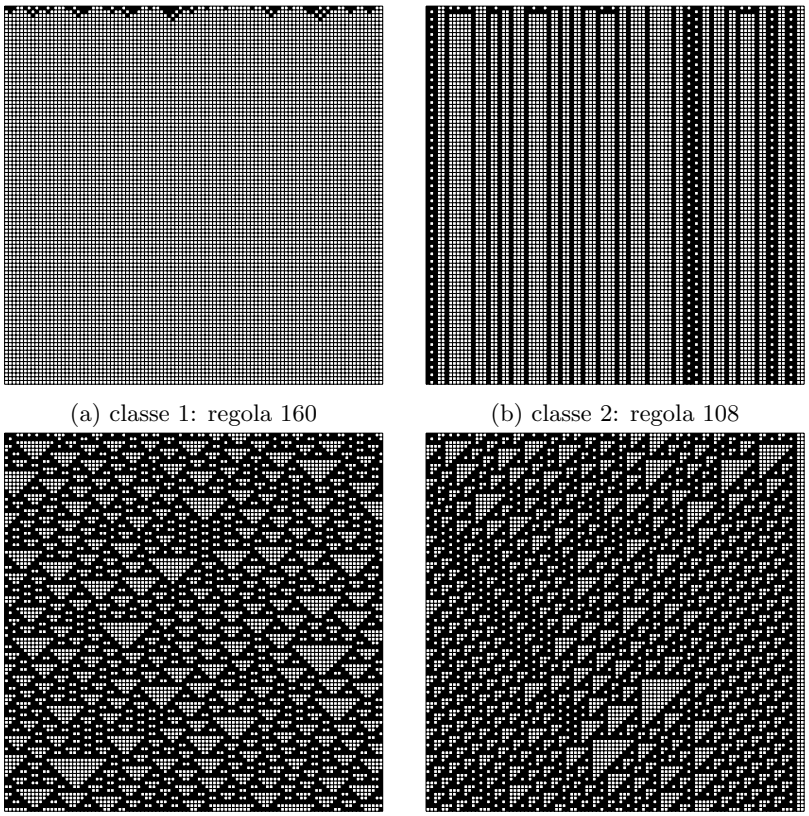
Definizione 2. *(automa cellulare elementare). Un automa cellulare (d, S, N, f) si dice* elementare *se e solo se:*

- $d = 1$
- $S = \{0, 1\}$
- $N = (-1, 0, 1)$

L'unica cosa che distingue un automa cellulare elementare dall'altro è la *regola di aggiornamento locale* $f : S^3 \longrightarrow S$. In teoria esistono quindi $2^8 = 256$ automi cellulari elementari. In realtà molti di questi sono equivalenti a patto di rinominare gli stati o invertire la destra con la sinistra nello spazio cellulare. In definitiva il numero di regole effettivamente differenti è 88.

Un modo comodo di riferirsi ad un automa cellulare elementare che ormai è diventato uno standard è quello di usare il suo *numero di Wolfram*. Una regola è specificata univocamente dal valore che assume su ogni possibile combinazione di valori in ingresso, ovvero dalla sequenza di otto bit

$$f(1,1,1)\ f(1,1,0)\ f(1,0,1)\ f(1,0,0)\ f(0,1,1)\ f(0,1,0)\ f(0,0,1)\ f(0,0,0)$$



(a) classe 1: regola 160 (b) classe 2: regola 108

(c) classe 3: regola 126 (d) classe 4: regola 110

Figura 2.1: Prime 100 righe dei *diagrammi spazio tempo* di un automa cellulare elementare per ogni *classe di Wolfram*, la configurazione iniziale è casuale.

che, se considerata come un numero in base binaria è il *numero di Wolfram* dell'automa cellulare. Talvolta la regola di aggiornamento di un automa cellulare elementare con numero di Wolfram n viene detta *regola* n.

Esempio 1. *(regola 110) L'espansione binaria ad* 8 *bit del numero decimale 110 è 01101110, quindi la regola di aggiornamento locale dell'automa cellulare elementare con numero di Wolfram* 110 *è* f *tale che:*

$$f(1,1,1) = 0 \; f(1,1,0) = 1 \; f(1,0,1) = 1 \; f(1,0,0) = 0$$
$$f(0,1,1) = 1 \; f(0,1,0) = 1 \; f(0,0,1) = 1 \; f(0,0,0) = 0$$

Wolfram ha studiato empiricamente il comportamento degli automi cellulari e ha proposto una loro suddivisione in quattro classi [25].

Definizione 3. *(classificazione di Wolfram).*

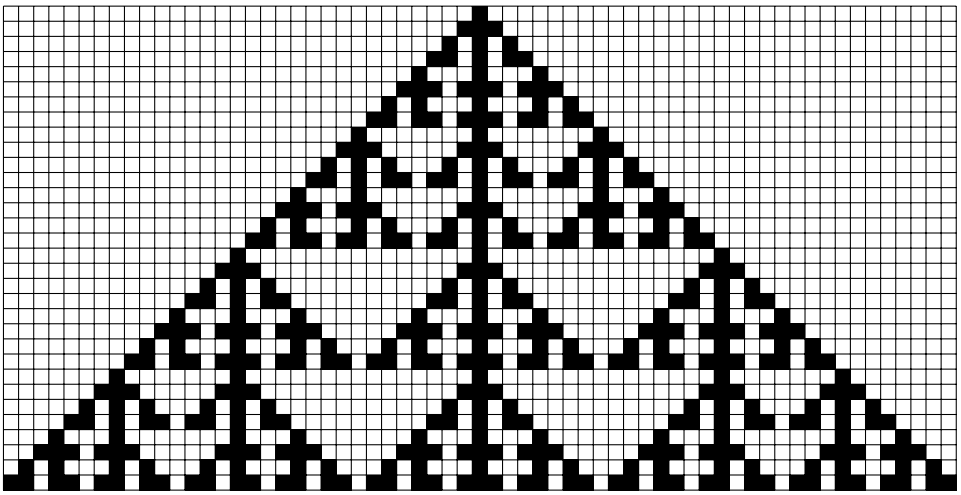


Figura 2.2: Prime 32 righe del diagramma spazio tempo dell'automa cellulare elementare con regola 150 e configurazione iniziale uguale a 0 ovunque tranne in una cellula.

(W1) *l'evoluzione di una qualunque configurazione iniziale porta tutte le cellule ad avere lo stesso stato (figura 2.1a)*

(W2) *l'evoluzione di una qualunque configurazione iniziale porta a strutture semplici, separate e stabili o che si ripetono periodicamente (figura 2.1b)*

(W3) *l'evoluzione di una qualunque configurazione iniziale porta a pattern caotici (figura 2.1c)*

(W4) *l'evoluzione di alcune configurazioni iniziali produce strutture localizzate con interazioni complesse (figura 2.1d)*

2.3 Termini e nozioni di base

Nel resto del capitolo definiamo alcuni termini e concetti basilari dello studio degli automi cellulari, e presentiamo i risultati necessari per le successive trattazioni.

Se consideriamo c come *configurazione iniziale*, possiamo considerare la sequenza di configurazioni dell'automa cellulare ottenute applicando ripetutamente G a c come la sequenza di passi di computazione di un modello di calcolo che abbia ricevuto c come dato in ingresso. Data una configurazione iniziale c si definisce *orbita* di c la sequenza

$$orb(c) = c, G(c), G^2(c), G^3(c), \ldots$$

Poiché l'aggiornamento dello stato delle cellule avviene a intervalli di tempo discreti e costanti, dato $t \in \mathbb{N}$ chiameremo $G^t(c)$ *stato della configurazione c al tempo t*.

A volte sarà utile considerare orbite bidirezionali (infinite) ovvero tali che per ogni configurazione ne esiste una precedente oltre che una successiva.

$$orb_b(c) = \ldots, c_{-2}, c_{-1}, c, c_1, c_2, \ldots$$

dove

$$\forall i \in \mathbb{Z}.G(c_i) = c_{i+1}$$

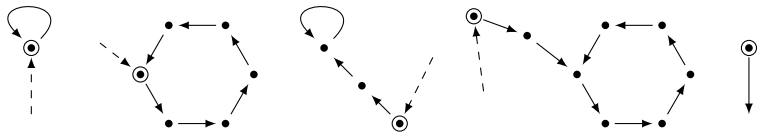
Questo approccio è importante nello studio degli automi cellulari come sistema dinamico. Si noti che in questa rappresentazione non ha senso parlare di configurazioni iniziali. È importante osservare che, data una configurazione c, la sua orbita unidirezionale è stabilita in modo univoco da G, mentre la sua orbita bidirezionale può non essere unica. Si pensi all'automa cellulare elementare con numero di Wolfram 110 dell'esempio 1. In questo caso esistono due configurazioni tali che la loro immagine è la configurazione in cui tutte le cellule sono nello stato 0: sé stessa e la configurazione in cui tutte le cellule sono nello stato 1.

Un modo spesso utile di rappresentare l'orbita di un automa cellulare è attraverso un *diagramma spazio-tempo*, ovvero un grafico di dimensione $d + 1$ nel quale d dimensioni servono a rappresentare le configurazioni dell'automa e in cui la dimensione aggiuntiva rappresenta la variabile tempo[2]. Il diagramma spazio-tempo relativo all'orbita unidirezionale di un automa cellulare unidimensionale è di gran lunga il caso più diffuso, e viene rappresentato come una sequenza di linee, la prima relativa alla configurazione iniziale, la seconda relativa alla configurazione al tempo $t = 2$ e così via (figura 2.2).

Talvolta si usa rappresentare le possibili configurazioni di un automa cellulare come un grafo orientato infinito, detto *spazio delle fasi*. In questo caso ad ogni nodo corrisponde una configurazione differente e da ogni nodo relativo ad una configurazione c esiste un unico arco uscente che lo collega al nodo che rappresenta la configurazione $G(c)$. Non è generalmente possibile trattare lo spazio delle fasi di un automa cellulare in modo diretto se non in casi molto particolari[3], per questo si preferisce in genere studiare lo spazio delle possibili configurazioni da un punto di vista topologico [4].

[2]Per orbite unidirezionali ($t \in \mathbb{N}$) i diagrammi spazio-tempo sono elementi di $S^{\mathbb{Z}^d \times \mathbb{N}}$, per orbite bidirezionali ($t \in \mathbb{Z}$) sono invece elementi di $S^{\mathbb{Z}^d \times \mathbb{Z}}$.

[3]Ad esempio lo spazio delle fasi di un automa cellulare lineare ristretto alle sole configurazioni spazialmente periodiche può essere descritto completamente [16].



(a) (b) (c) (d) (e)

Figura 2.3: Porzioni di *spazi delle fasi*, i nodi *cerchiati* in ogni figura rappresentano configurazioni rispettivamente: *punto fisso* (a), *periodica* (b), *infine fissa* (c), *infine periodica* (d) e *giardino dell'Eden* (e). Le frecce solide sono gli archi necessari perché la definizione sia soddisfatta. Quelli tratteggiati che puntano a nodi cerchiati corrispondono ad archi che possono essere presenti.

Esistono alcune classi di configurazioni che presentano un comportamento particolare rispetto alla funzione di transizione globale e che corrispondono a forme particolari nello spazio delle fasi. Una caratterizzazione equivalente a quella che daremo può essere data in base alle orbite di tali configurazioni.

Definizione 4. *Una configurazione c di un automa cellulare G si dice :*

punto fisso (di G) *se $G(c) = c$ (figura 2.3a);*

temporalmente periodica *se esiste un* periodo $p \in \mathbb{Z}_+$ *tale che $G^p(c) = c$ (figura 2.3b);*

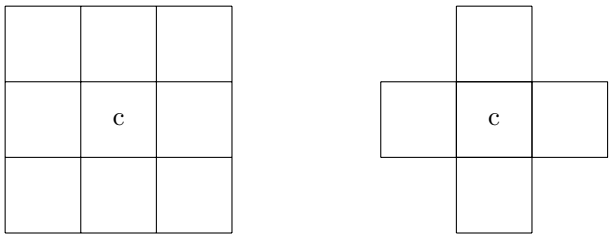
infine fissa *se esiste un $n \in \mathbb{N}$ tale che $G^{n+1}(c) = G^n(c)$, ovvero tale che G^n sia un* punto fisso *(figura 2.3c);*

infine temporalmente periodica *se esistono un $n \in \mathbb{N}$ e un $p \in \mathbb{Z}_+$ tali che $G^{n+p}(c) = G^n(c)$, ovvero tale che G^n sia* temporalmente periodica *di* periodo p *(figura 2.3d);*

giardino dell'Eden *se non esiste una* configurazione c' *tale che $G(c') = c$, ovvero se $\{c\}$ ha come controimmagine l'insieme vuoto (figura 2.3e).*

Enunciamo adesso alcuni risultati di immediata verifica riguardanti queste categorie di configurazioni. L'orbita unidirezionale di un *punto fisso* contiene solamente una configurazione. Una delle orbite bidirezionali di un *punto fisso* contiene solamente una configurazione. Ogni configurazione contenuta nell'orbita di una configurazione *temporalmente periodica* di *periodo p* è una *configurazione temporalmente periodica* di *periodo p* a sua volta. L'orbita di una configurazione *temporalmente periodica*, *infine fissa* o *infine temporalmente periodica* contiene solo un numero finito di configurazioni differenti. Un *giardino dell'Eden* non può appartenere a nessuna orbita bidirezionale e l'unica orbita unidirezionale a cui appartiene è la propria.

Alcuni indici di vicinato sono particolarmente frequenti nella descrizione di automi cellulari. I più diffusi sono il *vicinato* di *Moore* e quello di *Von Neumann*.



(a) (b)

Figura 2.4: I vicinati (a) M_1^2 e (b) V_1^2 e di una generica cellula c.

Definizione 5. *In un automa cellulare di dimensione d, l'*indice di vicinato *di* Moore *di raggio* $r \in \mathbb{Z}$ *(*M_r^d*) e l'*indice di vicinato *di* Von Neumann *di raggio r (*V_r^d*) sono definiti come segue:*

$$M_r^d = \{\vec{k} \in \mathbb{Z}^n \mid \|\vec{k}\|_\infty \leq r\}$$

$$V_r^d = \{\vec{k} \in \mathbb{Z}^n \mid \|\vec{k}\|_1 \leq r\}$$

I vicinati di *Von Neumann* e *Moore* per $d = 2$ ed $r = 1$ sono riportati nella figura 2.4. Si noti che l'automa cellulare *game-of-life* di Conway adotta il *vicinato di Moore* con raggio uno.

Nel caso particolare degli automi cellulari unidimensionali possiamo osservare che i vicinati di Von Neumann e di Moore di raggio uguale sono identici, pertanto si usa chiamarli semplicemente *indici di vicinato di raggio r*. Un caso particolare è l'*indice di vicinato di raggio* $\frac{1}{2}$ definito come $(0, 1)$. Un automa cellulare con indice di vicinato di raggio $\frac{1}{2}$ è detto *unidirezionale*. Gli automi cellulari elementari hanno tutti vicinato di raggio 1 per definizione; tuttavia è opportuno specificare che per alcuni di questi possono essere definiti automi cellulari *unidirezionali* equivalenti.

2.4 Configurazioni finite e periodiche

Dedichiamo questa sezione allo studio di due classi di configurazioni molto importanti, entrambe legate in qualche modo a vincoli che consentano rappresentazioni finite dello stato di un automa cellulare.

Talvolta uno stato q tale che $f(q, q, \ldots, q) = q$ viene chiamato *stato quiescente* dell'automa cellulare. Notiamo innanzitutto che un tale stato può esistere o meno a seconda della regola di aggiornamento locale dell'automa cellulare. Inoltre più stati possono verificare la condizione di quiescenza, in questo caso la scelta dello stato quiescente è arbitraria, ma una volta effettuata, lo stato quiescente è unico. Una configurazione Q viene detta *quiescente* se mappa tutte le cellule nello stesso stato quiescente q. È immediato verificare che una confi-

gurazione quiescente è sempre un punto fisso di G. Un automa cellulare si dice *nilpotente* se ogni configurazione iniziale evolve nello stato quiescente, ovvero se vale

$$\forall c \in S^{\mathbb{Z}^d} \,.\, \exists n \in \mathbb{N} \,.\, G^n(c) = Q$$

Sia $s \in S$, definiamo *s-supporto* di una configurazione $c \in S^{\mathbb{Z}^d}$ l'insieme delle cellule di c che non sono nello stato s. Una configurazione c si dice *s-finita* se $supp_s(c)$ è un insieme finito. Se q è lo stato quiescente dell'automa cellulare, allora le configurazioni *q-finite* sono dette semplicemente *finite*.

Definizione 6. *(configurazioni finite). Sia (d, S, N, f) un automa cellulare con stato* quiescente *$q \in S$ tale che $f(q, q, \ldots, q) = q$. Diremo che una configurazione $c \in S^{\mathbb{Z}^d}$ è* finita *se e solo se l'insieme* supporto

$$supp_q(c) = \{\vec{n} \in \mathbb{Z}^d \mid c(\vec{n}) \neq q\}$$

è un insieme finito.

Denotiamo con $\mathcal{F} \subset S^{\mathbb{Z}^d}$ l'insieme delle configurazioni finite di un automa cellulare di dimensione d e stati possibili S. È importante notare che $\mathcal{F}$ ha cardinalità numerabile, e che essendo q è lo stato quiescente $c \in \mathcal{F} \implies G(c) \in \mathcal{F}$. È sensato allora considerare la restrizione di G sulle sole configurazioni finite $G_{\mathcal{F}} : \mathcal{F} \longrightarrow \mathcal{F}$.

Sia $\vec{r} \in \mathbb{Z}^d$. Assumendo fissato S, definiamo la *traslazione* determinata da $\vec{r}$ $\tau_{\vec{r}}$ come la funzione di transizione globale dell'automa cellulare il cui indice di vicinato contiene unicamente $\vec{r}$ e la cui regola di aggiornamento locale è la funzione identità. Possiamo dire in altre parole che $\tau_{\vec{r}} : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ è la funzione che mappa ogni $c \in S^{\mathbb{Z}^d}$ in $c' \in S^{\mathbb{Z}^d}$ tale che

$$\forall \vec{n} \in \mathbb{Z}^d.\ c'(\vec{n}) = c(\vec{n} + \vec{r})$$

Possiamo immediatamente enunciare il seguente risultato [14].

Teorema 2. *Sia G la funzione di transizione globale di un automa cellulare e τ una funzione* traslazione*. Le funzioni G e τ commutano rispetto alla composizione funzionale, ovvero $G \circ \tau = \tau \circ G$.*

Una configurazione di un automa cellulare c su uno spazio di dimensione d si dice *spazialmente periodica* se esistono d elementi di $\mathbb{Z}^d$ linearmente indipendenti tali che la *traslazione* da loro determinata mappa c in c.

Definizione 7. *(configurazione spazialmente periodica). Dato una automa cellulare su spazio cellulare di dimensione d e con insieme degli stati S. Una sua*

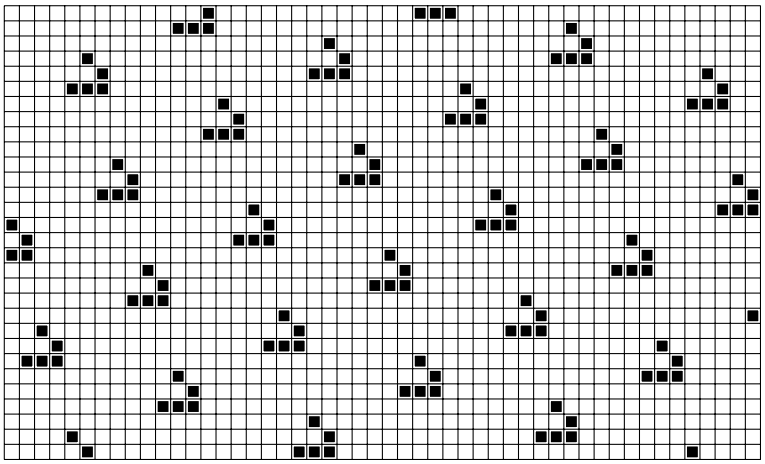


Figura 2.5: Una configurazione periodica bidimensionale secondo $\vec{r}_1 = (2, 7)$ e $\vec{r}_2 = (9, 3)$.

configurazione $c \in S^{\mathbb{Z}^d}$ *si dice* spazialmente periodica *se esistono* d *elementi di* $\mathbb{Z}^d$ *linearmente indipendenti* $\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_d \in \mathbb{Z}^d$ *tali che*

$$\forall \vec{n} \in \mathbb{Z}^d.\ c(\vec{n}) = c(\vec{n} + \vec{r}_i)$$

Dal punto di vista grafico, una configurazione spazialmente periodica può essere vista come un ipercubo che si ripete periodicamente in ognuna delle d dimensioni dello spazio (vedi figura 2.5). Denotiamo con $\mathcal{P} \subset S^{\mathbb{Z}^d}$ l'insieme delle configurazioni spazialmente periodiche di un automa cellulare di dimensione d e stati possibili S. Notiamo che anche in questo caso ci siamo ridotti a considerare solo un sottoinsieme numerabile di $S^{\mathbb{Z}^d}$. Inoltre dal teorema 2 possiamo dedurre che se la configurazione c è periodica per $\vec{r}_i \in \mathbb{Z}^d$, ovvero per ogni $\vec{n} \in \mathbb{Z}^d$ vale $c(\vec{n}) = c(\vec{n} + \vec{r}_i)$, allora

$$G(c) + \vec{r}_i = (\tau_{\vec{r}_i} \circ G)(c) = (G \circ \tau_{\vec{r}_i})(c) = G(c + \vec{r}_i) = G(c)$$

È immediato quindi ricavare che se c è spazialmente periodica allora anche $G(c)$ è spazialmente periodica.

Ha senso dunque considerare la restrizione della funzione G sulle sole configurazioni periodiche $G_{\mathcal{P}} : \mathcal{P} \longrightarrow \mathcal{P}$.

Le configurazioni finite e quelle periodiche sono usate nella simulazione di automi cellulari su computer. La restrizione alle sole configurazioni periodiche nella simulazione effettiva di un automa cellulare prende generalmente il nome di *condizioni periodiche di confine* su un array d-dimensionale finito di cellule. L'evoluzione di una configurazione periodica su spazio cellulare infinito può infatti essere perfettamente simulata su uno spazio finito formato da un ipercubo

nel quale le facce opposte sono "attaccate"[4].

Dato un insieme di stati S, e una dimensione d si può notare che per ogni configurazione spazialmente periodica c, il numero di configurazioni che sono spazialmente periodiche secondo lo stesso insieme di elementi $\vec{r_i}$ di c è limitato. In pratica sono tante quante le possibili assegnazioni di stato alle cellule contenute in uno degli ipercubi di cui è composta c. Data una qualunque configurazione spazialmente periodica sappiamo anche che la sua orbita conterrà unicamente configurazioni spazialmente periodiche secondo lo stesso insieme di elementi $\vec{r_i}$ e poiché queste sono finite, prima o poi una configurazione si ripeterà. Perciò una *configurazione spazialmente periodica* è sempre anche *infine temporalmente periodica*[5].

[4]Ovvero su una topologia quoziente definita a partire dalla relazione di equivalenza che lega due punti interni solo se sono uguali e due punti sulle facce dell'ipercubo se sono uno l'opposto dell'altro.

[5]Il viceversa non è vero, basti pensare ad una configurazione di *game-of-life* che contiene unicamente un oscillatore.

Capitolo 3

Alcune proprietà

3.1 Iniettività e suriettività

Trattiamo adesso alcune proprietà che caratterizzano classi di automi cellulari. Le definizioni di iniettività, suriettività e biettività per gli automi cellulari fanno riferimento alla *funzione di transizione globale* dell'automa cellulare.

Definizione 8. *Un automa cellulare A con funzione di transizione globale $G[A] : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ si dice:*

iniettivo *se $G[A]$ è una funzione iniettiva;*

suriettivo *se $G[A]$ è una funzione suriettiva;*

biettivo *se $G[A]$ è una funzione biettiva.*

Nel caso della suriettività possiamo notare che una configurazione ha come controimmagine l'insieme vuoto se e solo se è un giardino dell'Eden. Pertanto un automa cellulare *suriettivo* può essere equivalentemente definito come un automa cellulare che non ammette configurazioni *giardini-dell'Eden*.

Per parlare di iniettività e suriettività ci farà comodo definire formalmente il concetto di *pattern*. Intuitivamente sono pattern quelli che abbiamo presentato nell'introduzione per parlare delle classi di oggetti che si possono trovare in una configurazione dell'automa cellulare *game-of-life*. Dato un automa cellulare $A = (d, S, N, f)$ e un *dominio* $D \subseteq \mathbb{Z}^d$, diciamo che un *pattern* su A con *dominio* D è una configurazione parziale $p \in S^D$ che assegna uno stato ad ogni cellula del *dominio*. Un pattern si dice *finito* se il suo dominio è *finito*.

Dato un pattern, lo stato successivo delle cellule del suo dominio dipenderà solo dallo stato di una parte delle cellule dello spazio cellulare. L'insieme di

cellule che possono influenzare lo stato successivo delle cellule di un dominio D è detto vicinato del dominio D e indicato come $N(D)$, dove $N = \{\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m\}$ è l'indice di vicinato dell'automa cellulare, e contiene tutte le cellule che sono vicini di un elemento qualsiasi di D.

$$N(D) = \{\vec{x} + \vec{n}_i \mid \vec{x} \in D \wedge \vec{n}_i \in N\}$$

Risulta abbastanza naturale pensare allora ad una restrizione della funzione di aggiornamento globale che definisce unicamente l'evoluzione di un pattern su un dominio D. Tale riduzione ha come dominio le configurazioni su $N(D)$ e associa ad ognuna di esse la configurazione successiva su D.

$$G^{[N(D)\to D]} : S^{N(D)} \longrightarrow S^D$$

Un pattern finito p su dominio D si dice *orfano* se non esiste un pattern q su dominio $N(D)$ tale che $G^{[N(D)\to D]}(q) = p$. Il teorema seguente mette in relazione la presenza di pattern orfani in un automa cellulare con la suriettività [12].

Teorema 3. *Un automa cellulare ammette un* orfano *se e solo se non è* suriettivo.

Si noti che la parte interessante del teorema è che ogni automa cellulare non *suriettivo* ammette almeno un *orfano*. Il viceversa è banale in quanto ogni configurazione di S^{Z^d} che contiene un orfano è un elemento la cui controimmagine è vuota (è un giardino dell'Eden). Un modo intuitivo di vedere il risultato precedente è che ogni configurazione giardino dell'Eden contiene un pattern finito orfano.

Il seguente teorema, noto con il nome di *teorema del bilanciamento* afferma che in un automa cellulare *suriettivo* il numero di *pattern* differenti contenuti nella controimmagine di un *pattern finito* deve dipendere solo dalla dimensione del *dominio*, e non dal pattern stesso ovvero dagli stati delle cellule.

Il teorema è stato dimostrato prima per i soli spazi cellulari unidimensionali da Hedlund [9] e successivamente per spazi cellulari di dimensione qualunque da Marouka e Kimura [17].

Teorema 4. *Sia $A = (d, s, N, f)$ un automa cellulare suriettivo, e sia $D \in \mathbb{Z}^d$ un dominio finito. Allora, per ogni pattern $p \in S^D$ il numero di pattern $q \in S^{N(D)}$ tali che*

$$G^{[N(D)\to D]}(q) = p$$

è $|S|^{|N(D)|-|D|}$.

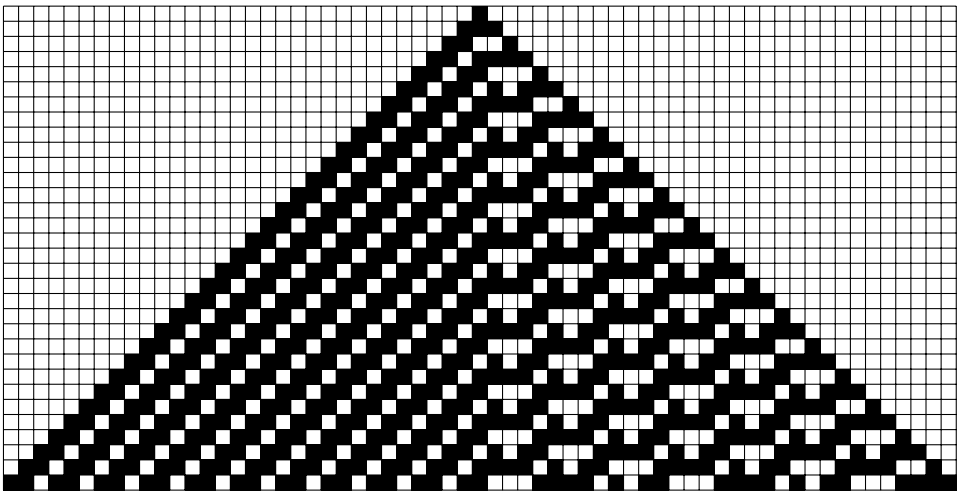


Figura 3.1: Prime 32 righe del diagramma spazio tempo dell'automa cellulare elementare con regola 62 e configurazione iniziale uguale a 0 ovunque tranne in una cellula.

Questo vuol dire che dato un dominio finito $D \subseteq \mathbb{Z}^d$, il numero di configurazioni parziali di $N(D)$ che portano a due configurazioni parziali di D qualunque deve essere lo stesso, altrimenti l'automa cellulare non può essere suriettivo. Come caso particolare, se si considera un dominio composto da un'unica cellula, otteniamo che un automa cellulare suriettivo deve avere una *regola di aggiornamento locale bilanciata*.

Esempio 2. *L'automa cellulare elementare con numero di Wolfram 62 (vedi figura 3.1) non è suriettivo. La regola di aggiornamento locale è la seguente:*

$$f(1,1,1) = f(1,1,0) = f(0,0,0) = 0$$
$$f(1,0,1) = f(1,0,0) = f(0,1,1) = f(0,1,0) = f(0,0,1) = 1$$

Per dimostrare che esistono giardini dell'Eden basta osservare che il numero di configurazioni del vicinato di una cellula che corrispondono ad una cellula in stato 1 *nella prossima generazione è maggiore dal numero di quelle che corrispondono ad una in stato* 0 *(5 contro 3).*

Due configurazioni c e d si dicono *asintotiche* se differiscono solo in un numero finito di cellule, ovvero se

$$diff(c,d) = \{\vec{x} \in \mathbb{Z}^d | c(\vec{x}) \neq d(\vec{x})\}$$

è un insieme finito. Un automa cellulare G è detto *preiniettivo* se $G(c) \neq G(d)$ per ogni coppia di configurazioni asintotiche c, d tali che $c \neq d$. Essendo le due configurazioni c e d diverse solo su un dominio finito risulta abbastanza semplice

constatare che un automa cellulare è *preiniettivo* se e solo se è *iniettivo sulle configurazioni finite*.

Moore ha dimostrato che un rapporto di dipendenza lega la preiniettività alla suriettività di un automa cellulare [18].

Teorema 5. *(Moore). Dato una automa cellulare G, se G è suriettivo allora G è preiniettivo.*

Sfruttando il teorema 3 possiamo affermare che se un automa cellulare non è suriettivo allora esiste una configurazione finita giardino dell'Eden. Da questo, grazie a considerazioni combinatorie è possibile dimostrare che vale anche l'inverso del teorema di Moore [19].

Teorema 6. *(Myhill). Dato una automa cellulare G, se G è preiniettivo allora G è suriettivo.*

In letteratura è spesso comune riferirsi al risultato complessivo come *garden-of-Eden theorem* [4][14][13][12].

Teorema 7. *(garden-of-Eden). Dato una automa cellulare G, G è preiniettivo se e solo se G è suriettivo.*

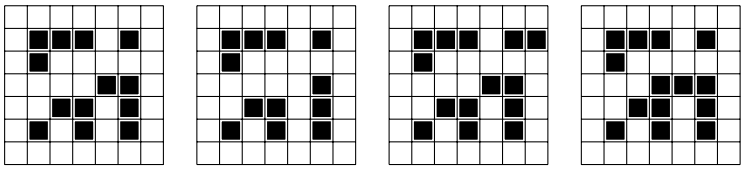
Notiamo che una conseguenza del teorema è che un automa cellulare *iniettivo* è anche suriettivo, e quindi *biettivo*. È infatti banale verificare che un automa cellulare iniettivo è anche preiniettivo.

Le relazioni fra iniettività e suriettività per configurazioni periodiche e finite sono ampiamente studiate, e i risultati sono spesso differenti nel caso degli automi cellulari di dimensione uno e in quello degli automi cellulari di dimensione maggiore [6]. Riportiamo qui solo alcuni risultati validi per tutti gli automi cellulari, indipendentemente dalla dimensione dello spazio cellulare.

- Se G è *iniettiva* allora anche $G_{\mathcal{F}}$ e $G_{\mathcal{P}}$ sono *iniettive*.
- Se $G_{\mathcal{F}}$ o $G_{\mathcal{P}}$ è *suriettiva* allora lo è anche G.
- Se $G_{\mathcal{P}}$ è *iniettiva* allora $G_{\mathcal{P}}$ è anche *suriettiva*.
- Se G è *iniettiva* allora $G_{\mathcal{F}}$ è *suriettiva*.

3.2 Automi cellulari come sistemi dinamici

Come abbiamo già accennato, un modo molto proficuo di studiare l'insieme delle configurazioni di un automa cellulare $A = (d, S, N, f)$ è attraverso una caratterizzazione topologica dell'insieme $S^{\mathbb{Z}^d}$. Da questo approccio deriva anche



(a) (b) (c) (d)

Figura 3.2: Quattro configurazioni differenti di *game-of-life*, la cellula $\vec{0}$ è al centro della griglia.

un metodo di studio del comportamento degli automi cellulari, alternativo a quello che abbiamo applicato finora, che li considera come sistemi dinamici (topologici) discreti.

Per prima cosa definiamo una metrica sullo spazio delle configurazioni $S^{\mathbb{Z}^d}$. La *distanza* fra due configurazioni c ed e è definita come $d(c, e)$.

$$d(c,e) = \begin{cases} 0 & se\ e = c \\ 2^{-min\{\|\vec{x}\|_\infty |\ c(\vec{x}) \neq e(\vec{x})\}} & se\ e \neq c \end{cases}$$

È semplice verificare che d è una metrica su $S^{\mathbb{Z}^d}$. In effetti si tratta di una ultrametrica[1], e la topologia indotta da essa è la topologia di Cantor. Intuitivamente due configurazioni sono tanto più vicine quanto più grande è l'ipercubo attorno a $\vec{0} \in \mathbb{Z}^d$ sul quale associano lo stesso valore ad ogni cellula. La distanza massima fra due configurazioni è 1, la distanza minima fra due configurazioni diverse non è definita.

Esempio 3. *Si prendano in considerazione le configurazioni di* game-of-life *in figura 3.2, dove si assume che la cellula $\vec{0}$ sia quella al centro della griglia*[2]*. Le distanze relative alle configurazioni c_a in figura 3.2a, c_b in figura 3.2b, c_c in figura 3.2c e d_d in figura 3.2d sono le seguenti:*

$$\begin{aligned} d(c_a, c_b) &= 2^{-1} = \frac{1}{2} \\ d(c_a, c_c) &= 2^{-3} = \frac{1}{8} \\ d(c_a, c_d) &= 2^0 = 1 \\ d(c_b, c_c) &= 2^{-min\{1,3\}} = \frac{1}{2} \\ d(c_c, c_d) &= 2^0 = 1 \end{aligned}$$

[1] Ovvero vale una forma più forte della disuguaglianza triangolare: $\forall x, y, x.\ d(x, y) \leq max\{d(x, z), d(y, z)\}$.

[2] Il numero di cellule vive nella configurazione di *game-of-life* in figura 3.2a cresce indefinitamente nel tempo.

Una base utile per la topologia è l'insieme dei *cilindri*. Dato un dominio finito $D \subseteq \mathbb{Z}^d$ e una configurazione $c \in S^{\mathbb{Z}^d}$, definiamo *cilindro* determinato da c e D ($Cyl(c, D)$) l'insieme di tutte le configurazioni le cui cellule sono nello stesso stato di quelle di c sul dominio D.

$$Cyl(c, D) = \{e \in S^{\mathbb{Z}^d} \mid \forall \vec{x} \in D.\ e(\vec{x}) = c(\vec{x})\}$$

Si noti che tutte le palle aperte $B_\varepsilon(c)$ sono cilindri, e che ogni cilindro è esprimibile come unione di palle aperte:

$$\begin{aligned} B_\varepsilon(c) &= \{e \in S^{\mathbb{Z}^d} \mid d(c, e) < \varepsilon\} \\ &= \{e \in S^{\mathbb{Z}^d} \mid \forall \vec{x}.\ \|\vec{x}\|_\infty \leq r \implies e(\vec{x}) = c(\vec{x})\} \end{aligned}$$

$$Cyl(c, D) = \bigcup_{e \in Cyl(c,D)} B_\varepsilon(e)$$

dove $\varepsilon = 2^{-r}$ e per r sufficientemente grande (tale che $\|\vec{x}\|_\infty \leq r$ per ogni $\vec{x} \in D$).

Di conseguenza i cilindri sono una base della topologia. Poiché i cilindri con la stessa base formano una partizione finita di $S^{\mathbb{Z}^d}$, ognuno di essi può essere rappresentato come unione del complementare degli altri. Quindi i cilindri, in quanto unione di finiti complementari di aperti, sono chiusi (oltre che aperti). Si può mostrare come gli insiemi di configurazioni che sono sia aperti che chiusi sono tutti e soli quelli ottenuti da unioni finite di cilindri. È importante osservare che l'insieme dei cilindri di uno spazio delle configurazioni è numerabile. Mettendo assieme quanto detto fin'ora abbiamo che i *cilindri* formano una base numerabile di insiemi per $S^{\mathbb{Z}^d}$ che sono sia aperti, sia chiusi.

Sotto questa topologia, possiamo dire che una sequenza di configurazioni $c_1, c_2, \ldots$ converge a $c \in S^{\mathbb{Z}^d}$ se e solo se per ogni $\vec{x} \in \mathbb{Z}^d$ esiste un $k \in \mathbb{N}$ tale che $c_i(\vec{x}) = c(\vec{x})$ per ogni $i \geq k$. In altre parole se lo stato di ogni cellula si stabilizza da un certo punto in poi della sequenza.

È possibile dimostrare che ogni sequenza di configurazioni ha una *sottosequenza convergente* [14]. Dunque $S^{\mathbb{Z}^d}$ è uno spazio compatto.

Un altro risultato che caratterizza $S^{\mathbb{Z}^d}$ riguarda le configurazioni spazialmente periodiche e finite. Mostrando che per ogni configurazione $c \in S^{\mathbb{Z}^d}$ esistono una sequenza di configurazioni in $\mathcal{P}$ e una sequenza di configurazioni in $\mathcal{F}$ tali che queste convergono a c, possiamo derivare che sia l'insieme delle configurazioni spazialmente periodiche, sia quello delle configurazioni finite sono densi in $S^{\mathbb{Z}^d}$ [14].

Anche le funzioni di transizione globali godono di importanti proprietà dal punto

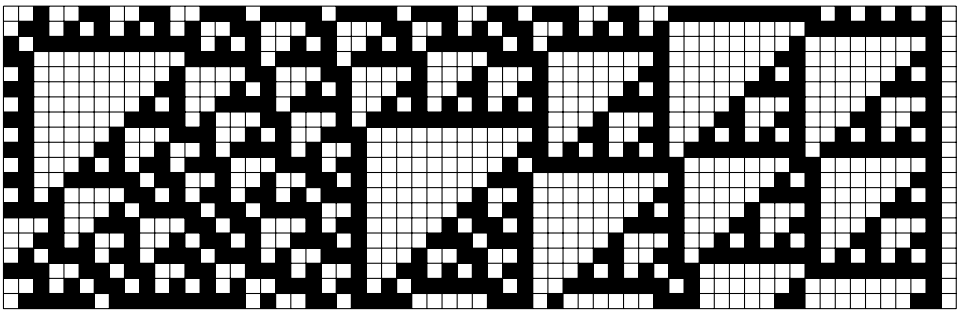


Figura 3.3: Prime 20 righe del diagramma spazio tempo dell'automa cellulare elementare con regola 102 e configurazione iniziale casuale.

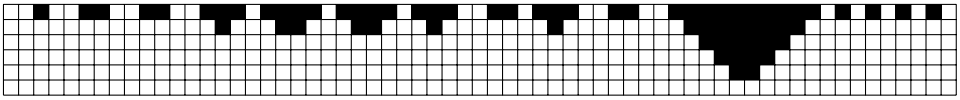


Figura 3.4: Prime 6 righe del diagramma spazio tempo dell'automa cellulare elementare con regola 128 e configurazione iniziale casuale.

di vista topologico. Data una funzione di transizione globale $G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ e una sequenza di configurazioni $c_1, c_2, c_3, \ldots$ convergente si dimostra facilmente che anche la sequenza $G(c_1), G(c_2), G(c_3), \ldots$ converge e in particolare che G è continua.

$$\lim_{i\to\infty} G(c_i) = G(\lim_{i\to\infty} c_i)$$

Intuitivamente la continuità di G riflette la località della regola di aggiornamento. Alla stessa maniera il fatto che G commuti con le traslazioni (vedi 2) impone che la regola di aggiornamento sia uguale per tutte le cellule. A livello intuitivo queste due proprietà definiscono completamente gli automi cellulari, perciò non risulta inatteso il seguente importantissimo risultato [9].

Teorema 8. *(Curtis-Hedlund-Lyndon). Dato un insieme finito S e un intero positivo $d \in \mathbb{Z}_+$. Una funzione $G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$ è la funzione di aggiornamento globale di un automa cellulare se e solo se è continua e commuta con le traslazioni su $\mathbb{Z}^d$.*

Coppie (X, F) dove X è uno spazio compatto ed $F : X \longrightarrow X$ è continua sono comunemente chiamati sistemi dinamici (topologici) [13]. Quando si vogliono studiare gli automi cellulari come sistemi dinamici, possiamo trascurare le configurazioni che non possono presentarsi da una certa generazione in poi. Non consideriamo interessanti le configurazioni giardino dell'Eden per G, per G^2 e così via. La parte rilevante dell'insieme delle configurazioni viene chiamato *insieme limite*.

Definizione 9. *Dato un automa cellulare (d, S, N, f) con funzione di aggior-*

namento globale G, il suo insieme limite è

$$\Omega_G = \bigcap_{n \in \mathbb{N}} G^n(S^{\mathbb{Z}^d})$$

Esempio 4. *Si consideri l'automa cellulare elementare numero* 102 *(vedi figura 3.3), anche noto come* xor. *La sua regola di aggiornamento locale è esprimibile come $f(s_{-1}, s_0, s_1) = s_0 \oplus s_1$, dove $\oplus$ è il simbolo che rappresenta la disgiunzione esclusiva. In quanto suriettivo, il suo insieme limite contiene tutte le configurazioni in $\{0,1\}^{\mathbb{Z}}$.*

Esempio 5. *Si consideri l'automa cellulare elementare numero* 128 *(vedi figura 3.4). La sua regola di aggiornamento locale è $f(s_{-1}, s_0, s_1) = s_{-1} \times s_0 \times s_1$. L'unico pattern che viene mappato in* 1 *è* 111*, l'automa cellulare non è sicuramente suriettivo. Notiamo che ogni configurazione finita prima o poi diventa la configurazione quiescente, ma che l'automa cellulare non è nilpotente, in quanto la configurazione in cui tutte le cellule sono nello stato* 1 *è un punto fisso.*

È immediato verificare il pattern $100\ldots001$ *è un orfano di G^n per n sufficientemente grande, pertanto in tutte le configurazioni che non sono giardini dell'Eden per un qualche G^n le cellule nello stato* 1 *devono formare un segmento contiguo. Si può dimostrare che l'insieme limite di questa regola contiene tutte le configurazioni della forma*

$$\ldots 111111 \ldots$$
$$\ldots 000000 \ldots$$
$$\ldots 111000 \ldots$$
$$\ldots 00111 \ldots$$
$$\ldots 000111 \ldots 111000 \ldots$$

In questo caso l'insieme limite è numerabile.

3.3 Invertibilità e leggi di conservazione

Gli automi cellulari possiedono molte proprietà comunemente associate al mondo fisico: sono massicciamente paralleli, omogenei e hanno solo interazioni locali. Se un automa cellulare viene visto come rappresentazione di fenomeni naturali ci si aspetterebbe che rispetti le stesse proprietà del sistema di riferimento. Questo non è sempre soddisfatto automaticamente. Tuttavia è possibile aggiungere delle restrizioni sulle regole di aggiornamento affinché gli automi cellulari rispettino altre proprietà di sistemi fisici o biologici.

Storicamente queste restrizioni hanno avuto grande riscontro nella simulazione di fenomeni naturali, si prendano come esempio i *lattice gas automata*: una classe di automi cellulari per la simulazione di urti di particelle che conservano massa e momento angolare e che si sono rivelati molto utili nella fluidodinamica. Ma la natura fisica degli automi cellulari è di grande rilevanza anche nel verso opposto. Dato che molti automi cellulari sono computazionalmente universali (vedi capitolo 4), alcuni anche molto semplici, se riuscissimo a sfruttare un fenomeno fisico microscopico per implementare uno di questi automi cellulari avremmo a disposizione uno strumento di calcolo massicciamente parallelo e Turing equivalente. Perché ciò sia possibile, ovviamente, è necessario che l'automa cellulare di cui cerchiamo un'istanza nel mondo fisico ubbidisca alle leggi della fisica.

In questa rassegna ci concentreremo in particolare sull'invertibilità, che gioca un ruolo importante in quanto molti dei sistemi della fisica microscopica sono *invertibili*. Accenneremo anche qualcosa per quanto riguarda le leggi di conservazione di quantità arbitrarie.

Come molte altre proprietà, anche l'invertibilità di un automa cellulare si definisce a partire dalla funzione di aggiornamento globale.

Definizione 10. *(automa cellulare invertibile). Dato un automa cellulare con funzione di aggiornamento globale G, se esiste un automa cellulare con funzione di aggiornamento globale G^{-1} tale che*

$$G \circ G^{-1} = G^{-1} \circ G = id$$

allora i due automi cellulari sono detti invertibili e l'uno è detto inverso dall'altro.

Sappiamo già che ogni funzione G biettiva è invertibile, ma dobbiamo verificare che l'inversa sia effettivamente la funzione di aggiornamento globale di un qualche automa cellulare. Sorprendentemente questo è sempre vero. Ovviamente ogni automa cellulare invertibile è anche biettivo. Viceversa supponiamo che G sia la funzione di aggiornamento globale di un automa cellulare biettivo. In quanto biettiva, G è invertibile: sia G^{-1} la sua funzione inversa.

$$\forall c \in S^{\mathbb{Z}^d} \ . \ (G^{-1} \circ G)(c) = (G \circ G^{-1})(c)$$

Poiché lo spazio delle configurazioni è compatto, sappiamo che l'inversa di una funzione continua e biettiva è una funzione continua a sua volta. Quindi G^{-1} è

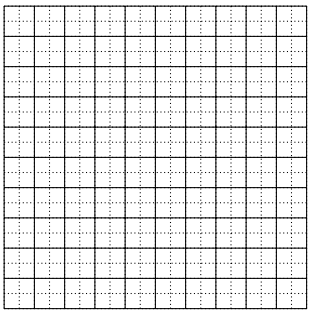


Figura 3.5: *Vicinato di Margolus*: la partizione delle cellule definita dalle linee tratteggiate si usa nella prima fase dell'aggiornamento, quella definita dalle linee continue nella seconda.

continua, ma è anche vero che commuta con ogni traslazione

$$\tau \circ G^{-1} = G^{-1} \circ G \circ \tau \circ G^{-1} = G^{-1} \circ \tau \circ G \circ G^{-1} = G^{-1} \circ \tau$$

quindi per il teorema 8 è la funzione di aggiornamento globale di un automa cellulare. Pertanto l'automa cellulare è invertibile [9].

Teorema 9. *Un automa cellulare è invertibile se e solo se è biettivo.*

Ricordiamo che per il teorema 7 un automa cellulare è biettivo se e solo se è iniettivo. Quindi per gli automi cellulari *l'invertibilità e la iniettività coincidono.*

Un metodo comodo di produrre automi cellulari invertibili è attraverso il *vicinato di Margolus* [15]. Si basa su una generalizzazione della definizione di automa cellulare che rimane comunque riconducibile alla definizione base che abbiamo adottato [14]. Consideriamo uno spazio cellulare bidimensionale e un insieme di stati S. Partizioniamo lo spazio cellulare in blocchi secondo una griglia due per due. L'aggiornamento non è più definito sulle singole cellule, ma sui blocchi, e avviene in due fasi distinte. Un automa cellulare di questo tipo è definito da una funzione biettiva $\pi : S^4 \longrightarrow S^4$.

La prima fase di aggiornamento prevede l'applicazione di π ad ogni blocco. Dopo la prima applicazione di π il partizionamento dello spazio cellulare cambia: la griglia si trasla di una cellula in orizzontale e in verticale (vedi figura 3.5). La seconda fase prevede una seconda applicazione della funzione π su ogni blocco e una nuova traslazione della griglia, sempre di una cellula in verticale e di una in orizzontale. La funzione viene applicata quindi a blocchi diversi nella prima e nella seconda fase. Si noti che la seconda traslazione porta ad un partizionamento identico a quello di partenza[3]: le partizioni delle cellule sono due e si alternano regolarmente per numero di traslazioni pari e dispari. La

[3]A volte ci si riferisce ai due partizionamenti come a *per coordinate pari* e *per coordinate dispari* a seconda che la cellula $(2, 0)$ sia nello stesso blocco della cellula $(1, 0)$ o della cellula $(3, 0)$.

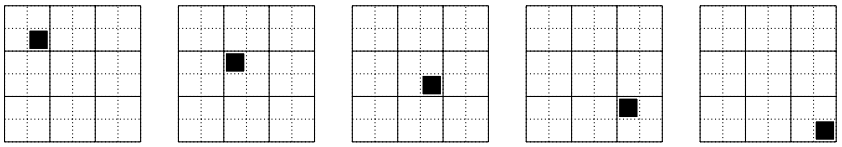


Figura 3.6: Propagazione diagonale di una particella. Le generazioni sono tre, gli aggiornamenti due, ognuno diviso in due fasi, la prima sulla base del partizionamento definito dalle righe tratteggiate.

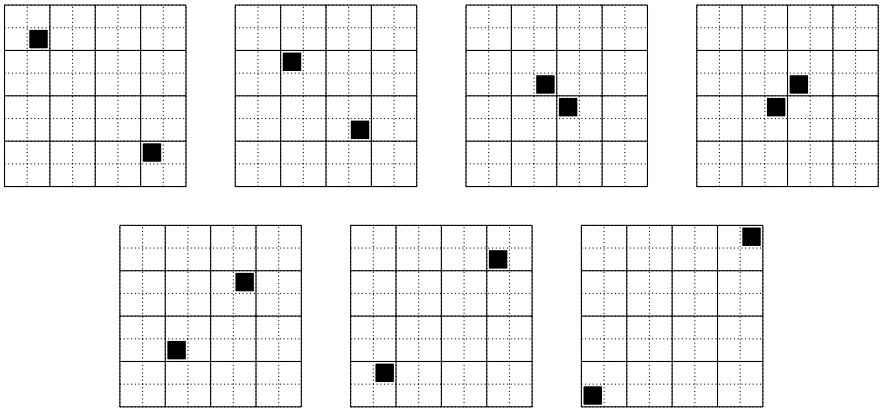


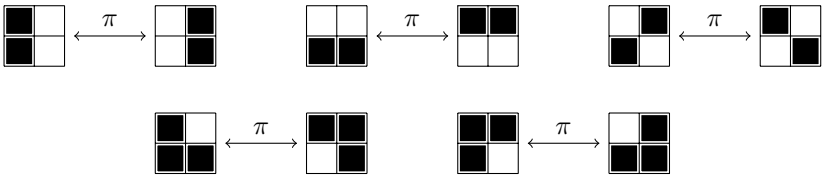
Figura 3.7: Urto frontale di due particelle. Le generazioni sono quattro, gli aggiornamenti tre, ognuno diviso in due fasi, la prima sulla base del partizionamento definito dalle righe tratteggiate.

combinazione di due applicazioni alternate di π (su ogni partizione) e della traslazione definisce la funzione di aggiornamento globale dell'automa cellulare.

La traslazione è una funzione biettiva, π lo è per ipotesi, pertanto l'automa cellulare risultante è *invertibile*. La funzione di aggiornamento globale dell'inversa corrisponde ad applicare le stesse fasi nello stesso ordine, partendo però dal partizionamento dello spazio cellulare opposto a quello dell'automa cellulare di partenza, e usando la funzione inversa di π.

Come abbiamo detto, questo modello può essere ricondotto alla definizione di automa cellulare a cui siamo abituati. L'idea è considerare ogni blocco due per due come cellula; in questo modo l'insieme degli stati che ogni cellula può assumere sono le possibili configurazioni di un blocco, ovvero S^4. Non è difficile a questo punto derivare la regola di aggiornamento che cattura il comportamento dell'automa originario, per effetto dell'alternanza del partizionamento si ottiene che l'indice di vicinato è quello di Moore con raggio uno (M_1^2).

Esempio 6. *Si consideri l'automa cellulare con vicinato di Margolus definito dalla funzione π uguale all'identità tranne nei seguenti casi:*



Si tratta del HPP-model*, un* lattice gas automata *molto semplice. Ogni cella nello stato* ■ *corrisponde ad una* particella*, e, a seconda della posizione nel blocco due per due, si muove in una delle quattro possibili direzioni diagonali. Ogni particella si muove nella direzione opposta alla sua posizione nel blocco. Ad esempio, Nella figura 3.6, una particella in alto a sinistra si muovo in diagonale verso sud-est. Si noti che, per ogni aggiornamento, abbiamo rappresentato anche lo stato della configurazione alla fine della prima fase, ma che le "vere" configurazioni oltre alla prima sono solo la terza e l'ultima. È abbastanza facile vedere che in questo modello le particelle mantengono la propria traiettoria e meno di scontrarsi frontalmente con un'altra particella, nel qual caso modificano la direzione di* 90° *(vedi figura 3.7).* π *è banalmente invertibile, quindi lo è anche l'automa cellulare* HPP-model*.*

Quando si pensa agli automi cellulari come a sistemi legati al mondo fisico, è necessario considerare l'importanza delle leggi di conservazione. Sono molte le quantità fisiche che possono essere conservative in un sistema, come la quantità di moto, l'energia o il momento angolare. L'idea è quella di trovare un formalismo che consenta la definizione di quantità arbitrarie definite su automi cellulari, in modo da poter studiare quali leggi di conservazione valgano in un dato automa cellulare e quali no, e possibilmente progettare automi cellulari perché conservino alcune quantità stabilite.

Il vicinato di Margolus è interessante anche in questo campo. Come esempio di questo notiamo quanto sia facile definire un automa cellulare che conservi il numero complessivo di cellule in un dato stato: è infatti sufficiente imporre come vincolo che il numero di cellule in quello stato non possa cambiare con l'applicazione di π.

Il formalismo che presentiamo sfrutta la nozione di *quantità additiva.* Molte alternative possono essere prese in considerazione, così come anche molte generalizzazione dei concetti che seguono.

Definizione 11. *(quantità additiva). Dato un automa cellulare* G *di dimensione* d *con insieme degli stati* S *e stato quiescente* q*, si dice quantità additiva sulle configurazioni di* G *una coppia*

$$A = (N, \varphi)$$

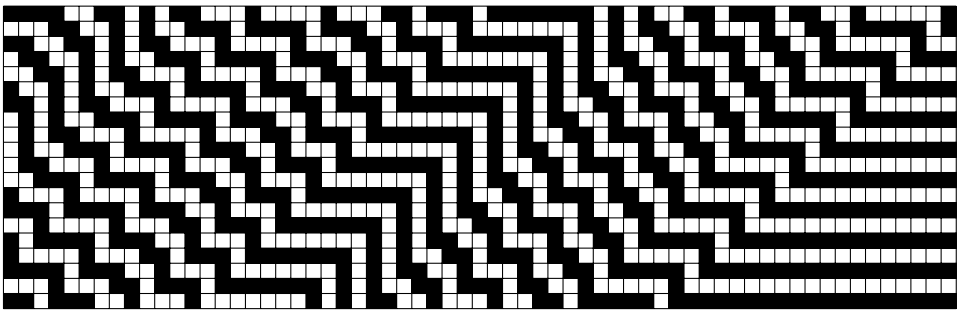


Figura 3.8: Prime 20 righe del diagramma spazio tempo dell'automa cellulare elementare con regola 85 e configurazione iniziale casuale.

dove:

- $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$ *è una tupla composta da* m *elementi distinti di* $\mathbb{Z}^d$ *(detto* indice di vicinato*)*[4]
- $\varphi : S^m \longrightarrow \mathbb{R}$ *detta* funzione di densità *tale che*

$$\varphi(q, q, \ldots q) = 0$$

Sia $c \in S^{\mathbb{Z}^d}$ una configurazione, la sua *densità* $\Phi_c : \mathbb{Z}^d \longrightarrow \mathbb{R}$ è una funzione che associa ad ogni cellula la propria densità.

$$\forall \vec{x} \in \mathbb{Z}^d . \Phi_c(\vec{x}) = \varphi(c(\vec{x} + \vec{n}_1), c(\vec{x} + \vec{n}_2), \ldots, c(\vec{x} + \vec{n}_m))$$

La *somma delle densità* delle singole cellule è finita se c è una configurazione finita.

$$\hat{\varphi}(c) = \sum_{\vec{x} \in \mathbb{Z}^d} \Phi_c(\vec{x})$$

Diremo che un automa cellulare G *conserva* la quantità additiva (N, φ) se $\hat{\varphi}(G(c)) = \hat{\varphi}(c)$ per ogni configurazione finita c.

Esempio 7. *L'esempio più comune di quantità additiva conservata da un automa cellulare è il numero di cellule che si trovano in un dato stato* $s' \in S$*. In questo caso la quantità additiva può essere definita formalmente come* $((\vec{0}), \varphi)$ *dove il vicinato di ogni cellula contiene solo sé stessa e la funzione di densità* φ *è definita come*

$$\varphi(s) = \begin{cases} 1 & se\ s = s' \\ 0 & altrimenti \end{cases}$$

Esempio 8. *Come esempio più stravagante possiamo chiederci se un automa cellulare elementare preservi il numero di cellule che hanno stato differente da*

[4]Questa tupla non è generalmente la stessa del vicinato dell'automa cellulare in considerazione.

quello della cellula immediatamente a sinistra, ma uguale a quello dalla cellula più a sinistra di ancora una posizione. Questa quantità additiva corrisponde ad $A = ((-2,-1,0),\varphi)$*, dove* $\varphi(s'',s',s) = \overline{(s'' \oplus s)} \wedge (s' \oplus s)$ *e* $\oplus$ *è il simbolo che rappresenta la disgiunzione esclusiva. Si noti che il vicinato della quantità additiva è diverso da quello dell'automa cellulare. Un automa cellulare elementare che soddisfa A è quello con numero di Wolfram* 85 *(vedi figura 3.8).*

Un primo risultato positivo riguardo la conservazione di quantità additive è che è decidibile se un dato automa cellulare conserva una data quantità. L'idea è che basta dimostrare che la modifica dello stato di un'unica cellula produce la stessa variazione nella somma delle densità indipendentemente da se viene applicata prima o dopo la funzione di aggiornamento G. Più precisamente possiamo dire che un automa cellulare G conserva (N,φ), se per ogni coppia c, c' che differiscono per lo stato di un unica cellula vale

$$\hat{\varphi}(c') - \hat{\varphi}(c) = \hat{\varphi}(G(c')) - \hat{\varphi}(G(c))$$

La dimostrazione si base sul fatto che ogni configurazione finita può essere trasformata in ogni altra configurazione finita da una serie finita di modifiche allo stato di un'unica cellula [14].

Un compito ancora più interessante è quello di determinare tutte le quantità additive conservate da un dato automa cellulare. Ad esempio è spesso importante accertarsi che un automa cellulare non conservi quantità che non sono conservative nel sistema fisico che si vuole simulare[5]. Dato un automa cellulare G e un indice di vicinato N, le funzioni di densità $\varphi : S^m \longrightarrow \mathbb{R}$ che definiscono quantità additive conservate da G formano uno spazio vettoriale su $\mathbb{R}$ [8]. Questo permette di caratterizzare finitamente l'insieme.

Teorema 10. *(Hattori e Takesue). Esiste un algoritmo che determina se una data quantità additiva è conservata da un dato automa cellulare. Per un dato automa cellulare G e indice di vicinato N è possibile trovare tutte le quantità additive conservate da G che abbiano indice di vicinato N.*

[5] È il problema principale di alcuni *lattice gas automata*, come l'*HPP-model*.

Capitolo 4

Universalità

4.1 Definire l'universalità computazionale

Abbiamo da subito chiarito che il potere espressivo degli automi cellulari è equivalente a quello delle macchine di Turing. Quindi, secondo la *tesi di Church-Turing*, è possibile progettare un automa cellulare in grado di calcolare qualunque problema che sia effettivamente risolvibile.

Nel caso della computazione su automi cellulari, un problema interessante deriva dalla difficoltà di descrivere le comuni nozioni di *dato in ingresso*, *dato in uscita* e *terminazione*. La funzione di aggiornamento globale è infatti totale sul dominio $S^{\mathbb{Z}^d}$, pertanto non esiste mai una configurazione finale e di conseguenza non è nemmeno banale definire un output. Nel capitolo 2 abbiamo affermato di poter considerare le configurazioni $c, G(c), G^2(c), G^3(c), \ldots$ come sequenza di passi di computazione di un algoritmo con $c \in S^{\mathbb{Z}^d}$ come dato in ingresso. Un'alternativa che è stata studiata è quella di considerare una parte della configurazione iniziale come dato in ingresso e il resto come una parte del modello di calcolo [4]. Vedremo che, in questa accezione, non è l'automa cellulare (d, S, N, f) a calcolare la funzione φ, ma la sua configurazione $c \in S^{\mathbb{Z}^d}$ di cui una "parte" è riservata alla rappresentazione codificata dell'input.

I problemi da risolvere per poter parlare di computazione in relazione agli automi cellulari sono dunque: come codificare gli argomenti della funzione da calcolare, come riconoscere la terminazione della computazione, come decodificare il risultato della computazione nell'immagine della funzione. La ricerca di una soluzione che sia adatta a qualsiasi automa cellulare, al fine di definire l'universalità computazionale per gli automi cellulari prende il nome di *universalità diretta*.

Un approccio completamente differente alla computazione, e dominante rispetto al precedente è la simulazione esplicita, all'interno di un automa cellulare, di un modello di calcolo noto (*universalità indiretta*). Se il modello simulato è computazionalmente universale, allora anche l'automa cellulare che lo simula deve esserlo. La definizione di computazione (e quindi di terminazione, input e output), deriva automaticamente dal modello simulato. Seguendo questo approccio tuttavia si rinuncia a trovare una definizione di computazione adatta ad ogni automa cellulare.

Esempi tipici di modelli di calcolo implementati all'interno di automi cellulari sono le reti logiche per spazi cellulari di dimensione due e le macchine di Turing per quelli di dimensione uno. Nel caso delle reti logiche, comunemente si implementano delle particelle per mezzo di pattern dell'automa cellulare. Queste particelle codificano informazioni booleane positive o negative e svolgono il ruolo della corrente elettrica nei circuiti logici reali. Il passo successivo consiste nel dimostrare che le interazioni fra queste particelle possono esprimere in qualche modo il comportamento delle porte logiche.

Nel caso unidimensionale la codifica del contenuto del nastro di una macchina di Turing è piuttosto naturale. Per catturare il comportamento della testina qualche precisazione è invece dovuta. È importante anche la simulazione dei *tag system*, sopratutto perché la tecnica impiegata è alla base della dimostrazione di universalità dell'automa cellulare elementare con numero di Wolfram 110.

L'universalità non è una proprietà esclusiva di automi cellulari disegnati ad hoc per soddisfare una definizione formale o per imitare il comportamento di un qualche modello di calcolo Turing equivalente. Sono molti gli esempi che caratterizzano l'universalità computazionale come fenomeno spontaneo. Un esempio famoso è *game-of-life*, che abbiamo già presentato nell'introduzione. È una congettura famosa quella secondo la quale alcuni automi cellulari elementari della quarta classe di Wolfram sarebbero computazionalmente universali. È stata recentemente dimostrata l'universalità dell'automa cellulare con numero di Wolfram 110.

Abbiamo sottolineato l'importanza della ricerca di automi cellulari invertibili, come modello di calcolo massicciamente parallelo adatto ad essere implementato da fenomeni microscopici. Perché questo abbia una reale utilità è anche necessario che gli automi cellulari invertibili siano Turing equivalenti. Fortunatamente risultati noti della teoria degli automi cellulari garantiscono l'esistenza di automi cellulari invertibili e computazionalmente universali di ogni dimensione. Un esempio bidimensionale molto noto è il *billiard ball model* di Margolus.

4.2 Universalità diretta

Presentiamo la formalizzazione del concetto di universalità computazionale proposta da Codd [2]. Non esiste attualmente una definizione condivisa in modo unanime di computazione riguardo gli automi cellulari. Quella che trattiamo nel seguito non è la più generale fra quelle ideate, ma ha il pregio di essere piuttosto semplice.

Il nostro obiettivo è quello di definire in quali casi possiamo affermare che un automa cellulare è in grado di calcolare ogni funzione Turing-calcolabile $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ su ogni possibile input $n \in \mathbb{N}$. Cominciamo con alcune definizioni preliminari.

Definizione 12. *Sia (d, S, N, f) un automa cellulare con funzione di transizione globale G e stato quiescente q.*

- *Due configurazioni $c, c' \in S^{\mathbb{Z}^d}$ sono dette* disgiunte *se $supp_q(c) \cap supp_q(c') = \emptyset$.*
- *Date due configurazioni disgiunte $c, c' \in S^{\mathbb{Z}^d}$ denotiamo con $c \cup c'$ e chiamiamo* unione *di c e c' la configurazione tale che*

$$\forall \vec{x} \in \mathbb{Z}^d.\ c \cup c'(\vec{x}) = \begin{cases} c(x) & se\ \vec{x} \in supp_q(c) \\ c'(x) & se\ \vec{x} \in supp_q(c') \\ q & altrimenti \end{cases}$$

- *Una* sottoconfigurazione *c' di una configurazione c è una configurazione tale che*

$$\forall \vec{x} \in supp_q(c').\ c(\vec{x}) = c'(\vec{x})$$

- *Una configurazione è detta* passiva *se è un punto fisso per G e* completamente passiva *se ogni sua sottoconfigurazione è passiva.*
- *Diciamo che una configurazione c* passa informazione *ad una configurazione c', disgiunta da c, se*

$$\exists t \in \mathbb{N}.\ \exists \vec{x} \in supp_q(G^t(c')).\ G^t(c \cup c')(\vec{x}) \neq G^t(c')(\vec{x})$$

- *Diciamo che una configurazione c è una* traslazione *di una configurazione c' se*

$$\exists \tau \in \mathbb{Z}^d\ .\forall \vec{x} \in \mathbb{Z}^d.\ c'(\vec{x}) = c(\vec{x} + \tau)$$

Definiamo adesso un insieme di configurazioni che useremo per codificare i numeri naturali $n \in \mathbb{N}$ che la funzione da calcolare riceve come dati in ingresso e restituisce come dati in uscita.

Definizione 13. *Sia $A = (d, S, N, f)$ un automa cellulare, un* dominio di Turing *per A è un insieme di configurazioni $T_A \subseteq S^{\mathbb{Z}^d}$ che soddisfa le seguenti condizioni:*

- *ogni configurazione in T_A è* finita *ed esiste una* funzione effettiva e iniettiva *$i_A : \mathbb{N} \longrightarrow T_A$;*
- *ogni configurazione in T_A è* completamente passiva*;*
- *le configurazioni in T_A sono* collettivamente passive*, ovvero per ogni sottoinsieme finito $T' \subset T_A$, per ogni insieme di traslazioni per gli elementi di T', se la traslazione degli elementi di T' porta a configurazioni disgiunte, allora la loro unione è completamente passiva.*

Intuitivamente la definizione precedente serve ad assicurarci che ogni "dato" non riceva informazioni durante la computazione da altri "dati". Per stabilire la terminazione di una computazione faremo uso di uno stato $s_{halt} \in S$, $s_{halt} \neq q$. La configurazione c corrisponde ad una terminazione se una cellula $\alpha \in supp_q(c)$ è nello stato s_{halt}. Un insieme di celle importante è $\bigcup_{t \in T_A} supp_q(t)$ che contiene la codifica dell'input all'inizio della computazione e quella dell'output al momento della terminazione. Infine la codifica da $\mathbb{N}$ a T_A è rappresentata da i_A, la decodifica dalla sua inversa (i_A^{-1}). Adesso abbiamo tutto quello che ci serve per definire la computazione per un automa cellulare.

Definizione 14. *Dato un automa cellulare $A = (d, S, N, f)$ con funzione di transizione globale G e stato quiescente q, su cui è definito un dominio di Turing T_A, diciamo che una sua configurazione $c \in S^{\mathbb{Z}^d}$, $c \notin T_A$ calcola $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$ se esiste una cellula $\alpha \in supp_q(c)$ e uno stato non quiescente $s_{halt} \in S$, $s \neq q$ tale che, per ogni n nel dominio di φ esiste un $t \in \mathbb{N}$ tale che:*

- $G^t(c \cup i_A(n))(\alpha) = s_{halt}$,
- $G^{t'}(c \cup i_A(n))(\alpha) \neq s_{halt}$ *per ogni* $t' < t$,
- *Siano*

$$c'(\vec{x}) = \begin{cases} q & \text{se } \vec{x} \in \bigcup_{t \in T_A} supp_q(t) \\ G^t(c \cup i_A(n))(\vec{x}) & \text{altrimenti} \end{cases}$$

  e

$$c''(\vec{x}) = \begin{cases} G^t(c \cup i_A(n))(\vec{x}) & \text{se } \vec{x} \in \bigcup_{t \in T_A} supp_q(t) \\ q & \text{altrimenti} \end{cases}$$

  allora c' non passa informazione a c'',
- $(G^t(c \cup i_A(n)))(\vec{x}) = (i_A(\varphi(n)))(\vec{x})$ *per ogni* $\vec{x} \in \bigcup_{t \in T_A} supp_q(t)$.

A questo punto risulta abbastanza naturale che una automa cellulare computazionalmente universale sia definito come una automa cellulare tale che per ogni funzione Turing-calcolabile, ammette una configurazione che la calcola.

Definizione 15. *(automa cellulare computazionalmente universale). Un automa cellulare* $A = (d, S, N, f)$ *è detto* computazionalmente universale *se esiste un dominio di Turing per esso e se, per ogni funzione parziale Turing-calcolabile* $\varphi : \mathbb{N} \longrightarrow \mathbb{N}$*, esiste una configurazione* $c \in S^{\mathbb{Z}^d}$*,* $c \notin T_A$*, che calcola* φ*.*

Come abbiamo già detto più volte questo non è altro che uno dei modi possibili per formalizzare il concetto di computazione su un automa cellulare (e di conseguenza di universalità computazionale). Molte scelte possono essere considerate arbitrarie e molte generalizzazioni sono possibili [23]. La ricerca di una buona definizione sembra attualmente essere legata a quella più generale di sistema dinamico computazionalmente universale [5].

4.3 Caso bidimensionale

Un approccio completamente differente da quello proposto nel capitolo precedente consiste nella simulazione di un modello di calcolo noto tramite un automa cellulare: l'universalità indiretta, sulla quale ci concentreremo per il resto del capitolo. Nel caso bidimensionale studieremo la simulazione, per mezzo di configurazioni finite, di macchine a stati finiti. Raggiunto questo obiettivo risulta evidente che l'unione di un numero illimitato di configurazioni finite possa simulare con facilità una macchina di Turing o altri formalismi noti.

Presentiamo l'automa cellulare *Copper* (in breve *Cu*) [21] come semplice automa cellulare adatto a simulare il comportamento di particelle su cavi.

Definizione 16. *(automa cellulare Copper) Chiamiamo* Copper *l'automa cellulare definito come* $(2, \{0, 1, 2\}, V_1^2, f)$ *dove:*

- *la dimensione dello spazio cellulare è* 2*;*
- *l'insieme degli stati è* $\{0, 1, 2\}$ *dove:*
  - 0 *corrisponde al* vuoto *(rappresentato graficamente da un quadrato bianco),*
  - 1 *corrisponde al* cavo *(rappresentato graficamente da un quadrato nero),*
  - 2 *corrisponde alla* particella *(rappresentata graficamente da una croce nera);*
- *l'indice di vicinato è di tipo* Von Neumann *con raggio* 1 *(*V_1^2*);*

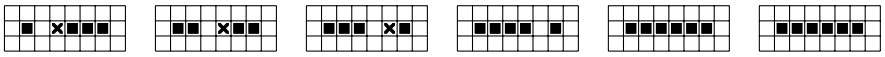


Figura 4.1: Una particella che si muove da sinistra a destra in un cavo.

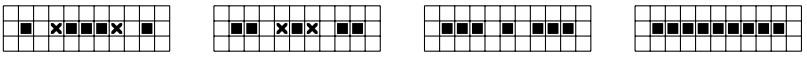


Figura 4.2: Due particelle che si scontrano su un cavo.

- *la regola di aggiornamento locale f è definita informalmente come*[1]*:*
  - *una cellula in stato* 0 *cambia il suo stato in* 1 *se le cellule a nord e sud o quelle ad est e ovest sono in stato* 1 *o* 2*, altrimenti rimane in stato* 0*;*
  - *una cellula in stato* 1 *cambia il suo stato in* 2 *se almeno due vicini diversi dalla cellula stessa sono nello stato* 1 *o* 2 *ed esattamente uno di questi è nello stato* 1 *o nello stato* 2*, altrimenti rimane in stato* 1*;*
  - *una cellula in stato* 2 *cambia il suo stato sempre in* 0*.*

Un cavo in Cu corrisponde ad una linea dritta composta da cellule in stato 1 e circondata da cellule in stato 0. Una particella che corre su un cavo è rappresentata da una cellula nello stato 2 seguita (nel verso dello spostamento della particella) da una in stato 0 (vedi figura 4.1). Due particelle che si muovono in verso opposto quando si incontrano svaniscono (vedi figura 4.2). Notiamo che il muoversi di una particella e lo scontro di due particelle non danneggiano il cavo[2].

Per completare la descrizione del comportamento delle particelle in Cu prendiamo in esame i *punti di intersezione*, ovvero le cellule comuni a due o più cavi. La connessione di due cavi non si comporta in maniera utile (la particella distrugge la connessione), sarà quindi esclusa dalle nostre costruzioni. Quando parleremo di punti di intersezione in generale intenderemo sempre il caso di almeno tre cavi.

Si prenda in considerazione una intersezione di $n \in \{3, 4\}$ cavi. Le situazioni che possono verificarsi sono le seguenti:

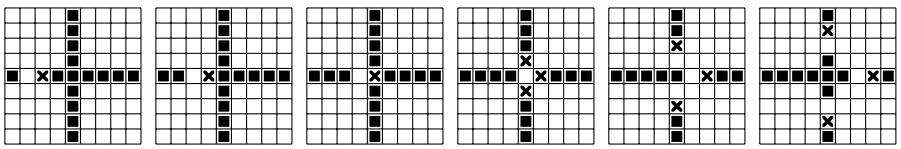


Figura 4.3: Una particella si propaga attraverso un punto di intersezione.

[1] Una definizione formale è facilmente derivabile da quella presentata, non consideriamo utile riportarla esplicitamente.

[2] In realtà per lo scontro vale solo se assumiamo che la distanza, intesa come numero di cellule cavo fra le due particelle, sia dispari. Questo non complica molto la costruzione, perciò sarà escluso dalla trattazione.

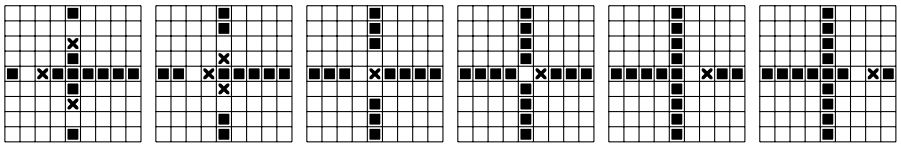


Figura 4.4: Le particella si fondono quando arrivano ad un punto di intersezione.

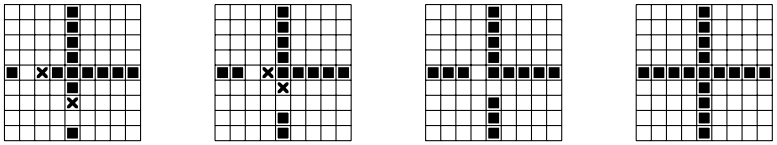


Figura 4.5: Le particella scompaiono quando arrivano ad un punto di intersezione.

1. se una sola particella arriva al punto di intersezione, si propaga in tutti i cavi eccetto quello da cui proviene (vedi figura 4.3);

2. se arrivano al punto di intersezione contemporaneamente $n - 1$ particelle, ovvero una da ogni cavo tranne uno, le particelle si fondono e una singola carica viene immessa sull'unico cavo da cui non è arrivata alcuna carica (vedi figura 4.4);

3. se arrivano al punto di intersezione contemporaneamente k particelle, con $k \neq n - 1$ e $k \neq 1$, le particelle svaniscono (vedi figura 4.5).

Sfruttando il comportamento delle particelle sulle intersezioni costruiamo delle porte logiche capaci di creare un qualsiasi circuito booleano. Definiamo innanzitutto il *diodo* (figura 4.6). Esso calcola semplicemente la funzione l'identità, ma ha l'importante proprietà di lasciar passare solo le particelle provenienti dall'ingresso (a sinistra nella figura) e bloccare quelle che si muovono nel verso opposto. Si noti infatti che una particella proveniente da sinistra si scinde alla prima intersezione propagandosi nei due cavi che non sono vicoli ciechi (caso 1), per fondersi nuovamente all'ultima intersezione, in quanto proveniente da due dei tre cavi che si intersecano (caso 2). Una particella proveniente da destra invece si diffonde alla prima intersezione (caso 1), ma all'ultima svanisce in quanto proveniente da due soli dei quattro cavi che si intersecano (caso 3).

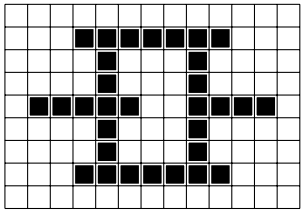
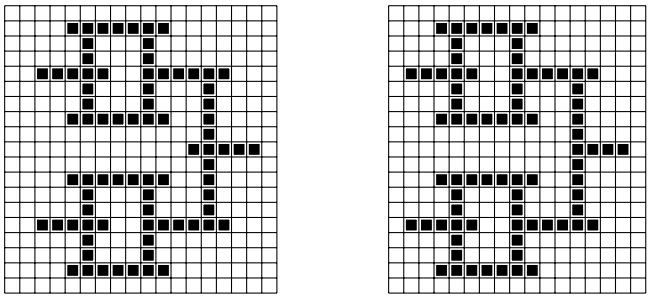


Figura 4.6: Un *diodo* permette il passaggio di particelle solo in un verso.



(a) (b)

Figura 4.7: Porte logiche XOR (a) e OR (b) implementate in Cu.

Osserviamo inoltre che è stato possibile simulare l'intersezione di esattamente due cavi (al fine di curvare il cavo) intersecandone tre di cui uno composto da una sola cellula (vicolo cieco).

Ci accontenteremo di implementare le porte logiche XOR e OR, in quanto sufficienti ad esprimere ogni possibile funzione booleana. In entrambi i casi le particelle in input attraversano un diodo per prima cosa, questo consente di evitare che da un cavo in ingresso possano uscire delle particelle. Nel caso dello XOR vogliamo che ci sia una particella in uscita solo quando una e una sola delle due potenziali particelle entra nella porta. La proprietà di avere una particella in uscita per una particella in ingresso e nessuna particella in uscita per due particelle in ingresso è banalmente soddisfatta da un'intersezione di quattro cavi in cui da uno non arrivino mai particelle. Anche in questo caso useremo un vicolo cieco (vedi figura 4.7a). Se nella porta entra una sola particella da un cavo di input, questa si propaga in tutti gli altri cavi (caso 1). Uno corrisponde all'uscita della porta, l'altro è un vicolo cieco per costruzione e l'ultimo è il cavo dell'ingresso da cui non è arrivata alcuna particella, in questo caso il diodo blocca la particella. Se nel circuito entrano due particelle invece, queste svaniscono quando arrivano all'intersezione dei quattro cavi (caso 3). Ovviamente, se nessuna particella entra nella porta, nessuna particella esce dalla porta.

Il caso della porta OR è simile al precedente, in questo caso vorremmo però che ci fosse una particella in uscita anche in corrispondenza di due particelle in ingresso. È sufficiente sostituire l'intersezione di quattro cavi con l'intersezione di tre cavi, corrispondenti ai due ingressi e all'uscita (vedi figura 4.7b). Come prima, se non arrivano particelle in ingresso, non abbiamo niente in uscita, e all'arrivo di una singola particella questa si propaga su tutti i cavi (caso 1). Nel caso di due particelle in ingresso invece, quando queste arrivano nell'ultima intersezione abbiamo che una particella viene immessa nel cavo di uscita (caso 2).

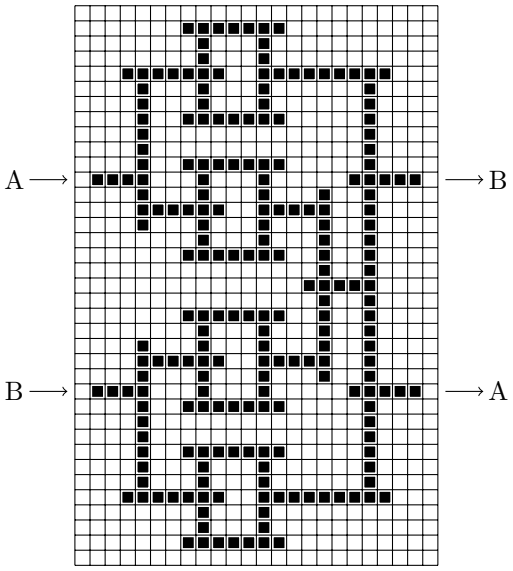


Figura 4.8: Incrocio di due cavi: le particelle provenienti dal cavo in ingresso in alto escono dal cavo in uscita in basso e viceversa.

Un'altra componente importante è l'incrocio di due cavi. Non sempre vogliamo che due cavi che si incrociano siano connessi, è ragionevole sperare di poter far attraversare ad un cavo una zona dello spazio cellulare occupata da un altro cavo senza che le eventuali particelle presenti su di loro interagiscano. L'incrocio di due cavi è ottenuto grazie alla porta XOR. In generale sappiamo che vale la seguente proprietà: $p \oplus (q \oplus p) = q$ dove $\oplus$ è il simbolo per la disgiunzione esclusiva. Dati due cavi in ingresso con particelle A e B posizioniamo le porte XOR in modo da comporre nell'ordine $A \oplus (B \oplus A)$ e $B \oplus (A \oplus B)$ (vedi figura 4.8). Si noti che, poiché lo stesso valore è atteso in ingresso da più porte, le eventuali particelle devono essere duplicate per mezzo di intersezioni. Anche in questo caso facciamo uso dei diodi per impedire alle particelle di percorrere la porta nel verso errato.

Risulta abbastanza chiaro in che modo sia possibile costruire una qualsiasi macchina a stati finiti per mezzo delle porte che abbiamo definito fino a qui. Il primo passo consiste nel definire una codifica booleana per lo stato interno, e per i simboli di input e output. In questo modo, sia la funzione che aggiorna lo stato interno della macchina, sia quella che definisce l'output sono funzioni booleane; possono quindi essere calcolate da circuiti ottenuti componendo le porte OR e XOR. Come abbiamo notato non è difficile duplicare le particelle e ridirigerle dove servono, è quindi facilmente implementabile il classico ciclo di aggiornamento dello stato interno della macchina a stati finiti. Notiamo che il ciclo di clock dipende sia dalla complessità delle funzioni booleane da calcolare,

sia dal tempo necessario affinché il valore dello stato all'istante successivo si sposti dall'output del circuito all'input.

Per simulare un formalismo computazionalmente universale è necessario collegare più macchine a stati finiti (l'output di una con l'input dell'altra e così via). Questo può creare qualche problema con il calcolo del ciclo di clock. Il modo più semplice di collegare un qualunque numero di macchine a stati finiti è di far sì che queste condividano lo stesso ciclo di clock, ovvero che tutte le macchine abbiano artificialmente il ciclo di clock maggiore. Questo è indesiderabile in quanto il periodo non sarebbe limitato a priori. La soluzione più semplice a questo problema è quella di usare delle memorie fra ogni coppia di macchine a stati finiti. Le memorie possono essere implementate a loro volta come macchine a stati finiti con un ciclo di clock molto piccolo τ. Sarà sufficiente a questo punto forzare ogni macchina ad avere un ciclo di clock che sia multiplo di τ.

La costruzione descritta per Cu, seppure informale, è abbastanza precisa da convincerci di avere di fronte un automa cellulare universale. Provando a generalizzare il risultato ottenuto possiamo affermare che, per simulare i circuiti booleani, un automa cellulare deve essere in grado di:

- definire *cavi*, ovvero le possibili direzioni dei segnali, che possono essere espliciti come in Cu o codificati in una proprietà della particella;
- modificare la *direzione* dei segnali;
- *ritardare* l'arrivo di un segnale per permettere la sincronizzazione;
- definire *incroci* di cavi che preservino lo stato dei segnali;
- definire un qualunque insieme universale di *porte logiche*;
- consentire la *duplicazione* dei segnali usando stati appositi o sfruttando regole dei cavi.

Alcuni accorgimenti utili per la simulazione di circuiti logici sono invece:

- l'uso di clock espliciti è comodo anche se spesso non essenziale[3];
- gli incroci fra cavi sono gratuiti se è possibile implementare lo XOR;
- il ritardo dei segnali è gratuito se questi possono essere ridiretti in ogni direzione.

[3]Se codifichiamo (come abbiamo fatto in Cu) l'assenza di segnale come valore booleano negativo $false$ allora il clock è necessario per fornire un output alla porta logica NOT. Si noti ad esempio che in Cu da sole cellule in stato 0 e 1 non verrà mai generata una particella. Un'alternativa è quella di modificare la rappresentazione e usare due cavi per ogni segnale, il primo contiene una particella se il segnale è $true$, l'altro se il segnale è $false$, assenza di particelle significa assenza di valore.

4.4 Caso unidimensionale

Nel caso unidimensionale la simulazione di circuiti booleani risulta essere piuttosto intricata, per questo si preferisce solitamente simulare direttamente un sistema di calcolo universale, in genere il più semplice possibile. Presentiamo come esempio la costruzione di un automa cellulare unidimensionale capace di simulare una macchina di Turing [21] [11]. Sembra abbastanza naturale vedere una somiglianza fra il nastro infinito della macchina di Turing e lo spazio cellulare unidimensionale. Come vedremo l'unica difficoltà è quella di simulare il comportamento del cursore. Parliamo anche dei concetti fondamentali per la simulazione del tag system, per il quale la costruzione non risulta altrettanto immediata.

La variante della macchina di Turing che useremo è la seguente. Una *macchina di Turing* è una tupla $(Q, \Sigma, \#, q_0, \delta)$ dove Q è l'insieme finito degli stati, Σ è l'alfabeto finito dei simboli del nastro, $\# \in \Sigma$ è il carattere *blank*, $q_0 \in Q$ è lo stato iniziale della macchia di Turing, e $\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{-1, 0, 1\}$ è la funzione di transizione parziale. La funzione di transizione esprime il comportamento della macchina, una regola di transizione $\delta(q, a) = (q', b, d)$ corrisponde a: se la macchina è nello stato q e legge sul nastro in corrispondenza del cursore il carattere a, scrive sul nastro b, passa allo stato q' e muove il cursore in accordo con d. Una configurazione della macchina di Turing è espressa come (q, z, c) dove $q \in Q$ è lo stato della macchina, $z \in \mathbb{Z}$ è la posizione del cursore sul nastro, $c \in \Sigma^{\mathbb{Z}}$ rappresenta il contenuto del nastro (associa ad ogni posizione il carattere che vi è attualmente scritto). La macchina esegue un passo di computazione evolvendo da una configurazione (q, z, c) ad una (q', z', c') se $\delta(q, c(z)) = (q'', b, d)$ è definita e vale $q'' = q'$, $z' = z + d$, $c'(z) = b$ e $\forall z'' \in \mathbb{Z}. z'' \neq z \implies c'(z'') = c(z'')$.

Sia (Q, Σ, q_0, δ) la macchina di Turing da simulare. Definiamo S, l'insieme degli stati dell'automa cellulare come $(Q \cup \varnothing) \times \Sigma$, dove $\varnothing \notin Q$. La configurazione (q, z, c) di una macchina di Turing è codificata come la configurazione c' dell'automa cellulare tale che:

$$\forall x \in \mathbb{Z}.\ c'(x) = \begin{cases} (\varnothing, c(x)) & se\ x \neq z \\ (q, c(x)) & se\ x = z \end{cases}$$

Lo stato di ogni cellula rappresenta quindi sia il carattere contenuto nella stessa posizione del nastro nella macchina di Turing, sia la presenza o meno del cursore in quella posizione e in tal caso lo stato della macchina di Turing.

È compito della regola di aggiornamento locale quello di simulare sia lo spo-

stamento del cursore, sia la riscrittura del nastro. Prendiamo come indice di vicinato quello di raggio 1 ($\{-1, 0, 1\}$). Dato lo stato attuale di una cellula, questa conosce sia il carattere letto dal cursore, sia lo stato attuale della macchina, può quindi, essendo nota la funzione di transizione, modificare il proprio stato. Allo stesso modo la cellula immediatamente a destra e sinistra di quella relativa alla posizione del cursore hanno accesso allo stato della cellula, ovvero conoscono sia il carattere che lo stato della macchina e possono calcolare in quale direzione il cursore si sposterà. Più in dettaglio possiamo dire che la regola di transizione $f : S^3 \longrightarrow S$ è

$$f(s', s, s'') = \begin{cases} (q', b) & \textit{se } s = (q, a) \textit{ e } \delta(q, a) = (q', b, 0) \\ (\varnothing, b) & \textit{se } s = (q, a) \textit{ e } \delta(q, a) = (q', b, d) \textit{ e } d \neq 0 \\ (q', a) & \textit{se } s = (\varnothing, a) \textit{ e } (\ (\ s' = (q, b) \textit{ e } \delta(q, b) = (q', c, 1)) \textit{ o} \\ & \qquad\qquad (s'' = (q, b) \textit{ e } \delta(q, b) = (q', c, -1))\ ) \\ s & \textit{in ogni altro caso} \end{cases}$$

Si può dedurre che se la configurazione attuale dell'automa cellulare simula la configurazione c della macchina di Turing, la configurazione successiva $G(c)$ dell'automa cellulare simula la configurazione della macchina di Turing dopo un passo di computazione.

Notiamo che l'automa cellulare qui costruito è unidimensionale, ha vicinato di raggio uno e il numero di stati dipende dalla macchina di Turing da simulare; sia m il numero di stati $q \in Q$ della macchina di Turing, sia n il numero di caratteri nell'alfabeto Σ della macchina di Turing, allora sono necessari $(m + 1) \times n$ stati per simulare questa macchina di Turing con la costruzione presentata.

In letteratura sono presenti numerosi esempi di automi cellulari molto più "piccoli" (in termini di numero di stati) capaci di simulare una macchina di Turing[4] [11].

Un altro modello la cui simulazione è importante per l'universalità negli automi cellulari unidimensionali è il *tag system*. I principi su cui è basata la simulazione ideata da Cook [3] sono gli stessi che gli hanno permesso di dimostrare l'universalità computazionale dell'automa cellulare elementare con numero di Wolfram 110.

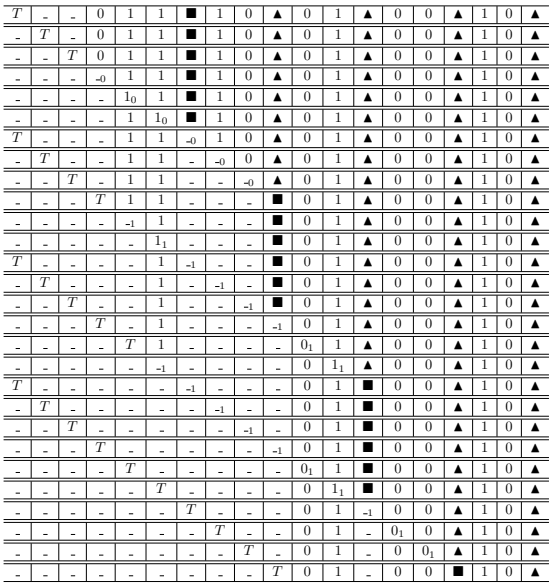
In realtà quelli che vengono implementati dalla costruzione di Cook sono una classe particolare di tag system, i *tag system ciclici*; parte non banale della dimostrazione sarebbe mostrare che hanno lo stesso potere espressivo dei tag system. Presentiamo brevemente il modello di calcolo da simulare. Un *tag sy-*

[4]Sebbene alcuni facciano uso indici di vicinato di raggio maggiore.

```
Cyclic Tag System
 Produzioni: (10, 01, 00)

Computazione
 Stringa Iniziale: 011
    Produzioni     Stringhe
    ----------     ----------
       10           011
       01            11
       00             101
       10              0100
       01                100
       00                  0001
       .                     .
       .                     .
```

(a)



(b)

Figura 4.9: Computazione di un *tag system ciclico* con produzioni $\{10, 01, 00\}$ e stringa iniziale 011 (a) e della sua simulazione su un automa cellulare (b) a confronto.

stem ciclico è definito da una tupla di n stringhe finite composte di 0 e 1 dette *produzioni* $P = (p_0, p_1, \ldots, p_n)$ dove per $i \in \{0, 1, \ldots, n\}$ vale $p_i \in \{0, 1\}^*$. La configurazione di un tag system corrisponde ad una stringa finita $s \in \{0, 1\}^*$. Ad ogni passo di computazione il primo carattere viene cancellato dalla stringa, successivamente se quello che si è cancellato era 1 si aggiunge in fondo alla stringa una p_i fra quelle in P. Le p_i vengono alternate in base all'ordine nella tupla, iterazione dopo iterazione, indipendentemente dal fatto che siano effettivamente state applicate (se il carattere letto era 1) o meno (se il carattere letto era 0). Una stringa $x\alpha$ all'iterazione i viene quindi trasformata in $t(x\alpha, i)$.

$$t(x\alpha, i) = \begin{cases} \alpha & se\ x = 0 \\ \alpha p_i & se\ x = 1 \end{cases}$$

La computazione termina solo quando si arriva ad ottenere la stringa vuota. Data una stringa in ingresso α la sequenza di configurazioni della computazione è $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m$ dove $\alpha_0 = \alpha$, $\alpha_m = \varepsilon$ e

$$\forall i \in \{1, 2, \ldots, m\}.\ \alpha_i = t(\alpha_{i-1}, i - 1 \bmod n)$$

Della costruzione che permette la simulazione di questo modello diamo solamente una breve descrizione informale in quanto i dettagli sono molti e una

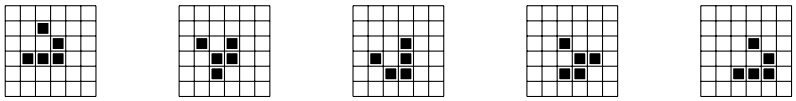


Figura 4.10: Glider con direzione sud-est.

trattazione precisa richiederebbe molto spazio e attenzione. Il problema principale è la non località della computazione: laddove una macchina di Turing legge e modifica il nastro unicamente in corrispondenza del cursore, nel tag system la lettura avviene all'inizio della stringa, mentre la scrittura avviene dalla parte opposta.

Diversamente dal caso precedente, ciò che costruiamo adesso è un automa cellulare capace di simulare qualunque tag system ciclico e non un modo per costruire un automa cellulare differente per ogni tag system. Le produzioni che definiscono il tag system saranno infatti parte della configurazione iniziale dell'automa cellulare. L'indice di vicinato è anche questa volta $\{-1, 0, 1\}$. Siano $(p_0, p_1, \ldots, p_n)$ le produzioni di un tag system circolare, e sia α la stringa in ingresso del tag system. La configurazione iniziale dell'automa cellulare sarà allora della forma $(T\_^k)^\omega \alpha \blacksquare (p_0 \blacktriangle \ldots \blacktriangle p_n \blacktriangle)^\omega$. Intuitivamente le cellule in stato T sono segnali (clock) che si propagano verso destra. L'intervallo di tempo fra un segnale e l'altro, ovvero la distanza fra una cellula nello stato T e la successiva andando verso sinistra, dipende dal valore di k. Non ci preoccuperemo di questo valore, assumendo che sia sempre adeguato alle nostre necessità. Quando il segnale raggiunge il primo carattere di α lo cancella ed emette un segnale verso destra che trasporta il valore appena cancellato. Quando il segnale raggiunge $\blacksquare$ lo rimuove e continua verso destra. Se il valore trasportato dal segnale è 1 allora questo attraversa p_0 senza modificarlo. Se il valore trasportato dal segnale è 0 allora cancella completamente p_0. In ogni caso quando il segnale raggiunge $\blacktriangle$ lo cambia in $\blacksquare$ e scompare. La computazione prosegue alla stessa maniera con il secondo carattere di α e il secondo segnale T. Si noti che come ci si aspetterebbe, alla seconda iterazione la produzione che viene applicata è p_1.

Per un esempio si veda la figura 4.9, nella quale mostriamo alcuni passi di esecuzione di un automa cellulare che simula il tag system ciclico con produzioni $\{10, 01, 00\}$ su stringa iniziale 011.

4.5 Universalità spontanea

Gli automi cellulari visti finora, basandosi sulla simulazione di modelli sequenziali, mascherano o negano la propria natura parallela. È interessante osservare l'universalità come fenomeno di automi cellulari che non siano disegnati ad hoc per simulare altri modelli di calcolo. Questo vuol dire che occorre "lavorare

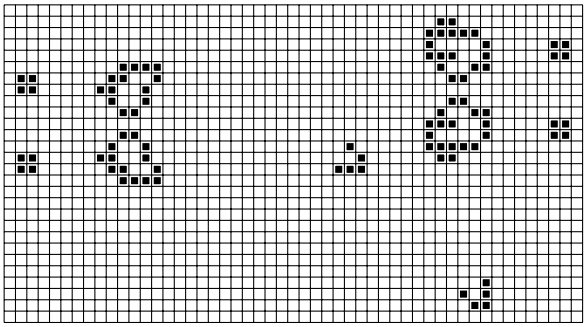


Figura 4.11: Gosper p46 gun.

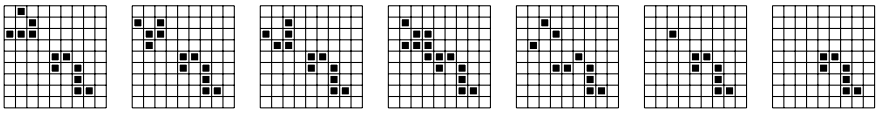


Figura 4.12: Collisione fra un glider e un eater.

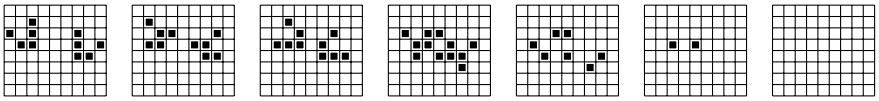


Figura 4.13: Crash fra due glider.

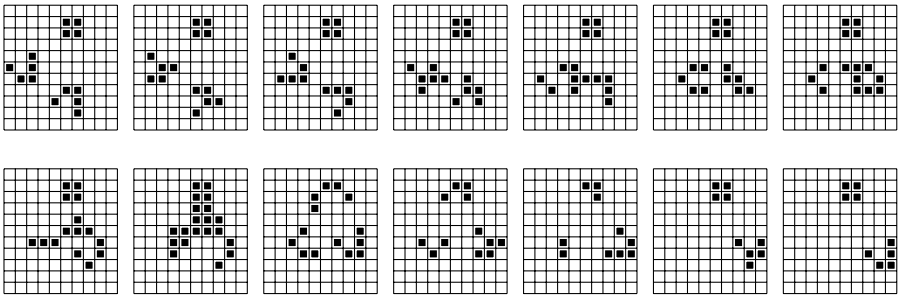


Figura 4.14: Collisione fra due glider nei pressi di un duplicator.

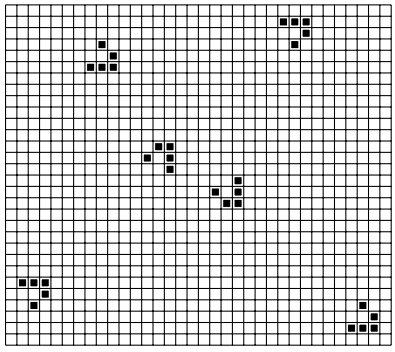


Figura 4.15: Crossing fra due flussi di glider.

a ritroso". Partendo quindi dall'automa cellulare bisogna cercare elementi che suggeriscano l'implementazione di un qualche modello di calcolo noto.

L'esempio di universalità spontanea più famoso è senza dubbio l'automa cellulare *game-of-life* di Conway [1], di cui abbiamo parlato informalmente nell'introduzione. Lo stesso Conway dimostra che il proprio automa cellulare è computazionalmente universale, una dimostrazione più formale viene successivamente fornita da Durand e Roka [7].

Mostreremo informalmente come applicare a *game-of-life* i risultati ottenuti per *Copper*. Saranno quindi implementate delle porte logiche, fornendo così il necessario per la costruzione di macchine a stati finiti [21]. Il caso in questione è profondamente diverso da quello di Cu, non esiste una rappresentazione esplicita dei cavi, i segnali sono rappresentati da particelle che viaggiano con una direzione che dipende dalla loro forma.

Per prima cosa presenteremo una lista di oggetti che useremo per comporre le porte logiche.

Glider : è la rappresentazione di una particella in movimento, come in Cu le particelle possono muoversi in quattro direzioni diverse, ma in questo caso le direzioni sono diagonali. Un *glider* si muove di una casella in orizzontale e di una in verticale ogni quattro generazioni (figura 4.10).

Gosper p46 gun : si tratta di un cannone che emette un glider ogni 46 generazioni (figura 4.11). Il motivo per cui è stato scelto questo e non altri cannoni con periodo minore è dovuto alla necessità di avere glider sufficientemente distanti l'uno dall'altro per consentire a due "cavi" di attraversarsi.

Eater : si tratta di un oggetto statico. Sotto precise condizioni di sincronizzazione, ha la particolare proprietà di distruggere i glider che collidono con lui e successivamente tornare nello stato iniziale senza "sporcare" altre cellule (figura 4.12).

Crash : sotto precise condizioni di sincronizzazione, due glider ortogonali che si scontrano si distruggono completamente senza lasciare cellule vive (figura 4.13). Questo oggetto corrisponde di per sé ad un pattern vuoto con opportune condizioni di sincronizzazione.

Duplicator : è un oggetto statico composto da un blocco di cellule vive due per due. Sotto precise condizioni di sincronizzazione, quando due glider si scontrano nei suoi pressi, se provengono da direzioni sfasate di 90°, allora vengono assorbiti e un nuovo glider viene emesso nella direzione di uno dei

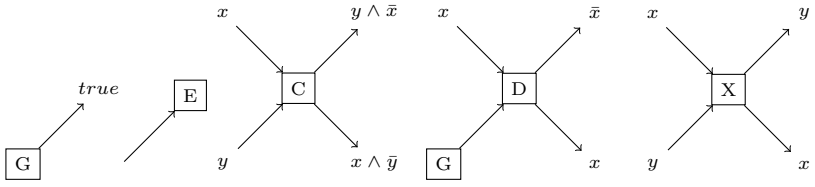


Figura 4.16: Porte logiche corrispondenti agli oggetti definiti: Gosper p46 gun (a), eater (b), crash (c), duplicator (d), crossing (e).

due (quale dipende dal punto in cui si incontrano rispetto alla posizione del duplicator, figura 4.14)[5].

Crossing : due "cavi" ovvero due flussi di glider, con direzione ortogonale, possono attraversarsi senza influenzarsi a vicenda e senza bisogno di costruzioni particolari. I glider di uno stream semplicemente passano nello spazio lasciato vuoto fra due glider successivi dell'altro stream (figure 4.15). Questo è il motivo per cui è stato scelto un cannone con periodo 46. Anche questo oggetto, come il crash, corrisponde ad un pattern vuoto con opportune condizioni di sincronizzazione.

In una configurazione possiamo quindi considerare come cavo ogni tragitto che possa essere percorso da un glider. Su ogni cavo viaggiano valori booleani con periodo fisso. La presenza di un glider corrisponde ad un valore booleano positivo, la sua assenza ad uno negativo. Da questo punto di vista un cavo collegato direttamente ad un gun (G in 4.16a) conterrà sempre valori positivi. Un eater può essere visto come la terminazione di un cavo (E in 4.16b). Un crash corrisponde ad una porta con due ingressi, x e y, e due uscite, una corrispondente a $x \wedge \bar{y}$ e l'altra $y \wedge \bar{x}$ (C in 4.16c). Considereremo per le nostre costruzioni solo duplicator (D in 4.16d) in cui uno dei due cavi è collegato ad un gun e l'altro è quello i cui glider "sopravvivono" in caso di scontro. In questo caso la coppia gun e duplicator è una porta con un ingresso x e due uscite, una x e l'altra $\bar{x}$. Infine il crossing (X in 4.16e) corrisponde semplicemente ad un intreccio di cavi che non si influenzano, una porta quindi con due ingressi e due uscite identiche.

A questo punto si tratta solo di comporre in maniera adeguata queste porte per ottenere le tradizionali AND, OR e NOT. Mostriamo anche come ottenere la duplicazione di un glider e l'attraversamento di due cavi aventi stessa direzione (anziché l'una perpendicolare all'altra). Data la costruzione schematica in figura 4.17, la verifica dei valori di output è banale applicando le formule precedenti[6].

[5] Il glider emesso è in realtà lievemente traslato rispetto a quello ricevuto e ha sincronizzazione differente, ma per i fini di questa trattazione non ci soffermeremo su questo genere di dettagli.

[6] La costruzione in figura 4.17d sembra eccessivamente complicata. In realtà il fatto che ogni glider attraversi esattamente due duplicator è ottimale per la sincronizzazione.

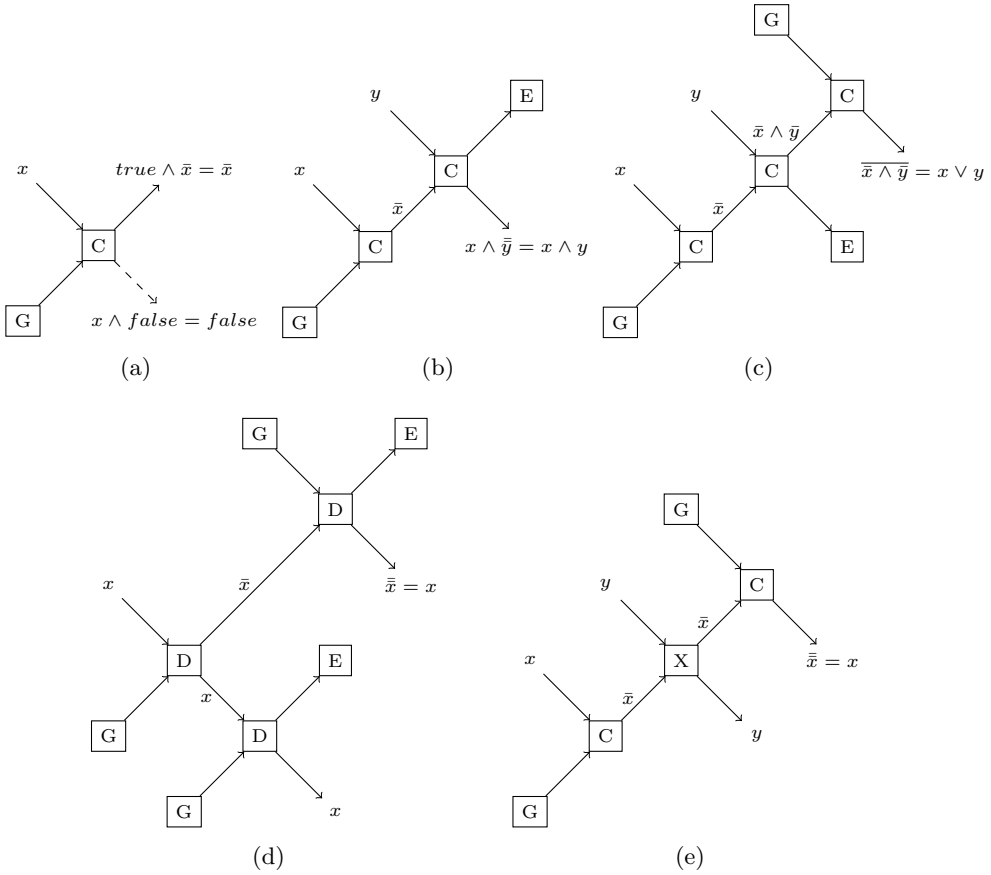


Figura 4.17: Porte logiche NOT (a), AND (b), OR (c); duplicazione di glider (d) e incrocio di cavi (e).

Prendiamo ora in esame gli automi cellulari elementari. Innanzitutto adesso possiamo parlare in maniera più approfondita della loro classificazione. Le definizioni proposte da Wolfram, che sono vaghe e qualitative (vedi definizione 3), sono state successivamente formalizzate da Culik e Yu che hanno anche dimostrato che la loro classificazione è indecidibile [10].

Definizione 17. *(classificazione di Culik-Yu) Un automa cellulare (d, S, N, f) con funzione di transizione globale G appartiene alla classe di indice minore fra quelle di cui soddisfa le proprietà.*

(CY1) *tutte le configurazioni finite evolvono in uno stato quiescente Q, ovvero*

$$\forall c \in \mathcal{F}.\ \exists n \in \mathbb{N}.\ G^n(c) = Q$$

(CY2) *tutte le configurazioni finite sono infine periodiche, ovvero*

$$\forall c \in \mathcal{F}.\ \exists m, n \in \mathbb{N}.\ m \neq n \wedge G^m(c) = G^n(c)$$

(CY3) *esiste un algoritmo che decide se una data configurazione finita appartiene all'orbita di un'altra configurazione finita data, ovvero per ogni $c \in \mathcal{F}$*

$$\{e \in \mathcal{F} \mid \exists n \in \mathbb{N}\ .\ G^n(c) = e\}$$

è un insieme ricorsivo[7]*.*

(CY4) *nessuna restrizione.*

Si noti che la nilpotenza di un automa cellulare è condizione sufficiente ma non necessaria per la sua appartenenza alla classe uno di Culik-Yu. È infatti possibile per un automa *CY1* che esista una configurazione non finita che non evolve nello stato quiescente Q. Uno di questi automi cellulari è presentato nell'esempio 5.

Una nota congettura di Wolfram afferma che alcuni, se non tutti, gli automi cellulari della quarta classe sarebbero computazionalmente universali[25]. Questa congettura si è dimostrata fondata grazie alla dimostrazione di universalità della regola 110 ad opera di Cook e Wolfram stesso [3][26].

La dimostrazione è molto complicata e fornire anche solo una breve traccia sarebbe molto complesso, per cui ci limiteremo a dare qualche indicazione. Quella che viene mostrata è una tecnica per la simulazione di una sottofamiglia dei tag system ciclici computazionalmente universale. La costruzione fa uso di

[7] L'orbita di una configurazione finita contiene solo configurazioni finite, quindi non è necessario in realtà specificare che e debba esserlo.

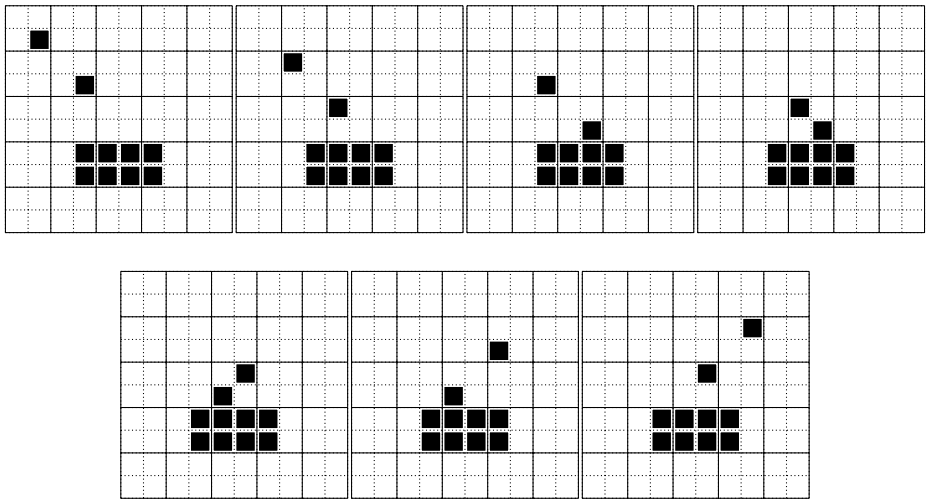


Figura 4.18: Urto di una particella su di un muro. Le configurazioni sono quattro, corrispondenti alla prima, la terza, la quinta e l'ultima figura; gli aggiornamenti sono tre, ognuno diviso in due fasi, la prima sulla base del partizionamento definito dalle righe tratteggiate, la seconda su quello definito da righe continue.

18 tipi differenti di particelle e 23 tipi di collisione. Le particelle sono raggruppate per codificare informazione e le interazioni desiderate sono costruite a partire dalle classi di collisioni osservate.

4.6 Caso invertibile

Abbiamo già messo in luce l'importanza degli automi cellulari invertibili, dovuta principalmente a legami con il mondo fisico. È una naturale domandarsi se esistano automi cellulari invertibili che siano anche computazionalmente universali. Nessuno di quelli visti finora verifica entrambe le proprietà. Si può dimostrare banalmente, infatti le regole di aggiornamento locale degli automi cellulari universali presentati non sono bilanciate (teorema 4).

Il primo risultato importante in questo senso è quello di Toffoli, che ha dimostrato che ogni automa cellulare di dimensione d può essere simulato da un automa cellulare invertibile di dimensione $d+1$ [24]. Questo, assieme all'esistenza di automi cellulari universali di ogni dimensione risolve il problema, tranne che nel caso $d = 1$. Fortunatamente è stato dimostrato da Morita e Harao, attraverso la simulazione diretta di un modello di calcolo invertibile, che esistono automi cellulari unidimensionali invertibili che sono computazionalmente universali.

Presentiamo rapidamente le idee che sono alla base di un automa cellulare bidimensionale piuttosto semplice che è invertibile e computazionalmente universale:

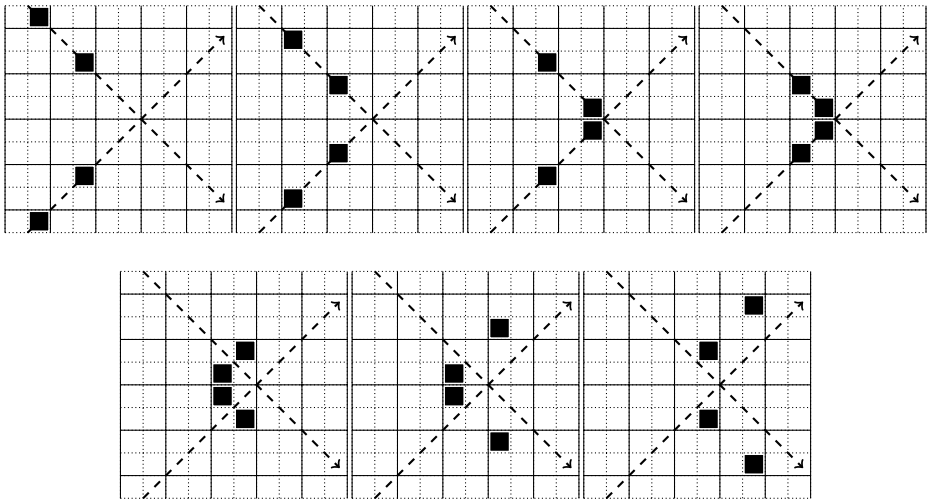
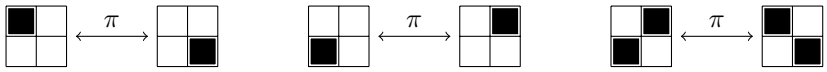


Figura 4.19: Urto di di due particelle fra di loro. Le configurazioni sono quattro, corrispondenti alla prima, la terza, la quinta e l'ultima figura; gli aggiornamenti sono tre, ognuno diviso in due fasi, la prima sulla base del partizionamento definito dalle righe tratteggiate, la seconda su quello definito da righe continue. Le frecce tratteggiate diagonali rappresentano la traiettoria che ciascuna palla seguirebbe in assenza di urti.

il *billiard ball model* [15]. Il nome deriva dal modello di calcolo simulato (*billiard ball computer*), nel quale i segnali sono rappresentati da palle da biliardo e la computazione avviene per mezzo di urti sia fra le palle stesse, sia con le pareti.

L'automa cellulare utilizza il vicinato di Margolus, che come abbiamo visto garantisce l'invertibilità. L'insieme degli stati è $S = \{0, 1\}$ rappresentati come sempre da quadrati bianchi (□) o neri (■). La funzione $\pi : S^4 \longrightarrow S^4$ su cui si basa la funzione di aggiornamento è una rotazione in alcuni casi, l'identità negli altri. Il valore di π nei casi in cui non coincide con l'identità sono i seguenti:



Si noti che π definisce una permutazione. Chiamiamo *particella* l'unico stato 1 in un blocco due per due. Osserviamo che, come nell'esempio 6, le particelle si propagano in diagonale. Una palla da biliardo è rappresentata da una coppia di particelle che si muovono nella stessa direzione secondo la stessa traiettoria. Il diametro di una palle è dato dalla distanza delle due particelle. Per i nostri scopi saranno sufficienti quelle di diametro minimo, in cui le due particelle distano due cellule in diagonale (lo spazio percorso da una particella da una generazione alla successiva). Le pareti sulle quali le palle rimbalzano sono rappresentate da linee di cellule in stato 1 (vedi figura 4.18). La posizione delle pareti deve essere tale che rimangano immobili. Per questo è necessario che in nessuna delle due partizioni (linee continue e linee tratteggiate) sia possibile che la cellula di un muro si trovi in un blocco con tre cellule in stato 0. I blocchi con due cellule in

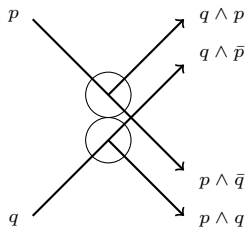


Figura 4.20: Caratterizzazione booleana dell'urto di due palle da biliardo.

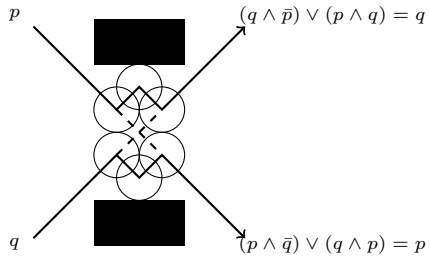


Figura 4.21: Incrocio di due cavi. La linea tratteggiata corrisponde alla traiettoria che si verifica in caso di assenza di urti.

stato 1 servono a simulare gli urti delle palle, sia fra di loro, sia con le pareti[8] (figure 4.19 e 4.18).

Come per *game-of-life* possiamo considerare le potenziali traiettorie delle palle come cavi. Al solito decideremo di codificare una palla come un segnale contenente un valore positivo, la sua assenza come un valore negativo. Sfruttando le regole relative agli urti è possibile costruire delle porte logiche. Cominciamo dando una caratterizzazione booleana all'urto di due palle (figura 4.20).

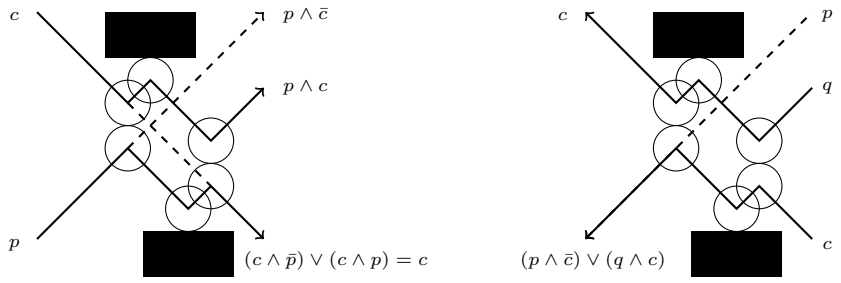


Figura 4.22: Implementazione di uno switch. La linea tratteggiata corrisponde alla traiettoria che si verifica in caso di assenza di urti. Può essere usata sia nel verso principale (a) per selezionare un percorso, che nell'inverso (b) per selezionare un input. La correttezza della formula per l'inverso vale solo se l'input non selezionato è nullo.

[8]Si noti che la scelta di usare due particelle per rappresentare una palla è dovuta proprio al comportamento dei blocchi con due cellule in stato 1. Grazie ad esso infatti, la traiettoria di una palla viene modificata di 90° quando questa rimbalza su una parete. Se una particella solitaria rimbalzasse su un muro tornerebbe invece indietro seguendo la stessa traiettoria di incidenza.

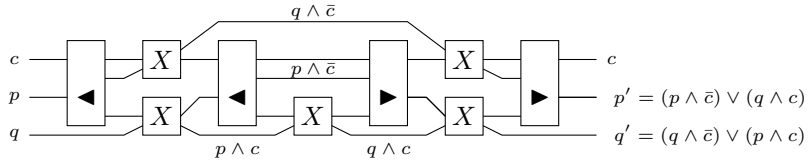


Figura 4.23: *Porta di Fredkin* implementata per mezzo di due switch (indicati come ◀), due switch inversi (indicati come ▶) e cinque incroci di cavi (indicati come X).

Possiamo sfruttare due muri per simulare l'incrocio di due cavi (figura 4.21). Poiché i segnali possono essere ridiretti a piacere per mezzo di muri, la sincronizzazione è garantita. È facile verificare che lo switch in figura 4.22 funziona come ci si aspetterebbe, dirigendo la palla in input p nella direzione stabilita dal bit di controllo c. Impiegheremo questa porta anche nel verso opposto, per selezionare un segnale fra due. La correttezza di questo uso è garantita se l'ingresso che non viene selezionato è negativo. Infine costruiamo la *porta di Fredkin* che implementa la funzione

$$f(c,p,q) = \begin{cases} (c,p,q) & se\ c = 0 \\ (c,q,p) & se\ c = 1 \end{cases}$$

Questa porta è universale, perciò possiamo usarla per costruire qualunque circuito logico. La sua implementazione richiede quattro switch, gli incroci fra cavi sono ottenuti per mezzo dei muri (vedi figura 4.23). Possiamo concludere che l'automa cellulare *billiard ball model* è computazionalmente universale.

Osserviamo che in questo modo abbiamo costruito anche un automa cellulare computazionalmente universale che conserva la quantità di cellule nello stato 1. In realtà, per ovvi motivi, il *billiar ball model* conserva tutte le quantità del proprio modello fisico di riferimento: massa, energia cinetica, momento angolare.

Capitolo 5

Conclusioni

Visto i concetti fondamentali della teoria degli automi cellulari ci siamo quindi soffermati sulla definizione di universalità computazionale, fornendo una rapida carrellata su come ottenerla sia per dimensione uno che per dimensioni maggiori. Abbiamo parlato dell'universalità sotto costrizioni legate a principi fisici e come fenomeno spontaneo di sistemi semplici.

Per ovvie ragioni, molte questioni importanti sono state escluse da questa trattazione. Abbiamo solo accennato in che modo gli automi cellulari possano essere visti come sistemi dinamici discreti; questo approccio è attualmente alla base sia delle applicazioni degli automi cellulari, sia dello studio del loro potere espressivo [5].

Non abbiamo fatto nessun riferimento alle *Wang tiles*, un sistema formale simile a quello presentato, ma statico, dal cui studio derivano molti risultati di questa teoria. Né abbiamo parlato di automi cellulari per il riconoscimento di linguaggi, che rappresentano un ambito di ricerca classico per questo modello di calcolo.

Anche dal punto di vista dell'universalità, quanto abbiamo presentato finora non è altro che una parte della storia. Esistono infatti numerose definizioni alternative di universalità per quanto riguarda gli automi cellulari. Alcune sono ispirate alla concezione originaria di Von Neumann e seguono il concetto di *universalità di costruzione*. Intuitivamente si cercano configurazioni capaci di costruire ogni altra configurazione finita dell'automa cellulare in questione [4].

La definizione di *universalità intrinseca* è invece una versione più stringente di quella che abbiamo visto, ma esclude la possibilità di paragonare gli automi cellulari ad altri modelli di calcolo. In questo caso, si richiede che un automa possa simulare ogni altro automa cellulare della stessa dimensione. Esistono automi cellulari computazionalmente universali che non sono intrinsecamente universali [21] [20].

La fascinazione verso gli automi cellulari come modello di calcolo trae sicuramente origine dalla profonda differenza con la visione umana di algoritmo e dall'intimo legame con il mondo fisico. Altri formalismi sono evidentemente ispirati dall'idea di un esecutore umano. Presi i dati del problema e un insieme di operazioni elementari, una rappresentazione simbolica dell'operatore agisce sulla memoria in passi successivi e localizzati, finché non riconosce di aver terminato il proprio compito. Anche modelli molto semplici, come la macchina di Turing, si basano in realtà solo su una semplificazione delle operazioni eseguibili e della struttura di memorizzazione, ma l'origine umana rimane palese. Contrariamente, nel caso degli automi cellulari, se si osserva l'evoluzione di una configurazione come il calcolo della soluzione di un problema, sembra quasi di trovarsi di fronte alle tubature idrauliche capaci di parlare il cinese teorizzate da Searle. Il funzionamento di un automa cellulare è infatti tanto distante da quello della mente umana[1] da sfidare la nostra stessa definizione di computazione. L'applicazione dei principi dell'intelligenza artificiale forte, riassumibili come "esistono macchine capaci di pensare" agli automi cellulari e più in generale ai sistemi dinamici modifica considerevolmente la portata dei risultati. Se prendiamo per buoni tali principi infatti, un automa cellulare (e quindi un sistema dinamico) universale, capace di simulare ogni macchina, può di certo comportarsi come una *macchina pensante*. Non solo, secondo molti entusiasti infatti[2], l'universalità sarebbe un fenomeno piuttosto comune in natura. In questo caso, nella nostra ricerca di intelligenza nel mondo inorganico, non dovremmo più guardare solo a computer accuratamente progettati, ma anche a sistemi fisici originatisi spontaneamente.

> CALL HIM by no name, for he had no name. He did not know the meaning of name, or of any other word. He had no language, for he had never come into contact with any other living being in the billions of light-years of space that he had traversed from the far rim of the galaxy, in the billions of years that it had taken him to make that journey. For all he knew or had ever known he was the only living being in the universe.
>
> He had not been born, for there was no other like him. He was a piece of rock a little over a mile in diameter, floating free in space. There are myriads of such small worlds but they are dead rock, inanimate matter. He was aware, and an entity. An accidental combination of atoms into molecules had made him a living being. To our present knowledge such an accident has happened only twice in infinity and eternity; the other such event took place in the primeval ooze of Earth, where carbon atoms formed sentient life that multiplied and evolved.
>
> ...
>
> Call him a thinking rock, a sentient planetoid.
>
> Call him a rogue, in the biological sense of the word rogue: an accidental variation.
>
> Call him a rogue in space.
>
> — Fredric Brown, *Rogue In Space*, 1957

[1] Sebbene sia ispirato in parte al funzionamento del cervello umano.
[2] Primo fra tutti Wolfram stesso.

Bibliografia

[1] Elwyn Ralph Berlekamp, John Horton Conway, , and Richard Kenneth Guy. *Winning Ways for your Mathematical Plays*, volume 4 of *Winning Ways for your Mathematical Plays*, chapter What is Life?, pages 927–961. A K Peters, Ltd, 2004.

[2] Edgar Frank Codd. *Cellular Automata*. Academic Press, New York, New York, 1968.

[3] Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

[4] Marianne Delorme. An introduction to cellular automata. Technical Report 98-37, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 1998.

[5] Jean-Charles Delvenne, Petr Kůrka, and Vincent D. Blondel. *Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, chapter Computational Universality in Symbolic Dynamical Systems, pages 104–115. Springer Berlin Heidelberg, 2005.

[6] Bruno Durand. *Global Properties of Cellular Automata*, volume 3 of *Nonlinear Phenomena and Complex Systems*, chapter Cellular Automata and Complex Systems, pages 1–22. Springer Netherlands, 1999.

[7] Bruno Durand and Zsuzsanna Róka. The game of life: universality revisited. Technical Report 98-01, Laboratoire de l'Informatique du Parallélisme, Normale Supérieure de Lyon, 1998.

[8] Tetsuya Hattori and Shinji Takesue. Additive conserved quantities in discrete-time lattice dynamical systems. *Physica D*, 49(3):295–322, 1991.

[9] Gustav Arnold Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Theory of Computing Systems*, 3(4):320–375, 1969.

[10] Karel Culik II and Sheng Yu. Undecidability of ca classification schemes. *Complex Systems*, 2(2):177–190, 1988.

[11] Alvy Ray Smith III. Simple computation-universal cellular spaces. *Journal of the Association for Computing Machinery*, 18(3):339–353, 1971.

[12] Jarkko Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, 334(1):3–33, 2005.

[13] Jarkko Kari. *Handbook of Natural Computing*, volume 1, chapter Basic Concepts of Cellular Automata, pages 3–24. Springer Berlin Heidelberg, 2012.

[14] Jarkko Kari. Cellular automata. dispense per il corso di Automi Cellulari dell'università di Turku, 2013.

[15] Norman Margolus. Physics-like models of computation. *Physica D*, 10(1-2):81–95, 1984.

[16] Olivier Martin, Andrew Odlyzko, and Stephen Wolfram. Algebraic properties of cellular automata. *Communications in Mathematical Physics*, 93(2):219–258, 1984.

[17] Akira Maruoka and Masayuki Kimura. Condition for injectivity of global maps for tessellation automata. *Information and Control*, 32(2):158–162, 1976.

[18] Edward Forrest Moore. Machine models of self-reproduction. In Richard Bellman, editor, *Mathematical Problems in the Biological Sciences*, volume 14 of *Proceeding of Symposia in applied mathematics*, pages 17–33. American Mathematical Society, 1962.

[19] John Myhill. The converse of moore's garden-of-eden theorem. *Proceedings of the American Mathematical Society*, 14(4):685–686, 1963.

[20] Nicolas Ollinger. Intrinsically universal cellular automata. *Electronic Proceedings in Theoretical Computer Science*, 1:199–204, 2009.

[21] Nicolas Ollinger. *Handbook of Natural Computing*, volume 1, chapter Universalities in Cellular Automata*, pages 189–229. Springer Berlin Heidelberg, 2012.

[22] Zsuzsanna Róka. Simulations between cellular automata on cayley graphs. Technical Report 94-40, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 1994.

[23] Klaus Sutner. *Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, chapter Universality and Cellular Automata, pages 50–59. Springer Berlin Heidelberg, 2005.

[24] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15(2):213–231, 1977.

[25] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D*, 10(1-2):1–35, 1984.

[26] Stephen Wolfram. *A new kind of science*. Wolfram Media Inc., Champaign, Illinois, 2002.