

SOC4J - Self-Organizing Cells for Jason

Luca Santonastasi*

Matteo Fattori†

December 22, 2015

Abstract

Si vuole studiare e realizzare il comportamento degli Automi Cellulari al fine di creare utilizzando dei "passi" univoci una simulazione del problema noto come WireWorld, in cui viene riprodotto il comportamento dei circuiti digitali sotto forma di gate logici (and,or,not..). Oltre al problema citato sopra lo scopo fondamentale dello studio è quello di trovare un modo per sviluppare tramite il paradigma ad Agenti gli Automi Cellulari in modo tale da semplificare lo sviluppo di esempi e simulazioni correlati su di essi.

*761222, luca.santonastasi@studio.unibo.it

†724159, matteo.fattori5@studio.unibo.it

1 Introduzione

Gli Automi Cellulari vengono visti come un'idealizzazione dei sistemi naturali, questi sono rappresentati da un insieme di celle immerse in un ambiente in cui ogni cella ha un certo numero di vicini che ne manipolano il comportamento.

In questa visione preliminare possiamo immaginare ogni cella come il contenitore di uno stato, rappresentato solitamente da un valore intero, che in base alla configurazione iniziale può variare in base al sistema naturale che si vuole simulare, in fase di istanziamento.

Un primo modello matematico che possa assimilare il comportamento di ogni automa cellulare è stato descritto dalla seguente equazione:

$$a_i^{(t+1)} = a_{i-1}^{(t)} + a_{i+1}^{(t)} \mod 2 \quad (1)$$

Questa regola¹ rappresenta il comportamento di ogni cella e come viene computato lo stato all'istante di tempo successivo, in questo caso il modulo vincola ogni singolo componente dell'automa ad avere solamente due stati possibili 1 o 0, cella viva o morta.

Anche se molto semplice dall'equazione si evince che il comportamento di ogni cella dipende unicamente da quello dei suoi vicini e questo comporta la generazione di un comportamento complesso che si osserva quando il sistema è a regime.

¹Stephen Wolfram Cellular Automata

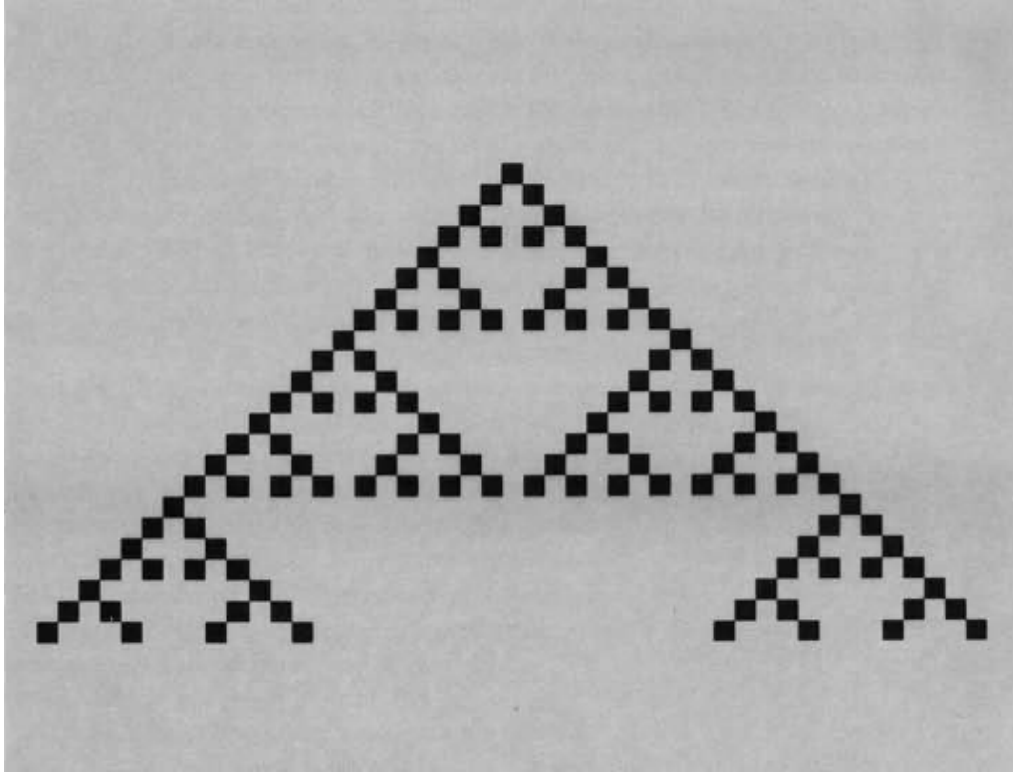


Figure 1: Pattern generato da un'unico automa cellulare con stato iniziale pari a 1.

Il pattern mostrato in Figure 1 viene generato utilizzando come configurazione iniziale un'unica cella con stato pari a 1 e applicando, un numero poco elevato di volte, l'equazione descritta precedentemente.

Se osserviamo il sistema a regime dopo molti step computazionali:

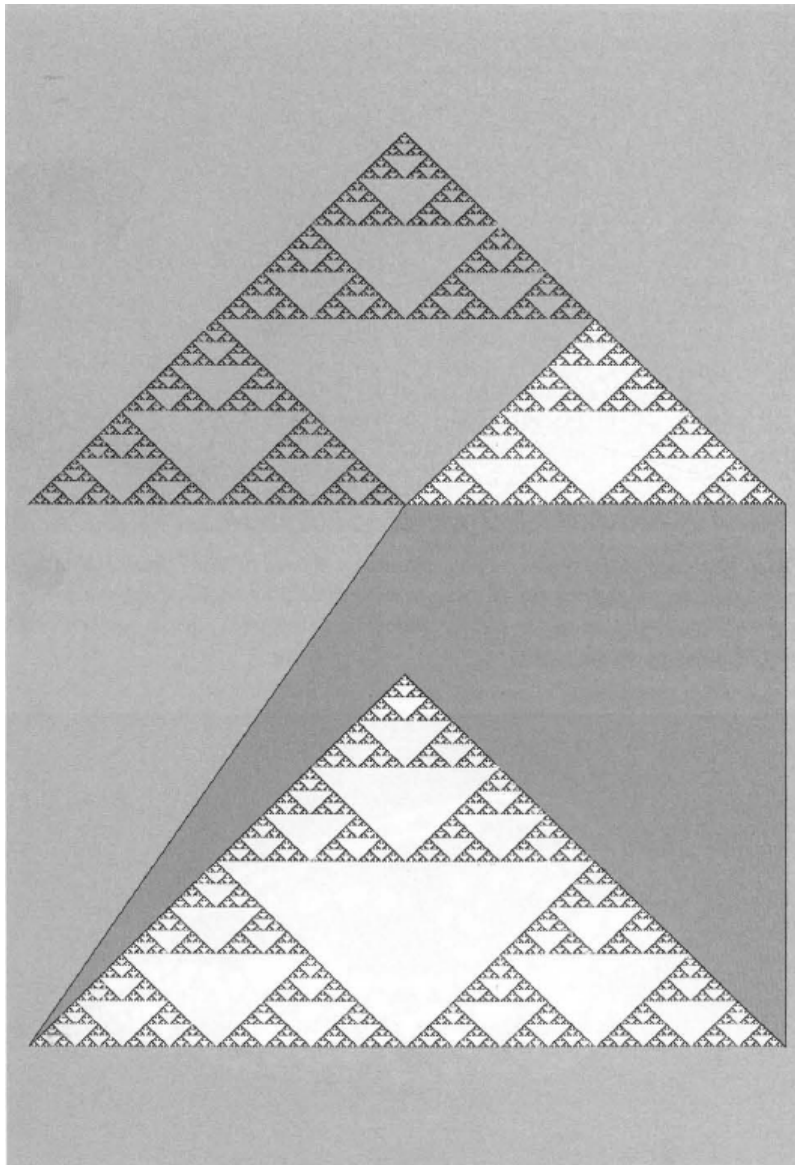


Figure 2: Pattern ripetuto.

Si evince dall'immagine che dopo un certo numero di step computazioni il pattern che gli automi cellulari generano viene ripetuto, questo è possibile grazie all'evoluzione descritta dal comportamento che ogni cella singolarmente assume, specificato dall'equazione 1.

Grazie a questa proprietà di ripetizione del pattern, i sistemi biologici diventano l'esempio fondamentale per spiegare tale comportamento evolutivo, in quanto entrambi sia questi che gli automi cellulari generano ripetutamente strutture equivalenti fra loro anche se il sistema parte utilizzando una configurazione iniziale generata in maniera casuale.

2 Proprietà degli Automi Cellulari

Per distillare alcune proprietà degli automi cellulari ci si basa sul comportamento di ogni singola cella, come descritto dall'equazione 1 ogni cella può avere due stati possibili e come semplificazione consideriamo un numero N finito di celle.

Abbiamo quindi 2^N possibili configurazioni assunte da questo insieme di celle e possiamo rappresentare l'evoluzione globale degli automi cellulari utilizzando un grafo delle transizioni a stati finiti in cui ogni nodo rappresenta una delle possibili 2^N configurazioni e il fatto che due nodi siano connessi significa il passaggio da una configurazione all'altra attraverso l'applicazione dell'equazione 1.

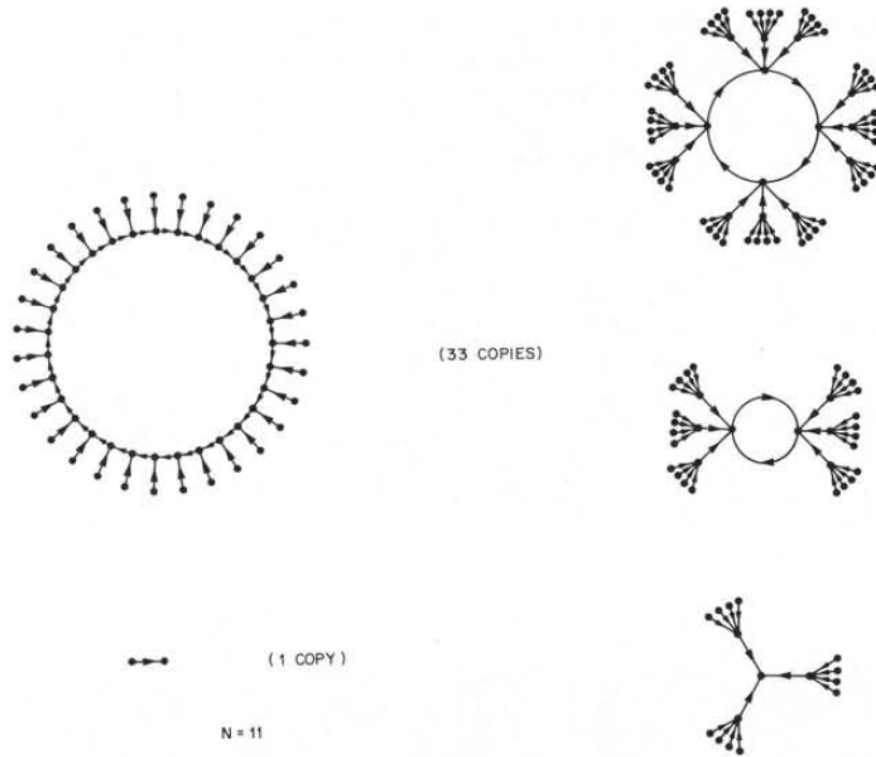


Figure 3: Rappresentazione formale.

Nella Figure 3 viene mostrato un esempio di utilizzo della rappresentazione appena descritta, in particolare l'automa considerato in questo caso ha $N=12$ celle disposte nel grafo lungo una circonferenza e ogni nodo rappresenta una delle 4096 possibili configurazioni.

Una proprietà fondamentale degli automi cellulari che si evince utilizzando questa rappresentazione è la presenza di traiettorie che si ricongiungono man mano che gli step avanzano, in particolare ogni stato ha un'unico successore nel tempo e alcuni/nessun predecessore.

Questa proprietà implica che si ha una perdita di informazioni durante l'evoluzione dell'automa, ovvero in un particolare stato del sistema non è possibile determinare la combinazione di stati precedenti che egli ha perseguito per raggiungere lo stato corrente, quindi l'evoluzione è un processo **irreversibile**.

Notiamo però guardando i grafi in figura che dopo un certo numero di step computazionali l'automa cellulare tende a percorrere un cammino che porta inevitabilmente a degli stati "attrattori" e, come se rimanesse intrappolato in essi, l'automa continua a riprodurre i pattern

proposti in base a tali stati durante tutta l'evoluzione.

3 Generalizzazione degli Automi Cellulari

Dalle discussioni affrontate fino a questo momento, si evince che il comportamento di una cella è basato unicamente sull'equazione 1, ma questa può essere ulteriormente generalizzata, per esempio introducendo una funzione arbitraria che manipola il valore di ogni cella al posto di utilizzare il modulo:

$$a_i^{(t+1)} = F(a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}) \quad (2)$$

Dato che ogni cella ha al più 3 vicini ed ognuno di essi può assumere 2 stati (compresa la cella considerata), questa funzione² assegna un "rule number" ad ogniuna delle 256 possibili configurazioni:

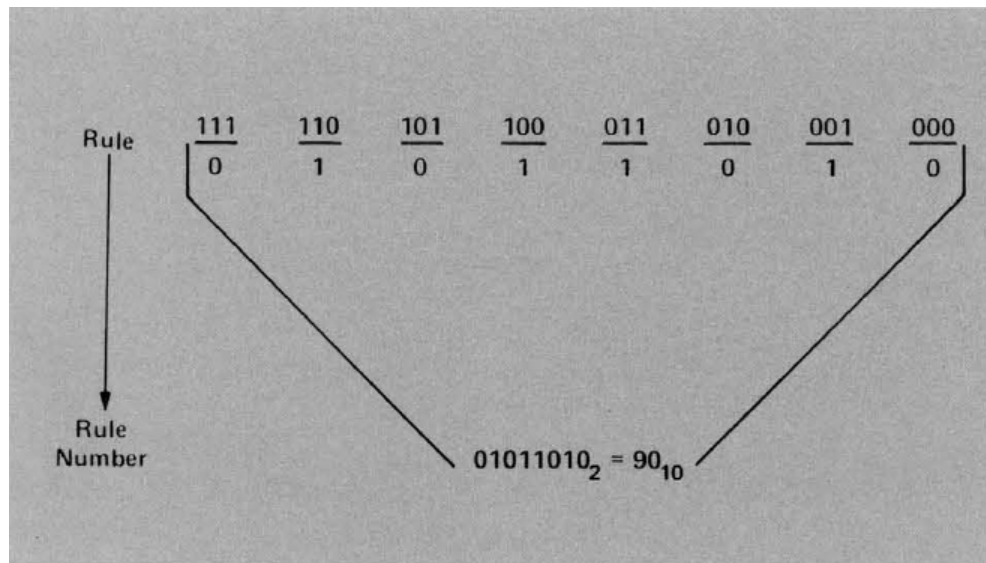


Figure 4: Regola 90.

In questo caso in figura è mostrata la "Rule 90" che rappresenta il "rule number" dell'equazione 1.

²Stephen Wolfram Cellular Automata

Questa prima generalizzazione è solamente parziale in quanto il comportamento di ogni cella dipende unicamente dai vicini "più stretti" ad essa e il valore di ogni cella è vincolato a 0 e 1, infatti possiamo pensare ad una generalizzazione più sofisticata degli automi cellulari introducendo il concetto di raggio r e un numero k arbitrario di valori assunti come stato dalle singole celle.

Quindi l'equazione 2 viene adattata come segue:

$$a_i^{(t+1)} = F(a_{i-r}^{(t)}, \dots, a_{i+r}^{(t)}) \quad (3)$$

Da cui ricaviamo che il numero di "Rule" differenti dato k e r cresce al variare di $k^{k^{2r+1}}$. In base a queste regole è stato possibile distinguere tra 4 classi diverse di automi cellulari caratterizzate come segue:

- Automi in cui i pattern sono generalmente stabilizzati in omogeneità.
- Automi in cui pattern evolvono in strutture prevalentemente stabili o oscillatorie.
- Automi in cui pattern evolvono in maniera apparentemente caotica.
- Automi in cui pattern diventano estremamente complessi e possono durare per lungo tempo usando strutture locali stabili.

Come si può notare dalle descrizioni, questa classificazione degli automi cellulari è basata unicamente sul loro differente comportamento evolutivo.

Concludiamo questa prima analisi sugli automi cellulari specificando che oltre alla rappresentazione dei sistemi naturali, questo modello ispirato alla natura può riprodurre anche situazioni critiche e può essere utilizzato per risolvere problemi di auto-organizzazione come ad esempio la gestione del traffico o la simulazione di modelli reali (la Macchina di Von Neumann, circuiti elettrici), inoltre abilita l'implementatore del modello a pensare in maniera concorrente in quanto è possibile immaginare un modello di interazione per questo tipo di sistemi basato sul paradigma ad agenti.

4 Gli Automi Cellulari ad Agenti

Dal modello degli automi cellulari studiato finora si capisce che ogni cella può avere un certo grado di autonomia e, più nel dettaglio, è in grado di decidere autonomamente in un certo istante di tempo t quale stato assumere all'istante $t+1$.

In questa visione l'autonomia è legata unicamente all'evoluzione di ogni cella che essendo situata in una locazione precisa all'interno del sistema ci abilita ad immaginare un comportamento puramente reattivo legato all'unico scopo che ogni cella nella scacchiera di un automa cellulare ha per tutta la vita, evolvere in seguito a informazioni provenienti dall'ambiente reagendo con un'azione pragmatica.

Essendo questo un sistema naturale, il comportamento è determinante per l'auto-organizzazione del sistema e, come nel nostro caso di studio, una cella è in grado *di cambiare il mondo* che la circonda (entro un certo raggio) grazie alla propria evoluzione.

Pertanto, le celle possono essere considerate come *agenti autonomi* in quanto hanno una limitata abilità di percezione dell'ambiente, possono elaborare informazione dall'ambiente e agire di conseguenza con un'azione pragmatica e infine non c'è un leader che dice loro come agire³.

Basandosi sul paradigma ad agenti quindi possiamo pensare di associare ad ogni cella un'unità fondamentale del paradigma, un'agente.

Dal modello precedentemente studiato si capisce che il comportamento di una cella dipende da quelle vicine. Quindi abbiamo bisogno di avere agenti *con stato* che eseguono una determinata azione a fronte del fatto che le celle vicine si sono evolute, pertanto il modo più semplice è di pensare ogni agente come *intenzionale*, cioè che ha uno *stato mentale* ed è deliberativo (o *goal-directed*) ovvero gli obiettivi vengono rappresentati in maniera esplicita al di fuori del loop di controllo.

Ogni agente, che è un'entità attiva, sarà pertanto responsabile della propria evoluzione che ad ogni step computazionale deve essere coordinata per fare in modo che il sistema com-

³<http://natureofcode.com/book/chapter-6-autonomous-agents/>

plessivo avanzi di uno step alla volta. Un agente può quindi computare lo step successivo se e solo se tutti gli altri agenti presenti nel sistema hanno terminato di computare lo step corrente.

Questo meccanismo è indispensabile ai fini di rispettare la corretta evoluzione proposta precedentemente dal modello degli automi cellulari e rappresenta il problema fondamentale che si deve risolvere, ovvero quello dell'interazione.

L'interazione dei vari componenti di un sistema ne determina la complessità. Si definisce quindi da subito un modello dell'interazione come illustrato in figura 5.

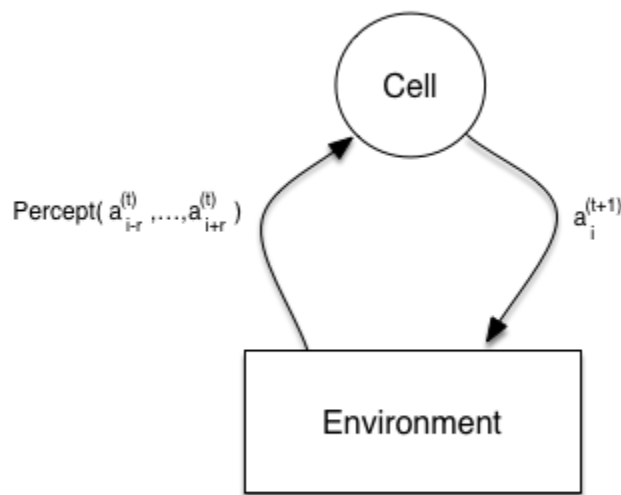


Figure 5: Interazione.

Dalla figura si vede che oltre agli agenti "Cell" nel sistema è presente un'entità attiva "Environment" che rappresenta l'ambiente in cui gli agenti sono situati, per semplicità si può pensare come una griglia di celle in cui ogni cella è un'agente.

Essendo questa nuova entità del sistema parte integrante del paradigma, possiamo dare ad essa il compito di coordinare l'evoluzione delle singole celle.

Avvalendoci del concetto di agente, sappiamo che l'ambiente può interagire attivamente con

ogni cella a fronte dell'avvenimento di opportuni eventi, nel nostro caso una cella mantiene o cambia il proprio comportamento in base allo stato delle celle vicine.

L'ambiente quindi non dovrà far altro che gestire le percezioni di ogni cella a fronte del fatto che una delle celle vicine ha cambiato il proprio stato.

Una volta appreso lo stato delle celle vicine, un agente dovrà evolvere di conseguenza e comunicare il risultato di questo processo all'ambiente.

5 WireWorld

WireWorld è un automa cellulare che può essere utile per simulare elementi di logica elettronica o gates. A discapito della semplicità delle sue regole, WireWorld è Turing-completo⁴.

Il modello degli automi cellulari di Wireworld prevede per ogni cella 4 stati che sono:

- Vuoto;
- Testa;
- Coda;
- Conduttore.

Come in ogni automa cellulare, il tempo procede in passi chiamati *generazioni* in cui ogni cella si comporta come segue:

- Vuoto evolve in Vuoto;
- Testa evolve in Coda;
- Coda evolve in Conduttore;
- Conduttore evolve in Testa se ci sono esattamente 1 o 2 Teste nel vicinato (di raggio 1), altrimenti resta Conduttore.

⁴<https://en.wikipedia.org/wiki/Wireworld>

Come si può vedere in figura 6, WireWorld può essere utile per simulare gate logici utilizzando le regole sopra descritte. Ad esempio possono essere realizzati gate come diodi, AND, OR, XOR, etc.. Da queste basi si può procedere fino ad arrivare a strutture più complesse con ROM, EPROM, etc..

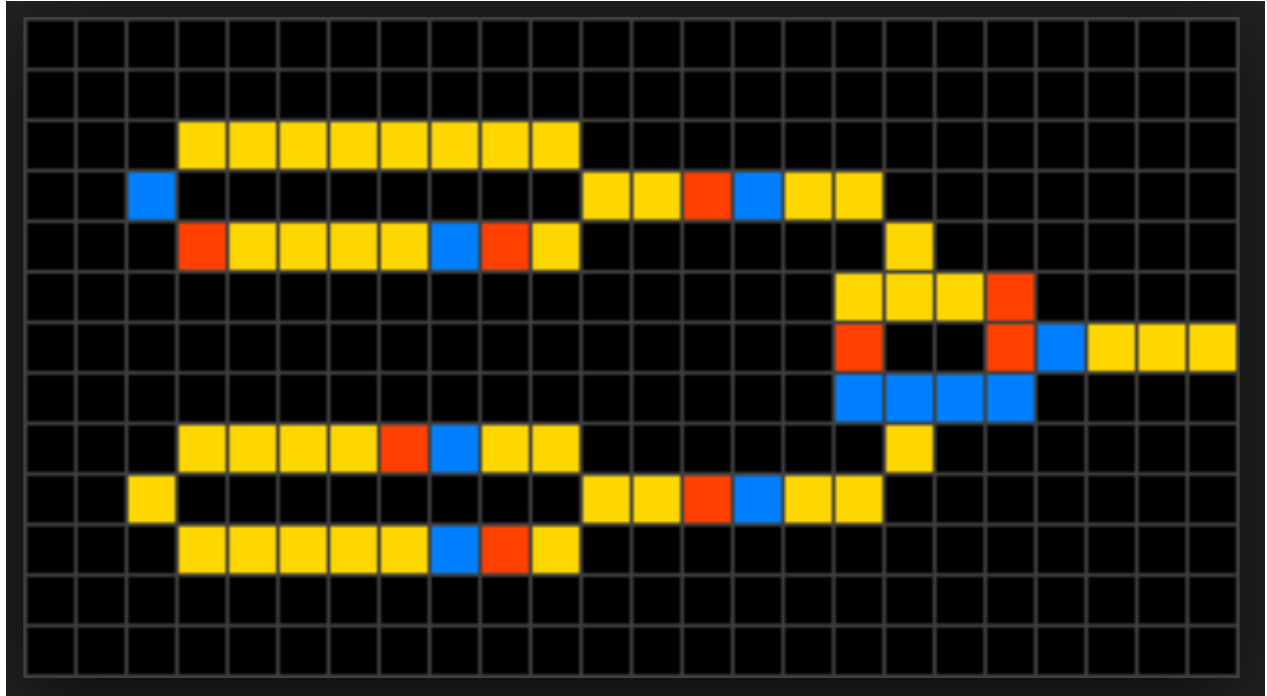


Figure 6: Esempio di XOR-gate con due generatori.

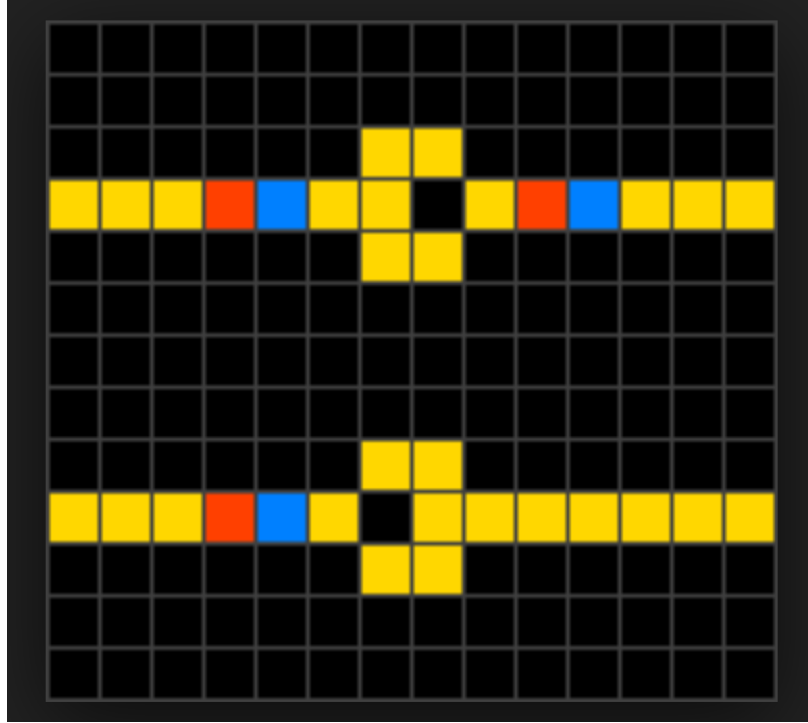


Figure 7: Esempio di diodi

6 Implementazione in *Jason* - WireWorld

Considerando le osservazioni effettuate in precedenza riguardo gli Automi Cellulari ad Agenti, si è deciso di implementare l'esempio proposto di WireWorld utilizzando *Jason* come linguaggio di riferimento.

Jason permette di specificare all'interno di file in formato *asl*, la struttura, interazione e comportamento di un agente.

Si è implementata ogni cella come un agente che reagisce solamente ai cambiamenti dell'ambiente attraverso le percezioni e in base ad esse decide autonomamente se cambiare stato o meno in accordo alle regole di WireWorld.

Per implementare l'ambiente si è sfruttato un implementazione presente nelle librerie *Jason* che permette all'ambiente di attendere che tutte le celle abbiano computato un ciclo di reasoning prima di avanzare allo step successivo (secondo un timeout massimo) cioè un *TimeSteppedEnvironment*⁵, che rappresenta un ambiente che "sincronizza" tutte le azioni

⁵<http://jason.sourceforge.net/api/jason/environment/TimeSteppedEnvironment.html>

degli agenti.

Si è deciso quindi di implementare un modello nel quale viene data una rappresentazione sottoforma di griglia del mondo in cui ogni cella di essa è un agente.

Si sono strutturati i file come illustrato in figura 8.

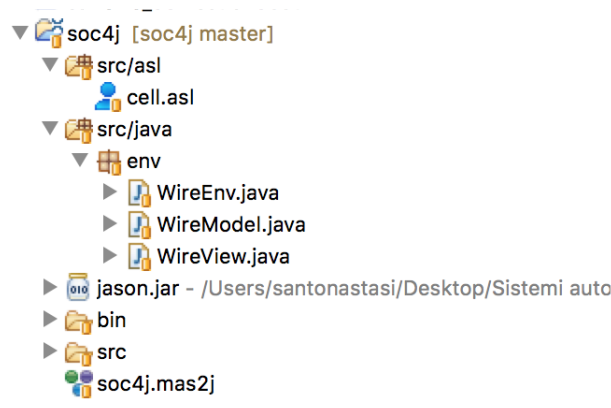


Figure 8: Struttura dei package

Nel file *soc4j.mas2j* si è andati a configurare il sistema come insieme di una griglia 60x60 composta quindi da 3600 agenti, ottenendo così quanto illustrato in Figura 9.

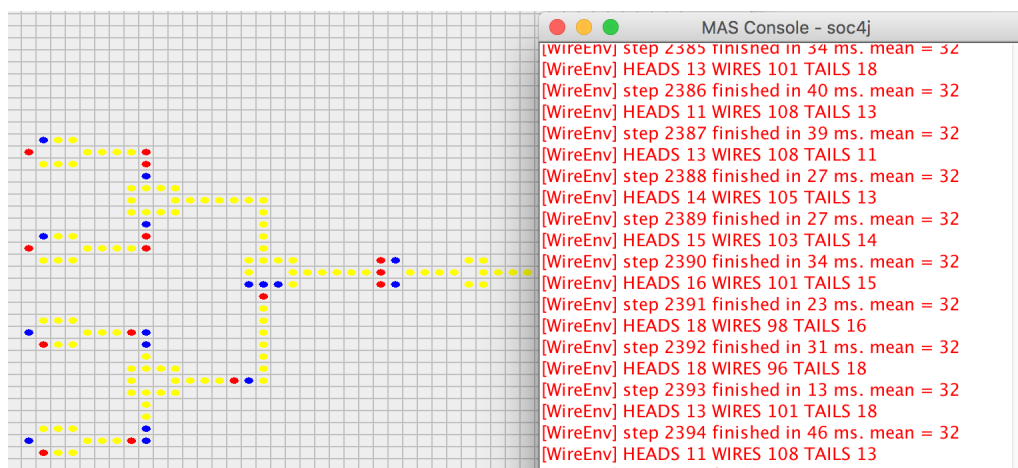


Figure 9: Esempio di esecuzione

Inizialmente il sistema si avvia con tutte le celle della griglia impostate nello stato *Vuoto*. È possibile interagire con l'interfaccia grafica cambiando lo stato degli agenti con un click del mouse.

7 Conclusione

Il modello proposto di WireWorld può essere utilizzato come base per arrivare a simulare persino dinamiche molto più complesse, come illustrato in figura 10.

In allegato al progetto vengono forniti un file *README.md* per configurare e gestire il progetto: <https://apice.unibo.it/xwiki/bin/view/Courses/Sa1516Projects-SOC4J>.

Inoltre, per provare eventuali altri gate oltre quelli visti in precedenza, è possibile visitare il sito <http://karlscherer.com/Wireworld.html>.

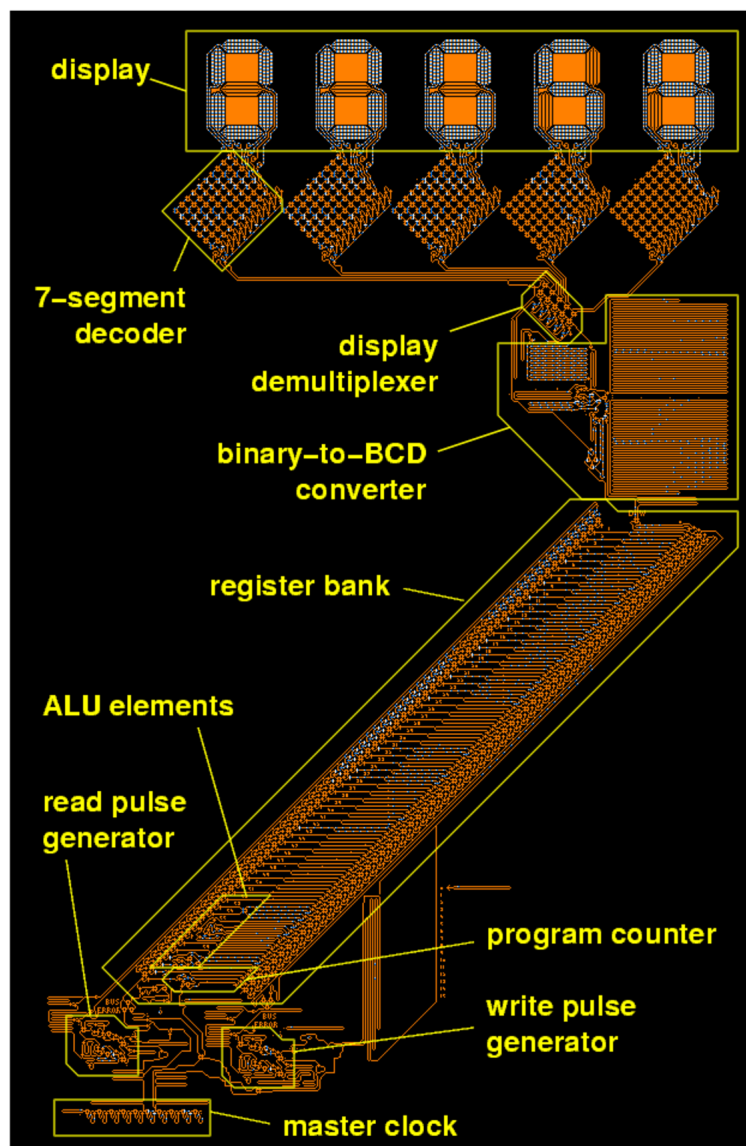


Figure 10: Esempio complesso