An implementation in Agda of Sutner's decision algorithms for injectivity and surjectivity of one-dimensional cellular automata

Silvio Capobianco[1,2] and Niccolò Veltri[1,3] *

[1] Institute of Cybernetics at TUT
[2] Email: silvio@cs.ioc.ee
[3] Email: niccolo@cs.ioc.ee

We discuss an Agda formalization of the algorithms, due to Klaus Sutner [3], that decide injectivity and surjectivity of one-dimensional *cellular automata* by equating these properties with features of cycles in specific finite labeled graphs.

A cellular automaton (briefly, CA) is a quadruple $\mathcal{A} = \langle d, Q, \mathcal{N}, f\rangle$ where the *dimension* $d \geq 1$ is an integer, the *alphabet* Q is finite and contains at least two elements, the *neighborhood* $\mathcal{N} \subseteq \mathbb{Z}^d$ has $N \geq 1$ elements $\nu_1, \ldots, \nu_N$, and the *local update rule* f is a function from Q^s to Q. The *global transition function* of the CA $\mathcal{A}$ on the set $\mathcal{C} = Q^{\mathbb{Z}^d}$ of d-dimensional *configurations* is defined as the synchronous application of the local update rule to each point of $\mathbb{Z}^d$, according to the formula

$$F_{\mathcal{A}}(c) = \lambda\, x : \mathbb{Z}^d.\, f\left(c(x+\nu_1), \ldots, c(x+\nu_N)\right) \;\; \forall c \in \mathcal{C}\,. \tag{1}$$

For $d = 1$ one can always take $\mathcal{N} = \{m, \ldots, m+N-1\}$ for suitable $m \in \mathbb{Z}$, and see f as a function of the word $q_1 \cdots q_N$ corresponding to the concatenation of its arguments $q_1, \ldots, q_N$.

Deriving properties of a CA's global transition function exclusively from its finitary formulation is, in general, undecidable. There are, however, important exceptions: in dimension 1, both injectivity and surjectivity of (1) are decidable. This is due to the *Garden of Eden theorem* ruling that a CA is surjective if and only if distinct configurations only differing in finitely many points have distinct images, and the characterization of injective 1D CA as those that are injective on the set of *periodic* configurations. (See [1] for an introduction to CA theory.)

Recall that the product of two labeled graphs $G_1 = (V_1, E_1, \mathcal{L})$ and $G_2 = (V_2, E_2, \mathcal{L})$ with the same set $\mathcal{L}$ of labels is the graph $G = (V, E, \mathcal{L})$ where $V = V_1 \times V_2$ and $((x_1, x_2), (y_1, y_2)) \in E$ with label $\ell \in \mathcal{L}$ if and only if $(x_1, y_1) \in E_1$ and $(x_2, y_2) \in E_2$ both witl label ℓ.

For $N \geq 2$, the *de Bruijn graph* of order N on the alphabet Q is the graph $G = (V, E)$ where $V = Q^{N-1}$ and $(u, v) \in E$ if and only if $u = xw$ and $y = wy$ for suitable $x, y \in Q$ and $w \in Q^{N-2}$. If $\mathcal{A} = \langle 1, Q, \mathcal{N}, f\rangle$ is a 1D CA with alphabet Q and neighborhood $\mathcal{N} = \{m, \ldots, m+N-1\}$, we can label $xwy \in Q^N$ with $f(xwy) \in Q$: we call the *Sutner graph* of the CA the product of the labeled graph so obtained with itself. By our discussion above, calling *diagonal* the subgraph generated by the pairs (w, w) with $w \in Q^{N-1}$, the following hold [3]:

1. A one-dimensional cellular automaton is injective if and only if no cycle in its Sutner graph touches a node outside the diagonal.

2. A one-dimensional cellular automaton is surjective if and only if no cycle in its Sutner graph joins the diagonal with the outside.

*This research was supported by the ERDF funded projects EXCS and Coinduction, the Estonian Ministry of Education and Research institutional research grant IUT33-13, and the Estonian Science Foundation grant no. 9398.

For the present work, which is still in progress, we use version 2.4.2.3 of the Agda programming language [5] with version 0.9 stable of the standard library. Agda is a dependently-typed functional programming language based on intuitionistic type theory. The standard library is rich in algebra and category theory, but only deals with acyclic graphs. As a side project, we start the development of a small Agda library for graphs, following what is done by [2] in Haskell: such library would, in our aims, include an efficient implementation of the depth-first search algorithm. The choice of Agda is motivated by the greater expressiviness of its type system, which allows not only to implement algorithms, but also to prove their correctness.

Our module `1DCA` is parameterized by three values `q m s : ℕ`. We implement the alphabet as the initial interval `Q` of the natural numbers with `suc q` elements, the neighborhood as an interval of starting point `-m` and size `N = suc (suc s)`, and the local update rule `f` as a function of type `Vec Q N → Q`. We define patterns as vectors on `Q`.

Configurations are implemented as pairs of *streams* (defined coinductively) over a given alphabet, with the convention that the configuration $(\ldots, q_{-2}, q_{-1}, q_0, q_1, \ldots)$ is represented by the ordered pair of streams $((q_{-1}, q_{-2}, ...) \rightarrow| (q_0, q_1, ...))$: the constructor of the type `Conf` of configurations is $\rightarrow|$, rather than the standard comma, to emphasize the role of the point $0 \in \mathbb{Z}$. Equality is defined through bisimilarity in the standard way. Such method allows to define *translations* (i.e., the functions $\sigma_t = \lambda\, c : \mathcal{C}.\, \big(\lambda\, x : \mathbb{Z}^d.\, c(x + t)\big)$) straightforwardly. A pattern `p` is turned into a periodic configuration by a function `periodic = λ p . behind p →| ahead p`, where `ahead p` is the periodic stream obtained by concatenating ω copies of `p`, and `behind p` the one similarly obtained from the reverse of `p`.

In order to update entire configurations by the global transition function, we must be able to update single points by the local update rule: having done this, we can corecursively apply the procedure to both the `ahead` and `behind` component of the configuration. The global transition function is defined *up to translations*: as the latter are bijections, this does not alter injectivity and surjectivity of the CA—which is our present focus.

The de Bruijn graph is implemented via its edge relation. From this, we construct the Sutner graph: two pairs of vectors `(xs , xs')`, `(ys , ys')` of length `suc s` are related if and only if `(xs, ys)` and `(xs', ys')` are both in the de Bruijn relation, and in addition `f` takes the same value on the words of length `N` corresponding to the two pairs. A path in the Sutner graph will then be the reflexive and transitive closure of the Sutner relation; a cycle, a *nonempty* path with the same endpoints; a loop, a cycle of length 1.

As a cycle of length 2 or more always belongs to a strongly connected component, and every strongly connected component with two or more nodes always contains a cycle (possibly a loop), the Sutner conditions for injectivity and surjectivity can be tested by Tarjan's strongly connected components algorithm [4] as follows:

1. The CA is injective if and only if the Sutner graph has no loops outside the diagonal, and no strongly connected components of size 2 or more that contain a point outside the diagonal.

2. The CA is surjective if and only if the Sutner graph has no strongly connected components that contain nodes both inside and outside the diagonal.

For condition 1 we can exploit that every loop in the Sutner graph is on a node of the form (q^{N-1}, p^{N-1}) with $q, p \in Q$.

We can prove in Agda that, if the global transition function is injective, then all cycles in the Sutner graph are contained in the diagonal. The proofs of the converse of the above, and of the corresponding statements for surjectivity, are currently being implemented.

Future work will deal with formalizations of cellular automata theory. In particular, we conjecture that the aforementioned Garden of Eden theorem can be formalized in Agda: which would allow to prove the other direction with regard to surjectivity.

References

[1] Kari, J. (2005) Theory of cellular automata: A survey. *Theor. Comput. Sci.* **334**, 3–33.

[2] King, D. and Launchbury, J. (1994) Lazy depth-first search and linear graph algorithms in Haskell. Glasgow Workshop on Functional Programming 1994. `doi:10.1.1.45.3876`

[3] Sutner, K. (1991) De Bruijn graphs and linear cellular automata. *Complex Systems* **5**, 19–31.

[4] Tarjan, R.E. (1972) Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1(2)**, 146–160.

[5] The Agda Wiki. `wiki.portal.chalmers.se/agda/`