A operação de diferença entre templates de autômatos celulares

Zorandir Soares Junior *
Programa de Pós-Graduação em Engenharia Elétrica e Computação
Universidade Presbiteriana Mackenzie

Pedro Paulo Balbi de Oliveira †
Programa de Pós-Graduação em Engenharia Elétrica e Computação
Universidade Presbiteriana Mackenzie

27 de outubro de 2015

Resumo

Este artigo apresenta a operação de diferença entre templates e sua aplicabilidade. Para isto introduz o que são templates e como o uso de templates pode ajudar na representação de famílias de autômatos celulares. Apresenta também algumas operações aplicáveis aos templates, assim como introduz a operação que gera templates de exceção.

Palavras-chave: *Autômatos celulares, templates, conservabilidade, problema de paridade.*

1 INTRODUÇÃO

Autômatos celulares (ACs) são sistemas dinâmicos tipicamente discretos em tempo, espaço e estados (WOLFRAM, 2002). Os ACs têm a capacidade de por meio de regras de comportamentos locais simples gerar comportamentos globais complexos. O estudo de problemas clássicos de autômatos celulares, como o problema da paridade (BETEL; DE

*zorandir@gmail.com
†pedrob@mackenzie.br

OLIVEIRA; FLOCCHINI, 2013) e o problema da densidade(OLIVEIRA, 2013), podem ajudar a compreender como esse comportamento complexo surge.

O problema de paridade consiste em encontra uma regra que determine se há um número par ou ímpar de células com o estado 1 na configuração inicial. Já o problema de densidade consiste em buscar uma regra que determine qual o estado mais frequente na configuração inicial. Há algumas maneiras de se buscar a solução para esses problemas, sendo a mais básica simplesmente testar todas as regras de uma determinada família de ACs afim de verificar se alguma dessas regras resolve o problema buscado. Essa abordagem faz sentido quando se busca por regras no espaço elementar, já que o mesmo contém apenas 256 regras. Entretanto para famílias de ACs maiores, essa abordagem se mostra bastante ineficiente.

Como estratégia de pesquisa em famílias maiores pode ser utilizado algoritmos genéticos. Algoritmos genéticos, e outras técnicas evolucionarias, são muito utilizadas na investigação do problema de classificação de densidade (WOLZ; OLIVEIRA, 2008).

Outra estratégia na busca de regras que resolvam determinado problema, é a restrição do espaço de busca por regras que apresentem determinada propriedade. Para conseguir representar um subespaço com determinada propriedade sem a necessidade de se enumerar todas as regras desse espaço, pode-se utilizar *templates*. Um template é uma estrutura de dados associada as tabelas de transições de uma família de ACs utilizando variáveis (DE OLIVEIRA; VERARDO, 2014a). A utilização de variáveis torna possível que um template represente um conjunto de regras que podem representar alguma propriedade de autômatos celulares.

Há uma série de operações que podem ser aplicadas aos templates, como expansão e intersecção. No presente artigo é introduzida a operação de diferença entre templates e a operação geradora de templates de exceção. Além disso são mostrados alguns exemplos de aplicação dessas operações.

Na próxima seção são dadas as noções básicas de autômatos celulares. Na Seção 3 é apresentada em mais detalhes o que são templates e suas principais operações. Na Seção 4 são apresentadas a operação de diferença entre templates e a operação geradora de templates de exceção, assim como suas aplicações. E por fim, na Seção 5 é apresentada as considerações finais sobre essas operações e trabalhos futuros.

2 Autômatos Celulares

Autômatos celulares são idealizações matemáticas simples dos sistemas naturais (WOLFRAM, 1994). Eles consistem em um reticulado de campos discretos usualmente

idênticos, onde cada campo pode assumir um conjunto finito de, geralmente, valores inteiros. Os valores dos campos evoluem em tempo discreto de acordo com regras usualmente determinísticas que especificam o valor de cada campo de acordo com os campos das vizinhanças (WOLFRAM, 1994).

As células de um autômato celular podem apresentar k estados. Esses estados são representados por valores inteiros no intervalo $[0, k-1]$. O estado de uma célula pode ser modificado pelas funções locais, que são o conjunto de regras que determinam o novo valor de uma célula baseado em seu estado atual e nos estados das células adjacentes. Para que as funções locais atualizem os valores de uma célula, é necessário que um raio r seja definido. Esse raio r representa o número de células adjacentes que serão analisadas em cada direção pelas funções locais.

Uma família, ou espaço, de autômatos celulares é definida pelo raio r e pelo número de estados k. Autômatos celulares unidimensionais de raio $r = 1$ e $k = 2$ são conhecidos como a família dos autômatos celulares elementares.

O número de regras de um espaço é dado pela Eq. (1):

$$k^{k^{2r+1}} \tag{1}$$

É fácil perceber que qualquer modificação nas variáveis k e r geram famílias com um total de regras muito grande. Uma boa estratégia para lidar com esse problema é a utilização de propriedades estáticas, que são propriedades obtidas através de restrições aplicadas à tabela de transições. Ao utilizar propriedades estáticas, pode-se restringir bastante o espaço de busca original. E com templates é possível representar um conjunto de regras com determinada propriedade estática.

Entretanto, para poder explicar o funcionamento dos templates e suas operações é interessante compreender os detalhes das propriedades de conservabilidade de estados, simetria interna e das regras invariantes a troca de cor, pois essas propriedades serão usadas como exemplo posteriormente.

2.1 Conservabilidade de Estados

Conservabilidade de estados é uma propriedade estática que determina que a soma dos estados de um determinado autômato celular não deve se alterar durante a evolução espaço-temporal, independente da configuração inicial.

De acordo com Boccara e Fukś (2002), um AC é conservativo quando cada uma de suas regras locais f de vizinhança $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ respeita as condições descritas na

Eq. (2).

$$f(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = \alpha_0 + (\sum_{i=0}^{n-2} f(0_0, 0_1, \ldots, 0_i, \alpha_1, \alpha_2, \ldots, \alpha_{n-1}) - f(0_0, 0_1, \ldots, 0_i, \alpha_0, \alpha_1, \ldots, \alpha_{n-i-1})) \quad (2)$$

2.2 Simetria Interna

A propriedade de simetria interna é representada pelo número de transições que permanecem iguais após a aplicação de uma transformação que nos leve a uma regra de equivalência dinâmica. Logo, faz-se necessário compreender o funcionamento das transformações de regras e classes de equivalência dinâmica para entender o que como funciona a simetria interna. As explicações a seguir são válidas para regras binárias, apesar de ser possível sua generalização para k estados.

Dado uma tabela de transições de um AC, existem três transformações que podem ser empregadas e que resultam em ACs com comportamentos dinâmicos equivalentes: *reflexão*, *conjugação* e *composição*. A reflexão é a transformação obtida ao refletir os bits das vizinhanças da tabela de transições. A conjugação é obtida ao inverter todos os estados das células da tabela de transições. Já a composição é a transformação obtida ao se efetuar a reflexão e a conjugação, independente da ordem.

Para exemplificar essas transformações e as equivalências dinâmicas considere a tabela de transição da regra 60, ilustrada pela Figura 1. Ao se aplicar a transformação por reflexão na regra 60 obtemos a regra 102 do espaço elementar, ilustrada pela Figura 2.

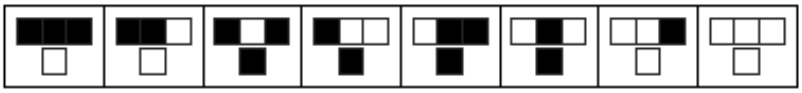

Figura 1: Tabela de transições da regra 60 do espaço elementar.

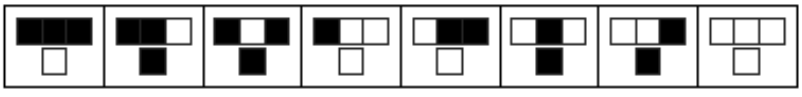

Figura 2: Tabela de transições da regra 102 do espaço elementar, obtida através da transformação de reflexão aplicada na tabela de transições da regra 60.

A simetria interna é representada pelo número de vizinhanças que permanecem iguais após aplicada uma determinada transformação. Exemplificando, a regra 60 dos ACs

elementares tem um valor de simetria interna por reflexão igual a 4, pois compartilha as transições de estado $((1,1,1),0)$, $((1,0,1),1)$, $((0,1,0),1)$ e $((0,0,0),0)$ com a regra resultante de sua transformação por reflexão, a regra 102.

2.3 Invariância a troca de cor

É considerado que um AC tem a propriedade de invariância a troca de cor se ele for invariante a aplicação de permutações nos estados das células de sua tabela de transição (SALO; TÖRMÄ, 2013). No caso binário só há uma permutação possível, que pode ser descrita como $\{0 \to 1, 1 \to 0\}$. Para autômatos celulares não binários há um número maior de permutações possíveis.

A propriedade de invariância a troca de cores está diretamente ligada a transformação por conjugação, sendo que as regras invariantes a troca de cor são as regras que possuem máxima simetria interna por conjugação.

3 Templates

Template é uma generalização de tabelas de transições de ACs por meio de variáveis. Um único template pode representar um conjunto de regras que podem apresentar determinada propriedade. Os templates foram criados por De Oliveira e Verardo (2014a) e implementada como um algoritmo na linguagem do software *Wolfram Mathematica* (WOLFRAM RESEARCH, 2015), atualmente disponíveis na biblioteca *open source CA-Templates* (VERARDO; DE OLIVEIRA, 2015) no GitHub.

Formalmente, um *template* é uma n-upla formada por k^{2r+1} itens, e cada item i representa uma função $g_i(x_0, x_1, \ldots, x_{k^{2r+1}-1})$. As variáveis x_i podem assumir qualquer estado entre 0 e $k-1$, logo no caso binário x_i pode assumir os valores 0 e 1. É possível limitar os valores possíveis de x_i através da notação $x_i \in C$, onde C é um conjunto representando os possíveis valores de x_i.

Exemplificando, o template do espaço elementar $T_1 = (1,1,1,1,1-x_1,x_2,x_1,0)$ representará todas as regras que tenham na posição 0 (sempre da direita pra esquerda) o estado 0, nas posições 4, 5, 6 e 7 o estado 1, nas posições 1 e 2 qualquer estado no intervalo $[0, k-1]$ e na posição 3 o estado complementar ao valor da posição 1. Deste modo o template T_1 representa o conjunto de autômatos celulares elementares $\{(1,1,1,1,1,0,0,0),(1,1,1,1,0,0,1,0),(1,1,1,1,1,1,0,0),(1,1,1,1,0,1,1,0)\}$, ou em sua forma decimal $\{248, 242, 252, 246\}$. O processo de encontrar todas as regras representadas por um template se chama *expansão*.

Já há implementado na biblioteca *open source CATemplates* (VERARDO; DE OLIVEIRA, 2015) diversas algoritmos geradores de templates que representem regras com determinada propriedade. O template $T_{comp} = (1-x_0, 1-x_4, 1-x_2, x_4, 1-x_1, x_2, x_1, x_0)$ dos ACs elementares, por exemplo, quando expandido gera apenas regras que apresentam máxima simetria interna por composição. Já o template $T_{inv} = (1-x_0, 1-x_1, 1-x_2, 1-x_3, x_3, x_2, x_1, x_0)$ dos AC elementares quando expandido gera apenas regras com a propriedade de invariância a troca de cor.

Além da possibilidade de expansão de templates, já foi desenvolvido por De Oliveira e Verardo (2014b) um algoritmo que permite gerar templates que representem a intersecção entre dois sub-espaços de regras representadas por templates. Para exemplificar, considere que se esteja a busca de um conjunto de regras que sejam ao mesmo tempo invariantes a troca de cor e apresentem máxima simetria interna por composição. Para encontrar esse conjunto de regras basta-se realizar dois passos.

O primeiro passo é igualar os dois templates que representam as propriedades desejadas, gerando assim um sistema de equações. Ao solucionar esse sistema de equações serão gerados os relacionamentos entre as variáveis, que quando aplicados aos templates recebidos como entrada, resultará no template que representa a intersecção.

Como exemplo, considere o template T_{comp}, que representa as regras com máxima simetria por composição, e o template T_{inv}, que representa as regras invariantes a troca de cor, ambos com $k = 2$ e $r = 1$. Para encontrar a intersecção entre esses dois template o primeiro passo consiste em igualar os templates gerando o sistema de equações representado por Eq. 3:

$$\begin{cases} 1-x_0 &= 1-x_0 \\ 1-x_4 &= 1-x_1 \\ 1-x_2 &= 1-x_2 \\ x_4 &= 1-x_3 \\ 1-x_1 &= x_3 \\ x_2 &= x_2 \\ x_1 &= x_1 \\ x_0 &= x_0 \end{cases} \tag{3}$$

Após isso, esse sistema é resolvido obtendo o conjunto solução $S = \{x_3 = 1 - x1, x_4 = x_1\}$. Esse conjunto solução é então aplicado como um conjunto se substituições aos templates recebidos como parâmetro. Caso os templates recebidos como parâmetro não apresentem restrições de variáveis, o resultado de ambas as substituições podem ser escolhidos, finalizando o processo e resultando no template $(1-x_0, 1-x_1, 1-x_2, x_1, 1-x_1, x_2, x_1, x_0)$ que representa a intersecção dos templates T_{comp} e T_{inv}, e por consequência representa todos as regras do espaço elementar que são ao mesmo tempo invariantes a

troca de cor e tenham máxima simetria por composição.

Caso pelo menos um dos templates apresente restrições de variáveis, uma segunda etapa do algoritmo deve ser feita. Nessa segunda etapa as expressões que restringem as variáveis são extraídas, gerando um conjunto que é então traduzido para um sistema de equações. Esse sistema de equações é resolvido, e seu conjunto solução é aplicado como um conjunto de substituições nos templates passados como parâmetro.

Inspirada na operação de intersecção, é introduzido neste artigo a operação de diferença entre templates.

4 Diferença entre templates e Templates de Exceção

A operação de diferença recebe dois templates como parâmetro, que chamaremos de $T_{minuendo}$ e $T_{subtraendo}$. Essa operação tem como resultado um conjunto de templates que representa todas as regras representadas pelo template $T_{minuendo}$ que não são representadas também pelo $T_{subtraendo}$.

A operação de diferença apresenta um processo com diversas etapas. A primeira etapa consiste em fazer a intersecção entre os dois templates passados como parâmetro, resultando num template T_i. Caso não haja intersecção entre os dois templates, o resultado da operação de diferença é o próprio $T_{minuendo}$. Caso haja intersecção, o template de intersecção T_i é igualado ao template $T_{minuendo}$, gerando assim combinações lógicas de equações. Então o algoritmo remove as equações tautológicas e aplica uma operação de negação nas equação, que no caso binário consiste em apenas efetuar as permutações $\rho = (0 \to 1, 1 \to 0)$ ao resultado final das equações. Neste momento o algoritmo troca o operador lógico $\wedge$ por $\vee$ e o sistema gerado por esse processo é solucionado gerando assim um conjunto de conjuntos de substituição que devem ser aplicados ao template $T_{minuendo}$. Caso não haja conjuntos de substituições, ou os conjuntos sejam inválido, todas as regras que pertencem a $T_{minuendo}$ também pertencem ao $T_{subtraendo}$ e, por consequência, o algoritmo retorna um conjunto vazio.

Para compreender esse processo, considere o template que representa as regras invariantes a troca de cor $T_{inv} = (1 - x_0, 1 - x_1, 1 - x_2, 1 - x_3, x_3, x_2, x_1, x_0)$ e o template de conservabilidade de estados $T_{con} = (1, 1 + x_2 - x_3, 1 - x_2, 1 - x_1 - x_2, x_3, x_2, x_1, 0)$. O primeiro passo para encontrar a diferença de T_{inv} para T_{con} é fazer a intersecção entre os dois templates, que no caso é $T_{int} = (1, 1 - x_1, 1 - x_2, 1 - x_1 - x_2, x_1 + x_2, x_2, x_1, 0)$. Como há intersecção, o próximo passo é igualar T_{inv} com o T_{int} gerando o sistema de equações

representado pela Eq. 4:

$$\begin{cases} 1 - x_0 &= 1 \\ 1 - x_1 &= 1 - x_1 \\ 1 - x_2 &= 1 - x_2 \\ 1 - x_3 &= 1 - x_1 - x_2 \\ x_3 &= x_1 + x_2 \\ x_2 &= x_2 \\ x_1 &= x_1 \\ x_0 &= 0 \end{cases} \tag{4}$$

Entretanto, esse sistema deve ser representado por meio de combinações lógicas de equações, como se pode ver na Eq. 5:

$$\begin{aligned} 1 - x_0 = 1 \wedge 1 - x_1 = 1 - x_1 \wedge 1 - x_2 = 1 - x_2 \wedge \\ 1 - x_3 = 1 - x_1 - x_2 \wedge x_3 = x_1 + x_2 \wedge x_2 = x_2 \wedge x_1 = x_1 \wedge x_0 = 0 \end{aligned} \tag{5}$$

O algoritmo elimina então todas as equações tautológicas e troca todo operadores lógico $\wedge$ por $\vee$ resultando o sistema representado pela Eq. 6:

$$1 - x_0 = 1 \vee 1 - x_3 = 1 - x_1 - x_2 \vee x_3 = x_1 + x_2 \vee x_0 = 0 \tag{6}$$

Por fim, é aplicado a operação de negação nas equações. No caso binário basta efetuar a permutação ρ, que também pode ser feita por meio da função $f(x) = 1 - (x)$. A Eq. (7) representa a combinação lógica de equações resultante dessas operações.

$$1 - x_0 = 1 - 1 \vee 1 - x_3 = 1 - (1 - x_1 - x_2) \vee x_3 = 1 - (x_1 + x_2) \vee x_0 = 1 - 0 \tag{7}$$

Após esses passos a combinação lógica de equações resultante é resolvida gerando o conjunto solução $S = \{\{x_0 \to 1\}, \{x_3 \to -x_1 - x_2 + 1\}\}$. Como S apresenta mais de um conjunto de substituições, ambos os conjuntos devem ser utilizados para realizar as substituições no template $T_{inv} = (1 - x_0, 1 - x_1, 1 - x_2, 1 - x_3, x_3, x_2, x_1, x_0)$. Com isso é obtido o conjunto de templates $\{(0, 1 - x_1, 1 - x_2, 1 - x_3, x_3, x_2, x_1, 1), (1 - x_0, 1 - x_1, 1 - x_2, x_1 + x_2, 1 - x_1 - x_2, x_2, x_1, x_0)\}$

Em diversos casos apenas as etapas descritas até aqui são suficientes para encontrar a diferença entre os dois templates. Mas há casos em que o template $T_{subtraendo}$, que é o T_{con} no exemplo dado, apresenta substituições nas variáveis que levam a regras inválidas. Para melhor compreender esse problema, basta verificar a regra gerada quando se expande o template $(1, 1 - x_1, 1 - x_2, 1 - x_1 - x_2, x_1 + x_2, x_2, x_1, 0)$ atribuindo o valor 1 às variáveis x_1 e x_2. Aos se fazer isto será obtida na posição 3 (da direita para esquerda) o valor 2, que está fora do intervalo inteiro $[0, k - 1]$.

Para contornar esse problema, é necessário que após os primeiros passos da operação de diferença entre templates, também se verifique se há templates de exceção na intersecção entre o $T_{minuendo}$ e o $T_{subtraendo}$, sendo que templates de exceção são os templates que apresentam um conjunto de substituições que levam um template passado como parâmetro a apresentar substituições fora do intervalo $[0, k-1]$.

Para exemplificar, vamos continuar a intersecção entre T_{inv} e T_{con}. Para isso considere o template $T_{int} = (1, 1-x_1, 1-x_2, 1-x_1-x_2, x_1+x_2, x_2, x_1, 0)$, que é a intersecção entre o esses dois templates para $k = 2$. A primeira etapa da operação de diferença entre os templates T_{inv} e o T_{con} ocorre normalmente e encontra os templates complementares $\{(x_7, x_6, x_5, x_4, x_3, x_2, x_1, 1), (x_7, x_6, x_5, x_1+x_2, x_3, x_2, x_1, x_0)\}$. Todavia é trivial perceber que qualquer expansão do template T_{int} que tenha o conjunto de substituições $\{x_1 = 1, x_2 = 1\}$ fará com que a posição 3 e 4 do template apresentem valores que não pertencem ao intervalo $[0, k-1]$. Logo, todos os templates que apresente $\{x_1 = 1, x_2 = 1\}$ são complementares ao template T_{int}. Assim gera-se o template $(x_7, x_6, x_5, x_4, x_3, 1, 1, x_0)$, que é o template de exceção de T_{int}.

Deste modo, toda regra representada pelo template $T_{minuendo}$, que é T_{inv} no exemplo dado, e que também seja representada pelos templates de exceção da intersecção do $T_{minuendo}$ com o $T_{subtraendo}$, o T_{int} no exemplo dado, deve ser representada em algum dos templates do conjunto de templates de diferença. Para que isto o algoritmo que encontra a diferença entre templates pega todos os templates de exceção encontrados, os intersecciona com o $T_{minuendo}$ e os adiciona ao conjunto de templates obtidos pelos primeiros passos da operação de diferença entre os templates. Com isso, no exemplo utilizado até agora, conjunto de templates de diferença resultante está representado pela Eq. 8:

$$\begin{aligned} &\{(0, 1-x_1, 1-x_2, 1-x_3, x_3, x_2, x_1, 1), \\ &(1-x_0, 1-x_1, 1-x_2, x_1+x_2, 1-x_1-x_2, x_2, x_1, x_0), \\ &(1-x_0, 0, 0, 1-x_3, x_3, 1, 1, x_0)\} \end{aligned} \tag{8}$$

O interessante da operação de diferença entre templates é que com ela é possível descobrir respostas para diversas perguntas não triviais, tais como: quais são as regras que apresentam conservabilidade de estados, mas não são apresentam ao mesmo tempo máxima simetria por composição e invariância a troca de cor? Para responder essa pergunta basta gerar o template de regras conservativas, e então subtrair dele a intersecção entre o template de regras com máxima simetria por composição e o template de invariância a troca de cor.

O resultado dessa operação retorna um conjunto de templates que representa todas as regras conservativas, excetuando a regra identidade.

5 Considerações Finais

No presente artigo são descritos os templates de autômatos celulares e apresentadas pela primeira vez a operação de diferença entre templates e a operação que encontra templates de exceção. Ambas operações podem se mostrar valorosas na compreensão do problema de paridade e do problema de densidade.

Também é demonstrado como é possível obter resposta para perguntas não triviais sobre buscas por regras com determinadas propriedades estáticas, por meio de templates, sem depender da utilização de algoritmos genéticos, evitando também buscas e testes extensivos por regras em toda uma família de ACs.

Entretanto a operação de diferença entre templates ainda funciona apenas para famílias com $k = 2$, o que é uma limitação que merece destaque por ser um possível trabalho futuro, já que a generalização dessa operação traria um grande avanço à biblioteca de templates.

Agradecimentos

Os autores agradecem à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) e à CAPES o apoio recebido para o desenvolvimento deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

BETEL, H.; DE OLIVEIRA, P. P. B.; FLOCCHINI, P. Solving the parity problem in one-dimensional cellular automata. *Natural Computing*, v. 12, n. 3, p. 323–337, 2013.

BOCCARA, N.; FUKŚ, H. Number-conserving cellular automaton rules. *Fundamenta Informaticae*, IOS Press, v. 52, n. 1-3, p. 1–13, 2002.

DE OLIVEIRA, P. P. B.; VERARDO, M. Representing families of cellular automata rules. *The Mathematica Journal*, v. 16, n. 8, 2014. Disponível em: <dx.doi.org/doi:10-.3888/tmj.16-8>.

DE OLIVEIRA, P. P. B.; VERARDO, M. Template based representation of cellular automata rules. In: ISOKAWA, T.; IMAI, K.; MATSIU, N.; PEPER, F.; UMEO, H. (Ed.). *20th International Workshop on Cellular Automata and Discrete Complex Systems*. Himeji, Japão, Julho 7-9: [s.n.], 2014. p. 199–204.

OLIVEIRA, P. de. Conceptual connections around density determination in cellular automata. In: KARI, J.; KUTRIB, M.; MALCHER, A. (Ed.). *Cellular Automata*

and Discrete Complex Systems. Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8155). p. 1–14. Disponível em: <http://dx.doi.org/10.1007/978-3-642-40867-0_1>.

SALO, V.; TÖRMÄ, I. Color blind cellular automata. In: *Cellular Automata and Discrete Complex Systems*. [S.l.]: Springer, 2013. p. 139–154.

VERARDO, M.; DE OLIVEIRA, P. P. B. *CATemplates*. [S.l.], 2015. Disponível em: <https://github.com/mverardo/CATemplates>.

WOLFRAM RESEARCH. *Wolfram Mathematica*. 2015. Disponível em: <http://www-.wolfram.com/mathematica/>.

WOLFRAM, S. *Cellular automata and complexity: collected papers*. [S.l.]: Addison-Wesley Reading, 1994.

WOLFRAM, S. *A new kind of science*. [S.l.]: Wolfram media Champaign, 2002.

WOLZ, D.; OLIVEIRA, P. P. D. Very effective evolutionary techniques for searching cellular automata rule spaces. *J. Cellular Automata*, v. 3, n. 4, p. 289–312, 2008.