Verificação de Gerações no Jogo da Vida de Conway

Henrique Nazaré Santos,
hnsantos@dcc.ufmg.br

May 29, 2012

1 Introdução

Autômatos celulares são sistemas determinísticos compostos por um vetor ou matriz de células idênticas. As células têm o comportamento de um autômato de estado finito. Todas células do autômato mudam de estado simultaneamente, de acordo com uma função de transição n, aplicada em intervalos discretos de tempo. A configuração do autômato em cada um dos intervalos é uma geração do mesmo.
O primeiro autômato celular foi proposto por John von Neumann nos anos 40 para modelar o crescimento de cristais. Este autômato, conhecido como Vizinhança de von Neumann, era um autômato bidimensional, com células postas em um tabuleiro – um espaço cartesiano, infinto, com dois eixos, sendo que cada célula ocupada um ponto $(x, y) \in (\mathbb{N}, \mathbb{N})$. A vizinhança de uma célula - uma função de agrupamento - é uma parte da função de transição de estado que define um conjunto de células as quais o estado de uma célula depende.

2 Jogo da Vida de Conway

Nos anos 70, o matemático britânico John Conway propôs um autômato celular chamado de Jogo da Vida, com a motivação de estudar as propriedades macroscópicas do crescimento populacional. Cada célula desse jogo tem dois estados, *dead* - morto - e *alive* - vivo. A evolução do jogo é produto apenas de sua configuração inicial, sendo portando, um jogo com zero jogadores. O critério usado para a escolha da função de transição era de dificuldade de previsão do crescimento das células. Ela pode ser definida pelas seguintes propriedades:

- *Neighborhood* (Vizinhança): A vizinhança de uma célula no ponto (x, y) são todos os células imediatamente adjacentes a ele – as que ocupam os pontos $(x-1, y+1)$, $(x, y+1)$, $(x+1, y+1)$, $(x-1, y)$, $(x+1, y)$, $(x-1, y-1)$, $(x, y-1)$, $(x+1, y-1)$.
- *Survival* (Sobrevivência): Se uma célula está viva e tem dois ou três vizinhos vivos, então ela continua viva.

- *Birth* (Nascimento): Se uma célula está morta e exatamente três vizinhos vivos, então ela fica viva.
- *Death* (Morte): Se uma célula está viva e tem zero, um, ou mais três vizinhos vivos, então ela continua morta.

3 Implementação do Jogo da Vida

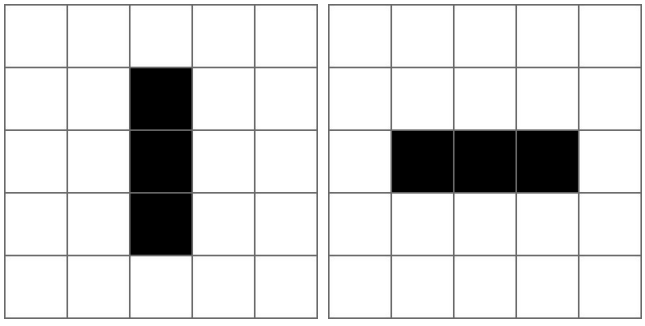
O Jogo da Vida foi modelado e implementado usando o NuSMV 2.5 e preprocessador CPP 4.4.5. O tabuleiro usado é um espaço cartesiano $k \times k$, com k finito. Ele é representado internamente como um vetor de tamanho k^2, indexado a partir do zero, que mapeia um ponto (x, y) do tabuleiro o para índice $x + y \cdot k$. Células vivas são representadas com o inteiro 1 e células mortas com 0. Uma implementação usando um tabuleiro quadrado de 25 posições e outro de 81 posições estão disponíveis.
Por ser de tamanho fixo, a vizinhança usada na função de transição teve que ser modificada para pontos nos extremos do tabuleiro. Só serão inclusos na vizinhança os pontos adjacentes e contidos no vetor, ou seja, pontos (x, y) tais que $0 < x+y \cdot k < k^2$.

4 Verificação de Gerações

Para uma dada configuração inicial do Jogo da Vida, podem ser verificadas propriedades em algumas ou em todas as gerações. Os padrões mais interessantes são mostrados abaixo, junto com a fórmula CTL que a verifica. Considere k o tamanho do tabuleiro e t o vetor do tabuleiro.

4.1 Longevidade

Uma configuração que tem, em qualquer geração, pelo menos uma célula viva satisfazem a propriedade de longevidade. Típicas configurações iniciais que a satisfazem são *still lifes* - seres inertes - e *oscillators* - osciladores. O oscilador mais simples, o *blinker*. Ele altera entre duas configurações infinitamente, elas estão mostradas abaixo. Um tabuleiro com o *blinker* está incluído no código fonte do programa.



Uma fórmula CTL que verifica esta propriedade é $AG(\sum_{i=0}^{k} t_i \neq 0)$.

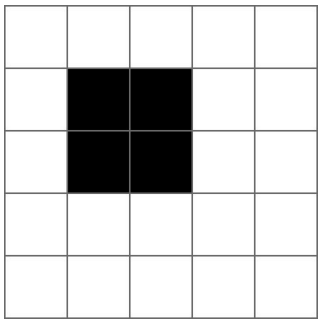
4.2 Extermínio

Uma configuração que tem, em uma geração qualquer, nenhuma célula viva satisfazem a propriedade de extermínio. Várias configurações randômicas tendem a levar à tabuleiro vazios.

Uma fórmula CTL que verifica esta propriedade é $AG(\sum_{i=0}^{k} t_i = 0)$.

4.3 Estabilidade

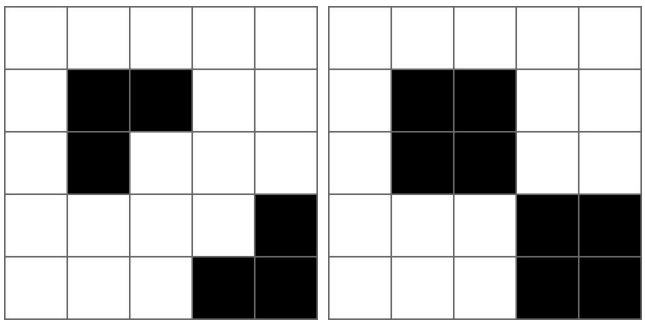
Uma configuração é dita ser estável, se há uma geração que é igual à anterior, ou seja, o autômato, eventualmente, para. Uma configuração estável chamado *block*, está mostrada abaixo.



Uma fórmula CTL que verifica esta propriedade é $EF(\bigwedge_{i=0}^{k} AG\ (AX(t_i) = t_i)))$.

4.4 Periodicidade

Osciladores alteram entre uma sequência finita de estados de forma infinitamente frequente. O oscilador mais simples, o *blinker* está mostrado acima. O *blinker* tem período 2, visto que ele retorna a uma configuração em uma geração g na geração $g + 2$. As gerações de um outro oscilador chamado *beacon* estão mostradas abaixo. Ele tem período 2, como o *blinker*. Podemos verificar se uma configuração tem periodicidade n com a fórmula CTL $!AG(\bigwedge_{i=0}^{k} (t_i = 1 \rightarrow AX_{n-1}\,(t_i = 1))\ \&\ (t_i = 0 \rightarrow AX_{n-1}(t_i = 0)))\&AG(\bigwedge_{i=0}^{k}((t_i = 1 \rightarrow AX_n(t_i = 1))\&(t_i = 0 \rightarrow AX_n(t_1 = 0))))$, com $AX_n(b) = \underbrace{AX(AX\ \ldots AX(b))\ldots}_{nvezes}$. Para verificar se o *beacon* tem período dois, por exemplo, temos a fórmula CTL $!AG(\bigwedge_{i=0}^{k}((t_i = 1 \rightarrow AX(t_1 = 1))\&(t_i = 0 \rightarrow AX(t_i = 0)))) \ \&\ AG(\bigwedge_{i=0}^{k}((t_i = 1 \rightarrow AX\ (AX\ (t_i = 1)))\ \&\ (t_i = 0 \rightarrow AX\ (AX\ (t_i = 0)))))$.



4.5 Propriedades celulares

Podemos verificar se todas a células do tabuleiro trocarem de estado pelo menos uma vez com a fórmula CTL $EF(\bigwedge_{i=0}^{k}(t_i = 1 \rightarrow EF\,(t_i = 0))\,\&\,(t_i = 0 \rightarrow EF\,(t_i = 1)))$.
Podemos verificar se todas a células do tabuleiro trocam de estado infinitamente frequente com a fórmula CTL $AG(\bigwedge_{i=0}^{k}(t_i = 1 \rightarrow EF\,(t_i = 0))\,\&\,(t_i = 0 \rightarrow EF\,(t_i = 1)))$.
Podemos verificar se pelo menos uma célula do tabuleiro troca de estado com a fórmula CTL $EF(\bigvee_{i=0}^{k}(t_i = 1 \rightarrow EF\,(t_i = 0))\,\&\,(t_i = 0 \rightarrow EF\,(t_i = 1)))$.