

Frog Chess



Projeto PLOG - 2019/20 - MIEIC

Turma 4 Grupo Frog_Chess_2

Professor das Aulas Práticas: Daniel Augusto Gama de Castro Silva

Autores

Mário Mesquita, up201705723 (up201705723@fe.up.pt)

Pedro Esteves, up201705160 (up201705160@fe.up.pt)

Porto, 20 de Outubro de 2019

Índice

O jogo: Frog Chess	3
História	3
Regras de Jogo	4
Número de Jogadores	4
Preparação	4
Objetivo	4
Desenvolvimento	5
Fim	6
Fontes de informação	7
Modelação do jogo em Prolog	8
Representação interna do estado do jogo	8
Descrição	8
Exemplos	8
Tabuleiro vazio	8
Estado inicial	9
Estado intermédio	10
Estado final	10
Visualização do tabuleiro em modo de texto	12
Descrição	12
Exemplos	13
Tabuleiro vazio	13
Estado inicial	14
Estado intermédio	15
Estado final	16

O jogo: *Frog Chess*

História

Pelas palavras do próprio **Joseph Brower, presidente e lead developer** na *Binary Cocoa* : “Frog Chess was created by Brian Grigsby. He is one of our game designers at Binary Cocoa. He actually came up with the idea many years ago (over 20 years) and never published or went forward with the idea. Many of our games are very original, so it's hard to say if any particular game inspired us. Frog Chess was going to be called Frog Checkers (as the pieces move more similarly to Checkers) but we decided on Frog Chess, because we wanted people to associate it with deeper strategy (it is much more strategic than Checkers.) So you could say that Checkers served some motivation, but the idea largely originated completely from Brian. We wanted the game to appeal to children, and having pieces jump over each other is boring. Frogs on the other hand, are significantly more exciting. We were able to find frogs that we could easily obtain that were fun to hold, and we felt it added some dynamic to the game in that there is a great tactile experience. Also, Frogs like to jump, so it made logical sense that in a game with a lot of jumping, you would find frogs. :)”.

Essencialmente, Frog Chess foi criado por **Brian Grigsby** e produzido pela *Binary Cocoa*, de uma ideia que já terá surgido há mais de 20 anos. Apesar de maioritariamente original, tem inspirações no jogo **Damas**, tendo sido escolhida a designação de Xadrez para realçar a profundidade da sua estratégia. O tema das rãs advém de uma tentativa de apelar às crianças, assim como do facto de fazer total sentido face ao estilo do jogo.



Figura 1 - Jogo Damas, inspiração para o Frog Chess

Regras de Jogo

Número de Jogadores

Frog Chess pode ser jogado por dois ou três jogadores. No entanto, nesta fase do projeto apenas serão considerados dois jogadores, o jogador verde (em vez do jogador amarelo que surge nas imagens) e o jogador cor-de-rosa.

Preparação

O jogo começa pela colocação das rãs de cada jogador no tabuleiro, inicialmente vazio, até que toda a zona de jogo (excluindo as bordas do tabuleiro, coloridas a verde na imagem abaixo) fique totalmente preenchido.

A escolha das posições é feita alternadamente entre os jogadores, uma rã de cada vez. Qualquer participante pode tomar a iniciativa de começar.

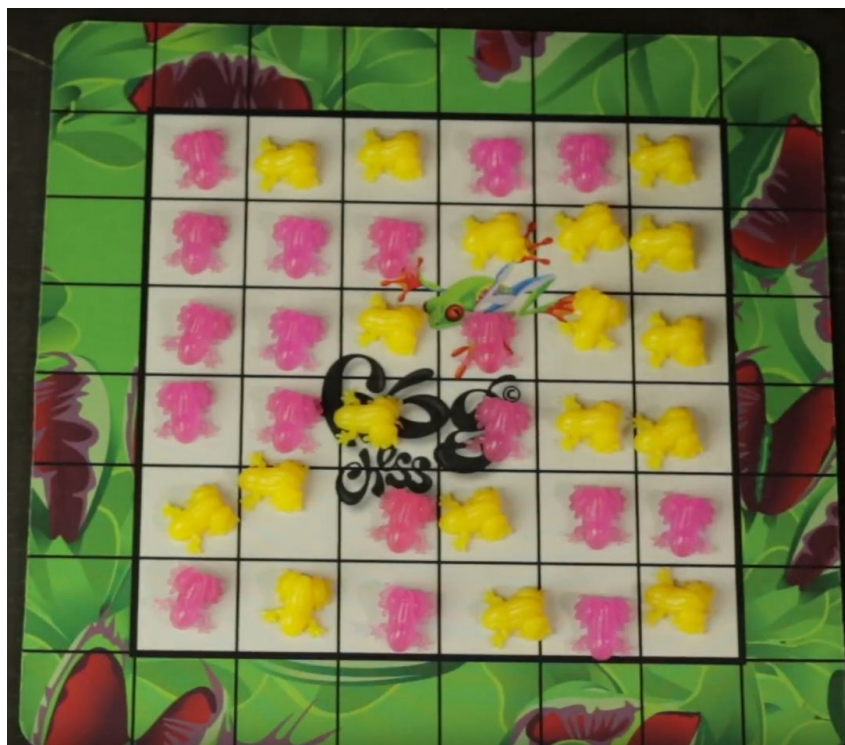


Figura 2 - Tabuleiro de jogo preenchido com rãs de dois jogadores. Note-se a zona “especial” a verde na borda do campo.

Objetivo

O objetivo final do jogo é ser a última rã a saltar.

Ao contrário do que talvez seria expectável, é possível ganhar o jogo com menos rãs no tabuleiro do que os adversários.

Uma estratégia utilizada frequentemente é tentar isolar as rãs dos adversários, impossibilitando-os de efetuar saltos.

Desenvolvimento

O jogo progride por turnos, em que as rãs dos jogadores **saltam por cima umas das outras**.

Quando uma rã salta por cima de outra, a segunda é **removida do tabuleiro**.

O salto pode ser feito em **todas as direções**, incluindo as diagonais, desde que exista uma rã para saltar por cima na proximidade.

Em cada turno apenas uma rã pode saltar. No entanto, podem ser feitos **saltos em sucessão** (enquanto existirem peças para saltar por cima), saltando até por cima das próprias rãs se necessário.

A **zona mais exterior** do tabuleiro, sinalizada de cor diferente, possui um comportamento adicional. Qualquer rã aqui colocada é removida no final do turno. Esta pode ser visitada e abandonada durante uma mesma jogada, sendo a rã removida apenas se lá permanecer no fim do turno.



Figura 3 - Desenvolvimento do jogo

Fim

O jogo acaba quando um jogador não consegue efetuar qualquer salto no seu turno, por qualquer motivo que seja. Pode acontecer que um jogador perca todas as suas rãs e não consiga efetuar qualquer ação, assim como é possível que, ainda que tenha rãs no tabuleiro, estas se encontrem isoladas de outras peças e, portanto, impossibilitadas de saltar.

Ganha o jogador que efetuou o último salto.

Normalmente, o jogo tem uma duração curta, inferior a 15 minutos.



Figura 4 - Jogador amarelo perdeu, uma vez que não pode efetuar qualquer salto.

Fontes de informação

Website oficial

- <https://binarycocoa.com/portfolio/frog-chess/>

Board game geek - Website dedicado a jogos de tabuleiro

- <https://boardgamegeek.com/boardgame/284634/frog-chess>

Blog de Brian Grigsby - Análise do jogo

- <http://guildmastergaming.blogspot.com/2018/08/frog-chess-by-brian-grigsby-game-review.html>

Email enviado ao presidente da *Binary Cocoa*, Joseph Brower, a requisitar informação.

Modelação do jogo em Prolog

Representação interna do estado do jogo

Descrição

Tendo em conta o objetivo final de ser a última rã a saltar, a representação do estado do jogo resume-se à representação das **rãs dentro do tabuleiro** e do **próximo jogador** a efetuar um movimento. Não é relevante representar as peças fora de jogo, uma vez que estas não são indicativo da condição de vitória, assim como qualquer outro elemento.

O tabuleiro de jogo será representado por uma **matriz** (lista de listas) 8 por 8, em que cada elemento corresponde a uma posição do tabuleiro.

Cada posição poderá assumir um de três valores, com os seguintes significados:

- **empty** : posição do tabuleiro vazia, isto é, sem uma rã;
- **green**: posição ocupada por uma rã do jogador verde;
- **pink**: posição ocupada por uma rã do jogador cor-de-rosa;

Exemplos

Tabuleiro vazio

Tabuleiro vazio, antes de os jogadores colocarem as suas rãs. Todas as 64 posições estão vazias.

```
/* 64 empty cells */
emptyBoard([
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty]
```


]).

Estado inicial

Como já referido na secção de Preparação, o estado inicial do jogo é obtido pela colocação alternada de rãs por cada jogador.

Embora as posições das rãs de cada jogador não sejam fixas, mantém-se inalterável o facto de que cada jogador começa o jogo com 18 peças, ocupando cada um metade da zona interior de jogo.

Apresenta-se uma possível configuração de estado inicial, correspondente à representada na Figura 2 acima.

```
/*
  18 green frogs
  18 pink frogs
  28 empty cells, the board edges
*/
initialBoard([
  [empty, empty, empty, empty, empty, empty, empty, empty],

  [empty, pink, green, green, pink, pink, green, empty],

  [empty, pink, pink, pink, green, green, green, empty],

  [empty, pink, pink, green, pink, green, green, empty],

  [empty, pink, pink, green, pink, green, green, empty],

  [empty, green, green, pink, green, pink, pink, empty],

  [empty, pink, green, pink, green, pink, green, empty],

  [empty, empty, empty, empty, empty, empty, empty, empty]
]).
```

Estado intermédio

Apresenta-se uma situação em que o jogo, tendo já começado, ainda se encontra a decorrer, correspondente à Figura 3.

```
/* The game is still going on, any player can jump. */
intermediateBoard([
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, pink, pink, empty, empty, empty, pink, empty],
  [empty, pink, pink, empty, green, empty, empty, empty],
  [empty, pink, empty, empty, green, green, green, empty],
  [empty, pink, pink, empty, pink, empty, empty, empty],
  [empty, green, pink, empty, empty, empty, pink, empty],
  [empty, empty, green, empty, green, pink, green, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty]
]).
```

Estado final

Um estado final é alcançado quando um jogador não consegue efetuar um salto no seu turno.

O primeiro estado final apresentado representa a Figura 4, em que o jogador verde perdeu por ter ficado apenas com uma rã isolada.

```
/* The only green frog in the bottom-right corner is isolated. Pink player wins. */
isolatedPiece([
  [empty, empty, empty, empty, empty, empty, empty, empty],
  [empty, pink, pink, empty, empty, pink, pink, empty],
  [empty, pink, pink, empty, empty, empty, pink, empty],
  [empty, pink, empty, empty, empty, empty, empty, empty],
  [empty, pink, pink, empty, empty, empty, empty, empty],
  [empty, empty, pink, empty, empty, empty, empty, empty],
  [empty, empty, empty, empty, empty, empty, green, empty],
  [empty, empty, empty, empty, empty, empty, empty, empty]])
```

O estado final seguinte representa uma outra situação de término, em que o jogador cor-de-rosa ficou sem rãs no tabuleiro e, portanto, impossibilitado de saltar. Como o jogador verde foi o último a movimentar uma peça, é o vencedor.

```
/*  
  Pink player has no frogs left and green player made the last move.  
  Green player wins.  
*/  
noPinkFrogsLeft([  
  [empty, empty, empty, empty, empty, empty, empty, empty],  
  [empty, empty, empty, empty, empty, empty, empty, empty],  
  [empty, empty, empty, empty, green, empty, empty, empty],  
  [empty, empty, empty, empty, empty, empty, empty, empty],  
  [empty, empty, empty, empty, empty, empty, empty, empty],  
  [empty, empty, green, empty, empty, green, empty, empty],  
  [empty, empty, empty, empty, empty, green, green, empty],  
  [empty, empty, empty, empty, empty, empty, empty, empty]  
]).
```

Visualização do tabuleiro em modo de texto

Descrição

Para a visualização do estado interno do jogo, foi desenvolvido o predicado ***display_game(+Board, +Player)*** que, para além do tabuleiro e respetivas peças em jogo, mostra também no ecrã o jogador que irá efetuar o próximo movimento.

De modo a facilitar a leitura do tabuleiro e a escolha da próxima jogada, é feita uma ordenação das colunas, usando números de 1 a 8, e das linhas, usando letras minúsculas de *a* a *h*. Assim, a próxima jogada de cada jogador, será escolhida conforme esta numeração, indicando a respetiva linha e coluna para onde quer movimentar as suas rãs.

Uma vez que são utilizados caracteres da **Tabela de Caracteres ASCII Completa** (*Extended ASCII characters*), é aconselhada a visualização do tabuleiro com a **fonte Terminal**. Devido ao tamanho da representação do tabuleiro, aconselha-se o **tamanho 9** da fonte mencionada. Deste modo, é possível visualizar o tabuleiro completo e sem falhas nos caracteres utilizados para a sua representação, não suportados pela grande parte das fontes.

Adicionalmente, deve ser utilizado o **SWI Prolog** para executar o programa, uma vez que são utilizados predicados apenas presentes nesta implementação. Destaca-se o predicado ***ansi_format(+ClassOrAttributes, +Format, +Args)***, que permite alterar as propriedades de qualquer carácter, nomeadamente as suas **cores**. Este é utilizado para a representação das rãs de cada jogador, atribuindo cores distintas a cada jogador.

Para a **representação das rãs** foi utilizada uma representação em **ASCII Art** retirada da página asciart.eu/animals/frogs.



Figura 5 - ASCII Art das rãs.

Exemplos

As representações seguintes seguem os exemplos presentes na secção anterior, “Representação interna do estado do jogo”, mostrando cada estado do tabuleiro. Em cada exemplo, é também apresentada a chamada efetuada em Prolog para que cada estado do tabuleiro seja impresso no ecrã.

Tabuleiro vazio

Tabuleiro vazio, antes de os jogadores colocarem as suas rãs. Todas as 64 posições estão vazias.

```
?- emptyBoard(B), display_game(B, 1).
```

	1	2	3	4	5	6	7	8
a								
b								
c								
d								
e								
f								
g								
h								


```

=====
| Player 1 Turn |
=====
  
```




Figura 6 - Representação de um tabuleiro vazio.

Estado inicial

O estado inicial do jogo é obtido pela colocação alternada de rãs por cada jogador.

```
?- initialBoard(B), display_game(B, 1).
```

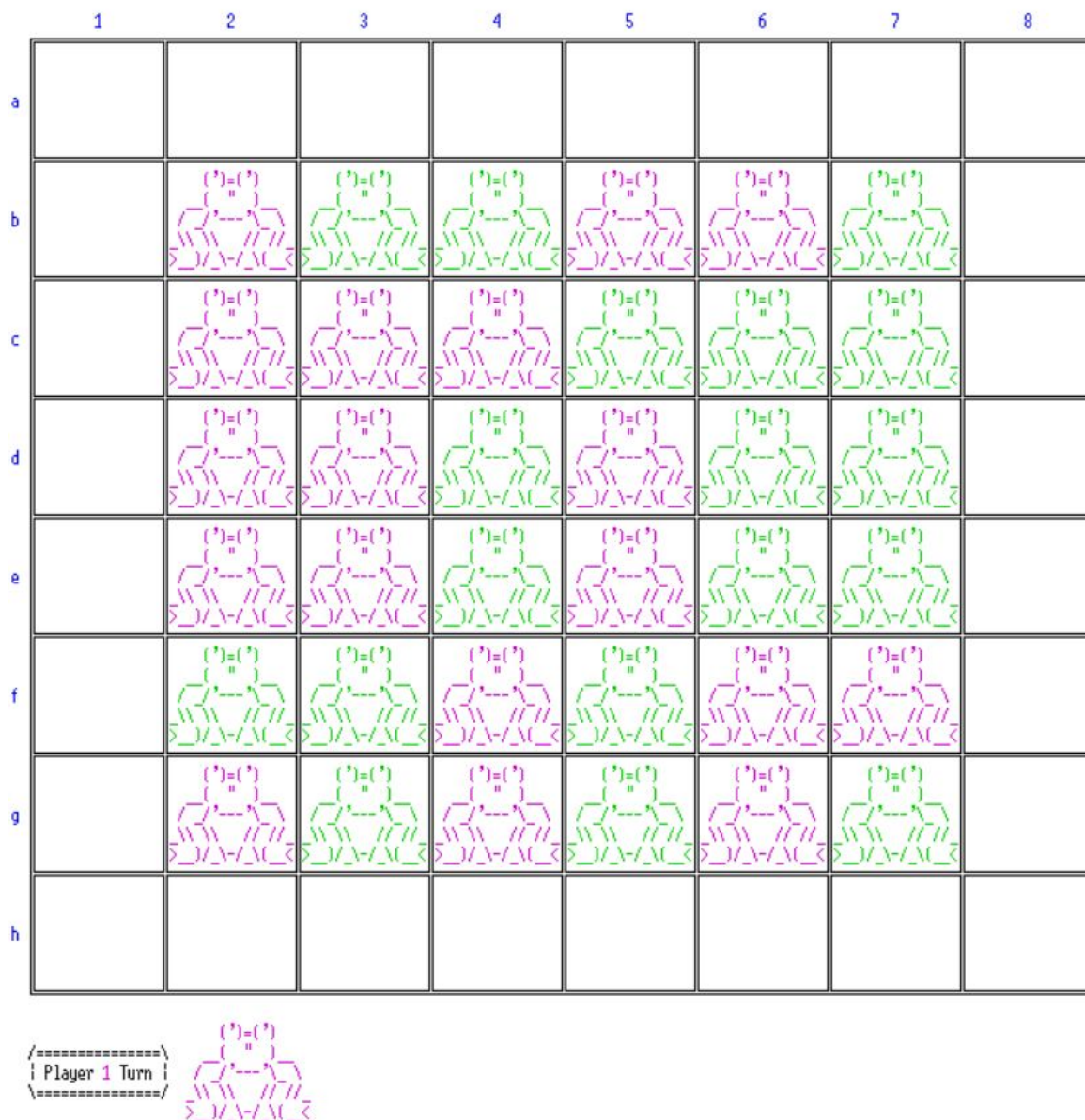


Figura 7 - Representação de um estado inicial.

Estado intermédio

Apresenta-se uma situação em que o jogo, tendo já começado, ainda se encontra a decorrer.

?- intermediateBoard(B), display_game(B, 2).

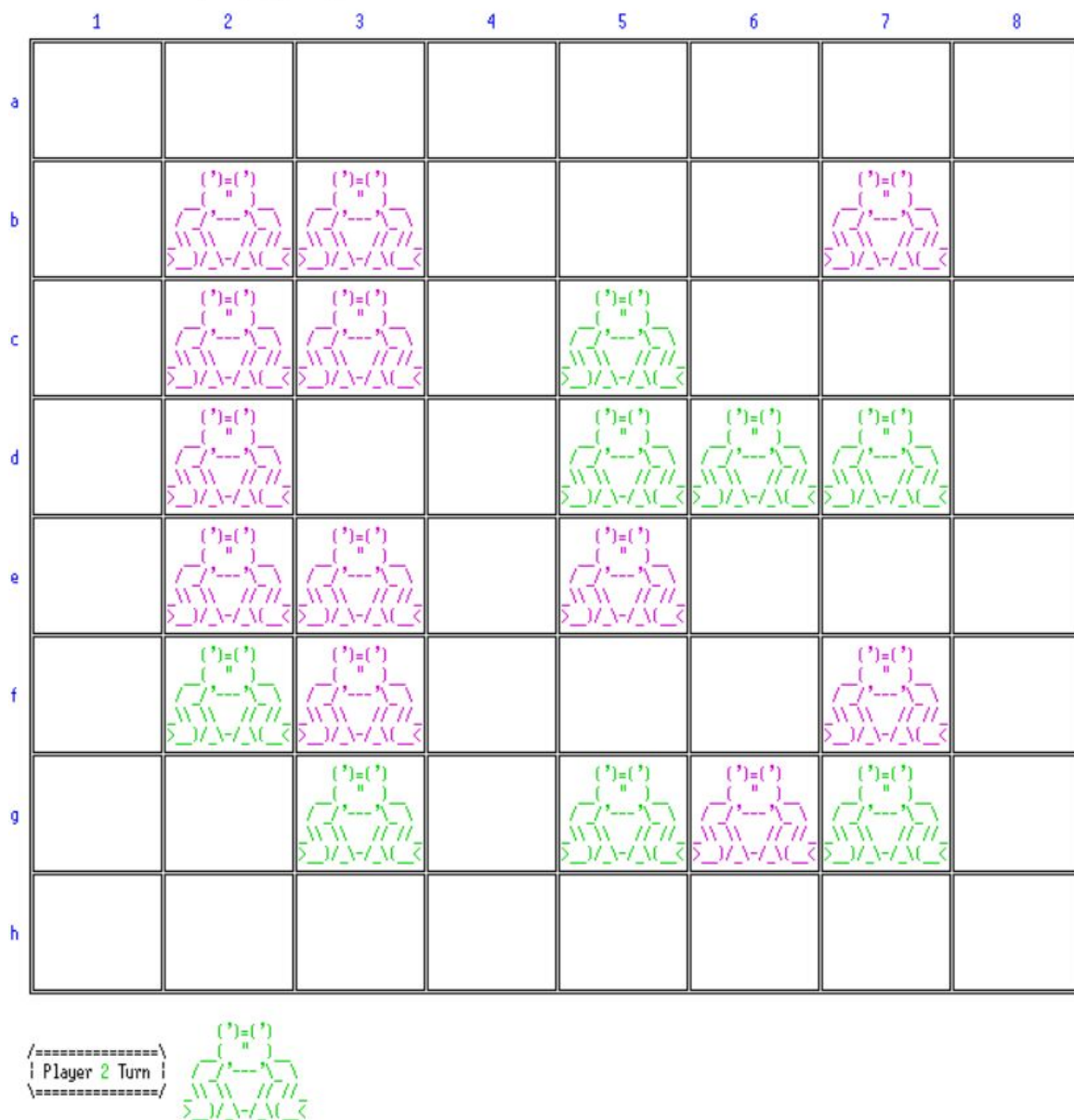


Figura 8 - Representação de um estado intermédio.

Estado final

Um estado final é alcançado quando um jogador não consegue efetuar um salto no seu turno.

Apresenta-se a representação das duas situações de estado final mencionadas na secção anterior.

```
?- isolatedPiece(B), display_game(B, 1).
```

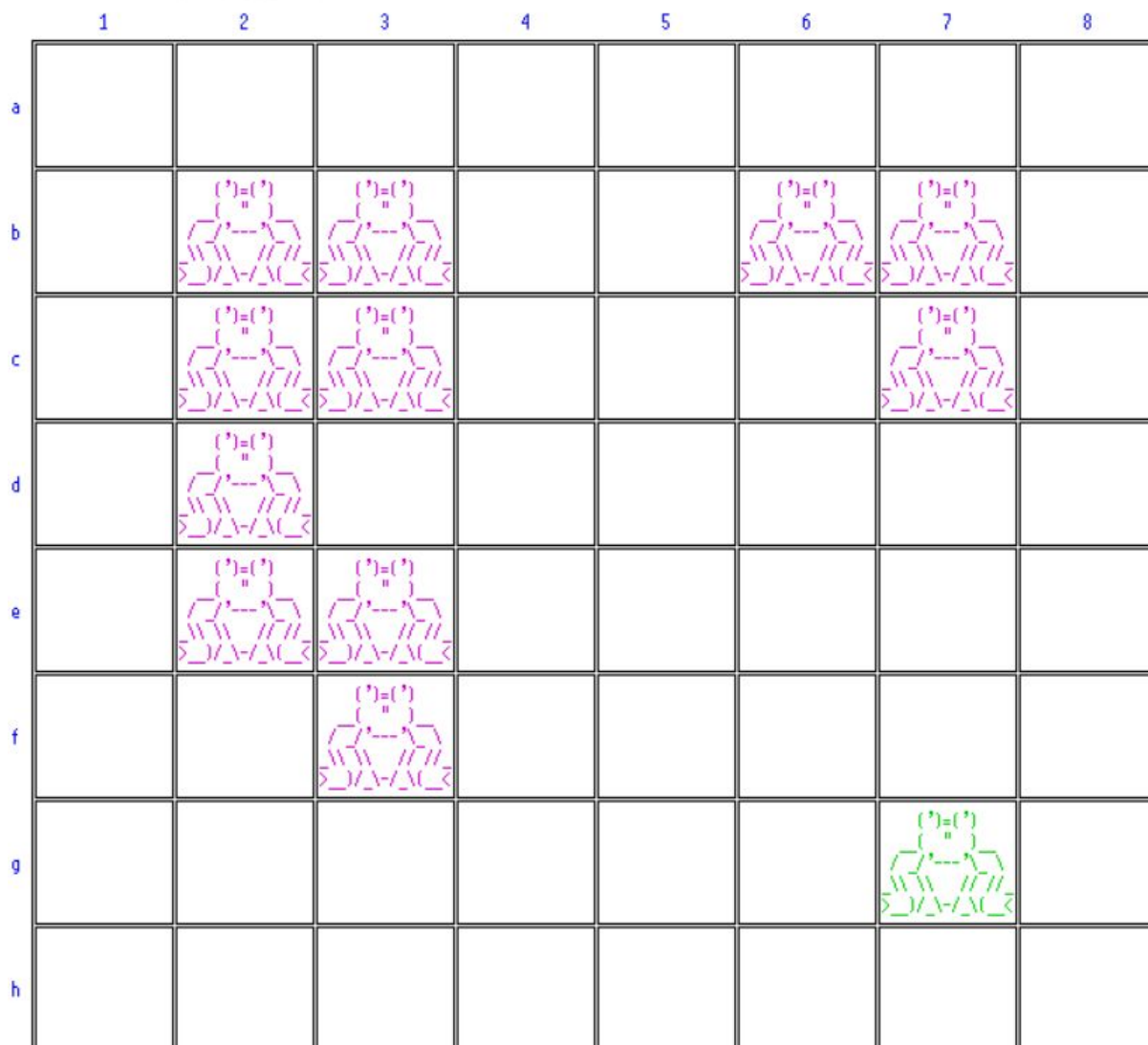


Figura 9 - Representação de um estado final, com uma rã isolada.

?- noPinkFrogsLeft(B), display_game(B, 1).

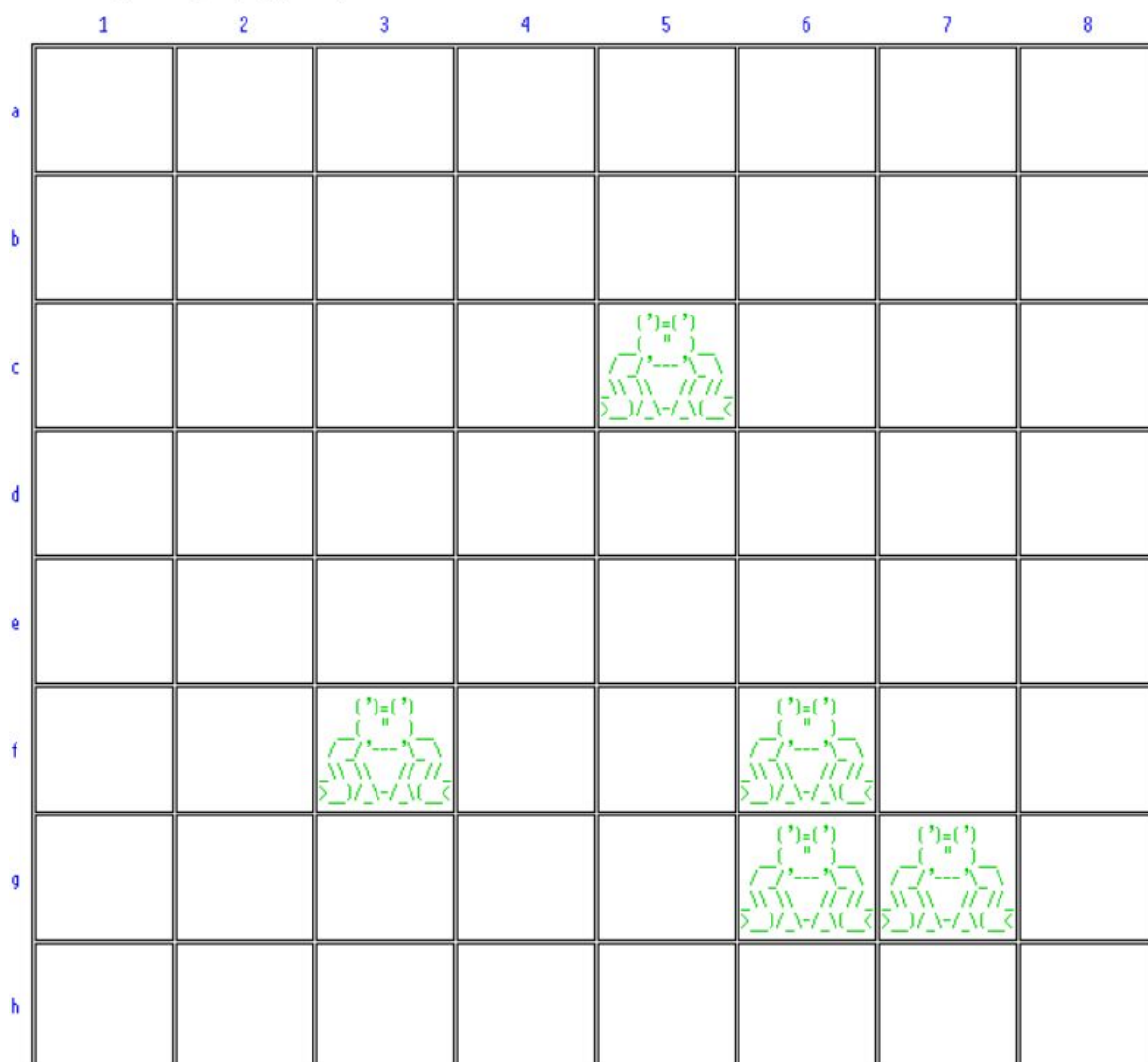


Figura 10 - Representação de um estado final, em que um jogador ficou sem rãs.