FERRAMENTA PARA CRIAÇÃO, VALIDAÇÃO E RESOLUÇÃO DE DESAFIOS LÓGICOS TEXTUAIS

Diego Esmerio da SILVA

Fundação Educacional do Município de Assis – FEMA
diesmerio@gmail.com

RESUMO: Este artigo aborda o processo de desenvolvimento de uma ferramenta gráfica para criação, validação e resolução de desafios lógicos textuais com recursos de extensibilidade e geração de arquivos de jogo para distribuição. Ao contrário de outras abordagens por força bruta, este projeto ataca o problema com o uso de uma inteligência artificial que visa simular o processo de resolução usado por jogadores humanos, possibilitando a criação e validação de desafios maiores sem que o tempo de resolução aumente de forma exponencial. O algoritmo usa um sistema de regras ativas com autonomia para observar e manipular o estado atual do desafio. Esta abordagem permite que o problema seja resolvido parcialmente conforme as regras sejam adicionadas, e desta forma visualizar em tempo real a complexidade do problema criado. Para evitar erros, as regras contam com recursos de detecção de contradições. No modo de criação de novos desafios a detecção de contradição é usada para que não seja possível adicionar regras que contradigam a solução previamente definida. No modo de jogo a detecção de contradição é usada para dar ao jogador um auxílio visual para que este veja os erros cometidos. Para validar o desafio, as regras devem ser suficientes para que a solução seja única e consequentemente não se faz necessário salvar a solução junto as regras.

PALAVRAS-CHAVE: Desafios lógicos; Inteligência artificial; Algoritmos

ABSTRACT: This paper describes the process used in the development of a tool for creation, validating and solving logic puzzles, with extensibility support and outputting of game files for subsequent playing. In contrast of brute-force heuristics, this project approaches the subject using an artificial intelligence that simulates the solving process used by humans, resulting in bigger and complex puzzles without the drawback of an exponential increase in the solving time. The

algorithm uses a system of active rules than can observe and make changes to the game state. This approach makes it possible to have a partial solution as rules are inserted and see in real-time as the solution unfolds, giving information about how complex the puzzle is. The rules have also a contradiction detection mechanism. In the creation mode that mechanism prevents the insertion of rules that contradicts the previous defined solution, making the process foolproof. In the game mode that mechanism gives a visual aid for helping the player in identifying mistakes. As the rules must describe a unique solution for the puzzle to be valid the solution do not need to be stored.

KEYWORDS: Logic puzzle; Artificial intelligence; Algorithms

1. Desafios lógicos

Os desafios lógicos tratados neste artigo seguem o seguinte modelo:

1. O desafio descreve um cenário com um número definido de itens que possuem características distintas;
2. Existe ao menos dois conjuntos que descrevem características;
3. Cada item está relacionado com um e somente um item de outro conjunto;
4. As dicas descrevem a relação entre os itens de cada conjunto;
5. As relações são transitivas[1], reflexivas[2] e simétricas[3];
6. Não há relações entre os itens de um mesmo conjunto.

Para facilitar o processo de resolução o desafio é frequentemente acompanhado por uma grade, representando a intersecção de todos os conjuntos e todos os itens destes. Por consequência de 5 e 6, ao descobrir um relacionamento verdadeiro entre dois itens todas as outras relações na mesma categoria são marcadas como falsas. Por consequência de 3, se houver n-1 relações falsas na intersecção entre um item e outro conjunto, a relação entre este item e o item restante é verdadeira. Desta forma é possível descrever um cenário maior utilizando poucas regras.

[1] Uma relação transitiva descreve uma relação onde se A está relacionado a B e B está relacionado a C, A está relacionado a C.
[2] Uma relação reflexiva descreve uma relação onde o item é relacionado a si próprio.
[3] Uma relação simétrica descreve uma relação onde se A está relacionado a B, logo B está relacionado a A.

2. Representação dos itens

Para representar os itens foram criadas classes para representar o jogo em vários níveis, estes são:

- Grade: Classe menos específica, contém o nome do problema, número de itens por categoria e lista de categorias. Usado principalmente para definir um lugar comum para acessar todos os elementos do jogo;
- Categoria: Define a característica que é distinta em todos os itens. Contém o nome da categoria, lista de propriedades e lista de itens;
- Propriedade: Define um valor numérico para descrever características quantitativas ou qualitativas comuns, tais como idade e preço ou presença ou ausência de uma característica, tais como sexo, paridade e cor base. Composto pelo nome da propriedade e um dicionário, usando o item como chave para obter o valor numérico;
- Item: Define o item propriamente dito, usado nas relações. Contém o nome do item.

Todos os itens de níveis mais internos contêm referências para seus elementos mais externos. O uso desta separação permite também uma fácil implementação de funções de serialização, útil para definir diferentes problemas usando a mesma descrição. O formato utilizado foi escolhido por ser facilmente lido e modificado com editores de texto, com informações separadas por barras verticais. Itens entre chaves duplas são opcionais. A especificação é composta por:

<Nome do desafio>|<Número de categorias>|<Número de itens por categoria>

<Nome da categoria 1>|<<Propriedade 1>>|(...)|<<Propriedade n>>

<Nome do item 1>|<<Valor da propriedade 1>>|(...)|<<Valor da propriedade n>>

(...)

<Nome do item n>|<<Valor da propriedade 1>>|(...)|<<Valor da propriedade n>>

(...)

<Nome da categoria n>|<<Propriedade 1>>|(...)|<<Propriedade n>>

<Nome do item 1>|<<Valor da propriedade 1>>|(...)|<<Valor da propriedade n>>

(...)

<Nome do item n>|<<Valor da propriedade 1>>|(...)|<<Valor da propriedade n>>

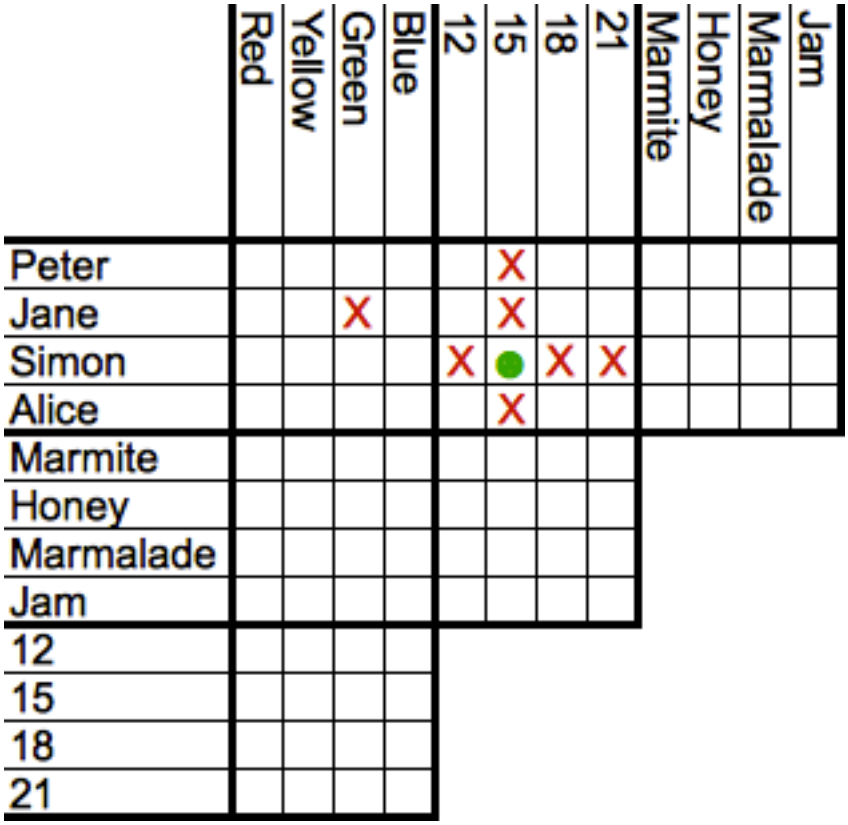

Figura 1 - Exemplo de grade para auxílio gráfico de solução

3. Representação dos relacionamentos

Para auxiliar na representação das regras de jogo foi desenvolvida uma estrutura composta por:

- Um dicionário para associar as relações verdadeiras entre itens, usando a categoria como chave;
- Uma lista de relações falsas, usando a mesma estrutura.

Esta composição permite obter a relação entre os itens facilmente. Para controlar as relações, um dicionário adicional é usado para acessar rapidamente a relação em que o item está contido. Assim, quando uma relação positiva é inserida, ambas estruturas são unidas e quando a relação é negativa, ambas são adicionadas na lista de relações falsas de cada relação. Desta forma, para verificar a relação entre dois itens, o seguinte algoritmo é utilizado:

a. Relação onde primeiro item está contido é pesquisada no dicionário auxiliar;
b. Categoria do segundo item é pesquisada no dicionário de relações;
c. Caso o item encontrado seja o mesmo item que o pesquisado, a relação é verdadeira.

d. Caso o item encontrado seja diferente do item pesquisado a relação é verdadeira com outro item da mesma categoria, portanto a relação é falsa;
e. Caso a categoria não seja encontrada, a lista de relações falsas é usada;
f. Caso o item pesquisado esteja contido na lista de relações verdadeiras de alguma relação, a relação é falsa;
g. Caso o item pesquisado não seja encontrado, a relação é desconhecida.

4. Regras invariantes

Algumas regras são comuns a todos os jogos e foram inseridas diretamente sobre o motor da inteligência artificial. Estas regras são usadas principalmente para evitar contradições e aumentar a densidade de informação descritas pelas dicas, tais como:

- Se uma relação entre dois itens é marcada como verdadeira, todas as relações verdadeiras e falsas de ambas relações são unidas e a todas as outras relações que continham referências a estas relações são atualizadas para conter apenas a relação unida;
- Se n itens podem apenas ser relacionados com os mesmos n itens de outra relação, nenhum outro item pode estar relacionado com estes. Caso contrário, ou algum item de uma categoria relacionaria com mais de um item de outra categoria ou algum item não teria relação com outro item desta categoria;
- Para suprir a limitação de regras serem limitadas a observar apenas ao estado atual das relações, foi adicionado um mecanismo de tentativa e erro de único nível, que testa se uma relação leva a um estado inválido. Desta forma, caso a regra “Item A é maior que item B” seja adicionada com “Item A é maior que item C”, a possibilidade de item A ser o segundo menor item leva a uma contradição e é eliminada.

5. Inteligência artificial

As regras externas descrevem as dicas e ficam em módulos externos ao programa principal. Para tal foram definidas interfaces comuns para possibilitar o carregamento em tempo de execução de novas regras, possibilitando a terceiros programarem novas regras de acordo com novas possibilidades. Estas

regras são divididas entre regras independentes e regras dependentes. As regras independentes são regras que descrevem uma relação de verdadeiro ou falso em uma única passagem, e não dependem do estado atual do jogo para fornecer novas informações. Afirmações como “Eduardo usava camiseta vermelha.”, “Juliana tem mais de 12 anos.” e “Renata não gosta de maçã.” podem ser descritas com regras independentes. Em contraste, as regras dependentes podem até definir parte do estado em uma primeira passagem, mas ficam ativas até que toda a informação seja obtida. “Renato é mais velho que Abreu.”, “Entre Sérgio e Carlos, um toca guitarra e o outro tem 25 anos em alguma ordem.” podem ser descritas com regras dependentes. É possível dizer que Renato não é o mais novo e Abreu não é o mais velho e que a pessoa de 25 anos não toca guitarra, mas não é possível saber toda a informação até que o estado tenha mais informações sobre estas relações.

Para obter estas informações, a inteligência segue os seguintes passos ao inserir uma regra:

1. Regra é adicionada em uma lista;
2. Regra passa pelo processo de inicialização[4];
3. Regra é então ativada;
4. Caso haja regras não finalizadas[5], todas estas são reativadas;
5. Caso uma regra fez uma modificação no estado atual, esta deve retornar verdadeiro.
6. Caso a regra já obteve toda a informação, esta deve marcar seu estado interno como finalizado.
7. Caso alguma regra retornou verdadeiro no passo 5, algoritmo retorna ao passo 4.
8. Mecanismo de tentativa e erro é ativado;
9. Se nenhuma contradição foi encontrada, o estado é revertido.
10. Caso relações verdadeiras contraditórias sejam encontradas, esta é marcada como falsa e algoritmo retorna ao passo 4.
11. Caso nenhuma informação nova seja obtida no passo 4 ou no passo 8, espera por uma nova regra.

[4] O processo de inicialização pode ser utilizado para definição de relações independentes dentro de regras dependentes, pois estas não dependem do estado e podem ser ativas uma única vez sem efeitos.
[5] Regras simples sempre retornam como finalizadas e são ativadas apenas uma vez.

Usando este algoritmo, a solução pode ser obtida de forma parcial ao simular o mesmo algoritmo que um jogador humano usaria ao solucionar o mesmo problema.

6. Tentativa e erro

Para minimizar os efeitos de força bruta, as regras podem utilizar de áreas de interesse, que consistem em categorias, propriedades, itens ou uma mistura de todos em uma lista, desta forma testando somente os elementos relevantes para a informação. Para tal, foi utilizado um algoritmo adaptativo que muda seu comportamento a partir da entrada de dados. Caso um elemento seja um item, este é testado em todas as outras categorias por contradições. Caso seja um elemento e uma categoria, este é testado apenas nas relações com esta categoria. Em caso de duas categorias, os itens de ambas são testados entre si. Este processo faz com que não haja testes em itens onde as regras não podem dar informações, maximizando a velocidade.

7. Restauração de estados

Para realizar os testes, faz-se necessário salvar os dados para restauração do estado anterior a este. Para este mecanismo foi utilizado o seguinte mecanismo:

1. Todas as relações conhecidas são salvas em uma tupla, composta pelos dois itens da relação binária e a relação entre estes;
2. Para evitar redundância, lista de relações é minimizada;
3. Caso uma relação é verdadeira, todos os itens na diferença são por consequência falsos, então estes não se fazem necessários.
4. Caso a relação entre A e B seja verdadeira e a relação de A e C é falsa, logo por transitividade a relação entre B e C é falsa, e esta não é necessária.
5. Ao final é obtido um conjunto mínimo de relações para representar todo o estado.

Nos testes de desenvolvimento, o uso de um algoritmo de restauração sem minimização causou um impacto expressivo no desempenho da inteligência artificial. Após a implementação otimizada de minimização, os testes que antes completavam em 12,3s passaram a completar em 0,7s. Este teste foi baseado em um desafio lógico creditado a Einstein, conhecido por *Zebra's Puzzle* e tem vários algoritmos de resolução documentados baseados em força bruta.

8. Regras implementadas

Para a realização do teste de conceito foram implementadas algumas regras que frequentemente aparecem em desafios lógicos. Estas são:

a. Verdadeiro
    i. Classe: Independente
    ii. Argumentos: Item A, Item B
    iii. Funcionamento: Relação entre A e B é verdadeira.
    iv. Contradição: Relação entre A e B é falsa.

b. Falso
    i. Classe: Independente
    ii. Argumentos: Item A, Item B
    iii. Funcionamento: Relação entre A e B é falsa.
    iv. Contradição: Relação entre A e B é verdadeira.

c. Todos diferentes
    i. Classe: Independente
    ii. Argumentos: Item A, Item B, ..., Item N
    iii. Funcionamento: Relação entre A, B, ..., N é falsa.
    iv. Contradição: Alguma relação entre A, B, ..., N é verdadeira.

d. Maior absoluto
    i. Classe: Independente
    ii. Argumentos: Item A, Propriedade B, inteiro C
    iii. Funcionamento: Item A é maior que C em relação a propriedade B.
    iv. Contradição: Item A é menor ou igual a C em relação a propriedade B.

e. Menor absoluto
    i. Classe: Independente
    ii. Argumentos: Item A, Propriedade B, inteiro C
    iii. Funcionamento: Item A é menor que C em relação a propriedade B.
    iv. Contradição: Item A é maior ou igual a C em relação a propriedade B.

f. Igual
    i. Classe: Independente
    ii. Argumentos: Item A, Propriedade B, inteiro C
    iii. Funcionamento: Item A é igual a C em relação a propriedade B.
    iv. Contradição: Item A é diferente de C em relação a propriedade B.
g. Maior
    i. Classe: Dependente
    ii. Argumentos: Item A, Item B, Propriedade C
    iii. Ativação: Item A não possui a menor propriedade C, item B não possui a maior propriedade C.
    iv. Funcionamento: Item A é maior que B em relação a propriedade C.
    v. Contradição: Item A é menor ou igual a B em relação a propriedade C.
    vi. Pontos de interesse: Item A, Item B, Propriedade C.
h. Diferença
    i. Classe: Dependente
    ii. Argumentos: Item A, Item B, Propriedade C, Inteiro D
    iii. Funcionamento: Relações onde não haja itens com diferença D entre A e B com relação a propriedade C são marcadas como falsas
    iv. Contradição: A diferença entre A e B em relação a C não é igual a D.
    v. Pontos de interesse: Item A, Item B, Propriedade C.
i. Diferença absoluta
    i. Classe: Dependente
    ii. Argumentos: Item A, Item B, Propriedade C, Inteiro D
    iii. Funcionamento: Relações onde não haja itens com diferença D para mais ou para menos entre A e B com relação a propriedade C são marcadas como falsas
    iv. Contradição: O módulo da diferença entre A e B em relação a C não é igual a D.
    v. Pontos de interesse: Item A, Item B, Propriedade C.

j. Diferença entre propriedades
    i. Classe: Dependente
    ii. Argumentos: Item A, Propriedade B, Propriedade C, Inteiro D
    iii. Funcionamento: A não pode assumir valores em relação a B que não sejam diferentes de C em D unidades.
    iv. Contradição: O valor de A em relação a B não é diferente de C em D unidades.
    v. Pontos de interesse: Item A, Propriedade B, Propriedade C.
k. Ou (1 para 2)
    i. Classe: Dependente
    ii. Argumentos: Item A, Item B, Item C
    iii. Ativação: B não pode ser igual a C.
    iv. Funcionamento: Se A é B logo A não é C e se A não é B, logo A é C.
    v. Contradição: A não é nem B nem C, A é B e C ou B é C.
    vi. Pontos de interesse: Item A, Item B, Item C.
l. Ou (2 para 2)
    i. Classe: Dependente
    ii. Argumentos: Item A, Item B, Item C, Item D
    iii. Ativação: A não pode ser igual a B e C não pode ser igual a D.
    iv. Funcionamento: Se A é C logo B é D e se A não é C, logo B é C e A é D.
    v. Contradição: A não é nem C nem D, A é C e D ou C é D e relações simétricas entre A e B, e C e D.
    vi. Pontos de interesse: Item A, Item B, Item C, Item D.

9. Interface gráfica

Foi programado várias telas, usando a tecnologia WPF[6] para uma melhor manipulação de telas dinâmicas, tais como as relações que mudam de estados desde as informações iniciais, onde o número de categorias, de propriedades e de itens em cada categoria não é fixo. A figura 2 e a figura 3 mostram as telas

[6] Windows Presentation Foundation – Framework para criação de interfaces gráficas usado no framework .NET.

em seus estados iniciais e modificados para diferentes combinações de propriedades e categorias.

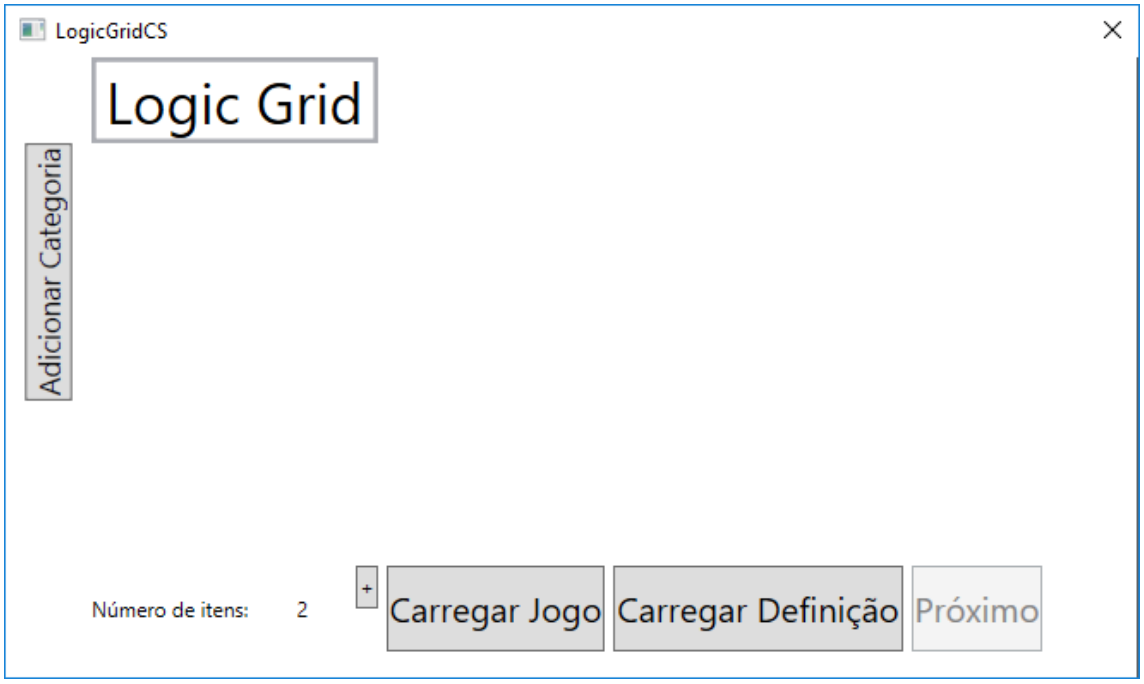

Figura 2

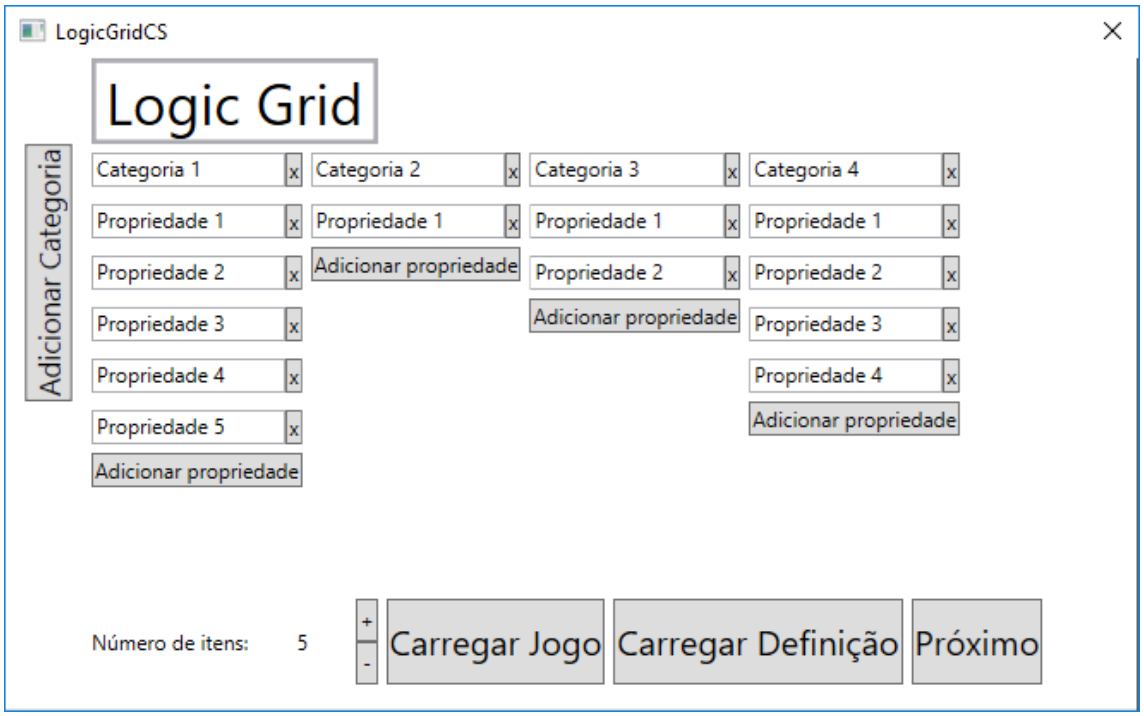

Figura 3

A figura 4 e 5 mostram respectivamente o modo de criação, onde a resposta é previamente definida (figura 6) e o modo de resolução, onde o jogo é aberto para receber qualquer tipo de regra.

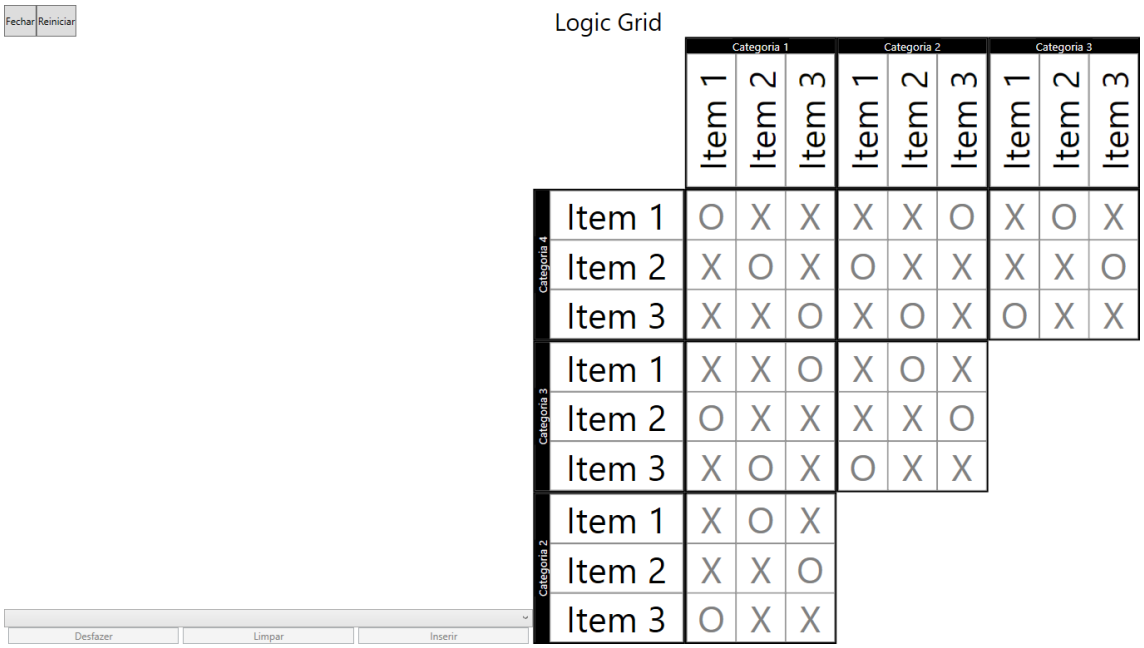

Figura 4

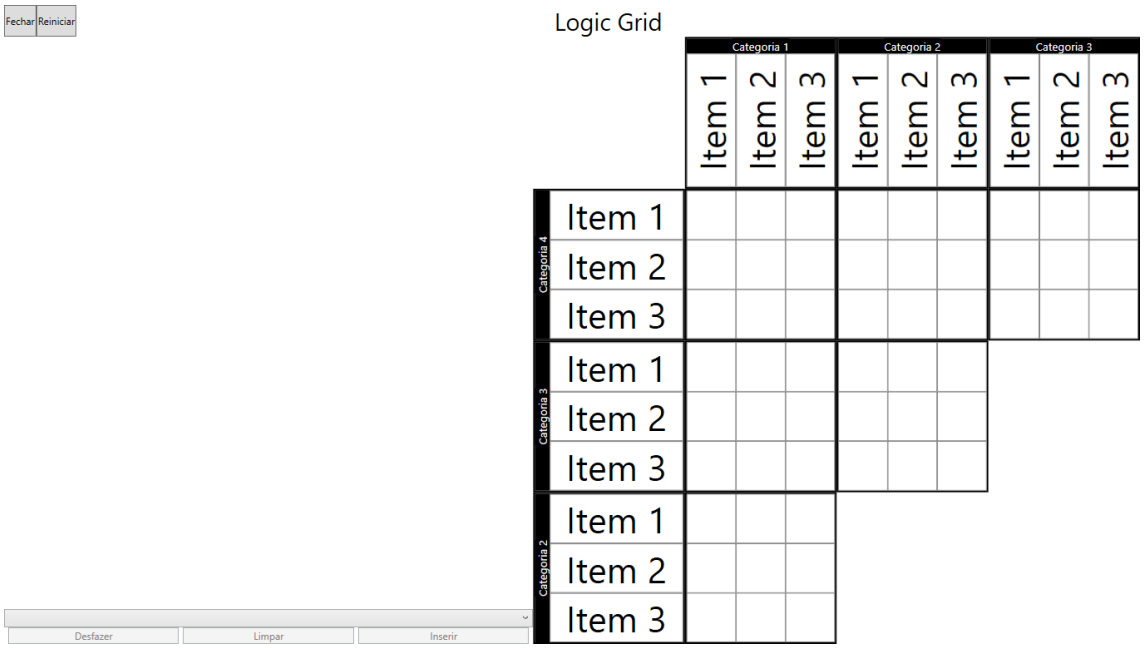

Figura 5

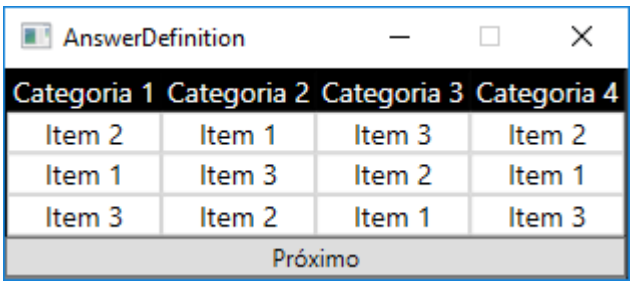

Figura 6

As funcionalidades de exportação de arquivos de jogos seguem outros conceitos não estão incluídos no escopo deste artigo.

10. Considerações finais

Este projeto mostra que é possível chegar a soluções de problemas com várias restrições sem o uso de força bruta e de soluções completas. Durante os testes não houveram problemas não representáveis pelas relações binárias.

11. Referência bibliográficas

Windows Presentation Foundation - Disponível em https://msdn.microsoft.com/pt-br/library/ms754130%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396. Acesso em dez 2015.

Zebra Puzzle – Rosetta Code – Diponível em http://rosettacode.org/wiki/Zebra_puzzle. Acesso em dez 2015.

Logic Puzzle – Disponível em https://en.wikipedia.org/wiki/Logic_puzzle. Acesso em dez 2015.