Prova da NP Completude de The Legend of Zelda: A Link to the Past

Gustavo Kundlatsch e Gustavo Raimundo

[1]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina(UFSC)

Abstract. *This paper has the objective to prove the NP completude of the game The Legend of Zelda: a Link to The Past, a classic game of the serie, released for the Super Nintendo Entertainment System (SNES). That is, if the game was played by an algorithm, the complexity to solve certain problems during it's run would be NP-Complete, so the game as a whole is NP-Complete. For this, we will use a proof framework, used to make similar proofs to other games of the same company.*

Resumo. *Este artigo tem como objetivo provar a NP completude do jogo The Legend Of Zelda: a Link to The Past, um dos clássicos da franquia, lançado para o Super Nintendo Entertainment System (SNES). Isto é, caso fosse jogado por um algoritmo, a complexidade para resolver certos problemas durante seu decorrer são NP-Completo, e assim, o jogo como um todo pode ser dito NP-Completo. Para isso, utilizaremos um framework que é utilizado para realizar provas semelhantes em diversos jogos da mesma empresa.*

1. Introdução

Diversos artigos acadêmicos falam da NP Completude da jogabilidade de variados tipos de jogos [Cormode 2004] [Forišek 2010] [Viglietta 2014]. Neste trabalho, vamos provar a NP Completude do clássico jogo da Nintendo "The Legend of Zelda: a Link to The Past", de 1991. Para isso, vamos usar a definição de [Aloupis et al. 2015] para caracterizar sua jogabilidade:

> Nós consideramos o problema de decisão de acessibilidade: dado uma fase ou *dungeon*, é possível alcançar o ponto de objetivo t a partir do ponto de início s?

Para isso, vamos reduzi-lo ao problema 3SAT (problema da satisfação booleana com três elementos), através de um framework, que será apresentado mais tarde.

Em tal prova, vamos assumir que nenhum *bug* ou *glitch* será utilizado para driblar qualquer mecânica do jogo, respeitando a coerência das mecânicas implementadas pelos desenvolvedores.

2. Apresentação dos Problemas

Como a prova apresentada nesse artigo utiliza um framework, ela pode se tornar um pouco abstrata demais. Nessa sessão apresentamos os problemas envolvidos e exemplos de cada um.

2.1. The Legend of Zelda: a Link to The Past

The Legend of Zelda: a Link to The Past é um jogo desenvolvido para Super Nintendo (que a partir desse momento será chamado apenas de Zelda, para facilitar a leitura), publicado pela primeira vez em 1991, no Japão. Nele, controlamos o protagonista Link, um garoto que tem como objetivo salvar a terra de Hyrule, impedir que o mago Ganon se torne o novo governante e salvar a princesa Zelda. Para isso, Link deve obter a Triforce, uma relíquia sagrada espalhada pelo mapa, que é composta por três triângulos equiláteros, que trazem ao portador sabedoria, coragem e poder.

Em termos de design, o jogo é um mapa aberto, em que o jogador pode andar livremente. A câmera é posicionada acima do plano, dando visão de uma parte do mapa total, sendo que quando o jogador se desloca para fora desse enquadramento a câmera se desloca para exibir uma nova parte do mapa. O jogo é uma mistura de combates e puzzles. Para avançar no jogo, é preciso explorar cavernas e construções, conversar com personagens e derrotar inimigos, assim obtendo novos itens e habilidades. Os itens podem ser achados dentro de baús, sendo que existem itens especiais que quando são encontrados seus baús permanecem abertos. Itens comuns podem voltam a ter seus baús fechados quando o jogador se move para outro pedaço do mapa e retorna ao pedaço original (inimigos têm o mesmo comportamento, ou seja, a não ser que sejam chefes especiais, os inimigos retornam ao mapa após morrerem caso haja essa troca de tela. Chefes especiais também liberam itens especiais). Como Zelda é um jogo de mapa aberto, qualquer inimigo pode ser enfrentado a qualquer momento, desde que seja possível chegar até ele, dando grande liberdade ao jogador.

Tais itens e habilidades não são em si importantes para a prova que iremos realizar, mas certamente poderiam ser utilizados para provar a NP-Completude por outros caminhos, como por exemplo realizando uma redução ao problema 1-Push 2D [Demaine et al. 2000].

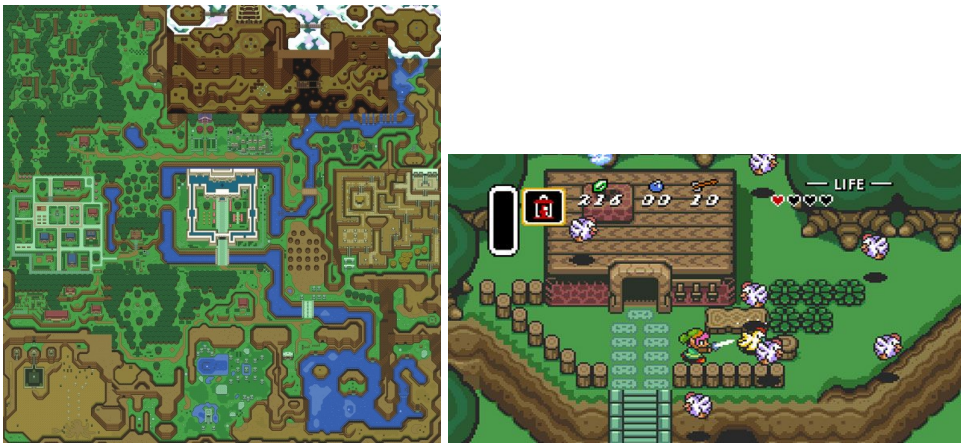

Figure 1. Imagens do jogo

Diversos elementos do jogo podem ser utilizados para provar sua NP-Completude. Nesse jogo específico da franquia existe um elemento bastante importante para esta prova, que o diferencia de seus anteriores: um gancho que Link pode utilizar para se mover em direção a qualquer bloco do cenário.

Figure 2. Exemplo de uso do gancho

2.2. 3-SAT

O problema de satisfazibilidade booleana (SAT) é um problema de decisão, cuja instancia é uma escrita com operadores lógicos AND, OR, NOT, variáveis, e parênteses. A questão desse problema de decisão é: dada uma expressão, há alguma atribuição de valores verdadeiros e falsos para as variáveis que torne toda a expressão verdadeira? Uma fórmula da lógica proposicional é dita satisfazível se e somente se é possível atribuir valores lógicos a suas variáveis de tal maneira que eles tornem a fórmula verdadeira. A prova da NP Completude do problema SAT é dada pelo teorema de Cook-Levin [Cook 1971].

O problema 3-SAT é um subconjunto do problema SAT, onde cada expressão lógica terá apenas três literais.

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

Figure 3. Exemplo do problema 3-SAT

3. Framework

O framework aqui apresentado foi originalmente proposto por [Aloupis et al. 2015], e tem como objetivo provar a NP-Hardness de jogos de plataforma. Para isso, ele implementa diversos dispositivos. Apesar de The Legend of Zelda: a Link to The Past não ser um jogo de plataforma, mostraremos como ele se encaixa neste modelo.

Essa framework funciona da seguinte maneira: O jogador começa na posição Start. Cada dispositivo de variável obriga o jogador a fazer uma escolha exclusiva de "verdadeiro" (x) ou "falso" ($\neg x$) como valor para a variável para uma fórmula booleana. Ambas as escolhas permitem ao jogador seguir caminhos que levam aos dispositivos de Cláusula, correspondentes as clausulas que contém aquela literal (x ou $\neg x$). Esses caminhos podem se cruzar, mas o dispositivo de Crossover previne que o jogador troque entre caminhos cruzados. Ao visitar o dispositivo de cláusula, o jogador pode desbloquear a cláusula (um estado permanente de mudança), mas não pode alcançar nenhum dos outros caminhos conectados ao dispositivo de cláusula. Por fim, depois de atravessar através

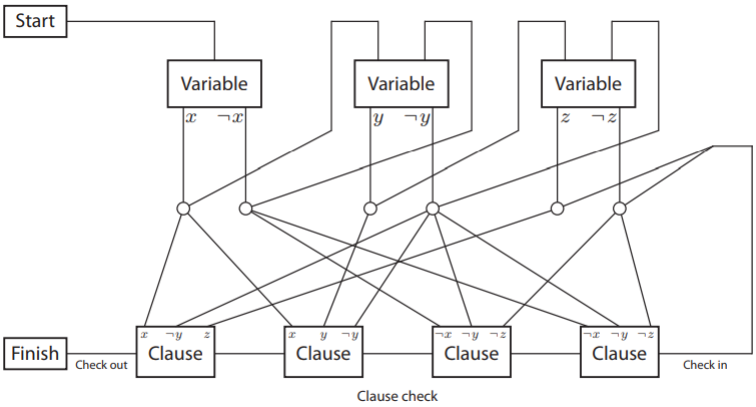

Figure 4. Framework geral para prova da NP Completude

de todos os dispositivos de variáveis, chegará a posição final. O jogador pode passear o caminho de checagem se e somente se cada cláusula for desbloqueada por algum literal. Portanto, basta implementar os dispositivos citados para provar a NP-Hardness de qualquer jogo de plataforma.

Os dispositivos devem seguir as seguintes propriedades:

Começo e Fim: Os dispositivos de começo e fim contém o ponto inicial e o objetivo do personagem, respectivamente.

Variável: Cada dispositivo de variável precisa forçar o jogador a escolher um entre dois caminhos, correspondendo a (xi) ou sua negação $\neg xi$ sendo escolhidos como o literal satisfeito, como por exemplo o caminho que foi escolhido ou o caminho que não pode ser transposto. Cada dispositivo de variável deve ser acessível por apenas e tão somente o dispositivo de variável anterior, independentemente da escolha feita no dispositivo anterior, no qual o caminho a entrada de um literal não permita a travessia de volta para a negação do literal.

Cláusula e Checagem: Cada dispositivo de cláusula deve ser acessível do (e inicialmente, apenas do) caminho vindo do literal correspondente aos literais que aparecem na cláusula da fórmula Booleana. Além disso, quando um jogador visita o dispositivo de cláusula dessa maneira, ele deve realizar alguma ação que ”destrave” o dispositivo. O caminho de checagem atravessa todo dispositivo de cláusula em sequência, e o jogador pode passar em todos os dispositivos de cláusula através do caminho de checagem se e apenas se o dispositivo de cláusula estiver desbloqueado. Assim o caminho de verificação pode ser totalmente atravessado apenas se todos os dispositivos de variável tiverem sido visitadas dos caminhos de literais. Se o jogador atravessar todo o caminho de verificação, ele pode acessar o dispositivo de fim.

Crossover: O dispositivo Crossover deve permitir a travessia através de duas passagens que se cruzam, de tal forma que não há vazamento entre elas.

4. Prova da NP-Completude

Para realizar a prova da NP-Completude de Zelda, vamos primeiro demonstrar que tal problema é NP-Hard, reduzindo-o para o problema 3SAT. Depois, mostraremos que é NP.

Como por definição o conjunto de problemas NP-Completo é a intersecção dos conjuntos de problemas NP com o conjunto de problemas NP-Hard, teremos provado que Zelda é NP-Completo.

4.1. NP-Hard

Teorema 1. *É NP-Hard decidir quando uma posição final é alcançável a partir de uma dada posição inicial em uma versão generalizada de The Legend Of Zelda: A Link to the Past.*

Proof. Existem várias maneiras de demonstrar esse teorema, uma vez que se uma parte do jogo for NP-Completa, todo o jogo também será. Para isso, utilizaremos apenas baús de tesouro e blocos, que servirão como alvo do gancho (vamos assumir que Link começa com esse item). Para realizar a prova, basta descrevermos o ambiente do jogo de acordo com os dispositivos do framework apresentado. Como esse não é um jogo de plataforma, não precisamos de um dispositivo específico de começo e fim, podendo o começo ser uma posição arbitrária dentro de uma caverna (onde normalmente ficam os puzzles que precisam do gancho) e uma posição arbitrária do mapa aberto, respectivamente.

O dispositivo de variável, mostrado na figura 4, funciona da seguinte maneira: Link se aproxima ou do canto superior esquerdo ou do canto superior direito, dependendo de qual valor foi escolhido na variável anterior. Então Link usa o gancho para ir até um baú no centro superior, e por fim utiliza o gancho em um dos dois baús de baixo. Uma vez que Link alcançou um dos baús de baixo, o outro se torna inalcançável. Observe como diversas barreiras ao redor dos corredores impedem link de usar o gancho em outros baús de direções indesejáveis.

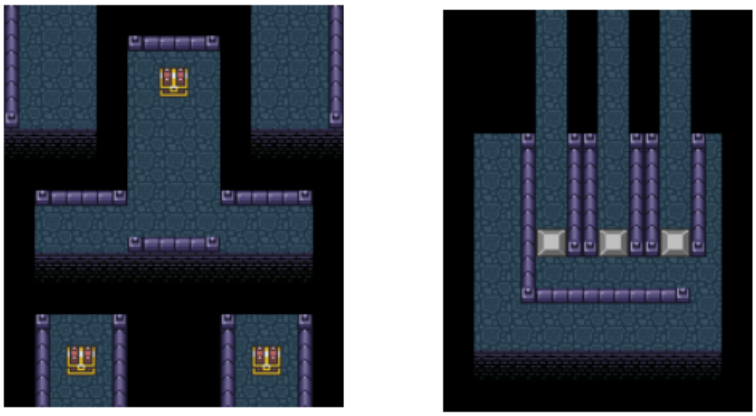

Figure 5. Dispositivos de variável e cláusula, respectivamente

O dispositivo de cláusula é ilustrado também na figura 4. Os três corredores de cima correspondem aos literais que aparecem na cláusula. Quando link visita um desses corredores, ele deve empurrar o bloco para frente, o que permite a ele utilizar o gancho em um dos blocos da direita depois, quando estiver atravessando o caminho de checagem (figura 5). Note que a barreira mais a esquerda do dispositivo de cláusula previne Link de ”pular” clausulas não satisfeitas, mesmo se ele puder usar o gancho arbitrariamente a longas distâncias.

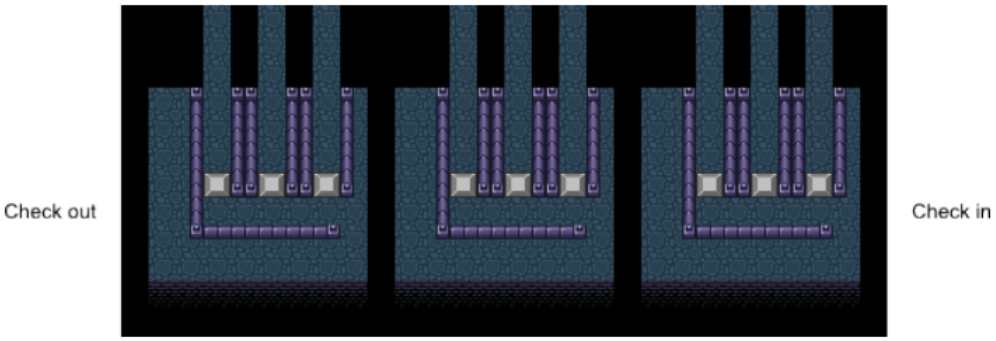

Figure 6. Caminho de checagem

Por fim, o dispositimo de crossover já é nativamente implementado no jogo, como mostra a figura 6.

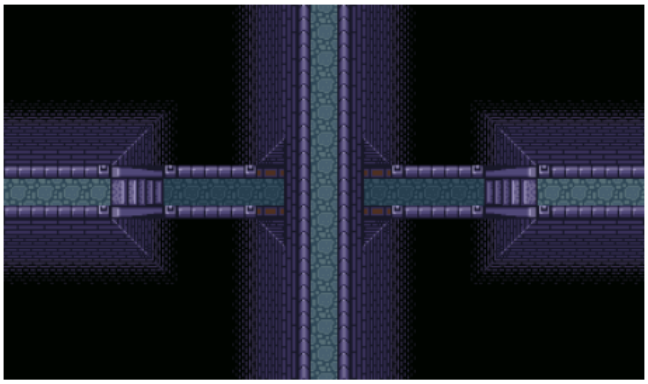

Figure 7. Dispositivo de Crossover

□

4.2. NP

Teorema 2. *Decidir quando uma posição final é alcançável a partir de uma dada posição inicial em uma versão generalizada de The Legend Of Zelda: A Link to the Past é NP.*

Proof. Para concluir a prova da NP-Completude de Zelda, basta provarmos que finalizar o jogo é um problema NP. Se mostrarmos um caminho que é sempre possível, então há um algoritmo de solução de localização de caminho com um tempo de execução que é no máximo polinomial no tamanho da entrada, ou seja, podemos mostrar por certificados que o problema é verificável em tempo polinomial, o que por definição o torna um problema NP.

Então, suponde que existe tal solução, vamos considerar qualquer caminho que Link pode fazer no mapa. Uma vez que só é possível abrir os baús que contém itens únicos do jogo uma vez, e que se matarmos todos os inimigos pelo menos uma vez teremos matado todos os inimigos que soltam itens especiais e também o chefe final do jogo, podemos dizer que o caminho que passa por cada um dos baús atualmente alcançáveis,

e elimina cada um dos inimigos alcançáveis uma vez (lembrando que todos os inimigos podem ser derrotados desde o primeiro momento do jogo), então existe uma solução polinomial ao tamanho da entrada.

Tal demonstração é similar as feitas por [Gabrielsen 2012] para outros jogos da Nintendo (Super Mario Bros., Donkey Kong Country e Metroid).

□

5. Conclusão

Neste artigo, mostramos como jogar The Legend of Zelda: A Link to the Past é um problema NP-Completo. O framework de prova utilizado e a demonstração que tal problema é NP poderiam ser utilizados para demonstrar que qualquer jogo da série é NP-Completo, contanto que utilizem os mesmos elementos (mapa aberto, gancho de movimentação e restrições de caminhos).

Esse tipo de trabalho é interessante para mostrar como pode ser computacionalmente custoso criar um algoritmo para jogar certos jogos, expondo a necessidade de se buscar soluções alternativas como por exemplo algoritmos genéticos, redes neurais ou programação de agentes para obter uma solução boa.

Por fim, é interessante avaliar como é acessível esse tipo de prova, mostrando que um estudante que tenha a base da teoria da computação pode demonstrar que jogar seu jogo favorito é (ou não) NP-Completo.

References

Aloupis, G., Demaine, E. D., Guo, A., and Viglietta, G. (2015). Classic nintendo games are (computationally) hard. *Theoretical Computer Science*, 586:135–160.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM.

Cormode, G. (2004). The hardness of the lemmings game.

Demaine, E. D., Demaine, M. L., and O'Rourke, J. (2000). Pushpush and push-1 are np-hard in 2d. *arXiv preprint cs/0007021*.

Forišek, M. (2010). Computational complexity of two-dimensional platform games. In *International Conference on Fun with Algorithms*, pages 214–227. Springer.

Gabrielsen, C. (2012). Video game motion planning reviewed np-complete.

Viglietta, G. (2014). Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621.