Graphical Abstract

Uma demonstração de que o jogo Sid Meier's Civilization: Beyond Earth é Turing Completo

Nicholas Wojeicchowski, Lucas Gonçalves Brach

Highlights

Uma demonstração de que o jogo Sid Meier's Civilization: Beyond Earth é Turing Completo

Nicholas Wojeicchowski, Lucas Gonçalves Brach

- Uma máquina de Turing é o modelo computacional mais poderoso conhecido
- Para afirmar que um modelo de computação é Turing Completo devemos construir uma máquina de Turing Universal para ele
- A partir das regras internas do jogo Sid Meier's Civilization: Beyond Earth podemos simular uma máquina de Turing[1]
- Há diversos exemplos de aplicativos que são Turing completos, como o jogo de cartas Magic: The Gathering[2], o Microsoft PowerPoint[3], entre outros
- O modelo de computação criado a partir do jogo Sid Meier's Civilization: Beyond Earth é menos eficiente que uma máquina de Turing tradicional

Uma demonstração de que o jogo Sid Meier's Civilization: Beyond Earth é Turing Completo

Nicholas Wojeicchowski[a], Lucas Gonçalves Brach[a]

[a]*Departamento de Ciência da Computação - Universidade do Estado de Santa Catarina, R. Paulo Malschitzki - 200, Joinville, 89219-710, Santa Catarina, Brasil*

Abstract

No contexto da ciência da computação, um modelo de computação é dito Turing Completo, ou universalmente computável se e somente se pode ser usado para manipular qualquer máquina de Turing. Em outras palavras podemos definir que um modelo de computação é Turing completo se é possível simular uma máquina de Turing a partir dele e se a partir de uma Máquina de Turing podemos simulá-lo também. O presente trabalho trata-se de uma análise do jogo digital Sid Meier's Civilization: Beyond Earth e a prova da sua Turing Completude, bem como uma explicação superficial das mecânicas do jogo, exemplos de algoritmos construídos com a partir das regras de computação definidas pelo jogo e uma comparação entre uma Máquina de Turing tradicional e o modelo de computação construído a partir do jogo.

Keywords: Turing Completude, Máquina de Turing Universal, Sid Meier's Civilization, Simulação, Teoria da Computação

1. Introdução

Nos dias atuais é quase impossível realizar qualquer tarefa do dia a dia sem o auxílio de um computador, seja ele um Desktop ou até mesmo um dispositivo móvel como um smartfone ou tablet, mas ouve um tempo em que o conceito de computador era apenas uma ideia abstrata e teórica, e é sobre essa ideia que o presente trabalho fala a respeito. Os nossos atuais computadores surgiram de um modelo de computação muito simples, a Máquina de Turing, uma Máquina de Turing é um dispositivo que é essencialmente um autômato finito com uma fita de dimensões infinitas em que é possível ler e escrever dados [4].

Em 1936, Alan Turing propôs a máquina de Turing como um modelo capaz de computar qualquer computação possível [5], entretanto vale ressaltar que esse não é o único modelo de computação capaz de fazer isso, todas as propostas sérias para um modelo de computação possuem o mesmo poder computacional (como a Máquina de Post e o Cálculo Lambda). A tese de Church-Turing diz que qualquer forma de computação permite computar apenas as funções μ-recursivas [4].

Quando um modelo possui a capacidade de computar qualquer computação possível ele é chamado de Turing Completo, a seção 3 abordará esse tema novamente com mais detalhes.

O modelo de proposto por Turing consiste um controlador finito e uma fita infinita dividida em células, em que cada célula armazena um único símbolo de um conjunto finito de símbolos possíveis, sendo uma célula a posição corrente do cabeçote de leitura e escrita. Além disso, faz movimentos baseados no estado atual da máquina e no símbolo lido da fita pelo cabeçote, de forma que a cada movimento, o estado possivelmente é alterado, um símbolo é sobrescrito na posição corrente do cabeçote e este move uma célula para esquerda ou direita. A computação inicia com a palavra de entrada, que consiste de uma sequência finita de símbolos disposta na fita, sendo as demais células preenchidas com o símbolo especial de branco. A máquina para aceitando assim que atinge um estado de aceitação, e para rejeitando assim que atinge um estado de rejeição[4]. Uma definição formal e mais aprofundada sobre esse dispositivo será dada na seção 3.

Para afirmar que dois modelos computacionais são equivalentes em poder computacional é preciso que eu consiga simular um modelo no outro. Variantes do modelo da máquina de Turing, como multi-fita, movimento estacionário ou não-determinismo apresentam todos o mesmo poder computacional, ou seja, aceitam a mesma classe de linguagens [6].

A proposta desse artigo é justamente provar que o jogo Sid Meier's Civilization: Beyond Earth é equivalente em poder computacional a uma máquina de Turing, e portanto afirmar que o jogo é Turing Completo, para isso é apresentado uma forma de simular uma máquina de Turing usando as mecânicas e regras internas do jogo, tais regras serão acuradas na seção 2.

As seções 4, 5 e 6 apresentam respectivamente, exemplos de uma máquina de Turing mapeada para o jogo, uma comparação entre a máquina de Turing tradicional e a máquina de Turing simulada no jogo e uma conclusão final sobre o trabalho.

2. Sobre o jogo

Desenvolvido por Sid Meier e Firaxis Games e publicado em 2014 pela Take-Two Interactive, *Civilization: Beyond Earth* é um *spin-off* da série de jogos digitais *Civilization*, caracterizada pelo gênero de estratégia baseada em turnos e nas mecânicas de 4X (explorar, expandir, extrair e exterminar), sendo o mais longevo e popular jogo multijogador de simulação de alta estratégia com vasto detalhamento histórico [7]. A ideia essencial dos jogos principais da série é construir um império capaz de resistir ao tempo, iniciando com o estabelecimento de uma civilização na Idade da Pedra e liderando o desenvolvimento até a Era da Informação, atravessando guerras, diplomacia e evolução tecnológica, econômica e cultural.

2.1. Enredo e características

Sid Meier's Civilization: Beyond Earth é tematizado em ficção científica e se passa num futuro incerto em que eventos globais levaram à desestabilização e consequente colapso da sociedade contemporânea, sujeitando a humanidade a buscar um novo início na imensidão do espaço e na colonização e desenvolvimento em planetas extraterrestres.

Os jogadores são responsáveis por uma colônia e têm o objetivo de atingir alguma das condições de vitória, como Contato, Dominação e Afinidade, evidenciando a boa gestão dos recursos e relacionamento com as demais colônias, visto que há também o combate, que ocorre entre duas entidades políticas que estão em guerra entre si, mais precisamente, uma colônia pode estar em guerra com outra colônia.

O mapa do jogo é finito e composto por divisórias hexagonais, podendo manifestar-se como diversos tipos de terreno: desertos, planícies, pradarias, colinas, e outros. Os hexágonos podem ainda ter modificadores como florestas, neve, rios, pântanos, e até contaminação alienígena.

As cidades são vitais para o sucesso da sua civilização. Elas permitem a pesquisa de novas tecnologias e acúmulo de riqueza, além da construção de Unidades, Edificações e Maravilhas: Unidades são elementos que podem se mover pelo mapa, podem ser classificadas em combatentes ou não combatentes, nesta última destacam-se os Trabalhadores, que encarregam-se de construções e plantações; cidades possuem mais que apenas casas, Edificações podem ser laboratórios, instalações, galpões e outros que representam as melhorias feitas à cidade; e Maravilhas que são edificações, invenções e conceitos especiais que requerem mais esforços para serem completados. [7]

2.2. Regras básicas

O jogo é multi jogador, e é separado em turnos, onde que em cada turno, o jogador executa todas ou algumas das ações disponíveis, como construir melhorias (Unidades, Edificações e Maravilhas), pesquisar tecnologias, melhorar políticas sociais e se concentrar na gestão geral da sua civilização.

Para cada melhoria construída é necessário uma quantidade determinada de turnos, além disso cada melhoria produz um recurso relacionado a ela, e esses recursos são utilizados para manter as unidades e edifícios que o jogador possui, além de possibilitar a aquisição de tecnologias e políticas sociais mais avançadas. Alguns exemplos desses recursos são Alimentos, Produção, Cultura e Ciência.

Uma ou mais unidades controladas pelo jogador, conhecidas como Trabalhadores, geralmente são encarregadas de construir melhorias em um terreno, o que implica diretamente no rendimento de recursos desse jogador. Por exemplo, para adquirir novas tecnologias precisa-se de uma certa quantidade do recurso ciência, para isso os Trabalhadores constroem melhorias nos terrenos (por exemplo, uma universidade) para aumentar o rendimento de ciência.

Vale ressaltar que os Trabalhadores só podem construir melhorias em terrenos pertencentes a uma Cidade. As cidades produzem mais cidadãos para trabalhar os terrenos e suas melhorias. Os jogadores são capazes de gerenciar a colocação de cidadãos para alterar seu rendimento de recursos. [7]

3. Demonstrando a Turing Completude do jogo

3.1. Definição da Máquina de Turing

O modelo de computação de propósito geral concebido por Turing é como um autômato finito que executa uma série de instruções predefinidas, porém dispõe de memória infinita e capacidade tanto de ler quanto escrever em qualquer posição desejada. Sipser [6] define a máquina de Turing como uma 7-tupla (Q, Σ, Γ, δ, q_0, q_a, q_r), de forma que:

1. Q é o conjunto de estados,
2. Σ é o alfabeto de entrada, tal que $\sqcup \notin \Sigma$,
3. Γ é o alfabeto da fita, tal que $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{E, D\}$ é a função programa,
5. $q_0 \in Q$ é o estado inicial,

6. $q_a \in Q$ é o estado de aceitação,
7. $q_r \in Q$ é o estado de rejeição, sendo $q_a \neq q_r$.

Uma máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ recebe como entrada uma palavra $w = w_1 w_2 ... w_n \in \Sigma^*$ nas n células mais à esquerda, o resto da fita é preenchida pelo símbolo especial de branco $\sqcup$. Para iniciar o processo de computação o cabeçote é posicionado na célula mais à esquerda da fita, em sequência ocorrem as transições de acordo com as regras da função programa δ, em que o símbolo lido da fita e o estado interno atual da máquina descrevem o estado seguinte, o símbolo a ser escrito na posição atual e o sentido do movimento para esquerda ou direita, continuando assim até atingir um estado de aceitação ou rejeição, podendo inclusive continuar infinitamente.

3.2. Máquinas de Turing Universais

Alan Turing argumenta que uma máquina U é dita universal quando essa recebe como entrada um outra máquina de Turing codificada M, e uma palavra w e faz-se o processamento de M com entrada w. Logo a máquina U é capaz de simular qualquer outra máquina de Turing.[5]

Observe que para criar uma máquina de Turing universal eu preciso de somente 2 símbolos (0 e 1 por exemplo, além do símbolo branco), pois qualquer outro símbolo pode ser convertido para binário.

Segundo Rogozhin[8], a definição formal de uma máquina de Turing U pode ser dada por:

$$M(c) = h(U(f(g(M, c))))$$

Onde M é qualquer máquina de Turing, c é a configuração de M, g é o número de Gödel[1] para o par (M, c), f é uma função que mapeia $g(M, c)$ para uma configuração de U e por último h que mapeia a configuração de U para a configuração de M.

Além disso, a função f é injetiva e a função h é total e sobrejetiva. Vale ressaltar que, a função de codificação f e a função de decodificação h são ambas recursivas. Este último requisito, garante que a universalidade esteja na máquina que afirmamos ser universal e não nas funções de codificação ou decodificação.

[1] O número de Gödel é um função que para cada simbolo ou fórmula de uma determinada linguagem atribui-se um número natural [9]

Atualmente há diversos estudos para a definição de máquinas de Turing com símbolos e estados mínimos, chegando a máquinas com apenas 2 símbolos e 2 estados [10], o presente trabalho irá usar a máquina de Turing Universal definida por Rogozhin [8] que possui 3 símbolos e 10 estados, chamada de U_{mt}(3:10), a figura 2 mostra o diagrama completo dessa máquina

3.3. Construindo uma máquina de Turing Universal com as regras do jogo

Para simular uma máquina de Turing com as regras do jogo, primeiramente deve-se fazer duas suposições[1]:

Suposição 1. *No jogo há infinitos turnos e somente um jogador na partida*

Suposição 2. *O mapa cresce infinitamente para todos os lados*

A suposição 1 deve-se ao fato de não limitar o jogo ao um número finito de turnos, pois uma máquina de Turing pode eventualmente entrar em *loop*. E não pode haver mais de um jogador na partida pois não pode haver influência externa durante a computação de uma máquina de Turing. A suposição 2 precisa ser considerada pois o mapa servirá como fita para a máquina de Turing que será simulada no jogo.

Conforme as mecânicas e regras de *Sid Meier's Civilization: Beyond Earth*, dentre as atividades possíveis que podem ser desenvolvidas por Trabalhadores, destacam-se a construção e destruição de Estradas, a Pilhagem, e a construção e destruição de *Terrascapes*, que trata-se de um tipo de fazenda otimizada para alta eficiência e capaz de fornecer Cultura[7]. Mais adiante veremos a importância destas características para a simulação de uma máquina de Turing.

A construção da Máquina de Turing Universal(U_{mt}) é baseada na ação de dois Trabalhadores: um é responsável pela alternância de estados e o outro pela operação na fita, ou seja, o cabeçote. A dupla desenvolve seu trabalho em seções disjuntas do mapa. Para a fita é considerada a infinitude dos hexágonos que não são propriedade do jogador, já os hexágonos que demarcam os estados precisam estar em sua posse para permitir a construção das melhorias avançadas, nesse caso os *Terrascapes*. Vale ressaltar que as operações de construção, reparação ou remoção de melhorias consome possivelmente alguns turnos.[1]

3.3.1. Simulando a fita

A fita constitui-se de uma sequência contínua de hexágonos pelo mapa. Tendo em vista que terrenos irregulares, como por exemplo morros e montanhas, requerem mais pontos para atravessar, assumiremos que o mapa é composto apenas de planícies. As unidades Trabalhador são dotadas da capacidade de construir/reparar, remover e pilhar Estradas, desta forma o conjunto Γ de símbolos da fita corresponde à situação do hexágono, ou seja, Γ = {Pilhada, Estrada, Vazio}. Para a construção da U_{mt} teremos a respectiva correspondência: $\Gamma = \{0, 1, \sqcup\}$.[1]

3.3.2. Simulando os estados

O estado atual é formado pelo número de hexágonos com a melhoria *Terrascape* em posse do jogador, desta forma, a cada transição são construídas ou removidas tantas melhorias quantas forem necessárias para corresponder ao estado indicado pela função programa δ. Para a construção da U_{mt} assumimos que o jogador possui ao menos nove hexágonos disponíveis para manutenção das *Terrascapes*, portanto Q = {0, ..., 9}. Uma forma equivalente de identificar o estado é através da quantidade de Cultura normalizada gerada no turno, visto que este recurso é obtido pelas *Terrascapes*.[1]

A figura 1 mostra como fica o mapa do jogo simulando uma máquina de Turing. Repare na figura que os terrenos hexagonais destacados em preto mostram a fita com o simbolo "Estrada", a célula destacada em verde, mostra a posição do cabeçote, e o hexágono em vermelho mostra a célula com o simbolo $\sqcup$. No canto inferior direito temos uma área roxa, essa área é a área de posse do jogador e os hexágonos em cinza são os hexágonos com a melhoria *Terrascape*, ou seja, o estado atual da máquina é o estado q_2, o terreno em amarelo mostra o trabalhador responsável pelo controle dos estados, que irá construir/destruir os *Terrascapes*. A configuração em um Máquina de Turing correspondente a imagem seria $1111111q_211\sqcup1111$.

Figura 1: Exemplo de uma simulação no mapa do jogo, adaptado de [1]

Com base nas regras definidas nas subseções 3.3.1 e 3.3.2 podemos criar uma função programa para $U_{mt}(3{:}10)$ e dessa forma provar que a partir das suposições 1 e 2 *Sid Meier's Civilization: Beyond Earth* é Turing completo. A tabela 1 mapeia cada função programa da máquina $U_{mt}(3{:}10)$ definida no diagrama da figura 2 para uma determinada ação no jogo.

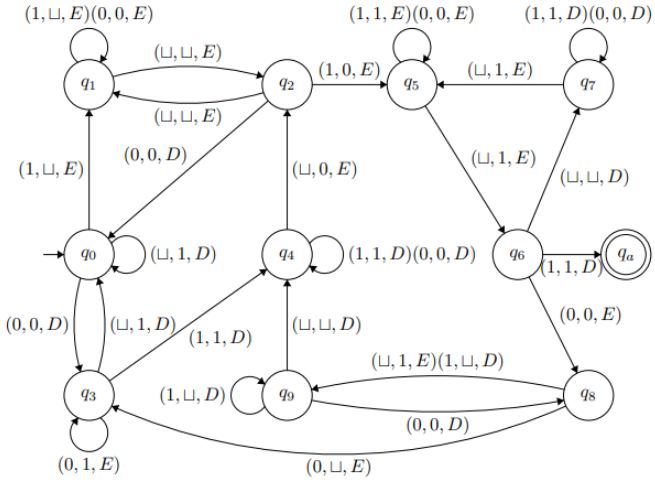

Figura 2: Diagrama de uma máquina de Turing Universal com 10 estados e com 3 símbolos, adaptado de [1]

Civ:BE (Q, Γ)	Comando (Q, Γ, {E, D})	U_{mt}(3:10)
0, Pilhada	c. 3 T, *na*, D	$q_0, 0, 0, D, q_3$
0, Estrada	c. 1 T, r. Melhoria, E	$q_0, 1, \sqcup, E, q_1$
0, Vazio	*na*, c. Estrada, D	$q_0, \sqcup, 1, D, q_0$
1, Pilhada	*na*, *na*, E	$q_1, 0, 0, E, q_1$
1, Estrada	*na*, r. Melhoria, E	$q_1, 1, \sqcup, E, q_1$
1, Vazio	c. 1 T, *na*, E	$q_1, \sqcup, \sqcup, E, q_2$
2, Pilhada	r. 2 T, *na*, D	$q_2, 0, 0, D, q_0$
2, Estrada	c. 3 T, pilha Estrada, E	$q_2, 1, 0, E, q_5$
2, Vazio	r. 1 T, *na*, E	$q_2, \sqcup, \sqcup, E, q_1$
3, Pilhada	*na*, repara Estrada, E	$q_3, 0, 1, E, q_3$
3, Estrada	c. 1 T, *na*, D	$q_3, 1, 1, D, q_4$
3, Vazio	r. 3 T, c. Estrada, D	$q_3, \sqcup, 1, D, q_0$
4, Pilhada	*na*, *na*, D	$q_4, 0, 0, D, q_4$
4, Estrada	*na*, *na*, D	$q_4, 1, 1, D, q_4$
4, Vazio	r. 2 T, c. e pilha Estrada, E	$q_4, \sqcup, 0, E, q_2$
5, Pilhada	*na*, *na*, E	$q_5, 0, 0, E, q_5$
5, Estrada	*na*, *na*, E	$q_5, 1, 1, E, q_5$
5, Vazio	c. 1 T, c. Estrada, E	$q_5, \sqcup, 1, E, q_6$
6, Pilhada	c. 2 T, *na*, E	$q_6, 0, 0, E, q_8$
6, Estrada	c. 4 T, *na*, D	$q_6, 1, 1, D, q_a$
6, Vazio	c. 1 T, *na*, D	$q_6, \sqcup, \sqcup, D, q_7$
7, Pilhada	*na*, *na*, D	$q_7, 0, 0, D, q_7$
7, Estrada	*na*, *na*, D	$q_7, 1, 1, D, q_7$
7, Vazio	r. 2 T, c. Estrada, E	$q_7, \sqcup, 1, E, q_5$
8, Pilhada	r. 5 T, r. Melhoria, E	$q_8, 0, \sqcup, E, q_3$
8, Estrada	c. 1 T, r. Melhoria, D	$q_8, 1, \sqcup, D, q_9$
8, Vazio	c. 1 T, c. Estrada, E	$q_8, \sqcup, 1, E, q_9$
9, Pilhada	r. 1 T, *na*, D	$q_9, 0, 0, D, q_8$
9, Estrada	*na*, r. Melhoria, D	$q_9, 1, \sqcup, D, q_9$
9, Vazio	r. 5 T, *na*, D	$q_9, \sqcup, \sqcup, D, q_4$

Tabela 1: Mapeamento da função programa da máquina U_{mt}(3:10), adaptada de [8]. A primeira coluna corresponde ao *estado atual, símbolo lido* no mapeamento da MT para o jogo. A segunda coluna corresponde ao comando *estado seguinte, símbolo escrito, movimento* fornecido aos Trabalhadores, em que *c.*, *r.*, T e *na* significam respectivamente constrói, remove, *Terrascape* e *não altera*. A terceira coluna corresponde à função programa de U_{mt}(3:10) na disposição *estado atual, símbolo lido, símbolo escrito, movimento, estado seguinte*. O estado de aceitação $q_a$foi definido como q_{10}.

4. Exemplos

Essa seção é dedicada a exemplos de máquinas de Turing e seu mapeamento para as regras do jogo, como na seção acima já foi provado que o jogo pode simular qualquer máquina de Turing, essa seção serve apenas para complementar o entendimento.

Seja MT A uma máquina de Turing que reconhece a linguagem $L = \{ 0^n1^n \mid n \in \mathbb{N} \}$, a figura 3 mostra do diagrama dessa máquina de Turing e a tabela 2 mostra o mapeamento para as regras do jogo.

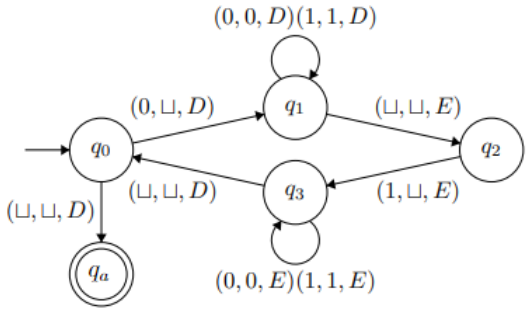

Figura 3: Diagrama da MT A

Civ:BE (Q, Γ)	Comando (Q, Γ, {E, D})	MT A
0, Pilhada	c. 1 T, r. Melhoria, D	$q_0, 0, \sqcup, D, q_1$
0, Vazio	c. 10 T, na, D	$q_0, \sqcup, \sqcup, D, q_a$
1, Pilhada	na, na, D	$q_1, 0, 0, D, q_1$
1, Estrada	na, na, D	$q_1, 1, 1, D, q_1$
1, Vazio	c. 1 T, na, E	$q_1, \sqcup, \sqcup, E, q_2$
2, Estrada	c. 1 T, r. Melhoria, E	$q_2, 1, \sqcup, E, q_3$
3, Pilhada	na, na, E	$q_3, 0, 0, E, q_3$
3, Estrada	na, na, E	$q_3, 1, 1, E, q_3$
3, Vazio	r. 3 T, na, D	$q_3, \sqcup, \sqcup, D, q_0$

Tabela 2: Mapeamento das funções programas para a máquina de Turing A

Seja MT B uma máquina de Turing que reconhece a linguagem $L =$ {todos os números pares em binário}. A figura 4 mostra o diagrama da máquina e a tabela 3 mostra o mapeamento correspondente para o jogo.

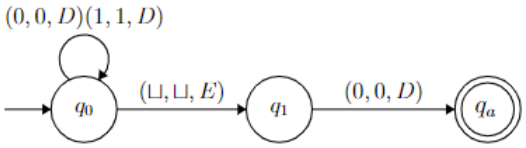

Figura 4: Diagrama da MT B

Civ:BE (Q, Γ)	Comando (Q, Γ, {E, D})	MT B
0, Pilhada	*na*, *na*, D	$q_0, 0, 0, D, q_0$
0, Estrada	*na*, *na*, D	$q_0, 1, 1, D, q_0$
0, Vazio	c. 1 *T*, *na*, E	$q_0, \sqcup, \sqcup, E, q_1$
1, Pilhada	c. 9 *T*, *na*, D	$q_1, 0, 0, D, q_a$

Tabela 3: Mapeamento da MT B para as regras do jogo

5. Comparando os modelos de computação

A seção 3 mostra que podemos simular qualquer máquina de Turing no jogo, com isso podemos afirmar que ambas possuem o mesmo desempenho computacional [6]. Entretanto o mesmo não é valido para desempenho de tempo, por isso essa análise torna-se relevante. A tabela 1 nos mostra que a máquina de Turing U_{mt}(3:10) vai do estado q_9 para o estado q_4, sendo essa transição a de maior diferença entre os estados. Como cada estado q_i da máquina é simulado no jogo com $_i$ sendo o número de *Terrascapes* construídos em nossa posse, sabemos que para essa máquina devemos construir/remover até 5 *Terrascapes* em uma única transição.

Suponto que cada *Terrascapes* necessite de t turnos e cada estrada necessite de n para serem construídos/removidos, o tempo máximo para cada instrução será de no máximo $5m+n$ turnos, fazendo com que a U_{mt}(3:10) do jogo seja temporalmente menos eficiente do que a U_{mt}(3:10) tradicional por um fator constante. [1]

6. Considerações finais

Este artigo procurou trabalhar em cima da seguinte questão, "O jogo *Sid Meier's Civilization: Beyond Earth* é Turing completo?" As principais colocações para essa questão foi de que sim, é Turing Completo, é possível

representar qualquer máquina de Turing com as regras do jogo, inclusive uma máquina de Turing Universal, que irá simular qualquer outra máquina de Turing. Dada a afirmação anterior podemos concluir que se a suposição 2 da seção 3 for válida o jogo *Sid Meier's Civilization: Beyond Earth* é indecidível, visto que se fosse decidível, implicaria que o problema da parada é decidível, o que é um absurdo [6]. Este trabalho se limita quando refere-se a aplicação prática de seus conceitos, visto que a ideia da construção de uma máquina de Turing usando o jogo não passa de um ideia teórica e hipotética, pois não é possível de fato construir algoritmos com o jogo, sem contar que a Turing completude do jogo só é possível a partir de duas suposições que na prática são impossíveis de se alcançar no jogo. Apesar disso, é de extrema importância o estudo de ideias como a apresentada nesse trabalho, pois mostra que é possível associar conceitos teóricos da computação, com várias coisas do nosso dia a dia, seja uma ferramenta de trabalho, um software de edição de imagens ou até mesmo um jogo digital.

Referências

[1] A. de Wynter, Turing completeness and sid meier's civilization, CoRR (2021).
URL `https://arxiv.org/abs/2104.14647`

[2] A. Churchill, S. Biderman, A. Herrick, Magic: The gathering is turing complete, CoRR (2019).
URL `http://arxiv.org/abs/1904.09828`

[3] T. Wildenhain, On the turing completeness of ms powerpoint, The Official Proceedings of the Eleventh Annual Intercalary Workshop about Symposium on Robot Dance Party in Celebration of Harry Q Bovik's (2017).

[4] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley Publishing Company, 2000.

[5] A. M. Turing, On computable numbers, with an application to the entscheidungsproblem, Proceedings of the London Mathematical Society s2-42 (1) (1937) 230–265.

[6] M. Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.

[7] FiraxisGames, Sid meier's civilization beyond earth manuals (2014). URL `https://www.2k.com/manual/civbe/index.html`

[8] Y. Rogozhin, Small universal turing machines, Bulletin of Symbolic Logic (2003).

[9] R. Zach, Kurt gödel, 'über formal unentscheidbare sätze der principia mathematica und verwandter systeme i' (1931), in: Monatshefte fur Mathematik und Physik, 2003, pp. 4–5.

[10] S. Cooper, B. Löwe, A. Sorbi, Computation and Logic in the Real World: Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007, Proceedings, Springer Berlin, Heidelberg, 2007.