Investigação do jogo Shapez.io para a verificação de sua Turing Completude

Marcos Paulo F. De Souza
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
mp022@ufmg.br

Orientador:
Vinicius Fernandes dos Santos

***Resumo*—Este artigo investiga a Turing Completude do jogo "Shapez.io". Para isso, foi suficiente demonstrar a existência da porta lógica NAND e de um sistema de "clock" para esta prova, pois esses são os elementos essenciais que conceitualmente permitem a implementação de uma máquina de Turing Universal [1]. Ao relacionar esse tema com o jogo, foi possível identificar que essa é uma questão levantada pela comunidade do "Shapez.io" há anos. Muitos participantes da comunidade alegam ter construído máquinas que comprovam a completude do jogo, uma delas é a máquina implementada por Antoine Dragnir [2]. Com essa máquina é possível demonstrar como implementar uma máquina de Turing Universal, e por isso ela foi analisada e explicada. Com isso, este artigo verifica a Turing completude do jogo "Shapez.io", provando a possibilidade da implementação da porta lógica NAND e de um sistema de "clock", e demonstra a possibilidade de implementar uma máquina de Turing Universal, ao apresentar a máquina de Antoine Dragnir que pode ser modificada para se tornar uma máquina de Turing Universal.**

***Index Terms*—Turing Completude, Shapez.io, Máquina de Turing, Máquina de Turing Universal, Jogos Digitais.**

I. Introdução

Um sistema Turing Completo pode ser descrito como qualquer sistema que possibilite a implementação de uma Máquina de Turing Universal (MTU) [3]. Portanto, precisamos demonstrar que o "Shapez.io" suporta uma MTU para comprovarmos a sua Turing Completude. A MTU foi criada por Alan Turing em 1936 no artigo "On Computable Numbers, with an Application to the Entscheidungsproblem" [4]. Por se tratar de um artigo antigo, usaremos as definições da terceira edição do livro "Introduction to the Theory of Computation" escrito por Michael Sipser em 2013 [5]. No livro de Sipser uma MTU é descrita como uma Máquina de Turing (MT) capaz de simular outras MT, sendo a MT uma máquina conceitual capaz de resolver qualquer problema computável [5]. Ela é composta por uma fita infinita, uma cabeça e uma função de transição. A fita funciona como a memória, estando inicialmente preenchida com a entrada da máquina, que poderá ser alterada durante o processamento. A cabeça pode mover-se, ler e escrever na fita a partir dos estados que serão assumidos pela máquina. A função de transição, a partir do estado atual e do que está sendo lido na fita, determina como a cabeça irá se comportar, qual será o próximo estado a ser assumido e o que deve ser escrito na fita.

Para comprovar a Turing Completude de um sistema não é necessário implementar uma MTU, basta provar que é possível fazê-lo. Para isso, é suficiente mostrar que o sistema possui os componentes básicos para a construção da MTU. Esses componentes são a porta lógica NAND, que é necessária para termos memória e o processamento da MTU, e o sistema de "clock", que tem a função de registrar a passagem do tempo [1]. Após a seção de referencial teórico, que explicará todos os conceitos básicos necessários para esse artigo e também como o jogo funciona, será realizada a prova da Turing Completude do jogo. Nas seções seguintes, como uma forma de demostrar como seria a construção de uma MTU, a máquina que é capaz de simular uma MT construída por um usuário da comunidade do "Shapez.io" será destrinchada. Na última seção será feito uma conclusão sintetizando todo o conteúdo do artigo e alguns comentários sobre trabalhos que ainda precisam ser realizados na área.

II. Referencial Teórico

Nessa seção será explicada toda a base teórica e o conhecimento sobre o jogo que são necessários para compreender a verificação da Turing completude do "Shapez.io". Primeiro serão definidas as regras de como o jogo funciona e os seus elementos básicos. Depois toda base teórica, passando pelo conceito de MT, MTU até a explicação de como será feita a prova da Completude do jogo.

A. Shapez.io

O "Shapez.io" foi inspirado no jogo "Factorio" e foi desenvolvido por Tobias Springer [6]. Ele é um jogo de código aberto que foi lançado em junho de 2020 na loja virtual de jogos digitais "Steam". Nele o jogador tem o objetivo de desenvolver fábricas e linhas de produção automatizadas para avançar nos níveis e desbloquear as construções do jogo [7].

A figura 1 representa uma fábrica que produz a forma "blueprint". Essa forma é chamada assim por ser uma moeda para utilizar a mecânica "blueprint" que permite rotacionar, recortar, copiar e colar fábricas feitas anteriormente pelo

jogador, assim evitando o retrabalho na construção das fábricas [8]. Essa figura traz o exemplo do que é uma fábrica de formas do “Shapez.io”.

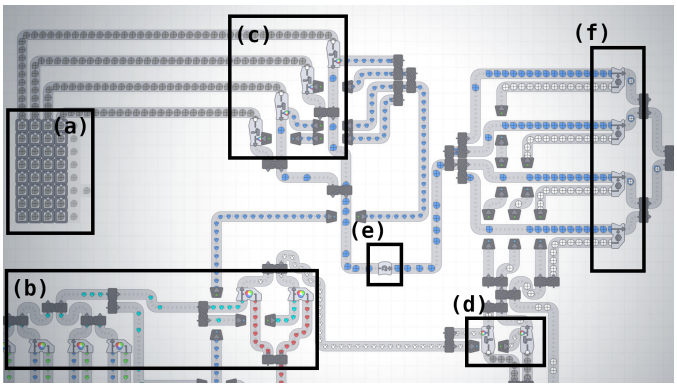

Figura 1. Fonte: https://store.steampowered.com/app/1318690/shapez/
Fábrica de ”blueprint”. Ela é composta por diversas construções que possuem diferentes funções. Os extratores **(a)** retiram a forma “gota” do mapa e as colocam em esteiras transportadoras. Na região **(b)** temos como principais construções as de mistura de tintas, que estão produzindo a tinta branca. As pintoras em **(c)** estão pintando as “gotas” de azul. Em **(d)** temos outras pintoras que estão pintando os círculos de branco. A construção **(e)** rotaciona as “gotas” azuis em 90 graus no sentido horário. Em **(f)** temos a construção de empilhar que posiciona o circulo branco sobre a ”gota” azul. Assim gerando o resultado final desta fábrica, a forma “blueprint”.

A maioria das fábricas que são desenvolvidas no decorrer do jogo são semelhantes a essa, com pouca ou sem nenhuma adaptabilidade sempre tendo as mesmas entradas e produzindo as mesmas saídas. Porém ao ultrapassar o nível 20 do jogo, os fios são desbloqueados abrindo uma quantidade enorme de possibilidades. Ao desbloquear os fios, o jogo que possuía apenas a camada das construções, se expande ganhando a camada dos fios. Ou seja, o mapa do jogo possui duas camadas que interagem entre si mas possuem as suas particularidades.

B. Construções do Shapez.io

Nessa subseção vamos explorar algumas das construções do jogo que serão utilizados durante esse artigo. Elas serão explicadas a partir das camadas que elas tem maior importância. Compreender essas construções permitirá compara-las com os componentes computacionais que elas podem simular, a partir dessas comparações a prova da Turing Completude se tornará trivial de ser verificada.

1) Camada de construções: Nessa camada é permitido colocar e retirar as construções que manipulam os elementos básicos do jogo que são as tintas e as formas. Como o foco desse trabalho é a Turing completude do jogo só explicaremos as construções que são necessárias para alcançar esse objetivo. Todas as construções dessa camada que são relevantes para isso estão representadas nas figuras 2 e 3. Cada uma dessas construções possui uma função definida e as próprias características. As construções mais relevantes para esse trabalho são o armazenamento **(b)** e o empilhador **(f)**, a partir das suas especificidades é possível simular portas lógicas que são essenciais para a Turing completude do jogo sem a utilização das portas lógicas disponibilizadas pelo jogo.

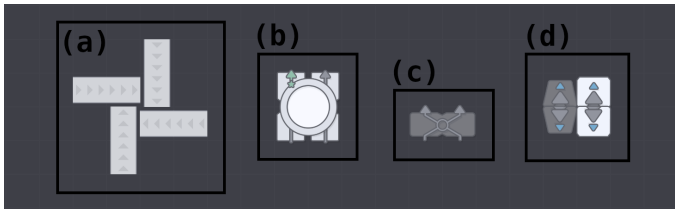

Figura 2. Fonte: De autoria própria
Construções da camada de construções que transportam e armazenam formas e tintas. **(a)** Esteiras que transportam formas em uma única direção. **(b)** Armazenamento capaz de guardar até 5 mil tintas ou formas de um único tipo. As entradas estão na parte inferior e as saídas na parte superior. Os elementos que chegarem pelas entradas serão armazenados enquanto não tiverem uma opção de saída. Dentre as saídas dessa construção a da esquerda possui uma estrela e tem a coloração verde pra indicar que ela possui a preferência. Por isso, teremos elementos na saída da direita apenas se a saída da esquerda estiver retirando o máximo de elementos que ela consegue e ainda tenham elementos para sair do armazenamento. **(c)** Divisor de elementos que possui as entradas na parte inferior e as saídas na parte superior. Ele separa igualmente os elementos que chegam nas suas entradas entre as suas saídas. **(d)** Túneis que passam elementos por baixo da camada, permitindo o transporte de elementos através de outras construções. Aqui temos suas duas versões, a versão da esquerda pode ter 5 quadrados separando a entrada do túnel com a saída e a versão da direito pode ter 9 quadrados separando a entrada e a saída do túnel.

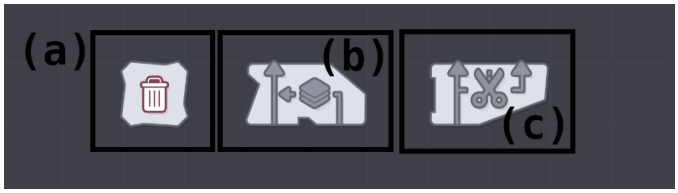

Figura 3. Fonte: De autoria própria
Construções da camada de construções que manipulam formas. **(a)** Lixeira que elimina qualquer elemento que chegue em qualquer um dos seus quatro lados. **(b)** Empilhador que possui as entradas na parte inferior e gera a saída na parte superior, colocando a forma da entrada da direita sobre a forma da entrada da esquerda. Caso uma das entradas não possua formas o empilhador não irá gerar uma saída e não irá consumir a forma recebida na entrada. **(c)** Cortadoras de formas que as separam as formas da entrada na parte inferior ao meio verticalmente gerando em duas formas nas saídas na parte superior.

2) Camada de fios: Nessa camada apenas as construções que manipulam valores digitais podem ser posicionadas, os valores digitais são uma representação das formas e tintas e também de valores binários. Como o foco desse trabalho é a Turing completude do jogo só explicaremos as construções que são necessárias para chegar a essa verificação. Todas as construções dessa camada que são relevantes para alcançar esse objetivo estão representadas na figura 4 e figura 5. As construções da figura 4 possuem a finalidade de serem as entradas e as apresentações dos valores dos circuitos. As construções da figura 5 realizam as funções de portões lógicos e de componentes computacionais como os transistores.

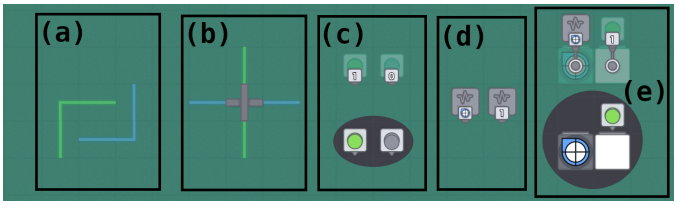

Figura 4. Fonte: De autoria própria
Construções que geram ou transportam valores digitais. **(a)** Fios que conectam construções transmitindo valores entre elas. Existem fios de duas cores diferentes que não se conectam quando estão adjacentes. **(b)** Cruzamento de fios que permite que dois fios passem por um mesmo quadrado sem se cruzarem. Podem ser usados com qualquer combinação de fios verdes e azuis. **(c)** Interruptor que gera um sinal constante que pode ser 1 ou 0 variando quando selecionado pelo jogador. Essa construção é posicionada nas duas camadas, podendo ser apertado pelo jogador na camada de construções mas gerando valores apenas na camada de fios. Os interruptores a esquerda, nas duas camadas, estão gerando o sinal 1 e os da direita estão gerando o sinal 0. **(d)** Gerador de sinal constante que difere do interruptor por gerar qualquer tipo de valor. A direita temos um gerando o valor da forma "blueprint" e o da esquerda está gerando o sinal 1. **(e)** Display que apresenta um valor. Assim como o interruptor está presente nas duas camadas, na camada de construções ele apresenta o valor que é recebido na camada de fios. Nesse exemplo vemos o da esquerda conectado a um sinal constante gerando a forma "blueprint" e o da direita conectado a um interruptor gerando o sinal 1.

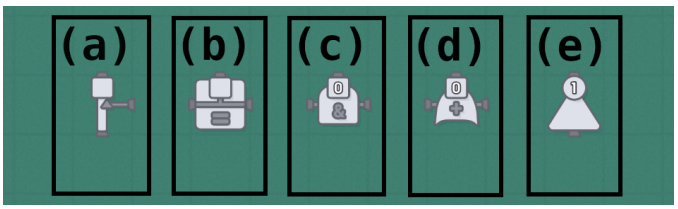

Figura 5. Fonte: De autoria própria
Construções com a função de componentes lógicos. **(a)** Transistor que repete o valor da entrada, que chega na parte inferior, na saída, na parte superior, caso um valor diferente de 0 seja recebido na entrada da sua lateral. **(b)** Componente de Igualdade que compara os valores das suas entradas, nas laterais, e envia para sua saída, na parte superior, 1 caso elas sejam iguais, 0 se forem diferentes ou se não estiver recebendo as duas entradas. **(c)** Portão E que verifica os valores das suas entradas, nas laterais, e envia para sua saída, na parte superior, 1 caso elas sejam 1, e 0 se alguma delas for diferente de 1. **(d)** Portão OU que compara os valores das suas entradas, nas laterais, e envia para sua saída, na parte superior, 1 caso alguma delas seja 1 e 0 se elas forem 0. **(e)** Portão Negar que nega a sua entrada, na parte inferior, e envia para sua saída, na parte superior. Ou seja, caso a entrada seja 1 a saída será 0, caso a entrada seja 0 a saída será 1.

C. Máquinas de Turing

A máquina de Turing foi definida por Alan Turing [4] e, em conjunto com o cálculo lambda definido por Alonzo Church [9], foi usada para estabelecer o conceito de algoritmo na tese de Church-Turing. Os algoritmos são a base da computação, entender os seus limites leva à compreensão do que é computável. Por isso, a MT tem extrema importância na área da computação, por representar o modelo fundamental que define o máximo poder computacional. Mesmo sendo tão poderosa, no livro de Michael Sipser [5] ela é comparada com um autômato finito, possuindo apenas o diferencial de ser composta por uma fita infinita e uma cabeça que é capaz de ler, escrever e se movimentar para a direita e para a esquerda.

Para melhor compreendermos a MT usaremos a definição formal descrita por Sipser [5]. Uma MT $\mathbf{M}$ pode ser definida formalmente como uma sétupla descrita pela igualdade $\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \mathbf{\Gamma}, \delta, \mathbf{q_0}, \mathbf{q_{aceita}}, \mathbf{q_{rejeita}})$. Cada elemento da sétupla é definido da seguinte forma:

- $\mathbf{Q}$: O conjunto finito de estados;
- $\mathbf{\Sigma}$: O conjunto finito de caracteres do alfabeto de entrada que não contém o símbolo vazio;
- $\mathbf{\Gamma}$: O conjunto finito com o alfabeto da fita onde o símbolo vazio e o $\mathbf{\Sigma}$ estão contidos;
- δ: A função de transição que dado um estado e um elemento de $\mathbf{\Gamma}$ retorna um outro estado, um outro elemento de $\mathbf{\Gamma}$ e um comando $\{$D, E$\}$;
- $\mathbf{q_0}$: O estado inicial;
- $\mathbf{q_{aceita}}$: Um conjunto com os estados de aceitação;
- $\mathbf{q_{rejeita}}$: Um conjunto com os estados de rejeição.

A computação de uma MT $\mathbf{M}$ se inicia com a cabeça estando inicialmente na posição mais à esquerda da fita, o estado inicial sendo $\mathbf{q_0}$ e recebendo a entrada $\mathbf{w} = \mathbf{w_1 w_2 \dots w_n}$ que pertence a $\mathbf{\Sigma}$ que está nas posições mais à esquerda da fita, o restante da fita a direita está preenchida infinitamente com o símbolo vazio [5]. Dado essas informações iniciais a computação se desenrola a partir da função de transição, lendo a fita na posição em que a cabeça está e considerando o estado atual será definido o que a cabeça irá escrever, para onde a cabeça irá se mover e qual o próximo estado de $\mathbf{M}$. A computação termina caso a máquina $\mathbf{M}$ assuma algum estado que está contido em $\mathbf{q_{aceita}}$ ou $\mathbf{q_{rejeita}}$. Caso essa condição nunca seja satisfeita, a máquina continuará a computação para sempre [5].

D. Linguagens

A configuração de uma MT representa a situação atual da sua computação e durante uma computação diversas configurações podem ser assumidas pela MT. Uma configuração de uma MT é composta pelo seu estado atual, a localização atual da sua cabeça e o conteúdo atual de sua fita [5]. A partir de uma configuração $\mathbf{C_1}$ é possível produzir uma configuração $\mathbf{C_2}$ se em um único passo a MT conseguir ir de $\mathbf{C_1}$ para $\mathbf{C_2}$. Uma configuração de aceitação é qualquer configuração que tenha como estado um estado de aceitação, da mesma forma uma configuração de rejeição possui um estado de rejeição. Podemos afirmar que as configurações de aceitação e de rejeição não produzem nenhuma outra configuração por possuir estados que são de parada da máquina [5]. Uma entrada $\mathbf{w}$ é aceita pela MT $\mathbf{M}$ se existir uma sequência de configurações $\mathbf{C_1}, \mathbf{C_2}, \dots, \mathbf{C_{Aceita}}$ onde a configuração $\mathbf{C_1}$ possua o estado inicial $\mathbf{q_0}$, a cabeça esteja na posição mais à esquerda da fita e a fita esteja preenchida com $\mathbf{w}$ nas posições mais à esquerda e por vazio nas posições restantes. Uma linguagem é composta pelo conjunto de entradas $\mathbf{w}$ que são aceitas por uma MT [5].

E. Máquinas de Turing Universais

As MT possuem uma limitação quanto às linguagens que elas aceitam, sendo necessárias diferentes MT para aceitar diferentes linguagens. Existe um tipo especial de MT que são

capazes de aceitar uma linguagem que seja aceita por qualquer outra MT, elas são chamadas de máquinas de Turing Universal. Elas possuem essa capacidade por conseguirem simular outras MT [5]. Para isso, essas máquinas recebem como entrada na fita a MT que elas devem simular e a entrada para essa máquina. A partir dessa descrição é possível comparar a MTU com os computadores atuais, onde o "Hardware" é semelhante a MTU por servir de base e oferecer as ferramentas para a execução do "Software" que é semelhante a MT que está sendo simulada, por implementar os algoritmos e resolver os problemas que recebe como entrada. Um exemplo de MTU que foi descrito por Sipser [5] é a máquina $\mathbf{U}$ que recebe como entrada uma MT $\mathbf{M}$ e uma sequência de carácteres $\mathbf{w}$ e simula a máquina $\mathbf{M}$ com a entrada $\mathbf{w}$. Caso a máquina $\mathbf{M}$ aceite $\mathbf{w}$, então $\mathbf{U}$ aceita $\mathbf{M}$ e $\mathbf{w}$, caso a máquina $\mathbf{M}$ rejeite $\mathbf{w}$ então $\mathbf{U}$ rejeita $\mathbf{M}$ e $\mathbf{w}$ e caso a máquina $\mathbf{M}$ não pare com $\mathbf{w}$ então $\mathbf{U}$ não irá parar com $\mathbf{M}$ e $\mathbf{w}$.

F. Turing Completude

A partir da definição de Astro Teller [10], podemos entender que as linguagens aceitas por um sistema Turing Completo contém todas as linguagens aceitas por qualquer MT. Portanto, para provar a completude do "Shapez.io" é necessário demonstrar que as linguagens aceitas pelo jogo são superconjunto das linguagens aceitas pelas MTs [10]. Uma das formas de realizar essa prova é demonstrar que o jogo suporta uma MTU. Isso é suficiente para a prova da Turing completude, pois por definição uma MTU é uma MT capaz de simular qualquer MT. Portanto, uma MTU é capaz de aceitar toda linguagem que é aceita por qualquer MT. Entretanto, para provar que o jogo suporta uma MTU não é necessário implementá-la, basta demonstrar que é possível faze-lo [1].

G. Método de Prova e Demonstração

A prova da Turing Completude de jogos digitais é um tema recorrente na literatura. Os jogos "Minecraft" [3], "Magic: The Gathering" [11], "Sid Meier's Civilization" [12], entre outros, são Turing Completos e essas provas foram feitas por meio da implementação de uma MTU. Simular uma MTU não é necessário para realizar a prova, basta comprovar a possibilidade de se implementar uma MTU. Por isso a presença da porta lógica NAND e de um mecanismo de "clock" no sistema que se deseja comprovar ser Turing Completo é suficiente, pois esses são os elementos básicos para a construção de uma MTU [1].

Através da possibilidade de implementar esses elementos se torna provado a Completude do jogo. Porém isso não é o suficiente para demonstrá-la. A demonstração só pode ser feita por uma implementação da MTU e por isso buscamos na comunidade do "Shapez.io", que já explorava a possibilidade da completude do jogo, jogadores que tenham feito e disponibilizado uma MTU [13]. Existe um jogador que alega ter montado uma MTU dentro do "Shapez.io" em um vídeo publicado no "YouTube" [2]. A máquina criada por esse jogador será destrinchada para exemplificar como uma MTU poderia ser montada a partir das portas lógicas que o jogo oferece.

III. Verificação da Turing Completude

Utilizando os portões lógicos E e Negar representados na figura 5, podemos construir o portão lógica NAND. Já um sistema de "clock" pode ser implementado usando uma sequência ímpar e maior que 1 de portões negar, esse sistema está representado na figura 11. Construir esses elementos usando as construções da camada de fios é uma tarefa simples, porém eles também podem ser construídos na camada de construções. É possível construir os portões lógicos E e OU Exclusivo e o sistema de "clock" através de esteiras, armazenamentos, balanceadores, empilhadores, cortadores e de túneis como podemos ver nas figuras 6, 7 e 9 respectivamente [14].

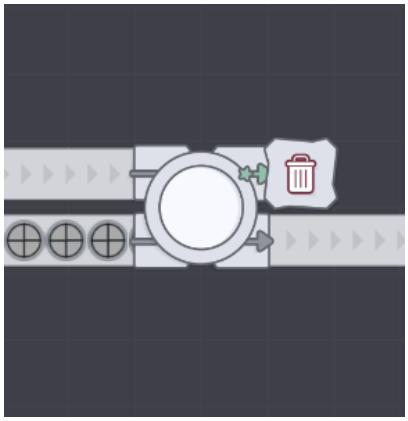

Figura 6. Fonte: De autoria própria
Portão E. Composto por um armazenamento e uma lixeira. Esse portão se aproveita da saída preferencial da construção de armazenamento, que está sempre eliminando a maior quantidade possível de elementos na lixeira, para simular o comportamento da porta lógica E.

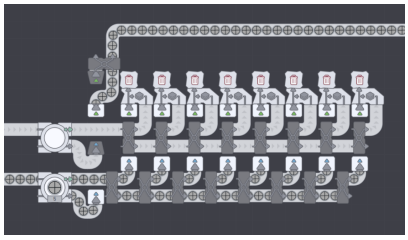

Figura 7. Fonte: De autoria própria
Portão OU Exclusivo. Essa fábrica se aproveita da peculiaridade do Empilhador, de exigir as duas entradas para gerar uma saída, para simular o portão lógico OU Exclusivo.

Para construir um NAND a partir dessas portas é preciso que o portão OU Exclusivo receba em uma entrada um sinal continuo de verdadeiro e na outra entrada a saída de um portão E. Dessa forma o portão OU Exclusivo age como um portão NEGAR o que torna essa uma construção do portão NAND. Uma demostração dessa implementação do NAND pode ser observada na figura 8.

O sistema de "clock" feito na camada de construções funciona de forma semelhante com o construído na camada

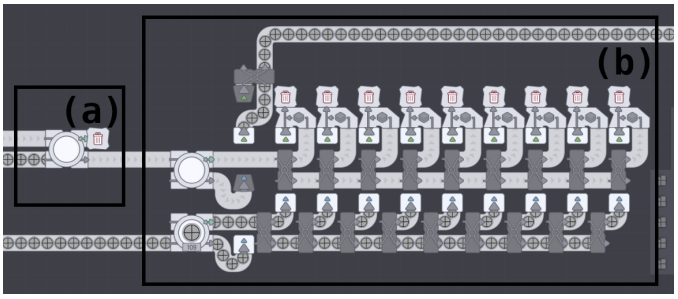

Figura 8. Fonte: De autoria própria
Portão NAND. Construída usando o Portão E **(a)** da figura 6 e o Portão OU Exclusivo **(b)** da figura 7. O portão Negar é implementado usando o Portão OU Exclusivo com uma das suas entradas recebendo um valor continuo.

de fios. O sistema retratado na figura 9 se aproveita do tempo de atraso que existe nas construções de empilhar e de cortar para gerar continuamente uma forma de saída mantendo uma forma circulando no circuito em períodos fixos de tempo.

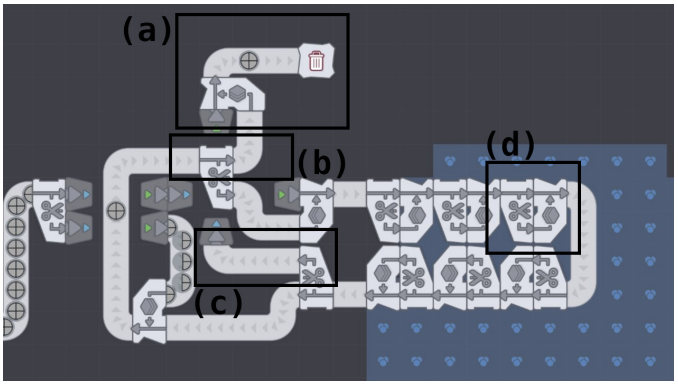

Figura 9. Fonte: De autoria própria
Sistema de ”clock” da camada de construções. A saída do sistema **(a)** gera uma forma a cada volta que uma forma da nesse circulo de esteiras. A saída é gerada a partir de metade da forma que é gerada em **(b)** no início do ciclo empilhada com a metade complementar gerada no final do ciclo em **(c)**. Esse sistema se aproveita das funções complementares das empilhadeiras e dos cortadores que quando unidas em sequência como em **(d)** geram na saída uma forma idêntica a recebida na entrada mas que possuem um tempo de espera para faze-lo.

Tendo essas implementações da porta lógica NAND e as implementações do sistema de ”clock” podemos afirmar que o jogo é Turing Completo. Ademais, ao possuir o portão NAND é possível ter memória, através da implementação de um Flip-Flop [15], que precisa de um sinal do sistema de ”clock”, e também qualquer outro portão lógico que seja necessário para a construção de uma MTU [1].

IV. Componentes computacionais para a Máquina de Turing Programável

Foi comprovado que o jogo é Turing Completo a partir da existência da porta NAND e do sistema de ”clock”. Porém, isso não demonstra como a construção da MTU pode ser feita no jogo. A Máquina de Turing Programável (MTP) implementada por Antoine Dragnir e exibida no vídeo “Proof that Shapez.io is Turing-complete” [2] pode ser usada como base para a construção de uma MTU. Ela pode ser chamada de MTP pois possui uma função de transição que pode ser programada para simular diferentes MT. Por isso, analisar e compreender o funcionamento da MTP será a forma como demonstraremos que uma MTU pode ser implementada no ”Shapez.io”. Durante o vídeo “Proof that Shapez.io is Turing-complete”, Antoine Dragnir [2] apresenta os dois modos de uso da MTP. O primeiro é o modo manual onde o jogador pode mover a posição da cabeça e alterar os valores armazenados na fita e o segundo é o modo Turing onde a máquina simula uma MT. Para conseguirmos entender por completo o funcionamento dessa máquina é preciso acessar o mapa que foi disponibilizado pelo Antoine Dragnir nos comentários dos seus vídeos [2]. Através desse mapa a máquina foi estudada e o seu funcionamento documentado.

Explicaremos o funcionamento da MTP em etapas, primeiro serão explicadas os componentes básicos, compostos pelas construções do jogo, que são combinados na composição da máquina. Então a MTP será divida em três partes, Função de Transição, Núcleo e Fita, que serão explicadas separadamente na próxima seção identificando os componentes básicos usados e as partes da MT que elas simulam.

Na figura 10 temos a MTP apresentada no vídeo “Proof that Shapez.io is Turing-complete” [2]. Para visualizar por completo a MTP é preciso aumentar a amplitude da visão do mapa ao máximo possível por causa da sua grande extensão. Porém quando isso é feito o jogo altera a visualização do mapa para uma versão mais simples, o que gera uma diferença no visual do jogo que pode ser percebido comparando a figura 22 com a 23.

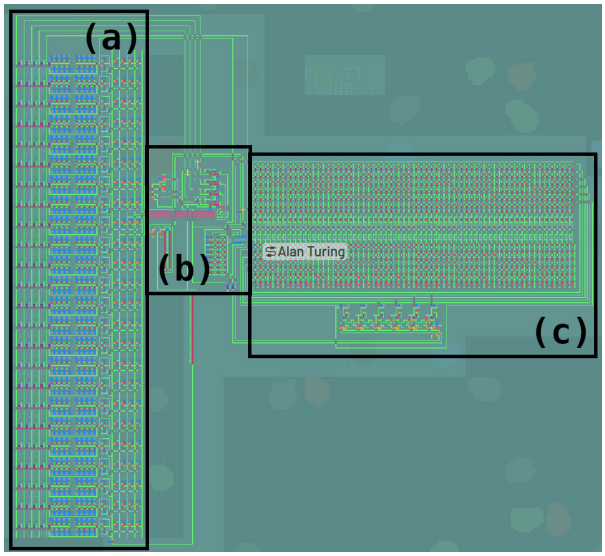

Figura 10. Fonte: De autoria própria
Máquina de Turing Programável criada por Antoine Dragnir. Ela é composta por: **(a)** Função de Transição Programável que comanda a movimentação da cabeça, a alteração do estado atual e a atualização da fita; **(b)** Núcleo que é composto pelo armazenamento do estado atual da fábrica, pelo armazenamento do modo de funcionamento atual, pelos controles para manipular a fita no modo manual e um sistema de ”clock” que marca a passagem de cada iteração; **(c)** Fita que armazena as valores da fita, a posição atual da cabeça e o sistema que altera a posição da cabeça.

A. Componentes Básicos

Alguns componentes são reutilizados em várias partes da MTP. Dentre eles temos componentes de armazenamento de dados, tanto binários como de elementos do jogo, de contagem de valores binários e de controle da passagem das iterações. Esses componentes serão explicados nessa subseção para possibilitar a compreensão do funcionamento da MTP.

1) Sistema de "clock": Nas figuras 11 e 12 podemos ver duas versões do sistema de "clock". Na versão que utiliza os transistores existe a possibilidade de desligar e ligar o sistema, através de um transistor qualquer que esteja no circuito. Esse componente, independente da versão, se aproveita do tempo de atraso que as construções precisam para a partir da entrada gerar uma saída, com isso é gerado um valor que circula infinitamente o componente. Quão maior for a quantidade de construções compondo o "clock" maior será o intervalo entre as trocas dos sinais.

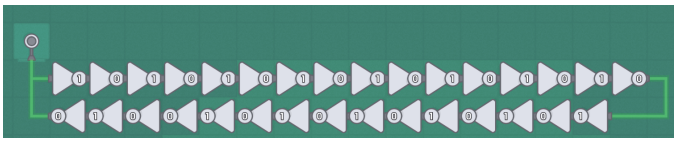

Figura 11. Fonte: De autoria própria
Componente de "clock" composto por portões negar. Se aproveita do tempo de atraso que o portão negar possui ao a partir da entrada gerar uma saída para manter um valor circulando infinitamente no componente.

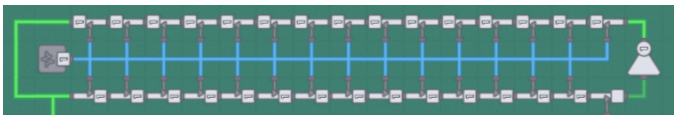

Figura 12. Fonte: De autoria própria
Componente de "clock" composto principalmente por transistores. Se aproveita do tempo de atraso que os transistores possuem para a partir da entrada gerar uma saída, assim gerando um valor que circula infinitamente no componente. Na parte inferior a direita da figura é possível visualizar um transistor que não está ligado ao fio azul no cento do sistema, esse é o transistor que possuí a função de desligar ou ligar o sistema.

2) Gerador de Pulso: Na figura 13 podemos ver um componente que se aproveita do tempo de atraso causado pelo portão Negar para gerar um pulso na sua saída. Essa funcionalidade é usada em diversos componentes que precisam apenas de um sinal ao invés de um valor constante.

3) Armazenamentos: Na figura 14 podemos ver os componentes de armazenamento que possuem dois tipos. O mais a esquerda é o armazenamento de cor e os outros três são versões de armazenamentos binários. No armazenamento de cores temos um exemplo com três cores. Esse é o armazenamento que compõem a fita, ou seja, a fita possui cores como alfabeto. Os outros três armazenamentos são equivalentes em funcionalidade mas estão construídos de formas diferentes. O funcionamento do armazenador binário é idêntico ao de um Flip-Flop [15], ou seja, podemos definir como 1 ou redefinir como 0 um valor binário armazenado nesse componente.

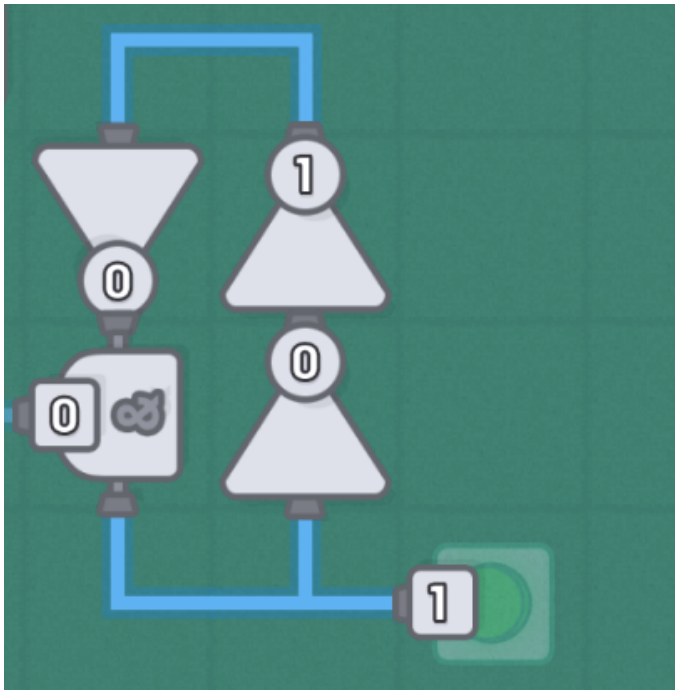

Figura 13. Fonte: De autoria própria
Gerador de pulso. Usa o tempo de atraso do portão Negar para gerar um pulso na sua saída ao invés de um valor continuo.

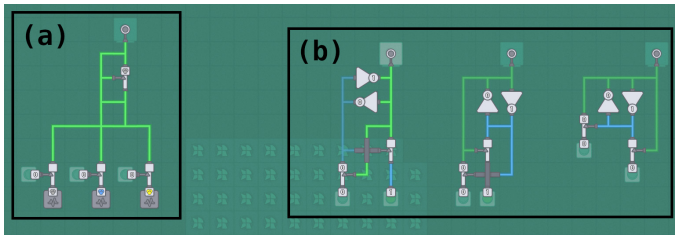

Figura 14. Fonte: De autoria própria
Armazenamentos de valores digitais e de binários. **(a)** Armazenamento de cor, nesse exemplo com três cores. É composto por apenas um transistor que por ter as suas entradas e a sua saída conectadas é capaz de armazenar um valor e o manter enquanto não for alterado. **(b)** Três versões do armazenamento binário, sendo a versão do meio a mais utilizada na fábrica. Esse armazenamento possui o funcionamento semelhante ao de um Flip-Flop [15], é composto por dois interruptores, um para definir o valor armazenado como 1 e o outro para redefinir o valor armazenado para 0.

4) Conversor de Binário: Na figura 15 podemos ver o componente que converte um número binário em uma posição de um vetor de Displays. Esse componente pode ser descrito como uma junção de literais binários [15], onde cada posição do vetor pode ser representado por um literal que equivale a apenas uma possibilidade do valor binário de entrada. Nessa implementação da figura 15 o número binário é representado por quatro interruptores onde cada interruptor representa um dígito do número da entrada.

5) Contador em binário: Na figura 16 o contador binário armazena um número binário e é capaz de incrementa-lo ou decrementa-lo. Nessa figura temos um composto por quatro cópias da versão mais simples do contador. Cada cópia repre-

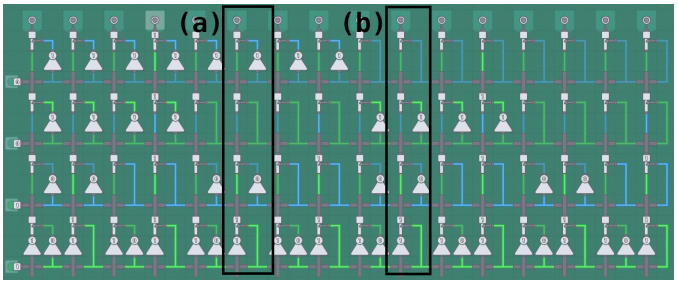

Figura 15. Fonte: De autoria própria
Conversor de binário que traduz um número binário para uma posição em um vetor de Displays. Podemos descrever cada posição do vetor como a representação de um número em binário. Nesse exemplo temos quatro interruptores no canto esquerdo da figura, cada um deles representa um digito do número binário. Considerando o interruptor mais próximo da parte superior da imagem como o mais significativo então podemos dizer que o Display da posição **(a)** só ficará branco quando o número de entrada for 0101 e da mesma forma o Display da posição **(b)** só ficará branco quando a entrada for 1001.

senta um digito de um valor binário, ao conectar essas cópias é possível representar números binários maiores, nesse caso com quatro dígitos.

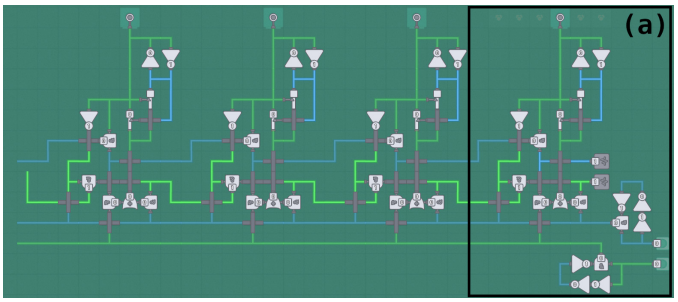

Figura 16. Fonte: De autoria própria
Contador binário de quatro dígitos. Esse exemplo é composto por quatro cópias da versão mais simples do contator **(a)**. Cada cópia representa um dígito do número binário, e todas elas estão conectadas com os comandos de incrementar e decrementar, sendo restringidas pelos sinais de entrada que recebem dos dígitos anteriores e restringem os dígitos a frente com as suas saídas.

Na figura 17 temos a versão mais simples do contador representando um valor binário de apenas um dígito. Aqui podemos ver as entradas e saídas do contador, além da entrada dos sinais de incrementar e decrementar também temos as entradas que indicam se o valor armazenado pode ser incrementado ou decrementado. Para permitir a expansão do número armazenado pelo contador, ele também possui saídas que indicam para o próximo digito se ele pode ser incrementado ou decrementado.

V. Funcionamento da Máquina de Turing Programável do Antoine Dragnir

Na figura 10 temos a máquina que será descrita em três partes, a função de transição, o núcleo e a fita. Ao descrever cada parte, será identificado os componentes básicos que foram utilizados e como elas simulam as suas versões da MT.

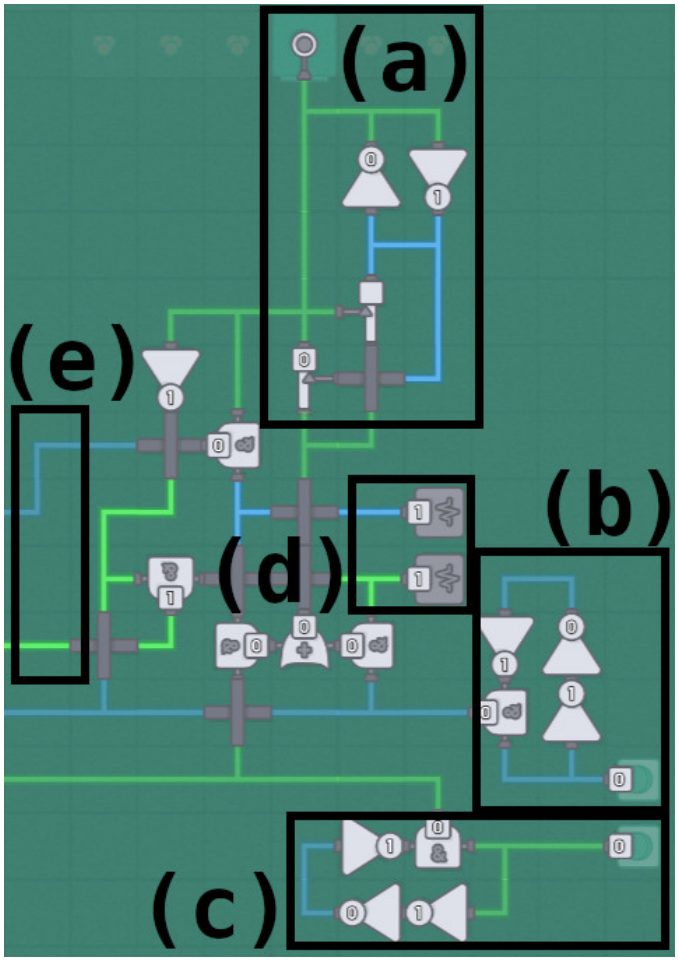

Figura 17. Fonte: De autoria própria
Contador binário de um digito. Podemos ver dois geradores de pulso, sendo o **(b)** para decrementar e o **(c)** para incrementar o número que está registrado no armazenador binário **(a)**. Quando um dos interruptores é acionado o resultado dependerá das entradas **(d)**, a entrada inferior indica que o armazenador pode ser decrementado e a superior que ele pode ser incrementado. Os valores da entrada e o digito armazenado geram os valores da saída **(e)**, assim como na entrada o inferior indica que o próximo digito pode ser decrementado e o superior que ele pode ser incrementado.

A. Função de Transição

A função de transição de uma MT pode ser representada como um grafo onde os nós são os estados e as arestas são as transições entre os estados condicionadas a partir dos valores lidos da fita. Tendo essa representação da função de transição podemos descrever essa parte da máquina como um conjunto de arestas programáveis desse grafo. A figura 18 registra essa parte da máquina e nela é possível identificar as "arestas" que são replicadas para compô-la. A forma como as arestas podem ser replicadas permite a expansão da função de transição, o que significa que não existe limite para a quantidade de arestas considerando a infinidade do mapa. Cada uma dessas arestas tem a função de armazenar e realizar a transição de um estado da máquina para outro alterando também a posição da cabeça e os valores da fita no processo.

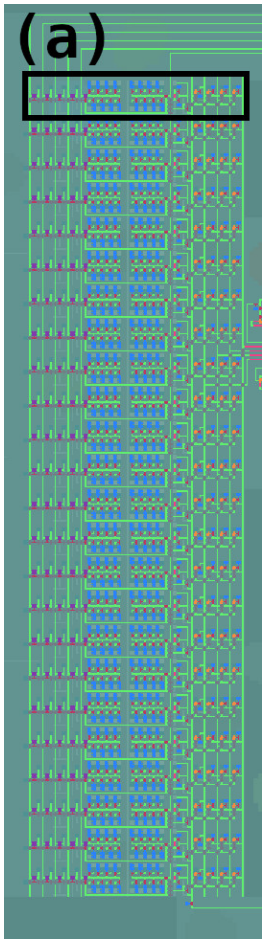

Figura 18. Fonte: De autoria própria
Função de transição da MTP. Ela é composta por um conjunto de arestas do grafo de transição **(a)** que podem ser programadas para simular uma MT.

1) Aresta do grafo de transição: Esse é o componente que armazena e executa a alteração de estado, a alteração de posição da cabeça da fita e a escrita na fita. Com um conjunto de cópias desse componente temos a simulação da função de transição de uma MT. Na figura 19 podemos ver uma aresta desse grafo mais de perto e identificar a escrita indicando o lado esquerdo como a entrada do componente e o lado direito como a saída. Tanto o valor esperado pela aresta como o valor armazenado nela para se tornar a saída podem ser alterados, essa possibilidade permite que qualquer função de transição possa ser simulada pela MTP.

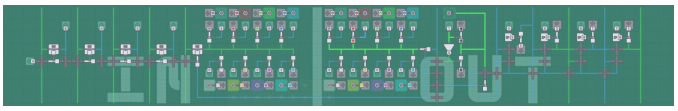

Figura 19. Fonte: De autoria própria
Aresta do grafo de transição da MTP. Na figura podemos perceber a entrada da aresta a esquerda que irá gerar a saída a direita caso os valores esperados sejam recebidos.

Na figura 20 vemos com mais detalhes a entrada desse componente. Nela é feita a comparação do estado atual da máquina com o estado esperado pelo aresta e do valor lido da fita com o valor esperado da aresta, apenas se esses valores forem idênticos que a entrada enviará um sinal para a saída do componente.

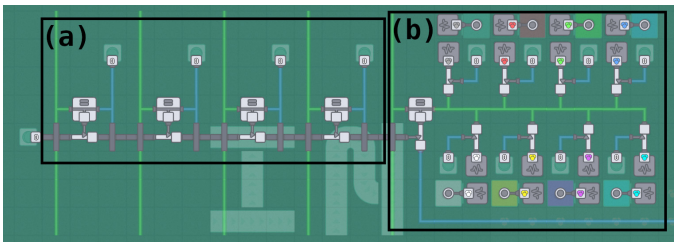

Figura 20. Fonte: De autoria própria
Entrada da aresta da função de transição da MTP. Esse componente irá comparar o estado atual da máquina recebido como entrada com os valores gerados pelos quatro interruptores, que representam o estado esperado pelo componente, usando os componentes de igualdade em **(a)**. Além da comparação dos estados temos também a comparação do conteúdo lido na fita com o esperado pelo componente em **(b)**. Os valores esperados podem ser alterados. Caso os valores das entradas sejam iguais aos valores esperados então a entrada envia um sinal para a saída da aresta.

Na figura 21 a saída do componente pode ser vista com mais detalhes. Ela armazena o valor que será armazenado na fita, a direção que a cabeça da fita irá se movimentar e o novo estado da máquina. Esses valores armazenados se tornam a saída da aresta caso a entrada envie o sinal confirmando que recebeu como entrada os valores esperados.

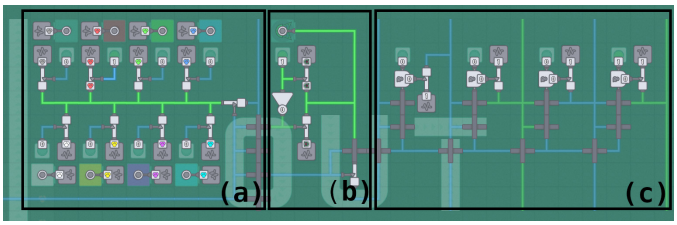

Figura 21. Fonte: De autoria própria
Saída da aresta da função de transição da MTP. Os valores armazenados, que podem ser alterados, se tornarão uma saída caso a entrada seja igual a esperada pela aresta. O conjunto de armazenamento de cores guarda a cor da saída que será escrita em **(a)**. À direita em **(b)** temos dois geradores de sinais constantes das formas direita e esquerda que indicam para onde a cabeça da máquina irá se mover. Os quatro próximos geradores de sinais constantes indicam o próximo estado da máquina em **(c)**.

B. Núcleo

Na figura 22 vemos um conjunto de componentes que exercem funções variadas na máquina. Dentre as funções temos um controle para o modo manual da máquina que permite alterar os valores armazenados na fita, um armazenamento binário de quatro dígitos que armazena o estado atual da máquina, um armazenamento binário que inicializa e indica se a máquina está no modo Turing e um sistema de ”clock”.

1) Sistema do modo Turing: A figura 23 apresenta em mais detalhes o armazenamento binário que indica se a máquina está no modo Turing. Ele recebe um sinal de um botão para iniciar o modo Turing. Quando esse modo é ativado o estado atual da

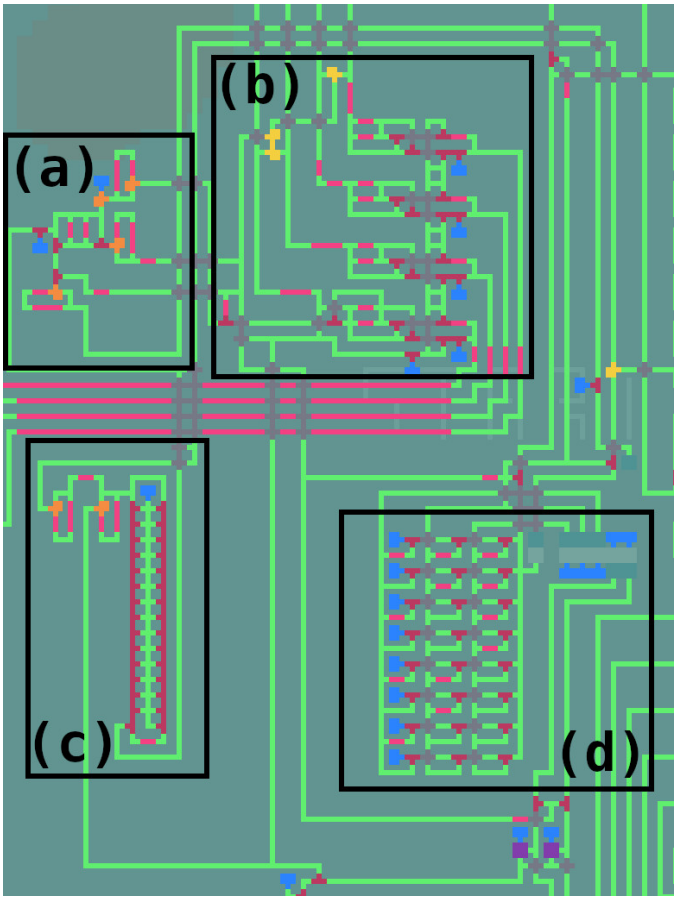

Figura 22. Fonte: De autoria própria
Núcleo da MTP. Essa parte da máquina possui diversas funções, dentre elas inicializar e indicar que o modo Turing está ativo **(a)**, armazenar o estado atual **(b)**, gerar o ”clock” da máquina **(c)**, controlar a posição da cabeça e escrever na fita no modo manual **(d)**.

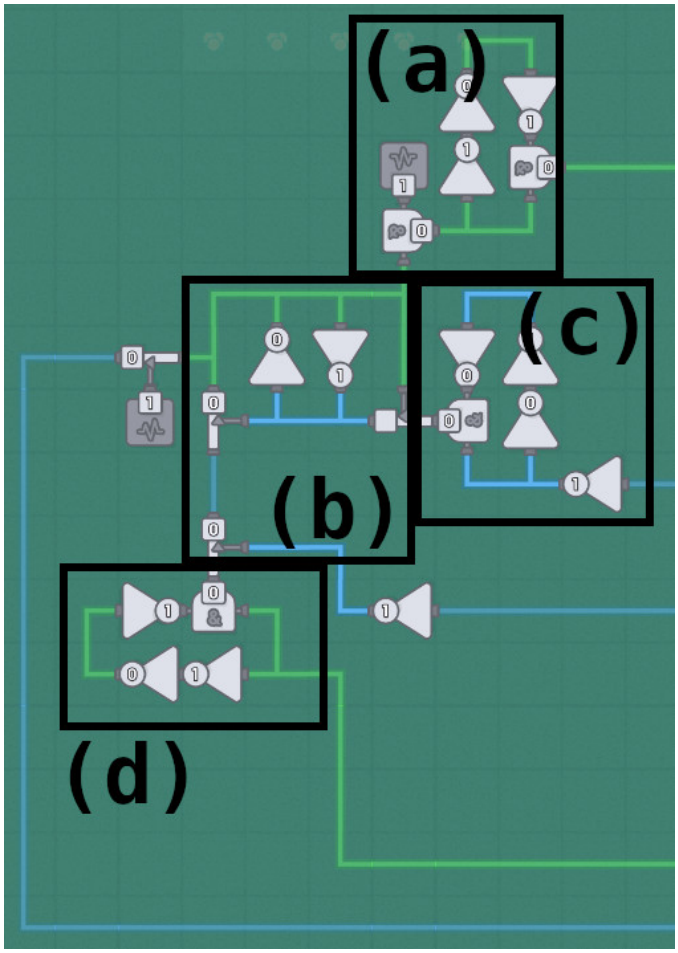

Figura 23. Fonte: De autoria própria
Sistema do modo Turing. Nessa figura temos cinco fios que ultrapassam a borda direita da figura, o fio verde na parte superior que sai do gerador de pulso **(a)** e o fio azul mais próximo a borda inferior que sai do armazenador binário **(b)** são as saídas desse componente, os fios restantes são entradas. O fio azul de saída de **(b)** se conecta ao sistema de clock o mantendo ativo enquanto a máquina estiver ligada. O fio verde que saí do gerador de pulso **(a)** se liga ao armazenamento do estado atual da máquina para definir o estado inicial da máquina. A entrada pelo fio verde que chega no gerador de pulso **(d)** está conectada a um interruptor que inicia o modo Turing. Essa entrada defini o armazenador binário **(b)** como 1. As outras duas entradas feitas por fios azuis, uma chegando ao transistor que sai de **(d)** e entra em **(b)** e outra chegando no gerador de pulso **(c)**, estão conectadas ao armazenamento do estado atual da máquina. Caso o estado da máquina seja o estado de parada, ou seja o equivalente em binário a 0, essas duas entradas serão do valor 1 e o armazenamento binário **(b)** irá receber o comando de redefinir o valor armazenado para 0, assim desativando o modo Turing.

máquina e atualizado para o estado inicial 0001 e esse modo só será desativado quando o estado atual da máquina se torna o estado de parada 0000. Enquanto o modo Turing estiver ativo o controle da fita do modo manual será desativado e só será reativado quando o modo Turing for desativado.

2) Sistema de ”Clock”: Na figura 24 temos o sistema de ”Clock”. Ele é composto por dois geradores de pulso e um componente de ”clock” na versão com transistores, essa é a versão utilizada por permitir ligar e desligar o ”clock”. O gerador de pulso mais próximo ao sistema de ”clock” se conecta à fita indicando que a saída da função de transição deve ser armazenada na fita. O outro gerador de pulso se conecta ao armazenamento do estado atual da máquina e é utilizado para atualizar o estado da máquina para o estado de saída da função de transição.

3) Armazenamento do estado atual: Na figura 25 podemos ver o sistema que armazena o estado atual da máquina. Ele é representado por um número binário de quatro dígitos. O estado atual é alterado pela função de transição a cada iteração. Quando o estado atual se torna 0000 um sinal é enviado para o sistema do modo Turing desligando o modo Turing.

C. Fita

A figura 26 registra a parte da máquina que simula a fita e a cabeça da MT. Essa Fita é composta pelo armazenamento da posição atual da cabeça e os valores armazenados na fita.

1) Sistema de movimentação da cabeça: Na figura 27 podemos ver o componente que envia o comando de incrementar ou decrementar a posição da cabeça na fita. Esse componente recebe comandos tanto da função de transição quanto do controle manual da fita. Porém o controle manual fica desativado enquanto a máquina está no modo Turing, isso

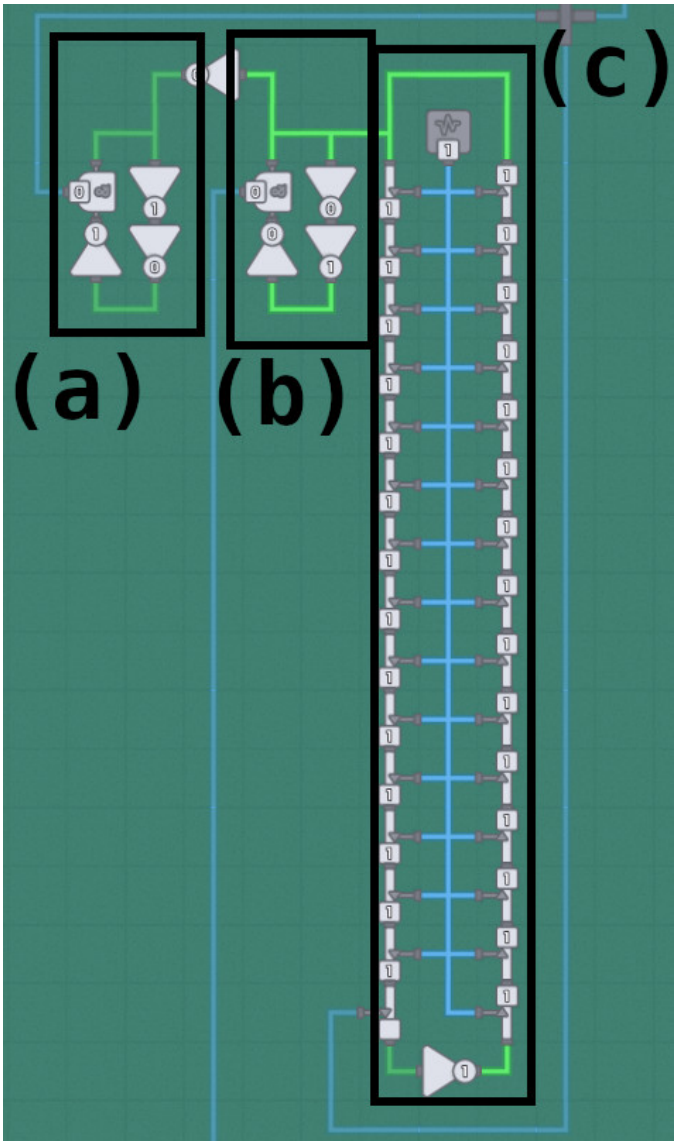

Figura 24. Fonte: De autoria própria
Sistema de ”clock” da MTP. Ele é composto por dois geradores de pulso **(a)** e **(b)**, além de um sistema de ”clock” na versão com transistores **(c)**, essa é a versão utilizada por permitir ligar e desligar. O gerador de pulso **(b)** se conecta à fita indicando que a saída da função de transição deve ser armazenada na fita. O gerador de pulso **(a)** se conecta ao armazenamento do estado atual da máquina e é utilizado para atualizar o estado da máquina para o estado de saída da função de transição.

garante que o usuário não atrapalhe a simulação da MT.

2) Armazenamento da posição atual da cabeça: Na figura 28 temos o armazenamento da posição atual da cabeça que é composta por um contador binário de seis dígitos. O armazenamento está ligada ao sistema de movimentação da cabeça da máquina que comanda a direção que a cabeça deve se mover. Esse armazenamento também está conectado ao armazenamento dos valores da fita indicando qual posição está sendo lida e será escrita.

3) Armazenamento da fita: Na figura 29 podemos ver parte do vetor de armazenamentos da fita. Ele é composto por

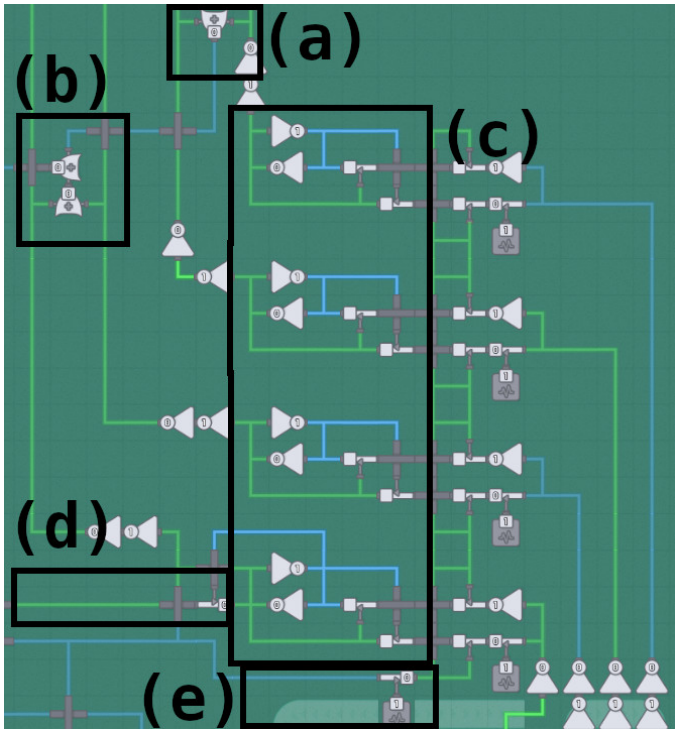

Figura 25. Fonte: De autoria própria
Armazenamento do estado atual da MTP. É composto por quatro armazenamentos binários **(c)** que juntos representam um número binário de quatro dígitos, esse número é o estado atual da máquina. A saída de **(c)** está conectada com a entrada da função de transição, e a entrada de **(c)** está conectado com a saída da função de transição após uma sequência de componentes lógicos de negação que atrasam a chegada dessa entrada. Existe também a saída para o sistema de inicialização do modo Turing **(b)** e **(a)** que é ativa quando o estado atual se torna o estado de parada, ou seja o equivalente em binário a 0. Além da entrada **(d)** que define o estado atual para o equivalente em binário do decimal 1 e da entrada que vem do sistema de ”clock” indicando quando o estado atual da máquina deve ser atualizado **(e)**.

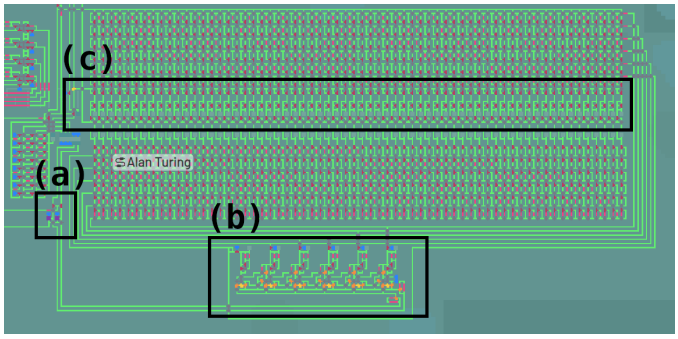

Figura 26. Fonte: De autoria própria
Fita da MTP. É composta por um sistema que manipula a posição da cabeça **(a)**, um contador binário **(b)** que armazena a posição da cabeça, dois conversores de binário um acima e outro abaixo do **(c)** que traduzem a posição da cabeça para acessar o vetor de armazenamento de cores **(c)**.

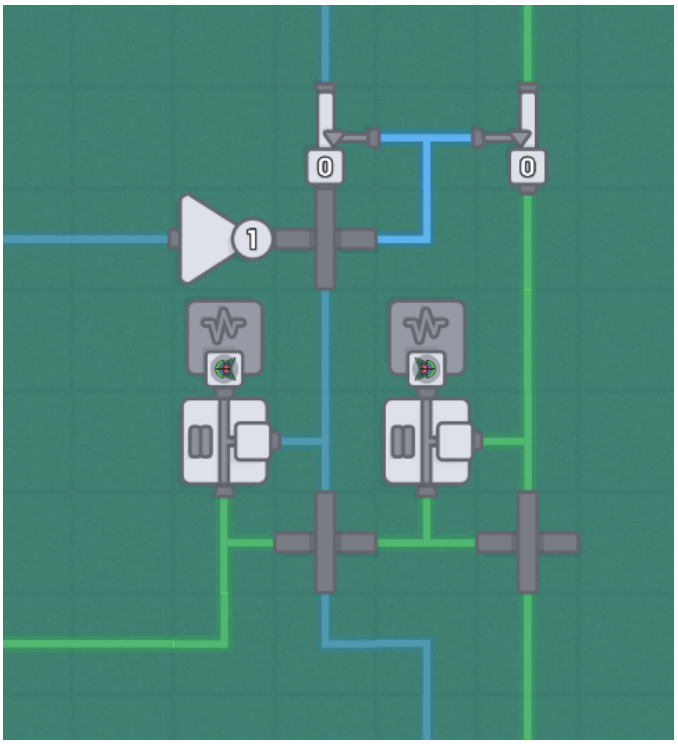

Figura 27. Fonte: De autoria própria
Sistema de movimentação da cabeça da MTP. Esse sistema possui quatro fios de entrada e dois de saída, os dois fios que estão na parte superior da figura vem do controle manual da máquina. O fio superior da parte esquerda da figura vem do sistema de armazenamento do estado atual da máquina, esse fio é o mesmo que é usado no sistema do modo Turing para desativá-lo, ele trás o valor 0 enquanto a máquina estiver no modo Turing. Com a negação desse valor a possibilidade de alterar a posição da cabeça pelo controle manual enquanto a máquina estiver no modo Turing é desabilitada. O fio da parte inferior à esquerda da figura é uma entrada desse sistema que está ligado à saída da função de transição. A partir dessa entrada a função de transição define qual será a próxima posição da cabeça. Os dois fios na parte inferior da figura estão conectados ao armazenamento da posição da cabeça sendo o fio a esquerda usado para decrementar a posição atual e o fio da direita para incrementar a posição atual do fio.

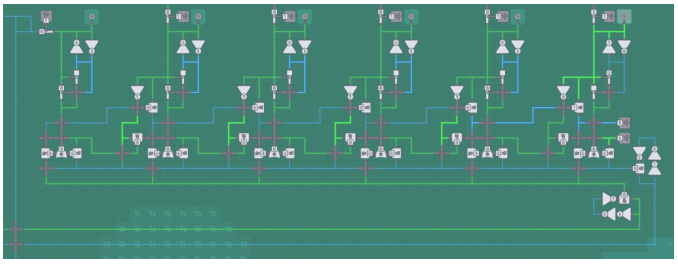

Figura 28. Fonte: De autoria própria
Armazenamento da posição atual da cabeça da MTP. É composta por um contador binário de seis dígitos. A entrada do armazenamento está ligada ao sistema de movimentação da cabeça da máquina, sendo dois fios de entrada, um para incrementar a posição atual e o outro para decrementar a posição atual. Cada um dos dígitos deste contador está conectado a um transistor, sendo essa a saída do contador que vai ser utilizada para indicar a posição da cabeça tanto visualmente por meio de um vetor de Displays quanto no sistema da fita, sendo usado para ler e para escrever na fita.

diversos armazenamentos de cores, sendo que cada posição da fita possui um armazenamento de cor. Por estar conectado ao sistema de armazenamento da posição da cabeça, apenas uma posição do vetor de armazenamentos pode ser lido ou alterado por vez.

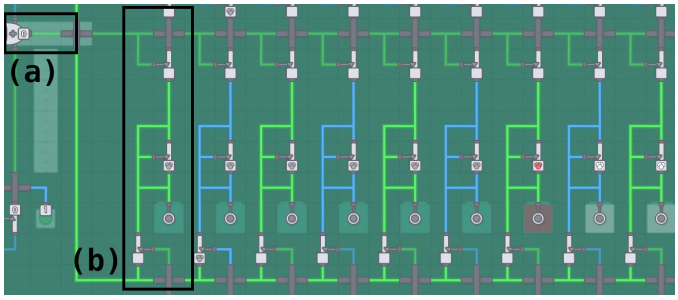

Figura 29. Fonte: De autoria própria
Vetor de armazenamentos de cores da Fita da MTP. É composto por diversos armazenamentos de cores, sendo que cada posição da fita possui um armazenamento de cor **(b)**. A entrada do armazenamento da fita está ligada à saída do conversor da posição atual da cabeça para o acesso a fita, a cor da saída da função de transição e ao ”clock” por **(a)** que manda o sinal para alterar a cor armazenada na posição atual da cabeça. A saída da fita está conectada ao conversor da posição atual da cabeça para um indicador da cabeça e a entrada da função de transição. A cor de saída da fita é determinada pela cor armazenada na posição atual da cabeça na fita.

VI. Conclusão

Comprovamos a Turing completude do jogo por meio da possibilidade de se criar uma porta lógica NAND, com as portas logicas do jogo e também com as construções que podem ser desbloqueadas diversos níveis antes das portas lógicas, e de se criar um sistema de ”clock”, que se aproveita do tempo de atraso que alguns componentes do jogo precisam para a partir de uma entrada gerar uma saída para criar um sinal que circula infinitamente.

Além disso, a Máquina de Turing Programável desenvolvida por Antoine Dragnir [2] foi analisada e é perceptível todas as semelhanças que ela possui com uma MT comum. Porém a possibilidade de programar a função de transição permite que qualquer MT seja simulada na MTP contanto que a função de transição seja configurada para isso. Por isso, a partir da MTP é possível desenvolver uma MTU o que reconfirma a Turing Completude do jogo.

A partir desse artigo temos uma demonstração de como jogos de fabricação focados em otimização e automação de processos podem possuir um poder computacional equivalente a Máquina de Turing. Sendo esse trabalho um exemplo de como provar essa equivalência. Uma questão que surge com o trabalho é a possibilidade de generalização desse método de prova para outros jogos de fabricação.

Referências

[1] M. Assis, “An origami universal turing machine design,” *arXiv preprint arXiv:2406.08490*, 2024.
[2] A. Dragnir. (2024) Proof that shapez.io is turing-complete. Youtube. [Online]. Available: https://www.youtube.com/watch?v=BSJvfum8Y4Y&ab_channel=AntoineDragnir

[3] J. A. Montoya Rodríguez and O. Rodríguez Aponte, "Steve turing: Minecraft es turing completo," 2023. [Online]. Available: http://hdl.handle.net/11349/39253
[4] A. Tueixg, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, pp. 345–363, 1936.
[5] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012. [Online]. Available: https://books.google.com.br/books?id=Akk8zQEACAAJ
[6] tobspr Games. (2024) shapez.io. GitHub. [Online]. Available: https://github.com/tobspr-games/shapez.io
[7] ——. (2020) shapez. Steam. [Online]. Available: https://store.steampowered.com/app/1318690/shapez/
[8] S. Wiki. (2023) Blueprints. [Online]. Available: https://shapezio.fandom.com/wiki/Blueprints
[9] A. Church, "An unsolvable problem of elementary number theory," *American journal of mathematics*, vol. 58, no. 2, pp. 345–363, 1936.
[10] A. Teller, "Turing completeness in the language of genetic programming with indexed memory," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, 1994, pp. 136–141 vol.1.
[11] A. Churchill, S. Biderman, and A. Herrick, "Magic: The gathering is turing complete," in *10th International Conference on Fun with Algorithms (FUN 2021)(2020)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 9–1.
[12] A. de Wynter, "Turing completeness and sid meier's civilization," *IEEE Transactions on Games*, vol. 15, no. 2, pp. 292–299, 2022.
[13] [deleted]. (2023) Is shapez.io turing complete? Reddit. [Online]. Available: https://www.reddit.com/r/shapezio/comments/p6xbnn/is_shapezio_turing_complete/
[14] Johandaonis. (2023) Logic gates. Reddit. [Online]. Available: https://www.reddit.com/r/shapezio/comments/hnjo15/logic_gates/
[15] R. J. Tocci, N. S. Widmer, and G. L. Moss, *Sistemas digitais*. Pearson Educación, 2010.