Modelagem e Análise de Vídeo Games baseadas em WorkFlow nets e Grafos de Estado

Franciny Medeiros Barreto

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2015

Franciny Medeiros Barreto

Modelagem e Análise de Vídeo Games baseadas em WorkFlow nets e Grafos de Estado

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software

Orientador: Dr. Stéphane Julia

Uberlândia
2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

B273m 2015

Barreto, Franciny Medeiros, 1992-
Modelagem e análise de vídeo games baseadas em WorkFlow nets e Grafos de Estado / Franciny Medeiros Barreto. - 2015.
98 f. : il.

Orientador: Stéphane Julia.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Redes de Petri - Teses. 3. Vídeo games - Teses. 4. Workflow - Teses. I. Julia, Stéphane. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Dedico este trabalho à minha mãe que muitas vezes desistiu de seus sonhos para que eu pudesse realizar os meus.

Agradecimentos

Agradeço a minha família, que é a minha fortaleza. Agradeço a minha mãe, Fátima, e aos meus irmãos, Amaro, Victor e Hugo, por todo amor e carinho que sempre me deram e pelo apoio incondicional. Agradeço as minhas cunhadas, Raísa e Isabel, por toda ajuda e apoio em todos os momentos.

A todos os meus amigos que fizeram o meu caminho muito mais feliz. Em especial agradeço a Emília, Bruno, Walter, Myllene, Maria Cristina, Joslaine, Reslley e Joyce pela amizade, pelo carinho, pela paciência e pela compreensão que sempre tiveram comigo. Agradeço a Natália, Hanailly, Selma, Priscilla e Hiago pelo companheirismo de sempre.

A todos os professores que contribuíram para a minha formação, que me apoiaram e me estimularam a chegar até aqui. Ao funcionário da secretaria do programa de pós-graduação em Ciência da Computação da UFU, Erisvaldo, pelo profissionalismo e por sempre ter me ajudado quando precisei.

Agradeço ao professor Stéphane Julia pelo profissionalismo, pela paciência e compreensão, pelos ensinamentos, pela confiança e orientação dados a mim ao longo desta pesquisa.

Agradeço principalmente a Deus, por tudo que tem feito e faz por mim, por ter sido o meu refúgio nos momentos mais difíceis e por me permitir desfrutar de Tua alegria. Que toda honra e toda glória seja dada a Ti, Senhor!

"Amei e odiei as palavras, espero tê-las usado bem."
(A menina que roubava livros - Markus Zusak)

Resumo

Este trabalho apresenta um método para a modelagem e análise qualitativa de cenários de vídeos games baseado em WorkFlow nets e grafos de estado para ajudar no processo de criação de vídeo games.

Em geral, um jogo possui um conjunto de atividades que devem ser executadas para alcançar um determinado objetivo do jogo, assim como um mundo virtual onde essas atividades são executadas. Para modelar tais atividades são utilizadas as WorkFlow-nets (redes de Petri que modelam processos de negócio). O mundo virtual é descrito através de um mapa topológico composto por várias áreas do jogo conectadas. As áreas do mundo virtual onde o jogador se encontrará durante o jogo são formalmente representadas por um tipo de rede de Petri denominado grafo de estado. Um mecanismo de comunicação, também baseado nas redes de Petri, é apresentado e tem como objetivo mostrar as interações que existem entre o modelo de atividades e o modelo do mapa topológico.

A análise qualitativa é realizada considerando o modelo global (modelo de atividades + modelo do mapa topológico). O método proposto para análise é derivado do algoritmo de verificação da propriedade *soundness*. Este método permite em particular a verificação da consistência de um cenário de vídeo games em termos de jogabilidade. A ferramenta CPN Tools é utilizada para a representação gráfica dos modelos e para a realização da análise.

A vantagem da abordagem proposta por este trabalho é a formalização do fluxo de atividades existente em um vídeo game e do mundo virtual por meio de um mesmo formalismo. A criação de um mecanismo de comunicação é importante para que os dois modelos trabalhem juntos simulando o comportamento do jogo. Assim, é possível que seja realizada uma análise global do jogo a fim de verificar a consistência tanto das atividades quanto do mundo virtual. O fato de trabalhar com o CPN Tools permite que seja realizada a análise do espaço de estado automaticamente por meio das funcionalidades de análise da ferramenta. Com o CPN Tools também é possível representar graficamente os modelos de modo simples e claro usando recursos da modelagem hierárquica.

Palavras-chave: redes de Petri, vídeo games, WorkFlow nets, grafos de estado, redes de Petri Coloridas, CPN Tools.

Abstract

This work presents a method for the modeling and qualitative analysis of video games scenarios based on WorkFlow nets and state graphs for aiding in the design process of video games.

In general, a game has a set of activities that must be performed to reach a certain goal of the game, as well as a virtual world where these activities are performed. To model these activities, WorkFlow-nets (Petri nets representing workflow processes) are used. The virtual world is described through a topological map composed of several connected areas. The areas of the virtual world the player will encounter during his evolution in the game are formally represented by a kind of Petri net called state graph. A communication mechanism, also based on Petri nets, is presented and aims to show the interactions that exist between the activity model and the topological map.

The qualitative analysis will be performed at a global model level (activity model + topological map). The method proposed for the analysis is based on an algorithm used to verify the soundness property of processes. This method will allow in particular the verification of the consistency of video game scenarios in gameplay terms. The CPN Tools is used for graphical representation of the models and to perform qualitative analysis.

The advantage of the proposed approach in this work is the formalization of the existing activity flow in a video game and of the virtual world through a same formalism. The creation of a communication mechanism is important in order for both models to work together and be able to simulate the game's behavior. Thus, a global analysis of the game can be performed to check the consistency of the activities and of the virtual world. The fact of working with the CPN Tools allows to perform automatically the state space analysis through the analysis functionalities of the tool. With the CPN Tools is also possible to represent graphically the models in a simply and clearly way using hierarchical modeling resources.

Keywords: Petri nets, video games, WorkFlow nets, state graphs, Colored Petri nets,

CPN Tools.

Lista de ilustrações

Sumário

Capítulo 1

Introdução

Um vídeo game é uma síntese de código, imagens, música e animação que se reúnem em forma de entretenimento. A criação de vídeo games difere da criação de *software* clássico por causa da fase de pré-produção e devido ao uso extensivo e integração de recursos multimídias. Além disso, a criação de vídeo games envolve algumas atividades que não necessariamente existem quando se considera o processo de desenvolvimento de *software* tradicional. Algumas dessas atividades são denominadas de *game design* e *level design*.

De acordo com (NATKIN; VEGA; GRÜNVOGEL, 2004), o *game design* é uma tarefa difícil que combina processos técnicos e artísticos. A fase de *game design* define os principais aspectos relacionados ao universo do jogo como, por exemplo, época e estilo, objetivo a ser alcançado, entre outros. Já na fase de *level desing*, são definidas as principais ações e os objetos do jogo (GAL et al., 2002).

Um jogo precisa ser interativo, entreter e dar liberdade controlada para o jogador (BATES, 2001). (ROLLINGS; MORRIS, 2003) afirmam que é preciso criar uma, ou mais, séries de desafios casualmente ligados em um ambiente simulado. É importante propor desafios que não sejam nem fáceis e nem difíceis demais de resolver. Além disso, é de fundamental importância garantir que a experiência do jogo leve a uma sucessão de metas dentro de um prazo razoável. Cumprir todas essas exigências não é uma tarefa meramente trivial, como exemplifica (NATKIN; VEGA; GRÜNVOGEL, 2004) com o problema da chave: dois quartos A e B são conectados por uma porta trancada. O avatar do jogador está localizado no quarto A e sua tarefa é entrar no quarto B, e o quarto B está trancado. Para realizar a tarefa, o jogador deve então destrancar a porta com a ajuda de uma chave. No entanto, a chave que destranca a porta está localizada no quarto B. Assim, o jogador nunca poderá abrir a porta e entrar no quarto B. Este é um exemplo trivial. Entretanto, em um jogo real, diferentes tarefas podem mudar a topologia do espaço virtual de uma forma bastante complexa (NATKIN; VEGA; GRÜNVOGEL, 2004). Consequentemente, prevenir erros similares a este se torna um grande problema.

Ao longo dos anos, os vídeo games evoluíram e se tornaram cada vez mais complexos.

De acordo com (HORROCKS, 1999), a linguagem natural não é uma ferramenta fácil para produzir grandes especificações. Por essa razão, novos conceitos e ferramentas precisam ser encontrados para suprir os desafios do desenvolvimento de jogos (HORROCKS, 1999). Em *softwares* e sistemas de engenharia é comum construir diagramas e gráficos para especificar visualmente conjuntos de requisitos.

Alguns trabalhos têm apresentado o uso de diagramas UML, para mostrar como os diferentes objetos em um jogo irão interagir de acordo com algumas ações que serão executadas pelo jogador ((ANG; RAO, 2004) e (RUCKER, 2003), por exemplo). De acordo com (OLIVEIRA; JULIA; PASSOS, 2011), diagramas UML são interessantes a medida que produzem uma estrutura de execução do jogo. No entanto, eles não apresentam em uma maneira explícita os possíveis cenários que existem em uma missão ou nível de jogo.

Métodos formais têm sido utilizados na modelagem de jogos como apresentado em (ARAÚJO; ROQUE, 2009), (THOMAS; YESSAD; LABAT, 2011), (OLIVEIRA; JULIA; PASSOS, 2011), (NATKIN; VEGA; GRÜNVOGEL, 2004), entre outros. Em (NATKIN; VEGA; GRÜNVOGEL, 2004), por exemplo, um novo tipo de rede de Petri denominado rede de transações é apresentado. Uma rede de transações permite modelar transações lógicas e temporais enquanto um mapa topológico do jogo é modelado por um tipo de grafo, chamado hiper-grafo. Neste tipo de grafo, as arestas são criadas dinamicamente. Mecanismos de comunicação entre os dois modelos tentam estabelecer a influência que um modelo tem sobre o outro. Porém, a análise formal é aplicada apenas na rede de transações (representada por uma rede de Petri). Assim, a validação de cenários de jogos não pode ser formalmente realizada usando ferramentas de análise de redes de Petri, devido à semântica operacional das redes de transação e dos hiper-grafos não serem as mesmas.

Já em (OLIVEIRA; JULIA; PASSOS, 2011), uma nova abordagem baseada em Work-Flow nets (AALST; HEE, 2004) foi proposta para especificar os cenários existentes em um nível de jogo. Nessa abordagem, o cálculo do sequente da lógica linear foi usado para provar a corretude de um tipo de critério chamado *soundness*, usado para validar os possíveis cenários que um jogador pode executar em uma missão de um jogo. No entanto, o trabalho de (OLIVEIRA; JULIA; PASSOS, 2011) considera apenas os modelos baseados nas atividades do jogador, e ignora por completo a visão de mapa topológico do jogo.

Neste trabalho, as redes de Petri serão exploradas para modelar as atividades existentes em um nível de jogo, bem como o mapa topológico correspondente do mundo virtual do jogo. Mecanismos de comunicação serão adicionados para estabelecer, através do formalismo das redes de Petri, as influências existentes entre o modelo de atividades do nível e o modelo do mapa topológico. Com isso, será obtido um modelo global composto pelo modelo de atividades e pelo modelo do mapa topológico, que poderá então ser analisado considerando certas importantes propriedades. Em particular, essas propriedades irão mostrar se será possível acessar todas as áreas de um mundo virtual e executar todas as

atividades planejadas em um nível de jogo, evitando assim problemas de inconsistência nos cenários de jogos.

Um dos pontos importantes para a execução prática deste trabalho será a utilização da ferramenta CPN Tools. O CPN Tools é um conjunto de ferramentas funcionais para a manipulação de modelos em redes de Petri Coloridas. Em particular, com o CPN Tools será possível representar graficamente os modelos de atividades de um nível de jogo, o modelo que representará o mapa topológico e o mecanismo de comunicação entre esses dois modelos de forma simples e clara. Além disso, será possível realizar a análise do modelo global obtido por meio de funcionalidades de análise disponíveis no CPN Tools.

1.1 Contribuições

Este trabalho apresenta as seguintes contribuições:

- definição de um modelo baseado nas WorkFlow nets para a formalização de cenários de vídeo games. Tal modelo tem por objetivo expressar o fluxo de atividades existentes em um nível de jogo.
- Definição de um modelo baseado nos grafos de estado (um tipo particular de redes de Petri) para a formalização da noção de mapa topológico de um jogo. Tal modelo tem por objetivo a representação do mundo virtual do jogo. A formalização do mapa topológico, por meio de um grafo de estado, consistirá da representação das regiões do mundo virtual e da representação das condições necessárias para se ter acesso a essas regiões.
- Definição de um mecanismo de comunicação para especificar as interações existentes entre o modelo de atividades e o modelo do mapa topológico. A representação de um mecanismo de comunicação, também por meio das redes de Petri, expressa de forma direta a relação existente entre os modelos, permitindo assim que ambos modelos possam ser tratados como um único modelo global que representa um nível de jogo.
- Análise qualitativa de cenários de vídeo games utilizando as boas propriedade das redes de Petri. A definição de um modelo global que envolve tanto o modelo de atividades de um nível de jogo quanto o modelo de mapa topológico, em um único formalismo, permite que seja realizada uma análise global do nível de jogo. A análise consistirá da verificação de certas propriedades importantes, que garantirá a consistência de cenários de vídeo games.

1.2 Organização da Dissertação

Esta dissertação está dividida em seis capítulos que são organizados como segue.

No capítulo 2 são apresentados fundamentos, relacionados a vídeo games, que são importantes para o conhecimento do processo de criação de vídeo games. Além disso, neste capítulo, também são apresentados vários trabalhos que contribuem para o processo de criação de jogos no contexto de métodos formais, especificamente através do uso das redes de Petri.

O capítulo 3 apresenta os fundamentos teóricos necessários para o entendimento da abordagem proposta. A seção 3.1 apresenta as definições de redes de Petri e de suas principais propriedades. A seção 3.2 apresenta os conceitos relacionados às WorkFlow nets, tais como a definição de uma WorkFlow net, o conceito de processo e roteamento e um critério importante de corretude chamado *soundness*. A seção 3.3 apresenta a definição das redes de Petri de tipo grafo de estado. A seção 3.4 apresenta os conceitos relacionados ao CPN Tools. E na seção 3.5 que são apresentas as considerações finais do capítulo 3.

O capítulo 4 apresenta a abordagem proposta neste trabalho. Em particular, a seção 4.1 apresenta o método proposto para a modelagem das atividades de um nível de jogo. Na seção 4.2 é apresentado o método proposto para a modelagem da noção de mapa topológico de jogo. A seção 4.3 apresenta a criação de um mecanismo de comunicação para representar o relacionamento entre o modelo de atividades e o modelo do mapa topológico. A criação deste mecanismo de comunicação por meio das redes de Petri Coloridas é apresentada na seção 4.4. A seção 4.5 apresenta o método proposto para a análise do modelo global de nível obtido com a construção dos modelos de atividade e de mapa topológico. E, por último, a seção 4.6 trás as considerações finais do capítulo 4.

O capítulo 5, por sua vez, apresenta um estudo de caso aplicado ao jogo Dream:scape para ilustrar a abordagem proposta neste trabalho. A seção 5.1 apresenta a especificação do jogo Dream:scpae. A seção 5.2 apresenta a representação do modelo topológico do jogo. A seção 5.3 apresenta os modelos de atividades relacionados a cada nível do jogo, bem como o modelo global obtido (modelo de atividades + mapa topológico) de cada nível do jogo. A análise de cada nível do jogo é apresentada na seção 5.4, e as considerações finais a respeito do capítulo 5 são apresentadas na seção 5.5.

Por fim, o capítulo 6 apresenta a conclusão deste trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Um dos maiores desafios de um projeto de jogo é fornecer conteúdo que seja agradável ao público. O único aspecto original de um jogo, que parece separá-lo do desenvolvimento de *software* tradicional é a exigência para que ele seja divertido (LEWIS; WHITEHEAD, 2011). Encontrar tal requisito não é meramente trivial. Diferentemente de outros aplicativos, os vídeo games proporcionam desafios únicos que resultam de várias disciplinas que compõe o desenvolvimento de jogos. (CLAYPOOL; CLAYPOOL, 2005) afirmam que para gerenciar os grandes projetos de jogos, a indústria de jogos usa uma mistura de técnicas e conceitos emprestados de áreas como o desenvolvimento de *softwares*, a indústria do cinema e conhecimento sobre jogos tradicionais.

Um jogo é um sistema fechado e formal que, subjetivamente, representa um subconjunto da realidade (CRAWFORD, 1982). O termo fechado diz respeito ao fato do jogo possuir um universo próprio e formal, em particular regras e leis que direcionam o jogador ao objetivo principal. Já para (KANODE CHRISTOPHER M. E HADDAD, 2009), um vídeo game é uma síntese de código, imagens, música e atuação que se reúnem em forma de entretenimento. Em geral, aplicações de jogos diferem de *softwares* tradicionais pela fase de pré-produção e por meio do uso e integração de recursos multimídias (KANODE CHRISTOPHER M. E HADDAD, 2009). Essa fase de pré-produção é um tipo de coleta de requisitos e criação de protótipos do jogo. O desenvolvimento desses jogos, do ponto de vista clássico da indústria de vídeo games, é composto de duas etapas principais denominadas *game design* e *level design* (GAL et al., 2002).

A etapa de *game design* pode ser definida como sendo a fase que descreve os aspectos principais do universo de um jogo, determinando cada detalhe de como o jogo irá funcionar (GAL et al., 2002). Época, estilo, contexto, objetivo a ser alcançado, como o usuário irá controlar o jogo, entre outros aspectos são definidos nessa fase. Em outras palavras, a natureza do jogo é determinada nesta fase. Os componentes principais da fase de *game design* (GAL et al., 2002) são:

- Contexto do jogo: época, estilo, referências históricas ou místicas.

- ❏ Cenário global: topologia, gráficos de navegação, personagens principais, natureza e hierarquia dos níveis.
- ❏ Principais características do jogo que o torna único.
- ❏ Os princípios de jogabilidade: modalidades, objetivos, regras e escolhas estratégicas principais.
- ❏ Imagens e sons.
- ❏ Princípios de ergonomia: interface, aprendizagem de jogo, opções como salvar ou reiniciar o jogo, entre outros.
- ❏ Classes dos objetos no jogo.

Na etapa de *level design*, todos os diferentes componentes do jogo se unem. A descrição de como o jogador irá interagir com os objetos do jogo, a descrição das missões, os objetivos a alcançar e o nível de execução dessas tarefas compõe o *level design*. Geralmente, nessa etapa, os problemas existentes do jogo se tornam mais aparentes (ROUSE; RYBCZYK, 2001).

Em geral, um jogo possui um ou mais níveis (*levels*). Um nível de jogo consiste de um espaço virtual, quebra-cabeças, ações principais e de um conjunto de objetos que irão interagir para o objetivo ser alcançado. Cada nível tem que ser significativo e tem de manter a coerência com o tema e a meta central do jogo. Dessa forma, é preciso levar em consideração a lógica de sequenciamento das ações relativas aos objetos do jogo, o posicionamento desses objetos no universo do jogo, bem como as limitações do mundo virtual.

Essas duas etapas do processo de criação de jogos, *game* e *level design*, são fundamentais para o sucesso do projeto. Aspectos importantes e decisivos para a história do jogo são elaborados e descritos nessas fases.

Em geral, os jogos eletrônicos estão surpreendendo seus públicos cada vez mais por meio de novas tecnologias e disciplinas utilizadas em seu desenvolvimento. Geralmente, o ambiente e a narrativa que muitos jogos apresentam têm a capacidade de submergir o jogador e motivá-lo a cumprir todas as tarefas propostas pelo jogo. Em particular, todo jogo deve possuir uma narrativa que move o jogador. Uma narrativa é uma sequência ordenada de eventos. Um evento é um elemento atômico da história ou alguma coisa significante que ocorre (COLLÉ; CHAMPAGNAT; PRIGENT, 2005). Além da narrativa, o ambiente também tem grande influência sob a motivação do jogador. O ambiente ajuda a proporcionar a sensação de realidade em um jogo e é caracterizado pela presença de cenários. Um cenário é um conjunto de sequências ordenadas de eventos (COLLÉ; CHAMPAGNAT; PRIGENT, 2005). Os cenários estão relacionados com o gênero do

jogo e, independente desse gênero, todo jogo possui um cenário, por mais simples que seja.

Em termos de *game design*, o cenário é a descrição clássica de uma sequência de ações das principais fases do jogo e como ocorre a navegação entre essas fases. Já em *level design*, os cenários correspondem ao posicionamento dos objetos em relação às missões do jogo. Isso induz uma sequência de ações parcialmente ordenadas que o jogador deve executar para chegar ao final de um determinado nível do jogo (GAL et al., 2002).

Quanto ao gênero, é possível encontrar várias classificações de jogos de acordo com vários critérios propostos pela literatura. (ROLLINGS; MORRIS, 2003) descrevem uma classificação baseada no foco principal do jogo:

- Jogos de ação: leva o jogador a apertar o mais rápido possível uma sequência de botões. Exemplo: jogos de luta como *Street Fighter*.

- Jogos de aventura: o jogador é um herói de um cenário complexo. Esse gênero é um dos mais relacionados com o cenário audiovisual clássico. Exemplo: *Metal Gear Solid* II.

- Jogos de estratégia: tem como característica decisões complexas que devem ser tomadas pelo jogador em um universo político, econômico ou militar. Exemplo: *R.U.S.E.*

- Jogos de simulação: levam o jogador a exercitar um esporte ou operar algum dispositivo físico simulado. Exemplo: *Pro Evolution Soccer*.

- Jogos de quebra-cabeça: o objetivo é resolver um difícil desafio analítico. Exemplo: *Heavy Rain*.

- Jogos de descobertas: consiste em descobrir uma história e resolver enigmas incorporados nos jogos. Jogos de descobertas são equivalentes a documentários no campo audiovisual. Exemplo: o jogo *Versalhes* leva o jogador a descobrir a vida do Rei Luiz XIV através de um jogo de enigmas.

- Jogos de RPG (*Role Playing Games*): é uma mistura dos jogos de ação, aventura e estratégia. Exemplo: *Final Fantasy* e *Legend of Zelda*.

(COLLÉ; CHAMPAGNAT; PRIGENT, 2005) afirmam que os editores de vídeo games tendem a criar mais e mais jogos baseados em cenários, para fazê-los mais atrativos. Dessa forma, os projetos de jogos se tornam então mais complexos. A Engenharia de Software se faz indispensável para ajudar a suprir os desafios de desenvolvimento, melhorando a gerência de projeto, diminuindo os riscos e garantindo o sucesso futuro da indústria de jogos (KANODE CHRISTOPHER M. E HADDAD, 2009).

Um dos maiores problemas do desenvolvimento de jogos encontrado por (KANODE CHRISTOPHER M. E HADDAD, 2009) é a utilização de metodologias de criação pobres, fazendo com que projetos não sejam finalizados no tempo estimado e com custos elevados. Isto ocorre devido ao gerenciamento errôneo das atividades na fase de pré-produção do jogo (prototipagem, engenharia de requisitos, estimativas, etc.). (LEWIS; WHITEHEAD, 2011) citam que as maiores falhas de desenvolvimento de jogos se encontram na etapa de levantamento de requisitos.

A principal forma existente atualmente para especificar conceitos de jogabilidade é por meio do uso da linguagem natural (NEIL, 2012). A ludologia (o estudo dos jogos em geral, particularmente dos vídeo games) aponta a necessidade de criar modelos para especificar os mecanismos dos jogos, pois a falta deste conhecimento tem sido um dos grandes problemas do tradicional projeto de jogos (REYNO; CUBEL, 2009).

Uma abordagem utilizada para modelagem de jogos é a de métodos formais. Os métodos formais são técnicas matemáticas, frequentemente suportadas por ferramentas para desenvolvimento de sistemas de *software* e *hardware*. Métodos formais usam modelos matemáticos para análise e verificação em certas partes do ciclo de vida de um programa (WOODCOCK et al., 2009). Os modelos formais são rigorosamente definidos e não contém especificações ambíguas (LEWIS; WHITEHEAD, 2011). Por meio de modelos formais, é possível que o projetista explore o projeto do jogo usando técnicas manuais ou automáticas para simular as passagens entre os níveis do jogo e verificar certas propriedades esperadas. Além disso, a modelagem por meio de modelos formais é barata, proporciona uma maneira rápida de conhecimento do projeto do que a produção de um protótipo e oferece a oportunidade de refinar o projeto antes de ter ocorrido os testes de jogo (LEWIS; WHITEHEAD, 2011) . (KINIRY; ZIMMERMAN, 2011) acreditam que a integração entre métodos formais e desenvolvimento de jogos pode causar efeitos positivos para essas duas áreas de estudo.

No contexto de métodos formais, as redes de Petri têm se destacado no que diz respeito ao desenvolvimento de projetos de jogos. Por serem consideradas como uma ferramenta gráfica e matemática para a modelagem e análise de sistema a eventos discretos, particulamente aplicável a muitos sistemas, as redes de Petri constituem uma abordagem interessante a ser aplicada a projetos de jogos (SYUFAGI; HARIADI; PURNOMO, 2013). As redes de Petri, vistas como uma simples ferramenta gráfica, são expressivas e de fácil entendimento, além de proporcionar técnicas de modelagem, simulação e verificação (ARAÚJO; ROQUE, 2009). Alguns trabalhos já consideram as redes de Petri como uma ferramenta eficiente para modelagem e análise de sistemas de jogos.

(ARAÚJO; ROQUE, 2009) ilustram com um estudo de caso como as redes de Petri podem ser utilizadas com certas vantagens em relação a outras linguagens de modelagem. Em (ARAÚJO; ROQUE, 2009) diagramas baseados em redes de Petri foram utilizados para representar as possíveis ações dos jogadores em relação aos objetivos do jogo. Assim,

o fluxo do jogo pode ser mapeado e simulado pelas redes de Petri. A notação gráfica das redes de Petri é simples e pode ser utilizada para modelar jogos complexos. Além disso, a estrutura matemática das redes de Petri permite que os sistemas modelados sejam formalmente analisados. Também a simulação dos comportamentos de jogo oferece a possibilidade de detectar problemas ainda na fase de projeto do jogo (ARAÚJO; ROQUE, 2009).

Em (THOMAS; YESSAD; LABAT, 2011) é apresentada a representação do fluxo de jogo utilizando redes de Petri. Para verificar o caminho de aprendizagem de um jogador, é preciso rastrear todas as ações para analisar e diagnosticar a aquisição do conhecimento do jogador. Para tanto, (THOMAS; YESSAD; LABAT, 2011) desenvolveram uma abordagem baseada em redes de Petri para modelar o comportamento exato do jogador e acompanhar o seu progresso no jogo. Assim, a rede criada representa um caminho ideal para aprendizagem e as transições da rede representam as ações realizadas pelo jogador. O caminho de aprendizagem é analisado então pela verificação de disparo efetivo das transições da rede de Petri. Para completar este método, os autores utilizaram a noção de ontologia para tentar explicar os erros de aprendizagem.

Em (OLIVEIRA; JULIA; PASSOS, 2011) e (OLIVEIRA, 2012), é apresentada uma nova abordagem baseada em um tipo particular de redes de Petri, denominado WorkFlow net, para especificar cenários existentes em um jogo. Nesta abordagem, uma WorkFlow net representa o fluxo de atividades que deve ser executado pelo jogador a fim de alcançar um objetivo específico no jogo, sem considerar os recursos necessários para tal. Esse fluxo de atividades é associado à noção de *quest*, ou seja, uma missão que o jogador deve executar. Em termos de modelo, cada *quest* é um subprocesso de uma WorkFlow net maior. A integração de diversas *quests* formam uma rede de *quests* e é por meio dela que o resultado completo da análise será estabelecido. Como cada *quest* é um subprocesso, caso aconteça alguma mudança em uma *quest* já estudada, um novo estudo das boas propriedades será feito apenas para a *quest* que sofreu alteração (OLIVEIRA, 2012). Além disso, os autores realizaram uma análise qualitativa por meio de árvores de prova da lógica linear. Em (OLIVEIRA; JULIA; PASSOS, 2011) a tradução dos modelos em redes de Petri para árvores da lógica linear, tem o objetivo de provar a propriedade *soundness* da rede que corresponde a consistência do cenário modelado do ponto de vista do jogo.

A lógica linear também foi utilizada para análise de jogos em (LEWIS; WHITEHEAD, 2011). Os autores explicam que o interesse de usar a lógica linear para análise de um cenário é a possibilidade de expressar ações e recursos do jogo sem reter informações inúteis. Na lógica linear, os personagens e recursos são representados por átomos. Enquanto que as relações implicativas são usadas para projetar eventos. Esse método proporciona a verificação de propriedades de cenários de jogos que podem ser validadas antes da execução dos jogos. As classes de propriedades principais citadas em (LEWIS; WHITEHEAD, 2011) são: jogabilidade (verifica se o curso do jogo está correto) e relevância do cenário

(verifica se o jogo é interessante). Após expressar os cenários utilizando fragmentos da lógica linear, o modelo é traduzido para um modelo em rede de Petri. Esse modelo permite gerar então as possíveis narrativas para um dado cenário.

Em (LEE; CHO, 2012) é apresentado um mecanismo para gerar enrendo de *quest* consistindo na representação de eventos baseados em redes de Petri. Um enredo de *quest* é uma sequência de eventos para alcançar um objetivo específico. E um evento consiste de ações que devem ser executadas. As rede de Petri são então utilizadas para representar um evento do jogo. Na rede, um lugar representa uma pré-condição para uma ação ou o armazenamento do resultado de uma ação. Os arcos representam relacionamentos casuais entre uma ação e sua pré-condição. As transições representam as ações do jogador e as fichas representam o estado de uma ação em um evento. O gerador de enredo baseado em redes de Petri, proposto em (LEE; CHO, 2012), foi então experimentado em uma plataforma de jogo comercial. De acordo com os autores, além de ser aplicável, o método proposto proporciona um método baseado em roteiro mais formal. Segundo (LEE; CHO, 2012), as redes de Petri ajudaram a identificar elementos passivos (tais como as condições) e elementos ativos (tais como as ações) da história, facilitou a representação de eventos independentes, assim como a representação de restrições e a sincronização dos eventos.

Uma nova abordagem para o processo de projeto de jogos que envolve a modelagem de relações espaço-temporais foi proposta por (NATKIN; VEGA; GRÜNVOGEL, 2004). A metodologia apresentada por (NATKIN; VEGA; GRÜNVOGEL, 2004) para combinar as relações espaciais e temporais utiliza os conceito de redes de Petri e hiper-grafos. Hiper-grafos são generalizações de grafos. Seus nós (vértices) são conectados por hiper-arestas que são definidas como um par ordenado de subconjuntos disjuntos de vértices. Sendo assim, uma hiper-aresta pode conectar vários nós. De acordo com (NATKIN; VEGA; GRÜNVOGEL, 2004), as hiper-arestas possuem mais possibilidades de apresentar relacionamentos espaciais do que as arestas de grafos comuns, que só podem conectar dois nós. Os hiper-grafos são utilizados então para representar a topologia do mundo virtual. Já as redes de Petri modelam as transações lógicas e temporais relacionadas ao jogo. Cada modelo representa um conjunto de atividades que pode ser executado pelo jogador em determinado momento do jogo. Para unir essas duas estruturas, (NATKIN; VEGA; GRÜNVOGEL, 2004) utilizaram "conexões" que tratam de substituir uma hiper-aresta do hiper-grafo pela árvore de alcançabilidade da rede de Petri. A cada vez que as tarefas são realizadas, os lugares do mapa topológico são liberados e as hiper-arestas substituídas. Embora o trabalho de (NATKIN; VEGA; GRÜNVOGEL, 2004) aborde o aspecto espaço-temporal do jogo, o uso de dois formalismos (rede de Petri e hiper-grafo) inviabiliza um estudo global do modelo de jogo. Para verificar a corretude do jogo, é necessário então analisar os modelos separadamente de acordo com seus formalismos.

2.1 Considerações Finais

Este capítulo apresentou fundamentos relacionados a vídeo games, como *game* e *level design*. Tais fundamentos fazem parte do processo de desenvolvimento de vídeo games e são importantes para o entendimento deste processo. Especialistas afirmam que é preciso utilizar diferentes métodos que auxiliem no processo de criação de jogos para que se possa atender as exigências cada vez maiores do mercado de vídeo games.

Várias contribuições foram apresentadas para o processo de desenvolvimento de jogos no contexto de métodos formais, especificamente em redes de Petri. Os métodos apresentados propõe a utilização das redes de Petri como ferramenta para auxiliar na representação do fluxo de atividades de jogos. A representação do fluxo de atividades ou de eventos que ocorrem em um jogo, descreve principalmente as ações que o jogador deve realizar para alcançar a meta do jogo. Com as redes de Petri, é possível representar formalmente tal fluxo de atividades e também rastrear as ações do jogador durante o jogo, caso seja necessário.

A grande maioria dos métodos propostos ignora completamente a representação do mundo virtual do jogo que é tão importante quanto o fluxo das atividades. O mundo virtual diz respeito ao ambiente em que o jogador executará as atividades. A proposta apresentada por (NATKIN; VEGA; GRÜNVOGEL, 2004) para a representação do mundo virtual utiliza um formalismo diferente do que é proposto para a representação do fluxo de atividades. O uso de formalismos diferentes impossibilita a análise global do jogo sendo necessário então realizar uma análise específica para cada formalismo. A análise global de um jogo consiste da análise do fluxo de atividades do jogo e a análise do ambiente onde as atividades serão executadas. Dessa forma, é possível identificar problemas que podem levar o jogo ao fracasso, como por exemplo, inconsistências no sequenciamento das atividades ou na distribuição de objetos a serem encontrados no mundo virtual.

No método proposto por este trabalho, tanto o fluxo de atividades quanto a representação do mundo virtual serão representados por meio do mesmo formalismo, as redes de Petri. Os modelos serão criados separadamente e um mecanismo de comunicação (também baseado em redes de Petri) será utilizado para descrever as influências que um modelo tem sobre o outro. Isso possibilitará que a análise global seja realizada, isso a fim de detectar possíveis inconsistências tanto nas atividades propostas quanto no ambiente topológico.

Capítulo 3

Fundamentos Teóricos

Este capítulo tem como objetivo introduzir conceitos relacionados com redes de Petri, WorkFlow nets, grafos de estado e CPN Tools que serão necessários para o entendimento deste trabalho. Sendo assim, as seções 3.1, 3.2, 3.3 e 3.4 trazem os conceitos sobre as redes de Petri, WorkFlow nets, grafos de estado e CPN Tools, respectivamente. E por fim, a seção 3.5 apresenta as considerações finais sobre o capítulo.

3.1 Redes de Petri

A teoria inicial das redes de Petri foi proposta em 1962 por Carl Adam Petri. Apesar de ser considerada uma teoria relativamente jovem, a teoria se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes (CARDOSO; VALETTE, 1997).

Redes de Petri são consideradas como uma ferramenta gráfica e matemática de representação formal que permite modelagem, análise e controle de sistemas a eventos discretos que comportam atividades paralelas, concorrentes e assíncronas (MURATA, 1989). Uma rede de Petri é um grafo bipartido e direcionado com dois tipos de nós chamados de lugares e transições. Os nós são conectados por meio de arcos direcionados. Conexões entre dois nós do mesmo tipo não são permitidas. Dessa forma, os elementos básicos que permitem a definição das redes de Petri são:

- **Lugar**: em uma rede de Petri um lugar é representado por um círculo. Um lugar pode ser interpretado como uma condição, um estado parcial, um procedimento, a existência de um recurso, etc..

- **Transição**: uma transição é representada graficamente por uma barra ou retângulo. São associadas as transições eventos que ocorrem em um sistema.

- **Ficha**: a ficha é representada por um ponto em um lugar. É um indicador significando que a condição associada ao lugar é verificada, como por exemplo, um recurso

disponível em um certo processo.

A figura 1 ilustra um exemplo de uma rede de Petri e seus elementos básicos.

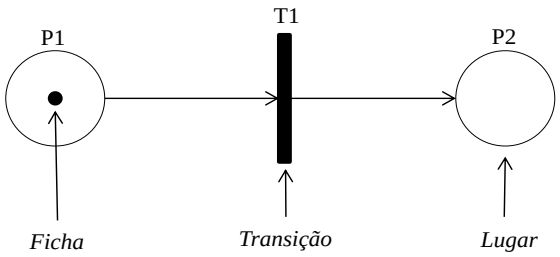


Figura 1 – Rede de Petri.

Formalmente, as redes de Petri são definidas da seguinte forma ((CARDOSO; VALETTE, 1997) e (MURATA, 1989)):

Definição 1 (Rede de Petri) *Uma rede de Petri é uma quádrupla R = <P, T, Pré, Pós> onde:*

- *P: é um conjunto finito de lugares;*
- *T: é um conjunto finito de transições;*
- *Pré: é uma relação que define os arcos que ligam os lugares às transições;*
- *Pós: é uma relação que define os arcos que ligam as transições aos lugares.*

Um lugar p é chamado lugar de entrada de uma transição t, se existe um arco direcionado de p para t. Por exemplo, na figura 1 o lugar *P1* é lugar de entrada da transição *T1*. Já um lugar é chamado lugar de saída de uma transição se existe um arco direcionado de t para p. Ainda na figura 1, o lugar *P2* é um lugar de saída de *T1*.

Em uma rede de Petri, a ocorrência de um evento no sistema é representada pelo disparo da transição ao qual este evento está associado. Uma transição t só pode ser disparada se em cada lugar de entrada de t existe pelo menos uma ficha. A ficha indica se a condição associada ao lugar está ou não satisfeita. O disparo de uma transição, segundo (CARDOSO; VALETTE, 1997), consiste de: (1) retirar as fichas dos lugares de entrada e (2) depositar fichas em cada lugar de saída. Retirar as fichas de um lugar indica que esta condição não é mais verdadeira após a ocorrência do evento. Por outro lado, ao depositar fichas em um lugar indica que esta atividade está sendo executada após a ocorrência do evento. Deste modo, no momento em que um evento ocorre, a rede passa de um estado discreto para outro, alterando sua marcação. Assim, o comportamento dinâmico do sistema é traduzido pelo comportamento da rede de Petri.

Um exemplo de funcionamento de um sistema por meio das redes de Petri pode ser visto na figura 2. A figura 2 ilustra um modelo em rede de Petri que representa algumas

tarefas em uma linha de montagem. Neste modelo, os lugares representam os recursos e os estágios do processo, enquanto que as transições representam os eventos. O modelo representa um pacote que precisa ser montado e enviado para outro setor. Para montar o pacote é preciso ter disponível parafusos, porcas e a máquina, que são representados no modelo pelos lugares *Parafusos*, *Porcas* e *Máquina*. As fichas nesses lugares indicam a disponibilidade desses recursos. Após a montagem do pacote, ele pode ser enviado ao próximo setor, o depósito. Uma vez enviado ao depósito, a máquina fica disponível para a montagem de outro pacote.

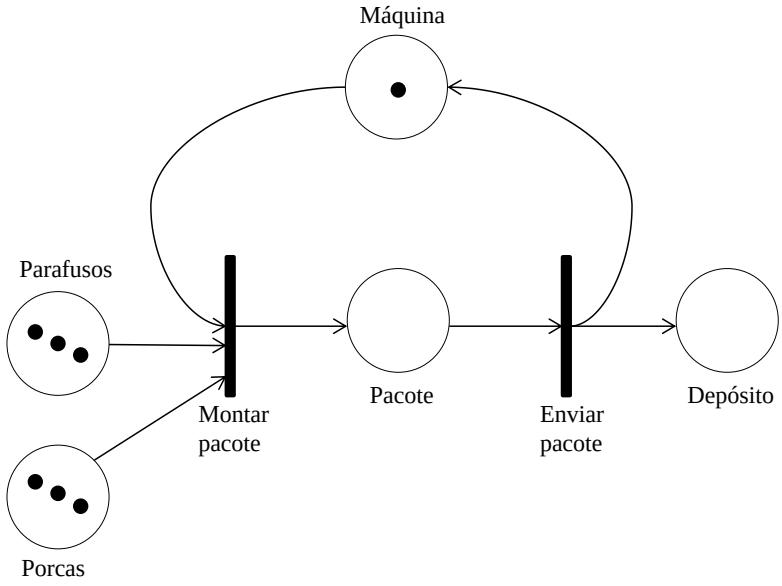
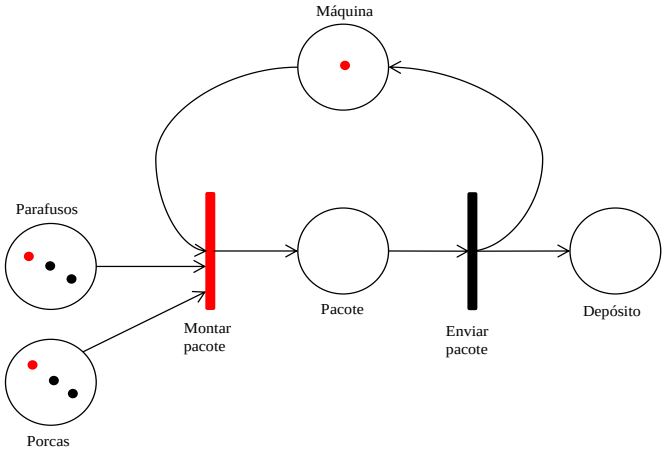


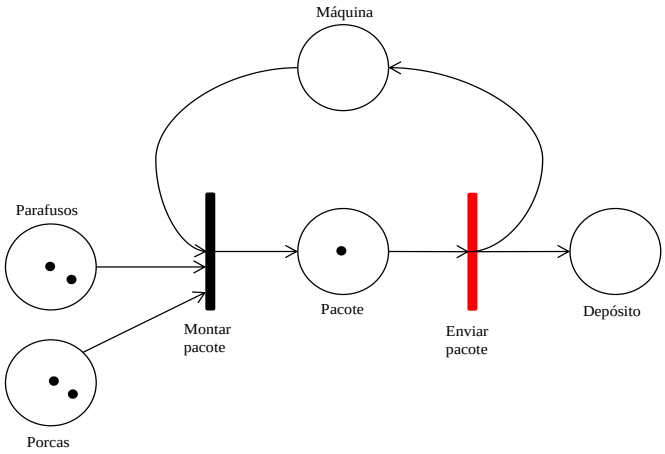
Figura 2 – Modelo em rede de Petri de uma linha de montagem.

A transição *Montar pacote* estará apta para ser disparada quando existir pelo menos uma ficha no lugar *Parafusos*, uma no lugar *Porcas* e uma no lugar *Máquina* (figura 3(a)). Após o disparo da transição *Montar pacote*, uma ficha é adicionada no lugar *Pacote*, que indica a montagem do conjunto. Uma vez com o pacote pronto, este pode ser enviado ao depósito (figura 3(b)). Este evento é representado pela transição *Enviar pacote*. Após o disparo desta transição, uma ficha é adicionada no lugar *Depósito* e outra é adicionada no lugar *Máquina*, indicando que este recurso está novamente disponível (figura 3(c)).

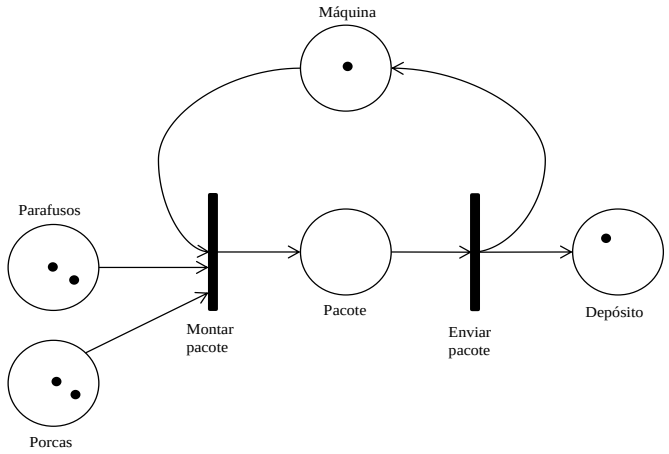
É importante notar que as interpretações dos lugares e fichas são bastante variadas. Podem representar entidades abstratas, como condições, e também entidades físicas como um depósito (como no exemplo da figura 2). Dessa forma, as redes de Petri permitem uma visão sintética do sistema a ser modelado e autoriza procedimentos de análise (CARDOSO; VALETTE, 1997). Uma revisão completa sobre as redes de Petri pode ser encontrada em (MURATA, 1989) e (CARDOSO; VALETTE, 1997).



((a)) Montar pacote.



((b)) Enviar pacote ao depósito.



((c)) Pacote enviado.

Figura 3 – Comportamento da rede de Petri.

3.1.1 Propriedades das redes de Petri

Redes de Petri não se resumem apenas a uma ferramenta que permite a modelagem de problemas que tenham atividades concorrentes, elas são utilizadas para descrever e analisar um sistema (RENÉ; HASSANE, 2010). Para tal, foi desenvolvida uma série de

métodos que permitem a análise de várias propriedades de sistemas a eventos discretos ((MURATA, 1989) e (PETERSON, 1981)).

A dinâmica de um sistema descrito por uma rede de Petri é dada pela evolução das marcações. De acordo com o conjunto de marcações acessíveis a partir da marcação inicial, são definidas algumas propriedades. Essas propriedades são definidas em (MURATA, 1989) como: alcançabilidade, limitabilidade, vivacidade e reiniciabilidade. Esse conjunto de propriedades é conhecido na literatura como boas propriedades.

- **Alcançabilidade**: essa propriedade indica a possibilidade de alcançar uma determinada marcação. Uma marcação é dita alcançável se existe uma sequência finita de disparo de transições que conduza a ela a partir da marcação inicial. Essa propriedade garante que certos estados serão alcançados ou não.

- **Limitabilidade** (*Boundedness*): uma rede de Petri é dita limitada se em cada lugar da rede, o número total de fichas nunca excederá a um inteiro positivo k. Neste caso, a rede é dita k-limitada. Se a rede for limitada ao inteiro 1, então diz-se que ela é binária.

- **Vivacidade** (*Liveness*): uma rede de Petri é dita viva se todas as suas transições são vivas. Uma transição T é dita viva se para cada marcação alcançável da rede, existe uma sequência de disparo S que sensibiliza T. Dessa forma, uma rede de Petri viva é uma rede onde todas as suas transições são disparáveis. Essa propriedade garante que o sistema é livre de *deadlock*.

- **Reiniciabilidade**: uma rede é dita reiniciável se é sempre possível voltar para a marcação inicial através de uma sequência de disparos, seja qual for a marcação considerada.

3.2 WorkFlow nets

Uma rede de Petri que modela um processo de negócio é chamada de WorkFlow net (WF-net) ((AALST, 1998) e (AALST; HEE, 2004)). Uma WF-net net satisfaz os seguintes requisitos (AALST, 1998):

- Uma WF-net tem apenas um lugar de início (i) e apenas um lugar de fim (o). Esses dois lugares são tratados como lugares especiais. O lugar i tem apenas arcos de saída e o lugar o tem apenas arcos de entrada. Uma ficha em i representa um caso que precisa ser tratado. Uma ficha em o representa um caso que já foi tratado.

- Em uma WF-net cada tarefa (transição) e condição (lugar) deve estar em um caminho que se encontra entre o lugar de início (i) e o lugar de término (o).

3.2.1 Processos

Um processo especifica quais tarefas precisam ser executadas e em qual ordem executá-las. Modelar um processo de negócio em termos de uma WF-net é bem direto: as tarefas são modeladas por transições, condições são modeladas por lugares e os casos são modelados pelas fichas (AALST, 1998).

Para ilustrar o mapeamento de um processo por uma WF-net, considera-se o exemplo apresentado em (AALST, 1997) que trata do processamento de pedidos relacionados a danos de carros. As tarefas necessárias para processar um pedido são: verificar seguro, contatar garagem, pagar danos e enviar carta. As tarefas *verificar seguro* e *contatar garagem* determinam se o pedido é justificado. Essas tarefas podem ser executadas em qualquer ordem. Se o pedido é justificado, então o dano é pago (tarefa *pagar danos*). Caso contrário, uma carta de rejeição é enviada ao requerente (tarefa *enviar carta*). A figura 4 ilustra a WF-net que representa este processo.

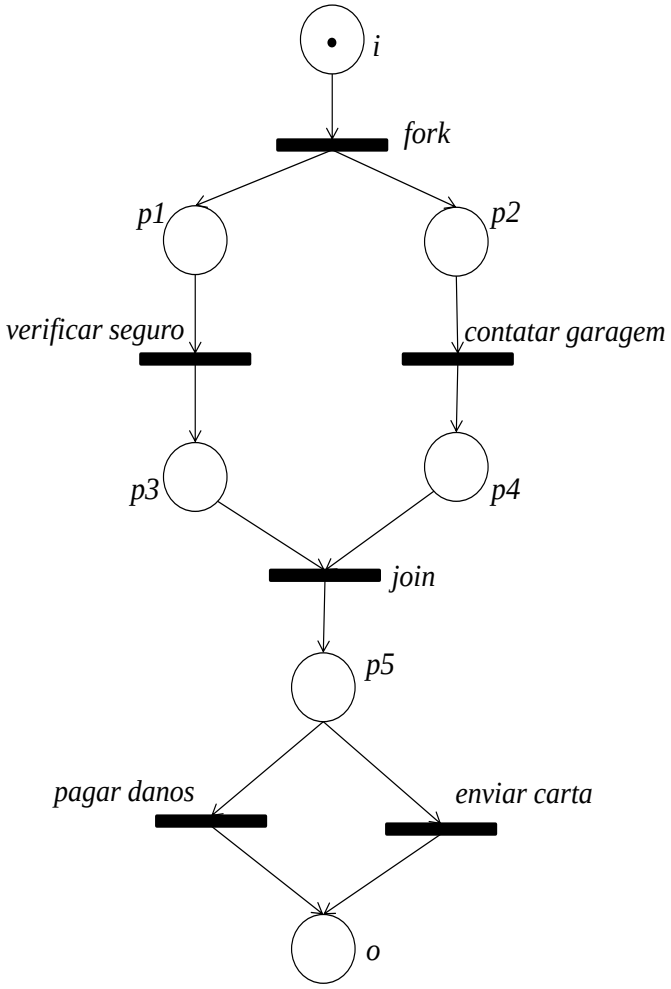
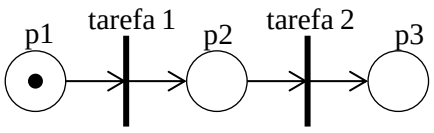


Figura 4 – WorkFlow net para o processo de pedidos relacionados a danos de carros.

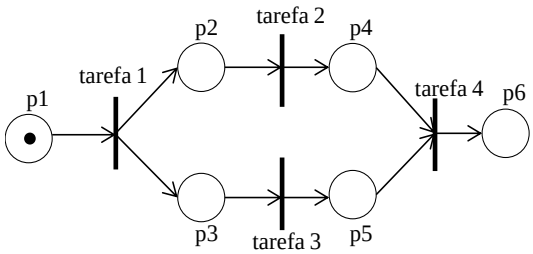
3.2.2 Roteamento

A ordem em que tarefas são executadas varia de caso para caso. Pelo roteiro de um caso ao longo de uma série de tarefas, é possível determinar quais tarefas precisam ser executadas e em qual ordem (AALST; HEE, 2004). Quatro construções básicas são consideradas para o roteamento de tarefas:

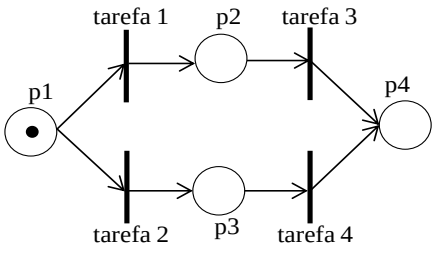
- **Sequencial** (figura 5(a)): refere-se a execução sequencial quando as tarefas precisam ser executadas uma após a outra. Quando uma tarefa precisa ser executada após a outra, têm-se então uma dependência entre essas tarefas.
- **Paralela** (figura 5(b)): quando mais de uma tarefa pode ser executada simultaneamente ou em qualquer ordem. Ambas tarefas podem ser executadas sem que uma interfira no resultado da outra.
- **Condicional** (figura 5(c)): quando pode existir uma escolha entre duas ou mais tarefas.
- **Iterativa** (figura 5(d)): quando é necessário repetir uma mesma tarefa ou uma sequência de tarefas várias vezes.



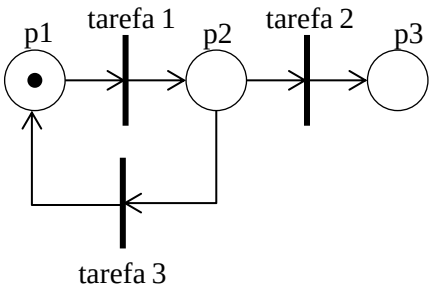
((a)) Sequencial.



((b)) Paralela.



((c)) Condicional.



((d)) Iterativa.

Figura 5 – Construções básicas para o roteamento de tarefas.

Na figura 4 as tarefas *verificar seguro* e *contatar garagem* são um exemplo de roteamento paralelo. Já as tarefas *pagar danos* e *enviar carta* compõe um roteamento condicional.

3.2.3 Soundness

Soundness é um critério de corretude definido para as WorkFlow nets. Uma WorkFlow net é dita *sound* se, e somente se, os seguintes requisitos forem satisfeitos ((AALST; HEE, 2004) e (AALST, 1998)):

1. Para cada ficha colocada no lugar de início i, uma, e apenas uma, ficha deve aparecer no lugar de término o.

2. Quando uma ficha aparece no lugar o, todos os outros lugares estão vazios para o caso em questão.

3. Para cada transição (tarefa), é possível evoluir da marcação inicial para uma marcação onde essa transição é sensibilizada, ou seja, em uma WorkFlow net não deve haver transição morta.

A propriedade *soundness* está relacionada com a dinâmica das WorkFlow nets. O primeiro requisito significa que todo caso será completado após um período de tempo. O segundo requisito significa que uma vez que um caso é completado, nenhuma referência a ele permanecerá no processo. Já o terceiro requisito afirma que todas as tarefas em uma WF-net podem ser, a princípio, executadas.

A propriedade *soundness* é um critério importante a ser satisfeito quando se trata de processos de negócios. No contexto das WF-nets, a prova desse critério está relacionada com a análise qualitativa. Em (AALST; HEE, 2004), alguns métodos para provar a propriedade *soundness* são apresentados. Um desses métodos em específico será discutido na seção seguinte.

3.2.4 Algoritmo de verificação da propriedade soundness

Em (AALST, 1998) e (AALST; HOFSTEDE, 2000), um método foi proposto para a verificação da propriedade *soundness* de uma WorkFlow net. Este método traduz a propriedade *soundness* através de duas propriedades bem conhecidas: *liveness* e *boundedness* (AALST; HEE, 2004).

Dado uma WorkFlow net PN = (P,T,F), deve-se decidir se PN é *sound*. Para tanto, define-se uma rede estendida $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$. $\overline{PN}$ é uma rede de Petri obtida adicionando-se uma transição extra t^*. A transição t^* conecta o lugar de término (o) ao lugar de início (i). A figura 6 ilustra a relação entre PN e $\overline{PN}$.

Para uma WorkFlow-net arbitrária PN e sua correspondente rede de Petri estendida $\overline{PN}$, o seguinte teorema (AALST, 1997) pode ser provado.

Teorema 1 (*Soundness*) *Uma WF-net é dita sound se, e somente se, ($\overline{PN}$, i) é viva (live) e limitada (boundedness).*

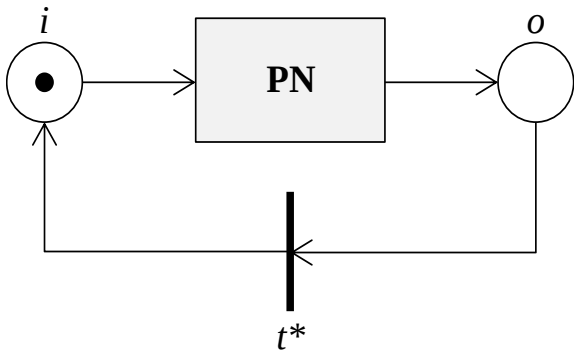


Figura 6 – Exemplo de uma rede de Petri estendida.

A prova deste teorema pode ser encontrada em (AALST, 1997) e (AALST, 1996).

Dessa forma, a verificação da propriedade *soundness* em uma WF-net, resume-se em verificar se a rede de Petri estendida $\overline{PN}$ é viva e limitada. Isso significa que ferramentas de análise baseadas em redes de Petri podem ser utilizadas para definir o critério *soundness* da rede ((AALST, 1996), (AALST, 1997) e (AALST, 1998)).

3.3 Grafos de estado

Uma rede de Petri não marcada é um grafo de estado se, e somente se, cada transição tiver exatamente um arco de entrada e um arco de saída (RENÉ; HASSANE, 2010), como ilustrado na figura 7.

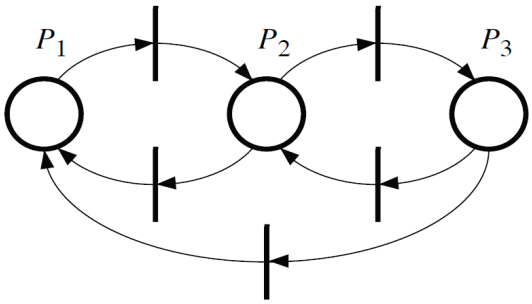


Figura 7 – Exemplo de um grafo de estado não marcado.

Uma rede de Petri marcada conhecida como grafo de estado, será equivalente a um grafo de estado no sentido clássico (representando um autômato que está em apenas um estado por vez) se, e somente se, contém exatamente uma ficha localizada em um dos lugares da rede. Na figura 8 é apresentado um exemplo de um grafo de estado com uma ficha. Em um grafo de estado, o peso de todos os arcos é igual a 1 (RENÉ; HASSANE, 2010).

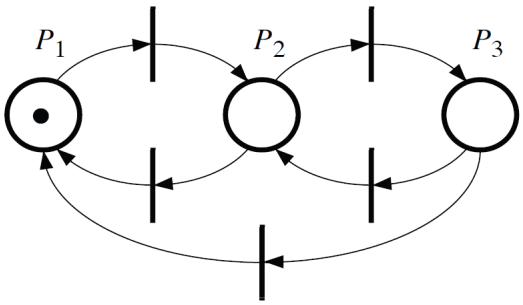


Figura 8 – Exemplo de um grafo de estado marcado com apenas uma ficha.

3.4 CPN Tools

As redes de Petri Coloridas são uma linguagem de modelagem gráfica que combina as redes de Petri e as linguagens de programação funcional (MILNER; HARPER; TOFTE, 1990). Uma das razões para a criação das redes de Petri Coloridas é o fato de que as redes de Petri (não coloridas) tendem a ser muito grandes para lidar. Outra razão é o fato de que as fichas geralmente representam objetos ou recursos em um sistema modelado (AALST, 1992). Esses objetos podem ter atributos, os quais não são facilmente representados por um ficha simples de uma rede de Petri que não identifica as fichas de um mesmo lugar. Foi então que (JENSEN, 1981) definiu o que hoje se conhece por redes de Petri Coloridas, ou CP-nets (*Colored Petri Nets*).

Nas redes de Petri Coloridas, as fichas possuem valores que são conhecidos como cores. Essas cores não significam apenas cores ou padrões, elas podem representar tipos de dados complexos (JENSEN; KRISTENSEN, 2009). Dessa forma, as redes de Petri Coloridas são projetadas para reduzir o tamanho do modelo, permitindo a individualização de fichas (usando as cores que lhe são atribuídas). Assim, diferentes processos ou recursos podem ser representados na mesma estrutura gráfica.

A aplicação prática da modelagem e análise das redes de Petri Coloridas depende fortemente da existência de ferramentas computacionais que ofereçam suporte à criação e manipulação de tais modelos. CPN Tools é uma ferramenta adequada para edição, simulação e para análise do estado de espaço (*state space*) dos modelos de redes de Petri Coloridas (JENSEN, 1998).

Nos modelos de redes de Petri Coloridas (CP-nets), cada lugar da rede tem um *tipo* associado. Esse *tipo* determina qual tipo de dado o lugar pode ter. Por exemplo, a figura 9 ilustra um exemplo de um modelo CP-net. O lugar *p2* tem um tipo de dado inteiro (*INT*) associado a ele. Já o lugar *p1* tem um tipo de dado composto pelo produto cartesiano de dois conjuntos de dados: *INT* (inteiro) e *DATA* (dados).

Durante a execução de uma CP-net, cada lugar poderá conter um número determinado de fichas. Cada ficha carrega um valor de dado que pertence ao tipo associado ao lugar. Por exemplo, o valor da ficha que está em *p1* na figura 9 é:

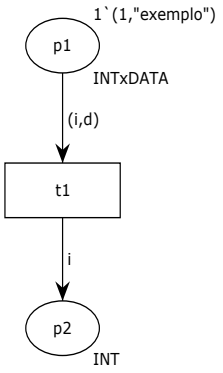


Figura 9 – Exemplo de uma rede de Petri Colorida.

```
1‘(1, ‘‘exemplo’’).
```

O coeficiente “ *1‘* ” na frente da ficha significa que há exatamente uma ficha que carrega esse valor. Da mesma forma, poderia haver mais fichas em *p1* com valores diferentes. Os valores das fichas são chamados de cores. Os tipos de dados são chamados de *color sets*. As cores podem representar tipos de dados complexos, como por exemplo, um registro onde um dos campos é do tipo real, outro uma cadeia de caracteres e um terceiro como uma lista de inteiros.

As ações em uma CP-net são representadas por meio das transições. Em uma transição, os arcos de entrada indicam que a transição pode remover fichas do lugar correspondente ao arco, enquanto que os arcos de saída indicam que a transição pode adicionar fichas aos lugares correspondentes. O número exato de fichas e seus valores de dados são indicados pela expressão do arco (localizada ao lado do arco). Por exemplo, na figura 9, a transição *t1* tem um arco de entrada e um arco de saída. A expressão do arco de entrada possui duas variáveis livres: **i** do tipo inteiro, e **d** do tipo *data*. Assim, *t1* só poderá ser sensibilizada se houver ao menos uma ficha que contenha valores do tipo $INT\text{x}DATA$. Para o exemplo da figura 9, durante a execução, as variáveis **i** e **d** possuem os seguintes valores:

```
i = 1
d = ‘‘exemplo’’.
```

O CPN Tools permite estruturar os modelos CP-net em módulos. O conceito de módulos em CP-nets é baseado em um mecanismo de estruturação hierárquica. A ideia básica da hierarquia por trás das CP-nets é permitir a construção de um amplo modelo combinando um número de pequenas redes de Petri Coloridas em um único modelo (JENSEN; KRISTENSEN, 2009).

A hierarquia das redes de Petri Coloridas oferecem um conceito conhecido como *fusion places* (lugares de fusão). Esse conceito permite especificar um conjunto de lugares que são considerados idênticos (JENSEN, 1998). Isto significa que todos esses lugares

representam um único lugar conceitual, ainda que sejam desenhados como vários lugares individuais. Esses lugares são chamados de *fusion places*, e um conjunto de *fusion places* é chamado de *fusion set* (conjunto de fusão). Qualquer coisa que acontece a um lugar do conjunto de fusão, também acontece aos outros lugares do conjunto. Assim, se uma ficha for adicionada/consumida de um dos lugares, uma ficha idêntica também será adicionada/consumida em todos os outros lugares do conjunto de fusão.

Na ferramenta CPN Tools, os *fusion places* são criados usando o recurso de hierarquia que a ferramenta oferece. Ao definir um *fusion place*, uma etiqueta é adicionada ao lugar. A etiqueta define o nome do conjunto de fusão ao qual o lugar pertence.

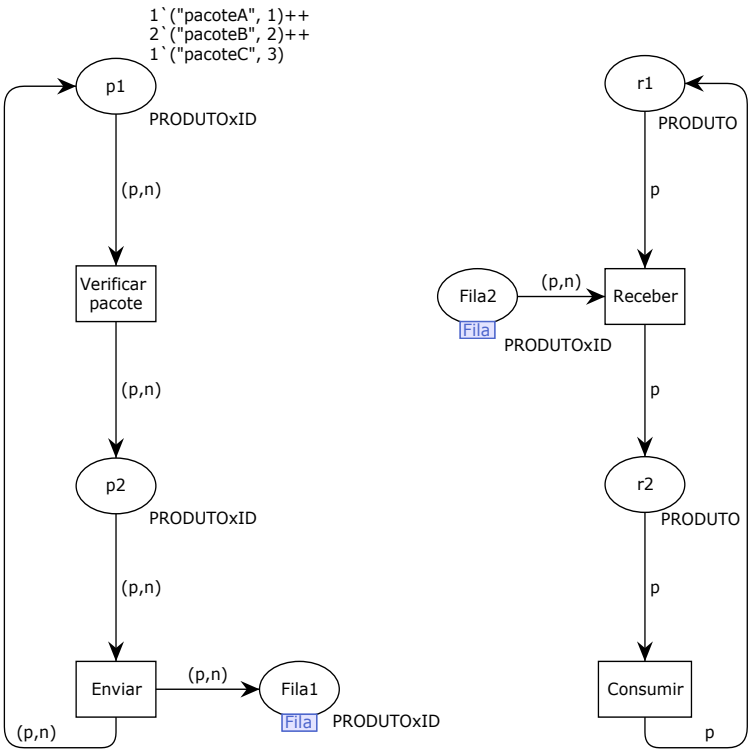


Figura 10 – Exemplo de uma rede de Petri Colorida com *fusion places*.

O exemplo da figura 10 representa um procedimento onde pacotes são verificados e enviados para uma outra entidade. Após verificados, os pacotes são então colocados em uma fila de espera para serem entregues. Ao serem recebidos, esses pacotes são então consumidos. A fila de espera é representada pelos lugares *Fila1* e *Fila2*. Assim, esses dois lugares são então definidos como *fusion places* e marcados com a etiqueta *Fila*, que representa o conjunto de fusão a que esses lugares pertencem.

Quando todos os membros de um conjunto de fusão pertencem a mesma parte ou página (no contexto de CPN Tools) da CP-net, e tem apenas uma instância, os *fusion places* são nada mais que um mecanismo conveniente para evitar vários cruzamentos de arcos (JENSEN; KRISTENSEN, 2009). Dessa forma, é possível simplificar a estrutura gráfica da rede sem mudar o seu significado.

3.5 Considerações Finais

Este capítulo apresentou os fundamentos teóricos necessários para o entendimento deste trabalho. Primeiramente, foi apresentado o conceito geral de redes de Petri e suas propriedades. Em seguida, foi apresentado um tipo especial de rede de Petri, as Work-Flow nets. As redes do tipo grafos de estado também foram apresentadas. Os conceitos relacionados ao CPN Tools foram também descritos neste capítulo.

Capítulo 4

Proposta

Este capítulo tem por objetivo apresentar como o fluxo de atividades e a noção de mapa topológico de jogos podem ser representados por meio das redes de Petri. Os modelos são criados separadamente, entretanto, para expressar as influências que um modelo tem sobre o outro, um mecanismo de comunicação, também baseado em redes de Petri, é utilizado. Dessa forma, é possível realizar uma análise global a fim de verificar certas propriedades que garantem o bom funcionamento do jogo. O jogo Silent Hill II é utilizado para demonstrar o método proposto neste capítulo.

Silent Hill II é um jogo de aventura/terror que começa em uma cidade deserta. O jogador assume o papel de James, um personagem que não possui objetivos muito claros no começo do jogo. James recebe uma estranha carta de sua esposa chamando-o para se encontrarem em Silent Hill. No entanto, a esposa de James está morta a três anos. Incentivado pelo mistério da carta, James retorna a Silent Hill. Para alcançar o objetivo do jogo, o jogador deve explorar o ambiente, lutar contra inimigos, coletar objetos e resolver enigmas.

Neste capítulo, a seção 4.1 apresenta a proposta para a modelagem das atividades de um nível de jogo (modelo lógico). A seção 4.2 apresenta a proposta para a modelagem do mapa topológico (modelo topológico). Já a seção 4.3, apresenta um mecanismo de comunicação criado para representar o relacionamento entre o modelo de atividades e o modelo topológico. A seção 4.4, apresenta a especificação deste mecanismo de comunicação por meio das redes de Petri Coloridas; em particular esta seção apresenta o uso de *fusion places* bem como as vantagens deste conceito para a abordagem proposta. O método proposto para a análise do nível do jogo é apresentado na seção 4.5. Por fim, a seção 4.6 apresenta as considerações finais deste capítulo.

4.1 Modelo Lógico

Para a representação do modelo lógico, é preciso considerar o conceito de nível (*level*). O uso do termo nível em vídeo games tem sido empregado há muito tempo e, de modo

geral, pode ser abordado de duas formas. A primeira associa o termo nível à dificuldade da fase do jogo. Já a segunda, associa nível a uma determinada etapa do jogo. Neste trabalho, é utilizada a segunda forma de abordagem. Assim, um nível é tido como uma parte do jogo.

A maioria dos jogos pode ser estruturada em um grupo de níveis. Um nível pode ser visto como um grupo de tarefas (atividades). Todo jogo possui um objetivo geral que o jogador precisa alcançar com o intuito de vencer. Em particular, cada nível tem um objetivo específico associado. Para alcançar esse objetivo é necessário que o jogador execute uma determinada sequência de tarefas. Essas tarefas devem ser executadas pelo jogador sequencialmente ou simultaneamente. Além disso, algumas tarefas são obrigatórias e outras opcionais. Quando todas as tarefas do nível são executadas, o objetivo do nível é alcançado e o jogador pode então passar de um nível para outro.

Uma vez que um jogo é composto por vários níveis e um nível é composto por uma sequência de atividades, as WorkFlow nets são adequadas para a modelagem de níveis de jogos. Uma WorkFlow net modela um processo de negócio. De fato, um nível de jogo é similar à estrutura clássica de um processo de negócio, pois ambos possuem um começo e um objetivo final que será alcançado após a execução de várias atividades. Assim, a estrutura de uma WorkFlow net torna possível a modelagem de níveis de vídeo games.

Para ilustrar esta abordagem, o primeiro nível do jogo Silent Hill II é utilizado. Para completar o primeiro nível de Silent Hill II, o jogador precisa executar a seguinte sequência de atividades:

- ❏ Encontrar o mapa de Silent Hill no estacionamento da plataforma de observação.
- ❏ Matar a criatura e pegar o rádio no túnel.
- ❏ Encontrar o bilhete no trailer.
- ❏ Encontrar um segundo mapa no Neely's Bar.
- ❏ Encontrar a chave em um cadáver na rua Martin.
- ❏ Ir até o apartamento Wood Side.

Após executar essas atividades, o jogador completa o primeiro nível do jogo e poderá passar para o próximo nível. A figura 11 representa o modelo lógico do primeiro nível de Silent Hill II. Na WorkFlow net, que representa o fluxo de atividades, as transições representam as atividades do nível enquanto que os lugares da rede representam as condições. O lugar inicial da rede representa o começo do jogo e o lugar final representa o fim do nível. Portanto, na rede da figura 11 o lugar *P0* representa o começo do primeiro nível e o lugar *P10* representa o objetivo que deve ser alcançado para finalizar o primeiro nível. A ficha no lugar *P0* representa o jogador.

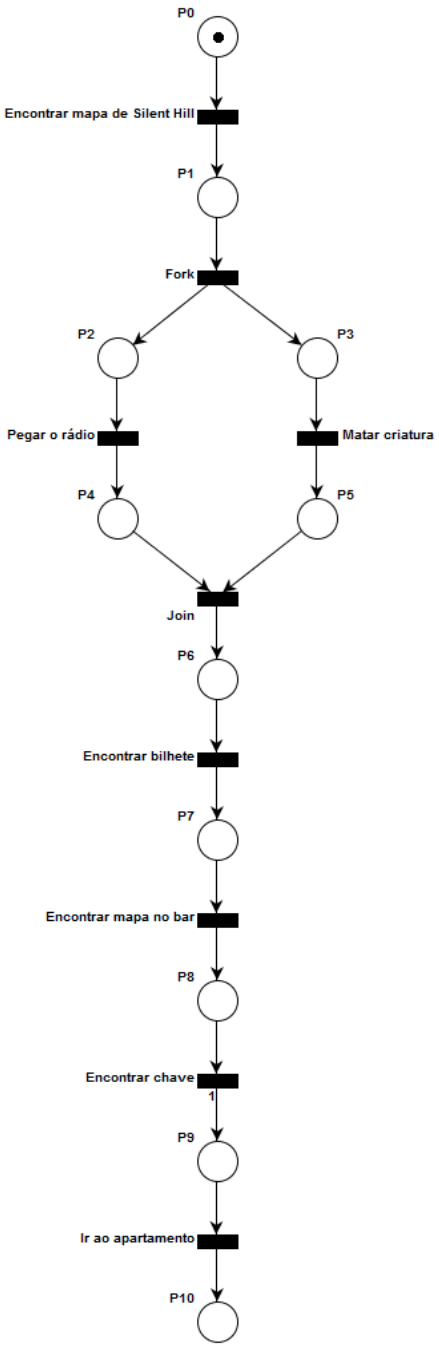


Figura 11 – Modelo lógico do primeiro nível de Silent Hill II.

A primeira tarefa do nível é *Encontrar mapa de Silent Hill*. As transições *Fork* e *Join* representam o início e o fim de uma rotina paralela, respectivamente. Fazem parte dessa rotina as atividades *Pegar rádio* e *Matar criatura*. As atividades *Encontrar bilhete*, *Encontrar mapa no bar*, *Encontrar chave* e *Ir ao apartamento* são atividades que pertencem a uma rotina sequencial, portanto devem ser executadas uma após a outra. A atividade *Ir ao apartamento* é a última atividade do primeiro nível. Após completar todas as atividades, o jogador se encontrará no lugar *P10*, o que significa que o objetivo do primeiro nível foi alcançado e que o jogador poderá passar para o nível seguinte.

4.2 Modelo Topológico

A estrutura topológica de um jogo consiste da descrição das regiões que compõe o mapa do jogo. Em outras palavras, a estrutura topológica é a representação do mundo virtual. Em um jogo, cada atividade é executada pelo jogador em alguma região específica do mapa topológico. Assim, a descrição das propriedades do mundo virtual bem como a evolução do jogador dentro deste mundo são indispensáveis para o processo de criação de jogos.

O mapa topológico de um jogo é composto por várias regiões (áreas). Essas regiões podem ser entendidas como um conjunto de lugares que o jogador poderá acessar como, por exemplo, o centro de uma cidade ou o interior de uma casa. O jogador poderá ir de uma área para outra livremente ou satisfazendo algum tipo de condição específica, dependendo dos objetivos do jogo.

A figura 12 ilustra um exemplo de mapa de jogo com suas respectivas regiões. O mapa em questão pertence ao jogo Silent Hill II.

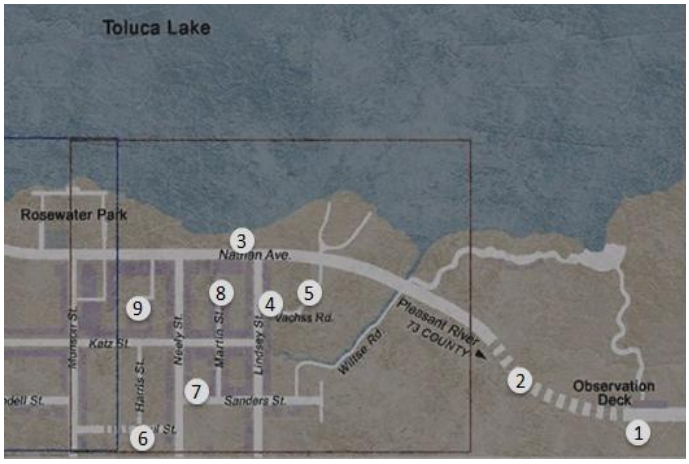


Figura 12 – Mapa parcial do primeiro nível de Silent Hill II.

Os números circulados na figura 12, representam subconjuntos especiais do mundo virtual onde o jogador deve executar tarefas específicas. Essas regiões são fixas, ou seja, não mudam durante a evolução do jogo. O nome de cada uma das regiões do mapa da figura 12 é dado a seguir:

1. Plataforma de observação.
2. Floresta.
3. Igreja.
4. Quintal.
5. Túnel.

6. Trailer.

7. Bar.

8. Rua Martin.

9. Apartamento.

Para representar o mapa topológico de um jogo, é utilizado uma rede de Petri do tipo grafo de estado. No grafo de estado, uma área específica do mapa é modelada por um lugar específico. As fronteiras entre as áreas são representadas por transições simples. A localização do jogador é representada por uma ficha em um lugar específico da rede. A orientação do arco representa em qual direção o jogador pode ir entre duas áreas adjacentes. A figura 13 ilustra a representação do mapa topológico do jogo Silent Hill II por meio de um grafo de estado.

As regiões ilustradas na figura 12, são representadas no grafo de estado da figura 13 por lugares com os mesmos nomes. Inicialmente, o jogador está localizado na *Plataforma de observação*. Assim, uma ficha nesse lugar representa a localização atual do jogador no começo do jogo. De acordo com a evolução do jogo, o jogador mudará sua localização. Entretanto, no começo o jogador não pode acessar todas as áreas conectadas. Para passar de uma área para outra é preciso respeitar algumas condições (requisitos), que dependerão de alguma atividade associada ao modelo lógico do nível. No mapa topológico, essas condições, na maioria dos casos, corresponderão a um objeto necessário para passar de uma área para a outra, como uma chave por exemplo. Por outro lado, algumas condições corresponderão simplesmente à realização de uma tarefa que o jogador deverá executar e não necessariamente à produção de algum item específico do jogo.

Uma maneira de representar formalmente essas condições, pode ser por meio de um recurso a ser produzido, como apresentado na figura 14, onde o disparo da transição *T32* representará a produção do recurso correspondente. O lugar *C1* marcado representará então uma condição válida necessária para habilitar a transição *T0*, que corresponde à liberação da passagem para a área *Floresta*.

A figura 15 ilustra o mapa topológico do primeiro nível de Silent Hill II com as suas respectivas condições.

Para o jogador passar da área *Plataforma de observação* (área 1) para *Floresta* (área 2), por exemplo, o jogador precisa encontrar o mapa de Silent Hill. Essa é a primeira atividade do jogo que, após realizada, ativará a condição associada com o lugar *C1*, liberando assim a passagem entre a área Plataforma de observação e a área Floresta, como indicado na figura 15. Em particular, o disparo da transição *T32* corresponderá ao fim da atividade correspondente *Encontrar o mapa de Silent Hill*. Uma vez com o mapa, o jogador poderá passar da área 1 para a área 2. Depois disso, as únicas áreas que o jogador poderá acessar são *Plataforma de observação*, *Floresta*, *Igreja*, *Quintal* e *Túnel*.

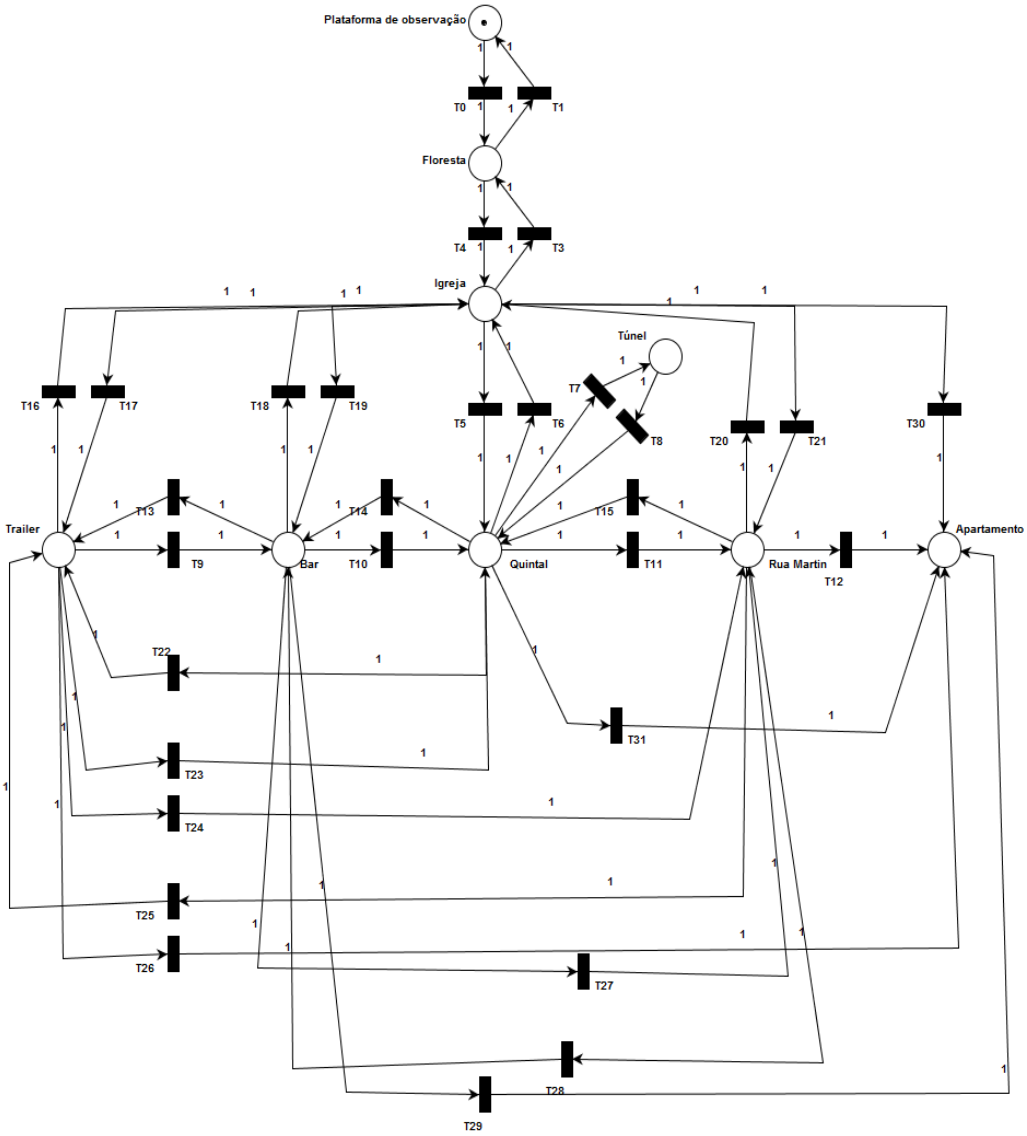


Figura 13 – Grafo de estado do mapa topológico de Silent Hill II.

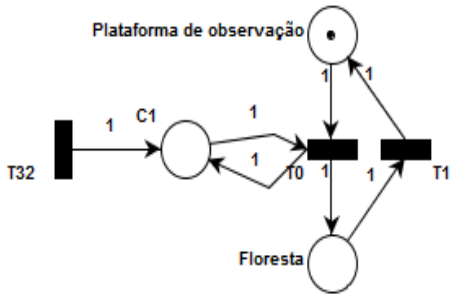


Figura 14 – Condição associada com o mapa topológico.

Para acessar as outras áreas o jogador precisa satisfazer as condições correspondentes. Por exemplo, para entrar na área *Apartamento* o jogador precisará encontrar uma chave. Uma vez com a chave, o jogador terá satisfeito a condição e consequentemente o acesso a

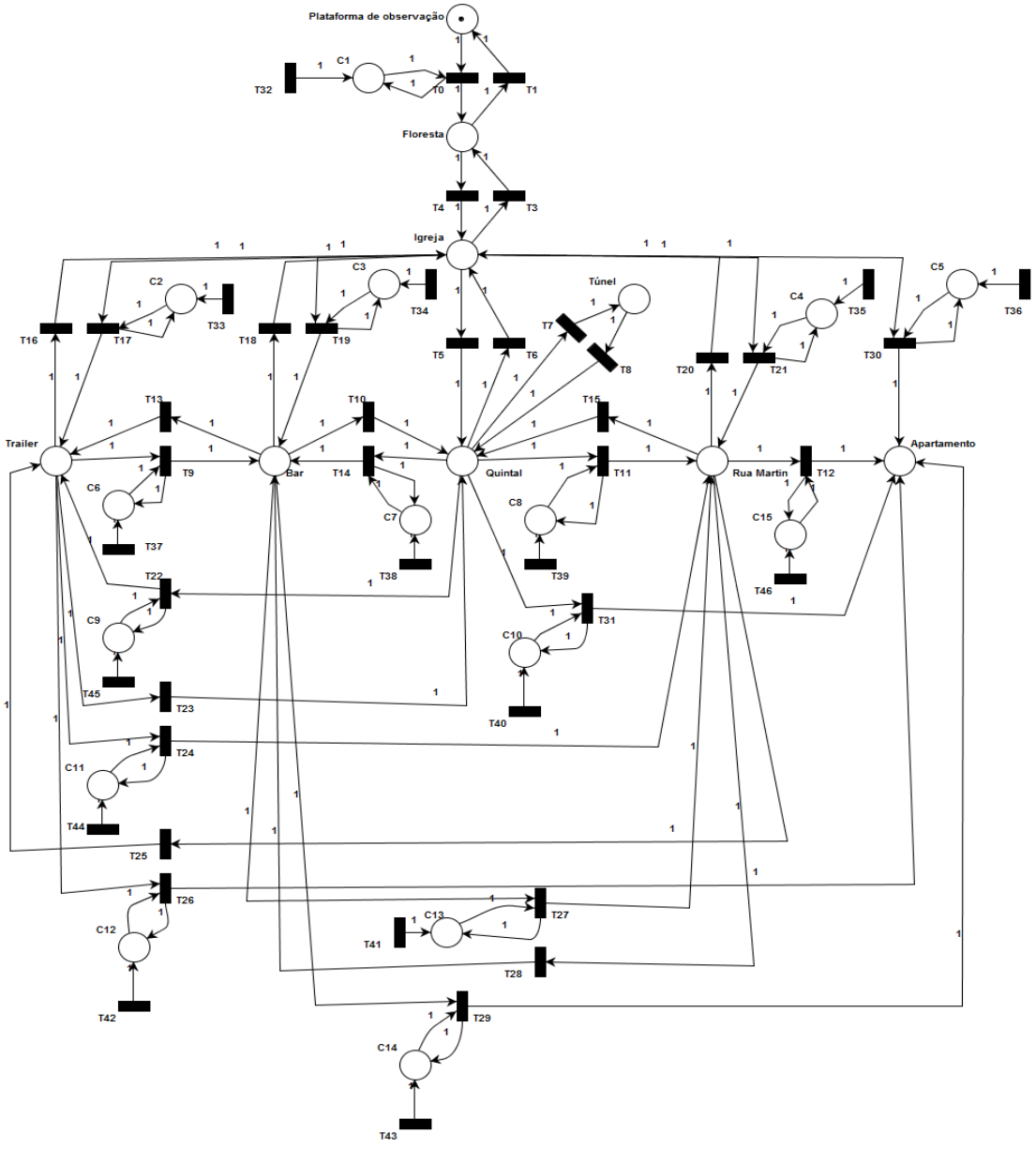


Figura 15 – Mapa topológico do primeiro nível de Silent Hill II com as condições associadas.

área *Apartamento* será liberado.

Após satisfazer uma condição, ela não será mais um obstáculo para a passagem do jogador e ele poderá ir e vir livremente entre as áreas liberadas. Uma vez que a passagem entre áreas adjacentes é liberada, ela continuará aberta até que o nível seja completado. Isso significa que quando uma condição é satisfeita (uma ficha produzida no lugar da condição), o lugar correspondente à condição continuará marcado o tempo todo.

4.3 Modelo de comunicação

Em sistemas de engenharia é comum trabalhar com dois modelos que se comunicam por meio de mensagens síncronas e assíncronas. Por exemplo, em (ANDREU; PASCAL; VALETTE, 1996), a análise e monitoramento de um processo reator é baseado em dois modelos que se comunicam. Um modelo representa os possíveis comportamentos do reator (modelo de um equipamento físico), enquanto que o outro modelo define a receita (modelo lógico) aplicado ao reator para produzir um produto específico. As vantagens em trabalhar com dois modelos distintos que se comunicam é que o modelo de um processo (como uma nova receita por exemplo) pode ser redefinido sem alterar o modelo do sistema físico. O mesmo vale para o outro lado, é possível alterar o modelo do sistema físico sem alterar o modelo lógico. Esse mesmo princípio pode ser aplicado no contexto de vídeo games.

Dado uma WorkFlow net que define o nível de um jogo e um grafo de estado que representa o mapa topológico correspondente, é possível identificar as influências que um modelo tem sobre o outro. No jogo Silent Hill II, o avatar James só pode realizar uma tarefa se ele estiver na área apropriada. Sendo assim, uma transição no modelo de nível estará habilitada apenas se o jogador estiver localizado em um lugar específico do grafo de estado (uma área específica do mapa topológico). Por outro lado, uma condição será ativada no modelo topológico, apenas se um certo item é produzido após completar uma atividade específica descrita no modelo lógico. Dessa forma, é possível definir um mecanismo de comunicação entre os dois modelos.

No jogo Silent Hill II, a condição para o jogador deixar a área *Plataforma de observação* é ter o mapa de Silent Hill. E para encontrar o mapa de Silent Hill o jogador precisa estar localizado na área *Plataforma de observação*. Essa situação é descrita na figura 16.

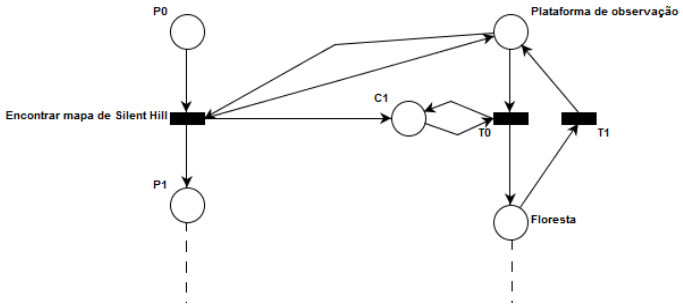


Figura 16 – Comunicação entre o modelo lógico e o modelo topológico.

A figura 16 apresenta um modo formal para especificar o mecanismo de comunicação entre o modelo lógico e o modelo topológico. No lado esquerdo da figura está representado um fragmento do modelo lógico. Do lado direito da figura está representado parte do mapa topológico. A transição *Encontrar mapa de Silent Hill* é habilitada se existir uma ficha no lugar *P0* e uma ficha no lugar *Plataforma de observação*. Depois de disparar a transição, são produzidas três fichas. Uma ficha retorna para o lugar *Plataforma de observação*, outra é produzida em *P1* (continuação do modelo lógico) e outra em *C1* (condição para

a liberação da área *Floresta*). Esse procedimento para relacionar os dois modelos pode ser visto como um tipo de mecanismo de comunicação.

A vantagem de se utilizar dois modelos distintos é que o modelo lógico pode ser alterado sem que o modelo topológico também o seja. O mesmo vale para o modelo topológico; é possível redefini-lo sem que o modelo lógico sofra alterações. Uma vez que existe modelos distintos para expressar as atividades e para expressar o mapa do mundo virtual, é possível simular então o funcionamento do jogo. Assim, o papel do mecanismo de comunicação é fazer com que os dois modelos trabalhem juntos, a fim de verificar o comportamento do jogo. É por meio da verificação do comportamento do jogo que os problemas podem ser encontrados e corrigidos ainda na fase de projeto ou antes da criação de um protótipo, por exemplo.

4.4 Modelagem utilizando o conceito de *fusion places*

Para manter os dois modelos distintos (pelo menos visualmente), uma abordagem interessante é usar o conceito de *fusion places* (lugares de fusão) proposto pelo CPN Tools. Os usuários do CPN Tools trabalham diretamente com a representação gráfica de modelos em redes de Petri Coloridas (CP-net). Sendo assim, é preciso adaptar o modelo lógico e o modelo topológico do jogo para o CPN Tools, para poder adaptar o conceito de *fusion places*.

4.4.1 Modelo lógico para o CPN Tools

Em uma rede de Petri ordinária, as fichas são indistinguíveis, enquanto que nas CP-nets são associados valores (cores) as fichas. Em particular, cada lugar de um modelo CP-net precisa ter um tipo associado a ele, denominado *color set*. Para adaptar o modelo lógico do jogo para o CPN Tools sem alterar a semântica do modelo, o mesmo *color set* *PLAYER* é associado a todos os lugares do modelo; *player* representa o valor associado à ficha que representa o jogador no modelo da figura 17. O modelo ilustrado na figura 17 é a versão em rede de Petri Colorida do modelo lógico, que foi apresentado na figura 11.

A variável *p* associada aos arcos do modelo, pertence ao mesmo tipo *PLAYER* e pode receber fichas da mesma cor. Por exemplo, antes de disparar a transição *Encontrar mapa de Silent Hill*, a variável *p* associada ao arco de entrada da transição recebe o valor da ficha localizada em *P0*. Depois de disparar a transição, o valor da ficha é transferido para a variável *p* associada ao arco de saída da transição, e uma nova ficha com o mesmo valor é produzida em *P1*.

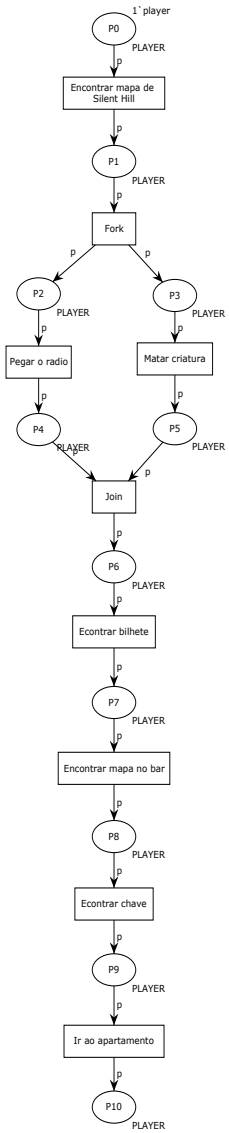


Figura 17 – Versão em rede de Petri Colorida do modelo lógico do primeiro nível de Silent Hill II.

4.4.2 Modelo Topológico para o CPN Tools

Assim como o modelo lógico, o modelo do mapa topológico também é adaptado para a ferramenta CPN Tools. A versão em rede de Petri Colorida do mapa topológico é representada na figura 18, e equivale ao modelo apresentado na figura 15. Para diferenciar os lugares que representam as áreas do mapa dos lugares que representam as condições, foram usados dois *color sets* diferentes. O tipo *PLAYER* é associado aos lugares que representam as áreas do mapa, e o tipo *COND* (bem como uma nova variável *c* do mesmo tipo) é associado aos lugares que representam as condições existentes no mapa topológico.

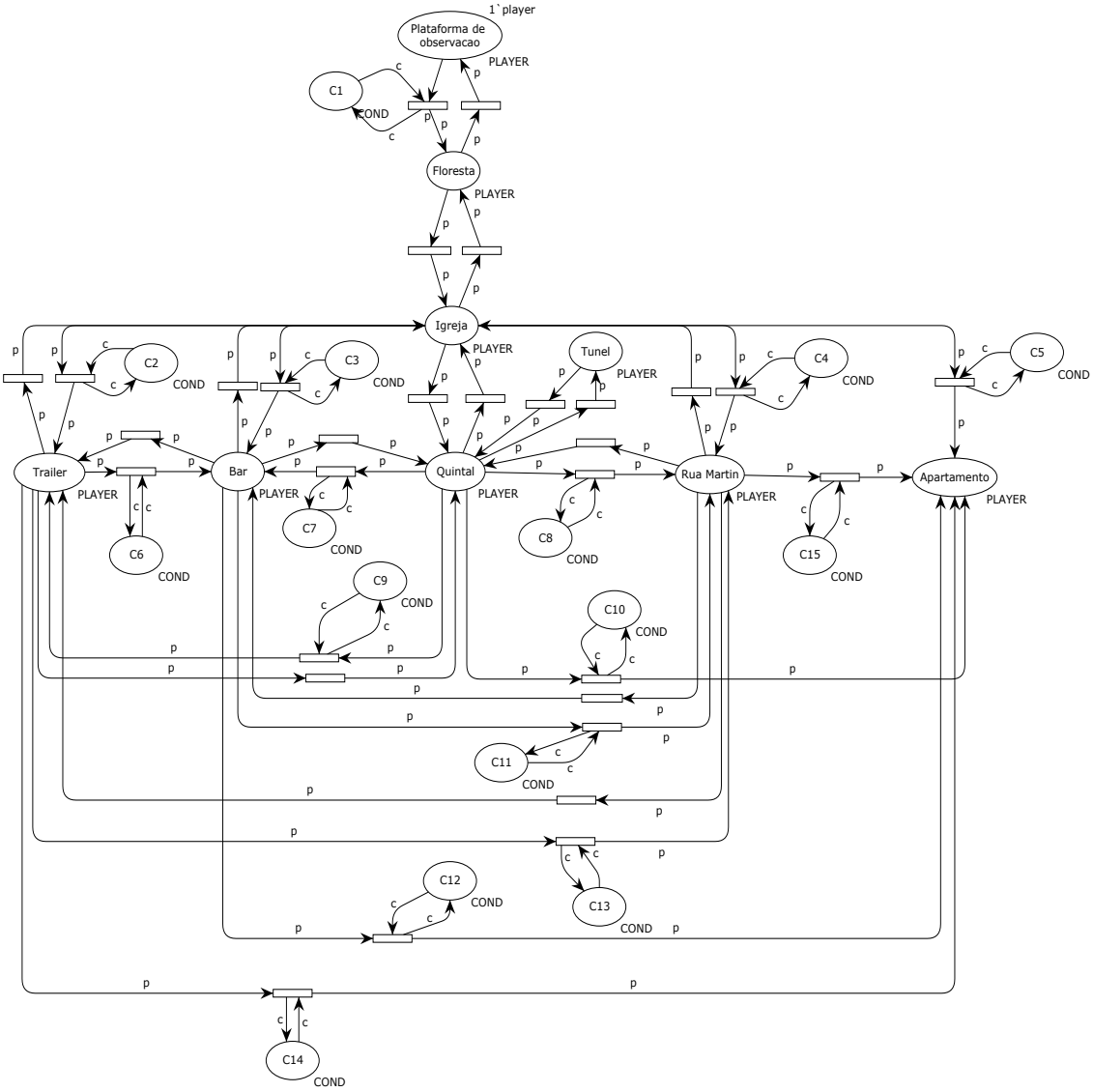


Figura 18 – Versão em rede de Petri colorida do modelo do mapa topológico de Silent Hill II.

4.4.3 Mecanismo de Comunicação para o CPN Tools

Os *fusion places* são utilizados para representar o mecanismo de comunicação entre o modelo lógico e o modelo topológico, e podem ser vistos na figura 19. O mecanismo apresentado na figura 19 é equivalente ao apresentado na figura 16. No entanto, tal mecanismo utiliza o conceito de *fusion places* disponível no CPN Tools.

A marcação que indica um *fusion place* é denominada *fusion tag*. A *fusion tag* é representada por um retângulo azul, na figura 19. Depois do disparo da transição *Encontrar mapa de Silent Hill*, uma ficha é produzida no lugar *c1* do modelo lógico. Uma mesma ficha é então reproduzida automaticamente no lugar correspondente *C1* do modelo topológico. Os lugares *c1* e *C1* estão marcados com a etiqueta *Mapa encontrado*, o que significa que esses lugares pertencem ao mesmo conjunto de fusão.

O exemplo da figura 19, ilustra apenas a primeira atividade do primeiro nível de Silent

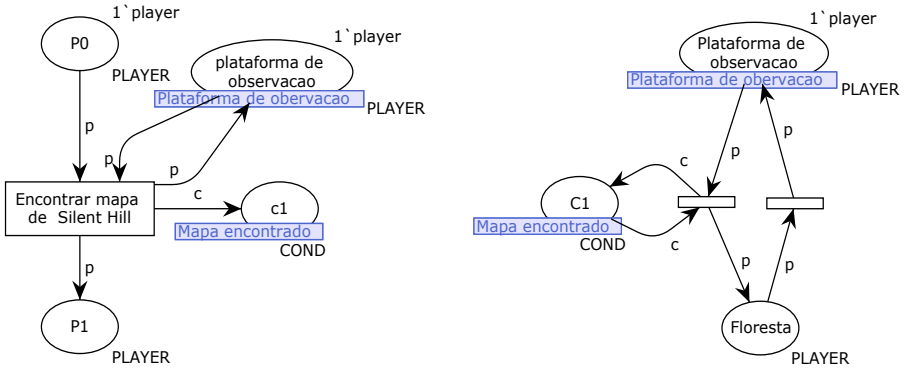


Figura 19 – Comunicação entre o modelo lógico e o modelo topológico usando o conceito de *fusion places*.

Hill II. A comunicação completa entre o modelo lógico e o modelo topológico do primeiro nível de Silent Hill II, utilizando o conceito de *fusion places*, pode ser vista na figura 20.

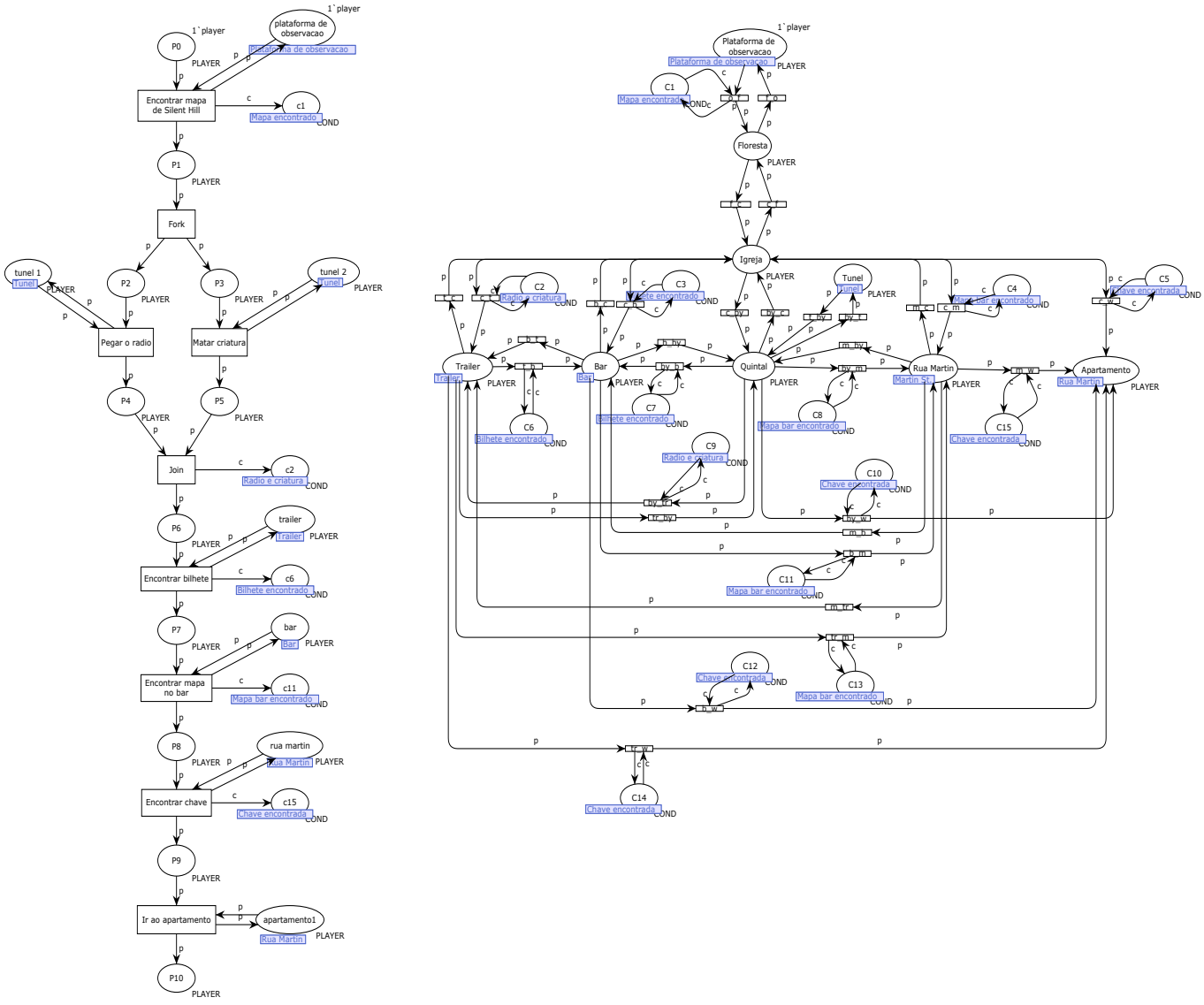


Figura 20 – Comunicação completa entre os modelos do primeiro nível de Silent Hill II.

O uso dos *fusion places* nesta abordagem age como uma conveniência de desenho que permite evitar vários cruzamentos de arcos, simplificando a estrutura gráfica da rede sem alterar seu significado. Dessa forma, os dois modelos distintos que se comunicam são graficamente melhor representados usando o CPN Tools. O CPN Tools possibilita implementar cada modelo separadamente e eventualmente modificá-los, como acontece no caso dos modelos que usam formas de comunicações assíncronas (comunicação através de lugares de comunicação sem a necessidade de usar a fusão de transições). Além disso, é possível utilizar as funcionalidades oferecidas pelo CPN Tools para verificar certas propriedades, que garantem o bom funcionamento do jogo.

4.5 Método para a análise

O modelo global do jogo consiste de um único modelo que é composto pelo modelo que representa a estrutura lógica do jogo, modelo que representa a estrutura topológica e o mecanismo de comunicação entre os dois modelos (*fusion places*). A partir de um modelo global, é possível analisar o jogo do ponto de vista de algumas propriedades importantes que garantem o bom funcionamento do jogo. Assim, o objetivo da análise dos modelos em redes de Petri de um vídeo game é garantir a consistência das atividades do jogo, bem como a boa construção do mundo topológico. O método para a análise proposto por esta abordagem é derivado do método apresentado em (AALST, 1997) sobre verificação da propriedade *soundness* das Workflow nets. No caso deste trabalho, o modelo global não é mais uma WorkFlow net, mas o conceito da propriedade *soundness* ainda pode ser aplicado para em particular verificar que todas as atividades do modelo lógico poderão ser realizadas pelo jogador.

O método de verificação da propriedade *soundness*, propõe criar uma rede de Petri estendida $\overline{PN}$, obtida adicionando uma transição extra t^* que conecta os lugares finais do modelo ao lugares de início do mesmo modelo. Se a rede $\overline{PN}$ é limitada (binária) e viva, de acordo com as definições dessas propriedades, então o modelo correspondente PN é considerado *sound*. Para verificar se o modelo estendido $\overline{PN}$ é vivo e limitado, é possível utilizar ferramentas de análise baseadas em redes de Petri, fornecidas por algumas funcionalidades disponíveis no CPN Tools. O CPN Tools fornece duas abordagens para análise: simulação e espaço de estado (*state space*). No método proposto por este trabalho, os modelos são analisados por meio da funcionalidade *state space*.

A ideia básica por trás do *state space* é calcular todos os estados alcançáveis a partir da marcação inicial de um modelo em rede de Petri, e representá-los como um grafo, onde os nós representam os estados alcançáveis e os arcos representam os possíveis disparos de transição. Algumas propriedades são então verificadas com base em componentes fortemente conexas do grafo obtido.

Para ilustrar o método de análise proposto, o modelo global obtido do primeiro nível

do jogo Silent Hill II foi utilizado. O modelo para análise é ilustrado na figura 21.

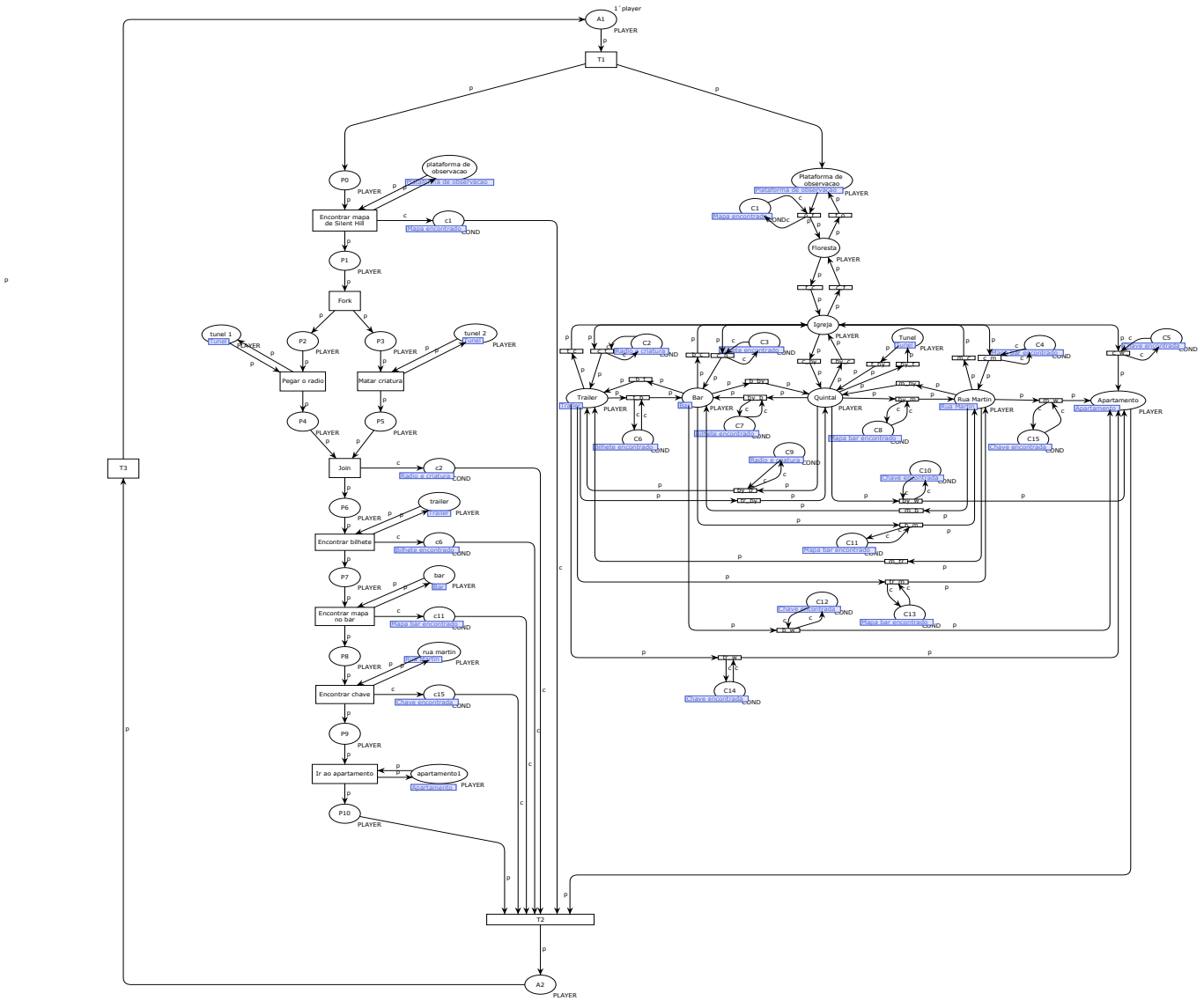


Figura 21 – Modelo para análise.

O lado esquerdo da figura representa o modelo lógico, onde todas as atividades do nível são representadas. O lado direito da figura representa o modelo topológico com todas as áreas do mapa do jogo. Para realizar uma análise global de ambos modelos e suas interações, foi criado um lugar de início comum denominado *A1* e um lugar final denominado *A2*.

No modelo da figura 21, a transição *T1* é uma bifurcação que produz uma ficha no lugar *P0* (lugar de início do modelo lógico) e uma ficha no lugar *Plataforma de observação* (o lugar que representa a área do mapa onde o jogador está inicialmente). A transição *T2* é uma junção que tem o propósito de consumir as fichas que (no caso *sound*) devem estar no lugar final do modelo lógico, no lugar que representa a área do mapa onde o jogador deve estar ao finalizar o nível, e nos lugares de condição marcados que são usados para liberar as passagens entre as diversas áreas do mapa. A transição *T3* irá por fim

reiniciar o modelo, transformando-o assim em um modelo cíclico para que seja realizada a verificação da propriedade *soundness*, no contexto de vídeo games, considerando as propriedades clássicas das redes de Petri (vivacidade e limitabilidade).

A funcionalidade da análise *state space* disponível no CPN Tools, calcula o grafo das marcações alcançáveis automaticamente e gera um relatório contendo as informações da análise. A figura 22 apresenta a primeira parte do relatório gerado após a análise do modelo da figura 21.

```
Statistics
------------------------------------

  State Space
     Nodes:  59
     Arcs:   149
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 22 – Estatísticas depois de aplicar a análise do *state space*.

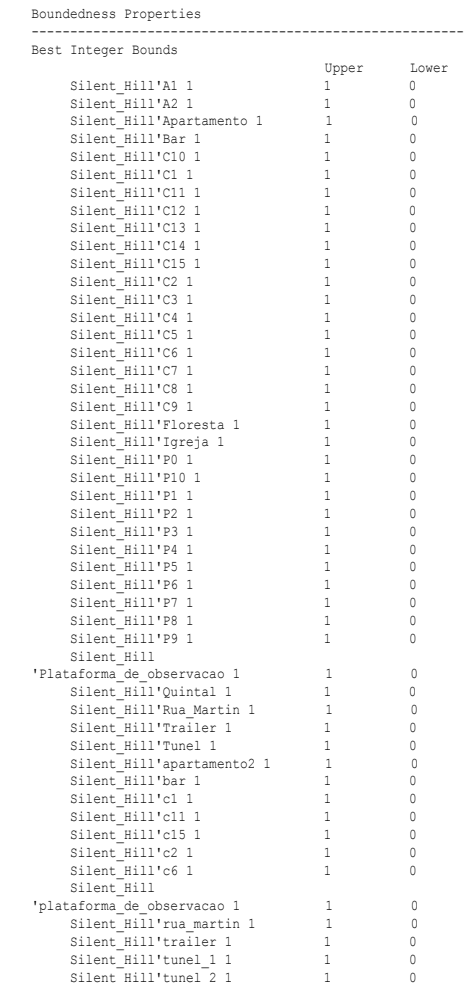
A figura 22 trás as informações estatísticas obtidas após aplicar a análise *state space*. Em particular, SCC-graph (*Strongly Connected Components* - Componentes fortemente conexas) indica que existe um componente fortemente conexo no grafo de alcançabilidade obtido após a execução da análise.

A segunda parte do relatório produzido pelo CPN Tools contém informações sobre a propriedade limitabilidade (*boundedness*) da rede. A figura 23 indica que o número máximo de fichas que um lugar no modelo estendido pode ter é um. Em outras palavras, a análise mostrou que não há duplicação de fichas no modelo global. De acordo com a definição da propriedade limitabilidade, definida no capítulo 3, o modelo global analisado é 1-limitado.

A última parte do relatório estatístico gerado pelo CPN Tools, indica as instâncias da propriedade vivacidade (*liveness*) da rede. A figura 24 mostra que todas as transições do modelo da figura 21 são vivas.

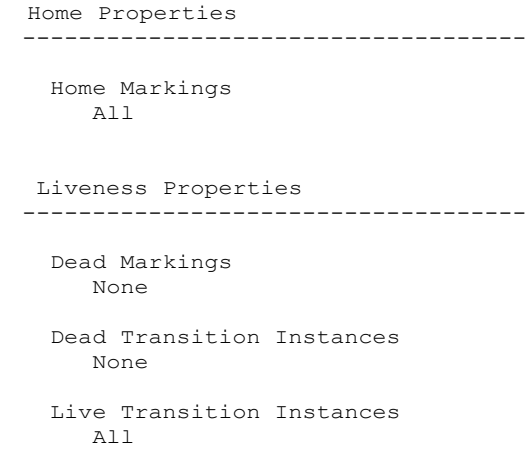
Como apresentado nas figuras 23 e 24, o modelo estendido da figura 21 é dito vivo e 1-limitado. Portanto, o modelo não estendido (apresentado na figura 20) é *sound*). Se baseando no ponto de vista do jogo, o modelo ser *sound* significa que todas as atividades do nível podem ser executadas e todas as áreas do mapa topológico podem ser eventualmente acessadas pelo jogador. Isso garante a consistência das atividades do nível de jogo, bem como a boa construção do mapa topológico.

A abordagem apresentada neste capítulo também identifica modelos que possuem inconsistências. Por exemplo, no primeiro nível de Silent Hill II o jogador precisa encontrar uma chave para ter acesso a área *Apartamento*. Essa chave pode ser encontrada em um



```
Boundedness Properties
----------------------------------------------------------
Best Integer Bounds
                                        Upper      Lower
     Silent_Hill'A1 1                   1          0
     Silent_Hill'A2 1                   1          0
     Silent_Hill'Apartamento 1          1          0
     Silent_Hill'Bar 1                  1          0
     Silent_Hill'C10 1                  1          0
     Silent_Hill'C1 1                   1          0
     Silent_Hill'C11 1                  1          0
     Silent_Hill'C12 1                  1          0
     Silent_Hill'C13 1                  1          0
     Silent_Hill'C14 1                  1          0
     Silent_Hill'C15 1                  1          0
     Silent_Hill'C2 1                   1          0
     Silent_Hill'C3 1                   1          0
     Silent_Hill'C4 1                   1          0
     Silent_Hill'C5 1                   1          0
     Silent_Hill'C6 1                   1          0
     Silent_Hill'C7 1                   1          0
     Silent_Hill'C8 1                   1          0
     Silent_Hill'C9 1                   1          0
     Silent_Hill'Floresta 1             1          0
     Silent_Hill'Igreja 1               1          0
     Silent_Hill'P0 1                   1          0
     Silent_Hill'P10 1                  1          0
     Silent_Hill'P1 1                   1          0
     Silent_Hill'P2 1                   1          0
     Silent_Hill'P3 1                   1          0
     Silent_Hill'P4 1                   1          0
     Silent_Hill'P5 1                   1          0
     Silent_Hill'P6 1                   1          0
     Silent_Hill'P7 1                   1          0
     Silent_Hill'P8 1                   1          0
     Silent_Hill'P9 1                   1          0
     Silent_Hill
'Plataforma_de_observacao 1             1          0
     Silent_Hill'Quintal 1              1          0
     Silent_Hill'Rua_Martin 1           1          0
     Silent_Hill'Trailer 1              1          0
     Silent_Hill'Tunel 1                1          0
     Silent_Hill'apartamento2 1         1          0
     Silent_Hill'bar 1                  1          0
     Silent_Hill'c1 1                   1          0
     Silent_Hill'c11 1                  1          0
     Silent_Hill'c15 1                  1          0
     Silent_Hill'c2 1                   1          0
     Silent_Hill'c6 1                   1          0
     Silent_Hill
'plataforma_de_observacao 1             1          0
     Silent_Hill'rua_martin 1           1          0
     Silent_Hill'trailer 1              1          0
     Silent_Hill'tunel_1 1              1          0
     Silent Hill'tunel 2 1              1          0
```

Figura 23 – Resultados da análise da propriedade limitabilidade.



```
Home Properties
-------------------------------------

  Home Markings
     All


 Liveness Properties
-------------------------------------

  Dead Markings
     None

  Dead Transition Instances
     None

  Live Transition Instances
     All
```

Figura 24 – Resultados da análise da propriedade vivacidade.

cadáver que está localizado na área *Rua Martin*. Uma vez com a chave o jogador poderá acessar a área *Apartamento*.

Se em vez de encontrar a chave no cadáver na área *Rua Martin* o projetista colocasse

a chave na área *Apartamento*, seria impossível o jogador executar a atividade *Encontrar chave* e, consequentemente, não poderia concluir o primeiro nível do jogo. Para ilustrar essa situação, o modelo de atividades do primeiro nível de Silent Hill II foi modificado. A transição *Encontrar chave* tem agora os lugares *P8* e *Apartamento1* como lugares de entrada. Isso significa que para executar a atividade *Econtrar chave* é preciso que o jogador esteja localizado na área *Apartamento*. A figura 25 apresenta a parte do modelo de atividades que foi modificada. De fato, essa modificação no modelo de atividades do primeiro nível de Silent Hill II gera inconsistência no cenário do jogo. Para entrar no apartamento o jogador deve primeiro encontrar a chave para destrancar a porta que dá acesso a esta área, mas como a chave se encontra dentro do apartamento o jogador nunca poderá pegar a chave e, consequentemente, nunca poderá acessar a área *Apartamento*.

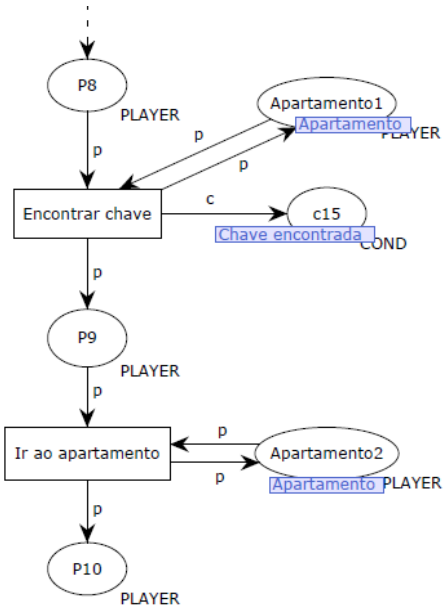


Figura 25 – Modelo do nível modificado.

Uma vez que o modelo de atividades sofreu alteração, o modelo global modificado (similar ao modelo apresentado na figura 21) pode ser submetido a análise. As informações da propriedade vivacidade (figura 26) indicam que as transições *Encontrar chave* e *Ir ao apartamento* são agora transições mortas, ou seja, elas não podem ser sensibilizadas. A partir deste resultado é possível deduzir que a chave que o jogador precisa encontrar deve ser localizada em outra área do mapa para que o nível possa ser concluído.

4.6 Considerações finais

Este capítulo apresentou uma abordagem baseada em redes de Petri para o processo de criação de vídeo games. Esta abordagem envolve os cenários de um nível de jogo, que correspondem as diversas atividades que o jogador precisa executar, bem como a noção de mapa topológico, onde são representadas as diversas áreas do mundo virtual nas quais

```
Liveness Properties
------------------------------------

 Dead Markings
     None

 Dead Transition Instances
     Silent_Hill'Ir_ao_apartamento 1
     Silent_Hill'Encontrar_chave 1
     Silent_Hill'T1 1
     Silent_Hill'T2 1
     Silent_Hill'T3 1
     Silent_Hill'b_w 1
     Silent_Hill'by_w 1
     Silent_Hill'c_w 1
     Silent_Hill'm_w 1
     Silent_Hill'tr_w 1
```

Figura 26 – Resultados da análise da propriedade vivacidade após a modificação do modelo do nível.

o jogador efetuará as atividades previstas. O primeiro nível do jogo Silent Hill II foi utilizado para ilustrar a abordagem proposta.

Para a representação das atividades foi utilizado uma rede de Petri do tipo WorkFlow net. Já para representar o mapa topológico do jogo, foi utilizado um tipo de rede de Petri denominado grafo de estado. Um mecanismo de comunicação foi então proposto para representar as influências que um modelo tem sobre o outro. As redes de Petri Coloridas foram utilizadas para especificar tal mecanismo de comunicação. Em particular, o conceito de *fusion places*, proposto pela ferramenta CPN Tools, foi usado para manter os dois modelos visualmente distintos.

Para verificar a corretude do modelo, um algoritmo baseado na análise do espaço de estado é automaticamente implementado por algumas das funcionalidades existentes no CPN Tools. O principal propósito da análise é verificar se todas as atividades do modelo de nível podem ser executadas e se todas as áreas do mundo virtual podem ser acessadas pelo jogador. O uso de uma uma ferramenta como o CPN Tools permitiu a implementação prática da abordagem proposta, tornando possível a modelagem e a simulação dos modelos criados.

Sendo assim, este capítulo apresentou uma abordagem para a modelagem e verificação formal de vídeo games.

Capítulo 5

Estudo de Caso

Este capítulo tem por objetivo apresentar a abordagem proposta no capítulo 4 por meio de um estudo de caso aplicado ao jogo Dream:scape (SPEEDBUMP, 2012).

5.1 Dream:scape

Dream:scape é um jogo baseado em uma narrativa cheia de mistérios e enigmas. O jogador controla o personagem Wilson, um paciente em coma à beira da morte que é dada a oportunidade de explorar suas memórias e descobrir a verdade sobre um mistério que o perseguiu por toda a vida. Para ajudar a recuperar os fragmentos da memória de Wilson, o jogador recebe um diário que contém um esquema do mapa do ambiente e um espaço para anotações sobre cada lugar. Essas anotações são adquiridas no decorrer do jogo, de acordo com a execução das tarefas. O jogo Dream:scape, segue uma narrativa linear que leva o jogador a explorar o ambiente, coletar itens e executar uma série de atividades para resolver o enigma da história.

Para mostrar a abordagem proposta neste trabalho, o Dream:scape foi dividido em cinco níveis. Cada nível possui uma sequência diferente de atividades que deve ser executada para alcançar um determinado objetivo. Já a representação do mapa topológico é comum para todos os níveis. No entanto, as áreas do mapa são liberadas de acordo com a execução das atividades. Assim, no primeiro nível do jogo, poucas áreas podem ser acessadas pelo jogador. Já no último nível, o jogador poderá acessar todas as áreas, uma vez que já tenha executado todas as atividades e, consequentemente, satisfeito todas as condições para liberação das áreas. A ideia básica do Dream:scape é coletar várias informações para desvendar o mistério que envolve a história. Para isso, é preciso explorar as regiões do mundo virtual. O jogador deve visitar então certos lugares e coletar informações diferentes a cada nível do jogo.

5.2 Representação do Mapa Topológico

O ambiente do jogo Dream:scape é o subconsciente do personagem Wilson. O diário que o jogador tem acesso trás o mapa que contém as áreas deste ambiente. São nessas áreas que as atividades do jogo devem ser executadas. A figura 27 ilustra o mapa que é apresentado ao jogador.

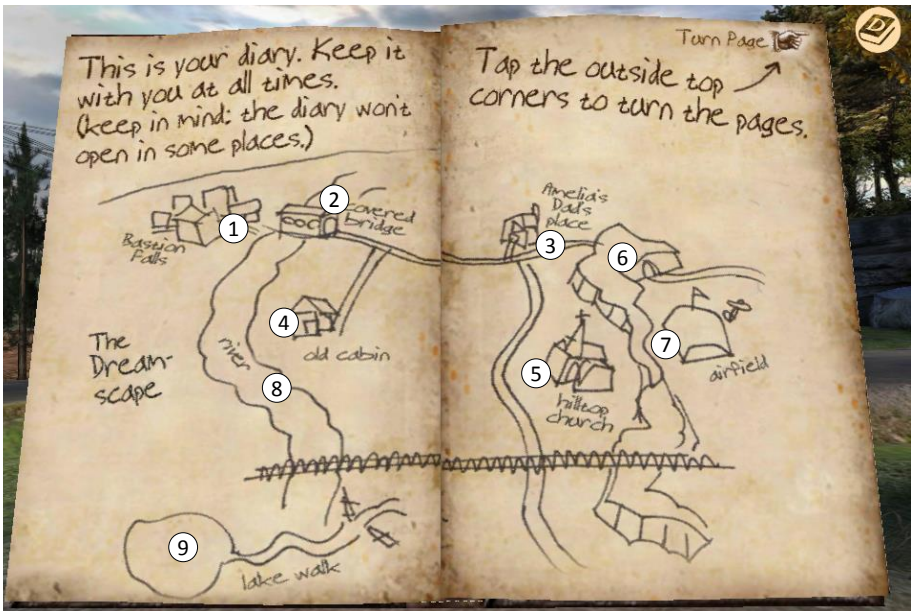


Figura 27 – Mapa topológico do jogo Dream:scape.

Os números circulados na figura 27 representam as regiões do mapa onde as atividades são executadas:

1. Bastion Falls.
2. Ponte coberta.
3. Casa do pai da Amélia.
4. Cabana.
5. Igreja.
6. Túnel.
7. Aeródromo.
8. Rio.
9. Lago

O mapa da figura 27 possui pouca funcionalidade. Por exemplo, ele não mostra onde o jogador está e ainda oculta algumas áreas do ambiente onde o jogador deverá executar

tarefas específicas do jogo. Uma das áreas que é oculta pelo mapa é a área denominada *Celeiro*. Outra área importante que não é apresentada no mapa é o *Demento scape*. É nesta área que o jogador executará a última atividade do jogo.

A figura 28 ilustra a representação do mapa topológico do Dream:scape por meio de um grafo de estado. Na figura 28, todas as áreas do jogo e as condições associadas a liberação das áreas são apresentadas.

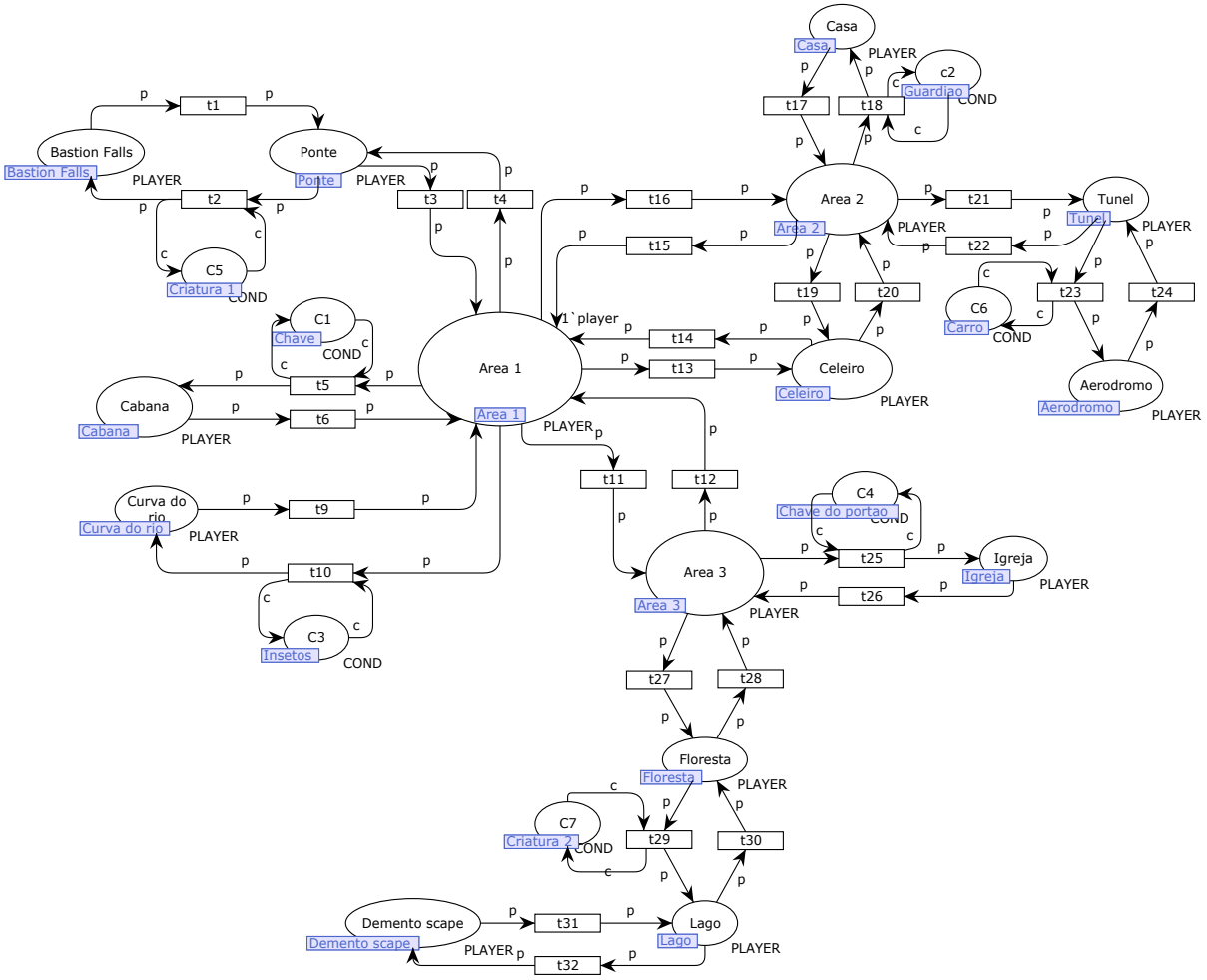


Figura 28 – Representação do mapa topológico do jogo Dream:scape por meio de um grafo de estado com condições de ativação.

As áreas de nomes *Area 1*, *Area 2* e *Area 3* são regiões comuns entre outras áreas. De modo geral, essas regiões são espaços que servem apenas como passagem de uma área para outra. Por exemplo, a *Area 1* faz fronteira com as áreas *Ponte*, *Cabana*, *Curva do rio*, *Celeiro*, *Area 2* e *Area 3*.

No início do jogo, o jogador se encontra na área de nome *Area 1*. Assim, a ficha no lugar *Area 1* representa a localização do jogador no começo do jogo. Inicialmente, o jogador não poderá ter acesso a todas as áreas do mapa. As áreas são liberadas de acordo com a evolução do jogo. No modelo topológico da figura 28, todas as condições de liberação de área são representadas e marcadas com as *fusion tags*. Uma *fusion tag*

em um lugar específico, indica qual a condição associada a este lugar. Ao todo, são sete condições:

1. Encontrar a chave da cabana (*fusion tag* Chave).
2. Espantar o guardião da casa (*fusion tag* Guardião).
3. Espantar os insetos (*fusion tag* Insetos).
4. Encontrar a chave do portão da Igreja (*fusion tag* Chave do portão).
5. Espantar a criatura da ponte (*fusion tag* Criatura 1).
6. Remover o carro do túnel (*fusion tag* Carro).
7. Derrotar a criatura da floresta (*fusion tag* Criatura 2).

Cada condição está associada a uma transição específica do grafo de estado, que representa a liberação da passagem para uma área específica do mapa topológico. Por exemplo, a condição 1 está associada à liberação da passagem para a área *Cabana*, e a condição 7 está associada à liberação da passagem para a área *Lago*.

Além dos lugares de condição, todos os lugares da rede (que representam as regiões do mapa) estão marcados com as *fusion tags*, que são utilizadas para a implementação do mecanismo de comunicação entre o modelo do mapa topológico e os modelos dos níveis.

5.3 Representação dos Níveis

O jogo Dream:scape foi dividido em cinco partes, ou seja, cinco níveis diferentes. Cada nível possui uma sequência de atividades que o jogador precisa executar para alcançar o objetivo final do jogo. O fluxo de atividades de cada nível é representado por WorkFlow nets. Dessa forma, cada nível possui uma WorkFlow net que o representa. Além disso, cada nível possui um modelo global que é composto por uma WorkFlow net (representação das atividades), por um grafo de estado (representação do mapa topológico) e pelo mecanismo de comunicação entre os dois modelos.

Assim como o grafo de estado do mapa topológico, apresentado na seção anterior, as WorkFlow nets que representam as atividades foram adaptadas para a ferramenta CPN Tools. O modelo global de cada nível, também é apresentado por meio do CPN Tools.

5.3.1 Nível 1

As atividades que correspondem ao nível 1 são:

- ❑ Pegar pista 1.

- ❏ Pegar chave.
- ❏ Recuperar memórias 1.
- ❏ Pegar pista 2.
- ❏ Pegar martelo.
- ❏ Consertar escadas.
- ❏ Pegar sanduíches e recuperar memórias 2.

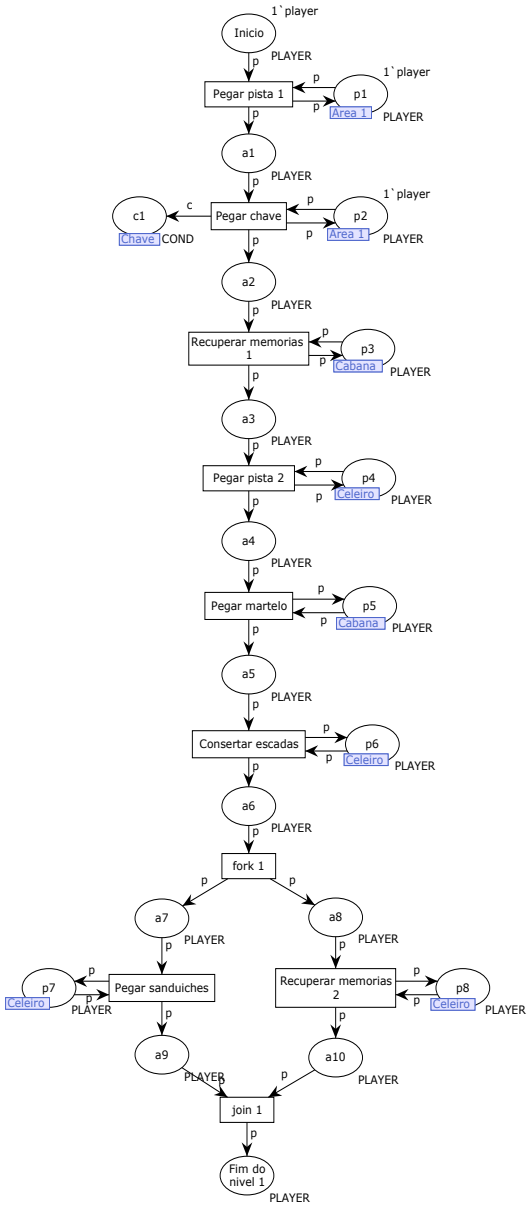


Figura 29 – Modelo lógico do nível 1.

Algumas atividades do jogo se repetem várias vezes, como por exemplo, as atividades *Pegar Pista* e *Recuperar memórias*. Sempre que o jogador precisa pegar algum item para

ter acesso a uma área específica, ele precisa encontrar a pista que leva a esse item. A tarefa *Recuperar memórias* repete toda vez que o jogador visita algum lugar importante que o faz lembrar de memórias do passado. Essas memórias são adquiridas e preenchem o espaço destinado as anotações dos lugares no diário que o jogador recebe no começo do jogo. Apesar de possuírem os mesmo nomes, essas atividades não trazem as mesmas informações para o jogador e não são executadas sempre nos mesmos lugares. Para diferenciá-las, as atividades são enumeradas de acordo com a ordem em que elas ocorrem no jogo. *Pegar pista 1*, *Recuperar memórias 1* e *Recuperar memórias 2* são exemplos de atividades que se repetem.

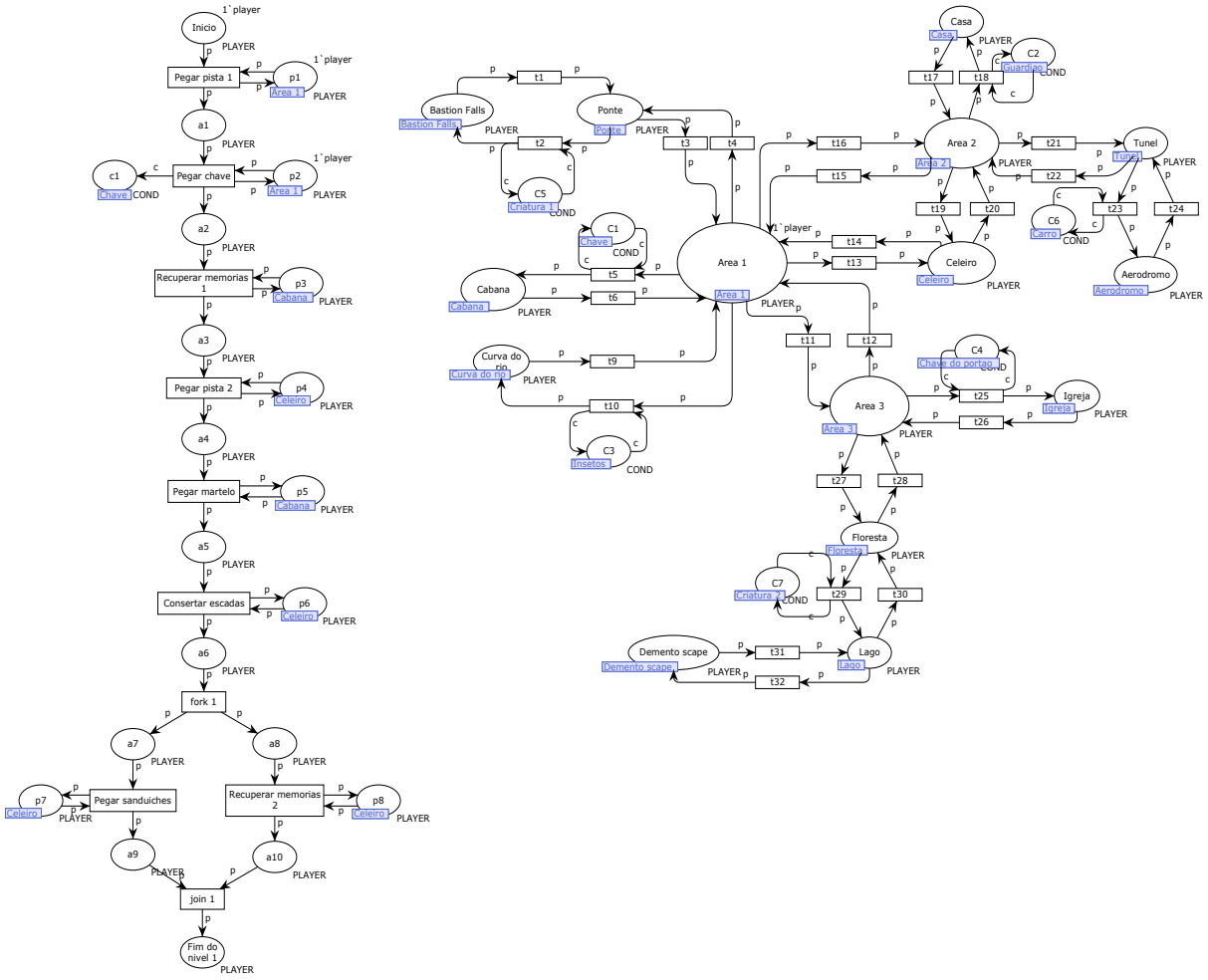


Figura 30 – Modelo global do nível 1.

A WorkFlow net que representa as atividades relacionadas ao nível 1 do jogo Dream:scape é apresentada na figura 29. Neste modelo, as atividades *Pegar pista 1*, *Pegar chave*, *Recuperar memorias 1*, *Pegar pista 2*, *Pegar martelo* e *Consertar escadas* são tarefas que pertencem a uma rotina sequencial. A atividade *Pegar chave* satisfaz a condição 1, que corresponde a liberação da área *Cabana* no modelo topológico da figura 28. As atividades *Pegar sanduiches* e *Recuperar memorias 2* correspondem a uma rotina paralela e também são as últimas atividades do nível 1. Os lugares *Inicio* e *Fim do nivel 1* representam o

começo do jogo e o final do nível 1, respectivamente.

O modelo global de um nível é composto pelo modelo de atividades do nível, modelo do mapa topológico e pelo mecanismo de comunicação que é responsável pela comunicação entre os dois modelos. O modelo global do nível 1 pode ser visto na figura 30. O lado esquerdo da figura 30 representa o modelo lógico do nível 1. Já o lado direito da figura 30 ilustra o grafo de estado que representa o mapa topológico do jogo. No início do jogo, o jogador está localizado na área 1 e não poderá acessar de imediato todas as áreas do mapa. Dessa forma, uma ficha no lugar *Area 1*, representa a localização atual do jogador.

5.3.2 Nível 2

Após terminar o nível 1, o jogador poderá passar para o nível 2 e executar as seguintes atividades:

- ❏ Espantar guardião.
- ❏ Recuperar memórias 3.
- ❏ Pegar pista 3.
- ❏ Pegar inseticida.
- ❏ Espantar insetos.
- ❏ Recuperar memórias 4.

O modelo lógico do nível 2 é ilustrado através da figura 31. O lugar *Nivel 2* representa o começo do nível 2. Neste nível, todas as atividades pertencem a uma rotina sequencial. Em particular, as atividades *Espantar guardiao* e *Espantar insetos* satisfazem as condições 2 e 3, respectivamente. Os lugares da rede da figura 31 que representam essas condições são os lugares *c2* e *c3*. Essas condições liberam as áreas *Casa* e *Curva do rio*. O lugar *Fim do nivel 2* representa o final do nível.

O modelo global do nível 2 é ilustrado na figura 32. O lado direito da figura 32 representa o modelo do mapa topológico. Já o lado esquerdo da figura ilustra o modelo de atividades do nível 2. No início do nível 2, o jogador está localizado na região de nome *Area 2*. Portanto, uma ficha no lugar *Area 2* do modelo do mapa topológico (lado direito da figura), representa a localização do jogador no começo do nível. O lugar de condição *C1* do modelo topológico, possui uma ficha. Isso indica que a condição associada ao lugar *C1* já foi satisfeita pelo jogador no nível anterior.

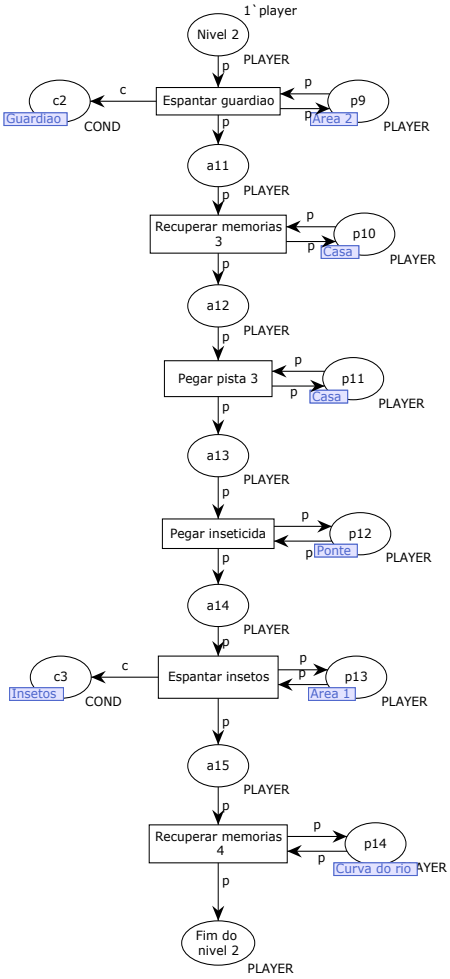


Figura 31 – Modelo lógico do nível 2.

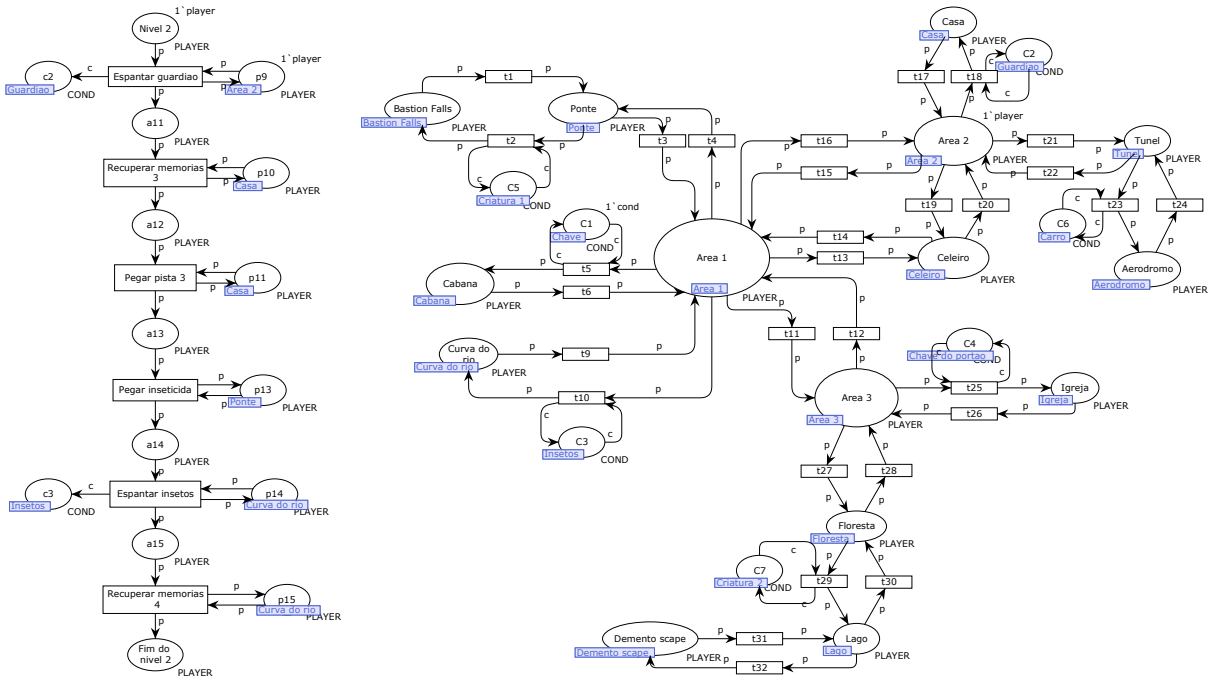


Figura 32 – Modelo global do nível 2.

5.3.3 Nível 3

O nível 3 corresponde a terceira parte do jogo. Neste nível, o jogador já terá executado todas as atividades dos níveis 1 e 2. Para completar o nível 3, o jogador deverá executar as seguintes atividades:

- ❏ Pegar pista 4.
- ❏ Pegar chave do portão.
- ❏ Pegar arma e recuperar memórias 5.
- ❏ Espantar criatura.
- ❏ Recuperar memórias 6.

A figura 33 ilustra a WorkFlow net que representa o modelo lógico do nível 3. Os lugares *Nivel 3* e *Fim do nivel 3* equivalem ao começo e ao fim do nível 3, respectivamente. As atividades *Pegar arma* e *Recuperar memorias 5* fazem parte de uma rotina paralela, ao passo que as demais atividades pertencem a uma rotina sequencial. Neste modelo, duas atividades correspondem à liberação de áreas do mapa topológico: *Pegar chave do portao* e *Espantar criatura*. A execução da atividade *Pegar chave do portao* satisfaz a condição 4, que corresponde à liberação da área *Igreja*. Já a atividade *Espantar criatura* satisfaz a condição 5, e corresponde à liberação da passagem para a área *Bastion Falls*.

O modelo global do nível 3 é representado na figura 34. O lado esquerdo da figura 34 ilustra o modelo de atividade no nível 3, enquanto que o lado direito ilustra o modelo do mapa topológico. A ficha no lugar *Area 3* do mapa topológico representa a localização do jogador no começo do nível 3. As fichas nos lugares *C1*, *C2* e *C3* indicam que as condições associadas a esses lugares já foram satisfeitas nos níveis anteriores.

5.3.4 Nível 4

O nível 4 é o penúltimo nível do jogo. Nesta fase, o jogador consegue descobrir um pouco mais sobre o mistério que precisa ser resolvido. Para isso, as seguintes atividades devem ser executadas:

- ❏ Pegar pista 5.
- ❏ Pegar gasolina.
- ❏ Remover carro.
- ❏ Recuperar memórias 7.
- ❏ Recuperar memórias 8.

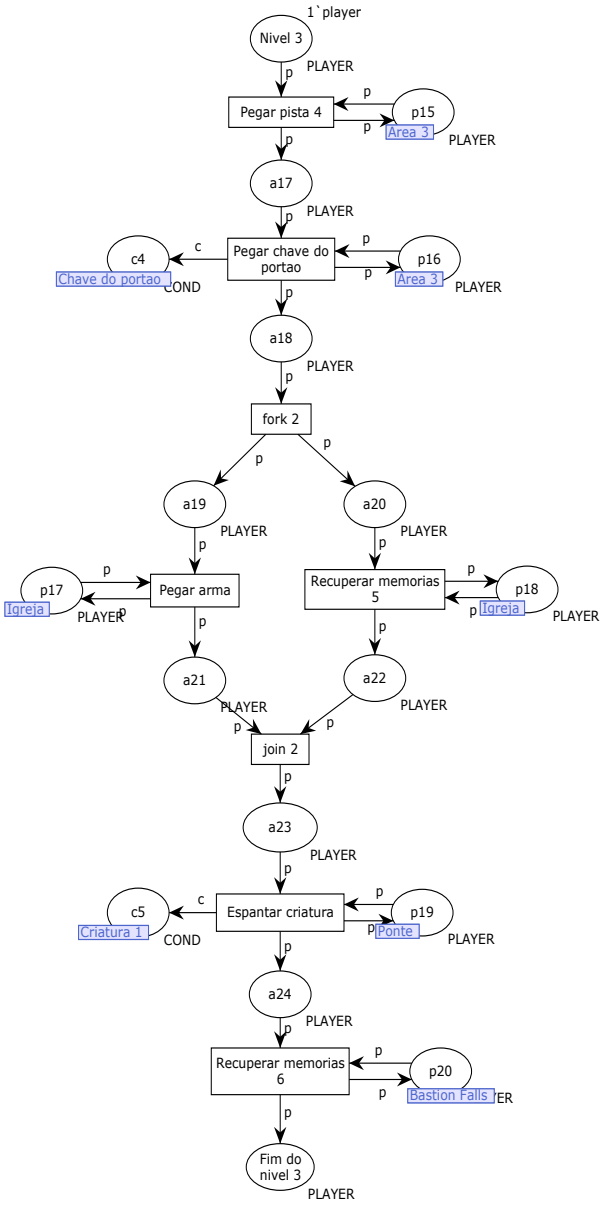


Figura 33 – Modelo lógico do nível 3.

Todas as atividades deste nível pertencem a uma rotina sequencial. Em particular, a atividade *Remover carro* satisfaz a condição 6 que libera a área *Aerodromo* do mapa topológico. A partir deste ponto, o jogador começa a descobrir ainda mais sobre o mistério que envolve a história do personagem. A WorkFlow net que representa o nível 4 é ilustrada na figura 35.

A figura 36 ilustra o modelo global do nível 4. O lado esquerdo da figura 36 ilustra o modelo do nível 4 e o lado direito ilustra o mapa topológico. No início do nível 4, o jogador está localizado na região do mapa topológico denominada *Tunel*. Assim, uma ficha no lugar *Tunel* representa a localização do jogador no começo do nível 4. Os lugares condição *C1*, *C2*, *C3*, *C4* e *C5* possuem uma ficha cada, o que indica que as condições associadas a esses lugares já foram satisfeitas pelo jogador nos níveis anteriores.

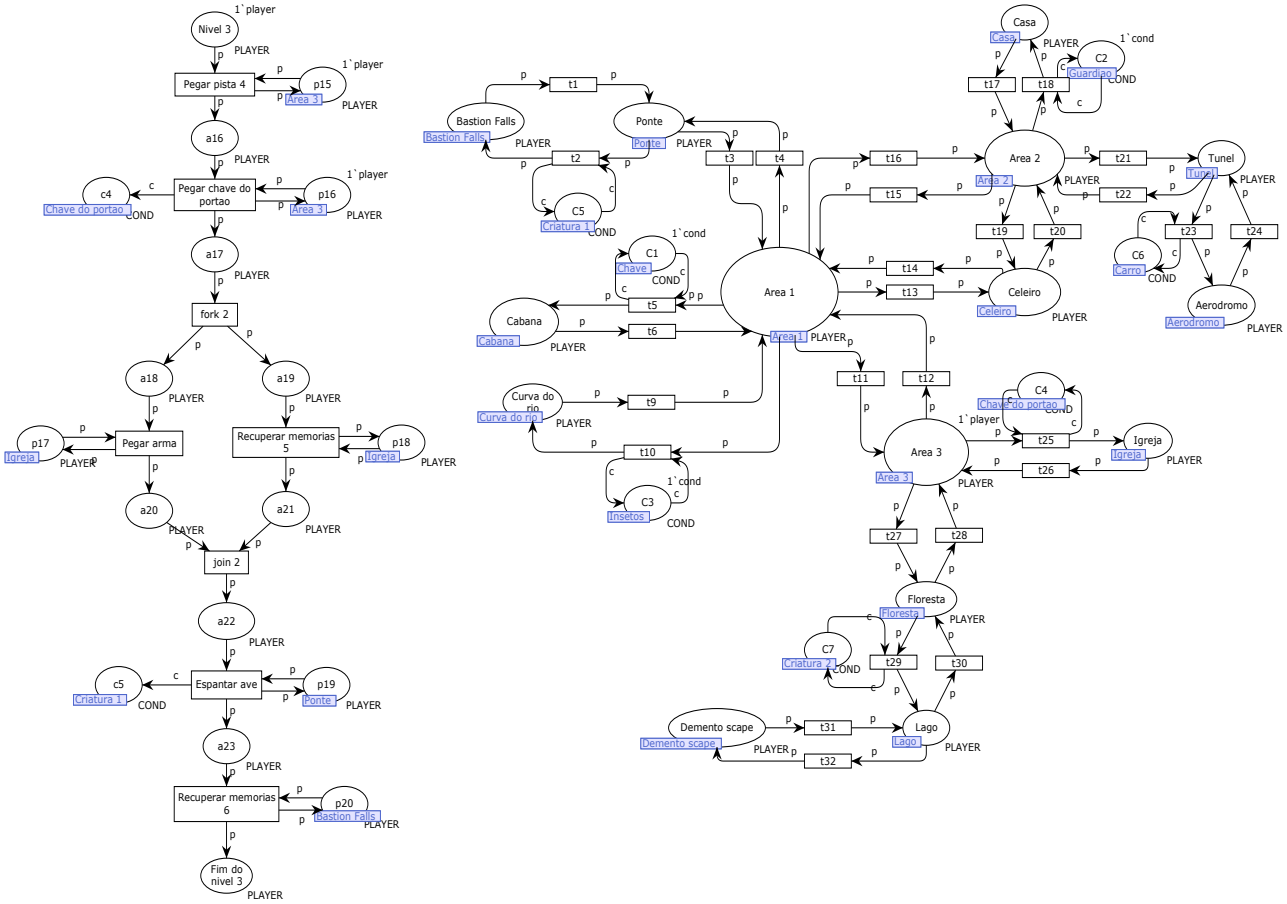


Figura 34 – Modelo global do nível 3.

5.3.5 Nível 5

A última parte do jogo Dream:scape é representada pelo nível 5. Neste nível, o jogador já terá acesso a quase todas as áreas do jogo e terá realizado todas as atividades dos níveis anteriores. Para concluir o jogo é preciso que o jogador execute as seguintes atividades do nível 5:

- ❏ Pegar pista 6.
- ❏ Pegar pista 7.
- ❏ Derrotar criatura.
- ❏ Entrar no lago.
- ❏ Abrir a caixa.

O modelo lógico que representa o nível 5 pode ser visto na figura 37. O lugar *Nivel 5* representa o começo do nível 5 e o lugar *Fim do jogo* representa o fim do nível 5 e, consequentemente, o fim do jogo. Todas as atividades do nível 5 pertencem a uma rotina sequencial. Em particular, a execução da atividade *Derrotar criatura* é a condição

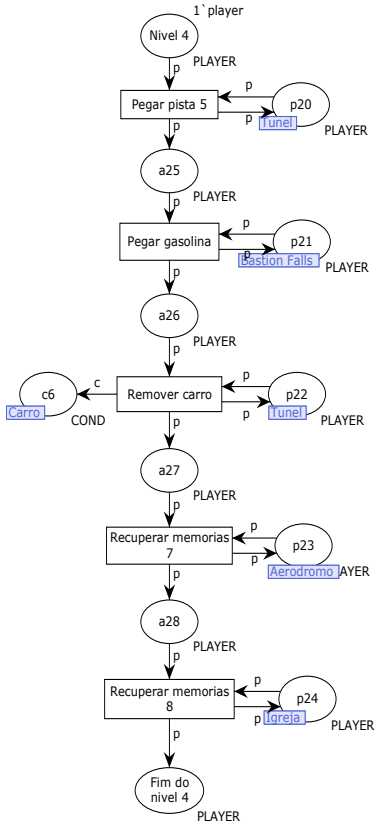


Figura 35 – Modelo lógico do nível 4.

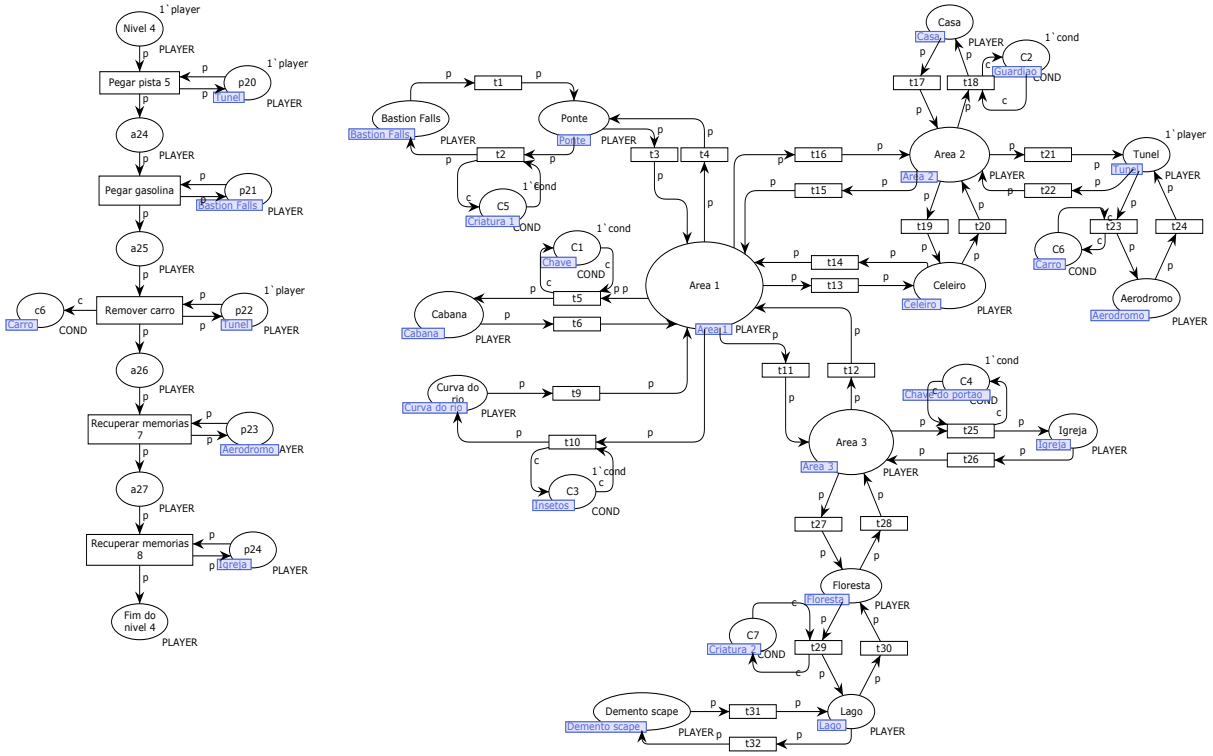


Figura 36 – Modelo global do nível 4.

necessária para liberar a passagem para a área *Lago*. A última atividade do jogo que

o jogador deve executar é a atividade *Abrir caixa*. Após a execução dessa atividade, o jogador conclui o mistério da história e termina o jogo.

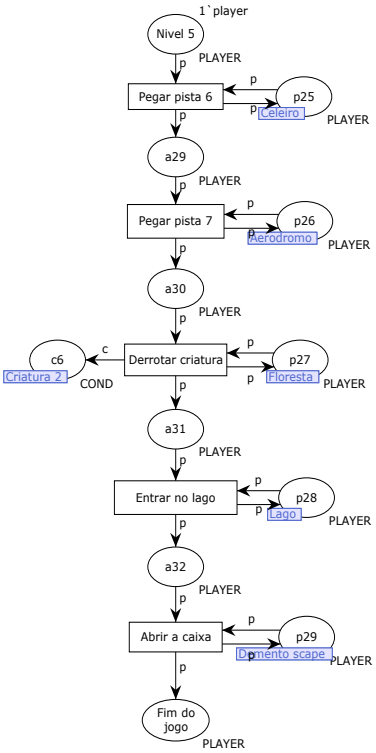


Figura 37 – Modelo lógico do nível 5.

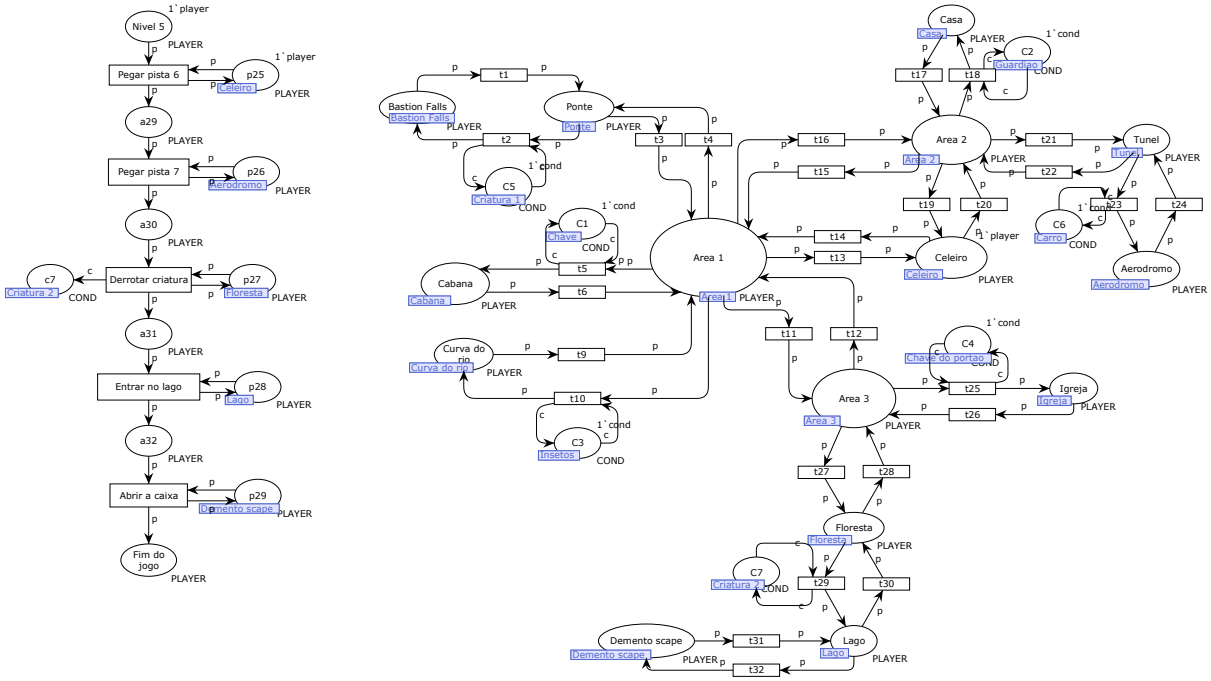


Figura 38 – Modelo global do nível 5.

No começo do nível 5, o jogador está localizado no lugar *Celeiro*. Portanto, o lugar *Celeiro*, do mapa topológico, possui uma ficha. No último nível do jogo, as condições 1,

2, 3, 4, 5 e 6 já foram satisfeitas. Assim, os lugares *C1*, *C2*, *C3*, *C4*, *C5* e *C6* possuem uma ficha, cada um. A condição número 7, representada pelo lugar de nome *C7* no mapa topológico, é satisfeita no nível 5. Ao fim do nível, o jogador deve estar localizado no lugar *Demento scape*, para realizar a última tarefa do jogo.

5.4 Análise

Para analisar o modelo global de cada nível será utilizada a abordagem do algoritmo de verificação da propriedade *soundness*, apresentada nos capítulos 3 e 4 deste trabalho.

A ideia é transformar cada modelo global do nível em um modelo cíclico aumentado, para poder analisar a propriedade *soundness* no contexto de jogos, considerando assim as propriedades clássicas das redes de Petri. Para tanto, em cada modelo global do nível, um lugar de início comum *A1* e um lugar final *A2* são criados. Uma transição *T1* é então criada para distribuir uma ficha no lugar que representa o início do nível, e outra ficha em algum lugar específico do grafo de estado para representar a localização do jogador no início do nível. No caso de níveis que já começam com condições satisfeitas (nível 2 em diante), o lugar correspondente a essa condição também receberá uma ficha. Uma outra transição *T2* é criada com o propósito de consumir as fichas do modelo ao final do nível. Essas fichas devem estar presentes no lugar final do modelo lógico, no último lugar de área onde o jogador deve estar ao terminar o nível, e em todos os lugares de condição marcados que foram utilizados para liberar as passagens entre duas áreas diferentes do mapa. Uma transição denominada *T3*, irá então reiniciar a rede.

5.4.1 Análise do nível 1

A figura 39 apresenta o modelo global do primeiro nível do Dream:scape modificado para a análise. O modelo da figura 39 é um modelo cíclico e equivale ao modelo acíclico apresentado na figura 30.

O lado esquerdo da figura 39 corresponde ao modelo de atividades do nível 1. Já o lado direito da figura corresponde ao modelo do mapa topológico. Na figura 39, a transição *T1* distribui uma ficha no lugar *Inicio*, que representa o início do jogo e também do nível 1, e outra ficha no lugar *Area 1*, que representa a localização do jogador no começo do jogo. A transição *T2* tem o propósito de consumir todas as fichas do modelo, que devem se encontrar nos lugares *Fim do nivel 1*, *Celeiro* e nos lugares de condição marcados com a *fusion tag Chave*. O disparo da transição *T3* irá então reiniciar o modelo modificado (figura 39).

Para a verificação da propriedade *soundness* no modelo global do nível 1, foi aplicada a funcionalidade de análise do *state space* disponível no CPN Tools. A figura 40 apresenta as informações relativas aos dados de análise produzidos pela enumeração do conjunto

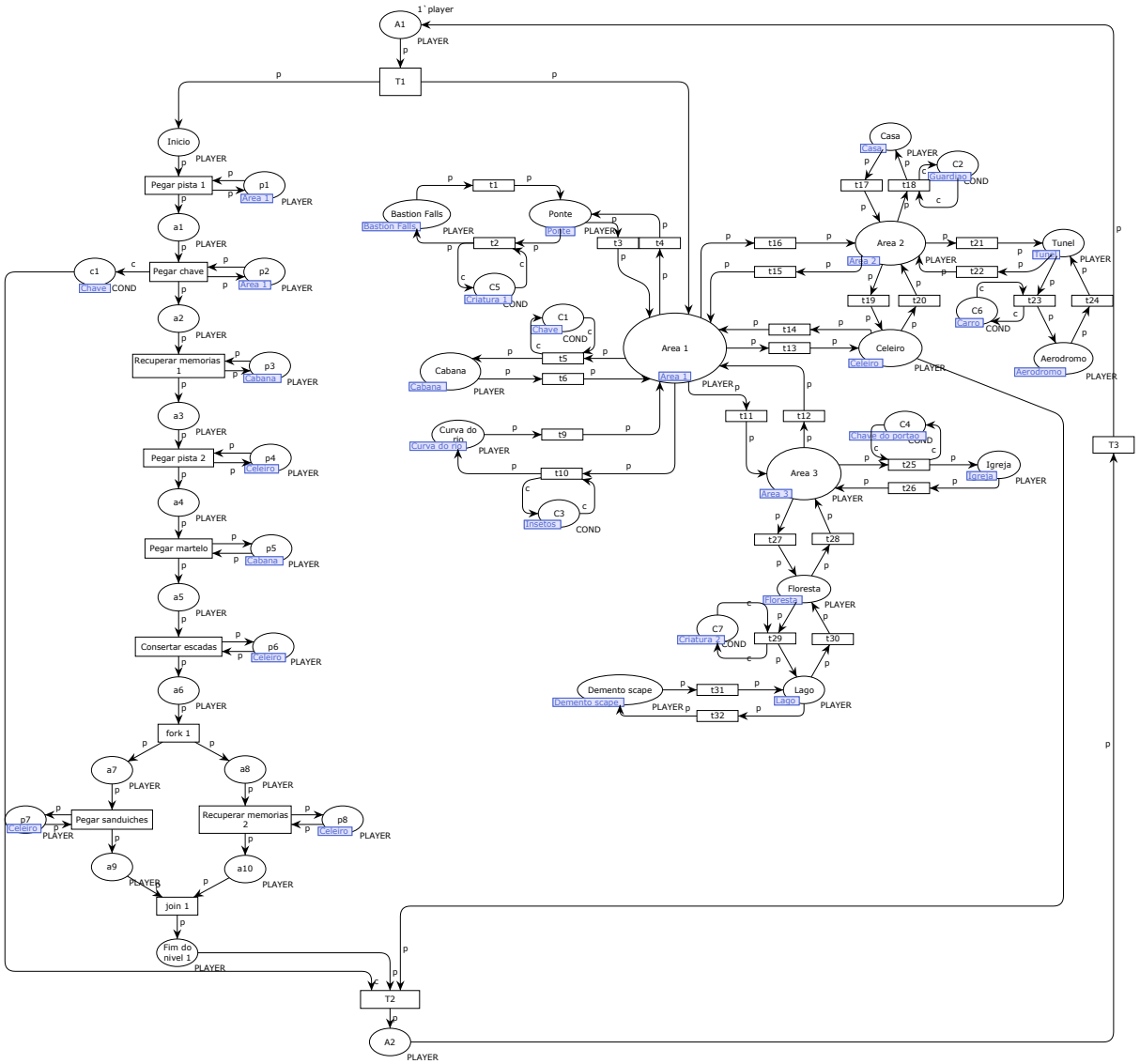


Figura 39 – Modelo global do nível 1 modificado para análise.

dos estados alcançáveis após aplicar essa funcionalidade. O SCC Graph indica que existe apenas um componente fortemente conexo no grafo de alcançabilidade obtido após a aplicação da análise *state space*.

```
Statistics
------------------------

  State Space
     Nodes:  96
     Arcs:   217
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 40 – Dados de análise produzidos após aplicar a análise do *state space* para o nível 1.

A figura 41 apresenta a segunda parte do relatório de análise gerado pelo CPN Tools e contém as informações sobre a propriedade *boundedness* da rede. Os lugares destacados em vermelho na figura 41 indicam o número máximo de fichas que esses lugares podem ter, que no caso é um. Em outras palavras, não houve duplicação de ficha. No entanto, há lugares em que este número é zero, como por exemplo o lugar *Nivel1'Aerodromo*. Os lugares que não possuem histórico de ficha correspondem a áreas do mapa topológico e a lugares de condição. De fato, no primeiro nível do jogo, algumas áreas do mapa estão bloqueadas e consequentemente o jogador não poderá acessá-las. Essas áreas serão liberadas nos próximos níveis do jogo, de acordo com a execução das atividades.

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Nivel1'A1	**1**	**0**
Nivel1'A2	**1**	**0**
Nivel1'Aerodromo	0	0
Nivel1'Area_1	**1**	**0**
Nivel1'Area_2	**1**	**0**
Nivel1'Area_3	**1**	**0**
Nivel1'Bastion_Falls	0	0
Nivel1'C1	**1**	**0**
Nivel1'C2	0	0
Nivel1'C3	0	0
Nivel1'C4	0	0
Nivel1'C5	0	0
Nivel1'C6	0	0
Nivel1'C7	0	0
Nivel1'Cabana	**1**	**0**
Nivel1'Casa	0	0
Nivel1'Celeiro	**1**	**0**
Nivel1'Curva_do_rio	0	0
Nivel1'Demento_scape	0	0
Nivel1'Fim_do_nivel_1	**1**	**0**
Nivel1'Floresta	**1**	**0**
Nivel1'Igreja	0	0
Nivel1'Inicio	**1**	**0**
Nivel1'Lago	0	0
Nivel1'Ponte	**1**	**0**
Nivel1'Tunel	**1**	**0**
Nivel1'a10	**1**	**0**
Nivel1'a1	**1**	**0**
Nivel1'a2	**1**	**0**
Nivel1'a3	**1**	**0**
Nivel1'a4	**1**	**0**
Nivel1'a5	**1**	**0**
Nivel1'a6	**1**	**0**
Nivel1'a7	**1**	**0**
Nivel1'a8	**1**	**0**
Nivel1'a9	**1**	**0**
Nivel1'c1	**1**	**0**
Nivel1'p1	**1**	**0**
Nivel1'p2	**1**	**0**
Nivel1'p3	**1**	**0**
Nivel1'p4	**1**	**0**
Nivel1'p5	**1**	**0**
Nivel1'p6	**1**	**0**
Nivel1'p7	**1**	**0**
Nivel1'p8	**1**	**0**

Figura 41 – Dados da análise da propriedade *boundedness* do nível 1.

A última parte do relatório produzido pelo CPN Tools apresenta as informações sobre a propriedade *liveness* da rede. A figura 42 mostra as transições mortas do modelo (aquelas que não foram disparadas) e as transições vivas do modelo (aquelas que foram disparadas). As transições mortas do modelo são apresentadas do lado esquerdo da figura, ao passo que as transições vivas são apresentadas do lado direito.

A análise identificou transições que não podem ser disparadas. Como no primeiro nível do jogo o jogador não terá acesso imediato a todas as áreas, as transições do mapa topológico que correspondem as passagens para essas áreas não serão disparadas. No entanto, todas as transições que correspondem as atividades do nível 1, foram disparadas.

Liveness Properties

Dead Markings

None

Dead Transition Instances	Live Transition Instances
Nivel1't1	Nivel1'Consertar_escadas
Nivel1't10	Nivel1'Pegar_chave
Nivel1't17	Nivel1'Pegar_martelo
Nivel1't18	Nivel1'Pegar_pista_1
Nivel1't2	Nivel1'Pegar_pista_2
Nivel1't23	Nivel1'Pegar_sanduiches
Nivel1't24	Nivel1'Recuperar_memorias_1
Nivel1't25	Nivel1'Recuperar_memorias_2
Nivel1't26	Nivel1'T1
Nivel1't29	Nivel1'T2
Nivel1't30	Nivel1'T3
Nivel1't31	Nivel1'fork_1
Nivel1't32	Nivel1'join_1
Nivel1't9	Nivel1't11
	Nivel1't12
	Nivel1't13
	Nivel1't14
	Nivel1't15
	Nivel1't16
	Nivel1't19
	Nivel1't20
	Nivel1't21
	Nivel1't22
	Nivel1't27
	Nivel1't28
	Nivel1't3
	Nivel1't4
	Nivel1't5
	Nivel1't6

Figura 42 – Instâncias da propriedade *liveness* do nível 1.

Após a aplicação da análise do *state space* no modelo global modificado da figura 39, conclui-se então que o modelo global modificado é 1-limitado e que todas as transições envolvidas no nível são vivas. Conclui-se também que o comportamento do primeiro nível foi de acordo com o esperado, pois todas as atividades do nível foram executadas e todas as áreas liberadas para o primeiro nível podem ser acessadas pelo jogador.

5.4.2 Análise do nível 2

O modelo global modificado para a análise pode ser visto na figura 43 e equivale ao modelo global apresentado na figura 32.

O modelo lógico do nível 2 é representado no lado esquerdo da figura 43 e no lado direito está representado o modelo do mapa topológico. A marcação do mapa topológico no nível 2 é diferente da marcação do mapa topológico apresentada no nível 1. A localização inicial do jogador no nível 2 é representada por uma ficha no lugar *Area 2*. O lugar com a *fusion tag Chave*, que corresponde a primeira condição do jogo, também possui uma ficha, indicando que a condição já foi satisfeita no nível anterior. Portanto, a transição *T1* distribui uma ficha no lugar *Nivel 2*, uma no lugar *Area 2* e outra no lugar *C1*.

A análise do modelo global do nível 2 é então realizada. A primeira parte do resultado da análise do modelo global modificado da figura 43 pode ser vista na figura 44. Assim, como no nível 1, o SCC Graph do nível 2 possui apenas um componente fortemente conexo.

A segunda parte do resultado da análise corresponde aos dados da propriedade *boundedness* do modelo e está ilustrada na figura 45. Os nomes dos lugares destacados em

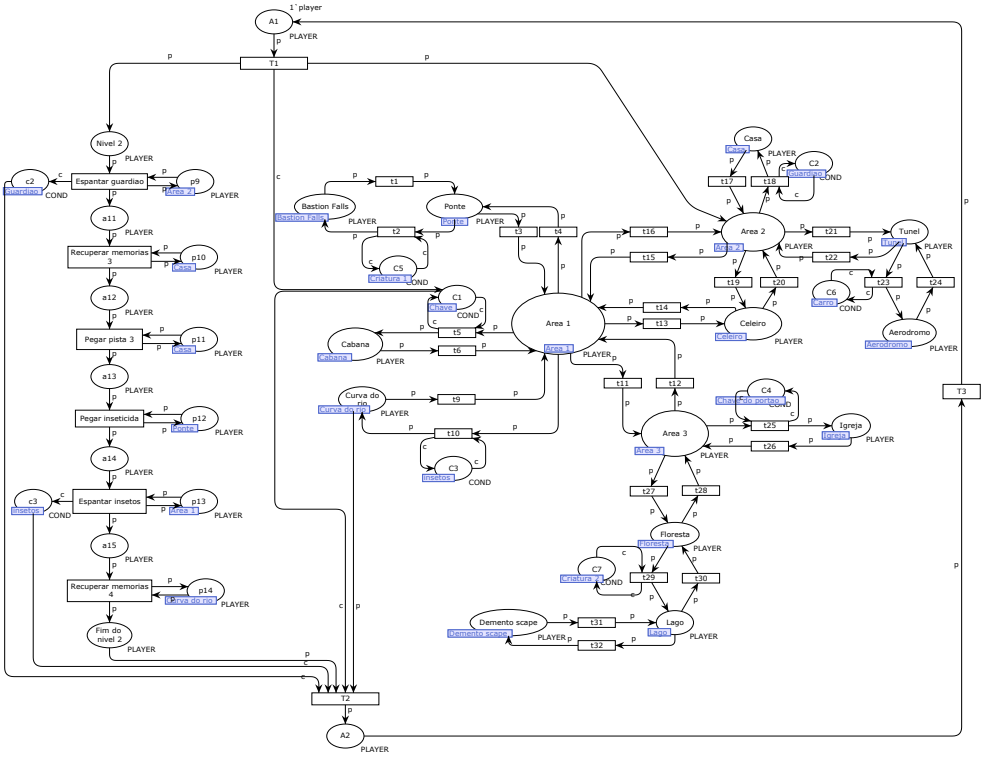


Figura 43 – Modelo global modificado para análise do nível 2.

```
Statistics
------------------------

  State Space
     Nodes:  66
     Arcs:   137
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 44 – Dados de análise produzidos após aplicar a análise do *state space* para o nível 2.

vermelho correspondem aos lugares do modelo que receberam (no máximo) uma ficha. Ainda no nível 2 não é possível acessar todos as áreas do mapa topológico, portanto, os lugares que não receberam fichas são os lugares de condição e as áreas do mapa topológico bloqueadas para o nível 2.

A terceira parte do resultado da análise diz respeito à propriedade *liveness* do modelo. A figura 46 apresenta as transições mortas e as transições vivas do modelo. As transições vivas (lado direito da figura 46) são as atividades do modelo lógico e as passagens entre as áreas liberadas para o nível 2. Já as transições mortas (lado esquerdo da figura 46) correspondem às passagens para as áreas que não são liberadas para o nível 2.

Com a análise, conclui-se que o nível 2 funcionou de acordo com o esperado, pois executou todas as atividades do nível e permitiu o acesso a todas as áreas liberadas para o nível 2. Como no nível 2 ainda não é possível acessar todas as áreas do mapa topológico,

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Nivel2'A1	**1**	**0**
Nivel2'A2	**1**	**0**
Nivel2'Aerodromo	0	0
Nivel2'Area_1	**1**	**0**
Nivel2'Area_2	**1**	**0**
Nivel2'Area_3	**1**	**0**
Nivel2'Bastion_Falls	0	0
Nivel2'C1	**1**	**0**
Nivel2'C2	**1**	**0**
Nivel2'C3	**1**	**0**
Nivel2'C4	0	0
Nivel2'C5	0	0
Nivel2'C6	0	0
Nivel2'C7	0	0
Nivel2'Cabana	**1**	**0**
Nivel2'Casa	**1**	**0**
Nivel2'Celeiro	**1**	**0**
Nivel2'Curva_do_rio	**1**	**0**
Nivel2'Demento_scape	0	0
Nivel2'Fim_do_nivel_2	**1**	**0**
Nivel2'Floresta	**1**	**0**
Nivel2'Igreja	0	0
Nivel2'Lago	0	0
Nivel2'Nivel_2	**1**	**0**
Nivel2'Ponte	**1**	**0**
Nivel2'Tunel	**1**	**0**
Nivel2'a11	**1**	**0**
Nivel2'a12	**1**	**0**
Nivel2'a13	**1**	**0**
Nivel2'a14	**1**	**0**
Nivel2'a15	**1**	**0**
Nivel2'c2	**1**	**0**
Nivel2'c3	**1**	**0**
Nivel2'p10	**1**	**0**
Nivel2'p11	**1**	**0**
Nivel2'p12	**1**	**0**
Nivel2'p13	**1**	**0**
Nivel2'p14	**1**	**0**
Nivel2'p9	**1**	**0**

Figura 45 – Dados da análise da propriedade *boundedness* do nível 2.

Liveness Properties

Dead Markings

None

Dead Transition Instances	Live Transition Instances
Nivel2't1	Nivel2'Espantar_guardiao
Nivel2't2	Nivel2'Espantar_insetos
Nivel2't23	Nivel2'Pegar_inseticida
Nivel2't24	Nivel2'Pegar_pista_3
Nivel2't25	Nivel2'Recuperar_memorias_3
Nivel2't26	Nivel2'Recuperar_memorias_4
Nivel2't29	Nivel2'T1
Nivel2't30	Nivel2'T2
Nivel2't31	Nivel2'T3
Nivel2't32	Nivel2't10
	Nivel2't11
	Nivel2't12
	Nivel2't13
	Nivel2't14
	Nivel2't15
	Nivel2't16
	Nivel2't17
	Nivel2't18
	Nivel2't19
	Nivel2't20
	Nivel2't21
	Nivel2't22
	Nivel2't27
	Nivel2't28
	Nivel2't3
	Nivel2't4
	Nivel2't5
	Nivel2't6
	Nivel2't9

Figura 46 – Instâncias da propriedade *liveness* do nível 2.

as transições que representam as passagens para as áreas bloqueadas não são disparadas.

Portanto, o modelo global do nível 2 é 1-limitado e todas as transições envolvidas com o nível estão vivas.

5.4.3 Análise do nível 3

A figura 34 ilustra o modelo global do nível 3. Neste nível, três das sete condições de liberação de áreas do jogo já foram satisfeitas nos níveis anteriores. Portanto, a transição *T1* distribuirá uma ficha nos lugares que estão marcado com as *fusion tags Chave*, *Guardiao* e *Insetos*. No nível 3 o jogador começa na área *Area 3* do mapa topológico. Assim, o lugar *Area 3* também receberá uma ficha, bem como o lugar *Nivel 3*, que indica o começo do nível. A figura 47, ilustra o modelo global modificado para a análise.

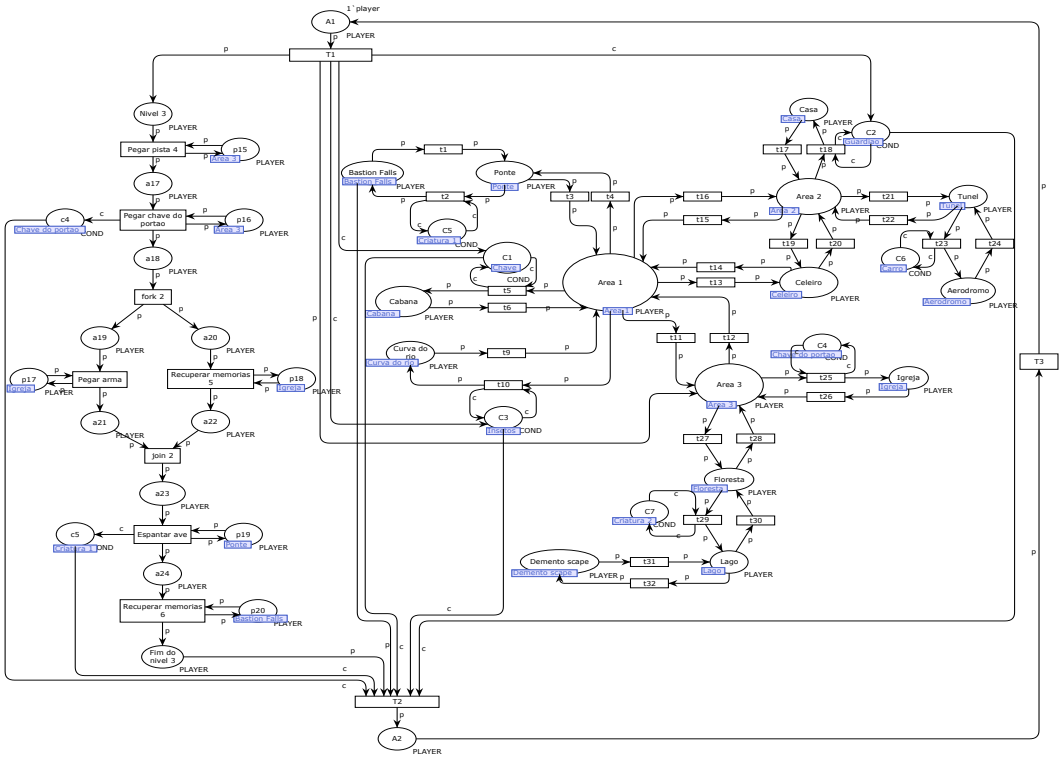


Figura 47 – Modelo global modificado para análise do nível 3.

As informações relativas aos dados de análise produzidos pela enumeração do conjunto dos estados alcançáveis apresentada na figura 48, indica que o grafo de alcançabilidade do modelo analisado possui apenas um componente fortemente conexo (SCC Graph).

A figura 49 ilustra os resultados da propriedade *boundedness* do modelo submetido à análise do *state space*. Como pode ser visto na figura 49, apenas cinco lugares do mapa topológico ainda não receberam ficha. Em particular, esses lugares correspondem a lugares de condição e à áreas do mapa topológico que estão bloqueadas para o nível 3.

A figura 50 apresenta as instâncias da propriedade *liveness*. É possível notar que todas as transições que representam as atividades do nível 3 foram sensibilizadas. Além disso, mais transições do grafo de estado foram sensibilizadas. Isso significa que no nível 3 o

```
Statistics
------------------------
  State Space
     Nodes:  112
     Arcs:   253
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 48 – Dados de análise após aplicar a análise do *state space* para o nível 3.

```
Boundedness Properties
----------------------------------------

Best Integer Bounds
                        Upper      Lower
 Nivel3'A1                1          0
 Nivel3'A2                1          0
 Nivel3'Aerodromo         0          0
 Nivel3'Area_1            1          0
 Nivel3'Area_2            1          0
 Nivel3'Area_3            1          0
 Nivel3'Bastion_Falls     1          0
 Nivel3'C1                1          0
 Nivel3'C2                1          0
 Nivel3'C3                1          0
 Nivel3'C4                1          0
 Nivel3'C5                1          0
 Nivel3'C6                0          0
 Nivel3'C7                0          0
 Nivel3'Cabana            1          0
 Nivel3'Casa              1          0
 Nivel3'Celeiro           1          0
 Nivel3'Curva_do_rio      1          0
 Nivel3'Demento_scape     0          0
 Nivel3'Fim_do_nivel_3    1          0
 Nivel3'Floresta          1          0
 Nivel3'Igreja            1          0
 Nivel3'Lago              0          0
 Nivel3'Nivel_3           1          0
 Nivel3'Ponte             1          0
 Nivel3'Tunel             1          0
 Nivel3'a17               1          0
 Nivel3'a18               1          0
 Nivel3'a19               1          0
 Nivel3'a20               1          0
 Nivel3'a21               1          0
 Nivel3'a22               1          0
 Nivel3'a23               1          0
 Nivel3'a24               1          0
 Nivel3'c4                1          0
 Nivel3'c5                1          0
 Nivel3'p15               1          0
 Nivel3'p16               1          0
 Nivel3'p17               1          0
 Nivel3'p18               1          0
 Nivel3'p19               1          0
 Nivel3'p20               1          0
```

Figura 49 – Dados da análise da propriedade *boundedness* do nível 3.

jogador pode acessar mais áreas do mapa topológico. Embora o número de transições vivas tenha aumentado, ainda há transições mortas. Essas transições mortas, correspondem as passagens para as áreas bloqueadas para o nível 3.

Assim como o nível 2, o nível 3 se comportou de forma correta. As atividades do nível foram executadas e as áreas do mapa liberadas para o nível 3 foram acessadas.

5.4.4 Análise do nível 4

O nível 4 é o penúltimo nível do jogo. Nesta etapa, o jogador já poderá ter acesso a quase todas as áreas do mapa topológico. A figura 51, apresenta o modelo global

Liveness Properties

Dead Markings

None

Dead Transition Instances	Live Transition Instances
Nivel3't23	Nivel3'Espantar_ave
Nivel3't24	Nivel3'Pegar_arma
Nivel3't29	Nivel3'Pegar_chave_do_portao
Nivel3't30	Nivel3'Pegar_pista_4
Nivel3't31	Nivel3'Recuperar_memorias_5
Nivel3't32	Nivel3'Recuperar_memorias_6
	Nivel3'T1
	Nivel3'T2
	Nivel3'T3
	Nivel3'fork_2
	Nivel3'join_2
	Nivel3't1
	Nivel3't10
	Nivel3't11
	Nivel3't12
	Nivel3't13
	Nivel3't14
	Nivel3't15
	Nivel3't16
	Nivel3't17
	Nivel3't18
	Nivel3't19
	Nivel3't2
	Nivel3't20
	Nivel3't21
	Nivel3't22
	Nivel3't25
	Nivel3't26
	Nivel3't27
	Nivel3't28
	Nivel3't3
	Nivel3't4
	Nivel3't5
	Nivel3't6
	Nivel3't9

Figura 50 – Instâncias da propriedade *liveness* do nível 3.

modificado para a análise. O modelo da figura 51 equivale ao modelo da figura 36.

A transição T1 distribui uma ficha para o lugar *Nivel 4*, outra para o lugar *Tunel* (localização do jogador no início do nível) e uma ficha nos lugares de condição *C1*, *C2*, *C3*, *C4* e *C5* (condições que já foram satisfeitas nos níveis anteriores).

As informações relativas aos dados de análise produzidos pela enumeração do conjunto dos estados alcançáveis após aplicar a análise do *state space* para o nível 4, pode ser vista na figura 52. Em particular, o SCC Graph indica apenas um componente fortemente conexo no grafo de acançabilidade.

A figura 53 apresenta os resultados da propriedade *boundedness*. Os nomes destacados em vermelho representam os lugares da rede que receberam no máximo uma ficha. Como esperado, apenas três lugares não receberam ficha. Esses lugares correspondem a duas áreas do mapa topológico e a uma condição.

As instâncias da propriedade *liveness* são apresentadas na figura 54. As transições vivas estão ao lado direito da figura e as transições mortas ao lado esquerdo. Todas as transições que representam as atividades do nível 4 foram disparadas, o que indica que todas as atividades do nível 4 foram realizadas. As únicas transições que não podem ser disparadas são as transições associadas aos lugares bloqueados para o nível 4.

Com a análise, é possível concluir então que o modelo global modificado é 1-limitado e que todas as transições envolvidas com o nível estão vivas. O comportamento do nível 4 foi condizente com o esperado, uma vez que todas as atividades do nível foram executadas e todas as áreas liberadas para o nível 4 podem ser acessadas.

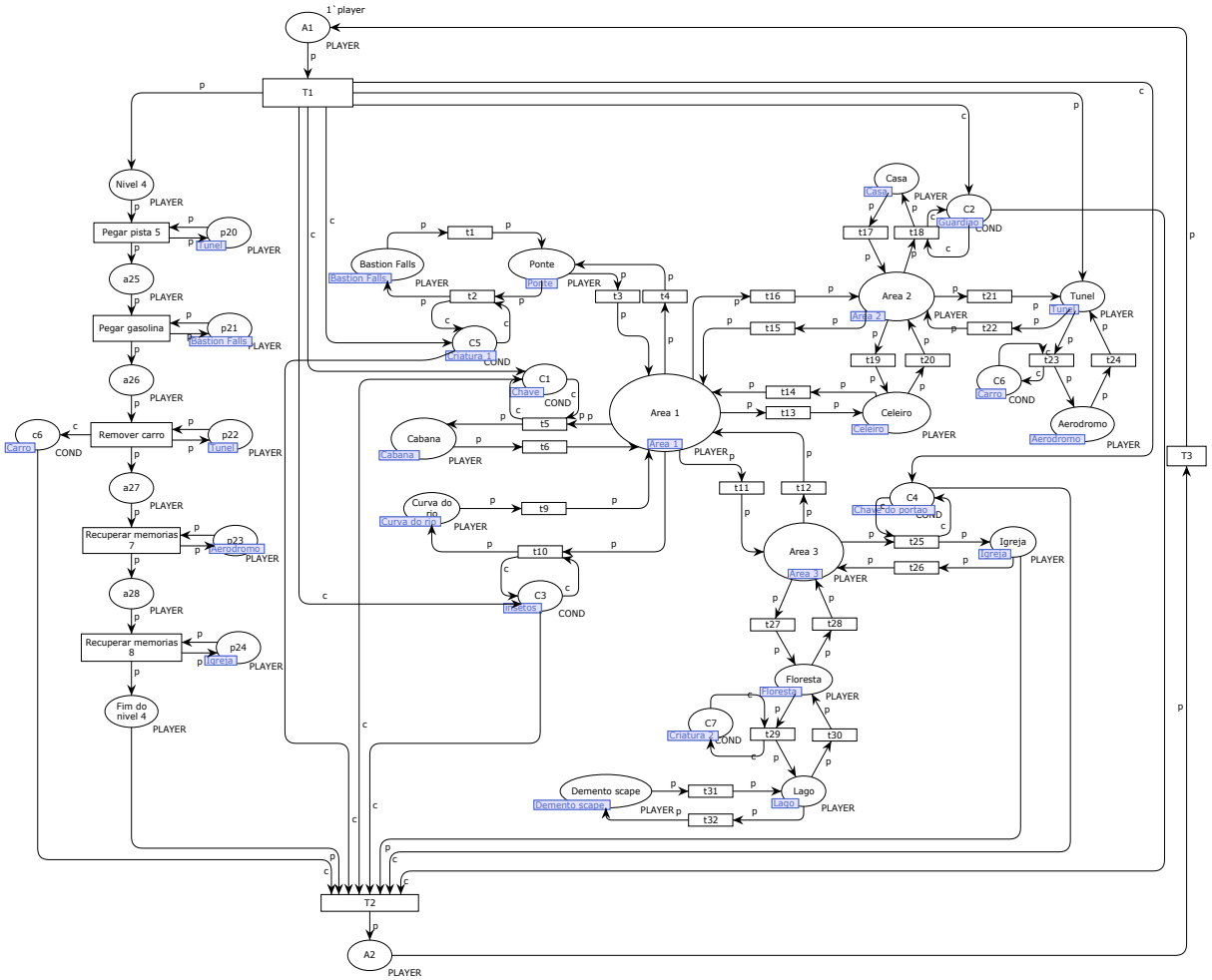


Figura 51 – Modelo global modificado para análise do nível 4.

```
Statistics
------------------------

  State Space
     Nodes:  77
     Arcs:   158
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 52 – Dados de análise após aplicar a análise do *state space* para o nível 4.

5.4.5 Análise do nível 5

A última etapa do jogo Dream:scape é representada pelo nível 5. No nível 5, o jogador acalçará o objetivo e concluirá o jogo. O modelo global do nível 5 modificado para a análise é apresentado na figura 55. No começo do nível 5, o jogador deve estar na área *Celeiro* e uma ficha é adicionada neste lugar pela transição *T1*. Todos os lugares de condição do mapa topológico recebem também uma ficha, com exceção do lugar *C7*, que corresponde a última condição do jogo e que será satisfeita no nível 5. O lugar *Nivel 5*,

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Nivel4'A1	**1**	**0**
Nivel4'A2	**1**	**0**
Nivel4'Aerodromo	**1**	**0**
Nivel4'Area_1	**1**	**0**
Nivel4'Area_2	**1**	**0**
Nivel4'Area_3	**1**	**0**
Nivel4'Bastion_Falls	**1**	**0**
Nivel4'C1	**1**	**0**
Nivel4'C2	**1**	**0**
Nivel4'C3	**1**	**0**
Nivel4'C4	**1**	**0**
Nivel4'C5	**1**	**0**
Nivel4'C6	**1**	**0**
Nivel4'C7	0	0
Nivel4'Cabana	**1**	**0**
Nivel4'Casa	**1**	**0**
Nivel4'Celeiro	**1**	**0**
Nivel4'Curva_do_rio	**1**	**0**
Nivel4'Demento_scape	0	0
Nivel4'Fim_do_nivel_4	**1**	**0**
Nivel4'Floresta	**1**	**0**
Nivel4'Igreja	**1**	**0**
Nivel4'Lago	0	0
Nivel4'Nivel_4	**1**	**0**
Nivel4'Ponte	**1**	**0**
Nivel4'Tunel	**1**	**0**
Nivel4'a25	**1**	**0**
Nivel4'a26	**1**	**0**
Nivel4'a27	**1**	**0**
Nivel4'a28	**1**	**0**
Nivel4'c6	**1**	**0**
Nivel4'p20	**1**	**0**
Nivel4'p21	**1**	**0**
Nivel4'p22	**1**	**0**
Nivel4'p23	**1**	**0**
Nivel4'p24	**1**	**0**

Figura 53 – Dados da análise da propriedade *boundedness* do nível 4.

Liveness Properties

Dead Markings
None

Dead Transition Instances	**Live Transition Instances**
Nivel4't29	Nivel4'Pegar_gasolina
Nivel4't30	Nivel4'Pegar_pista_5
Nivel4't31	Nivel4'Recuperar_memorias_7
Nivel4't32	Nivel4'Recuperar_memorias_8
	Nivel4'Remover_carro
	Nivel4'T1
	Nivel4'T2
	Nivel4'T3
	Nivel4't1
	Nivel4't10
	Nivel4't11
	Nivel4't12
	Nivel4't13
	Nivel4't14
	Nivel4't15
	Nivel4't16
	Nivel4't17
	Nivel4't18
	Nivel4't19
	Nivel4't2
	Nivel4't20
	Nivel4't21
	Nivel4't22
	Nivel4't23
	Nivel4't24
	Nivel4't25
	Nivel4't26
	Nivel4't27
	Nivel4't28
	Nivel4't3
	Nivel4't4
	Nivel4't5
	Nivel4't6
	Nivel4't9

Figura 54 – Instâncias da propriedade *liveness* do nível 4.

do modelo lógico, também receberá uma ficha.

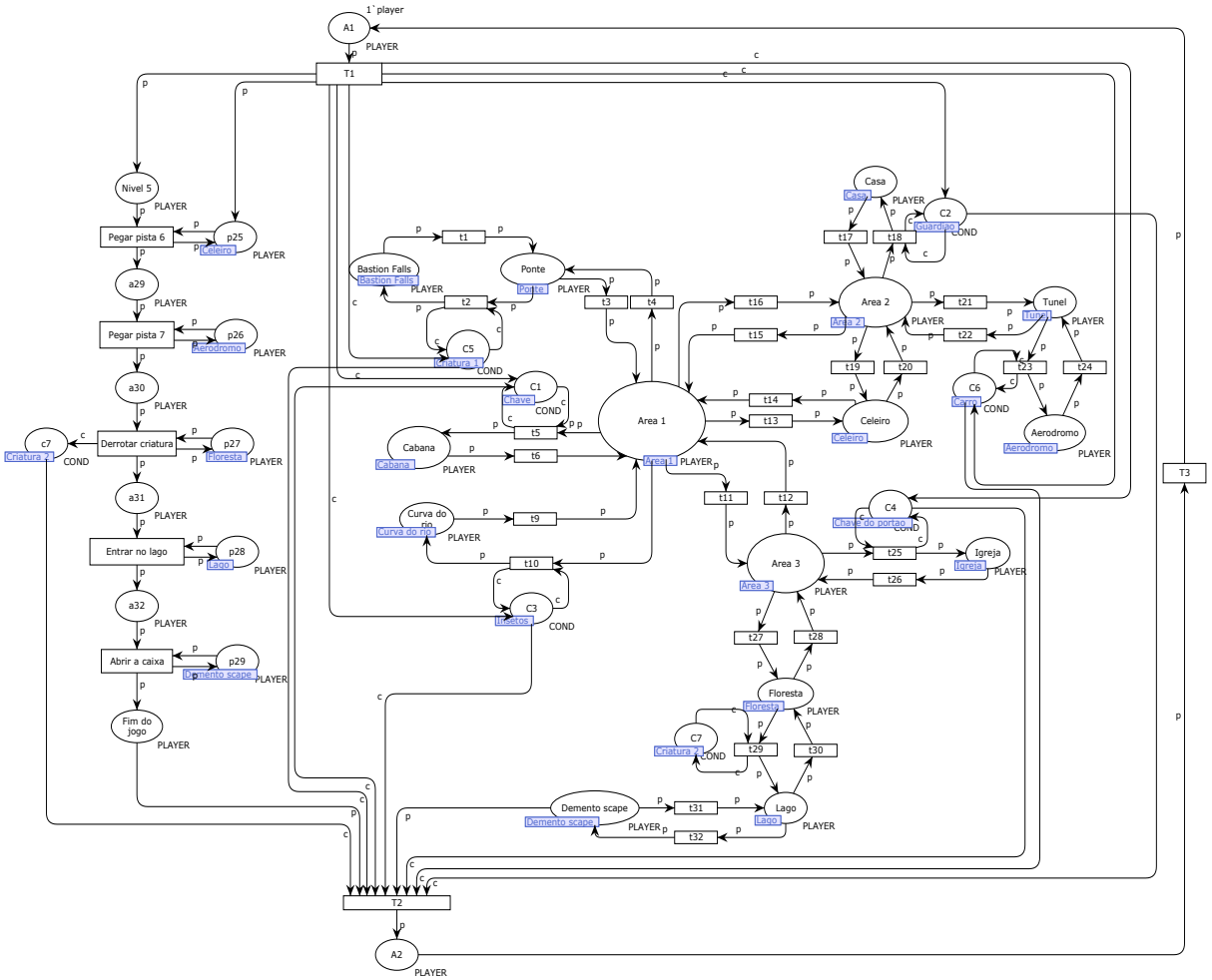


Figura 55 – Modelo global modificado para análise do nível 5.

Após aplicar a análise do *state space* para o nível 5, as informações relativas aos dados de análise produzidos pela enumeração do conjunto dos estados alcançáveis são representadas na figura 56. Assim como nos níveis anteriores, o grafo de alcançabilidade do modelo analisado possui apenas um componente fortemente conexo.

```
Statistics
------------------------

  State Space
     Nodes:  86
     Arcs:   176
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  1
     Arcs:   0
     Secs:   0
```

Figura 56 – Dados de análise após aplicar a análise do *state space* para o nível 5.

Os dados da propriedade *boundedness* são representadas na figura 57. Vê-se claramente na figura 57 que todos os lugares da rede receberam no máximo uma ficha, o que indica que a rede é limitada pelo número um.

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Nivel5'A1	1	0
Nivel5'A2	1	0
Nivel5'Aerodromo	1	0
Nivel5'Area_1	1	0
Nivel5'Area_2	1	0
Nivel5'Area_3	1	0
Nivel5'Bastion_Falls	1	0
Nivel5'C1	1	0
Nivel5'C2	1	0
Nivel5'C3	1	0
Nivel5'C4	1	0
Nivel5'C5	1	0
Nivel5'C6	1	0
Nivel5'C7	1	0
Nivel5'Cabana	1	0
Nivel5'Casa	1	0
Nivel5'Celeiro	1	0
Nivel5'Curva_do_rio	1	0
Nivel5'Demento_scape	1	0
Nivel5'Fim_do_nivel_5	1	0
Nivel5'Floresta	1	0
Nivel5'Igreja	1	0
Nivel5'Lago	1	0
Nivel5'Nivel_5	1	0
Nivel5'Ponte	1	0
Nivel5'Tunel	1	0
Nivel5'a29	1	0
Nivel5'a30	1	0
Nivel5'a31	1	0
Nivel5'a32	1	0
Nivel5'c7	1	0
Nivel5'p25	1	0
Nivel5'p26	1	0
Nivel5'p27	1	0
Nivel5'p28	1	0
Nivel5'p29	1	0

Figura 57 – Dados da análise da propriedade *boundedness* do nível 5.

As instâncias da propriedade *liveness* para o nível 5 podem ser vistas na figura 58. A figura 58 mostra que todas as transições da rede são vivas. Portanto, a modelo global modificado do nível 5 é vivo. Nos níveis anteriores nem todas as transições eram vivas, isso se deve ao fato de que algumas áreas estavam bloqueadas para o nível em questão. Já no nível 5 todas as áreas estão desbloqueadas, uma vez que todas as atividades dos níveis anteriores já foram executadas e todas as condições de liberação de área já foram satisfeitas.

Liveness Properties

Dead Markings
None

Dead Transition Instances
None

Live Transition Instances
None

Figura 58 – Instâncias da propriedade *liveness* do nível 5.

De acordo com a análise, o modelo global modificado do nível 5 é vivo e 1-limitado. Portanto, o modelo global não modificado do nível 5 (figura 38) pode ser considerado *sound*. Do ponto de vista do jogo, isso significa que todas as atividades podem ser executadas e que todos os lugares do mapa topológico podem ser eventualmente acessados.

5.5 Considerações Finais

A abordagem apresentada neste trabalho propõe a modelagem e análise qualitativa de cenários de vídeo games utilizando as redes de Petri. A modelagem de um cenário de vídeo game é composta pelo modelo de atividades do nível e pelo modelo do mapa topológico do jogo. O modelo de atividades expressa as tarefas do nível que o jogador precisa executar, e o mapa topológico expressa a evolução do jogador no ambiente durante a execução das atividades.

Esta abordagem foi apresentada neste capítulo por meio de um estudo de caso aplicado ao jogo Dream:scape. O jogo Dream:scape foi divido em cinco níveis. Para cada nível um modelo global foi criado. O modelo global por sua vez, foi composto do modelo de atividades do nível e do modelo do mapa topológico. Além disso, mecanismos de comunicação baseados na noção de lugares de fusão do CPN Tools foram considerados para relacionar os dois modelos.

Todos os modelos foram adaptados para a ferramenta CPN Tools. A ferramenta além de proporcionar uma representação gráfica mais clara dos modelos, também possibilita a utilização de funcionalidades para análise, como por exemplo, a funcionalidade de análise *state space* e a funcionalidade de simulação passo a passo. A análise *state space* aplicada aos modelos no CPN Tools, gera automaticamente um relatório com as propriedades da rede analisada. Com os dados desse relatório, é possível verificar a quais propriedades o modelo analisado satisfaz. Já com a funcionalidade de simulação passo a passo do CPN Tools, é possível verificar minuciosamente o comportamento da rede, verificando passo a passo os disparos das transições e a disposição das fichas durante a execução do modelo.

O jogo foi analisado então por níveis. Cada modelo global de nível foi submetido à análise do *state space* através da funcionalidade disponível no CPN Tools. Basicamente, a análise dos níveis consistiu da verificação da propriedade *soundness*. Como os modelos foram adaptados para o CPN Tools, foi possível utilizar a funcionalidade de análise *state space*, disponível no CPN Tools. Com a análise do *state space* foi possível verificar que o modelo global do nível 1, nível 2, nível 3 e nível 4, não são *sound*, de acordo com o critério definido para a propriedade *soundness* adaptada ao contexto de vídeo games. Esse resultado já era esperado, pois as áreas do mapa topológico do jogo vão sendo liberadas de acordo com a execução das atividades nos níveis. Dessa forma, cada nível se comportou de maneira correta, ou seja, todas as atividades relacionadas ao nível podem ser executadas e todos as áreas do mapa liberados para o nível podem ser acessadas. Então, ao fim do jogo (nível 5), todos os lugares estarão disponíveis para o jogador acessar. Assim, o modelo global do nível 5 foi verificado como sendo *sound*. Com tudo isso, conclui-se então que é possível executar todas as atividades do jogo Dream:scape e acessar a todas as áreas do mapa topológico do jogo. De modo geral, pode-se dizer que o jogo é correto em termos de execução das atividades e da construção do mapa topológico.

Capítulo 6

Conclusão

Neste trabalho foi apresentado uma abordagem baseada em redes de Petri para auxiliar no processo de criação de vídeo games. A abordagem apresentada consiste da modelagem e análise qualitativa de cenários de vídeo games.

Em geral, um jogo possui várias atividades que o jogador precisa executar para alcançar os objetivos do jogo e um mundo virtual onde essas atividades serão executadas. Um jogo também possui vários níveis, e cada nível contém sua sequência de atividades. Para a representação das atividades, foi utilizado um tipo de rede de Petri denominado WorkFlow net. As WorkFlow nets, em seu sentido clássico, modelam processos de negócio. Um nível de jogo é similar à estrutura clássica de um processo de negócio, uma vez que ambos possuem um começo e um objetivo final que será alcançado após a execução de várias atividades. Assim, as WorkFlows nets se mostraram adequadas para a modelagem das atividades de um nível de jogo.

Para representar o mapa topológico de um jogo, foi utilizado um tipo de rede de Petri denominado grafo de estado. Os lugares do grafo de estado representaram as regiões do mapa topológico. Já as transições representaram as fronteiras entre as regiões do mundo virtual. A essas transições, foram associados lugares denominados de lugares de condição, onde cada lugar de condição representou uma condição necessária para ter acesso a uma determinada região do mundo virtual.

A relação entre o modelo que representa as atividades de um nível e o modelo que representa o mapa topológico foi representada por um mecanismo de comunicação, também baseado em redes de Petri. O mecanismo de comunicação associa as transições do modelo de atividades aos lugares de condição do modelo do mapa topológico, e associa os lugares que representam as regiões do mapa às transições do modelo de atividades. Em outras palavras, isso significa que a execução de uma determinada atividade é condição necessária para se ter acesso a uma determina região do mundo virtual, assim como a execução de uma atividade só acontecerá quando o jogador estiver localizado em uma região específica do mapa. Portanto, o mecanismo de comunicação especifica as influências que um modelo tem sobre o outro.

A análise qualitativa pôde ser então realizada no modelo global obtido com a junção do modelo de atividades de um nível e do modelo do mapa topológico. O método proposto para a análise é baseado no algoritmo de verificação da propriedade *soundness*. A propriedade *soundness* foi adaptada ao contexto de vídeo games. Através deste método, foi possível verificar a consistência de um cenário de vídeo game utilizando ferramentas baseadas em redes de Petri. Em particular, a funcionalidade de análise *state space*, da ferramenta CPN Tools, foi utilizada para a verificação da propriedade *soundness* adaptada ao contexto de vídeo games. Por meio da análise, foi possível verificar, para um jogo dado, se as atividades previstas para um nível foram executadas e se todas as áreas do mapa topológico foram acessadas. Esse tipo de verificação garante que não haverá erros no jogo em termos de execução das tarefas e da disposição das áreas e localização de itens de jogo a serem recolhidos no mundo virtual.

O uso do CPN Tools possibilitou também a representação gráfica e a utilização do conceito de *fusion places*. Foi possível representar os modelos de atividades e de mapa topológico utilizando o CPN Tools sem que perdessem os seus significados. Além disso, o uso do conceito de *fusion places* foi fundamental para manter os dois modelos visualmente distintos, deixando a representação gráfica dos modelos simples e clara.

Comparando esta abordagem com outros trabalhos que tratam da modelagem de jogos, a principal vantagem é a representação do fluxo de atividades e do mundo virtual de um jogo em um único formalismo, tornando possível a concepção de um modelo de análise que verifica a consistência das atividades e a boa construção do mundo virtual de um jogo.

6.1 Trabalhos Futuros

Como proposta de trabalho futuro, seria interessante considerar uma modelagem hierárquica da abordagem proposta. A modelagem hierárquica dos níveis permitiria uma representação explorativa do jogo interconectando todos os níveis existentes. A ferramenta CPN Tools poderia ser utilizada para a representação gráfica dos modelos, uma vez que a ferramenta disponibiliza funcionalidades para modelagem hierárquica. Além disso, a validação e análise dos modelos hierárquicos poderiam ser realizadas por meio da funcionalidade de simulação disponível no CPN Tools.

A abordagem apresentada neste trabalho trata apenas de situações onde o jogador consegue executar corretamente todas as atividades. Seria interessante especificar por meio da modelagem quais as próximas ações possíveis quando um jogador falha ao executar alguma tarefa do jogo. Também seria interessante considerar um modelo temporal com a estimação dos tempos através de fórmulas da Lógica Linear, e a noção de objetos (itens) de jogos que são adquiridos pelo jogador de acordo com a evolução do jogo.

Um outro trabalho interessante seria estender a abordagem proposta neste trabalho

para cenários de jogos *multiplayer* (multijogador), ou seja, um trabalho que abordaria a situação em que há mais de um jogador ativo no jogo. Seria interessante descrever por meio dos modelos em redes de Petri o comportamento de um jogo com mais de um jogador onde a ação de um jogador influencia direta ou indiretamente nas ações dos demais jogadores. Neste caso, a análise quantitativa e qualitativa dos modelos poderia ser realizada utilizando Lógica Linear e a funcionalidade de simulação do CPN Tools.

6.2 Contribuições em Produção Bibliográfica

Além da escrita desta dissertação, este trabalho teve como resultados a publicação e apresentação dos seguintes artigos:

- *A Straightforward Introduction to Formal Methods Using Coloured Petri Nets* (BARRETO et al., 2014), aceito para publicação pela *16th International Conference on Enterprise Information Systems* (ICEIS 2014), realizado em Lisboa - Portugal.
- *Modeling and analysis of video games based on Workflow nets and State Graphs* (BARRETO; JULIA, 2014), aceito para publicação pela *24th Annual International Conference on Computer Science and Software Engineering* (CASCON 2014), realizado em Markham - Canadá.

Referências

AALST, W. M. van der. Verification of workflow nets. In: **Application and Theory of Petri Nets 1997**. [S.l.]: Springer, 1997. p. 407–426.

_____. The application of petri nets to workflow management. **Journal of circuits, systems, and computers**, World Scientific, v. 8, n. 01, p. 21–66, 1998.

AALST, W. M. van der; HOFSTEDE, A. H. ter. Verification of workflow task structures: A petri-net-baset approach. **Information Systems**, v. 25, n. 1, p. 43 – 69, 2000. ISSN 0306-4379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437900000089>.

AALST, W. V. D.; HEE, K. M. V. **Workflow management: models, methods, and systems**. [S.l.]: MIT press, 2004.

AALST, W. van der. Timed coloured Petri nets and their application to logistics. In: **International Symposium on Physical Design**. [S.l.: s.n.], 1992.

_____. **Structural Characterizations of Sound Workflow Nets**. 1996.

ANDREU, D.; PASCAL, J.; VALETTE, R. Events as a key of a batch process control system. In: **Proc. CESA'96, Symp. On Discrete Events and Manufacturing Systems**. [S.l.: s.n.], 1996.

ANG, C. S.; RAO, G. S. V. R. K. Designing interactivity in computer games: a uml approach. **Int. J. Intell. Games Simulation**, v. 3, n. 2, p. 62–69, 2004.

ARAÚJO, M.; ROQUE, L. Modeling games with petri nets. **Breaking New Ground: Innovation in Games, Play, Practice and Theory. DIGRA2009. Londres, Royaume Uni**, 2009.

BARRETO, F. M. et al. A straightforward introduction to formal methods using coloured petri nets. In: **16th international conference on enterprise information systems**. [S.l.: s.n.], 2014. (ICEIS 2014, v. 2), p. 145–152.

BARRETO, F. M.; JULIA, S. Modeling and analysis of video games based on workflow nets and state graphs. In: **Proceedings of 24th Annual International Conference on Computer Science and Software Engineering**. Riverton, NJ, USA: IBM Corp., 2014. (CASCON '14), p. 106–119. Disponível em: <http://dl.acm.org/citation.cfm?id=2735522.2735535>.

BATES, B. **Game Design: The Art and Business of Creating Games**. [S.l.]: Prima Publishing, 2001.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. [S.l.]: Editora da UFSC, 1997.

CLAYPOOL, K.; CLAYPOOL, M. Teaching software engineering through game design. In: **Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education**. New York, NY, USA: ACM, 2005. (ITiCSE '05), p. 123–127. ISBN 1-59593-024-8. Disponível em: <http://doi.acm.org/10.1145/1067445.1067482>.

COLLÉ, F.; CHAMPAGNAT, R.; PRIGENT, A. Scenario analysis based on linear logic. In: **Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology**. New York, NY, USA: ACM, 2005. (ACE

05). ISBN 1-59593-110-4. Disponível em: <http://doi.acm.org/10.1145/1178477.1178583>.

CRAWFORD, C. **The art of computer game design**. [S.l.]: Osborne/McGraw-Hill, 1982.

GAL, V. et al. Writing for video games. **Proceedings Laval Virtual (IVRC)**, Citeseer, 2002.

HORROCKS, I. **Constructing the User Interface with Statecharts**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201342782.

JENSEN, K. Coloured petri nets and the invariant-method. **Theoretical Computer Science**, v. 14, n. 3, p. 317 – 336, 1981. ISSN 0304-3975. Disponível em: <http://www.sciencedirect.com/science/article/pii/0304397581900499>.

_____. An introduction to the practical use of coloured petri nets. In: **Lectures on Petri Nets II: Applications**. [S.l.]: Springer, 1998. p. 237–292.

JENSEN, K.; KRISTENSEN, L. **Coloured Petri Nets**. [S.l.]: Springer, 2009. ISBN 978-3-642-00283-0.

KANODE CHRISTOPHER M. E HADDAD, H. M. Software engineering challenges in game development. In: LATIFI, S. (Ed.). **ITNG**. [S.l.]: IEEE Computer Society, 2009. p. 260–265. ISBN 978-0-7695-3596-8.

KINIRY, J. R.; ZIMMERMAN, D. M. Verified gaming. In: **Proceedings of the 1st International Workshop on Games and Software Engineering**. New York, NY, USA: ACM, 2011. (GAS '11), p. 17–20. ISBN 978-1-4503-0578-5. Disponível em: <http://doi.acm.org/10.1145/1984674.1984681>.

LEE, Y.-S.; CHO, S.-B. Dynamic quest plot generation using petri net planning. In: **Proceedings of the Workshop at SIGGRAPH Asia**. New York, NY, USA: ACM, 2012. (WASA '12), p. 47–52. ISBN 978-1-4503-1835-8. Disponível em: <http://doi.acm.org/10.1145/2425296.2425304>.

LEWIS, C.; WHITEHEAD, J. The whats and the whys of games and software engineering. In: **Proceedings of the 1st International Workshop on Games and Software Engineering**. New York, NY, USA: ACM, 2011. (GAS '11), p. 1–4. ISBN 978-1-4503-0578-5. Disponível em: <http://doi.acm.org/10.1145/1984674.1984676>.

MILNER, R.; HARPER, R.; TOFTE, M. **The Definition of Standard ML**. [S.l.]: MIT Press, 1990.

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541–580, April 1989.

NATKIN, S.; VEGA, L.; GRÜNVOGEL, S. A new methodology for spatiotemporal game design. In: **Mehdi, Q. and Gough, N.,(Eds.). Proceedings of CGAIDE 2004, 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education**. [S.l.: s.n.], 2004. p. 109–113.

NEIL, K. Game design tools: Time to evaluate. In: **Proceedings of DiGRA Nordic 2012 Conference: Local and Global Games in Culture and Society**. [S.l.: s.n.], 2012.

OLIVEIRA, G. W. d. Modelagem e análise de video games usando as workflow nets ea lógica linear. **Faculdade de Computação UFU**, 2012.

OLIVEIRA, G. W. de; JULIA, S.; PASSOS, L. M. S. Game modeling using workflow nets. In: **SMC**. IEEE, 2011. p. 838–843. ISBN 978-1-4577-0652-3. Disponível em: <http://dblp.uni-trier.de/db/conf/smc/smc2011.html#OliveiraJP11>.

PETERSON, J. **Petri net theory and the modeling of systems**. [S.l.]: Prentice-Hall, Englewood Cliffs, 1981.

RENÉ, D.; HASSANE, A. **Discrete, continuous, and hybrid Petri Nets**. Berlin; London: Springer, 2010. Disponível em: <http://www.amazon.com/Discrete-Continuous-Hybrid-Petri-Nets/dp/3642106684/ref=dp_ob_image_bk>.

REYNO, E. M.; CUBEL, J. . C. Automatic prototyping in model-driven game development. **Computers in Entertainment**, v. 7, n. 2, 2009. Disponível em: <http://dblp.uni-trier.de/db/journals/cie/cie7.html#ReynoC09>.

ROLLINGS, A.; MORRIS, D. **Game Architecture and Design: A New Edition [Taschenbuch]**. [S.l.]: New Riders, 2003.

ROUSE, R.; RYBCZYK, M. L. **Game Design: Theory and Practice (With CD-ROM)**. Wordware Publishing, 2001. Paperback. ISBN 1556227353. Disponível em: <http://www.amazon.de/exec/obidos/ASIN/1556227353>.

RUCKER, R. v. B. **Software engineering and computer games**. [S.l.]: Pearson Education, 2003.

SPEEDBUMP. **Dream:scape, electronic game, developed by Speedbump and published by iTunes**. 2012.

SYUFAGI, M. A.; HARIADI, M.; PURNOMO, M. H. Petri net model for serious games based on motivation behavior classification. **Int. J. Computer Games Technology**, v. 2013, 2013.

THOMAS, P.; YESSAD, A.; LABAT, J.-M. Petri nets and ontologies: Tools for the "learning player"assessment in serious games. In: **ICALT**. [S.l.]: IEEE Computer Society, 2011. p. 415–419.

WOODCOCK, J. et al. Formal methods: Practice and experience. **ACM Comput. Surv.**, v. 41, n. 4, 2009.