UNIVERSIDADE DO VALE DO RIO DO SINOS

PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO
EM COMPUTAÇÃO APLICADA

Leandro Motta Barros

Um Modelo para Prover Consistência Narrativa em *Interactive Storytelling*

São Leopoldo
2007

Leandro Motta Barros

Um Modelo para Prover Consistência Narrativa em *Interactive Storytelling*

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de mestre em Computação Aplicada.

Orientadora:
Prof. PhD. Soraia Raupp Musse

São Leopoldo

2007

AGRADECIMENTOS

Inicio, mais uma vez, agradecendo aos meus pais por todo o apoio que sempre me deram em tudo.

Agradeço à Soraia, minha orientadora, por mais dois bons anos de trabalho e ótimo convívio.

Obrigado à minha família, em especial à minha dinda Lilia, que sempre dá um jeito nos animar, e à minha prima Mariana, por motivos já explicados (e documentados!) há coisa de dois anos atrás e que não vou ficar repetindo.

É mais fácil fazer as coisas quando há pessoas legais por perto. Por isso, meu agradecimento também aos vários amigos e amigas da Unisinos com quem convivi de forma mais ou menos próxima durante a realização deste trabalho: Ana Paula, Antonio, André (Chefe), Bicho, Chips, Cristiano (Abóbora), Etiene, Evandro, Fabi, Gonzaga, Jezer, Júlios (JJJ e Ferronato), Klaser, Marcelos (Borba (que, aliás, é o pai do Ugh) e Salizar (S4)), Maurício, Mierlo, Mírian, Otávio, Renato (Mikitow), Root, Rossana, Ruthiano, Scheila, Schramm e Solon. Talvez um pouco menos próximos que estes outros, mas também responsáveis por umas tantas conversas desopilantes, Cadu, Lucas, Hisham e Rafael (Camarão). Obrigado também aos que deveriam estar nesta lista, mas que eu esqueci de incluir...

Obrigado aos professores e ex-professores do PIPCA com quem tive um contato mais próximo durante minha passagem pelo Programa: Marcelo, Ney, Osório e Renata.

Quase que a totalidade deste trabalho foi desenvolvida com o uso de *softwares* livres e/ou de código aberto. Meus agradecimentos aos responsáveis pelo desenvolvimento de ferramentas como Cal 3D, CEGUI, Emacs, GCC, Gimp, Gnuplot, GoboLinux, LaTeX, Lua, Open Scene Graph e Wings 3D. Mais ou menos no mesmo espírito, Martin Fleurent e Ray Larabie criaram, respectivamente, o estilo (*skin*) de elementos da interface gráfica e a fonte utilizada no protótipo implementado.

Diversos algoritmos de planejamento foram avaliados durante a realização deste trabalho. Isto teria sido impossível se os pesquisadores que criaram este algoritmos não tivessem disponibilizado suas implementações.

Este trabalho contou com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Y si no fuese porque imagino…, ¿qué digo imagino?, sé muy cierto, que todas estas incomodidades son muy anejas al ejercicio de las armas, aquí me dejaria morir de puro enojo.

—Don Quijote de la Mancha

RESUMO

Interactive Storytelling (IS) é uma área de pesquisa que busca o desenvolvimento de técnicas que permitam a criação de sistemas computacionais interativos com ênfase em aspectos dramáticos e narrativos. Uma das abordagens utilizadas em pesquisas na área de IS consiste no uso de algoritmos de planejamento para definir a seqüência de eventos que compõe a história. Este trabalho se propõe a investigar técnicas que permitam avançar rumo à solução de três problemas encontrados por trabalhos que seguem a abordagem baseada em planejamento: (i) a abordagem não é inerentemente consistente narrativamente; (ii) o usuário pode levar a história a situações sem saída, em que o algoritmo de planejamento é incapaz de encontrar uma solução para a história; e (iii) uma carga de trabalho muito grande é posta sobre o autor das histórias.

Para enfrentar estes problemas, são apresentados três mecanismos que podem ser incluídos em modelos de IS baseados em planejamento: (i) uma técnica que permite fazer as histórias geradas pelo sistema seguir um arco de tensão definido pelo autor da história; (ii) um mecanismo que busca antever e evitar situações sem saída; e (iii) um método que permite refinar alguns parâmetros da história com base na avaliação de um grupo de usuários de teste, reduzindo a carga de trabalho sobre o autor da história.

As idéias apresentadas no trabalho foram implementadas em um protótipo chamado Fabulator, que exibe as histórias através de gráficos 3D, com personagens articulados. O Fabulator foi utilizado para a realização de experimentos com usuários. Os resultados obtidos indicam que o modelo é capaz de atingir seus objetivos.

Palavras-chave: *Interactive Storytelling*, entretenimento digital interativo, narrativas, planejamento.

ABSTRACT

Interactive Storytelling (IS) is a research area that aims the development of techniques allowing the creation of interactive systems with emphasis in dramatic and narrative aspects. One approach used in previous work in IS is the use of planning algorithms to define the sequence of events that compose the story. This work intends to investigate techniques that allow to advance towards the solution of three problems found in previous work based on the planning approach for IS: (i) this approach is not inherently narratively consistent; (ii) the user can bring the story to a dead end, in which the planning algorithm cannot find a solution to the story; and (iii) an excessively high workload is put on the stories authors.

In order to challenge these problems, three mechanisms that can be added to planning-based IS systems are being proposed: (i) a technique that allows to make the system-generated stories follow a tension arc defined by the story author; (ii) a mechanism that strives to foresee and avoid dead ends; and (iii) a method that allows to fine tune some story parameters based on the evaluation of a group of test users, reducing the workload on the story author.

The ideas presented in this work have been implemented in a prototype application called Fabulator, which displays the stories using 3D graphics and articulated characters. Fabulator was used to perform experiments with users. The results suggest that the model is capable to achieve its goals.

Keywords: Interactive storytelling, interactive digital entertainment, narratives, planning.

LISTA DE FIGURAS

1 Um exemplo parcial de um grafo E/OU. 18
2 O protótipo de Cavazza *et al.* 19
3 Descrição da ação STRIPS `use-rachels-pda`. 19
4 Teatrix, o protótipo de Machado *et al.* 21
5 Telas do sistema de Ciarlini *et al.* 21
6 O sistema Erasmaganza de Crawford. 22
7 A arquitetura do IDtension de Szilas. 23
8 Tela do protótipo de Magerko. 24
9 O Façade, de Mateas e Stern. 25
10 Arco Aristotélico no Façade. 28
11 Exemplos de planos para o objetivo "ter meias e tênis vestidos". 30
12 A ação `GivePresent` escrita em PDDL. 33
13 A ação `ExaminePlace` escrita em PDDL 33
14 Arquitetura do Fabulator. 41
15 Modelo de arco de tensão em histórias do tipo Enigma. 42
16 Respeitando arcos de tensão. 44
17 Antevendo situações sem saída. 46
18 Duas formas de evitar situações sem saída. 47
19 Exemplo de arquivo `Storyworld.lua` para definição de uma história. . . . 51
20 Exemplo de arquivo para definição de um personagem. 52
21 Duas telas do Fabulator 53
22 Lista de ações exibida após clique em um personagem da história. 54
23 Tela com o "questionário" para ajuste dos parâmetros c_p. 55
24 Modo de depuração do Fabulator 56
25 Arco de tensão desejado para *A Estória de Ugh 2*. 60
26 Arco de tensão da história produzida pelo usuário 5. 63
27 Arco de tensão da história produzida pelo usuário 8. 64

28 Arco de tensão da história produzida pelo usuário 7. 65
29 Vista superior do cenário d'*A Estória de Ugh 2*. 95

LISTA DE TABELAS

1 Exemplo de definição do verbo “ofender”. 22
2 Resumo da análise dos algoritmos de planejamento. 38
3 Dados adicionais sobre algoritmos de planejamento. 39
4 Endereços com as implementações dos algoritmos analisados. 39
5 Pistas d’*A Estória de Ugh 2*. 58
6 Avaliação dos usuários quanto aos valores ideais para os parâmetros c_p. 60
7 Discrepância total (E) observada nos experimentos com usuários finais. 61
8 Símbolos da lógica de predicados usados neste trabalho. 77

LISTA DE ABREVIATURAS E SIGLAS

ABL *A Behavioral Language*

CG Computação Gráfica

IA Inteligência Artificial

IPC *International Planning Competition*

IPG *Interactive Plot Generator*

IDS *Iterative Deepening Search*

IS *Interactive Storytelling*

MMORPG *Massively Multiplayer Online Role-Playing Game*

MUD *Multi-User Dungeon*

NPC *Non-Player Character*

OSG *Open Scene Graph*

PDA *Personal Digital Assistant*

PDDL *Planning Domain Description Language*

RPG *Role-Playing Game*

STRIPS *Stanford Research Institute Problem Solver*

SUMÁRIO

1 Introdução **13**

1.1 *Interactive Storytelling* 14

1.2 *Interactive Storytelling* baseada em planejamento 15

1.3 Objetivos deste trabalho 16

1.4 Organização desta dissertação 16

2 Revisão Bibliográfica **17**

2.1 Abordagens baseadas em planejamento 17

2.2 Abordagens morfológicas 20

2.3 Abordagens heurísticas 21

2.4 Contextualização deste trabalho no estado-da-arte 25

3 Fundamentos **27**

3.1 Narrativas 27

3.1.1 Narratologia segundo Mieke Bal 27

3.1.2 Arcos de tensão 28

3.1.3 O enredo mestre “Enigma” 29

3.2 Planejamento 29

3.3 Avaliação de algoritmos de planejamento 31

3.3.1 Avaliando algoritmos de planejamento do ponto de vista de IS 31

3.3.2 Análise dos algoritmos 35

3.3.3 A escolha 37

4 Modelo **40**

4.1 Arquitetura para o desenvolvimento do trabalho 40

4.2 Modelando e respeitando arcos de tensão 42

4.2.1 Modelando arcos de tensão 42

4.2.2 Respeitando arcos de tensão 43

4.3 Antevendo e evitando situações sem saída 45
4.3.1 Antevendo situações sem saída 46
4.3.2 Evitando situações sem saída 47
4.4 Ajustando parâmetros com base na avaliação de usuários 48
4.5 O modelo no contexto da teoria narrativa de Bal 49

5 Protótipo **51**
5.1 Definição de uma história 51
5.1.1 Informações gerais sobre a história 51
5.1.2 Definição do problema de planejamento 52
5.2 Visualização 53
5.3 Interface com o usuário 53
5.4 Planejamento e detecção de situações sem saída 54
5.5 Ajuste de parâmetros com base na avaliação de usuários 55
5.6 Auxílios para depuração e análise de resultados 55

6 Resultados **57**
6.1 A Estória de Ugh 2 57
6.2 Experimentos com usuários 59
6.2.1 Usuários de teste 59
6.2.2 Usuários finais 60
6.2.2.1 Discrepância total 60
6.2.2.2 Análise das histórias resultantes 61
6.2.3 Observações adicionais 65

7 Conclusões **67**
7.1 Trabalhos Futuros 69

Referências **72**

Apêndice A – Símbolos da Lógica de Predicados **77**

Apêndice B – "A Estória de Ugh": Primeiro Ato **78**
Arquivo `ugh.pddl` 78
Arquivo `ugh_act1.pddl` 80

Apêndice C – Linguagem Para Descrição de Problemas de Planejamento **81**

Apêndice D – "A Estória de Ugh 2" **87**

Arquivo `Storyworld.lua` 87

Arquivo `Problem` 87

Cenário 95

1 INTRODUÇÃO

Could you tell a wise man,
By the way he speaks or spells?
Is this more important,
Than the stories that he tells?
— Mike & Brian Hugg

Jogos para computadores e consoles estão estabelecidos, já há vários anos, como a forma mais popular de entretenimento digital interativo. Do ponto de vista tecnológico, o desenvolvimento nesta área é notável, sobretudo no que diz respeito à aplicação de técnicas de áreas como Computação Gráfica (CG), Inteligência Artificial (IA) e Redes de Computadores. Porém, uma análise menos técnica e mais "conceitual", permite observar que os jogos desenvolvidos nas últimas décadas, com raras exceções, têm explorado fundamentalmente os mesmos aspectos: coordenação entre os sistemas motor e visual, resolução de quebra-cabeças e gerenciamento de recursos (CRAWFORD, 2004).

Vem ganhando força, contudo, a idéia de que é possível explorar de forma bastante positiva aspectos geralmente negligenciados por jogos. Em particular, acredita-se que seja possível criar sistemas computacionais interativos fortemente baseados em aspectos dramáticos e narrativos. Mais do que isso, acredita-se que estas idéias podem ser desenvolvidas levando em consideração aspectos tanto da Computação quanto das Artes, de modo a originar uma nova mídia para contadores de histórias e uma nova forma de expressão artística.

Murray (2003) considera que há uma grande variedade de formas para utilizar computadores com propósitos narrativos. Algumas destas formas são relativamente simples, e já são exploradas há vários anos. Um exemplo destas formas são grupos de discussão na Internet sobre seriados da televisão, que, por vezes, são usados como fonte de idéias pelos roteiristas. Outro exemplo são sistemas multi-usuários — desde os antigos *Multi-User Dungeon*s (MUDs) até os modernos *Massively Multiplayer Online Role-Playing Game*s (MMORPGs)—, em que histórias interessantes podem surgir da interação entre os usuários. Formas mais complexas para usar computadores de formas ligadas à narrativa também existem ou podem ser imaginadas. Estas formas vão desde agentes capazes de manter diálogos com o usuário de um sistema computacional até sistemas utilizando Realidade Virtual em que o usuário participa como protagonista de uma história que se desenrola à sua volta.

Para Glassner (2004), a forma ideal para unir aspectos de histórias e jogos são ambientes multi-usuário que estimulem o surgimento de narrativas. Essencialmente, o autor propõe que os usuários sejam colocados em um ambiente virtual tridimensional, no qual cada um deles interpreta um papel. Neste ambiente, devem ocorrer eventos que introduzam obstáculos aos objetivos dos personagens e estimulem a interação entre os usuários. Na visão de Glassner, seria possível, inclusive, que grupos de usuários contratassem "usuários profissionais" para atuar juntamente com eles nos ambientes virtuais, e criar mais situações dramaticamente interessantes.

Crawford (2004) vê as histórias como um meio utilizado pelos seus autores para expor seus pontos-de-vista sobre algum assunto. Se uma história é contada oralmente, para um único ouvinte, o contador da história é capaz de adaptar esta história às reações do espectador, à medida que conta. Desta forma, ele é capaz de transmitir suas idéias de forma muito mais eficiente ao ouvinte. Autores que fazem uso de mídias de massa, como Literatura ou Cinema, são capazes de contar suas histórias para platéias muito maiores. Eles perdem, porém, a capacidade de adaptar a história à audiência. De acordo com Crawford, sistemas computacionais interativos podem ser capazes de unir o melhor destas duas alternativas. Em outras palavras, "sistemas contadores de histórias" podem permitir que autores atinjam um grande número de espectadores e adaptem a história contada a cada um deles.

Existe, portanto, uma crença de que sistemas computacionais interativos podem se tornar uma nova mídia para expressão. Porém, para que esta visão possa ser concretizada, é preciso vencer uma nova série de desafios tecnológicos e científicos. No que diz respeito à criação de sistemas interativos centrados em narrativas, um dos grandes desafios consiste em ser capaz de gerar histórias consistentes e interessantes, que possam ser significativamente diferentes dependendo das ações executadas pelo usuário. Conciliar estas características de maneira satisfatória é uma tarefa extremamente difícil e uma das principais motivações da área de *Interactive Storytelling* (IS).

1.1 *Interactive Storytelling*

Interactive Storytelling é o nome normalmente dado à relativamente nova área de pesquisa que busca desenvolver técnicas que permitam a criação de sistemas interativos com foco em narrativas. Um dos problemas fundamentais enfrentado pela área é a conciliação entre interatividade e consistência narrativa.

Interação é metaforicamente definida por Crawford como "um processo cíclico no qual dois atores alternadamente escutam, pensam e falam." (CRAWFORD, 2003). No contexto de um sistema centrado em narrativas, isto significa que um sistema só pode ser considerado interativo se for capaz de receber ordens do usuário, processá-las de modo a determinar uma resposta, e transmitir esta resposta de volta ao usuário. E, como o processo é cíclico, o usuário deverá ser capaz de compreender a resposta do sistema e pensar numa nova ação a ser executada, de modo a reiniciar o ciclo.

Consistência narrativa é a expressão utilizada neste trabalho para designar a qualidade de seqüências de eventos que correspondam a boas histórias. Esta é uma definição bastante subjetiva, pois não existe uma regra que permita quantificar o quão boa é uma narrativa. A bibliografia referente a narrativas, contudo, cita aspectos que são considerados ingredientes importantes para boas histórias (ARISTÓTELES, 1966; TOBIAS, 1993; FIELD, 1995). Neste trabalho, está se considerando que quanto mais destes aspectos forem observados numa história, mais consistente narrativamente ela é. Vendo este conceito sob a ótica de sistemas computacionais, um sistema seria dito narrativamente consistente se for capaz de gerar seqüências de eventos compreendidas por seus usuários como boas histórias.

Dadas as definições de interatividade e consistência narrativa, é possível compreender mais claramente a dificuldade de conciliar estes dois aspectos. Em mídias tradicionais, sem interatividade, a consistência narrativa pode ser totalmente garantida pelo autor da história. Em IS, o

"espectador" é um agente ativo na história, e pode potencialmente pôr a consistência narrativa em risco. Claramente, há um conflito entre consistência narrativa e interatividade. Em um sistema de IS, deseja-se que as ações do usuário tenham influência real e profunda no desenrolar da trama, mas, ao mesmo tempo, deseja-se impedir que ele seja capaz de transformar a história em algo sem sentido (AYLETT, 1999; GREEFF; LALIOTI, 2001).

1.2 *Interactive Storytelling* baseada em planejamento

Diversas abordagens vêm sendo exploradas por diferentes grupos de pesquisa em IS buscando solucionar o problema da conciliação de interatividade e consistência narrativa. (O Capítulo 2 apresenta uma perspectiva da área.) Uma destas alternativas baseia-se no uso de algoritmos de planejamento para determinação dos eventos que compõem a história. Uma das vantagens bastante interessantes desta abordagem é a possibilidade de atender com relativa simplicidade aos requisitos de interatividade esperados de um sistema de *Interactive Storytelling*. Contudo, IS baseada em planejamento também apresenta alguns inconvenientes.

O primeiro inconveniente decorre do fato de que nem todo plano produzido por um algoritmo de planejamento corresponde a uma boa história. Ou, utilizando a terminologia introduzida anteriormente, o uso de algoritmos de planejamento não garante a consistência narrativa. É preciso, portanto, que modelos de IS baseados nessa abordagem incluam mecanismos adicionais que permitam satisfazer este requisito.

Uma segunda limitação do uso de algoritmos de planejamento diz respeito à possibilidade do jogador levar a história para uma "situação sem saída". Uma situação sem saída é uma para a qual seja impossível gerar um plano que conduza a história ao desfecho imaginado por seu autor. Neste caso, a história precisa ser encerrada prematuramente, pois o algoritmo de planejamento não será capaz de determinar um meio de encerrá-la devidamente. Assim, idealmente, modelos de IS baseados em planejamento devem incorporar meios de evitar que situações como esta ocorram.

Por fim, um terceiro problema que merece destaque é a enorme carga de trabalho posta sobre o autor de uma história para um sistema de IS. Este problema não é exclusivo da abordagem baseada em planejamento: em geral, trabalhos que oferecem bons resultados exigem que o autor das histórias especifique uma grande quantidade de dados que servem de entrada para o sistema. Em alguns casos, isto pode chegar a inviabilizar a criação de histórias de comprimento razoável. Desta forma, é preciso buscar formas de aliviar a carga de trabalho dos autores de histórias para IS.

Resumindo, os modelos de IS baseados em planejamento possuem ao menos três importantes problemas que precisam ser enfrentados:

1. Eles *não* são inerentemente consistentes narrativamente.
2. Eles podem chegar a situações sem saída.
3. Eles impõem uma carga de trabalho muito grande sobre os autores das histórias.

1.3 Objetivos deste trabalho

O principal objetivo deste trabalho é investigar mecanismos que permitam avançar em direção à solução dos três problemas enfrentados na IS baseada em algoritmos de planejamento apresentados na seção anterior. Esta investigação utilizará como plataforma um modelo de IS desenvolvido em trabalhos anteriores (BARROS, 2004; BARROS; MUSSE, 2005). Deseja-se ainda que o modelo desenvolvido durante a execução do trabalho esteja fundamentado em teorias desenvolvidas no contexto de áreas tradicionalmente ligadas às narrativas. De forma mais específica, são objetivos deste trabalho:

1. Propor um método que busque aumentar a consistência narrativa das histórias geradas. O método proposto busca fazer com que as histórias sigam um arco de tensão especificado pelo seu autor.[1]
2. Desenvolver um mecanismo que procure evitar que a história chegue em situações sem saída. Este mecanismo deve ter a preocupação de ser "integrado à história", ou seja, sempre que possível ele deve intervir na história de uma forma que seja justificada pelos eventos anteriores.
3. Prover um meio que permita reduzir um pouco da carga de trabalho posta sobre o autor de uma história para um sistema de IS.
4. Desenvolver um protótipo que implemente os três pontos descritos nos ítens anteriores.
5. Utilizar o protótipo para avaliar os resultados obtidos.

1.4 Organização desta dissertação

Esta dissertação está dividida em 7 capítulos. O próximo apresenta uma revisão de trabalhos na área de IS e contextualiza o presente trabalho no estado-da-arte. O Capítulo 3 introduz alguns conceitos necessários à compreensão do restante do trabalho. Em seguida, o Capítulo 4 descreve o modelo de IS proposto. O protótipo que implementa este modelo é descrito no Capítulo 5. Na seqüência, o Capítulo 6 apresenta resultados obtidos. Conclusões encerram o texto no Capítulo 7.

[1] Arcos de tensão, descritos na Seção 3.1.2, são uma forma de especificar de que forma o nível de tensão vivenciado pelo espectador de uma história varia com o tempo.

2 REVISÃO BIBLIOGRÁFICA

En resolución, él se enfrescó tanto em su lectura, que se le passaban las noches leyendo de claro em claro, y los dias de turbio en turbio; y así, del poco dormir e mucho leer, se le secó el celebro de manera que vino a perder el juicio.
— Miguel de Cervantes

Este capítulo tem dois objetivos: descrever os principais trabalhos na área de *Interactive Storytelling* (IS) encontrados na literatura e contextualizar o presente trabalho no estado-da-arte. Ao buscar estes objetivos, contudo, deve-se levar em consideração uma característica distinta da área na qual este trabalho encontra-se inserido: as tecnologias são desenvolvidas e aplicadas com propósitos artísticos e/ou expressivos.

Assim, objetivos e convicções artísticas guiam muitas decisões técnicas tomadas por pesquisadores da área. De fato, segundo Mateas, "para ser efetiva, a pesquisa técnico-artística deve continuamente avaliar se a tecnologia está servindo aos objetivos artísticos e expressivos" (MATEAS, 1997). Desta forma, embora este capítulo concentre-se em aspectos técnicos dos trabalhos descritos, busca-se dar uma noção dos objetivos artísticos de cada um deles.

Analisando a literatura da área, é possível observar que há diversas abordagens sendo exploradas por diferentes grupos de pesquisa. As próximas seções apresentam os trabalhos mais representativos de cada uma das principais abordagens utilizadas. É importante acrescentar que a divisão dos trabalhos em "grupos de abordagens" utilizado aqui é, de certa forma, arbitrária. Outras divisões seriam possíveis e alguns trabalhos possuem características de mais de uma abordagem.

2.1 Abordagens baseadas em planejamento

Alguns trabalhos em IS são fortemente baseados no uso de algum tipo de algoritmo de planejamento. Esta seção descreve trabalhos com esta característica.

O trabalho desenvolvido por Cavazza, Charles e Mead (CAVAZZA; CHARLES; MEAD, 2001, 2002b, 2002a; MEAD; CAVAZZA; CHARLES, 2003) tem como objetivo permitir que histórias com uma estrutura narrativa bem definida tenham seu final alterado em conseqüência da interação do usuário. Os autores vislumbram um sistema em que o usuário possa assistir a uma história e, caso deseje, influenciar o desenrolar da trama. Com o objetivo de manter a narrativa gerada coerente com o enredo imaginado pelo autor da história, as possibilidades de interação são intencionalmente reduzidas: o papel do usuário se limita a mover objetos pelo cenário e sugerir comportamentos aos personagens.

O autor da história descreve o comportamento de cada um dos personagens principais através de um grafo E/OU (RICH; KNIGHT, 1991), que define hierarquicamente os possíveis

planos que o personagem pode seguir para atingir seu objetivo na narrativa. As ações efetivamente executadas pelos personagens durante a história são computadas por um algoritmo de planejamento que opera sobre este grafo. A Figura 1 reproduz parte de um grafo E/OU que codifica os meios pelos quais o protagonista de uma história poderia cumprir seu objetivo: marcar um encontro com a personagem chamada Rachel. Como se vê, o primeiro nível do grafo subdivide a história em quatro etapas distintas (obter informações, ganhar a afeição, ficar a sós e fazer o convite), enquanto os níveis inferiores acrescentam as possíveis variações no enredo.

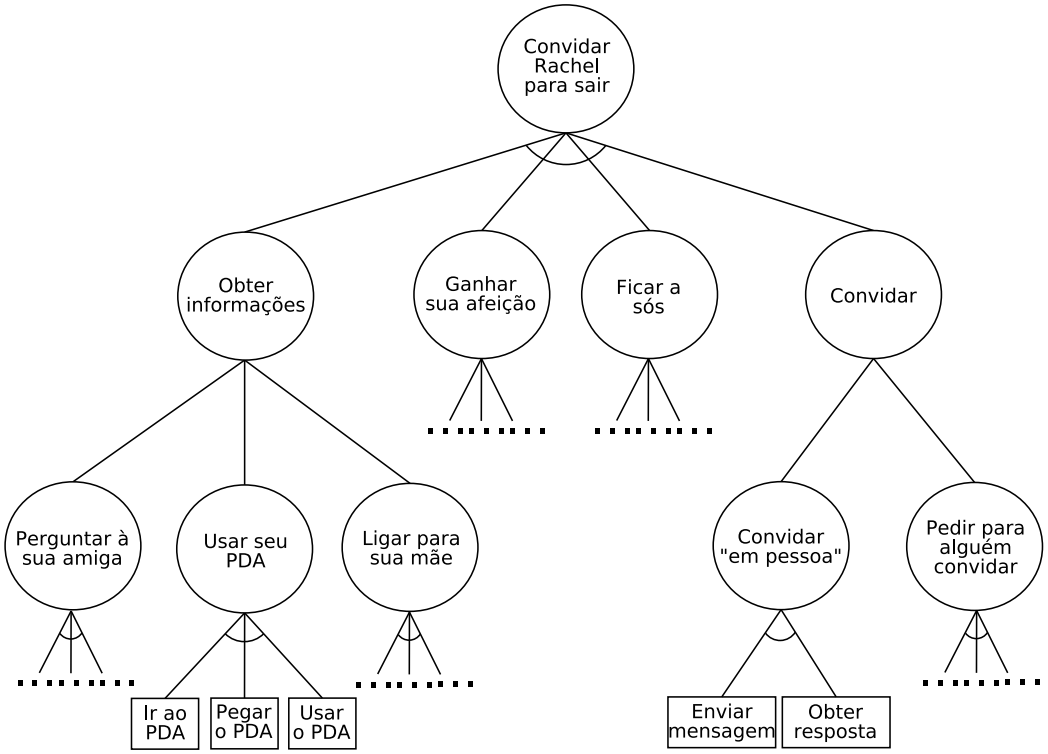


Figura 1: Um exemplo parcial de um grafo E/OU (Adaptado de (CAVAZZA; CHARLES; MEAD, 2002a)).

Os autores deste trabalho afirmam que boa parte do interesse dramático nas histórias geradas pelo modelo surge a partir da interação entre os planos de diferentes personagens. Infelizmente, porém, este processo é discutido apenas superficialmente. O protótipo implementado possui visualização 3D baseada no motor do jogo *Unreal Tournament*, conforme mostra a Figura 2.

Este mesmo grupo, com a colaboração de Lozano e Bisquerra, explorou em (CHARLES et al., 2003) a possibilidade de utilizar uma linguagem baseada em STRIPS (RUSSELL; NORVIG, 2002) e algoritmos de planejamento para a geração das histórias. Em STRIPS, o estado do mundo é representado por um conjunto de predicados, e cada ação que pode ser executada pelo sistema é descrita através de um conjunto de pré-requisitos para sua execução e um conjunto de alterações no estado do mundo que a sua execução provoca. Como exemplo, a Figura 3 mostra a definição da ação `use-rachels-pda`. O contexto em que esta ação é utilizada é o mesmo descrito anteriormente: o protagonista da história marcando um encontro com a personagem Rachel. A ação é utilizada pelo protagonista para buscar no *Personal Digital Assistant* (PDA) de Rachel alguma idéia para presenteá-la. A definição da ação determina que ela só pode ser executada caso o protagonista precise de uma idéia para presente (ou seja, a proposição `need-gift-idea` deve fazer parte do estado do mundo) e o PDA de Rachel deve estar livre

Figura 2: O protótipo de Cavazza *et al.* (CHARLES; CAVAZZA, 2004)

(`pda-free`). Se a ação for executada, o protagonista da história saberá que Rachel gostaria de ganhar flores (a proposição `info-gift-flowers` é adicionada ao estado do mundo) e não mais precisará de uma idéia para presente (`need-gift-idea` é removido do estado do mundo).

```
(def-operator: use-rachels-pda
   (make-operator
      :pre-conditions '(:need-gift-idea)
      :exe-condition :pda-free
      :add-list '(:info-gift-flowers)
      :delete-list '(:need-gift-idea)))
```

Figura 3: Descrição da ação STRIPS `use-rachels-pda`. (CHARLES et al., 2003)

Young, Riedl e Saretto (YOUNG, 1999, 2001; RIEDL; SARETTO; YOUNG, 2003) propõem uma arquitetura para IS chamada Mimesis, em que a história é representada por um plano gerado por um algoritmo de planejamento. Uma das características principais desta arquitetura é a inclusão de mecanismos de *detecção de exceções* e *mediação*. Exceções são ações executadas pelo usuário que comprometem a cadeia de causas e efeitos representada no plano que modela a história. Mecanismos de mediação são meios utilizados para tratar as exceções que ocorrem durante a história. Duas formas de mediação são propostas pelos autores:

- *Acomodação:* O plano é modificado de modo que a ação do usuário se encaixe na cadeia de causas e efeitos sem prejuízo para a história. Por exemplo: o plano original previa que um ladrão iria arrombar um cofre e roubar o dinheiro que estava ali guardado. Porém, antes da chegada do ladrão, o usuário abriu o cofre e esqueceu-o aberto (sem pegar seu conteúdo). Neste caso, o plano poderia ser alterado de forma simples: o ladrão não mais precisaria arrombar o cofre, bastaria pegar o dinheiro.
- *Intervenção:* Caso a ação do usuário seja particularmente danosa à história, o sistema pode fazer com que ela falhe, de modo que o estado do mundo permaneça inalterado.

> Um exemplo simples citado pelos autores consiste numa tentativa por parte do usuário de matar algum personagem importante da história. Neste caso, o sistema intervém, fazendo com que a arma falhe ou que o tiro "passe raspando" pelo alvo.

Com relação ao Mimesis, também é interessante notar que busca-se levar aspectos dramáticos em consideração durante a geração do plano. Assim, as ações que compõem a história são escolhidas de modo a garantir, por exemplo, a existência de conflito entre protagonista e antagonista e o respeito ao arco aristotélico (Seção 3.1.2). A visualização das histórias, em 3D, é feita através do motor do jogo *Unreal Tournament*.

2.2 Abordagens morfológicas

No final da década de 1920, motivado por encontrar uma forma de categorizar obras literárias, o formalista Vladimir Propp realizou um estudo sobre a morfologia dos contos de fadas (PROPP, 1983), ou seja, buscou descobrir que partes compõem estas obras e como estas partes se relacionam entre si. Analisando um corpus composto de cem contos de fadas populares russos, Propp verificou que todos os textos eram compostos por um número pequeno de *funções narrativas* organizadas em padrões bastante rígidos (uma função narrativa é definida como uma "acção de uma personagem, definida do ponto de vista de seu significado no desenrolar da intriga" (PROPP, 1983, p. 60)).

Alguns trabalhos de IS utilizam os resultados de Propp como base. Neste sentido, podem ser citados os trabalhos de Machado, Paiva e Brna (2001) e Grasbon e Braun (2001). Ambos baseiam-se na idéia de que o autor da história deve definir um conjunto de cenas, associando cada uma delas a uma função narrativa. Durante a execução, o modelo garante que as cenas exibidas sigam os padrões descritos por Propp, de modo a garantir a consistência da história gerada. A Figura 4 mostra uma tela do sistema de Machado *et al.*, que é destinado a crianças. O protótipo de Grasbon e Braun utiliza interface em modo texto.

Com relação a esta abordagem, há um ponto que deve ser destacado: o próprio Propp comenta que todas as suas conclusões são válidas apenas para contos de fadas do folclore: histórias de outros gêneros narrativos, ou mesmo "contos de fadas modernos" não irão necessariamente seguir os padrões que ele descreveu. Como conseqüência, basear um trabalho de IS na "Morfologia do Conto" de Propp implica em limitar o universo de histórias que podem ser geradas.

Pozzer (2005) e Ciarlini *et al.* (2005) descrevem um sistema de IS projetado para aplicação em TV interativa. Na implementação apresentada, a geração e a visualização da história acontecem em momentos distintos, e toda interação ocorre durante o processo de geração da história. Assim, num primeiro momento, o usuário interage com o sistema a fim de produzir um roteiro que lhe agrade para, em seguida, assistir à sua dramatização (em 3D, como mostrado na Figura 5) como se fosse um filme.

A geração dos enredos é realizada com o uso do *Interactive Plot Generator* (IPG), uma ferramenta desenvolvida pelos próprios autores com base nas funções narrativas de Propp. O IPG permite que o usuário, interativamente, explore as possíveis variações que a história pode ter, sem desrespeitar os padrões morfológicos descritos por Propp.

Figura 4: Teatrix, o protótipo de Machado *et al.* (2001)

Figura 5: Telas do sistema de Ciarlini *et al.* (2005)

2.3 Abordagens heurísticas

Nos trabalhos descritos até aqui, a definição dos eventos que compõem a história é baseada principalmente no uso de algoritmos propostos em outras áreas da Computação (normalmente, IA no caso de algoritmos de planejamento, e algoritmos de grafos no caso das abordagens morfológicas). Em contraste a estes, há um grupo de trabalhos que determina os acontecimentos da história através de procedimentos heurísticos. Estes procedimentos são criados especialmente para IS, mas carecem de uma definição formal mais rigorosa. São criados de modo que, na prática, apresentem resultados satisfatórios, mas em geral não oferecem nenhum tipo de garantia quanto à sua capacidade de solucionar os problemas para os quais são criados. Os trabalhos descritos a seguir utilizam esta estratégia.

Crawford vem trabalhando com IS em sua empresa Erasmatazz há aproximadamente uma década, buscando criar ferramentas de autoria acessíveis a pessoas sem perfil técnico. Em seu sistema de autoria, o Erasmatron, o autor da história define um conjunto de *verbos* (ações que

podem ser executadas pelos personagens). Cada verbo possui associado uma lista de papéis que podem ser assumidos por diferentes personagens durante a sua execução. Por exemplo (conforme ilustra a Tabela 1), um verbo "ofender" poderia ter associado dois papéis: o personagem ofendido e uma testemunha da ofensa. Para cada um destes papéis, há uma lista de verbos representando as possíveis reações. Assim, em resposta à ofensa, o ofendido poderia "responder à ofensa" ou "baixar a cabeça". Para cada uma destas possíveis reações, o autor da história deve definir uma *equação de inclinação*, que determina a probabilidade de que esta seja a escolhida dentre as opções.

Tabela 1: Exemplo de definição do verbo "ofender".

Papel	**Possíveis Reações**
Ofendido	Responder à ofensa Baixar a cabeça
Testemunha	Defender o ofendido Ajudar a ofender Ignorar

O sistema dispõe ainda de alguns mecanismos para gerar automaticamente uma série de comportamentos de personagens que são comuns a todas as histórias. Exemplos desses comportamentos são locomover-se entre diferentes cenários e conversar com outros personagens, incluindo a possibilidade de espalhar boatos e mentiras.

O sistema de visualização das histórias, Erasmaganza, possui uma visualização 2D que enfatiza as expressões faciais dos personagens (Figura 6). Uma descrição da tecnologia do Erasmatron é encontrada em (CRAWFORD, 2004) e as premissas nas quais o modelo de Crawford está baseado estão descritas em (CRAWFORD, 1999).

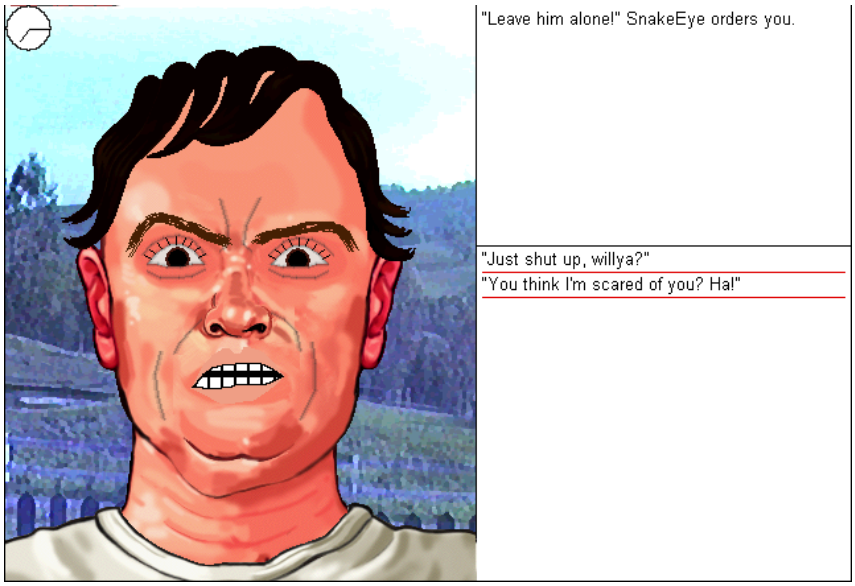


Figura 6: O sistema Erasmaganza de Crawford.

IDtension, o modelo proposto por Szilas (1999, 2001, 2003), busca permitir que as ações do usuário alterem profundamente as histórias geradas, sem, no entanto, violar os princípios

dramáticos fundamentais. A arquitetura proposta pelo autor (Figura 7), é baseada em uma *lógica narrativa*, que é utilizada para determinar todas as ações possíveis para cada personagem. Outro módulo do sistema, o *narrador virtual*, determina quais ações são realmente executadas, baseando a decisão em critérios como consistência, conflito e surpresa, que são desejáveis em narrativas. Para gerar conflito, por exemplo, o modelo utiliza um sistema baseado em *valores* morais (uma ação "roubar" poderia ser avaliada como negativa se comparada com o valor "honestidade"). O modelo busca gerar conflito promovendo situações que obriguem os personagens a executar ações que vão contra seus valores para conseguir atingir seus objetivos. Durante a execução, o sistema constrói um modelo de usuário, com o objetivo de identificar, por exemplo, quais são os valores mais importantes para o usuário. O protótipo que implementa este modelo utiliza uma interface baseada em texto e exibe ao usuário as suas opções de ação em um menu.

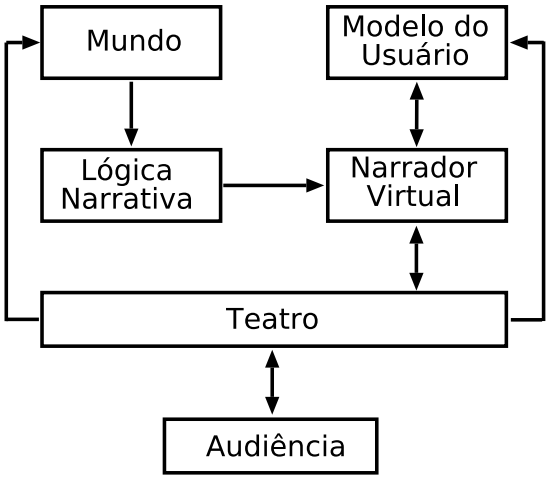


Figura 7: A arquitetura do IDtension de Szilas. Adaptado de (SZILAS, 2003).

O trabalho de Magerko (2002, 2003, 2005) tem como objetivo técnico guiar heuristicamente as interações do usuário dentro de um enredo abstratamente definido pelo autor da história. Artisticamente, deseja-se oferecer ao usuário uma experiência rica, flexível e narrativamente intensa, na qual ele controla o protagonista da história. Além disso, deseja-se que o autor seja capaz de "contar a história que deseja contar". Para atingir estes objetivos, Magerko propõe uma arquitetura em que a história é representada pelo autor como um conjunto de "*plot points*" parcialmente ordenados. Cada *plot point* é composto de três elementos:

- Um conjunto de pré-requisitos que devem ser observados no mundo para que o *plot point* possa ser utilizado.
- Um conjunto de ações que devem ser representadas durante o *plot point*, como a execução de animações ou a interpretação de algum diálogo específico pelos *Non-Player Character*s (NPCs).
- Restrições temporais especificando uma janela de tempo na história durante o qual o *plot point* deve ser utilizado. Isto permite ao autor especificar o ritmo com que a história deve avançar.

Em tempo de execução, o sistema se encarrega de fazer com que os *plot points* sejam utilizados respeitando a ordem parcial, os pré-requisitos e as restrições temporais. Como o usuário

tem liberdade de agir durante a história, o sistema introduz um módulo chamado Diretor que tem como função controlar os NPCs e o ambiente de modo a persuadir o usuário a manter-se dentro do enredo pré-estabelecido pelo autor da história. Assim, caso seja necessário, por exemplo, que o usuário esteja presente em um determinado local do cenário em um certo momento, o Diretor pode ordenar que algum NPC diga alguma frase que desperte a curiosidade do usuário em ir ao local em questão.

Outra contribuição do trabalho é a utilização de um modelo estatístico usado para prever as ações que serão executadas pelo usuário. Com base nestas previsões, o Diretor é capaz de intervir na história de forma mais antecipada e sutil, de modo a guiar o jogador pelo enredo pré-estabelecido sem que ele perceba isto. A visualização das histórias geradas é baseada no motor do jogo *Unreal Tournament*, como ilustrado na Figura 8.

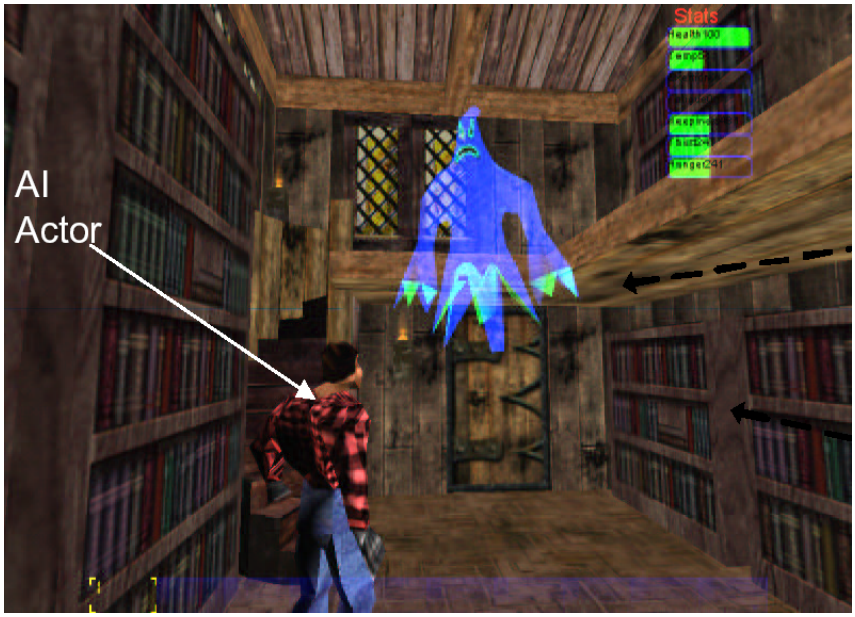


Figura 8: Tela do protótipo de Magerko (2002).

O Façade, desenvolvido por Mateas e Stern (2003, 2005b), é considerado o primeiro sistema de IS totalmente funcional a ser disponibilizado para o público em geral. O objetivo dos autores era envolver o usuário em histórias relativamente curtas, com cerca de 20 minutos de duração, mas com ação dramática unificada e emocionalmente intensa. Os autores desejavam ainda que as ações do jogador tivessem influência significativa em quais eventos ocorrem na história. De fato, consideram que o usuário deve "jogar a história" várias vezes, explorando variações para apreciar o Façade devidamente. Por fim, desejava-se que o jogador pudesse interagir a qualquer momento, sem que houvessem pontos em que sua participação fosse explicitamente solicitada.

No Façade, o usuário interpreta o papel de um velho amigo de Grace e Trip, um casal que o convida para um jantar. Em pouco tempo, fica evidente que o casamento dos amigos está ruindo, e as ações do usuário durante as discussões que surgem podem ser decisivas para o desfecho dos fatos. Utilizando o teclado e o *mouse*, o usuário é capaz de realizar ações como mover-se pelo apartamento de Grace e Trip, manipular objetos e abraçar ou beijar os amigos. A maior parte da interação, contudo, se dá por meio de linguagem natural (MATEAS; STERN, 2004b): caso o usuário deseje falar alguma frase para os NPCs, basta digitá-la e teclar ENTER.

Internamente, o Façade está organizado como uma coleção de *beats*, pequenos fragmen-

tos da história, com cerca de um ou dois minutos de duração (mais curto, portanto, que uma cena típica do Cinema). A seqüência com que os *beats* são utilizados é determinada por um módulo chamado *Drama Manager*, que toma esta decisão levando em consideração tanto as ações do usuário quanto uma série de informações que são associadas a cada *beat* pelos autores da história.

O comportamento dos NPCs é programado em uma linguagem chamada *A Behavioral Language* (ABL) (MATEAS; STERN, 2004a). Esta linguagem, permite especificar seqüências de ações que os personagens podem executar para atingir determinados objetivos, possibilitando, inclusive, programar objetivos compartilhados entre diferentes personagens, que são capazes de cooperar para atingi-lo. Mais de 100 mil linhas de código ABL são utilizadas no Façade. O Façade possui visualização 3D baseada em um *rendering* não-fotorealístico que oferece um destaque às expressões dos personagens, como ilustrado na Figura 9.

Figura 9: O Façade, de Mateas e Stern (2003).

2.4 Contextualização deste trabalho no estado-da-arte

Este trabalho está inserido no grupo de trabalhos baseados em algoritmos de planejamento: o modelo utilizado como base para as investigações propostas é o de Barros e Musse (2005), que por sua vez é baseado no trabalho de Charles *et al.* (2003). A principal diferença do presente trabalho em relação a seus predecessores diretos é introdução de mecanismos que buscam minimizar três problemas existentes nesta área, descritos na Seção 1.2.

Uma das propostas deste trabalho é fazer com que o arco de tensão das histórias geradas pelo sistema siga um arco determinado pelo autor da história. Uma solução para este problema é proposta por Mateas e Stern para o sistema Façade (MATEAS; STERN, 2003), mas num contexto de IS baseado em uma abordagem heurística. O grupo de Michael Young menciona ter tratado este problema na abordagem baseada em planejamento, mas a literatura consultada não fornece detalhes sobre o método proposto (YOUNG, 1999, 2001; RIEDL; SARETTO; YOUNG, 2003).

Para tratar o problema da história atingir situações sem saída, Young propôs mecanismos de intervenção (YOUNG, 1999). Esta solução é efetiva no sentido de que é capaz de eliminar o problema por completo, mas pode ser frustrante para o usuário, pois é baseada na idéia de efetivamente ignorar algumas de suas ações. A solução proposta neste trabalho não é capaz de eliminar o problema por completo, mas, quando aplicável, espera-se que seja capaz de solucionar o problema de uma forma mais integrada à história, reduzindo a frustração do usuário. A solução aqui proposta é de certa forma semelhante aos mecanismos de predição do comportamento do usuário descritos por Magerko (2005), mas possui uma diferença importante: o trabalho de Magerko assume a existência de um roteiro pré-definido de forma consideravelmente mais rígida do que no presente trabalho.

Por fim, todas as tentativas de simplificação da tarefa de autoria para IS encontradas na bibliografia são baseadas na construção de interfaces para autores de histórias (CRAWFORD, 2004; DONIKIAN; PORTUGAL, 2004). A alternativa proposta neste trabalho utiliza a avaliação de versões preliminares da história por um grupo de usuários de teste. Esta avaliação é usada para ajustar parâmetros que tenham sido definidos inicialmente pelo autor da história. Esta proposta é, portanto, significativamente diferente das alternativas propostas na literatura.

Detalhes sobre as soluções propostas por este trabalho são apresentados no Capítulo 4.

3 FUNDAMENTOS

In order to make an apple pie from scratch, you must first create the universe.
— Carl Sagan

Este capítulo, dividido em três seções, aborda tópicos relacionados a áreas nas quais este trabalho está fundamentado. A primeira seção apresenta alguns elementos de áreas relacionadas à Narrativa que serão mencionados em outras partes deste trabalho. A Seção 3.2 descreve os aspectos relacionados a planejamento que são fundamentais para uma devida compreensão deste texto. O capítulo encerra com seção dedicada a uma análise realizada em diversos algoritmos de planejamento existentes, que teve como objetivo selecionar qual algoritmo seria utilizado para o desenvolvimento deste trabalho.

3.1 Narrativas

Conforme discutido na Introdução, uma das características desejadas em sistemas de *Interactive Storytelling* é a consistência narrativa. Para que este objetivo possa ser atingido, é necessário buscar conhecimento em áreas que estudam narrativas. As subseções a seguir descrevem alguns elementos destas áreas que serão utilizados adiante neste trabalho para contextualizá-lo dentro de teorias narrativas e justificar certas decisões tomadas.

3.1.1 Narratologia segundo Mieke Bal

Esta seção apresenta algumas definições de aspectos relacionados aos textos narrativos, apresentadas por Mieke Bal em sua teoria narrativa (BAL, 1998). Primeiramente, cabe discutir a aplicabilidade desta teoria em uma mídia como IS. A princípio, Bal considera como "texto narrativo" apenas textos escritos, mas reconhece que sua teoria é capaz de explicar aspectos narrativos de outras mídias, como histórias em quadrinhos e cinema. Por isso, considera-se que esta teoria pode ser utilizada como base para trabalhos na área de IS. De fato, na bibliografia é possível encontrar trabalhos que utilizaram esta teoria para fundamentar decisões (YOUNG, 1999)).

Na terminologia usada no trabalho de Bal, um *ator* é um agente (não necessariamente humano) que executa ações. Um *acontecimento* é a transição de um estado a outro, ou seja, um evento ou ação que faça a narrativa progredir. Para que um texto narrativo seja estudado, é útil analisá-lo como sendo composto de três camadas:

- *Fábula* é o nível mais baixo de um texto narrativo, e é definido pela autora como "uma série de acontecimentos lógica e cronologicamente relacionados que os atores causam

ou experimentam". Em outros termos, a fábula é uma seqüência de eventos e ações relacionadas entre si que ocorre no universo da narrativa.

- *História*, a segunda camada, é "uma fábula apresentada de uma certa maneira". Durante o processo de conversão de uma fábula em história, podem ocorrer diferentes "transformações", como reordenação dos acontecimentos, acréscimo de detalhes e definição do ponto de vista em que a narrativa será relatada.

- *Texto*, por fim, é a camada em que "um agente narra uma história", e corresponde à forma final do texto narrativo. No caso de um texto escrito, ela representa a própria redação do texto, as palavras e frases escolhidas pelo autor para que o narrador relate os acontecimentos.

3.1.2 Arcos de tensão

O conflito é um elemento fundamental nas narrativas: não é possível criar um enredo interessante sem obstáculos entre o protagonista e seus objetivos (TOBIAS, 1993). O conflito é a fonte de tensão da história, e a tensão é uma variável cuidadosamente controlada pelos autores de histórias: dependendo da sensação que os autores pretendem passar à sua platéia, eles fazem a tensão aumentar ou diminuir à medida que a trama se desenrola.

O nível de tensão da história em função do tempo forma uma curva, que recebe o nome de *arco de tensão*. O arco de tensão mais comumente utilizado é o Arco Aristotélico (MATEAS; STERN, 2005a), composto de um período de crescimento da tensão, um clímax e um período final de redução da tensão. Existe uma certa liberdade, contudo, para definir arcos diferentes. Como um exemplo, Zagalo (2004) comenta que, em filmes produzidos nas últimas décadas, é freqüente a sobreposição de vários pequenos arcos de tensão secundários sobre um arco principal.

No contexto de IS, há alguns relatos de tentativas de fazer as histórias geradas seguirem arcos de tensão, como no trabalho de Young (1999) e, de forma mais explícita, no Façade de Mateas e Stern (2003), que procura seguir um arco como o da Figura 10.

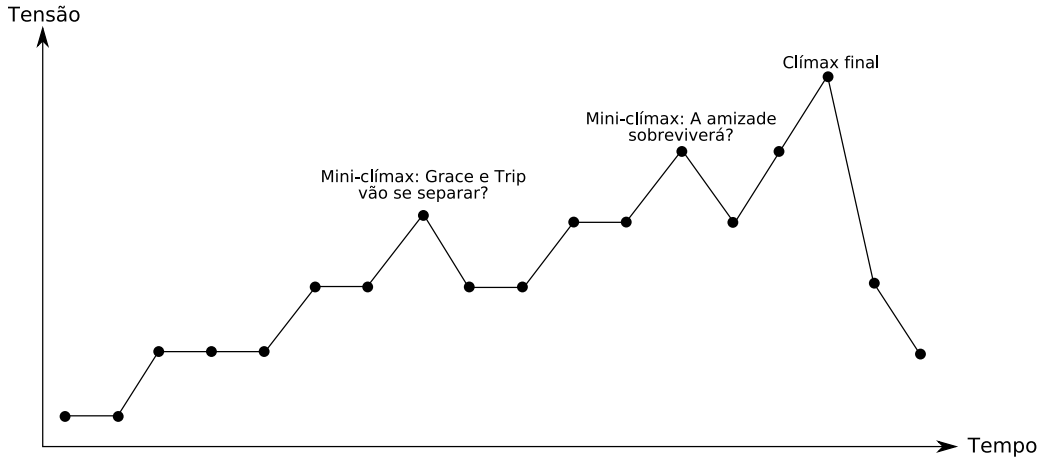


Figura 10: Arco Aristotélico no Façade, incluindo dois "sub-clímaxes" (Adaptado de (MATEAS; STERN, 2003)).

3.1.3 O enredo mestre "Enigma"

Histórias podem ser agrupadas de acordo com suas características e, com efeito, diferentes categorizações foram propostas ao longo dos tempos. Tobias, por exemplo, apresenta uma classificação de histórias em vinte *enredos mestres* (TOBIAS, 1993). Um destes enredos mestres é o Enigma (*riddle*), que tem como principais representantes as histórias de detetive. De acordo com o autor, histórias baseadas neste enredo mestre devem ser organizadas em três atos, que devem conter os seguintes elementos:

1. Apresentação de fatos genéricos, como pessoas, locais e eventos.
2. Revelação de fatos específicos, de uma descrição detalhada de como as pessoas, locais e eventos estão relacionados entre si.
3. A solução do enigma, em que a verdade sobre o fato obscuro é revelada.

A tensão é um ingrediente essencial para uma boa narrativa, e segundo Tobias, em histórias deste tipo, "a tensão deve vir do conflito entre o que aconteceu em oposição ao que parece ter acontecido." Infelizmente, porém, o autor não faz comentários mais específicos sobre o arco de tensão neste tipo de narrativa.

Esta dissertação concentra-se no tratamento de histórias que seguem enredos do tipo Enigma. Em particular, o modelo de arco de tensão descrito na Seção 4.2.1 foi desenvolvido para narrativas deste gênero. Da mesma forma, a história criada para obtenção de resultados e subseqüente avaliação (*A Estória de Ugh 2*, Seção 6.1) segue este mesmo enredo mestre.

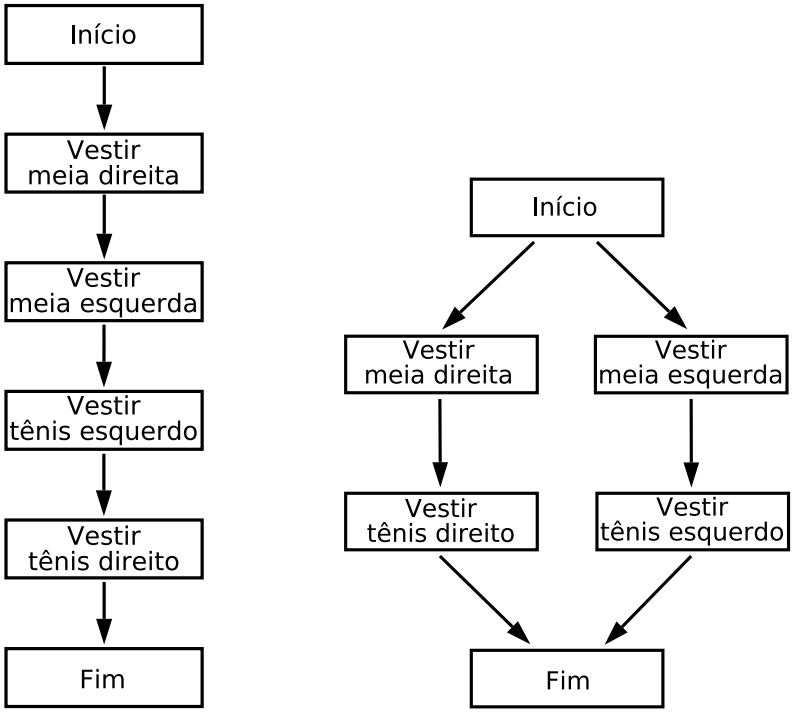
3.2 Planejamento

Planejamento é o nome dado à tarefa que consiste em encontrar uma seqüência de ações que permita atingir um objetivo.[1] Daí vem naturalmente a definição de *plano*: uma seqüência de ações que permite atingir um objetivo. Planos podem ser totalmente ou parcialmente ordenados, como pode ser visto na Figura 11. Um *plano totalmente ordenado* é aquele em que, para qualquer par de ações distintas que façam parte deste plano, está especificado qual delas deve ser executada antes. Em contraste, um *plano parcialmente ordenado* é aquele em que existe pelo menos um par de ações em que a ordem de execução não é definida (por exemplo, o plano da Figura 11(b), não especifica se "vestir tênis direito" deve ser executado antes de "vestir meia esquerda" ou vice-versa).

O número de *passos paralelos* necessários para executar um plano parcialmente ordenado corresponde ao número de instantes de tempo necessários para sua execução, considerando que ações podem ocorrer simultaneamente. Como exemplo, o plano representado na Figura 11(b) pode ser executado em dois passos paralelos: (1) vestir meia direita, vestir meia esquerda; e (2) vestir tênis direito, vestir tênis esquerdo.

Algoritmos de planejamento trabalham sobre uma representação do estado em que o mundo se encontra. Uma das representações possíveis consiste em considerar o *estado do mundo*

[1] As definições relacionadas com planejamento encontradas nesta seção foram retiradas de (RUSSELL; NORVIG, 2002).



(a) Plano totalmente ordenado (b) Plano parcialmente ordenado

Figura 11: Exemplos de planos para o objetivo "ter meias e tênis vestidos".

como sendo uma conjunção de literais proposicionais ou de primeira ordem. Deste modo, $Vestido(MeiaDireita) \wedge Vestido(TenisDireito) \wedge \neg Vestido(MeiaEsquerda)$ é um exemplo de estado do mundo.[2]

O *objetivo* de um plano é um estado do mundo parcialmente especificado. "Parcialmente especificado" significa que, para definir o objetivo de um plano, não é preciso descrever um estado do mundo de forma completa; basta indicar os aspectos do mundo que interessam. Por exemplo, se o objetivo do plano é simplesmente ter os tênis vestidos, não é preciso incluir no objetivo nenhum predicado a respeito da camisa ou chapéu. Um estado de mundo satisfaz um objetivo se contém todos os literais deste objetivo. Por exemplo, o objetivo $Vestido(MeiaEsquerda) \wedge Vestido(MeiaDireita)$ é satisfeito pelo estado $Vestido(MeiaDireita) \wedge Vestido(TenisDireito) \wedge Vestido(MeiaEsquerda)$, mas não pelo estado $Vestido(MeiaDireita) \wedge Vestido(TenisDireito) \wedge \neg Vestido(TenisEsquerdo)$.

A execução de uma *ação*[3] é o meio utilizado para alterar o estado do mundo. Uma ação é composta por quatro partes: um nome, uma lista de parâmetros, uma pré-condição e um efeito. A *pré-condição* é uma condição que necessita ser verdadeira para que a ação possa ser utilizada; o *efeito* representa quais alterações a execução da ação causa no estado do mundo.

Um algoritmo de planejamento é dito *ótimo* se gerar, garantidamente, sempre o melhor plano possível, segundo algum critério de otimalidade. Certos algoritmos de planejamento permitem associar um *custo* a cada ação, de modo que ações de menor custo são privilegiadas

[2]Um resumo da notação da lógica de predicados, utilizada em algumas partes deste trabalho, é apresentada no Apêndice A.

[3]Na literatura de IA, ações são ocasionalmente chamadas de *operadores*.

durante a criação do plano.

A idéia de representar ações da forma descrita acima foi introduzida por uma linguagem clássica criada em 1970 chamada STRIPS[4]. À medida que novas aplicações de planejamento necessitavam de linguagens mais expressivas, STRIPS foi sendo estendida. Atualmente, é comum chamar de linguagem "do tipo STRIPS" (*STRIPS-like*) a qualquer linguagem de descrição de problemas de planejamento que se baseie na idéia de representar ações através de pré-requisitos e efeitos. Mais recentemente, boa parte das implementações de algoritmos de planejamento passaram a aceitar como entrada uma linguagem chamada *Planning Domain Description Language* (PDDL) (EDELKAMP; HOFFMANN, 2004). Esta linguagem, que é "do tipo STRIPS", foi criada com o objetivo de uniformizar a sintaxe e semântica da linguagem reconhecida pelos participantes da *International Planning Competition* (IPC) (MCDERMOTT, 2006).

3.3 Avaliação de algoritmos de planejamento

Esta seção apresenta uma avaliação de diversos algoritmos de planejamento que possuem implementação disponível. Esta avaliação foi desenvolvida pelo autor deste texto com o objetivo de selecionar o algoritmo mais adequado para a realização deste trabalho.

3.3.1 Avaliando algoritmos de planejamento do ponto de vista de IS

As referências que descrevem protótipos de IS baseados em planejamento costumam ser muito breves ao justificar sua escolha por um determinado algoritmo de planejamento. De fato, não foi possível encontrar nenhuma discussão razoavelmente aprofundada a esse respeito. A seguir, são propostos diversos aspectos que podem ser utilizados para avaliar algoritmos de planejamento do ponto de vista de IS. Estes aspectos foram selecionados com base na experiência na realização de trabalhos anteriores (BARROS, 2004; BARROS; MUSSE, 2005), mas acredita-se que seja plausível supor que boa parte destes aspectos se aplique à maioria dos trabalhos em IS baseados em um algoritmo de planejamento.

Aspecto 1: *Suporte a requisitos de linguagem adicionais.* Conforme discutido ao final da Seção 3.2, a maioria dos algoritmos de planejamento atuais utiliza PDDL como linguagem de entrada. Porém, PDDL foi projetada como uma linguagem modular: sistemas de planejamento são obrigados a suportar apenas um pequeno conjunto de requisitos, mas podem opcionalmente aceitar uma linguagem de entrada que suporte requisitos extras.

Cada requisito adicional suportado por um algoritmo de planejamento provê maior poder de expressão à linguagem de entrada ou simplifica a descrição de certas ações. Estas duas possíveis conseqüências são desejáveis para aplicações de IS: mais poder de expressão permite a criação de ações mais "interessantes" de um ponto de vista narrativo, e facilidade para escrever ações estimula a experimentação (que acaba resultando em histórias melhores).

Dos requisitos opcionais descritos na especificação da PDDL, destacam-se os cinco seguintes como sendo particularmente úteis para IS:

[4] A sigla STRIPS significa *Stanford Research Institute Problem Solver*, numa referência à aplicação que introduziu a linguagem.

1. *Hierarquias de tipos (requisito `:typing` da PDDL).* Linguagens que exigem uma declaração do tipo de cada objeto utilizado são capazes de identificar erros mais facilmente durante o processo de autoria das ações. O benefício do suporte a hierarquias de tipos (ou "sub-tipos") é mostrado pelo exemplo a seguir. Normalmente, é necessário definir algum predicado como `at (what, where)` para representar a localização de personagens e objetos no ambiente virtual. Caso hierarquias de tipos não sejam suportadas, é necessário especificar duas versões diferentes deste predicado, um para cada um dos tipos para o parâmetro `what` (por exemplo, `character-at (a-character, where)` e `prop-at (a-prop, where)`). O suporte a hierarquias de tipos, permite declarar uma única versão do predicado, aceitando um parâmetro de um tipo como `placeable-thing`, e os tipos `character` e `prop` são criados como sub-tipos de `placeable-thing`.

2. *Operador de igualdade (requisito `:equality` da PDDL).* Freqüentemente é necessário verificar a igualdade entre constantes e variáveis. Um operador de igualdade é particularmente útil quando utilizado em conjunto com efeitos condicionais (discutidos abaixo), pois permite fazer com que um certo efeito só seja utilizado quando a ação é executada por um certo personagem, por exemplo. Outro uso comum do operador de igualdade (em conjunto com pré-condições negativas, também discutidas abaixo) é garantir que dois parâmetros de uma ação sejam ligados a objetos distintos. Isto é ilustrado na Figura 12, na qual a pré-condição `(not (= ?giver ?receiver))` é utilizada para garantir que um personagem não possa dar um presente a si mesmo.

3. *Pré-condições negativas (requisito `:negative-preconditions` da PDDL).* Como discutido acima, pré-condições negativas são úteis quando usadas com o operador de igualdade. Sua utilidade, porém, vai além disso. Elas podem ser usadas, por exemplo, com efeitos condicionais para permitir que um personagem comporte-se de maneira diferente caso ele não saiba de algo importante (`(not (knows ?some-character some-information))`).

4. *Efeitos condicionais (requisito `:conditional-effects` da PDDL).* Efeitos de ações em IS são freqüentemente dependentes de alguma condição, e é exatamente este o propósito dos efeitos condicionais. Novamente, a Figura 12 mostra um exemplo: na definição materialista da ação `GivePresent`, quem recebe o presente passa a ser amigável a quem dá o presente, apenas caso ele goste do presente dado.

5. *Pré-condições existenciais (requisito `:existential-preconditions` da PDDL).* A possibilidade de utilizar quantificadores existenciais em pré-condições abre algumas possibilidades interessantes. Na Figura 13, por exemplo, é definida uma ação `ExaminePlace` que é usada por um personagem que deseja procurar pistas para solucionar um crime. Utilizando um quantificador existencial na pré-condição, é possível expressar que o personagem examinador só pode executar esta ação caso esteja sozinho no local. A ação `GivePresent` (Figura 12) também utiliza um quantificador existencial, neste caso para expressar que tanto quem dá quanto quem recebe o presente devem estar no mesmo local.

Aspecto 2: *Capacidade de gerar planos parcialmente ordenados.* Num contexto de IS, as ações de um plano representam eventos da história, como ações executadas por personagens.

```
(:action GivePresent
   :parameters (?giver ?receiver - character ?present - thing)

   :precondition (and (not (= ?giver ?receiver))
                      (has ?giver ?present)
                      (exists (?p - place)
                              (and (at ?giver ?p)
                                   (at ?receiver ?p))))

   :effect (and (not (has ?giver ?present))
                (has ?receiver ?present)
                (when (likes ?receiver ?present)
                      (friendly-to ?receiver ?giver))))
```

Figura 12: A ação `GivePresent` escrita em PDDL. A maior parte dos requisitos adicionais, para a linguagem de entrada discutidos nesta seção, são usados aqui.

```
(:action ExaminePlace
   :parameters (?examiner - character ?where - place)

   :precondition (and (at ?examiner ?where)
                      (not (exists (?c - character)
                                   (and (not (= ?c ?examiner))
                                        (at ?c ?where))))))

   :effect (and (when (= ?where tughs-cave)
                      (has ?examiner tughs-ticket))
                (when (= ?where zonks-cave)
                      (knows ?examiner zonks-cave-has-wet-paint))))
```

Figura 13: A ação `ExaminePlace` escrita em PDDL. Um quantificador existencial é usado para determinar que o personagem examinador deve estar sozinho no local onde a ação é executada.

Pode ser desejável que ações ocorram simultaneamente durante uma história, e este tipo de paralelismo é oferecido por planos parcialmente ordenados.

Planos parcialmente ordenados podem ser positivamente explorados mesmo se os objetivos do modelo de IS em questão necessitam de um plano estritamente seqüencial. Como diferentes planos totalmente ordenados podem ser extraídos de um único plano parcialmente ordenado, um algoritmo de planejamento que gere planos parcialmente ordenados pode ser usado como uma forma de encontrar diferentes alternativas para o mesmo conjunto de eventos (esta idéia já foi explorada em (CIARLINI et al., 2005)).

Aspecto 3: *Otimalidade.* Um plano não-ótimo tipicamente implica em adicionar à história alguns eventos que não são necessários para levar o mundo a um estado objetivo. Até certo ponto, isto não causa nenhum problema sério: não se pode esperar que alguns eventos "extras" degradem significativamente a percepção que o usuário tem da história. Planos "altamente não-ótimos", por outro lado, tendem a resultar em personagens agindo de maneira tola, o que é totalmente indesejável. Assim, embora otimalidade não seja um requisito estrito para um

algoritmo de planejamento usado num sistema de IS, algoritmos que sejam "demasiadamente não-ótimos" devem ser evitados.

Ainda relacionado a este tópico, deve-se enfatizar que é preciso ter cuidado ao discutir otimalidade de uma perspectiva de IS, pois ela pode passar uma idéia equivocada dependendo da métrica de otimalidade utilizada. Por exemplo, para planos parcialmente ordenados, uma métrica comum é o número de passos paralelos (Seção 3.2) necessários para solucionar o problema, independentemente do número total de ações no plano. Com esta métrica, um plano contendo ações desnecessárias (a ponto de provocar comportamento tolo dos personagens) pode ser considerado ótimo. Um exemplo prático disto é brevemente discutido na Seção 3.3.2, na descrição do sistema de planejamento SatPlan.

Aspecto 4: *Suporte a ações com custos.* Alguns algoritmos permitem associar um custo a cada ação, e privilegiam o uso de ações de menor custo. Este recurso já foi explorado com sucesso em IS em um trabalho anterior (BARROS; MUSSE, 2005), e o presente trabalho propõe utilizá-lo novamente, como será apresentado adiante, na descrição do modelo (Capítulo 4).

Aspecto 5: *Eficiência.* Como IS envolve interatividade, planos normalmente precisam ser gerados enquanto o sistema está rodando (por exemplo, pode ser necessário criar um novo plano porque o usuário executou alguma ação que invalidou o plano anterior (BARROS; MUSSE, 2005)). Portanto, algoritmos de planejamento mais rápidos são preferíveis. Além disso, sistemas de planejamento mais eficientes permitem a criação de histórias mais longas, com mais personagens, objetos e ações.

Aspecto 6: *Necessidade de conhecimento do domínio.* Alguns algoritmos de planejamento são totalmente automáticos, ou seja, para produzir um plano eles necessitam apenas da descrição do problema de planejamento. Outros algoritmos também exigem como entrada alguma forma de conhecimento do domínio, ou seja, algum tipo de informação relativa ao problema sendo tratado, codificada de alguma forma que possa ser utilizada pelo algoritmo para produzir o plano em menor tempo. A primeira versão do Fabulator (o protótipo apresentado em (BARROS, 2004; BARROS; MUSSE, 2005) e descrito na Seção 4.1), simplesmente tratou o problema de planejamento como uma busca no espaço de estados do mundo, e utilizou o algoritmo A* (RUSSELL; NORVIG, 2002) para resolver o problema de busca. Este algoritmo exige conhecimento do domínio na forma de uma função heurística, e na experiência obtida no desenvolvimento do Fabulator, foi verificado que isto desencorajava a experimentação (porque cada alteração na história exigia uma mudança correspondente na heurística, para obter desempenho satisfatório). Considerando este fato, algoritmos totalmente automáticos, ou seja, algoritmos que não exigem conhecimento do domínio, são preferíveis.

Aspecto 7: *Suporte a variáveis numéricas.* Sistemas de planejamento clássicos representam os estados do mundo como uma conjunção de predicados booleanos. Para aplicações de IS, isso pode ser um fator limitante, pois muitos aspectos interessantes de histórias não são booleanos por natureza. De fato, o "pensamento booleano" que domina alguns trabalhos em IS já sofreu fortes críticas (CRAWFORD, 2004). Alguns sistemas de planejamento admitem o uso de variáveis numéricas (além de variáveis booleanas) para descrever o estado do mundo, permitindo ir além de condições rígidas do tipo "verdadeiro ou falso" e adicionar "nuances" às histórias. Há relatos de uso com sucesso de variáveis numéricas em IS (por exemplo, (CIARLINI et al., 2005; MATEAS; STERN, 2005b)).

3.3.2 Análise dos algoritmos

Nesta seção é apresentada a análise propriamente dita realizada com vários sistemas de planejamento. Os algoritmos foram avaliados com base nos aspectos discutidos na seção anterior. Para avaliação de desempenho, o primeiro ato de "A Estória de Ugh" (Apêndice B) foi adaptado para cada um destes sistemas. Os tempos de execução apresentados são médias de três execuções em um computador com processador Pentium 4 HT a 2,8 GHz e 512 MB de RAM.

Para esta análise, foram selecionados algoritmos satisfazendo dois critérios: (i) possuir uma implementação disponibilizada pelos seus autores e (ii) ser capaz de tratar o problema de teste (primeiro ato de "A Estória de Ugh") sem erros.

Graphplan (BLUM; FURST, 1997) foi o primeiro algoritmo de planejamento a converter o problema de planejamento para uma estrutura de dados intermediária chamada *Grafo de Planejamento*, obtendo desempenho surpreendente quando comparado às abordagens de planejamento anteriormente utilizadas. Graphplan é um algoritmo ótimo e gera planos parcialmente ordenados, mas sua linguagem de entrada é bastante limitada: nenhum dos requisitos adicionais discutidos nas Seção 3.3.1 é suportado. Com relação ao desempenho, o problema de teste foi solucionado em 0,138 segundos.

Grafos de Planejamento foram utilizados em diversos trabalhos subseqüentes. Um destes trabalhos é o IPP, *Interference Progression Planner* (KOEHLER et al., 1997), que iniciou como uma extensão do Graphplan com uma linguagem de entrada melhorada: todos os requisitos adicionais de linguagem apresentados são suportados. O IPP ainda é capaz de gerar planos parcialmente ordenados, mas pode ou não ser ótimo, dependendo de como é configurado.[5] O problema de teste foi resolvido em média em 0,106 segundos quando o IPP estava configurado para rodar otimamente, e 0,032 segundos quando não.

O sistema de planejamento SatPlan (KAUTZ; SELMAN, 1992) é baseado em uma idéia diferente: a transformação de um problema de planejamento em um problema de satisfabilidade que pode ser resolvido com diferentes algoritmos. A versão do SatPlan testada, SatPlan_2004, de fato combina a abordagem de "planejamento como satisfabilidade" com Grafos de Planejamento. A linguagem de entrada aceita pelo SatPlan_2004 é bastante limitada: apenas hierarquias de tipos são suportadas da lista de recursos desejados. Do lado positivo, este sistema de planejamento é capaz de gerar planos parcialmente ordenados e é ótimo (porém, veja a discussão abaixo). A eficiência do SatPlan depende do algoritmo usado para resolver o problema de satisfabilidade. Foram realizados testes com três deles, e os tempos para resolver o problema de teste foram de 0,457 segundos (utilizando o `jerusat1.3`), 0,517 segundos (`satz-rand`) e 0,247 segundos (`siege`).

O SatPlan é um bom exemplo de um ponto apresentado na Seção 3.3.1, durante a discussão sobre otimalidade. O algoritmo é tecnicamente ótimo, mas seu critério de otimalidade é o número de passos paralelos necessários para resolver o problema. Nos testes realizados, verificou-se que ações desnecessárias foram incluídas nos planos gerados. O algoritmo `jerusat1.3` foi particularmente problemático neste sentido, produzindo comportamento claramente tolo em personagens.

Outro algoritmo que segue o mesmo princípio (ou seja, combinar as abordagens baseadas

[5] O IPP pode ser executado com uma opção chamada RIFO (*Removing Irrelevant Facts and Operators*), que abre mão da otimalidade em troca de melhor desempenho.

em Grafos de Planejamento e planejamento como satisfabilidade) é o LPG (*Local search for Planning Graphs*) (GEREVINI; SERINA, 2002). Com exceção de efeitos condicionais, todos os requisitos de linguagem adicionais discutidos anteriormente são suportados. Além disso, ele gera planos parcialmente ordenados e suporta ações com custos e variáveis numéricas. O LPG não é ótimo. A implementação do LPG apresentou erros ao tentar solucionar o problema de teste, então foi testada a implementação do LPG-*TD* (GEREVINI et al., 2004), uma versão do LPG com alguns recursos adicionais que não foram utilizados nesse experimento. Ele solucionou o problema de teste em 0,680 segundos quando variáveis numéricas foram utilizadas, e 0,169 segundos quando não.

Uma terceira abordagem para o planejamento foi introduzida pelo HSP, o *Heuristic Search Planner* (BONET; GEFFNER, 2001): ele trata o planejamento com um problema de busca, usando uma heurística independente de domínio para guiar o processo. O HSP gera apenas planos totalmente ordenados e não é ótimo. A linguagem de entrada do HSP tem suporte a hierarquias de tipos e possui um operador de igualdade, mas carece de todos os outros recursos discutidos anteriormente. O problema de teste foi resolvido em 0,031 segundos. Convém ainda notar que o HSP já foi utilizado com sucesso em IS (CHARLES et al., 2003).

Foram feitas pesquisas com o objetivo de encontrar heurísticas admissíveis para o HSP, de modo a torná-lo ótimo. Isto levou ao desenvolvimento do HSP* (HASLUM; GEFFNER, 2000), uma família de algoritmos de planejamento ótimos. Além da otimalidade, o HSP* introduz outras importantes melhorias ao HSP: suporte a pré-condições negativas, ações com custos, variáveis numéricas e capacidade de gerar planos parcialmente ordenados. Os tempos para solucionar o problema de teste, contudo, foram consideravelmente mais altos do que os obtidos com o HSP: 3,641 segundos (para a variante hsp_0), 3,642 segundos (hsp_a), 247,71 segundos (hsp_b), 7,25 segundos (hsp_c) e 3,65 segundos (hsp_d).

Uma quarta abordagem para o planejamento foi introduzida com o TLplan (BACCHUS; KABANZA, 2000), que é baseado em busca, mas utiliza conhecimento do domínio fornecido pelo usuário (codificado na forma de uma *fórmula de controle* escrita em uma lógica temporal) para podar o espaço de estados e obter melhor desempenho. Este é o único dos algoritmos testados que requer o uso de conhecimento do domínio, e observou-se que este é um grave empecilho para seu uso em IS. Com efeito, não foi possível criar uma fórmula de controle boa o suficiente para resolver o problema de teste em um tempo razoável. Não está se sugerindo que seja impossível criar uma fórmula de controle adequada para este problema. É provável que pessoas com maior familiaridade com lógicas temporais sejam capazes de criá-la. Porém, aparentemente, é muito mais simples utilizar este formalismo para problemas clássicos de IA, como o Mundo dos Blocos (*Blocks World*) usado como exemplo em (BACCHUS; KABANZA, 2000). Para "problemas de planejamento de IS", é muito difícil identificar situações que possam ser podadas da busca.

Uma exploração mais cuidadosa sobre o uso do TLplan em IS pode ser um trabalho interessante, pois ele é considerado surpreendentemente eficiente, possui uma linguagem de entrada extremamente expressiva, tem suporte a variáveis numéricas, ações com custos e, com uma fórmula de controle adequada, é ótimo.

Os demais algoritmos discutidos aqui combinam aspectos das diferentes abordagens descritas acima. FF (*Fast Forward*) (HOFFMANN; NEBEL, 2001), por exemplo, foi criado unindo idéias novas com idéias introduzidas por HSP, Graphplan e IPP. Este algoritmo não é ótimo, produz planos totalmente ordenados e não é capaz de tratar variáveis numéricas e ações com

custos. Porém, suporta todos os requisitos extras discutidos na Seção 3.3.1 e foi o algoritmo mais rápido para obter a solução do problema de teste, levando apenas, em média, 0,015 segundos.

Algumas extensões ao FF foram propostas. Uma delas é o Metric-FF (HOFFMANN, 2003), que adiciona suporte a variáveis numéricas. Adicionalmente, embora ações com custos não sejam explicitamente suportadas, elas podem ser facilmente "emuladas" com variáveis numéricas, pois o Metric-FF permite que seja definida uma métrica de otimalidade (uma função de variáveis numéricas que deve ser maximizada ou minimizada). Como esperado, estes recursos adicionados implicaram em um incremento no tempo de execução: este sistema de planejamento levou 0,059 segundos para resolver o problema de teste quando algumas variáveis numéricas foram utilizadas. Sem o uso de variáveis numéricas, o tempo necessário para resolver o problema foi de 0,018 segundos.

Marvin, *Macro-Actions from Reduced Versions of the INstance* (COLES; SMITH, 2004), também é baseado no FF, do qual ele herdou a estratégia de busca. Este algoritmo introduziu uma etapa de pré-processamento, durante a qual uma versão reduzida do problema de entrada é automaticamente extraído e resolvido. Depois, esta "solução intermediária" é usada para auxiliar na busca pela solução final. Marvin suporta a mesma linguagem de entrada que o FF, mas é capaz de gerar planos parcialmente ordenados. O problema de teste foi resolvido em 0,017 segundos.

O último algoritmo nesta análise, STAN 4 (ou *Hybrid STAN*) (FOX; LONG, 2001), parte da observação que, embora planejamento em geral seja uma tarefa difícil, certos problemas de planejamento apresentam características que permitem solucioná-los de maneira eficiente com o uso de algoritmos específicos. Assim, o sistema inicia analisando o problema; caso ele seja reconhecido como sendo um problema para o qual existe um algoritmo mais específico (e eficiente), o algoritmo específico é utilizado ao invés do algoritmo geral. STAN não suporta nenhum dos requisitos de linguagem adicionais descritos na Seção 3.3.1, mas é capaz de gerar planos parcialmente ordenados.

Os resultados desta análise são resumidos na Tabela 2. Adicionalmente, a Tabela 3 provê uma referência para cada algoritmo analisado e uma lista de algoritmos nos quais ele é baseado. Finalmente, a Tabela 4 indica os endereços de onde foram obtidas as implementações dos sistemas de planejamento estudados.

3.3.3 A escolha

Observando os resultados da análise apresentada na seção anterior, é possível notar que não há um algoritmo que reúna todas as características desejadas discutidas na Seção 3.3.1. É necessário, portanto, verificar quais são as características mais importantes para o trabalho que está sendo proposto, e selecionar o algoritmo que melhor atenda a estes requisitos.

O modelo proposto neste trabalho baseia-se no uso de ações com custos, de modo que é essencial que este recurso possa ser utilizado no algoritmo selecionado. Além disso, com base na experiência de trabalhos anteriores, é possível afirmar que ser capaz de expressar as idéias facilmente durante a criação do protótipo leva a resultados superiores, de modo que algoritmos suportando um grande número de requisitos adicionais (Seção 3.3.1) são mais adequados para as necessidades deste trabalho. Por fim, antevendo trabalhos futuros, considera-se que suporte

Tabela 2: Resumo da análise dos algoritmos de planejamento. As colunas indicam suporte a hierarquias de tipos (HT), operador de igualdade (OI), pré-condições negativas (PN), efeitos condicionais (EC), pré-condições existenciais (PE), otimalidade do algoritmo (Ot), capacidade de gerar planos parcialmente ordenados (PPO), suporte ações com custos (AC), variáveis numéricas (VN) e tempo para solucionar o problema de teste (Tempo). Notas: (1) A variante hsp_b do HSP* levou consideravelmente mais tempo: acima de 4 minutos. (2) Pode ser ótimo ou não, dependendo de como for configurado. (3) O primeiro tempo mostrado é para a configuração ótima, o segundo para a configuração não-ótima. (4) O primeiro tempo mostrado é com o uso de variáveis numéricas, o segundo sem. (5) Não é explicitamente suportado, mas pode ser facilmente emulado. (6) É ótimo desde que uma fórmula de controle adequada seja utilizada. (7) Não foi possível criar uma fórmula de controle apropriada para o problema de teste; por isso, o tempo não pode ser verificado.

Algoritmo	HT	OI	PN	EC	PE	Ot	PPO	AC	VN	Tempo
FF	✓	✓	✓	✓	✓	✗	✗	✗	✗	$0,015$ s
Graphplan	✗	✗	✗	✗	✗	✓	✓	✗	✗	$0,138$ s
HSP	✓	✓	✗	✗	✗	✗	✗	✗	✗	$0,031$ s
HSP*	✓	✓	✓	✗	✗	✓	✓	✓	✗	$3,641$–$7,25$ s (1)
IPP	✓	✓	✓	✓	✓	(2)	✓	✗	✗	$0,106$ / $0,032$ s (3)
LPG-*TD*	✓	✓	✓	✗	✓	✗	✓	✓	✓	$0,680$ / $0,169$ s (4)
Marvin	✓	✓	✓	✓	✓	✗	✓	✗	✗	$0,017$ s
Metric-FF	✓	✓	✓	✓	✓	✗	✗	(5)	✓	$0,059$ / $0,018$ s (4)
SatPlan_2004	✓	✗	✗	✗	✗	✓	✓	✗	✗	$0,247$–$0,512$ s
STAN 4	✗	✗	✗	✗	✗	✗	✓	✗	✗	$0,093$ s
TLplan	✗	✓	✓	✓	✓	(6)	✗	✓	✓	(7)

a variáveis numéricas é uma característica adicional interessante.

Dados estes requisitos, optou-se por utilizar o Metric-FF para o desenvolvimento deste trabalho. Ações com custos podem ser facilmente "emuladas" neste algoritmo com o uso de métricas de otimalidade definidas pelo usuário. Sua linguagem de entrada suporta todos os requisitos adicionais discutidos, além de variáveis numéricas. A própria capacidade de definir métricas de otimalidade pode ser eventualmente útil. O algoritmo que mais se aproxima do Metric-FF em termos de recursos é o LPG, que apresentou desempenho significativamente inferior nos testes realizados.

Tabela 3: Dados adicionais sobre algoritmos de planejamento: referências e outros algoritmos nos quais eles se baseiam.

Algoritmo	Referência	Baseado em
FF	(HOFFMANN; NEBEL, 2001)	HSP, Graphplan, IPP
Graphplan	(BLUM; FURST, 1997)	—
HSP	(BONET; GEFFNER, 2001)	—
HSP*	(HASLUM; GEFFNER, 2000)	HSP
IPP	(KOEHLER et al., 1997)	Graphplan
LPG-*TD*	(GEREVINI et al., 2004)	SatPlan, Graphplan
Marvin	(COLES; SMITH, 2004)	FF
Metric-FF	(HOFFMANN, 2003)	FF
SatPlan_2004	(KAUTZ; SELMAN, 1992)	SatPlan, Graphplan
STAN 4	(FOX; LONG, 2001)	Graphplan
TLplan	(BACCHUS; KABANZA, 2000)	—

Tabela 4: Endereços de onde foram obtidas as implementações dos algoritmos de planejamento analisados.

Algoritmo	URL
FF	`http://members.deri.at/~joergh/ff.html`
Graphplan	`http://www.cs.cmu.edu/~avrim/graphplan.html`
HSP	`http://www.ldc.usb.ve/~bonet/#software`
HSP*	`http://www.ida.liu.se/~pahas/hsps/`
IPP	`http://www.informatik.uni-freiburg.de/~koehler/ipp.html`
LPG-*TD*	`http://zeus.ing.unibs.it/lpg/`
Marvin	`http://planning.cis.strath.ac.uk/Marvin`
Metric-FF	`http://members.deri.at/~joergh/metric-ff.html`
SatPlan_2004	`http://www.cs.rochester.edu/u/kautz/satplan`
STAN 4	`http://planning.cis.strath.ac.uk/STAN`
TLplan	`http://www.cs.toronto.edu/~fbacchus/tlplan.html`

4 MODELO

Para fundar o estabelecimento adquiriria 1000 tartarugas prontas a pôr; cada tartaruga, de cada assentada, desova 400 ovos para a primeira ninhada. No segundo ano, outros 400000 ovos; no terceiro ano, as primeiras 400000 tartarugas novas já começariam a pôr, na razão de 400 ovos cada uma, ou sejam, 160400000 ovos para o choco do quarto ano; e assim, sucessivamente multiplicados e somados, tirada a prova dos noves, os algarismos patenteavam que ao cabo de sete anos eu teria um viveiro de 700000000000000000000000000000000 de tartarugas, e que daí em diante, com vigilância e economia, começaria então eu a desfrutar em manteiga de ovos, sopa e cascas (ou cascos) um regular rendimento, justa compensação do rigor do meu jogo de algarismos.

— Romualdo

Na Introdução, foram apresentados três problemas existentes em modelos de IS baseados em planejamento:

1. Eles *não* são inerentemente consistentes narrativamente.
2. Eles podem chegar a situações sem saída.
3. Eles impõem uma carga de trabalho muito grande sobre os autores das histórias.

O objetivo principal deste trabalho é investigar novos métodos que permitam minimizar estes problemas. Adicionalmente, deseja-se que estes métodos estejam fundamentados em conceitos e teorias propostas por áreas ligadas às narrativas. A primeira seção deste capítulo apresenta a arquitetura básica utilizada para a investigação. As seções seguintes detalham os mecanismos propostos para a minimização dos problemas supracitados. A última seção do capítulo contextualiza o modelo dentro de uma teoria narrativa.

4.1 Arquitetura para o desenvolvimento do trabalho

O modelo que serve de base para este trabalho (BARROS, 2004; BARROS; MUSSE, 2005) está fundamentado em uma arquitetura com três módulos principais, conforme mostrado na Figura 14.

O módulo de *seleção de ações* é responsável por determinar uma seqüência de ações capaz de transformar o mundo de seu estado corrente para um estado objetivo. Esta seqüência de ações é criada com o uso de um algoritmo de planejamento, e o estado objetivo é especificado pelo autor da história. De fato, o modelo não trabalha com um único objetivo, mas com três, utilizados em seqüência. Desta forma, o modelo é capaz de gerar histórias que seguem a divisão em três atos, utilizada desde o teatro grego clássico (ARISTÓTELES, 1966) até o cinema contemporâneo (FIELD, 1995).

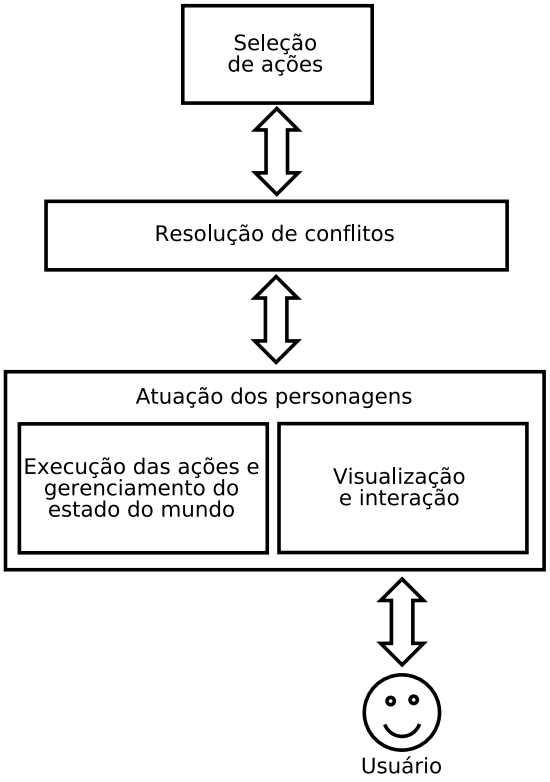


Figura 14: Arquitetura do Fabulator (BARROS, 2004).

Os planos gerados pelo algoritmo de planejamento incluem tanto ações que devem ser executadas pelos NPCs quanto pelo protagonista da história (que é controlado pelo usuário). Como o usuário tem liberdade de executar as ações que quiser, não há nenhuma garantia de que ele vá "cumprir" o que estava previsto no plano. Por isso, um novo plano é gerado caso alguma das seguintes situações seja detectada:

- O usuário executou alguma ação que tornou o plano inválido, ou seja, modificou o estado do mundo de forma que, caso a execução do plano atual prossiga, um estado objetivo não será atingido.
- O arco de tensão observado na história está significativamente diferente de um arco de tensão desejado pelo autor da história (conforme detalhado na Seção 4.2).

O módulo de *atuação dos personagens* tem como funções manter e gerenciar o estado do mundo e prover mecanismos para visualização da história e interação. Este módulo recebe ordens para executar ações tanto do módulo de seleção de ações quanto do usuário, e encarrega-se de exibir graficamente a execução destas ações e alterar o estado do mundo adequadamente. A visualização das histórias é feita utilizando gráficos 3D.

Finalmente, o módulo de *resolução de conflitos* busca eliminar um possível problema no momento em que um novo plano é criado pelo módulo de seleção de ações. O problema em questão é a possibilidade do novo plano determinar que um personagem deve executar uma ação diferente da ação que ele executava anteriormente. Neste caso, o módulo de resolução de conflitos define se a execução da nova ação deve iniciar imediatamente, interrompendo a ação anteriormente em execução, ou se deve aguardar a conclusão da execução desta ação. Esta decisão é baseada em prioridades associadas a cada ação.

4.2 Modelando e respeitando arcos de tensão

Autores procuram fazer com que suas narrativas sigam um determinado arco de tensão, como discutido na Seção 3.1.2. Por isso, considera-se que sistemas de IS também devem ser capazes de gerar histórias que respeitem um arco de tensão especificado pelo autor. Para atingir este objetivo, dois aspectos devem ser considerados. Em primeiro lugar, é preciso modelar o conceito de tensão da história. E, em seguida, é necessário dotar o modelo de IS de meios que procurem fazer com que a tensão observada nas histórias siga o arco desejado.

4.2.1 Modelando arcos de tensão

O escopo deste trabalho foi limitado a histórias que sigam o enredo mestre Enigma (Seção 3.1.3). Não foi possível encontrar na literatura informações detalhadas a respeito da tensão (e de arcos de tensão em particular) neste tipo de narrativa. Com efeito, a informação mais concreta a esse respeito é a afirmação de que "a tensão deve vir do conflito entre o que aconteceu em oposição ao que parece ter acontecido" (TOBIAS, 1993). Por esta razão, o modelo de arcos de tensão aqui apresentado é uma contribuição original deste trabalho.

O modelo parte da premissa de que a tensão aumenta à medida que novas informações são descobertas ou reveladas, de modo a aproximar o conhecimento que o usuário tem (o que parece ter acontecido) da verdade (o que realmente aconteceu). O auge da tensão é o momento em que o usuário precisa apenas de uma última e decisiva pista para descobrir a solução do enigma. O desfecho da história é o momento em que o conhecimento do usuário se iguala ao que realmente aconteceu, ou seja, é o momento em que o enigma foi solucionado. Assim, *no modelo proposto, conhecimento e tensão se equivalem*. A Figura 15 ilustra estes conceitos.

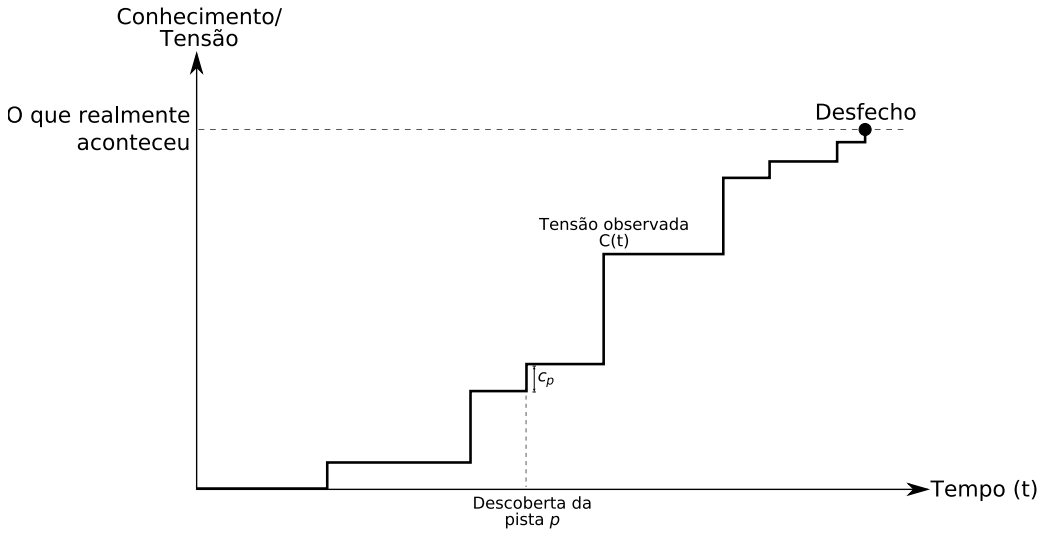


Figura 15: Modelo de arco de tensão em histórias do tipo Enigma.

Com respeito à afirmação de que tensão e conhecimento se equivalem, deve-se fazer uma observação importante. Aqui, o termo "tensão" é utilizado com um sentido mais restrito que a noção intuitiva e informal de "tensão de uma história". Em particular, uma história pode possuir diversas fontes de tensão, e o modelo proposto procura ser capaz apenas de tratar da tensão produzida pelo enigma em si. Como um exemplo, pode-se imaginar uma passagem

de uma história em que o protagonista, capturado por inimigos, encontra-se isolado enquanto aguarda que seu destino seja decidido por seus captores. Intuitivamente, seria possível dizer que este é um momento tenso da história. Porém, esta tensão não provém de um enigma e, portanto, não é considerada pelo presente modelo.

Formalizando as idéias apresentadas anteriormente, o arco de tensão de uma história é uma função do tempo, denotada por $C(t)$. Por exemplo: no início de uma narrativa, caso o usuário não saiba nada a respeito do enigma, seu conhecimento (e, por conseqüência, a tensão) é zero, ou seja $C(0) = 0$. O valor de C é incrementado cada vez que uma pista é revelada ao usuário. A magnitude deste incremento depende da pista revelada: para uma pista p, o incremento será de c_p. Os valores de c para cada pista da história são parâmetros do modelo.[1] Assim, se uma pista p for descoberta num instante t, o valor de C é incrementado de c_p neste instante. Na Figura 15 está indicado um instante em que isto ocorre.

Convém observar que, neste modelo, $C(t)$ é uma função monotônica não-decrescente, pois o conhecimento do usuário jamais diminui. Este fato deve ser enfatizado, pois o conceito de arco de tensão apresentado na Seção 3.1.2 admite que o nível de tensão sofra reduções em determinados trechos da história. Espera-se que futuramente o modelo aqui apresentado possa ser estendido para tratar reduções no conhecimento do usuário (por exemplo, devido a mentiras contadas por NPCs). Com esta extensão, seria possível representar arcos com redução no nível de tensão em certos pontos da história.

4.2.2 Respeitando arcos de tensão

Para que as histórias geradas possam seguir um determinado arco de tensão, é preciso antes de mais nada definir qual é este arco desejado. Seguindo as idéias apresentadas na subseção anterior, o arco de tensão desejado é uma função do tempo, definida pelo autor da história e denotada por $C^*(t)$. Dadas as definições da curva de tensão observada $C(t)$ e a desejada $C^*(t)$, é possível definir uma nova função,

$$D(t) = C^*(t) - C(t),$$

que mede a discrepância entre estas duas curvas num dado instante. Estes conceitos são ilustrados na Figura 16.

Idealmente, não deveria haver discrepância entre os arcos desejado e observado, ou seja, gostaria-se que $D(t) = 0$ para todos os valores possíveis de t. Na prática, porém, sempre existirá alguma diferença entre estes dois arcos. É preciso, portanto, definir uma forma de medir a "discrepância total" observada em uma história. Aqui, propõe-se que esta grandeza seja representada por E e definida como

$$E = \int_{t=0}^{T} |D(t)| \, ,$$

onde T é a duração total da história. Deste modo, a idéia intuitiva de respeitar um arco de tensão especificado pelo autor da história pode ser mais formalmente definida como minimizar E.

[1] A princípio, estes parâmetros deveriam ser ajustados pelo autor da história, mas a Seção 4.4 descreve uma metodologia que permite que eles sejam refinados com base em uma avaliação feita por usuários de versões preliminares da história.

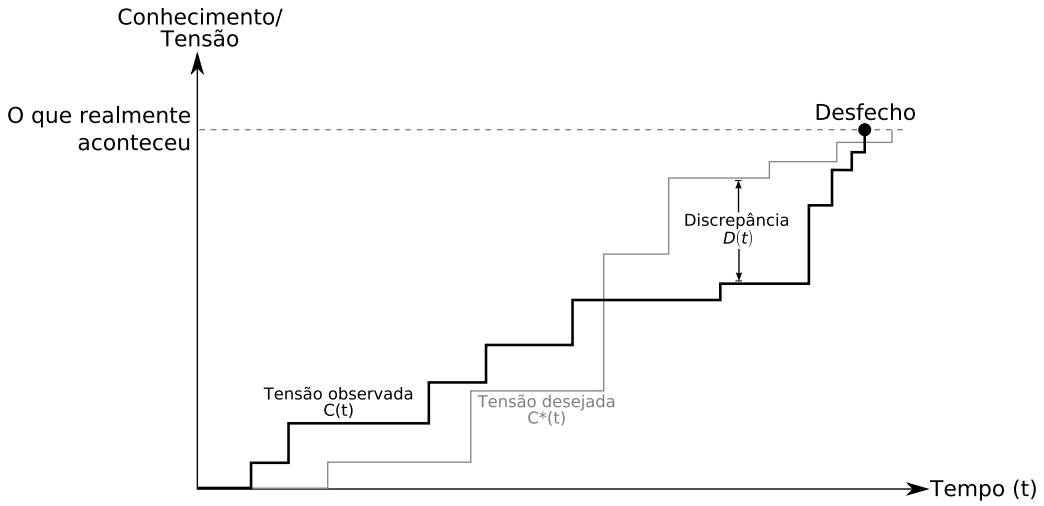


Figura 16: Respeitando arcos de tensão.

No modelo proposto, o valor de $C(t)$ só se altera quando uma pista é descoberta pelo usuário. Assim, para reduzir a discrepância $D(t)$ (e, conseqüentemente, minimizar E), é preciso aumentar ou diminuir o ritmo com que pistas são descobertas, o que é feito alterando-se dinamicamente o nível de participação que os NPCs têm na descoberta das pistas. Em outras palavras, caso o usuário esteja tendo dificuldades para encontrar pistas ($C(t) < C^*(t)$), os demais personagens da história devem passar a agir de modo a auxiliar o usuário em sua tarefa. Analogamente, caso o usuário esteja encontrando pistas muito rapidamente ($C(t) > C^*(t)$), os NPCs farão o mínimo possível para auxiliar o usuário na sua busca.

A forma adotada para alterar o nível de participação dos NPCs na solução do enigma consiste na geração de novos planos, variando o custo associado às ações executadas por estes personagens. Denota-se por $N(t)$ o valor utilizado para este custo num instante t. Desta forma, se for detectado, por exemplo, que o usuário está tendo dificuldades, basta gerar um novo plano com N reduzido. Ao gerar este novo plano, o algoritmo de planejamento prioriza as ações de custo mais baixo. Como resultado, existe a tendência de que os personagens coadjuvantes atuem mais efetivamente para localizar pistas.

Dada esta idéia inicial, dois aspectos ainda precisam ser definidos: que custo exatamente deve ser utilizado para as ações dos NPCs e quando deve ser disparada a criação de um novo plano. Com relação ao custo a ser utilizado, *a priori* não há como saber a relação existente entre o valor utilizado e os planos gerados. Ou seja, para definir, por exemplo, qual deve ser o valor dos custos para que a participação dos NPCs seja pequena, é preciso realizar experimentos.

Assim, definiu-se manter o custo das ações do protagonista fixo em 100 e, em seguida, gerar vários planos para um problema de planejamento[2] utilizando diferentes custos para as ações dos NPCs. Analisando o nível de participação dos NPCs nos planos gerados, notou-se que o custo ideal para as ações dos NPCs em um determinado instante (denotado $N^*(t)$) pode ser aproximado por uma função linear da discrepância neste mesmo instante ($D(t)$), ou seja,

$$N^*(t) = k_1 D(t) + k_2,$$

[2]O problema de planejamento utilizado nestes testes corresponde ao primeiro ato d'*A Estória de Ugh 2*, apresentada na Seção 6.1.

com $k_1 < 0$. O parâmetro k_1 determina a inclinação da reta, e pode ser utilizado para ajustar o quão drásticas serão as alterações em $N^*(t)$. Em outras palavras, aumentando o valor de k_1 (em módulo), o custo utilizado para as ações dos NPCs sofrerá alterações maiores para uma variação fixa na discrepância observada. O parâmetro k_2, por sua vez, pode ser interpretado como o nível de dificuldade da história. Quanto maior o seu valor, maior será o custo utilizado para as ações dos NPCs, e, portanto, maior a dificuldade.

Os testes descritos acima também permitiram definir os valores para k_1 e k_2 utilizados nos experimentos relatados no Capítulo 6: $k_1 = -1,75$ e $k_2 = 100$.

O segundo aspecto, ou seja, o critério para determinar em que momento disparar a criação de um novo plano, foi definido da seguinte forma: um novo plano é gerado, utilizando o custo ideal para ações dos NPCs computado conforme descrito acima, quando pelo menos uma das seguintes condições for observada num instante t:

1. $|D(t) > k_3|$, ou seja, se a discrepância entre a tensão desejada e a tensão observada (em módulo) estiver acima de um certo limiar.
2. $N(t)/N^*(t) > k_4$ ou $N^*(t)/N(t) > k_4$, ou seja, se a diferença percentual entre o custo ideal para ações dos NPCs e o custo utilizado no plano corrente for superior a um certo limiar.

A primeira destas duas condições faz com que novos planos sejam gerados quando a discrepância absoluta for muito alta. A segunda condição faz com que novos planos sejam gerados quando a discrepância $D(t)$ voltar a níveis aceitáveis, para que os NPCs voltem a atuar no seu ritmo normal. Além destes dois aspectos, o modelo prevê que um intervalo de pelo menos 30 segundos seja observado entre duas gerações consecutivas de planos, para que haja tempo dos personagens executarem as ações previstas no novo plano.

Os parâmetros k_3 e k_4 definem os limites de discrepância que o modelo admite antes de gerar um novo plano. Quanto maiores forem os seus valores, mais conivente será o sistema com diferenças entre a tensão observada e a desejada. Porém, cabe observar que utilizar valores excessivamente baixos para estes parâmetros causa um alto número de replanejamentos sem proporcionar benefícios em termos de redução da discrepância total (pois, neste caso, novos planos são gerados antes que os NPCs tenham tempo de agir). Nos experimentos do Capítulo 6, foram utilizados $k_3 = 20$ e $k_4 = 1,25$.

4.3 Antevendo e evitando situações sem saída

A possibilidade do usuário levar a história a uma situação sem saída é inerente à abordagem baseada em algoritmos de planejamento. Uma solução apresentada para este problema são os mecanismos de intervenção propostos em (YOUNG, 1999) e descritos na Seção 2.1. Este tipo de solução é eficaz no sentido de que pode eliminar totalmente a possibilidade das situações sem saída, mas apresenta um inconveniente: ela tende a provocar frustração no usuário, já que algumas de suas ações, para todos efeitos práticos, são ignoradas pelo sistema sem uma explicação condizente com a história.

A alternativa aqui proposta não busca eliminar completamente o problema das situações sem saída, mas, nos casos em que for aplicável, pode permitir uma solução mais integrada ao

contexto história, reduzindo a frustração do usuário. Além disso, esta alternativa atua de forma antecipada, ao contrário da solução de Young, que atua no momento exato em que o usuário levaria a história a uma situação sem saída. Acredita-se que isto também contribua para tornar a intervenção menos evidente ao usuário (infelizmente, não é possível fazer esta afirmação com certeza, pois Young não apresenta resultados de experimentos com usuários, que possam ser comparados com os resultados obtidos neste trabalho). A idéia proposta é composta de duas etapas: antever situações sem saída e agir para evitar que elas ocorram. As subseções seguintes descrevem estas etapas.

4.3.1 Antevendo situações sem saída

O usuário de um sistema de IS possui liberdade de executar as ações que forem de sua vontade. Porém, ele deve escolher suas ações dentre um repertório limitado e conhecido, que é especificado pelo autor da história durante o processo de autoria. Assim, embora não seja possível saber o que o usuário fará, é possível saber tudo que ele *pode* fazer.

Isto permite que se crie uma árvore com os possíveis comportamentos futuros do usuário, conforme mostrado na Figura 17. Nesta figura, o estado corrente do mundo é e_0, na raiz da árvore. Neste estado, o usuário tem como opções de ação $a_1, a_2, \ldots, a_n$. Caso ele execute a ação a_2, ele irá alterar o mundo para o estado e_2, e passará a ter um novo conjunto de ações à sua disposição $(a_1, a_2, \ldots, a_m)$.

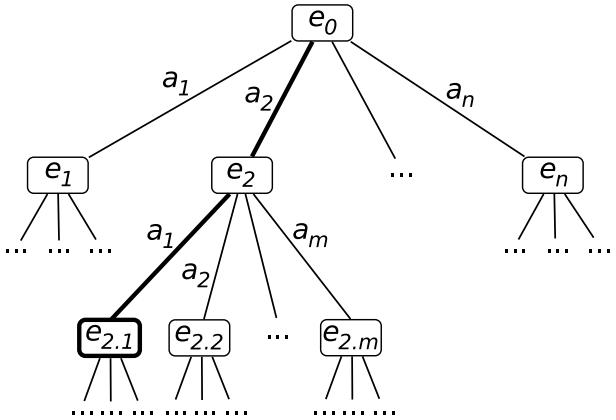


Figura 17: Antevendo situações sem saída. Uma simulação dos possíveis próximos passos do usuário permite verificar que o estado $e_{2.1}$ é uma situação sem saída.

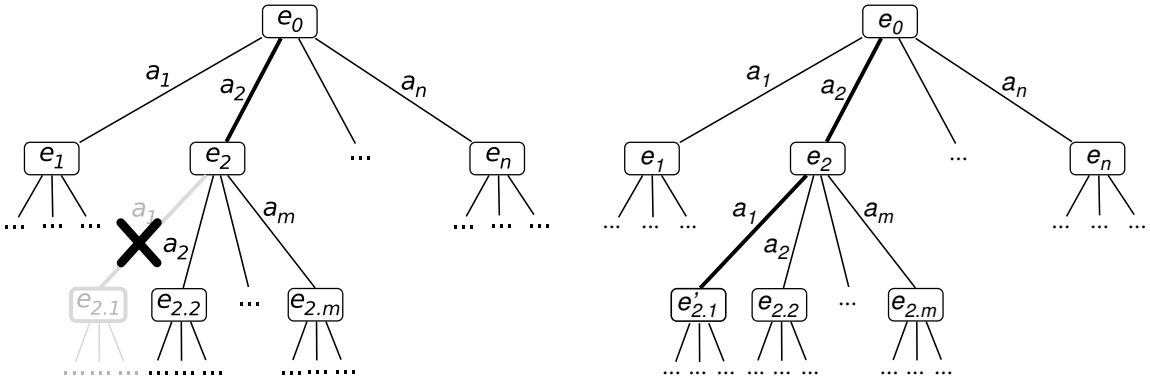
Para identificar situações sem saída, basta expandir alguns níveis desta árvore e testar, para cada estado expandido, se o algoritmo de planejamento é capaz de gerar um plano que transforme o estado em questão em um estado objetivo. Na Figura 17 está ilustrado um caso em que, se o usuário executar a seqüência de ações $\langle a_2, a_1 \rangle$, ele chegará a um estado $(e_{2.1})$ que corresponde a uma situação sem saída. Seria desejável, portanto intervir de alguma forma na história para evitar que o usuário levasse-a até esse estado.

É preciso ainda discutir uma limitação deste método de detecção de situações sem saída: tipicamente, o número de ações disponíveis para o usuário é bastante grande, de modo que o fator de ramificação da árvore tende a ser alto. Desta forma, ao contrário do que seria desejado, é inviável expandir muitos níveis da árvore. Ainda assim, acredita-se que o método proposto é interessante, pois esta limitação pode ser futuramente minimizada. Uma opção neste sentido

é a utilização de um modelo de usuário que possibilite estimar quais são as ações futuras do usuário mais prováveis, e expandir prioritariamente os ramos da árvore que apresentem uma boa probabilidade de ocorrer. Modelos de usuário para previsão do comportamento do usuário já foram utilizados em IS com propósito similar a esse (MAGERKO, 2005).

4.3.2 Evitando situações sem saída

Uma vez que um determinado estado seja identificado como sendo uma situação sem saída, é necessário intervir de alguma maneira para evitar que o usuário leve o mundo até este estado. Neste modelo, isto pode ser feito de duas formas diferentes, ilustradas na Figura 18.



(a) O estado do mundo é alterado de forma que o pré-requisito da ação a_1 não mais será atendido. O estado $e_{2.1}$ torna-se inacessível.

(b) O estado do mundo é alterado de modo que, caso o usuário execute a seqüência de ações $\langle a_2, a_1 \rangle$, atingirá um estado válido $e'_{2.1}$, ao invés da situação sem saída.

Figura 18: Duas formas de evitar situações sem saída.

A primeira forma consiste em alterar o estado do mundo de modo que o pré-requisito de alguma das ações que levariam à situação sem saída não seja mais atendido. No exemplo da Figura 18(a), havia sido detectado que a seqüência de ações $\langle a_2, a_1 \rangle$ levaria a uma situação sem saída. Para evitá-la, o estado do mundo foi modificado de modo que a ação a_1 não tivesse mais seu pré-requisito respeitado. Desta forma, elimina-se a possibilidade do usuário executar a_1 e, portanto, da história atingir a situação sem saída representada pelo estado $e_{2.1}$.

A segunda forma de evitar situações sem saída opera modificando o estado do mundo de maneira que a execução da seqüência de ações identificada como problemática não leve a história a uma situação sem saída. Na Figura 18(b), por exemplo, o estado do mundo foi alterado de tal forma que a seqüência de ações $\langle a_2, a_1 \rangle$ levará a história a um estado "válido" $e'_{2.1}$, ao invés do estado $e_{2.1}$.

Estas duas formas de evitar situações sem saída são exemplificadas de forma prática, respectivamente, pelas ações *DropAndDirty* e *GrowNewStrawberry*, descritas na Seção 6.1.

Deve-se tomar cuidado, contudo, com a forma de intervenção adotada, pois alterar o estado do mundo de forma injustificada reduz a consistência narrativa. Propõe-se, pois, a introdução de dois tipos de ações no modelo: *ações de pretexto* e *ações de intervenção*. Ações de pretexto são ações que não possuem pré-requisitos, e, portanto, podem ser executadas a qualquer mo-

mento. O propósito das ações de pretexto é introduzir no estado do mundo alguns predicados que sejam pré-requisitos para ações de intervenção. As ações de intervenção, por sua vez, são utilizadas para evitar que o usuário leve a história a uma situação sem saída previamente detectada. São as ações de intervenção que realizam as alterações no estado do mundo mencionadas nos parágrafos anteriores e ilustradas na Figura 18. Como exemplo, poderia existir uma ação de pretexto definida da seguinte forma:

Ação: *ChoveChuva*
Pré-requisito: Nenhum
Efeito: *OcorreuTemporal*

Em seguida, poderiam ser definidas várias ações de intervenção que tenham *OcorreuTemporal* como pré-requisito:

Ação: *CaiPonte*
Pré-requisito: *OcorreuTemporal*
Efeito: ¬*CasaAcessível*

Ação: *PegaGripe (personagem)*
Pré-requisito: *OcorreuTemporal*
Efeito: *Gripado (personagem)*

Ação: *ApagaIncendio (prédio)*
Pré-requisito: *OcorreuTemporal*
Efeito: ¬*EmChamas (prédio)*

Caso em algum momento seja detectado que a história atingirá uma situação sem saída se o usuário chegar em casa e caso a ação de pretexto *ChoveChuva* tenha sido executada anteriormente, será possível executar a ação de intervenção *CaiPointe*, que evitará a situação problemática, com o pretexto de que a ponte caiu em conseqüência do temporal.

Tecnicamente, seria possível que uma ação de pretexto fosse executada já com uma ação de intervenção específica em vista. Porém, isto tornaria a intervenção demasiadamente explícita ao usuário. Por isso, neste modelo, ações de pretexto são executadas aleatoriamente: a cada 10 segundos, há uma probabilidade de 5% de uma ação de pretexto ser executada. Evidentemente, executar ações de pretexto em momentos aleatórios aumenta as chances de ocorrer um caso em que a situação sem saída não pode ser evitada pelo método descrito. Para estes casos, poderia ser utilizado o mecanismo de intervenção proposto por Young (1999), que sempre é capaz de solucionar o problema, mas prejudica a consistência narrativa de forma mais intensa.

4.4 Ajustando parâmetros com base na avaliação de usuários

A criação de conteúdo para os sistemas de IS descritos na literatura exige um esforço considerável por parte dos autores de histórias. Laura Mixon relata que levou um ano inteiro de trabalho para produzir uma história para o sistema Erasmaganza de Crawford (MIXON, 1997). Mateas e Stern escreveram mais de 100.000 linhas de código para programar o comportamento dos dois personagens do Façade (MATEAS; STERN, 2004a).[3] Tipicamente, os modelos de IS

[3]Cabe ainda acrescentar que estas mais de 100.000 linhas código foram escritas em ABL, uma linguagem criada especialmente para programar o comportamento de personagens, não em uma linguagem de propósito geral.

propostos na literatura possuem uma grande quantidade de parâmetros, cujos valores são informados ao sistema pelo autor da história. Ajustar os valores destes parâmetros é uma tarefa que exige muitos testes e demanda muito tempo. Abaixo, é apresentado um método bastante simples, que utiliza a avaliação de usuários de versões preliminares da história para ajustar certos parâmetros.

A idéia consiste em permitir que o autor da história defina alguns parâmetros apenas de forma preliminar, sem preocupar-se em encontrar um valor ótimo. Posteriormente, o valor destes parâmetros é refinado utilizando como base a opinião de um grupo de usuários de teste.

Neste trabalho, este método foi aplicado aos parâmetros c_p, introduzidos na Seção 4.2.1, que definem o quão importante cada pista é para o desenrolar da história. O autor da história define um valor inicial para o parâmetro, c_{p_0}. Em seguida, um primeiro usuário testa a história e, ao final, informa ao sistema um valor a_{p_1}, que corresponde à sua avaliação sobre qual deve ser o valor ideal para o parâmetro c_p. Com base em a_{p_1}, o valor de c_p é ajustado para um valor c_{p_1}, e o processo é repetido com o próximo usuário de testes do grupo.

Para atualizar o valor de c_p após a avaliação do i-ésimo usuário, foi utilizada a fórmula

$$c_{p_i} = \frac{ic_{p_{i-1}} + a_{p_i}}{i+1},$$

que equivale à média aritmética de todas as avaliações realizadas até então, incluindo o valor inicial estimado pelo autor da história.

Para obter a avaliação de um usuário de teste, ao final de uma execução, é mostrada uma lista de todas as pistas descobertas durante a história. Ele deve, então, indicar o quão importante considerou cada uma delas. Em seguida, os parâmetros em questão são automaticamente atualizados.

4.5 O modelo no contexto da teoria narrativa de Bal

Conforme comentado no início deste capítulo, desejava-se fundamentar o modelo em conceitos e teorias propostas por áreas ligadas à narrativa. Assim, esta seção estabelece um paralelo entre o modelo descrito acima e a teoria narrativa de Bal (Seção 3.1.1).

Um dos motivos pelos quais algoritmos de planejamento são considerados ferramentas adequadas para *Interactive Storytelling* é a estreita relação existente entre os conceitos de plano e narrativa (RIEDL; YOUNG, 2003; CHARLES et al., 2003). Neste sentido, é interessante começar observando a similaridade entre as definições de "ação" no contexto da IA e "acontecimento" na Narratologia: ambas são definidas como sendo os meios de alterar o estado do mundo. De fato, ações são um modelo bastante adequado do conceito de acontecimento. Segundo a estratificação proposta por Bal, a camada mais fundamental de uma narrativa é a fábula, definida como "uma série de acontecimentos lógica e cronologicamente relacionados". Algoritmos de planejamento operam justamente identificando relações entre ações, de modo a criar uma cadeia de causas e efeitos que atinjam um objetivo especificado. Mais uma vez, a relação entre um conceito da IA e um da Narratologia é bastante clara: planos modelam fábulas natural e apropriadamente.

Na Seção 3.1.1, comentou-se que algumas transformações podem ocorrer na seqüência de

acontecimentos quando ela é convertida de fábula para história. Neste modelo, nenhuma dessas possíveis transformações ocorre, e a fábula é utilizada diretamente como história. Em outras palavras, o plano gerado no nível da fábula é utilizado diretamente no nível do texto, sem sofrer alterações. Este é, certamente, um aspecto do modelo que pode ser futuramente melhorado.

Por fim, a camada do texto neste modelo corresponde à visualização gráfica da narrativa. Cada ação executada é dramatizada num ambiente virtual tridimensional, através da execução de animações e exibição de mensagens textuais.

A discussão apresentada nos últimos parágrafos demonstra que o modelo utilizado neste trabalho está alinhado com as idéias da Narratologia. Considera-se que esta é uma característica importante deste trabalho, pois a busca pela consistência narrativa em IS deve aproveitar o conhecimento de áreas ligadas à narrativa.

5 PROTÓTIPO

> *The goal of Computer Science is to build something that will last at least until we've finished building it.*
> — `fortune(6)`

O modelo descrito no capítulo anterior foi implementado em um protótipo chamado Fabulator. O Fabulator foi desenvolvido em ambiente GNU/Linux, nas linguagens C++ (STROUSTRUP, 2000) e Lua (IERUSALIMSCHY, 2003). O presente capítulo detalha aspectos relativos a este protótipo.

5.1 Definição de uma história

O Fabulator não executa uma história fixa. Ao invés disso, todas as informações necessárias para a definição de uma história são lidas de arquivos de dados. Desta forma, diferentes histórias podem ser executadas sem a necessidade de alterações ou recompilação do Fabulator. As subseções a seguir descrevem os principais arquivos de dados que servem de entrada para o Fabulator.

5.1.1 Informações gerais sobre a história

Um arquivo chamado `Storyworld.lua` especifica informações gerais relativas à história, tais como título, modelo 3D a ser utilizado como cenário e o arco de tensão ideal, desejado pelo autor. A Figura 19 mostra um exemplo.

```
Metadata = {
   Title = "A Estória de Ugh 2",
   Description = "A saga continua.",
}

SceneryModelFile = "Scenery/Neandertown.3ds"

DesiredTensionArc = {
   [0] = 0,     -- no tempo zero, a tensão ideal é zero
   [45] = 10,   -- aos 45s, a tensão deve ir a dez
   [70] = 25,   -- aos 70s, a tensão deve ir a 25
}
```

Figura 19: Exemplo de arquivo `Storyworld.lua` para definição de uma história.

5.1.2 Definição do problema de planejamento

Um segundo arquivo, `Problem`, contém definições da história utilizadas pelo algoritmo de planejamento, tais como ações disponíveis, estado inicial do mundo e estados objetivo. O Apêndice C apresenta a especificação completa do formato deste arquivo. A seguir, são destacados apenas alguns pontos relacionados à forma com que estas informações são utilizadas pelo Fabulator.

Três tipos são tratados de forma especial pelo protótipo: `Character`, `Place` e `Clue`. Objetos destes tipos são tratados pelo sistema como, respectivamente, personagens, lugares e pistas. Cada personagem definido no arquivo `Problem` deve possuir um arquivo Lua correspondente, definindo alguns aspectos relativos a este personagem, como seu nome, posição inicial e aparência. Um exemplo é mostrado na Figura 20. O arquivo indicado no campo `ModelConfigurationFile` contém informações sobre o modelo 3D a ser utilizado (por exemplo, que malha de polígonos, materiais e animações serão utilizadas).

```
Name = "Ugh"
Protagonist = true
StartingPosition = { -26.2, 3.5, -1.3 }
StartingOrientation = { 0.0, 0.0, -0.6 }
ModelConfigurationFile = "Data/Stories/Ugh/Characters/Ugh.cfg"
```

Figura 20: Exemplo de arquivo para definição de um personagem.

Objetos definidos como sendo do tipo `Place` precisam ter um campo "`position`" na seção "`DATA`" (Apêndice C) de sua especificação, indicando a posição no espaço 3D que corresponde a este local. Esta informação é utilizada para fazer a ligação entre o módulo responsável pelo planejamento com os módulos responsáveis pelo planejamento e pela visualização. Pistas devem ser definidas como objetos do tipo `Clue` na especificação da história.

Os predicados `At`, `Has` e `Knows` também tem um sentido especial para o Fabulator: são utilizados para detectar a localização de objetos e personagens, bem como para aferir quais objetos estão em posse de quais personagens. É com base nos predicados `Has` presentes no estado mundo que o inventário de objetos carregados pelo protagonista (Seção 5.3) é construído. O predicado `Knows` é utilizado para indicar quais personagens conhecem quais pistas. Durante a execução, o estado do mundo é monitorado para verificar a inclusão de novos predicados deste tipo, para detectar em que instante o usuário descobriu novas pistas.

Com relação a ações, espera-se que exista uma ação `GoTo` definida no arquivo `Problem`, e que esta seja a única forma possível para alteração da localização dos personagens no ambiente. Ações de pretexto e ações de intervenção (introduzidas na Seção 4.3.2) são diferenciadas de ações "normais" por possuírem um campo "`DATA`" com o conteúdo "`pretext_action=true`" ou "`intervention_action=true`".

O autor de uma história pode ainda associar *scripts* na linguagem Lua (IERUSALIMSCHY, 2003) quando um personagem iniciar ou concluir a execução de uma ação. Estes *scripts* podem ser utilizados, por exemplo, para mostrar um texto ao usuário indicando os resultados da ação executada.

5.2 Visualização

O Fabulator exibe as histórias através de gráficos tridimensionais. A visualização é baseada na biblioteca *Open Scene Graph* (OSG) (OSFIELD, 2006), que provê uma camada escalável, portável, eficiente e facilmente estensível sobre OpenGL. Além disso, o OSG provê mecanismos para carga de modelos 3D e imagens em diversos formatos e permite que as aplicações facilmente façam uso de dispositivos de saída estereoscópicos (ativos, passivos e anaglíficos). Personagens são animados com o uso da biblioteca CAL3D (HEIDELBERGER, 2006), que oferece recursos para animação baseada em esqueletos, utilizando recursos como deformação de malhas (*skinning*) e combinação de diferentes animações (*motion blending*). Por fim, elementos de interface gráfica, tais como janelas e botões, foram implementados utilizando a biblioteca CEGUI (EDDIE, 2006). A Figura 21 mostra duas telas do Fabulator.

Figura 21: Duas telas do Fabulator: à esquerda, a tela de abertura; à direita, uma tela durante a execução de uma história.

5.3 Interface com o usuário

A maior parte da interação do usuário com o Fabulator se dá através do *mouse*. Toda vez que o cursor do *mouse* é colocado sobre algo que possa ativar uma ação, é mostrado um pequeno texto indicando este fato. Para ordenar que o protagonista da história caminhe até um ponto, basta clicar sobre o ponto desejado (o texto "Andar até aqui" é mostrado quando esta ação é possível). O sistema utiliza o algoritmo A* (RUSSELL; NORVIG, 2002) para computar uma trajetória até o ponto desejado. A trajetória gerada passa apenas por polígonos aproximadamente alinhados com o plano do solo, de modo a evitar que personagens subam paredes, por exemplo.

Outras ações são executadas a partir de um clique em objetos ou personagens que estejam visíveis na tela. Ao clicá-los, é mostrada ao usuário uma lista de todas as ações relacionadas ao objeto ou personagem clicado. A Figura 22 mostra um exemplo: após clicar em um personagem da história, foi exibida uma lista de ações relativas a este personagem, como por exemplo falar-lhe algo, ou presentear-lhe com objetos. A lista inclui todas as ações relacionadas à entidade clicada cujo sujeito é o protagonista da história e que possuem os pré-requisitos satisfeitos no momento.

Figura 22: Lista de ações exibida após clique em um personagem da história. O inventário de objetos possuídos pelo protagonista da história também pode ser visto, na parte superior da imagem.

Objetos carregados pelo protagonista são exibidos em um "inventário" na parte superior da tela. No canto superior esquerdo é exibido um hexágono que representa o local onde o protagonista se encontra no momento, a partir do qual podem ser executadas ações como "examinar o local". Isto também pode ser visto na Figura 22.

Por padrão, a câmera segue o protagonista da história a uma distância fixa. Mas, usando as setas do teclado, o usuário tem a possibilidade de reposicioná-la. Este recurso é particularmente útil para os momentos em que algum objeto acaba ficando entre a câmera e o protagonista.

5.4 Planejamento e detecção de situações sem saída

Conforme discutido na Seção 3.3.3, o algoritmo de planejamento utilizado neste trabalho é o Metric-FF (HOFFMANN, 2003). O protótipo utiliza a implementação fornecida pelos próprios autores deste algoritmo.

O mecanismo utilizado para antever situações sem saída (descrito na Seção 4.3.1) foi implementado através de uma busca no espaço de estados do mundo usando a estratégia *Iterative Deepening Search* (IDS) (RUSSELL; NORVIG, 2002), que oferece um bom equilíbrio em termos de tempo de execução e memória requerida.

Para explorar arquiteturas multiprocessadas e com múltiplos núcleos (*multicore*), vários estados são expandidos e testados simultaneamente, com o uso de múltiplas *threads*. O número de *threads* utilizadas para a detecção de situações sem saída é determinado por uma entrada no arquivo de configuração do Fabulator.

5.5 Ajuste de parâmetros com base na avaliação de usuários

Uma opção no arquivo de configuração do protótipo permite habilitar o modo de refino de parâmetros, que implementa a funcionalidade descrita na Seção 4.4. Se este modo estiver ativado, ao final da história é mostrada ao usuário uma tela que lhe permite avaliar cada uma das pistas descobertas (Figura 23). Uma vez que a avaliação é concluída, os valores indicados pelo usuário são salvos em um arquivo. Na próxima execução da história, os valores dos parâmetros c_p serão atualizados com base nesta avaliação, conforme especificado na Seção 4.4.

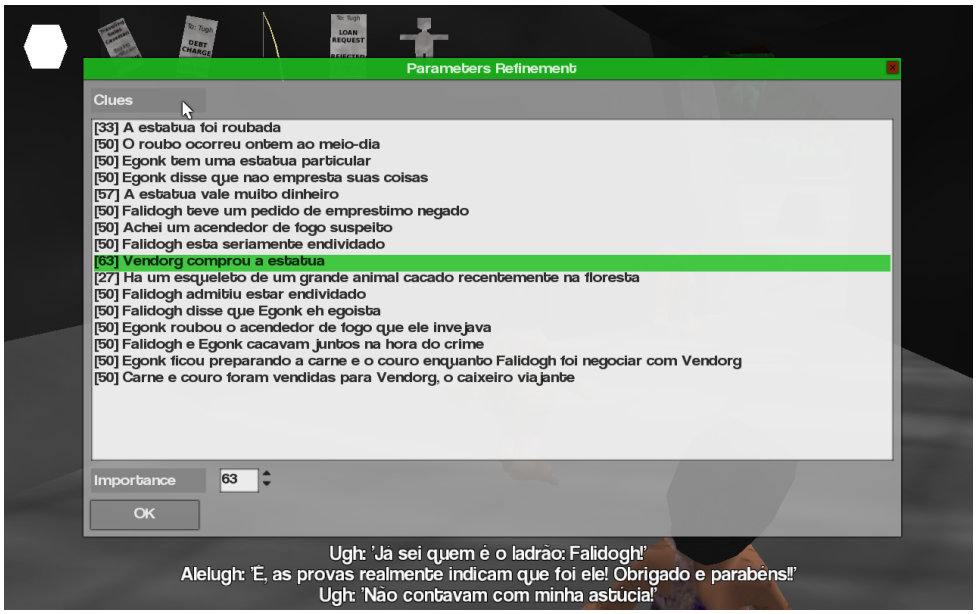


Figura 23: Tela com o "questionário" para ajuste dos parâmetros c_p.

5.6 Auxílios para depuração e análise de resultados

O protótipo inclui uma série de recursos que facilitam a depuração de histórias e a análise de resultados. Um modo especial do Fabulator permite aferir o estado atual do mundo, o plano corrente (Figura 24) e a ação sendo executada por um NPC. Este modo de depuração ainda permite ver as regiões consideradas caminháveis (formadas pelos polígonos com inclinação inferior a 45° em relação à horizontal), testar o módulo de planejamento de trajetórias dos personagens e verificar a localização no espaço tridimensional dos lugares definidos no arquivo que especifica o problema de planejamento.

Com relação à análise de resultados, o Fabulator cria um arquivo de registro (*log*) para cada execução de história. Este arquivo contém todos os eventos relevantes, incluindo execução de ações pelo usuário e pelos NPCs, planos gerados e situações sem saída antevistas. Além deste registro textual, o Fabulator gera uma figura incluindo o arco de tensão desejado e o observado durante a história, com marcas identificando que pistas foram responsáveis pelo aumento do nível de tensão em um determinado ponto do arco de tensão observado. As figuras 26, 27 e 28 (apresentadas no Capítulo 6) foram produzidas a partir destas figuras geradas automaticamente.

Figura 24: O modo de depuração do Fabulator, com o plano corrente sendo mostrado no terminal.

6 RESULTADOS

> *E tu non t'incantare troppo su queste teche. Di frammenti della croce ne ho visti molti altri, in altre chiese. Se tutti fossero autentici, Nostro Signore non sarebbe stato suppliziato su due assi incrociate, ma su di una intera foresta.*
> — *Guglielmo da Baskerville*

Este capítulo apresenta resultados obtidos com o uso do Fabulator (o protótipo descrito no capítulo anterior). Primeiramente, é apresentada uma descrição da história criada para a realização dos experimentos. Em seguida, são expostos resultados obtidos através de testes realizados com usuários do sistema.

6.1 A Estória de Ugh 2

A Estória de Ugh 2 é uma versão estendida d'*A Estória de Ugh* original (BARROS, 2004). Esta nova versão foi criada especialmente para a realização de experimentos com usuários para avaliação do modelo apresentado nesta dissertação. Assim como a versão original, esta versão estendida segue o enredo mestre Enigma (Seção 3.1.3). Esta seção descreve os principais aspectos da *Estória de Ugh 2* em alto nível. Uma descrição formal completa desta história é apresentada no Apêndice D.

A história está ambientada na comunidade de trogloditas de Neandertown, na idade da pedra. Os habitantes desta comunidade reverenciavam uma estátua, que era deixada sobre uma espécie de altar localizado próximo às cavernas habitadas pelos habitantes da comunidade. Um dia, Alelugh (o sacerdote de Neandertown) é atingido por um golpe de tacape e a estátua é roubada. Recomposto do golpe, Alelugh contrata Ugh, um conhecido detetive, para investigar o crime e descobrir o culpado. Outros dois personagens, ambos moradores de Neandertown participam da história: Falidogh e Egonk. Vendorg, o caixeiro viajante, também é mencionado durante o desenrolar dos acontecimentos, mas não aparece "pessoalmente" na história.

Assim, n'*A Estória de Ugh 2*, o usuário assume o papel do detetive Ugh e deve solucionar o caso do desaparecimento da estátua. Durante a história, o usuário irá descobrir que a estátua foi roubada por Falidogh, que estava seriamente endividado. Aproximadamente na hora do crime, Falidogh estava com Egonk caçando na floresta. Porém, quando conseguiram caçar um grande animal, decidiram vender a carne e o couro para Vendorg, que estava em Neandertown. Enquanto Egonk preparava a carne e o couro do animal caçado, Falidogh disse que iria até o porto onde o barco de Vendorg estava atracado para acertar a venda. Porém, passou antes pelo altar, atingiu Alelugh, roubou a estátua e vendeu-a secretamente para Vendorg. Para dar maior interesse à história, foram incluídas algumas pistas que fazem o usuário desconfiar também de Egonk, que revela-se bastante egoísta, tendo inclusive roubado no passado um acendedor de fogo que invejava. Ao todo, há 16 pistas n'*A Estória de Ugh 2*, que estão listadas na Tabela 5.

Tabela 5: Pistas d'*A Estória de Ugh 2*.

Número	**Pista**
1	A estátua foi roubada.
2	Falidogh teve um empréstimo negado.
3	Falidogh está endividado.
4	Falidogh admitiu estar endividado.
5	A estátua vale muito dinheiro.
6	Egonk tem uma estatueta pessoal.
7	Existe um acendedor de fogo suspeito.
8	Egonk roubou um acendedor de fogo que ele invejava.
9	Egonk disse que não empresta suas coisas.
10	Falidogh disse que Egonk é egoísta.
11	O crime ocorreu ontem ao meio-dia.
12	Falidogh e Egonk caçavam juntos na hora do crime.
13	Há um esqueleto de um animal caçado recentemente na floresta.
14	Carne e couro foram vendidas para Vendorg.
15	Egonk preparava a carne e o couro enquanto Falidogh negociava com Vendorg.
16	Vendorg comprou a estátua.

A seqüência de três objetivos para o algoritmo de planejamento (cada um correspondendo a um ato da história, conforme discutido na Seção 4.1) foi especificada de modo que *A Estória de Ugh 2* siga a estrutura para histórias do tipo Enigma descrita por Tobias (1993):

1. Apresentação de fatos genéricos, como pessoas, locais e eventos. No caso, durante a etapa inicial da história, o usuário fica sabendo as pistas 1 a 9 da Tabela 5. Assim, ao final do primeiro ato, o usuário deve estar desconfiando tanto de Falidogh quanto de Egonk.

2. Revelação de fatos específicos, de uma descrição detalhada de como as pessoas, locais e eventos estão relacionados entre si. N'*A Estória de Ugh 2*, isto corresponde a descobrir as pistas de 10 a 15, em que a seqüência de eventos do roubo durante a caçada fica clara.

3. A solução do enigma, em que a verdade sobre o fato obscuro é revelada. Na história em questão, isto corresponde à descoberta da pista 16 e à acusação de Falidogh como o culpado.

Convém observar que o Fabulator (e modelo no qual o Fabulator está baseado) não impõe nenhuma restrição quanto à ordem em que as pistas devem ser descobertas: a idéia é justamente que o usuário tenha liberdade de investigar o mistério da forma que desejar. Deste modo, é possível, por exemplo, que o usuário descubra a pista 16 logo no início da história. Contudo, o modelo com a história organizada em atos garante que os NPCs só atuem de modo a atingir os objetivos do ato corrente, favorecendo a estrutura clássica em três atos. De qualquer forma, *A Estória de Ugh 2* foi pensada de modo que ela faça sentido independentemente da ordem em que as pistas são descobertas.

Com relação ao final da história, o usuário tem à disposição uma ação *Acusar*, utilizada para dizer a Alelugh quem ele acha que foi responsável pelo roubo da estátua. Para evitar que

o usuário possa simplesmente acusar um suspeito qualquer sem a devida investigação, definiu-se que é necessário "convencer" Alelugh da culpa do suspeito. Para tanto, o usuário deve contar ao sacerdote todas as pistas relevantes para o esclarecimento do enigma antes de acusar o culpado. Caso o usuário acuse o suspeito errado, ou faça uma acusação antes de ter contado todas as pistas relevantes, Alelugh simplesmente diz que "não há evidências suficientes". Não foi incluído nenhum limite quanto ao número de "acusações erradas" que podem ser feitas.

Foram incluídas duas ações de intervenção e uma ação de pretexto na história. A ação de pretexto d'*A Estória de Ugh 2* é a ação *LetItRain*, que representa uma chuva em Neandertown. As ações de intervenção, por sua vez, são *GrowNewStrawberry* e *DropAndDirty*. *GrowNewStrawberry* faz um novo morango nascer na horta, o que pode evitar uma situação sem saída que ocorreria se o usuário desse o morango para Egonk ou Alelugh, quando ele deveria ser dado a Falidogh. *DropAndDirty* faz o protagonista deixar um objeto cair numa poça de lama, sujando-o. Objetos sujos não podem ser dados de presente a outros personagens, de modo que esta ação pode evitar que o usuário dê algum objeto importante a um NPC não cooperativo, o que também levaria a uma situação sem saída.

6.2 Experimentos com usuários

Para avaliar o modelo proposto neste trabalho, um grupo de nove usuários testou *A Estória de Ugh 2* no Fabulator (o número de participantes nos experimentos foi limitado pelo tempo disponível para a sua realização). Os quatro primeiros usuários foram utilizados para testes preliminares e ajuste dos parâmetros c_p (conforme descrito na Seção 4.4). Os demais, utilizaram o sistema como se fossem usuários finais.

Cada usuário participante dos experimentos recebeu uma descrição da história que jogaria, seu objetivo e instruções de como controlar o protagonista. Em seguida, jogou *A Estória de Ugh 2* e, após, foi entrevistado. As entrevistas foram conduzidas de modo a permitir que os usuários falassem com a maior liberdade possível sobre os pontos positivos e negativos de sua experiência.

6.2.1 Usuários de teste

Os quatro primeiros usuários foram utilizados para testes preliminares d'*A Estória de Ugh 2*. Estes usuários responderam ao questionário para avaliação relativa aos valores que consideram ideais para os parâmetros c_p (este questionário é mostrado na Figura 23, na Seção 5.5). A Tabela 6 mostra as respostas dos usuários de teste. A última linha desta tabela contém a média dos valores sugeridos pelos usuários, ou seja, o valor efetivamente utilizado para os parâmetros nos experimentos realizados com os usuários finais.

Este grupo inicial de usuários também permitiu que fossem feitas observações com relação ao tempo aproximado que seria levado para completar a história. Com base nestas observações, foi definido o arco de tensão ideal para *A Estória de Ugh 2*, ilustrado na Figura 25. Com relação a esta figura, cabe observar que o número de "degraus" na curva que define o arco de tensão desejado não precisa ser necessariamente igual ao número de pistas da história. Este desacoplamento permite que pistas sejam incluídas ou excluídas durante o processo de autoria, sem exigir mudanças no arco desejado.

Tabela 6: Respostas dos usuários de teste quanto aos valores que consideram ideais para os parâmetros c_p. Os valores mostrados estão normalizados de modo que o somatório de todos os c_p seja igual à tensão total da história, especificada pelo autor (que, no caso, vale 80). Os números identificando as pistas correspondem aos mostrados na Tabela 5.

	Pistas															
Usuário	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0,6	6,3	6,1	6,1	6,1	4,6	5,1	5,3	4,8	3,1	3,1	5,0	5,4	5,5	6,4	6,3
2	3,6	3,6	7,2	3,6	5,8	5,8	5,8	5,8	3,6	5,1	3,6	3,6	4,4	7,2	3,6	7,2
3	8,0	3,2	5,6	4,0	5,6	5,2	2,4	4,8	4,0	4,8	4,0	6,4	2,4	5,6	7,2	7,2
4	3,9	4,7	4,7	4,3	4,2	3,9	5,5	7,1	3,9	3,9	3,9	5,5	4,3	7,5	5,1	7,5
Média	4,0	4,5	5,9	4,5	5,4	4,9	4,7	5,8	4,1	4,2	3,7	5,1	4,1	6,5	5,6	7,1

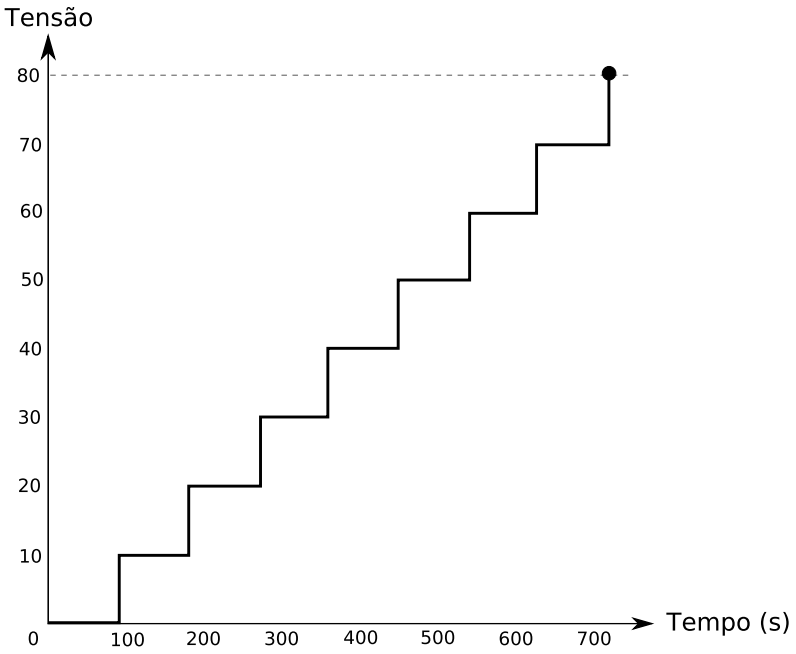


Figura 25: Arco de tensão desejado para *A Estória de Ugh 2*.

6.2.2 Usuários finais

O segundo grupo de usuários foi tratado como um grupo de "usuários finais" do Fabulator e d'*A Estória de Ugh 2*. Abaixo, são apresentados alguns resultados obtidos nestes experimentos.

6.2.2.1 Discrepância total

Os experimentos foram organizados de modo a permitir uma avaliação quantitativa da parte do modelo que busca fazer com que a história siga um arco de tensão especificado pelo autor. Para tanto, esta funcionalidade foi desativada para metade dos usuários. Desta forma, tornou-se possível comparar os valores de E (a discrepância total da história, definida na Seção 4.2.2) obtidos utilizando ou não as técnicas apresentadas neste trabalho.

Observou-se que o uso do modelo apresentado neste trabalho reduziu consideravelmente a

discrepância total das histórias geradas. Em média, as histórias produzidas com o modelo completamente habilitado apresentaram $E = 6581,6$. Desabilitando a funcionalidade que busca fazer com que os arcos sejam seguidos, o valor aumentou para $E = 47527,9$. Os valores de E para cada um dos usuários deste segundo grupo é apresentado na Tabela 7.

Tabela 7: Discrepância total (E) observada nos experimentos com usuários finais.

Usuário	**Modelo habilitado?**	***E***
5	S	9337,51
6	N	63804,00
7	N	31251,80
8	S	4434,91
9	S	5972,39

6.2.2.2 Análise das histórias resultantes

Aqui é apresentada uma análise das histórias resultantes dos experimentos com os usuários. O objetivo é apontar situações reais em que determinados aspectos do modelo descrito neste trabalho puderam ser observados trabalhando. Para facilitar a compreensão, os trechos de histórias apresentados a seguir incluem apenas os trechos e eventos relevantes à compreensão do aspecto específico que está sendo discutido. Os tempos dos eventos listados estão no formato "*minutos*:*segundos*", contados a partir do início da história.

O primeiro trecho, reproduzido abaixo, faz parte da história gerada pelo usuário número 5. Ele demonstra uma situação em que o sistema detecta que o nível de tensão da história está consideravelmente acima do desejável.

01:02 Neste ponto da história, o plano determina que várias ações "importantes" devem ser executadas por um NPC: Egonk deverá examinar dois locais (a caverna de Falidogh e a horta), onde encontrará pistas que indicam que Falidogh está seriamente endividado. Depois, Egonk deverá contar estas pistas ao protagonista Ugh. O plano completo (que foi gerado utilizando um valor igual a 100 para as ações dos NPCs) é mostrado abaixo.
$ExaminePlace(Egonk, FalidoghsCave)$
$Take(Egonk, LoanRequest, FalidoghsCave)$
$ExamineThing(Egonk, LoanRequest)$
$GoTo(Egonk, FalidoghsCave, Garden)$
$ExaminePlace(Egonk, Garden)$
$Take(Egonk, DebtCharge, Garden)$
$ExamineThing(Egonk, DebtCharge)$
$GoTo(Egonk, Garden, Patio)$
$TalkAbout(Egonk, Ugh, FalidoghHadLoanRejected)$
$TalkAbout(Egonk, Ugh, FalidoghIsIndebted)$
$TalkAbout(Ugh, Falidogh, FalidoghIsIndebted)$
$Take(Ugh, PersonalStatue, Patio)$
$GivePresent(Ugh, Egonk, PersonalStatue)$
$GoTo(Ugh, Patio, EgonksCave)$
$ExaminePlace(Ugh, EgonksCave)$
$Take(Ugh, FireStarter, EgonksCave)$

$GoTo(Ugh, EgonksCave, AlelughsCave)$
$GivePresent(Ugh, Alelugh, FireStarter)$

01:03 Egonk, conforme indicado no plano, examinou a caverna de Falidogh, onde encontrou um documento.

01:04 Egonk pegou o documento encontrado.

01:04 Egonk examinou o documento encontrado e verificou que tratava-se de um pedido de empréstimo negado a Falidogh.

01:18 Egonk andou até o pátio, onde o protagonista se encontrava.

01:20 Egonk, sempre seguindo o plano, contou ao protagonista que Falidogh teve um empréstimo negado. Esta é pista número 2 da história (Tabela 5). Neste instante, o sistema verifica que o nível corrente de tensão está consideravelmente acima do nível de tensão desejado para este momento da história. Por isso, é disparada a geração de um novo plano, utilizando custos para ações dos NPCs igual a 129 (o plano utilizado até então fora criado usando um custo igual a 100). Neste novo plano, as ações executadas pelos NPCs foram reduzidas quase ao mínimo possível. Em particular, as ações que fariam Egonk examinar a horta para encontrar uma nova pista e contá-la a Ugh foram removidas do plano. Ao invés disso, espera-se que o próprio protagonista execute estas ações. O plano completo é mostrado a seguir.
$GivePresent(Ugh, Egonk, PersonalStatue)$
$GoTo(Falidogh, Patio, AlelughsCave)$
$GoTo(Ugh, Patio, EgonksCave)$
$ExaminePlace(Ugh, EgonksCave)$
$Take(Ugh, FireStarter, EgonksCave)$
$GoTo(Ugh, EgonksCave, AlelughsCave)$
$GivePresent(Ugh, Alelugh, FireStarter)$
$GoTo(Ugh, AlelughsCave, Garden)$
$ExaminePlace(Ugh, Garden)$
$Take(Ugh, DebtCharge, Garden)$
$ExamineThing(Ugh, DebtCharge)$
$GoTo(Ugh, Garden, AlelughsCave)$
$TalkAbout(Ugh, Falidogh, FalidoghIsIndebted)$

A Figura 26 mostra o arco de tensão observado durante o experimento com o usuário número 5, incluindo uma indicação do momento discutido acima, em que o sistema registra uma discrepância excessivamente alta entre o arco de tensão observado e o desejado.

O segundo trecho de história, resultado do teste com o usuário número 8, mostra um caso oposto ao anterior: o Fabulator detecta que, em um determinado instante, o nível de tensão está abaixo do desejado. Como no caso anterior, um novo plano é gerado. Desta vez, porém, o novo plano faz com que os NPCs atuem para auxiliar o usuário:

03:08 Neste ponto da história, o usuário executou uma ação que invalidou o plano corrente. Assim, conforme descrito na Seção 4.1, foi criado um novo plano. Nesta altura, o nível de tensão observado estava ligeiramente acima do desejado, de modo que o plano gerado (utilizando custo igual a 103 para as ações dos NPCs) não previu que NPCs agissem de modo a auxiliar o usuário:
$GoTo(Ugh, Forest, Patio)$
$TalkAbout(Ugh, Egonk, EgonkHasAPersonalStatue)$
$GoTo(Ugh, Patio, AlelughsCave)$
$GivePresent(Ugh, Alelugh, FireStarter)$
$GoTo(Ugh, AlelughsCave, Garden)$

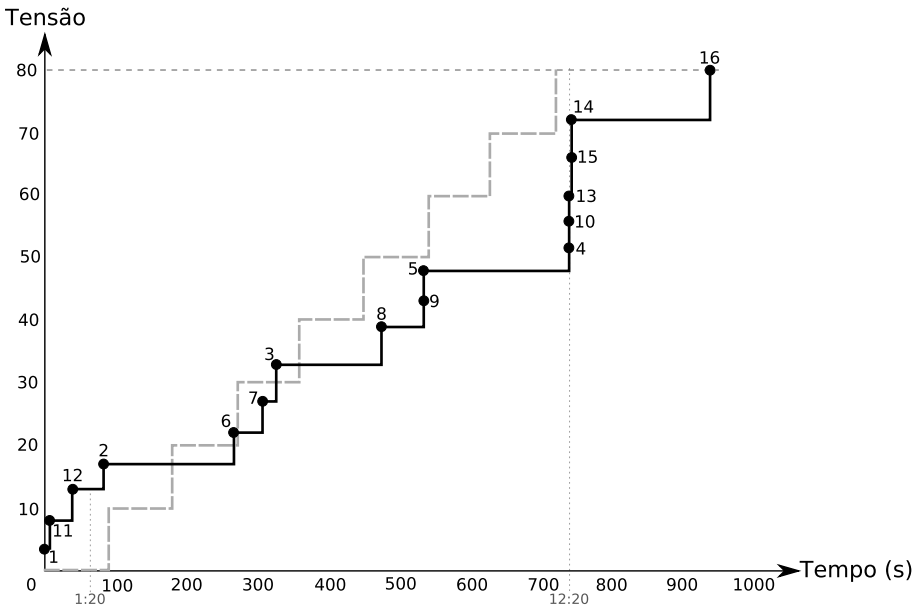


Figura 26: Arco de tensão da história produzida pelo usuário 5 (linha contínua preta). Os números ao lado dos pontos referem-se às pistas encontradas pelo usuário, conforme a Tabela 5. Como referência, o arco de tensão ideal é mostrado com uma linha tracejada, em tom mais claro. Também estão indicados os tempos mencionados no texto.

$ExaminePlace(Ugh, Garden)$
$Take(Ugh, DebtCharge, Garden)$
$ExamineThing(Ugh, DebtCharge)$
$GoTo(Ugh, Garden, AlelughsCave)$
$TalkAbout(Ugh, Falidogh, FalidoghIsIndebted)$

06:00 Durante quase 3 minutos, o jogador não foi capaz de localizar nenhuma pista nova. Assim, o nível de tensão observado tornou-se substancialmente menor que o desejado. Esta situação foi detectada e foi gerado um novo plano, com o custo das ações dos NPCs igual a 75 (contra 103 do plano anterior). O novo plano, mostrado abaixo, prevê que Egonk deverá ir até a horta e examiná-la, para descobrir uma pista e contá-la a Ugh.
$TalkAbout(Ugh, Egonk, EgonkHasAPersonalStatue)$
$GoTo(Ugh, Patio, AlelughsCave)$
$TalkAbout(Ugh, Alelugh, SuspectFireStarter)$
$GoTo(Egonk, Patio, Garden)$
$ExaminePlace(Egonk, Garden)$
$Take(Egonk, DebtCharge, Garden)$
$ExamineThing(Egonk, DebtCharge)$
$GoTo(Egonk, Garden, AlelughsCave)$
$TalkAbout(Egonk, Ugh, FalidoghIsIndebted)$
$TalkAbout(Ugh, Falidogh, FalidoghIsIndebted)$

06:14 Seguindo o plano, Egonk caminhou até a horta.

06:15 Egonk examinou a horta e encontrou um documento.

06:17 Egonk pegou o documento encontrado.

06:17 Egonk examinou o documento, e verificou tratar-se de uma cobrança de dívida endereçada a Falidogh.

06:40 Egonk andou até a caverna de Alelugh, onde Ugh se encontrava.

07:00 Egonk contou ao protagonista Ugh que Falidogh estava seriamente endividado. Com a revelação desta pista, o nível de tensão observada aproximou-se do desejado.

O arco de tensão completo para o usuário número 8 é mostrado na Figura 27. Este é um caso em que o arco de tensão observado segue o desejado particularmente bem, após os eventos descritos no trecho de história descrito acima.

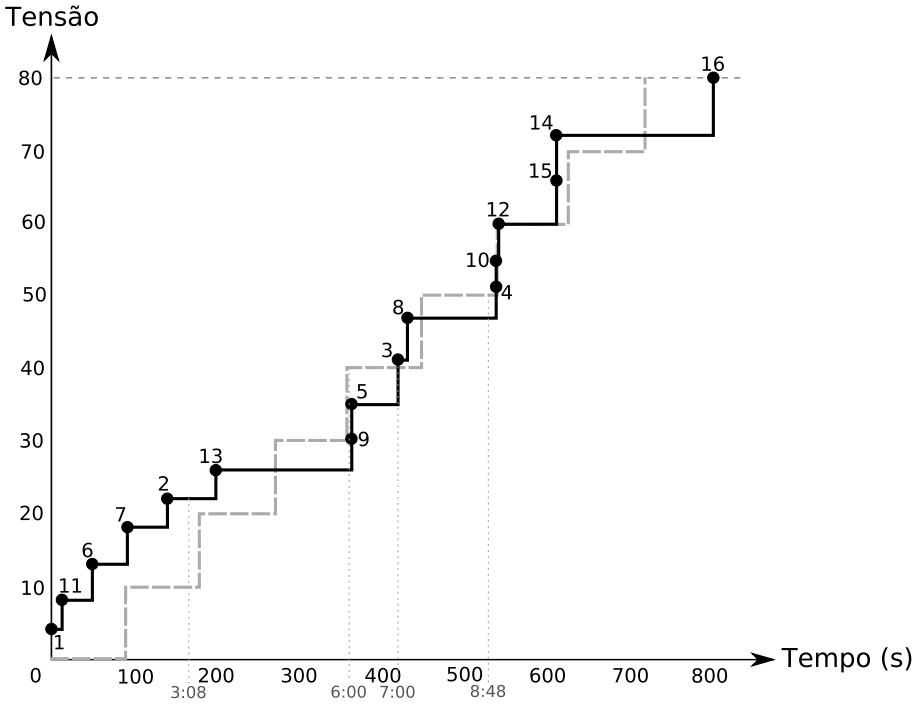


Figura 27: Arco de tensão da história produzida pelo usuário 8 (linha contínua preta). Os números ao lado dos pontos referem-se às pistas encontradas pelo usuário, conforme a Tabela 5. Como referência, o arco de tensão ideal é mostrado com uma linha tracejada, em tom mais claro. Também estão indicados os tempos mencionados no texto.

Também com o usuário 5 (Figura 26), pôde ser observado um momento em que ocorreu algo similar ao mostrado acima: próximo ao tempo 12:20, a maior parte das diversas pistas reveladas resultou de NPCs contando informações que haviam descoberto, com o objetivo de elevar o nível de tensão a um patamar próximo ao desejado.

Para ilustrar a atuação de ações de pretexto e ações de intervenção (cujo funcionamento é descrito na Seção 4.3.2), é apresentado abaixo mais um trecho da história produzida pelo usuário 8.

07:00 Já próximo ao final do primeiro ato da história, bastaria ao usuário conversar com os suspeitos a respeito de algumas pistas que poderiam incriminá-los para que a história avançasse à sua segunda etapa. Assim, o plano para este momento da história consistia simplesmente de duas ações:
$TalkAbout(Ugh, Falidogh, FalidoghIsIndebted)$
$TalkAbout(Ugh, Alelugh, SuspectFireStarter)$

08:21 Após algum tempo de busca (trabalhando em segundo pano, em *threads* dedicadas a isso), o sistema detecta que a história entrará numa situação sem saída caso o usuário execute a seguinte seqüência de ações:

1. Assustar o personagem Falidogh. Falidogh irá ficar brabo com o protagonista e irá se recusar a conversar com ele. Deve-se lembrar que é necessário conversar com Falidogh para concluir o primeiro ato da história.
2. Dar o morango (que ele possuía) a Egonk. A única forma de fazer as pazes com Falidogh era dar-lhe o morango de presente. Como o único morango da história estaria sendo dado a outro personagem, não haveria como voltar a ser amigo de Falidogh e, portanto, não haveria como concluir a história.

08:48 Com o pretexto de que a chuva irrigou a terra, o sistema fez um novo morango crescer na horta. Deste momento em diante, a história não entrará mais numa situação sem saída caso o usuário assuste Falidogh e dê o morango a Egonk, pois haverá um novo morango que poderá ser colhido e dado a Falidogh. Deve-se observar que nada impede que a ação *GrowNewStrawberry* seja executada novamente num momento futuro da história. Assim, caso fosse necessário, novos morangos poderiam ter sido criados.

Para concluir a análise das histórias geradas pelos usuários, a Figura 28 mostra o arco de tensão observado no experimento com o usuário número 7. Este foi um dos usuários que utilizou o sistema com a parte que busca adaptar os arcos de tensão desabilitada. É possível observar claramente que a partir de um certo ponto (aproximadamente no tempo 250 segundos), o usuário progrediu muito mais lentamente que o desejado. E, como o sistema não reagiu a isto, a discrepância total observada na história foi muito alta.

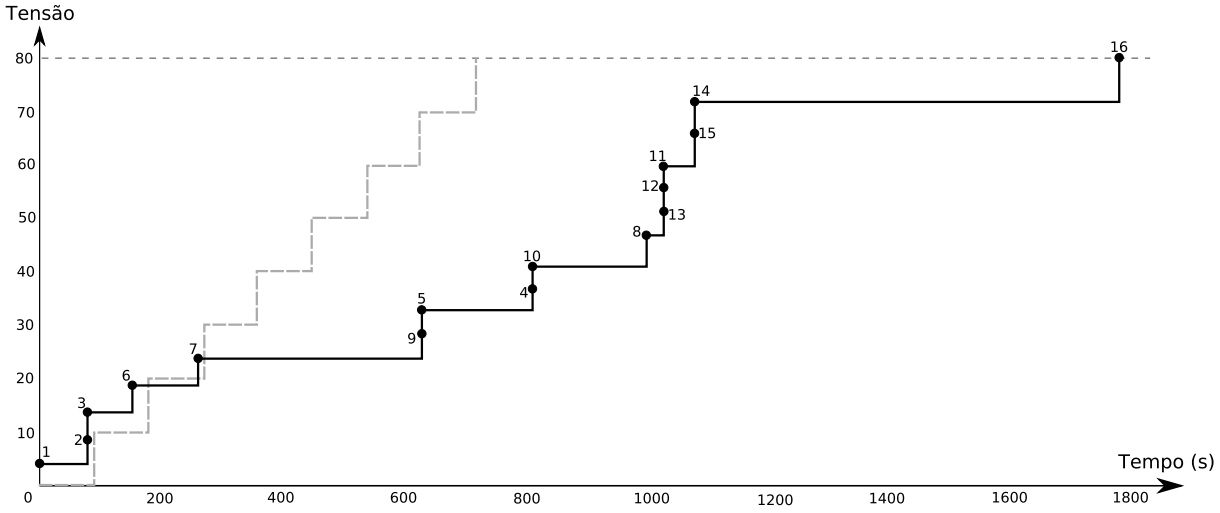


Figura 28: Arco de tensão da história produzida pelo usuário 7 (linha contínua preta). Os números ao lado dos pontos referem-se às pistas encontradas pelo usuário, conforme a Tabela 5. Como referência, o arco de tensão ideal é mostrado com uma linha tracejada, em tom mais claro.

6.2.3 Observações adicionais

É muito importante que sistemas interativos sejam avaliados por usuários. Entretanto, são extremamente raros na literatura os relatos de sistemas de *Interactive Storytelling* sendo testados desta maneira. Assim, considera-se importante relatar alguns pontos observados durante os experimentos realizados e comentários feitos pelos usuários nas suas entrevistas. É interessante notar que estas observações e comentários dos usuários podem ser divididos em dois grupos:

alguns se referem a aspectos do modelo em si; outros, relacionam-se a aspectos "periféricos", como a interface com o usuário do protótipo.

O Fabulator relata ao usuário diversos eventos através de mensagens textuais na parte inferior da tela. Isso ocorre, por exemplo, quando NPCs executam certas ações ou falam algo ao protagonista. Alguns usuários não perceberam mensagens importantes. Outros, que levaram a história para alguma situação em que apareceram muitas mensagens em um curto intervalo de tempo, relataram ter tido dificuldades para acompanhá-las.

Conforme descrito na Seção 6.1, para que o usuário concluísse *A Estória de Ugh 2*, ele deveria descobrir todas as pistas, contá-las ao sacerdote e fazer a acusação. O objetivo deste protocolo, evitar que usuários simplesmente arriscassem um palpite antes de concluir a investigação, foi atingido. Porém, houve efeitos colaterais. Primeiramente, a maioria dos usuários levou bastante tempo para concluir a história após ter descoberto todas as pistas: hesitaram em fazer a acusação, ou simplesmente preferiram procurar mais pistas (que não existiam). Além disso, a maioria dos usuários teve dificuldade de lembrar quais pistas já haviam sido contadas ao sacerdote, e precisou contar novamente todas as pistas, uma a uma, para poder fazer a acusação com sucesso.

A maior parte dos usuários conseguiu compreender a história sem grande dificuldade. Um usuário comentou que "pelas leis do Direito, acho que as provas ali não seriam suficientes, mas tudo indicava que era ele [o culpado]". Um dos usuários foi até o final da história achando, equivocadamente, que o criminoso era o Egonk, que "não tinha um caráter muito bom."

Muitos dos usuários demonstraram ter criado uma expectativa de que os NPCs apresentassem comportamentos mais complexos. Muitos reclamaram explicitamente que os NPCs sempre usavam a mesma "resposta padrão" quando conversavam sobre um assunto que não tivesse uma resposta especial programada. Um usuário afirmou que gostaria de ter podido "ir jogando um contra o outro até que o culpado acabasse se entregando". Outro, perguntou se era possível "fazer eles [os NPCs] conversar [entre eles]". Um terceiro afirmou que "suspeitava que eles fossem cúmplices, e se contasse alguma coisa do Falidogh pro Egonk, o Egonk ia contar pro Falidogh e daí não ia me dar mais pistas."

Foi possível observar alguns problemas com relação à modelagem do cenário d'*A Estória de Ugh 2*. O "pier", um dos locais onde havia uma pista importante, foi considerado "muito escondido" por alguns usuários. Em algumas situações, foi possível notar que não estava claro para os usuários que um determinado ponto do cenário era considerado um local diferente de outro ponto próximo. Por exemplo, alguns demoraram para perceber que o "pier" e a "praia" eram locais distintos (embora próximos).

Nenhum usuário mencionou as ações de pretexto e intervenção executadas durante a história. Considera-se isso um fato bastante positivo, pois uma das principais motivações do método apresentado era resolver situações sem saída de uma forma mais sutil que abordagens descritas na literatura.

Por fim, os usuários que testaram o Fabulator com o recurso de ajuste dos arcos de tensão desligado passaram por pelo menos um longo período em que não conseguiram progredir na história. Durante estes períodos, expressaram claramente sensações desagradáveis, do tédio à irritação. Neste sentido, os experimentos com usuários indicam que o método proposto para fazer as histórias seguir um arco de tensão idealizado pelo autor da história foi capaz de manter o interesse dos usuários na narrativa.

7 CONCLUSÕES

Smoke! That means there is a village nearby... or a fire. Or a village on fire!
— Groo, the Wanderer

Este texto apresentou um modelo para *Interactive Storytelling* baseado em planejamento que busca gerar histórias mais consistentes narrativamente através de mecanismos que buscam atingir três objetivos:

1. Fazer com que as histórias geradas sigam um arco de tensão desejado, especificado pelo autor da história.
2. Evitar situações sem saída através de intervenções na história de maneira transparente ao usuário.
3. Permitir que certos parâmetros da história sejam refinados por um grupo de usuários de teste, reduzindo a carga de trabalho sobre o autor da história.

A bibliografia na área já contava com trabalhos que buscavam enfrentar os mesmos problemas. As abordagens adotadas neste trabalho, contudo, apresentam contribuições nestes três pontos. Mateas e Stern (2003) já haviam trabalhado buscando fazer com que as histórias geradas pelo sistema de IS Façade seguissem um arco de tensão aristotélico. Porém, os eventos que compõem as histórias do Façade são selecionados de maneira heurística, e não através de um algoritmo de planejamento. Deste modo, a abordagem utilizada por Mateas e Stern não é aplicável a sistemas de IS baseadas em planejamento. Young (1999) já havia identificado a possibilidade de haver situações sem saída em IS baseada em planejamento e proposto uma solução para o problema. A solução apresentada no presente trabalho, contudo, intervém nas histórias de uma maneira mais sutil e justificada por eventos ocorridos anteriormente, de maneira que impõem um impacto menor à consistência narrativa. Da mesma forma, trabalhos anteriores buscaram simplificar a tarefa de autoria de conteúdo para sistemas de IS, mas através do uso de interfaces gráficas específicas para este fim (CRAWFORD, 2004; DONIKIAN; PORTUGAL, 2004).

O modelo apresentado neste trabalho pode ser enquadrado dentro da teoria narrativa de Bal, uma pesquisadora da área da Narratologia, conforme descrito na Seção 4.5. Uma das premissas deste trabalho é que modelos de IS devem buscar inspiração em estudos de áreas ligadas à narrativa. Por isso, acredita-se que uma característica importante do modelo é o fato dele ser "narratologicamente plausível".

Um protótipo foi desenvolvido, implementando as idéias aqui apresentadas, e este protótipo foi utilizado para a realização de testes com usuários. Dentre todos os trabalhos na área de IS

estudados, sabe-se de apenas dois que foram realmente utilizados por pessoas que não participaram do seu desenvolvimento.[1] Assim, acredita-se que a realização destes experimentos seja um indicativo de que o trabalho atingiu um nível de maturidade considerável. A amostra de usuários estudada é reconhecidamente pequena, incapaz de oferecer qualquer garantia estatística sobre os resultados obtidos. Ainda assim, a análise apresentada no Capítulo 6 indica que o modelo é capaz de produzir resultados satisfatórios.

O próprio protótipo desenvolvido é um resultado importante do trabalho, já que possui potencial de servir de base para futuros trabalhos na área. O protótipo foi desenvolvido utilizando bibliotecas disponíveis livremente (OSG (OSFIELD, 2006) para visualização e renderização, CAL3D (HEIDELBERGER, 2006) para animação, CEGUI (EDDIE, 2006) para interface gráfica com o usuário e Lua (IERUSALIMSCHY, 2003) para configuração e *scripting*) e a implementação do algoritmo de planejamento Metric FF disponibilizada pelos próprios criadores do algoritmo. O autor desta dissertação foi responsável pela integração destas ferramentas, bem como pelo desenvolvimento das demais partes do protótipo, incluindo o *parser* para a linguagem de especificação de histórias (Apêndice C), rotinas para manutenção e atualização do estado do mundo, ferramentas para depuração e análise de histórias e todo o modelo de IS descrito no Capítulo 4.

Foi possível observar claramente que o método utilizado para fazer as histórias seguirem o arco de tensão desejado reduziu consideravelmente a discrepância total observada nas histórias. Os usuários que testaram o sistema com esta funcionalidade desligada demonstraram, em certos pontos da história, estar entediados ou irritados por não conseguirem avançar em direção à solução do enigma. Nos usuários que utilizaram o sistema completo, estes sentimentos negativos foram bastante reduzidos.

O método utilizado para antever e evitar situações sem saída demonstrou funcionar bem na prática. Nenhum dos usuários sequer mencionou a execução das ações de pretexto e intervenção na história. Julga-se que isso seja um indício de que o método é capaz de atuar de forma razoavelmente sutil, sem causar frustração ao usuário.

Os quatro primeiros usuários que testaram o Fabulator foram utilizados como o grupo de usuários de teste d'*A Estória de Ugh 2*. Eles responderam ao questionário relativo à importância que cada uma das pistas teve para a história. Com base nesta avaliação dos usuários, os parâmetros c_p do modelo foram refinados para os experimentos com os demais usuários.

O estudo realizado com algoritmos de planejamento sob a ótica de IS (Seção 3.3) também é considerada uma contribuição importante deste trabalho. Esta é, até onde se sabe, a primeira análise deste tipo realizada, tendo inclusive sido aceita para publicação em periódico (BARROS; MUSSE, 2007).

Por fim, confrontando os resultados deste trabalho com o Estado-da-Arte em IS, pode-se dizer que o modelo apresentado obteve avanços no que diz respeito à busca por consistência narrativa em sistemas interativos.

[1] O Erasmatron, de Crawford, esteve comercialmente disponível por um certo período de tempo alguns anos atrás. O Façade, de Mateas e Stern, está disponível para o público em geral como um *download* gratuito.

7.1 Trabalhos Futuros

O modelo apresentado neste trabalho utiliza algoritmos de planejamento de forma bastante determinante, visto que *todas* as ações executadas por NPCs são determinadas por este algoritmo. Esta abordagem tem se demonstrado capaz de adaptar a história às ações dos usuários, fazendo com que a narrativa flua em direção ao seu desfecho. Porém, a abordagem tem uma desvantagem particularmente indesejada: o comportamento apresentado pelos personagens é bastante simples. Este problema foi destacado, de várias maneiras diferentes, pelos usuários que testaram o Fabulator e *A Estória de Ugh 2*. Alguns trabalhos na área (por exemplo, (MATEAS; STERN, 2003; RIEDL; YOUNG, 2003)) utilizam arquiteturas com dois níveis: o nível mais alto, que atua como uma espécie de "diretor" da história, determina objetivos que devem ser cumpridos pelos personagens para que a narrativa progrida adequadamente. O segundo nível modela os personagens, que buscam atingir os objetivos determinados pelo "diretor" demonstrando, caso seja necessário, comportamentos mais complexos.

Explorar uma arquitetura deste tipo parece uma possibilidade promissora. Neste caso, um modelo muito similar ao utilizado neste trabalho poderia ser utilizado como "diretor", enquanto novos módulos se encarregariam de prover comportamentos mais interessantes aos personagens. Nesta mesma linha, não deve ser ignorada a possibilidade de que o próprio modelo de personagens utilize algoritmos de planejamento (neste caso, haveria planejamento nos dois níveis do modelo).

Além de solucionar alguns dos problemas relatados por usuários, esta arquitetura teria um efeito colateral interessante: o algoritmo de planejamento deixaria de lidar com detalhes de baixo nível da história, de modo a reduzir o espaço de busca com o qual o algoritmo trabalha. Isso permitiria tratar histórias mais complexas e melhoraria a escalabilidade do modelo.

O modelo de arco de tensão utilizado (Seção 4.2.1), também pode ser melhorado no futuro. O modelo atual, em que conhecimento e tensão se equivalem, não é capaz de tratar reduções no nível de conhecimento do usuário. Isto impede o tratamento de situações em que um NPC mente ao protagonista. Desta forma, alterações no modelo de arco de tensão podem ser necessárias para permitir que futuramente sejam implementados comportamentos mais complexos para os NPCs (como dotá-los da capacidade de mentir).

Um segundo aspecto levantado pelos usuários do Fabulator foi a dificuldade de compreensão de alguns eventos que ocorreram na história. A maior parte destes casos estava relacionada a ações executadas pelos NPCs que não eram devidamente percebidas ou compreendidas pelos usuários. De fato, no modelo atual, os NPCs simplesmente executam as ações definidas pelo algoritmo de planejamento, sem verificar se estas ações serão corretamente percebidas pelo usuário. Solucionar este problema, evidentemente, é desejável.

Ainda a respeito deste mesmo ponto, é possível relacionar o problema com um aspecto levantado na Seção 4.5, que contextualiza o modelo na teoria narrativa de Bal: no modelo atual, a fábula é utilizada diretamente como história,[2] sem sofrer alterações. Segundo a teoria narrativa em questão, na transformação da fábula para história, um dos aspectos a ser definido é o ponto de vista em que a narrativa será contada. No caso, deveria se fazer algo para garantir que as ações fossem mostradas do ponto de vista do usuário, ou seja, de alguma forma que facilitasse sua compreensão.

[2] Aqui, "fábula" e "história" são usados com o sentido proposto por Bal (Seção 3.1.1).

Outra alteração que pode ser considerada futuramente diz respeito à representação do cenário em que a história ocorre. O Fabulator utiliza um cenário contínuo, sobre o qual os personagem se locomovem livremente. Internamente, porém, o modelo está baseado em um cenário discreto, em que cada personagem, num instante qualquer, deve estar em um lugar de uma lista de locais possíveis. Este conflito entre o uso de um cenário contínuo sobre um modelo que internamente trabalha com locais discretos é fonte de uma série de problemas práticos.

Por exemplo, quando um personagem executa uma ação de ir de um ponto a outro do cenário, ele dramatiza esta ação executando uma animação de caminhada pelo ambiente. Em algum instante durante a execução desta animação, deverá se alterar o estado do mundo para refletir a mudança na localização do personagem. Este instante poderia coincidir com o começo ou final da animação, mas isso impediria, por exemplo, que o usuário conversasse com um NPC que estivesse cruzando seu caminho (pois o sistema consideraria que os personagens não se encontram no mesmo local naquele instante). Uma alternativa seria alterar o estado do mundo várias vezes durante a execução da caminhada, refletindo que o personagem esteve por vários lugares até atingir seu destino. Isto, porém, também possui implicações negativas: alterações freqüentes no estado do mundo podem requerer um número muito grande de replanejamentos em espaços curtos de tempo, prejudicando o desempenho e podendo causar instabilidades no sistema.

A adoção de um cenário totalmente discreto evitaria este tipo de problema, pois eliminaria qualquer dúvida com relação à localização de um personagem enquanto ele se desloca de um ponto a outro do cenário. Poderia evitar também alguns inconvenientes observados por usuários, como locais do cenário considerados "muito escondidos" e a dificuldade de compreender que um determinado ponto do ambiente é considerado um local distinto de um ponto próximo. Adicionalmente, o uso de cenários discretos é defendido por Crawford (2004) como uma forma de dar maior ênfase a aspectos narrativos, pois histórias são amplamente independentes de relações espaciais complexas.

Expressões faciais e sons possuem grande importância na comunicação humana, mas não foram exploradas neste trabalho. Futuramente, seria interessante adotar esses recursos e averiguar de que forma eles podem ser utilizados para transmitir a história mais eficientemente aos usuários do sistema.

Futuramente também seria importante trabalhar em ferramentas que auxiliem no processo de autoria de conteúdo para IS. Uma ferramenta para edição e visualização gráfica da estrutura das histórias seria particularmente útil. Caso tal ferramenta fosse baseada em algum formalismo como Redes de Petri (MURATA, 1989), poderia ser utilizada para identificar situações sem saída potenciais.

Finalmente, há diversas melhorias que podem ser implementadas no protótipo. Algumas possibilidades que, de acordo com os experimentos com usuários, seriam importantes são:

- Utilizar textos coloridos para representar as falas dos personagens: se cada personagem for identificado por uma cor, será mais simples identificar quem está falando.
- Representar graficamente eventos como a chuva, ao invés de exibir um texto descrevendo o evento.
- Melhorar o controle de câmera.

- Incluir uma forma simples do usuário verificar quais pistas já foram contadas ao sacerdote.
- Melhorar a localização de personagens e objetos no cenário, evitando que um apareça "dentro do outro" (detecção de colisão mais precisa).

É interessante observar que estes pontos não estão diretamente relacionados com o modelo apresentado, nem com a implementação do modelo propriamente dito. Assim, à primeira vista, estes aspectos poderiam parecer de pouca relevância para a evolução futura do trabalho. A importância destes pontos está no fato deles estarem relacionados com a experiência que as pessoas têm ao utilizar o protótipo. Detalhes de implementação passam a ter um papel muito importante a partir do momento em que se realizam testes com usuários. Usuários tendem a concentrar suas críticas nestes detalhes, de modo que minimizar estes problemas pode ser visto como uma forma de forçar os usuários a prestar atenção e criticar os aspectos narrativos de sua experiência. E isto permite uma melhor avaliação dos resultados obtidos pelo modelo e identificação de aspectos "genuinamente relacionados a *Interactive Storytelling*" que podem ser melhorados.

REFERÊNCIAS

ARISTÓTELES. *Poética*. Porto Alegre, Brasil: Editora Globo, 1966. 264 p.

AYLETT, R. Narrative in virtual environments – towards emergent narrative. In: *Working Notes of the AAAI Fall Symposium on Narrative Intelligence*. [S.l.]: The AAAI Press, Menlo Park, Estados Unidos, 1999. p. 83–86.

BACCHUS, F.; KABANZA, F. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, Nova York, Estados Unidos, v. 116, n. 1–2, p. 123–191, 2000.

BAL, M. *Teoría de la Narrativa: una introducción a la narratología*. Quinta ed. Madri, Espanha: Cátreda, 1998. 164 p. ISBN 84-376-0504-0.

BARROS, L. M. *Incluindo Elementos da Narrativa em Interactive Storytelling*. São Leopoldo, Brasil, 2004. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação, Universidade do Vale do Rio dos Sinos).

BARROS, L. M.; MUSSE, S. R. Introducing narrative principles into planning-based interactive storytelling. In: ZHIYING, S. Z.; PING, L. S. (Ed.). *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005)*. Valência, Espanha: ACM Press, 2005. p. 35–42. ISBN 1-59593-110-4.

BARROS, L. M.; MUSSE, S. R. Planning algorithms for interactive storytelling. *ACM Computers in Entertainment*, 2007. (Aceito para publicação).

BLUM, A.; FURST, M. Fast planning through planning graph analysis. *Artificial Intelligence*, Nova York, Estados Unidos, v. 90, p. 281–300, 1997.

BONET, B.; GEFFNER, H. Planning as heuristic search. *Artificial Intelligence Special Issue on Heuristic Search*, Nova York, Estados Unidos, v. 1, n. 129, p. 5–33, 2001.

CAVAZZA, M.; CHARLES, F.; MEAD, S. J. Characters in search of an author: AI-based virtual storytelling. In: *Proceedings of ICVS 2001: International Conference on Virtual Storytelling*. Avignon, França: Springer, 2001. (Lecture Notes in Computer Science, v. 2197), p. 145–154.

CAVAZZA, M.; CHARLES, F.; MEAD, S. J. Character-based interactive storytelling. *IEEE Intelligent Systems, special issue on AI in Interactive Entertainment*, Los Alamitos, Estados Unidos, v. 17, n. 4, p. 17–24, Julho/Agosto 2002.

CAVAZZA, M.; CHARLES, F.; MEAD, S. J. Emergent situations in interactive storytelling. In: *ACM Symposium on Applied Computing*. Madri, Espanha: ACM Press, Nova York, Estados Unidos, 2002. p. 1080–1085.

CHARLES, F.; CAVAZZA, M. Exploring the scalability of character-based storytelling. In: ACM. *Proceedings of the ACM Joint Conference on Autonomous Agents and Multi-Agent Systems*. Nova York, Estados Unidos: ACM Press, New York, Estados Unidos, 2004. p. 872–879.

CHARLES, F. et al. Planning formalisms and authoring in interactive storytelling. In: GÖBEL, S. et al. (Ed.). *Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment*. Darmstadt, Alemanha: Fraunhofer IRB Verlag, 2003.

CIARLINI, A. E. M. et al. A logic-based tool for interactive generation and dramatization of stories. In: ZHIYING, S. Z.; PING, L. S. (Ed.). *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005)*. Valência, Espanha: ACM Press, 2005. p. 133–140.

COLES, A. I.; SMITH, A. J. *Marvin: Macro-Actions from Reduced Versions of the Instance*. Junho 2004. Fourth International Planning Competition Booklet (IPS'04), ICAPS 2004.

CRAWFORD, C. Assumptions underlying the Erasmatron interactive storytelling engine. In: *Working Notes of the AAAI Fall Symposium on Narrative Intelligence*. North Falmouth, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 1999.

CRAWFORD, C. *The Art of Interactive Design: A Euphonious and Illuminating Guide to Building Successful Software*. San Francisco, Estados Unidos: No Starch Press, 2003. 385 p. ISBN 1-886411-84-0.

CRAWFORD, C. *Chris Crawford on Interactive Storytelling*. Indianápolis, Estados Unidos: New Riders Games, 2004. 384 p. ISBN 0321278909.

DONIKIAN, S.; PORTUGAL, J.-N. Writing interactive fiction scenarii with DraMachina. In: GÖBEL, S. et al. (Ed.). *Technologies for Interactive Digital Storytelling and Entertainment, Second International Conference, TIDSE 2004, Darmstadt, Alemanha, Junho 24–26, 2004, Proceedings*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3105), p. 101–112. ISBN 3-540-22283-9.

EDDIE, C. *CEGUI: Crazy Eddie's GUI System*. 2006. Disponível em `http://www.cegui.org.uk`. Acesso em 29 de dezembro de 2006.

EDELKAMP, S.; HOFFMANN, J. *PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition*. Freiburg, Alemanha, Janeiro 2004.

FIELD, S. *Manual do Roteiro: Os Fundamentos do Texto Cinematográfico*. Rio de Janeiro, Brasil: Editora Objetiva, 1995. 223 p.

FOX, M.; LONG, D. Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning. In: NEBEL, B. (Ed.). *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 01)*. Seattle, Estados Unidos: Morgan Kaufmann, 2001. p. 445–452.

GEREVINI, A. et al. *Planning in PDDL2.2 Domains with LPG-TD*. Junho 2004. Fourth International Planning Competition Booklet (IPS'04), ICAPS 2004.

GEREVINI, A.; SERINA, I. LPG: a planner based on local search for planning graphs. In: *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02)*. Toulouse, França: AAAI Press, 2002.

GLASSNER, A. *Interactive Storytelling: Techniques for 21st Century Fiction*. [S.l.]: AK Peters, 2004. 511 p. ISBN 1568812213.

GRASBON, D.; BRAUN, N. A morphological approach to interactive storytelling. In: FLEISCHMANN, M.; STRAUSS, W. (Ed.). *cast01 // living in mixed realities*. Bonn, Alemanha: netzspannung.org, 2001. p. 337–340.

GREEFF, M.; LALIOTI, V. Interactive storytelling with virtual identities. In: *IPT/EGVE 2001: 5th International Projection Technology Workshop and 7th Eurographics Workshop on Virtual Environments*. Stuttgart, Alemanha: ACM Press, Nova York, Estados Unidos, 2001.

HASLUM, P.; GEFFNER, H. Admissible heuristics for optimal planning. In: *Proceedings of the Fifth International Conference on Artificial Intelligence Planning & Scheduling (AIPS 2000)*. Breckenridge, Estados Unidos: [s.n.], 2000. p. 140–149.

HEIDELBERGER, B. *CAL3D: Character Animation Library*. 2006. Disponível em `http://home.gna.org/cal3d/`. Acesso em 28 de dezembro de 2006.

HOFFMANN, J. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research, special issue on the 3rd International Planning Competition*, v. 20, p. 291–341, 2003.

HOFFMANN, J.; NEBEL, B. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, v. 14, p. 253–302, 2001.

HUTH, M. R. A.; RYAN, M. D. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge, Reino Unido: Cambridge University Press, 2000. 387 p.

IERUSALIMSCHY, R. *Programming in Lua*. Rio de Janeiro, Brasil: Lua.org, 2003. 288 p.

KAUTZ, H. A.; SELMAN, B. Planning as satisfiability. In: NEUMANN, B. (Ed.). *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*. Viena, Áustria: Wiley, 1992. p. 359–363.

KOEHLER, J. et al. Extending planning graphs to an ADL subset. In: STEEL, S.; ALAMI, R. (Ed.). *Recent Advances in AI Planning. Fourth European Conference on Planning (ECP'97)*. Toulouse, França: Springer, 1997. (Lecture Notes in Artificial Intelligence, v. 1348), p. 273–285. ISBN 3-540-63912-8.

MACHADO, I.; PAIVA, A.; BRNA, P. Real characters in virtual stories: Promoting interactive story-creation activities. In: *Proceedings of ICVS 2001: International Conference on Virtual Storytelling*. Avignon, França: Springer, 2001. (Lecture Notes in Computer Science, v. 2197), p. 127–134.

MAGERKO, B. A proposal for an interactive drama architecture. In: *AAAI Spring Symposium on AI and Interactive Entertainment*. Stanford, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 2002.

MAGERKO, B. Mediating the tension between plot and interaction. In: *Imagina 2005*. Monaco, Monte Carlo: [s.n.], 2005.

MAGERKO, B.; LAIRD, J. E. Building an interactive drama architecture. In: GÖBEL, S. et al. (Ed.). *Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment*. Darmstadt, Alemanha: Fraunhofer IRB Verlag, 2003. p. 226–237.

MATEAS, M. *An Oz-Centric Review of Interactive Drama and Believable Agents*. Pittsburg, Estados Unidos, 1997.

MATEAS, M.; STERN, A. Façade: An experiment in building a fully-realized interactive drama. In: *Game Developers Conference, Game Design track*. [S.l.: s.n.], 2003.

MATEAS, M.; STERN, A. Life-like characters: Tools, affective functions, and applications. In: ____. Berlin, Alemanha: Springer, 2004. cap. A Behavior Language: Joint Action and Behavioral Idioms.

MATEAS, M.; STERN, A. Natural language understanding in façade: Surface-text processing. In: GÖBEL, S. et al. (Ed.). *Technologies for Interactive Digital Storytelling and Entertainment, Second International Conference, TIDSE 2004, Darmstadt, Alemanha, Junho 24–26, 2004, Proceedings*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3105). ISBN 3-540-22283-9.

MATEAS, M.; STERN, A. Build it to understand it: Ludology meets narratology in game design space. In: *Digital Games Research Conference (DiGRA)*. Vancouver, Canadá: [s.n.], 2005.

MATEAS, M.; STERN, A. Structuring content in the Façade interactive drama architecture. In: AAAI. *Proceedings of the AAAI First Annual Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'05)*. Marina del Rey, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 2005.

MCDERMOTT, D. V. *Drew V. McDermott: International Planning Competition*. 2006. Disponível em `http://www.cs.yale.edu/homes/dvm`. Acesso em 13 de junho de 2006.

MEAD, S. J.; CAVAZZA, M.; CHARLES, F. Influential words: Natural language in interactive storytelling. In: *Proceedings of the 10th International Conference on Human-Computer Interaction*. Creta, Grécia: Lawrence Erlbaum Associates, Mahwah, Estados Unidos, 2003.

MIXON, L. J. *Shattertown Sky v1.0: A Post-Partum*. 1997. Disponível em `http://www.erasmatazz.com/library/SHATTERTOWNSKY.html`. Acesso em 15 de junho 2006.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, Abril 1989.

MURRAY, J. H. *Hamlet no Holodeck: O Futuro da Narrativa no Ciberespaço*. São Paulo, Brasil: Editora UNESP, 2003. 283 p.

OSFIELD, R. *Open Scene Graph*. 2006. Disponível em `http://www.openscenegraph.org`. Acesso em 28 de dezembro de 2006.

POZZER, C. T. *Um Sistema para Geração, Interação e Visualização 3D de Histórias para TV Interativa*. Tese (Doutorado) — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, Março 2005.

PROPP, V. *A Morfologia do Conto*. Segunda ed. Lisboa, Portugal: Editora Vega, 1983. 286 p.

RICH, E.; KNIGHT, K. *Artificial Intelligence*. Segunda ed. Nova York, Estados Unidos: McGraw-Hill, 1991.

RIEDL, M. O.; SARETTO, C. J.; YOUNG, R. M. Managing interaction between users and agents in a multi-agent storytelling environment. In: *Proceedings of AAMAS 2003: the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Melbourne, Austrália: ACM Press, Nova York, Estados Unidos, 2003.

RIEDL, M. O.; YOUNG, R. M. Character-focused narrative generation for execution in virtual worlds. In: *Virtual Storytelling (Proceedings of ICVS 2003: International Conference on Virtual Storytelling)*. Toulouse, França: Springer-Verlag Heidelberg, 2003. (Lecture Notes in Computer Science, v. 2897), p. 47–56.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Segunda ed. Englewood Cliffs, Estados Unidos: Prentice Hall, 2002. 1132 p. ISBN 0137903952.

STROUSTRUP, B. *The C++ Programming Language*. Special third. Reading, Estados Unidos: Addison-Wesley, 2000. 911 p.

SZILAS, N. Interactive drama on computer: Beyond linear narrative. In: *Working Notes of the AAAI Fall Symposium on Narrative Intelligence*. North Falmouth, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 1999. p. 150–156.

SZILAS, N. A new approach to interactive drama: From intelligent characters to an intelligent virtual narrator. In: *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*. Stanford, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 2001. p. 72–76.

SZILAS, N. IDtension: a narrative engine for interactive drama. In: GÖBEL, S. et al. (Ed.). *Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment*. Darmstadt, Alemanha: Fraunhofer IRB Verlag, 2003.

TOBIAS, R. B. *20 Master Plots: and how to build them*. Cincinnati, Estados Unidos: Writer's Digest Books, 1993. 240 p. ISBN 1-58297-239-7.

YOUNG, R. M. Notes on the use of plan structures in the creation of interactive plot. In: *Working Notes of the AAAI Fall Symposium on Narrative Intelligence*. North Falmouth, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 1999.

YOUNG, R. M. An overview of the Mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. In: *Spring Symposium on AI and Interactive Entertainment*. Stanford, Estados Unidos: The AAAI Press, Menlo Park, Estados Unidos, 2001.

ZAGALO, N.; BARKER, A.; BRANCO, V. Story reaction structures to emotion detection. In: ACM. *Proceedings of the SRMC'04, ACM Multimedia 2004 — Workshop on Story Representation, Mechanisam, Context*. Nova York, Estados Unidos, 2004. p. 33–38.

APÊNDICE A – SÍMBOLOS DA LÓGICA DE PREDICADOS

A Tabela 8 apresenta uma breve descrição dos símbolos da lógica de predicados utilizados neste trabalho. Uma descrição mais aprofundada da sintaxe e semântica desta lógica pode ser encontrada em (HUTH; RYAN, 2000).

Tabela 8: Símbolos da lógica de predicados usados neste trabalho.

Símbolo	**Significado**
$\vee$	Disjunção. Uma sentença na forma $p \vee q$ (“p ou q”), é verdadeira se pelo menos um dos símbolos envolvidos representa uma sentença com valor verdadeiro.
$\wedge$	Conjunção. Uma sentença na forma $p \wedge q$ (“p e q”), é verdadeira apenas se tanto p quanto q forem verdadeiros.
$\neg$	Negação. Uma sentença na forma $\neg p$ (“não p”), tem o valor verdade contrário ao de p.
$\forall$	Quantificação universal. Uma sentença na forma $\forall x\, P(x)$ (“para todo x, $P(x)$”) é verdadeira apenas se o predicado $P(\cdot)$ for verdadeiro para todos os objetos do mundo, quando estes objetos são colocados no lugar de “$\cdot$”.
$\exists$	Quantificação existencial. Uma sentença na forma $\exists x\, P(x)$ (“existe x tal que $P(x)$”) é verdadeira apenas caso exista pelo menos um objeto no mundo para o qual a expressão $P(\cdot)$ é verdadeira, quando este objeto é colocado no lugar de “$\cdot$”.

APÊNDICE B – "A ESTÓRIA DE UGH": PRIMEIRO ATO

Este apêndice apresenta a definição do primeiro ato de "A Estória de Ugh", que é utilizada na Seção 3.3.2 para avaliar o desempenho de diversos algoritmos de planejamento. A definição é dividida em dois arquivos. O primeiro, `ugh.pddl`, contém a descrição do domínio, ou seja, dos objetos existentes no mundo e as ações disponíveis ao algoritmo de planejamento. O segundo arquivo, `ugh_act1.pddl`, contém a descrição do problema propriamente dito, que compreende o estado inicial do mundo e o estado objetivo.

Arquivo `ugh.pddl`

```
(define (domain ughs-story)
  (:requirements :strips :typing :equality :conditional-effects :existential-preconditions)

  (:types place information movable-stuff - object
          thing character - movable-stuff)

  (:constants ugh tugh zonk - character
              ughs-cave tughs-cave zonks-cave altar volcano beach garden - place
              statue zonks-card tughs-ticket club strawberry fish - thing
              statue-was-stolen tugh-sold-his-mother zonks-cave-has-wet-paint
                 zonk-is-iconoclast club-has-paint-marks
                 tugh-painted-zonks-cave - information)

  (:predicates (at ?what - movable-stuff ?where - place)
               (has ?who - character ?what - thing)
               (knows ?who - character ?what - information)
               (likes ?who - character ?what - thing)
               (friendly-to ?the-who ?the-friend - character)
               (wont-give ?who - character ?what - thing)
               (everybody-knows ?what - information)
               (is-the-thief ?who - character))

  (:action GoTo
           :parameters (?who - character ?from ?to - place)

           :precondition (and (at ?who ?from)
                              (not (= ?from ?to)))

           :effect (and (at ?who ?to)
                        (not (at ?who ?from))))

  (:action Take
           :parameters (?who - character ?what - thing ?where - place)

           :precondition (and (at ?who ?where)
                              (at ?what ?where))
```

```
        :effect (and (not (at ?what ?where))
                     (has ?who ?what)))

(:action TalkAbout
        :parameters (?speaker ?listener - character ?topic - information)

        :precondition (and (friendly-to ?listener ?speaker)
                           (not (= ?speaker ?listener))
                           (knows ?speaker ?topic)
                           (exists (?p - place)
                                   (and (at ?speaker ?p)
                                        (at ?listener ?p))))

        :effect (and (knows ?listener ?topic)
                     (when (and (= ?listener zonk)
                                (= ?topic zonks-cave-has-wet-paint))
                           (knows ?speaker tugh-painted-zonks-cave))
                     (when (and (= ?listener tugh)
                                (= ?topic zonks-cave-has-wet-paint)
                                (not (knows tugh club-has-paint-marks)))
                           (knows ?speaker tugh-painted-zonks-cave))))

(:action ImitateDinosaur
        :parameters (?imitator ?spectator - character ?where - place)

        :precondition (and (at ?imitator ?where)
                           (at ?spectator ?where)
                           (not (= ?imitator ?spectator)))

        :effect (and (not (friendly-to ?spectator ?imitator))
                     (when (has ?spectator zonks-card)
                           (and (not (has ?spectator zonks-card))
                                (at zonks-card ?where)))))

(:action ExaminePlace
        :parameters (?examiner - character ?where - place)

        :precondition (and (at ?examiner ?where)
                           (not (exists (?c - character)
                                        (and (not (= ?c ?examiner))
                                             (at ?c ?where))))))

        :effect (and (when (= ?where tughs-cave)
                           (has ?examiner tughs-ticket))
                     (when (= ?where zonks-cave)
                           (knows ?examiner zonks-cave-has-wet-paint))))

(:action ExamineThing
        :parameters (?examiner - character ?what - thing)

        :precondition (has ?examiner ?what)

        :effect (and (when (= ?what tughs-ticket)
                           (knows ?examiner tugh-sold-his-mother))
                     (when (= ?what zonks-card)
                           (knows ?examiner zonk-is-iconoclast))
                     (when (= ?what club)
                           (knows ?examiner club-has-paint-marks))))

(:action GivePresent
        :parameters (?giver ?receiver - character ?present - thing)

        :precondition (and (not (wont-give ?giver ?present))
                           (not (= ?giver ?receiver))
                           (has ?giver ?present)
                           (exists (?p - place)
                                   (and (at ?giver ?p)
                                        (at ?receiver ?p))))

        :effect (and (not (has ?giver ?present))
                     (has ?receiver ?present)
                     (when (likes ?receiver ?present)
                           (friendly-to ?receiver ?giver))))
```

```
  (:action AssumeTheft
           :parameters (?thief - character ?location - place ?stolen-thing - thing)

           :precondition (and (at ?thief ?location)
                              (is-the-thief ?thief)
                              (has ?thief ?stolen-thing))

           :effect (at ?stolen-thing ?location)))
```

Arquivo `ugh_act1.pddl`

```
(define (problem ughs-story-act1)
  (:domain ughs-story)
  (:init
   (at ugh ughs-cave)
   (at tugh tughs-cave)
   (at zonk zonks-cave)
   (at club altar)
   (at strawberry garden)
   (at fish beach)
   (has tugh statue)
   (has zonk zonks-card)
   (knows ugh statue-was-stolen)
   (likes zonk strawberry)
   (friendly-to zonk ugh)
   (friendly-to tugh ugh)
   (wont-give tugh statue)
   (wont-give tugh tughs-ticket)
   (wont-give zonk zonks-card)
   (is-the-thief tugh))

  (:goal (and (knows ugh tugh-sold-his-mother)
              (knows ugh zonk-is-iconoclast)))
```

APÊNDICE C – LINGUAGEM PARA DESCRIÇÃO DE PROBLEMAS DE PLANEJAMENTO

Este apêndice apresenta uma descrição completa da sintaxe da linguagem utilizada neste trabalho para descrever problemas de planejamento. Esta linguagem possui os mesmos recursos de PDDL (EDELKAMP; HOFFMANN, 2004), mas utiliza uma sintaxe diferente. A gramática da linguagem e exemplos são apresentados à medida que os recursos da linguagem são descritos.

A descrição de um problema de planejamento é composta de uma seqüência de seções numa ordem específica:

⟨*Problema de planejamento*⟩ ::=⟨*Seção TYPES*⟩
⟨*Seção OBJECTS*⟩
⟨*Seção PREDICATES*⟩
⟨*Seção WORLD_STATE*⟩
⟨*Seção ACTIONS*⟩
⟨*Seção GOAL*⟩+

É importante notar que mais de uma seção GOAL é aceita. Objetivos são utilizados em seqüência, de modo a permitir aos autores que criem histórias seguindo a estrutura clássica em três atos (Seção 4.1).

Assim como em PDDL (com o requisito `:typing`), esta linguagem tem suporte a tipos e subtipos para facilitar a definição de ações. Na verdade, todos objetos e variáveis que aparecem na descrição do problema devem ter um tipo, e todos os tipos utilizados devem ser previamente declarados numa seção TYPES. Esta seção é iniciada pela palavra reservada `TYPES` e seguida de uma lista de um ou mais pares de identificadores separados por dois pontos:

⟨*Seção TYPES*⟩ ::=‘`TYPES`’ (⟨*Identificador*⟩ ‘`:`’ ⟨*Identificador*⟩)+

⟨*Identificador*⟩ ::=(‘`a`’...‘`z`’ | ‘`A`’...‘`Z`’) (‘`a`’...‘`z`’ | ‘`A`’...‘`Z`’ | ‘`0`’...‘`9`’ | ‘_’)*

Um par de identificadores no formato “`A: B`” define um novo tipo “`A`” como sendo um subtipo do tipo “`B`”. Um tipo “`object`” é intrínseco à linguagem, e representa a raiz da hierarquia de tipos. Uma declaração válida de tipos é mostrada logo a seguir.

```
TYPES
   Personagem: object
   Heroi: Personagem
   Lugar: object
   Coisa: object
```

Um objeto é uma instância de uma variável de algum tipo definido na seção TYPES. Todos os objetos referenciados em algum ponto do problema devem ser declarados na seção OBJECTS:

⟨*Seção OBJECTS*⟩ ::=‘OBJECTS’
(⟨*Identificador*⟩ ⟨*Identificador*⟩ [⟨*Dados adicionais*⟩])+

⟨*Dados adicionais*⟩ ::=‘DATA’ ‘(’ (⟨*Identificador*⟩ ‘=’ ⟨*string*⟩)+ ‘)’

⟨*string*⟩ ::=‘"’ ⟨*Caracter diferente de “aspas”*⟩+ ‘"’

Cada objeto é composto de um par de identificadores (seu tipo e nome, respectivamente). Opcionalmente, um objeto pode ter alguns dados adicionais associados a ele. Estes dados adicionais são utilizados para fazer a ligação entre o módulo responsável pelo planejamento e o restante do sistema, conforme discutido na Seção 5.1.2. Cada um dos dados adicionais é composto de um identificador, um sinal de igual (“=”) e uma *string*.

Uma declaração de alguns objetos é exemplificada abaixo:

```
OBJECTS
   Personagem Joao
   Personagem Gigante
   Lugar Castelo          DATA (position = "-9.6, 23.3, -1.0")
   Lugar Bosque           DATA (position = "3.0, -36.3, -0.6")
   Coisa SementeDeFeijao
```

Assim como tipos e objetos, todos os predicados que podem ser utilizados na descrição do problema devem ser declarados antes de sua utilização.

⟨*Seção PREDICATES*⟩ ::=‘PREDICATES’ ⟨*Predicado*⟩+

⟨*Predicado*⟩ ::=⟨*Identificador*⟩ ‘(’ [⟨*Identificador*⟩ (‘,’ ⟨*Identificador*⟩)*] ‘)’

O primeiro identificador é o nome do predicado. Os identificadores na lista entre parênteses indicam o tipo de cada um dos parâmetros do predicado sendo declarado. Não é possível sobrecarregar predicados (ou seja, declarar dois predicados com o mesmo nome, mas com listas de parâmetros distintas). É possível, contudo, obter efeito semelhante através do uso de hierarquias de tipos. Assim, o seguinte trecho representa uma declaração válida de predicados:

```
PREDICATES
   Em (Personagem, Lugar)
   Gosta (Personagem, Personagem)
   Tem (Personagem, Coisa)
   EhMau (Personagem)
   Choveu()
```

O estado do mundo contém todas as informações da história que podem ser relevantes ao algoritmo de planejamento. É importante deixar claro que isto inclui, caso seja relevante, aspectos que intuitivamente poderiam não ser consideradas parte do mundo, como o estado mental e a personalidade dos personagens. O estado do mundo num determinado instante é composto de um conjunto de predicados que são verdadeiros neste mesmo instante.

O estado inicial do mundo é descrito na seção WORLD_STATE:

⟨*Seção WORLD_STATE*⟩ ::='WORLD_STATE' (['NOT'] ⟨*Predicado*⟩)+

Na definição do estado inicial do mundo, todos os parâmetros dos predicados devem ser objetos definidos na seção OBJECTS:

```
WORLD_STATE
   Em (Joao, Bosque)
   Em (Gigante, Castelo)
   Tem (Joao, SementeDeFeijao)
   EhMau (Gigante)
   NOT Choveu()
```

As ações que o algoritmo de planejamento tem à disposição para alterar o estado do mundo são declaradas na seção ACTIONS. Conforme explicado na Seção 3.2, cada ação possui um nome, uma lista de parâmetros, um pré-requisito e um efeito. Além destes, como no caso dos objetos, é possível associar dados adicionais a cada ação, de modo a realizar a ligação entre o módulo de seleção de ações e o restante do sistema. O uso destes dados adicionais é descrito na Seção 5.1.2.

O pré-requisito de uma ação pode ser um simples predicado ou uma expressão complexa, envolvendo quantificadores, conjunções e disjunções de predicados. Para que a ação possa ser utilizada em um certo estado de mundo, é preciso que esta expressão seja verdadeira neste estado.

O efeito pode ser predicado simples, uma conjunção de predicados ou um "predicado condicional". A semântica dos efeitos das ações é a seguinte: um predicado p, se usado como efeito, faz com que p seja adicionado ao estado do mundo e $\neg p$ seja removido. Uma conjunção de predicados aplica esta mesma regra para cada um dos seus componentes. E um predicado condicional aplica esta regra caso a sua condição seja verdadeira.

⟨*Seção ACTIONS*⟩ ::=‘ACTIONS’ ⟨*Ação*⟩+

⟨*Ação*⟩ ::=⟨*Identificador*⟩ ‘(’ [⟨*Lista de vari'aveis*⟩] ‘)’
‘PREREQUISITE’ ⟨*Expressão*⟩
‘EFFECT’ [‘NOT’] (⟨*Predicado*⟩ | ⟨*Conjunção*⟩ | ⟨*Condicional*⟩)
⟨*dados adicionais*⟩

⟨*Lista de variáveis*⟩ ::=⟨*Identificador*⟩ ⟨*Identificador*⟩
[‘,’ ⟨*Identificador*⟩ ⟨*Identificador*⟩]

⟨*Expressão*⟩ ::=⟨*Predicado*⟩
| ⟨*Conjunção*⟩
| ⟨*Disjunção*⟩
| ⟨*Quantificador universal*⟩
| ⟨*Quantificador existencial*⟩
| ⟨*Igualdade*⟩
| ⟨*Condicional*⟩
| ‘NOT’ ⟨*Expressão*⟩

Na definição de ⟨*Lista de variáveis*⟩, em cada um dos pares de identificadores, o primeiro elemento é o tipo da variável, e o segundo é seu nome.

Conjunções e disjunções são utilizadas, respectivamente, para unir predicados através de conectivos E e OU, com a semântica usual:

⟨*Conjunção*⟩ ::=‘AND’ ‘(’ ⟨*Expressão*⟩+ ‘)’

⟨*Disjunção*⟩ ::=‘OR’ ‘(’ ⟨*Expressão*⟩+ ‘)’

Os quantificadores existencial e universal, como o nome sugere, correspondem aos quantificadores $\exists$ e $\forall$ da lógica de predicados (HUTH; RYAN, 2000), possuindo, inclusive, a mesma semântica:

⟨*Quantificador existencial*⟩ ::=‘EXISTS’ ⟨*Lista de vari'aveis*⟩ ‘(’ ⟨*Expressão*⟩ ‘)’

⟨*Quantificador universal*⟩ ::=‘FORALL’ ⟨*Lista de vari'aveis*⟩ ‘(’ ⟨*Expressão*⟩ ‘)’

Um predicado “EQUAL”, utilizado para testar se duas variáveis, ou uma variável e uma constante referenciam o mesmo objeto faz parte da linguagem.

⟨*Igualdade*⟩ ::=‘EQUAL’ ‘(’ ⟨*Identificador*⟩ ‘,’ ⟨*Identificador*⟩ ‘)’

Para testar por desigualdade, basta utilizar a negação deste predicado: `NOT EQUAL (x, y)`.

Uma expressão condicional é composta por duas outras expressões: uma condição e um efeito. Esta construção corresponde aos efeitos condicionais do requisito `:conditional-effects` da PDDL.

⟨*Condicional*⟩ ::=‘IF’ ⟨*Expressão*⟩ ‘THEN’ ⟨*Expressão*⟩

Uma seção ACTIONS com duas ações é mostrada a seguir.

```
ACTIONS
   PlantaPeDeFeijao (Personagem Quem, Lugar Onde)
      PREREQUISITE AND
                   (
                      Em (Quem, Onde)
                      Tem (Quem, SementeDeFeijao)
                   )
      EFFECT AND
             (
                NOT Tem (Quem, SementeDeFeijao)
                ExistePeDeFeijao (Onde)
             )
      DATA
      (
         subject = "Quem"
         animation = "Plant"
      )

   Furta (Personagem Ladrao, Personagem Furtado, Coisa Objeto)
      PREREQUISITE AND
                   (
                      Dorme (Furtado)
                      Tem (Furtado, Objeto)
                      EXISTS Lugar L
                      (
                         AND
                         (
                            Em (Ladrao, L)
                            Em (Furtado, L)
                         )
                      )
                   )
      EFFECT AND
             (
                Tem (Ladrao, Objeto)
                NOT Tem (Furtado, Objeto)
             )
      DATA
      (
         subject = "Ladrao"
         animation = "Steal"
         priority = 20
      )
```

Finalmente, uma seção GOAL é utilizada para especificar um objetivo para o algoritmo de planejamento:

⟨*Seção GOAL*⟩ ::=‘GOAL’ ⟨*Expressão*⟩

De acordo com esta especificação, uma seção GOAL poderia ser definida da seguinte forma:

```
GOAL
   Tem (Joao, OvosDeOuro)
   Em (Gigante, Castelo)
   Em (Joao, Casa)
   NOT EXISTS Lugar L
   (
      ExistePeDeFeijao (L)
   )
```

APÊNDICE D – "A ESTÓRIA DE UGH 2"

Este apêndice apresenta a definição completa de "A Estória de Ugh 2", utilizada nos experimentos descritos no Capítulo 6. Aqui, a história é apresentada como foi utilizada de entrada para o Fabulator. Uma discussão sobre esta história em um nível mais alto é apresentada na Seção 6.1.

Arquivo `Storyworld.lua`

```
Metadata = {
   Title = "A Estoria de Ugh: Edicao Gold Diamond Pro 2",
   Description = [[
Neandertown. 165012 a.C.
Um sacerdote inconsciente.
Uma estatua roubada.
Assuma o papel de Ugh, um troglodita detetive, e tente descobrir o culpado.]],
}

SceneryModelFile = "Scenery/Neandertown.3ds"

DesiredTensionArc = {
   [0] = 0,
   [90] = 10,
   [180] = 20,
   [270] = 30,
   [360] = 40,
   [450] = 50,
   [540] = 60,
   [630] = 70,
   [720] = 80,
}
```

Arquivo `Problem`

```
#
# The second incarnation of "Ugh's Story"
# A test storyworld by Leandro Motta Barros
#

TYPES
    MoveableStuff: object
    Character: MoveableStuff
    Place: object
    Clue: object
    Thing: MoveableStuff

OBJECTS
   Character Ugh        # The protagonist
   Character Alelugh    # The priest
   Character Falidogh   # The bankrupt
   Character Egonk      # The egotist

   Place FalidoghsCave DATA ( position = "-9.6, 23.3, -1.0" readable_form = "caverna do Falidogh" )
   Place EgonksCave DATA ( position = "2.8, 23.4, -0.3" readable_form = "caverna do Egonk" )
   Place AlelughsCave DATA ( position = "-32.3, -0.1, -1.4" readable_form = "caverna do Alelugh" )
   Place Patio DATA ( position = "-0.8, 0.1, 0.0" readable_form = "patio" )
   Place Beach DATA ( position = "3.0, -36.3, -0.6" readable_form = "praia" )
   Place Pier DATA ( position = "19.5, -38.8, -0.6" readable_form = "pier" )
   Place Forest DATA ( position = "25.1, -16.2, 0.0" readable_form = "floresta" )
   Place Garden DATA ( position = "-3.0, -25.8, -0.1" readable_form = "horta" )
```

```
   Thing DebtCharge DATA ( model = "DebtCharge.3ds" readable_form = "cobranca de divida" )
   Thing LoanRequest DATA ( model = "LoanRequest.3ds" readable_form = "pedido de emprestimo" )
   Thing BuyingCertificate DATA ( model = "BuyingCertificate.3ds" readable_form = "comprovante de compra" )
   Thing Strawberry DATA ( model = "Strawberry.3ds" readable_form = "morango" )
   Thing Fish DATA ( model = "Fish.3ds" readable_form = "peixe" )
   Thing Carcass DATA ( model = "Carcass.3ds" readable_form = "esqueleto" )
   Thing FishingPole DATA ( model = "FishingPole.3ds" readable_form = "vara de pescar" )
   Thing ManyFishingPoles DATA ( model = "ManyFishingPoles.3ds" readable_form = "muitas varas de pescar" )
   Thing FireStarter DATA ( model = "FireStarter.3ds" readable_form = "acendedor de fogo" )
   Thing PersonalStatue DATA ( model = "PersonalStatue.3ds" readable_form = "estatueta particular" )


   # Clues for the first act (minimal set of clues for finishing first act, that is)
   Clue FalidoghHadLoanRejected DATA ( readable_form = "Falidogh teve um pedido de emprestimo negado" )
   Clue FalidoghIsIndebted DATA ( readable_form = "Falidogh esta seriamente endividado" )
   Clue FalidoghAdmittedHavingDebts DATA ( readable_form = "Falidogh admitiu estar endividado" )
   Clue StatueWorthsALot DATA ( readable_form = "A estatua vale muito dinheiro" )
   Clue EgonkHasAPersonalStatue DATA ( readable_form = "Egonk tem uma estatua particular" )
   Clue SuspectFireStarter DATA ( readable_form = "Achei um acendedor de fogo suspeito" )
   Clue EgonkStolenTheFireStarterHeEnvied DATA ( readable_form = "Egonk roubou o acendedor de fogo que ele invejava" )
   Clue EgonkSaidHeDoesntBorrowHisStuff DATA ( readable_form = "Egonk disse que nao empresta suas coisas" )
   Clue FalidoghSaidEgonkIsEgotist DATA ( readable_form = "Falidogh disse que Egonk eh egoista" )

   # Clues for the second act
   Clue CrimeHappenedYesterdayAtNoon DATA ( readable_form = "O roubo ocorreu ontem ao meio-dia" )
   Clue FalidoghAndEgonkWereHuntingTogether DATA ( readable_form = "Falidogh e Egonk cacavam juntos na hora do crime" )
   Clue BigCarcassOfHuntedAnimal
   DATA ( readable_form = "Ha um esqueleto de um grande animal cacado recentemente na floresta" )
   Clue MeatWasSoldToVendorg DATA ( readable_form = "Carne e couro foram vendidas para Vendorg, o caixeiro viajante" )
   Clue EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg
   DATA ( readable_form = "Egonk ficou preparando a carne e o couro enquanto Falidogh foi negociar com Vendorg" )

   # Clues for the third act
   Clue VendorgBoughtTheStatue DATA ( readable_form = "Vendorg comprou a estatua" )

   # Bonus clue
   Clue StatueWasStolen DATA ( readable_form = "A estatua foi roubada" )


PREDICATES
   True()
   Rained()
   IsDirty (Thing)
   IsProtagonist (Character)
   At (MoveableStuff, Place)
   IsNonPortable (Thing)
   Has (Character, Thing)
   Knows (Character, Clue)
   Likes (Character, Thing)
   FriendlyTo (Character, Character)
   WontGive (Character, Thing)
   WontTell (Character, Clue)
   WontExaminePlace (Character, Place)
   HasToldClue (Clue)
   IsRevealed (Thing) # was the thing already revealed/discovered?
   IsCrimeSolved()


WORLD_STATE
   True()
   At (Ugh, AlelughsCave)
   At (Alelugh, AlelughsCave)
   At (Falidogh, FalidoghsCave)
   At (Egonk, EgonksCave)
   At (Strawberry, Garden)
   At (ManyFishingPoles, EgonksCave)
   At (Carcass, Forest)

   Knows (Ugh, StatueWasStolen)
   Knows (Alelugh, CrimeHappenedYesterdayAtNoon)
   Knows (Falidogh, FalidoghAndEgonkWereHuntingTogether)
   Knows (Egonk, FalidoghAndEgonkWereHuntingTogether)
   Knows (Egonk, BigCarcassOfHuntedAnimal)
   Knows (Egonk, MeatWasSoldToVendorg)
   Knows (Egonk, EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg)

   IsNonPortable (Carcass)
   IsNonPortable (ManyFishingPoles)

   Has (Egonk, PersonalStatue)

   Likes (Egonk, Fish)
   Likes (Falidogh, Strawberry)

   FriendlyTo (Falidogh, Ugh)
   FriendlyTo (Egonk, Ugh)

   WontGive (Falidogh, DebtCharge)
   WontGive (Falidogh, LoanRequest)
   WontGive (Falidogh, Strawberry)
   WontGive (Falidogh, BuyingCertificate)
   WontGive (Egonk, Fish)
   WontGive (Egonk, Strawberry)
   WontGive (Egonk, PersonalStatue)
   WontGive (Egonk, FireStarter)

   WontExaminePlace (Falidogh, FalidoghsCave)
```

```
   WontExaminePlace (Falidogh, Garden)
   WontExaminePlace (Egonk, EgonksCave)

   WontTell (Egonk, EgonkHasAPersonalStatue)
   WontTell (Egonk, EgonkStolenTheFireStarterHeEnvied)
   WontTell (Egonk, EgonkSaidHeDoesntBorrowHisStuff)
   WontTell (Egonk, FalidoghSaidEgonkIsEgotist)
   WontTell (Egonk, SuspectFireStarter)
   WontTell (Falidogh, FalidoghHadLoanRejected)
   WontTell (Falidogh, FalidoghIsIndebted)
   WontTell (Falidogh, FalidoghAdmittedHavingDebts)
   WontTell (Falidogh, StatueWorthsALot)
   WontTell (Falidogh, BigCarcassOfHuntedAnimal)
   WontTell (Falidogh, MeatWasSoldToVendorg)
   WontTell (Falidogh, EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg)
   WontTell (Falidogh, VendorgBoughtTheStatue)

   IsProtagonist (Ugh) # This is mandatory


ACTIONS
   # GoTo
   GoTo (Character Who, Place From, Place To)
      PREREQUISITE AND (
                     NOT EQUAL (Who, Alelugh) # Priest must stay at home, he's wounded
                     NOT EQUAL (From, To),
                     At (Who, From))
      EFFECT AND (
                 At (Who, To),
                 NOT At (Who, From))
      DATA (
         subject = "Who"
         readable_form = "Ir para <To>")


   # Take
   Take (Character Who, Thing What, Place Where)
      PREREQUISITE AND (
                      NOT IsNonPortable (What),
                      At (Who, Where),
                      At (What, Where))
      EFFECT AND (
                NOT At (What, Where),
                Has (Who, What))
      DATA (
         subject = "Who"
         animation = "7"
         readable_form = "Pegar <What>"
         active_objects = "What"
         after_script = "Fabulator:showText (ReadableParams.Who..': \'Pego.\'')")


   # TalkAbout
   TalkAbout (Character Speaker, Character Listener, Clue Topic)
      PREREQUISITE AND (
                     NOT WontTell (Speaker, Topic)
                     NOT EQUAL (Speaker, Listener)
                     Knows (Speaker, Topic)
                     EXISTS Place P (
                        AND (
                           At (Speaker, P),
                           At (Listener, P))))
      EFFECT AND (
                Knows (Listener, Topic)
                IF AND (
                      EQUAL (Listener, Alelugh)
                      EQUAL (Speaker, Ugh))
                THEN HasToldClue (Topic)

                # Talk with 'Alelugh' about 'StatueWasStolen'
                IF AND (
                       EQUAL (Listener, Alelugh)
                       EQUAL (Topic, StatueWasStolen))
                THEN Knows (Speaker, CrimeHappenedYesterdayAtNoon)

                # Talk with 'Falidogh' about 'FalidoghIsIndebted'
                IF AND (
                       FriendlyTo (Falidogh, Ugh)
                       EQUAL (Listener, Falidogh)
                       EQUAL (Topic, FalidoghIsIndebted))
                THEN AND (
                         Knows (Speaker, FalidoghAdmittedHavingDebts)
                         Knows (Speaker, FalidoghSaidEgonkIsEgotist))

                # Talk with 'Egonk' about 'EgonkHasAPersonalStatue'
                IF AND (
                       FriendlyTo (Egonk, Ugh)
                       EQUAL (Listener, Egonk)
                       EQUAL (Topic, EgonkHasAPersonalStatue))
                THEN AND (
                         Knows (Speaker, EgonkSaidHeDoesntBorrowHisStuff)
                         Knows (Speaker, StatueWorthsALot))

                # Talk with 'Falidogh' about 'CrimeHappenedYesterdayAtNoon'
                IF AND (
                       FriendlyTo (Falidogh, Ugh)
                       EQUAL (Listener, Falidogh)
                       EQUAL (Topic, CrimeHappenedYesterdayAtNoon))
```

```
                THEN Knows (Speaker, FalidoghAndEgonkWereHuntingTogether)

                # Talk with 'Egonk' about 'CrimeHappenedYesterdayAtNoon'
                IF AND (
                       FriendlyTo (Egonk, Ugh)
                       EQUAL (Listener, Egonk)
                       EQUAL (Topic, CrimeHappenedYesterdayAtNoon))
                THEN Knows (Speaker, FalidoghAndEgonkWereHuntingTogether)


                # Talk with 'Egonk' about 'BigCarcassOfHuntedAnimal'
                IF AND (
                       FriendlyTo (Egonk, Ugh)
                       EQUAL (Listener, Egonk)
                       EQUAL (Topic, BigCarcassOfHuntedAnimal))
                THEN AND (
                        Knows (Speaker, EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg)
                        Knows (Speaker, MeatWasSoldToVendorg))

                # Talk with 'Alelugh' about 'SuspectFireStarter'
                IF AND (
                       EQUAL (Listener, Alelugh)
                       EQUAL (Topic, SuspectFireStarter))
                THEN Knows (Speaker, EgonkStolenTheFireStarterHeEnvied))
      DATA (
         subject = "Speaker"
         animation = "8"
         readable_form = "Falar para <Listener>: '<Topic>'"
         active_objects = "Listener"
         before_script = "Fabulator:showText (ReadableParams.Speaker..': \''..ReadableParams.Topic..'\'')"
         after_script = "
if AngryArray == nil then
   AngryArray = { }
end

if Params.Speaker ~= 'Ugh' then
   -- Fabulator:showText ('Cu-co! Cu-co! Cu-co! Cu-co!')
   return
end

function npcAnswer (what)
   Fabulator:showText (ReadableParams.Listener..': \''..what..'\'')
end

if AngryArray[Params.Listener] then
   npcAnswer ('Você me assustou antes! Não quero papo, seu bobão!')
else
   if not TalkToAnswersTable then
      TalkToAnswersTable = {
         Alelugh = {
            CrimeHappenedYesterdayAtNoon = 'Sei, fui eu mesmo que te contei isso!',
            StatueWasStolen = 'Uma informação que pode ajudar: o crime foi ontem, ao meio-dia.',
            FalidoghHadLoanRejected = 'Humm... parece que ele está precisando de dinheiro.',
            FalidoghIsIndebted = 'Xiii, temos um endividado em Neandertown!',
            FalidoghAdmittedHavingDebts = 'Bom, mas ele só tá confirmando o que já sabíamos.',
            StatueWorthsALot = 'Isso verdade! Uma estátua dessas vale muito no mercado negro!',
            EgonkHasAPersonalStatue = 'Não me surpreende. Egonk é bem devoto, mas sempre faz os rituais sozinho.',
            EgonkSaidHeDoesntBorrowHisStuff = 'Ele tem fama de egoísta mesmo.',
            FalidoghSaidEgonkIsEgotist = 'Todo mundo diz isso...',
            SuspectFireStarter = { 'Esse acendedor é meu e estava sumido há um tempo.',
                                   'O Egonk invejava esse acendedor de fogo e o usurpou!' },
            FalidoghAndEgonkWereHuntingTogether = 'Não vejo nenhum problema nisso. Mas fica um de álibi do outro.',
            BigCarcassOfHuntedAnimal = 'Só o esqueleto? Que será que fizeram com o resto?',
            MeatWasSoldToVendorg = 'Sim, Vendorg, o caixeiro viajante, esteve por aqui ontem.',
            EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg =
               'Isso quer dizer que eles ficaram sem álibi por uns momentos.',
            VendorgBoughtTheStatue = 'Ó, puxa. Mas... quem será que vendeu?',
         },

         Egonk = {
            EgonkHasAPersonalStatue = { 'Como você sabe disso? Mas admito, tenho sim. E ela é minha! Só minha!',
                                        'Não empresto minhas coisas! Você sabe quanto vale uma estátua? Muito!' },
            CrimeHappenedYesterdayAtNoon = 'Eu sei. Nesta hora eu estava caçando com o Falidogh.',
            BigCarcassOfHuntedAnimal = {
               'Vendemos a carne e o couro para Vendorg, o caixeiro viajante.',
               'Eu fiquei preparando o bicho enquanto o Falidogh foi ao porto negociar com Vendorg.' },
         },

         Falidogh = {
            FalidoghIsIndebted = { 'Sim, eu tenho dívidas, é verdade.',
                                   'E prá piorar, aquele egoísta do Egonk não quer me emprestar dinheiro!' },
            CrimeHappenedYesterdayAtNoon = 'Foi bem na hora em que eu o Egonk caçávamos na floresta.',
         }
      }
   end -- of 'TalkToAnswersTable' definition

   local entry = TalkToAnswersTable[Params.Listener][Params.Topic]
   if entry then
      if type (entry) == 'string' then
         Fabulator:showText (ReadableParams.Listener..': \''..entry..'\'')
      elseif type (entry) == 'table' then
         for k,v in ipairs (entry) do
            Fabulator:showText (ReadableParams.Listener..': '..v..'\'')
         end
      else
         print 'FROM LUA: Argh!'
      end
   else
```

```
     Fabulator:showText (ReadableParams.Listener..': \'É mesmo?\'')
   end
end -- if AngryArray[...]

if Params.Listener == 'Alelugh' then
   if not CluesToldArray then
      CluesToldArray = { }
   end
      CluesToldArray[Params.Topic] = true
end -- if Params.Listener == 'Alelugh'")


   # ImitateDinosaur
   ImitateDinosaur (Character Imitator, Character Spectator, Place Where)
      PREREQUISITE AND (
                     At (Imitator, Where)
                     At (Spectator, Where)
                     NOT EQUAL (Imitator, Spectator))
      EFFECT AND (
                IF NOT EQUAL (Spectator, Alelugh)
                THEN NOT FriendlyTo (Spectator, Imitator)
                IF Has (Spectator, PersonalStatue)
                THEN AND (
                        NOT Has (Spectator, PersonalStatue)
                        At (PersonalStatue, Where)))
      DATA (
         subject = "Imitator"
         animation = "4"
         readable_form = "Imitar dinossauro para <Spectator>"
         active_objects = "Spectator"
         before_script = "Fabulator:showText (ReadableParams.Imitator..': \'GRRRRRRRRAAAAAAAAAAUUUUURRRRR!!!\'')"
         after_script = "Fabulator:showText (ReadableParams.Spectator..': \'Que susto!\'')

                         if AngryArray == nil then
                            AngryArray = { }
                         end

                         if Params.Imitator ~= 'Ugh' then
                            return
                         end

                         if Params.Spectator ~= 'Alelugh' then
                            AngryArray[Params.Spectator] = true
                         end

                         if Params.Spectator == 'Alelugh' then
                            Fabulator:showText(
                               ReadableParams.Spectator..': \'Deixa de palhaçada e vai descobrir quem roubou a estátua!\'')
                         elseif Params.Spectator == 'Egonk' and HaveImitated == nil then
                            HaveImitated = true
                            Fabulator:showText 'Egonk ficou brabo com você. E deixou cair algo.'
                         elseif Params.Spectator == 'Falidogh' then
                            Fabulator:showText 'Falidogh ficou brabo com você.'
                         end")


   # ExaminePlace
   ExaminePlace (Character Examiner, Place Where)
      PREREQUISITE AND (
                     At (Examiner, Where)
                     NOT WontExaminePlace (Examiner, Where))
      EFFECT AND (
                # Examine 'FalidoghsCave'
                IF AND (
                       EQUAL (Where, FalidoghsCave)
                       NOT IsRevealed (LoanRequest))
                THEN AND (
                    At (LoanRequest, Where)
                    IsRevealed (LoanRequest))

                # Examine 'EgonksCave'
                IF AND (
                       EQUAL (Where, EgonksCave)
                       NOT IsRevealed (FireStarter))
                THEN AND (
                    At (FireStarter, Where)
                    IsRevealed (FireStarter))

                # Examine 'Garden'
                IF AND (
                       EQUAL (Where, Garden)
                       NOT IsRevealed (DebtCharge))
                THEN AND(
                         At (DebtCharge, Where)
                         IsRevealed (DebtCharge)))
      DATA (
         subject = "Examiner"
         animation = "6"
         readable_form = "Examinar <Where>"
         active_objects = "Where"
         after_script = "if Params.Where == 'FalidoghsCave' and not FalidoghsCaveExamined then
                            FalidoghsCaveExamined = true
                            Fabulator:showText(
                               ReadableParams.Examiner..': \'Achei alguma coisa aqui em '..ReadableParams.Where..'.\'')
                         elseif Params.Where == 'EgonksCave' and not EgonksCaveExamined then
                            EgonksCaveExamined = true
                            Fabulator:showText(
                               ReadableParams.Examiner..': \'Achei alguma coisa aqui em '..ReadableParams.Where..'.\'')
```

```
                    elseif Params.Where == 'Garden' and not GardenExamined then
                       GardenExamined = true
                       Fabulator:showText(
                          ReadableParams.Examiner..': \'Achei alguma coisa aqui em '..ReadableParams.Where..'.\'')
                    elseif Params.Where == 'Forest' then
                       Fabulator:showText (ReadableParams.Examiner..': \'Tem um esqueleto de animal aqui.\'')
                    elseif Params.Where == 'Pier' then
                       Fabulator:showText (ReadableParams.Examiner..
                          ': \'Esse é pier onde Vendorg, o caixeiro viajante, atraca seu barco.\'')
                       Fabulator:showText(
                          ReadableParams.Examiner..': \'Vendorg compra e vende de tudo.\'')
                    else
                       Fabulator:showText (ReadableParams.Examiner..': \'Não vejo nada de anormal por aqui.\'')
                    end")


# ExamineThing
ExamineThing (Character Examiner, Thing What)
   PREREQUISITE OR (
                  Has (Examiner, What)
                  AND (
                     IsNonPortable (What)
                     EXISTS Place P (
                        AND (
                           At (Examiner, P)
                           At (What, P)))))
   EFFECT AND (
            # Examine 'LoanRequest'
            IF EQUAL (What, LoanRequest)
            THEN Knows (Examiner, FalidoghHadLoanRejected)

            # Examine 'DebtCharge'
            IF EQUAL (What, DebtCharge)
            THEN Knows (Examiner, FalidoghIsIndebted)

            # Examine 'PersonalStatue'
            IF EQUAL (What, PersonalStatue)
            THEN Knows (Examiner, EgonkHasAPersonalStatue)

            # Examine 'FireStarter'
            IF EQUAL (What, FireStarter)
            THEN Knows (Examiner, SuspectFireStarter)

            # Examine 'BuyingCertificate'
            IF EQUAL (What, BuyingCertificate)
            THEN Knows (Examiner, VendorgBoughtTheStatue)

            # Examine 'Carcass'
            IF EQUAL (What, Carcass)
            THEN Knows (Examiner, BigCarcassOfHuntedAnimal)

            # Examine 'ManyFishingPoles'
            IF EQUAL (What, ManyFishingPoles)
            THEN Has (Examiner, FishingPole))
   DATA (
      subject = "Examiner"
      animation = "5"
      readable_form = "Examinar <What>"
      active_objects = "What"
      after_script = "if Params.Examiner ~= 'Ugh' then
                        return
                     end

                     if Params.What == 'LoanRequest' then
                        Fabulator:showText 'É a resposta do banco a um pedido de empréstimo do Falidogh. Foi negado.'
                     elseif Params.What == 'DebtCharge' then
                        Fabulator:showText 'É uma intimação cobrando uma grande dívida do Falidogh.'
                     elseif Params.What == 'PersonalStatue' then
                        Fabulator:showText 'É uma estatueta pessoal, para adoração solitária no conforto do lar.'
                        Fabulator:showText 'E tem o nome do Egonk gravado nela.'
                     elseif Params.What == 'FireStarter' then
                        Fabulator:showText 'É um acendedor de fogo. Tem \'Egonk\' gravado nele.'
                        Fabulator:showText 'Mas ainda dá para perceber um outro nome, que tentaram apagar: Alelugh.'
                     elseif Params.What == 'BuyingCertificate' then
                        Fabulator:showText(
                           'É um certificado de compra. Alguém vendeu a estátua para Vendorg, o caixeiro viajante!')
                     elseif Params.What == 'Carcass' then
                        Fabulator:showText 'É um esqueleto de um animal caçado há um dia, no máximo.'
                        Fabulator:showText 'O bicho é enorme. Que será que fizeram com toda carne e couro?'
                     elseif Params.What == 'ManyFishingPoles' then
                        Fabulator:showText 'Caniços e mais caniços! Peguei um.'
                     else
                        Fabulator:showText 'Parece normal.'
                     end")


# GivePresent
GivePresent (Character Giver, Character Receiver, Thing Present)
PREREQUISITE AND (
               NOT WontGive (Giver, Present)
               NOT EQUAL (Giver, Receiver)
               NOT IsDirty (Present)
               Has (Giver, Present)
               EXISTS Place P (
                  AND (
                      At (Giver, P),
                      At (Receiver, P))))
EFFECT AND (
```

```
          NOT Has (Giver, Present)
          Has (Receiver, Present)
          IF Likes (Receiver, Present)
          THEN FriendlyTo (Receiver, Giver)

          # Giving the 'FireStarter' to 'Alelugh'
          IF AND (
                 EQUAL (Receiver, Alelugh)
                 EQUAL (Present, FireStarter))
          THEN AND(
                   Knows (Giver, SuspectFireStarter)
                   Knows (Giver, EgonkStolenTheFireStarterHeEnvied))

          # Giving the 'PersonalStatue' to 'Egonk'
          IF AND (
                 NOT IsDirty (Present)
                 EQUAL (Receiver, Egonk)
                 EQUAL (Present, PersonalStatue))
          THEN AND (
                   Knows (Giver, EgonkHasAPersonalStatue)
                   Knows (Giver, EgonkSaidHeDoesntBorrowHisStuff)
                   Knows (Giver, StatueWorthsALot)))
DATA (
   subject = "Giver"
   animation = "5"
   readable_form = "Dar <Present> de presente para <Receiver>"
   active_objects = "Receiver Present"
   before_script = "Fabulator:showText(
      ReadableParams.Giver..': \'Oi! Tome este '..ReadableParams.Present..' de presente!\'')"
   after_script = "if AngryArray == nil then
                      AngryArray = { }
                   end

                   if (Params.Receiver == 'Egonk' and Params.Present == 'Fish')
                      or
                      (Params.Receiver == 'Falidogh' and Params.Present == 'Strawberry') then
                      AngryArray[Params.Receiver] = false
                      Fabulator:showText (ReadableParams.Receiver..': \'Oh, um presente! Você não é tão mal assim...\'')
                   elseif Params.Receiver == 'Alelugh' and Params.Present == 'FireStarter' then
                      Fabulator:showText (ReadableParams.Receiver..
                         ': \'Esse acendedor é meu e estava sumido há um tempo.\'')
                      Fabulator:showText (ReadableParams.Receiver..
                         ': \'O Egonk invejava esse acendedor de fogo e o usurpou!\'')
                   elseif Params.Receiver == 'Egonk' and Params.Present == 'PersonalStatue' then
                      Fabulator:showText (ReadableParams.Receiver..
                         ': \'Ei, isso é meu! É a estátua que eu uso para fazer os rituais. Sozinho!\'')
                      Fabulator:showText (ReadableParams.Receiver..
                         ': \'Não empresto minhas coisas para ninguém. Ninguém! Entendeu?\'')
                      Fabulator:showText (ReadableParams.Receiver..
                         ': \'Ainda mais coisas caras. Sabe quanto vale uma estátua? Vale muito. Muito!\'')
                   end")


# ToFish (can't have object and action with same name...)
ToFish (Character Who, Place Where, Thing Instrument)
PREREQUISITE AND (
                NOT EQUAL (Who, Falidogh)
                EQUAL (Instrument, FishingPole)
                Has (Who, Instrument)
                At (Who, Where)
                OR (
                    EQUAL (Where, Beach)
                    EQUAL (Where, Pier)))
EFFECT AND (
           IF EQUAL (Where, Beach)
           THEN At (Fish, Beach)

           IF AND (
                  EQUAL (Where, Pier)
                  NOT IsRevealed (BuyingCertificate))
           THEN AND(
                    At (BuyingCertificate, Pier)
                    IsRevealed (BuyingCertificate)))
DATA (
   subject = "Who"
   animation = "5"
   readable_form = "Pescar"
   active_objects = "Instrument"
   after_script = "Fabulator:showText (ReadableParams.Who..': \'Pesquei algo!\'')")


# Accuse
Accuse (Character Detective, Character Listener, Character Accused)
PREREQUISITE AND (
                IsProtagonist (Detective)
                EQUAL (Listener, Alelugh)
                EXISTS Place P (
                   AND (
                      At (Detective, P)
                      At (Alelugh, P))))
EFFECT IF AND (
             EQUAL (Accused, Falidogh)
             HasToldClue (FalidoghHadLoanRejected)
             HasToldClue (FalidoghIsIndebted)
             HasToldClue (FalidoghAdmittedHavingDebts)
             HasToldClue (StatueWorthsALot)
             HasToldClue (FalidoghAndEgonkWereHuntingTogether)
             HasToldClue (BigCarcassOfHuntedAnimal)
```

```
                HasToldClue (MeatWasSoldToVendorg)
                HasToldClue (EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg)
                HasToldClue (VendorgBoughtTheStatue))
          THEN IsCrimeSolved()
   DATA (
      subject = "Detective"
      animation = "8"
      readable_form = "Acusar <Accused> do roubo."
      active_objects = "Listener"
      before_script = "Fabulator:showText (ReadableParams.Detective..
         ': \'Já sei quem é o ladrão: '..ReadableParams.Accused..'!\'')"
      after_script = "if not CluesToldArray then
                         CluesToldArray = { }
                      end

                      function priestAnswer(what)
                         Fabulator:showText (ReadableParams.Listener..': \''..what..'\'')
                      end

                      if Params.Accused == 'Ugh' then
                         priestAnswer 'Isso é uma confissão?! Nãão... trabalhe sério, pô!'
                      elseif Params.Accused == 'Alelugh' then
                         priestAnswer 'Eu sou o ladrão?! Isso não tem o menor fundamento. Vai investigar melhor.'
                      elseif Params.Accused == 'Egonk' then
                         priestAnswer 'Não, não há evidências suficientes para acusá-lo.'
                      elseif Params.Accused == 'Falidogh' then
                         if CluesToldArray.FalidoghHadLoanRejected
                            and CluesToldArray.FalidoghIsIndebted
                            and CluesToldArray.FalidoghAdmittedHavingDebts
                            and CluesToldArray.StatueWorthsALot
                            and CluesToldArray.FalidoghAndEgonkWereHuntingTogether
                            and CluesToldArray.BigCarcassOfHuntedAnimal
                            and CluesToldArray.MeatWasSoldToVendorg
                            and CluesToldArray.EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg
                            and CluesToldArray.VendorgBoughtTheStatue
                         then
                            priestAnswer 'É, as provas realmente indicam que foi ele! Obrigado e parabéns!!'
                            Fabulator:showText (ReadableParams.Detective..': \'Não contavam com minha astúcia!\'')
                         else
                            priestAnswer 'Não, não há evidências suficientes para acusá-lo.'
                         end
                      end")


   # Pretext actions

   # LetItRain
   LetItRain()
      PREREQUISITE True()
      EFFECT Rained()
      DATA (
         pretext_action = "true"
         before_script = "Fabulator:showText ('Está chovendo!')")


   # Intervention actions

   # GrowNewStrawberry
   GrowNewStrawberry()
      PREREQUISITE AND (
                      NOT At (Strawberry, Garden)
                      Rained())
      EFFECT At (Strawberry, Garden)
      DATA (
         intervention_action = "true"
         after_script = "Fabulator:showText ('Com toda aquela chuva, um morango novo cresceu na horta.')")


   # DropAndDirty
   DropAndDirty (Character Who, Thing What)
      PREREQUISITE AND (
                      NOT EQUAL (What, Strawberry)
                      NOT EQUAL (What, Fish)
                      NOT IsDirty (What)
                      Has (Who, What)
                      Rained())
      EFFECT IsDirty (What)
      DATA (
         intervention_action = "true"
         after_script = "Fabulator:showText ('Ops, deixei o '..ReadableParams.What..' cair numa poça. Ficou imundo.')")

# First Act
GOAL
   AND (
      Knows (Ugh, FalidoghHadLoanRejected)
      Knows (Ugh, FalidoghIsIndebted)
      Knows (Ugh, FalidoghAdmittedHavingDebts)
      Knows (Ugh, StatueWorthsALot)

      Knows (Ugh, EgonkHasAPersonalStatue)
      Knows (Ugh, EgonkStolenTheFireStarterHeEnvied)
      Knows (Ugh, EgonkSaidHeDoesntBorrowHisStuff)
      Knows (Ugh, FalidoghSaidEgonkIsEgotist))


# Second Act
GOAL
   AND (
```

```
      Knows (Ugh, CrimeHappenedYesterdayAtNoon)
      Knows (Ugh, FalidoghAndEgonkWereHuntingTogether)
      Knows (Ugh, BigCarcassOfHuntedAnimal)
      Knows (Ugh, MeatWasSoldToVendorg)
      Knows (Ugh, EgonkPreparedMeatWhileFalidoghNegotiatedWithVendorg))


# Third Act
GOAL
   IsCrimeSolved()
```

Cenário

Para dar uma noção dos espaços que podem ser visitados pelos usuários, a Figura 29 mostra uma vista superior do cenário utilizado n'*A Estória de Ugh 2*.

Figura 29: Vista superior do cenário d'*A Estória de Ugh 2*.