

Guiding Interactive Drama

Peter Weyhrauch

January, 1997

CMU-CS-97-109₂

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Thesis Committee:

Joseph Bates, chair

Jaime Carbonell

Roger Dannenberg

Brenda Laurel, Interval Research Corporation

© 1997 Peter Weyhrauch

All rights reserved

This work was supported in part by Fujitsu Laboratories, Mitsubishi Electric Research Laboratories, and Justsystem Corporation.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any other parties.

Keywords: computer science, artificial intelligence, interactive drama, interactive story, interactive fiction, interactive art, interactive entertainment, abstract adversary search, memoized search, sampled search, sampling search, SAS, SAS+, MFC, MMFC, aesthetic evaluation function, user experience, Oz Project



School of Computer Science

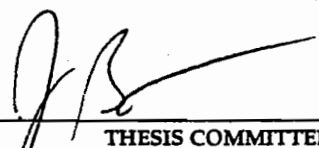
DOCTORAL THESIS
in the field of
Computer Science

Guiding Interactive Drama

PETER WEYHRAUCH

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

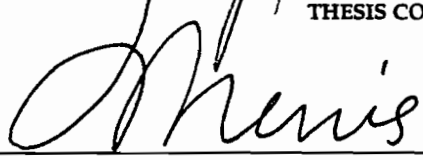
ACCEPTED:



THESIS COMMITTEE CHAIR

1/31/97

DATE

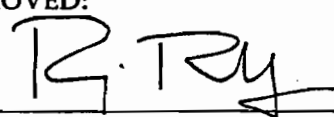


DEPARTMENT HEAD

May 16, 1997

DATE

APPROVED:



DEAN

May 16, 1997

DATE

Abstract

In traditional drama, an audience watches a story presented by characters on a stage. *Interactive Drama* changes the audience to a single User who enters that stage, interacting with the characters, and participating in the story.

This thesis concerns computer-based interactive drama. The computer system allows a User to interact with simulated worlds which are inhabited by dynamic and complex autonomous characters and shaped by a flexible, aesthetically pleasing story.

Like a traditional playwright, the artist who creates an interactive drama (ID) has a set of themes and ideas to be conveyed. Unlike the actions of characters in a play, however, the actions of the User in an ID are not known at story creation time, so an ID can not have a set script. The problem is to shape the experience of the User to conform to the set of themes and ideas to be conveyed, given that the actions of the User are not under the control of the artist. This work proposes solving this problem by dynamically monitoring and subtly guiding the experience of the User.

This dissertation describes an architecture called Moe that is designed to provide dramatic guidance. Moe uses abstract, adversary search with an aesthetic evaluation function to decide how and when to guide the User's experience. The first technical achievement of this work is a demonstration of the ability to capture a dramatic aesthetic (for one interactive drama) in an automated evaluation function that can examine an experience and determine its quality, much as a movie critic determines the quality of a film. This result is supported by statistically demonstrating a high degree of correlation ($r = .87$) between the evaluation function and the human artist.

The second main technical achievement of this work is the implementation of three search strategies (SAS and SAS+, both based on random sampling; and MMFC, which uses memoization to implement a full-depth search) and a search state that seem capable of effectively modelling and guiding an interactive dramatic experience. The abstract search state includes the important aspects of the physical world, the characters, the presentation medium, and the User, including her mental, emotional, and physical states. Moe has not yet been connected to a running interactive experience, but instead Moe has been tested with a variety of Simulated User Types. On "average" Users, SAS, SAS+, and MMFC are able to improve Users' experiences from the 50th percentile, to the 94th, 98th, and 99th percentile, respectively. This is a large and artistically meaningful increase.

This dissertation describes the first implementation of a system designed to provide centralized dramatic guidance in an interactive drama. Future work remains to determine whether the success of Moe in simulation can be effectively transferred to success with real Users.

For my mom, dad, and sister

Acknowledgements

First, I must thank Joe Bates. He has helped me in more ways than I can recall. Joe is a mentor who has a great knowledge of many different fields and has an ability to use this knowledge to help solve problems. Luckily, he has shown me some of this. By creating the Oz Project, Joe has given me the fabulous opportunity to work on something new, exciting, and fun. Finally, Joe has convinced me that research is best done in the mind, not on the computer. And, perhaps more importantly, that no problem is too hard. He is both an advisor and a friend.

My committee made my work better. Jaime brought a great deal of perspective to my thesis. Through our discussions I discovered his fantastic ability to quickly pinpoint important issues and steer me toward good ideas. Roger brought his experience from working with computers in the arts. His concrete suggestions about methodology for this new area were very helpful. Finally, Brenda. Besides laying tremendous groundwork in this area, she has also been an inspiration. She knew what I was talking about, and she let me know that people were waiting for this. She was always positive and always encouraging and most of all excitable. I left every meeting with her feeling as if I wanted to (and could) conquer the world.

Of course I would not have been able to do this without the support of the SCS. This is an incredible place that fosters some of the best work in the world. Obviously the place is the people, but I think I must thank especially the founders and visionaries of this department. They made it what it is today.

Thanks to Cathy Copetas. And of course one can't leave here without knowing, liking (and perhaps at times harassing) Sharon. I have a special place in my heart for Sharon.

I want to thank the Oz Group: Joe, Bryan, Scott, Mark, Phoebe, Michael, and members past. They are an incredible group who have a remarkable capacity for listening, thinking, and then commenting. They have the patience and vision to change the world. Phoebe also throws a mean party.

Thanks to Koichi and Dr. Morita for supporting me for these years.

Old and new, I want to thank my many friends for all the fun we had, and the support they have given me over the years. Without them I would literally be insane. My high school buddies: Rich, Jon, and Raj. My college friends: Livia, Chun, Irene, Kate, RF, David, and especially Amy. Shanghua (my first friend at CMU), and Manpreet (my second). DJ James—what we've been through. The girls: Patty, Terri, Maria. My officemates, past and present. My dancing fools: Victoria and Judy. Shirley, Todd, Anwar, James, Juan, Dina, Lisa, Tom, Jim and Cathy, Mike, Lori, Matt and Linda, Belinda, Andrew, csgrad88, and many others. Finally, I want to thank especially two friends who have made tremendous contributions to my life and work here.

Bryan Loyall has been a special person for me. We've worked together, we've lived together, we've travelled together. Heck, we've even given tag-team talks together. Professionally, he is my sounding board. All ideas must be bounced off Bryan for approval. But that isn't his only function of course: sometimes better ideas come bouncing back! Personally, he has been a constant friend and support.

And then there's that genius/troll, Mr. Corey Kosak. We all have something to learn from his flawed yet brilliant mind. Of my friend, I can only say this. Of all the souls I have encountered in my travels, his was the most ... *bitter*.

My family has been always supportive. My mom, even with her constant prodding ("Just get it done!"), has always known that I would find the right path. And my dad, who has always challenged my mind along the way. My sister Katie was my second half when we were young. She and I are still together. Yasuko and Harold, who have given me more than I deserved. Daniel: he is little, but has a loud voice. Aunt Judy, my constant supporter, and cousins Deb and Kim, and my grandmother. There is also a special mention for Gosper. I count him as my family, too.

And finally, my sweetheart Charmaine, for her love and support.

Contents

1	Introduction	1
1.1	Definition of Interactive Drama	1
1.2	Authoring Interactive Drama	3
1.3	The Oz Architecture for Interactive Drama	5
1.4	Guiding Interactive Drama: Related Approaches	7
1.5	The Moe Architecture for Dramatic Guidance	9
1.6	Related Work	13
1.7	Road Map To Dissertation	20
	Part I The Evaluation Function	21
2	<i>Tea For Three</i>	23
2.1	Background	23
2.2	The USER MOVES	26
2.3	The MOE MOVES, briefly	31
2.4	USER MOVES are Abstractions	33
2.5	Two Abstract Scenarios	35
3	The Evaluation Function	41
3.1	Overview	42
3.2	The <i>Thought Flow</i> Feature	43
3.3	The <i>Activity Flow</i> Feature	46
3.4	The <i>Options</i> Feature	48
3.5	The <i>Motivation</i> Feature	52
3.6	The <i>Momentum</i> Feature	55
3.7	The <i>Intensity</i> Feature	57
3.8	The <i>Manipulation</i> Feature	71
3.9	Normalized Values and Weights: Adding it Up	72
3.10	Discussion	72

4	Testing the Evaluation Function	75
4.1	Determining Success	75
4.2	Study Methodology	78
4.3	Study Results	80
4.4	Discussion	86
 Part II Search		 89
5	MOE MOVES	91
5.1	Introduction to Search	92
5.2	The MOE MOVES	96
5.3	Creating a Set of MOE MOVES	104
6	The Search State	109
6.1	Motivation For Having a Search State	109
6.2	The Search State	112
6.3	A Search State Example	120
7	Searching Strategies	135
7.1	Full-Depth Adversary Search	135
7.2	Sampling Adversary Search (SAS, SAS+)	143
7.3	Memoized Future Contribution Search (MFC)	148
8	Evaluation and Discussion of Search	159
8.1	An Evaluation Method	160
8.2	Data Gathered	163
8.3	Discussion of Results	169
8.4	Conclusion to Part II	171
 Part III Conclusion		 173
9	Conclusion	175
9.1	Contribution of the Thesis	175
9.2	Future Work, Limitations, and Improvements	176
9.3	Future Of The Art	180

Part IV Appendices	183
A Other Rater Comparisons	185
B Legal/When Functions for MOE MOVES	187

List of Figures

1.1	The Oz System Architecture for ID	5
1.2	A Partial Screen Shot of Two Woggles	6
1.3	Plot Graph of <i>Tea For Three's</i> USER MOVES	8
1.4	Moe's Decision Making System: <i>Adversary Search with Evaluation</i>	9
1.5	A Complete Interactive Drama System	11
2.1	<i>Tea For Three</i> DAG: four USER MOVES	26
2.2	<i>Tea For Three</i> DAG: eight USER MOVES	28
2.3	<i>Tea For Three</i> DAG: eleven USER MOVES	29
2.4	<i>Tea For Three</i> DAG: fourteen USER MOVES	30
2.5	<i>Tea For Three</i> DAG: all USER MOVES	30
3.1	<i>Thought Flow</i> in the Good Scenario	44
3.2	<i>Thought Flow</i> in the Bad Scenario	45
3.3	<i>Activity Flow</i> in the Good Scenario	47
3.4	<i>Activity Flow</i> in the Bad Scenario	48
3.5	12 Goals used by <i>Options</i>	49
3.6	Tracking Goals for <i>Options</i> in the Good Scenario	50
3.7	Tracking Goals for <i>Options</i> in the Bad Scenario	51
3.8	Table of Activities used to pursue Goals	53
3.9	<i>Motivation</i> in the Good Scenario	54
3.10	<i>Motivation</i> in the Bad Scenario	55
3.11	<i>Momentum</i> in the Good Scenario	56
3.12	<i>Momentum</i> in the Bad Scenario	57
3.13	Ideal Graph of User's Excitement	58
3.14	Tracking Excitement During Made-up Scenario	60
3.15	Excitement Graph from Made-Up Scenario	60
3.16	Table of Nine Facts and their Importance	61
3.17	Types of Knowledge and their Numerical Representation	62
3.18	Excitement Chart and Graph for the Good Scenario	68
3.19	Excitement Chart and Graph for the Bad Scenario	69

3.20	Matching Actual to Ideal Excitement in Made-Up Scenario	70
3.21	Matching Actual to Ideal Excitement in the Good Scenario	71
3.22	Matching Actual to Ideal Excitement in the Bad Scenario	71
3.23	Weights of the Features of the Evaluation Function	72
4.1	An Example Scenario, as Presented to Raters	77
4.2	Distribution of Values and Mapping to Stars	78
4.3	Results of Evaluation Function and Three Raters	81
4.4	“Ev Fn vs. Artist” Does Well in Comparison	86
5.1	A Complete Interactive Drama System	92
6.1	The History and Projected Future	110
6.2	The Basic Search State	112
6.3	Causing MOE MOVES that add to SS.candidateUserMoves	114
6.4	MOE MOVES that create active move substitutions	115
6.5	MOE MOVES (<i>hints</i>) that immediately change SS.moveMultipliers	118
6.6	MOE MOVES that go into SS.futureHints	119
6.7	DAG of USER MOVES and precedence relations	119
7.1	ChooseMOEMOVE in Pseudocode	139
7.2	MoeNode and UserNode in Pseudocode	140
7.3	DAG of USER MOVES and precedence relations	141
7.4	SAS:MoeNode and SAS:UserNode in Pseudo-code	144
7.5	An algorithm for SHEF and SousSHEF written in pseudo-code	145
7.6	Effectiveness of SAS, SAS+ on “Average Users”	148
7.7	FC:MoeNode and FC:UserNode in Pseudo-code	150
7.8	MFC:MoeNode and MFC:UserNode in Pseudo-code	151
7.9	Effectiveness of SAS, SAS+, MMFC on “Average Users”	158
8.1	A Simulated Interactive Drama System	161
8.2	An algorithm for CHOOSEUSERMOVE written in pseudo-code	162
8.3	Bad User that Ignores Hints	164
8.4	Bad User that Obeys Hints Probabilistically	164
8.5	Bad User that Always Does Hints	165
8.6	Average User that Ignores Hints	165
8.7	Average User that Obeys Hints Probabilistically	166
8.8	Average User that Always Does Hints	166
8.9	Good User that Ignores Hints	167
8.10	Good User that Obeys Hints Probabilistically	167

8.11 Good User that Always Does Hints	168
8.12 All Search Means Plotted vs. User Type	168

Chapter 1

Introduction

Dusk in Cairo. You are walking down a dusty street, on your way to a museum. You wish to discover the origin of a symbol found in a book belonging to your father the archeologist. Your father had told you, before he died, that this book contained his life's work. Suddenly, you hear footsteps. A man grabs the shoulder of your leather jacket. He says he is Horace, a friend of your father's, and if you come with him, he can help you. You stop in Cafe Tut — he is charming and surprisingly knowledgeable about your father. You absorb his every word, until you suddenly get an eerie feeling. Horace is asking strange questions about the book. Your father's book. You then notice a suspicious gun-like bulge in his suit jacket. You slowly get up, mentioning the restroom, and then bolt out of the cafe, barely evading Horace's hands as you jump into the nearest taxi. "Quick, get me to the museum," you scream over the sound of the screeching tires. You relax as you see Horace getting smaller though the rear window of the cab. "Yes, Sir," the gold-toothed cabbie replies with a smile, heading quickly in the wrong direction.

1.1 Definition of Interactive Drama

To me, Interactive Drama should be like what the description above suggests. You find yourself immersed in a fantasy world where there seem to be no limits. You are free to interact with the many exciting characters, explore the beautiful terrain, or follow up on any lead. Yet something is *happening*. It isn't just random. Although you feel as if any of number of possible adventures awaits you, you are confident that your experience will be good. Some force is controlling things to make the world dramatic. Although you can never be sure your experience will have a happy ending, you will always be happy with your experience.

The following is Laurel's definition of interactive drama (which she also calls Interactive Fantasy) from her dissertation *Toward The Design Of A Computer-Based Interactive Fantasy System*[21]:

An “interactive drama,” then, is a first-person experience within a fantasy world, in which the user may create, enact, and observe a character whose choices and actions affect the course of events just as they might in a play. The structure of the system proposed in the study utilizes a playwriting expert system that enables first-person participation of the user in the development of the story or plot, and orchestrates system-controlled events and characters so as to move the action forward in a dramatically interesting way.

Although this definition is ten years old, the words say almost all of what I want to say. The main difference is the technology I propose for providing dramatic guidance. Instead of saying “playwriting expert system” I would say “drama manager based on adversary search with an aesthetic evaluation function.” (The basic idea of search with evaluation for interactive drama was proposed by Bates in [4].) I include Laurel’s definition here not only because it is relevant, but because in many ways I see my work as the logical successor to her work, and thus my definition is very similar to hers.

My definition: *Interactive Drama* (ID) is the presentation by computers of rich, highly interactive worlds, which are inhabited by dynamic and complex characters, and also inhabited by a User, whose experience is shaped in this world by a dramatic destiny.

The job of an interactive drama system is to subtly guide the experience of the User, so that she retains her freedom while fulfilling her destiny. This is the real challenge of interactive drama.

My thesis is that Interactive Drama is possible. Although I will not prove this thesis, my evidence is the implementation and verification of two core components of a designed drama manager: one, an aesthetic evaluation function that can judge the quality of a User’s experience; and two, an adversary search mechanism that can effectively guide the User’s experience by using this evaluation function. These have been implemented to guide one interactive drama, called *Tea For Three*.

1.1.1 More Definition

For this dissertation, I shall call the person interacting in an interactive drama the *User* or *she*. I shall call the work of art an *interactive drama*, and I shall call the creator of the interactive drama, *the artist*, or *he*¹. *Guiding* an interactive drama means gently guiding the experience of the User to fulfill her destiny. Thus, the title of this dissertation is *Guiding Interactive Drama*.

An important phrase in my definition is “highly interactive.” The word “interactive” distinguishes ID from traditional media, while “highly interactive” indicates that the User

¹In order to break any mental preconceptions of the gender of the User, I am choosing to use a feminine pronoun when referring to the User. In order to make the distinction between the User and artist linguistically clear, I shall refer to the artist who creates an interactive drama as he. I refer to the reader of this dissertation as *you*. Sometimes *I* means the writer of this dissertation; sometimes *I* means the artist who created *Tea For Three*. I will point out the difference when necessary.

is choosing what to do, say, and think at all times. This is in contrast to other interactive media such as hypertext, where the User is given only a small number of fixed choices.

If this were conventional drama, the author alone would decide exactly what happens to the protagonist. In my model of interactive drama, the User is the protagonist, and the pursuit of her goals provides the central action of the plot. This is important since it means the choices the User makes must be significant to the plot. To me, the power of interactive drama will come from forcing the User to make difficult choices. How to make these decisions truly relevant and interesting is one of the challenges faced by artists creating interactive drama.

“Dramatic destiny” is the other half of this same challenge. Even though the User has the freedom to choose what to do, say, and think, she must have a destiny, which is the artist’s vision for a particular interactive drama. Unlike Aristotle’s notion of destiny, which is a sequence of actions and events specified by the gods (or in this case the artist), my notion of destiny is abstracted from exact events. A destiny is a set of qualities that the action must have.

As an example of what I mean, consider this overly simple destiny: *This story has three parts: a beginning, where a goal is chosen by or given to the User; a middle, where she pursues this goal unsuccessfully; and an end, where finally the goal is satisfied, and all her frustration is dispelled.* Notice this destiny doesn’t say what the goal is or exactly how her frustration is dispelled. A destiny has to be like this, because in the medium of interactive drama, the artist cannot control all events. In particular, the artist cannot control the actions of the User. As I’ve said above, one of the key challenges in interactive drama is balancing the freedom of the User against the destiny given by the artist. However, as we shall see below, a destiny still has a very specific instantiation in any interactive drama.

1.2 Authoring Interactive Drama

The definition of interactive drama that I have given requires the interactive drama system to guide the experience of the User to her destiny. What is left out of this definition is exactly what this destiny is. This is where the artist who is creating the interactive drama comes in. The experience that John Woo would want you to have is probably very different from the one that Woody Allen would try to give you.

The question is: what exactly does the artist have to do to get the experience of the User to match the destiny he envisions. The answer is that the artist must create (or direct the creation of) all aspects of the interactive drama: the physical world, the characters, and the specification of the destiny.

The physical world includes the design and workings of the settings and objects that make up the realm of that “rich, highly interactive world.” John Woo probably has you roaming around Hong Kong with a gun. But watch out, because in his world nobody ever runs out of bullets. On the other hand, Woody Allen probably has you walking around in Manhattan, armed with little more than feelings of anxiety. The exact details matter. Where Users go matters. What Users see matters. How Users think about the world matters.

Everything that influences the experience of a User matters to the artist, so every detail of the physical world must be under the control of the artist.

The same is true for the “dynamic and complex characters” that inhabit this world. The artist will specify the personality and behavior of all the characters. In John Woo’s world, most characters are trying to kill the User, but that doesn’t stop the User and her friend from sharing a touching moment. In Woody Allen’s world there will be many interesting and possibly dangerous personalities as well. The point is that every artist has a different set of unique characters that will be the cast of his interactive drama. As above, no artist wants to give up any control over the design of his characters. The design of interactive characters with real personalities is, of course, a hard problem. See [32, 25] for discussions about and references related to this aspect of interactive drama.

This leads us to the “dramatic destiny.” The destiny of the User is what *happens*. In John Woo’s world, the User has lots of exciting fights, gets double-crossed a couple of times, makes some good friends, may or may not succeed in his objectives, but always has to figure out what’s important. In Woody Allen’s world, the User gets caught up in a domestic situation where life is thrown out of balance, only to be rebalanced later, in some humorous way.

In a non-interactive medium, an artist maintains the protagonist’s destiny by creating a set of scenes that cause the destiny to be fulfilled. However, in an interactive drama, as I have defined it, the User is free to say, do, and think whatever she wants. Here is the crux of the problem. How can an artist have any control over the experience of a User that is free to do, say, or think anything?

The answer is that the artist specifies a destiny, which encodes the broad qualities the experience must have, rather than just one specific sequence of events that have that quality. This answer implies that the artist must give up the specific kind of control he enjoyed in non-interactive media. (See Section 9.3 for speculations on why, in the future of this art form, the artist might reject this control even if he had it.) What he can control is the quality of the experience, and that is the real defining property of interactive drama as an art form.

Creating the destiny is really a two-faceted process. Not only must the artist specify the destiny, but he must also specify the destiny in such a way that a computer can guide the User’s experience to fulfill that destiny. Because the artist will not be available to make judgements about what should happen in an interactive drama, the artist must give the computer system the ability to do it for him.

As we shall see, the purpose of all the technology implemented for this thesis is to allow the artist to specify a destiny in such a way as to let the system guide the User’s experience at a later time.

The next section describes a complete system incorporating all aspects of an interactive drama. In it, we will see parts for the world, the characters, and a drama manager, which is the part of the system responsible for maintaining the User’s destiny.

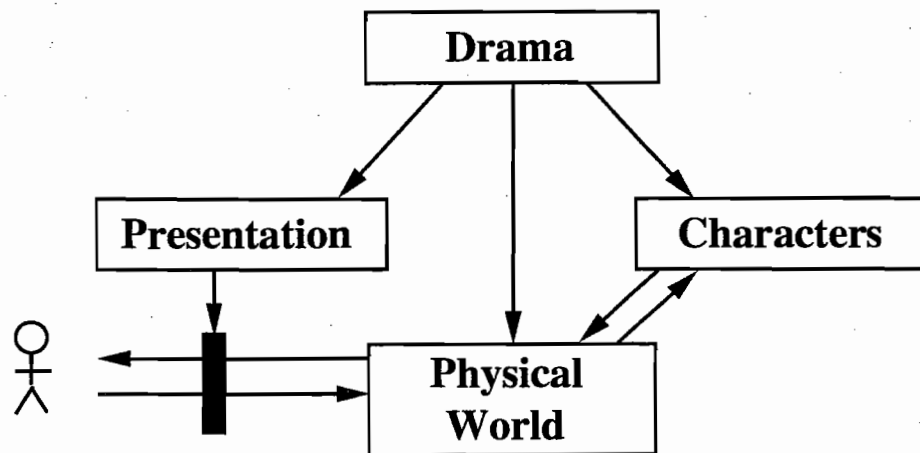


FIGURE 1.1: The Oz System Architecture for ID

1.3 The Oz Architecture for Interactive Drama

The Oz Project is a group of researchers that share this dream to create interactive drama as I've defined it. Together, we (the Oz Project researchers) have been designing and developing computer systems and technology that allow artists to create and present ID[4]. In particular, we have been creating important pieces of a complete interactive drama system: physical worlds, characters, and drama, as well as ways to connect these pieces together. Let's take a look at the framework that Oz has developed and mention some of the work on worlds and characters that complement my work on drama.

Figure 1.1 shows the basic architecture proposed by Oz for creating interactive drama. The Oz architecture includes a simulated physical world, several autonomous characters, a User, a presentation system, and a drama manager. The drama manager is the system that tries to subtly guide the User's experience so that she can fulfill her dramatic destiny. The physical world module contains a model of the scenery, the inanimate objects, each character's body, and of the User's body. Outside the physical world, a model of mind and personality controls each character's actions. The User's actions are controlled by the User. Sensory information is passed from the physical world to the User through an interface controlled by a presentation system. As shown, the drama manager can influence the characters' minds, the physical world, and the presentation system. As we will see, the drama manager also monitors the physical world, characters, and the User's actions.

The Oz Project has three primary research foci: characters, presentation, and drama. As in traditional media, we believe each of these areas is important for creating rich interactive drama.

In our research on characters we study how to create computer controlled agents that can inhabit our interactive dramas. We believe these characters should appear reactive, goal-directed, emotional, modestly intelligent, and capable of using natural language [5, 6, 26]. Of particular interest are Neal Reilly's recent PhD dissertation *Believable Social and Emotional Agents*[32] and Loyall's forthcoming PhD dissertation *Believable Agents:*

Building Interactive Personality[25]. Sengers is studying transitions between behaviors in order to create agents that can combine many behaviors and still make coherent sense to the User.[35]

Currently, the project has two different presentation models for interactive drama: textual and animated. None of the systems described below have dramatic guidance, but instead consist of just a physical world, characters, and a User.

The textual system uses text as the interface medium. The world and characters are described through text, and the User's actions are given to the computer through text. An early Oz text-based world featured a cat named *Lyotard*, who lives in a vacated apartment. While taking care of the apartment, the User tries to make friends with Lyotard by winning his trust with affection or food[7]. More recently, several small, text-based interactive dramas have been implemented by Neal Reilly, including: *Robbery World*, *Office Politics*, and *The Playground*[32].

For the most part, Oz presentation research deals with how to generate English narrative text[17, 19]. Please see Mark Kantrowitz's forthcoming PhD dissertation[18], which deals with one aspect of this problem.

In the animated interactive dramas, the world and characters are presented graphically in real-time. Humans can interact with the system by controlling a User-Creature with a mouse. In our first animated interactive drama, *Edge Of Intention*[8], the User plays with, fights with, and possibly makes friends with a trio of cute creatures named *Woggles*. Loyall[25] has extended the interface to include text-bubble output from the characters and typing input for the User. Figure 1.2 shows two *Woggles* vying for the User's attention. Sengers is creating another graphical interactive drama, *Industrial Graveyard*, which contains discarded objects attempting to eke out a marginal existence under the unfriendly eye of an overseer.

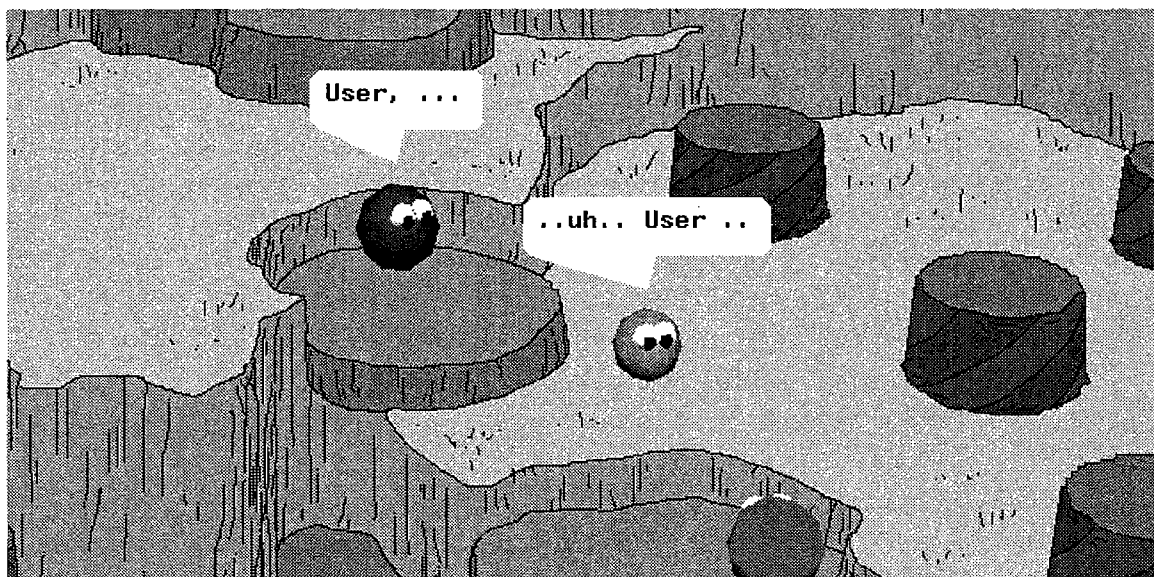


FIGURE 1.2: A Partial Screen Shot of Two Woggles

As I mentioned, in our research on drama we study how to specify a dramatic destiny so that it can be used to guide the experience of the User[3, 20].

Before I describe my approach to interactive drama, I will list some other approaches that have been suggested or used for providing dramatic guidance in an Oz-like world. After that, I will describe my drama system, Moe. Overall, this work thesis provides evidence that Moe can be used to guide interactive drama.

1.4 Guiding Interactive Drama: Related Approaches

The first possible approach to drama is to have no drama component at all; just have an exciting place with exciting characters and lots of things to do. The idea is that if there is just enough stuff going on, then a number of dramatic things should happen by chance, and thus the world will seem dramatic. This is the approach taken by *Edge of Intention*, for example.

A second approach is to have an implicit, decentralized dramatic structure enforced by the world or characters. This is the approach taken by most adventure games and video games. From the original Colossal Cave all the way to Broderbund's MYST, this type of game gets more exciting as you uncover more of the story by moving around and solving puzzles. Likewise, from Atari's Asteroids to id's Doom, video games get more exciting as the tasks become harder and the creatures get harder to kill. A recent AAAI paper by Sgouros, et al.[36] explores applying this technique via characters to "an interactive travel story environment for Greek mythology."

DEFACTO

A third approach is to have a fixed story sequence, with the presentation monitored and controlled, when necessary, by a central director. This type of director was implemented by Galyean in a work called *Dogmatic* (described in his PhD dissertation, *Narrative Guidance of Interactivity*[15]). Galyean shows how directable characters and cinematic techniques can be used to ensure that the fixed story happens, even though the User is free to act in an immersive 3D world. I believe this type of system would actually make an excellent complement to Moe, because although Galyean used a fixed story sequence, it seems as if his presentation-directing system could be applied to a variable story sequence, such as one produced by a User in an interactive drama guided by Moe.

A fourth approach is to have an explicit, centralized drama manager. This is the model I have chosen to use. Let's look at two previously reported examples of the centralized approach.

In her dissertation[21], Laurel proposes a centralized drama system based on an expert system, called the PLAYWRIGHT. As each new incident in the plot is to be created, the PLAYWRIGHT gives all characters the formal specifications for the next incident. One by one, the characters submit suggestions of action back to the PLAYWRIGHT, which simulates the effect of the actions and evaluates the results of the actions according to the formal specifications. The PLAYWRIGHT chooses to enact the first acceptable suggestion, or, in the case of no acceptable suggestions, mandates a different action of its own design. The PLAYWRIGHT also has the ability to modify its models, the characters, and its

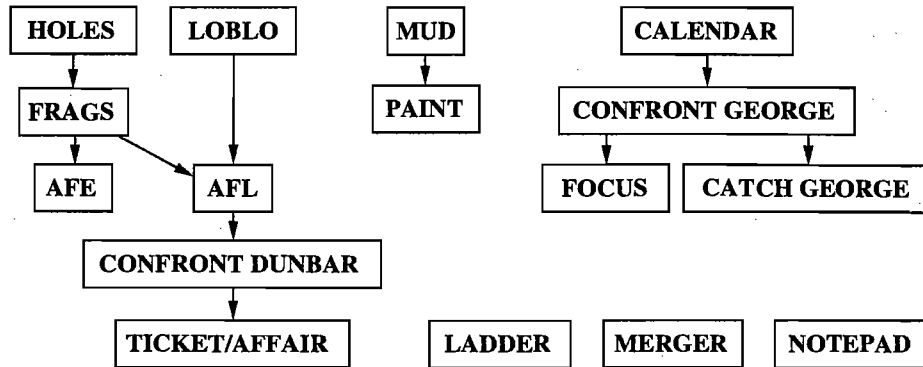


FIGURE 1.3: Plot Graph of *Tea For Three's* USER MOVES

problem solving strategies, as well as improve its own behavior by using positive and negative feedback.

Laurel's PLAYWRIGHT is different from Moe in several ways. First, PLAYWRIGHT was a suggested design, not an implemented system. Second, she proposed using a rule-based expert system, while I am proposing to use an adversary search-based approach. Third, PLAYWRIGHT's inference engine (described above) is different from Moe's approach of providing occasional dramatic guidance.

However, of all the work I've seen, Laurel's is the most closely related. She discussed or brought up many issues in her dissertation that later show up concretely in mine.

A second centralized approach was considered by Oz researchers and others in a series of live experiments. We explored an approach called the Plot Graph Model[20]. From the article:

The basic idea of the plot graph is that the major scenes of the story form a partial order and are thus linked together as a directed acyclic graph. The nodes of the graph represent events and situations that are the major moments of the story. These are similar to the more simple plot fragments in Lebowitz's UNIVERSE system [23]. The edges of the graph represent the "must precede" relation and are further annotated with hints or obstacles that can affect how the User gets from one scene to the next. The graph is used to help direct the User to experience the story the author has created.

This plot graph is different from a hypertext structure, because it is not a "choice graph." There is a frontier of available next nodes, any of which could happen. Unlike hypertext, a node that doesn't happen immediately can always happen later. Figure 1.3 shows the plot graph of the USER MOVES (defined later) for *Tea For Three*, the interactive drama my work concerns. As you will see in the section on search, Moe uses the plot graph for legal USER MOVE generation only, not for deciding how to guide the User's experience.

The Plot Graph Model was our first attempt to provide a mechanism for guiding the User's experience in an interactive drama. Probably the biggest criticism of this model is

d.iff.
b-t-w
PLAYWRIGHT
+
MOE

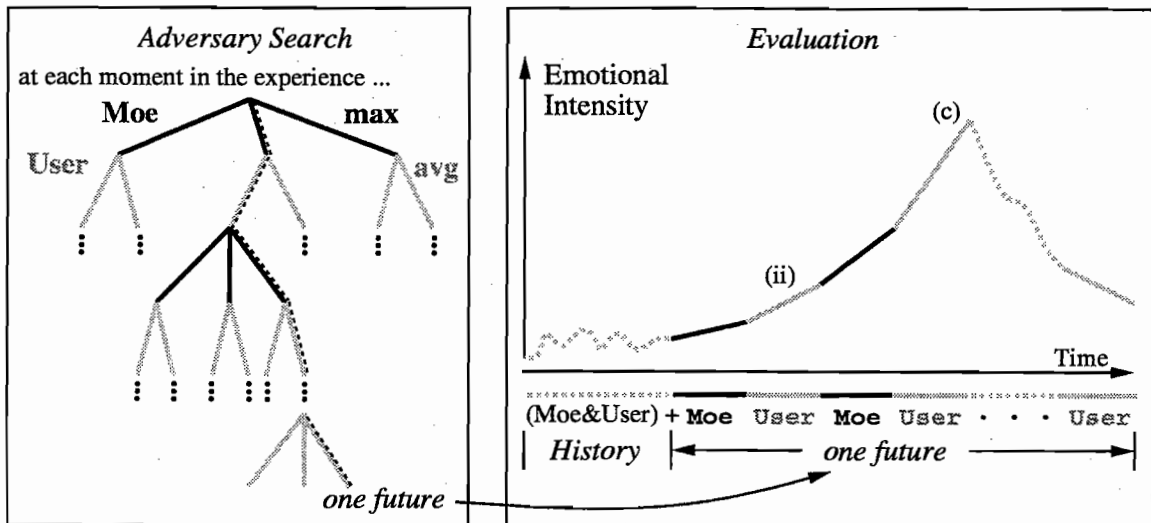


FIGURE 1.4: Moe's Decision Making System: *Adversary Search* with *Evaluation*

that it has no means of deciding which node of the plot graph should go next, but instead pursues all nodes on the frontier simultaneously. In the paper, we concluded:

We find the simple plot graph story model somewhat useful for defining interactive stories, but suggest that extensions are necessary.

In a sense, Moe embodies these necessary extensions to the Plot Graph Model.

1.5 The Moe Architecture for Dramatic Guidance

In an interactive drama, the simulated world and characters provide both great freedom and a powerful short-term experience. The art form also dictates a long-term structure called a destiny. Without it, this form would be like a movie where you say "The characters were good, but the plot was awful."

As I've stated above, the purpose of all the technology implemented for this thesis is to allow the artist to specify a destiny in such a way as to let the system guide the User's experience at a later time, so that she may fulfill her destiny.

Moe is the name of the architecture that has been designed to take this specification and do the guidance. Moe's purpose is to ensure that the User fulfills the artist-given destiny during the interactive drama, while simultaneously allowing the User to remain free. This section provides an overview of how Moe can achieve its purpose.

Figure 1.4 shows the two main components of Moe's decision-making procedure: adversary search and evaluation.² Let's look at how these work together:

²These two components of Moe have been fully implemented. As we shall learn, others have not.

My model is that Moe has a little bag of tricks (called MOE MOVES) that it can use to guide the experience at any moment. For example, a MOE MOVE could bring a new character into the scene, suddenly give a character a strong emotion, or cause a character to drop dead. These MOE MOVES are the way that Moe can guide the User's experience toward the destiny envisioned by the artist.

At any given time in the experience, the User has seen and done many things. Depending on that exact experience of the User, it might be appropriate for Moe to provide dramatic guidance. Thus, Moe's problem is to decide which MOE MOVE to make (if any) at a given time. To do this, Moe uses adversary search. The box on the left of Figure 1.4 shows the kind of adversary search Moe uses to decide which MOE MOVE to make.

As the diagram shows, this looks much like a game-tree search where Moe is one player, and the User is the other. At the top level max node, the drama manager can make a number of MOE MOVES. These are indicated in black. At the next level, for each MOE MOVE the drama system makes, the User can make a number of responses, each of which represents some significant action that the User takes. These significant actions are called USER MOVES. These are shown in gray. At the next level of the search, Moe makes its response to each USER MOVE. This alternating process proceeds till the search reaches the end of the experience.

Each path through the tree corresponds to a possible future of the experience. The path indicated by the dotted line is one particular future, notated *one future*. The evaluation function (as explained next) assigns a value to each of the possible futures, and the search mechanism backs up the values finally to the root, in order to choose the MOE MOVE that maximizes the expected value of the experience.

The box on the right of Figure 1.4 shows the process of evaluation. What has happened so far in the experience is represented by the MOE and USER MOVES in *History*. What is projected by the search are the MOVES in *one future*. The evaluation function works by examining each MOVE in the complete experience (*History* plus *one future*), according to a variety of dramatic criteria. A sample criterion, *Emotional Intensity*, is shown in the diagram. The emotional intensity is rated by plotting the shape of the experience over time and comparing it to an ideal shape.

The experience shown is close to ideal. It starts off slowly, has an inciting incident (ii), which leads to rising action, and finally a climax (c), followed by the resolution to the end. The evaluation function would rate this experience highly. Other experiences will have different shapes and thus will get different ratings. For example, a flat experience would score poorly, as would an experience that climaxed too soon.

There are four main differences between search in this domain and more traditional game-tree search. First, this search uses an average instead of a minimum for backing up at User Nodes. This is because the search doesn't model the User as an adversary who wants to defeat Moe, but as a somewhat random agent. Second, the evaluation function is not static: it operates over the entire history. Third, USER and MOE MOVES don't strictly alternate, since this is not a formal game with rules. For example, Moe could choose to do two things at once, which would mean making two MOE MOVES in a row. Notice that this implies the search diagram is misleading—it shows only alternating MOVES. And fourth,

guidance is
Depends
on
experiences
of
User

diff.
from
game-tree
search

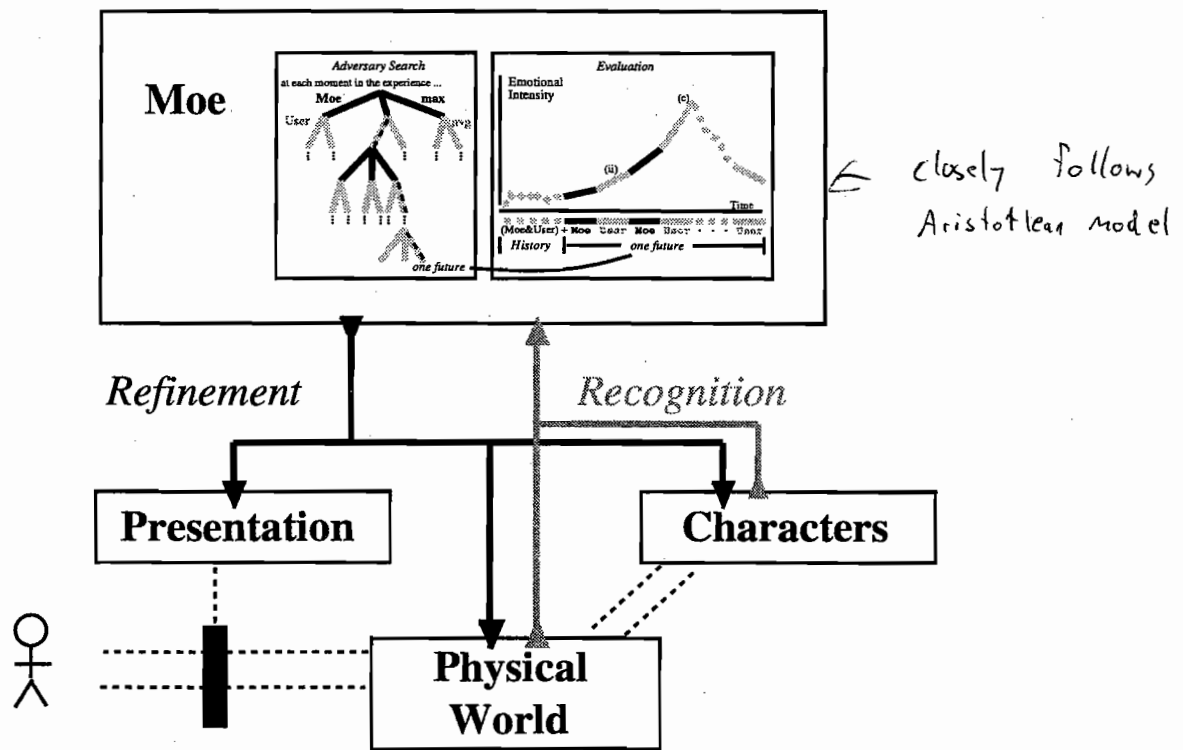


FIGURE 1.5: A Complete Interactive Drama System

the search is over abstract MOVES, which has some effect in the implementation, as you will see later.

Part I of this dissertation describes the work on evaluation in detail, including the actual artistic features, and how they are implemented. Part II describes work on search in detail.

In order for Moe to create the History that forms the basis of the search and evaluation, Moe must be able to “see” what’s happening with the User, the world, and the characters. Likewise, in order to guide the User, Moe must be able to “do” things to all aspects of the system. “Seeing” is the recognition of USER MOVES. “Doing” is the refinement of MOE MOVES. Figure 1.5 shows how Moe connects with the rest of the system through recognition and refinement. omniscience

At the bottom of the diagram is the simulated world with characters. This is where the action happens for the User. As I’ve stated, the User moves around in this world, interacting with characters, places, and objects. The User can fall in love, take walks, open doors, pick locks, pick fights, fly planes, charter boats, use the Force, or take a nap. This is where the experience is situated. This part of the world is created by the artist.

Connected from the simulated world to Moe, are the recognizers. As part of the creation of an interactive drama, an artist must break down the interactive drama into the possible key events, major actions, and significant chunks of story, called USER MOVES. USER MOVES are related to the plot-fragments proposed by Lebowitz[22, 23]. Please see the related work section (Section 1.6) for more details. USER MOVES

Each USER MOVE has its own recognizer, which is a little program, also created by the artist. Some USER MOVES are simple and have simple recognizers. For example, it is easy to make a program that recognizes when the User has found a gun or seen a report. Other USER MOVES will have more complex recognizers. For example, recognizing when the User has had a certain type of conversation with a character is hard in general. As you will see, although I have implemented all the USER MOVES, I have not written the recognizers for each. In Chapter 9 I speculate on what it might take to create recognizers for *Tea For Three*.

In the Moe model, when a USER MOVE is recognized, it is added to the end of the *History* along with the other USER and MOE MOVES.

Over the course of the experience many USER MOVES will be recognized. Thus, at any point during the User's experience, Moe has recognized a sequence of MOVES, which we have called *History*, which is an abstract transcript representing the experience of the User so far. As we have seen, Moe uses *History* to make its decisions.

Also connected to the rest of the system are the refiners. Refiners are little programs written for each MOE MOVE that allow MOE MOVES to make concrete changes to the world. The form of a refiner may vary. For example, the mechanism of a refining program could be a set or sequence of commands, a planner, or some sort of reactive plan executor. Exactly how these refiners are implemented and exactly what they do is specified by the artist.

As the arrows suggest, the concrete commands in the refining programs might affect the characters, the simulated world, or the presentation system. Character commands change the goals, plans, or emotions, or any other part of the character's mind. World commands change any part of the simulated world, including the physical state of objects, how objects work, or what objects exist in the world. Presentation commands change how the world is presented to the User. For example, a command might change the tone of the text or what incidental music is playing. As with recognition, I have not implemented the refiners for the MOE MOVES.

In the Moe model, when a MOE MOVE is refined, it is added to the end of the *History* along with the other USER and MOE MOVES.

As the experience progresses, MOE MOVES are refined, and USER MOVES are recognized. In my model, searches are initiated after any USER MOVE is recognized. The job of the search engine is to choose the MOE MOVE (if any) which maximizes the chance that the User's experience is as close as possible to her destiny, which has been specified by the artist in the evaluation function. The search is done over the abstract MOVES invented by the artist. A legal MOVE generator uses the abstract state of the experience to give the search a set of legal next MOVES. During the search, the evaluation function gives the value of every projected experience, so that comparisons can be made. When the search is over, a MOE MOVE may have been chosen. If so, this MOE MOVE is refined.

By performing this search, Moe has projected a set of possible futures that the experience could take from a certain point, judged them with the evaluation function, and then chosen a MOE MOVE that maximizes the expected value of the experience, which is the measure of how well the experience matches the destiny given by the artist.

recognizers

This section has described the basic model of the Moe Architecture.

However, for this thesis, not all parts of Moe have been implemented. In particular, the simulated world, characters, recognizers, and refiners of *Tea For Three* have not been implemented. However, search and an evaluation function, including all USER and MOE MOVES, have been implemented. The discussion of USER MOVES, the evaluation function, MOE MOVES, and the search form the bulk of this dissertation.

Their implementation was made possible by imagining that *Tea For Three's* simulated world, presentation system, recognizers, and refiners actually have been implemented. As you shall see, in the chapters where the search and the evaluation function are validated (Chapters 4 and 8), appropriate models have been created to overcome the lack of a complete working system.

The work reported in this dissertation takes steps toward a complete interactive drama system. By showing that search and evaluation can work to a useful degree, I provide evidence that the Moe architecture can be used to create drama managers that can guide an interactive drama, and thus that interactive drama is possible.

1.6 Related Work

Entertainers, artists, and researchers have been working in many areas relevant to interactive drama. In this section, I describe some of this other work and relate it to my definition.

I shall consider three broad classes of work: other interactive entertainment, traditional arts, and AI research on story. Some other related work is presented elsewhere. For example, in Section 1.4 I describe related approaches for providing dramatic guidance.

This section does not serve as complete survey of all related areas and techniques. A better summary of work in story understanding and story generation is provided by Ryan[34]. Many other areas of AI, psychology, and the arts can be useful and related to this work. In particular I have not mentioned much character (agent) or presentation work, since work in those areas is related in a complementary sense. Instead, I describe work that I have found particularly useful or relevant to my work on interactive drama.

1.6.1 Other Interactive Entertainment

One of the original forms of computer-based interactive entertainment was the text adventure game. I'm speaking of such games as the original Adventure, Zork, or Infocom's Deadline. In these games, the User takes the role of an adventurer, solving puzzles in order to achieve some larger goals, such as getting more treasure, finding new and cool locations, or finding more information. The User types commands to a command processor which understands limited English input. The User then reads the game's textual output, which is a description of what she sees or hears.

Inspiration for my work has come in part from selected interactive fiction games. In a sense, I view myself as carrying forth the tradition of those early efforts. However,

while they were interactive, even such gems as Infocom's Planetfall and Deadline had flat characters and, to my knowledge, little explicit dramatic knowledge. My work differs in that I am trying to put far greater emphasis on rich characters and the dramatic nature of the experience.

Interestingly enough, some of these early works did encapsulate dramatic principles in their design. For example in the murder mystery Deadline, it was required of the User to suspect one character falsely in order to find evidence to convict the actual murderer. This is a rough application of Aristotle's principle (see below) that reversals are good things to have in your tragedy. The best of these old games are useful springboards from which to produce ideas about interactive drama. In fact, *Tea For Three*, the interactive drama that I have (partially) implemented, is based on Deadline. But what is the difference?

In adventure games, solving puzzles provides much of the excitement. Typically, the User must solve a series of puzzles to finish the game. If the puzzles are of increasing difficulty or interest, the adventure games provides the User with a building sense of tension or excitement. Often, such as with the game Deadline or MYST, solving a puzzle will release information relating to a larger story. In this case, as more puzzles are solved, more information is gained, and the User can feel to a degree as if she is participating in a traditional story. This works well, often very well, as you would know if you've ever lost a friend for a few days to some addictive new game.

In my work, I want the User to feel fully that she is participating in a traditional story. For me, adventure games fall short for two reasons: first, puzzles take the place of characters, and second, the story is neither primary nor truly interactive. In my dream world of the future, the bread-and-butter of interactive drama is emotional, charged, and fascinating interactions with characters. Instead of the story being revealed coincidentally with puzzles that have nothing to do with the story, the story *is* the series of meetings between the User and the characters. As with good traditional stories there is both conflict and resolution, but the conflict is with other characters or perhaps with the inherent nature of the Universe (good vs. evil, e.g.), not with how to get past a particular door or open a certain box. In my dream world, the User's actions dictate the reactions of the characters and the result of the story. There is no one right answer, no one predetermined result.

Ultimately, the experience of the User should be a collaboration between the wishes of the artist, as it has been encoded into the computer, and the wishes of the User, as they are expressed in the actions she takes in the world.

Another way one might approach interactive drama is through some sort of hypertext system. This could mean choose-your-own adventure books or interactive movies with explicit choice points, or text and maybe pictures linked together by hyperlinks.

An advantage this medium has over a computer generated medium is that between each choice point the presentation of material is linear (i.e., non-interactive). Therefore, each segment could be as rich and rewarding as the finest traditional literature or art. The question for this medium is how to make the User feel like a participant in the story. Literally, the User dictates the outcome of the story, but how is this more interactive than simply choosing one book off the shelf versus another?

I think hypertext can make the User feel like a participant if the author pays attention to the relationship between the choices of the User and the content of the unfolding story. To the extent that stories in this medium will approach the type of experience I mean by interactive drama, I think the system must keep information about the User, and that both links and content will change in response to User choices. I am interested in this approach, and I think it could be a powerful medium for creating a certain kind of experience, but I have chosen to use a different model, since I think it is better suited to do precisely what I want to do.

Video games are another type of interactive entertainment. Playing a video game can be a very powerful experience, since the player can use her imagination to immerse herself completely into the action.³ Somehow a player can achieve an almost meditative state where no outside stimulation can make it into the player's brain. Playing the game becomes the player's whole world.

The relationship between adventure games and interactive drama is similar to the relationship between video games and interactive drama. I would like to take the powerful sense of immersion provided by video games and use it in interactive drama. Once again, the difference lies in the characters and story. The characters in a video game are to be killed and the story revolves around how long the player can avoid being killed.

The question is how to take that amazing capability for the player to identify with her little ship and do the same thing in a story setting filled with three-dimensional characters. Games like LucasArt's Dark Forces try to combine adventure games with video games, pushing the envelope of these two media. Unfortunately, the game still lacks a truly interactive story and meaningful character interactions.

Another interaction medium is the Multi-User Dungeon (MUD). MUDs can be quite fascinating. One could look at a MUD as a kind of chaotic adventure game with many human participants. To me as a participant, the most interesting part of a MUD is talking to the other people in the MUD.

A MUD could be like my version of interactive drama if it always promised a coherent interactive story. However, MUD's usually don't have a coherent story like an adventure game. It's more like real life. This is possibly precisely because each participant is an independent human looking to achieve her own goals. If one wanted to impose a story, one could try to control the world and situation enough to make an interactive drama, but since all the characters of the story are independent the amount of control the artist will have is minimal.

Although a MUD has appeal, I rejected trying to force MUD's to become interactive dramas. My model uses computer controlled characters in a simulated world. This is so the system can be in control of all the circumstances of the User's experience. Unlike a MUD, then, my model of interactive drama has only one human User, the rest of the characters being in the service of the interactive story through the control of the computer.

³Males are probably more commonly the ones who become immersed.

MUD's

1.6.2 Artistic Sources

There is an immense, varied body of theory and practice in drama and narrative that extends back for millennia. In general, I have found it difficult to apply this work to interactive drama directly. This is for two reasons. First, when this work makes claims (e.g., “show, don’t tell”), the underlying reasoning is often unclear. These claims can only be fully understood through personal experience using them. Without this, it is difficult to know how to create explicit judgement rules that can be interpreted by a computer. Second, even if these claims were crystal clear, it would be hard to know exactly how or whether to apply them, since all the claims are about non-interactive forms.

In spite of the difficulties, examining this work is very important for the creation of explicit artistic models. In the future, new conventions and rules will surely be created for interactive drama. But for now, one must adapt the known traditional models, in order to begin.

The philosopher Aristotle is one early theorist whose ideas have influenced almost all dramatic criticism. In his *Poetics*[1], Aristotle outlines his ideas about drama. For example, he defines the nature, kinds, and parts of tragic plots. He discusses how tragedy can best achieve its function through its six parts: plot, character, reason, diction, music, and spectacle. These and related concepts, such as the three unities of plot and the notions of reversals and recognitions, can help form a vocabulary for discussing interactive drama and thus provide possible building blocks for an automated system. I have found reading the *Poetics* very helpful for clarifying in my mind which aspects of a drama are relevant and must be represented. It is also a good source of ideas about what makes a drama work well.

Similarly, Freytag gives us his Triangle[14], which is a graphical representation of a play’s plot. This provides insight into how we might represent and evaluate aspects of a destiny.

In her dissertation[21], Laurel presents the ideas of both of these theorists and gives some suggestions on how they might be applied to interactive drama. Laurel’s suggestive analysis of Freytag’s Triangle is clearly my inspiration for creating the *Intensity* feature, which you will learn about in Chapter 3, where I describe the evaluation function in detail.

Not surprisingly, many conflicting theories of drama and literature exist, and it is useful to understand them in order to imagine varieties of interactive drama. As an example, Brecht advocates that some plays exhibit his “alienation effect,” which means the audience should be prevented from emotionally identifying with the characters and thus be unable to experience catharsis [10]. Since some artists of interactive drama may prefer the approach of Brecht (or of yet other theorists) over that of Aristotle (who advocates catharsis), an architecture such as Moe should ideally be capable of implementing any interactive drama, regardless of which theory of drama the artist uses.

Further, an artist may prefer a narrative, rather than dramatic, style. Narrative style of writing is exemplified by novels: description, internal thoughts, and digressions are acceptable. This is opposed to dramatic structure, found in plays, where action must tightly follow action. Propp’s *Morphology of the Folktale*[31] provides an explicit grammar for

representing Russian fairy tales. Studying his type of representation might suggest ways to represent interactive fairy tales as well as other interactive narrative forms.

I have created the evaluation function for *Tea For Three* based on my own aesthetic sense, which has been influenced by a great number of sources. I have drawn from dramatic theories (Aristotle[1], Freytag[14]), narrative theories (Propp[31], Polti[30]), assorted exotic critical theory (Thomas and Johnston[40], McCloud[27]), screenplay writing tips, and my personal experience watching TV, movies, and plays, reading books and comics, and thinking about my reactions. PhD theses by Laurel[21] and Sloane[38] have been useful for helping me understand these ideas in the context of interactive art. All of these have provided advice (much of it contradictory) on how to analyze stories, and what makes them good.

After mulling over these different points of view and my own, I came to the following conclusion: the place to start is to model the process by which the User comes to know and have a reaction to her interactive story-world, both emotionally and intellectually. The transcript of this process forms the basis of the evaluation function.

As you will see, the influences mentioned above often have specific analogs in the evaluation function. I am sure that many aspects of these different theories have become second nature to me, so that I don't even recognize the connections any more. In a way, you can see the process of creating the evaluation function as a task in specifying precisely the knowledge that has helped formed my aesthetic.

1.6.3 Story work in AI

The area of artificial intelligence that is most closely related to ID is work about story. In particular, story generation and story understanding are two areas that I looked at for ideas about representation and techniques.

The work in story generation that interests me most is story generation in the context of a simulation.

One example of such a system is Meehan's Talespin[29]. Talespin works by using a particular physical setting populated by a number of characters that have goals. The story generated is the description of the characters trying to achieve their goals. During generation, if the system needs more information it questions the human about the character's goals, states, and relationships. By carefully coordinating the answers to these questions that arise in the generating process, the human can cause Talespin to generate stories like simple Aesop fables.

Talespin is interesting because it has a similar architecture to an interactive drama system: somewhat interesting characters in a simulated world. There are two main differences between a system like Talespin and my work. First, there is no User character, so the stories generated are not interactive. Second, there is no long-term dramatic guidance that helps generate the stories.⁴ However, Talespin is an exploration of the theory that stories

⁴I suppose one could see the whole Talespin program as a kind of interactive entertainment where the User is the author and thus drama manager, but this is a different type of experience than interactive drama.

are based on the problem solving process of characters with goals. We see aspects of that theory in the evaluation function.

In their work on a system called TAILOR[39], Smith and Witten build on the ideas in Talespin. Like Talespin, they use the problem solving process of characters to tell stories. TAILOR differs from Talespin in many ways, including the fact that TAILOR accepts no human input. The one difference that interests me most is that TAILOR (in one of its forms) has a protagonist and an explicit antagonist that is trying to thwart the antagonist.

The story telling process is based on describing the back and forth actions taken by the protagonist and the antagonist. The protagonist has the goal that drives the story, while the antagonist is trying to foil that goal. To decide what to do, the protagonist character uses an adversary search, where the antagonist is the adversary.

By putting an adversary in the story, the authors are adding conflict to the world, which would presumably tend to create better stories. However, like Talespin, TAILOR also lacks long-term dramatic guidance. Without guidance these stories are kind of like descriptions of chess games. The problem is the program cannot distinguish a dramatically good game (evenly matched, or even a come-from-behind victory) from a dramatically bad one (a blowout).

Another system is Lebowitz's soap opera generator, called UNIVERSE [22, 23]. The purpose of UNIVERSE is to generate an endless narrative, in outline form, about a cast of characters in a soap opera. The generation is done through a planning mechanism that uses plan-like plot fragments (with subgoals that might include other plot fragments) which come from the soap opera genre. By continuously churning the lives of the characters, UNIVERSE can produce a serial format story, with no distinct beginning or ending, just a continuous sequence of dramatically interrelated episodes.

An important difference between Talespin and UNIVERSE is that plot fragments are expressing authorial goals, not character goals. The characters (with complex traits and relationships) maintain the consistency of the plot. The plot fragments direct the lives of the characters in more interesting authorial directions, but do not break the character's consistency. Thus, the plot fragments are a way of generating more interesting stories that would be naturally created by an unguided simulation, such as Talespin.

This work is interesting for several reasons. First, the selection of plot fragments⁵, including the subgoals used to achieve them, is interesting because it shows what Lebowitz thought were the relevant high-level story moments in a soap-opera. I used the plot fragments as a source of inspiration for USER MOVES, but USER MOVES have ended up representing smaller chunks of interaction, perhaps closer to the subgoals contained in plot fragments. Second, the exact construction of the plot-fragments forms an implicit aesthetic for soap operas. This aesthetic is not directly applicable to my work, however, since interactive drama is about dramatic climax and resolution, instead of a never ending story. In the future, Lebowitz's methods for defining personalities and relationships, and how they relate to plot fragments, could provide ideas about the implementation of MOE MOVES, in terms of how characters interact with the User to bring about USER MOVES.

⁵Obtained through private communication.

Again, the main difference between work in story generation and my own is that the stories generated by these systems are not interactive. Both address elements of drama in conflicts and character, but story generation does not address the possibility of a human User. Work in story generation often provides interesting examples upon which one might draw, but ultimately the proposed models cannot be used directly since they represent dramatic principles implicitly rather than explicitly.

Story understanding is another area that seems important to the creation of an evaluation function. I describe two pieces of work here that are more relevant than others, but then explain why this area is less relevant than one might think.

The first relevant piece of work is Lehnert's theory of plot units [24]. She proposes a system to summarize narrative stories by first breaking them into simple and more complex plot units, which are organizations based on positive, negative, and mental (neutral) affect states, and then using these plot units to generate the summary. I find this work interesting for two reasons. First, like Lebowitz, her plot units serve as examples of structures for analyzing stories, and in particular how complex story actions relate and overlap with each other. Second, her speculations on recognizing plot units might be useful for creating recognizers for USER MOVES.

Also interesting is Dyer's work on *In-depth Understanding* [13]. He has created a system which parses and understands stories, and then answers questions about those stories. Of most interest to me are his set of Thematic Abstraction Units, which are complex structures of narrative events, similar to Lehnert's plot units. Thus, in the same way, they are related to USER MOVES or possibly whole evaluation functions.

There are three main reasons story understanding work is less related to my goals than would seem likely.

First, most deal with understanding text, so a large portion of their effort is taken up by the language of stories. A drama system doesn't need any knowledge of natural language, since it is interpreting the meaning of events. In my model, all natural language knowledge goes in the characters.⁶

Second, the usual goal of story understanding is to understand what happens in the story. The goal of the evaluation function is to rate the quality of the experience. These goals are related, but different. One possible way to rate a story is to first understand it. However, an evaluation function does not necessarily have to understand the story in order to rate it.

Finally, story understanding researchers seem to favor generality, so story understanding programs are designed to read stories in general. This is different from this work in interactive drama, where the evaluation function rates a variety of experiences with respect to exactly one aesthetic and exactly one, narrow, physical world and set of characters. What this means is that an evaluation function needs to use only the dramatic structures that are related to the aesthetic and specific world, not a general framework of all structures. Further, I would propose that many artists would prefer specific (possibly idiosyncratic)

⁶This is a bit strong. In the future of interactive drama I wouldn't want to rule out the drama system making judgements about the language used either by the User or the characters. For example, directing the characters to use language similar to the User might be an artistic goal.

structures in their own evaluation functions, as opposed to the more general structures used in understanding.

1.7 Road Map To Dissertation

In this chapter we have seen an architecture for a complete interactive drama system. I have posed a component called the drama manager (Moe) that will observe the actions and experience of the User and try to subtly guide her experience.

Part I of the dissertation describes the work in building an evaluation function for the drama manager. I show how one interactive drama, *Tea For Three*, has been broken down into abstract pieces, the USER MOVES, which represent possibly significant moments in the experience. I show how these MOVES have been annotated with relevant information, and how the evaluation functions uses this information to rate (aesthetically) a scenario, which is a sequence of MOVES that represents the experience of the User. Finally, I give statistical evidence suggesting that to a useful degree the aesthetic of *Tea For Three's* artist has been encoded into the evaluation function.

Part II of the dissertation describes the work on adversary search, which is how Moe provides dramatic guidance using the evaluation function. First I describe MOE MOVES, the representation of how the User's experience can be guided at a single point, including the complete set of MOE MOVES for *Tea for Three*. Second, I define a search state that can represent the current situation of any experience, including what USER MOVES have happened, what MOE MOVES have happened, and their effect on the future experience. We then see how a variety of search strategies can use the search state to project future possibilities of the experience in order to decide whether to provide dramatic guidance now. In particular, we describe Sampling Adversary Search, which is a very shallow search that uses a sampling evaluation function, and Memoized Future Contribution Search, which is a modified and then memoized full-depth search, which works based on certain properties of *Tea For Three's* evaluation function. Finally, I give empirical evidence that shows the success of the search strategies, using abstract models of the User's experience.

The conclusion includes a summary of the contributions, plus a description of future work in this area.

Part I

The Evaluation Function

Chapter 2

Tea For Three

Before I describe the mechanism of the evaluation in function in detail, I want to explain *Tea For Three*, the interactive drama we will be considering. This description has several parts.

First, I will describe the background and starting point of the User's experience in this murder mystery. This will include a description of *Tea For Three's* "solution."

Second, I will describe the process of breaking down an interactive drama into USER MOVES. The USER MOVES define the significant moments of an interactive drama, in this case, *Tea For Three*. Breaking the experience into USER MOVES is done by the artist in order to facilitate the future evaluation and guidance of the User's experience.

Third, I will list and briefly describe the MOE MOVES for *Tea For Three*. The set of MOE MOVES is also created by the artist. The full of MOE MOVES is found in Chapter 5.

Fourth, I will give examples of concrete User experiences that serve to demonstrate the abstract nature of USER MOVES. Fifth, I will describe in detail two abstract transcripts: one from a good experience, one from a bad experience. In Chapter 3 these two scenarios will serve as examples that illustrate concretely the mechanism of the evaluation function.

When you have read this chapter, you should understand that a USER MOVE represents a significant moment of an experience, have a rough understanding of *Tea For Three*, including that its genre is murder mystery, and be familiar with the two example scenarios, because they are used in the next chapter.

2.1 Background

2.1.1 Relationship to "Deadline"

Tea For Three is inspired by Infocom's 1982 adventure game *Deadline*. I chose to recreate and modify an existing interactive drama instead of making a completely new one for two reasons. One, I thought it would take less time. And two, by using an established interactive drama, I wouldn't have to worry about whether this new interactive drama worked as a piece of interactive entertainment.

I picked *Deadline* because I thought it had several desirable properties. First, it had a set of characters that reacted to the User to a degree, and developed as the story developed. Second, it had a plot that seemed to match the type of experience I wanted to give the User. Finally, *Deadline* was in a very familiar and formulaic genre, Murder Mystery. This was desirable for two reasons. One, familiar genres are easily understood by computer scientists, who were my primary audience. And two, I believed a formulaic genre would be easier to analyze, in terms of its structure and what makes it good.

Tea For Three is different from *Deadline* in a number of ways, but similar in others. The plot has been simplified, the number of characters has been reduced, and what those existing characters do has been reduced. This was done for the most part to change the type of experience the User will have from a forty-plus hour marathon (as with *Deadline*), to a streamlined two hour dramatic experience, which is what I intended for *Tea For Three*. However, despite its simplification, most of the flavor of the experience is retained, including the setting, the character names, and the main lines of action. The descriptions of *Tea For Three* throughout this dissertation do not depend on any knowledge of *Deadline*.

2.1.2 The Setup

Tea For Three is a whodunnit where the User plays the role of a police detective. At the beginning of the experience she is presented with the apparent suicide of Marshall Robner and given the task of figuring out if the suicide was genuine, or if it was murder.¹ Ultimately, she will figure out who killed the man, how, and why.

There are three other characters in *Tea For Three*, four if you count the deceased. The dead man is Marshall Robner, a rich industrialist and philanthropist. The first live character is George Robner, Marshall's son. He is a surly young man who constantly argued with his father. His apparent motive is that he had been having trouble with his father over money, and was known to have been threatened with disinheritance.

The second character is Baxter, who was Marshall Robner's business partner. At the beginning, the User does not know exactly how Baxter might fit in, but it has been rumored that their business was not doing well and that Robner and Baxter had been arguing about the future direction of their company.

The third character is Dunbar, Robner's personal secretary and the last person known to have seen Robner alive. She delivered him tea around midnight, and the next day his body was discovered behind locked doors, apparently dead because of a self-administered overdose of Ebullion, his anti-depressant medicine. Robner was supposedly feeling bad about his business and depressed in general.

The User starts her investigation with just these descriptions of the presumed suicide, the set of characters, and their apparent motives. Given this information, she must seek physical clues and talk to the characters in order to figure out what happened. The process of trying to solve this mystery forms the User's experience in *Tea For Three*.

¹This setup deviates from the plot of *Deadline*, as do other aspects of *Tea For Three*. I will not point them all out.

In the next section, I will show how the mystery-solving process has been broken into a set of USER MOVES, each of which represents one significant moment of the process. For example, finding Baxter's true motive or finding how the murder was committed are both significant moments. As we shall also see, a complete experience is represented by a sequence of USER MOVES. In the next chapter we shall see that any complete sequence of USER MOVES, representing one way to solve the mystery, can be judged by the evaluation function for its dramatic goodness.

In order to make it easier to understand the USER MOVES that describe the solving of the mystery, I will first describe the complete solution.

2.1.3 The Solution

The night Marshall Robner died, he was working in his library on the second floor. Dunbar, his personal secretary, made his tea as usual. However, on that night she dissolved ground-up Loblo pills in his tea, so he couldn't detect it. (Loblo was her hypertension medicine.) She then left and went to bed. Robner drank his tea and took his Ebullion, as she knew he would. The combination of Loblo and Ebullion killed him.

To cover things up after Robner was dead, Baxter snuck over to the Robner residence and went into the library through the balcony window, using a ladder from the shed. Baxter replaced the cup contaminated with Loblo with a fresh cup, locked the library door from the inside, and snuck back out. Unfortunately for him, he dropped the contaminated cup while leaving, and it broke on the ground. He scooped up what he thought were all the pieces, took the ladder back to the shed, and then left the residence. Unknown to Baxter, he left muddy footprints on the balcony, scraped the paint on the balcony railing with the ladder, and left rather sizeable holes where the ladder pressed into the muddy ground. He also left several fragments from the contaminated cup, which became lodged in the holes.

Baxter wanted Robner dead because Robner was blackmailing Baxter by threatening to release scandalous information regarding one of Baxter's previous business deals, which involved defrauding the Focus Company. In the weeks before the murder Robner and Baxter had disagreed about whether their company should merge with another. Robner was preventing the merger by blackmailing Baxter. Because of this threat and a large number of previous threats, Baxter decided to kill his partner.

Baxter enlisted the help of Dunbar since she had access to Robner, and she agreed, in part because they were lovers, in part because he threatened to have her fired, and in part because she was an unstable woman. Baxter's alibi is that he went to a concert, alone, on the night of the murder, so he couldn't have committed the murder. In fact, he went with Dunbar, but they lied about it. Unfortunately for them, Dunbar kept her concert ticket with her.

The job of the User as detective is to uncover this chain of events and prove that Baxter and Dunbar conspired to kill Robner. As usual with this genre, the detective must establish means, motive, and opportunity in order to solve the mystery. In the next section we will describe the mystery solving process for each of these three aspects. In particular, we will break this the process into significant moments, which we have called USER MOVES.

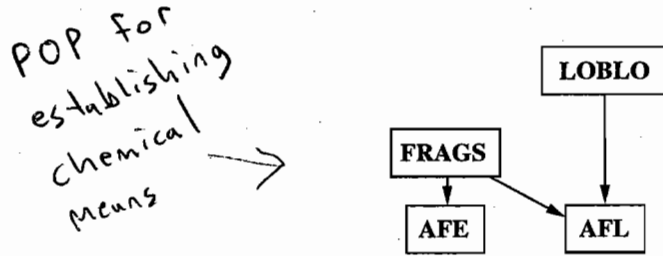


FIGURE 2.1: *Tea For Three* DAG: four USER MOVES

2.2 The USER MOVES

The first part of the solution that we will consider is the *means*: how the murderer caused death and made the murder look like a suicide. I have broken *means* into two parts, which we will examine sequentially. The first part we will consider is what caused the death, which I call the *chemical means*. The second part is how the murder was covered up to look like a suicide, which I call the *means of escape*. Later in the section we shall consider *motive* and *opportunity*.

To establish the *chemical means*, the User must take three actions. First, she has to find the fragments from the contaminated cup left by Baxter. Second, she has to find Dunbar's medicine, Loblo. These two can happen in either order. Finally, she must ask her police labs to analyze the fragments for Loblo. By doing this she will receive a report that tells her death was caused by a fatal combination of Ebullion and Loblo, which was earlier mistaken for an overdose of only Ebullion. This report establishes the *chemical means*.

Before establishing that Loblo and Ebullion killed Robner, the User might have another theory: that Dunbar dissolved ground-up Ebullion pills in Robner's tea, in order to cause the overdose. In this case, she might analyze the fragments for Ebullion instead of Loblo. If she does this, the police labs will return a negative report, stating that no Ebullion was found on the fragment. (However, please see Section 2.3 for how this report might be used by Moe.)

As I have said before, these four significant moments are called USER MOVES. In order to represent the temporal relationships between them, which will be used in legal MOVE generation, the artist places them in a directed acyclic graph (DAG).

Figure 2.1 shows the DAG containing the four USER MOVES, which are listed by their nicknames. Finding the fragment is called FRAGS, analyzing the fragment for Ebullion is called AFE, analyzing the fragment for Loblo is called AFL, and finding the Loblo is called LOBLO.

The meaning of the arrows in the graph is that the USER MOVE at the tail of the arrow must happen before the USER MOVE at the head of the arrow. As shown in in Figure 2.1, the User must have the fragments before analyzing them for either Ebullion or Loblo, and the analysis for Loblo can only come after finding the Loblo. However, there is no necessary ordering between analyzing for Ebullion and finding the Loblo.

The partial order given by the DAG represents only the physical constraints based on logic and physics. Notice that there are five full orders of the USER MOVES consistent with

this partial order:

1. FRAGS, AFE, LOBLO, AFL
2. FRAGS, LOBLO, AFE, AFL
3. FRAGS, LOBLO, AFL, AFE
4. LOBLO, FRAGS, AFE, AFL
5. LOBLO, FRAGS, AFL, AFE

If *Tea For Three* consisted of just those four USER MOVES, the job of the evaluation function would be to take as input any of those five possible total orders and return a number between zero and ten, which would represent the quality of the experience. In fact, the evaluation function works only on completed experiences, but I use this as an illustration now, since we can see the five complete possible orders before us. The number of possible complete orders for the full experience would be quite unmanageable.

Just to give a flavor of what an evaluation function might make of these orders, I will point out what I consider to be a flaw found in orders 3 and 5. Can you see it? In those orders, the mystery of how the murder was committed is discovered in event three. Thus, USER MOVE AFE at the end is irrelevant. To me, this makes a bad moment of the experience.

Recall that each order is an abstract representation of the User's experience in a concrete physical world simulation. Later in this chapter I will give some example computer interactions that demonstrate this.

The next part of the mystery that we will consider is the *means of escape*, which is how the murderer covered up the murder, making it look like a suicide. There also happen to be four USER MOVES that pertain to *means of escape*. As I described before, Baxter crept into the library with the ladder, replaced the cup, locked the library door, and snuck out again. However, he left four clues the User can find, which serve to establish the *means of escape*. The User can find the muddy footprints on the balcony (MUD), find the scraped paint on the outside of the balcony's railing (PAINT), find the holes made by the ladder below the balcony (HOLES), and find the muddy-footed ladder itself (LADDER). Figure 2.2 shows how these USER MOVES are related to the previous USER MOVES and each other.

Notice that there are several independent parts of the DAG. In fact, LADDER is all by itself. In *Tea For Three*, the fragments of the cup are actually hidden inside the holes, so HOLES must come before FRAGS. Also, to see the scraped paint, the User must walk on the balcony. The system will always describe the muddy footprints any time the User is on the balcony. Thus it is impossible to see the scraped paint without first seeing the muddy footprints. Therefore, MUD precedes PAINT.

Again, as above, if this were the whole set of USER MOVES possible in the experience, the evaluation function would take as an argument any of the 1,176 (I won't list them all here) possible total orders consistent with the partial order and return a rating of the order's goodness as a story.

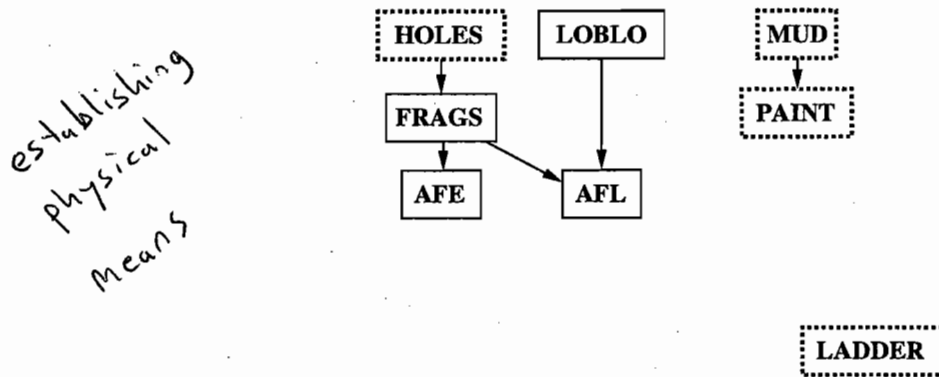


FIGURE 2.2: *Tea For Three* DAG: eight USER MOVES

In addition to *means*, the User must establish who had the *motive* and the *opportunity* to commit the crime. This part of the investigation happens at the same time as the investigation of the *means*. One neat property of *Tea For Three* is that the User could suspect all three characters at the same time, depending on what she knows. Let's consider the *motive* of each of the three characters in turn, including descriptions of *opportunity* where necessary.

As I mentioned earlier, initial police reports showed that his father, Marshall Robner, had threatened George with disinheritance. As the investigation begins, the User should have a possible theory: that George had actually been disinherited, and in order to reinstate his inheritance he killed his father and destroyed the new will. In fact, Marshall Robner had written a new will disinheriting George, but George didn't know this at the time of the murder. The new will is located in a safe in a secret room off the library. In addition, Marshall Robner's date calendar has a notation from a few days before the murder indicating that this new will had been completed, and that Robner needed to contact his lawyer.

There are three USER MOVES that relate to George's motive for killing his father. The first occurs when the User finds the calendar notation (CALENDAR). This confirms George has been disinherited. After finding the notation, the User can show it to George (CONFRONT GEORGE), in order to provoke a reaction. George will shake visibly when the User shows the calendar notation to him, and then he will start to fidget, as if he has something to do. George wants to destroy the new will, so when he has the chance, he will get it from the secret room, and destroy it by tossing it in the lake outside. The third USER MOVE (CATCH GEORGE) occurs when the User catches George with the new will. At that point George will break down and admit he was destroying the will, but still maintain his innocence.

Figure 2.3 shows what the new DAG looks like. As usual, the evaluation function must be able to judge any of the 194,040 total orders consistent with the partial order. Notice that establishing George's *motive* can be interleaved with establishing the *means* of the murder. This is a property of interactive experiences that can make evaluation difficult. The evaluation function must be able to rate both an experience where the *means* is established, then the *motive*, as well as an experience where the USER MOVES of each are interleaved. In Chapter 3 we shall see how it can do just that.

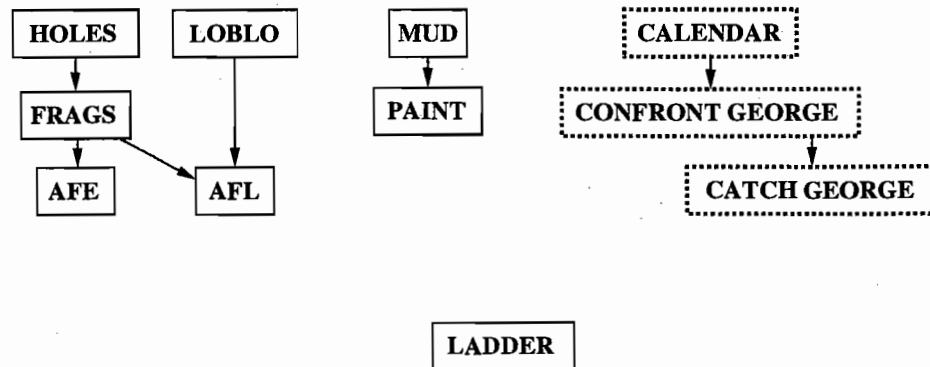


FIGURE 2.3: *Tea For Three* DAG: eleven USER MOVES

The next character to consider is Baxter. At the beginning of the investigation, he has no apparent *opportunity*, because, as I described earlier, he claimed to be at a concert by himself the night of the murder. In a moment I will describe how the User establishes Baxter's *opportunity*.

Meanwhile, the User has to establish Baxter's *motive* as well. As I mentioned before, Baxter wanted Robner dead because of Robner's long-term blackmail. At the beginning of the investigation, the User knows only that Baxter and Robner were business partners, and that they were disagreeing about whether their company should merge with another company. There are three USER MOVES which concern Baxter's *motive*. Relative to each other, these three USER MOVES can happen in any order.

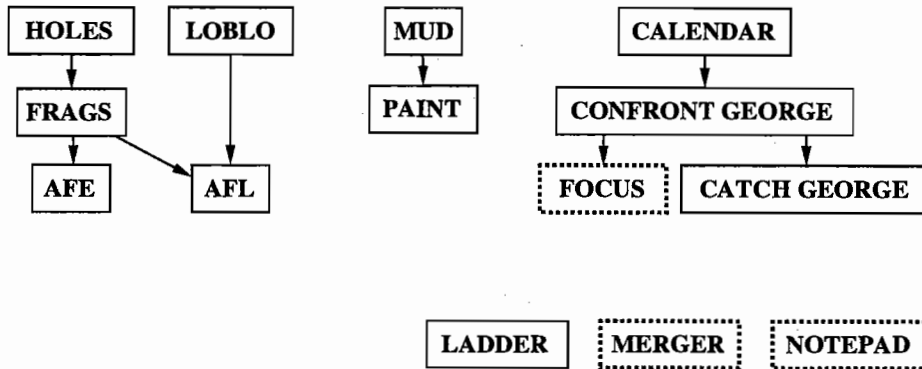
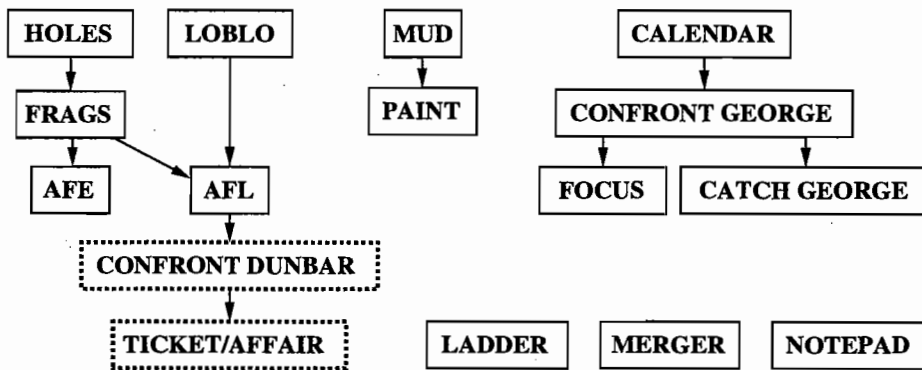
The first USER MOVE (MERGER) is a discussion Baxter has with the User. In the conversation, Baxter claims that, just before Robner's death, Robner and Baxter were in full agreement about their company's plans to merge with another company. However, Baxter's delivery is insincere and suspicious. In reality, Robner never agreed with the merger.

The second USER MOVE (NOTEPAD) happens when the User discovers some indentations on a notepad (found in the library) that appear to be made from pressing a pen through from the page above. The indentations are fragmentary, but it appears that the previous sheet of paper contained a note from Robner to Baxter warning Baxter to back off on the merger, or else Robner would be forced to bring Baxter's scandal out into the open.

The third USER MOVE (FOCUS) happens when the User investigates the safe (in the secret room) that George opens to get the new will. The safe contains not only the new will, but also all the documentation pertaining to the Focus Scandal, which Marshall was using to blackmail Baxter. Finding these papers provides the precise physical evidence showing Baxter's *motive*.

Figure 2.4 shows the DAG with the three new USER MOVES. As you can see, MERGER and NOTEPAD can happen at any time, while FOCUS is constrained to occur only after CONFRONT GEORGE. This partial order is consistent with over 211 million total orders.

The User must also consider Dunbar's possible *motive*. At the beginning of the investigation, she has no apparent *motive*. As I explained earlier, her true *motive* is based on her

FIGURE 2.4: *Tea For Three* DAG: fourteen USER MOVESFIGURE 2.5: *Tea For Three* DAG: all USER MOVES

love for Baxter. There are two USER MOVES in the User's process of discovering this.

The first USER MOVE is CONFRONT DUNBAR, where the User confronts Dunbar with the report indicating that Loblo, Dunbar's medicine, was used to kill Robner. Dunbar becomes visibly upset by this news.

The second USER MOVE is TICKET/AFFAIR, where the User intimidates Dunbar, which causes her to get nervous. Dunbar then decides to smoke, pulls out her cigarettes, and accidentally drops her concert ticket from the night of the murder. When the User sees this ticket, she breaks down completely, admitting she was with Baxter at the concert, she returned with him that night, she was his lover, and that she and Baxter committed the murder together. After this USER MOVE, the User knows that Baxter and Dunbar committed the murder together. This also destroys Baxter's alibi, showing he had *opportunity* to cover up the crime.

Notice that this is not necessarily the last USER MOVE of the story, since the User may not have established Baxter's *motive*, for example, or fully investigated George.

Figure 2.5 shows *Tea For Three's* complete DAG of all sixteen USER MOVES. This partial order is consistent with over 2.24 billion total orders. These possible total orders represent the universe of possible User experiences, when described by USER MOVES. The evaluation function must return a rating for any of these over 2.24 billion total orders.

While the descriptions of the USER MOVES are still fresh in your mind, I will describe the MOE MOVES that can be used by Moe to guide the User's experience. In Chapter 5 I will describe MOE MOVES in more detail, but for now we'll examine how they relate to the USER MOVES.

2.3 The MOE MOVES, briefly

There are eighteen different MOE MOVES that the system can use to guide the experience of the User. This section describes each briefly with respect to the USER MOVES. Each description gives the name of the MOE MOVE in **boldface**, along with a description of the changes the MOE MOVE makes to the world or characters, and finally how it relates to the USER MOVES. Again, these MOE MOVES cause changes to the way the world, characters, or presentation normally work. So, when the description says "Dunbar walks to the User..." it means that the MOE MOVE changes Dunbar's goals so that she will take that action in the world.

- 1. Remove Dunbar** Remove Dunbar from the world simulation. This prevents any USER MOVE involving Dunbar from happening: CONFRONT DUNBAR and TICKET/AFFAIR.
- 2. Dunbar Confronts User** If Dunbar is out of world, bring her in, somewhere away from the User. Dunbar then walks to the User, and forces the User to confront her with the Loblo Report. To do this, Dunbar may break the User's suspension of belief. Please see Chapter 5 for a more in-depth discussion of this issue, which, as you will see, arises for a number of MOE MOVES. MOE MOVE 2 causes USER MOVE CONFRONT DUNBAR.
- 3. Dunbar Confesses to User** Dunbar walks to the User and performs the whole sequence of dropping the ticket, breaking down, and confessing to the murder. Again, this may be unbelievable. This causes USER MOVE TICKET/AFFAIR.
- 4. Baxter Explains Merger** Baxter walks to the User and has the conversation where he explains that Robner and he agreed about their company's impending merger. May be unbelievable. This causes USER MOVE MERGER.
- 5. George Confronts User** Similar to MOE MOVE 2. George walks to the User, and forces the User to confront him with the Calendar. May be unbelievable. This causes USER MOVE CONFRONT GEORGE.
- 6. George is Caught** George walks to the User with the will in hand, breaks down, and confesses. Again, may be unbelievable. This causes USER MOVE CATCH GEORGE.
- 7. Delay the AFL Report** Normally, when the User requests the police labs to analyze the fragments for Loblo, the police perform the analysis and immediately return the report. For MOE MOVE 7, the report is not returned, but instead delayed. This delays USER MOVE AFL. See MOE MOVE 8 below.

- 8: Return the AFL Report** The police return the report to the User, causing USER MOVE AFL.
- 9: Combine AFE and AFL** When the police analyze the fragments for Loblo, they also do an analysis for Ebullion, and put both analyses in the report returned to the User. This combines USER MOVE AFL with USER MOVE AFE, making them happen at the same time.
- 10: Describe Holes and Fragments Together** When the holes are described, mention that there are some kind of ceramic fragments in the hole as well. This combines USER MOVE HOLES with USER MOVE FRAGS.
- 11: George Shoots Skeet** George walks over to lake, starts the skeet machine, and makes a lot of noise with his shotgun. If the User is in or near the library, the noise should attract the attention of the User, causing her to step out on the balcony, and thus see the muddy footprints. This MOE MOVE tries to cause USER MOVE MUD.
- 12: George goes on Balcony** George walks to balcony, makes eye contact with the User, and scurries back inside. This only works if the User is outside. The idea is to get the User thinking about the balcony, and hopefully get her to return to the scene of the crime and eventually see the muddy footprints. Thus, this MOE MOVE also tries to cause USER MOVE MUD.
- 13: George Confesses Secret Room** While confessing to taking the will, George mentions that he found the will in a safe in a secret room off the library. The idea is to get the User to go to the secret room and thus to find the Focus scandal papers in the safe. This MOE MOVE tries to cause USER MOVE FOCUS.
- 14: Describe the Mud and the Scraped Paint Together** When the system describes the muddy footprints, it also describes the paint that has been scraped off the railing. This combines USER MOVE MUD with USER MOVE PAINT.
- 15: Create Ladder Tracks** Put some track marks leading off to the shed from underneath the balcony. The idea is to lure the User to the shed to discover the ladder. Tries to cause USER MOVE LADDER.
- 16: Fragments Move to Shed** Remove the fragments from the holes and put them in the shed next to the ladder. The fragments would be in the shed there if Baxter had broken the cup there, instead of under the balcony. This combines USER MOVE LADDER with USER MOVE FRAGS.
- 17: George Doesn't take Will** Even though George knows about the new will, he won't take it yet. This prevents USER MOVE CATCH GEORGE.
- 18: AFE report mentions mystery substance** In the police report saying that there was no Ebullion found on the ceramic fragment, include a paragraph stating that the police did find some other substance, perhaps another medicine, which they were unable to

identify. The idea is to cause the User to search for this mystery substance, possibly searching medicine cabinets. In particular, this MOE MOVE is trying to get the User to find the Loblo in Dunbar's medicine cabinet, thus causing USER MOVE LOBLO.

2.4 USER MOVES are Abstractions

In Section 2.2, I described the abstract USER MOVES for *Tea For Three*. In this section, I will give three examples of concrete experiences that include the USER MOVE MUD. This is for two reasons.

First, I want to demonstrate that the USER MOVES are abstract, and thus show that the number of concrete instantiations of each is unlimited. We should note that as a consequence, a total order of USER MOVES is also representing an unlimited number of possible concrete User experiences. In the next section I will give two total orders of USER MOVES, one from a bad experience, one from a good, and for each describe a possible concrete experience that it could represent.

Second, two of the three examples come from the text-based version of *Tea For Three*. This is interesting because it shows what it would be like to interact in *Tea For Three*.

Here is the first example fragment of a concrete interactive session. This is the User interacting in the text-based version of *Tea For Three*, as I imagine it. At the USER> prompt, the User types her command. The output of the system follows after a blank line.

Concrete transcript #1:

USER> look

You are in the library. To the north is the balcony,
to the south is the hallway.

USER> go on the balcony

You step out the doorway onto the balcony. You see
muddy footprints on the surface of the balcony. There
is a balcony railing, and you can see a lake in the distance.

USER> examine the footprints

There is nothing special about the footprints.

USER> jump off the balcony

You cannot jump off the balcony.

USER> go south

You are now back in the library.

During this interactive session recorded in transcript #1, the system would have recognized the USER MOVE MUD happening, as soon as the footprints were seen. However, the scraped paint on the railing was not seen (this would have been USER MOVE PAINT).

This second transcript is almost the same, except that the User has to unlock the door, and that instead of trying to jump off the balcony, she examines the railing.

Concrete transcript #2:

USER> look

You are in the library. To the north is the balcony, to the south is the hallway.

USER> go on the balcony

The door is locked.

USER> unlock the balcony door with the red key

The door is now unlocked.

USER> go on the balcony

You step out the doorway onto the balcony. You see muddy footprints on the surface of the balcony. There is a balcony railing, and you can see a lake in the distance.

USER> examine the footprints

There is nothing special about the footprints.

USER> examine the railing

You notice that on the outside surface of the railing, the paint has been scraped off.

USER> go south

You are now back in the library.

As in transcript #1, the system recognizes USER MOVE MUD as soon as the User sees the footprints. How the User comes onto the balcony doesn't matter; it is irrelevant to the abstract transcript whether the door was locked or not. Notice that in transcript #2, USER MOVE PAINT has also occurred.

The important idea to understand is that a single USER MOVE is an abstraction of the many concrete User actions and observations occupying the time surrounding that USER MOVE. These details, such as whether the door was locked or not, are irrelevant to the overall experience, and therefore are not included in the abstract transcript. Any event that is important must have an associated USER MOVE.

This is a fairly large abstraction, as potentially hundreds or even thousands of concrete actions will be abstracted into a sequence of sixteen USER MOVES. This abstraction is necessary both for simplifying the evaluation function and for making the search tractable. Of course, one must remember that the User doesn't know about these abstractions, and experiences everything in the concrete simulation.

Here is yet a third transcript. It is a transcript of a full-sensory experience as might happen in Star Trek's Holodeck. I will describe it in the first person, from the point of view of the User.

Concrete transcript #3:

I see a glass door, partially obscured by an off-white curtain. I part the curtain with my hand and with my other reach out and take hold of the door's handle. It is iron, cool to the touch. I firmly turn the handle until I feel the door come free. I push open the door and feel a rush of fresh air from outdoors. I step outside. There is a strange crunching noise coming from my feet. Looking down, I notice that the floor of the balcony is covered with muddy footprints. Curious to see what lays below, I lean over the railing and take a look down at the garden. A small detail catches my eye: there is paint scraped off the outside of the railing. I look out over the lake, but quickly look away. The sun is too bright, and there appears to be nothing more to see. I walk back through the door, and close it, carefully pulling the curtain out of the door's way.

What transcript #3 suggests is a degree of medium independence. In this session, USER MOVES MUD and PAINT both happened, but the medium was not text. It was some sort of futuristic Holodeck. The point is that although *Tea For Three* has been designed for text-based interaction, *Tea For Three* may be well suited for other media. In Chapter 9, I discuss this issue in more detail.

2.5 Two Abstract Scenarios

This section contains descriptions of two abstract transcripts, which will be used to illustrate the process of evaluation. An abstract transcript is called a *scenario*. I have handcrafted these scenarios so that one of them will represent a "good" experience, and one will represent a "bad" experience. In Chapter 3, we will see how the evaluation function rates each. By using a good and a bad example, we can better see how an experience might be successful or unsuccessful.

The purpose of this description is to let you see what I think might be happening in the head of the User as each USER MOVE happens. As I describe the good and bad experiences, you should get an idea for what sort of aesthetic I have tried to encode into the evaluation function. I should remind you that I am the artist who (inspired by Deadline) created *Tea For Three*, and therefore this is my specific aesthetic. Since this is not a universal aesthetic, you may not agree with all or any aspects of it.

In any case, here are the two experiences. The USER MOVE is indicated in **boldface**, with the description following each. The first experience is the good one, in my opinion.

Good Experience:

NOTEPAD The User goes to the library, searches the desk, and finds the notepad. She uses her pencil to scratch the pad, finding the cryptic notation indicating that Robner was blackmailing Baxter. She thinks to herself that maybe Baxter had a motive to kill Robner, although it is not clearly a murder, yet.

CALENDAR She looks around the desk a little more, finding the calendar. She flips through the calendar, catching the notation of a new will. Perhaps George has been disinherited, she thinks. She now feels as if she could talk to either George or Baxter about these things, or possibly look around a bit more.

MUD The User decides look around, stepping out on the balcony. She notices the muddy footprints on the floor of the balcony.

PAINT Next she checks out the railing and finds the scraped paint on the outside. At this point she is getting an idea that perhaps this was no suicide.

MERGER She decides to confront Baxter with her questions, but Baxter provides a plausible, if suspicious, explanation. This eases her suspicions away from Baxter, but only slightly.

CONFRONT GEORGE Next she goes to George and confronts him with the notation indicating a new will had been written by his father. George is visibly shaken, and heads off.

FOCUS Somehow, without catching George, the User heads back to the library and finds the open safe with the Focus scandal papers. She peruses them and realizes that Marshall had been blackmailing Baxter the whole time. She thinks this makes an excellent motive.

CATCH GEORGE Next, she finds George and catches him getting rid of the will. I think in a way that this ordering of these two events is perhaps a flaw in this story, since she would have to basically run to catch George and she probably wouldn't have a motive to do that at this point in the story. In any case, she now has established motives for both George and Baxter.

LOBLO So, she goes to investigate the third character trying to find some kind of link, perhaps, since she still doesn't know how Robner was killed. Looking around in Dunbar's bedroom, the User finds the Loblo. The bottle is immediately suspicious, but it is not clear to her how it fits in with everything else. She saves it for later, and heads outside to follow up on the lead given by the muddy footprints and the scraped paint.

LADDER Outside is a shed. The User looks around in the shed and finds a ladder, perhaps a method for getting up to the balcony, she thinks.

HOLES Excited, she heads under the balcony to see if there is any evidence below. Indeed, she finds the holes shaped just like the ladder feet. Together with the muddy footprints, the scraped paint, and the ladder, this provides a technique for getting away with the murder, but she still needs to confirm cause of death, and figure out how Baxter or George did it.

FRAGS Curious, she digs around in the dirt a little and comes up with the fragments. This is exciting since it seems to indicate some party switched the cups. Perhaps somebody put extra Ebullion in Robner's cup and forced the overdose.

AFE She sends the fragment to be analyzed for Ebullion, but her hunch only partially pays off. There is no Ebullion. But, there is some unidentified substance.

AFL Something in the User's mind clicks, and she decides to check the fragments out for Loblo. In a dramatic turn, she learns from the police report that the fragments were covered with Loblo, and that a lethal combination of Loblo and Ebullion killed Robner.

CONFRONT DUNBAR Sensing the kill, she takes the report to Dunbar, confronting her with (what she thinks might be) her crime. Hopefully she can get Dunbar's motive somehow, she thinks. Dunbar is shaken.

TICKET/AFFAIR The User continues to pressure Dunbar. Dunbar drops a ticket stub as she is taking out her cigarettes. The User scoops it up and suddenly it all falls into place. Dunbar breaks down, admitting to being with Baxter that night, admitting to being his lover, and admitting they both did this evil deed. The User is pleased as punch at her intelligence and spunk.

For several reasons, I think this was a good experience for the User. First, it seems that from the point of view of the User, events in the experience flow in a logical order. This means that there are no coincidences (which are bad), and most events happen in response to the actions of the User. Second, the User seems to be following a clear course of action. She has goals and she seems to be following them. Finally, the User seems to be experiencing the joy of unravelling this mystery in a way that makes sense. She doesn't get any redundant information, so every USER MOVE adds to the progression of the experience.

Let me emphasize that my interpretations are guesses. One might think that if we can't *know* exactly what is going on with the User, we are lost. However, if we examine other forms of entertainment, we find that we are not lost. The process of creating any work of entertainment involves some amount of guesswork. For example, the director of a film uses his or her judgement about how best to tell a story. The director does not know exactly how everybody will react to the finished movie, but he or she has an educated guess that a certain technique will be effective. Sought after directors are those that can maximize the appeal (and profit) of movies, based on those good guesses. The same is true for interactive drama. A characteristic of an effective artist will be the ability to accurately infer the reactions of most Users.

A question may have occurred to you, even if we agree for the moment that the artist's guess is completely accurate. What if the artist's idea of what is pleasing to a User is different from the actual User's idea of what is good? It's a good question. The same question could be applied to any form of entertainment or art. And the answer is the same. The User probably doesn't have the exact same ideas, but will buy into the artist's aesthetic when it is close. If the artist and User are too different, the User probably should move onto another artist. In Chapter 4, I will argue that I have encoded an aesthetic (my own) into the evaluation function. There is a general question as to whether this is worthwhile, if my aesthetic is unique to me and pointless to all others. I shall address this exact question as well.

In order to show you how the evaluation function operates on a bad experience, I will use the following bad experience as the second example. There are worse experiences than this, but an experience that is just gibberish does not clearly illustrate the evaluation function. As with the good experience, I've included my interpretations of some of the concrete details.

Bad Experience:

LOBLO Suspecting Dunbar from the beginning, since she was the last known person to see Robner alive, the User noses around in Dunbar's bedroom, including her medicine cabinet. She finds a suspicious looking medicine, but is not really sure how it might fit in.

HOLES The User then heads out of the house, to behind the house. She hasn't yet found anything relevant at the crime scene or been on the balcony. She does however, find some holes below the balcony.

MERGER She heads back into the house and talks to Baxter about the merger. Baxter explains that Robner did agree with the merger before he died. The User is now suspicious. "The business partner doth protest too much, methinks."

MUD Next, the User heads out onto the balcony, finding muddy footprints on the floor. At this point she has a good idea that somebody left the library by somehow sneaking over the edge of the balcony.

NOTEPAD The User searches the scene, finding the notepad indicating blackmail. Now she is really suspicious of Baxter.

FRAGS She heads back under the balcony to check out the holes again, to see, perhaps, if she missed anything. She finds the ceramic fragment, and an idea clicks – maybe somebody switched the cups.

PAINT The User now heads back to the balcony and checks it out more thoroughly, finding the scraped paint on the outside.

CALENDAR She steps back inside, and searches the library some more. She finds the calendar and flips through it, finding some evidence that perhaps George had a real motive.

LADDER Oddly, she heads outside to the shed and looks inside. She finds the ladder, which she now realizes is how the murderer escaped over the balcony. (The reason I find this odd is that she could have looked in the shed during any of the other four trips she took right past it.)

AFL Her next step is to analyze the fragments for Loblo. Good move, but again, why not before? Anyway, she receives the report. The User now knows how this murder was completed and covered up. She now needs to find out who did it.

CONFRONT DUNBAR She heads for Dunbar, and shoves the report in Dunbar's face. Dunbar cringes, but doesn't break down. The conversation goes nowhere, but the User is still suspicious.

CONFRONT GEORGE The User then then heads for George and puts the calendar in his face. He is visibly shaken, and eventually leaves the scene, looking preoccupied.

AFE In a stunningly odd move, the User decides to analyze the fragments for Ebullion. She gets the report back, indicating no Ebullion. Of course, she already knew how he died.

CATCH GEORGE Back on track, she finds George, and catches him with the will. He breaks down and admits to trying to destroy it since he was being disinherited, but insists that he is innocent.

TICKET/AFFAIR The User then goes back to Dunbar and pressures her again. Dunbar breaks down, after dropping the ticket, and admits everything about her and Baxter. So now the User knows who did it and why.

FOCUS But, there is one final piece of physical evidence to get, so she eventually returns to the scene, enters the secret room, and finds the Focus scandal papers, proving Baxter's motive.

Although this experience is not terrible, I think it has some flaws. The first flaw is that not everything flows logically from one event to the next. Sometimes, it seems as if events pop up without any reason. Second, there are several unnecessary steps in the experience. The most glaring example is when the User analyzes the fragments for Ebullion even though she knows about the Loblo. Finally, the User seems to bounce around a lot, both geographically and mentally. Some may argue that this is not a flaw, but, as I mentioned before, these are my distinct opinions.

It is important to understand these scenarios are not stories in the traditional sense, to be understood or enjoyed vicariously. The scenarios are the abstract representation of what happened to the User as she travelled, acted, perceived, and thought inside a fantasy world. We found[20] that while a User may be engaged during an interactive drama, an outside observer might be bored. So, when thinking about these transcripts, try to imagine yourself actually doing those things in the *Tea For Three* world. It is an unconventional critical technique, perhaps, but a necessary one.

In this chapter, we have learned about USER MOVES, become familiar with *Tea For Three*, and seen two example scenarios, one good and one bad. In the next chapter, I will use those two example scenarios to show how the evaluation function works.

Chapter 3

The Evaluation Function

In Chapter 2, we learned about *Tea For Three*, including the selection of USER MOVES. I also created two example scenarios, one good and one bad.

In this chapter we learn about the evaluation function. First, I give an overview of how the evaluation function works, showing that I chose to implement it as a weighted sum of normalized feature values. Then, by using the two scenarios as examples, I explain each feature's motivation and show how to compute its raw and normalized value. At the end of the chapter, I explain how the feature values are combined in a weighted sum.

This chapter shows *how* the evaluation function works, but it doesn't show whether it works *correctly*. In Chapter 4, I will argue that working correctly means successfully encoding one artist's non-trivial aesthetic. I will then describe a study that suggests that this evaluation function has, to a useful degree, succeeded in encoding the artist's non-trivial aesthetic.

In Section 1.6.2, I stated that I have studied dramatic, literary, and popular theories in order to form my own personal aesthetic as an artist and critic. Since I am the artist who created *Tea For Three*, it is my personal aesthetic that is encoded in this evaluation function. Thus, the specific features, parameters, and techniques used in the evaluation function come from my accumulated knowledge.

One can see the process of creating an evaluation function as knowledge engineering. In this case, I am both expert and engineer. I am engineering my accumulated knowledge about how to rate a User's experience.

Throughout this chapter, I try to explain the motivation behind the various choices I have made. In this way, I hope this chapter can serve as both a complete description of one evaluation function, as well as an informal guide for creating different evaluation functions based on other artists' aesthetics or critical theories.

After you have read this chapter you should understand in detail how the evaluation function works. You should also appreciate the motivations for many of the choices I made in its design.

3.1 Overview

1/0
The input to the evaluation function is a sequence of USER MOVES and MOE MOVES, called a *scenario*. The output of the evaluation function is a real number between 0 and 10, inclusive, which represents the subjective aesthetic quality of the scenario. Zero is the worst rating; ten the best.

Input scenarios are restricted in two ways. First, the total order of the input scenario must be consistent with the partial order given in Figure 2.5. Second, the scenario must contain all sixteen USER MOVES. For most of this chapter we will consider scenarios that contain no MOE MOVES. Chapter 5 explains in detail how evaluation is affected by the MOE MOVES.

A scenario's rating is calculated by first calculating values for each feature of the evaluation function, and then adding the values together in a weighted sum. In Section 3.9, I discuss how these weights are chosen. There are seven features considered by the evaluation function. Most of the rest this chapter will be a description and illustration of each of the features on our two example scenarios. The seven features of *Tea For Three's* evaluation function are:

Thought Flow Measures whether one event in the User's experience relates logically to the next.

Activity Flow Measures how bored the User feels by walking around uselessly.

Options Measures how much freedom the User perceives she has.

Motivation Measures whether the User's actions are motivated by her goals.

Momentum Measures the proximity of certain events that I (as artist) prefer to happen together.

Intensity Measures whether the User's excitement builds while solving the mystery.

Manipulation Measures how manipulated the User feels in response to MOE MOVES.

Before the feature values can be combined in a weighted sum, the evaluation function must compute a value for each of the features. Three steps are required to compute each feature value.

First, the scenario is analyzed, MOVE by MOVE, to determine a measurement relevant to the feature. For example, with *Thought Flow*, the measurement equals the number of USER MOVES that have a particular relationship with their preceding USER MOVE. So, each USER MOVE is checked against the previous USER MOVE, and, if the relationship holds, the *Thought Flow* count is increased by one. After the function considers each USER MOVE the feature has a raw score.

Second, the raw score is judged for goodness. My aesthetic dictates that *Thought Flow* is better when it is larger. For other features, such as *Manipulation*, smaller raw scores are better.

Finally, we normalize the raw score to a scale from 0 to 10. We map the worst possible (lowest for *Thought Flow*) value to zero and the best possible value (highest for *Thought Flow*) to ten, and linearly interpolate intermediate values.

3.1.1 Simplifications

The task of evaluation has been simplified in three ways. First, the input scenario must have all sixteen USER MOVES. Second, this evaluation function is not general – it works only for scenarios from *Tea For Three*. Third, the evaluation function is required to understand only what a typical person might be feeling or thinking while having the experience that is represented by the input scenario. In Chapter 9, I will discuss these simplifications in more detail, speculating on whether and how they might be eased in future work.

The rest of this chapter contains a detailed description of the features of the evaluation function and how they are combined to compute a final rating for a scenario. I will use the two scenarios from Chapter 2 to illustrate how the implemented evaluation function works.

3.2 The Thought Flow Feature

The first feature we will consider is *Thought Flow*. *Thought Flow* measures the consistency of the User's thought process. I think it makes a better experience if the User rides one train of thought for a few USER MOVES in a row, instead of bouncing back and forth between topics. To me, this adds a sense of consistency and allows a sort of logic of action to build up. To compute *Thought Flow*, the function has to know what the possible trains of thought (topics) are, and which USER MOVES are associated with which topics.

In the design of this system I have used simple and specific representations rather than general frameworks. In this case, I decided that there were only five topics in *Tea For Three* that the evaluation function needed to consider. I also decided that there would be no structure relating topics, such as sub-topics or groups of topics.

The five topics are: *find the chemical means*, *find the means of death*, *think about Dunbar being guilty*, *think about Baxter being guilty*, and *think about George being guilty*. To me, these five captured the essential avenues of investigation found in *Tea For Three*. As the artist, I have annotated each of the USER MOVES with the subset of the five topics relevant to that USER MOVE. For example, NOTEPAD has been annotated with the set {*think about Baxter being guilty*}, which, as you can see, has only one element. The annotations for the other USER MOVES will be evident in the examples.

To compute the raw score for *Thought Flow*, each USER MOVE of the scenario is considered in order. The function compares the topic set of each USER MOVE to the topic set of its predecessor. If there is a non-empty intersection, then the raw score for *Thought Flow* is increased by one. Thus, the total score for a scenario is the number of times that two consecutive USER MOVES share a topic of thought. Let's look at how *Thought Flow* is computed on our first example scenario, the good scenario.

annotates
USER MOVES
w/ topics

Thought Flow in the Good Scenario				
Time	USER MOVE	Topic Set	Flow?	Count
1	find the NOTEPAD	<i>baxter</i>		
2	find the CALENDAR notation	<i>george</i>		
3	find the MUDDY footprints	<i>escape</i>		
4	find the scraped PAINT	<i>escape</i>	Yes	1
5	discuss the MERGER with Baxter	<i>baxter</i>		
6	CONFRONT GEORGE with calendar	<i>george</i>		
7	find the FOCUS scandal papers	<i>baxter</i>		
8	CATCH GEORGE with the will	<i>george</i>		
9	find the LOBLO	<i>chemical, dunbar</i>		
10	find the LADDER	<i>escape</i>		
11	find the HOLES	<i>escape</i>	Yes	2
12	find the ceramic cup FRAGMENTS	<i>chemical</i>		
13	Analyze the Frags for Ebullion (AFE)	<i>chemical</i>	Yes	3
14	Analyze the Frags for Loblo (AFL)	<i>chemical, dunbar</i>	Yes	4
15	CONFRONT DUNBAR with the report	<i>dunbar</i>	Yes	5
16	get the TICKET/AFFAIR story	<i>dunbar, baxter</i>	Yes	6
Total Thought Flow				6
Key:		<i>dunbar</i> = think about Dunbar being guilty		
<i>escape</i> = find the means of escape		<i>baxter</i> = think about Baxter being guilty		
<i>chemical</i> = find the chemical means		<i>george</i> = think about George being guilty		

FIGURE 3.1: *Thought Flow* in the Good Scenario

Figure 3.1 shows the computation of *Thought Flow* for the good scenario. The first column lists the time of the USER MOVES. (This is really its order in the sequence of sixteen.) The next column lists the USER MOVE. To the right is the topic set for each USER MOVE. The key of topic names is below the table. The next column shows whether that USER MOVE has *Thought Flow*. That is, whether its set of topics has a non-empty intersection with the set of topics for the previous USER MOVE. The far right column gives the cumulative count for *Thought Flow*. The final row summarizes the raw score of the feature for the scenario. At the bottom is the key to the topics.

We can see that at time fourteen (AFL) the good scenario has *Thought Flow*. This is because the topic set for AFL is $\{chemical\ dunbar\}$, and the topic set for the previous USER MOVE, AFE, is $\{chemical\}$, and the intersection, $\{chemical\ dunbar\} \cap \{chemical\} = \{chemical\}$, is not empty. Similarly, six USER MOVES have *Thought Flow*: PAINT, HOLES, AFE, AFL, CONFRONT DUNBAR, and TICKET/AFFAIR. Since a total of six USER MOVES have *Thought Flow*, the raw score for this feature is six.

As it turns out for *Tea For Three*, the minimum *Thought Flow* is zero while maximum *Thought Flow* is eleven. The weight for this feature is 1/6th of the total rating. (Later in this chapter, we shall discuss how feature minima and maxima and feature weights are

Thought Flow in the Bad Scenario				
Time	USER MOVE	Topic Set	Flow?	Count
1	find the LOBLO	<i>chemical, dunbar</i>		
2	find the HOLES	<i>escape</i>		
3	discuss the MERGER with Baxter	<i>baxter</i>		
4	find the MUDDY footprints	<i>escape</i>		
5	find the NOTEPAD	<i>baxter</i>		
6	find the ceramic cup FRAGMENTS	<i>chemical</i>		
7	find the scraped PAINT	<i>escape</i>		
8	find the CALENDAR notation	<i>george</i>		
9	find the LADDER	<i>escape</i>		
10	Analyze the Frags for Loblo (AFL)	<i>chemical, dunbar</i>		
11	CONFRONT DUNBAR with the report	<i>dunbar</i>	Yes	1
12	CONFRONT GEORGE with calendar	<i>george</i>		
13	Analyze the Frags for Ebullion (AFE)	<i>chemical</i>		
14	CATCH GEORGE with the will	<i>george</i>		
15	get the TICKET/AFFAIR story	<i>dunbar, baxter</i>		
16	find the FOCUS scandal papers	<i>baxter</i>	Yes	2
Total Thought Flow				2

FIGURE 3.2: *Thought Flow* in the Bad Scenario

determined.) By linearly interpolating the raw score between the minimum and maximum raw scores, the evaluation function computes the normalized value for the good scenario:

$$\begin{aligned}
 \text{Normalized Value} &= 10 * \frac{(\text{Raw Score} - \text{min_value})}{(\text{max_value} - \text{min_value})} \\
 &= 10 * \frac{(6 - 0)}{(11 - 0)} \\
 &\approx 5.45
 \end{aligned}$$

On the normalized scale, this scenario is slightly above the half way mark (five out of ten). By itself this is not great, but the value for the scenario is the weighted sum of all the normalized feature values. Let's consider the bad scenario next.

Figure 3.2 shows how the bad scenario gets a raw score of 2 for *Thought Flow*. Normalized, this scenario gets a 1.82. The very low *Thought Flow* count is consistent with my earlier feelings (see Chapter 2) that this scenario represents an experience that jumped around a lot, and didn't follow a logical progression. Only twice in this scenario does the User do something on a topic related to what came directly before.

Now, as I mentioned before, *Thought Flow* may not be important to some artists, and thus this lack of *Thought Flow* wouldn't hurt their opinion of the experience. But, since I

am encoding *my* aesthetic, it is important to have agreement, which we have, between my intuitive feelings and these feature values.

3.3 The Activity Flow Feature

Activity Flow is similar to *Thought Flow*, except that instead of measuring how the User's thoughts are bouncing around from topic to topic, it measures how the User's body is bouncing around from location to location. I think the User's experience is better when she does not run from place to place doing only one thing in each spot, only to return later to do something else. I prefer the User to do all her business in a place at one time. That way, the amount of uninteresting walking around is minimized, and the experience flows more smoothly.

As with *Thought Flow*, each USER MOVE has been annotated with a set of associated activities. There are seven activities and one special wild-card activity. The activities are: *search the scene of the crime*, *search the balcony*, *search outside*, *search Dunbar's bedroom*, *talk to Dunbar*, *talk to George*, and *talk to Baxter*. As with topics of thought, there are no special structures relating these activities. However, there is a special activity called *any*, which means it is consistent with any of the above activities. This activity is reserved for the USER MOVES associated with analyzing, Analyze the Fragments for Ebullion (AFE) and Analyze the Fragments for Loblo (AFL), since they could really happen anywhere at any time. Again, the mapping will be evident in the examples.

As above, the charts (Figures 3.3 and 3.4) go from left to right: the time, the USER MOVE, the activity set, the *Activity Flow* indicator, and finally the cumulative count for *Activity Flow*. The bottom row contains the summary, while the bottom part contains the key to the activities.

At time twelve in the good scenario (Figure 3.3), we can see that FRAGS (find the ceramic cup fragments), makes a good example of the normal type of *Activity Flow*. The activity set for FRAGS is $\{s/out\}$ (s/out = search outside), and the activity set for its predecessor, HOLES, is also $\{s/out\}$. Since the intersection, $\{s/out\}$, is not empty, FRAGS has *Activity Flow*. AFL (Analyze the Frags for Loblo) is an example of the special case for *any*. The activity set for AFL contains *any*, so AFL automatically gets a point for *Activity Flow*.

Notice that CONFRONT DUNBAR does not have *Activity Flow*. This is because *any* matches only previous USER MOVES, not future USER MOVES. One might argue that it should do future matches, or that CONFRONT DUNBAR should be compared against the first earlier USER MOVE not annotated with *any*. I chose to do only adjacent matching for simplicity, and I chose to do only backward matching somewhat arbitrarily. One should notice that using future matching actually makes little difference to the feature value, since the new raw score would almost always be greater by two, and thus the normalized score would be unaffected. (It would be greater by one if AFL and AFE were adjacent, or if either were the last USER MOVE.) This is an example of how a feature doesn't quite capture the whole truth. A more complex mechanism could probably do a slightly better job. Please see Chapter 9 for future work in this area.

Activity Flow in the Good Scenario				
Time	USER MOVE	Activity Set	Flow?	Count
1	find the NOTEPAD	<i>s/scene</i>		
2	find the CALENDAR notation	<i>s/scene</i>	Yes	1
3	find the MUDDY footprints	<i>s/scene s/balc</i>	Yes	2
4	find the scraped PAINT	<i>s/scene s/balc</i>	Yes	3
5	discuss the MERGER with Baxter	<i>tk/bax</i>		
6	CONFRONT GEORGE with calendar	<i>tk/geo</i>		
7	find the FOCUS scandal papers	<i>s/scene</i>		
8	CATCH GEORGE with the will	<i>tk/geo</i>		
9	find the LOBLO	<i>s/dun</i>		
10	find the LADDER	<i>s/out</i>		
11	find the HOLES	<i>s/out</i>	Yes	4
12	find the ceramic cup FRAGMENTS	<i>s/out</i>	Yes	5
13	Analyze the Frags for Ebullion (AFE)	<i>any</i>	Yes	6
14	Analyze the Frags for Loblo (AFL)	<i>any</i>	Yes	7
15	CONFRONT DUNBAR with the report	<i>tk/dun</i>		
16	get the TICKET/AFFAIR story	<i>tk/dun</i>	Yes	8
Total Activity Flow				8
Key:				
<i>s/scene</i> = search the scene of the crime		<i>any</i> = special activity; matches any		
<i>s/balc</i> = search balcony		<i>tk/bax</i> = talk to Baxter		
<i>s/out</i> = search outside		<i>tk/geo</i> = talk to George		
<i>s/dun</i> = search Dunbar's bedroom		<i>tk/dun</i> = talk to Dunbar		

FIGURE 3.3: Activity Flow in the Good Scenario

In *Tea For Three*, Activity Flow ranges from two to nine. The good scenario is one short of the maximum Activity Flow. Therefore, as we can see below, the normalized value for the good scenario is 8.57:

$$\text{Normalized Value} = 10 * \frac{(8 - 2)}{(9 - 2)} = 10 * 6/7 \approx 8.57$$

This high value is consistent with my feeling that this scenario has a certain flow to it.¹ It appears the User is not moving around needlessly.

The weight for this feature is the same as *Thought Flow*, 1/6th of the total rating.

Let's look at the Activity Flow of the bad scenario. We see in Figure 3.4 that the bad scenario scores four out of a possible nine, giving it a normalized score of 2.86. I thought that the User seemed to be moving around too much in this scenario, and that is consistent with the low 2.86 value.

¹Again, what I mean is that I imagine that the User's concrete experience has flow.

Activity Flow in the Bad Scenario				
Time	USER MOVE	Activity Set	Flow?	Count
1	find the LOBLO	<i>s/dun</i>		
2	find the HOLES	<i>s/out</i>		
3	discuss the MERGER with Baxter	<i>tk/bax</i>		
4	find the MUDDY footprints	<i>s/scene s/balc</i>		
5	find the NOTEPAD	<i>s/scene</i>	Yes	1
6	find the ceramic cup FRAGMENTS	<i>s/out</i>		
7	find the scraped PAINT	<i>s/scene s/balc</i>		
8	find the CALENDAR notation	<i>s/scene</i>	Yes	2
9	find the LADDER	<i>s/out</i>		
10	Analyze the Frags for Loblo (AFL)	<i>any</i>	Yes	3
11	CONFRONT DUNBAR with the report	<i>tk/dun</i>		
12	CONFRONT GEORGE with calendar	<i>tk/geo</i>		
13	Analyze the Frags for Ebullion (AFE)	<i>any</i>	Yes	4
14	CATCH GEORGE with the will	<i>tk/geo</i>		
15	get the TICKET/AFFAIR story	<i>tk/dun</i>		
16	find the FOCUS scandal papers	<i>s/scene</i>		
Total Activity Flow				4

FIGURE 3.4: Activity Flow in the Bad Scenario

3.4 The Options Feature

Options measures how much freedom the User feels she has. In *Tea For Three*, I don't want the User to feel as if there is only one right path at any given point. I want her to feel as if there are several significant options and that she is free to pursue any of them. *Options* is a measure of interactivity: Does the User feel as if she has real control over her actions, that there is sufficient freedom to act, and also that those choices have meaning in the experience?

I favor giving the User more choices at the beginning of the experience at the expense of more choices at the end. In fact, I might argue that being constrained at the end is a good thing. Therefore, the number of options at the beginning is more important to this feature's value than the number of options at the end.

In order to calculate the number of significant options available to the User, I have identified twelve important goals (shown in Figure 3.5) that the User might have at any time. Each goal constitutes one possible option for the User to pursue. Obviously there are other possible goals, but I have decided these are the important ones for measuring *Options* in *Tea For Three*.

There are three steps to calculate the raw score for *Options*. First, the function counts how many of the twelve goals the User has for each time in the story; i.e., after each USER

1. **Talk to Dunbar about her Loblo** This goal is valid from the time the User finds the LOBLO till the time she CONFRONTS DUNBAR (LOBLO to CONFRONT DUNBAR).
2. **Talk to George about the new will** This goal is valid from the time the User finds the CALENDAR notation indicating there is a new will till the time she confronts George (CALENDAR to CONFRONT GEORGE).
3. **Talk to Baxter about Blackmail** This goal is valid from the time the User finds the NOTEPAD indicating that Baxter is being blackmailed till the time that the User discusses the MERGER with Baxter (NOTEPAD to MERGER).
4. **Challenge Dunbar** This goal is valid from the time the User CONFRONTS DUNBAR with the report indicating Loblo with Ebullion is the cause of death till the time she finds the TICKET and Dunbar confesses her AFFAIR with Baxter (CONFRONT DUNBAR to TICKET/AFFAIR).
5. **Challenge Baxter** This goal is valid anytime after the User discusses the MERGER with Baxter (after MERGER).
6. **Find the means-of-escape upper** This goal is valid from the the time the User finds either the HOLES or the LADDER to the time she sees the MUDDY footprints or the scraped PAINT (HOLES or LADDER to PAINT or MUD).
7. **Find the means-of-escape lower** This goal is valid from the time the User find the MUDDY footprints or the scraped PAINT to the time she finds the HOLES (MUD or PAINT to HOLES).
8. **Find Clues in General** Always valid.
9. **Find clues about Dunbar** Valid until the User finds the LOBLO (until LOBLO).
10. **Analyze the fragments for Ebullion** Valid from the time the User finds the FRAGMENTS till the time she actually does the Analysis For Ebullion (FRAGS to AFE).
11. **Analyze the Fragments for Loblo** Valid from the time the User has found both the FRAGMENTS and the LOBLO till the time she does the Analysis For Loblo (FRAGS and LOBLO to AFL).
12. **Find mystery substance on the fragments** Valid from the time the User gets the Analysis For Ebullion report to the time she finds the LOBLO (AFE to LOBLO).

FIGURE 3.5: 12 Goals used by *Options*

Active Goals (and sums) in the Good Scenario														
Time	USER MOVE	Whether Numbered Goal is Active											Σ	
		1	2	3	4	5	6	7	8	9	10	11		12
0	at scenario's start								•	•				2
1	NOTEPAD			•					•	•				3
2	CALENDAR		•	•					•	•				4
3	MUD		•	•				•	•	•				5
4	PAINT		•	•				•	•	•				5
5	MERGER		•			•		•	•	•				5
6	CONFRONT GEORGE					•		•	•	•				4
7	FOCUS					•		•	•	•				4
8	CATCH GEORGE					•		•	•	•				4
9	LOBLO	•				•		•	•					4
10	LADDER	•				•		•	•					4
11	HOLES	•				•			•					3
12	FRAGS	•				•			•		•	•		5
13	AFE	•				•			•			•		4
14	AFL	•				•			•					3
15	CONFRONT DUNBAR				•	•			•					3
16	TICKET/AFFAIR					•			•					2

FIGURE 3.6: Tracking Goals for *Options* in the Good Scenario

MOVE has happened. Second, the function weights the number of goals at each time by the inverse of the time. Third, the raw score equals the sum of all the weighted goal counts. This technique achieves the effect of making the early goals more relevant to the overall measure.

Calculating *Options* requires the function to know which of the twelve goals the User has at any given time. To do this, each goal has an associated expression which relates the sequence of USER MOVES so far to whether the User has the goal. By evaluating the expression for each goal, the function can determine the number of goals the User has.

Figure 3.5 contains a description of each of the twelve goals. The description includes a goal number (used in the two examples), a short goal description, and the method for determining when the goal is active.

Figure 3.6 shows which goals, numbered one through twelve, the User has throughout the first scenario. A "•" appears in the column of a goal whenever that goal is present in the scenario after that USER MOVE. The chart also indicates at far right how many goals are active at that time in the scenario. As I've said, both the goals and goal numbers are calculated just after the USER MOVE on that line has happened.

To compute the raw score, the function weights each goal total for each time by the inverse of the time. The function counts time as starting at the beginning before the first event. So the score is the number of goals before the scenario begins divided by one, plus

Active Goals (and sums) in the Bad Scenario														
Time	USER MOVE	Whether Numbered Goal is Active												Σ
		1	2	3	4	5	6	7	8	9	10	11	12	
0	at scenario start								•	•				2
1	LOBLO	•							•					2
2	HOLES	•					•		•					3
3	MERGER	•				•	•		•					4
4	MUD	•				•			•					3
5	NOTEPAD	•				•			•					3
6	FRAGS	•				•			•		•	•		5
7	PAINT	•				•			•		•	•		5
8	CALENDAR	•	•			•			•		•	•		6
9	LADDER	•	•			•			•		•	•		6
10	AFL	•	•			•			•		•			5
11	CONFRONT DUNBAR		•		•	•			•		•			5
12	CONFRONT GEORGE				•	•			•		•			4
13	AFE				•	•			•					3
14	CATCH GEORGE				•	•			•					3
15	TICKET/AFFAIR					•			•					2
16	FOCUS					•			•					2

FIGURE 3.7: Tracking Goals for *Options* in the Bad Scenario

the number of goals after the User finds the NOTEPAD divided by two, plus the number of goals after she finds the CALENDAR divided by three, etc. Here is the full computation of the Raw Score:

$$\begin{aligned}
 \text{Raw Score} &= (2/1) + (3/2) + (4/3) + (5/4) + (5/5) + (5/6) \\
 &\quad + (4/7) + (4/8) + (4/9) + (4/10) + (4/11) + (3/12) \\
 &\quad + (5/13) + (4/14) + (3/15) + (3/16) + (2/17) \\
 &\approx 11.62
 \end{aligned}$$

The score for *Options* ranges from 7.73 to 13.28, so this scenario scores 7.01 on the normalized scale. In Figure 3.6, we can see that the number of goals builds up right away, from two to three to four to five. This is a good start, since immediately the User feels as if she has a number of different avenues that she can explore. This is one reason why I find this scenario good. In the User's mind, the options of what to do increase early.

Notice this is different than a choose-your-own-adventure, where options are explicit. Here, the options are significant courses of action from the point of view of the User.

Like *Activity Flow* and *Thought Flow*, the weight of *Options* is 1/6th of the total rating.

Figure 3.7 shows the goals being tracked in the bad scenario. *Options* is calculated as above, weighing the number of goals at each time by the inverse of the time:

$$\begin{aligned}
 \text{Raw Score} &= (2/1) + (2/2) + (3/3) + (4/4) + (3/5) + (3/6) \\
 &\quad + (5/7) + (5/8) + (6/9) + (6/10) + (5/11) + (5/12) \\
 &\quad + (4/13) + (3/14) + (3/15) + (2/16) + (2/17) \\
 &\approx 10.54
 \end{aligned}$$

The bad scenario scores 5.07 on the normalized, ten-point scale. This scenario doesn't have as many options at the beginning, so the score is lower. However, this scenario does have a rather thick part (in terms of the number of goals) toward the end of the middle. This is good, since it means that the User perceives freedom then, but, as I've said, freedom is more important in the beginning.

3.5 The *Motivation* Feature

Motivation measures whether the User has motivation for performing her actions in the world. I often think of a successful experience as one in which the User has a series of goals, which she alternately acquires and satisfies. Each piece of the scenario is like a little story itself, with a beginning, where the goal is acquired; a middle, where the goal is pursued; and an end, where the goal is satisfied. However, the small story will not make sense if the goal is satisfied before the User even knows she has the goal. So, *Motivation* lets the evaluation function measure whether the goal is present when the goal-satisfying USER MOVE takes place.

There are actually two types of agency represented by this feature. The first measures whether a goal is present in the User's mind when it is satisfied. The second measures whether any goal is being pursued. This second aspect of the feature is a bit more subtle.

Basically, I feel it is better to have something happen in pursuit of an unrelated goal than it is for something to happen while in pursuit of no goal. For an extreme example, suppose the fragments were not buried in the holes. As an artist, I would like it better if the User found the fragments in Dunbar's bedroom than if she found them on the front lawn. This is because in the first case the User is looking for clues about Dunbar, and in the second case the User has just wandered around and found the fragments by luck. There is reason (in my mind) to the first case and none in the second.

Calculating this feature requires tracking the goals of the User, tracking the activities of the User, and knowing the relationship between the two. The features *Activity Flow* and *Options* already provide the evaluation function the ability to track activities and goals.

The relationship between goals and activities is similar to the traditional relationship between a goal and a plan for that goal. An activity is associated with a goal if the User could logically engage in that activity in order to pursue the goal.

Figure 3.8 shows the mapping from the twelve goals to their associated activities. First given is the number of the goal, then the name of the goal. Following the name of the goal is the set of activities associated with that goal. Recall that the abbreviations for the

No.	Goal Description	Activity Set
1	Talk to Dunbar about her Loblo	<i>tk/dun</i>
2	Talk to George about the new Will	<i>tk/geo</i>
3	Talk to Baxter about Blackmail	<i>tk/bax</i>
4	Challenge Dunbar	<i>tk/dun</i>
5	Challenge Baxter	<i>tk/bax</i>
6	Find the means-of-escape upper	<i>s/scene s/balc</i>
7	Find the means-of-escape lower	<i>s/out</i>
8	Find Clues in General	<i>s/scene</i>
9	Find clues about Dunbar	<i>s/dun</i>
10	Analyze the fragments for Ebullion	
11	Analyze the fragments for Loblo	
12	Find mystery substance on the fragments	<i>s/scene s/dun</i>

FIGURE 3.8: Table of Activities used to pursue Goals

activities are found in Figure 3.3. Notice some goals give rise to more than one activity and some give rise to no activities.

Because the evaluation function considers both forms of *Motivation*, the calculation of the raw score is fairly straightforward. The basic idea is this: the feature scores a point if the activity associated with the current USER MOVE is one of the activities indicated by the User's previous goals. If that is true, then the User is participating in an activity motivated by her goals, and the raw score of *Motivation* is increased by one.

Mechanically the process is as follows: after each USER MOVE, the function figures out which goals the User had just before that USER MOVE. For each goal she had, the function determines the activity set for that goal. The activity sets from each of her previous goals are unioned. This union forms the complete set of activities that the User could have been using to pursue her complete set of previous goals. The complete set of activities is compared to the set of activities associated with the current USER MOVE. If the intersection is non-empty, the feature scores a point.

Figure 3.9 shows how this works in practice on the good scenario. The first column from the left shows the time.

The next two columns show the User's previous mental state. The first column gives the goals (by number) the User had just before the current USER MOVE. The second, labeled **Motivated Activities**, gives the complete set of activities that can be used to achieve any of the User's previous goals. (The goal numbers and activity abbreviations in this chart are the same ones used in Figure 3.8 and Figure 3.3.)

There is a subtlety here. Because the User always has Goal 8: *Find Clues in General*, the activity *s/scene* is always motivated. For this reason, *s/scene* is not considered in the calculation of *Motivation*.

The next two columns show the current USER MOVE and its associated **Activity Set**.

Motivation in the Good Scenario						
t	Prev. Goals →	Motivated Activities	USER MOVE →	Activity Set	Motivation?	Count
1	8 9	<i>s/dun</i>	NOTEPAD	<i>s/scene</i>		
2	3 8 9	<i>s/dun tk/bax</i>	CALENDAR	<i>s/scene</i>		
3	2 3 8 9	<i>s/dun tk/geo tk/bax</i>	MUD	<i>s/scene s/balc</i>		
4	2 3 7 8 9	<i>s/out s/dun tk/geo tk/bax</i>	PAINT	<i>s/scene s/balc</i>		
5	2 3 7 8 9	<i>s/out s/dun tk/geo tk/bax</i>	MERGER	<i>tk/bax</i>	Yes	1
6	2 5 7 8 9	<i>s/out s/dun tk/geo tk/bax</i>	CONFRONT GEORGE	<i>tk/geo</i>	Yes	2
7	5 7 8 9	<i>s/out s/dun tk/bax</i>	FOCUS	<i>s/scene</i>		
8	5 7 8 9	<i>s/out s/dun tk/bax</i>	CATCH GEORGE	<i>tk/geo</i>		
9	5 7 8 9	<i>s/out s/dun tk/bax</i>	LOBLO	<i>s/dun</i>	Yes	3
10	1 5 7 8	<i>s/out tk/dun tk/bax</i>	LADDER	<i>s/out</i>	Yes	4
11	1 5 7 8	<i>s/out tk/dun tk/bax</i>	HOLES	<i>s/out</i>	Yes	5
12	1 5 8	<i>tk/dun tk/bax</i>	FRAGS	<i>s/out</i>		
13	1 5 8 10 11	<i>tk/dun tk/bax</i>	AFE			
14	1 5 8 11	<i>tk/dun tk/bax</i>	AFL			
15	1 5 8	<i>tk/dun tk/bax</i>	CONFRONT DUNBAR	<i>tk/dun</i>	Yes	6
16	4 5 8	<i>tk/dun tk/bax</i>	TICKET/AFFAIR	<i>tk/dun</i>	Yes	7
Total Motivation						7

FIGURE 3.9: Motivation in the Good Scenario

If the intersection of **Motivated Activities** (column three) and **Activity Set** (column five) is non-empty, then the next column indicates this USER MOVE has *Motivation*. The final column is the cumulative count of *Motivation*. The total *Motivation* is found in the final row.

As an example, consider the row at $t = 9$. Just before USER MOVE LOBLO, the User has Goals 5, 7, 8, and 9. As we see in Figure 3.8, Goal 5 motivates $\{tk/bax\}$, Goal 7 motivates $\{s/out\}$, Goal 8 motivates $\{s/scene\}$, and Goal 9 motivates $\{s/dun\}$. The union of these activity sets is the complete set of motivated activities: $\{tk/bax, s/out, s/scene, s/dun\}$. The function removes $s/scene$, as described above, and the result $\{tk/bax, s/out, s/dun\}$ is shown in the column labeled **Motivated Activities** (although in a different order).

The next two columns show that the USER MOVE is LOBLO, and that LOBLO is associated with the **Activity Set** $\{s/dun\}$. Since $\{tk/bax, s/out, s/dun\} \cap \{s/dun\}$ is non-empty, LOBLO has *Motivation*. Thus, a “Yes” appears in column six, and the count is increased by one.

As we can see from the last row, the good scenario has a raw score of seven for the *Motivation* feature. *Motivation* ranges from five to seven in *Tea For Three*, so this is a perfect ten on the normalized ten-point scale. This is consistent with my feeling that this scenario flowed logically from one event to the other. The weight of this feature is 1/12th of the total score.

The variation in the raw score for *Motivation* is not large. The score ranges from five

Motivation in the Bad Scenario						
t	Prev. Goals →	Motivated Activities	USER MOVE →	Activity Set	Motivation?	Count
1	8 9	<i>s/dun</i>	LOBLO	<i>s/dun</i>	Yes	1
2	1 8	<i>tk/dun</i>	HOLES	<i>s/out</i>		
3	1 6 8	<i>tk/dun s/balc</i>	MERGER	<i>tk/bax</i>		
4	1 5 6 8	<i>tk/dun tk/bax s/balc</i>	MUD	<i>s/scene s/balc</i>	Yes	2
5	1 5 8	<i>tk/dun tk/bax</i>	NOTEPAD	<i>s/scene</i>		
6	1 5 8	<i>tk/dun tk/bax</i>	FRAGS	<i>s/out</i>		
7	1 5 8 10 11	<i>tk/dun tk/bax</i>	PAINT	<i>s/scene s/balc</i>		
8	1 5 8 10 11	<i>tk/dun tk/bax</i>	CALENDAR	<i>s/scene</i>		
9	1 2 5 8 10 11	<i>tk/dun tk/geo tk/bax</i>	LADDER	<i>s/out</i>		
10	1 2 5 8 10 11	<i>tk/dun tk/geo tk/bax</i>	AFL			
11	1 2 5 8 10	<i>tk/dun tk/geo tk/bax</i>	CONFRONT DUNBAR	<i>tk/dun</i>	Yes	3
12	2 4 5 8 10	<i>tk/dun tk/geo tk/bax</i>	CONFRONT GEORGE	<i>tk/geo</i>	Yes	4
13	4 5 8 10	<i>tk/dun tk/bax</i>	AFE			
14	4 5 8	<i>tk/dun tk/bax</i>	CATCH GEORGE	<i>tk/geo</i>		
15	4 5 8	<i>tk/dun tk/bax</i>	TICKET/AFFAIR	<i>tk/dun</i>	Yes	5
16	5 8	<i>tk/bax</i>	FOCUS	<i>s/scene</i>		
Total Motivation						5

FIGURE 3.10: Motivation in the Bad Scenario

to seven. This means that *Motivation* will not by itself distinguish a good scenario from a bad scenario. So, for *Tea For Three*, I feel that *Motivation* is a flavoring on top of the basic, more important ingredients for a good experience. In the future work section of Chapter 9, I will discuss how various features may apply to other classes of interactive drama.

We can see in Figure 3.10, the bad scenario has a raw score of five for *Motivation*. The normalized value is zero, so this scenario scores the minimum possible. As I mentioned above, *Motivation* may not be useful for distinguishing good from bad, but I find it comforting that *Motivation* is consistent with my feelings about these two scenarios.

Motivation measures whether the User is trying to accomplish one of her goals. This is related to the *Options* feature. I want the User not only to have options (as measured by *Options*), but also to be pursuing one of them (as measured by *Motivation*). *Motivation* measures whether the User is experiencing a cycle of goal acquisition and goal satisfaction. As I mentioned above, I believe this feeling is important to a good experience.

3.6 The Momentum Feature

While creating the evaluation function for *Tea For Three*, I found that the features I had created did not convey everything that I wanted to convey about a good scenario. I felt there were subtle reasons why some pairs of USER MOVES worked well when one followed the

Momentum in the Good Scenario			
Time	USER MOVE	Momentum?	Count
1	find the NOTEPAD		
2	find the CALENDAR notation		
3	find the MUDDY footprints		
4	find the scraped PAINT	Yes	1
5	discuss the MERGER with Baxter		
6	CONFRONT GEORGE with calendar		
7	find the FOCUS scandal papers		
8	CATCH GEORGE with the will		
9	find the LOBLO		
10	find the LADDER		
11	find the HOLES		
12	find the ceramic cup FRAGMENTS	Yes	2
13	Analyze the Frags for Ebullion (AFE)		
14	Analyze the Frags for Loblo (AFL)		
15	CONFRONT DUNBAR with the report		
16	get the TICKET/AFFAIR story	Yes	3
Total Momentum			3

FIGURE 3.11: *Momentum* in the Good Scenario

other immediately. Instead of coming up with general mechanisms to explain this subtlety, I thought it would be easier to create a feature that would allow me to associate any two USER MOVES, without specifying the reason. This feature is called *Momentum* because it scores when the second USER MOVE of the pair immediately follows the first.

Because it does not consider context, *Momentum* is possibly risky for an artist. *Momentum* will score for a pair of MOVES regardless of what has happened in the past. For this reason, only three pairs of USER MOVES have *Momentum* in *Tea For Three*.

The three pairs of USER MOVES are: HOLES then FRAGS, MUD then PAINT, and CONFRONT DUNBAR then TICKET/AFFAIR.

“HOLES then FRAGS” is there because of a flaw in *Tea For Three*. In my opinion there is no good time to find the FRAGS, except right after the HOLES. *Activity Flow*, *Thought Flow*, and *Motivation* represent this to a degree, but not as strongly as I feel about it. Similarly, there is a natural flow between MUD and PAINT, and between CONFRONT DUNBAR and TICKET/AFFAIR. These pairs are not required, and it isn’t even that the experience seems more tidy with them, but that *without* them, the experience seems flawed.

Figure 3.11 shows how *Momentum* is calculated on the good scenario. As you can see, all three ordered pairs of USER MOVES are found in this scenario, so it’s raw score is 3, the maximum. *Momentum* ranges from 0 to 3; on the normalized scale this scenario has value 10. The weight of *Momentum* is 1/12th of the total rating.

Momentum in the Bad Scenario			
Time	USER MOVE	Momentum?	Count
1	find the LOBLO		
2	find the HOLES		
3	discuss the MERGER with Baxter		
4	find the MUDDY footprints		
5	find the NOTEPAD		
6	find the ceramic cup FRAGments		
7	find the scraped PAINT		
8	find the CALENDAR notation		
9	find the LADDER		
10	Analyze the FRags for Loblo (AFL)		
11	CONFRONT DUNBAR with the report		
12	CONFRONT GEORGE with calendar		
13	Analyze the FRags for Ebullion (AFE)		
14	CATCH GEORGE with the will		
15	get the TICKET/AFFAIR story		
16	find the FOCUS scandal papers		
Total Momentum			0

FIGURE 3.12: *Momentum* in the Bad Scenario

Figure 3.12 shows *Momentum* for the bad scenario. The chart is quite empty. In this scenario, PAINT followed FRAGS, TICKET/AFFAIR followed CATCH GEORGE, and FRAGS followed CONFRONT GEORGE. Thus, the raw score and normalized value are both 0. This evaluation is consistent with my intuition that this scenario was in many ways awkward. In contrast, the good scenario felt smooth to me, consistent with its maximum score.

Let's next move to the most complex and interesting feature of the evaluation function, called *Intensity*. If you're the caffeinating type, this might be a good time to get another cup of coffee.

3.7 The *Intensity* Feature

When I'm reading a good book, I turn the page because I want to know what's going to happen next. If I'm reading a great book, I'll stay up all night, just to see how it ends. I want the User of one of my interactive dramas to have the same feeling. Instead of spoon-feeding goals to the User, the artist should engage the User's imagination, so that it becomes the User's goal to proceed. I want the User to be pursuing her own selfish, exciting, and engaging goals. Her curiosity and expectations should lead her onward, deeper into the experience. If we (as artists) can hook a User, we can start to explore the full potential of this medium.

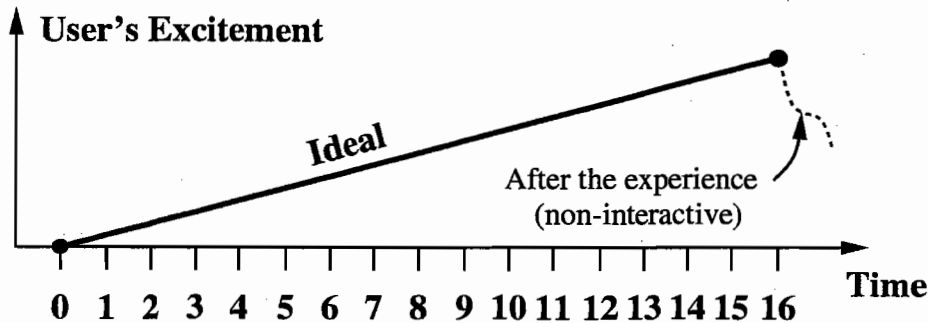


FIGURE 3.13: Ideal Graph of User's Excitement

Intensity is trying to measure the User's excitement. Specifically, *Intensity* measures how or whether the excitement of the User's experience builds over time, finally ending in a dramatic climax. I prefer an experience to draw the User in immediately and then keep building tension and excitement until the final conflict, when the experience is resolved one way or another. Sometimes the tension is cyclical, with ups and downs, but overall it should keep building. A big drop before the end can be disastrous. Once the experience builds to the point where things are really rolling, it must lead to a powerful climax.

If I plotted the excitement of a User's experience over time, I prefer the graph to look like a ramp. Figure 3.13 shows what such an ideal experience looks like. In my mind, there is a resolution after the climax of the interactive drama, but it happens during the non-interactive thoughts of the User, after the experience.

The idea behind *Intensity* is to match the graph of the User's excitement against this ideal graph. There are two parts to this process. The first is to create the graph for a particular User's experience, and the second is to match it against the ideal. Section 3.7.1 describes how to create the excitement graph from a scenario. Section 3.7.2 describes how to do the match against the ideal graph. *Intensity* measures how closely the User's excitement graph matches the ideal graph.

3.7.1 Creating the User's Excitement Graph

In *Tea For Three*, the excitement comes from solving the mystery. Therefore, the evaluation function needs to model both what the User knows, and what she suspects. These give rise to the surprises and reversals that produce the User's strongest emotions. This is similar to the complication measure identified by Laurel[21].

For *Tea For Three*, I have identified nine important facts that can be used to track the User's knowledge. For example, *Baxter has a Motive*, *LOBLO is the Cause of Death*, and *Dunbar is Guilty*. Naturally there are many other facts that the User might consider, but in the same way that I restricted the goals the evaluation function would consider, I have restricted the facts to these very important nine facts. Later in this section, I describe these facts in detail.

As I've said, in my model excitement comes from solving the mystery. Practically, this means gaining knowledge about the mystery. For *Tea For Three*, knowledge is measured by the nine facts. The User can either know or suspect each fact is true or false. If the User's knowledge of any of these facts change, she should be excited. For example, when the User verifies that LOBLO *is the cause of death*, by receiving the report, she should be quite excited. Naturally, verifying some facts will be more exciting than others. This shall be reflected (as described later) by annotating a goal with its importance.

excitement =
change in
knowledge

Each USER MOVE in the scenario can change the knowledge of the User. How it changes the User's knowledge depends on what she has already experienced, and thus already knows. Later in this section, I will explain exactly how the User's knowledge of each fact is calculated at any time in the scenario.

By examining the sequence of USER MOVES, the function tracks the knowledge of the User over time. As I've said, the changes in the User's knowledge give rise to her excitement. Thus, an excitement graph plots the excitement of the User over time. The excitement at each point in time is given by the change in knowledge at that time. Mathematically, the excitement graph is related to the derivative of the knowledge graph.

The question is: how does a change in the User's knowledge of a fact specifically relate to excitement. The answer is that it depends on how important the fact is, and exactly how the User's knowledge changed. More important facts cause more excitement. More dramatic changes in knowledge cause more excitement.

I have created a knowledge system (described in detail later in this section) that allows more than just an absolute knowledge of true or false. For example, a User can have no knowledge of a fact, have a hunch the fact is true, think a fact is probably true, or know a fact is true. There are also corresponding representations for when the User thinks a fact is false. Specific changes from one level of confidence to another lead to specific excitement levels. We will see that a reversal of the User's knowledge, which is changing from knowing a fact is false to knowing it is true, (or vice versa), causes the most excitement.

The User's total excitement at any time is the sum of the individual amounts of excitement she derives from learning about (changing knowledge levels) each of the nine facts. The excitement graph is simply the plot over time of the User's excitement.

defn

Let's look at a small, made-up example to see how this works.

Suppose there were only one fact: *The Butler did it*. At the beginning, let's imagine the User knows nothing about this fact. Further suppose that there are three USER MOVES in the made-up scenario. Let's say that after the first USER MOVE the User suspects *The Butler did it*. Let's say during second USER MOVE the User learns nothing new, but after the third USER MOVE she knows *The Butler did it*. Perhaps the butler confesses during the third USER MOVE.

Figure 3.14 shows how the function can track the excitement derived from the fact *The Butler did it* during the made-up scenario. The first two columns give the time and sequence of USER MOVES in this made-up scenario.

The third column shows the User's knowledge of the fact *The Butler did it* after each USER MOVE. For example, after the User find the Butler's letter opener (FIND BUTLER'S

Description of User's Knowledge	Abbrev.	Numerical Representation
"I know for a fact it's true"	K	7
"I believe it's true"	S3	3
"It's probably true"	S2	2
"It's possibly true"	S1	1
"I have a hunch that it's true"	S0	0.5
"I know nothing about it"	\emptyset	0
"I have a hunch that it's false"	\neg S0	-0.125
"It's possibly false"	\neg S1	-.25
"It's probably false"	\neg S2	-.5
"I believe it's false"	\neg S3	-.75
"I know for a fact it's false"	\neg K	-1.75

FIGURE 3.17: Types of Knowledge and their Numerical Representation

The Knowledge System

Before I describe how the User's knowledge of each fact is computed, I want to explain the different ways a User can suspect or know a fact. This is called the knowledge system. The knowledge a User has of a fact is what is in the User's mind, not what is true in "reality." This is a simple system that approximates the mental state of the User.

As I mentioned before, there are several different types, or levels, of knowledge for each fact. Figure 3.17 shows the complete set of ways the User can know or suspect a fact is true or false. The first column is what the User would say about the fact. The second column gives the abbreviation for this level of knowledge. The final column gives the numerical representation of this knowledge level.

There are a number of immediate questions pertaining to this knowledge system that I will answer. One, what is the difference between the levels of suspicion? Two, why doesn't the User simultaneously think a fact is "possibly true" and "possibly false"? Three, where do these odd-looking numbers come from and why are the negative numbers smaller in magnitude than the positive numbers?

There are four levels of suspicion (S0 through S3). Each is stronger than the previous. S0 is a *hunch*, which means it is not based on physical evidence. S1 to S3 are based on physical evidence, but connote stronger and stronger feelings of certainty.

For example, consider Fact 4: *George was actually disinherited*. At the beginning of the scenario the User has a hunch this has happened. She has no physical evidence, therefore her knowledge level for Fact 4 is S0. After she finds the calendar notation indicating a new will has been written (USER MOVE CALENDAR) she has a pretty strong suspicion this fact is true. Her knowledge level is S2. After she confronts George with the calendar (USER MOVE CONFRONT GEORGE), she believes it is true, but doesn't know for sure. Her knowledge

level is S3. Finally, when she catches George getting rid of the will (USER MOVE CATCH GEORGE), she knows George has been disinherited. Her knowledge level is K.

Determining a User's particular level of suspicion is a subjective process. For example, one could argue that after CALENDAR the User should only have knowledge level S1, for example. Whatever the case, the artist must specify his opinion of the User's knowledge for each fact in each situation.

The next question is: why doesn't the User simultaneously feel S1 and \neg S1 about a fact? They are certainly consistent. But, the idea of this knowledge system is to first capture which theory the User more firmly believes (true or false), and then assess the strength of that belief.

This is related to question three: where do the odd-looking numbers come from and why are the negative numbers smaller than the positive numbers? The numbers are artistically chosen to reflect the excitement of making changes from one knowledge level to another. For example, there is a bigger jump from S3 to K than from \emptyset to S3. This is because I believe (as the artist) that incremental jumps in suspicion are less intense than that final moment when the User finally knows something is true.

The values for the negative truth levels are all exactly one-fourth the magnitude of their positive counterparts. It is my feeling that for *Tea For Three* finding out a significant fact is false is not as interesting as finding out it is true. This comes from a particular property of the two facts that are in "reality" false: *The means of death was extra Ebullion added into Robner's tea cup* and *George is guilty*. If the User knows these are false, she still hasn't solved the mystery. This knowledge only narrows down the possibilities.

Perhaps another way to achieve this same effect would be to lower the importance of the facts that are false in "reality". There are many ways that this simple system could be improved. Please see the discussion of future work in Chapter 9.

Computing What the User Knows

The last detail required for creating the excitement graph is computing the User's knowledge level for each fact at any time. After this, I will show the charts and graphs of the User's excitement for the two example scenarios.

Each of the nine facts of *Tea For Three* has its own function that takes as input the history of USER MOVES to that point and returns the knowledge level of the User for that fact. The functions are straightforward; more like tables than calculations.

For some facts, the User's knowledge is calculated from just the set of USER MOVES that have happened. For others, the User's knowledge is derived from both the set of USER MOVES and from the knowledge level of facts that have been previously calculated. There are no circular dependencies. The derived knowledge levels could have been calculated with just the USER MOVES, but I found it convenient to refer to previously calculated levels.

One nice property of *Tea For Three* is that each fact's knowledge function is fairly simple, and most of them are independent of the others. What follows is the method of determining the knowledge level for each fact.

Fact 1: The means of escape was over the balcony

This function is better explained with a chart than with pseudocode. Fact 1 depends on four USER MOVES: MUD, PAINT, HOLES, and LADDER. Each legal combination of the four USER MOVES leads to a specific knowledge value. The following chart shows how to compute the User's knowledge of Fact 1.

User's Knowledge of Fact 1				
MUD	PAINT	HOLES	LADDER	Knowledge Level
				\emptyset
•				S1
•	•			S2
			•	\emptyset
•			•	S2
•	•		•	S3
		•		S2
•		•		S3
•	•	•		K
		•	•	S2
•		•	•	S3
•	•	•	•	K

The first four columns indicate whether the USER MOVE at the top of the column has happened. A "•" in a column means that USER MOVE has happened. A row represents one possible description of the history of USER MOVES. For example, the third row represents any history where MUD and PAINT have happened, but HOLES and LADDER have not. Notice that this function doesn't depend on the order of the USER MOVES. As you will see, the User knowledge of facts do not usually depend on order, with one exception.

The last column gives the User's knowledge based on the history represented by the first four columns. For example, if the User history included events HOLES and MUD, but not PAINT or LADDER, then we can see in the eighth row that the User's knowledge of Fact 1 is S3. As I have mentioned above, I (as the artist) specified the User's knowledge levels found in this table, based on my best guess of the User's knowledge.

There are only twelve rows in this chart instead of sixteen (2^4) since PAINT cannot happen unless MUD happens first. Of course, this is because MUD precedes PAINT in *Tea For Three's* plot graph (Figure 2.5).

A function format is used to describe how to compute the User's knowledge of the rest of the facts. There are three commonly called sub-functions. One, Happened (USER MOVE A) is true if the argument USER MOVE A has already happened. Two, USER MOVE A Precedes USER MOVE B is true if the USER MOVE A happened before USER MOVE B. Three KnowledgeOf(Fact A) refers to the previously computed value of the User's knowledge for Fact A. These functions use the history of USER MOVES as an implicit argument.

I will just list the knowledge functions without comment, unless comment is necessary.


```

% Fact 2: The means of death was Loblo and Ebullion
KnowledgeOfFactTwo (history)
  if ( Happened (AFL) and
      Happened (FRAGS) and
      Happened (LOBLO) )
    then return K
  if ( Happened (FRAGS) and Happened (LOBLO) )
    then return S2
  if ( Happened (FRAGS) ) then return S0
  else return ∅

```

```

% Fact 3: Baxter had a legitimate motive
KnowledgeOfFactThree (history)
  if ( Happened (FOCUS) and
      ( Happened (MERGER) or Happened (NOTEPAD) ) )
    then return K
  if ( Happened (NOTEPAD) and
      Happened (MERGER) and
      NOTEPAD Precedes MERGER )
    then return S2
  if ( Happened (NOTEPAD) )
    then return S3
  if ( Happened (FOCUS) or Happened (MERGER) )
    then return S0
  else return ∅

```

```

% Fact 4: George was actually disinherited
KnowledgeOfFactFour (history)
  if ( Happened (CALENDAR) and
      Happened (CONFRONT GEORGE) and
      Happened (CATCH GEORGE) )
    then return K
  if ( Happened (CALENDAR) and
      Happened (CONFRONT GEORGE) )
    then return S3
  if ( Happened (CALENDAR) )
    then return S2
  else return S0

```



```

% Fact 7: Dunbar is guilty
KnowledgeOfFactSeven (history)
  if ( Happened (TICKET/AFFAIR) )
    then return K
  if ( Happened (CONFRONT DUNBAR) )
    then return S2
  if ( Happened (LOBLO) )
    then return S1
  if ( KnowledgeOf (Fact 2) = K or S3 )
    then return S0
  else return  $\emptyset$ 

```

```

% Fact 8: Baxter is guilty
KnowledgeOfFactEight (history)
  if ( Happened (TICKET/AFFAIR) )
    then return K
  if ( Happened (MERGER) )
    then return S2
  else return  $\emptyset$ 

```

```

% Fact 9: George is guilty
KnowledgeOfFactNine (history)
  if ( KnowledgeOf (Fact 7) = K or
        KnowledgeOf (Fact 8) = K )
    then return  $\neg K$ 
  else return Max (Half (KnowledgeOf (Fact 1)),
                  Half (KnowledgeOf (Fact 2)),
                  Half (KnowledgeOf (Fact 4)) )

```

Excitement Graphs for the two Scenarios

Given that the evaluation function can compute the knowledge level for each fact at any time, it can compute the User's excitement over time. The chart in Figure 3.18 shows computation of the excitement for the good scenario. The total excitement at each time is plotted below the chart.

The first two columns of the chart give the time and USER MOVE at that time. The next nine columns show the User's knowledge levels and her derived excitement for each of the nine facts. The last column shows the total excitement, which is calculated by summing the User's feeling of excitement from each individual fact. To top row of the table shows the fact number, and just below that the importance of each fact. Recall, the excitement of each

Excitement Chart of the Good Scenario											
t	Fact Number →	1	2	3	4	5	6	7	8	9	Total Excite.
	Importance →	3	4	2	2	4	2	5	5	5	
	USER MOVE	k,e	k,e	k,e	k,e	k,e	k,e	k,e	k,e	k,e	
0	Beginning ...	∅	∅	∅	S0	∅	∅	∅	∅	∅	0
1	NOTEPAD	↓	↓	S3,6	↓	↓	S1,3	↓	↓	↓	9
2	CALENDAR	↓	↓	↓	S2,3	↓	↓	↓	↓	S1,5	8
3	MUD	S1,3	↓	↓	↓	↓	↓	↓	↓	↓	3
4	PAINT	S2,3	↓	↓	↓	↓	S2,2	↓	↓	↓	5
5	MERGER	↓	↓	S2,2	↓	↓	↓	↓	S2,10	↓	12
6	CONFRONT GEORGE	↓	↓	↓	S3,2	↓	↓	↓	↓	↓	2
7	FOCUS	↓	↓	K,10	↓	↓	↓	↓	↓	↓	10
8	CATCH GEORGE	↓	↓	↓	K,8	↓	↓	↓	↓	S2,5	13
9	LOBLO	↓	S0,2	↓	↓	↓	↓	S1,5	↓	↓	7
10	LADDER	S3,3	↓	↓	↓	↓	S3,2	↓	↓	↓	5
11	HOLES	K,12	↓	↓	↓	↓	K,8	↓	↓	↓	20
12	FRAGS	↓	S2,6	↓	↓	-S0,0.5	↓	↓	↓	↓	6.5
13	AFF	↓	↓	↓	↓	-K,6.5	↓	↓	↓	↓	6.5
14	AFL	↓	K,20	↓	↓	↓	↓	↓	↓	↓	20
15	CONFRONT DUNBAR	↓	↓	↓	↓	↓	↓	S2,5	↓	↓	5
16	TICKET/AFFAIR	↓	↓	↓	↓	↓	↓	K,25	K,25	-K,18.75	68.75

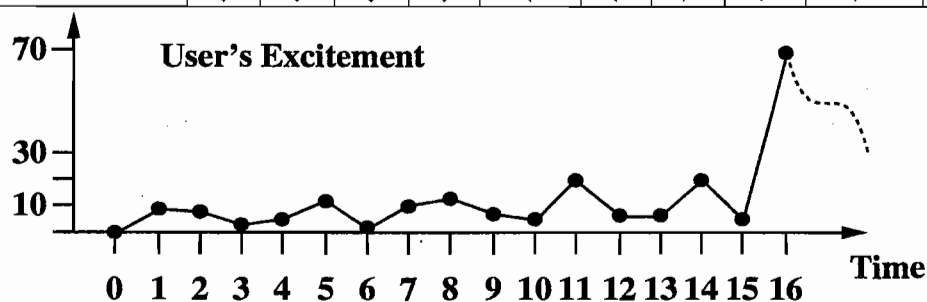


FIGURE 3.18: Excitement Chart and Graph for the Good Scenario

fact is calculated by multiplying the numeric knowledge level change by the importance of the fact.

In the nine columns, if the User's knowledge level hasn't changed, this is denoted by a "↓". If her knowledge level has changed, then her new knowledge level is given, along with the excitement derived from this change. As an example, at $t=8$, the User has just caught George. At this point, the User's knowledge of Fact 4: *George was actually disinherited* goes from S3 to K. The numeric knowledge level change is $|7 - 3| = 4$. The importance of Fact 4 is 2. Thus, the derived excitement is $4 * 2 = 8$. As you can see, the column contains "K,8" which shows the new knowledge level and the derived excitement.

The graph of the User's excitement throughout the good scenario is shown below the chart in Figure 3.18. As before, the dotted line at the end represents the excitement level of the User after the experience is over.

As we can see, the experience has ups and downs, but in general builds till the climax at time sixteen. There are good intermediate high points at times eleven and fourteen. Although this is not a perfect shape, it is a good shape. This scenario builds slowly and has an intense climax, which are characteristics of a good experience. In Section 3.7.2, we will

Excitement Chart of the Bad Scenario											
t	Fact Number →	1	2	3	4	5	6	7	8	9	Total Excite.
	Importance →	3	4	2	2	4	2	5	5	5	
	USER MOVE	k,e	k,e	k,e	k,e	k,e	k,e	k,e	k,e	k,e	
0	Beginning ...	∅	∅	∅	S0	∅	-S2	∅	∅	∅	0
1	LOBLO	↓	S0,2.0	↓	↓	↓	S0,2.0	S1,5	↓	↓	9.0
2	HOLES	S2,6	↓	↓	↓	↓	S2,3.0	↓	↓	S1,5	14.0
3	MERGER	↓	↓	S0,1.0	↓	↓	↓	↓	S2,10	↓	11.0
4	MUD	S3,3	↓	↓	↓	↓	S3,2	↓	↓	↓	5.0
5	NOTEPAD	↓	↓	S3,5.0	↓	↓	↓	↓	↓	↓	5.0
6	FRAGS	↓	S2,6.0	↓	↓	-S0,0.5	↓	↓	↓	↓	6.5
7	PAINT	K,12	↓	↓	↓	↓	K,8	↓	↓	S2,5	25.0
8	CALENDAR	↓	↓	↓	S2,3.0	↓	↓	↓	↓	↓	3.0
9	LADDER	↓	↓	↓	↓	↓	↓	↓	↓	↓	0.0
10	AFL	↓	K,20	↓	↓	-K,6.5	↓	↓	↓	↓	26.5
11	CONFRONT DUNBAR	↓	↓	↓	↓	↓	↓	S2,5	↓	↓	5.0
12	CONFRONT GEORGE	↓	↓	↓	S3,2	↓	↓	↓	↓	↓	2.0
13	AFE	↓	↓	↓	↓	↓	↓	↓	↓	↓	0.0
14	CATCH GEORGE	↓	↓	↓	K,8	↓	↓	↓	↓	↓	8.0
15	TICKET/AFFAIR	↓	↓	↓	↓	↓	↓	K,25	K,25	-K,18.75	68.75
16	FOCUS	↓	↓	K,8	↓	↓	↓	↓	↓	↓	8.0

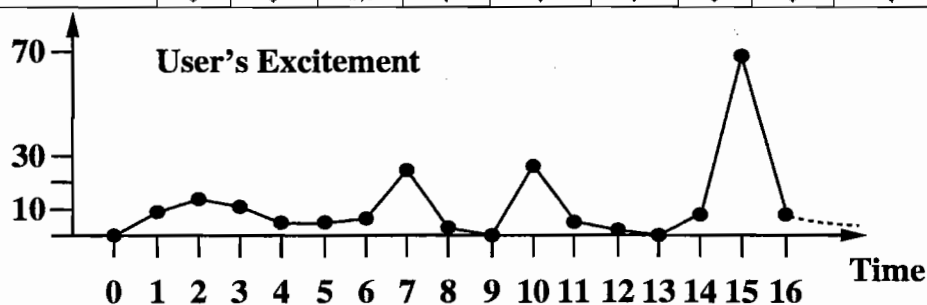


FIGURE 3.19: Excitement Chart and Graph for the Bad Scenario

see how the evaluation function quantifies this intuition.

Figure 3.19 gives the chart and graph of the User's excitement in the bad scenario. As you can see, it does generally build to a climax. There are a few reasons why this graph is not as good as the the graph for the good scenario. First, at time sixteen there is a relatively unexciting last moment. This is not good. As I've said before, I prefer the end to be the most exciting moment. Second, although times seven and ten are rather exciting, they are followed by periods of very low excitement. In the good scenario, the excitement never dips back down to zero. Third, there are only four peaks in the excitement graph, which is not as good as the six in the first scenario. This is not *a priori* bad, but since there are large valleys of unexciting time, it is bad.

This concludes the section on how to create the excitement graph for a scenario. The next section shows how *Intensity* is calculated, by comparing the actual excitement graph against an ideal excitement graph.

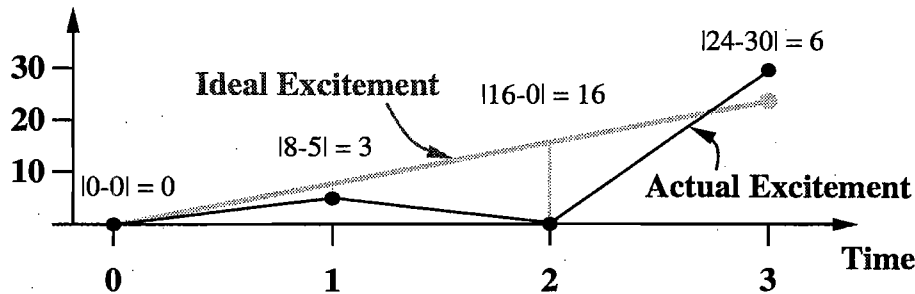


FIGURE 3.20: Matching Actual to Ideal Excitement in Made-Up Scenario

3.7.2 Matching Against the Ideal Excitement Graph

To illustrate how the a scenario's excitement graph is compared to an ideal excitement graph, let's return to our made-up example from the beginning of Section 3.7.1. Figure 3.20 shows the ideal excitement graph (in gray) from our made-up scenario superimposed over the actual excitement graph (in black) from Figure 3.15.

The ideal excitement graph goes from zero at the beginning to twenty-four at the end. The shape of the excitement graph, including its maximum value, is an artistic decision. At every time, the value of the ideal graph is based on the linear interpolation between the maximum (in this case twenty-four) and minimum (zero) values of the ideal. Thus, at time one the ideal value is eight, and at time two, the ideal value is sixteen.

To do the match, the evaluation function compares the actual excitement graph with the ideal excitement graph. For each time, the function takes the linear distance between the two graphs, and then weights the distance by the time (plus one). Figure 3.20 shows the distances between the actual graph and the ideal graph for each time: 0, 3, 16, and 6. The raw score is the sum of all the weighted distances.

The reason *Intensity* weights the distance by the time (plus one) is that I (as artist) care more about a highly exciting ending than a non-exciting beginning. This is similar to how *Options* considers early goals more important than later ones.

As I've described, the raw score for this made-up scenario is calculated in this way:

$$\begin{aligned} \text{Raw Score} &= (1 * 0) + (2 * 3) + (3 * 16) + (4 * 6) \\ &= 78 \end{aligned}$$

Thus, the raw score of *Intensity* is 78, in this made-up example. As usual, in the real case the raw score would be normalized into the ten-point scale to find the normalized value. Notice that a lower score represents a closer match. This is the one feature where the highest raw score is mapped to the lowest normalized score.

That is the overview of how the raw score for *Intensity* is computed from an actual excitement graph. Let's take a look at how this works on the two example scenarios.

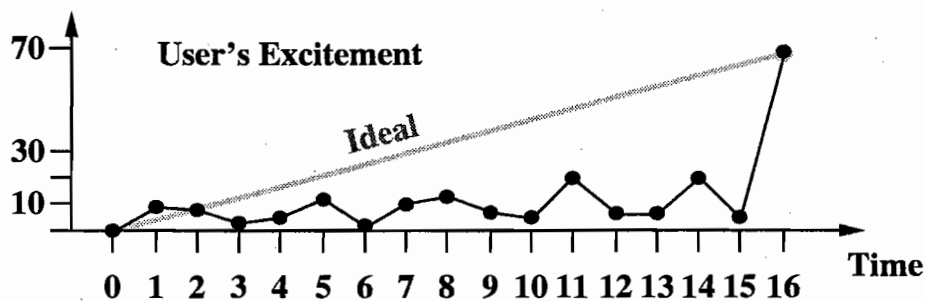


FIGURE 3.21: Matching Actual to Ideal Excitement in the Good Scenario

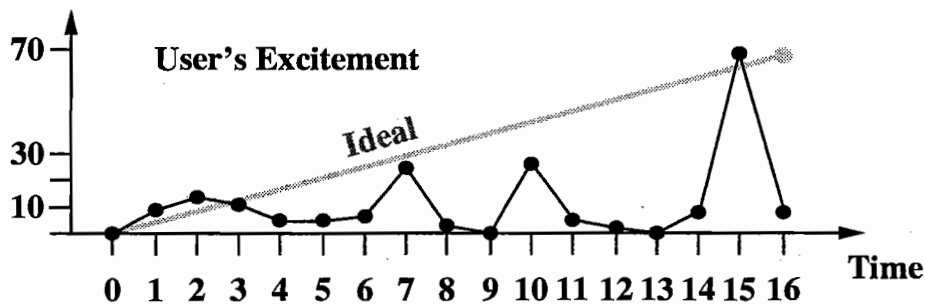


FIGURE 3.22: Matching Actual to Ideal Excitement in the Bad Scenario

Figure 3.21 shows the excitement graph from the good scenario along with the ideal excitement graph of *Tea For Three*. The ideal graph starts a zero (at time zero) and ends at sixty-eight (at time sixteen). The function calculates a raw score of 3939.8 for *Intensity*. The raw score for *Intensity* ranges from 3799 (the best match) to 6415 (the worst match). Therefore, the good scenario has a normalized value of 9.47, which is very high. This is consistent with my feelings that this scenario's excitement graph had a good shape.

Figure 3.22 shows the ideal and actual graphs for the bad scenario. It has a raw score of 4358.0, which gives it a normalized value of 7.86. This is consistent with my feelings about the bad scenario's excitement graph. It is fairly good, but has a few flaws.

Intensity is an important feature. It accounts for one-third of the total value of a User's experience.

3.8 The Manipulation Feature

The final feature of *Tea For Three's* evaluation function is called *Manipulation*. It is a measure of how much the User perceives her experience has been pushed-around or manipulated by the system. Since *Manipulation* occurs only with MOE MOVES, its discussion is deferred until I describe the MOE MOVES. See Section 5.1.3.

Feature	Weight
<i>Thought Flow</i>	1/6
<i>Activity Flow</i>	1/6
<i>Options</i>	1/6
<i>Motivation</i>	1/12
<i>Momentum</i>	1/12
<i>Intensity</i>	1/3
<i>Manipulation</i>	-1

FIGURE 3.23: Weights of the Features of the Evaluation Function

3.9 Normalized Values and Weights: Adding it Up

As I described in the beginning of this chapter, the evaluation function is a weighted sum of the normalized value of each feature. The weights, shown in Figure 3.23, have been hand-chosen to represent the importance of each feature's value to the entire value for the scenario. The value returned by the evaluation function is simply the sum of the normalized values from each feature multiplied by their respective weights.

Manipulation is a non-standard feature of the evaluation function. The sum of the weights of the first six features is one. As it turns out, the raw score for *Manipulation* is actually simply subtracted from the total evaluation function based on the other features. This contradicts some aspects of my previous discussion, but not in serious ways. Please see Chapter 5 for more details on *Manipulation*.

The final detail is that normalizing the raw scores of each feature requires both the minimum and the maximum raw score possible for that feature. Because I could not easily examine every scenario, and because I could not imagine any proofs for determining these values, I chose to use an empirical method to calculate approximately correct values.

I used five million randomly generated scenarios to calculate the range of raw scores for each feature. Although this is not a theoretical method which guarantees the highest and lowest values, it is useful, because raw scores that don't show up in five million examples are very rare, and hence may be safely ignored. It is safe to ignore these outliers because there is no requirement (in the system) that a scenario score in the range.

3.10 Discussion

There is one interesting aspect of the evaluation function that I would like to discuss.

I believe that there are two different types of features in *Tea For Three*. The first exist because this is interactive entertainment, the second because this is like traditional non-interactive entertainment.

The first type of feature measures how the User is judging her experience at a moment. They measure the quality of the interaction. Most of the features are of this type: *Options*,

Motivation, Thought and Activity Flow, and Manipulation.

Notice that these features are typically absent from traditional criticism, because they deal with the process of interacting. One could argue that when we watch a play or movie we are feeling vicariously how the characters are feeling, but I would maintain that this is different from how we would feel if we *were* the characters. Perhaps *Manipulation* is one example of a feature that would be present in both interactive and non-interactive media.

The second type of feature measures the quality of the scenario in a more traditional way. Although the User is experiencing the interactive drama in first person, I believe she eventually judges it in a way similar to how an audience judges it. The difference is that she judges her experience's story-like feeling through her memory, thinking back on what happened. *Intensity* is this type of feature.

The evaluation function for *Tea For Three* is a complex program that defies easy analysis. The next chapter attempts to bypass the detailed analysis of its correctness by judging it by its ability to rate scenarios. In the next chapter, I argue that the evaluation function has, to a useful degree, captured the artist's (my) aesthetic for *Tea For Three*.

Chapter 4

Testing the Evaluation Function

We have seen in the previous chapter how the evaluation function works. The next and final step of this part (Part I) of the dissertation is try to judge how well the evaluation function is working.

The first section of this chapter argues that the relevant measure of success is how well a particular artist's aesthetic has been encoded into a particular evaluation function. If the evaluation function says a scenario is good, the artist should agree. If the evaluation function says it's bad, the artist should also agree. Likewise with every rating in between. Let's refer to the degree to which they actually agree as the *accuracy of the match* between the artist and the evaluation function.

The ultimate goal of an evaluation function is to have as much accuracy as possible. In our case, success occurs if the evaluation function implemented for *Tea For Three* encodes *my* aesthetic, because I am the artist who has created the evaluation function.

The rest of this chapter will address several aspects of this goal. First, it is important to understand why it is difficult to determine the accuracy of the match when the domain, artistic criticism, is subjective and when the evaluation function is judging scenarios, which are abstract representations.

This leads to the next section, where we discuss a methodology for doing a study for determining the accuracy of the match. We will learn how this methodology addresses these concerns and others. After presenting the methodology, I will present the results of the study, including some rudimentary analysis, which will show a large degree of correlation between the evaluation function's ratings and the artist's (my) ratings across scenarios. Finally, I shall argue that this evaluation function has succeeded adequately in encoding the artist's non-trivial aesthetic.

4.1 Determining Success

I believe that an evaluation function should be judged by how well it captures a particular aesthetic. For *Tea For Three* this means the aesthetic of the artist. In this subjective domain there is no objective truth as to what makes a scenario good or bad. Critics disagree all

the time. Thus, the evaluation function cannot be judged against some “right” answer. However, to me it is clear that the evaluation function should be returning a set of ratings consistent with *some* aesthetic, be it from a single person or a group of people.

In the case of *Tea For Three*, I think it would be right for the evaluation function ratings to match those of the artist himself. One could argue this evaluation function should match other aesthetics, such as a professional writer’s, or popular opinion, or maybe a professional critic’s. I would agree these are valid choices, but I have chosen to try to match the artist’s aesthetic.

This is a good choice because it is the artist who will be crafting the evaluation function, and he will likely want to take control of the creative output of the system, which is the experience of the User. The closer this experience matches the Artist’s vision the better. The evaluation function is a logical place to encode aspects, and possibly the core, of this vision.

So, for the purposes of this research on interactive drama, success in the implementation is achieved when there is great accuracy of the match between the artist and the evaluation function. The bulk of the consideration in the methodology and analysis is given to assessing this accuracy.

4.1.1 Difficulties

A first source of difficulty arises from the possible criticism that the artist’s (my) aesthetic is simply boring and of no interest. If this is the case, one might argue, these results will show us nothing about what it might be like to encode an interesting aesthetic, because we might presume that an interesting aesthetic is somehow more complex and subtle. As an extreme example consider a trivial, idiosyncratic aesthetic: “Take the numerical value (1 to 16) of the fifth USER MOVE and divide by 2.”

To assess the interestingness of the aesthetic I have employed two experts to rate the scenarios along with me. If one or both of the experts share the artist’s aesthetic, then it would be hard to claim the aesthetic was uninteresting. It probably isn’t great art, but at least it isn’t so uninteresting as to be non-useful. On the other hand, if both experts seem to agree with each other and disagree with the artist and evaluation function, that would leave open the possibility of an uninteresting aesthetic. As we shall see, one expert was in disagreement while the other showed a high enough degree of agreement to suggest that the aesthetic was interesting.

Another possible source of difficulty is inconsistency. It may be hard to match an aesthetic exactly if the artist is not consistent. A scenario that is rated highly might be rated less well the next day. For this reason, it may be impossible to have perfect match, even if we created the ultimate evaluation function. In the study, each rater rates each scenario twice, in order to assess the internal consistency of the rater. This gives us some information on how to judge how close two raters could be, even if they are the same person.

Another thing that makes this rating comparison difficult is the abstract, non-interactive representation used for scenarios. Figure 4.1 shows an example scenario as it was actually

Story #1.

Find Mud.
 Find Notepad.
 Find Ladder.
 Merger explained.
 Find Calendar.
 Confront George.
 Find Holes.
 Catch George.
 Find Paint.
 Find Frags.
 Anal. frags for Ebul.
 Find Focus papers.
 Find Loblo.
 Anal. frags for Loblo.
 Confront Dunbar.
 Ticket/affair.

Evaluation: zero * ** *** **** ***** *****

FIGURE 4.1: An Example Scenario, as Presented to Raters

presented to the human raters. Simply determining a rating is difficult for several reasons.

First, these scenarios are not interesting themselves (as fiction, for example.) Only somebody familiar with *Tea For Three's* characters and events could rate a scenario. For this reason I have chosen as my outside experts Professor Joseph Bates (Carnegie Mellon, Computer Science) and an English Professor familiar with *Deadline* who is interested in literary theory (English Prof, for short). Both of them are familiar with *Tea For Three* because both are familiar with Infocom's *Deadline*, the game on which it is based. Before the study, I conveyed to both the differences between the two.

Second, these scenarios don't convey all the details of the final product. What this process might be similar to is one where a critic is presented with the outline of a movie. The actual movie might be great or stink, but the claim the critic is going to make is whether he or she thinks the movie would be good based on competently executing the outline.

Third, because the representation is a non-interactive description of an interactive experience, the raters are going to have to get inside the head of the User. They are going to have to ask themselves, if somebody went through this set of USER MOVES, would it be a satisfying experience. This is not the same as asking whether it would be satisfying to watch. I have no specific solutions for these last two difficulties which seem inherent to the problem.

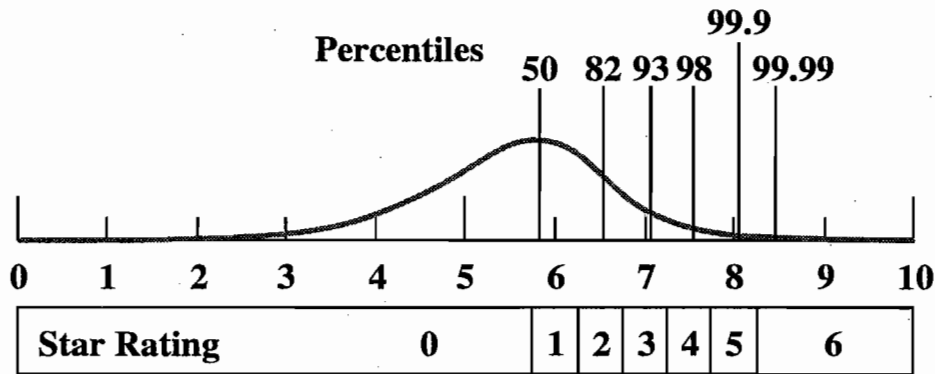


FIGURE 4.2: Distribution of Values and Mapping to Stars

The final difficulty is that there is no one right way to measure the accuracy of the match. Scientists use many measures, each of which has different meanings and connotations. Because of this, as you will later see, we have chosen to measure accuracy with a variety of established measures.

4.1.2 Nature of the Result

I want to remind you that my result is an existence proof of the ability to encode one artist's aesthetic (that is interesting) into an evaluation function. I will make no claims that this is an artistically great aesthetic. I will make no claims that this evaluation function represents the only or best mechanism for encoding an aesthetic. I will not claim that this evaluation function paradigm is capable of encoding all or most aesthetics. The basic contribution of this part of the dissertation is to show one aesthetic that has been encoded into one evaluation function. In Section 9.2, I speculate briefly about how this evaluation function might apply to other interactive dramas, other styles of interactive drama, and other artists.

4.2 Study Methodology

The basic form of the study was to take a suite of thirty-five scenarios and present them to the evaluation function and three human raters, and compare the results. The actual form of the study was a little more complex than that.

First, I considered the real-valued zero-to-ten rating system to be too complicated for a person to use. Upon observing many sample scenarios, I broke the scale of ratings from 0 to 10 into seven different ranges, denoted by a star-rating (*) from zero to six stars.

Figure 4.2 shows the distribution of randomly generated scenarios¹ distinguished by the rating the evaluation function gave to each. Below that is the range for each of the seven star values. The bottom 50th percentile receive zero stars. This is because I feel scenarios

¹Generated by choosing randomly among legal USER MOVES at each point in the scenario.

in that range were very bad. One and Two star scenarios are bad, Three okay, Four is good, and Five and Six are near perfect. Any evaluation function rating can be mapped into a star rating. Thus, the human raters use the more intuitive six-star rating, and compare that to the mapped evaluation function ratings.

The test suite of thirty-five scenarios was created by a random process that did not involve the Artist. (Artist is capitalized to show this is the artist who created *Tea For Three*). To facilitate the selection of distinguishable scenarios from each star-rating, I chose a smaller, midpoint-centered range for each star-rating. Many scenarios were chosen at random. When a scenario (as rated by the evaluation function) fell within one of the smaller ranges, it was selected to be one of the thirty-five stories. Only the first five scenarios from each star-rating was included. The final suite of thirty-five scenarios thus contained five scenarios from each of the seven star-ratings.

I already had the evaluation function ratings for each of the stories. The next question was how to present the suite of stories to the human raters, so that we could make an accurate and fair comparison.

I had several criteria that I wished to have in my presentation method. Let me explain how these scenarios were presented, along with the benefits of each aspect of the presentation.²

I gave each rater each scenario twice, so that his or her internal consistency could be measured. Effectively, I doubled the number of scenarios to seventy.

Because I wanted to ensure that the raters were showing consistency over time (as opposed to just having one lucky day) I divided the rating period into five days. On each day, the rater would rate one-fifth of the scenarios, or fourteen scenarios.

The seventy scenarios were randomly distributed with the provision that the same scenario never appeared on the same or consecutive days. This ensured that the daily distribution of fourteen scenarios was unknown: all could be good, all bad, or spread out. In this way, there was no way for a rater to game the test.

The same scenario was separated by at least one day so that the rater was less likely to remember what rating he or she gave the scenario the first time. Although this would not be enough time to forget something like a movie, this is reasonable in our case, since as Figure 4.1 shows, these scenarios are mechanically represented, and thus would be difficult to recall, especially given that he or she would be rating many other similar scenarios.

This methodology resulted in a set of ratings for each of the thirty-five scenarios. Each scenario had been rated once by the evaluation function, and twice by the Artist, Bates, and the English Prof. Since the evaluation function is deterministic, there is no need for the evaluation function to rate the scenarios twice. Figure 4.3 shows the raw data that was collected. **A** and **B** refer to the two different rating times for each scenario. The scenarios have been ordered to facilitate your reading of the table. As we can see, for example, Scenario One (near the bottom of Figure 4.3) received a Six-Star Rating from the

²This methodology was designed with the help of Professor Bonnie John, Computer Science and Psychology, Carnegie Mellon.

evaluation function, two Four-Star Ratings from both the Artist and Bates, and a Three-Star and a Six-Star Rating from the English Prof.

4.3 Study Results

It is clear from the results given in Figure 4.3 that there is not an exact match between any two raters. The question then becomes how can we make sense of the data.

In the fields of statistics and social science there are many ways that sets of data such as these are analyzed, although none is seen as universally correct. I propose to use three types of measurement. The first seeks to determine what degree of agreement exists between the ratings of one rater and the ratings of another. These are the various agreement coefficients. The second are simple vector differences between rating sets. The third type seeks to determine if there is a linear relationship between the ratings of one rater and the ratings of another. These are called linear regressions and correlations.

Because there is no definitive and correct answer, I cannot tell you definitively which model of analysis is correct. Therefore, for each model or method I will explain why it might be relevant. The set of all models will form the basis of comparison between sets of ratings. That way, we can see the problem analyzed from a variety of directions, any or all of which could be accepted by a given reader.

The “result” of this section is that the evaluation function has encoded to a useful degree the aesthetics of the artist. This is suggested by the data presented in Figure 4.4, which shows how different pairs of raters compare to each other. Overall, the evaluation function to artist comparison is closer than all other comparison pairs, except the comparison of the artist to himself.

4.3.1 Models of Comparison

The first model of comparison is a coefficient of agreement. In its basic form, this is a measure of how often one rater agrees exactly with another rather. The simplest agreement coefficient is percentage match ($M_{\leq 0}$), which is just what percent of the time do the two raters agree exactly. Another simple agreement coefficient is to define agreement as rating within one ($M_{\leq 1}$) star, and compute the percentage of matches on that basis. To visualize these measures (as well as the ones that follow) I have created a scatterplot for each set of data.

A related agreement coefficient is called Cohen’s Kappa (κ)[11]. This is also measuring percentage match, but corrected for chance. κ scores zero when percent matching is due to chance, up to one for a perfect match, and below zero when the match is less likely than chance.

All three of these coefficients are useful because they can assess a base level of absolute agreement between two raters. However, they do not say anything about how close the mismatch is. 0 vs. 2 is considered the same badness (not matching) as 0 vs. 6. This leads us to two agreement coefficients which take distance into account.

Scenario No.	Ev. Fn.	Artist		Bates		English Prof	
		Set A	Set B	Set A	Set B	Set A	Set B
8	0	0	0	1	1	0	4
9		0	0	1	3	2	5
18		0	0	1	2	0	1
25		0	0	2	3	4	5
28		0	0	1	1	2	5
22	1	0	1	2	2	5	4
29		0	1	2	3	6	4
34		0	1	3	3	0	4
30		0	2	3	3	3	4
10		2	2	3	3	1	3
4	2	1	0	1	2	1	3
15		1	2	1	2	5	3
7		1	3	2	3	6	6
21		2	2	3	4	5	5
20		2	3	2	3	1	4
23	3	0	1	2	4	4	5
32		1	1	1	2	5	4
24		2	2	4	4	6	6
12		2	3	3	3	5	5
26		4	4	4	4	5	3
31	4	2	3	4	4	1	4
14		3	3	4	4	6	3
16		3	3	3	4	4	4
6		4	5	3	4	5	6
19		4	5	5	5	5	3
35	5	3	4	5	5	5	6
11		4	4	4	6	4	4
2		4	5	2	4	3	3
17		5	5	5	5	3	5
27		6	6	6	5	5	3
3	6	3	4	2	5	5	6
1		4	4	4	4	3	6
5		5	5	3	4	3	4
13		5	5	5	5	3	4
33		6	6	6	6	3	4

FIGURE 4.3: Results of Evaluation Function and Three Raters

The first is Weighted Kappa (κ_w)[12]. κ_w is like κ except that (in the case of my weights) non-matches measure disagreement based on the square of the distance between ratings. Thus, far away ratings are penalized more than close ratings. (κ says only exact matches count.) κ_w , like κ , is also corrected for chance.

The final agreement coefficient we will consider is Robinson's A (A_R)[33]. It is a measure also weighting agreement by squares of differences and is computed by the formula $A = 1 - D/D_{max}$ where D is the actual distance and D_{max} is the theoretically maximum distance for the problem. The primary difference between these last two seems to be that κ_w takes chance into account whereas A_R does not. Thus, one might consider κ_w the better measure.

These are not the only agreement coefficients available, but just a few to look at. These statistics serve as a guideline to show that some degree of agreement is going on. Because $M_{\leq 0}$, $M_{\leq 1}$, and κ all consider matching as binary, they are probably less informative than κ_w and A_R .

These agreement coefficients are related to the next category of measurement, Linear and Squared Differences. These values are calculated by simply averaging either the linear or squared distance between ratings, over all the ratings given. These are useful because they give an absolute measure of distance between raters. The squared version is especially useful because it penalizes far misses much more than near misses.

When one of the raters is the evaluation function, we supply two additional measures as a benchmark. These measures are the average squared differences based on two simple "evaluation functions": the first guesses randomly and the second always guesses 3. A random guesser turns out to be very bad, with average distances very high. A "three" guesser often fairs better, but still not as well as the strongly related raters. The reason we look at these two statistics is to make sure the evaluation function is doing better than trivial programs.

This leads to the final statistical technique, the linear regression. We use a regression instead of a correlation because the scenario samples are not randomly drawn from the population.³ As I've described, the scenarios are distributed evenly among star-ratings, as determined by the evaluation function.

A linear regression measures a linear relationship between the ratings of one rater and the ratings of another rater. The output of the regression is a measure r and its square r^2 as well as the *slope* and *intercept* of the line that best fits the data in a least squares sense.

r is called the correlation coefficient and it is a measure of the goodness of the match, similar to κ_w and A_R . It ranges from -1 to $+1$, with zero indicating no correlation, $+1$ indicating maximum correlation in the positive direction (best fit line has positive slope), and -1 indicating maximum negative correlation (best fit line has negative slope). Unlike κ_w and A_R , correlation doesn't require matching, just a linear relationship. For example, if rater one is always rating scenarios two star-ratings less than rater two, the linear regression will get $r = 1$, but the agreement coefficients will not be that high.

³Actually, if we did this, it is not clear what population to draw from. The distribution of values and thus scenarios can and should be different under the influence of dramatic guidance.

To assess consistency of a rater, we also make comparisons between the two rating sets that each rater gave. Each of the thirty-five scenarios has been rated twice. Because there is no sense of which scenario rating is the first or second, I have plotted each scenario twice in the scatterplot. Thus, a scenario that received ratings 3 stars and 5 stars would be plotted both at (3,5) and at (5,3). In this case, the correlation coefficient r becomes a measure of the intra-class as opposed to the inter-class correlation. Robinson[33] argues that this might be a better measure of agreement, and his measure is based on it, but we use it only for consistency comparisons.

There is a test called the t -test that will show if the correlation is statistically significant. I have shown the computed t value and the required t value for significance at the 99.9% level, given the correct number of degrees of freedom (usually $70 - 2 = 68$)⁴. Fisher's r -to- Z transformation to compute a confidence interval for r does not apply here because, again, the samples are not drawn randomly from a bivariate normal population. Normally in regression, confidence interval estimates for the population correlation are not possible. These analysis techniques are described in [28] and [16].

This leads to the question of which measure is best. As I mentioned above, there appears to be no consensus about which is best. My informal evidence is based on taking one measure from each category. Let's turn our attention to the main comparison we wish to make.

⁴Actually, I am using the entry for 60 degrees of freedom because my table[28] of the t -distribution jumps from 60 to 120 degrees of freedom.

4.3.2 Artist and Evaluation Function

To get some benchmarks for how consistent a single rater might be, let's take a look at how consistent the artist is when rating scenarios. Let's also use this as an example of how to read the data in the comparison.

Statistical Comparison: Artist vs. Artist																																																																			
Star Rating Scatterplot	Linear Regression	Agreement Coefficients:																																																																	
<div style="display: flex; align-items: center; justify-content: center;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 5px;">6</td><td></td><td></td><td></td><td></td><td></td><td></td><td style="padding: 2px 5px;">4</td></tr> <tr><td style="padding: 2px 5px;">5</td><td></td><td></td><td></td><td></td><td style="padding: 2px 5px;">3</td><td style="padding: 2px 5px;">6</td><td></td></tr> <tr><td style="padding: 2px 5px;">4</td><td></td><td></td><td></td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">6</td><td style="padding: 2px 5px;">3</td><td></td></tr> <tr><td style="padding: 2px 5px;">3</td><td></td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">3</td><td style="padding: 2px 5px;">4</td><td style="padding: 2px 5px;">2</td><td></td><td></td></tr> <tr><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">6</td><td style="padding: 2px 5px;">3</td><td></td><td></td><td></td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td></td><td></td><td></td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">10</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">1</td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">3</td><td style="padding: 2px 5px;">4</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">6</td></tr> </table> <div style="margin-left: 10px; font-size: small;">Artist</div> </div>	6							4	5					3	6		4				2	6	3		3		1	3	4	2			2	1	1	6	3				1	5	2	1	1				0	10	5	1							0	1	2	3	4	5	6	$r = .91$ $r^2 = .84$ <i>slope</i> = .91 <i>intercept</i> = .214 <u>t-test</u> computed <i>t</i> value: 18.56 required <i>t</i> value: 3.46 Passes at over 99.9%. $N = 70$	$M_{\leq 0} = .54$ $M_{\leq 1} = .94$ $\kappa = .54$ $\kappa_w = .91$ $A_R = .96$ Average Distances: $d^1 = .51$ $d^2 = .63$
6							4																																																												
5					3	6																																																													
4				2	6	3																																																													
3		1	3	4	2																																																														
2	1	1	6	3																																																															
1	5	2	1	1																																																															
0	10	5	1																																																																
		0	1	2	3	4	5	6																																																											

The big square on the left shows the scatterplot of scenario ratings plotted against one another. Each scenario is plotted according to what star rating it received from each rater. One rater is shown to the left, with corresponding star-ratings. The other shown below. The total in each cell is the number of scenarios that received the two ratings corresponding to the location. Because this is a consistency plot, the Artist appears on both axes. Likewise, each scenario has been entered twice. For example, we can see from the chart that five scenarios have been rated by the artist with both a zero and a one-star rating, and that these show up as 5's in cells (0,1) and (1,0). We can see from the scatterplot as a whole a pretty clear relationship between the artist and himself. The off-diagonal values come from inconsistency.

The section on the far right contains the agreement coefficients. We see that the artist agrees with himself exactly ($M_{\leq 0}$) in 54% of the cases, and rates stories within one star-rating ($M_{\leq 1}$) in 94% of the cases. Unweighted and weighted kappa (κ , κ_w) are .54 and .91 respectively. κ matches $M_{\leq 0}$ closely because the star-ratings are distributed fairly evenly. Finally, Robinson's-A (R_A) scores .96 for the artist.

Below the agreement coefficients we see the distances, both linear and squared, averaged over all scenarios. The average linear distance between ratings (d^1) was .51, whereas the average of the squared distances (d^2) was .63.

Finally, the middle column shows the analysis based on a linear regression. We can see the slope and intercept of the best-fit line, and the correlation coefficient, r , and its square, r^2 . $r = .91$ for the artist to artist comparison. We also see the results of a *t*-test for this

data, showing a value of 18.56 for t that clearly exceeds the required value of 3.46 for the 99.9% statistical significance level (for 68^5 degrees of freedom.)

The artist to artist comparison sets up a maximum achievable consistency for this artist's aesthetic. We should accept as indistinguishable any evaluation function that can be as close to the artist as the artist is to himself. Notice also that since the artist is somewhat inconsistent, it would be impossible to match the artist exactly. In any case, let's look at how the artist does compare to the evaluation function:

Statistical Comparison: Evaluation Function vs. Artist								
Star Rating Scatterplot				Linear Regression		Agreement Coefficients:		
Artist	6					2	2	
	5					2	3	4
	4				2	2	4	3
	3			2	1	5	1	1
	2		3	4	3	1		
	1		3	3	3			
	0	10	4	1	1			
		0	1	2	3	4	5	6
				Evaluation Function				
				$r = .87$ $r^2 = .76$ <i>slope</i> = .83 <i>intercept</i> = -.01 <i>t-test</i> computed t value: 14.63 required t value: 3.46 Passes at over 99.9% $N = 70$		$M_{\leq 0} = .36$ $M_{\leq 1} = .84$ $\kappa = .36$ $\kappa_w = .84$ $A_R = .92$		
						Average Distances: $d^1 = .83$ $d^2 = 1.3$ random $d^2 = 4.1$ 3's $d^2 = 3.9$		

As we can see, in every category of comparison, the evaluation function differs from the artist by more than the artist differed from himself. Thus, we can see there is no complete success. Yet, there is obviously a pretty strong relationship. An r value of .87 is high (especially compared with the .91 maximum). Likewise, being within one star-rating 84% of the time is pretty good.

We seem to be looking at an informal, non-provable likeness between the artist and the evaluation function. However, to get a better feel for the power of the connection between the evaluation function and the artist, I have made a ranked comparison (see Figure 4.4) using three statistics: Average Squared Distance, r , and A_R . As you can see below, "Ev. Fn. vs. Artist" ranks third, second, and second best among the comparisons. In particular, notice that except for the "Bates vs. Bates" Squared Distance, the Evaluation Function and the Artist are closer than everything else but the Artist to himself, including internal consistencies. To me, this indicates that we have made a significant amount of progress toward encoding an aesthetic. Thus I say the evaluation function has encoded the artist's aesthetic to a useful degree.

Because this is not a provable comparison, it does not make sense to belabor the statistics specifically.

⁵As above, actually 60.

Overall Comparison using Selected Measures:					
Squared Distance (d^2)		Correlation (r)		Agreement (A_R)	
Artist vs. Artist	.63	Artist vs. Artist	.91	Artist vs. Artist	.96
Bates vs. Bates	1.1	Ev. Fn. vs. Artist	.87	Ev. Fn. vs. Artist	.92
Ev. Fn. vs. Artist	1.3	Artist vs. Bates	.76	Bates vs. Bates	.86
Ev. Fn. vs. Bates	2.0	Bates vs. Bates	.72	Ev. Fn. vs. Bates	.83
Artist vs. Bates	2.1	Ev. Fn. vs. Bates	.72	Artist vs. Bates	.82
Prof vs. Prof	3.7	Prof vs. Prof	.25	Prof vs. Prof	.62
Ev. Fn. vs. Prof	5.7	Ev. Fn. vs. Prof	.25	Ev. Fn. vs. Prof	.59

FIGURE 4.4: "Ev Fn vs. Artist" Does Well in Comparison

4.3.3 Aesthetic is Interesting

The following comparison shows that there is a reasonably strong relationship between the ratings of the Artist and the ratings of Bates. This provides informal evidence that the aesthetic isn't so idiosyncratic so as to be uninteresting. Notice in this particular scatterplot each scenario is plotted four times, once for each possible pair of ratings.

Statistical Comparison: Artist vs. Bates							
Star Rating Scatterplot				Linear Regression		Agreement Coefficients:	
Bates	6				2		6
	5				3	5	10 2
	4	1	1	8	8	12	4
	3	9	4	11	6	1	3
	2	9	9	2	3	2	1
	1	13	4	1			
	0						
		0	1	2	3	4	5
				Artist			
				$r = .76$ $r^2 = .58$ <i>slope</i> = .56 <i>intercept</i> = 1.86 <i>t-test</i> computed <i>t</i> value: 13.94 required <i>t</i> value: 3.373 Passes at over 99.9% $N = 140$		Average Distances: $M_{\leq 0} = .29$ $M_{\leq 1} = .69$ $\kappa = .28$ $\kappa_w = .66$ $A_R = .82$ $d^1 = 1.1$ $d^2 = 2.1$	

4.3.4 Other Comparisons

The rest of the comparisons are found in Appendix A.

4.4 Discussion

We have seen an analysis of gathered data that suggests that the evaluation function is encoding the artist's aesthetic to a useful degree. We could do more statistical crunching

of the numbers, but this won't change the conclusion, since the statistics are suggesting, rather than proving, that we have a success.

I believe that from a practical point of view, r is probably the best measure of agreement. This is because in order for the drama manager to work correctly, all that is required is that there be a monotonically increasing function mapping the output of the evaluation function to the rating of the artist. Thus a strong linear relationship is effective. On the other hand, this might suggest the use of something like the Spearman rank-order coefficient, which is based on relative ranking rather than absolute numeric ratings. The only problem is that in terms of a study it might be very hard for a human to sort thirty-five scenarios.

Some critic of this work might say "Well, this comparison between you as artist and the evaluation function is ridiculous, because you implemented the function. Of course you can imitate it." I hope you can see from the previous chapter that executing the evaluation function in my head would be hopeless. To make this criticism one would also have to explain how Bates, who did not know the evaluation function nearly as well, could match it rather well.

A related criticism is that I could have changed my aesthetic to match the evaluation function. I suppose that could have happened, though I don't think it did. Again, one would have to account for Bates, who did not implement the evaluation function.

However, even if my aesthetic did change, that would be okay, as long as we still believe the aesthetic is interesting. This is because the creative process (in this case encoding the aesthetic) can often change an artist's perspective and opinion of things. This is a valid change that does not render the aesthetic invalid.

Finally, somebody once asked how much I had to tweak the evaluation function when I was creating it. The fact is that I designed it and then implemented it, with only a few minor tweaks in the *Intensity* feature. If I were to go back and improve it (a good area for future work), it might match my ratings even more closely.

Part II

Search

Chapter 5

MOE MOVES

My thesis, that *interactive drama is possible*, is supported by two pieces of evidence. First, that an evaluation function can encode an artist's aesthetic. Second, that an adversary search mechanism can effectively guide the User's experience by using this evaluation function.

The previous three chapters support this first piece of evidence. Chapter 2 describes one interactive drama, *Tea For Three*, and shows how the artist created a set of USER MOVES which represent the significant moments of *Tea For Three*. Chapter 3 explains how *Tea For Three's* evaluation function can take a scenario of USER and MOE MOVES and return a rating indicating its aesthetic quality. Finally, Chapter 4 describes a study that shows the evaluation function has, to a useful degree, encoded the aesthetic of the artist.

The next four chapters support the second piece of evidence. This first chapter contains a description of the MOE MOVES for *Tea For Three*. MOE MOVES are the mechanism by which Moe can affect the User's experience, in order to guide her to her destiny. Chapter 6 describes the search state, which is the mechanism that the search uses to represent the important changes (to the world, characters, presentation system, and User's mind) that happen when MOE MOVES are refined, and must have happened in order for USER MOVES to be recognized. Chapter 7 describes three different search strategies that use this search state. Finally, Chapter 8 describes experiments that suggest Moe can use these three strategies to guide an interactive drama effectively.

This chapter contains three sections. The first is an introduction to search, which includes descriptions of refinement, recognition, and *Manipulation*. The full description of these terms has been delayed until here. The second section describes the MOE MOVES of *Tea For Three* in detail. This description depends on a knowledge of the USER MOVES of *Tea For Three* (see Chapter 2). The final section gives some guidelines by which I selected the MOE MOVES. My hope is that these guidelines can serve as an example for future artists.

After reading this chapter, you should be familiar with the MOE MOVES of *Tea For Three*. You will also have an appreciation for how an artist selects a set of MOE MOVES.

Understanding this chapter depends on having a broad-stroke understanding of Moe. If you haven't read the description of Moe in Chapter 1, I would recommend that you go back and read it.

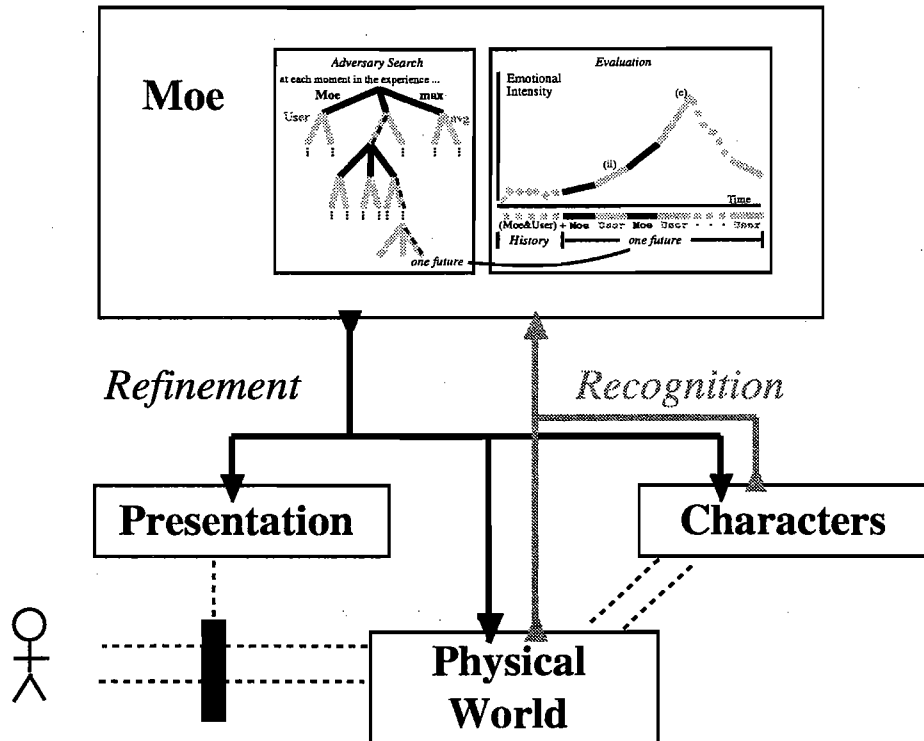


FIGURE 5.1: A Complete Interactive Drama System

5.1 Introduction to Search

This section describes recognition, refinement, and *Manipulation*. Recognition (*seeing*) and refinement (*doing*) are the two ways that Moe connects to the rest of an interactive drama system. *Manipulation* is a feature of the evaluation function that measures the negative cost of manipulating the User's experience.

Figure 5.1 shows a complete interactive drama system. The core of the system is a simulated physical world inhabited by autonomous interactive characters.

The world is presented to the User through a presentation system, which converts raw data from the simulated world into an understandable, and possibly stylistic, presentation. Usually, I think of these parts as being implemented on a computer, but the medium is not fixed. The world could be presented through text, animated graphics, fixed images, music, and through other means.

The User's actions take place in the simulated world. She is acting according to her own ideas of what is the right thing to do at any time. She could explore spaces, chat with people, pet animals, scream wildly, smoke, or do whatever she chooses.

In my model, each particular world is created by an artist (or artists) who wants the User to have a particular dramatic, story-like experience while in the world. Although the User has complete freedom to act within the limits of the world, the artist would like the experience of the User has to seem (to her) like a story. This story-like aspect of her

destiny

experience is called her destiny. A more detailed description of destinies and authoring is found in Chapter 1. Such a created world that has a destiny is called an interactive drama.

Moe's job is to ensure that the experience of the User matches the destiny given by the artist. Moe connects to the experience of the User in two ways: by *seeing* what the User does, and by *acting* to subtly guide that experience. In Figure 5.1, these are the arrows labelled "Recognition" and "Refinement".

5.1.1 Recognition

As described in Chapter 2, the artist must break down the User's possible experiences into significant moments, called USER MOVES. For each USER MOVE, the artist must write a recognizer, which is a little program that can identify when these significant moments have happened. Recognizers can examine the actions of the User, the bodies and minds of the characters, as well as the simulated world.

For example, consider the USER MOVE MERGER, from *Tea For Three*. MERGER happens when the User has a conversation with Baxter about the merger of his company with another company. If recognizers were general mechanisms, this recognizer would have to keep track of the dialogue between Baxter and the User, and somehow interpret that language to infer when the User learned the information about the merger.

My model is that recognizers should not be general, but as specific as possible. By doing this, implementing recognizers is easier. For example, let's imagine a very specific recognizer for MERGER. Instead of tracking the conversation, the recognizer just asks Baxter if the conversation took place. Baxter, the computer character, knows when it has explained the merger to the User. It can keep this information in a special place in it's mind. To recognize MERGER, the recognizer simply accesses this state in Baxter's mind.

As I mentioned in Chapter 1, the recognizers for *Tea For Three* have not been implemented. Although this makes evaluating the effectiveness of search problematic, Chapter 8 proposes a model for testing different search strategies without using real users interacting in a running simulated world.

5.1.2 Refinement

Guiding the experience of the User is Moe's second form of connection to the User. To achieve this goal, Moe uses MOE MOVES, which can change the behavior of the simulated world, characters, or presentation system. Each MOE MOVE has an associated program, written by the artist, that makes the changes to the other parts of the system. Running this program is called *refining* the MOE MOVE. This process is represented in Figure 5.1 by the arrows labelled "refinement."

In its most general form, a refiner is a little program that can reason about the current situation, possibly do some planning (or anything, really), and then make changes in either the character's minds, the physical world simulation, or the presentation system. In *Tea For Three*, for example, refiners can move the ceramic fragments to the shed, cause George

def

to go to the lake and shoot skeet, or change the presentation system so when it describes the muddy footprints on the balcony (seeing the mud would be recognized as the USER MOVE MUD), it also automatically describes the scraped paint on the railing (recognized as USER MOVE PAINT).

Refining a single MOE MOVE may change any number of aspects of the system, including changes that bear on future actions of the User. For example, *Tea For Three* has a MOE MOVE that will change the future behavior of the world simulation, by providing extra information in a report. When the User asks the police lab (part of the simulated world) to analyze the ceramic fragments for Loblo, they normally write a report indicating that the fragments did have traces of Loblo, and that Loblo was the cause of death.

If the example MOE MOVE is refined, the police lab will augment the report to include a small section showing a negative analysis for Ebullion. This gives the User additional useful information. Notice, the User did not order this analysis. It was done by the police lab, because the MOE MOVE was previously refined.

The set of MOE MOVES is like a bag of tricks that Moe can use to control aspects of the User's experience. Each interactive drama will have its own set of artist created MOE MOVES, crafted specifically to have certain effects in that interactive drama. For *Tea For Three*, I created eighteen MOE MOVES that are designed to be able to influence the User's opportunities and actions in a variety of ways.

In order to have maximum effect, the individual MOE MOVES should complement each other. In Section 5.3, I describe some guidelines for selecting an effective set of MOE MOVES.

As with Recognition, the actual programs for refining *Tea For Three's* MOE MOVES haven't been implemented. However, part of the design of a MOE MOVE is how it will be refined. The search needs to have an accurate model of each MOE MOVE in order to make an accurate decision. The refinement process of each of *Tea For Three's* MOE MOVES appears in Section 5.2.

5.1.3 Manipulation

In Chapter 3, I mentioned a feature of the evaluation function called *Manipulation*. *Manipulation* measures the degree to which refining a MOE MOVE manipulates or constrains the User in a way that makes no sense, except as a way to advance the plot. In *Tea For Three*, *Manipulation* detracts from the quality of the User's experience. Each MOE MOVE has an associated function which determines its *Manipulation* cost.

When a MOE MOVE is being refined, its *Manipulation* cost function might examine the history of the experience as well as the process of refinement to determine its *Manipulation* cost. When a MOE MOVE is made *within a search*, the refinement process can not be observed. Therefore, only the history may be examined, and the likely cost must be estimated.

Let's look at an example of how the refinement process can affect the *Manipulation* cost of a MOE MOVE. Suppose the User is heading down a hallway. A simple MOE MOVE

example might be trying to get the User to enter a particular room. The plan for refining this MOE MOVE has two parts. First, have someone sitting in the room playing music to attract the attention of the User. Second, if the first part of the plan fails, have the person call out "I think you ought to come in here because something important is about to happen."

The first case probably seems natural to the User, since it's pretty normal to be playing music and there doesn't appear to be any overt, plot-related goal to the music. If only this portion of the plan were needed to get the User to enter the room, the function computing the *Manipulation* cost would return zero, indicating that the refinement of this MOE MOVE didn't feel manipulative at all.

On the other hand, the execution of the second half of the plan will probably feel contrived to the User, since the words spoken by the character directly mention the plot. This would tend to destroy the User's suspension of disbelief. The User would find the interaction unbelievable and manipulative. In this case, the function would return a non-zero *Manipulation* cost.

The *Manipulation* cost of a MOE MOVE might also depend on the concrete choices of the User in the simulated world, not just on the parts of the refinement program that are executed. For example, suppose the User is in a room with four doors, and Moe is refining a MOE MOVE that is trying to make the User use one particular door.

The refinement of this MOE MOVE can use a heavy-handed technique: lock all three other doors. This certainly works, but is it manipulative? Only if the User notices. Thus, if the User chooses the unlocked (preferred by Moe) door first, there is no *Manipulation* cost, since she didn't even know that the other doors were locked.

example However, at the other extreme, if the User tries all three locked doors first, she will probably realize that she is being herded by the system. This will feel very unsatisfying because she is being forced to choose the "correct" door. Therefore, this MOE MOVE should have a large *Manipulation* cost. As you can see, the function computing the *Manipulation* must monitor the actions of the User, in order to be completely accurate.

In general, *Manipulation* cost functions are sensitive to both the context and process of refinement. However, because the refinement programs for *Tea For Three's* MOE MOVES haven't been implemented and tested, and Moe hasn't been connected to a running simulated world, I have hand-chosen a single, fixed *Manipulation* cost for each MOE MOVE. If the refinement of a MOE MOVE seems to risk manipulating the User, I have assigned a representative non-zero *Manipulation* cost. This cost is supposed to represent an average, probabilistic cost for refining the MOE MOVE in the absence of the ability to monitor the User or the refinement process.

Since the search is considering experiences where MOE MOVES are being refined in the future, such a probabilistic representation is also appropriate. This is because the MOE and USER MOVES are too abstract to consider the details of the refinement. Thus, even if the simulated world and refinement programs were fully implemented, the search would still need a way to estimate *Manipulation*. Of course, more complex function that depend on the previous behavior of the User or other factors could be developed. I have chosen a fixed, representative value as a simple initial model.

Manipulation affects the evaluation of an experience by simply subtracting its cost from the value returned by the evaluation function without *Manipulation*. For example, suppose an experience with a given sequence of USER MOVES is given a value of 6.84 by the evaluation function, without considering *Manipulation*. Further suppose that three MOE MOVES were made during that experience, and that their individual *Manipulation* costs were 0.2, 0.0, and 0.25. The true value of the scenario, given by the evaluation function considering *Manipulation*, is:

$$6.84 - (0.2 + 0.0 + 0.25) = 6.39 .$$

For scenarios rated above 5.0, a decrease of .5 in the value of a scenario is a noticeable decrease in its aesthetic quality. This is like losing “one-star” from the rating of the experience. (The Six-Star rating system is described in Chapter 4.) The *Manipulation* costs of *Tea For Three’s* MOE MOVES range from .25 to zero.

There is no proof of correctness for values that I have chosen, but the values are based on the same artistic intuition that I used to create the values found in the evaluation function. During the discussion of the MOE MOVES in the next section, I explain some of the reasons behind the *Manipulation* costs.

5.1.4 The Search

At any given point in an interactive drama, the User has experienced certain things and made certain decisions. I call this the history of the experience so far. The search represents this history as a sequence of USER MOVES and previously refined MOE MOVES.

The search projects all possible futures of the User’s experience. The projected experiences are, of course, abstract representations of what might happen, leaving out many concrete details. The search includes all possible USER MOVES and any MOE MOVES that Moe might refine to help guide the User’s experience.

By rating each projected future with the evaluation function, and backing up all the values, Moe can choose which MOE MOVES, if any, would most improve the User’s chance of having a good experience. Without getting into more details, a game-tree search paradigm is used to determine whether to make MOE MOVES. See Chapter 7 for details.

That concludes the introduction to search. The next section describes *Tea For Three’s* MOE MOVES, which, when refined, are the mechanism by which Moe can guide the experience of a User. To conclude this chapter, Section 5.3 gives some suggestions about how an artist might think about selecting a set of MOE MOVES for his interactive drama.

5.2 The MOE MOVES

Tea For Three has eighteen MOE MOVES. This section gives a description of each, including how the MOE MOVES effect the User’s experience, how they might be refined, and their *Manipulation* costs. Certain descriptions illustrate important information about

MOE MOVES. I would advise reading these descriptions before proceeding with the other chapters that describe search. If you do not intend to read the search chapters, but want a briefer description of the MOE MOVES, please see Section 2.3.

Moe Move 1: Remove Dunbar (No Manipulation Cost)

This is a simple MOE MOVE. Dunbar is literally removed from the simulated world. Of course, this should be done only if the User is not in contact with Dunbar, but notice Dunbar can easily slip away if necessary. The intent of this MOE MOVE is to prevent a concrete sequence of events that could be recognized as either of the two USER MOVES, CONFRONT DUNBAR or TICKET/AFFAIR. In both, the User must talk to Dunbar or show her various items. If Dunbar is not in the world, this cannot happen. I call this type of MOE MOVE a *denier*, because it denies the User the opportunity to take actions that will be recognized by Moe as USER MOVES.

Because there is almost always a way to remove Dunbar without the User noticing, I have assigned no *Manipulation* cost to this MOVE. This might break down if somehow the User has trapped Dunbar in a closet, but recall that these *Manipulation* costs are average case assumptions and don't rely on context or details of the refinement. In a more sophisticated cost model, the refiner could notice whether Dunbar is trapped and assess a higher penalty in that case.

Moe Move 2: Dunbar Confronts User (Manipulation Cost: 0.2)

I call this type of MOE MOVE a *causer* since it causes a sequence of events that will be recognized as a certain USER MOVE. The purpose of this MOE MOVE is to *cause* the USER MOVE CONFRONT DUNBAR to be recognized, which happens when the User confronts Dunbar with the report showing that the cause of death involved her medicine, Loblo.

When this MOE MOVE is refined, Dunbar goes to the User and tries to make the User confront her with the report. This requires that the User have the report. Dunbar will approach the User and first give the User an opportunity to confront Dunbar. If that fails, Dunbar will ask about the papers that make up the report. The User should then show them to Dunbar. In the extreme, Dunbar resorts to taking the report from the User.

The trick is to make this interaction plausible (i.e., believable). I modelled this MOE MOVE as having some manipulation, since I didn't believe this interaction would always appear natural. For example, the User might get suspicious that Dunbar was doing something harmful and unnatural to herself. The *Manipulation* cost of this move is 0.2 .

Moe Move 3: Dunbar Confesses to User (Manipulation Cost: 0.25)

This MOE MOVE is a causer, like the previous one, except that Dunbar goes to the User to make the USER MOVE TICKET/AFFAIR be recognized. This happens when Dunbar confesses to being Baxter's lover and to helping him commit the murder. This must occur after the

User confronts Dunbar with the report, which would have been recognized as CONFRONT DUNBAR.

To refine this MOE MOVE, Dunbar first finds the User and engages her in conversation. This gives the User the opportunity to intimidate and pressure Dunbar. If the User pushes, then Dunbar gives in and confesses. If not, Dunbar may drop the concert ticket, while removing cigarettes (since she is nervous), in order to prompt her own breakdown.

Dunbar should act as if she feels much remorse and sorrow. As with causing the previously described confrontation, refining this MOE MOVE may not be possible without seeming odd to the User. I have assigned a *Manipulation* cost of .25 to this MOVE. As you will see, this MOE MOVE is tied for the highest *Manipulation* cost, which, as I've mentioned above, corresponds to half a "star".

Moe Move 4: Baxter Explains Merger (Manipulation Cost: 0.1)

This MOE MOVE is designed to cause the USER MOVE MERGER to be recognized, which happens when Baxter informs the User about the alleged thoughts of the deceased regarding the upcoming merger between their and another company. Importantly, this piece of information is refuted by the USER MOVE NOTEPAD, showing part of Baxter's motive for murder.

If this MOE MOVE is refined, Baxter will first find the User and strike up a conversation. In the course of the conversation Baxter will talk about the impending business merger. This will seem natural most of the time, but some Users might find it a little odd, depending on exactly how the information comes up. Thus, I have made the *Manipulation* cost of this MOE MOVE 0.1 .

Moe Move 5: George Confronts User (Manipulation Cost: 0.2)

This MOVE is designed to cause the USER MOVE CONFRONT GEORGE to be recognized, which happens when the User confronts George with the calendar notation that indicates a new will has been created by his dead father. George knew his father might disinherit him, so this provides a motive for George.

Refining this MOE MOVE makes sense only if the User has already found the calendar notation, which would have been recognized as CALENDAR. If this MOE MOVE is refined, George first comes up to the User and stands there nervously, wondering what the calendar is all about. He is naturally concerned that there may be a new will, but he doesn't know for sure. The interaction could happen in several ways. The User might take the opportunity to confront George, or George might have to ask the User about the calendar, or, in the worst case, George may take the calendar and look at it himself. In this way, this MOE MOVE is similar to MOE MOVE 2: DUNBAR CONFRONTS USER.

As with MOE MOVE 2, depending on how the confrontation happens, it may seem natural or unnatural. I think it will tend to seem contrived. For this reason, I have assigned a *Manipulation* cost of .2 to this MOE MOVE.

Moe Move 6: George is Caught (Manipulation Cost: 0.25)

This MOE MOVE is designed to cause the USER MOVE CATCH GEORGE to be recognized, which happens when the User catches George in the act of disposing the new will that he has acquired from the safe in the secret room. When George is caught, he has a breakdown. This MOE MOVE can be refined anytime after George finds out about the new will. That happens when George is confronted with the calendar notation, which is recognized as CONFRONT GEORGE.

If this MOE MOVE is refined, George first must get the will, if he doesn't already have it, and then let himself get caught. He does this by finding the User and acting very nervous around her. In the best case, the User asks George about the will herself, but at the other extreme George will have to force the conflict and pretend to break down under the pressure.

Because George's actions will probably seem unbelievable to the User, this MOE MOVE has a *Manipulation* cost of 0.25. Again, this MOVE is tied for the highest *Manipulation* cost.

Moe Move 7: Delay the "Analyze for Loblo" Report (No Manipulation Cost)

When the User requests the police lab to analyze the ceramic fragments for Loblo, some small amount of time passes, and then the report is returned to the User. The User's receiving the report is recognized as the USER MOVE AFL. This MOE MOVE is supposed to delay the return of that report.

When this MOE MOVE is refined, it changes the simulated world (the police lab) so that the report will be delayed indefinitely. I call this type of MOE MOVE (surprise!) a *delayer* since it is delaying the sequence of events which will ultimately be recognized as a USER MOVE. This is the only delayer used in *Tea For Three*. Below, we see how the next MOE MOVE (number eight) can be used to cause the delayed USER MOVE (AFL) to happen any time after it has been delayed.

Because the User has no previous knowledge of a reasonable amount of time to wait for such a report to be prepared, this MOE MOVE has no *Manipulation* cost. Any amount of time is plausible. One might argue that this MOE MOVE could feel manipulative if the User has previously received the AFE report in a much shorter amount of time. Future work in this area is to determine exactly how manipulated the User feels by MOE MOVES.

Notice that this MOE MOVE must be made before the User requests the report, otherwise the police lab will return it right away, and Moe would lose its opportunity to delay AFL.¹ Several MOE MOVES have this property. Because of this, the search might seem less like an alternation of MOVES, and more like an anticipation by Moe of what the User might do. Instead of reacting to the User's request for the report, Moe must anticipate it. I call this kind of MOE MOVE an *anticipating* MOE MOVE.

There is another related limitation. There are no provisions in the Moe Architecture for unmaking a MOE MOVE. For example, once Moe has refined MOE MOVE 7, the police

¹This is not the whole truth. A more detailed discussion follows in the description of MOE MOVE 17 and in the next chapter.

will always delay the report after creating it. Obviously, if the refinement of a MOE MOVE changes the world in some non-reversible way, it cannot be undone. But some MOE MOVES can be undone. For example, MOVES that affect the future behavior of the characters or world.

It would be useful if some architectural feature allowed Moe to unmake such MOE MOVES. However, no such feature exists, so if the artist wants the capability to “undo” a MOE MOVE, he must create another MOE MOVE to do it. For example, MOE MOVE 8 (described next) provides a way to “undo” MOE MOVE 7.

Moe Move 8: Return the “Analyze for Loblo” Report (No Manipulation Cost)

Assuming that MOE MOVE 7 has been previously refined, this MOE MOVE will cause the report to be delivered to the User, and the USER MOVE AFL to be recognized. As mentioned above, the report contains the positive results of the analysis of the ceramic fragments for Loblo.

The real potential power of this pair of MOE MOVES is to let Moe return the report at just the right moment in the User’s experience, spurring her on to an exciting conclusion. Like its partner, this MOE MOVE also has no *Manipulation* cost, since the User won’t know the correct amount of time an analysis should take. To sustain believability, the report will be introduced naturally, not just after the previous USER MOVE has been recognized.

Moe Move 9: Combine AFE and AFL (No Manipulation Cost)

Receiving the successful analysis for Loblo (recognized as AFL) is often more dramatic than receiving the negative analysis for Ebullion (recognized as AFE). The intent of this MOVE is to force these events to happen in a dramatically proper order. When refined, this MOE MOVE changes the future behavior of the police labs. When the User requests the analysis for Loblo, they will include a negative Ebullion report as the preface to the positive Loblo report. When the User receives this combined report, it will be recognized as the sequence of USER MOVES, AFE then AFL.

This is a new type of MOE MOVE which I call a *move substitution* since it substitutes the recognition of one USER MOVE with the recognition of a sequence of two USER MOVES. This MOE MOVE is also anticipating, since it must be refined before USER MOVE AFL.

There is no *Manipulation* cost for this MOE MOVE. Logically, it makes sense for the police lab to do an analysis for Ebullion, even though the User didn’t request one, since Ebullion is the original official cause of death. Thus, the User should find it plausible that the lab includes the result of such an analysis as well, stating that they worked at their own initiative.

Moe Move 10: Describe Holes and Fragments Together (No Manipulation Cost)

Usually, the User must examine or dig in the holes in the backyard to find the ceramic fragments, since they are buried. However, when this MOE MOVE is refined, the fragments

are unburied, leaving them in plain sight. In addition, the presentation system is directed to describe the fragments after the holes. This means that when the User takes the single action of looking around under the balcony, the system will describe first the holes and then the fragments. Therefore, her observations will be recognized as the sequence of USER MOVES, HOLES then FRAGS.

As above, this is a move substitution. This is also an anticipating MOE MOVE since it must be refined before the USER MOVE HOLES is recognized. There is no *Manipulation* cost, since it is plausible to find the fragments on the ground next to the holes.

Moe Move 11: George Shoots Skeet (No Manipulation Cost)

This MOE MOVE designed to cause the USER MOVE MUD by encouraging the User to step out onto the balcony. It only works when the User is in the library.

When this MOE MOVE is refined, George gets his shotgun, takes it out to the lake (which is within earshot of the library), and shoots some skeet. The success of this MOE MOVE depends on the User reacting to the gun shots by going out on the balcony to investigate. Once out on the balcony, the User will see the muddy footprints, which will be recognized as MUD. There is no *Manipulation* cost for this MOE MOVE, since it is plausible that George would shoot skeet.

I call this sort of MOE MOVE that may or may not work a *hint*. It encourages the USER MOVE MUD, but doesn't necessarily cause it. That is, the User may ignore the shot or investigate it in another way, such as by going outside through the front door.

Moe Move 12: George goes on Balcony (No Manipulation Cost)

This MOE MOVE is like the previous, except it is used when the User is outside, in the back yard. When this MOE MOVE is refined, George goes out on the balcony, looks around a bit, makes eye contact with the User, and then goes back inside. This MOVE is designed to give the User the idea to check out the balcony in the near future.

Like the previous MOE MOVE, this is a hint for the USER MOVE MUD, encouraging, but not forcing the User to go onto the balcony. Since the User shouldn't find it strange that George was on the balcony, there is no *Manipulation* cost for this MOE MOVE.

Moe Move 13: George Confesses Secret Room (No Manipulation Cost)

This MOE MOVE is a hint for the USER MOVE FOCUS. FOCUS is recognized when the User goes to the secret room, opens the safe, and finds the Focus Company papers indicating that Baxter was being blackmailed by the deceased.

When this MOE MOVE is refined, it changes what happens during the concrete sequence of events that are normally recognized as the USER MOVE CATCH GEORGE. Normally, during CATCH GEORGE, George breaks down in front of the User, admitting that he is trying to destroy the new will, which he is carrying. If this MOE MOVE is refined, George will also tell the User about the secret room and safe during his breakdown. Knowing that

there is a secret room should encourage the User to find the room, and thus find the Focus Company papers. Notice this is an anticipating MOE MOVE, which must be refined before USER MOVE CATCH GEORGE is recognized.

There is no *Manipulation* cost for this MOVE, since it seems plausible that George could spill all sorts of information to the User, if he is ashamed and upset. The location of the safe seems like a moderate, but pertinent, detail, and one that George was not hiding: after getting the new will, he leaves the secret room's door open and detectable.

Moe Move 14: Describe the Mud and the Scraped Paint Together (No Manipulation Cost)

Normally, when the User steps on the balcony, the muddy footprints are described, which is recognized as the USER MOVE MUD. In order to see the scraped paint on the railing, the User has to explicitly examine the railing. This is recognized as the USER MOVE PAINT.

However, when this MOE MOVE is refined, the presentation system is directed to describe the mud and the scraped paint together (in that order). Thus, when the User steps onto the balcony, she will see both in sequence, which will be recognized as the sequence of USER MOVES, MUD then PAINT.

This is a move substitution with the sequence MUD then PAINT substituted for the USER MOVE MUD. It is also an anticipating MOE MOVE that must be refined before MUD is recognized. There is no *Manipulation* cost for this MOVE since it is plausible that the scraped paint would be immediately obvious to a casual observer.

Moe Move 15: Create Ladder Tracks (No Manipulation Cost)

This MOE MOVE is a hint for the USER MOVE LADDER, which is recognized when the User finds the ladder in the shed. Ordinarily, there are no marks indicating that Baxter has moved the ladder from the balcony to the shed.

When this MOE MOVE is refined, the simulated world is changed to show tracks leading into the shed. The apparent explanation is that Baxter accidentally dropped the ladder right at the entrance to the shed, and, because of the dark, couldn't see the marks he left. These marks should encourage the User to enter the shed and find the ladder, which will be recognized as LADDER.

There is no *Manipulation* cost for this MOVE, since it is believable that Baxter accidentally dropped or dragged the ladder. One could argue that the User would feel manipulated if she had previously passed by the shed and *not seen* the marks. The presentation system must avoid this.

Moe Move 16: Fragments Move to Shed (No Manipulation Cost)

Normally, the ceramic fragments from the broken cup are buried in the holes underneath the balcony. However, sometimes the quality of the User's experience will be higher if the ladder and fragments are found together.

When this MOE MOVE is refined, the physical world is changed so that the fragments are moved to the shed, unhidden, and the presentation system is directed to describe the ladder and fragments together, in that order. Therefore, when the User enters the shed, she spots both the ladder and the fragments. Seeing the ladder and fragments in sequence is recognized as the sequence of USER MOVES, LADDER then FRAGS. This MOE MOVE is a move substitution which substitutes the sequence LADDER then FRAGS for the single USER MOVE LADDER.

As you might imagine, this MOE MOVE is incompatible with MOE MOVE 10: DESCRIBE HOLES AND FRAGMENTS TOGETHER. As we will see in Chapter 7, Legal Move Generation ensures that at most one of those two is refined.

There is no *Manipulation* associated with this MOVE, since the cup could have broken in the shed instead of below the balcony. In fact, one might argue that finding the fragments in the shed is more natural.

As you might have inferred, this MOE MOVE has another consequence. It removes the temporal relationship between USER MOVE HOLES and USER MOVE FRAGS: FRAGS no longer needs to be recognized after HOLES. As we shall see in Chapter 7, changing this relationship will affect Legal Move Generation.

In addition to destroying the old relation, this MOVE creates a new, but implicit, relation between LADDER and FRAGS. This relation is not represented by an explicit edge in the precedence DAG (see Figure 2.5), but implicitly in the move substitution.

Moe Move 17: George Doesn't take Will (No Manipulation Cost)

When the User shows George the notation in the calendar indicating a new will has been created by the deceased (recognized as the USER MOVE CONFRONT GEORGE), George decides to destroy the new will. Normally, George waits until he is left alone, and then he executes his plan. He goes to the secret room, opens the safe, takes the new will, goes to the lake, and finally destroys the new will by throwing it in the lake.

If this MOE MOVE is refined, George won't take and won't try to destroy the will. He will just wait. This MOE MOVE denies CATCH GEORGE and FOCUS. CATCH GEORGE is denied because George doesn't have the new will. FOCUS is denied because the User cannot access the safe (where both the new will and the Focus Company papers are located) unless George has accessed it previously. The effects of this MOE MOVE can be reversed by refining MOE MOVE 6: GEORGE IS CAUGHT.

Since it is believable for George to hide out for some length of time, the User cannot tell if Moe has refined this MOE MOVE or not. Thus, it has no *Manipulation* cost.

Because this MOE MOVE denies FOCUS, it must be refined before George takes the new will from the safe. Ordinarily, this would imply that this MOE MOVE must be an anticipating MOE MOVE, since the only way to *guarantee* that it is refined before George takes the will is to refine it before the USER MOVE CONFRONT GEORGE is recognized.

However, I have decided to make an exception for this one MOE MOVE, since there is time between when George is confronted and when he gets the new will. A new Moe

decision can be made in that time.

The other anticipating MOE MOVES cause changes that effect the concrete sequences of actions that are recognized as USER MOVES. Thus, there is no time to make a new decision. For example, Moe cannot halt the description of the balcony just after describing the mud in order to decide if the scraped paint should also be described. That decision must be made in advance.

One could argue that I should also make an exception for MOE MOVE 7: DELAY THE AFL REPORT, since there should be time after the report is requested and before it is returned. However, since MOE MOVE 7 is a delayer, its search mechanics are different. In the next chapter, we shall see that the search system has no separate concept for requesting and receiving the AFL report, and therefore that MOE MOVE 7 must remain an anticipating MOE MOVE.

Moe Move 18: “Analyze for Ebullion” Report includes Mention of a Mystery Substance (No Manipulation Cost)

Normally the “Analyze for Ebullion” report contains just the negative result and no other information. When this MOE MOVE is refined, the report is changed to include a reference to a mystery substance found on the fragments. The lab will say it couldn’t match this substance to any “commonly available medicine.” This gives the User incentive to check the medicine cabinets, especially Dunbar’s, since she gave the deceased his tea.

The mystery substance is Loblo, located in Dunbar’s medicine cabinet. When the User finds the Loblo, the system recognizes the event as the USER MOVE LOBLO. Thus, this MOE MOVE serves as a hint for LOBLO. Again, refining this MOE MOVE does not guarantee that the User will find Loblo in the near future, but it should encourage her to. There is no *Manipulation* cost for this MOVE, since either version of the report is plausible.

That concludes the description of the MOE MOVES. Knowing about the MOE MOVES is required for understanding the rest of the chapters on search. The next chapter shows how the search state represents the effects of each of the different types of MOE MOVES described here.

5.3 Creating a Set of MOE MOVES

There are no recipes that tell an artist exactly how to create a wonderful piece of art. Instead, among other things, an artist learns from established examples, draws from personal experience, and reacts to criticism. There is no one right way to do it.

Creating an interactive drama is such an artistic pursuit. The artist must create the settings, characters, and story. The artist must also create abstract models of these elements that can be used by Moe. The process of creating the abstract models is an important part of the artistic process.

By describing aspects of the process I used to create the MOE MOVES for *Tea For Three*, I can illustrate some guidelines that I think are useful for creating a set of MOE MOVES for another interactive drama. These are the techniques that I found successful. Again, this is not a recipe for how to select MOE MOVES. Instead, I hope future artists can use this description for guidance or inspiration.

Techniques
for
creating
MOE
MOVES

Brainstorm First, Select Later

When I started creating the MOE MOVES for *Tea For Three*, I didn't know exactly what MOVES I wanted. For this reason, I started by brainstorming as many MOE MOVES as possible. While brainstorming, I tried to get coverage in two ways. First, I created at least a few MOE MOVES that affected each USER MOVE. Second, I created MOE MOVES that worked through all three refinement channels: the character's minds, the physical world simulation, and the presentation system.

I found this technique useful because it freed me from having to choose the correct MOE MOVES at first. By considering many possible MOVES, I was actually able to understand many possible subtleties of *Tea For Three*. This helped me figure out MOE MOVES with low *Manipulation* costs, for example. After brainstorming, I had a giant sheet of paper with about two hundred potential MOE MOVES.

Choose a target number of Moe Moves, but remain flexible

Two hundred seemed like a large number to me, so I decided I should set a target for how many MOE MOVES I would pick. First, this would help me decide when I was finished. Second, the limit would focus my efforts to pick only the most powerful MOE MOVES.

I considered three separate criteria to determine my target. First, since each MOE MOVE increases the search depth, and therefore the search time, too many MOE MOVES would make the search intractable. For this reason, a small number of MOE MOVES must be chosen. Second, for unexplainable reasons, I found comfort in choosing a number of MOE MOVES close to the number of USER MOVES. Finally, from a resource point of view, I only had enough time to consider a minimal set of MOE MOVES. To design more than an adequate number of MOE MOVES would have been a bad use of my time.

This reasoning led me to choose a target of fifteen to twenty MOE MOVES. When another artist chooses his own target for another interactive drama, he should use his own reasoning.

First choose Causers, Deniers, Delayers. Then choose Powerful Hints and Move Substitutions.

Since I could choose only a limited number of MOE MOVES, I wanted to choose the MOE MOVES that were best able to guide the User's experience. Therefore, I started by choosing causers, deniers, and delayers, waiting until later to select hints and the other types of MOE MOVES.

The most powerful MOE MOVES are the ones that can cause important USER MOVES without much *Manipulation*. In conjunction with causing, the ability to deny or delay USER MOVES is also powerful, because it gives Moe the power to control the order of USER MOVES. Move substitutions are less powerful for controlling long-term structure, but can be good for local structure. Hints are only as powerful as they are effective.

Make sure to get coverage of all important User Moves

Not only did I want a set of individually powerful MOE MOVES, but I wanted a set of MOE MOVES that had coverage, which is the ability to guide all or most aspects of the User's experience. Therefore, I chose MOE MOVES that would be able to affect the order of all important USER MOVES.

Ideally, every one of these MOE MOVES would have been a causers without *Manipulation* costs. As it turned out, I had to choose a variety of types.

Divide the experience in other ways, if necessary

During the selection process, I found it useful to divide *Tea For Three* into five sections, and make sure that I had MOE MOVES for guiding each section. The five sections came from the five topics of thought used by the *Thought Flow* feature of the evaluation function: proving Dunbar is guilty, proving Baxter is guilty, proving George is guilty, figuring out the physical means of the murder, and figuring out the chemical means of death. (See Chapter 3 for the description of *Thought Flow*.)

If I hadn't divided the experience in this way, I might have left out MOE MOVES that were important. If an experience is naturally divided into important parts, an artist should consider each part of the experience separately, in order to make sure each part is guidable.

Prefer characters as the refining medium

I believe that characters are the best mechanism for the refinement of MOE MOVES. (This belief came in part from experience gained while performing a set of live experiments[20].) For this reason, I was very interested in exploring MOE MOVES that used characters in their refinement process. Whenever possible, I chose to use such MOE MOVES.

This turned out to be a very valuable technique. Character MOVES are flexible, powerful, and potentially have very low *Manipulation* costs. An artist should consider using characters whenever possible.

Stop when you have fulfilled you criteria

How does the process of selecting MOE MOVES stop? I stopped when I had achieved my goals. Using only eighteen MOE MOVES, I had coverage of all the major USER MOVES. I had used many causers, deniers, and delayers. And I had many MOE MOVES that used

characters. A few MOE MOVES could have been added or subtracted from my set, but there were no gaping holes or redundancies.

Certainly, the set I chose was not the optimal set, but it is a workable set. In Chapter 8, we see evidence that suggests this set of MOE MOVES can be used to effectively guide *Tea For Three*. Even though these MOE MOVES are not optimal, they were chosen with sensitivity to certain criteria. I hope future artists will be able to take advantage of this explanation in order to make effective choices of their own.

In this chapter we learned about the MOE MOVES for *Tea For Three*, including how they are refined. In the next chapter we learn how the search state represents the process of refining MOE MOVES by representing the important changes to the world, characters, presentation system, and User that happen when MOE MOVES are refined.

Chapter 6

The Search State

The previous chapter described the MOE MOVES of *Tea For Three*. This included a description of how each MOE MOVE should be refined, and its *Manipulation cost*. Although the refiners have not been implemented, examining their design has led to the creation of several categories of MOE MOVES. These include *causers*, *deniers*, *hints*, *move substitutions*, *delayers*, and *future hints*. The last three can be thought of as the *anticipating* versions of the first three.

This chapter describes the search state used by the search to represent the refinement of MOE MOVES and the recognition of USER MOVES. The chapter starts by discussing the motivation for having a search state. Next, each field of the search state is described in detail, including how it represents various USER or MOE MOVES. The final part of the chapter is an extended example of Moe using the search state to consider one complete scenario.

The next chapter describes three search algorithms that use this search state. Recall, the search is used by Moe to decide which MOE MOVE to refine (if any) at some point in the User's experience, in order to guide the User to her destiny. Chapter 8 describes an experiment that provides evidence suggesting that these three search algorithms can be used by Moe to guide an interactive drama effectively.

After you read this chapter, you will understand the search state that Moe uses in its search. You should understand how the search state is an abstraction of the concrete simulated world that represents only those details needed by the abstract adversary search.

6.1 Motivation For Having a Search State

During an experience, the User is interacting in the simulated world with characters as presented by the presentation system. Let's call the actual actions and perceptions of the human User the *concrete* experience of the User.

Certain of her interactions with the characters and physical world have been recognized by Moe as USER MOVES (see Chapter 2). Let's call the sequence of USER MOVES that has been recognized the *abstract* experience of the User.

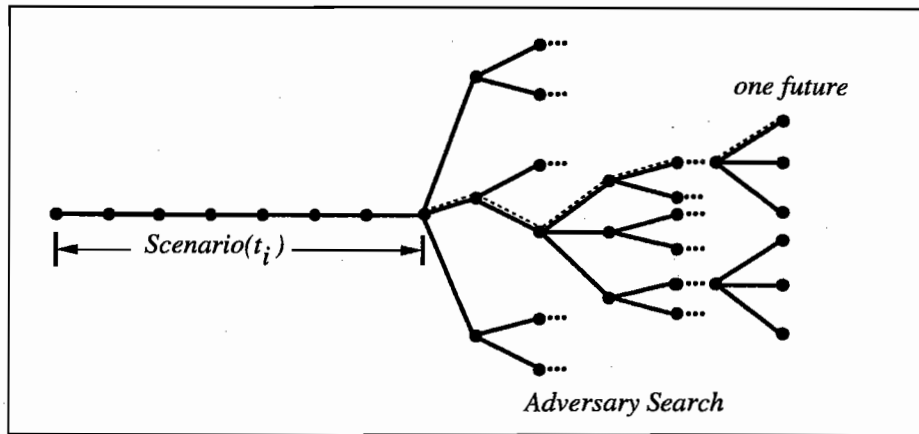


FIGURE 6.1: The History and Projected Future

Moe refines MOE MOVES (described in the previous chapter) to change aspects of the concrete simulation in order to guide the User's experience. If Moe has refined any MOE MOVES during the User's experience, they are included in the *abstract* description of the experience as well.

The sequence of USER and MOE MOVES that abstractly describes the experience of the User is called a *scenario*. Just to be clear, many different concrete experiences could be described by the same *scenario*. A given *scenario*, therefore, represents many concrete User experiences.

At any given time during the experience of the User, Moe must decide whether refining a MOE MOVE is beneficial to the experience of the User. Moe uses adversary search to make this decision.

In order to limit the size of the search, the search program uses an abstract representation. It does not consider the concrete experience of the User, because that would involve too many details and, therefore, too many variations to search over, making the search intractable.

Let t_i be the time that Moe initiates a search. Recall that Moe is initiating the search because it needs to determine which of the currently available MOE MOVES to refine, if any. Before t_i , the User has had a certain concrete experience, and Moe has refined certain MOE MOVES. Therefore, there is a sequence of USER and MOE MOVES that represents the experience of the User so far. Call this $scenario(t_i)$. $scenario(t_i)$ is the input that Moe uses to decide which MOE MOVE to refine at t_i , if any.

As I mentioned, the search program is trying to decide if a given MOE MOVE would increase the expected quality of the User's experience. Conceptually, it does this by projecting all possible futures of the experience from the time t_i . That is, it projects all future experiences starting with $scenario(t_i)$.

Since the search program is working in the abstract, this means taking $scenario(t_i)$ and projected from it all possible (legal) scenarios, which are sequences of USER and MOE MOVES.

adversary search

uses abstract representation

scenario(t_i)

searches through all possible futures from t_i

Figure 6.1 shows this process. At the left is what has already happened to the User, represented by $scenario(t_i)$. The right portion of the diagram shows that the search is expanding the entire tree of possible future scenarios that start with $scenario(t_i)$. As we mentioned in Chapter 1, the search calls the evaluation function with each possible future scenario, backs up the values using a game-tree paradigm, and thus decides the MOE MOVE, if any, that maximizes the expected value of the User's experience. The dotted scenario represents on possible future experience of the User. See Chapter 7 for more details.

Since the search program is using abstract sequences of USER and MOE MOVES to project the future possibilities of the User's experience, the program must somehow be able to model their effects. To do this, the search program stores relevant information in a *search state*. The search state implicitly provides a search space over which the search is performed.

The search state represents the effects of refining MOE MOVES and recognizing USER MOVES. As you will see in the next section, the search state does not represent all the details of the refinement or recognition, but rather the important information. The search state represents the important changes to the world, characters, presentation system, and the User's mental state that happen when MOE MOVES are refined or must have happened in order for USER MOVES to be recognized. For *anticipating* MOE MOVES, this information will include the future behavior of the world, characters, or presentation system.

Notice this search state is different from the one employed by a basic chess-playing program. A basic chess-playing program uses the board position as its search state. Thus, there is an isomorphism between the real board and the program's image of the board. Moe's search state contains abstracted descriptions of the "real world." Moe's search state must store only relevant information.

When a search is initiated, a new search state is created. We denote the search state by **SS**. As we shall see, **SS** has a number of fields that store information relevant to the search. The initial values of the fields of **SS** (at t_i) are based on $scenario(t_i)$. As the search begins, **SS** does not yet represent any projected USER or MOE MOVES.

As the search proceeds, different USER and MOE MOVES are *made*, which means the search is pretending that each MOVE is either *recognized* or *refined* next in the future of the User's experience.¹ As each MOVE is *made*, its effect on **SS** is calculated, updating **SS**.

Thus, at any point during the search, **SS** contains information from the MOVES in $scenario(t_i)$ (the USER and MOE MOVES which have been *recognized* and *refined* during the concrete experience), as well as information from the projected USER and MOE MOVES (those *made* by the search).

The next section describes Moe's search state, **SS**, in detail. Section 6.3 contains an example of how the search state is used by the search as it projects *one* possible future scenario of MOE and USER MOVES.

¹Actually, as you will see, sometimes a USER MOVE that is made gets delayed for a time.

- history** ordered list of USER MOVES and MOE MOVES that were recognized or refined before the search, followed by those that have happened during the search
- usedUserMoves** the set of USER MOVES contained in **SS.history**
- usedMoeMoves** the set of MOE MOVES contained in **SS.history**
- lastUserMove** the last USER MOVE in **SS.history**
- mergerThenNotepad** records whether MERGER happened before NOTEPAD in **SS.history**
- candidateUserMoves** an ordered list of USER MOVES that are about to be added to **SS.history**. Candidates might not be added, depending on other aspects of the search state.
- illegalUserMoves** set of USER MOVES that are illegal because of previously refined or made MOE MOVES
- moveSubstitutions** the set of currently active move substitutions. For example, the sequence MUD,PAINT is substituted for the single USER MOVE, MUD
- movesWhichWillBeDelayed** set of USER MOVES that will be delayed in the future. They are put on this list because of MOE MOVES
- movesWhichHaveBeenDelayed** set of USER MOVES that have been delayed. Previously, these MOVES would have been in **SS.movesWhichWillBeDelayed**
- moveMultipliers** a set of currently active hints. Each hint is for a specific USER MOVE. Each has an integer value, which says how much more likely this USER MOVE is to happen than is normally the case; and a timeout, which is how much longer this hint is active. For example, (MUD: value 2, timeout 3) means MUD is twice as likely as normal for the next three USER MOVES
- futureHints** the set of hints created by anticipating MOE MOVES. Each has a trigger USER MOVE which tells the system when the hint should move to **SS.moveMultipliers**
- edges** the set of edges that show the precedence relationships between USER MOVES

FIGURE 6.2: The Basic Search State

6.2 The Search State

Figure 6.2 shows the fields of the search state. In **boldface** is the name of the field. For each, there is a description of the field. The rest of this section describes each of the fields in more detail.

6.2.1 SS.history

SS.history is a straightforward field. It is used to keep a record of which USER MOVES and MOE MOVES have *happened* in the experience. A MOVE has *happened* if it has been

recognized or refined before the search, or if it has been made during the search. Actually, sometimes USER MOVES are delayed for a time when they are made, and thus they won't be part of **SS.history** until later in the search. We don't say those USER MOVES have *happened* until they are added to **SS.history**. (For details, see **SS.candidateUserMoves**, below.)

When a search is initiated, a new search state is created. The value of **SS.history** is set to be the sequence of USER and MOE MOVES which have been *recognized* and *refined* during the experience so far. We have previously called this *scenario*(t_i).

During the search, as each MOE MOVE is made, it is added to **SS.history**. As each USER MOVE is made, it is added to **SS.candidateUserMoves**, and usually subsequently added to **SS.history**. Thus, at any point during the search, **SS.history** contains the MOVES of *scenario*(t_i) followed by (with some minor variation) the MOVES made during the search.

6.2.2 **SS**.{usedUserMoves, usedMoeMoves, lastUserMove, mergerThenNotepad}

These four fields are derived directly from **SS.history**. They are computed by the search for efficiency, since this information is accessed frequently. Chapter 7 explains another reason for keeping this information distinct from **SS.history**.

SS.usedUserMoves is the set of USER MOVES in **SS.history**, while **SS.usedMoeMoves** is the set of MOE MOVES that are in **SS.history**. **SS.lastUserMove** stores the last USER MOVE in **SS.history**, and **SS.mergerThenNotepad** stores simply whether MERGER or NOTEPAD came first in **SS.history**.

6.2.3 **SS.candidateUserMoves**

SS.candidateUserMoves is an ordered list of USER MOVES which are supposed to come next during the search. **SS.candidateUserMoves** is a mechanism of the search. It does not represent any aspect of the User's experience. Usually, there is just one candidate USER MOVE, and it is directly added to **SS.history**.

The reason the search state has **SS.candidateUserMoves** is that previously refined or made MOE MOVES can change the behavior of the search with respect to certain USER MOVES. These USER MOVES will not be added directly to **SS.history**. For example, a USER MOVE might be delayed, or a sequence of USER MOVES might need to be substituted for a given USER MOVE. We shall see more detailed examples later in this section.

USER MOVES are added to **SS.candidateUserMoves** in several different ways. The two most common are through the normal process of projecting legal USER MOVES during the search, and through *causing* MOE MOVES. We shall see examples of both in Section 6.3.

Figure 6.3 shows which MOE MOVES directly add USER MOVES to **SS.candidateUserMoves**. When any of those MOE MOVES is made, the USER MOVE it causes is added to **SS.candidateUserMoves**.

info
from history
stored for
efficiency

MOE MOVE	USER MOVE it causes
2: DUNBAR CONFRONTS USER	CONFRONT DUNBAR
3: DUNBAR CONFESSES TO USER	TICKET/AFFAIR
4: BAXTER EXPLAINS MERGER	MERGER
5: GEORGE CONFRONTS USER	CONFRONT GEORGE
6: GEORGE IS CAUGHT	CATCH GEORGE
8: RETURN THE AFL REPORT	AFL

FIGURE 6.3: Causing MOE MOVES that add to **SS.candidateUserMoves**

6.2.4 **SS.illegalUserMoves**

SS.illegalUserMoves introduces the first good example of the search state's ability to represent the concrete effects of MOE MOVES. In this case, **SS.illegalUserMoves** stores the effects of *denying* MOE MOVES.

When deniers are refined, they make changes to the physical world, characters, and presentation system so that specific events in the world are no longer possible. Thus, they deny the User the opportunity to take certain actions that are recognized as USER MOVES. For example, when MOE MOVE 1: REMOVE DUNBAR is refined, it removes Dunbar from the world, denying the User the opportunity to interact with Dunbar. Interacting with Dunbar is required for the recognition of CONFRONT DUNBAR and TICKET/AFFAIR.

The search state must represent these changes to the world, characters, or presentation. However, since the search is working in the abstract, the details of the refinement are not considered. Therefore, the search state does not need to represent the details. The search state needs to represent only the fact that the User has been denied the opportunity to take certain actions that are recognized as USER MOVES. The search state represents this fact by putting the denied USER MOVE into **SS.illegalUserMoves**.

Notice again how this is different from a basic chess program. The search has distilled a huge amount of concrete details that happen in the concrete experience into a single change in the search state. This is what it means for the search to be abstract.

SS.illegalUserMoves does not contain every non-legal USER MOVE. USER MOVES may be illegal for several reasons: they've happened already, they cannot logically happen because other USER MOVES must happen first, or they are in the set **SS.illegalUserMoves** because they have been denied. As we shall see in the section on Legal MOVE Generation, **SS.illegalUserMoves** filters the set of potentially legal USER MOVES by removing its members.

There are three MOE MOVES that add USER MOVES to **SS.illegalUserMoves**, while one MOE MOVE can remove one of the previously added USER MOVES. First, MOE MOVE 1: REMOVE DUNBAR adds the USER MOVE CONFRONT DUNBAR to **SS.illegalUserMoves**. It doesn't add TICKET/AFFAIR because TICKET/AFFAIR must follow CONFRONT DUNBAR and thus TICKET/AFFAIR is implicitly denied.

MOE MOVE	(Trigger→Substituted Sequence)
9: COMBINE AFE AND AFL REPORTS	(AFL→AFE,AFL)
10: DESCRIBE HOLES AND FRAGMENTS TOGETHER	(HOLES→HOLES,FRAGS)
15: DESCRIBE MUD AND PAINT TOGETHER	(MUD→MUD,PAINT)
16: MOVE FRAGMENTS TO SHED	(LADDER→LADDER,FRAGS)

FIGURE 6.4: MOE MOVES that create active move substitutions

Second, MOE MOVE 17: GEORGE DOESN'T TAKE WILL adds CATCH GEORGE and FOCUS to **SS.illegalUserMoves**. Subsequently, FOCUS can be removed by making MOE MOVE 6: GEORGE IS CAUGHT. Thus, MOE MOVE 6 not only has the effect of adding CATCH GEORGE to **SS.candidateUserMoves**, but it also removes FOCUS from **SS.illegalUserMoves**.

Third, MOE MOVE 16: FRAGMENTS MOVE TO SHED adds FRAGS to **SS.illegalUserMoves**. This might seem strange, but recall that MOE MOVE 16 also creates the move substitution (LADDER→LADDER,FRAGS). Later in the search, FRAGS will be added to **SS.candidateUserMoves** as a result of that move substitution. Therefore, FRAGS must become illegal.

6.2.5 SS.moveSubstitutions

SS.moveSubstitutions introduces the search state's ability to store the effects of *anticipating* MOE MOVES.

Recall that when an anticipating MOE MOVE is refined, it makes changes the world, characters, or presentation that alter a concrete sequence of events that happen in the *future*. In this way, an anticipating MOE MOVE changes the concrete sequence of events that will be recognized, sometime in the future, as a USER MOVE.

For example, MOE MOVE 9: COMBINE AFE AND AFL changes the behavior of the police labs so that when, in the future, the User requests the police labs to analyze the fragment for Loblo, the police also do an analysis for Ebullion and create a joint report.

The search state must be able to represent this change to the future behavior of the world. One such type of change is a MOVE SUBSTITUTION. MOE MOVE 9, above, is a move substitution because it makes concrete changes to the world that result (sometime in the future) in two USER MOVES being recognized instead of just one. Specifically, the sequence of USER MOVES AFE,AFL will be recognized instead of just AFL.

In the search, this is represented by substituting a sequence of two USER MOVES for a single USER MOVE. (AFL→AFE,AFL) is the move substitution that represents the refinement of MOE MOVE 9. The "→" indicates the substitution to be made. Thus, in the search, AFE,AFL is substituted for AFL. Figure 6.4 shows the four move substitutions used in *Tea For Three*, and which MOE MOVES cause them.

In the search, move substitutions have their effect when the search is choosing to change a candidate USER MOVE into an actual USER MOVE. That is, moving it from

SS.candidateUserMoves to **SS.history**. If the next candidate USER MOVE is to be substituted for, then the substitution happens at that point.

For example, suppose (AFL→AFE,AFL) were an active move substitution and AFL were the next candidate USER MOVE. Without the move substitution, AFL would simply be added to **SS.history**. With the move substitution, the search removes AFL from **SS.candidateUserMoves** and puts the sequence AFE,AFL in its place.

At that point, AFL is again the next candidate USER MOVE, and so it will be added to **SS.history**. The reason AFL isn't substituted for again is that after (AFL→AFE,AFL) makes its substitution, it is no longer considered active.

After AFL is added to **SS.history**, AFE becomes the next candidate USER MOVE, and thus it is also added to **SS.history**. The result is that when AFL happens, the move substitution causes the sequence AFE,AFL to happen instead.

Move substitutions are an example of why **SS.candidateUserMoves** is needed. If AFL were immediately entered into **SS.history**, the search would have no opportunity to make the move substitution. By using **SS.candidateUserMoves** the system has an opportunity to make the necessary changes to the sequence of USER MOVES.

SS.moveSubstitutions is the set of currently active move substitutions.

6.2.6 **SS.movesWhichWillBeDelayed, SS.movesWhichHaveBeenDelayed**

SS.movesWhichWillBeDelayed represents the future effects of another type of *anticipating* MOVE MOVE: *delayers*. In *Tea For Three* there is only one delayer: MOE MOVE 7: DELAY THE AFL REPORT. When MOE MOVE 7 is refined it directs the police to change their future behavior. In the future, when the User requests the labs to analyze the fragments for Loblo, the police labs will not immediately return the report. Instead, they will keep it, delaying its return to the User. The search represents this future action of the labs by putting AFL into **SS.movesWhichWillBeDelayed**.

To see how this works in the search, assume that AFL has been delayed, and thus AFL is in **SS.movesWhichWillBeDelayed**. When AFL becomes the next candidate USER MOVE (comes to the front of **SS.candidateUserMoves**), this represents the moment in the concrete world when the User requests the police to analyze the fragments for Loblo.

Since AFL is in **SS.movesWhichWillBeDelayed**, it is not directly moved from **SS.candidateUserMoves** to **SS.history**. Instead, it is removed from **SS.candidateUserMoves** and **SS.movesWhichWillBeDelayed** and added to **SS.movesWhichHaveBeenDelayed**.

The time AFL is in **SS.movesWhichHaveBeenDelayed** represents the time that the police labs are keeping the report. AFL stays there until MOE MOVE 8: RETURN THE AFL REPORT is made. This represents the moment when the police labs are directed to return the report to the User. Receiving the report should be recognized as AFL. Thus, AFL is removed from **SS.movesWhichHaveBeenDelayed** and added to **SS.candidateUserMoves**, from where it should move to **SS.history**.

6.2.7 SS.moveMultipliers

SS.moveMultipliers is a structure that represents the effects of *hints* in the search. Recall, a hint is a MOE MOVE that, when refined, changes aspects of the world, characters, or presentation in order to encourage the User to experience a sequence of concrete events that will be recognized as a USER MOVE.

When a hint is refined, a small part of the human User's actual mind is changed in response to what she sees or hears. Thus, when **SS.moveMultipliers** represents what the User is likely to do, it is actually representing one small part of the human User's mind.

For example, consider the hint MOE MOVE 12: GEORGE GOES ON BALCONY. When MOE MOVE 12 is refined, George goes to the balcony and makes eye contact with the User. When the User sees George's actions, this will encourage her (alter her actual mind) to go to the balcony, where she will see the muddy footprints, which will be recognized as USER MOVE MUD. My judgement tells me that when the User sees George's actions, her mind has been changed in such a way that she is (roughly) twice as likely to go to the balcony as she is to take another course of action. (Future work in this area is to determine the exact effects of hints.)

It is plausible that the User might do something significant on her way to the balcony. However, if she takes two significant actions, I no longer consider the User to be more likely to go to the balcony than elsewhere. Therefore, I believe this *hint* persists for the period of time in the concrete world that comprise the next two USER MOVES. Thus, after the second different USER MOVE is recognized, I consider the hint to be a failure.

To represent the effect of hints in the search state, every USER MOVE is represented in **SS.moveMultipliers** by a *value* and a *timeout*. *value* represents how likely it is that this USER MOVE will be recognized next, compared to the average USER MOVE. For example, if MUD has a *value* of 2, then that represents a concrete experience where MUD, compared to the average MOVE, is twice as likely to be recognized. The default *value* for USER MOVES is 1.

More than one USER MOVE may have a *value* greater than one. For example, MUD can have *value* 3 at the same time that LADDER has *value* 2. In that case, MUD is three times as likely to be recognized next, while LADDER is twice as likely to be recognized next, compared to the average USER MOVE.

timeout represents how much longer the hint is valid. *timeout*, as I described above, is measured in number of USER MOVES that have been recognized. The default *timeout* is 0, which means either the hint has failed, or there is no hint. Whenever the *timeout* is zero, the *value* becomes one, the default.

Non-zero *timeouts* indicate how much longer the hint is valid. Every time any USER MOVE is recognized (which is represented by adding it to **SS.history**), the *timeout* of every hint is decremented, to a minimum value of zero. When the *timeout* of a given USER MOVE reaches zero, this means the hint is no longer having an effect, so *value* changes back to 1.

As an example, suppose MUD's entry in **SS.moveMultipliers** has a *value* of 2 and a *timeout* of 2. After, say, USER MOVE LADDER happens, MUD's entry will have *value* 2, and a decremented *timeout* of 1. After another USER MOVE (say, HOLES) happens MUD's

MOE MOVE	USER MOVE	Value	Timeout
11: GEORGE SHOOTS SKEET	MUD	40	1
12: GEORGE GOES ON BALCONY	MUD	2	2
16: MAKE LADDER TRACKS	LADDER	16	1

FIGURE 6.5: MOE MOVES (*hints*) that immediately change **SS.moveMultipliers**

entry returns to the default *value* 1, since its *timeout* dropped to 0, indicating this hint is no longer having an effect.

Figure 6.5 shows the three MOE MOVES that immediately create hints, and thus immediately change the entries in **SS.moveMultipliers**. I say immediately because the next section describes the set of anticipating hints, which will affect **SS.moveMultipliers** some time in the future, after they are made.

6.2.8 SS.futureHints

SS.futureHints represents the changes in the world, characters, or presentation that will result in hints being created some time in the future. These changes happen when an anticipating MOE MOVE hint is refined. Thus, when the search makes such a MOE MOVE, these changes must be represented.

For example, when MOE MOVE 13: GEORGE CONFESSES SECRET ROOM is refined, George is directed to give the User the location of the secret room when he is eventually caught getting rid of the will. The effect of this confession is to encourage the User to go to the secret room and find the Focus Scandal Papers.

In the concrete world, this means that when George confesses, the concrete events that will be recognized as the USER MOVE FOCUS become (in the artist's mind) three times as likely for the period of time during which the next two USER MOVES are recognized. The search would represent this information as: after CATCH GEORGE, FOCUS becomes three times as likely for the next two USER MOVES.

Thus, every anticipating MOE MOVE hint has an associated structure, a *future hint*, that represents the changes it makes to the concrete world. Recall, a hint consists of a timeout and value for a given USER MOVE. For example, MUD is twice as likely (value 2) for the next two USER MOVES (timeout 2). A *future hint* consists of a hint and a triggering USER MOVE, which causes the hint to become active.

For example, the *future hint* for MOE MOVE 13 has the trigger (CATCH GEORGE), and the hint (USER MOVE FOCUS: value 3, timeout 2). Figure 6.6 shows the two MOE MOVES in *Tea For Three* that add *future hints* to **SS.futureHints**.

In the search, when the trigger USER MOVE of a *future hint* is added to **SS.history**, the hint portion of the *future hint* is added to **SS.moveMultipliers**, and the *future hint* is removed from **SS.futureHints**.

MOE MOVE	Trigger	USER MOVE	Value	Timeout
13: GEORGE CONFESSES SECRET ROOM	CATCH GEORGE	FOCUS	3	2
18: MENTION MEDICINE	AFE	LOBLO	2	2

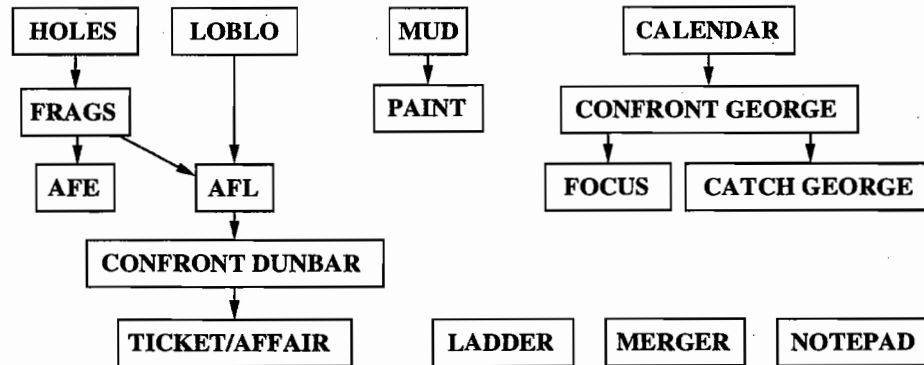
FIGURE 6.6: MOE MOVES that go into **SS.futureHints**

FIGURE 6.7: DAG of USER MOVES and precedence relations

To continue our above example, suppose the following *future hint* were in **SS.futureHints**:

(Trigger: CATCH GEORGE; Hint: FOCUS, value = 3, timeout = 2)

When CATCH GEORGE is added to **SS.history**, the entry for FOCUS in **SS.moveMultipliers** is updated to have a value of 3 and a timeout of 2. In addition, the *future hint* is removed from **SS.futureHints**.

6.2.9 SS.edges

SS.edges contains the set of edges that show the precedence relationships between pairs of USER MOVES. This is the DAG of USER MOVES from Chapter 2. Edges are used for Legal MOVE Generation, which will be explained in the next chapter.

Figure 6.7 shows the default set of edges. The USER MOVE at the head of the arrow must happen after the USER MOVE at the tail of the arrow. Thus, PAINT must follow MUD, both AFE and AFL must follow FRAGS, but MERGER may happen at any time.

MOE MOVE 16: MOVE FRAGMENTS TO SHED is the only MOE MOVE which changes **SS.edges**. When MOE MOVE 16 is refined, it removes the edge connecting USER MOVE HOLES to USER MOVE FRAGS. This edge was there to represent that the User must find the holes in the ground before she can find the ceramic fragments in the holes. When the fragments are moved to the shed, they are no longer related to the holes, so the link is incorrect, and thus removed.

One might think changing **SS.edges** might cause the evaluation function to stop working. In general this is true, since the evaluation function exploits the fact that the scenarios it rates must be consistent with the DAG. However, by careful design, this change does not affect *Tea For Three's* evaluation function.

6.3 A Search State Example

The previous section explains how the search state represents the actions and mind of the User, or changes to the concrete world, characters, and presentation. During a search, refining a MOE MOVE in the concrete world is represented by making a MOE MOVE in the search. The effects of the refinement are represented in the various fields of the search state. Likewise, the effects of recognition are represented. For each search state field, we learned what information it represents, and how that information is updated by making MOE and USER MOVES.

This section contains an extended example showing the search state being used in a search that is initiated at the beginning of a User's experience. That is, before any MOE MOVES have been refined or any USER MOVES have been recognized.

In the actual search, Moe would be considering all legal USER and MOE MOVES at every point. In this example, we are going to consider only one future scenario. At each point in the search, this example examines either one USER MOVE or one MOE MOVE. As each USER or MOE MOVE is considered, I show the changes it makes to the search state.

Thus, although a search will normally expand all future scenarios, in this example we will see only one future scenario. This example illustrates most of the mechanisms used in processing the search state.

The initial search state looks like this. Notice, curly brackets, “{}” represent a set; whereas square brackets “[]” represent an ordered list.

```

SS.history [ ]
SS.usedMoeMoves { }
SS.usedUserMoves { }
SS.lastUserMove none
SS.mergerThenNotepad no
SS.candidateUserMoves [ ]
SS.illegalUserMoves { }
SS.moveSubstitutions { }
SS.movesWhichWillBeDelayed { }
SS.movesWhichHaveBeenDelayed { }
SS.moveMultipliers all USER MOVES have Move Multipliers with weight 1 and
timeout 0.
SS.futureHints { }
SS.edges { HOLES→FRAGS, FRAGS→AFE, FRAGS→AFL, LOBLO→AFL,
AFL→CONFRONT DUNBAR, CONFRONT DUNBAR→TICKET/AFFAIR,
MUD→PAINT, CALENDAR→CONFRONT GEORGE,
CONFRONT GEORGE→FOCUS, CONFRONT GEORGE→CATCH GEORGE }

```

In the example that follows, I will show only the parts of the search state that have changed from the initial values given above. The steps in the description are numbered. These numbers roughly (but not exactly) correspond to the number of USER or MOE MOVE considered by the search so far.

1. The first MOVE of the scenario considered here is MOE MOVE 14: DESCRIBE THE MUD AND THE SCRAPED PAINT TOGETHER. Recall that Moe has the option of making a MOE MOVE at any time during the search, including before any USER MOVES are considered. To be clear, at any Moe decision point, Moe considers making all legal MOE MOVES, and making no MOE MOVE. This example shows a scenario where Moe considers making MOE MOVE 14 at the beginning.

By considering MOE MOVE 14, the search is considering changing the presentation system so that it will describe the scraped paint on the balcony railing whenever it describes the muddy footprints. To represent this, the search state is changed to indicate that when Moe considers the USER MOVE MUD, it will instead consider the sequence MUD,PAINT. As we can see, this information is stored in **SS.moveSubstitutions**:

```

SS.history [ 14 ]
SS.usedMoeMoves { 14 }
SS.moveSubstitutions { (MUD→MUD,PAINT) }

```

Notice also that **SS.history** and **SS.usedMoeMoves** have been updated to represent the fact that the search is considering making MOE MOVE 14.

2. Next, I show one possible MOE MOVE that the search might consider after making MOE MOVE 14. Again, the search would really be considering all legal MOE MOVES and making no MOE MOVE, but this example, again, shows just one future scenario.

The next MOE MOVE is MOE MOVE 16: FRAGMENTS MOVE TO SHED. When MOE MOVE 16 is refined in the simulated world, the ceramic fragments are moved from the holes under the balcony to the shed, next to the ladder. That means that when the User finds the ladder in the shed, she will also find the fragments, since they are in plain sight. As you can see, this is also represented in **SS.moveSubstitutions**:

```

SS.history [ 14, 16 ]
SS.usedMoeMoves { 14, 16 }
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT), (LADDER→LADDER,FRAGS) }

```

As above, **SS.history** and **SS.usedMoeMoves** have been updated.

When MOE MOVE 16 is refined, it has another effect. It removes the temporal connection between finding the holes and finding the fragments. This was previously represented by an edge in **SS.edges**. Thus, to represent this change, the edge (HOLES→FRAGS) is removed from **SS.edges**. For brevity, I will not show this change, since **SS.edges** will never change again in this example.

This change is also represented by putting FRAGS into **SS.illegalUserMoves**, since the search model is that this USER MOVE can only happen (in the search) as part of the move substitution.

After making MOE MOVE 16, the search would normally be considering making all next legal MOE MOVES or making no MOE MOVE. As you will see below, this example shows the case where Moe considers doing nothing at this point. To imagine doing nothing, the search must consider that the User has done something that has been recognized as a USER MOVE.

When a USER MOVE happens in the search, it represents the concrete events in the world that will be recognized as that USER MOVE. When a USER MOVE happens, the search state is updated to represent these changes to the world and User.

Of course, the search is not considering all the concrete details that might be recognized as the USER MOVE. Instead it is pretending that some sequence (unknown to the search)

of concrete events has been recognized as the USER MOVE. During this example, I will suggest a set of concrete events that would be recognized as a given USER MOVE. These will be written in *italics*.

3. After making MOE MOVE 16, suppose a USER MOVE is next made.

At this point in the search, the search considers all legal USER MOVES: MUD, LADDER, HOLES, LOBLO, CALENDAR, NOTEPAD, and MERGER. Suppose USER MOVE HOLES is considered. *The first thing the User does is go out behind the house and find the holes underneath the balcony. Recall from the previous MOE MOVE that the fragments are now in the shed, so the User couldn't find them here.*

When a USER MOVE is made, it is added to **SS.candidateUserMoves**:

```

SS.history [ 14, 16 ]
SS.usedMoeMoves { 14, 16 }
SS.candidateUserMoves [ HOLES ]
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT), (LADDER→LADDER,FRAGS) }

```

Because no part of the search mechanism prevents it, HOLES is removed from **SS.candidateUserMoves** and added to **SS.history**. **SS.lastUserMove** is also updated.

```

SS.history [ 14, 16, HOLES ]
SS.usedMoeMoves { 14, 16 }
SS.usedUserMoves { HOLES }
SS.lastUserMove HOLES
SS.candidateUserMoves []
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD, PAINT), (LADDER→LADDER, FRAGS) }

```

4. After making USER MOVE HOLES, the search, as usual, considers making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making MOE MOVE 12: GEORGE GOES ON BALCONY.

MOE MOVE 12 is a hint for the USER MOVE MUD. Thus, the following hint will be added to **SS.moveMultipliers**: (MUD: value 2, timeout 2)

As we shall see in Chapter 7, this means that the next two times the search considers the legal MOVES of the User, the backup mechanism will doubly count the result of the User doing MUD. However, for this example, we are not considering the backup process.

As usual, when MOE MOVE 12 is made, **SS.history** and **SS.usedMoeMoves** are both updated. As you can see, **SS.moveMultipliers** has also been updated.

```

SS.history [ 14, 16, HOLES, 12 ]
SS.usedMoeMoves { 12, 14, 16 }
SS.usedUserMoves { HOLES }
SS.lastUserMove HOLES
SS.candidateUserMoves []
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT), (LADDER→LADDER,FRAGS) }
SS.moveMultipliers { (MUD: value 2, timeout 2) }

```

5. After making MOE MOVE 12, consider the path where the search makes MOE MOVE 15: CREATE LADDER TRACKS.

MOE MOVE 15 is a hint like MOE MOVE 12, above. It adds another non-default entry to **SS.moveMultipliers**, as well as updating **SS.history** and **SS.usedMoeMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15 ]
SS.usedMoeMoves { 12, 14, 15, 16 }
SS.usedUserMoves { HOLES }
SS.lastUserMove HOLES
SS.candidateUserMoves []
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT), (LADDER→LADDER,FRAGS) }
SS.moveMultipliers { (MUD: value 2, timeout 2), (LADDER: value 16, timeout
1) }

```

This search state represents MUD being twice as likely to be recognized, while LADDER is sixteen times as likely. Again, as we shall see in Chapter 7, this information is used when the search backs up the values at a User search node.

Notice that because no new USER MOVE has happened (added to **SS.history**), the timeouts of MUD's move multiplier (in **SS.moveMultipliers**) hasn't been decreased by one.

6. After making MOE MOVE 15, suppose a USER MOVE is next made.

The legal USER MOVES are MUD, LADDER, LOBLO, CALENDAR, NOTEPAD, and MERGER. Suppose USER MOVE LADDER is considered. *The User leaves the area underneath the balcony and heads back to the house. On the way, she enters the shed and finds the ladder.*

Since LADDER is made, it is added to **SS.candidateUserMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15 ]
SS.usedMoeMoves { 12, 14, 15, 16 }
SS.usedUserMoves { HOLES }
SS.lastUserMove HOLES
SS.candidateUserMoves [ LADDER ]
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT), (LADDER→LADDER,FRAGS) }
SS.moveMultipliers { (MUD: value 2, timeout 2), (LADDER: value 16, timeout
  1) }

```

Because the move substitution (LADDER→LADDER,FRAGS) is in **SS.moveSubstitutions**, LADDER is not immediately added to **SS.history**. Instead, three things happen: LADDER is removed from **SS.candidateUserMoves**, the sequence LADDER,FRAGS is put in its place, and the move substitution is removed from **SS.moveSubstitutions**:

```

SS.history [ 14, 16, HOLES, 12, 15 ]
SS.usedMoeMoves { 12, 14, 15, 16 }
SS.usedUserMoves { HOLES }
SS.lastUserMove HOLES
SS.candidateUserMoves [ LADDER, FRAGS ]
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.moveMultipliers { (MUD: value 2, timeout 2), (LADDER: value 16, timeout
  1) }

```

At this point, LADDER is added to **SS.history**, and removed from **SS.candidateUserMoves**. LADDER is also added to **SS.usedUserMoves**, and becomes the value of **SS.lastUserMove**. Further, since LADDER has happened, the entry for LADDER in **SS.moveMultipliers** reverts back to the default, which is value 1 and timeout 0. Also, since a USER MOVE has happened, the timeouts of all USER MOVES in **SS.moveMultipliers** are decremented (unless already zero). The new search state looks (without showing the default entries) like this:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER ]
SS.usedMoeMoves { 12, 14, 15, 16 }
SS.usedUserMoves { LADDER, HOLES }
SS.lastUserMove LADDER
SS.candidateUserMoves [ FRAGS ]
SS.illegalUserMoves { FRAGS }
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.moveMultipliers { (MUD: value 2, timeout 1) }

```

7. Because no part of the search state affects it, FRAGS is removed from **SS.candidateUserMoves** and added to **SS.history**. At this point FRAGS is removed from **SS.illegalUserMoves**. Also, since another USER MOVE has happened, the timeouts of all other USER MOVES in **SS.moveMultipliers** are decreased by one. Thus, MUD's entry now has timeout zero, so its value reverts back to the default. By changing MUD's multiplier value to one, the search state now represents the case where the User is not more likely to be choosing MUD next.

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS ]
SS.usedMoeMoves { 12, 14, 15, 16 }
SS.usedUserMoves { LADDER, HOLES, FRAGS }
SS.lastUserMove FRAGS
SS.candidateUserMoves []
SS.moveSubstitutions { (MUD→MUD,PAINT) }

```

8. After making USER MOVE FRAGS, the search, as usual, considers making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making MOE MOVE 18: REPORT FROM AFE INCLUDES MENTION OF A MYSTERY MEDICINE at this point.

Notice that the search didn't consider a MOE MOVE between LADDER and FRAGS. This is because the search's model is that those USER MOVES are recognized at the same time. Therefore, there is no time to refine a MOE MOVE in between.

In the concrete experience, the report that is normally returned says there is no excess Ebullion on the ceramic fragment. When MOE MOVE 18 is refined, the police labs are changed, so that when they eventually prepare the report, they augment it to include mention of some mystery substance (possibly a medicine) that they were unable to identify.

This should encourage the User to find this mystery substance, which should make the finding the Loblo more likely. Finding the Loblo is recognized as the User Move LOBLO.

Thus, by making this MOE MOVE, the search is pretending that it is directing the police labs to change their future behavior. This is represented in the search state by adding the following future hint to **SS.futureHints**:

```
(trigger:AFE; hint:LOBLO, value 2, timeout 2)
```

As we shall see, when AFE is added to **SS.history**, the hint (LOBLO: value 2, timeout 2) will be added to **SS.moveMultipliers**.

In addition to adding a future hint, MOE MOVE 18 is added to **SS.usedMoeMoves** and **SS.history**:

```
SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18 ]
SS.usedMoeMoves { 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS }
SS.lastUserMove { FRAGS }
SS.candidateUserMoves []
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.futureHints { (trigger:AFE; hint:LOBLO, value 2, timeout 2) }
```

9. After making MOE MOVE 18, the search, as usual, considers making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making no MOE MOVE at this point, which means a USER MOVE is next made.

At this point in the search, the search considers all legal USER MOVES: MUD, AFE, LOBLO, CALENDAR, NOTEPAD, and MERGER. This example shows the path where the search considers USER MOVE AFE at this point. *The User requests that the lab analyze the ceramic fragments for Ebullion. A report is returned indicating that there is no Ebullion on the fragment. The report has been augmented by the police labs (as directed by the past refinement of MOE MOVE 18) to include mention of some mystery medicine that the lab was unable to identify.*

To make AFE, the search adds it to **SS.candidateUserMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18 ]
SS.usedMoeMoves { 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS }
SS.lastUserMove { FRAGS }
SS.candidateUserMoves [ AFE ]
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.futureHints { (trigger:AFE; hint:LOBLO, value 2, timeout 2) }

```

The search then removes AFE from **SS.candidateUserMoves** and adds it to **SS.history** and **SS.usedUserMoves**. AFE also becomes the value of **SS.lastUserMove**. Because AFE is the trigger of a future hint, this future hint now becomes active. Thus, the hint is added to **SS.moveMultipliers** and the future hint is removed from **SS.futureHints**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE ]
SS.usedMoeMoves { 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS, AFE }
SS.lastUserMove AFE
SS.candidateUserMoves []
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.moveMultipliers { (LOBLO: value 2, timeout 2) }

```

10. Again, let's examine the path where the search considers making no MOE MOVE.

The legal USER MOVES are: MUD, LOBLO, CALENDAR, NOTEPAD, and MERGER. This example shows the path where the search considers USER MOVE LOBLO at this point. *Spurred on by the mention of the mystery medicine, the User searches around, finally finding the bottle of Loblo in Dunbar's medicine cabinet.*

LOBLO is added to **SS.candidateUserMoves**. Because there is no reason not to, LOBLO is removed from **SS.candidateUserMoves**, and added to **SS.history** and **SS.usedUserMoves**. As usual, it becomes the value of **SS.lastUserMove**. Because LOBLO has been added to **SS.history**, its entry in **SS.moveMultipliers** reverts to the default:


```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO ]
SS.usedMoeMoves { 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove LOBLO
SS.candidateUserMoves []
SS.moveSubstitutions { (MUD→MUD,PAINT) }

```

11. After making USER MOVE LOBLO, the search considers making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making MOE MOVE 7: DELAY THE “ANALYZE FOR LOBLO” REPORT.

This is an anticipating MOE MOVE. Refining this MOVE changes the future behavior of the police labs, so that when the User requests the ceramic fragments to be analyzed for Loblo, the lab will delay the return of the report to the User. To represent this in the search state, AFL is added to **SS.movesWhichWillBeDelayed**. As usual, MOE MOVE 7 is added to **SS.history** and **SS.usedMoeMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7 ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove LOBLO
SS.candidateUserMoves []
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.movesWhichWillBeDelayed { AFL }

```

12. After making MOE MOVE 7, suppose a USER MOVE is next made.

The search considers all legal USER MOVES: MUD, AFL, CALENDAR, NOTEPAD, and MERGER. This example shows the path where the search considers USER MOVE MUD at this point. *After finding the Loblo in Dunbar’s cabinet, the User finally enters the scene of the alleged crime. Curious about the library door being locked from the inside, she investigates the balcony. Stepping out on the balcony, she discovers both muddy footprints and a scraped railing. (Both of these are described together because of the move substitution).*

In the search, MUD is first added to **SS.candidateUserMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7 ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove LOBLO
SS.candidateUserMoves [ MUD ]
SS.moveSubstitutions { (MUD→MUD,PAINT) }
SS.movesWhichWillBeDelayed { AFL }

```

Second, the search performs the move substitution (MUD→MUD,PAINT). MUD is removed from **SS.candidateUserMoves**, the sequence MUD,PAINT is put in its place, and the move substitution is removed from **SS.moveSubstitutions**.

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7 ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove LOBLO
SS.candidateUserMoves [ MUD, PAINT ]
SS.moveSubstitutions { }
SS.movesWhichWillBeDelayed { AFL }

```

Processing proceeds normally, with MUD then PAINT being added to **SS.history** and **SS.usedUserMoves**. PAINT becomes the value of **SS.lastUserMove**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove PAINT
SS.candidateUserMoves [ ]
SS.movesWhichWillBeDelayed { AFL }

```

13. After making USER MOVE MUD, suppose another USER MOVE is next made.

The legal USER MOVES are AFL, CALENDAR, NOTEPAD, and MERGER. This example shows the path where the search considers USER MOVE AFL at this point. *After finding the muddy footprints and scraped paint, the User decides to analyze the fragments for Loblo. However, it seems to be taking a while for the report to get back.*

This is an example of a USER MOVE being delayed. At first, AFL is added to **SS.candidateUserMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove PAINT
SS.candidateUserMoves [ AFL ]
SS.movesWhichWillBeDelayed { AFL }

```

Because AFL is in **SS.movesWhichWillBeDelayed**, it is not added to **SS.history**. Instead, AFL is removed from **SS.movesWhichWillBeDelayed** and **SS.candidateUserMoves** and added to **SS.movesWhichHaveBeenDelayed**.

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT ]
SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }
SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO }
SS.lastUserMove PAINT
SS.candidateUserMoves []
SS.movesWhichWillBeDelayed {}
SS.movesWhichHaveBeenDelayed { AFL }

```

This represents the time in the concrete world during which the User is waiting for the return of the report, and presumably doing other things. In the search, AFL will remain in **SS.movesWhichHaveBeenDelayed** until the search makes MOE MOVE 8: RETURN THE "ANALYZE FOR LOBLO" REPORT.

14. After making AFL (and having it be delayed), the search considers making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making no MOE MOVE at this point, which means a USER MOVE is next made.

In fact, this example shows the path where Moe considers making six USER MOVES in a row at this point. In order, the next six USER MOVES shown in this example are: MERGER, NOTEPAD, CALENDAR, CONFRONT GEORGE, FOCUS, and finally CATCH GEORGE.

In turn, each of these USER MOVES has been added to **SS.candidateUserMoves** then added to **SS.history**, **SS.usedUserMoves**, and, as appropriate, become the value of **SS.lastUserMove**. Since MERGER comes before NOTEPAD in **SS.history**, this boolean is set to true. Here is what the search state looks like after Moe considers making these six USER MOVES:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT, MERGER, NOTEPAD, CALENDAR, CONFRONT GEORGE, FOCUS, CATCH
  GEORGE ]

SS.usedMoeMoves { 7, 12, 14, 15, 16, 18 }

SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO, CAL-
  ENDAR, CONFRONT GEORGE, CATCH GEORGE, FOCUS, NOTEPAD, MERGER }

SS.lastUserMove CATCH GEORGE

SS.mergerThenNotepad yes

SS.candidateUserMoves []

SS.movesWhichHaveBeenDelayed { AFL }

```

15. After making the last of the six USER MOVES, the search, as usual, considers next making all legal MOE MOVES and making no MOE MOVE. This example shows the path where the search considers making MOE MOVE 8: RETURN THE “ANALYZE FOR LOBLO” REPORT at this point. *After waiting for some time, the User receives the analysis for Loblo. The report states that Loblo in combination with Ebullion was the cause of death.*

In the search, MOE MOVE 8 causes AFL to move from **SS.movesWhichHaveBeenDelayed** to **SS.candidateUserMoves**. MOE MOVE 8 is added to **SS.history** and **SS.usedMoeMoves**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT, MERGER, NOTEPAD, CALENDAR, CONFRONT GEORGE, FOCUS, CATCH
  GEORGE, 8 ]

SS.usedMoeMoves { 7, 8, 12, 14, 15, 16, 18 }

SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO, CAL-
  ENDAR, CONFRONT GEORGE, CATCH GEORGE, FOCUS, NOTEPAD, MERGER }

SS.lastUserMove CATCH GEORGE

SS.mergerThenNotepad yes

SS.candidateUserMoves [ AFL ]

SS.movesWhichHaveBeenDelayed { }

```

At this point, AFL is removed from **SS.candidateUserMoves** and added to **SS.history** and **SS.usedUserMoves**. It also becomes the value of **SS.lastUserMove**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT, MERGER, NOTEPAD, CALENDAR, CONFRONT GEORGE, FOCUS, CATCH
  GEORGE, 8, AFL ]

SS.usedMoeMoves { 7, 8, 12, 14, 15, 16, 18 }

SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO,
  AFL, CALENDAR, CONFRONT GEORGE, CATCH GEORGE, FOCUS, NOTEPAD,
  MERGER }

SS.lastUserMove AFL

SS.mergerThenNotepad yes

SS.candidateUserMoves []

```

16. After making MOE MOVE 8 (and having AFL happen), suppose a USER MOVE is next made.

In fact, this example shows the path where Moe considers making the last two USER MOVES of the scenario. In order, the last two USER MOVES shown in this example are: CONFRONT DUNBAR followed by TICKET/AFFAIR. *The User takes the report to Dunbar and accuses her of murder. Dunbar, feeling incredible remorse, breaks down and confesses she poisoned the deceased, and conspired with Baxter, her lover, to cover it all up.*

In the search, these two USER MOVES (one after the other) are added to **SS.candidateUserMoves**, then removed, and then added to **SS.history**, **SS.usedUserMoves**, and as appropriate, become the value of **SS.lastUserMove**:

```

SS.history [ 14, 16, HOLES, 12, 15, LADDER, FRAGS, 18, AFE, LOBLO, 7, MUD,
  PAINT, MERGER, NOTEPAD, CALENDAR, CONFRONT GEORGE, FOCUS, CATCH
  GEORGE, 8, AFL, CONFRONT DUNBAR, TICKET/AFFAIR ]

SS.usedMoeMoves { 7, 8, 12, 14, 15, 16, 18 }

SS.usedUserMoves { MUD, PAINT, LADDER, HOLES, FRAGS, AFE, LOBLO, AFL,
  CONFRONT DUNBAR, TICKET/AFFAIR, CALENDAR, CONFRONT GEORGE,
  CATCH GEORGE, FOCUS, NOTEPAD, MERGER }

SS.lastUserMove TICKET/AFFAIR

SS.mergerThenNotepad yes

```

This is the end of the extended example of one path considered by the search.

In this chapter we have learned how the search state represents the important details of the concrete world that occur while refining MOE MOVES and recognizing USER MOVES.

By examining a detailed example path of the search through one future scenario, we have seen how the search manipulates this abstract model. The next chapter describes three search algorithms that use this search state.

Chapter 7

Searching Strategies

The previous chapter describes the search state used by the search to represent the refinement of MOE MOVES and the recognition of USER MOVES. The search state does not represent all the details of the refinement or recognition, but rather the important information. In this way, the search state represents the important changes to the world, characters, presentation system, and User that happen when MOE MOVES are refined and that must happen when USER MOVES are recognized.

This chapter describes the adversary search algorithms used by Moe. The first section describes the basic full-depth search. This description includes how search in the domain of interactive drama is different from search in the domain of Chess. Because full-depth search is intractable for *Tea For Three*, I have created three quick search algorithms that Moe can use instead. The last two sections of this chapter discuss these three algorithms, explaining how each is related to full-depth search.

The next chapter proposes a method to evaluate the three search algorithms. Evaluation is not straightforward because *Tea For Three* has not been completely implemented. The method simulates the experience of the User to gather its data. As you will see, the data suggest that if Moe were connected to an implemented system, it would be able to guide the experience of a User effectively.

When you have read this chapter you will understand the three search strategies used by Moe. You will also understand their theoretical foundation.

If you have not read the the description of Moe in Chapter 1 and at least scanned the previous two chapters, I would recommend that you do. This chapter assumes a knowledge of these.

7.1 Full-Depth Adversary Search

This section describes full-depth adversary search in the domain of interactive drama. First, I explain the important differences between the domain of interactive drama and the domain of Chess and how these differences affect the implementation. Second, I describe the implementation, including Legal MOVE Generation. Finally, I explain that full-depth

search is intractable for interactive dramas the size of *Tea For Three*. This motivates the rest of the chapter, which describes three quick search algorithms that can be used for *Tea For Three*.

7.1.1 Interactive Drama vs. Chess as a domain

In computer science, a Chess playing program is often used to illustrate adversary search. For that reason I want to explain adversary search in the domain of interactive drama by explaining the four important ways that interactive drama differs from Chess, and how these differences affect the implementation and testing of adversary search for interactive drama.

Moe is similar to a basic Chess playing program. Moe models the interaction between Moe and the User as a two-player game, like Chess, where Moe is one player and the User is the other. Using this model, Moe decides which MOE MOVE to make at a given time, if any, by performing a search similar to a two-player game-tree search.

However, there are four important differences between a basic Chess program and Moe. First, the opponent (the User) is modelled as a naive player that isn't trying to defeat Moe. Second, the rules of the game are different, so in particular Moe has the option of refining multiple MOE MOVES or no MOE MOVES on its turn. Third, the evaluation function (as it is implemented) rates only complete scenarios. Ordinarily in Chess there exist static evaluation functions. Fourth, the game has no absolute winning or losing. Moe's success comes from improving (on average) the User's experience. The rest of this section describes these differences in more detail.

Naive Opponent

Moe models itself as a knowledgeable player: Moe knows all the rules of the game, can use adversary search to project the effects of MOVES on the game state, and wants to do its best to improve the User's experience.

Moe models the User as a naive player, because she is unaware of the game. The User's mind is focused on acting and reacting in a simulated world with characters. In her head, she may be creating descriptions of what happens (similar to USER MOVES I hope), but ideally she will be unaware of Moe's actions. Even if she feels manipulated, she should not feel as if her experience is a game with Moe.

Moe also models the User as a player that doesn't want to "win". The User wants to experience good interactive drama. Moe wants the User's experience to be good interactive drama. Therefore, the User does not want to defeat Moe. If she did, her experience would probably be unsatisfying.

Modelling the opponent (the User) as a naive player who doesn't want to defeat Moe is very different from the model employed by most Chess playing programs. Most Chess programs assume that their opponent is as skilled as they are, and that their opponent always makes the best possible move to try to win.

The practical significance of this difference is that the Moe search does not use the usual minimax calculation technique[2] used by most Chess programs. Instead, it uses a strategy that I call *avg-max*. Let's look at how *avg-max* works.

As described in Section 1.5, adversary search works by expanding a tree of nodes. A node that represents a choice point for Moe is called a Moe Node. Likewise, a node that represents a choice of the User is called a User Node.

Since Moe is trying to give the User the best possible experience, it always chooses the MOE MOVE which gives the User's experience the highest expected value (as given by the evaluation function.) Thus, the search calculates the value at a Moe Node by taking the maximum expected value of all child nodes. That's why this strategy is called *avg-max*. This is the same as the minimax model.

However, Moe models the User as naive player who doesn't want to "win." Therefore, the search is concerned with the average, typical USER MOVE. Thus, the search calculates the value at a User Node by taking the *average* expected value of all child nodes. That's why the strategy is called *avg-max*. Notice this is in contrast to the minimax model, which is trying to anticipate the worst possible USER MOVE, and thus would use the minimum value.

I have designed Moe's search to take an average at User Nodes because averaging represents a simple guess of what the User might do. More sophisticated calculations based on keeping statistics or using more complex models certainly exist, but since I do not yet have real Users and have no intuition about what complex models might be appropriate, I have chosen to use this simple method first. Future work in this area might include developing more sophisticated methods.

Moves Don't Alternate

Unlike Chess, turns in interactive drama do not strictly alternate. During the User's experience, USER MOVES are being recognized, and MOE MOVES are being refined. If Moe decides that refining two MOE MOVES simultaneously will improve the User's experience, there is no reason Moe cannot do this. Likewise, Moe may decide to refine no MOE MOVES.

This flexibility is reflected in the implementation of the search, where MOE and USER MOVES don't necessarily alternate. As above, Moe may make any number of MOE MOVES in a row or make no MOE MOVE at all. Thus, **SS.history** could contain many USER MOVES or many MOE MOVES in a row.

Because of this flexibility, the implementation of the search at Moe Nodes is somewhat more complex. Moe Nodes spawn as children both Moe Nodes and User Nodes. When Moe makes a MOE MOVE, the search creates a child Moe Node. This is so two or more MOE MOVES can be made in a row. When Moe considers making no MOE MOVE, the search creates a child User Node. This gives the User the opportunity to make a USER MOVE.

The implementation at User Nodes is more standard. User Nodes spawn only Moe Nodes as children. That is because Moe always has the option of making a MOE MOVE after any USER MOVE.

No Static Evaluation Function

As implemented, the evaluation function is valid only for complete scenarios. This is in contrast to Chess, where static evaluation functions can rate a board position at any time during the game. The lack of static evaluation functions for interactive drama means that Moe cannot use the standard Chess program technique of searching for several ply and then using a static evaluation function.

Moe must instead expand all search paths to completion. I call this a full-depth search. As we shall see, this causes performance problems. Two of our search strategies attempt to overcome this limitation by using modified evaluation functions (based on the original) that can be used like static evaluation functions.

Not Win-Lose Game

Moe's model is that the User follows her own agenda, which may or may not lead naturally to a good experience. Therefore, Moe's job is to use MOE MOVES to guide the User to a better experience than she would have had on her own. When possible this should be a great experience. However, sometimes the choices made by the User preclude a great experience. In this case, Moe should do as much as possible to improve her experience.

Moe's success is based on whether it can improve the experience of the User. As I have mentioned, sometimes the User's actions preclude the User from having a great experience. Therefore, there is no absolute *winning* or *losing* for Moe. This is in contrast to Chess, where one player or the other wins (or it is a tie.)

This creates a problem. How does one judge the success of a dramatic guidance system? In Chapter 8, I pose a model for making this judgement.

7.1.2 Implementation of Full-Depth Search

Section 6.3 showed an extended example of the search considering one future scenario. At the beginning of that example, the search state represented the beginning of the experience, before any USER MOVES had been recognized and before any MOE MOVES had been refined. Let's call that initial search state **SS0**.

As I've mentioned before, a search can be initiated at any time during the User's experience. Let t_i be the time that Moe initiates a search. Before t_i , the User has had a certain concrete experience, and Moe has refined certain MOE MOVES. Therefore, there is a sequence of USER and MOE MOVES that represents the experience of the User so far. Call this *scenario*(t_i).

The input to the search initiated at t_i is the search state, **SS**, that has been derived from *scenario*(t_i). By using the techniques described in the previous chapter, the input **SS** is created by incrementally modifying **SS0** with the effects of each USER or MOE MOVE in *scenario*(t_i).

Recall, that **SS** has many fields. Of particular interest to this discussion is **SS.history**, whose initial value is the sequence of USER and MOE MOVES in *scenario*(t_i).

```

% Returns MOE MOVE which maximizes expected value
ChooseMOEMOVE (SS)
  if (the experience is over) then return NO MOVE
  if (no legal Moe Moves) then return NO MOVE
  results[NO MOVE] = UserNode (SS)
  foreach M ∈ {LEGAL MOE MOVES}
    results[M] = MoeNode (UpdateSearchState (SS, M))
  return the M which maximizes results[M]

```

FIGURE 7.1: ChooseMOEMOVE in Pseudocode

The full-depth search algorithm consists of one initially called function, and two mutually recursive functions. Together, these functions will generate the search tree in a depth-first manner and ultimately return which MOE MOVE (if any) to refine at t_i .

The initially called function is named ChooseMOEMOVE. The mutually recursive functions are named MoeNode and UserNode.

Moe calls ChooseMOEMOVE to determine which MOE MOVE to refine, if any, at t_i . ChooseMOEMOVE's argument is the search state (SS) that represents the User's experience so far.

ChooseMOEMOVE returns either a MOE MOVE to refine or NO MOVE, depending on what will maximize the expected value of the User's experience. Figure 7.1 shows the implementation of ChooseMOEMOVE in pseudocode.

First, ChooseMOEMOVE checks to see if *the experience is over*. This is the same as checking whether all sixteen USERMOVES are in SS.history. ChooseMOEMOVE also checks if there are no legal MOE MOVES. (The discussion of Legal MOVE Generation appears in Section 7.1.3.) If either of these are true, NO MOVE is returned.

Assuming the experience is not over and that there are legal MOE MOVES to make, ChooseMOEMOVE then determines which of those legal MOE MOVES to make, if any.

First, it computes the expected value of making NO MOVE by calling UserNode with its argument, SS. SS is not updated because the state of the experience hasn't changed.

Second, it computes the expected value of making each of the legal MOE MOVES by calling MoeNode with an updated search state. The updated search state is created by modifying SS with the effect of each MOE MOVE.

Notice that ChooseMOEMOVE calls MoeNode to determine the expected value of making MOE MOVES. This is because Moe is allowed to make more than one MOE MOVE in a row. Likewise, ChooseMOEMOVE calls UserNode with its argument SS because Moe is allowed to make no MOE MOVE.

Finally, ChooseMOEMOVE returns the MOE MOVE (possibly NO MOVE) that maximizes the expected value of the User's experience. If more than one MOE MOVE is optimal, then

```

% returns expected value, given next MOVE could be a MOE MOVE
MoeNode (SS)
  if (the experience is over) then return EvaluationFunction (SS.history)
  if (no legal User or Moe Moves) then return  $-\infty$ 
  results[NO MOVE] = UserNode (SS)
  foreach M  $\in$  {LEGAL MOE MOVES}
    results[M] = MoeNode (UpdateSearchState (SS, M))
  return Max (results)

```

```

% returns expected value, given next MOVE is a USER MOVE
UserNode (SS)
  if (the experience is over) then return EvaluationFunction (SS.history)
  if (no legal User Moves) then return  $-\infty$ 
  foreach U  $\in$  {LEGAL USER MOVES}
    results[U] = MoeNode (UpdateSearchState (SS, U))
  return weighted Average (results)

```

FIGURE 7.2: MoeNode and UserNode in Pseudocode

Moe chooses among them arbitrarily.

The two mutually recursive functions that generate the rest of the search tree are named MoeNode and UserNode. These are shown in Figure 7.2.

MoeNode is the same as ChooseMOEMOVE except in three ways. First, when *the experience is over*, MoeNode calls the EvaluationFunction on **SS.history**. This is the search reaching a “leaf-node.” Second, it returns the value $-\infty$ if there are no legal MOVES. $-\infty$ is the value given to an incomplete experience. And third, instead of returning the MOE MOVE that maximizes the expected value, MoeNode returns the maximum expected value.

UserNode is similar to MoeNode with a few key differences. First, UserNode requires that the next MOVE in the search be a USER MOVE. Moe’s model is that UserNode returns the expected value of the experience given that the next MOVE *must* be a USER MOVE. For this reason, UserNode checks the availability of legal USER MOVES only, independent of whether there are any legal MOE MOVES. Second, instead of returning the maximum expected values, UserNode returns the weighted average of the expected values. The weights are provided by the move multipliers (in **SS.moveMultipliers**) described in the previous chapter.

Notice that unlike the User Node of a minimax search, which would return the minimum expected value, UserNode returns the weighted average. This is consistent with Moe’s model that the User is a naive player that is neither aware of nor concerned with winning.

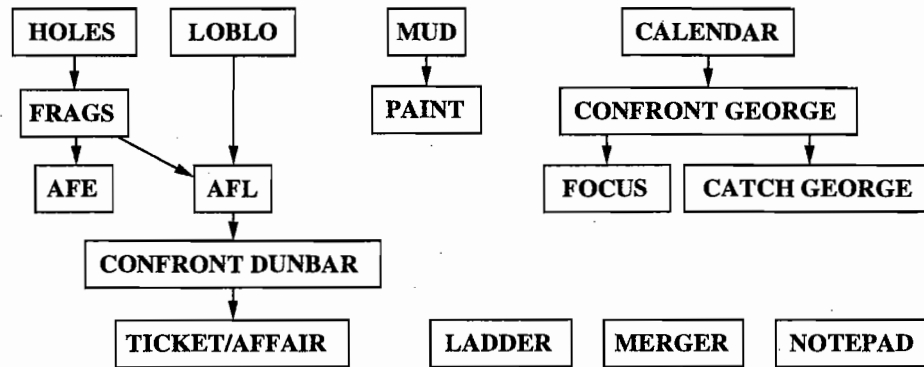


FIGURE 7.3: DAG of USER MOVES and precedence relations

As we've mentioned, this algorithm is similar to those used by standard Chess programs. The main differences are: one, the MOVES don't strictly alternate; two, the search uses a weighted average of values (instead of a minimum) at User Nodes; and three, the search has no static evaluation function.

7.1.3 Legal MOVE Generation:

The last part of the full-depth search that needs to be described is Legal MOVE Generation. There are two separate types of Legal MOVE Generation in Moe's search: one for MOE MOVES and the other for USER MOVES. Each is described separately.

Legal USER MOVE Generation

The set of legal USER MOVES is determined in two steps. First, Moe calculates the possible set of USER MOVES that are legal according to the temporal precedence relationships between USER MOVES. Second, Moe removes any USER MOVES that are illegal because of previously refined MOE MOVES.

As described in Chapter 2, the USER MOVES are related to each other by precedence. These relations are represented in the directed acyclic graph (DAG) shown in Figure 7.3. An arrow from one USER MOVE to another means that the USER MOVE at the tail of the arrow must come before the USER MOVE at the head of the arrow. For example, MUD must come before PAINT, while there is no relation between FRAGS and LOBLO.

The possible set of legal USER MOVES is calculated by removing the USER MOVES in **SS.history** from the DAG. Any USER MOVES which have no arrows pointing to them are considered possible legal USER MOVES.

From the possible set, Moe subtracts (using set difference) all the USER MOVES in the union of **SS.illegalUserMoves** and **SS.movesWhichHaveBeenDelayed**. The remaining USER MOVES are the set of legal USER MOVES.

Legal MOE MOVE Generation:

Every MOE MOVE has a function called `LegalWhen` that determines whether the MOE MOVE is legal in the current situation. Thus, the set of legal MOE MOVES is the set of MOE MOVES not in `SS.history` (MOE MOVES can be made only once) whose `LegalWhen` functions return true.

Each `LegalWhen` function has been created by the artist specifically for the particular MOE MOVE. The `LegalWhen` functions for the MOE MOVES of *Tea For Three* are implemented as boolean expressions over a standard set of conditions.

The conditions are whether or not certain USER or MOE MOVES are in `SS.history`, whether certain USER MOVES are in `SS.movesWhichHaveBeenDelayed`, and which *activities* (as defined in Chapter 3) describe `SS.lastUserMove`.

Let's look at an example. Consider MOE MOVE 9: COMBINE AFE AND AFL. MOE MOVE 9 uses the following expression in its `LegalWhen` function:

```
not happened (AFL)
and not happened (AFE)
and happened (FRAGS)
and happened (LOBLO)
```

This expression is both correct and efficient. For correctness, only the first two terms are needed, since it makes temporal sense to make MOE MOVE 9 any time before either of the two reports have been returned.

However, to be efficient, the search should not consider making this MOVE until it is relevant, which is when the User might request the analysis of the fragments for Loblo. This can only happen after the USER MOVES, FRAGS and LOBLO. Thus, the last two terms prevent the search from considering this MOE MOVE before then.

Let's look at another example. Consider MOE MOVE 11: GEORGE SHOOTS SKEET. The expression for this move is:

```
hasn't happened (MUD)
and last activity (search scene)
```

There are two interesting aspects of this expression. First, it is overly restrictive. This MOE MOVE, when refined, might actually work even if the last activity were not *search scene*. For example, the system could wait for the User to visit the library, but not actually require a USER MOVE to be recognized.

However, by restricting the situations in which Moe may refine this MOE MOVE, it becomes a much more powerful hint, capable of almost always causing (it's a hint with value 40) the USER MOVE MUD. Without the restriction, this hint would be much less powerful.

The second interesting aspect of this expression is that it requires that the MUD hasn't happened. In a smarter system, the Legal MOVE Generator could infer this part of the

expression, since this MOVE is a hint for MUD. Moe does no such reasoning. Moe requires all dependencies to be explicit.

These are two examples of the LegalWhen expressions used for Legal MOE MOVE Generation. Each of *Tea For Three's* LegalWhen expressions has been carefully crafted by the artist. As you can see, LegalWhen can be used to achieve other effects, such as increased search efficiency or increased MOE MOVE power. Appendix B contains the LegalWhen expressions for *Tea For Three's* eighteen MOE MOVES.

7.1.4 The Full-Depth Search is Intractable

The above description plus the descriptions in the previous chapters fully describe full-depth search for interactive drama. If this search is implemented in a straightforward manner, it has a prohibitive problem: for experiences the size of *Tea For Three*, the search is intractable.

A standard Chess program will limit its search time by limiting its search depth. The search described here must expand the entire search tree, which, as we will discuss below, has far more than billions of nodes. Therefore, near the beginning of the experience, the full-depth search is intractable.

If half of the experience (eight USER MOVES) has happened, the full-depth search terminates in a reasonable amount of time. This is because the maximum search depth is reduced, and thus the search time is reduced drastically. Thus, if *Tea For Three* could be described by half as many USER MOVES, Moe could use the straightforwardly implemented full-depth search. For bigger experiences, Moe needs better solutions.

The following two sections describe three tractable search algorithms. The first search algorithm is called Sampling Adversary Search (SAS). SAS uses random sampling to create a partial evaluation function which the search uses to limit its search depth. The second search algorithm, SAS+, is a variation of SAS. The third search algorithm is called Memoized Future Contribution Search (MFC). MFC is a dynamic programming algorithm that exploits a symmetry in the evaluation function. MFC is a very quick algorithm that retains the functionality of the full-depth search.

7.2 Sampling Adversary Search (SAS, SAS+)

depth-limited
use of
MOE-MOVES
+
random
sampling
of future
USER MOVES

SAS is a modified version of the full-depth search algorithm that uses a partial evaluation function to limit its search depth. Instead of expanding the entire search tree, SAS searches to a fixed depth.

As SAS expands the search tree, it keeps track of how many new MOE or USER MOVES are added to **SS.history**. If SAS reaches its fixed depth limit at a given node, SAS considers that node a *leaf node* and uses a heuristic evaluation function to calculate the value of the node.

```

% returns expected value, given next MOVE could be a MOE MOVE
SAS:MoeNode (SS)
  if (maximum search depth reached) then return SHEF (SS)
  if (the experience is over) then return EvaluationFunction (SS.history)
  if (no legal User or Moe Moves) then return  $-\infty$ 
  results[NO MOVE] = SAS:UserNode (SS)
  foreach M  $\in$  {LEGAL MOE MOVES}
    results[M] = SAS:MoeNode (UpdateSearchState (SS, M))
  return Max (results)

```

```

% returns expected value, given next MOVE is a USER MOVE
SAS:UserNode (SS)
  if (maximum search depth reached) then return SHEF (SS)
  if (the experience is over) then return EvaluationFunction (SS.history)
  if (no legal User Moves) then return  $-\infty$ 
  foreach U  $\in$  {LEGAL USER MOVES}
    results[U] = SAS:MoeNode (UpdateSearchState (SS, U))
  return weighted Average (results)

```

FIGURE 7.4: SAS:MoeNode and SAS:UserNode in Pseudo-code

To determine the value of a *leaf node*, SAS uses a *Sampling Heuristic Evaluation Function* (SHEF). The SHEF is calculated by averaging the expected value of a number of random samples, each of which represents one possible future experience. SAS is called a *sampling* search because it uses a *sampling* evaluation function.

The two functions for that implement SAS are shown in Figure 7.4. The modifications from the full-depth search are straightforward. The only difference is that SAS:MoeNode and SAS:UserNode first check the condition *maximum search depth reached*, and if true, call the SHEF.

maximum search depth reached is calculated examining the **SS.history**. Whenever a MOE MOVE or USER MOVE is added to **SS.history**, the depth of the search increases by one. When the fixed search depth is reached, *maximum search depth reached* will be true.

The reason Moe uses **SS.history** to calculate the depth is that only **SS.history** represents how many MOE or USER MOVES have happened. If the search depth were calculated by the search functions, SAS:MoeNode and SAS:UserNode would have to keep track of move substitutions, delayed USER MOVES, etc. **SS** centralizes the bookkeeping, including the number of new MOE and USER MOVES, and thus the depth.


```

% Sampling Heuristic Evaluation Function
SHEF (SS, N)
  totalValue = 0
  dotimes (N)
    totalValue += SousSHEF (SS)
  return totalValue ÷ N

SousSHEF (SS)
  if (the experience is over) then return EvaluationFunction (SS.history)
  if (no legal User Moves) then return  $-\infty$ 
  choose random U ∈ {LEGAL USER MOVES}
  return SousSHEF (UpdateSearchState (SS, U))

```

FIGURE 7.5: An algorithm for SHEF and SousSHEF written in pseudo-code

7.2.1 The Sampling Heuristic Evaluation Function (SHEF)

The Sampling Heuristic Evaluation Function (SHEF) has a simple implementation, shown in Figure 7.5. The SHEF takes a search state **SS** and an integer **N** as arguments. The value returned by SHEF is equal to the average of **N** calls to its helper function, **SousSHEF**.

Each call to **SousSHEF** represents taking one random sample of the future of the User's experience. Thus SHEF is calculating the expected value of the experience at a point by averaging the values of **N** random samples of how the experience might go in the future.

SousSHEF calculates its value by projecting one random future of the experience from the point represented by its argument, **SS**. To create a random future, **SousSHEF** incrementally increases the length of the scenario by adding new **USER MOVES**, until the experience is over. Each new **USER MOVE** is chosen randomly from available legal **USER MOVES**.

For the studies presented in the next chapter, I have chosen **N** to be thirty-two. The larger **N** is, the more accurate the SHEF should be, but the longer it will take to compute. In practice (as you will see in the next chapter), SAS is fairly effective using thirty-two samples. Future work in this area involves understanding how to choose **N** in order to balance accuracy and efficiency.

Relationship of SAS to Full-Depth Search

SAS is different from full-depth search in two important ways. First, the implementation of the SHEF assumes that no future dramatic guidance will be given. Second, even under the previous assumption, the SHEF returns an approximately correct value.

The random samples used by the SHEF represent a possible future experience of the User. Importantly, **SousSHEF** chooses only **USER MOVES**. This means that the SHEF is

limiting the User's experiences to those that have no dramatic guidance in the future.

If the SHEF expanded the entire search tree from that point using only USER MOVES, it would return the actual expected value of the User's experience, assuming no future dramatic guidance. By taking only N samples, the SHEF is returning a value that approximates the true value. As I've mentioned above, the larger N is, the more accurate this value should be.

7.2.2 SAS+

SAS considers making MOE MOVES only to a fixed search depth. This implies that while SAS might provide effective local guidance, its strategic vision is limited. This could affect Moe's decision process in subtle ways.

However, SAS's strategic blindness is in one way evident: SAS cannot use certain *pairs* of MOE MOVES. The most striking example is the pair, MOE MOVE 7 and MOE MOVE 8. MOE MOVE 7 delays the return of the "Analyze for Loblo" report (USER MOVE AFL), whereas MOE MOVE 8 returns the report on demand. In full-depth search, these two make a powerful combination, often capable of determining exactly when AFL will be recognized.

However, SAS (with a small fixed depth) will rarely choose to make MOE MOVE 7. This is because the SHEF cannot consider making MOE MOVE 8 in the future of that same search, and therefore the experience will be incomplete and receive a value of $-\infty$.

When SAS considers MOE MOVE 7, AFL will be added to **SS.movesWhichWillBeDelayed**. Eventually, during the SousSHEF's random selection of legal USER MOVES, AFL will be chosen and then moved to **SS.movesWhichHaveBeenDelayed** (This is described in the previous chapter). AFL will then be an illegal USER MOVE.

Because AFL is illegal and SAS will not move it to **SS.history** by making MOE MOVE 8, the experience will be incomplete. That is, the experience will not be over, but there will be no legal USER MOVES. In that case, SousSHEF gives the experience a value of $-\infty$.

Because SousSHEF has given a value of $-\infty$ to each of the N samples, the SHEF will also return $-\infty$. Thus, to SAS, MOE MOVE 7 functions as a horrible MOE MOVE that destroys the User's experience.

SAS+ is a modified version of SAS that helps solve this problem by changing Legal USER MOVE Generation in the SHEF. In SAS+, delayed or denied USER MOVES are considered legal during the execution of SousSHEF. This means that delayed or denied USER MOVES (such as AFL, above) will eventually be chosen by SousSHEF, and a value for a complete experience can be returned.

Relationship of SAS+ to Full-Depth Search

By legalizing denied and delayed USER MOVES during the execution of SousSHEF, SAS+ anticipates guidance that will come from making MOE MOVES in the future. However, SAS+ does not mimic how full-depth search would consider making these MOE MOVES.

The heuristic used by SAS+ to consider future MOE MOVES is different from full-depth search in three important ways.

First, SAS+ is not assessing any *Manipulation* cost for making the future MOE MOVES. This implies that SAS+ may be more likely to make the first MOE MOVE of a pair than full-depth search.

Second, SousSHEF chooses its USER MOVE sequence randomly. That means SAS+ has no control over when the future MOE MOVES are refined. This implies that SAS+ may be less likely to make the first MOE MOVE of a pair than full-depth search.

Third, since SousSHEF is not considering the actual MOE MOVES, it may be implicitly making MOE MOVES at times when they are illegal.

It is important to remember that SAS+ is using heuristics. They may not be optimal, they may not be based in reality, and they may have strange behavior. In practice, however, we shall see that SAS+ does quite a bit better than SAS.

7.2.3 SAS and SAS+

This section shows two aspects of SAS and SAS+. First, it explains how SAS and SAS+ are time limited algorithms. Second, it shows some preliminary results that demonstrate SAS and SAS+ are working to a useful degree. Chapter 8 discusses the meaning of these results and the method of obtaining them.

SAS, SAS+ are Time Bounded Algorithms

SAS and SAS+ search to a depth and call a heuristic evaluation function. Usually, a search's heuristic evaluation function does not depend on the search depth. That is, it takes a similar amount of time anywhere in the search. The running time of the SHEF, on the other hand, increases linearly with the number of remaining USER MOVES in the experience.

Because the SHEF is slow, SAS and SAS+ are only feasible for depths of one, two, or maybe three, in my implementation. Searching to larger depths takes too long. In order to be able to gather a meaningful amount of data, I have chosen to restrict SAS and SAS+ to depth two.

One could imagine building hybrid SAS algorithms that increased their search depth over time. However, that effort is relegated to future work.

Sneak Preview of Effectiveness Results

I want to give you a sneak preview of how well SAS and SAS+ can guide a User's experience. For now let's look at how these algorithms are doing on "average" Users. Chapter 8 describes in detail how search data is gathered, shows additional data in the form of graphs, and explains how to interpret the data.

Figure 7.6 shows the preliminary data. Briefly, this data is gathered by performing many trials for each search algorithm. Each trial represents a User experiencing *Tea For Three*

Search Algorithm	Mean	Percentile	Star Rating
None	5.9	50th	between 0-1 stars
SAS depth 1	7.0	93th	lower 3 stars
SAS depth 2	7.1	94th	upper 3 stars
SAS+ depth 1	7.3	97th	between 3-4 stars
SAS+ depth 2	7.4	98th	between 3-4 stars

FIGURE 7.6: Effectiveness of SAS, SAS+ on “Average Users”

under the influence of dramatic guidance. The mean value achieved by a search algorithm is the mean value of the set of experiences gathered during the trials that used that search algorithm. The percentile score is calculated by taking the percentile score of the mean with respect to the scale of experiences with no dramatic guidance. Star-Ratings describe the mean value. (Star-ratings are described in Chapter 4.)

The first row shows that in *Tea For Three* the average User’s experience with no dramatic guidance has mean value 5.9, which of course scores at the 50th percentile. Data for SAS and SAS+ are given for both depths of one and two.

My first observation is that, on average, the Users’ experiences are getting much better when Moe guides them. Second, it is interesting that the depth of SAS and SAS+ has so little effect on their ability to guide an experience. Deeper depths do better, but only by a small amount. Finally, although the percentile scores appear very high, SAS+ at depth two (on average) still only gets Star Ratings of moderate quality. There is room for improvement, which leads us to the next section.

7.3 Memoized Future Contribution Search (MFC)

Recall for a moment the full-depth search described at the beginning of this chapter and shown in Figures 7.1 and 7.2. *MoeNode* and *UserNode* are two functions that return the expected value of an experience. Both functions take search states (*SS*) that represent the state of the experience at some time. The value returned by either function is computed by searching to the bottom of the tree, where all the scenarios are complete, and backing up the values through the tree using the avg-max method described earlier.

The full-depth search is intractable, as I’ve mentioned. However, it turns out that a dynamic programming algorithm exists that more quickly computes the same values as the full-depth search. The approach is to memoize the tree as it is searched.

The basic idea behind memoizing full-depth search is to store the values computed by *MoeNode* and *UserNode* in a table indexed by *SS*, and if either function ever gets an identical *SS* as an argument, it would use the stored value instead of searching again. A benefit of memoization is that if the complete table of memoized values can be calculated,

then subsequent calls to ChooseMOEMOVE will result in at most depth-one searches. This is because the values returned by any call to MoeNode will be in the table.

The problem with trying to memoize the full-depth search described here is that the table is too large to compute and store feasibly. Even if one considers just the search states (SS) with sixteen USER MOVES and no MOE MOVES in SS.history, one counts over 2.4 billion legal SS's. The set of all SS's is much larger, since an SS.history could contain instead any legal subset of the sixteen USER MOVES and up to eighteen MOE MOVES in any legal order. Thus, memoization will not work for the full-depth search as implemented.

However, two properties about how evaluation and search work on *Tea For Three* allow us to change the search (without changing its results) in order to shrink the table to a manageable size. After we consider each of these properties I will return to the description of the memoization.

Property One: Decomposable Evaluation

Recall that the argument to the evaluation function is a scenario, which is a sequence of USER and MOE MOVES that serves as an abstract description of the User's actual or projected experience. Consider a complete scenario consisting of the sequence of moves $m_1, m_2, m_3, \dots, m_n$. The *Tea For Three* evaluation function considers each m_i of the scenario, in order, and calculates its contribution, C_{m_i} . Each C_{m_i} depends on only m_1, \dots, m_i .¹ The value of the complete scenario is $\sum_{i=1}^n C_{m_i}$. Notice that the value contributed by the MOVES in any prefix m_1, \dots, m_j is well defined.

Thus, for any legal search state SS with SS.history = m_1, \dots, m_j we can define the contribution of m_1, \dots, m_j to any scenario with that prefix as ContributionOf(SS.history), whose value is given by $\sum_{i=1}^j C_{m_i}$. As above, ContributionOf(SS.history) is well defined.

This observation leads us to create two new functions that can be used for searching: FC:MoeNode and FC:UserNode (shown in Figure 7.7). "FC" stands for *future contribution* because we can think of them as returning the expected avg-max contribution of the future MOVES of the scenario. These new functions are related to MoeNode and UserNode by the following equations:

$$\text{FC:MoeNode (SS)} = \text{MoeNode (SS)} - \text{ContributionOf (SS.history)}$$

$$\text{FC:UserNode (SS)} = \text{UserNode (SS)} - \text{ContributionOf (SS.history)}$$

As you can see in Figure 7.7, I have modified MoeNode and UserNode (from Figure 7.2) in small ways in order to implement FC:MoeNode and FC:UserNode. The main difference is that the value contributed by each MOVE is added as it considered by the search, and therefore when the experience is over (the "leaf node" case), the functions return a value of zero. This is because there are no future MOVES to make a contribution. ValueContributedByMove(SS, m) is simply C_m , as above.

¹The fact that the evaluation function can work this way is possibly idiosyncratic to *Tea For Three* and not necessarily a property of all interactive dramas. See the discussion in Section 7.3.2 for more details.

```

% returns expected future contribution to experience
FC:MoeNode (SS)
% future contribution is zero at end
if (the experience is over) then return 0
if (no legal User or Moe Moves) then return  $-\infty$ 
results[NO MOVE] = FC:UserNode (SS)
foreach M  $\in$  {LEGAL MOE MOVES}
  results[M] = ValueContributedByMove (SS, M) +
              FC:MoeNode (UpdateSearchState (SS, M))
return Max (results)

```

```

% returns expected future contribution to experience
FC:UserNode (SS)
if (the experience is over) then return 0
if (no legal User Moves) then return  $-\infty$ 
foreach U  $\in$  {LEGAL USER MOVES}
  results[U] = ValueContributedByMove (SS, U) +
              FC:MoeNode (UpdateSearchState (SS, U))
return weighted Average (results)

```

FIGURE 7.7: FC:MoeNode and FC:UserNode in Pseudo-code

By examining the new implementation you should be able to verify that the behavior of the search is not changed: The same number and types of nodes will be expanded during the search in the same order, and the same MOE MOVE will be chosen.

In particular, you should be able to convince yourself that the two recursive steps preserve the correct behavior because, for Max and Average,

$$\text{Max}(x_1, \dots, x_n) = C + \text{Max}(x_1 - C, \dots, x_n - C)$$

and

$$\text{Average}(x_1, \dots, x_n) = C + \text{Average}(x_1 - C, \dots, x_n - C)$$

for any constant C . In our case, C would be $\text{ContributionOf}(\text{SS.history})$. Further, this implies that the subtraction of C does not affect the Max operation used to choose the MOE MOVE returned by ChooseMOEMOVE .

Property Two: Order Independence

The second property about both evaluation and search is more surprising. It turns out, as we shall see, that the future contributions for *Tea For Three* (i.e., those calculated by

```

% returns expected future contribution to experience
MFC:MoeNode (SS)
% Note: memoize based on  $i_{SS}$ 
if (value has been memoized) then return value
if (the experience is over) then Memoize and return 0
if (no legal User or Moe Moves) then Memoize and return  $-\infty$ 
results[NO MOVE] = MFC:UserNode (SS)
foreach  $M \in \{\text{LEGAL MOE MOVES}\}$ 
  results[M] = ValueContributedByMove (SS, M) +
    MFC:MoeNode (UpdateSearchState (SS, M))
Memoize and return Max (results)

```

```

% returns expected future contribution to experience
MFC:UserNode (SS)
% Note: memoize based on  $i_{SS}$ 
if (value has been Memoized) then return value
if (the experience is over) then Memoize and return 0
if (no legal User Moves) then Memoize and return  $-\infty$ 
foreach  $U \in \{\text{LEGAL USER MOVES}\}$ 
  results[U] = ValueContributedByMove (SS, U) +
    MFC:MoeNode (UpdateSearchState (SS, U))
Memoize and return weighted Average (results)

```

FIGURE 7.8: MFC:MoeNode and MFC:UserNode in Pseudo-code

FC:MoeNode(SS) and FC:UserNode(SS)) do not depend on the ordering of **SS.history**, except for the last USER MOVE and whether MERGER or NOTEPAD came first. That is, future contributions, which do depend on the ordering of the future MOVES, do not depend on the ordering of past MOVES.

Let's define a modified search state i_{SS} (*the table index derived from SS*) to be all of **SS** except **SS.history**. Notice that i_{SS} contains the unordered MOVES in **SS.history**, since these are identical to the union of **SS.usedUserMoves** and **SS.usedMoeMoves**. i_{SS} also contains the last USERMOVE and whether MERGER or NOTEPAD came first in **SS.lastUserMove** and **SS.mergerThenNotepad**. As you can see, i_{SS} is equivalent to the information mentioned above.

Below, we will show **Property Two**: that for any two search states, SS_a and SS_b , if $i_{SS_a} = i_{SS_b}$ then

$$\begin{aligned} \text{FC:MoeNode}(SS_a) &= \text{FC:MoeNode}(SS_b) \text{ and} \\ \text{FC:UserNode}(SS_a) &= \text{FC:UserNode}(SS_b). \end{aligned}$$

The significance of **Property Two** is that the search can memoize FC:MoeNode and FC:UserNode using i_{SS} as the table index. Figure 7.8 shows the implementation of these memoizing functions, named MFC:MoeNode and MFC:UserNode. (MFC stands for *memoized future contribution*.) Contrary to the usual case, notice that MFC:MoeNode and MFC:UserNode are not memoizing with their argument, SS, but rather with the index derived from SS, which is i_{SS} .

The previous analysis determined that memoizing would be infeasible, because it required the complete SS as the table index. However, since many different SS's have the same i_{SS} , the size of the memoizing table will be smaller in this new case. The size reduction could be huge when one considers that the number of unordered sets (from i_{SS}) of MOVES is $O(2^n)$, while the number of ordered sets of MOVES (from SS) is $O(n!)$.

The punchline is that the memoizing table can be computed. In fact, for *Tea For Three*, the table contains 2,789,703 entries², which makes the calculation feasible on existing workstations.

MFC has been successfully implemented and executed using the functions shown in Figure 7.8. The first phase of the execution is the successful creation of a table indexed by i_{SS} that contains all the possible values. This happens the first time ChooseMOEMOVE is called with the starting search state SS0 (i.e., one with an empty history). The second phase of the execution comprises the subsequent searches that take a small amount of time (depth-one search plus table lookup), but return the same MOE MOVE as the basic full-depth search.

The important points about MFC are:

- MFC gives the same results as full-depth search
- After the table has been created, MFC is fast
- Using future contributions and changing the index to i_{SS} allows the table to be created

7.3.1 Proof of Property Two

MFC uses i_{SS} as the memoizing index. In this proof, I will show that for any arguments, SS_a and SS_b , if $i_{SS_a} = i_{SS_b}$ then the MFC search-node functions return the same value.

To show this for the complete algorithm, I show that each part of the algorithm preserves **Property Two**. If all parts preserve the property, then the whole algorithm preserves the property.

There are only six relevant subordinate functions in the search that use the search state: the test for the end of the experience, Legal USER MOVE Generation, Legal MOE MOVE Generation, calculating the weighted average at a User Node, calculating the contribution of each MOVE to the experience, and updating the search state with either USER or MOE MOVES.

²This value is determined empirically

For the first five of these subfunctions, the way that I show the property is preserved is to show that the execution of the function depends on only the information in i_{SS} .

For the last subfunction, the way that I show **Property Two** is preserved is by showing that the value returned by the MFC functions depends only on i_{SS} . I cannot use the technique used for the previous five subfunction because updating the search state *does depend* on **SS**. However, I show that the only field of **SS** that depends on information not in i_{SS} is **SS.history**, and as I have shown above, this does not affect the other five subfunctions and thus does not affect the values of the MFC functions.

I will proceed systematically through each of the six subordinate functions, showing that each preserves **Property Two**. Once I am through, I will have shown that **Property Two** is true.

Experience is Over

The experience is over when all sixteen **USER MOVES** have happened. This can be checked by examining **SS.usedUserMoves**, which is in i_{SS} . Thus, this function depends on only the contents of i_{SS} .

Legal USER MOVE Generation

Legal **USER MOVE** generation uses four parts of the search state: **SS.usedUserMoves**, **SS.illegalUserMoves**, **SS.movesWhichHaveBeenDelayed**, and **SS.edges**. As explained earlier, **MOVES** in **SS.usedUserMoves** are removed from the DAG denoted by **SS.edges**. Any **USER MOVES** in the DAG that have no arrows pointing to them are possible legal **USER MOVES**. Using set subtraction, the set of legal **USER MOVES** is computed by subtracting **SS.illegalUserMoves** and **SS.movesWhichHaveBeenDelayed** from the possible set.

As you can see, all of these fields are contained in i_{SS} . Thus, this function depends on only the contents of i_{SS} .

Legal MOE MOVE Generation

Legal **MOE MOVE** Generation depends on **SS.usedMoeMoves**, **SS.usedUserMoves**, **SS.lastUserMove**, and **SS.movesWhichHaveBeenDelayed**. The set of legal **MOE MOVES** is generated by taking the set of **MOE MOVES** that haven't happened (from **SS.usedMoeMoves**) and checking each with a legality function. These functions are implemented as boolean expressions over a standard set of conditions. The conditions are whether or not certain **USER** or **MOE MOVES** have happened (calculated from **SS.usedUserMoves** and **SS.usedMoeMoves**), whether certain **USER MOVES** have been delayed (calculated from **SS.movesWhichHaveBeenDelayed**), and which activities (as defined by the evaluation function – see Chapter 3) describe **SS.lastUserMove**.

As you can see, all these fields are contained in i_{SS} . Thus, this function depends on only i_{SS} .

The Weighted Average at a User Node

The weights that are used in the Weighted Average are stored in **SS.moveMultipliers**.

This field is contained in i_{SS} . Thus, this function depends on only i_{SS} .

Contributions of Moves

Let's call the information needed to calculate the contribution of a MOE or USER MOVE the *context* of the calculation. By examining each *feature* of the evaluation function in turn we can determine that the context required is **SS.usedUserMoves**, **SS.lastUserMove**, and **SS.mergerBeforeNotepad**. By showing this I show that calculating the contributions of MOVES depends only on the contents contained in i_{SS} .

The features are: *Manipulation*, *Thought Flow*, *Activity Flow*, *Momentum*, *Options*, *Motivation*, and *Intensity*.

Manipulation requires no context. The contribution of each MOE MOVES is simply: each one takes away its *Manipulation Cost*, completely independent of the context in which it was made. USER MOVES have no manipulation. Thus, *Manipulation* adds nothing to the context needed.

Let's consider the remaining features, realizing that we no longer need to consider MOE MOVES, since they don't contribute to the feature values.

Activity Flow, *Thought Flow*, and *Momentum* are computed by comparing the activities, thoughts, and identity of the current USER MOVE to those of **SS.lastUserMove**. These three features thus indicate that the context to compute the contribution of a MOVE must include **SS.lastUserMove**.

Options is computed by determining the number of active goals at this time, and dividing this by the number of USER MOVES up to this point plus one. There are twelve goals. To determine if a goal is active, the system evaluates a boolean-expression over whether certain USER MOVES have happened. For example, the goal "Talk to George about the new Will" is active when "happened(CONFRONT GEORGE) AND happened(CALENDAR)" is true.

These boolean expressions depend on which USER MOVES have happened and which haven't, but not the order. Thus, to compute the set of active goals and determine the number of USER MOVES up to this point, which is required to compute *Options*, the system needs only **SS.usedUserMoves**. The *Options* feature thus indicates that the context must include **SS.usedUserMoves**.

Motivation is calculated by comparing the activities of the current USER MOVE to the set of activities indicated by the set of currently active goals. The set of active goals, as above, requires **SS.usedUserMoves**. Computing the activities motivated by these goals and comparing them to the current activities requires no context. The *Motivation* feature thus adds nothing to the context needed.

The *Intensity* feature is calculated by creating a curve that represents the excitement of the User over time, and comparing that curve to the ideal curve. There are two steps:

creating the curve and comparing the curve. Comparing the curve is done on a point by point basis, and does not consider context. This leaves creating the curve.

The curve is created by measuring the excitement of the User experience for each USER MOVE. Excitement is derived from the knowledge gained because of that USER MOVE. Thus, if the system knows the previous knowledge and the current knowledge, the system can calculate the difference, and thus the excitement.

The knowledge of the User consists of knowledge levels for nine different facts. The knowledge level for each fact can be determined by functions that examine which USER MOVES have happened, plus whether MERGER comes before NOTEPAD.

Thus, computing *Intensity* is based on **SS.usedUserMoves** and **SS.mergerThenNotepad**. Therefore, *Intensity* adds **SS.mergerThenNotepad** to the context needed.

By examining each evaluation function feature in turn, we know that the context required to determine the contribution of a MOE or USER MOVE is **SS.usedUserMoves**, **SS.lastUserMove**, and **SS.mergerThenNotepad**. These are all contained in the contents of i_{SS} . Thus, calculating the contribution of a MOVE depends on only i_{SS} .

Updating the Search State

As I mentioned above, I cannot show updating the search state depends only on i_{SS} , because updating **SS.history** depends on the previous **SS.history**.

However, updating all the other fields of **SS** depend on only i_{SS} . In particular, the two fields of **SS** that seem like they might depend on **SS.history** actually don't. **SS.lastUserMove** simply takes on the value of the last USER MOVE to update **SS**. Likewise, **SS.mergerThenNotepad** can be determined by either, one, the previous value of **SS.mergerThenNotepad**, or, two, **SS.usedUserMoves** and the USER MOVE that is updating the **SS**.

Because **SS.history** is not used to update the other fields and the other five subfunctions do not depend on **SS.history** and the search routines do not depend on **SS.history**, the values returned by **MFC:MoeNode** and **MFC:UserNode** do not depend on **SS.history**.

Thus, even though updating the search state depends on the full **SS**, the only parts of the **SS** that can affect the value of **MFC:MoeNode** and **MFC:UserNode** are in i_{SS} . Thus, we do not need to update or even record **SS.history** in the MFC search.

In this proof, I have shown that each of the five subordinate functions do not depend on the complete **SS**, but rather on i_{SS} . For the final step of the proof I have shown that the values of the MFC functions are independent of **SS.history**, since nothing but **SS.history** depends on **SS.history**. In this way, I have shown property two is true: the value of the search functions depends on only the information in i_{SS} , not the full **SS**.

7.3.2 Discussion

The following subsection contains more information about MFC. First, I discuss storing the memoizing table. Next, I discuss how MFC was originally designed as a benchmark,

and how because it is a space-bound algorithm, it will not scale well with the increasing size of the ID or any violation of the underlying assumptions. Next, I show that the implemented algorithm is slightly different than MFC because it doesn't really use $-\infty$ but instead -100. The effects of this are discussed. Finally I give a sneak preview of the implemented algorithm's performance.

Storing the memoizing table

For *Tea For Three* MFC creates a table with 2,789,703 entries (determined empirically). The values are stored as floating point numbers stored in a simple splay tree[37], using the compressed representation of i_{SS} as a key.

Since a splay tree stores both the index and the data in the table, the actual index used is a compression of the data contained in i_{SS} . The search state has many fields that are not changed during any search. They are there for the ease of implementation. However, since these fields or parts of fields never change, they never distinguish two i_{SS} 's.

An example would be the array of move multipliers. Since there are a limited set of MOE MOVES, not every USER MOVE can have its probability multiplied. Those that never change can be removed from the compressed index.

As another example, the set of Edges that are used to compute legal USER MOVES can be in only one of two states, depending on whether a MOE MOVE 16 has been refined. Therefore, the representation of the edges field of the state can be compressed into one bit.

As a further example of data compression, this would be the same bit that represents whether MOE MOVE 16 has happened. The point is that many redundant or stable aspects of i_{SS} were removed, resulting in an index of 49 bits.

MFC probably won't scale well

MFC was designed primarily as a benchmark for testing the other search algorithms. MFC performs a search that returns the same results as the theoretically ideal full-depth search. Thus, for any search algorithms that are approximating full-depth search (e.g., SAS and SAS+), MFC establishes an upper bound and desired target for how effective they could be at guiding the experience of the User.

MFC has two problems. The first problem is that the memoizing table is quite large, already pushing the limits of today's workstations. The second problem is that unless the simplifying assumptions are valid, we cannot create this table at all. It is unlikely, for these two reasons, that MFC will scale well to other stories that are bigger.

The severe limitation of MFC is the size of the table. As you can see, the table is quite big. If there were many more MOE or USER MOVES, MFC would become unfeasible. For this reason, future work might include efforts to find new symmetries, analogous to order independence, that we can exploit to reduce the size of the table.

In addition, creating the table at all is based on the validity of two assumptions: first, that the search mechanism does not depend on the ordered History of the experience; and

second, that the contribution of any MOVE to the scenario value does not depend on future MOVES. The real question is whether the two simplifying assumptions are valid.

They are true for *Tea For Three*, but might not be true for other interactive dramas. Specifically, a different sort of evaluation function might need the entire scenario to correctly calculate the contribution for any MOVE. In our previous terminology: C_{m_i} would depend on $m_1, \dots, m_i, \dots, m_n$.

This assumption is valid for features that consider only the previous USER MOVE, such as *Activity Flow* and *Thought Flow*, but likely invalid for features like *Intensity* are modified to work for other ID's.

Perhaps this will become an important design criterion for evaluation functions of the future. Like any form of expression, artists will have to work within the rules of the medium. If they want to try to use MFC on other ID's the size of *Tea For Three*, we have established an important rule.

When $-\infty = -100$

The algorithm that I implemented is slightly different from MFC. Let's call this actual algorithm MMFC (Mistaken Memoized Future Contribution). Instead of returning $-\infty$ in the case of SS with no legal MOVES, the MMFC functions return -100 .

For the case of a *minimax* algorithm with values from 0 to 10, -100 is the same as $-\infty$, from the point of view of the search.

However, FC uses an avg-max method of backing up values in the search tree. For this reason, the -100 is incorporated into the scores, using average, just as if it were a valid value.

As a result of this, MMFC penalizes incomplete experiences, but does not rule them out completely. In fact, the choice of what value to give an incomplete scenario is an artistic decision. The search is not affected by how the value was determined, be it 0, -100 , $-MAXINT$, or $-\infty$. In the most general case, one could consider building an evaluation function for incomplete scenarios, based on their merit, relative to other complete scenarios. It may be that some incomplete scenarios will score better than some complete ones.

The choice of $-\infty$ is actually in one way detrimental. It loses information because when backing up, a Moe Node cannot distinguish between User Nodes whose averages are $(9.8 + -\infty)/2$ and $(2.3 + -\infty)/2$. The Moe Node should choose the MOVE with the 50% chance of a good scenario as opposed to the 50% chance of a bad scenario, even though the other 50% of the time the scenario is bad no matter what. Perhaps this is an argument for using some more complex system.

The consequence of my implementation mistake is that the proof that MFC returns the same results as full-depth search does not apply to MMFC. This is because -100 is returned for the future contribution of any SS with no legal moves. In order for the proof to apply, the functions must return $(-100 - ContributionOf(SS.history))$.

In spite of this, I think MMFC will give results close to a full-depth search that returns -100 as the total value of any incomplete scenario. In any case, the results of MMFC

Search Algorithm	Mean	Percentile	Star Rating
None	5.9	50th	between 0-1 stars
SAS depth 1	7.0	93th	lower 3 stars
SAS depth 2	7.1	94th	upper 3 stars
SAS+ depth 1	7.3	97th	between 3-4 stars
SAS+ depth 2	7.4	98th	between 3-4 stars
MMFC	7.6	99th	upper 4 stars

FIGURE 7.9: Effectiveness of SAS, SAS+, MMFC on “Average Users”

should be a lower bound for the results of MFC for average users (i.e., where the search model is correct.)

Future work is to fix the mistake in the algorithm, and decide what value is appropriate for incomplete scenarios, so that the “real” data can be collected. For now, all the effectiveness results in this and the next chapter refer to MMFC.

Sneak Preview of Effectiveness Results

As with SAS and SAS+ I want to give a sneak preview of the results for MMFC. Figure 7.3.2 shows the information from before, as well as the data for MMFC. The mean for MMFC is 7.6, which is the 99th percentile of experiences under no dramatic guidance. The star value is upper four stars, which is quite respectable. We can see MMFC does better than SAS+.

The 99th percentile seems quite high. Although this is quite good, it might be able to get better. In particular, since MMFC is nearly the best possible search given the models we have, this implies one area of future work might be finding more powerful MOE MOVES.

Chapter 8

Evaluation and Discussion of Search

The previous chapter begins by describing the implementation of full-depth search in the domain of interactive drama. However, for interactive dramas the size of *Tea For Three*, full-depth search is intractable. Therefore, I have implemented three quick search algorithms that Moe can use to guide the experience of a User in *Tea For Three*. The previous chapter ends by describing each of these algorithms, called SAS, SAS+, and MMFC.

This chapter has two main parts. First, I describe the method by which I have evaluated each of these search algorithms. Second, I report and discuss the results.

The three algorithms are judged based on Moe's performance when using that algorithm to decide which MOE MOVES to make. The first section of this chapter describes this method. However, because *Tea For Three* is not completely implemented, this evaluation is not straightforward. Therefore, part of the description of this method includes describing how I have gathered my data by simulating a variety of Users experiencing *Tea For Three*.

The second part of this chapter describes the data I collected by the previously described method. The basic result is that all three search algorithms seem to work well for guiding interactive drama. However, one must interpret this success knowing that this data has been collected from a simulation, not from real Users. A peculiar result is that SAS+ performs almost as well as MMFC. In the discussion of the data I speculate about why that might be true for *Tea For Three*.

The fact that these three search algorithms work well is very significant. As I've mentioned before, my thesis, that *interactive drama is possible*, is supported by two pieces of evidence. First, that an evaluation function can encode an artist's aesthetic. And second, that an adversary search mechanism can effectively guide the User's experience by using this evaluation function. The results of the evaluation described in this chapter suggest that this second piece of evidence is true.

This chapter is the last chapter on search itself. The next chapter is the conclusion of the dissertation, where I summarize the contributions reported earlier and describe future work in interactive drama.

8.1 An Evaluation Method

Even with a successful drama manager, a given single User's experience may be given a low rating since the individual User's actions may preclude a good experience. For this reason, I judge Moe a success if the expected value of the average User's experience is high.

Therefore, to judge each of the search algorithms, I ran a large number of simulated Users through *Tea For Three* under the guidance of that algorithm and collected the results. I use simulated Users since *Tea For Three* is not fully implemented and thus there are no human Users. The ratings are calculated by calling the evaluation function with the scenario that represents the individual User's concrete experience.

As a benchmark, I also ran a large number of Users with no drama manager at all. Section 8.2 presents the data that I collected. As you will see, these search algorithms are quite effective.

8.1.1 The User Experience Simulator (SUE)

If *Tea For Three* were implemented, a User would sit in front of a computer, typing and reading for a time, until the mystery had been solved. The concrete set of actions and perceptions of the User would be her experience, and the sequence of USER MOVES recognized and MOE MOVES refined would be her abstract experience. The evaluation function (which encodes the artist's judgement of quality for experiences) would be called on the abstract experience, and a single data point would be collected.

The problem is to simulate this process of gathering a single data point. To describe how I have done this, let's recall the complete interactive drama system. As we know, Moe is implemented, but the rest of the systems are not yet implemented for *Tea For Three*.

In order to gather data, these unimplemented systems must be simulated. Because simulating each of the systems individually would have been as time consuming as simply implementing them, I chose to simulate all systems at an abstract level. Thus, all of the systems are simulated at once by one system, called the User Experience Simulator (nicknamed SUE).

Figure 8.1 shows how SUE connects to Moe. The portion of the diagram in gray within SUE is there to suggest that SUE simulates those systems, but they do not literally exist within SUE.

As you can see, the SUE must maintain the same interfaces to Moe that exist in the complete interactive drama system. From Moe's point of view, SUE must respond to the refinement of MOE MOVES and provide the USER MOVES that have been recognized. These are shown by the arrows labeled *Refinement* and *Recognition*.

To gather a single data point, Moe and SUE play a little game that represents the experience of one User in *Tea For Three*. The state of the game is the search state that represents the history of the User's experience so far. When Moe moves, Moe takes the current search state and updates the search state with a number of MOE MOVES. This

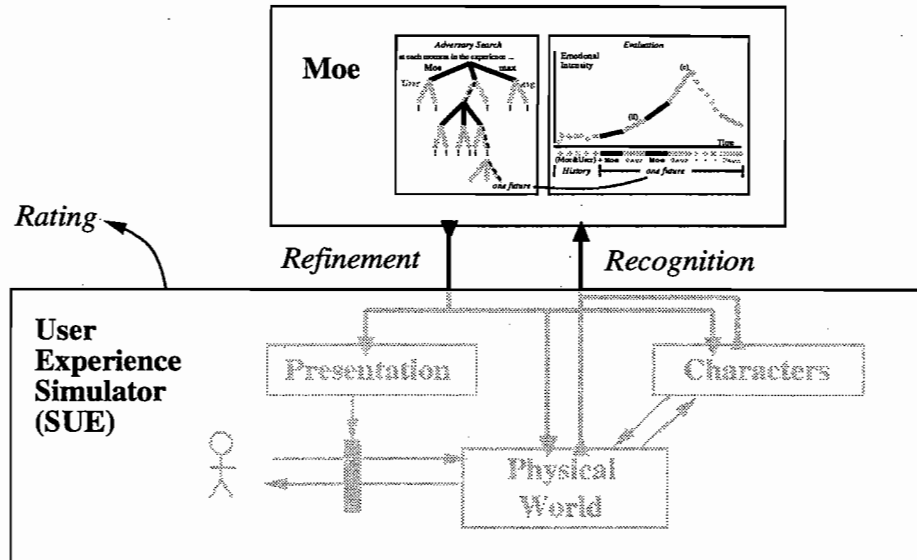


FIGURE 8.1: A Simulated Interactive Drama System

represents the refinement of MOE MOVES in the User's experience. When SUE moves, SUE takes the current search state and updates it with the USER MOVE that has been recognized next. This represents the simulated User's actions and perceptions in the world. When the experience is over, the evaluation function is called on the search state and the value is gathered.

We know from previous chapters how Moe decides which MOE MOVES to refine, if any. The question is how does SUE decide which USER MOVE should come next. One possibility is simply to use the same model of Users that Moe uses. However, since real human Users will not exactly obey Moe's model, I would like the simulated Users to vary in their styles of behavior. This will give us a rough idea of how Moe will perform for a variety of types of Users. The following section describes how SUE selects the next USER MOVE for nine different types of simulated Users.

8.1.2 Nine Types of Users

Moe's search contains a model of the User. Moe assumes that causing, denying, and delaying, etc., MOE MOVES determine the set of available legal USER MOVES. These are reasonable assumptions since they reflect logical constraints on the User.

However, Moe further assumes that each of the legal USER MOVES is equally likely (the avg-max assumption), and that those equal probabilities can be altered by refining *hints*. These two assumptions may not be correct, so I have chosen other possibilities for them.

Corresponding to these two assumptions, I have created two axes along which I have classified Users: Skill and Cooperation. Both axes have three representative values. Therefore, there are a total of nine different types of Users. These classifications are not

```

% returns the USER MOVE that the simulated User next makes
ChooseUserMove (SS,S,C)
  consider U ∈ {LEGAL USER MOVES}
  foreach U
    if (C = Ignore) then value[U] = 1
    else value[U] = SS.moveMultipliers[U].value
  if (C = Always Do)
    then foreach U
      if (value[U] > 1) then return U
  order U with SHEF, by expected quality of future experience
  if (S = Bad)
    then foreach U in lower half
      value[U] = 3 * value[U]
  if (S = Good)
    then foreach U in upper half
      value[U] = 3 * value[U]
  select random U based on probability weights in value[U]

```

FIGURE 8.2: An algorithm for CHOOSEUSERMOVE written in pseudo-code

definitive. There are other possible axes and other values, but these serve to get some User variation over two important Moe assumptions.

Skill—corresponds to the avg-max assumption. I have chosen to classify Users in three types (listed below), based on their skill at participating in interactive drama. Again, this is probably not the most accurate model, but it does give variation to the User types.

1. **Bad** Three times as likely to choose below average USER MOVES
2. **Average** Chooses all USER MOVES equally (Moe's model)
3. **Good** Three times as likely to choose above average USER MOVES

To determine which USER MOVES are above or below average, SUE uses the SHEF, described in the previous chapter.

Cooperation—corresponds to the probabilistic model of hints. Some Users will take hints, others will ignore them. The three types of Users are listed below.

1. **Ignore Hints** Hints have no effect on this type of User
2. **Probability** User obeys hints according to Moe's model
3. **Always Do** User automatically follows a hint as if it were a causing MOE MOVE

Figure 8.2 shows an implementation of SUE's single function. ChooseUserMove determines the next USER MOVE for any of the nine types of Users. SS is the search state representing the current experience of the simulated User, while S and C represent the skill and cooperation of the simulated User. The next section presents results for each of these nine types.

8.2 Data Gathered

The data gathered for each User type is presented in two forms: a graph of the distribution of ratings and a chart summarizing the distribution. The number of samples in the distributions are as follows: SAS has 1,695 samples for each distribution, SAS+ 1,305 samples, MMFC 11,010 samples, and None has 63,093 samples. SAS and SAS+ were run at depth 2.

Each graph shows, in overlay, how well all four methods worked with a particular type of User. Each algorithm is represented by a differently colored or weighted line: MMFC by thick black, SAS+ by thick gray, SAS by thin gray, and None by thin black.

The horizontal (x-axis) gives the experience rating for each simulated User's experience. The ratings have been put into discrete buckets of size one-tenth. A given rating has been placed in the bucket below or equal to it. So, for example, a rating 3.98 will go into the bucket with label 3.9.

The vertical (y-axis) shows the frequency of the values. The curves have been plotted by connecting each rating-frequency pair to the next with a straight line.

In addition to the distributions being represented graphically, a table contains the mean, the 99-percent confidence interval for the mean, and the standard deviation for each distribution. The mean and confidence interval both reach the second decimal place. Ordinarily this degree of accuracy would have little meaning, but since these are means, more accuracy is relevant. The means, confidence intervals, and standard deviations were all calculated according to the methods found in [16].

Next, each chart describes the reported mean by a Star Rating. The absolute Star Rating of this mean is given. Notice this is independent of the type of User. The Star Rating system is described in Chapter 4.

A final rating has been included, which is the mean of the values, disregarding the *Manipulation Cost*. This is included to give you an idea of how manipulative these different algorithms are.

For a quick summary of this data, please see the summary presented in in Figure 8.12 on Page 168.

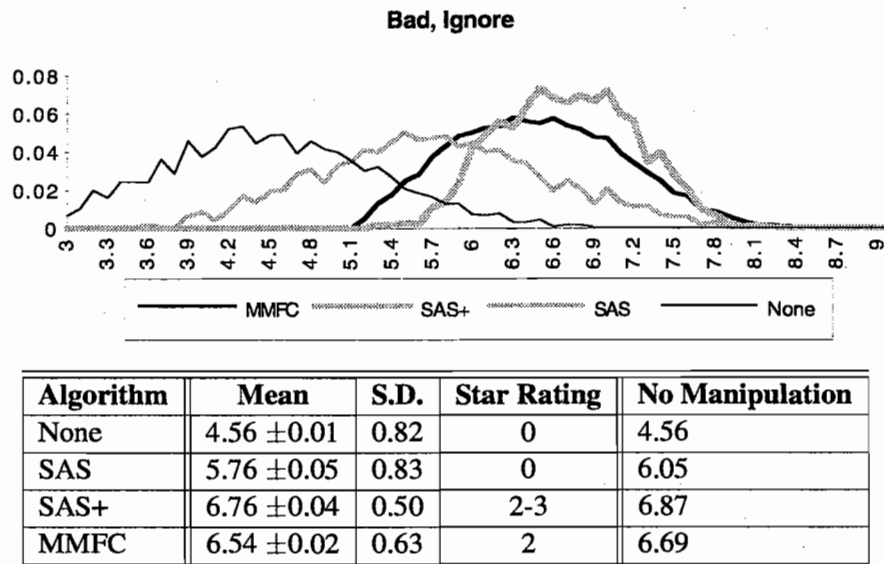


FIGURE 8.3: Bad User that Ignores Hints

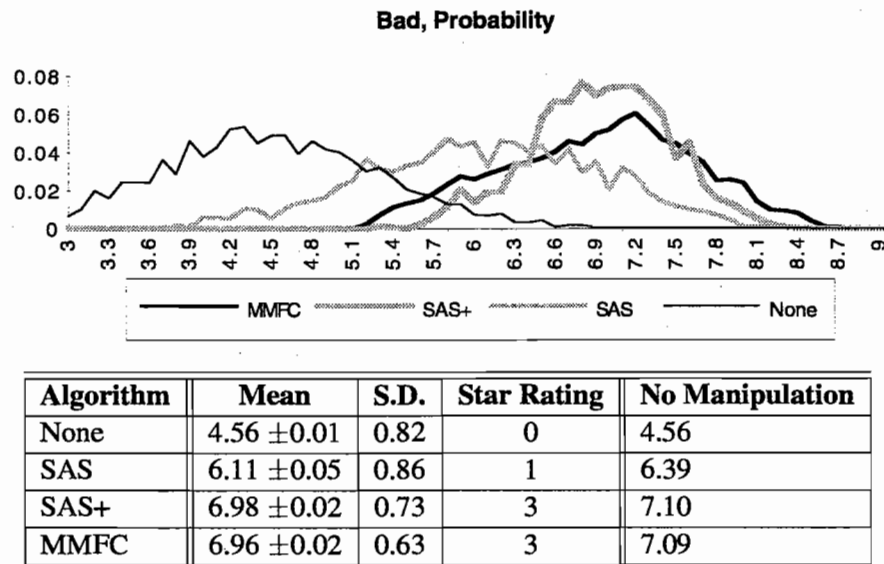


FIGURE 8.4: Bad User that Obeys Hints Probabilistically

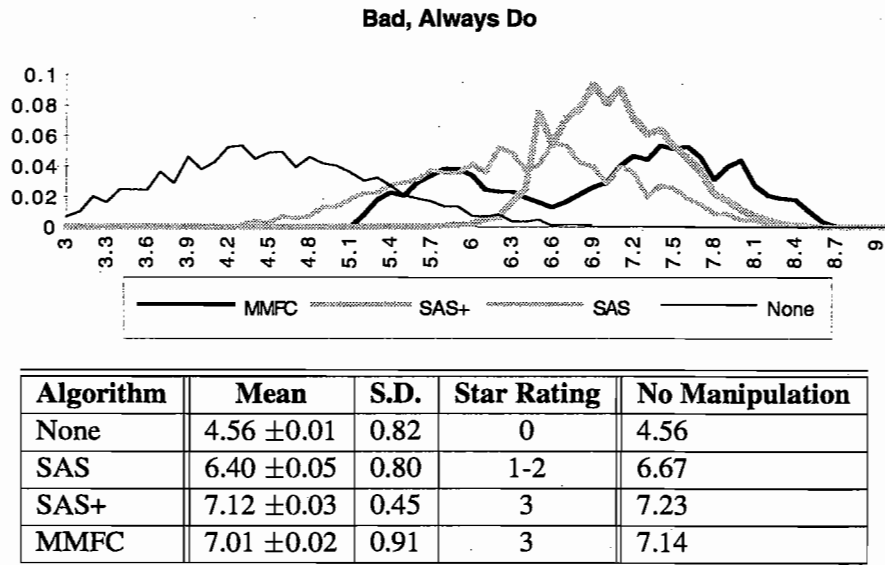


FIGURE 8.5: Bad User that Always Does Hints

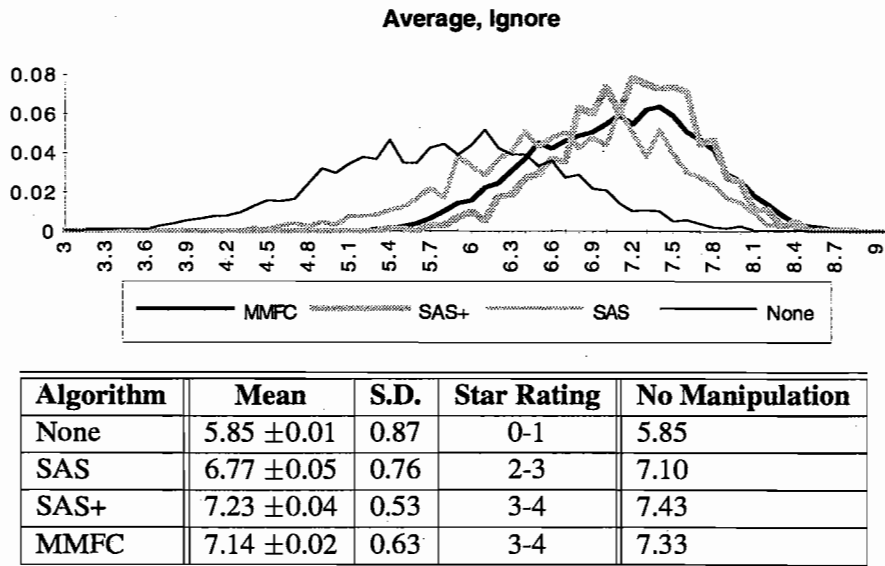


FIGURE 8.6: Average User that Ignores Hints

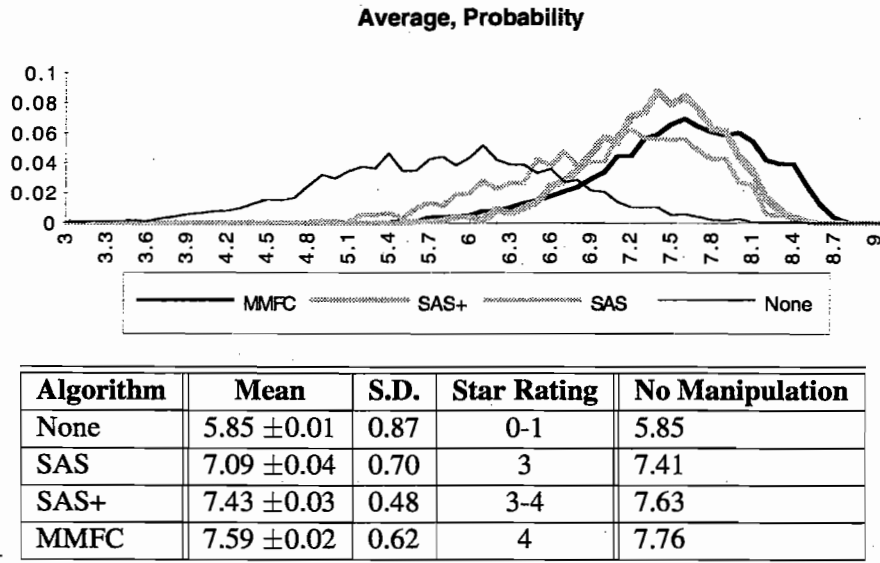


FIGURE 8.7: Average User that Obeys Hints Probabilistically

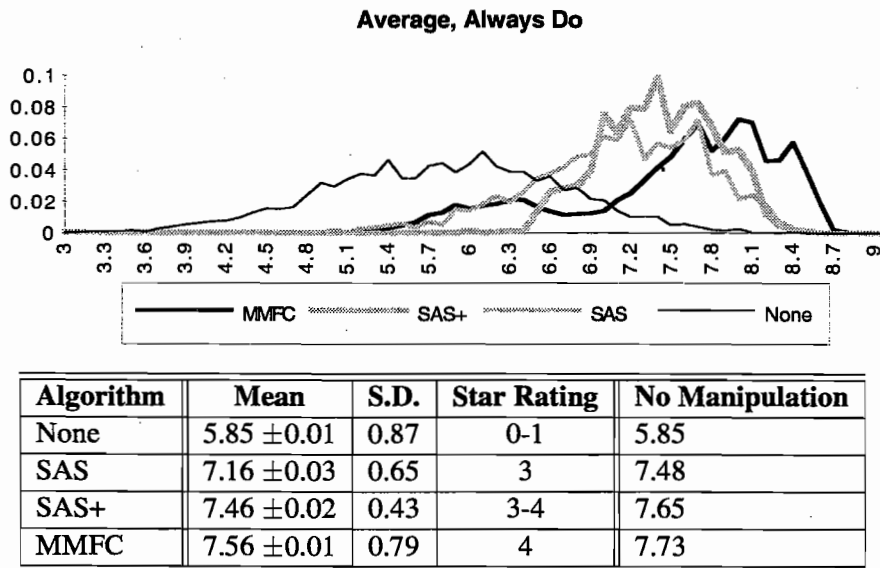


FIGURE 8.8: Average User that Always Does Hints

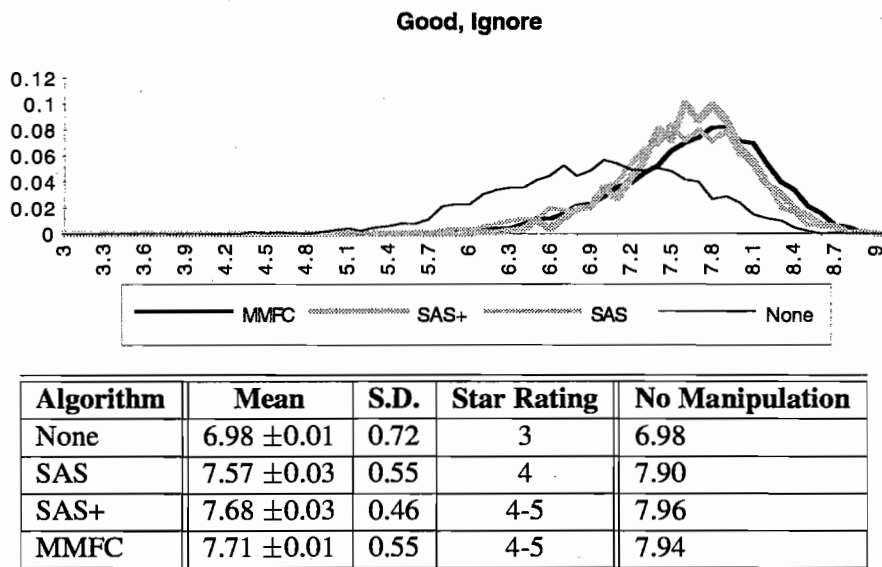


FIGURE 8.9: Good User that Ignores Hints

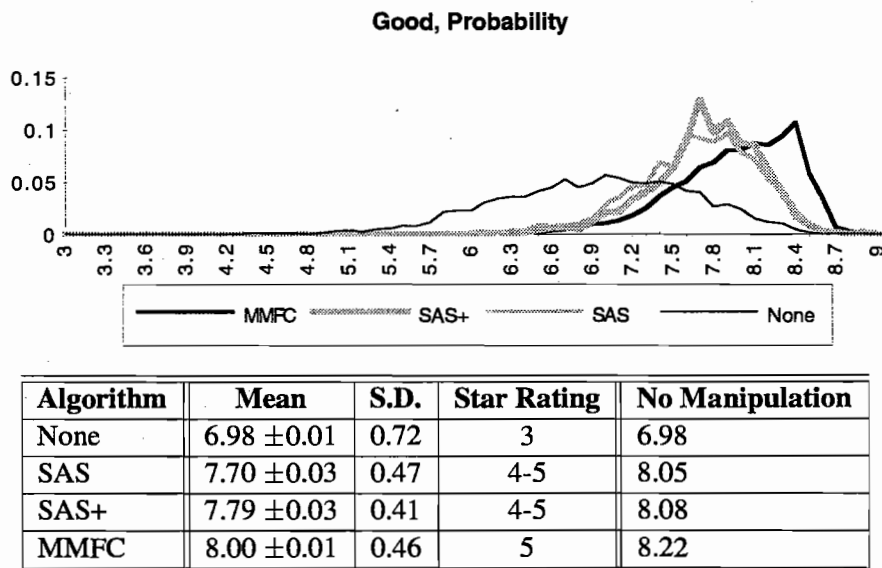
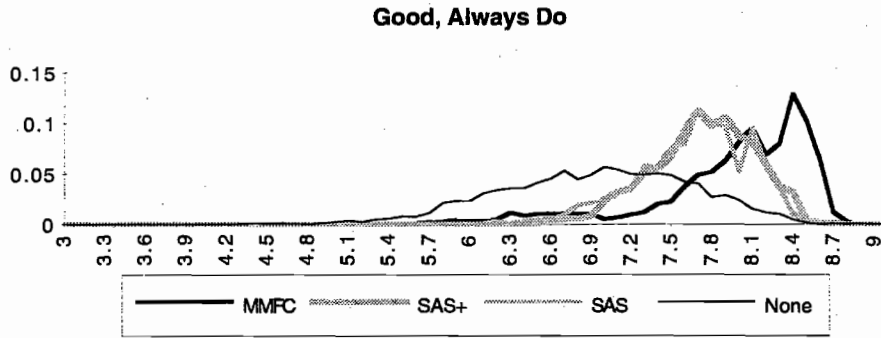


FIGURE 8.10: Good User that Obeys Hints Probabilistically



Algorithm	Mean	S.D.	Star Rating	No Manipulation
None	6.98 ±0.01	0.72	3	6.98
SAS	7.70 ±0.03	0.45	4-5	8.05
SAS+	7.78 ±0.03	0.38	4-5	8.05
MMFC	8.01 ±0.02	0.61	5	8.21

FIGURE 8.11: Good User that Always Does Hints

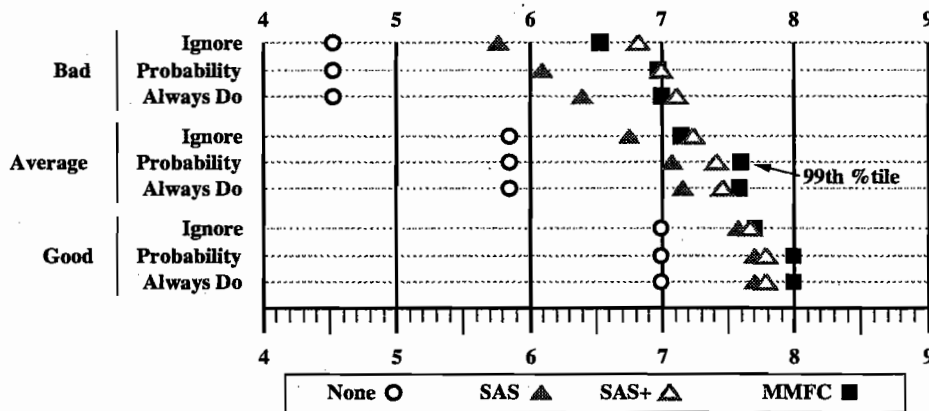


FIGURE 8.12: All Search Means Plotted vs. User Type

8.3 Discussion of Results

8.3.1 Search Algorithms Work

Figure 8.12 shows just the means of each algorithm for each of the nine types of Users. The vertical axis (y-axis) is User Types, while the horizontal axis (x-axis) shows the mean values for each of the four algorithms.

The basic result of this experiment is this: MMFC, SAS+, and SAS are doing a good job of providing dramatic guidance for simulated Users. As we can see in the Figure 8.12, these search strategies are significantly changing the mean ratings for User experiences.

Consider, for example, MMFC on “Average” “Probability” Users. MMFC is able to increase the mean User experience from 5.85, which represents zero to one stars, to 7.59, which represents a Star Rating of four stars. 7.59 is in the 99th percentile of ratings for this type of User without guidance. This is a large and artistically meaningful increase.

Even the worst algorithm, SAS, guiding Average Users that Ignore hints increases the mean rating of Users from 5.85 to 6.77, which represents an average Star-Rating between two and three stars.

As you can see, these algorithms are having an obvious beneficial effect. If this data were for a fully implemented system, that would mean that these search strategies were capable in practice of guiding a User’s experience in *Tea For Three*. Therefore, the degree to which we believe SUE is an accurate model of human Users is the degree to which we believe Moe is capable of guiding actual Users.

Therefore, as I have said many times before, this data, while not a proof, *suggests* that Moe is capable of guiding a human User’s experience in *Tea For Three*.

The rest of this section looks at various interpretations of the data. They supplement the main result.

8.3.2 Four Comments on this Evaluation

First, evaluating the search algorithm is different from evaluating the interactive drama as a whole. This is because the success of an interactive drama may be unrelated to guidance: it might primarily depend on characters, music, or graphics.

Second, the search algorithm should be judged by how well it caused the intended abstract experience to happen, not by whether or not that experience was pleasing to the User. By using the evaluation function I avoided the question of whether the intended effect was noticeable or enjoyed by the User. That is a different question altogether.

Third, in general having no dramatic guidance cannot be used as a benchmark. This is because, in general, an interactive drama might have been crafted specifically under the assumption that Moe must be used. Without Moe, the experience might be incomplete. For example, the experience may rely on certain MOE MOVES being refined. However, this method is applicable to *Tea For Three*, because *Tea For Three* does not depend on Moe’s

guidance. In *Tea For Three's* case, Moe simply tries to make the experience of the User better.

Finally, the results of the experiment indicate whether Moe is enhancing (on average) the experiences of the Users. There is no way to tell if a single, given User's experience will be better.

8.3.3 Observations About The Data

Here are three observations about the data. I do not know why these facts seem to be true, but useful future work might include trying to understand or explore them.

Manipulation

The average rating (over all User types) for MMFC is 7.39 with *Manipulation* and 7.56 without. Likewise, SAS+ gets 7.28 with and 7.55 without. It appears that on average SAS+ and MMFC can guide Users to the same level of experience, if the evaluation function did not consider *Manipulation*. So in a sense SAS+ and MMFC are achieving the same *Manipulation*-less results, but MMFC can do it using less *Manipulation*.

Standard Deviations

An interesting feature of this data is that SAS+ consistently has the smallest variance, as shown by the standard deviation. MMFC usually has the second smallest. In general the standard deviations tend to be largest with Average Users, and smallest with Good Users.

Bimodal Distributions

MMFC seems to have an odd behavior for Users that "Always Do" hints. For some reason, the distributions appear bimodal. In Figure 8.5 we find a pronounced bimodal distribution with modes at 5.9 and 7.5. In Figure 8.8 we also see a bimodal distribution, though it is less pronounced. Finally, by closely examining Figure 8.11 one can see the first mode has practically disappeared, but it does seem to be there in the 6.3 to 6.9 range. Future work might include trying to eliminate the first hump.

8.3.4 Speculations on MMFC vs. SAS+

Because MMFC is performing a full-depth search, one would expect MMFC to be the best algorithm for all types of Users. However, in several cases SAS+ beats MMFC. To me, this is surprising. This section contains my speculations on why MMFC is not better than SAS+ in all cases.

Long-term Strategy Not Needed

One possibility is that *Tea For Three* requires little strategic guidance. In *Tea For Three* there are only a few combinations of MOE MOVES that work together, and these pairs can be adequately employed by SAS+. This implies, in effect, that the problem is not so hard. A search algorithm that maximizes the local benefit of guidance can do as well as a strategic search.

One possible reason for this could be my choice of MOE MOVES. If the MOE MOVES are very powerful, it does not take a smart search to figure out how to use them. On the other hand, maybe the set of MOE MOVES is too weak and thus there are no strategic combinations of MOVES that are particularly effective. If MMFC could use better or different types of MOE MOVES, perhaps it could exceed SAS+ all the time.

As an aside, the fact that SAS+ works well implies that random sampling with $N = 32$ is a good heuristic for determining the expected value of an experience given no future guidance. So, it appears that if one is going to use a local search, one can effectively use random sampling.

Assumptions in MMFC Wrong

We can see in Figure 8.7 that for an Average User who Probabilistically follows Hints, MMFC has a higher mean than SAS or SAS+. In this case, the assumptions of the SUE and the assumptions in the search are the same.

In the cases where SAS+ beats MMFC, the search assumptions of MMFC are incorrect. This seems to be inhibiting MMFC's ability to guide *Tea For Three*. Perhaps since MMFC is the "smarter" algorithm, it is hurt more by incorrect models.

Future work in this area is to use these search algorithms for interactive dramas other than *Tea For Three* in order to see which of these speculations are true.

8.4 Conclusion to Part II

As I mentioned at the beginning of this chapter, my thesis, that *interactive drama is possible*, is supported by two pieces of evidence. First, that an evaluation function can encode an artist's aesthetic. Second, that an adversary search mechanism can effectively guide the User's experience by using this evaluation function.

Part I, on the evaluation function, supports the first piece of evidence. Part II supports this second piece of evidence. Chapter 5 contains a description of the MOE MOVES for *Tea For Three*. Chapter 6 describes the search state, which is the mechanism that the search uses to represent the User's concrete experience. Chapter 7 describes three different search strategies that use this search state. Finally, this chapter describes data that suggest Moe can use these three strategies to guide an interactive drama effectively.

In the next and final chapter, I summarize the contributions reported in this dissertation, and discuss future work.

Part III
Conclusion

Chapter 9

Conclusion

9.1 Contribution of the Thesis

This dissertation describes the first reported implementation of a system designed to provide centralized dramatic guidance in an *interactive drama*, as I've defined it.

There are two specific technical achievements in this work. The first is the demonstration of the ability to capture one dramatic aesthetic in an automated evaluation function. The second is the creation of three search strategies and a search state that seem capable of guiding interactive drama, based on this evaluation function. Both of these achievements include the ability to break down an experience into USER and MOE MOVES and annotate these MOVES with relevant information.

The evaluation function is the part of the complete interactive drama system that judges the aesthetic quality of a User's experience based on its abstract description. My achievement is to show that, to a useful degree, it is possible to encode a dramatic aesthetic into an evaluation function. This was demonstrated in Chapter 4 by showing a strong correlation between the artist's and evaluation function's ratings for an example set of experiences.

The evaluation function works by considering the abstract description of the User's experience from a variety of aesthetic perspectives. Each perspective is specified and implemented by the artist in an evaluation function feature. By assigning importance weights to the various features, the complete evaluation function can be calculated by adding the individual feature values in a weighted sum.

Part of my contribution is the set of features I created for *Tea For Three*. Although the set of features another artist might use for another interactive drama could be very different, I hope that my features can be used as inspiration for what types of features might be useful. This is especially true for the features that relate to interactivity, which is one of the important characteristics of interactive drama.

In order to create an evaluation function, an artist must break down the interactive drama into significant experience chunks, called USER MOVES. The choice of which USER MOVES to use is an artistic one. Again, I hope that the set of USER MOVES for *Tea For Three* can serve as a model for future artists.

The search program is the part of a complete interactive drama system that decides how and when to provide dramatic guidance. My contribution in search is to show that various search algorithms can be applied successfully to the problem of guiding the experience of simulated Users. Three search strategies have been implemented. SAS and SAS+ are based on a Sampling Heuristic Evaluation Function. MMFC is based on memoizing full-depth search. On “average” Users, SAS, SAS+, and MMFC are able to guide experiences from the 50th percentile, to the 94th, 98th, and 99th percentile, respectively. Future work is to determine if these results apply to implemented interactive dramas with human Users.

The search program works by projecting the future of a User’s experience through the abstraction of USER and MOE MOVES. A significant portion of my contribution is adapting adversary search to the domain of interactive drama. Doing this has included creating a search state to represent abstractly the concrete effects of the USER MOVES and six different types of MOE MOVES. By identifying interactive drama as a new domain, this work raises new issues and creates new problems in search.

In the same way that the evaluation function and USER MOVES provide a model, I hope that the search state, search algorithms, and MOE MOVES will inspire artists.

The challenge of interactive drama is to create a system capable of retaining the apparent freedom of the User, while ensuring that the User fulfills her Destiny. A real artistic contribution of this work is one definition of destiny and a suggestion (backed up by concretely implemented technology) that an artist can use this definition to take on the challenge of creating interactive drama. I hope I have turned the question of “Can we do this?” into “How should we do this?”

In 1991, Bates briefly mentioned the idea of using abstract adversary search with an evaluation function to guide an interactive drama.[4] My work is based on his original insight.

Besides Bates, my work is most closely related to Laurel’s work[21]. Both of us report similar goals. Where mine builds on hers is the creation of implemented mechanisms for providing dramatic guidance.

Other technical efforts in interactive drama have concentrated on decentralized approaches or ways to affect the unfolding of a fixed plot. Thus, there are no reported systems that I may compare directly to Moe.

9.2 Future Work, Limitations, and Improvements

This section contains a number of ideas for future work. They are organized by system component and listed (approximately) from most concrete to most speculative. This is not necessarily an order of importance or the order in which I would proceed. Of course, this is not an exhaustive list.

Tea For Three’s evaluation function does not match the artist as well as the artist matches himself. There are several ways to proceed to improve the evaluation function. Each might be done individually, or they could be used in combination. Not only would these techniques

be useful for improving *Tea For Three's* evaluation function, but I suspect they would be useful for different evaluation functions.

Evaluation Function Weights

The evaluation function is a weighted sum of feature values. Although I chose each weight carefully, these weights are probably not optimal for the features as implemented.

By using linear optimization techniques, I tried to determine the optimal weights, given the metric of minimizing the squared distance between the artist's ratings and the evaluation function's ratings. Unfortunately this technique is complicated by the non-linear nature of the transformation from the 0 to 10 scale (produced by the evaluation function) to the 0 to 6 star scale (used by humans.) For the data analyzed I found only slightly better weights than the one's I chose.

One area of future work is designing techniques for optimizing the weights based on empirical data supplied by the artist. He first gives reasonable weights, and then supplies a number of ratings, which are used to calculate the optimal weights. If an artist were so inclined, he might even let User's rate their impression of the experience and calculate the optimal weights according to their opinions.

Weighted Sum?

Optimizing weights is only one possible way to improve the evaluation function. Although a weighted sum has worked quite well, it might turn out that a weighted sum will never be capable of expressing the optimal evaluation function. For instance, a weighted sum cannot easily express the evaluation function of an artist that says "I need either *Thought Flow* OR *Activity Flow*, but not necessarily both."

This simple example suggests that other combination techniques besides summing should be investigated. For example, an artist might use other mathematical operations such as taking the product, maximum, or minimum of two feature values. Or, an artist might consider a boolean (logic) system, instead of a numeric system. Obviously different artists will want different things. Future work in this will entail artistic exploration of the many options.

More Accurate Features

Regardless of the weights or combination method, it is very important to have accurate features. Features necessarily have models, which are inherently approximate. Creating more specific and accurate models could make the features more accurate. This is true for all features of *Tea For Three*.

Consider, for example, the *Intensity* feature. The knowledge system employed by *Intensity* is imprecise and somewhat arbitrary. I could imagine implementing a more flexible knowledge system that would let an artist better represent the User's state of mind. This would involve more complication for the artist, but *Intensity* would be more accurate.

Of course, part of the design of features is to balance accuracy and complication. If a feature fails to capture the desired information, the artist should probably improve the accuracy just until it's good enough.

Feedback For Features

Like many of Moe's models, Moe's architecture itself is a simple first model. There are many ways that information might flow through the system to improve Moe's performance. One example of this is providing concrete information to the features.

Consider the *Activity Flow* feature. It could be made more accurate by allowing the recognition of USER MOVES to give feedback. Instead of just matching against the previous USER MOVE, *Activity Flow* could measure the intervening non-USER MOVE activity.

Another example is *Manipulation*. For MOE MOVES that have a *Manipulation* cost, the value is usually based on the expected values from several distinct outcomes. By knowing the outcome, the *Manipulation* can be more accurately calculated.

For example, suppose a MOE MOVE's *Manipulation* cost were based on three possibilities: refinement goes smoothly (no *Manipulation*), acceptably (some), or very badly (the most *Manipulation*). Once the MOVE has been refined, one of three values will be used. Notice, the search would still have to use the guess.

Partial Experience Evaluation

The Sampling Heuristic Evaluation Function (SHEF) gives the system a way to evaluate a partially completed experience by projecting futures. This is the basis of SAS and SAS+.

A better way to evaluate partial experiences might be to create evaluation function features specifically designed to judge them. Such features could be used to improve SAS by supplementing or replacing the SHEF. This is important because in the future Moe might abandon the avg-max assumption. In that case, SHEF (or its variations) may no longer work well.

Improving Moe's Models

There are many places in the entire system where simple models were used in order to facilitate the creation of the system. Though these choices are plausible approximations, they are fairly arbitrary and probably wrong. One category of future work is to investigate these models to determine if better models exist.

One model that needs to be investigated is the avg-max model for search. There are probably much better predictive procedures than just assuming a random distribution of next USER MOVES. A simple next step would be to keep records of many actual Users and then do context dependent prediction.

The same sorts of records could be kept for the effectiveness of MOE MOVES in guiding the User's experience. The values given as multipliers are quite speculative, and more correct models would presumably result in better decisions.

In fact, it would be nice to get feedback on a variety of parameters contained in Moe. For example, evaluation function weights, manipulation costs, search depths, and sample sizes. Exactly how to get the feedback is a difficult question, but a worthy subject of future research. If this process could at all be automated that would be good. It might be helpful to do a sensitivity analysis on all these parameters, in order to identify parameters to consider first.

MOE MOVES

The search results show search is not 100% effective. One of my speculations is that the set of MOE MOVES is not powerful enough. An obvious first step is to create better MOE MOVES in the types we already have. Perhaps more importantly, future work should involve creating more powerful types of MOE MOVES, especially those that might work in combinations. Even if creating new types does not improve the theoretical strength of Moe, artistically it may allow more subtle and flexible options for an artist creating MOE MOVES.

Improving Search Algorithms

Rather than pushing forward search research for other domains, my work in search has been an application of adversary search to a new domain. Future work in this area is to try to apply established search techniques to this domain, or create new ones.

In my work so far, I have considered various other techniques besides sampling and memoization. I tried various versions of forward pruning and beam search, but empirically I didn't find my application of them effective. Others might have more success. I considered using α - β [2], but ultimately rejected it because I felt the pruning would be limited under the avg-max assumption. A more careful analysis could lead to using α - β , or to the creation of more useful variations.

Connecting to a Simulated World

Probably, the most important next step is to connect Moe to a working version of *Tea For Three*. This involves implementing the physical world, characters, and presentation system, as well as creating the programs for refinement and recognition.

One challenge of implementing the refinement and recognition programs is the creation of interfaces between Moe and the various subcomponents. This is especially true of characters, because we do not want to compromise the believability of the characters for the sake of plot. Recently, Blumberg and Galyean have presented a potentially useful approach to the problem of directing autonomous characters[9].

An important part of this process is to attempt to understand how the results reported in this dissertation (for simulated Users) apply to an implemented system. This could involve coming up with new and different critical criteria for how to evaluate the different components.

A New, Graphical, Real-time Interactive Drama

After implementing *Tea For Three* I would next attempt to create a new interactive drama in a real-time graphical medium. Real knowledge will be gained by investigating the applicability of Moe to other types of interactive drama, other media, and other conditions. Here I present only a few, basic issues.

One requirement is the creation of a new evaluation function. As I've said before, some of the old features will be useful, but I suspect that new features will need to be created, especially if the interactive drama is significantly different in genre. It remains to be seen whether new feature creation is easy or hard.

The current evaluation function is based on the assumption that all sixteen USER MOVES must happen. I would probably relax this assumption in my next interactive drama. This means new features measuring the size and scope of the User's experience will need to be created.

This relates to a weakness of the current evaluation function. Because there is no representation of time passing in the abstract model, there are no features that deal with timing and pacing. This is important for a text-based model and becomes especially important in a real-time world.

Finally, Moe would have to be adapted to real-time requirements. Moe should probably be modified to make very quick, but perhaps non-optimal decisions. This would be followed by longer and longer computations as time permits, which will presumably yield better decisions. A new way of judging search algorithms would then become appropriate, based on how quickly it can make a reasonable decision.

9.3 Future Of The Art

Currently there is a large amount of debate about interactive drama as an artistic medium. The first question is whether it is possible to create interactive drama on a computer, and the second question is whether people will really enjoy it at all. My personal opinion is that interactive drama is possible, and that many people will enjoy it.

Critics cite many reasons why interactive drama is not possible. The basic objection is that there is no way to allow the User freedom to act AND to give the User a dramatic experience. There is an apparent contradiction between these two aspects of the User's experience.

As I argued in Chapter 3, I agree that there is no way to employ the traditional techniques of dramatic control used in non-interactive media. What I have proposed instead is that the artist give up direct control of the sequence of events and instead define the interactive drama with a destiny. The technology of this dissertation suggests that freedom and destiny are not contradictory.

I believe that not only are freedom and destiny consistent, but that future artists should be trying to exploit the freedom of the User to make better dramatic effect. This can happen in two ways.

The first is identified by a phrase: *Freedom Means Control*. The artist of the future should ultimately realize that the User can be given a more powerful experience by observing her actions. Just as skilled salesman is more effective than an advertisement and a skilled conversationalist can be more effective than a leaflet, a skilled interactive drama system should be more effective than a non-interactive medium. By observing the actions and reactions of the User, the system can customize the presentation, characters, music, and drama to an individual User. Interactive drama can be seductively successful.

The second way that freedom can be used is to make Users confront situations directly. Seeing the destruction of a relationship in a movie may be emotional, but it doesn't compare to having your own relationship end. I am not suggesting that interactive drama will ever be as intense as the dramatic moments of real life, but I am saying that making choices and taking action is difficult. How powerful would it be if an artist could make a User search into her soul for an answer about herself? Non-interactive media do not directly require choices or actions. Interactive dramas can demand these of their Users.

I believe the future of interactive drama will include an exploration of how to best exploit these two properties. This medium has the chance to be among the most powerful. Let's get to work.

Part IV
Appendices

Appendix A

Other Rater Comparisons

Statistical Comparison: Bates vs. Bates																																																																		
Star Rating Scatterplot	Linear Regression	Agreement Coefficients:																																																																
<table border="1"> <tr><td>6</td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>2</td></tr> <tr><td>5</td><td></td><td></td><td>1</td><td></td><td></td><td>8</td><td>1</td></tr> <tr><td>4</td><td></td><td></td><td>2</td><td>4</td><td>10</td><td></td><td>1</td></tr> <tr><td>3</td><td></td><td>1</td><td>4</td><td>8</td><td>4</td><td></td><td></td></tr> <tr><td>2</td><td></td><td>4</td><td>2</td><td>4</td><td>2</td><td>1</td><td></td></tr> <tr><td>1</td><td></td><td>4</td><td>4</td><td>1</td><td></td><td></td><td></td></tr> <tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> <p style="text-align: center;">Bates</p>	6					1	1	2	5			1			8	1	4			2	4	10		1	3		1	4	8	4			2		4	2	4	2	1		1		4	4	1				0									0	1	2	3	4	5	6	$r = .72$ $r^2 = .52$ <i>slope</i> = .72 <i>intercept</i> = .901 <i>t-test</i> computed <i>t</i> value: 8.639 required <i>t</i> value: 3.46 Passes at over 99.9% $N = 70$	$M_{\leq 0} = .49$ $M_{\leq 1} = .86$ $\kappa = .48$ $\kappa_w = .72$ $A_R = .86$ Average Distances: $d^1 = .69$ $d^2 = 1.1$
6					1	1	2																																																											
5			1			8	1																																																											
4			2	4	10		1																																																											
3		1	4	8	4																																																													
2		4	2	4	2	1																																																												
1		4	4	1																																																														
0																																																																		
	0	1	2	3	4	5	6																																																											

Statistical Comparison: English Prof vs. English Prof																																																																		
Star Rating Scatterplot	Linear Regression	Agreement Coefficients:																																																																
<table border="1"> <tr><td>6</td><td></td><td></td><td></td><td>2</td><td>1</td><td>3</td><td>4</td></tr> <tr><td>5</td><td></td><td></td><td>2</td><td>5</td><td>4</td><td>4</td><td>3</td></tr> <tr><td>4</td><td>2</td><td>2</td><td></td><td>4</td><td>4</td><td>4</td><td>1</td></tr> <tr><td>3</td><td></td><td>2</td><td></td><td>2</td><td>4</td><td>5</td><td>2</td></tr> <tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td>2</td><td></td></tr> <tr><td>1</td><td>1</td><td></td><td></td><td>2</td><td>2</td><td></td><td></td></tr> <tr><td>0</td><td></td><td>1</td><td></td><td></td><td></td><td>2</td><td></td></tr> <tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table> <p style="text-align: center;">English Prof</p>	6				2	1	3	4	5			2	5	4	4	3	4	2	2		4	4	4	1	3		2		2	4	5	2	2						2		1	1			2	2			0		1				2			0	1	2	3	4	5	6	$r = .25$ $r^2 = .06$ <i>slope</i> = .25 <i>intercept</i> = 2.92 <i>t-test</i> computed <i>t</i> value: 2.119 required <i>t</i> value: 3.46 FAILS at 99.9% $N = 70$	$M_{\leq 0} = 0.2$ $M_{\leq 1} = .54$ $\kappa = 0.2$ $\kappa_w = .25$ $A_R = .62$ Average Distances: $d^1 = 1.5$ $d^2 = 3.7$
6				2	1	3	4																																																											
5			2	5	4	4	3																																																											
4	2	2		4	4	4	1																																																											
3		2		2	4	5	2																																																											
2						2																																																												
1	1			2	2																																																													
0		1				2																																																												
	0	1	2	3	4	5	6																																																											

Statistical Comparison: Evaluation Function vs. Bates																																																																		
Star Rating Scatterplot	Linear Regression	Agreement Coefficients:																																																																
Bates	$r = .72$ $r^2 = .52$ $slope = 0.5$ $intercept = 1.75$ <u>t-test</u> computed t value: 8.529 required t value: 3.46 Passes at over 99.9% $N = 70$	$M_{\leq 0} = .27$ $M_{\leq 1} = .73$ $\kappa = .27$ $\kappa_w = .67$ $A_R = .83$																																																																
		Average Distances:																																																																
		$d^1 = 1.1$																																																																
		$d^2 = 2.0$																																																																
		random $d^2 = 3.1$																																																																
		3's $d^2 = 2.0$																																																																
<table border="1"> <tr> <td>6</td><td></td><td></td><td></td><td></td><td></td><td>2</td><td>2</td> </tr> <tr> <td>5</td><td></td><td></td><td></td><td></td><td>2</td><td>5</td><td>3</td> </tr> <tr> <td>4</td><td></td><td></td><td>1</td><td>5</td><td>6</td><td>2</td><td>3</td> </tr> <tr> <td>3</td><td>2</td><td>7</td><td>3</td><td>2</td><td>2</td><td></td><td>1</td> </tr> <tr> <td>2</td><td>2</td><td>3</td><td>4</td><td>2</td><td></td><td>1</td><td>1</td> </tr> <tr> <td>1</td><td>6</td><td></td><td>2</td><td>1</td><td></td><td></td><td></td> </tr> <tr> <td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td> </tr> </table> Evaluation Function	6						2	2	5					2	5	3	4			1	5	6	2	3	3	2	7	3	2	2		1	2	2	3	4	2		1	1	1	6		2	1				0									0	1	2	3	4	5	6		
6						2	2																																																											
5					2	5	3																																																											
4			1	5	6	2	3																																																											
3	2	7	3	2	2		1																																																											
2	2	3	4	2		1	1																																																											
1	6		2	1																																																														
0																																																																		
	0	1	2	3	4	5	6																																																											

Statistical Comparison: Evaluation Function vs. English Prof																																																																		
Star Rating Scatterplot	Linear Regression	Agreement Coefficients:																																																																
English Prof	$r = .25$ $r^2 = .06$ $slope = 0.2$ $intercept = 3.3$ <u>t-test</u> computed t value: 2.128 required t value: 3.46 FAILS at 99.9% $N = 70$	$M_{\leq 0} = .17$ $M_{\leq 1} = 0.4$ $\kappa = .17$ $\kappa_w = .22$ $A_R = .59$																																																																
		Average Distances:																																																																
		$d^1 = 1.9$																																																																
		$d^2 = 5.7$																																																																
		random $d^2 = 3.8$																																																																
		3's $d^2 = 3.3$																																																																
<table border="1"> <tr> <td>6</td><td></td><td>1</td><td>2</td><td>2</td><td>2</td><td>1</td><td>2</td> </tr> <tr> <td>5</td><td>3</td><td>1</td><td>3</td><td>5</td><td>2</td><td>3</td><td>1</td> </tr> <tr> <td>4</td><td>2</td><td>4</td><td>1</td><td>2</td><td>3</td><td>2</td><td>3</td> </tr> <tr> <td>3</td><td></td><td>2</td><td>2</td><td>1</td><td>2</td><td>4</td><td>4</td> </tr> <tr> <td>2</td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td>2</td><td></td><td>1</td><td></td><td></td> </tr> <tr> <td>0</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td> </tr> </table> Evaluation Function	6		1	2	2	2	1	2	5	3	1	3	5	2	3	1	4	2	4	1	2	3	2	3	3		2	2	1	2	4	4	2	2							1	1	1	2		1			0	2	1							0	1	2	3	4	5	6		
6		1	2	2	2	1	2																																																											
5	3	1	3	5	2	3	1																																																											
4	2	4	1	2	3	2	3																																																											
3		2	2	1	2	4	4																																																											
2	2																																																																	
1	1	1	2		1																																																													
0	2	1																																																																
	0	1	2	3	4	5	6																																																											

Appendix B

LegalWhen Functions for MOE MOVES

List of all LegalWhen expressions:

MOE MOVE 1: REMOVE DUNBAR:

 happened (AFL)
 and not happened (CONFRONT DUNBAR)
 and not happened (MOE MOVE 2: DUNBAR CONFRONTS USER)

MOE MOVE 2: DUNBAR CONFRONTS USER:

 happened (AFL)
 and not happened (CONFRONT DUNBAR)

MOE MOVE 3: DUNBAR CONFESSES TO USER:

 happened (CONFRONT DUNBAR)
 and not happened (TICKET/AFFAIR)

MOE MOVE 4: BAXTER EXPLAINS MERGER:

 not happened (MERGER)

MOE MOVE 5: GEORGE CONFRONTS USER:

 happened (CALENDAR)
 and not happened (CONFRONT GEORGE)

MOE MOVE 6: GEORGE IS CAUGHT:

 happened (CONFRONT GEORGE)
 and not happened (CATCH GEORGE)

MOE MOVE 7: DELAY THE AFL REPORT:

happened (LOBLO)
and happened (FRAGS)
and not happened (AFL)

MOE MOVE 8: RETURN THE AFL REPORT:

delayed (AFL)
and not happened (AFL)

MOE MOVE 9: COMBINE AFE AND AFL:

not happened (AFL)
and not happened (AFE)
and happened (FRAGS)
and happened (LOBLO)

MOE MOVE 10: DESCRIBE HOLES AND FRAGS TOGETHER:

not happened (HOLES)
and not happened (MOE MOVE 16: FRAGS TO SHED)

MOE MOVE 11: GEORGE SHOOTS SKEET:

not happened (MUD)
and current act = search scene

MOE MOVE 12: GEORGE GOES ON BALCONY:

current act = search outside
and not happened (MUD)

MOE MOVE 13: GEORGE CONFESSES SECRET ROOM:

not happened (CATCH GEORGE)
and not happened (FOCUS)

MOE MOVE 14: DESCRIBE THE MUD AND SCRAPED PAINT TOGETHER:

not happened (MUD)
and not happened (PAINT)