43

An Architecture for Character-Rich Social Simulation

Michael Mateas and Josh McCoy

43.1 Introduction

More progress has been made creating NPCs that engage in purely autonomous activity or small-group behavior organized around combat (e.g., squad behavior) than creating NPCs that participate in ongoing social activity. Social activity often has no explicit functional goal, but rather involves social actors responding to, displaying, and changing social state. Social action strongly depends on the history of previous social acts. Complex social actions are often best done through dialog, making use of the full richness of natural language to refer to feelings, relationship states, and history. Finally, social action is embedded in a rich social context—actions often have ramifications across multiple social actors. These features of social action are not easily satisfied by NPC architectures that emphasize individual decision making focused on moment-by-moment action selection to accomplish primarily functional goals.

This chapter describes the Comme il Faut (CiF) social simulation architecture. CiF was used to create *Prom Week*,* a social puzzle and storytelling game that was a technical

* The design and implementation of *Prom Week*, as well as significant development of CiF, was carried out by a dedicated core team consisting of Josh McCoy, Aaron A. Reed, Ben Samuel, Mike Treanor, Michael Mateas, and Noah Wardrip–Fruin, and by a larger team who provided additional programming, writing, art, music, sound, and animation work. A full credit list is available at http://promweek.soe.ucsc.edu/?page_id=25.

excellence nominee in the Independent Game Festival at GDC 2012 and a nominee for IndieCade 2012. Comme il Faut, which in French roughly translates as "as it should be," satisfies the requirements outlined above. In CiF the concept of a multicharacter social interaction is a first-class AI construct. Characters use many details of the social state, including the history of prior interactions, to decide how to participate in these multicharacter social exchanges. The system automatically retargets dialog within these exchanges to the particularities of specific characters with their detailed personalities and history in specific social situations. The goal of this architecture is to enable casts of characters to engage in rich social interaction, speaking the kind of concrete dialog typically associated with hand-authored dialog trees, but utilizing a level of emergent social simulation typically associated with simulation games.

The rest of this chapter first introduces the released experimental game *Prom Week*, which uses CiF to support a form of interactive storytelling in which the gameplay revolves around solving social puzzles to accomplish story goals. The bulk of the chapter describes CiF's major architectural elements. The chapter then concludes with a description of different interaction styles that a CiF-like architecture can support and a description of related work. The goal of this chapter is to provide enough detail to allow the reader to borrow elements of the CiF architecture in their own work, without drowning the reader in implementation details.

43.2 CiF and *Prom Week*

While an early version of CiF was developed as a stand-alone social simulation system [McCoy and Mateas 09], the architecture and authoring approaches evolved considerably with the development of *Prom Week* [McCoy et al. 10a, McCoy et al. 10b]. Throughout the rest of this chapter, all the examples of different kinds of CiF authoring (knowledge representation) will be from *Prom Week*. Of course, if a CiF-like architecture was used in a different game, different content would be authored, but creating the CiF architecture in the context of creating a complete, playable experience forced us to scale the architecture to the full complexities of authoring and development.

The initial inspiration for developing CiF came from authoring limitations experienced during the creation of the interactive drama *Façade* [Mateas and Stern 02; Mateas and Stern 03]. *Façade* is an interactive drama in which the player, from a first-person perspective, has a short 20-minute interaction with a couple whose marriage is falling apart. *Façade*'s character behaviors are organized around dramatic beats, approximately 1-minute interactions in which the characters work together to convey some aspect of the story or their personality. In *Façade* these behavior clusters were authored for specific characters expressing specific content [Mateas and Stern 07]. Variation in performance was implicitly encoded in behaviors and was not reusable between characters. Our work on CiF started with the goal of generalizing multicharacter exchanges into *reusable* units that support character-specific performance variation *explicitly*. Dynamically retargeting animations allows animation authoring effort to be reused across multiple characters. Similarly, we want to enable dynamically targeted NPC dialog, in which multicharacter dialog performances are authored more generally, and then targeted to specific characters

in specific situations. Starting with this initial goal, our final architecture provides the following major features:

- Multicharacter social exchanges are explicitly represented separately from any specific characters. Given a cast of characters, with traits and social state declaratively represented, social exchanges are retargeted for specific characters.
- Characters decide what they want to do and who they want to do it with based on soft decision making (not Boolean flags or rigid preconditions) that can take into account hundreds of considerations.
- Social interactions don't just cause a single change in social state, but have cascading consequences across multiple characters.
- The performance details and outcomes of previous social interactions are stored in an episodic memory and used both during social exchange performances (social exchange dialog can refer to past performances) and to help determine which social exchanges characters want to engage in.

Together, these properties provide the foundation for simulating a *social* world, in which characters are embedded in a constantly evolving sea of social state. After first briefly describing *Prom Week*, we will then walk through the details of these different elements of the architecture.

43.2.1 Prom Week

In *Prom Week*, available for free at promweek.soe.ucsc.edu, the player controls the lives of 18 high school students during the week leading up to the prom. The gameplay consists of the player selecting pairs of characters and choosing a social exchange that the first character initiates with the second. Given a selected pair of characters, CiF determines what actions the first character most wants to initiate, which are presented in a menu to the player, and how the second character responds to a selected interaction. In Figure 43.1, the player is choosing among the five social exchanges CiF most wants to perform between the two selected characters, and is currently highlighting Pick-Up Line, an exchange with the goal of initiating a dating relationship. Once a social exchange is selected, CiF decides how the two characters will perform the social exchange. Figure 43.2 shows an in-progress performance of a Brutal Break-Up, a social exchange with the intention of terminating a dating relationship. The player is trying to accomplish story goals, such as having a character date a specific other character, make a certain number of friends, or have a certain number of good or bad things happen to them. The story goals for one of the *Prom Week* scenarios are shown in Figure 43.3. Because of the underlying social modeling performed by CiF, story goals have multiple emergent, nonprescripted solutions, and different combinations of story goals accomplished by the player open up different endings at the prom. In this way, *Prom Week* combines storytelling and character exploration with simulation-based gameplay.

To help the player master the underlying social system, the *Prom Week* interface allows the player to explore the reasons why characters were motivated to initiate a social exchange, why they responded to a social exchange the way they did, why social exchanges caused the effects they did, and the current social state. Figure 43.4 shows the interface the player can use to explore why a social exchange happened the way it did. The explanations

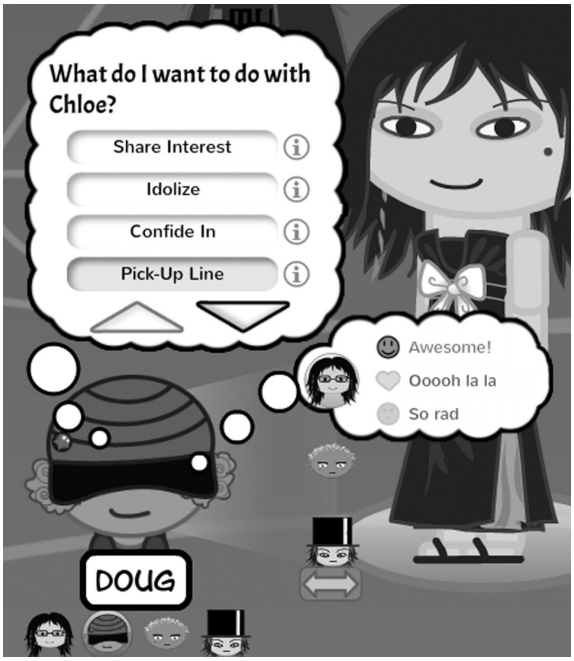

Figure 43.1

Selecting a social exchange.

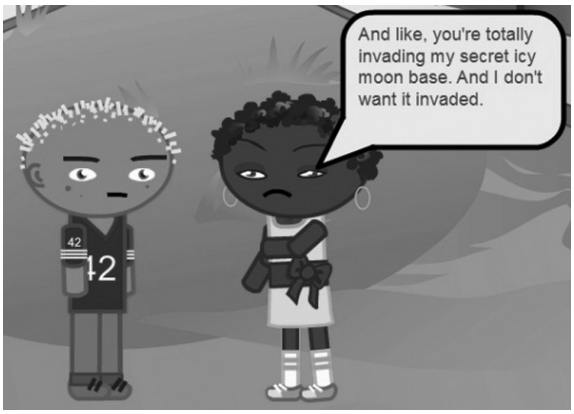

Figure 43.2

Social exchange performance.

are generated from an analysis of the most important rules that fired to influence the exchange. Figure 43.5 shows the interface the player can use to explore the current social state, guiding strategic selection of characters and social exchanges. Finding the right amount of complex simulation state to expose to the player in a usable fashion was one of the major design challenges for *Prom Week*.

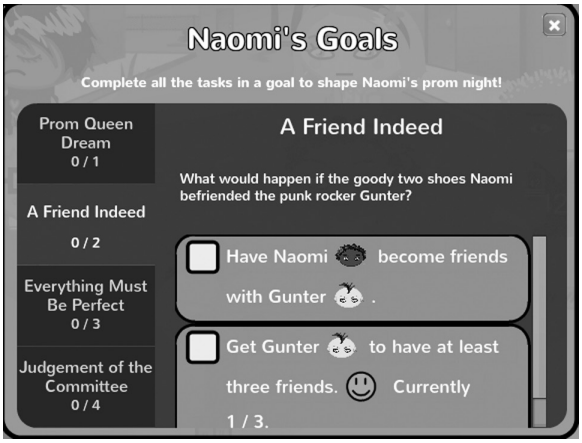

Figure 43.3

Story goals.

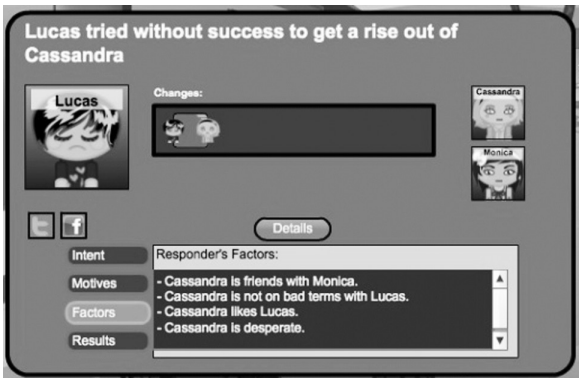

Figure 43.4

Outcome explorer interface.

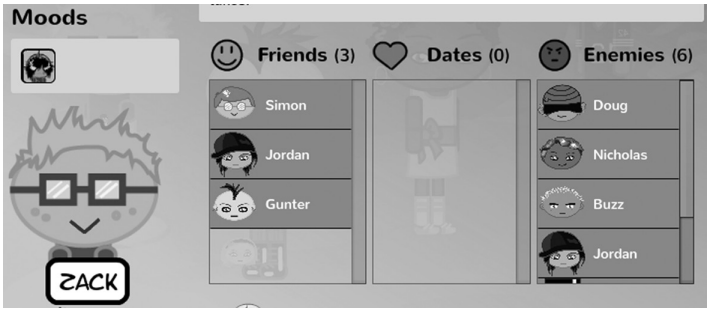

Figure 43.5

Social state explorer interface.

As the player makes successful social exchanges happen, they earn social influence points. Social influence points can be spent to look ahead at the outcome of a social exchange, the factors influencing the social exchange, and/or to override the outcome. Strategic management of social influence points becomes another element of the gameplay.

43.3 CiF Architecture

The primary knowledge representation element in CiF is the social exchange, a collection of patterns of (primarily dialog) interaction where the exact performance and social outcome varies based on the personality-specific attributes of the characters involved and the current social state. The idea of social exchanges was inspired by Erving Goffman's work in sociology on dramaturgical analysis of social interaction [Goffman 59]. Goffman viewed social interaction in everyday life as dramatic performances designed to express properties of one's personality or to change social state. He used the metaphor of theater to analyze how people draft others to perform roles in their performances, organize space and objects into a "stage" and "props" for a performance, and so forth. In CiF, social exchanges comprise a taxonomy of performances organized around what elements of social state they are designed to change or express. Before diving into the details of social exchanges, it is necessary to first understand how characters and social state are represented.

43.3.1 Characters

Characters consist of the elements described in Table 43.1.

Due to the emphasis being placed on social exchanges, the representation of each character is relatively thin, consisting of a small amount of declarative information (that is, information that specifies to the underlying AI system the properties of the character, but leaves it to the system to figure out *how and when* to express these properties). What makes a character rich and unique is their situation in the social world and their history, rather than a bunch of character-specific AI behavior.

Traits and statuses describe the personality and character state of a character. Traits are permanent properties of a character which heavily impact social exchange play. Given a character `x`, examples include `trait(brainy, x)`, `trait(stubborn, x)`, `trait(attention hog, x)`, and `trait(sex magnet, x)`. Traits only have meaning to the degree that they are referenced in preconditions and influence rules to determine which social exchanges a character wants to initiate and to select among different instantiations of social exchanges.

Table 43.1 Character representation

Character	
Name	The character's name.
Gender	The gender of the character.
Traits	A vector of the character's traits.
Statuses	A vector of the statuses held by the character.
Prospective memory	The character's desires to play social exchanges, represented as a vector of volitions.
Character-specific phrases	Character-specific natural language generation template fill-ins (greetings, pejoratives, exclamations, etc.).

Statuses are temporary, optionally directional, binary social effects that result from social exchange play. Statuses capture transitory states in an agent's mood (e.g., `status(cheerful, x)`), sharp spikes of emotion between agents (e.g., `status(hasACrushOn, x, y)`), and social states (e.g., `status(popular, x)`). They are useful in capturing transitory but potent social situation and character states. When a status is posted by a social exchange, it includes a duration specification of how long the status lasts before it times out.

Prospective memory is a vector of numeric volitions (desires) for characters to engage in specific social exchanges with specific characters. These volitions are computed by initiator influence rules, as described in the *Social Exchanges* section below.

Character-specific phrases are filled into dialog templates in social exchanges. This is one way that the general social exchange dialog is retargeted to specific characters. The character-specific phrases defined in *Prom Week* include `%greeting%` (opening greeting phrase), `%shocked%` (expression of shock), `%positiveAdj%` (positive adjective), `%pejorative%` (a nasty thing to call someone), and `%sweetie%` (a term of endearment). Whenever any of these tags appear in social exchange dialog, they are replaced by the appropriate character-specific phrase. A character definition should include one or more phrases for each of the character-specific tags used in social exchanges.

43.3.2 Social State

The social state of the world is captured by four different representations: social networks, relationships, the cultural knowledge base, and the social facts knowledge base.

Social networks are bidirectional fully connected networks where the edge values measure the feelings between characters. *Prom Week* has three networks: a romance network, which represents how interested characters are in pursuing intimate relationships with each other, a friendship network, which represents how much characters like each other, and a "coolness" network, which represents how much respect characters have for one another. If `x` has a romance network value of 80 towards `y`, but `y` only has a value of 20 towards `x`, the agents see their situation differently. Social network values are the private feelings characters feel towards each other. Just because two characters have mutually high friendly feelings towards each other does not mean that they automatically have the publicly recognized social state of being friends. Characters with high mutual friendly feelings will be strongly inclined to engage in a social exchange that results in them becoming friends, but a social exchange is necessary to enact the process of turning mutual private friendly feelings into a publicly recognized friendship (and similarly for romance, etc.).

Relationships represent publicly recognized social relationships between characters. In *Prom Week* the three relationships are friends, dating, and enemies. Unlike social network values, which have a numeric range, relationships are binary: two characters either are or aren't friends, are or aren't dating, are or aren't enemies. The distinction between social networks and relationships enables the representation of dramatically interesting (and lamentably true to life) states. For example, given three characters `x`, `y`, and `z`, CiF can represent states such as `relationship(dating, x, y)`, `network(romance, x, y, 20)`, `network(romance, y, x, 95)`, and `network(romance, x, z, 80)`, which translates into `x` and `y` are dating, `y` is head over heels in love with `x`, while `x` has fallen out of love with `y` but has eyes for a third character, `z`. Given the initiator and responder influence rules we developed for *Prom Week* (see *Social Exchanges* section, below), this state would

give x higher volition to want to break up with y but make y more likely not to accept the breakup (to interpret it as a joke or a temporary fight), make x more likely to want to flirt with z to increase romance and possibly to start dating (which would have the consequence of making y angry and lowering y's romance feelings towards x), and make y more likely to want to initiate positive exchanges with x to increase romance but also to initiate negative exchanges with x out of jealousy or anger (e.g., if y sees x flirt with someone else). The relationship representation also allows such complex social relationships as being friends and enemies at the same time (frenemies) – our goal was to support rich and dramatically interesting social states.

The cultural knowledge base (CKB) is a way to further define the world that CiF-driven agents inhabit, providing them with a variety of topics to bond over and squabble about The design intent for creating the CKB is to have a representation of props that is sociologically rich. As props are much more than simple physical objects in dramaturgical analysis, CiF needs a way to understand the cultural importance of items in relationship to the storyworld. The CKB used for *Prom Week* has many items, including zombie movies, chainsaws, and webcomics. Every agent has one or more connections to these items, linked through the uni-directional phrases *likes*, *dislikes*, *wants*, and *has*. Gunter dislikes bobbleheads. Oswald likes web comics. Phoebie likes zombie movies. Additionally, every object in the CKB can be associated with universally agreed-upon properties in the social world (e.g., chainsaws are bad ass, dodgeball is mean). This allows for agents to interact with each other based on their individual opinions of objects in the world. The CKB can be queried to search for patterns of attitudes characters hold for objects. Consider the example query:

```
CKB(item,(x, likes), (y, dislikes), lame)
```

There are four parts (only one has to be specified) to a CKB query: 1) the item to look for, `item`, 2) the first subjective label, e.g., `(x, likes)`, 3) the second subjective label, e.g., `(y, dislikes)`, and 4) the truth label, e.g., `lame`. This query will match an item that x likes, y dislikes, and is universally regarded as `lame`, which could perhaps contribute to y's volition to poke fun at x.

Finally, the social facts knowledge base (SFKB) keeps track of the social history of the story world so that it can be queried for socially relevant information. The SFKB influences what social exchanges characters want to engage in and how they respond, and also affects performances of social exchanges, where the specifics of previous exchanges are brought up in conversation. Typically, games don't record much of the interaction history and use it to make decisions or refer to it in character dialog. By making this history a part of the architecture, CiF provides history-dependence in the social simulation that goes far beyond what can be accomplished with ad hoc techniques such as flags.

The SFKB stores an entry for every social exchange played and for every trigger rule that causes social state change. This entry includes the details of who was involved, as well as any items from the cultural knowledge base mentioned, a natural language generation template that can be used to turn the entry into text for use in performances, and an abstract social exchange label (such as *mean*, *funny*, *nice to*) that can be used for querying. For example, if Edward initiates the Bully social exchange with Chloe, engag-

ing in a specific instantiation in which x makes fun of y's SAT score, the following entry is stored in the SFKB.

```
(SocialGameContext exchangeName = "Bully" initiator = "Edward" responder =
"Chloe" initiatorScore = "15" responderScore = "10" time = "5" effectID =
"10" other = ""(SFKBLabel type = "mean"))
```

This entry records the name of the social exchange, the characters involved, the scores computed by the initiator and responder influence rules, a time stamp (for *Prom Week* this is discrete turns, but it could be continuous time as well), an effect ID that indicates which specific instantiation of Bully happened (in this case, making fun of SAT scores), and a label saying that this was a "mean" exchange. SFKB labels are used to support more abstract queries. Rather than querying for specific exchange names or instantiations, often a query only cares about the general tone, finding past events in which one character did something mean, or nice, or lame, etc., to another character. For example, if Chloe later initiates a Backstab exchange with Edward, an instantiation of backstab can query for past mean things Edward did to Chloe: `[SFKBLabel(mean, responder, initiator,0) window(10)]`, in this case looking through the past 10 social exchanges. The matched entry is then turned into text, resulting in the performance "You know when you made fun of my SAT score?", which the initiator (in this case Chloe) says just before revealing what she's done to get even. In this way, the SFKB supports a compounding effect of history, where the characters refer more and more to past events that have happened, with the past events affecting decision making.

43.3.3 Social Exchanges

Social exchanges are the heart of CiF. The rich social state described in the previous section exists to provide socially interesting reasons for characters to initiate social exchanges; the exchanges then change the social state. Table 43.2 describes the components of a social exchange.

The intent captures the initiator's purpose for initiating the social exchange. These purposes involve changing social state, such as increasing another character's friendship feelings towards the initiator (a social network value change) or initiating or terminating a relationship (such as a dating relationship). If one views social exchanges as complex plan operators, the intent is like the postcondition of the operator. The difference from a

Table 43.2 Social exchange structure.

Social exchange	
Name	The name of the social exchange ("Ask Out," "Text Message Breakup," etc.).
Intent	The intended social change associated with the social exchange.
Precondition	A condition that must be true for the social exchange to be applicable in the current social environment.
Initiator influence rules	The social considerations applicable to the initiator of this social exchange.
Responder influence rules	The social considerations applicable to the responder of this social exchange.
Instantiations	The details of a specific performance including the NLG dialog templates and character animations. Instantiations have a condition that must be true for it to be performed and a specification of how the performance changes the social world.

traditional plan operator is that the social game may be rejected by the responder, which usually has the consequence of causing an almost opposite state change. For example, character x who feels high romance towards y may initiate an "Ask Out" game towards y, with the intent of initiating the dating relationship with y. But just because x initiates "Ask Out", doesn't mean that it somehow magically forces y to start dating; y might have a status of being angry towards x, or have low romance towards x, or have a status of being jealous of x because x has been spending time (engaging in multiple positive social exchanges) with a good friend of y's. There are many factors that may contribute to y rejecting the "Ask Out" exchange, with the reject performance perhaps resulting in y having even lower romance towards x. Additionally, social exchanges can cause cascading effects via trigger rules. In this example, x might gain the status of being embarrassed, and a third party character who wasn't even part of the exchange might gain the status of pitying x. Even if a social exchange is accepted, and the desired intent occurs, there are still often multiple cascading effects of additional social state change, which is a further difference from traditional plan operators. For *Prom Week* there are 12 different intents, 6 for the three social network values and 6 for the three relationships: increase/decrease friendship network, increase/decrease romance network, increase/decrease cool network, initiate/terminate dating relationship, initiate/terminate friendship relationship, and initiate/terminate enemy relationship.

Preconditions are standard predicate conditions that rule a social exchange in or out depending on whether the precondition is satisfied in the current context. One of the goals of CiF is to minimize such hard decision making, relying more on soft decisions that can take many factors into account and that aren't liable to break in unexpected states. Therefore, by convention, preconditions are used very sparingly, only to encode the minimum conditions for a social exchange to make sense. For example, the social exchange "Text Message Breakup," which has the intent of terminating a dating relationship, only makes sense if the initiator and responder are currently dating. Many social exchanges in *Prom Week* have no preconditions (meaning they are always available for consideration).

Initiator influence rules are used to determine the volition (desire) for a character to initiate a social exchange with other characters. Responder influence rules are used to determine whether a responder accepts or rejects the social exchange. Every social exchange has two or three roles: an initiator I, a responder R, and an optional third agent referred to as the other, O. The influence rules refer to these roles in predicate arguments, with CiF appropriately filling in the roles depending on which character a volition is being computed for. Each social exchange has specific initiator and responder rules that combine with a much larger number of inherited general rules called microtheories, to determine volition and accept/reject decisions. The general form of these rules is:

```
<condition> → <increment/decrement volition for an intent>
```

We will first consider examples of initiator influence rules. For the Annoy social exchange in *Prom Week*, whose intent is to decrease the responder's friendship feelings for the initiator, two of the 18 influence rules are:

```
network(romance, I, R) > 66 && trait(I, inarticulate) → +3
[SFKBLabel(cool, R, I) window(10)] → -3
```

The first rule says that if the initiator feels a lot of romance towards the responder (>66) and the initiator has the trait of being inarticulate, add 3 to their tendency to initiate the Annoy social exchange with the responder; characters who can't express themselves well will have a tendency to express being romantically attracted to someone by bothering them. The second rule, which uses a social facts knowledge base query, says that if the responder initiated an exchange with the initiator within the last 10 social exchanges, and the exchange was labeled as cool, subtract 3 from their tendency to initiate the Annoy social exchange; a character is less likely to annoy someone who initiated a cool social exchange with them in the recent past.

While every social exchange has a small number of specific rules, a much larger number of shared rules are stored in microtheories. The purpose of microtheories is to facilitate knowledge reuse; to support writing, just once, rules that might be used across many social exchanges. The microtheory library constitutes a large repository of rules, split between dozens of microtheories. A microtheory consists of a definition and a set of influence rules. The definition of a microtheory is a condition, often times consisting solely of one predicate. For example, `relationship(friends,x,y)` is the definition of the Friends microtheory. Only microtheories whose definitions evaluate to true in the current context are considered when calculating volitions. The microtheory's rule set then provides a general representation of the social "common sense" associated with the condition, for example, in the Friends microtheory, that friends are more likely to get along, and less likely to become enemies, than strangers. So, given the example of the Annoy social exchange, if the initiator and responder happen to be friends, then the Friends microtheory definition will be satisfied, and all the rules in the Friends microtheory will be considered in addition to the Annoy-specific initiator influence rules. In this case, the Friends microtheory contains rules that will subtract from the tendency to engage in behavior with the intent to decrease the buddy network, decreasing the volition to engage in Annoy. Of course, more than one microtheory can be active during volition formation. For example, if in addition to being friends the initiator has the trait of being self-destructive, then the Self-Destructive microtheory will also be active. This microtheory captures general tendencies for a self-destructive character, which in the case of Annoy will actually add to the tendency to engage in friendship decreasing intents.

Forming volitions for a character involves summing the volitions on the right-hand sides of the satisfied initiator influence rules for every potential responder (and other) for every social exchange. At the end of this process is a vector of numbers corresponding to the character's desire to engage in each of the social exchanges with each of the characters. A variety of policies can then be used to decide what action to take, such as weighted random selection from the top N for autonomous action, or placing the top N in a menu for the player to select from as was done in *Prom Week*. The volition calculation process can be made efficient through a combination of rule-caching techniques such as Rete [Forgy 82] and by limiting the set of potential responders (for example to just nearby characters or characters present in a scene) for which volitions are calculated.

Influence rules seek to combine the advantages of utility methods and Boolean predicate representations. Utility methods have well-known advantages including converting disparate decisions into a single comparable basis (numbers), and improving robustness of decision making by avoiding unaccounted-for corners of the state space for which the

binary decision making mechanism doesn't have a match [Mark 09]. The predicate calculus, on the other hand, is great at representing complex conjunctions of states, such as "If x feels high romance towards y, and a third party z did something mean to x, followed by y doing something mean to z, then x will have a higher desire to initiate a dating relationship with y." Such complex conjunctive logic is unwieldy to express through combining algebraic functions, especially given the ease with which new conjunctive expressions can be incrementally added. By having predicate calculus expressions on the left-hand side of rules, but having them add weights to a sum on the right-hand side, the rules work together to compute a complex utility surface, combining the benefits of numeric and logic-based decision making.

Responder influence rules are similar to initiator influence rules, except that they are used to score how the responder feels about the exchange that she is included in. In a process very similar to desire formation, the responder gets to determine how they feel about the exchange. If the responder score is too low, the responder will reject the exchange, resulting in a different (and often opposite) social effect than the social exchange intent. In addition to the exchange-specific responder influence rules, microtheories are also used for computing the responder score. For *Prom Week*, the default accept/reject boundary is 0; responder scores higher than 0 result in accepting the social exchange. The responder accept/reject boundary is one of the parameters adjusted by the difficulty settings.

Finally, a social exchange has a set of instantiations, each consisting of effects and natural language generation templates. Instantiations are divided into accept and reject instantiations. Each instantiation is a different possible way a social exchange can play out. Associated with instantiations are conditions that are tested to see if the instantiation is valid in the current context. Every exchange has a generic accept and reject instantiation that places no conditions on the instantiation. More specialized instantiations have additional conditions, and play out the exchange in more specialized ways, in addition to having more effects. For example, one accept instantiation on the Break Up social exchange has the condition `trait(cold,i)`—this instantiation not only leads to terminating the dating relationship, but has additional repercussions, such as terminating the friends' relationship. If multiple instantiations have conditions which evaluate as true, the most salient is chosen, with saliency being computed as a weighted sum of the number of true predicates in each condition (the weight associated with a predicate type indicates how important that predicate is for determining the specificity of a social context).

The instantiation performance consists of lines of dialog, represented using natural language generation templates, to be spoken by the characters during the exchange. In the case of *Prom Week*, the dialog lines are tagged with animations to be performed by the characters. For example, one of the accept instantiations of Reminisce has the condition that the responder has done something recently to embarrass a third party, and the initiator is enemies with the same third party. In this instantiation, the initiator and responder bond (increasing friend feelings) by reminiscing about how the responder embarrassed this disliked third party. The template dialog for this instantiation is:

```
I: Hey%r%. Man, I can't stand%o%...
R: Tell me about it. Hey, remember that time when%SFKB_(embarrassed,r,o)%?
I: Oh god, I totally do!%pronoun(o,he/she)% totally had that coming for
being such a%pejorative%!
```

The bolded elements of the dialog demonstrate some of the natural language generation tags supported by CiF, including an SFKB reference and a character-specific pejorative. In the case of Simon and Monica talking about Oswald, the dialog could turn into:

Simon: Hey **Monica**. Man, I can't stand **Oswald**...
Monica: Tell me about it. Hey, remember that time when **I broke up with Oswald in the middle of his tennis match just to make him lose**?
Simon: Oh god, I totally do! **He** totally had that coming for being such a **n00b**!

The final element of CiF's architecture is the trigger rules. These are effects (and associated conditions) that are shared across all social exchanges. In addition to the social state change caused by the social exchange itself, additional changes may be caused by trigger rules. The trigger rules capture the cascading consequences of social exchanges, as well as state changes crossing multiple social exchanges. Consider the following example trigger rule:

```
~relationship(enemies x, y) && trait(x, cat: nice) &&
[SFKBLabel(cat: negative, z, y) window(7)] &&
~[SFKBLabel(cat: negative, x, y) window(7)] →
status(pities, x, y)
```

This rule says that if x and y aren't enemies, and x has one of the traits that falls into the more general category of "nice" traits, and a third party z has done something that falls into the more general category of negative interactions to y in the last 7 social exchanges, and x hasn't themself done something mean to y in the last 7 exchanges, then x gains the status of pitying y. Or, more succinctly, nice people will pity those who have mean things happen to them as long as they haven't done mean things themselves to the same person. This rule, like all trigger rules, can fire after any social exchange.

43.4 Interaction Approaches

Now that the architecture has been described, this section describes different ways a CiF-like architecture can be integrated into the player interaction loop. The current version of CiF treats social exchanges as discrete dialog units; thus CiF currently supports a discrete-choice model of social interaction.

Prom Week utilizes a god-game interface in which the player doesn't play a specific character, but tells characters what to do. When the player selects two characters, CiF computes the volition for the first character to initiate every social game with the second. The system then displays the top volition actions in a menu, with some intermixing to ensure variety of intentions appearing in the menu. Characters only occasionally take autonomous action. When a character develops an extremely high volition to initiate a social exchange, the player is informed that the action will take place in a small number of moves. This was added to *Prom Week* to create an additional challenge element. For example, the player may be trying to achieve the story goal of having a character date three other characters at the same time. But often, when the player has succeeded in having the character initiate a second dating relationship, the first romantic partner will develop a high volition to play a social exchange terminating the relationship. This autonomously initiated exchange puts a sharp limit on the number of turns the player has left to achieve the goal.

In *Mismanor*, an experimental RPG built using CiF [Sullivan et. al. 10; Sullivan 12], CiF was extended to include information and objects. This version of CiF, dubbed CiF-RPG, supports characters in forming volitions to perform actions related to gaining and sharing information, and gaining, losing, and using objects, in addition to network and relationship intents. Quests often have social, nonprescripted solutions enabled by the CiF model. In *Mismanor* the player plays a specific player character. The player character is modeled like any CiF character, having a collection of traits, statuses, and so forth. Traits are assigned during the character creation process. When the player interacts with an NPC, the menu options presented are determined by the highest volition social exchanges the player character wants to perform with that NPC. In a sense, CiF takes an active, dynamic role in supporting role play. NPCs are also given opportunities to engage in autonomous actions with the player and each other.

Another use of CiF would be as a simulation to create a rich social background around the player. For example, in an RPG, the characters in a village could go about their daily lives engaging each other in social exchanges driven by CiF. The social exchanges could be used to dynamically reveal backstory and frequently reference actions of the player character. In this more autonomous mode, action selection could be performed using any one of a number of well-known utility selection methods (e.g., weighted random based on the volitions, random among the top n, etc.).

Currently, we are working to extend CiF to support moment-by-moment interaction, moving away from the currently atomic social exchanges. In this interaction model, the player controls a real-time character employing individual gestures and limited natural language interaction. To accomplish this, instantiations are represented as collections of ABL behaviors [Mateas and Stern 02], with bottom-up recognition of player-initiated intents.

43.5 Related Work

Here, we briefly describe the most relevant related work. *The Sims 3* employs a social model similar to CiF. Its characters have traits and desires that inform the social practices (social norms and clusters of expectations) they perform [Evans 08]. A Sim can be involved in more than one practice at a time, and a practice can involve more than one Sim at a time. Thus, the *Sims 3* AI represents social practices as nonatomic interactions that can be intermixed, unlike the atomic interactions of CiF. However, the Sims engage in more abstract interactions, and do not use concrete dialog. Further the characters don't have backstories or make complex use of history in decision making. Finally, CiF supports reference to third parties outside of the immediate social exchange, supporting more complex social decision making and cascading social effects.

More recently, Evans has developed *EL*, a deontic logic that "distinguishes between what is in fact the case and what should be the case" [Evans 11]. For example, an *EL* rule can capture inferences such as "if it is the case that x's stomach is empty and that y is food, then it should be the case that x should eat y." While CiF relies upon weight/intent pairs on the right-hand side of a rule to determine character desires, *EL*'s rules infer what a character should be doing as well as what the character thinks other characters should be doing. *L* is a deontic epistemic logic (it can represent what is the case, what should be the case, what has been seen to be communicated, and what is intended to be communicated) [Evans 09] that has been implemented in *EL*. It's a response to Evan's critique of his own earlier work in which social practices are represented as activities that characters unproblematically know

how to participate in, versus being activities in which characters actively work to help each other maintain state in the activity (the ethnomethodological approach to human activity). The same critique applies to CiF.

Evans and Short have developed Praxis, a system with many similarities to CiF that lies behind the commercially released game Versu [Evans and Short 12; Evans and Short 13]. In Praxis, social practices are represented as collections of states that provide social affordances for character actions. Unlike CiF social exchanges, which are atomic performances, practices involve multiple state changes with opportunities for player choice. This also supports multiple practices intermixing. Character traits as well as judgements characters make about each other influence the performance of these practices.

43.6 Conclusion

The CiF architecture enables casts of characters to engage in rich social interaction, speaking the kind of concrete dialog typically associated with hand-authored dialog trees but utilizing a level of emergent social simulation typically associated with simulation games.

To accomplish this, multicharacter social exchanges are explicitly represented separately from any specific characters. Given a cast of characters, with traits and social state declaratively represented, social exchanges are retargeted for specific characters. Thus, social exchanges provide both an abstraction layer for reasoning about social exchanges and a mechanism for generalizing reusable dialog.

Characters decide what they want to do and who they want to do it with, based on soft decision making (not Boolean flags or rigid preconditions) that can take into account hundreds of considerations. This allows complex relationship states and history across the entire cast of characters to influence individual decisions.

Social interactions don't just cause a single change in social state, but have cascading consequences across multiple characters. These cascading effects help create a sense of a living world, with the characters enmeshed in a constantly changing social context.

The performance details and outcomes of previous social interactions are stored in an episodic memory and used both during social exchange performances (social exchange dialog can refer to past performances) and to help determine which social exchanges characters want to engage in. This provides a strong sense of history dependency typically missing from NPCs.

Together, these properties provide the foundation for simulating a *social* world, in which characters are embedded in a constantly evolving sea of social state.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0747522.

References

[Evans 08] R. Evans. "Re-expressing normative pragmatism in the medium of computation." *Proceedings of Collective Intentionality VI*. 2008. Available online (http://philpapers.org/rec/EVARNP).

[Evans 09] R. Evans. "The logical form of status-function declarations." *Etica and Politica/Ethics and Politics*, XI, 2009, 1, pp. 203–259. Available online (http://www2.units.it/etica/2009_1/EVANS.pdf).

[Evans 11] R. Evans. 2011. "Using exclusion logic to model social practices." In *Agents for Games and Simulations II*, pp. 163–178. Springer Lecture Notes in Computer Science, 2011. Available online (http://www.springerlink.com/index/W2675776V4411H58.pdf).

[Evans and Short 12] R. Evans and E. Short. Talk during the AI summit session "Beyond Eliza: Constructing socially engaging AI." *Game Developers Conference*, San Francisco CA. 2012.

[Evans and Short 13] R. Evans and E. Short. "Versu-A Simulationist Storytelling System." IEEF Transactions on Computational Intelligence and Artificial Intelligence in Games, forthcoming 2013.

[Forgy 82] C. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem." *Artificial Intelligence*, 19, pp. 17–37, 1982.

[Goffman 59] E. Goffman. *The Presentation of Self in Everyday Life*. Garden City, NY: Doubleday, 1959.

[Mark 09] D. Mark. *Behavior Mathematics for Game AI*. Boston, MA: Course Technology, 2009.

[Mateas and Stern 02] M. Mateas and A. Stern. "A behavior language for story-based believable agents." *IEEE Intelligent Systems*, Vol. 17, Number 4, 2002, pages 39–47. Available online (http://users.soe.ucsc.edu/~michaelm/publications/mateas-is-2002.pdf).

[Mateas and Stern 03] M. Mateas and A. Stern. "Integrating plot, character and natural language processing in the interactive drama Façade." *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*, Darmstadt, Germany. March 24–26, 2003. Available online (http://users.soe.ucsc.edu/~michaelm/publications/mateas-tidse2003.pdf).

[Mateas and Stern 07] M. Mateas and A. Stern. "Procedural authorship: A case-study of the interactive drama Façade." In *Second Person: Role-Playing and Story in Games and Playable Media*, edited by Patrick Harrigan and Noah Wardrip-Fruin, pp. 183–208, Boston, MA: MIT Press. 2007. Available online (http://users.soe.ucsc.edu/~michaelm/publications/mateas-second-person-2007.pdf).

[McCoy and Mateas 09] J. McCoy and M. Mateas. "The Computation of Self in Everyday Life: A Dramaturgical Approach for Socially Competent Agents." Intelligent Narrative Technologies II, Papers from the 2009 AAAI Spring Symposium. AAAI Technical Report, SS-09-06, AAAI Press, Menlo Park, CA, 75–82, 2009. Available online (http://users.soe.ucsc.edu/~mccoyjo/publications/AAAI-INT2-09-McCoy.pdf).

[McCoy et al. 10a] J. McCoy, M. Treanor, B. Samuel, B. Tearse, M. Mateas, and N. Wardrip-Fruin. "Comme il Faut 2 : A fully realized model for socially-oriented gameplay." In *Proceedings of Foundations of Digital Games (FDG 2010) Intelligent Narrative Technologies III Workshop (INT3)*. Monterey, California, 2010. Available online (http://games.soe.ucsc.edu/sites/default/files/CiF-FDG2010-IntelligentNarrativeTechnologies3.pdf).

[McCoy et al. 10b] J. McCoy, M. Treanor, B. Samuel, B. Tearse, M. Mateas, and N. Wardrip-Fruin. 2010. "Authoring game-based interactive narrative using social games and Comme il Faut." In *Proceedings of the 4th International Conference and Festival of the Electronic Literature Organization: Archive and Innovate*, 2010. Providence, Rhode Island, 2010. Available online (http://games.soe.ucsc.edu/sites/default/files/TheProm-ELOAI.pdf).

[Sullivan et al. 10] A. Sullivan, N. Wardrip-Fruin, and M. Mateas. "Rules of engagement: Moving beyond combat-based quests." In *Proceedings of Foundations of Digital Games, Intelligent Narrative Technologies III Workshop*, Monterey, California, June 18, 2010. Available online (http://games.soe.ucsc.edu/sites/default/files/rulesofengagementcameraready.pdf).

[Sullivan 12] A. Sullivan. "The Grail Framework: Making Stories Playable on Three Levels in CRPGs." Ph.D. Dissertation, University of California Santa Cruz, 2012. Available online (http://www.asdesigned.com/dissertation.pdf).