Social Story Worlds with Comme il Faut

Joshua McCoy, Mike Treanor, Ben Samuel, Aaron Reed, Michael Mateas, and Noah Wardrip Fruin,
UC Santa Cruz

***Abstract*—Narrative in video games has often played a subservient role, due to the complexity of hand-authoring truly meaningful player choices that can affect an ongoing story. This article presents *Comme il Faut* (*CiF*), an artificial intelligence sys0074em that matches character performances to appropriate social context, with the goal of enabling authors to write high-level rules governing expected character behavior in given social situations, rather than specific fixed choice points in a curated narrative structure. CiF characters are associated with three primary sets of characteristics: traits, relationships and statuses, which encode both the permanent and temporal qualities defining a character at a given moment. A complete social history is also stored and considered by characters in the rules governing their behavior. We use the example of *Prom Week*, a complete game utilizing CiF as its narrative engine, to illustrate how it successfully creates complex narratives that are unique for each player and directed by those players' attempts to make progress towards story goals..**

***Index Terms*—artificial intelligence, interactive drama, emergent narrative, game design,**

I. Introduction

Narrative in video games and other forms of interactive entertainment has often played a subservient role in gameplay and system design. When it does exist in a substantial form, narrative is used as either a tool to justify the setting of a game or doled out in static portions as rewards for completing other aspects of gameplay. *BioShock*'s undersea dystopia allows for distinctive art direction and packages of narrative to reward progress, but the setting and plot are largely irrelevant to the player's interaction style or strategy.

Current games with strong storytelling components do not offer many options for the player to influence the story. The "beads on a string" model of interactive narrative [1], for instance, links sequences of narratively-motivated gameplay into a linear order, collapsing and eliminating most consequences of player choice each time the next "bead" is reached. Allowing for branching structures at discrete player choice points creates an exponentially increasing authorial burden, meaning designers tend to avoid structures that allow for real choice, to the detriment of meaningful player agency within an interactive narrative. Both problems, of coupling narrative more tightly to gameplay and reducing the authorial burden from choice points, can be addressed by implementing computational models. Models of storytelling domains allows complexity to be handled procedurally, opening up a new space for expression not possible with hand-authored interactive narratives.

Manuscript received October 5, 2012. Funding for this work was in part provided by the National Science Foundation award IIS-0747522.

Comme il Faut (*CiF*) [2] is an AI system that uses these techniques to enable an interactive, authorable model of social interaction for autonomous agents. Social exchanges are the primary structure of rep-resenting social interactions in *CiF*. Social exchanges are defined as multi-character social interactions whose function is to modify the social state existing within and across the participants.

Through the use of social exchanges along with additional encoded social context, *CiF* lowers the authoring burden needed to create the social aspects of an interactive story by allowing the author to specify the rules and general patterns of how social interaction should take place. With the separation of patterns of social behavior from the norms that govern their use, authors can explicitly encode the reasoning of domains of social norms which can be reused across all social behaviors. The encoding of social norms is comprised of individual rules each of which encompass a social consideration. Because of this rules-based encoding, additional domain knowledge can be easily added to the existing base of rules and be immediately used by *CiF*. When the rules are used in conjunction with social exchanges, the character behaviors generated by *CiF* are rich and surprising.

In this paper, we contribute a detailed description of the structures with which *CiF* represents social knowledge and how this knowledge is employed to simulate social interactions between characters in a story world. Situations from a video game, *Prom Week*, provides concrete examples of how *CiF* can be used to enable social behavior in characters for interactive storytelling in a way that is tractable to author and flexible for the player. We also present an evaluation of the narratives assembled jointly by players and *CiF* for *Prom Week*. This evaluation examines both how unique players' paths through *Prom Week*'s are and how well players achieved their story goals.

II. Related Work

CiF relates to other interactive narrative technologies, deeper models of characters use in virtual worlds, *CiF*'s relation to game AI techniques, and how *CiF* in *Prom Week* compares to existing video games.

Narrative generation systems [3–6] model enough of a story world to create stories. In comparison, *CiF* does not attempt to model an entire story world. Instead it deeply models the myriad of considerations necessary for a character to follow norms during social interactions. As such, *CiF* is meant to be the social reasoning component encompassed by a narrative generation system. Similar goals have been attempted through analysis of crowd-sourced data to discover common play

interaction patterns [7] but our approach is fundamentally different, driven by a rules-driven AI system rather than pattern matching from a player-generated corpus.

There are many systems in the domain of modeling interactions between characters or virtual humans based on cognitive or psychological models that reason over competing capacities of a prescribed set of desires [8–10]. *CiF* is an implementation of an alternate, norms-based vision of modeling what characters should be doing. This approach gives characters the affordance to reason over what desires are appropriate for the situation and then to negotiate between those relevant desires [11]. Through modeling normal patterns of social behavior with a context of general social norms, the amount of story space covered by each authoring effort is increased over that of authoring for a single social state.

In comparison to hierarchical task networks [12] and behavior trees [13], the operators, or patterns of social behavior, in *CiF* make use of larger sets of domain knowledge to judge their appropriateness for the current context. Instead of encapsulating domain knowledge implicitly in hierarchically layered operators or behaviors using a small number of (possibly procedural) pre or post conditions, *CiF* chooses characters' behaviors based on all applicable rules in a large rulebase that encodes normal social behavior authored for a particular story world.

The Sims 3 is an example of a culturally influential and commercially successful video game that has a highly dynamic social space [14]. Its characters, known as Sims, have traits and desires that inform the social practices (social norms and cluster of expectations) they perform [15]. Two major differences between the systems are in the complexity of the statements of social norms and the use of history in those statements. *CiF* provides a level of complexity similar to first order logic in that parties outside of the social exchange can be referenced (*x* is cheating on *y* if *x* and *y* are dating and there is a character *z* also dating *x*) where *The Sims 3* can only reference the two characters in an interaction. *CiF* also allows for both back story (history of the story world before the player is involved) and play history to be used in reasoning and social exchange performance, a feature completely missing from *The Sims 3*. These richer rules found in *CiF* allow for each individual authoring effort to be more potent while enabling an entire new set of social reasoning to the characters.

III. Comme il Faut

A. Characters

Due to the emphasis in *CiF* on social norms and how they guide social exchanges, the representation of each character is thin. What makes characters rich and unique is their relational situation in the social world and their interconnected history. This is a direct artifact of the sociological base of *CiF*; the characters are modeled as semiotic selves. The system determines the most salient social influences for a character by considering a full context of social norms, history and current circumstance. To start with, however, characters are associated with three primary sets of characteristics: traits, relationships and statuses.

Traits

Traits are permanent properties of a character which heavily impact the possible social exchanges he or she can play (e.g. `trait(brainy,x)`). Though in reality personality traits can change over a long enough time scale, since *CiF* is generally used in short-form narratives, character traits are not able to be changed.

Relationships

Relationships are binary states that provide detailed information about the significant social connections between agents. *CiF's* relationships are bidirectional and have significant impact on social exchange play. For example, *Prom Week* has three different types of relationships, including `relationship(friends,x,y)`, `relationship(dating,x,y)`, and `relationship(enemies,x,y)`. Relationships are stored in a network that contains all of the relationships between two characters as an edge, as *CiF's* notion of relationships is public, binary, and considered bi-directional.

Statuses

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 other salient but ephemeral facts (e.g. `status(popular,x)`). They are useful in capturing transitory but potent social situations and character states; being angry, embarrassed, or cheerful can all have major effects on a character's performance, while not being permanent.

As statuses aretemporary, the status data structure includes a duration element. Unlike relationships (which are shared equally between two characters), statuses are associated with a single character. This is why the predicate notation of statuses always includes at least one character variable, like `x`. A character's status can optionally be associated with a second character. For example, a character `x` can pity or be angry at a second character, `y`: `status(pity,x,y)` or `status(angryAt,x,y)`.

B. Social State

Social Networks

Social networks are bi-directional fully connected networks where the edge values measure the feelings between characters. Examples include a romance network, which measures how interested characters are in pursuing intimate relationships with each other, and a "coolness" network which is an approximate record of how much respect characters have for one another. If `x` has a romance network value of 80 towards `y`, but `y` only has 20 towards `x`, the agents see their situation differently.

Networks being bi-directional and distinct from relationships permits interesting (and lamentably true to life) states like:

`relationship(dating, x, y)`,
`network(romance,x,y)` is 20,
`network(romance,y,x)` is 95 and
`network(romance,x,z)` is 80,

This example translate to `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`. These directional differences in social networks represent the internal feelings of characters toward each other (as opposed to the public nature of the relationship *dating*), allow for a way of encoding dramatic tension, and provide good hooks for rules. In the example above, even through `x` is dating `y`, `x`'s low feelings of romance toward `y` would make ending that relationship more likely.

Cultural Knowledge Base

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 was to create a sociologically rich representation of props. 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. Also, the relationship of characters to the cultural items is very important. The interplay between what a prop represents to a social group and how an individual relates to that prop can vary widely. This deviation is a great source for interactions based on social norms associated with a prop and is an interesting tool for seeding a storyworld with drama.

The CKB used for *Prom Week* has many items, including zombie movies, roses, 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 webcomics, Phoebie likes zombie movies. Additionally, every object in the CKB can be associated with universally agreed-upon properties in the social world (e.g. roses are romantic, dodgeball is mean). This allows for agents to interact with each other based on both collective opinion and individual relationships to the objects in the world. The CKB can be queried to search for patterns of attitudes characters hold for objects:

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

In this example, the CKB is queried to request a CKB item culturally seen as lame that character x likes and character y dislikes, which could perhaps contibute to y's volution to make fun of x. There are four parts to a CKB query, three of which can be omitted in any query: 1) the item to look for, `item` 2) the first subjective label, `(x, likes)` 3) the second subjective label, `(y, dislikes)` 4) the truth label, `lame`. Another example:

```
CKB(item,(x, likes), (y, likes),
funny)
```

This statement might be used in the left-hand-side of an influence rule for a bonding social exchange, as the agents find common ground in shared tastes.

C. Social Exchanges

Social exchanges are performances arising from particular social contexts. The purpose of a social exchange can range from wanting to display coolness and sophistication to accomplishing a social state change with a particular character, like making peace with an enemy. As with all social performances, there are contexts where the exchange is appropriate and others where a break of social expectations would result. To ensure contextually appropriate performances, social exchanges contain conditions that determine when the social exchange is appropriate in general. Some of these conditions, called preconditions, determine if the social exchange is generally applicable (characters cannot break up if they are not dating). To determine the validity of specific performances (called an instantiation) in the social exchange, there are additional conditions for each instantiation. These checks are binary and any precondition or instantiation condition that fails their evaluation cannot be performed in the current social state.

If the general domain precondition check deems the social exchange possible, *CiF* determines the character's desire to start the exchange toward another character. Social exchanges include role-specific rules that help determine desirability. These are the initiator and responder influence rule sets. These influence rule sets are used in conjunction with the authored rules for general social normalcy to determine the overall desirability of a social exchange.

Social exchanges make use of the abstraction of influence rules (described in their own section below) and an array of parameterized performances of the social exchange. Every social exchange has an initiator `i`, a responder `r`, and an optional third agent referred to as the other, `o`. These roles are designed to be extremely general so they can capture many performances across a wide range of social exchanges while being specific enough to make sense when encoding performances. The initiator influence rule set and the responder influence rule set serve distinct functions when processing social exchanges. The initiator influence rule set is used in the desire formation process to determine a character's volitions to play a social exchange with other characters. The responder influence rule set factors in how the responder feels about the exchange that she was included in. In a process very similar to desire formation, the responder gets to determine how they feel about the exchange – social exchanges were designed to keep the agency of the responder intact. Each instantiation can therefore be either a reject or accept type.

The rules, social change, and performances, when considered in tandem, provide the real encoding of the authorial intent of the social exchange – the name is simply a label that should be succinct and readily evoke the domain of the exchange. An authoring advantage of the social exchange abstraction is that additional detail can be added to the social exchange by simply adding more effect and instantiation pairs.

At the data structure level, social exchanges are comprised of an intent, a set of preconditions, influence rule sets for `i` and `r`, a set of effects, and a set of instantiations. As mentioned above, the intent encodes the change `i` wants to make on the social state. For example, the intent of the social exchange "Ask Out" is `relationship(dating,i,r)`.

As previously stated, preconditions are conditions which must hold true for the social exchange to happen. The social exchange "Breakup" has a precondition of `relationship(dating,i,r)`; before `i` can breakup with `r`, they must be dating.

Next, a social exchange has a set of effects, where an effect is made up of a pair of rules called the effect conditions and the social changes, and a label marking the effect as either 'accepted' or 'rejected.' The effect conditions dictate what must be true for this effect to take place, and the effect changes outline how the social state of the world is affected based on this particular effect playing out. At a high level, an effect represents one possible trace through a social exchange. At minimum, a social exchange should have two effects—a generic effect for the case in which the game is accepted (the sum of all of the rules factoring into `r`'s considerations was positive) and another for rejection (the sum was negative). However, through the use of effect conditions, additional considerations can be taken into account which may impact the social space in additional ways. For example, given `trait(cold,i)`, "Break Up" may not only lead to `relationship(~dating(i,r))`, but could also have more serious repercussions such as `relationship(~friends(i,r))` or `status(AngryAt(i,r))`. If multiple effects have conditions which evaluate as true, the most salient effect is chosen, with saliency being a function of which true condition rule has the most predicates.

The final component to a social exchange is a set of instantiations. An instantiation has a performance consisting of lines of dialogue, each tagged with animations that communicate state change and the justifications for the state change using hand-authored, templated natural language. As mentioned earlier, each instantiation is associated with some social change that reflect the performance and a condition that must be true for the instantiation to be performed.

D. Social History

In role performances in dramaturgical analysis as well as in semiotic view of self, an actor's history and experiences are a major factor in all aspects of performance. *CiF's* social facts database (SFDB) is a data structure that keeps track of the social history of the storyworld so that it can be queried for socially relevant information by *CiF*'s processes.

The SFDB stores a context entry for social exchanges played and every trigger rule that affects social state change (see the visualization in. Additionally, any specifically mentioned cultural item, character reference, or social exchange label (such as mean, funny, and nice to) are stored in a social exchange context entry for use in future social exchanges. Through this, agents may reason over socially complex thoughts that take into account not only their current state, but the social history that led to them possessing this state. For example, Naomi will be more interested in dating Simon if Simon has done several nice things to her recently. Relevant historical facts in the SFDB can also be explicitly referenced during performance realization, in which state change is communicated to the user through hand-written instantiations.

E. Rule System

CiF's rule system is the mechanism by which social reasoning is encoded. A rule detects a specific condition in the social space. When evaluated, the left-hand-side (condition) of the following rule can detect when two characters have a strongly romantic dating relationship:

```
relationship(dating,x,y) and
Network(romance,x,y) > 66
```

This condition is made of up distinct parts that are consistent with the previously detailed ways of representing the social state. The first, `relationship(Dating,x,y)`, is a simple check to determine if `x` and `y` are dating. The second is a look at `x`'s feeling of romance to `y` in a social network, or `Network(romance,x,y) > 66`. These parts are primitives in *CiF* and can be quickly and easily checked for truth. Rule primitives in *CiF* are known as predicates.

Rules in *CiF* are Horn clauses meaning that their predicates are conjunctive (or connected by a logical "and") and each has an implication[1]. The benefits of conjunctive rules are that they are easier to author (this removes the dependency of authors knowing logical operations by creating an "everything must be true" rule authoring environment) and they are more efficient to evaluate. This evaluation efficiency trick is used by the Prolog logical inference programming language [16] and is achieved by making rule evaluation deterministic. If other logical operators were used, such as "or" or disjunctions, rule evaluation would become non-deterministic and would result in a large and more unpredictable search space. The drawbacks of being limited to conjunctive rules is that more abstracted situations, such as a rule to capture if two characters are either dating or friends with a high level of romance, could not be written as the following condition:

```
(relationship(Dating,x,y) and
relationship(Friends,x,y)) or
network(Romance,x,y) > 66
```

However, the space denoted by the disjunctive rule can still be represented in *CiF* in the form of two separate left-hand-sides, or conditions:

```
relationship(Dating,x,y) and
network(Romance,x,y) > 66
relationship(Friends,x,y) and
network(Romance,x,y) > 66
```

Rules are evaluated by *CiF* in nearly all of its processes. As it is intended that a majority of rules will contain character variables, specific characters must be bound to rules for them to be properly evaluated. *CiF*'s processes manage variable binding internally. Every process has a different context: forming desires considers every social exchange for every combination of two or three characters, while social exchange play has a determined set of characters.

To add an additional level of utility, *CiF* allows rules to be created, valuated, and evaluated external to its processes. An

[1] Horn clauses in their definite form are disjunctions of literals with at most one positive literal (i.e. $\neg p \vee \neg q \vee r$). A logically equivalent form that is conceptually useful for influence rules in *CiF* is conjunctive and an implication ($p \wedge q \rightarrow r$). A conjunctive expression with an implication is useful in *CiF* rules as each defines a social state and, when true, implies things like a weight to a desire or that a social exchange is possible.

TABLE I
EXAMPLE INFLUENCE RULES FROM PROM WEEK

Condition (Left-Hand-S	Weight to an Intent (Right-Hand-Side)		Description
`status(CheatingOn,r,i)`	`intent(relationship(Friends,i,r))`	`-15`	If you are being cheated on, you want to be friends with them less.
`relationship(Dating,i,r)` and `status(AngryAt,i,r)`	`intent(network(Romance,i,r) +)`	`- 1`	If you are dating someone you are angry at, your desire to be romantic with them is lessened.
`status(HasACrushOn,i,o)` and `SFDB(Romantic,r,o)`	`intent(~Relationship(Friends,i,r))`	`+3`	If your crush has done something romantic to you in the past, you are less likely to be friends with them (in favor or being "more than friends")

application that employs *CiF* can create rules that can be evaluated at any time. External rules are required to provide such a binding of characters to character variables.

CiF uses rules to reason over the social world when making decisions about social exchanges. To calculate a character's will, or volition, to perform social exchanges, some rules are given a weight to aid in comparing social concerns. The rule data structure is used in or as a foundation of every data structure in *CiF* that needs to query the social world. The remainder of this section is a discussion of rules, the predicates that form these rules, and several ways in which rules are used.

Predicates

Predicates are the binding between the current social state as modeled by *CiF* and the authoring in social interaction patterns and social norms. They are representational primitives that can be evaluated for truth in a specific social state. Predicates have three areas for configuration. First is a set of up to three characters or character variables that will bind to characters during evaluation. Next is a predicate type corresponding to aspects of the social environment modeled by *CiF* consisting of character traits, relationships, statuses, social network values, history in the social facts database (SFDB), and cultural items in the world found in the cultural knowledgebase (CKB), which are described in detail in the **Error! Reference source not found.** section.

The final area for configuration is the details of exactly how the predicate is evaluated, or the evaluation mode. A predicate can be evaluated via a few methods. These different modes of evaluation are a key feature as they allow the predicate to capture more sophisticated concepts of social space. *CiF* supports four modes of predicate evaluation: true now, intent, true in history and times true.

In true now mode, the rule is immediately evaluated for truth. This is the default (and most common in *Prom Week*) evaluation mode for predicates. SFDB labels cannot be True Now by design.

The intent mode is used during desire formation and encoding the intended consequences of social exchanges. While their primary use is internal to *CiF*, external rules that contain Intent evaluation mode predicates can be used if a social exchange reference is passed to the rule's evaluation method.

Every predicate other than trait and CKB predicates can be evaluated in True in History mode. True in History evaluations determines if the predicate is in a change rule in any SFDB entry (primarily social exchange and trigger entries). SFDB type predicates default to True in History mode. SFDB type predicates and other types of predicates that use the True in History mode seem similar at first glance. In fact they are similar. The main difference lies in authoring use (SFDB labels are meant to capture an impression of a social exchange like "mean" or "nice"), evaluation efficiency (comparing a predicate to all predicates that have taken effect in the past is expensive), and specificity (there are circumstances where knowing if a character increased their cool network value toward another character by 33 less than four turns ago is very useful).

The times true mode determines how many times the predicate is true in the current social state. For example, to get the status of popular, a character needs to have three or more friends; or a character could be "wicked cool" if more than four other characters have a cool network value towards that character higher than 66. This predicate both allows for rules involving more than three characters and simplifies writing long rule conditions. For example, take a long condition:

```
relationship(Dating,x,y) and
relationship(Dating,x,z) and
relationship(Dating,x,w) and
relationship(Dating,x,u)
```

This could be rewritten as a single "times true" rule. Times true requires a couple pieces of information. The first is how many times does the social state represented by the predicate need to be true? In the above examples, this would be set to four. Second, what character variable bindings should be held static: the accepted values are "first," "second," and "both." In the above example, we could set the first character variable `x` to be static. *CiF* would then determine how many characters could be bound to `y` to make the predicate evaluate to true. The number of true bindings is then compared to the Times True number to finalize the evaluation.

The example of the "Wicked Cool" status would have "second" as the character variable binding to be held as static. With a Times True predicate of `network(Cool,x,y)` and a Times True number of four, the first character variable `x` could be bound to other members of the cast to see if the number of true bindings is four or above.

TABLE II
TEMPLATES IN *CiF*'S NLG SYSTEM

NLG Tags	Examples and Explanations
Roles	**%i% %r% %o%** The name of the character bound to the role slot.
Role Possessive	**%ip% %rp% %op%** The corresponding character name in its possessive form.
Character Locutions	**%greeting% %shocked% %positiveAdj% %pejorative% %sweetie%** Character-specific utterances.
Pronouns	**%pron(ROLE,MALEFORM/FEMALEFORM)%**
SFDB Entry	**%SFDB_(LABEL,ROLE1,ROLE2,WINDOW)%** Inserts a SFDB reference of a previously played social exchange that matches the label, roles, and occurs in a window of time.
CKB	**%CKB_((ROLE_1,SUBJECTIVE_LABEL1),(ROLE_2,SUBJECTIVE_LABEL2),(TRUTH_LABEL))%** Inserts the name of an item that matches the specified CKB query.
Conditional Statement	**%if(ruleID,text to display)elseif(ruleID,text to display) else(text to display)%** Inserts text according to rule evaluation. There can be arbitrarily many elseif clauses.
Topics of Conversation	**%toc1% %toc2% %toc3%** Either an SFDB entry lookup or CKB item that is determined when the template is first processed and is stored to be used in the rest of the performance. A topic of conversation is metadata to the template and is specified by either a CKB or SFDB NLG tag.

Combinations of these evaluation types yield interesting results. For example, if an SFDB predicate is evaluated with the Times True mode, it will return how many times that particular SFDB label was encountered by the characters assigned to the predicate's roles in the past within a history window, allowing for characters to know, for example, how many times another character has been romantic towards them in the last ten moves. Some evaluation modes can be combined. Times True and True in History can be used in the same predicate to perform detailed mining of the social history; *CiF* could find out how many times a character has been cheated on or broken up with, and some storyworlds might trigger a "freak out" behavior if many bad things happen to a character.

Armed with a detailed description of predicates and a general understanding of rules, rules external to *CiF* can be constructed. At a high level, the process is not complex; create a rule, fill the rule with predicates, then evaluate the rule (the right-hand-side of the rule is to be handled by the external process).

Influence Rules

Influence rules are *CiF* rules where the left-hand-side is a social condition and the right-hand-side consists of a weight and intent pair. *CiF*'s processes evaluate influence rules and add the weight to a character's desire toward the intent predicates when the rule's condition evaluates to true. Intents can be any predicate type that is mutable (which means the CKB and Trait predicate types are ineligible) as intents imply changing the social world in some way. Though *CiF* supports multiple intent predicates per influence rule, that capability has not been used in any significant way. Some influence rules authored for *Prom Week* are can be seen in Table 1.

Most story-focused games model a character's willingness to engage in a behavior with a simple story progression point or characteristic threshold value. To enable greater dynamism, *CiF* employs influence rule sets (IRSs) — sets of rules that influence the desires of the agents to engage in social exchanges. The right-hand-side of every rule inside of an IRS is a weight that represents how important the rule is in determining intents, where an intent is the intended change in social state after performing a social exchange (e.g. have two characters start to date). All rules, both in all initiator IRSs and in all microtheories (discussed below) are considered and their weights tallied—the social exchanges with the highest scored weights represent the social exchanges the initiatorwants to perform most. A similar scoring mechanism is used for the responder `r`, with one small caveat; `r` need only decide whether to accept or reject the proposed social exchange's intent.

Time Ordered Rules

During the development of *CiF*, we encountered authoring situations where temporal reasoning was useful, especially capturing chains of social state change in history. When a character has a second character do something mean to them, and then a third person is mean to the second, the first character should have an increased desire to be friends with the third. This "knight in shining armor" influence rule would be impossible to capture without encoding its chronology. Time ordered rules are an alternate evaluation mode to rules that allows for this type of temporal evaluation.

Time Ordered evaluation mode for rules follows an alternate evaluation path from the default True Now mode. Each predicate has a Time Order property that places the predicates into time groups (the default Time Order value is 0 which means current time). The predicates are evaluated in ascending Time Order value and are evaluated in True in History mode.

All rules with a Time Order less than 1 are evaluated without temporal ordering constraints (this is not shown in code as the predicates are evaluated in the default True Now mode). This function tolerates gaps in order, meaning a rule can have predicates of orders 0, 3, 9, 100. Gaps in Time Order values are ignored. If there are multiple predicates of the same order in the rule, they must all be true after the next lowest order and before the next highest order. Any predicate of the same order is considered true as long as all other predicates of an identical are true.

Microtheories

The power of influence rule sets is great, but if each set of rules contains repetitions of influence considerations that also apply in other situations, we have found that rule sets can become unwieldy and difficult to maintain during revisions. To address this, we have introduced the concept of

microtheories in *CiF* to capture knowledge about social dynamics that apply across multiple social exchanges. The use of microtheories is an authoring strategy that helps tame the complexity of what is essentially a big bag of rules. The microtheory library constitutes a large repository of rules, split between dozens of microtheories. A microtheory consists of a definition and a pair of influence rule sets. 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 rule set then provides a general understanding of what it means to be friends; the first set applies to `i`'s considerations, the second to `r`'*s*. For example, friends are more likely to get along, and less likely to become enemies, than strangers.

Rules in microtheories are essentially shared by all social exchanges. This abstraction permits the initiator and responder IRSs associated with specific exchanges to focus on capturing the nuances which differentiate social exchanges from one another. For example, `status( feelsSuperiorTowards, x, y)` would generally negatively impact `x`'s desire to befriend `y`, which is reflected its own microtheory. However, when taken in the context of the social exchange "Give Advice," it is reasonable that `x` would want to give advice to `y`, a social exchange that—given the right context—can lead two characters to friendship.

F. Performance Script Generation

CiF generates a performance scripts that are customized to the acting characters via a template-based natural language generation (NLG) system. This system populates designated areas of dialogue templates that are as simple as character names or as complex as conditional statements determined by arbitrary *CiF* rules.

The NLG templates are composed of literal and procedural pieces. The procedural pieces are denoted by pairs of % (percent signs) in a template while any text not wrapped in %s are output with no modification from the NLG system. For example, a small template, "Isn't %pron(o,he/she)% cool!?!" would be transformed into "Isn't she cool!?!" if the character bound to the other role was female. A complete list of template text options and their usage is in Table 2.

An example template that uses many of the tags can be found in *Prom Week*'s Declare War social exchange. Performance (or instantiation) number 18 includes all three roles (initiator, responder, and other) and makes use of the more straight-forward tags. The instantiation is paired with a condition rule and social change rules shown respectively below and is followed by the NLG template (the tags are bold):

```
Relationship(Dating,i,o) ^
   relationship(Enemies,i,r) ^
   relationship(Friends,r,o)
Network(buddy,r,o) + 10 ^
   Network(buddy,i,o) +10 ^
   SFDB(Nice,o,r) ^ SFDB(Nice,o,i)
```

Initiator: %greeting% %r%.
Responder: What do you want ***%i%****?*
Initiator: I'm sick of you hanging out with ***%o%****. If you keep this up I'm going to have to take drastic actions.*
Responder: What are you talking about?
Other: Hey guys, what are you up to?
Responder: Your ***%gender(i,boyfriend,girlfriend)%*** *is acting crazy. It's like* ***%pron(i,he/she)%*** *is trying to start WW3 or something.*
Initiator: I just can't handle it when you hang out with ***%pron(r,him/her)%*** *all the time.*
Other: Look, ***%r%*** *is just my friend, you are my* ***%gender(i,boyfriend,girlfriend)%****. You two are going to have to deal with that.*
Responder: Whatever happens I'm not being friends with ***%pron(i,him/her)%****.*
Initiator: Me neither.
Other: Chill the freak out. Both of you.
Initiator: Consider this war postponed… for now.

If the template was processed with Buzz as the initiator, Simon as the responder, and Naomi as the other, the template would be transformed into the following dialogue (the text generated for the tags is bold):

Buzz: Uhhhh Simon.
Simon: What do you want ***Buzz****?*
Buzz: I'm sick of you hanging out with ***Naomi****. If you keep this up I'm going to have to take drastic actions.*
Simon: What are you talking about?
Naomi: Hey guys, what are you up to?
Simon: Your ***boyfriend*** *is acting crazy. It's like* ***he*** *is trying to start WW3 or something.*
Buzz: I just can't handle it when you hang out with ***her*** *all the time.*
Naomi: Look, Simon is just my friend, you are my ***boyfriend****. You two are going to have to deal with that.*
Simon: Whatever happens I'm not being friends with ***him****.*
Buzz: Me neither.
Naomi: Chill the freak out. Both of you.
Buzz: Consider this war postponed… for now.

The following example, Bicker instantiation 15, has a topic of conversation *%toc1%* defined[2] by the tag *%SFDB_(Romantic,r,o)%* and shows how a topic of conversation tag is used:

Initiator: Hey ***%r%****… You've really been hanging out with a lot of new faces lately…*
Responder: So?

[2] Topics of conversation ensure that multiple references to CKB or SFDB queries will be constant throughout a performance. If a performance has a topic of conversation, the query that will replace topic of conversation tags is ran once and used throughout the performance. This is necessary as CKB or SFDB queries randomly select an entry from all entries that satisfy the query.

Initiator: I dunno, it just makes me a little uncomfortable that you've been spending more time around them, than me…

Responder: Listen, you gotta get over this 'little uncomfortable' thing. When you say stuff like this, it really makes me feel suffocated.

Initiator: I just wish you'd stop giving me reasons to worry.

Responder: What reasons to worry!? Sheesh ***%sweetie%****, just quit worrying so much.*

Responder: Although, your worries are kind of justified… Like that time when ***%toc1%****…*

Initiator: You're right ***%sweetie%****. I'm sorry. I love you.*

Responder: Yeah… Totally.

With Kate as the initiator, Monica as the responder, and Nicholas as the other, *%toc1%* would be replaced with a textual reference to a past social exchange between Monica and Nicholas. In this case, that exchange would be when "%i% kissed %r% behind the bleachers after tennis practice". In these references to the past, the roles would be filled in with the context of the past exchange with respect to the characters in the current exchange (basically matching the characters of the past to the pronouns of the present). As the past social exchange had Monica as the initiator and Nicholas as the responder, *%toc1%* would be spoken by Monica as "I kissed Nicholas behind the bleachers after tennis practice". Here is the final dialogue:

Kate: Hey ***Monica****… You've really been hanging out with a lot of new faces lately…*

Monica: So?

Kate: I dunno, it just makes me a little uncomfortable that you've been spending more time around them, than me…

Monica: Listen, you gotta get over this 'little uncomfortable' thing. When you say stuff like this, it really makes me feel suffocated.

Kate: I just wish you'd stop giving me reasons to worry.

Monica: What reasons to worry!? Sheesh ***love-monkey****, just quit worrying so much.*

Monica: Although, your worries are kind of justified… Like that time when ***I kissed Nicholas behind the bleachers after tennis practice****…*

Kate: …

IV. Evaluation

To evaluate *Prom Week* and *CiF* in light of system responsiveness and variation to player actions, play traces can be analyzed to determine how *CiF* is responding to real play situations. Even with the large amount of variation supported by *CiF* in a storyworld as content-rich as *Prom Week*, there are reasons why players could potentially be exploring a very small space of the possible story. The cast of characters in a level could have very little desire to interact with one another. Overly restrictive story goals could be constraining player choice into narrow spaces of interaction. The balance of microtheories and applicable social exchanges could leave few social exchanges for the player to choose from. Even with involving players from *Prom Week*'s alpha to its release, only a small slice of the possible game states could be seen from user testing.

To gain a better understanding of the variation in stories that players experience in the wilds of public release, a holistic and detailed understanding of the play traces is useful.

A. Play Traces from Prom Week

As players interact with *Prom Week*, the system saves their interactions with the game. These traces provide data for saving and continuing play sessions and contain the information needed to re-simulate the social state created by the player. After the player exits a play session or completes a level, *Prom Week* sends a trace to a server. The trace is associated with an anonymized ID that represents the player and is used to track a player across play sessions.

Each play trace consists of the game events chosen by the player that have an effect on the social world. Each event is stored with enough context to recreate the social world constructed by the player given *Prom Week*'s initial state. The SFDB was designed to keep a record of *CiF*'s activities and the social exchanges played, statuses timed out, and triggers fired. Additionally, *Prom Week* uses the SFDB to store when and how the player uses story points. When sent to the server, the SFDB is made into XML with included data about the level.

From when play traces were first collected in beta on December 10th, 2011 to May 17th, 2012, players have generated a total of 28,407 traces. Of these traces, 7,074 took place in tutorial levels, 504 were of the goal-less freeplay mode, and the remaining 5,425 took place in *Prom Week*'s stories. Only the 5,425 story play traces generated after the official release of *Prom Week* on February 14th 2012 are used in this evaluation.

The story play traces were generated each time a level was successfully ended (either the level clock was clicked or the player ran out of time) or a story ending was reached (a prom ending was seen). The release version of *Prom Week* had 5 playable stories: Doug, Oswald, Simon, Monica, Edward and Lil (for a small time right after release, Naomi's story was also playable).

B. Gameplay Customized Storyworld Exploration

To get a sense of how *CiF*'s simulation and *Prom Week*'s gameplay impact the actual choices presented to the player, level traces were analyzed and visualized using the Façade Log Analysis and Visualization Tool [17], [18], a visualization tool that aims to enhance the current toolset for studying interactive narratives. This tool helped in forming an understanding of how players were interacting with the released version of *Prom Week*. Even though the player has many options of social exchanges to choose from, it is not clear without evaluation that there are enough paths through the story space to satisfy the whims of each individual player. Furthermore, story goals, level casts, and the desires of the characters themselves may restrict the options available in such a way that many players will be forced down a narrow few paths in their pursuit of story goals.

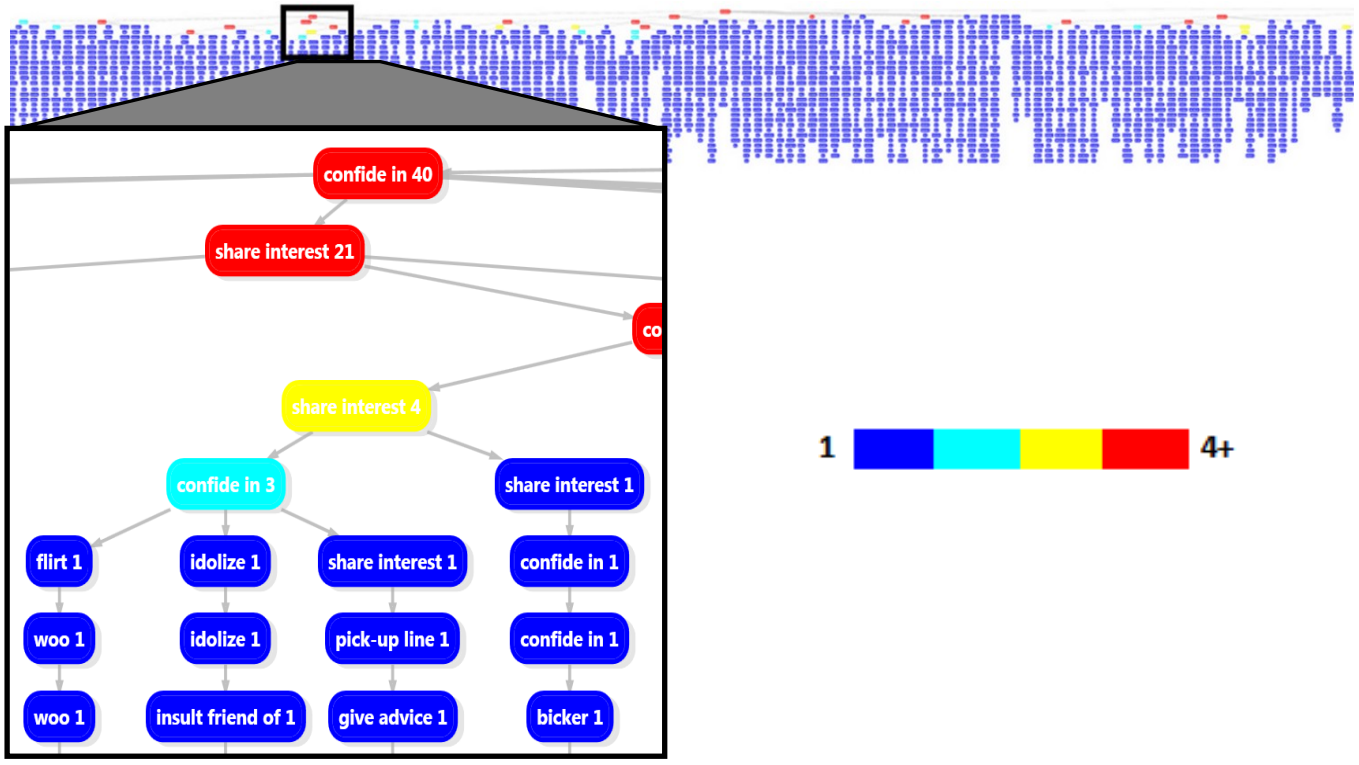

Figure 1. A play trace graph showing how often each distinct path through Simon's story was taken (shown by the color and number associated with each node). The large band of nodes seen at the top of the diagram represents approximately one third of the total size of the complete map. The cutout shows a section of the map in detail including examples of social exchanges (like "pick-up line" and "confide in") that appeared in more than one play trace. The majority of play traces are unique.

We were pleased to discover that there was a very large degree of variation in the way that players navigated the social space. Examining a tree map representing the social moves selected during the final level of Simon's campaign reveals that, of the 263 unique playthroughs we analyzed, no two were exactly alike; the space was rich enough to allow for an entirely unique play trace per player. Figure 1 is a tree graph of the play traces analyzed for Simon's campaign. Each node represents a selected social exchange, each of which results in changes to the game state (e.g. relationships starting or ending). A path through the tree is the sequence of social exchanges a player made from the starting state in the first level (the root), to an ending (a leaf). Although there are a fixed amount of maximum turns in Simon's campaign, not all paths in the tree are the same length as players have the option of skipping remaining turns and jumping ahead to the next level. The color of the nodes is a heat map indicating frequency of node visitation along that specific path; red is frequently visited (i.e. several players followed that exact same route up to the point of that node), and dark blue means visited only once (i.e. the route to that node was experienced by only a single player). For readability purposes, the nodes have been collapsed to the names of social exchanges selected, when in actuality gameplay moves are identified by the social exchange and the two characters to perform that social exchange. Including this differentiator would have further increased the branching of the tree, but we claim that it is already branchy enough for the purposes of validating our hypothesis of high variability.

The average indegree (times a node was encountered by a player) of a node in this graph is approximately 1.11; though as mentioned above there are a few nodes towards the beginning that are selected many times—"share interest" and "confide in" are popular starting moves, happening 91 and 40 times respectively—the vast majority of story traces have nodes that are visited precisely once. This means the play trace is unique because no other trace is composed of the same sequence of social exchanges.

Performing n-gram analysis[3] revealed some interesting statistics on the patterns of sequences of social moves played (this analysis is explore in more detail in the next section). Using 1-gram analysis, there are 38 unique social moves that players employed on this level, out of a total possible 39 social moves that exist in the game. Using 3-gram analysis, we have 2521 unique patterns, of which only 80 appear more than 10 times. With 6-gram analysis, there are 5066 unique patterns of social exchanges, one of which occurred 16 times, another 10 times, and all the rest less than 5 times. The fact that so many separate patterns exist, with so little repetition, indicates that players were able to find their own way through the story space. Moreover, the n-grams that have the most repetition are situations in which the same social exchange was played multiple times in a row. Though apparently there is a player type that relies on a strategy of brute force (for example, attempting to 'woo' six times in a row), they are dwarfed by the number of other patterns exhibited.

[3] N-gram analysis is used to find repeated patterns of varying lengths in corpora.

Another interesting point was discovered by examining the tree graph of social exchanges. The sheer breadth of the tree gives a positive view of just how much variability there is in player choice; not only does the system allow for variability, but players are taking advantage of it as well. Additionally, though there are only 11 nodes that players chose for the first move, there are 79 different nodes selected for the second, and 143 for the third. By the fourth turn, nearly every gameplay trace is unique. Even traces with subtle differences in gameplay actions (for example, the sequence of social actions "reminisce", "confide in", "ask out" as opposed to "confide in", "reminisce", ask out") can result in remarkably different traversals through the social state, as *Prom Week* keeps track of the specific social exchanges and instantiations that the user has seen and incorporates them into future social exchange selection. Moreover the specific ordering of social changes also impacts the formulation of which social exchanges characters want to play with each other, thus even seemingly similar play traces can be considered unique.

The general trend of paths becoming unique can be seen across the stories and is even more prevalent in the more difficult stories of the late game. Take Oswald's story as an example, which has 390 level traces that all begin in the same starting state. Twenty-five different opening moves were selected with an average indegree of 15.6. After the second turn the average drops to 2.36. The average dips to 1.27 after the third turn, and hits 1.07 after the fourth.

The above supports our first hypothesis of the variability in *Prom Week*. The low average indegree indicates that we are approaching a completely unique playthrough experience for each player; the large number of unique n-grams even for small n indicate that these unique playthroughs consist of different patterns of play; and the rapid branching factor means that the little overlap that does exist between players quickly separates into distinct traces. Given all of this, we claim that *Prom Week* was successful in providing a game space with large amounts of variability, even if, as we see below, players selected between only a handful of the total possible options on the first turn.

The relatively low variability seen during the first turn is actually positive evidence for our second hypothesis: that *Prom Week* is specifically providing large variability in the service of making stories playable. There are five characters in Simon's first level, and each character wants to engage in five possible social exchanges with each other character (the top five social exchanges character A wants to perform with B given the desires computed by *CiF* for character A). Since the player picks a unique initiator and responder, this means that there are at least 100 potential opening social exchanges (the actual number is a little higher, as players can spend story points to unlock additional options).

The fact that, of these hundred starting options, only eleven were ever pursued between all of the gameplay traces implies that players are not choosing moves at random, but attempting to accomplish specific story goals. The beginning of each level provides framing text which contextualizes the characters' relationships to each other with respect to campaign goals, and offers small hints about how to accomplish the goals. The hints take the form of advising the player on which characters to form relationships with, but offer no advice on which specific social exchanges to try. This means that player actions are being motivated by story goals without being dictated by them, providing a solid foundation for our second hypothesis.

C. Strategy Driven Play

To determine if *Prom Week* promotes strategic play, this section analyzes the player-driven paths through *Prom Week* with respect to the successful completion of story goals. To be seen as an indicator for strategic play, large portion of the story paths - variable though they may be - need to lead to successful goals. Story goals in *Prom Week* represent story states for the player to make true in the storyworld. For example, in Simon's campaign, the player is tasked with accomplishing five distinct goals, including having Simon make five friends, having Simon begin dating someone, and giving Simon an "ideal rival" by making him friends and enemies with the same person. The combination of goals accomplished determines which ending for the campaign the player receives. Though endings are mostly pre-written to leverage authorial control, there still exists template dialogue within endings that allows for explicit references to specific social exchanges that were chosen by the player throughout the course of gameplay. This gives every choice the player makes—and not just goal completion—an impact on the campaign's climax.

D. Story Goal Completion

Figure 2 shows another view of the 263 traces which start at Simon's first level and progress their way through the end of his campaign. In this graph the color of the nodes shows the impact of that social exchange on story goals. Story goal completion ranges from dark blue to green, progress toward the goals is in the range of light blue to orange, and moving the social state away from the story goal (antiprogress) is colored orange or red. This data was generated by taking the same level traces used to generate Figure 2and running them through *CiF*, keeping track of the goal accomplishments at each game turn.

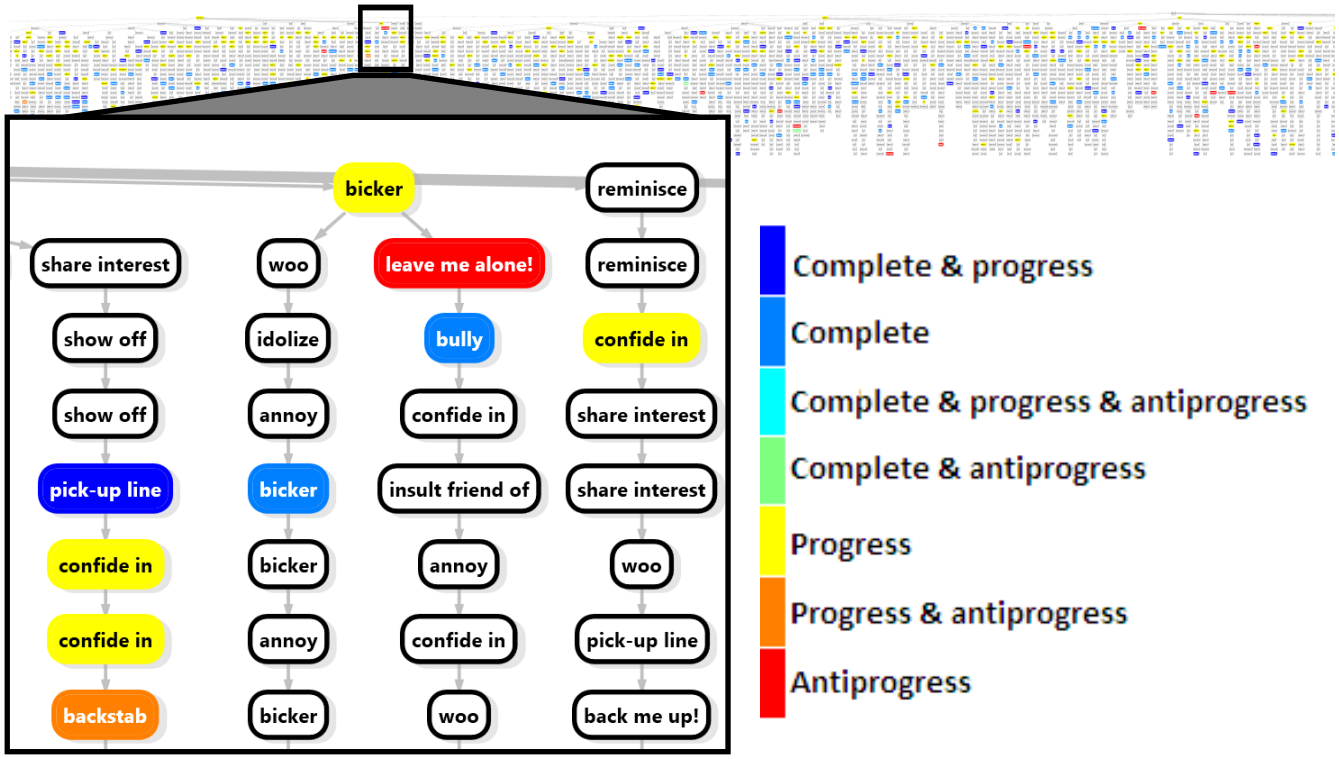

Figure 2. A tree displaying the amount of progress towards goals in Simon's campaign. The color of the nodes represents the type of goal progress. There are three types of goal progress that can be combined in any way. Complete (Blue) means a goal was completed, progress (yellow) means that one aspect of a goal was made true, and antiprogress (red) means that an aspect of a goal that used to be true was made false. White nodes mean that no progress (or antiprogress) was directly made by making that social exchange, though the social state was still changed which could lead to progress in future turns. The large band of nodes along the top still represents about 1/3 of the total play traces of Simon's story.

Simon's campaign is the third non-tutorial level in *Prom Week* and is of intermediate difficulty. Though some goals can be accomplished in just a single turn (across all 263 traces for Simon's campaign, only 13 completed a goal on the first turn, and only 17 completed a goal on the second), the rest take several turns to complete. As seen inFigure 2, the story goals were completed by players at many points along the story paths. Of all of Simon's traces, only a single one did not contain any goal progress. All others exhibited at least some amount of effort towards achieving story goals.

Even though Simon's campaign is of intermediate difficulty, players still displayed an aptitude for achieving goals. Between all of the play traces, goal completion (on any of Simon's five goals) was reached a total of 610 times (average of 2.32 goals per player). If every trace from every file had accomplished all five goals, the total would be 1,315, which means that around 46% of all possible Simon goals were achieved. Goal progress was made a total of 837 times (average of 3.18 times per player), and goal antiprogress was made a total of 44 times (average of 0.18 times per player).

A concern when designing goals is that *Prom Week*'s gameplay—manipulating social relationships within a setting of cascading social influences in the pursuit of story goals—is fairly unique. Since *Prom Week* serves as an introduction to this genre of social puzzle game for most players, figuring out the nuances of the system to make story progress could have proven to be a challenge. Although the goal completion rate is perhaps a little low for a campaign of only intermediate difficulty, the results are encouraging because not only were players motivated to pursue story goals, they were also able to create a strong enough internal model of the storytelling system to be able to pursue story goals with some amount of success.

REFERENCES

[1] P. Harrigan and N. Wardrip-Fruin, Eds., Second Person: Role-Playing and Story in Games and Playable Media. The MIT Press, 2007, p. 432.
[2] 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), 2010.
[3] M. Lebowitz, "Creating characters in a story-telling universe," Poetics, vol. 13, no. 3, pp. 171–194, Jun. 1984.
[4] J. Meehan, The metanovel : writing stories by computer. [New Haven]: Yale University Department of Computer Science, 1976.
[5] S. Turner, The Creative Process: A Computer Model of Storytelling and Creativity. Psychology Press, 1994.
[6] B. Tearse, M. Mateas, and N. Wardrip-Fruin, "MINSTREL Remixed," in Proceedings of the Intelligent Narrative Technologies III Workshop on - INT3 '10, 2010, pp. 1–7.
[7] J. Orkin and D. Roy, "The restaurant game: Learning social behavior and language from thousands of players online," Journal of Game Development, vol. 3, no. December, pp. 39–60, 2007.
[8] R. S. Aylett, S. Louchart, J. Dias, A. Paiva, and M. Vala, "Fearnot!: an experiment in emergent narrative," in Proceedings of Intelligent Virtual Agents (IVA05), 2005, p. 305.
[9] S. Marsella and J. Gratch, "EMA: A process model of appraisal dynamics," Cognitive Systems Research, vol. 10, no. 1, pp. 70–90, 2009.
[10] M. Si, S. Marsella, and D. V. Pynadath, "Modeling appraisal in theory of mind reasoning," Autonomous Agents and Multi-Agent Systems, vol. 20, no. 1, pp. 14–31, May 2009.

[11] R. Evans, “The Logical Form of Status-Function Declarations,” Etica & Politica, vol. 11, no. 1, pp. 203–259, 2009.
[12] K. Erol, J. Hendler, and D. S. Nau, “Semantics for Hierarchical Task-Network Planning.” 1995.
[13] D. Isla, “Handling Complexity in the Halo 2 AI,” Game Developers Conference, p. 12, 2005.
[14] The Sims Studio, “The Sims 3.” Electronic Arts, 2009.
[15] R. Evans, “Re-Expressing Normative Pragmatism in the Medium of Computation,” Proceedings of Collective Intentionality VI, 2008.
[16] A. Colmerauer and P. Roussel, “The birth of Prolog,” in Computer, 1996, vol. 28, no. 3, pp. 37–52.
[17] S. Sali and M. Mateas, “Using Information Visualization to Understand Interactive Narrative: A Case Study on Façade,” in Proceedings of the Fourth International Conference on Interactive Digital Storytelling, 2011, vol. 7069.
[18] S. Sali, “Playing With Words: From Intuition To Evaluation Of Game Dialogue Interfaces,” UC Santa Cruz, 2012.