A Lightweight Videogame Dialogue Manager

James Ryan, Michael Mateas, and Noah Wardrip-Fruin
Expressive Intelligence Studio
University of California, Santa Cruz
{jor, michaelm, nwf}@soe.ucsc.edu

ABSTRACT

We present a fully procedural alternative to branching dialogue that is influenced by theories from linguistic pragmatics and technical work in the field of dialogue systems. Specifically, this is a dialogue manager that extends the *Talk of the Town* framework, in which non-player characters (NPCs) develop and propagate subjective knowledge of the gameworld. While previously knowledge exchange in this framework could only be expressed symbolically, such exchanges may now be rendered as naturalistic conversations between characters. The larger conversation engine currently lacks a player interface, so in this paper we demonstrate our dialogue manager through conversations between NPCs. From an evaluation task, we find that our system produces conversations that flow far more naturally than randomly assembled ones. As a design objective, we have endeavored to make this dialogue manager lightweight and agnostic to its particular application in *Talk of the Town*; it is our hope that interested readers will consider porting its straightforward design to their own game engines.

Keywords

dialogue management, conversation, non-player characters, natural language generation

INTRODUCTION

Nearly thirty years after its influential early use in games like *Space Rogue* (1989) and *The Secret of Monkey Island* (1990), branching dialogue still prevails in game design today as the dominant approach to implementing character conversations (Freed, 2014). The fundamental appeal of this approach is that it supports extended dialogue interaction while still providing guarantees about content coherence. This is because branching dialogue is authored in directed graphs that explicitly define all valid conversations as the possible paths through those graphs. Due to this explicitness, incoherent conversations can only occur during gameplay as a result of authoring errors, which are easier to correct than complex runtime procedures are to debug. These guarantees about content coherence, however, come at the cost of authoring ease, player agency, and dynamism to system state.

Branching dialogue is hard to author. Like all directed graphs, conversation graphs with even modest branching factors may blow up after just a handful of levels in the graph. As such, authors of such graphs often force branches to converge onto one another as a way of tempering this unwieldy growth, but this tactic tends to make the specification less readable (and thus more difficult to work with). Even with dedicated software at hand, these notions work to make the authoring of branching dialogue both burdensome and cumbersome (Bateman and Adams, 2007; Freed, 2014).

Branching dialogue also inhibits player agency. At any decision point, the range of dialogue options is typically explicitly presented to the player, but players do not appreciate decision

Proceedings of 1st International Joint Conference of DiGRA and FDG

points with many explicit choices (Szilas and Ilea, 2014)—because of this (and because graphs become less manageable as the branching factor grows), the effective choice space at player decision points is typically limited, and this of course damages player agency. Moreover, the authoring tactic of branch convergence renders some player choices meaningless and, worse, even signals this meaninglessness to the player (Wardrip-Fruin, 2009).

Finally, the rigid linking of content in these graphs makes it difficult to author conversations that are dynamic to the underlying system state. Authors may place flags on graph nodes to specify that certain dialogue options only be available when some variable aspect of system state holds (Wardrip-Fruin, 2009), but because an author must explicitly link these nodes to other nodes, she effectively must compose whole paths through the graph that are appropriate with regard to that aspect of system state. But games can get into very many states, many of which may not be anticipated at authoring time, and it is not feasible to author entire conversations for each, or even many. As such, games tend to feature conversations that are appropriate across most variations of system state, but this yields dialogue interactions that are rarely dynamic to (potentially interesting) changes in the state.

In this paper, we present a fully procedural alternative to branching dialogue that is relatively easy to author for, is designed to foster player agency, and may yield dialogue interactions that are highly dynamic to the system state. Just as *story management* works to disentangle the inextricable linking of nodes in a branching story graph by employing a policy for dynamically selecting story events (Mateas, 2005), our method disentangles the graph structure of branching dialogue by employing a policy for dynamically selecting non-player character (NPC) dialogue. Rather than specifying unwieldy directed graphs, by our approach an author composes individual lines of dialogue and annotates them for the concerns central to this selection policy. Because the policy may select an NPC response given *any* player dialogue selection, the player's choice space is effectively the entire pool of dialogic content. This advances player agency significantly, but it also presents the considerable challenge of designing a player interface that facilitates selection from such a large choice space. As discussed below, we are currently investigating potential interface solutions, and so in this paper we demonstrate our dialogue manager by feeding it onto itself, *i.e.*, by producing conversations between NPCs (who each select dialogue using the same policy). Finally, our method yields dialogue exchanges that are highly dynamic to the underlying system state. Instead of composing whole conversations pertaining to individual state variations, by our approach authors may quickly craft many variants of individual lines of dialogue that each pertain to unique aspects of underlying state. Because content is not inextricably linked at authoring time, our system is always free to select (at NPC conversation turns) the particular variants that best express the underlying game state.

As we noted above, the fundamental appeal of branching dialogue is that it provides guarantees about conversational coherence. The potential drawback of our method, then, is that our policy for selecting NPC dialogue could produce incoherent conversational sequences. To test this, we elicited human judgements as to the naturalness of conversations produced by our system. As we report below, procedural conversations yielded by our system were rated as flowing more naturally than randomly assembled conversations.

The dialogue manager that we present here is an extension to an AI framework that we have

presented elsewhere, called *Talk of the Town* (Ryan et al., 2015). In this framework, NPCs form and propagate subjective knowledge about other characters and places in the gameworld. The dialogue manager that we present here integrates with this framework to render the exchange of such knowledge as naturalistic conversations between characters. More precisely, certain lines of dialogue, when delivered to another character, assert propositions about the gameworld that the receiving character may adopt as her own subjective belief (which she may then propagate to other characters in the same way). Though we have implemented our dialogue manager as an extension to this framework, we attempt to describe its operation in this paper such that interested readers may port it to their own applications. Finally, while we *are* employing natural language generation (NLG) to produce NPC dialogue in *Talk of the Town*, due to space considerations we limit our concerns in this paper to the dialogue manager that handles conversation flow and sends targeted requests to the NLG system (which we plan to present in a separate publication).

RELATED WORK

Our approach is rooted intellectually in prominent theories in the field of *linguistic pragmatics*, namely *speech act theory* (which posits that utterances have pragmatic force; Austin, 1975), Grice's *cooperative principle* (conversations are cooperative exchanges with implicit rules; Grice, 1970), and Brown and Levinson's *politeness theory* (speakers work to affirm their interlocutors; Brown and Levinson, 1987). Architecturally, we adapt to the videogame domain technical approaches from the field of *dialogue systems*; as we note below, we have been particularly influenced by the work of David Traum and collaborators (Traum and Allen, 1994; Traum and Larsson, 2003). Additionally, we view our current approach as a generalized refinement of earlier work in which we procedurally recombined *Prom Week* dialogue exchanges using a notion of speech acts (Ryan et al., 2014), and as an instantiation of the *modular content* strategy that we have articulated elsewhere (Ryan et al., 2015).

A handful of systems in the videogame domain have supported procedural conversations. Many parser-based interactive-fiction (IF) games have featured open-ended dialogue with NPCs, but typically without any modeling of the conversation beyond valid player queries and associated responses. Notable in this area, however, is the *Threaded Conversation* extension to the IF platform *Inform 7*, which supports stateful conversations with authorable parameters for flow management (Reed, 2010). *Versu*, also by Emily Short (in collaboration with Richard Evans), takes this approach much further by modeling speech acts, which may select who should speak next, what speech act that character should perform, and what topic she should address (Short and Evans, 2013). *Façade* also employs a notion of speech acts, in its case by mapping arbitrary player dialogue inputs to discrete acts that inform the operation of its reactive-planning ecosystem, components of which select NPC dialogue (Mateas and Stern, 2004). *Bot Colony* likewise maps free-text player input to speech acts (Joseph, 2012). Very recently, the *LabLabLab* project has produced multiple games centered on freeform conversation with NPCs, but without mapping to speech acts (Lessard, 2016). While conversations in *Versu*, *Façade*, and *Bot Colony* take place in real time, in our game, as in the *LabLabLab* games, conversation proceeds by discrete turns. However, like the majority of these systems, we rely crucially on a notion of speech acts, as we discuss below. Of all the systems we have listed, it seems that *Versu* is most similar to ours. Characters in both systems may pursue conversational goals, and both systems utilize a dialogue pool that is

available to all characters (but with contextual usage constraints asserted by authored preconditions). Our system renders the exchange of character knowledge in natural language and, similarly, characters in Versu form subjective beliefs that pertain to story concerns, and the delivery of dialogue by one character may cause another to change her beliefs. The major distinction of our contribution relative to the other work that we have outlined here is that we have endeavored to produce a lightweight, streamlined design for a videogame dialogue manager that does not make commitments to a specification environment, agent-modeling approach, or game engine. Whereas, for instance, dialogue management in *Versu* and *Façade* is deeply embedded in the Praxis (Evans and Short, 2014) and ABL (Mateas and Stern, 2004) language environments, respectively, we endeavor here to support readers who may seek to port our design to their own game applications with relative ease.

BACKGROUND: *TALK OF THE TOWN*

This work extends the *Talk of the Town* framework, which supports non-player characters who build up knowledge of the gameworld that may then be propagated or may deteriorate in a number of interesting ways (Ryan et al., 2015). In this section, we provide a brief overview of this framework, focusing on its simulation of character knowledge, which we have now integrated with the dialogue manager presented in this paper.

Each *Talk of the Town* gameworld is a procedurally generated American small town in which NPCs build up knowledge about businesses, homes, and their fellow residents through first-hand observation, as well as by hearing things from other people. Knowledge formation and propagation (as well as deterioration) happens online during the simulation, in which characters act out daily routines across day and night timesteps. A character's composite knowledge of the world is structured as an ontology of interlinked *mental models* (Ryan et al., 2015). Each mental model pertains to a particular entity (character, business, or home) and is composed of a number of *belief facets* corresponding to particular attributes of that entity (*e.g.*, name, hair color, address). Crucially, characters adopt new belief facets (and potentially revise existing ones) in response to *evidence*, and the *strength* of a given belief facet is the sum total of the strength of all the pieces of evidence that support it. While we have implemented nine types of evidence that work to characterize a wide array of knowledge phenomena (Ryan et al., 2015), our purposes here concern only the four evidence types that relate to character knowledge propagation:

- **Statement.** A *statement* occurs when a character says something about the world to another character that she herself believes.
- **Declaration.** When a character delivers a statement, her *own* belief that she is imparting will grow stronger by virtue of her having uttered it; we call such buttressing a *declaration*.
- **Lie.** A *lie* occurs when a character says something about the world to another character that she herself does *not* believe.
- **Eavesdropping.** An *eavesdropping* occurs when a nearby character overhears a statement or lie for which they are not the intended recipient.

Upon encountering a new piece of evidence, a character will either adopt a new belief facet, augment the strength of a currently held belief facets that it supports, or, if it contradicts her current belief facet, enact a belief-revision procedure (Ryan et al., 2015). In this revision

procedure, the strength of the new evidence is weighed against the strength of the character's currently held belief facet that the new evidence contradicts. If the new evidence is stronger, the character will revise her belief; if it is weaker, she will not revise her belief, but will remember the evidence in the case that future evidence also supports it.

While previously we described our implementation of character knowledge propagation as a trading of purely symbolic representations, in this paper we present an extension to this framework in which knowledge is transmitted over the course of naturalistic character conversations that are rendered in surface-level natural language. Below, we describe how our dialogue manager integrates with the simulation of knowledge phenomena that we have just outlined to instantiate *statement*, *declaration*, *lie*, and *eavesdropping* pieces of evidence as knowledge is transmitted by natural-language utterances.

A LIGHTWEIGHT DIALOGUE MANAGER

In this section, we describe the dialogue manager that we have developed as an extension to the *Talk of the Town* framework. As noted above, we have targeted a straightforward, application-agnostic design so that others may port the system to their own applications.

Preliminary Definitions

For our purposes, a *conversation* is a dialogue exchange initiated by one character toward another character that plays out over a series of *turns*, during each of which one character delivers to the other character one line of dialogue. As this definition expresses, our system currently only supports dyadic conversations, and the only notion of time at play here is the progression of a conversation by discrete turns (*i.e.*, we are not modeling real-time conversation). Briefly, we will introduce terms for different types of *conversational roles* that we will be using throughout the rest of the paper: a *conversant* is either character in a conversation; an *initiator* is the character who initiates a conversation, in which case the other character is the *recipient*; on a given conversation turn, the *speaker* is the character who delivers a line of dialogue, and the other character is the *interlocutor*.

Core Notions

Before proceeding to discuss the operation of our dialogue manager in detail, in this subsection we first introduce a set of core notions that are central to this operation.

Dialogue Moves

The central notion underpinning our system is the *dialogue move*, a term we take from dialogue systems research (Traum and Larsson, 2003) and appreciate for its ludic flavor. Dialogue moves in our system are like speech acts, except messier and more numerous. Rather than reifying only high-level discursive acts like `ask` or `tell` (or even relatively more specific ones, like `yes-no question` or `affirmative answer`), in our authored set of dialogue moves we feature, in addition to these staples, very low-level moves like `ask about the weather`, `tell where you work`, `say something positive`, and `confirm or correct first name`. By our approach, a set of dialogue moves is always extensible, with new moves being added whenever an author deems it necessary or convenient. As such, move sets will vary considerably by application and may become very large.

A line of dialogue may be delivered by a character to *perform* a dialogue move, and a given line may perform multiple (potentially very many) dialogue moves. For illustration, we offer a selection of disconnected example lines of dialogue that we have authored, along with the dialogue moves that they perform in our system:

"Oh, greetings, Andrew." `greet`, `greet back`, `use interlocutor first name`

"I'm alright. Yourself?" `answer`, `answer how are you`, `answer how are you neutrally`, `ask`, `ask how are you`, `make small talk`

"Yes, the weather is wonderful." `agree`, `agree about the weather`, `agree that the weather is good`, `say something positive`, `say something positive about the weather`, `make small talk`, `talk about the weather`

Conversational Obligations

The delivery of a given line of dialogue may *obligate* a conversational party to perform a dialogue move on a subsequent conversation turn. As an example, in our system the line *What do you think about this weather?* obligates the interlocutor to perform the dialogue move `assert about the weather` on a subsequent turn. Speaks may even obligate themselves to perform some move, *e.g.*, a line "There's something I need to tell you...". Here, we again borrow from David Traum's work, particularly the notion of *discourse obligations* proffered by Traum and Allen (1994).

Conversational Goals

Characters may pursue *conversational goals* for which dialogue moves serve as the planning operators. A plan for a conversational goal consists of a series of steps, where each step is either a dialogue move (to be performed by a certain conversational party) or a subgoal (with a plan of its own that must be carried out); if the last step of a plan is realized, then the goal is satisfied (regardless of whether any or all earlier steps have been realized). If the next step in a character's plan relies on the other character performing some move, the plan is *on hold*. Currently, plans are not constructed dynamically, but instead are preauthored and stored in a plan library. (As we discuss below, we intend in future work to have characters dynamically construct conversational plans.) Below we provide examples of three goals (one of which is a subgoal) and their associated plans. The step prefix `[me]` indicates that the goal pursuer must perform the move specified by the step, while `[them]` requires the other conversational party to perform the move, and `[either]` means the step will be realized if either character performs the specified move.

- `GOAL:LEARN NAME OF STRANGER IN PUBLIC`
  - `GOAL:INTRODUCE SELF TO STRANGER IN PUBLIC`
    - `[either] make small talk (x4)`
    - `[me] introduce self`
  - `[me] request name`
  - `[them] introduce self`
- `GOAL:END CONVERSATION`
  - `[me] wrap up conversation`

`[them] say goodbye`

`[me] say goodbye back`

Note that, with regard to the goal `LEARN NAME OF STRANGER IN PUBLIC`, a character could simply request another character's name immediately at the beginning of a conversation, but this would be brash and awkward. In the plan we have specified, the character pursuing this goal will engage in small talk for several turns (and introduce herself) before requesting the other character's name. This encodes normative social behavior, such that characters pursuing goals may act believably and mannerly, even as such commitments delay the satisfaction of their goals.

Above, we noted that conversational obligations prescribe that their obligated parties perform specified dialogue moves, but one might wonder why they do not instead prescribe that certain conversational goals be pursued. The answer is that, from the perspective of authorial control, we have found that it is powerful to use obligations as a governor of local coherence and conversational goals as a governor of global (conversation-level) coherence.

Conversational Frames

The dialogue manager may activate initial obligations and goals from any of a set of *conversational frames* that we have hand-authored; here, we use the term 'frame' in a broad Minskian sense (Minsky, 1977). Whether the dialogue manager will activate a given frame depends on the social context of the conversation. More technically, a frame will be activated if its preconditions are met, given the state of the world at the start of the conversation. Let us illustrate by again providing implemented examples:

- `FRAME:FACE-TO-FACE`

  Preconditions: `conversation modality is face-to-face`

  Initiator obligations: `greet`

- `FRAME:PHONE CALL`

  Preconditions: `conversation modality is phone call`

  Initiator obligations: `report identity`

  Recipient obligations: `answer phone`

  Recipient goals: `GOAL:LEARN CALLER IDENTITY`

- `FRAME:STRANGERS IN PUBLIC`

  Preconditions: `conversation is in public`, `initiator and recipient are strangers`

  Initiator goals: `GOAL:LEARN NAME OF STRANGER IN PUBLIC`

  Recipient goals: `GOAL:LEARN NAME OF STRANGER IN PUBLIC`

- `FRAME:EXTREMELY INTROVERTED RECIPIENT`

  Preconditions: `recipient is extremely introverted`

  Recipient goals: `GOAL:END CONVERSATION`

Note that because these frames can be stacked upon one another (*i.e.*, each and every frame whose preconditions are met will be activated in a conversation), we do not need to reduplicate specifications across them. For this reason, only the frame `FRAME:FACE-TO-FACE` asserts an obligation on the initiator to greet the recipient, but of course this frame will be activated whenever, *e.g.*, the frame `FRAME:STRANGERS IN PUBLIC` is activated, since conversations between strangers in public will always be face-to-face. Further, it is simple to specify frame variations that differ according to considerations of, *e.g.*, character personality—one simply asserts frame preconditions sensitive to these considerations and then authors the frames to prescribe socially believable obligations and goals, given the, *e.g.*, personality considerations. This elegant stacking of frames allows an author to define capsules of social context without having to worry about cumbersome notions like inheritance. Because frame preconditions can specify anything about the world state, an author can also specify frames that pertain to, for instance, past interactions or non-conversational goals that characters may have:

- `FRAME:ALREADY INTERACTED THIS TIMESTEP`

  Preconditions: `conversants already interacted this timestep`

  Initiator obligations: `explain reason for another conversation`

- `FRAME:INITIATOR SEEKS PROMOTION`

  Preconditions: `initiator has a goal to be promoted at work, recipient is initiator's boss`

  Initiator goals: `GOAL:DISCUSS PROSPECT OF PROMOTION`

The reader might wonder why the frame `FRAME:FACE-TO-FACE` does not assert an obligation on the *recipient* to perform a `greet` dialogue move. The reason for this is that upon the initiator performing her obligated `greet` move (which the frame *does* prescribe), the recipient will likely become obligated by that initiator line of dialogue to greet back in turn. In this way, frame authoring here is not about scripting entire dialogue sequences, but rather specifying an initial set of goals and obligations that may spark dialogue sequences that are believable in their particular social contexts.

Conversational Violations

Lines of dialogue may have associated *conditional conversational violations*, which are specified by rules of the form `violation name` ← `conditions`. If a line of dialogue is deployed and the conditions for one of its violation rules hold, then that line will incur a *violation* in addition to performing its associated dialogue moves. Such violations are recognized by the other conversational party, who will have an opportunity (or an onus, depending on the severity of the violation) to *handle* it, for instance by ignoring, acknowledging, highlighting, belaboring it, etc. How a character handles a violation depends on the particular line that incurred the violation, the character's personality, and aspects of the social state. On a technical level, we manage the handling of a given violation by having the interlocutor seek to perform a dialogue move constituting such handling.

Topics of Conversation

As an ancillary notion, we reify *topics of conversation*, which a line of dialogue may be used to *address* during a *lull* in a conversation. By 'lull', we mean conversation turns in

which no character is obligated to perform a dialogue move and no character is actively pursuing a conversational goal. In such a case, a character might reference an existing topic of conversation (*e.g.*, work or the weather) to fill in the lull.

Propositions

We reify the semantic content of a line of dialogue as a set of *propositions* about the world. (Here, we do not mean an expression of propositional logic, but rather the linguistic notion of a 'proposition' as a meaning of an utterance.) As an example, the line *My name is Jill* would have an associated proposition `first name(speaker, 'Jill')`. Note that we only care about representing propositional content that corresponds to the belief facets that make up character mental models in *Talk of the Town*, which means that we do not seek to exhaustively reify all the conceivable propositions of a line. Below, we give a technical account of how exactly propositions are used to transmit character knowledge.

Effects

Finally, a line of dialogue may have associated *conditional effects* that yield changes to the general world state if the specified conditions hold; these are defined by rules of the form `effect` ← `conditions`. In our implementation, `effect` will be a method call that may be executed to carry out the effect, *e.g.*, `interlocutor.lower_affinity(speaker)`.

Turn-Taking

Loosely following Traum and Allen (1994), conversation turns are allocated by the system according to a simple policy: if a character has an unresolved obligation, allocate the turn to him or her; otherwise, if a character has a conversational goal whose plan is not on hold, allocate it to him or her. If there is a tie (*e.g.*, both characters have an unobstructed plan and no obligations) or there is a lull in the conversation, then the turn is allocated probabilistically with consideration to the conversants' respective extroversion personality components. Speakers will spend their turns accordingly (*i.e.*, will seek to resolve obligations or pursue goals), and speakers without obligations or goals will either address a topic of conversation, perform the dialogue move `make small talk`, or adopt a goal to end the conversation.

Requesting Dialogic Content

So far, we have skirted issues of dialogue authoring and content selection; this is because we wish to tease apart these concerns from a general description of the design of our dialogue manager (in support of readers who may wish to port it to their own applications, which will likely have different authoring and content constraints). Maintaining this stance, we will in this section first provide an account of the basic constraints on dialogic content that this system imposes, before briefly discussing the technical details of content selection and deployment in our specific implementation of it.

While the dialogue manager decides *what* a character will say, an external *content source* will decide *how* the character will say what they say. Specifically, once a conversation turn has been allocated, the dialogue manager makes a request to its content source to ask for a line of dialogue that either performs a targeted dialogue move (if the speaker will attempt to resolve an obligation or pursue a goal) or addresses a targeted topic of conversation. Here, the content source may be either a base of preauthored dialogue or an NLG module. In

either case, the content source must be able to reason over the following information about all preauthored/generable lines of dialogue:

- **Preconditions.** A list of rules specifying what must be true (or not true) about the gameworld in order for this line to be deployed.
- **Dialogue moves.** A list of the dialogue moves a line may be used to perform.
- **Obligations pushed**. A list of the dialogue moves this line obligates its interlocutor to perform next.
- **Speaker obligations pushed.** A list of the dialogue moves this line obligates its *speaker* to perform next.
- **Conditional violations.** A list of rules specifying the conditional conversational violations for this line.
- **Topics introduced.** A list of the topics of conversation introduced by this line.
- **Topics addressed.** A list of the topics of conversation addressed by this line.
- **Propositions.** A list of the propositions that will be asserted upon deploying this line.
- **Lie conditions.** A list of conditions that, if satisfied, mean that the speaker does not herself believe the propositional content of a line of dialogue, which is how we define lying in *Talk of the Town* (as explained above).
- **Conditional effects.** A list of rules specifying the conditional effects for this line.

Upon receiving a dialogue request, the content source must furnish a line (through either content selection or generation) that satisfies the request (*i.e.*, performs the targeted move or addresses the targeted topic) and has its preconditions met. To evaluate preconditions, the content source must have access to the current world state.

The Expressive Power of Preconditions

Because preconditions may specify anything about the world state, an author may use them cleverly to prescribe that the content source furnish the *most expressive* variant of the lines that perform the targeted dialogue move (or address the targeted topic of conversation), given the character who is speaking and the context of the conversation and the gameworld at the time of the request being made. Let us consider an example pertaining to the common authoring strategy of having characters say the same thing differently according to varying personality models; this strategy is at play in, *e.g.*, *Prom Week* and *Versu* (McCoy et al., 2013; Evans and Short, 2014). Specifically, we will consider three ways to perform the dialogue move `say you should get back to work` that vary by character personality:

- **Amiable:** "Apologies, but I should probably get back to work before my boss catches a look this way!"
- **Neutral:** "Well, I better get back to work now."
- **Cold:** "You're distracting me from my duties."

By cleverly specifying the following preconditions, an author may prescribe that the content source always pick the variant that best expresses the speaker's personality:

- **Amiable:** `speaker has personality trait 'amiable'`

- **Neutral:** `speaker does not have personality trait 'amiable'`, `speaker does not have personality trait 'cold'`
- **Cold:** `speaker has personality trait 'cold'`

While dialogue preconditions are more intuitively a way of specifying when *not* to use a given line, they may actually be used to flexibly prescribe *which* line, from a pool of candidates, is most expressive. As we have just shown, this is done by specifying disjoint precondition sets such that the only satisficing line for a given speaker will (in the case of this example) be the one that best expresses her personality. (More powerfully yet, authors might employ a notion of precondition *specificity*, where lines with the most specific satisfied preconditions are selected.) Beyond issues of character personality, preconditions can be used in this same way to prescribe that a content source furnish dialogue that best expresses any aspect of the conversation or the gameworld that holds at the time that the dialogue request is made.

Our NLG Module

In authoring content for the conversation system we have implemented for *Talk of the Town*, we utilize a tool that we have developed called Expressionist. With this tool, an author defines a probabilistic context-free grammar whose terminal derivations are lines of dialogue that are explicitly annotated for the specific concerns governing content selection in the target application (Ryan et al., 2015); in our case, these concerns are the types of information about lines of dialogue that we enumerated above. Specifically, we have implemented an NLG module that operates over a grammar authored in Expressionist to generate bespoke lines of dialogue according to requests made by the dialogue manager. Due to space considerations, further discussion of this NLG module is beyond the scope of this paper, though we plan to present it in a future publication.

Updates

Once a line of dialogue has been furnished by a content source, the system deploys it, and then the dialogue manager uses the information about it outlined above to execute updates to the conversation state, the world state, and the subjective beliefs of the conversants; in this section, we will discuss these notions in turn.

Updates to the Conversation State

To update the conversation state, the system first processes the dialogue moves performed by the deployed line, which may resolve conversational obligations or cause plan steps in conversational goals to be realized, which in turn may cause those goals to be satisfied. Additionally, certain dialogue moves may terminate the conversation (*e.g.*, `storm off` or `say goodbye back`), cause new conversational obligations to be asserted, new conversational violations to be incurred, or new topics of conversation to be introduced or addressed.

Updates to the World State

Updates to the general world state are made according to any conditional effects of a deployed line that hold at the time of its delivery. In our implementation, we include arbitrary function calls in the definition of effect rules, which are then executed to yield such effects.

Knowledge Transmission

Upon deployment, a line of dialogue transmits the knowledge asserted by its propositions, which causes updates to the conversants' beliefs to be executed accordingly; this is the aspect of the conversation engine that hooks into the deep modeling of character knowledge, outlined above, that underpins *Talk of the Town* (Ryan et al., 2015). Specifically, each time a proposition is asserted by a line of dialogue, four types of evidence (defined above) may be instantiated: a *statement*, *declaration*, *lie*, or *eavesdropping*. Whether a proposition causes a *statement* or *lie* piece of evidence to be instantiated depends on the line of dialogue's lie conditions. Whether a line of dialogue is eavesdropped by a nearby character is determined probabilistically at the start of a conversation turn. As new evidence is instantiated, the recipients of the evidence consider whether to adopt new beliefs (or refine existing ones) in the way described above (and more deeply by Ryan et al., 2015). Characters who adopt new beliefs may then themselves assert propositions about the world that express these new beliefs, thus propagating them. This may even occur over the course of a conversation, for instance, in the case of a character learning her interlocutor's name and then using that very name in a later conversation turn (as seen in the example conversation below).

Symmetric Design

We note that our system design is symmetric with regard to its treatment of player and non-player conversants, which is an unusual approach in dialogue systems (where task-based, service-oriented paradigms reign) and in videogame dialogue engines (though *Versu* is a strong counterexample). By our design, players and NPCs will all be held to the same conversational standards (*e.g.*, with regard to obligations or violations) and will all select from the same content source. (Our system does not currently support player interaction, but below we outline our vision for a player interface.) We like this approach for multiple reasons. First, because it does not rely on a human to be in the loop, the system may be used to produce procedural conversations among NPCs (an example of which we provide in the next section); this could bolster *background believability*, and it could also be used to render story beats that exclusively involve NPCs. Relatedly, because it does not require human inputs, it is quite easy to test the system (by feeding it back on itself) and to refine and verify content specifications. Lastly, by putting players and NPCs on the same conversational playing field, the system will interestingly be able to recognize when players defy their obligations or incur conversational violations; we believe this will afford a rich space of NPC behaviors pertaining to the checking of specific disruptive player behaviors that in many systems would not be recognizable.

EXAMPLE CONVERSATION

For illustrative purposes, in this section we will work through an example procedural conversation; for each of its lines, we provide a trace of how the line affects both the conversational state and the gameworld. The lines that appear here were generated from a space of roughly 3M generable lines. This particular conversation occurs on the night of August 14, 1979, at the bar of the Legnon Avenue Hotel, in a town called Piperton (pop. 322). It is an exchange between strangers, initiated by Marion Orgel, a 64-year-old woman, toward recipient Kevin Leppert, a 27-year-old man. For this conversation, the dialogue manager activated the frames `FACE-TO-FACE` and `STRANGERS IN PUBLIC`, both defined above.

Marion Orgel: Hello.

— Turn allocated according to `obligation('greet')`
— Performed `move('greet')`
— Pushed `obligation('greet back')`

Kevin Leppert: Hi.

— Turn allocated according to `obligation('greet back')`
— Performed `move('greet back')`

Marion Orgel: Yours is a face I don't recognize. Are you from Piperton?

— Turn allocated according to `plan step(either, 'make small talk')`
— Performed `move('make small talk')`, `move('ask are you from here')`
— Pushed `obligation('answer are you from here')`

Kevin Leppert: Yup, born and raised.

— Turn allocated according to `obligation('answer are you from here')`
— Performed `move('make small talk')`, `move('answer are you from here')`

Marion Orgel: What do you do?

— Turn allocated according to `plan step(either, 'make small talk')`
— Performed `move('make small talk')`, `move('ask where do you work')`
— Introduced `topic('work')`
— Pushed `obligation('answer where do you work')`

Kevin Leppert: I'm a Stocker at 6th Street Grocers.

— Turn allocated according to `obligation('answer where do you work')`
— Performed `move('make small talk')`, `move('answer where do you work')`
— Addressed `topic('work')`
— Asserted propositions `workplace(Kevin Leppert, 6th Street Grocers)`, `job title(Kevin Leppert, 'stocker')`
— Pushed `obligation('acknowledge response')`

Marion Orgel: Oh, okay.

— Turn allocated according to `obligation('acknowledge response')`
— Performed `move('acknowledge response')`

Marion Orgel: I'm Marion, by the way.

— Turn allocated according to `plan step(Marion Orgel, 'introduce self')`
— Performed `move('introduce self')`
— Satisfied `goal(Kevin Leppert, 'LEARN NAME OF STRANGER IN PUBLIC')`
— Asserted proposition `first name(Marion Orgel, 'Marion')`
— Pushed `obligation('introduce self back')`

Kevin Leppert: My name is Kevin.

— Turn allocated according to `obligation('introduce self back')`
— Performed `move('introduce self back')`
— Satisfied `goal(Marion Orgel, 'LEARN NAME OF STRANGER IN PUBLIC')`
— Asserted proposition `first name(Kevin Leppert, 'Kevin')`
— Pushed `obligation('say nice to meet you')`

Marion Orgel: Nice to meet you.

— Turn allocated according to `obligation('say nice to meet you')`
— Performed `move('say nice to meet you')`
— Pushed `obligation('say nice to meet you back')`

Kevin Leppert: And you as well.

— Turn allocated according to `obligation('say nice to meet you back')`

— Performed `move('say nice to meet you back')`

Kevin Leppert: I have to set off. Maybe I'll see you around here again.

— Turn allocated probabilistically (due to lull in the conversation)

— Performed `move('wrap up conversation')`

— Pushed `obligation('say goodbye')`

Marion Orgel: Goodbye, Kevin.

— Turn allocated according to `obligation('say goodbye')`

— Performed `move('say goodbye')`, `move('use interlocutor first name')`

— Asserted proposition `first name(Kevin Leppert, 'Kevin')`

— Pushed `obligation('say goodbye back')`

Kevin Leppert: Bye.

— Turn allocated according to `obligation('say goodbye back')`

— Performed `move('say goodbye back')`

— Terminated conversation

EVALUATION

In evaluating a system like ours, there is potential to unintentionally end up evaluating the quality of the authored content that it displays as its output, rather than the operation of the system itself. With this in mind, here we seek to evaluate not the *quality* of the conversations that our system produces (which would really be an appraisal of authoring performance), but rather the *naturalness* of those conversations with regard to how they flow from line to line (since this is what the dialogue manager controls). Toward this, we conducted a preliminary experiment comparing randomly assembled conversations to ones produced by our system (both using the same content base of lines generated by our NLG system). By using random assembly as a baseline, we home in on the effect that our dialogue manager's *organizational policies* (*e.g.*, turn-taking, conversational goals and obligations) have on conversational naturalness. In this section, we discuss our experimental design and results.

Experiment

Seven respondents who are not associated with the project were administered a survey in which we asked about the naturalness of sixty unrelated conversations. Specifically, for each conversation, a single question was asked—*How natural is the flow of this conversation?*—and participants indicated their judgement using an integer scale with values between 1 ("less natural") and 5 ("more natural"). The sixty conversations were randomly sorted, with each belonging to one of two types: thirty *procedural* conversations generated using all of the dialogue manager's organizational policies that we outlined above, and thirty *random* conversations whose generation is explained below. Both types of conversations featured randomly selected conversants in the same simulated town on the same simulation timestep. To randomly generate conversations, a number of turns between five and fifteen (a typical range for our procedural conversations) was randomly determined, and each turn was then randomly allocated to one of the two conversants. During each turn, a dialogue move was randomly selected and a request was made to our NLG system to target that move. Lines whose preconditions were not met (given the world state at a particular conversation turn) were not generated, as this could produce inherently nonsensical lines, as well as lines that

contradict a character's personality model (which could produce a type of incoherence that is unrelated to the kind that we sought to evaluate). As such, the random conversations may more precisely be thought of as conversations that are yielded when only our dialogue manager's organizational policies are ablated.

Results

The human respondents rated the flow of procedural conversations as far more natural than that of random conversations. Specifically, procedural conversations earned a mean naturalness rating of 4.05 ($\sigma = 0.97$), versus 1.34 ($\sigma = 0.82$) for random conversations; this difference was statistically significant ($t(418) = 30.96$; $p < 0.0001$). Though this is not an evaluation of all aspects of our system (*e.g.*, content selection, authorial expressivity, etc.), the fact that procedural conversations earned ratings that far exceeded those of the random baseline (and, further, were fairly high in absolute terms) indicates that the system's organizational policies do well to engender natural conversational flow.

CONCLUSION AND FUTURE WORK

We have presented a fully procedural alternative to branching dialogue in the form of an implemented dialogue manager that extends the *Talk of the Town* framework, in which nonplayer characters (NPCs) develop and propagate subjective knowledge of the gameworld. While previously the trading of character knowledge could only be expressed symbolically, such exchanges may now be rendered in surface-level natural language. From a human-evaluation task, we found that our system produces NPC conversations that flow far more naturally than randomly assembled ones. It is our hope that interested readers will port our system design to their own game engines.

Toward Dynamic Conversational Planning

While in our implementation characters currently rely on a preauthored plan library to pursue their conversational goals, we plan to explore dynamic conversational planning in future work. One approach to this would be to more richly specify plan components in a *hierarchical task network* (Erol, 1996). More dynamically still, characters could even chain backward from goal dialogue moves by reasoning over conversational obligations to infer backward-chaining rules. That is, characters would infer rules that chain from desired dialogue moves backward to lines asserting obligations that would elicit performance of those desired moves by the appropriate conversants.

Toward a Player Interface

We are currently tackling the tricky issue of providing a player interface to the larger conversation system that we are developing. (Lacking this, we have even been exploring the unusual design space of using a human actor as an interface to *Talk of the Town* gameworlds; Ryan et al., 2016.) Our goal for such an interface is to allow the player to select from lines of dialogue (rendered in surface-level natural language) that come from the same content pool that NPCs select from. Because such lines would be explicitly annotated for all the concerns that our dialogue manager reasons about, the system would deeply understand the player's dialogue choices (via the symbolic mark-up), even though she may make her selections by considering nuances of natural language (which the system cannot understand). The problem, however, is that we must build an interface by which a player may select from

millions of dialogue choices. From preliminary exploration, we anticipate that our solution will combine: system suggestions that rely on the current conversational context, a hierarchical menu by which the player may navigate dialogue moves and select example lines that perform those moves, and a free-text interface that uses a machine-learning model to map unconstrained player inputs to lines in the content pool.

BIBLIOGRAPHY

Austin, J. L. (1975). *How to do things with words*. Oxford University Press.

Bateman, C. M. and E. Adams (2007). *Game writing: Narrative skills for videogames*. Charles River Media.

Brown, P. and S. C. Levinson (1987). *Politeness: Some universals in language usage*. Cambridge University Press.

Erol, K. (1996). Hierarchical task network planning.

Evans, R. and E. Short (2014). Versu—a simulationist storytelling system. *TCIAIG*.

Freed, A. (2014). Branching conversation systems and the working writer. *Gamasutra*.

Grice, H. P. (1970). *Logic and conversation*.

Joseph, E. (2012). Bot colony—a video game featuring intelligent language-based interaction with the characters. In *Proc. GAMNLP*.

Lessard, J. (2016). Designing natural-language game conversations. In *Joint Conference of DiGRA-FDG*.

Mateas, M. (2005). Beyond story graphs: Story management in game worlds. In *Story Generators: Approaches for the Generation of Literary Artefacts*.

Mateas, M. and A. Stern (2004). Natural language understanding in Façade: Surface-text processing. In *Proc. TIDSE*.

McCoy, J., M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin (2013). Prom week: Designing past the game/story dilemma. In *Proc. FDG*.

Minsky, M. (1977). Frame-system theory. *Thinking: Readings in Cognitive Science*.

Reed, A. (2010). *Creating Interactive Fiction with Inform 7*. Cengage Learning.

Ryan, J. O., C. Barackman, N. Kontje, T. Owen-Milner, M. A. Walker, M. Mateas, and N. Wardrip-Fruin (2014). Combinatorial dialogue authoring. In *Interactive Storytelling*.

Ryan, J. O., A. M. Fisher, T. Owen-Milner, M. Mateas, and N. Wardrip-Fruin (2015). Toward natural language generation by humans. In *Proc. Intelligent Narrative Technologies*.

Ryan, J. O., M. Mateas, and N. Wardrip-Fruin (2015). Open design challenges for interactive emergent narrative. In *Interactive Storytelling*.

Ryan, J. O., A. Summerville, M. Mateas, and N. Wardrip-Fruin (2015). Toward characters who observe, tell, misremember, and lie. In *Proc. Experimental AI in Games*.

Ryan, J. O., A. J. Summerville, and B. Samuel (2016). Bad News: A game of death and communication. In *Proc. CHI Conference on Human Factors in Computing Systems*.

Short, E. and R. Evans (2013). Versu: Conversation implementation. https://emshort.wordpress.com/2013/02/26/versu-conversation-implementation/.

Szilas, N. and I. Ilea (2014). Objective metrics for interactive narrative. In *Proc. ICIDS*.

Traum, D. R. and J. F. Allen (1994). Discourse obligations in dialogue processing. In *Proc. Association for Computational Linguistics*.

Traum, D. R. and S. Larsson (2003). The information state approach to dialogue management. In *Current and New Directions in Discourse and Dialogue*.

Wardrip-Fruin, N. (2009). *Expressive Processing*. MIT Press.