Architectural Issues for Compositional Dialog in Games

Ian Horswill
Northwestern University
ian@northwestern.edu

Abstract

Making a game that supports generative conversation with NPCs involves a number of technical and design challenges, ranging from designing new game mechanics to making a Prolog interpreter run under Unity. I discuss the software architecture of a game, scheduled for initial release in spring 2015, that attempts to solve some of these problems.

Introduction

There has been surprisingly little NLP in the character AI of shipped games. While parser-based interactive fiction systems (Short, 2011), such as Adventure (Crowther & Woods, 1976), provide a user-interface based on typed imperative sentences, they rely entirely on procedural attachment (i.e. they produce no internal logical form) and typically have little, if any character AI. *Façade* had very sophisticated character AI, but it's natural language system had no compositional semantics (Mateas & Stern, 2004). More recent conversation-oriented games such as *Versu* (Evans & Short, 2014) and *Prom Week* (McCoy, *et al*., 2011) have abandoned typed input entirely, in favor of menus of fixed dialog options. This is not to say there aren't examples of sophisticated NLP in games; many researchers have developed research prototypes (Reed et al., 2011) or used games as test beds for NL research (Endrass, *et al.*, 2014; Koller, *et al*., 2004). But I have been unable to find instances of shipped games with gameplay that involves compositional NLP.

There are a number of reasons for this. Current game genres do not provide good opportunities for this kind of interaction [ANONYMIZED], and so new game mechanics and genres must be developed to support them. And complex AI systems do not generally fit well into the run-time environments of conventional game engines where 1 millisecond per frame is considered a lot of CPU.

In this paper, I discuss the issues in integrating simple compositional dialog into a tile-based RPG game, scheduled for initial release in spring 2015. I will talk briefly about designing game mechanics for this kind of interaction, and then discuss designing character AI and NLP that fit cleanly into a contemporary game engine.

Game Design

The game (working title, *MKULTRA*) is a mystery set in an alternate history where the CIA's mind control experiments of the 50s, 60s, and 70s, have borne fruit. Its technical goal is to bring composition NL dialog and full reactive planning to simulationist games such as *The Sims* and *Versu*. Gameplay involves two primary mechanics, information gathering through dialog with NPCs (particularly question answering), and mind control, where players solve problems by injecting false beliefs into the knowledge bases of NPCs to manipulate their behavior. Space precludes a more detailed discussion of the design issues with the game; see [ANONYMIZED] for further discussion.

One of the primary design challenges is to make a fluid user interface for typed NL input. Parser-based games are prone to "hunt the verb" gameplay where the player flails trying to find an input the system will understand. *MKULTRA* uses a bidirectional grammar, allowing it to display possible valid completions of a player's input as they type. This both reduces typing time and unobtrusively exposes the player to examples of the system's grammar and lexicon.

Architecture

The system is built on Unity3D. AI code uses a custom Prolog interpreter designed to interoperate well with Unity.

Framework architecture

Each character runs as a separate AI system with a separate knowledgebase; all inherit from a shared, global KB. Each KB contain both a standard Prolog database, and a separate eremic logic (Evans, 2010) database used as blackboard/working memory, and for communication with Unity components.

The control architecture can be thought of as a least-common-denominator reactive planning system (Bonasso *et al*., 1997; Mateas & Stern, 2002). Characters are structured as a set of "concerns" that can record local state, process event messages, propose and score actions, and create and destroy subconcerns.

Events

Characters are primarily event-driven. Low level C# code sends event messages to the AI system, which dispatches them to the relevant concerns. After all events have been processed for a given tick, the system selects an action. Actions are considered a kind of event, so once an action is executed, it is reported back to the character and other characters in the area as an event.

Events are represented as event descriptions (Prolog terms). While not very expressive – we can't, for example, distinguish two separate events that happen to have the same description – it's fast and sufficient for our needs.

Construals

Events in the real world don't have unique descriptions. "I'd like a gin and tonic" is always an assertion; when addressed to a bartender, it's also a request for a drink, but not when uttered at an Alcoholic's Anonymous meeting.

Multiple description is modeled using a two-place relation, $\mathrm{construe}(A, B)$, which states that any event with description A also has description B. When handling events, characters compute the set of all construals of the event and process each construal. This allows a more modular implementation of the kinds of detailed reasoning about social norms and their violations seen in Versu. For example, the Prolog rules:

```
construe(request(Agent, Patient, _),
         uppity_act(Agent)) :-
   subordinate(Agent, Patient).
construe(uppity_act(Agent),
         norm_violation(Agent)).
```

State that requests by subordinates to their superiors are uppity (a gross oversimplification, over course), and that uppity acts are norm violations.

Action selection and problem solving

Selection of actions to deliver to the game engine is performed using a variant of the utility-based methods found in recent AI games like *Prom Week* and *Versu.* Concerns are first polled to propose actions. Then, for each proposed action, its construals are computed, and each concern is polled to score each construal. The system then executes the action with the best overall score:

$$\arg\max_{a\in\mathrm{actions}} \sum_{\substack{d\in\mathrm{construals}(a)\\ c\in\mathrm{concerns}}} \mathrm{score}(c, d)$$

To extend this scheme to handle subgoaling and planning, we use an on-line problem solver based on Sibun's Salix (1992), which was in turn based on McDermott's NASL (McDermott, 1978). Given a task T to perform, it proposes it for immediate execution if it is an action. Otherwise, it determines all possible strategies (decompositions) for *T*. If there is only one, it executes it. If there are multiple strategies, it recursively searches for a metastrategy to resolve the conflict. Custom metastrategies can be specified for a task domain, such as Salix's strategies for discourse planning. In the absence of a custom metastrategy, it chooses a generic strategy, such as utility-based scoring, the use of a preference relation, or random selection.

Natural language processing

The natural language system handles single-clause utterances, optionally wrapped in one or more modal verbs. Thus LFs have the form:

$$Q(\Box(\neg(A_1(x_1) \wedge \ldots \wedge A_n(x_n) \wedge P(y_1, \ldots, y_m))))$$

Where Q is a sequence of zero or more quantifiers, $\Box$ is zero or more modal operators, the negation is optional, the A_i are predicates for intersective adjectives, and P is the predicate for the interior clause. Although the grammar supports quantifiers in the style of Montague's (1973) PTQ semantics, I don't currently have a use case for quantified NPs in actual game dialog. Anaphora resolution is not yet implemented, but the restriction to single-clause sentences should allow the use of relatively simple anaphora resolution.

Low-level parsing and generation

The parser-generator began as a very heavily modified version of the definite clause grammar of Pereira and Shieber (1987), extended to be efficiently bidirectional, and to support mood, polarity, tense, aspect, person, number, and gender features, as well as a number of grammatical constructions such as pronouns and PPs that were not previously supported.

DCGs offer a number of advantages for games: they're relatively easy to make bidirectional; they're very easy to implement; and they can also be salted with raw Prolog code to execute during the parsing process. In addition, quips (i.e. human authored dialog, Short, 2011) are easily added as specialized, character-specific productions. They generalize the slotted string mechanisms used in recent games and IF systems (Evans & Short, 2014; Montfort, 2007; Nelson, 2006).

Discourse planning

Larger scale generation is based on Sibun's Salix (1992) discourse planner. The primary appeal of Salix is its incrementality, since this allows planning time to be spread out over many update cycles of the game engine, while still allowing the character to being speaking as soon as the first increment has been chosen. However, actually making Salix run in the polled architecture of a game engine required a major rewrite using explicit continuations so as to support interruptibility and durative actions.

Conversation and interaction rituals

Ritualized interactions between characters are usually implemented as state machines, which can be painful to code and debug. We use a generalization of DCGs one might call "event logic grammars." For example, the rules:

```
conversation >--> opening, content, closing.
opening >--> [ greet(I,R), greet(R,I) ].
closing >--> { partner(P) }, [exit_ss(P)].
closing >--> [parting(X, Y), parting(Y, X)].
```

state that a conversation begins with an opening, which consists of a greeting from the initiator I to the recipient R, followed by a reciprocal greeting from R to I. It ends with a closing, which consists either of an exchange of partings initiated by either conversational partner, or by one's partner exiting one's social space (i.e. walking away).

ELGs reduce coordination to a parsing problem. Let $L(G)$ be the language generated by the event grammar. Then after a sequence of events s, the possible next events are simply the possible events e for which $s + e$ is a prefix of some string in $L(G)$, i.e. $E = \{e \mid \exists s'.(s + e + s') \in L(G)\}$. E, which is easily computed using a variant of DCG parsing, is the set of relevant events to listen for. Moreover, the set of possible actions the character can perform at this point is simply the subset of E for which the character is the agent.

References

Bonasso, P., Firby, R. J., Gat, E., & Kortenkamp, D. (1997). Experiences with an Architecture for Intelligent Reactive Agents. *Journal of Theoretical and Experimental Artificial Intelligence*, *9*(2-3).

Crowther, W., & Woods, D. (1976). Colossal Cave Adventure.

Endrass, B., Klimmt, C., Mehlmann, G., Andre, E., & Roth, C. (2014). Designing User-Character Dialog in Interactive Narratives: An Exploratory Experiment. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(2), 166–173.

Evans, R. (2010). Introducing Exclusion Logic as a Deontic Logic. In *Deontic Logic in Computer Science, Proceedings of the 10th International Conference, DEON 2010, Lecture Notes in Computer Science Volume 6181* (pp. 179–195). Fiesole, Italy: Springer.

Evans, R., & Short, E. (2014). Versu - A Simulationist Storytelling System. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(2), 113–130.

Koller, A., Debusmann, R., Gabsdill, M., & Striegnitz, K. (2004). Put my galakmid coin into the dispenser and kick it: Computational Linguistics and Theorem Proving in a Computer Game. *Journal of Logic, Language and Information*, *13*(2), 187–206.

Mateas, M., & Stern, A. (2002). A Behavior Language for Story-Based Agents. *IEEE Intelligent Systems*, *17*(4), 39–47.

Mateas, M., & Stern, A. (2004). Natural Language Understanding in Façade: Surface-text Processing. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*. Darmstadt, Germany.

McCoy, J., Treanor, M., Samuel, B., Wardrip-Fruin, N., & Mateas, M. (2011). Comme il Faut: A System for Authoring Playable Social Models. In V. Bulitko & M. O. Riedl (Eds.), *Proceedings of the 7th AI and Interactive Digital Entertainment*. Stanford, CA: AAAI Press.

McDermott, D. (1978). Planning and acting. *Cognitive Science*, *2*(2), 71–100.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In P. Suppes, J. Moravcsik, & J. Hintikka (Eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics* (pp. 221–242). Dordrect, NL.

Montfort, N. (2007). *Generating Narrative Variation in Interactive Fiction*. University of Pennsylvania.

Nelson, G. (2006). Inform 7.

Pereira, F. C. N., & Shieber, S. (1987). *Prolog and Natural Language Analysis*. Brookline, MA: Microtome Publishing.

Reed, A. A., Samuel, B., Sullivan, A., Grant, R., Grow, A., Lazaro, J., … Wardrip-Fruin, N. (2011). A Step Towards the Future of Role-Playing Games: The SpyFeet Mobile RPG Project. In *Proceedings of the Seventh Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-11)*. Stanford, CA.

Short, E. (2011). NPC Dialog Systems. (K. Jackson-Mead & J. R. Wheeler, Eds.)*IF Theory Reader*. Boston, MA: > Transcript On Press.

Sibun, P. (1992). *Locally Organized Text Generation*. University of Massachusetts, Amherst.