An interactive, generative Punch and Judy show using institutions, ASP and emotional agents

Matt Thompson [1] and **Julian Padget** and **Steve Battle**

Abstract. Using Punch and Judy as a story domain, we describe an interactive puppet show, where the flow and content of the story can be influenced by the actions of the audience. As the puppet show is acted out, the audience reacts to events by cheering or booing the characters. This changes the emotional state of each agent, potentially causing them to change their actions, altering the course of the narrative. An institutional model is used to ensure that the narrative is constrained to remain consistent with the Punch and Judy canon.

1 Introduction

Agent-based approaches for interactive narrative generation use intelligent agents to model the characters in a story. The agents respond to the interactions of a player with dialogue or actions fitting the shape of a story. However, these agents have little autonomy in their actions, bound as they are to the strict requirements of their role in the narrative.

An institutional model can be used as normative framework for governing the actions of agents in a story. By describing the rules of a narrative in terms of social expectations, the agents are encouraged to perform certain types of actions while still remaining free to break free of these expectations. As in society in the real world, breaking agreed norms comes with consequences, and only generally happens in exceptional circumstances.

One situation where this is desirable is with the use of emotional agents. An agent experiencing an extreme emotion in an emotional model (such as rage or depression) may be allowed to act unusually or uncharacteristically. Allowing characters to break from narrative norms enables them to be ‘pushed too far’ by circumstances, with results that add an extra dimension of richness to a story.

Through this implementation, we introduce two novel approaches: (i) the use of an institutional model to describe a narrative ‘world’ or domain, and (ii) how emotional models can give intelligent agents some degree of autonomy to both act in idiosyncratic ways and to react emotionally to input from the audience.

The puppets in the show are each belief-desire-intention (BDI) agents with a valence, arousal, dominance (VAD) emotional model described in section 5. The story is modelled by a set of institutional norms (section 6.1) that describe the Punch and Judy story domain in terms of Propp’s ‘story moves’ [8] (section 3). The agents communicate with their environment using the Bath Sensor Framework, described in section 6.3 [6]. In the final sections, we describe the animation system that functions as the agents’ environment (section 6.4), and how the audience interacts with the system (section 7).

[1] University of Bath, United Kingdom, email: m.r.thompson@bath.ac.uk

2 Propp moves and roles

To express story events as an institution, we must look to narrative theory for inspiration. Instead of describing parts of the Punch and Judy story explicitly (such as ‘Punch is expected to hit the policeman in this scene’), it is more desirable to describe scenes in a more abstract way (‘The villain fights the victim in this scene’). The use of more general story components allows us to reuse them in multiple scenes, or even in other stories.

Narratology, and structuralism in particular, supply such generalised building blocks for stories. Russian formalism is an early movement in narrative theory to formalise the elements of narrative, of which Vladimir Propp is a prominent figure.

In order to direct the course of the narrative, we use a model built upon Propp’s 1928 formalism of Russian folktales, *The Morphology of the Folktale* [8]. In this formalism, Propp identifies recurring characters and motifs in Russian folklore, distilling them down to a concise syntax with which to describe stories.

In this formalism, characters have *roles*, such as *hero*, *villain*, *dispatcher*, *false hero*, and more. Characters performing a certain role are able to perform a subset of *story moves*, which are actions that make the narrative progress. For example, the *dispatcher* might send the *hero* on a quest, or the *victim* may issue an *interdiction* to the *villain*, which is then *violated*.

Propp defines a total of 31 distinct story functions, some of which can have subtle variations from story to story. Each function is given a number and symbol in order to create a succinct way of describing entire stories. Examples of such functions are:

- One of the members of a family absents himself from home: *absentation*.
- An interdiction is addressed to the hero: *interdiction*.
- The victim submits to deception and thereby unwittingly helps his enemy: *complicity*.
- The villain causes harm or injury to a member of the family: *villainy*.

Each of these functions can vary to a great degree. For example, the *villainy* function can be realised as one of 19 distinct forms of villainous deed, including *the villain abducts a person*, *the villain seizes the daylight*, and *the villain makes a threat of cannibalism*.

These functions are enacted by characters following certain roles. Each role (or *dramatis personae* in Propp’s definition) has a *sphere of action* consisting of the functions that they are able to perform at any point in the story. Propp defines seven roles that have distict spheres of action: *villain*, *donor*, *helper*, *princess*, *dispatcher*, *hero*, and *false hero*.

In a typical story, one story function will follow another as the tale progresses in a sequential series of cause and effect. However, Propp’s formalism also allows for simultaneous story functions to occur at once.

2.1 Propp example: sausages and crocodile scene

The common elements of Punch and Judy are easily described in terms of Propp's story functions. Here we pick one scene from the Punch and Judy show to use as an example: the scene where Punch battles a crocodile in order to safeguard some sausages.

In this scene, Joey the clown (our narrator) asks Punch to guard the sausages. Once Joey has left the stage, a crocodile appears and eats the sausages. Punch fights with the crocodile, but it escapes. Joey then returns to find that his sausages are gone.

The appropriate story functions are:

1. Joey tells Punch to look after the sausages (*interdiction*).
2. Joey has some reservations, but decides to trust Punch (*complicity*).
3. Joey gives the sausages to Punch (*provision or receipt of a magical agent*).
4. Joey leaves the stage (*absentation*).
5. A crocodile enters the stage and eats the sausages (*violation*).
6. Punch fights with the crocodile (*struggle*).
7. Joey returns to find that the sausages are gone (*return*).

3 Institutional model

An institution describes a set of 'social' norms describing the permitted and obligated behaviour of interacting agents. Noriega's 'Fish Market' thesis [7] describes how an institutional model can be used to regiment the actions of agents in a fish market auction. Cliffe [3], Baines and Lee [6] extend this idea to build systems where institutions actively regulate the actions of agents, while still allowing them to decide what to do. Adapting this idea to the world of narrative, we use an institutional model to describe the story world of Punch and Judy in terms of Propp moves and character roles.

Institutional models use deontic logic to describe obligations and permissions that act on interacting agents in an environment. By combining this approach with Propp's concepts of *roles* and *story moves*, we describe a Propp-style formalism of Punch and Judy in terms of what agents are *obligated* and *permitted* to do at certain points in the story.

For example, in one Punch and Judy scene a policeman enters the stage and attempts to apprehend Punch. According to the rules of the Punch and Judy world, Punch has an obligation to kill the policeman by the end of the scene (as this is what the audience expects to happen, having seen other Punch and Judy shows). The policeman has an obligation to try his best to catch Punch. Both agents have permission to be on the stage during the scene. The policeman only has permission to chase Punch if he can see him (Punch is obligated to hide from him at the start of the scene).

The permissions an agent has constrain the choices of actions available to them at any given moment. Obligations affect the goals of an agent. Whether or not an agent actively tries to fulfil an obligation depends on their emotional state.

3.1 Institution example

Here we continue the 'sausages and crocodile' scene example from section 3.1, taking the Propp story functions and describing them as an institutional model.

We define our institution in terms of *fluents*, *events*, *powers*, *permissions* and *obligations*.

3.1.1 Fluents

Fluents are properties that may or may not hold true at some instant in time. *Institutional events* are able to *initiate* or *terminate* fluents at points in time. A fluent could describe whether a character is currently on stage, the current scene of a story, or whether or not the character is happy at that moment in time.

Domain fluents ($\mathcal{D}$) describe domain-specific properties that can hold at a certain point in time. In the Punch and Judy domain, these can be whether or not an agent is on stage, or their role in the narrative (equation 1).

$$\mathcal{D} = \{\text{onstage, hero, villain, victim, donor, item}\} \quad (1)$$

Institutional fluents consist of *institutional powers*, *permissions* and *obligations*.

An **institutional power** ($\mathcal{W}$) describes whether or not an external event has the authority to meaningfully generate an institutional event. Using Propp as an example, an *absentation* event can only be generated by an external event coming from a *donor* character (such as their leaving the stage). Therefore, any characters other than the donor character would not have the institutional power to generate an *absentation* institutional event when they leave the stage.

Equation 2 shows a list of possible empowerments, essentially a list of institutional events.

$$\mathcal{W} = \{\text{pow(introduction, interdiction, give, absentation,} \\ \text{violation, return)}\} \quad (2)$$

Permissions ($\mathcal{M}$) are external actions that agents are permitted to do at a certain instant in time. These can be thought of as the set of *socially permitted* actions available to an agent. While it is possible for an agent to perform other actions, societal norms usually prevent them from doing so.

For example, it would make sense in the world of Punch and Judy if Punch were to give the sausages to the Policeman. It is always Joey who gives the sausages to Punch. Also, it would be strange if Joey were to do this in the middle of a scene where Punch and Judy are arguing. We make sure agents' actions are governed so as to allow them only a certain subset of permitted actions at any one time. Equation 3 shows a list of permission fluents.

$$\mathcal{M} = \{\text{perm(leavestage, enterstage, die, kill,} \\ \text{hit, give, fight)}\} \quad (3)$$

Obligations ($\mathcal{O}$) are actions that agents *should* do before a certain deadline. If the action is not performed in time, a *violation event* is triggered, which may result in a penalty being incurred. While an agent may be obliged to perform an action, it is entirely their choice whether or not they actually do so. They must weigh up whether or not pursuing other courses of action is worth suffering the penalty that an unfulfilled obligation brings.

Anybody who has seen a Punch and Judy show knows that at some point Joey tells Punch to guard some sausages, before disappearing offstage. Joey's departure is modelled in the institution as the *absentation* event. It could be said that Joey has an obligation to leave the stage as part of the *absentation* event, otherwise the story function is violated. Equation 4 shows how this would be described in the institution.

$$\mathcal{O} = \{\text{obl(leavestage, absentation, viol(absentation))}\} \quad (4)$$

3.1.2 Events

Cliffe's model specifies three types of **event**: *external events* (or 'observed events', $\mathcal{E}_{obs}$), *institutional events* ($\mathcal{E}_{instact}$) and *violation events* ($\mathcal{E}_{viol}$).

External events are observed to have happened in the agents' environment, which can *generate institutional events* which act only within the institional model, *initiating* or *terminating* fluents, permissions, obligations or institutional powers. An external event could be an agent leaving the stage, an agent hitting another, or an agent dying. Internal events include narrative events such as scene changes, or the triggering of Propp story functions such as *absentation* or *interdiction* (described in section 3). Violation events occur when an agent has failed to fulfil an obligation before the specified deadline. These can be implemented in the form of a penalty, by decreasing an agent's health, for example.

$$\begin{aligned}\mathcal{E}_{obs} = \{&\text{startshow, leavestage, enterstage, die, give,}\\ &\text{harmed, hit, fight, kill, escape}\} \quad (5)\\ \mathcal{E}_{instact} = \{&\text{introduction, interdiction, give, absentation,}\\ &\text{violation, return, struggle, defeat, complicity,}\\ &\text{victory, escape}\} \quad (6)\\ \mathcal{E}_{viol} = \{&\text{viol(introduction), viol(interdiction), viol(give),}\\ &\text{viol(absentation), viol(violation), viol(return),}\\ &\text{viol(struggle), viol(defeat), viol(complicity)}\\ &\text{viol(victory), viol(escape)}\} \quad (7)\end{aligned}$$

3.1.3 Event Generation and Consequences

An **event generation** function, $\mathcal{G}$, describes how events (usually external) can generate other (usually institutional) events. For example, if an agent leaves the stage while the *interdiction* event holds, they trigger the *leavestage* event. This combination generates the *absentation* institutional event (equation 11).

Event generation functions follow a $\langle\text{preconditions}\rangle \rightarrow \{\text{postconditions}\}$ format: $\langle\mathcal{G}(\mathcal{X},\mathcal{E})\rangle \rightarrow \{\mathcal{E}_{out}\}$, where $\mathcal{X}$ is a set of fluents that hold at that time, $\mathcal{E}$ is an event that has occurred, and $\mathcal{E}_{out}$ are the events that are generated. They are generally used to generate internal, institutional events from external events.

Consider the Punch and Judy scenario described in section 3.1. There are seven institutional events (story functions) that occur during this scene: *interdiction*, *complicity*, *receipt* (from Propp's *receipt of a magical agent*) *absentation*, *violation*, *struggle*, *return*. These institutional events are all generated by external events. The *interdiction* is generated when Joey tells Punch to protect the sausages. Punch agreeing amounts to *complicity*. Joey *gives* punch the sausages (*receipt*), then leaves the stage (*absentation*). The crocodile eating the sausages is a *violation* of Punch's oath, the agents fight (*struggle*), then Joey enters the stage again (*return*).

It is desirable that these story function occur in this sequence in order for a satisfying narrative to emerge. Agents may decide to perform actions that diverge from this set of events, but the institution is guiding them towards the most fitting outcome for a *Punch and Judy* world. For this reason, a currently active story function can be the precondition for event generation. For example, the *receipt* event may only be triggered if an agent externally performs a *give* action **and** if the *complicity* event currently holds (equation 10).

Examples of event generation function for this scenario, complete with preconditions, are listed in equations 8 to 14.

$$\begin{aligned}\mathcal{G}(\mathcal{X},\mathcal{E}) :&\langle\emptyset, \textit{tellprotect}(\text{donor, villain, item})\rangle\\ &\rightarrow \{\textit{interdiction}\} \quad (8)\\ &\langle\{\textit{interdiction}\}, \textit{agree}(\text{villain}))\rangle\\ &\rightarrow \{\textit{complicity}\} \quad (9)\\ &\langle\emptyset, \textit{give}(\text{donor, villain, item}))\rangle\\ &\rightarrow \{\textit{receipt}\} \quad (10)\\ &\langle\{\textit{interdiction}\}, \textit{leavestage}(\text{donor})\rangle\\ &\rightarrow \{\textit{absentation}\} \quad (11)\\ &\langle\{\textit{interdiction}\}, \textit{harmed}(\text{item})\rangle\\ &\rightarrow \{\textit{violation}\} \quad (12)\\ &\langle\{\textit{interdiction, absentation}\},\\ &\textit{enterstage}(\text{donor}),\ \text{onstage(villain)}\rangle\\ &\rightarrow \{\textit{return}\} \quad (13)\\ &\langle\emptyset, \textit{hit}(\text{donor, villain})\rangle\\ &\rightarrow \{\textit{struggle}\} \quad (14)\end{aligned}$$

Consequences consist of fluents, permissions and obligations that are *initiated* ($\mathcal{C}^{\uparrow}$) or *terminated* ($\mathcal{C}^{\downarrow}$) by institutional events. For example, the institutional event *give* could initiate the donor agent's permission to leave the stage, triggering the *absentation* event (equation 16). When the *interdiction* event is currently active and a *violation* event occurs, the interdiction event is terminated (21). Equations 15 to 22 describe the initiation and termination of fluents in the Punch and Judy sausages scenario detailed in section 3.1.

$$\begin{aligned}\mathcal{C}^{\uparrow}(\mathcal{X},\mathcal{E}) :&\langle\emptyset, \text{interdiction}\rangle\\ &\rightarrow \{\text{perm(give(donor, villain, item))}\} \quad (15)\\ &\langle\emptyset, \text{receipt}\rangle\\ &\rightarrow \{\text{perm(leavestage(donor))}\} \quad (16)\\ &\{\textit{active}(\textit{interdiction})\}, \text{violation}\rangle\\ &\rightarrow \{\text{perm(enterstage(dispatcher))}\} \quad (17)\\ &\{\textit{active}(\textit{absentation}), \textit{active}(\textit{violation})\}, \text{return}\rangle\\ &\rightarrow \{\text{perm(hit(donor, villain))}\} \quad (18)\\ \mathcal{C}^{\downarrow}(\mathcal{X},\mathcal{E}) :&\langle\emptyset, \text{interdiction}\rangle\\ &\rightarrow \{\text{perm(give(donor, villain, item))}\} \quad (19)\\ &\langle\{\textit{active}(\textit{interdiction})\}, \text{absentation}\rangle\\ &\rightarrow \{\text{perm(leavestage(donor))}\} \quad (20)\\ &\langle\{\textit{active}(\textit{interdiction})\}, \text{violation}\rangle\\ &\rightarrow \{\textit{active}(\textit{interdiction})\} \quad (21)\\ &\langle\{\textit{active}(\textit{absentation}), \textit{active}(\textit{violation})\}, \text{return}\rangle\\ &\rightarrow \{\textit{active}(\textit{absentation})\} \quad (22)\end{aligned}$$

4 VAD emotional model

In order to make the agents acting out the Punch and Judy show more believable, we apply an emotional model to affect their actions and decisions. For this, we use the valence-arousal (circumplex) model first described by Russell [10].

In order to give each character its own distinct personality, we extend this model with an extra dimension: dominance, as used by

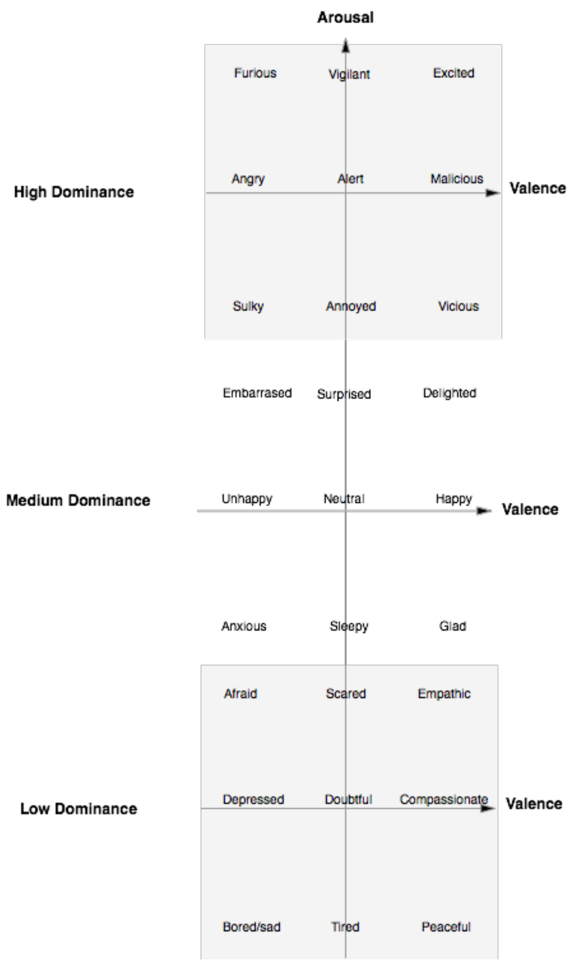

Figure 1. VAD emotional values, adapted from Ahn et al [1]

Ahn et al in their model for conversational virtual humans [1]. This dominance level is affected by the reactions of the audience to the agents' actions. For example, Judy may become more dominant as her suggestions to hit Punch with a stick are cheered on by the audience, emboldening her into acting out her impulses.

Figure 1 shows how valence, arousal and dominance values map to identifiable emotions. Valence, arousal and dominance can each have a value of low, medium or high. This allows the agents to have a total of 27 distinct emotional states.

Valence and arousal levels of each agent are affected by the actions of other agents. For example, a character being chased around the stage by Punch will see their valence level drop while their arousal increases. According to Russell's circumplex model of emotion [10], this would result in them becoming *afraid* (if their dominance level is low).

An agent's emotional state affects its ability to fulfil its institutional obligations. An agent that is *furious* would have no problem carrying out an obligation that requires them to kill another agent. If that same agent is *happy* or *depressed*, however, they might not have the appropriate motivation to perform such a violent action.

5 Architecture

5.1 Multi-Agent System

We use the JASON framework for belief-desire-intention (BDI) agents [2], programming our agents in the AgentSpeak language.

The VAD emotional model is represented inside each agent as a set of beliefs. Each agent has beliefs for its *valence*, *arousal* and *dominance* levels, each of which can take the value of low, medium or high. This combination of VAD values creates one of the 27 emotional states shown in figure 1, affecting whether or not an agent breaks from its permitted or obliged behaviour.

5.2 Institutional Framework

To describe our institutional model, we use instAL [3], a DSL for describing institutions that compiles to AnsProlog, a declarative programming language for Answer Set Programming (ASP). instAL's semantics are based upon the Situation Calculus [9] and the Event Calculus [5]. It is used to describe how external events generate institutional events, which then can initiate or terminate fluents that hold at certain instances in time. These fluents can include the permissions and obligations that describe what an agent is permitted or obligated to do at specific points in time.

For example, if an agent with the role of *dispatcher* leaves the stage, it generates the *absentation* Propp move in the institution:

```
leaveStage(X) generates intAbsentation(X) if
    role(X, dispatcher), activeFunction(
    interdiction);
```

The *absentation* institutional event gives the crocodile permission to enter the stage if there are any sausages on the stage. It also terminates the permission of the absented agent to leave the stage, as they have already done so:

```
intAbsentation(X) initiates perm(enterStage(
    croc)) if objStage(sausages);
intAbsentation(X) terminates onStage(X), perm(
    leaveStage(X));
```

instAL rules like those shown above are compiled into AnsProlog ASP rules. Once the instAL model is compiled to AnsProlog, we use the *clingo* answer set solver [4] to ground the logical variables, and 'solve' queries by finding all permissions and obligations that apply to any agents, given a sequence of events as the query input. The agents' percepts are then updated with their permitted and obliged actions from that moment in time onwards.

5.3 Bath Sensor Framework

The Bath Sensor Framework (BSF) [6] is a framework supporting publish/subscribe-style communication between distributed software components, in this case connecting intelligent agents with their virtual environments. It uses the XMPP publish/subscribe protocol to allow the communication between agents and their environments. Each agent subscribes to receive notifications of environment changes via XMPP server, which relays messages between publishers and subscribers. If any environment change occurs, all subscribed agents are informed of the changes.

This allows agents' environments to be created using entirely different technologies and programming languages from the agents themselves. In our case, BSF is especially useful as the animation engine that acts as the agents' environment is written in Javascript and runs in the browser. This means that the clingo solver and JASON agent

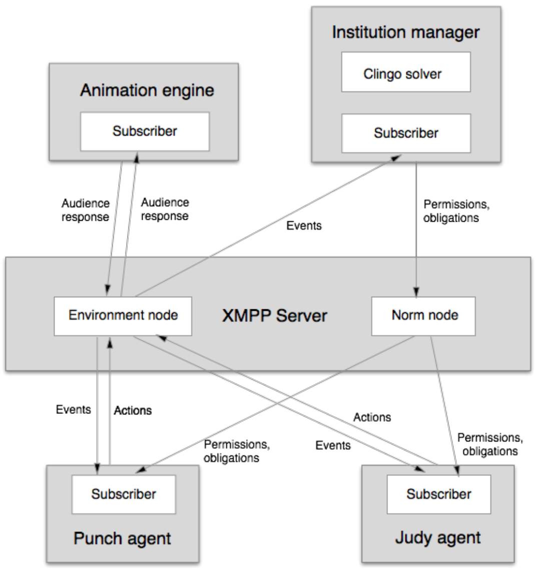

Figure 2. System architecture

framework can run on a central web server and communicate to any connected clients using BSF and XMPP.

Figure 2 shows how BSF is used to coordinate the components of the system. An XMPP server runs two publish/subscribe nodes. One node is for events related to changes in the environment (the *environment* node), the other is for changes in agents' permissions and obligations (the *norm* node).

All agents (in this case, Punch, Judy, the Policeman, etc) are subscribed to both the environment and norm nodes. They can also publish events to the environment node, but not the norm node. Only the institution manager (connected to the *clingo* solver) can publish permissions and obligations to the norm node. This manager (labelled in figure 2 as *institution manager*) is subscribed to the environment node of the XMPP server, watching it for events. These events then get passed to the *clingo* solver with the institutional model, which outputs the new permissions and obligations, publishing them to the norm node.

The animation engine is subscribed to the environment node, watching it for any events that need animating for the puppet show. In addition, it can publish input from the audience ('cheers' or 'boos') as events to the same node.

5.4 Animation

The animation engine that shows the visual output of the agents actions is written in Javascript and the Phaser game framework. It runs entirely in a browser, and communicates with BSF using the Strophe XMPP library.

If the user allows the program access to their microphone, they can cheer or boo the actions of the agents by shouting into the microphone. Otherwise, they can simulate these actions by clicking on 'cheer' or 'boo' buttons at the bottom of the screen.

6 Audience Interaction

The puppet show is designed to be run in front of either a single user's computer, or on a large display in front of an audience. The

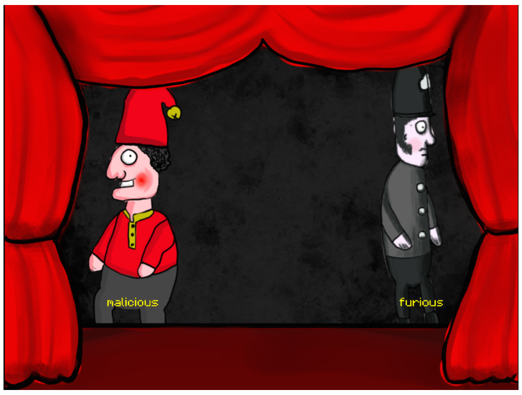

Figure 3. A screenshot of the Punch and Judy show

user/audience is instructed to cheer or boo the actions of the characters of the show, which will be picked up by a microphone and 'heard' by the agents. This will then affect the emotional state of the agents and change the actions they make in the show. Their actions are constrained by the set of 'Punch and Judy' world norms as described in the institutional model.

There are many different ways in which the audience's responses can affect the outcomes of the show. If the audience craves a more 'traditional' Punch and Judy experience, then they can cheer Punch into beating and killing all of his adversaries (including his wife, Judy). Alternatively, a more mischievous audience could goad Judy into killing Punch and then taking over his role as sadist and killer for the rest of the show. The narrative outcomes are dependent on how the audience responds to the action, yet still conform to the rules of the Punch and Judy story world.

7 Conclusion

With our approach to interactive narrative generation, we regulate the rules of the story domain using an institutional model. This model describes what each agent is permitted and obligated to do at any point in the story. This approach alone would be too rigid, however. Though the audience's interactions (cheering or booing) may alter the course of the narrative, the agents would still have to blindly follow a pre-determined set of paths. By giving our agents emotional models that change their willingness to follow the narrative, a degree of unpredictability is added to each run-through of the show, giving the impression that the agents are indeed characters capable of free will.

REFERENCES

[1] Junghyun Ahn, Stéphane Gobron, David Garcia, Quentin Silvestre, Daniel Thalmann, and Ronan Boulic, 'An NVC emotional model for conversational virtual humans in a 3d chatting environment', in *Articulated Motion and Deformable Objects*, 47–57, Springer, (2012).
[2] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, volume 8, John Wiley & Sons, 2007.
[3] Owen Cliffe, Marina De Vos, and Julian Padget, 'Specifying and reasoning about multiple institutions', in *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, 67–85, Springer, (2007).
[4] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider, 'Potassco: The potsdam answer set solving collection', *Ai Communications*, **24**(2), 107–124, (2011).
[5] Robert Kowalski and Marek Sergot, 'A logic-based calculus of events', in *Foundations of knowledge base management*, 23–55, Springer, (1989).

[6] Jeehang Lee, Vincent Baines, and Julian Padget, ‘Decoupling cognitive agents and virtual environments’, in *Cognitive Agents for Virtual Environments*, eds., Frank Dignum, Cyril Brom, Koen Hindriks, Martin Beer, and Deborah Richards, volume 7764 of *Lecture Notes in Computer Science*, 17–36, Springer Berlin Heidelberg, (2013).

[7] Pablo Noriega, *Agent mediated auctions: the fishmarket metaphor*, Citeseer, 1999.

[8] Vladimir Propp, ‘Morphology of the folktale. 1928’, *Trans. Svatava Pirkova-Jakobson. 2nd ed. Austin: U of Texas P*, (1968).

[9] Raymond Reiter, ‘The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression’, *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, **27**, 359–380, (1991).

[10] James A Russell, ‘A circumplex model of affect.’, *Journal of personality and social psychology*, **39**(6), 1161, (1980).