A Phone That Cures Your Flu: Generating Imaginary Gadgets in Fictions with Planning and Analogies

Boyang Li and Mark O. Riedl
School of Interactive Computing, Georgia Institute of Technology
{boyangli, riedl}@gatech.edu

Abstract

Most computational story generation systems lack the ability to generate new types of imaginary objects that play functional roles in stories, such as lightsabers in *Star Wars*. We present an algorithm that generates such imaginary objects, which we call gadgets, in order to extend the ontological expressivity of existing, planning-based story generation systems. The behavior of a gadget is represented as a plan including typical events that happen when the gadget is used. Our algorithm creates gadgets by extrapolating and merging one or more commonly known objects in order to achieve a narrative goal provided by an existing story generator. We extend partial-order planning to establish open conditions based on analogies between concepts related respectively to common objects and the gadget. We show the algorithm is capable of generating gadgets created by human.

Introduction

Since early days of Artificial Intelligence (AI), one of the goals has been to procedurally simulate the human ability of storytelling. Many story generation systems (Meehan 1981; Lebowitz 1985; Turner 1992; Pérez y Pérez and Sharples 2001; Cavazza, Charles, and Mead 2002; Riedl and Young 2010; Gervás et al. 2005) begin with a predefined world configuration. Such configurations include unchangeable facts about the fictional world such as what objects exist, how they relate to each other and what events can happen. With the initial world configuration, story generators build stories, the execution of which transform and evolve the world. As most story generators accept the initial world as a given rather than construct their own, they are limited in their creativity and expressivity. Some story generators can create characters and objects either before the story (Lebowitz 1984) or when stories need them (Dehn 1981; Riedl and Young 2006; Swartjes and Theune 2009; Ware and Young 2010). These systems are still limited in their ability to construct the world because they can only initiate new instances of known types of objects; they cannot create new types of objects.

In contrast, creative literary works produced by human often feature objects never envisioned before, such as lightsabers in *Star Wars* and the magic mirror in *Snow White*. These objects possess special powers to achieve the impossible or the improbable due to futuristic technologies or magic. They facilitate and sometimes dictate story development. In fact, the gadget story is proposed as one of the four subgenres of science fiction (Malmgren 1991). The ability to create such objects, which we call gadgets, can relax the reliance on human-constructed fictional worlds of AI story generators and greatly improve their expressivity.

We present a computational approach for creating new types of magical and science fiction objects by extrapolating and combining existing object types. The approach described here augments the creativity of plan-based story generators such as that by Riedl and Young (2006). We empower a traditional story planner with the ability to plan with analogies. We incrementally modify behaviors of known objects based on a consistent set of analogies with backward chaining and combine behaviors of multiple objects to create a new behavior. The process results in a new gadget that can cause desired changes in the fictional world that are impossible or improbable to achieve by other means.

Background and Related Work

Cognitive research on narrative comprehension suggests that a story can be modeled as a sequence of interrelated events that transform a fictional world. In the mental

models of readers, narratives are segmented into discrete event structures (Zacks, Speer, and Reynolds 2009). Furthermore, people can perceive events of different granularities organized in hierarchies where a large event can include several small events (Zacks and Tversky 2001). Causality and temporality between events are important constituents of mental models of narratives, directly affecting comprehension (cf. Zacks, Speer, and Reynolds 2009; Zwann, Magliano, and Graesser 1995). Causal relationships between events allow readers to make inference about narratives and missing causal links may hinder comprehension (Trabasso and van den Broek 1985).

An AI formalism that corresponds to such a mental model and captures a sequence of events as well as temporal and causal relationships between them is a partial-order plan. In story generation, a plan may be used to imitate mental models of stories and make inferences about readers' perception of stories (Young 1999). This leads to the development of story planners (Lebowitz 1985; Riedl and Young 2010; Porteous and Cavazza 2009; Li and Riedl 2010). Story planners require both an initial state and the outcome to be completely specified before a story can be made. The initial state describes the world before the story happens, and the goal situation describes changes the story caused when it ends. The planning algorithm generates a plan as a feasible path linking the beginning and the end of the story. Traditional story planners, however, have limited expressivity because they have to accept both a given beginning and a given outcome.

AI storytellers by Dehn (1981), Lebowitz (1984) and Riedl & Young (2006) are capable of creating some aspects of their own fictional worlds. Dehn (1981) models a storyteller as an autonomous agent that deliberately places objects and characters in the fictional world to achieve author goals as the story develops. Lebowitz (1984) constructs the cast of characters for fictional world before story generation takes place. Riedl & Young (2006) extend story planning with the ability to accept or deny certain facts in the initial state, such as existence of an object (thus creating an object), an attribute of an existing object, or a relation between two existing objects. This approach is further elaborated upon by Swartjes and Theune (2009) and Ware and Young (2010). However, in these systems, types of objects and characters dynamically created must be known. These types specify object behaviors, so type information must be known in advance so that created objects and characters behave correctly.

In this paper, we empower story generators with the ability to create new types of objects previously unknown in service of a story being created by a story generator. Thus, our system is more creative than those systems reviewed above. Ryan (1991) proposes that readers re-construct the fictional world while reading. Initially they assume unmentioned aspects of the fictional world as minimally departing from reality. Learning about the fictional world bit by bit through the story can be thought of as a search for possible worlds the story could be set in. In this light, our procedure is tantamount to re-configuring the possible world of the story. We create gadgets as minimal departures of common objects so that readers' knowledge of common objects can help them understand the gadget and accept it easily.

Partial-Order Story Planning

Following plan-based story generators, we model a story as a *partial-order plan*. A partial-order plan consists of actions as well as temporal and causal links between actions. Actions encode preconditions, which must be true for the event to occur, and effects, which become true once the event completes. Preconditions and effects are first-order logic predicates stating facts, such as `contain(box, candy)`. The type of the predicate is `contain`, and `box` and `candy` are objects it takes as arguments. A *causal link*, denoted as $a_1 \rightarrow^c a_2$, indicate that an effect of event a_1 establish a precondition c necessary for event a_2. Causal links act as protected intervals during which the truth of predicate c in the world must be maintained. *Temporal links* indicate ordering constraints between events. Both events and predicates can be parameterized with symbolic references to objects of known types. For example, an `Move(?o, ?l1, ?l2)` event, taking one object `?o` and two locations `?l1, ?l2` as arguments, has the effect that the object is moved from one location to another.

Before planning, an initial world state and a desired goal situation are specified as two sets of predicates. All predicates in the goal situation and all preconditions of events in the plan must be established. Otherwise, they are called open conditions. An action library contains all un-parameterized action templates. During story planning, actions are drawn from the library, given appropriate arguments, and inserted into the plan. A causal link can extend from the initial state or from an effect of an action to establish an open condition. A planning algorithm is a refinement search that gradually adds events, causal links and temporal links to produce a sound plan – one that does not contain open conditions and guarantees to reach the goal situation from the initial state. See Weld (1994) for more details on partial-order planning.

Gadget Generation

We formulate the gadget generation problem as follows: find a new type of object which, when used by a character in the story, causes the desired change in world state. We should be able to describe the object, or gadget, in sufficient details that it can be accepted and believed by

readers. Many gadgets in fictions are imaginary, but readers usually understand and accept them easily. To promote readers' suspension of disbelief, we adhere to two principles. First, the gadget's behavior is similar to objects readers already understand, so knowledge about the old can be brought to understand the new, as suggested by the minimal departure principle (Ryan 1991). A lightsaber, for example, is similar to a sword, so readers understand it is used to slash, pierce or perform other functions of a sword. Second, the gadget caters to biological traits and limits of humans. For example, a person should not have to walk on water in order to use a gadget unless she or he already possesses that ability; although humans in the fictional world may possess special abilities, modifying human behaviors to suit a particular gadget is outside the scope of gadget generation. Thus, in order to keep the believability of gadgets, our system use common objects as prototypes for gadgets and prefers to minimize modifications to prototypes. Our algorithm creates a new object type through a combination of analogical mapping of elements from the prototype to the gadget and planning to fill in additional details. During the modification, we prevent gadgets from requiring unnatural human behaviors.

This paper focuses on generating step-by-step behaviors for gadgets. We represent the behavior of an object as a partial-order plan called a *usage frame*. A usage frame describes the sequence of events expected to happen during a typical use of the object, including how people operate it and how it affects the world. A usage frame can take different arguments, as an object can be used by different people in different occasions. A usage frame is summarized into a single event of the object being used inside the story plan, forming an event hierarchy. Such a hierarchy supports flexible description of gadgets in different media. That is, a narrative text may simply mention the gadget is used, but a movie or comic may show each step of its usage.

An example usage frame of a toy phone, made of two cups attached by a string, is shown in Figure 1. The goal state of the frame and some causal links are omitted for clarity. Boxes denote events in the frame. To differentiate from events in the story plan, we refer to events in usage frames as *actions*. Thick arrows denote causal links, and dashed arrows denote temporal links. In other words, the frame describes the situation when two people each pick up one cup of the toy phone and one speaks into a cup. The phone transmits the voice so it is heard at the other end. Dotted arrows in the frame denote *closure events*. Closure events restore the world to a normal or routine state after other events change it. They are not necessary for a gadget's intended purpose, but they complete the frame and may improve coherence of the story. Here, the events where people release the two cups of the toy phone are closure events, which "close" the events where the cups are

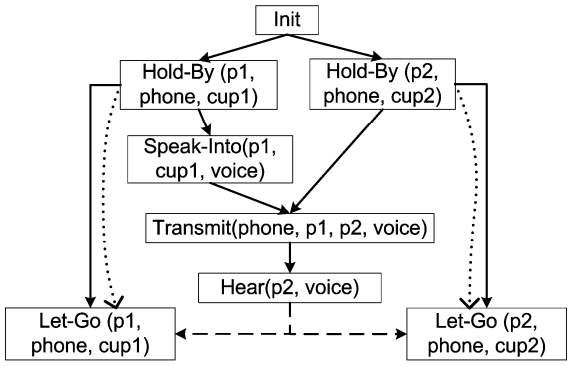

Figure 1. The usage frame of a normal toy phone.

picked up. Those two events prevent people from holding on to the phone after using it.

The narrative generation process is initiated by a partial-order story generator, which supplies narrative goals that the gadget should accomplish. After that, a common object is identified as a prototype. The usage frame of the prototype is modified incrementally based on necessary analogies to become the usage frame of the gadget. The algorithm attempts to preserve the structure of the prototype while ensuring the presence of causally necessary actions. As part of an iterative process, the system may determine if usage frames of more than one common object should be combined. For example, if an open condition requires the gadget to fly, the algorithm can retrieve an airplane as a second prototype and transplant its flying operation onto the gadget.

We should note that both the appearance and the behavior are necessary to describe a gadget; the more detailed its description, the more vivid and believable the gadget becomes. Further, when presentational aspects of storytelling come into play, one needs to be able to describe or show how the gadget is used in a way that is suitable for the media used (text, computer games, comics, cinematography, etc.). However, presentation of the gadget appearance in the story is beyond the scope of the paper.

Narrative Goals of Gadgets

As a partial-order planning story generator iteratively establishes open conditions, it may invoke gadget generation when a gadget is deemed the best option to achieve an open condition p in the story, which becomes the narrative goal of the gadget. There are three reasons to generate a gadget to achieve a goal. First, the goal may be impossible or too difficult to achieve without assistance of a gadget (e.g. stopping the rain). Second, the goal may require unpleasant actions or significant time commitments

from the protagonist, such as housework, that the character in question generally wants to avoid. The third reason is the lack of reliable means to achieve the goal, such as winning a lottery. Admittedly, a story planner can make any events happen no matter how unlikely. However, a story character in constant pursuit of unlikely events appears irrational. The believability of the story is further damaged if such a character always appears successful. Here, a gadget which makes the improbable happen can believably justify this outcome and rescue the story.

In this paper, we assume one or more narrative goals are given by a story generation system and focus on subsequent gadget generation. Once the gadget usage frame is complete, we summarize it to form a single "use-gadget" event and insert the event into the story in order to establish the narrative goal.

Retrieving a Prototype Object

We use a knowledge base of existing objects that are known a priori. Usage frames of these objects are manually authored, stored and indexed by predicates they are typically employed to achieve. When gadget generation is initiated to achieve a narrative goal *p*, the system searches for tools that achieve a predicate analogous to *p*. Saunders and Gero (2004) propose that an artifact is usually considered the most creative when it is neither too similar nor too dissimilar to what we already know. Following that, an object with optimally moderate similarity is attempted first, and becomes the prototype of the new gadget. The algorithm may backtrack and try a different tool if the first trial fails.

Computing Analogies

Analogy is critical in retrieval of the prototype object and subsequent transformation. We employ Sapper (Veale and Keane 1994) as the analogy-making engine. All known object types, predicates, and actions are stored in a pre-authored semantic network that contains attributes of and relations between objects. Objects types are considered analogous if they share attributes or are involved in the same relations. Analogies between predicates and actions are recursively supported by analogies or literal matches between their corresponding arguments. In addition, semantic roles, such as subject, object, etc., of arguments in predicates and events are annotated to facilitate mapping. Furthermore, we utilize the notion of spatial signatures (Veale and Keane 1992) to capture similarities between predicates and actions. For instance, climbing a staircase and advances of career both implies upward movements, so a metaphor can be created between them. Figure 2 shows some examples of spatial signatures, which help to establish analogies between two predicates `heard` and `infected-by`, and two actions `Speak-Into` and

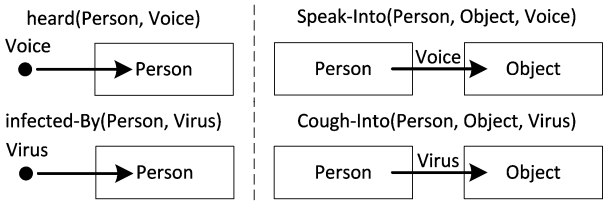

Figure 2. Spatial signatures of some predicates (left) and actions (right)

`Cough-Into`. The basic idea in transformation is that if two object types, predicates, or actions are analogous enough they can stand in place for each other in a creative domain.

Constructing the Usage Frame

The primary function of gadget generation is to construct a usage frame for an unknown gadget. Extending the partial-order planning (POP) algorithm (Weld 1994), we propose new analogy-based methods to establish open conditions in the gadget usage frame.

The gadget usage frame starts as an empty plan with an empty initial state and the narrative goal *p* being the only open condition. During planning, an action may be added to the gadget frame to establish an open condition, and preconditions of the new action will become new open conditions. The algorithm continues to add actions and predicates to the gadget frame in a back-chaining manner until all open conditions are satisfied. Given an open condition c_g in the gadget frame, there are four methods to establish it, as shown in Figure 3:

- Insert an action with an effect equal to c_g
- Modify the initial state of the usage frame by inserting c_g into the frame's initial state
- Reuse existing predicates from the initial state or effects of existing actions
- Assume it is achieved by special "gadget powers"

During each iteration of the search, each of the four methods is attempted, which may generate one or more frames in which c_g is satisfied. These frames become part of the search frontier. At each iteration of the algorithm, the best frame is chosen from the search frontier based on heuristic values. We backtrack to alternative frames when we cannot satisfy an open condition using any of these methods. We next describe how analogical reasoning is incorporated into planning and each of the four methods.

As with traditional story planners, the main purpose of the algorithm is to construct a gadget's usage frame by incorporating new content into the frame. A traditional POP planner inserts actions (see Weld 1994), and the planner by Riedl and Young (2006) additionally inserts predicates into the initial state of the plan. Here, we introduce *projection* as a new method of inserting content into a plan to establish open conditions. A projection

copies an element from the prototype usage frame – either an action or a predicate in the initial state – and inserts it into the new gadget usage frame either literally or through an analogical transformation. A literal projection simply copies an element over. An analogous projection transforms the element projected based on analogies between the two frames. In order to keep the resemblance between the gadget and the prototype, we prefer literal projections to analogous projections. When an action or a predicate is projected, all referenced frame arguments are also copied over into the gadget frame. Each element can only be projected once. Following the flow of the algorithm in Figure 3, we discuss projection of actions, projection of predicates into the usage frame initial state, then reuse of elements, and finally special rules.

Projecting and Inserting Actions

Before projecting any actions from the prototype frame to the gadget frame, we first find correspondence of conditions between the two frames. We look for a predicate in the prototype frame that corresponds to the open condition c_g. To do so, we first find within the gadget frame the action which c_g is a precondition of. We call this action B_g. If an earlier projection has projected an action B_p in the prototype frame to become action B_g in the gadget frame, one precondition c_p of B_p must have become c_g. This precondition c_p of the projected action B_p is the predicate we look for. If no such projection happened, we look for a precondition c_p of any action in the prototype frame that is most analogous to c_g. If c_g is a predicate in the goal situation, we find the predicate in the prototype goal situation that is the most analogous to c_g.

Once c_p is identified, we look at how it is satisfied in the prototype frame. If c_p is satisfied by an effect of action A_p in the prototype frame, we then project A_p to the gadget frame to satisfy c_g. However, how exactly A_p is projected is determined by the relationship between c_p and c_g. If c_p is identical to c_g, we can directly copy A_p into the gadget frame (shown as A1 in Figure 3). We refer to this as a literal projection. If c_p is not identical but analogous to c_g, there are two possible analogous projections. The first is to transform A_p by directly changing its constraints and effects, so that its effect c_p will match c_g (shown as A2). We call this operation an analogical transformation, and will explain the details later. If this fails or is not applicable, we look in the action library for an action A_g such that (1) the action A_g is analogous to the action A_p, and (2) A_g establishes precondition c_g (shown as A3). Note here the analogy is made between two actions. To compare A2 and A3, method A3 draws an unmodified action from the event library, whereas A2 modifies a known action to create a new action depending on the analogy between c_p and c_g. Finally, if none of A1, A2 or A3 works, we simply insert an action satisfying c_g from the library (A4). Note the method A4 does not involve projection and exists in traditional POP. It is used only as a last resort.

The CONSTRUCT-GADGET algorithm takes the prototype frame F_p, the gadget frame F_g, an action template library, and a narrative goal p.

1. Add p to the open condition list of the gadget frame F_g.

2. Choose an open condition c_g in F_g. Find in F_p the corresponding predicate c_p and the initial state or action a_p that establishes it. Non-deterministically do one of the following:

- Insert An Action: If a_p is an action then insert a new action a_g created by one of the following methods (try from top to bottom until one succeeds):
  - A1. If $c_g = c_p$ then $a_g = a_p$.
  - A2. Analogically transform a_p to create a new action a_g with the same type which achieves p.
  - A3. Find an action a_p from the library with a different type such that a_p is analogous to a_g and achieves p.
  - A4. Find an action a_p from the library with a different type such that a_p achieves p.
- Revise Initial State: If a_p *is* the initial state of F_p, insert c_g into the initial state of the prototype frame. Find which of the following is true to compute heuristic value.
  - I1. $c_g = c_p$
  - I2. c_p is analogous to c_g.
  - I3. None of the above.
- Reuse Existing Elements: Solve c_g with reuse if:
  - R1. c_g exists in the initial state
  - R2. c_g is an effect of an existing action
  - R3. An existing action can be analogically transformed to produce c_g without affecting other causal links.
- Use Gadget Power: Remove c_g. Consider it as repaired by high-tech or magical powers of the gadget. Only applies to limited types of predicates.

3. Establish corresponding causal links as in Weld (1994). Remove c_g from the open condition list of F_g).

4. Pick another open condition from F_g and repeat 2-4 until F_g contains no more open conditions.

Figure 3. Repair open conditions in gadget frames

Projecting and Inserting Predicates into Initial State

When the corresponding condition c_p in the prototype frame is established by the initial state, instead of an action, we can establish a non-goal open condition c_g in the gadget frame in the same way. The second method is used to insert predicates into the gadget's initial state to establish open conditions, provided the inserted predicates do not conflict with existing ones. Again, we prefer to use projection to insert predicates into initial states, and prefer literal projections to analogous projections.

Our preference distinguishes three different cases with decreasing priority. We can perform a direct projection, adding c_p directly to the gadget's initial state, if we find the predicate c_p in the prototype frame identical to c_g (shown as I1). If c_p is not identical but analogous to c_g, we still insert c_p to the gadget's initial state (I2), but the operation

becomes an analogous projection rather than a literal projection. If neither applies, we force the insertion of c_p into the initial state as a last resort (I3). The three methods produce the same initial state, but they generate different heuristic values for the frame produced.

Reusing Existing Actions and Predicates

The third method reuses an effect of an existing action (R1) or a predicate from the initial state (R2) to establish the open condition c_g within the gadget frame, as in traditional POP. We can also analogically transform an existing action so that it establishes c_g, only when doing so does not break predicates already established (R3).

Gadget Powers

Finally, the "gadget power" method simply removes c_g without resolving it, which is an appeal to magical or high-tech properties of the gadget itself. For example, the requirement that direct line of sight can be removed from a telescope, resulting in a gadget that can see through walls. Whether this method is used is controlled by rules capturing the human author's intuition about when this should be allowed and what gadget powers can accomplish. We reckon that overusing this method may remove too many open conditions, break analogy between gadgets and common objects and damage believability. Its use may be domain-specific and currently very limited.

Analogical Modification of Actions

In this section we explain the technique used in methods A2 and R3 to modify actions to achieve new effects based on analogies. Traditional POP assigns variables for actions such that their effects become identical to the open condition to be established. Variables can only take objects of that type. Constraints on variable types restrict the kind of predicates that can be satisfied. To make ends meet, gadget generation allows variables to take objects of different types than the constraints specified, as long as an analogy can be established between the old type and the new type. Suppose we try to make a phone that transmit flu virus instead of voice. The variable `?voice` in the action `Transmit(?phone, ?person1, ?person2, ?voice)` is of type `Voice`. An analogy between voice and flu virus will allow the variable to take an object of type `Virus`, thereby allowing the phone to transmit it. This transformation is justified on the ground that analogous object may stand for each other in a creative domain.

We constrain the use of such transformations by requiring the action must not be performed by human. As explained before, a believable gadget should not require inconvenient actions on the user's part. For instance, a gadget should not require a person to eat a stone, even if an analogy is established between the stone and a cookie (e.g. based on their shapes and colors). In addition, when multiple transformations are performed during the generation of one gadget frame, all analogies made must be consistent. A type from a prototype frame can be considered analogous to only one type in the gadget frame.

Closing and Summarizing the Gadget

When all open conditions in the gadget frame are established, we add closure actions and summarize the frame. If a corresponding initiating action has been projected, the closure action is projected into the gadget frame with the same projection method. This may create new flaws in the gadget frame. However, since the narrative goal will be achieved before the closure actions take effect, adding closure actions is strictly necessary. If the cost to repair the new flaws becomes too high, the algorithm can choose to ignore closure actions.

The summarization generates a "use gadget" meta-event from the gadget frame to insert into the story plan. Its preconditions include all predicates from the gadget frame's initial state, and effects are accumulated from the effects of all actions in the gadget frame. Frame arguments become parameter variables of the meta-event. The usage frame is originally built for one particular narrative goal. The meta-action allows us to use the gadget in the story plan more than once and with different parameters, such as different users, to achieve different narrative goals of the same type.

Example

Our algorithm is tested against gadgets taken from a classic Japanese manga named *Doraemon*. Doraemon is a cat-like robot coming from the future to accompany a primary school student, Nobita. The repeated theme of the manga series is Doraemon helping Nobita with daily problems such as exams and bullies by using high-tech gadgets indistinguishable from magic. Dream-fulfilling gadgets that solve intractable problems are the highlights of *Doraemon* (Schilling 1993). In this section, we step through the algorithm to illustrate that our algorithm can produce a flu-transmitting phone, a gadget from *Doraemon* Volume 2 Episode 14, based on a toy phone as its prototype. To keep the description concise, we assume the refinement search works non-deterministically, i.e. always making the correct choices. In reality, it will make mistakes and backtrack.

Gadget generation is initiated by the story generator when the need to transmit flu from person A to person B arises. In symbolic form, the open conditions to be established are expressed as `infected-by(bully, virus)` and `not(infected-by(norbita, virus))`. Although it is possible to achieve both conditions without assistance of gadgets, it is not desirable in the story; waiting for flu to cure is unpleasant and may have monetary costs, and there are usually no reliable ways to infect someone with flu.

Hence, the gadget generation is initiated to fulfill the two open conditions as narrative goals. The gadget usage frame is created with no actions and the two goals as open conditions. We retrieve a toy phone from the knowledge base of known tools as the prototype based on the analogy between one of its effects `heard(p2, voice)` and `infected(bully, virus)`. The usage frame of a toy phone is shown in Figure 1.

During object retrieval, analogies are established between two predicates `heard(p2, voice)` and `infected-by(bully, virus)`, which are in turn supported by the analogy between the verb `heard` and `infected-by`, as well as the analogy between the types of corresponding frame arguments and primitives. First, `p2` and `bully` are of the same type, `People`. Second, the type of the frame argument `voice`: `Voice` and the type of the primitive `virus`: `Flu Virus` are found to be analogous. `Flu Virus` and `Voice` are analogous because they share similar attributes, such as invisible and transient, and play similar roles in the similar spatial signatures. Finally, `heard` and `infected-by` are analogous with regard to their spatial signatures, as shown in Figure 2. These analogies are kept consistent during the generation process. Frame variable `p2` is bound to the primitive `bully`.

The refinement search works backwards from the proposition that initially triggered the gadget generation: `infected-by(bully, virus)`. The action achieving a similar predicate in the phone frame is `Hear(p2, virus)`. The actor of this action is a person, so we cannot analogically transform this action. Instead, we find a similar action from the library: `Infected-By(p2, virus)` (method A3), and add it to the gadget frame. The newly added action brings in a precondition `near(p2, virus)`. We notice that an analogy can be built between `near(p2, virus)` and one of the effects of the `Transmit` action, `near(p2, voice)`, and that this analogy is consistent with existing analogies. As the actor of `Transmit` is the toy phone, an analogical transformation can reconcile the two predicates, which yields the action `Transmit(phone, p1, p2, virus)` (method A2). In other words, this transformation allows the phone to transmit flu virus based on the analogy between flu virus and voice.

We then try to satisfy three preconditions of the `Transmit` action in the gadget frame: `holding(p1, cup1)`, `holding(p2, cup2)`, and `inside(virus, cup1)`. The first two preconditions are satisfied by directly copying the two actions `Hold(p1, phone, cup1)` and `Hold(p2, phone, cup2)` from the phone frame into the gadget frame using literal projection. In the phone frame, the action `Speak-Into(p1, cup1, voice)` achieves the precondition `inside(virus, cup1)`, which is analogous to the open condition `inside(virus, cup1)` in the gadget frame. As its actor is of the type `People`, we apply method A3 instead of A2. From the action library, we find an action `Cough-Into`, given appropriate arguments, can achieve the effect `inside(virus, cup1)`. Method A3 requires an analogy between `Cough-Into` and `Speak-Into`, which is supported by their spatial signatures (shown in Figure 2), and the analogy between `Flu Virus` and `Voice`. The action `Cough-Into` is inserted into the gadget frame.

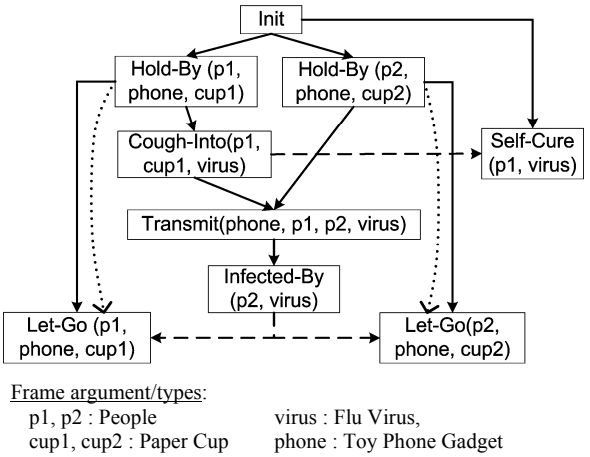

Figure 4. The usage frame of the flu-transmitting gadget phone

After that, the refinement search tries to establish the other narrative goal `not(infected-by(norbita, virus))`. No actions in the prototype frame achieves an effect analogous enough to this goal. Thus, we add the action `Self-Cure(p1, virus)` into the gadget frame (method A4). By putting the action in the gadget frame, rather than the story plan, the power of the gadget can be considered to reduce the duration of the action and render it painless. Remaining open conditions are satisfied by inserting them into the initial state of the gadget frame (methods I1, I2, and I3).

Finally, we add closure actions, which are added using the same projection method as the actions they close. As the two `Hold-By` actions are literally projected from the phone frame, we literally project the two `Let-Go` actions. The final gadget frame with frame arguments is shown in Figure 4. For clarity, some causal links and the goal state are omitted.

The usage frame is summarized into a "use-gadget" event. Predicates in the initial state of gadget frame become preconditions of the event. Some preconditions may still be impossible or undesirable to achieve in the story. Gadget generation can be initiated again to fulfill those preconditions by retrieving another prototype and projecting its elements into the old gadget frame to fulfill these open conditions. Space limitation forbids the presentation of another example that merges two prototype objects. However, the differences are minor. Two analogical constraints apply in merging multiple prototypes: First, the second prototype object should be analogous to the first because it is more intuitive to

combine analogous objects than dissimilar objects. Second, consistency of analogies requires each object type from one frame can be considered analogous to only one object type in another frame.

Discussion and Conclusions

High-tech gadgets and magical artifacts capable of the impossible appear in many stories by human writers. However, AI story generators today lack the ability to create new types of objects. We propose a significant extension to current story generation systems: the ability to create new types of objects to serve narrative purposes. Our system generates the behavior of a gadget by modifying behaviors of known objects based on a set of analogies. Our example illustrates that our algorithm, given sufficient knowledge, can generate gadgets featured in some high-quality stories produced by human.

For an artifact to be considered creative, Boden (2009) asserts it must be (a) valuable, useful or entertaining, (b) significantly different from artifacts known or created previously, and (c) not easily predicted by consumers of the artifact. Our algorithm generates gadgets that are different from any known objects and achieve narrative goals other objects cannot ordinarily achieve. We believe the process is creative. Our algorithm combines aspects of combinational and transformational creativity since it can (1) combine multiple objects and (2) transform rules of the fictional world in which the story generator searches for the best story, thereby expanding the story space that can be explored by the story generator. Thus, gadget generation enhances the creativity of story generators and can be seen as another step towards computers with human-level narrative intelligence.

References

Boden, M. A. 2009. Computer Models of Creativity. *AI Magazine* 30 (3): 23-34.

Cavazza, M., Charles, F., and Mead, S. J. 2002. Planning Characters' Behavior in Interactive Storytelling. *Journal of Visualization and Computer Animation* 13 (2): 121 - 131.

Dehn, N. 1981. Story Generation after Tale-Spin. In *Proceedings of 7th International Joint Conference on Artificial Intelligence*.

Gervás, P., Díaz-agudo, B., Peinado, F., and Hervás, R. 2005. Story plot generation based on CBR. *Knowledge-Based Systems* 18 (4-5): 235-242.

Lebowitz, M. 1984. Creating Characters in A Story-Telling Universe. *Poetics* 13:171-194.

Lebowitz, M. 1985. Story-telling as Planning and Learning *Poetics* 14:483-502.

Li, B., and Riedl, M. O. 2010. An Offline Planning Approach to Game Plotline Adaptation. In *Proceedings of 6th Conference on Artificial Intelligence for Interactive Digital Entertainment*.

Malmgren, C. D. 1991. *Worlds Apart: Narratology of Science Fiction*: Indiana University Press.

Meehan, J. 1981. TALE-SPIN. In *Inside Computer Understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Pérez y Pérez, R., and Sharples, M. 2001. MEXICA: A Computer Model of a Cognitive Account of Creative Writing. *Journal of Experimental and Theoretical Artificial Intelligence* 13:119-139.

Porteous, P., and Cavazza, M. 2009. Controlling Narrative Generation with Planning Trajectories: The Role of Constraints. In *Proceedings of 2nd International Conference on Interactive Digital Storytelling*.

Riedl, M. O., and Young, R. M. 2006. Story Planning as Exploratory Creativity: Techniques for explanding the narrative search space. *Computational Creativity* 24 (3):303-323.

Riedl, M. O., and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research* (39):217-268.

Ryan, M.-L. 1991. *Possible Worlds, Artificial Intelligence, and Narrative Theory*. Bloomington: Indiana University Press.

Saunders, R., and Gero, J. 2004. Curious Agents and Situated Design Evaluations. *AI for Engineering, Design, Analysis, and Manufacturing* 18 (2):153-161.

Schilling, M. 1993. Doraemon: Making dreams come true. *Japan Quarterly* 40 (4):405-417.

Swartjes, I. M. T., and Theune, M. 2009. Late commitment: virtual story characters that can frame their world. Technical Report TR-CTIT-09-18, Univ. of Twente, the Netherlands.

Trabasso, T., and van den Broek, P. 1985. Causal Thinking and the Representation of Narrative Events. *Journal of Memory and Language* 24 (5):612-630.

Turner, S. R. 1992. Minstrel: A Computer Model of Creativity and Storytelling: Computer Science Dept., UCLA.

Veale, T., and Keane, M. T. 1992. Conceptual Scaffolding: A spatially founded meaning representation for metaphor comprehension. *Computational Intelligence* 8 (3).

Veale, T., and Keane, M. T. 1994. Metaphor and Memory: Symbolic and Connectionist. Issues in Metaphor Comprehension. In *European Conference on Artificial Intelligence Workshop on Neural and Symbolic Integration*.

Ware, S. G., and Young, R. M. 2010. Rethinking Traditional Planning Assumptions to Facilitate Narrative Generation. In *AAAI Fall Symposium on Computational Models of Narrative*.

Weld, D. 1994. An Introduction to Least Commitment Planning. *AI Magazine* 15 (4):27-61.

Young, R. M. 1999. Notes on the Use of Plan Structures in the Creation of Interactive Plot. In *Proceedings of AAAI Fall Symposium on Narrative Intelligence*.

Zacks, J. M., Speer, N. K., and Reynolds, J. R. 2009. Segmentation in Reading and Film Comprehension. *Journal of Experimental Psychology: General* 138 (2):307-327.

Zacks, J. M., and Tversky, B. 2001. Event structure in perception and conception. *Psychological Bulletin* 127:3-21.

Zwann, R. A., Magliano, J. P., and Graesser, A. C. 1995. Dimensions of Situational Model Construction in Narrative Comprehension. *Journal of Experimental Psychology: General* 21 (2):386-397.