Creative Gadget Design in Fictions: Generalized Planning in Analogical Spaces

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

ABSTRACT
Science-fiction and fantasy stories often contain objects never envisioned previously. Inventing gadgets like lightsabers or mythical creatures like griffins is a creative task. Traditional computational storytelling systems are limited in their expressivity because they cannot create new types of objects or gadgets. The Japanese manga series *Doraemon* exemplifies the role of new and creative gadgets in creating fun and successful stories. We surveyed five volumes of *Doraemon* and identified 9 cognitive strategies of gadget creation, unified in a 5-step process. We present an algorithm to create new types of gadgets in the context of story generation. The algorithm is a combination of partial-order planning and analogical reasoning. Although *Doraemon* is our motivating example, we can also generate gadgets commonly seen in other science fictions and fairy tales.

Author Keywords
Story generation, fictional gadget, computational creativity

ACM Classification Keywords
H.4.m [Information Systems Applications]: Miscellaneous; J.5 [Arts and Humanities]: Literature

General Terms
Algorithms, Design.

INTRODUCTION

This paper attempts to investigate and simulate a major creative capability exhibited by human storytellers: the ability to create imaginary objects. Many science fictions, fantasies, and fairy tales contain imaginary objects that do not exist in our world. These objects usually have special powers, supposedly due to futuristic technologies or magic, which allow them to accomplish impossible deeds. Lightsabers in *Star Wars* and the magic mirror in *Snow White* are two famous examples. These objects contribute significantly to the fun of reading and sometimes dictate story development. The "gadget story" is proposed as one of the four subgenres of science fiction [10]. The cognitive construction of such objects, which we call *gadgets*, is a creative act that we attempt to imitate with an Artificial Intelligence (AI).

Simulating the human ability to write stories has long been an objective of AI, and some storytelling systems are considered as creative [5, 11]. However, almost all storytelling systems require a pre-specified micro-world which defines all characters, objects, places, and their inter-relationships. A few systems can modify this configuration to a limited extent [14, 19, 24]. We are not aware of storytelling systems that can create new types of objects previously unknown. In contrast, human authors' ability to perform this imaginative task is well-known, as we can see from examples as disparate as *Star Wars*, *Snow White*, and *Doraemon*.

The hugely successful, 45-volume Japanese manga *Doraemon* is often considered as a Japanese cultural icon and one of the most prominent illustrations of the human ability to imagine new objects. Doraemon is a cat-like robot coming from the future to accompany and help a primary school student, Nobita. The repeated theme of the series is that Doraemon helps Nobita to cope with problems such as exams and bullies by using high-tech gadgets indistinguishable from magic. However, in the end they usually backfire and cause unexpected consequences, emphasizing the importance of self-reliance. Dream-fulfilling gadgets that solve intractable problem are the highlights of *Doraemon* [18]. In this paper, we focus on *Doraemon* as an example of creative use of gadgets in fiction.

In an attempt to reverse engineer the creative processes of *Doraemon*'s creator, we surveyed five volumes of the manga. From an information-processing viewpoint, we identified 9 techniques that appear to be used by the *Doraemon* authors to create gadgets. These techniques are unified into a 5-step process. We then present an algorithm for generating gadgets based on our taxonomy, utilizing a combination of planning, analogical reasoning, and knowledge of everyday objects and tools to create gadgets serving narrative purposes. The algorithm extends partial-order planning to systematically explore in space of analogies. We show our algorithm can reproduce gadgets in *Doraemon*. To our best knowledge, this is the first attempt at the generation of novel gadgets as part of AI storytelling.

For an artifact to be considered creative, Boden [1] 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. Hence, we believe the process is creative. Our algorithm combines aspects of combinational and transformational creativity since it can combine multiple objects and transforms rules of the fictional world in which the story happens.

BACKGROUND AND RELATED WORK

Following cognitive research on narrative comprehension, we model a story as a sequence of events that happen in and transform a fictional world. Cognitive science reveals that readers build mental models of narratives, which capture events described by stories. It is found that people naturally segment continuous narratives such as texts and movies into discrete event structures [27]. Furthermore, people can perceive events of different granularities organized in hierarchies, where a large event can include several small events [28]. Causality and temporality between events are also important constituents of mental models of narratives, directly affecting comprehension (cf. [27, 29]). Causal relationships between events allow readers to make inferences about narratives and missing causal links can hinder comprehension [20].

A corresponding AI formalism that 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. This leads to the development of story planners [9, 12, 14, 15, 26]. Story planners require both an initial state and a goal situation to be specified as inputs before generation takes place. The initial state describes the world before the story happens, and the goal situation describes changes the events caused when the story ends. The planning algorithm generates a plan as a feasible path linking the beginning and the end of the story. Traditional story planners have limited expressivity because they have to accept both a given beginning and a given outcome.

From a narratological perspective, Ryan [16] considers comprehension of stories as reconstruction of fictional worlds. Ryan makes two observations. First, readers learn about the fictional world bit by bit throughout the story, rather than everything at the beginning. Second, readers generally assume aspects not mentioned in the story to depart minimally from our world.

One implication of readers learning bit by bit throughout the story is that they are usually happy to believe that facts learned later existed all along. For example, when the story describes a lightsaber for the first time, readers can easily accept that this apparently novel technology existed in the fictional world the entire time. This implies that, with proper justification, we can introduce facts and novel objects later in the story without complete loss of believability. This observation motivates story planners that dynamically assert facts in the story world. Riedl and Young [14] proposed a story generation technique that starts out with a pre-defined micro-world, but allows some of the details of the world to be retroactively modified to suite the narrative arc. From a narratological perspective, their system can be considered to start from a set of possible worlds and later settle on one where the best story can be developed. The idea is later generalized [19, 24]. Our work on gadget generation complements these techniques for world-altering during storytelling. The world-altering story planning techniques described above can adjust relationships between objects and attributes of objects, and instantiate new objects of known types. These techniques, however, cannot create new types of objects. The work presented in this paper directly addresses the problem of creating an object of previously unknown type.

The second of Ryan's observations, the theory of minimal departure, suggests that readers re-use existing knowledge to interpret newly encountered fictional worlds. For example, we are willing to assume most characters in movies and novels pay at restaurants, even when such details are often omitted. In AI terminology, when the reader learns that a person eats at a restaurant , they bring into the fictional world the knowledge frame of restaurant visits, which includes paying. Based on this observation, we believe the connection between a common object that people are familiar with and the unfamiliar gadget is crucial for the understanding of a gadget. When the gadget is analogous to a common object, old knowledge can shed light on the new, which allows the new gadget to be easily understood and accepted. For instance, since we understand a lightsaber is analogous to a sword, we can accept that a lightsaber deflects upon hitting another lightsaber blade, even though physics indicates two beams of laser would actually pass through each other. Take another example from the *Doraemon* manga, a piece of toast bestowing the power of memorization on its user (Volume 2 Story 1, abbreviated as D2.1 thereafter). Although the analogy between the gadget and a toast cannot completely explain how to use the gadget, it provides some hints; readers can easily understand using the toast gadget involves eating it. We believe that for imaginary gadgets, such analogies are crucial for its comprehension and intuitive appeal. A good analogy can enhance the intuitive appeal of the gadget. Gadget generation in this paper focuses on finding the correct common object and modifies it to make a gadget while preserving the analogy between the two.

To reason about analogies between the common object and the gadget, we draw from the research on analogical reasoning. Several AI programs, such as SME [2], ACME [8], and Sapper [22, 23] attempted to simulate the human ability to recognize analogies. An analogy maps between two conceptual spaces including entities with attributes and interrelationships. SME distinguishes between the mapping between surface properties (such as color and size) and relations (such as causality or inequality). Mappings of only relations are considered as true analogies, whereas mappings of only surface properties are superficial and mappings of both are literal similarities. Nonetheless, other researchers [8, 22, 23] utilize both properties and relations in analogical mapping. Since most mappings are not perfect, we consider both surface and relational features contribute to intuitive appeal of analogies. Most relevant to our work, Sapper utilizes both types of features, and allows analogies to be built incrementally via the incremental rule. This well fits into the refinement search paradigm of partial-order planning.

STRATEGIES OF NOVEL GADGET DESIGN

As fictional gadgets exist only in the imagination of the author and the reader, it is often impossible to provide a detailed scientific account for their mechanisms. No one ever read HAL's code or understood how a time machine works, but it does not make them less happy reading those stories. This differentiates design of fictional gadgets from design in the real world, where a working mechanism must be designed for any product. Computational systems that automate the design task have been investigated (cf. [6]). The most similar to our work are systems that utilize analogy in design, such as [13] and [7].

For a gadget to be accepted by readers, it must be comprehensible. In our opinion, to comprehend a gadget is to be able to recognize it and to make predictions about it; the understanding consists of its appearance and behavior. We consider anyone who knows how to use a phone to transmit voice as "understanding" a phone, even though they may not understand the underlying physics. Hence, we propose that to generate a gadget is to generate a reasonable behavior for it and its corresponding appearance. The appearance should intuitively match the behavior. For example, a gadget that can fly is likely to have wings or propellers. However, the coupling between the appearance and the behavior is not the focus of this paper.

Across all 45 volumes, it is estimated that *Doraemon* contains about a thousand gadgets [18]. We closely analyzed the first five volumes of *Doraemon* for a total of 87 stories and attempted to deconstruct the process of gadget creation. Consequently, we identified 9 concrete strategies of gadget design. Several exclusion criteria are applied in our study. We disregard gadgets that interact with time (e.g. time machines) since such interactions may result in logical inconsistencies problematic for AI systems. We ignore stories where multiple gadgets are used in combination and none is described well enough for our analysis. We further exclude gadgets creating scientific simulations and virtual reality (e.g. the virtual reality skiing field in D2.15), or directly taken from fairytales and obviously existing ideas, such as Aladdin's lamp (D1.15) and Cupid's bow and arrows (D3.12). After these exclusions, we identified 60 gadgets from the five volumes, and found that the 9 strategies proposed here can explain 55 or 91.6% of these gadgets.

Next, we describe specific strategies with examples from the manga. However, as with many cognitive phenomena, gadget creation is a fluid and organic process. Strategies may have fuzzy boundaries and can be used in combination. The main aim here is to illustrate different techniques rather than providing a strict taxonomy.

Strategy I: Default Form Factors

The simplest strategy is to use default form factors for certain types of functions. Robots (D2.2), hand puppets (D3.9) and dolls (D2.6) are used when the gadget is supposed to exhibit human behaviors, for example, to love our protagonist, to be a life guide, or to tell hidden desires. Gadgets that alter mental states, such as temperament (D5.1), are sometimes portrayed as pills. When other strategies fail to generate gadgets, a default form like a micro-computer may be used. Examples include gadgets that perform facial plastic surgeries (D4.7) and make dim sum (D2.13). The appearances and behaviors of the gadgets do not deviate much from the original objects. Robots can be turned on or off, and pills are directly swallowed.

Strategy II: Symbolized Abilities

Some characters are exemplars of their unique abilities. Sherlock Holmes's ability to reason and Superman's ability to fly are very well known. In addition, these characters are also typically associated with and identified by some personal items, such as Holmes's cap and pipe and Superman's red and blue suit. Hence, a conceptual slippage (cf. [23]) may associate iconic personal items with special abilities. Such associations are employed to create gadgets. D3.7 features a Superman Costume Set that allows the wearer to fly at a low altitude. D3.4 features a Holmes Mystery Solver Set, including a cap, a pipe, a magnifying glass and a walking stick, which helps the user to explain seemingly mysterious events and find culprits. D5.15 features a black belt, a symbol of Judo masters. The wearer of the belt gets the ability to throw away anyone touching her. This strategy relies crucially on finding the conceptual association between the personal item, its owner, and a special ability.

Strategy III: Typical Components of Typical Scenes

Sometimes a particular ability can be conceptually associated with a typical scene. For example, ghost stories often contain an image of flickering lights, or will-o'-the-wisps in darkness. Thus, a gadget that makes ghost story come true (D2.3) can adopt the form of a lamp that looks like a will-o'-the-wisp. As another example, a garden-under-moonlight scene may involve fragrant flowers, singing crickets and bright moonlight. Not surprisingly, a gadget that simulates the sound of crickets appears as a flower (D4.6). This strategy seems to work by first imagining a typical scene associated with the function, and extract a component from the typical scene.

Strategy IV: Decoraters and Decoratees

Another association employed by the author of *Doraemon* is between an object and the one it decorates. Examples include personal accessories or clothes and corresponding body parts. In the Holmes Mystery Solver Set (D3.4), the cap enhances the ability to reason since the cap sits upon the head where reasoning happens. D1.10 features a lipstick that grants its wearer the ability to say extremely pleasant compliments. A slightly different example is Gravity Paint (D5.2). Walls painted with it generate gravity and work like floors, i.e. allowing people to stay on them without falling. This strategy requires little additional transformation after the initial prototype is retrieved. The intuitive appeal of gadgets generated relies on the "decorating" relationship.

Strategy V: Reversed Use of Models

When a model represents a certain aspect of the real world, modifying the model can be imagined to affect the real world. The practice of inserting needles into voodoo dolls suggests a long historic trace behind this thought. D4.1 features a camera producing dolls very much the same as voodoo dolls. D3.2 contains a wrist watch that, when tweaked, changes the actual date. D3.11 features a camera that reverses the ordinary picture taking procedure. You can change someone's dress by putting a picture of the desired dress in the camera and pressing the shutter towards her. When the camera is not loaded with a picture, however, all her clothes will disappear.

A similar example of causality reversal appears in D4.17. The gadget is a revolver used to play Russian roulette. In the original Russian roulette, one needs good luck to survive. In the story, the revolver contains bullets of good and bad luck. She who takes a bullet will become extremely luck or unlucky for a short period.

Strategy VI: Substituting Operands

Strategy VI allows a known tool to operate on things it cannot ordinarily operate on. The flu-transmitting phone (D2.14) transmits flu instead of voice. The friend remote control (D4.9) controls people, not home appliances. D1.9 features a pair of scissors that can cut one's shadow off, which can then work as a slave. Other than the operand, behaviors of these gadgets (e.g. cutting, transmitting, controlling) usually do not change significantly from the prototype object. It seems to us that the new operand should bear resemblance to the old, which would justify the substitution. Both flu virus and voice are invisible to the naked eye and can spread in space. A shadow may look as thin as paper.

Strategy VII: Combining Multiple Tools

From flying cars to lightsabers, combining different things is a great source of inspiration for gadgets. Many mythical creatures are also combinations of common animals. The Pegasus is a horse combined with wings of a bird. A griffin is an eagle plus a lion. The memory toast (D2.1) is an exemplar from the *Doraemon* manga. The gadget helps people to memorize things on a book page. First, put a memory toast on that page, and the printings will be transferred from the page onto the toast. After that, eat the toast, and you will have precise memory of anything on that page. This gadget seems to be a combination of silly putty, which provides the printing transfer, and a toast, which provides the edibility. The alarm clock in D3.5 is a robot whose body and head are replaced by a clock.

Strategy VIII: Relaxing Preconditions or Constraints

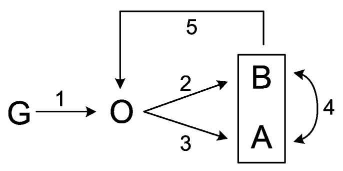
Tools in the real world have constraints we may wish to remove or relax. A telescope cannot see through a wall. A pencil cannot write by itself but needs someone to write with it. This strategy modifies or removes these constraints from the original object. If we remove the precondition that two rooms connected by a door must be adjacent in space, we create one of the most well-known gadgets in *Doraemon*: the Anywhere Door, walking across which can bring the user anywhere within a 10-lightyear radius. The smart pencil (D1.11) writes right answers to exam questions by itself. A mailbox in D2.17 predicts the reply to your letter before it is sent, thereby removing a constraint of time. Telescopes can see through walls in D4.11 and into the future in D4.10.

Strategy IX: Extending or Reversing Effects

Sometimes a gadget looks like an ordinary object and is also used in a similar manner, but it can create extraordinary effects. Though the effects are extraordinary, they are usually not arbitrary. In *Doraemon*, effects of ordinary objects are often replaced by something similar to the original effects but better. Body cream is usually used to alleviate the harsh feeling of winter. D1.14 enhances this ability to create a cream gadget that reverses the feeling of temperature. When one wears the cream, winter feels like summer. Ringing a bell is often a signal for gathering. The bell in D2.4 can summon people to play with you while in dreams. Sometimes the new effects are reversals of the old. A mirror tells the truth in Snow White, but in D2.8 it tells lies.

The Generalized Process

A general gadget generation process unifying these strategies is shown in Figure 1, with each step numbered. Real-world design usually starts with a desired function and seeks a structure to realize it [4]. Similarly, in the first step our process starts with a given function, and retrieves an object from all known objects as a gadget prototype. Steps 2 and 3 generate a behavior and an appearance for the gadget respectively. Our approach to generating the behavior and appearance of the gadget is to reuse corresponding components of the prototype object directly and/or adapt aspects of the prototype to suit the function. The fourth step indicates mutual influences between the usage and appearance, reflecting the fact that the gadget can be iteratively refined. The fifth step indicates that the appearance or behavior may prompt a retrieval of additional prototype objects. For example, if during the generation of the behavior we realize the gadget should be able to fly, we can decide to retrieve an airplane and transplant its wings onto our gadget. Steps 1-3 are major



F = desired function O = prototype object
A = appearance of gadget B = behavior of gadget
→ = transformation / derivation

Figure 1. The process for gadget generation.

steps, which are performed each time a gadget is produced. In contrast, steps 4-5 are "maintenance" steps, reflecting that gadget generation often can go back-and-forth or circuitously rather than always straightforward. In summary, gadget generation is the generation of an appearance and a behavior for the gadget, mediated by one or more analogies with known objects and tools.

Gabora [3] notes that creative problem solving often involves alternation between an associative phase, where possible or similar solutions are retrieved, and an analytic phase, where products of the first phase is amended to meet quality and realistic constraints. Here, the prototype retrieval step corresponds to the associative phase. The subsequent goal-driven analytic phase attempts to fill any causal gaps and minimize unnatural human behavior during interaction with the gadget. If any problems arise during the analytic phase, the associate phase can be restarted.

The gadget generation process generalizes strategies I-IX presented before. Strategy I makes use of default prototype objects, which supply default appearance and behavior for certain functions. Strategies II-V focus on the association between the desired function and the prototype object and often perform little subsequent adaptation. These strategies utilize focused retrieval. Strategy IV, for instance, retrieves only those prototype objects that decorate another object or organ related to the function. Strategies VI-IX perform substantial adaptation after the retrieval and may be applicable to a wider type of prototype objects. Strategy V, for example, modifies the prototype object by reversing causes and effects.

The gadget generation process includes different tasks, each relying on different types of computation and knowledge. The retrieval of prototype objects relies crucially on efficient memory organization. Once an object is retrieved for strategies II-V, little to no subsequent transformation is needed. Strategies VI-IX require careful analysis and transformation of a series of causes and effects in the prototype to produce a coherent description of how the gadgets interact with the world. Generating the appearance of gadgets is different from the previous two tasks and concerns the relationship between behaviors and structures that enable them. The generation of visual appearance of novel gadgets is left for future work.

In this paper, we focus on the functional aspect of gadgets and implement mainly Step 2, with some consideration for Step 1. In the next section, we propose an algorithm that first retrieves a prototype object and then transforms it to produce gadgets similar to those generated by Strategies VI and VII. Strategies VI is supported by the ability to transform known actions analogically to create new action. Strategy VII is supported by iterative merging of multiple prototype frames. Analogical transformation can also handle special cases of Strategy IX when the new effect is analogous, but not opposite, to the old effect. Efficient memory organization and appearance generation are left for future work. Although the efficiency of the algorithm can be negatively affected without efficient retrievals of known objects, we believe in a divide-and-conquer approach and currently use a simple retrieval method as a surrogate.

GADGET STORY GENERATION

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

Following Young [26], there has been a growing trend of modeling a story as a partial-order plan (e.g. [9, 12, 14, 15]), where generating a story is equivalent to solving a planning problem with aesthetic constraints. In our work gadget generation is a computational process that augments story planning by automatically producing a new type of device that can be incorporated into the story. The story planner provides a narrative goal and the gadget generation process produces a plausible gadget that, when used, changes the world to achieve the goal. In the next sections, we provide a general background on AI planning and then describe our representation for gadgets and the algorithm for producing them using a combination of planning and analogical mapping.

AI Planning Background

Planning produces a sound plan, or a sequence of actions that guarantees to achieve a goal situation from an initial world. The goal situation and the initial world are two sets of first-order logic expressions or predicates. An action has preconditions and effects, also expressed as predicates. Before an action can occur, its preconditions must be true. After the action occurred, its effects become true. A sound plan properly links a series of actions so that the goal conditions are achieved by effects of some actions – whose preconditions are in turn achieved by earlier actions – and preconditions of the earliest actions are in the initial state. Actions take objects and/or characters as parameters. Objects belong to hierarchically organized types. For example, the object `my-car` belongs to the type `Car`, which is a subtype of `Vehicle`.

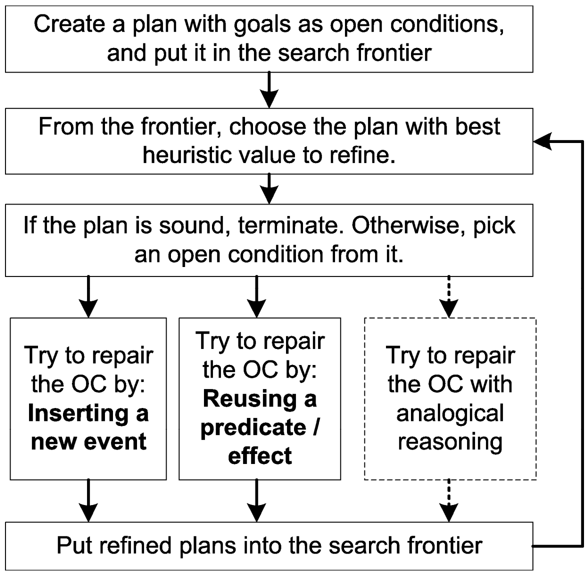
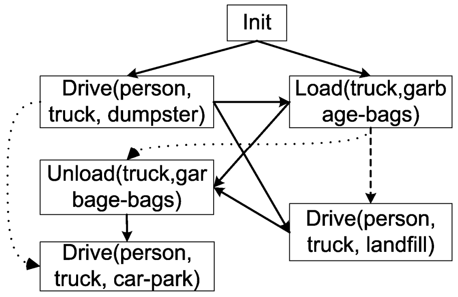

Create a plan with goals as open conditions, and put it in the search frontier
From the frontier, choose the plan with best heuristic value to refine.
If the plan is sound, terminate. Otherwise, pick an open condition from it.
Try to repair the OC by: **Inserting a new event**
Try to repair the OC by: **Reusing a predicate / effect**
Try to repair the OC with analogical reasoning
Put refined plans into the search frontier

Figure 2. A brief outline of partial-order planning.

Partial-order planning (POP) is primarily concerned with selecting correct actions from an action library, giving them correct parameters and inserting them into the plan to achieve open conditions. An open condition (OC) is a precondition of an action or a goal predicate not yet achieved. An OC can be achieved (or in POP terms, *repaired*) with an identical effect from an action. The action may be an existent action in the plan or newly inserted into the plan for the purpose. In addition, an OC can be achieved with an identical predicate in the initial state. A repair operation produces a refined plan, where a matching predicate (either from the initial state or an effect) is linked to the OC with a *causal link*. Note that if an action is inserted during the repair, it may bring in new preconditions which become new OCs. When there are multiple possible ways to repair an OC (e.g., different actions with the same effect), each will yield a refined plan, which are put into the search frontier. A heuristic function estimates the quality of a plan and number of further refinements it needs to reach a sound plan. From the search frontier, the plan with the best heuristic value is selected as the candidate for the next iteration of OC repairs. The process continues until a sound plan containing no open conditions is found on the frontier. See Weld [25] for more details of partial-order planning, such as causal threats. The POP algorithm is outlined in Figure 2 as a flowchart. In gadget generation, we extend POP by employing analogical reasoning to repair OCs, as indicated by the dotted box in Figure 2.

Usage Frames

We represent the gadget's behavior as a plan describing how the gadget interacts with its user and the world during a typical use. This plan is called a *usage frame*. It portrays a typical scenario of the gadget being used, including actions that typically happen right before and after its use. In the story plan, a usage frame is summarized as a single meta-


Init
Drive(person, truck, dumpster)
Load(truck,garbage-bags)
Unload(truck,garbage-bags)
Drive(person, truck, landfill)
Drive(person, truck, car-park)

Frame-level variable/type:	person/Person,
truck/Truck	garbage-bags/Bagged-Trash
landfill/Landfill	car-park/ Land-Location
dumpster/Land-Location	

Figure 3. The usage frame of a garbage truck.

action, forming an action hierarchy. Such a hierarchy supports flexible description of gadgets in different media.

As an example, Figure 3 shows the usage frame for a garbage truck. The truck is first driven to the dumpster to collect bagged trash, and then to the landfill to unload it, and finally returned to the car park. Solid arrows denote causal links. Dashed arrows denote temporal links, which indicate orderings of actions. Dotted arrows in the frame denote *closure actions*, which restore the world to a normal or routine state after other actions change it. Closure actions are not necessary for a gadget's intended purpose, but they complete the frame and may improve story coherence. Here, the action where the truck unloads garbage is a closure action, which "closes" the actions where garbage is loaded so that the truck is usable again.

Usage frames deviate slightly from the general plan data structure in that we use a set of variables, called *frame variables,* instead of objects. All actions in usage frame take frame variables as parameters. This allows us to identify parameters of different actions to *co-resolve* (i.e. bind to the same object) without committing until the gadget is used in the story. Values of frame variables depend on how the gadget is used in the story. We also find it convenient to distinguish between actions performed by a human and those performed by machines, tools, or natural occurrences.

Computing Analogies

An key aspect of gadget's believability and appeal is the resemblance between the gadget's usage frame and an usage frame of an ordinary object. Analogy is critical in both retrieving the ordinary object and in subsequent transformation. We employ Sapper [22, 23] as the analogy making engine. Representing knowledge in a semantic network, Sapper lays dormant bridges between concepts that share enough properties or participate in same relations. When we determine if two concepts are analogous, dormant bridges may be activated to support the analogies. We can make analogies between object types, predicates, and actions, but not across categories. All known object types, predicates and actions are stored in the semantic network. Objects types are considered analogous if they share properties or are involved in relations of the same type. Analogies between predicates and actions are supported by analogies or matches between corresponding parameters. Semantic roles, such as subject, object, etc., of each parameter in predicates and actions are annotated to facilitate mapping. Furthermore, we utilize the notion of *spatial signatures* [21] to capture similarities between predicates and actions. Spatial signatures capture the embodied understanding of verbs as movement patterns, which can reveal hidden connections between actions. For instance, climbing a staircase and a rise in social status both imply upward movements. Thus, a metaphor can be created between them. Two predicates or actions with matching spatial signatures and analogous corresponding parameters will be considered highly analogous. All these comparisons are incorporated in Sapper's activation spreading process. The basic idea in transformation is that if two object types, predicates, or actions are analogous enough then they can stand in place for one another in a creative domain.

Initiating Gadget Generation

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 an improbable goal, such as winning a lottery. Admittedly, a story planner can create coincidences without considering probabilities. However, two many coincidences can damage the believability of a story. A gadget which makes the improbable happen can believably justify an unlikely outcome and rescue the story.

After the gadget usage frame is completed, it is summarized into a single “use gadget” meta-action and put in the story. At this time, frame variables will be assigned to objects and characters in the story in a way that is consistent with the usage of the gadget and the larger story context. If a new narrative goal needs to be achieved by gadgets when one gadget already exists in the story, the existing gadget can be adapted to satisfy a new narrative goal, or a new gadget can be generated. As a special case, when a precondition of a "use gadget" meta-action initiates gadget generation, an additional prototype will be retrieved to merge with the existing gadget. This paper omits details and assumes a story generation system capable of selecting the open conditions to be achieved by gadgets.

Retrieving a Prototype

Prototypes are *a priori* known object types from our ontological hierarchy. Prototypes become the basis from which we create new gadgets. Usage frames of these objects are 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 known tools that achieve a predicate analogous to p. Saunders and Gero [17] 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 whose effect is optimally moderately analogous to p is first attempted as the prototype for the new gadget. The algorithm may backtrack and try a different tool.

Constructing the Gadget Frame

To construct a usage frame for a new gadget, we extend POP with new methods, informed by the usage frame of the retrieved prototype object, to achieve open conditions. The gadget usage frame starts as an empty usage frame – an empty plan – with an empty initial state and the narrative goal p being the only open condition. Gadget generation incrementally repairs open conditions by trying each of the methods, putting the resulted usage frames into the search frontier, and starting the next round of repairs with the usage frame having the best heuristic value. In the next few sections, we introduce the newly proposed methods to repair open conditions during gadget generation.

Analogical reasoning and partial-order planning are unified in the idea called projection. Projection provides a tactic to achieve open conditions by copying an element from the prototype usage frame – either an action or a predicate in the initial state – and inserting it into the new gadget usage frame either literally or through analogical transformations. A literal projection simply copies an element over. An analogous projection transforms the projected element based on analogies between the two frames before copying it over. 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. Table 1 lists all projection methods alongside traditional methods gadget generation takes from POP [25] and the Initial State Revision story planner [14].

Projection allows the gadget usage frame to imitate the prototype usage frame and achieves its own goals at the same time. Given an open condition in the gadget frame, we first attempt to find the same condition or an analogous condition in the prototype frame. If such a condition is found, the gadget frame tries to imitate the way the prototype frame achieves the original open condition. As mentioned previously, the condition could have been achieved by an action's effect or a predicate in the initial state. In the former case, we attempt to project the action. In the latter, we attempt to project the predicate into the initial state of the gadget frame. In both cases, literal projections are attempted before analogous projections as literal projections provide closer imitation of the prototype.

	Projection		Traditional
	Literal	**Analogous**	
Insert an New Action	Copy an action from prototype frame to gadget frame	A. Analogically transform an action from the prototype frame B. Select an analogous action from the action library	Insert an action directly (same as POP [25])
Modifying the Initial State	Copy an predicate from prototype frame to gadget frame	Analogically transform a predicate from the prototype frame	Insert a predicate to gadget frame directly (similar to ISR [14])

Table 1. Traditional and project methods that insert actions and modify the initial state.

Projecting and Inserting Actions

In order to achieve an open condition *c*, a literal projection copies an action with the effect *c* from the prototype frame into the gadget frame. However, often we cannot find such an action in the prototype frame, and we will attempt an analogous projection. Analogous projection empowers partial-order planning to systematically explore in the space of analogies.

The first method of analogous projection is to transform an action in the prototype frame based on analogies in order to produce a new action that achieves *c*. We call this analogical transformation. As mentioned earlier, an action takes parameters of predefined types. Take, for example, an action from a telephone usage frame: `Transmit(voice?, person1?, person2?)`, which transmits voice between two people. A question mark denotes a parameter. Paratermized actions can be applied in different situations (e.g. transmitting voices between different people). This action has one effect `close-by(voice?, person2?)`. `voice?` is of type `Voice` and `person1?` and `person2?` are of the type `Person`. Suppose the stories requires us to create a gadget that transmits flu viruses, and one open condition in the gadget frame is `close-by(virus?, person2?)` where `virus?` is of type `Virus`. Normally, the transmit action cannot take parameters of the type `Virus`. Analogical transformation creates a new action that can do so based on the analogy between flu viruses and voice. Hence, after the transformation we are able to achieve the desired open condition. In order to keep intuitive appeal of the gadget and prevent nonsensical actions, analogical transformation is only allowed when the actor of the action is not human. Even though a stone may look like a cookie, a person cannot eat the stone like a cookie. On the other hand, it is reasonable if a high-tech or magical gadget interacts with this stone as if it is a cookie. The heuristic value, or the desirability of an analogical transformation is positively correlated to the analogies used during the transformation.

If analogical transformation is not applicable due to our restriction on actors, we can apply the third analogous projection. Suppose an action A_p in the gadget frame has an effect c_p, which is analogous to our open condition *c*. To keep the resemblance between the gadget and the prototype, we look for another action A_g from the action library that is analogous to A_p and has the effect *c*.

If any of the above three projection methods succeeds, an action with the effect *c* will be inserted into the gadget frame, achieving the desired open condition. However, if none is applicable, we can still use the conventional POP method that inserts any action from the action library that achieves *c* into the gadget frame. This final method does not imitate the prototype and has lower preference.

Projecting and Inserting Predicates into Initial States

An open condition *c* in the gadget frame can also be achieved by a predicate in the initial state. Similar to actions, we first try to perform a literal projection, i.e. copy the same predicate *c* which exists in the initial state of the prototype frame directly into the gadget frame. If such a predicate does not exist, we then try to perform an analogous projection. If a predicate c_p in the prototype frame is analogous to *c*, we can insert c into the gadget frame. This is possible because we consider *c* as resulted from a analogical transformation of c_p. Finally, we may directly insert *c* into the initial state of the gadget frame directly. This is similar to the functionality in the Initial State Revision story planner [14]. In fact, all three methods produce the same refined usage frame where *c* is inserted into the initial state. However, frames produced by the three methods have decreasing heuristic values because they differ in degree of imitation. The first method preserves the most of the prototype frame, while the last method does not imitate the prototype frame at all.

Other Methods to Repair Open Conditions

Besides the projection methods introduced above, our algorithm is also capable for reusing effects of actions and predicates in the initial state, just like traditional POP. In addition, we may also assume an open condition is resolved by the "power of the gadget". 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 hurt believability. Its use may be domain-specific. Currently, we only use it to remove any knowledge requirement of gadgets because gadgets in *Doraemon* rarely need complex skills or knowledge.

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 optional. If the cost becomes too high, the algorithm can choose to ignore closure actions.

The summarization generates a "use gadget" meta-action 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-action. The meta-action can then be used to achieve narrative goals of the same type as the usage frame.

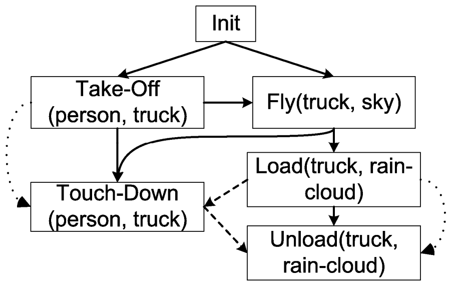
EXAMPLES

Our algorithm can generate some highly complex gadgets, including some from the *Doraemon* manga, such as a truck collecting rain clouds and the flu-transmitting phone (D2.14). Due to limited space, we show how these gadgets can be created schematically with major decisions during their generation.

The first example is a truck that collects rain cloud to stop the rain. The initiating narrative goal in the story is `not(at(rain-cloud, sky))`. The system compares this condition with all known tools, and finds that a garbage truck can achieve an analogous effect: `not(at(garbage-bags, dumpster))`. The usage frame of garbage truck (shown in Figure 3) is retrieved as the prototype. The analogies between rain cloud and bagged garbage and between sky and dumpster are made at this time. Based on the analogies, we analogously project the action `Load(truck, garbage-bags)` from the truck frame to create a new action `Load(truck, rain-cloud)`.The algorithm choose not to project `Drive(person, truck, dumpster)` to the gadget frame. Although `dumpster` is considered analogous to `sky`, the analogy is rather stretched, so the analogical transformation yields a low heuristic value and we prefer not to project. After we repair remaining open conditions of this action by revising the initial state, gadget planning completes.

Later, the story generator realizes that it cannot find a way to deliver the cloud-collecting truck to the sky where the clouds are – that is, `at(truck, sky)` cannot be achieved. The cloud-collecting truck is blended with the prototype frame of an airplane to give it the ability to fly. The final gadget frame is in Figure 4, showing the result of the double-blend.

In the next example, a flu-phone gadget is built to achieve the narrative goal `infected-by(bob, flu)`. In words, the flu is transferred from one person to another via phone. A



Frame-level variable/type: person/Person, truck/Truck-Gadget, sky/Air-Location, rain-cloud/Rain-Cloud

Figure 4. The final usage frame of the flying cloud-collecting truck.

telephone frame is retrieved based on the analogy between the flu virus and voice, and the analogy between understanding voices and being infected by viruses. These analogies allow us to modify effects of the main transmission action of the telephone frame. The action `Speak(person?, voice?)` is projected from the telephone prototype frame to `Cough(person?, virus?))`. As the actor of `Speak` is a person, analogical transformation is not applicable. This projection is done by finding an analogous action from the action library.

CONCLUSIONS AND FUTURE WORK

Based on our survey of the *Doraemon* manga, we have identified a general process for generating novel gadgets in fictions, which unifies nine different strategies. The process first retrieves one or more known objects, then adapts and combines them to generate a typical behavior for the gadget. After that, an appearance is generated to match the behavior. In this paper, we implement mainly two strategies of gadget generation: Strategies VI and VII, with special cases of Strategy IX. We present an extended planning algorithm that generates a usage frame of the gadget. The algorithm is a planning process using analogically generalized actions and is informed by other known plans. Analogically generalized actions expand the space of plans that can be generated. Known plans, when used as guidance, focus planning on the parts of the expanded space that are more reasonable and intuitively appealing. Our algorithm can generate complex gadgets, including some from *Doraemon* and other science fictions as well as mythical creatures. Future work will address performance issues of the algorithm and other steps and strategies in the gadget generation process.

Our algorithm can be considered as creative from multiple perspectives. We generate an unknown gadget, which serves narrative purposes, satisfying the novel and useful criteria [1] and the notion of c-creativity [11]. As the need arises in the story, gadgets are created on the fly and expand the spaces of stories that can be generated. Boden [1] classifies this as transformational creativity. Combining multiple known objects to create a new gadget fits the category of combinational creativity. Gero [4] defines innovative design as assigning variables with values outside their typical ranges and creative design as introducing new variables. Binding variables analogously may be considered as innovative, whereas a combination of two objects takes variables from both, and is hence creative design.

Our work suggests that a goal-driven analogy making process is a viable approach for computational creativity. Fictional gadgets are vivid illustrations of the importance of human imagination in writing stories. Any progress in the field of computational storytelling will require advances in computational creativity, of which the algorithm in this paper can be considered one such example.

ACKNOWLEDGMENT
This paper is based upon work supported by the National Science Foundation under Grant No. IIS-1002748.

REFERENCES

1. Boden, M. A. Computer Models of Creativity. *AI Magazine*, 30, 3 (2009), 23-34.
2. Falkenhainer, B., Forbus, K. D. and Gentner, D. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41, 1 (1989), 1-63.
3. Gabora, L. Revenge of the 'neurds': Characterizing creative thought in terms of the structure and dynamics of memory. *Creativity Research Journal*, 22, 1 (2010), 1-13.
4. Gero, J. S. Design Prototypes: A knowledge representation schema for design. *AI Magazine*, 11, 4 (1990), 26-36.
5. Gervás, P. Computational approaches to storytelling and creativity. *AI Magazine*, 30, 3 (2009), 49-62.
6. Goel, A. K. Design, analogy, and creativity. *IEEE Expert*, 12, 3 (1997), 62-70.
7. Helms, M., Vattam, S. and Goel, A. Compound analogies, or how to make a surfboard disappear. Proc. CogSci 2008 (2008).
8. Holyoak, K. J. and Thagard, P. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13 (1989).
9. Li, B. and Riedl, M. O. An offline planning approach to game plotline adaptation. *Proc. AIIDE'10* (2010).
10. Malmgren, C. D. Worlds Apart: Narratology of Science Fiction. Indiana University Press, 1991.
11. Pérez y Pérez, R. and Sharples, M. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems*, 17, 1 (2004), 15-29.
12. Porteous, P. and Cavazza, M. Controlling narrative generation with planning trajectories: The role of constraints. *Proc. ICIDS'09* (2009).
13. Qian, L. and Gero, J. S. A design support system using analogy. *Proc. AID '92* (1992), 795-813.
14. Riedl, M. O. and Young, R. M. Story planning as exploratory creativity: Techniques for explanding the narrative search space. *Computational Creativity*, 24, 3 (2006), 303-323.
15. Riedl, M. O. and Young, R. M. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39 (2010), 217-268.
16. Ryan, M.-L. *Possible worlds, artificial intelligence, and narrative theory*. Indiana University Press, 1991.
17. Saunders, R. and Gero, J. Curious agents and situated design evaluations. *AI for Engineering, Design, Analysis, and Manufacturing*, 18, 2 (2004), 153-161.
18. Schilling, M. Doraemon: Making dreams come true. *Japan Quarterly*, 40, 4 (1993), 405-417.
19. Swartjes, I. M. T. and Theune, M. *Late commitment: virtual story characters that can frame their world.* Technical Report TR-CTIT-09-18, University of Twente, Enschede, Netherlands, 2009.
20. Trabasso, T. and van den Broek, P. Causal Thinking and the Representation of Narrative Events. *Journal of Memory and Language*, 24, 5 (1985), 612-630.
21. Veale, T. and Keane, M. T. Conceptual Scaffolding: A spatially founded meaning representation for metaphor comprehension. *Computational Intelligence*, 8, 3 (1992).
22. Veale, T. and Keane, M. T. Metaphor and memory: Symbolic and connectionist. issues in metaphor comprehension. *Proc. ECAI 1994 Workshop on Neural and Symbolic Integration* (1994).
23. Veale, T. and O'Donoghue, D. Computation and Blending, *Cognitive Linguistics*, 11, 3/4 (2000), 253-281.
24. Ware, S. G. and Young, R. M. Rethinking traditional planning assumptions to facilitate narrative generation. *Proc. INT3* (2010).
25. Weld, D. An Introduction to least commitment planning. *AI Magazine*, 15, 4 (1994), 27-61.
26. Young, R. M. Notes on the use of plan structures in the creation of interactive plot. *Proc. INT1* (1999).
27. Zacks, J. M., Speer, N. K. and Reynolds, J. R. Segmentation in reading and film comprehension. *Journal of Experimental Psychology: General*, 138, 2 (2009), 307-327.
28. Zacks, J. M. and Tversky, B. Event structure in perception and conception. *Psychological Bulletin*, 127 (2001), 3-21.
29. Zwann, R. A., Magliano, J. P. and Graesser, A. C. Dimensions of situational model construction in narrative comprehension. *Journal of Experimental Psychology: General*, 21, 2 (1995), 386-397.