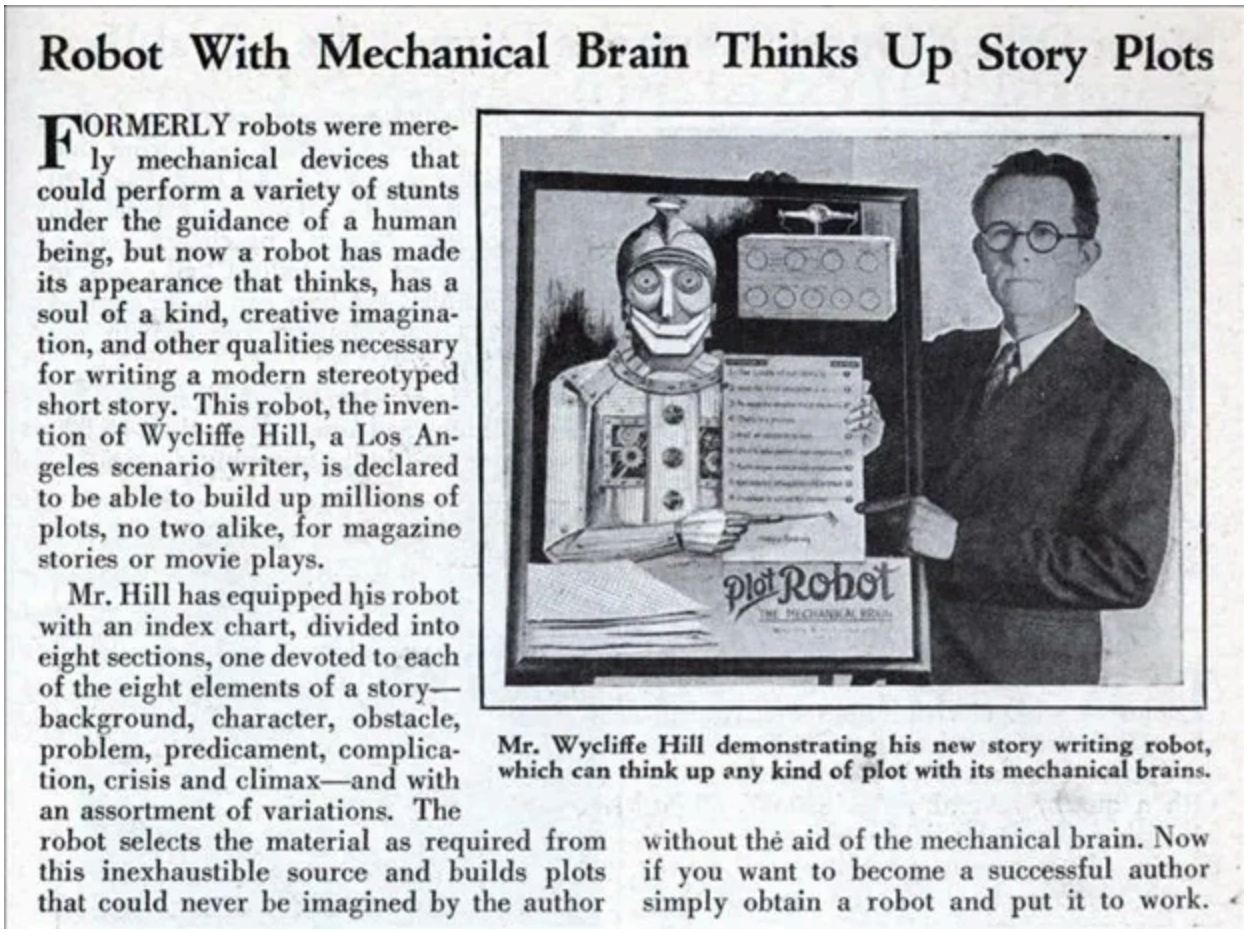
An Introduction to AI Story Generation

Mark Riedl Follow 26 min read · Jan 4, 2021

Contributing authors (alphabetical): Amal Alabdulkarim, Louis Castricato, Siyan Li, and Xiangyu Peng.

Updated: June 28, 2021



Robot With Mechanical Brain Thinks Up Story Plots

FORMERLY robots were merely mechanical devices that could perform a variety of stunts under the guidance of a human being, but now a robot has made its appearance that thinks, has a soul of a kind, creative imagination, and other qualities necessary for writing a modern stereotyped short story. This robot, the invention of Wycliffe Hill, a Los Angeles scenario writer, is declared to be able to build up millions of plots, no two alike, for magazine stories or movie plays.

Mr. Hill has equipped his robot with an index chart, divided into eight sections, one devoted to each of the eight elements of a story—background, character, obstacle, problem, predicament, complication, crisis and climax—and with an assortment of variations. The robot selects the material as required from this inexhaustible source and builds plots that could never be imagined by the author without the aid of the mechanical brain. Now if you want to become a successful author simply obtain a robot and put it to work.

Mr. Wycliffe Hill demonstrating his new story writing robot, which can think up any kind of plot with its mechanical brains.

This is an article from Popular Mechanics in 1931.

I research (among other things) *automated story generation.* I have not taught my course on AI storytelling in a number of years. I have put this primer together as a resource that I think my students need to know to get started on research on automated story generation. Anyone interested in the topic of automated story generation may find it informative. Since I have been actively researching automated story generation for nearly two decades, this primer will be somewhat biased toward work from my research group and collaborators.

1. What is Automated Story Generation?

Automated story generation is the use of an intelligent system to produce a fictional story from a minimal set of inputs. Let's tease this apart.

- **Narrative:** The recounting of a sequence of events that have a continuant subject and constitute a whole (Prince, 1987). An **event** describes some change in the state of the world. A "continuant subject" means there is some relationship between the events—it is about something and not a random list of unrelated events. What "relates" events is not entirely clear but I'll get to that later.
- **Story:** A narrative that tells a story has certain properties that one comes to expect. All stories are narratives, but not all narratives are stories. Unfortunately I cannot point to a specific set of criteria that makes people regard a narrative as a story. One strong contender, however, is a structuring of events in order to have a particular effect on an audience.
- **Plot:** A plot is the outline of main incidents in a narrative.

We might distinguish between automated story generation and automated plot generation. Recently, some have started distinguishing story generation from plot generation as whether the output of the system reads as an outline of main events versus having natural language that describes aspects of the story that are not strictly events, like descriptions, dialogue, and other elaborations. The distinction seems to be: does it read like a high-level outline of events, or does it look like something someone might find in a book. I'm not certain I would make this distinction, but I see the appeal for those that are more interested in surface form versus structure (two equally important and valid perspectives).

The *fictional* criteria distinguishes automated story generation from other storytelling technologies such as news writing, where the events are those that really happened in the real world. That is, news relies on the real world as the "generator" and then creates a natural language prose. News

generation is an important problem, but one that, in my opinion, should be separated out from automated story generation.

Minimal input is a criteria that I add to automated story generation to distinguish from *story retelling*. Story retelling is a problem where most or all of the story/plot is given and the automated system is producing an output that tracks the input closely. For example, one might give a story retelling system a trace of facts about a story in some abbreviated or structured form and the system might generate natural language prose that conveys those facts in order. In this case the "story" was already known but the surface form of the telling is variable. As to what a "minimal" set of input is is subject to debate. Should it be a single prompt for the start of the story? Should it be the start and a goal? Should it be a small number of plot points that get filled in? What about given domain knowledge? For machine learning systems, should we consider the corpus that is trained on as an input? This probably needs more work but I am loath to provide a definition that is overly narrow.

2. Why Study Automated Story Generation?

We can look at this question from a few angles. The first is applications. Aside from the grand challenge of an AI system that can write a book that people would want to read, storytelling appears in many places in society.

- **Human-AI coordination:** there are times when it is easier to communicate via narrative. For example, communicating via vignettes helps with coordination because it sets expectations against which to guage the appropriateness of behavior. Humans often find it easier to explain via vignettes, and are often able to more easily process complex procecural information via vignettes.

- **Human-AI rapport:** Telling and listening to stories is also a way that humans build rapport.
- **Explainable AI:** Explanations can help humans understand what an AI system does. For sequential decision making tasks (e.g. robotics) this might entail a temporal component to the explanation resembling a story.
- **Computer games:** many computer games feature stories or plots, which can be generated or customized. Going beyond linear plots, interactive stories are those in which the user assumes the role of a character in a story *and is able to change the story* with their actions. To be able to respond to novel user actions requires the ability to adapt or re-write the plot.
- **Training and education:** inquiry-based learning puts learners in the role of experts and scenarios can be generated to meet pedagogical needs (similar to interactive stories above).

Beyond application areas, story generation strikes at some fundamental research questions in artificial intelligence. To make a story requires planning with language.

To tell a story, an intelligent system has to have a lot of knowledge including knowledge about how to tell a story and knowledge about how the world works. These concepts need to be grounded to be able to tell coherent stories. The grounding doesn't have to be in vision or physical manipulation; knowledge can be grounded in shared experience. Shared experience is related to commonsense reasoning, an area of AI research that is believed to be essential for many practical appliactions.

Story generation is an excellent way to know if an intelligent system truly *understands* something. To understand a concept, one must be able to put that concept into practice — telling a story in which a concept is used correctly is one way of doing that. If an AI system tells a story about going to a restaurant, as simple as that sounds, we discover very quickly what the system doesn't understand when it messes up basic details.

Story generation requires an intelligent system to have *theory of mind,* a model of the listener to reason about what needs to be said or what can be left out and still convey a comprehensible story.

3. Narratology and Narrative Psychology

Before diving into technology, let's look at some of the things we can learn from narratology and narrative psychology. **Narratology** is a humanistic field that concerns itself study of narratives. **Narrative psychology** is a branch of psychology that looks at what happens in the human mind when reading stories.

There are a lot of branches of narratology. As a technologist I find the most value in structural narratology, which has provided frameworks for thinking about the structure of narratives. I have found some of their theories and frameworks to be operationalizable (which is not to say that other branches of narratology are not useful, just that I haven't been able to directly map their contributions to system engineering).

Structural narratology, drawing heavily from Bal (1998), analyze narratives at two levels:

- **Fabula:** The fabula of a narrative is an enumeration of all the events that occur in the story world between the time the story begins and the time the story ends. The events in the fabula are temporally sequenced in the order that they occur, which is not necessarily the same order in which they are told. Most notably, the events in the fabula might not all exist in the final telling of the narrative; some events might need to be inferred from what is actually told. For example: "John departs his house. Three hours later John arrives at the White House. John mutters about the traffic jam." The fabula clearly contains the events "John departs house" and "John arrives at the White House" and "John mutters". We might infer that John also drove a car and was stuck in a traffic jam — an event that was not explicitly mentioned and furthermore would have happened between "depart" and "arrive" instead of afterward when the first clue is given.
- **Sjuzhet:** The sjuzet of a narrative is a subset of the fabula that is presented via narration to the audience. It is not required to be told in chronological order, allowing for achronological tellings such as flash forward, flashback, ellipses (gaps in time), interleaving, achrony (randomization), etc.

Some further distinguish a third layer, **text** or **media**, which is the surface form that the reader/audience directly interfaces with. The text/media is the specific words or images from which the fabula is inferred.

While in reality human writers cognitively operate at all levels simultaneously, the distinction between fabula, sjuzhet, and text can be useful for operationalizing story generation systems. For example, some systems use a tripartite pipeline [citations needed] in which (1) an explicit fabula is generated as an exhaustive list of events, (2) a sjuzhet is generated by selecting a subset of events and possibly reordering them, and (3) the

generation of natural language sentences that describe each event in the sjuzhet in greater and more readable detail.

One line of narrative psychology looks at the mental models that readers create when reads a story. The psychology of narrative comprehension point to the importance of *causal relations* (Trabasso 1982) when human readers model the relationships between story events. In narrative, a causal relation between two events is an indication that the temporally latter event is made possible in part by the presence of the temporally former event (*narrative causality* is not the same as causality as used in statistics or Bayesian inference, where causality between random variables means that the value of one random variable affects the distribution of values of the other random variable). Narrative causality can be due to *direct cause* — a wheel falling off a car causes it to crash — or through *enablement* —the could not have crashed if the ignition hadn't been turned on.

Trabasso (1982) determined that human narrative understanding involved modeling of the causal relations between events in stories. A causal relation exists between two events *e1* and *e2* when *e1* occurs before *e2* and *e2* could not occur if *e1* had not occurred. Although causal relations implies that *e1* causes *e2*, it is more accurate to say that the presence of a causal relation captures enablement: *e1* enables *e2*.

Graesser et al. (1991) hypothesize that readers also model the goal hierarchies of story world characters, recognizing that characters emulate real humans in having goals and that actions also enable goals and intentions. Some events are goal events and some events are sub-goal events that are necessary steps in achieving a final goal event. For example, the event "John shoots Fred" might be a goal event preceded by a number of sub-goal events such as "John buys a gun" and "John loads the gun". These sub-

goal events precede the goal event and are also related to the goal event as part of a hierarchy. Some events also directly cause a character to form an intention, which manifests itself as a goal event later in the story (or at least the attempt to achieve a goal event). For example "Fred threatens to kidnap John's children" might be an event that causes John to intend to shoot Fred. This captures the notion of *direct cause* between two events as mediated by the mental state of a character.

4. Non-Learning Story Generation Approaches

Let's get into technologies. This cannot be exhaustive, so I have attempted to create some some broad classes and give some examples of each. This section looks at non-machine-learning based approaches. Non-learning systems dominated much of the history of automated story generation. They can be produce good plots though the emphasis on natural language output has been reduced. The key defining feature of these techniques — for the most part — is the reliance on knowledge bases containing hand-coded knowledge structures.

4.1. Story Grammars

Computational grammars were designed to decide whether an input sequence would be accepted by a machine. Grammars can be reversed to make generative systems. The earliest known story generator (Grimes 1960) used a hand-crafted grammar. The details are largely lost to history.

```
A LION HAS BEEN IN TROUBLE FOR A LONG TIME. A DOG STEALS SOMETHING THAT
BELONGS TO THE LION. THE HERO, LION, KILLS THE VILLAIN, DOG, WITHOUT A
FIGHT. THE HERO, LION, THUS IS ABLE TO GET HIS POSSESSION BACK.
```

The earliest known story generated by a grammar-based story generation system (1960).

In 1975, David Rumelhart (1975) published a grammar for story understanding. It was followed by a proposed story grammar by Thorndyke (1977).

Syntactic Rules and Semantic Interpretation Rules

(1) Story -> Setting + Episode
    => ALLOW (Setting, Episode)
(2) Setting -> (States)*
    => AND (State, state,.....)
(3) Episode -> Event + Reaction
    => INITIATE (Event, Reaction)
(4) Event -> {Episode | Change-of-state | Action | Event + Event}
    => CAUSE ($Event_1$, $Event_2$) or ALLOW ($Event_1$, $Event_2$)
(5) Reaction -> Internal Response + Overt Response
    => MOTIVATE (Interval-response, Overt Response)
(6) Internal Response -> {Emotion | Desire}
(7) Overt Response -> {Action | (Attempt)*}
    => THEN ($Attempt_1$, $Attempt_2$,.....)
(8) Attempt -> Plan + Application
    => MOTIVATE (Plan, Application)
(9) Application -> (Preaction)* + Action + Consequence
    => ALLOW (AND(Preaction, Preaction,..),
       {CAUSE | INITIATE | ALLOW} (Action, Consequence))
(10) Preaction -> Subgoal + (Attempt)*
    => MOTIVATE [Subgoal, THEN (Attempt,.....)]
(11) Consequence -> {Reaction | Event}

The Rumelhart story grammar.

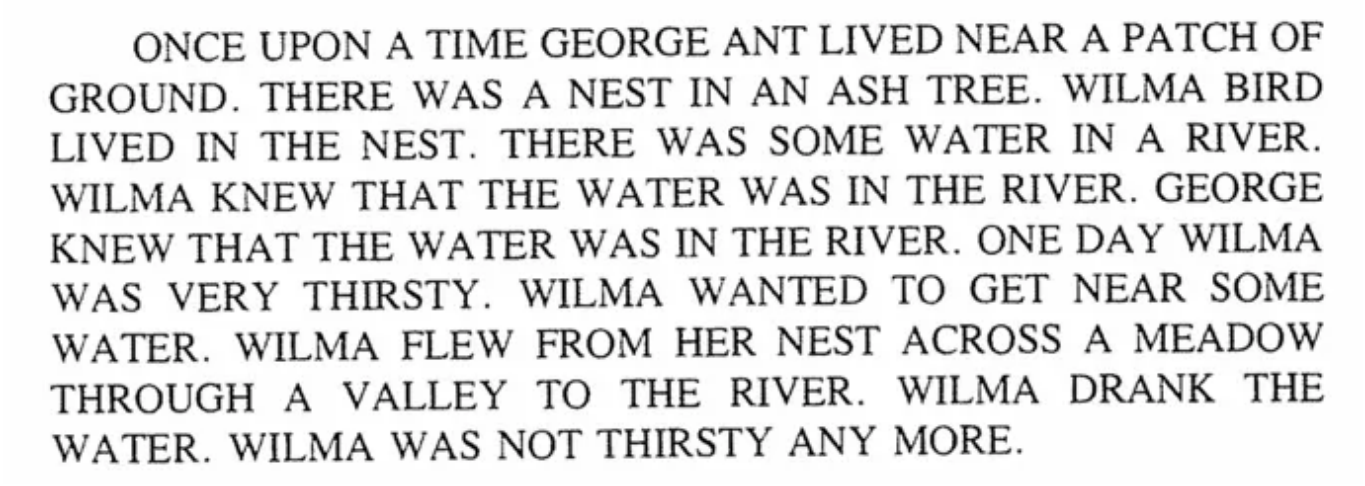
Black and Wilensky (1979) evaluate the grammars of Rumelhart and Thorndyke and come to the conclusion that they are not fruitful for story understanding. Rumelhart (1980) responds that Black and Wilensky misunderstood. Mandler and Johnson (1980) suggest that Black and Wilensky are throwing the baby out with the bathwater. Wilensky (1982) revisits story grammars and doubles-down on his critique. Wilensky (1983)

then goes on to propose an alternative to grammars called “story points”, which resemble schemata for plot points (see next section) but aren’t generative. Rumelhart goes on to work on neural networks and invents the back-propagation algorithm.

4.2. Story Planners

Story planners start with the premise that the story generation process is a goal-driven process and apply some form of symbolic planner to the problem of generating a fabula. The plan is the story.

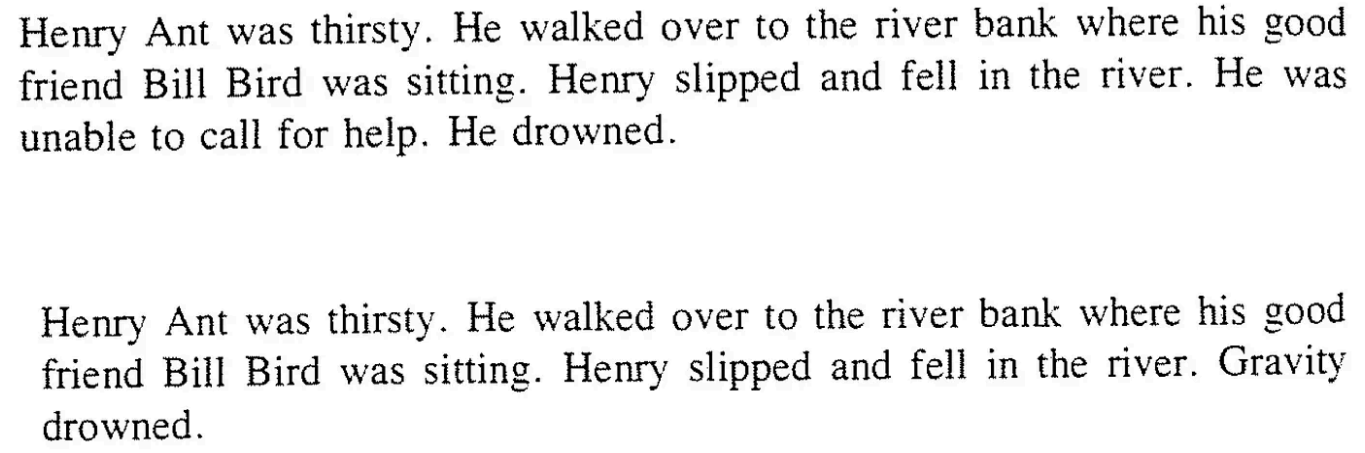
The system that is most commonly acknowledged as the first “intelligent” story generator is **Tale Spin** (Meehan 1977). Tale Spin originated out of work on story understanding, flipping things around to do story generation instead. Tale Spin builds off two core ideas. First is *conceptual dependency theory*, a theory of human natural language understanding that (for the purpose of conciseness) boils everything down into a small number of primitives: ATRANS (transfer an abstract relationship such as possession), PTRANS (transfer physical location), PROPEL (apply a force such as a push), GRASP (grasping an object), MOVE (movement of a body part). Second is **scripts**, which are procedural formulae (made up of conceptual dependencies) for achieving goals. Given a goal for a character, Tale Spin would generate a story by picking a script. This might result in further goals by the main character or other characters and further script selection.



ONCE UPON A TIME GEORGE ANT LIVED NEAR A PATCH OF GROUND. THERE WAS A NEST IN AN ASH TREE. WILMA BIRD LIVED IN THE NEST. THERE WAS SOME WATER IN A RIVER. WILMA KNEW THAT THE WATER WAS IN THE RIVER. GEORGE KNEW THAT THE WATER WAS IN THE RIVER. ONE DAY WILMA WAS VERY THIRSTY. WILMA WANTED TO GET NEAR SOME WATER. WILMA FLEW FROM HER NEST ACROSS A MEADOW THROUGH A VALLEY TO THE RIVER. WILMA DRANK THE WATER. WILMA WAS NOT THIRSTY ANY MORE.

A story generated by the Tale Spin system.

One of the more notable contributions of Tale Spin were the "mis-spun tales", which demonstrated the tight connections between the knowledge engineering of the scripts and the quality of the stories, as illustrated by example runs that failed in interesting ways.



Henry Ant was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. He was unable to call for help. He drowned.

Henry Ant was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. Gravity drowned.

Mis-spun tales generated by the Tale Spin system.

The **Universe** system followed with a more modern understanding of planning. We would now call Universe a greedy hierarchical task planner. Universe had a character generator (Lebowitz 1984) and a story planner (Lebowitz 1985). Universe, which generates soap operas, uses a library of schemas that indicate how a scene plays out. The schema can reference sub-

schemas. Starting with a high-level scene goal (e.g., “churn two lovers”), the system would pick a schema that described how to make two characters unhappy. The schema would contain a number of sub-goals. The system would construct the story by iteratively decomposing sub-goals.

```
PLOT FRAGMENT: forced-marriage
CHARACTERS: ?him ?her ?husband ?parent
CONSTRAINTS: (has-husband ?her)      {the husband character}
             (has-parent ?husband)   {the parent character}
             ( < (trait-value ?parent 'niceness) -5)
             (female-adult ?her)
             (male-adult ?him)
GOALS: (churn ?him ?her) {prevent them from being happy}
SUBGOALS: (do-threaten ?parent ?her "forget it") {threaten ?her}
          (dump-lover ?her ?him)        {have ?her dump ?him}
          (worry-about ?him)            {have someone worry about ?him}
          (together * ?him)             {get ?him involved with someone else}
          (eliminate ?parent)           {get rid of ?parent (breaking threat)}
          (do-divorce ?husband ?her)    {end the unhappy marriage}
          (or (churn ?him ?her)         {either keep churning or}
              (together ?her ?him))     {try and get ?her and ?him back together}
```

A plot fragment schema from the Universe system.

```
* (tell '(((churn neil liz)))

working on goal - (CHURN NEIL LIZ)
Several plans to choose from FORCED-MARRIAGE LOVERS-FIGHT JOB-PROBLEM
- using plan FORCED-MARRIAGE

working on goal - DO-THREATEN STEPHANO LIZ "forget it")
- using plan THREATEN

>>> STEPHANO threatens LIZ: "forget it"

working on goal - (DUMP-LOVER LIZ NEIL) - using plan BREAK-UP

>>> LIZ tells NEIL she doesn't love him

working on goal - (WORRY-ABOUT NEILL) - using plan BE-CONCERNED
Possible candidates - MARLENA JULIE DOUG ROMAN DON CHRIS KAYLA
Using MARLENA for WORRIER

>>> MARLENA is worried about NEIL

working on goal - (TOGETHER * NEIL)
Several plans to choose from SEDUCTION DRUNKEN-SNEAK-IN
SYMPATHETIC-UNION JOB-TOGETHER
- using plan SEDUCTION
Possible candidates - DAPHNE RENEE
Using DAPHNE for SEDUCER

>>> DAPHNE seduces NEIL

working on goal - (ELIMINATE STEPHANO)
Several plans to choose from ATTEMPTED-MURDER EXPOSE
- using plan ATTEMPTED-MURDER
Possible candidates - RENEE ALEX
Using RENEE for KILLER

>>> RENEE tries to kill STEPHANO

working on goal - (DO-DIVORCE TONY LIZ) - using plan DIVORCE

>>> LIZ and TONY got divorced

working on goal - (TOGETHER LIZ NEIL)
no acceptable plans
```

A story generated by the Universe system.

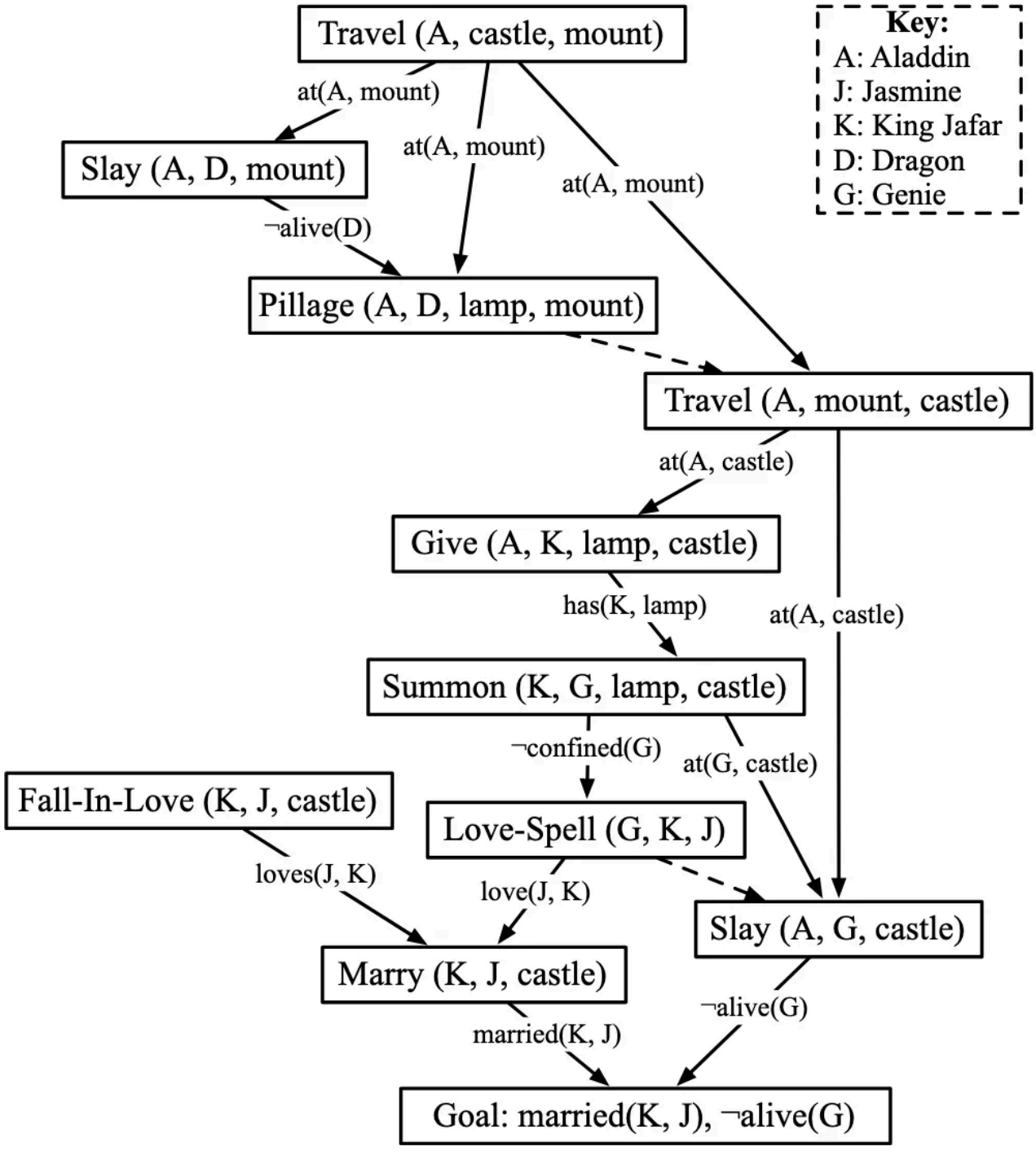
In the mid-90s symbolic planning became much more formalized in predicate logic. Formally, planners find a sequence of actions that transforms an initial state (given as a list of fact propositions) to a state in which a goal state holds (given as a list of propositions that must be made true). Action schema consisted of an action, operands, a precondition and an

effect. A precondition is a set of propositions that must be true for an action to execute. An effect is a description of how the world changes in terms of propositions that become true and those that become false.

```
      action: Eat (?monster, ?victim, ?location)
precondition: knows(?monster, ?victim),
              alive(?monster), alive(?victim),
              ¬eaten( ?victim),¬full(?monster),
              at(?monster, ?location),
              at(?victim, ?location),
              ?monster ≠ ?victim
      effect: eaten(?victim)
              in(?victim, ?monster), full(?monster),
              ¬at(?victim, ?location)
```

An action schema for a POCL planner.

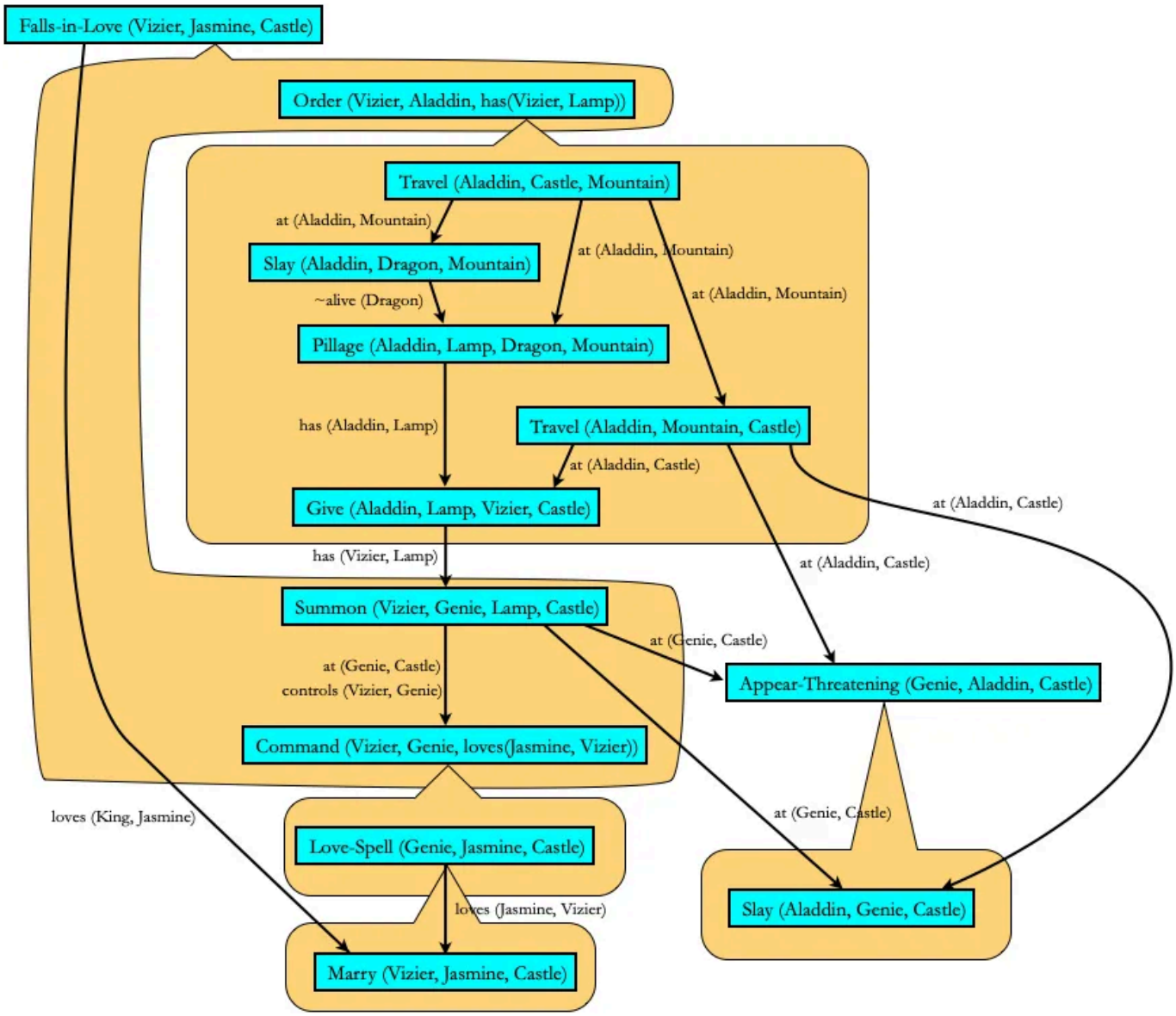
There are many ways of doing planning. A Partial-Order Causal-Link (POCL) planner chains from the goal backward to the initial state by seeking actions that make goal conditions true, and then seeking actions that make the preconditions of those actions true. This process continues until everything grounds out in the propositions listed in the initial state. An argument can be made that a least-commitment plan resembles the mental model of a reader because each chain from a precondition of some action to an effect of another action resembles enablement relations.



A story plan generated by a POCL planner from Riedl and Young (2010).

One of the problems with symbolic planners is that enablement isn't the only consideration for stories. If everything is driven by enablement of preconditions it ends up looking like all characters are collaborating on a shared goal — the given goal state. My **Fabulist** system (Riedl 2010) used a

newly invented type of POCL planner that built plan structures that created the appearance of character goal hierarchies and character intentions. Fabulist uses the tripartite framework: it first generates a plan data structure as the fabula, sub-selects actions to be part of the sjuzhet, then uses templates to render actions into natural language.



A story plan generated by Fabulist. The orange bubbles show actions that are part of goal hierarchies.

There is a woman named Jasmine. There is a vizier named Jafar. This is a story about how Jafar becomes married to Jasmine. There is a magic genie. This is also a story about how the genie dies.

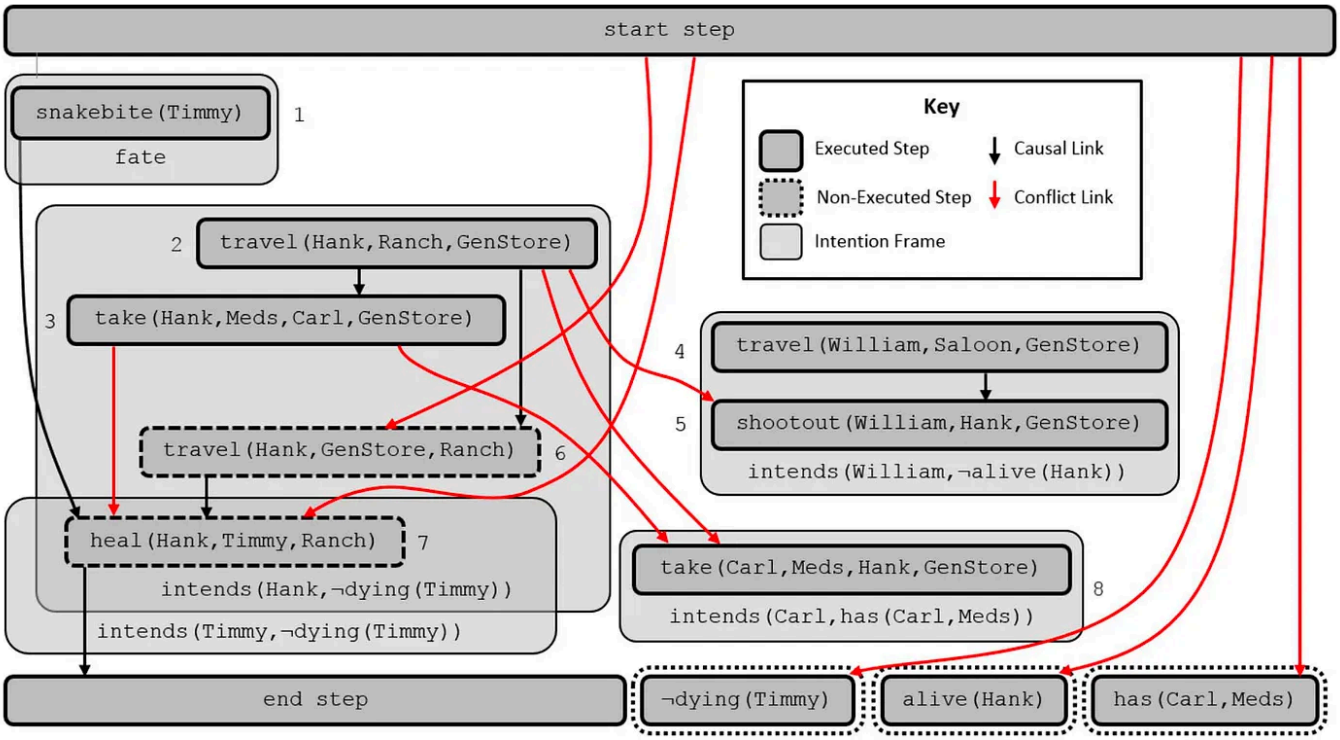
There is a magic lamp. There is a dragon. The dragon has the magic lamp. The genie is confined within the magic lamp.

Jafar is not married. Jasmine is very beautiful. Jafar sees Jasmine and instantly falls in love with her. Jafar wants to marry Jasmine. There is a brave knight named Aladdin. Aladdin is loyal to the death to Jafar. Jafar orders Aladdin to get the magic lamp for him. Aladdin wants Jafar to have the magic lamp. Aladdin travels from the castle to the mountains. Aladdin slays the dragon. The dragon is dead. Aladdin takes the magic lamp from the dead body of the dragon. Aladdin travels from the mountains to the castle. Aladdin hands the magic lamp to Jafar. The genie is in the magic lamp. Jafar rubs the magic lamp and summons the genie out of it. The genie is not confined within the magic lamp. Jafar controls the genie with the magic lamp. Jafar uses the magic lamp to command the genie to make Jasmine love him. The genie wants Jasmine to be in love with Jafar. The genie casts a spell on Jasmine making her fall in love with Jafar. Jasmine is madly in love with Jafar. Jasmine wants to marry Jafar. The genie has a frightening appearance. The genie appears threatening to Aladdin. Aladdin wants the genie to die. Aladdin slays the genie. Jafar and Jasmine wed in an extravagant ceremony.

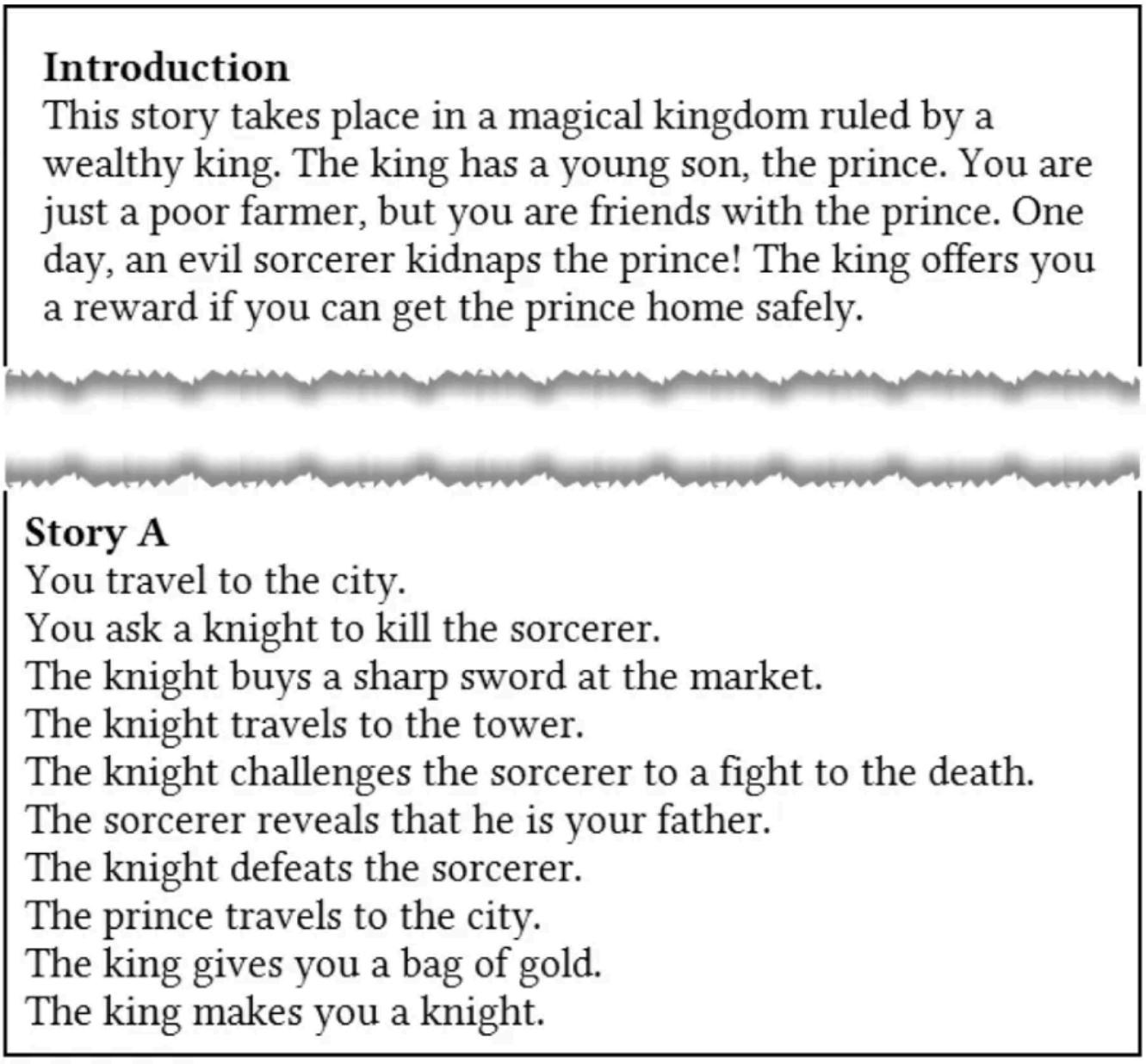
The genie is dead. King Jafar and Jasmine are married. The end.

A story generated by Fabulist corresponding to the above plan data structure.

Stephen Ware (2014) further modified POCL planners to ensure the appearance of character conflict (CPOCL). Planners like to avoid conflict because conflict breaks one or more of the plans of characters.



An example CPOCL plan with character conflict and un-executed actions.



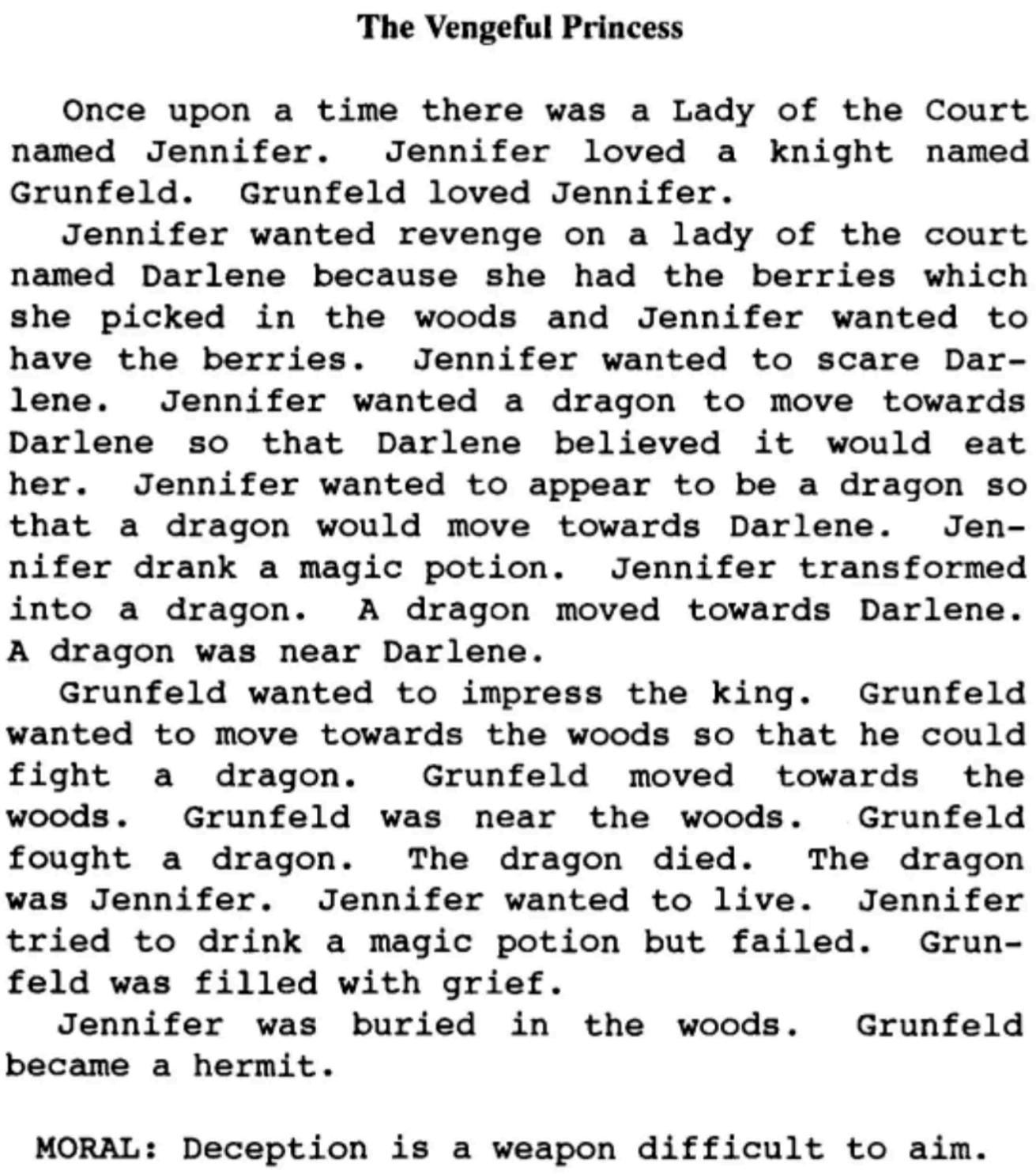
A story generated by CPOCL.

4.3. Case Based Reasoning

Case based reasoning is a theory of intelligence based on the idea that most reasoning is not done from first principles but instead adapts memories of solutions to related problems to new contexts. When a problem is encountered, the agent retrieves a solution to an older related problem, applies the old solution to the new problem, adapts the old solution to better fit the needs of the current problem, and then stores the new solution.

There is reason to believe that humans tell stories by adapting existing stories to new contexts. Case Based Reasoning approaches to story generation typically assume the agent has access to a library of existing stories and uses the above retrieve-apply-adapt-store process to transform an old story (or several old stories) into a new story.

The **Minstrel** system (Turner 1993) was arguably the most famous case-based story generation system. It used a lot of special rules for adaptation.



The Vengeful Princess

Once upon a time there was a Lady of the Court named Jennifer. Jennifer loved a knight named Grunfeld. Grunfeld loved Jennifer.

Jennifer wanted revenge on a lady of the court named Darlene because she had the berries which she picked in the woods and Jennifer wanted to have the berries. Jennifer wanted to scare Darlene. Jennifer wanted a dragon to move towards Darlene so that Darlene believed it would eat her. Jennifer wanted to appear to be a dragon so that a dragon would move towards Darlene. Jennifer drank a magic potion. Jennifer transformed into a dragon. A dragon moved towards Darlene. A dragon was near Darlene.

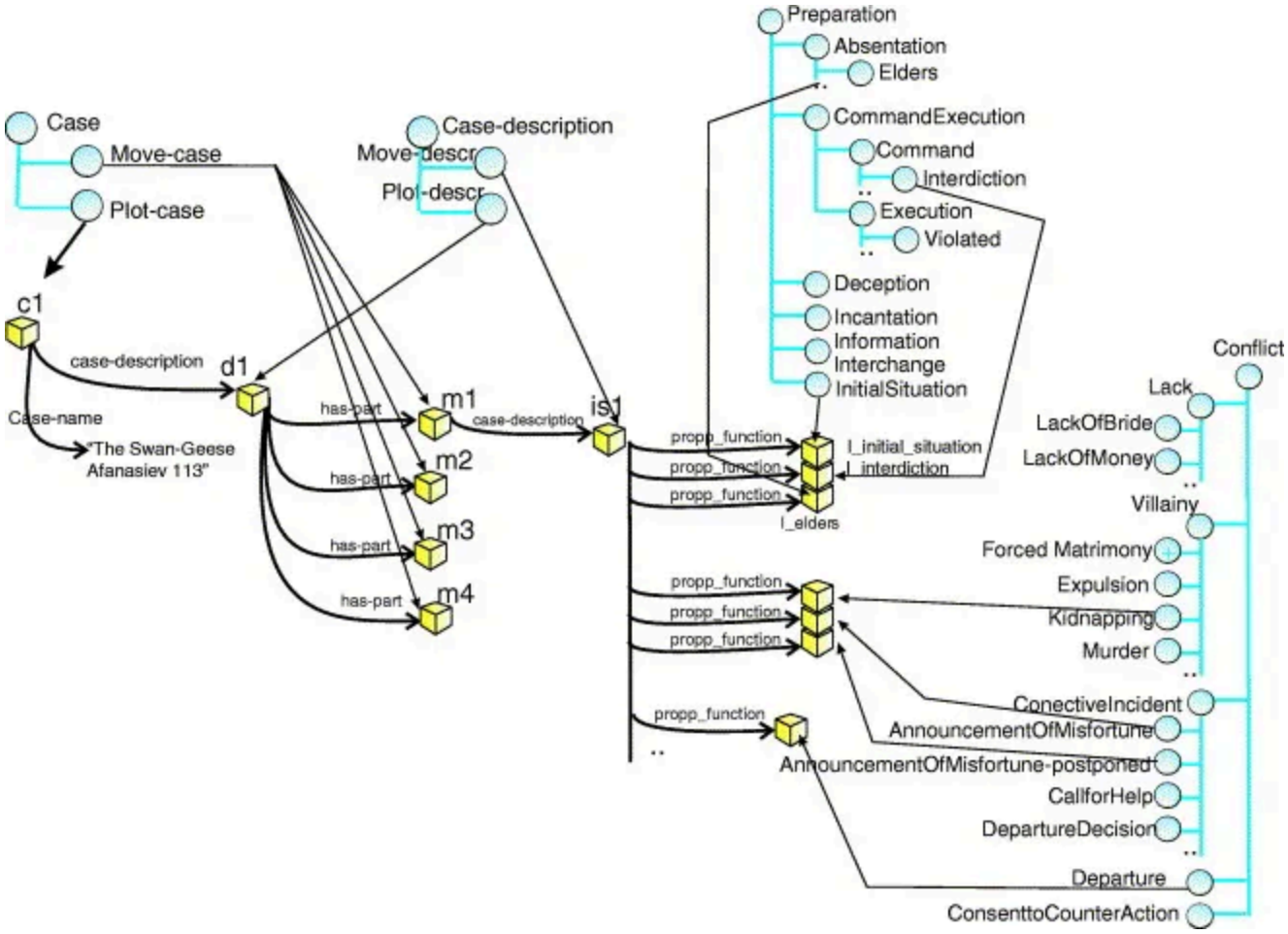
Grunfeld wanted to impress the king. Grunfeld wanted to move towards the woods so that he could fight a dragon. Grunfeld moved towards the woods. Grunfeld was near the woods. Grunfeld fought a dragon. The dragon died. The dragon was Jennifer. Jennifer wanted to live. Jennifer tried to drink a magic potion but failed. Grunfeld was filled with grief.

Jennifer was buried in the woods. Grunfeld became a hermit.

MORAL: Deception is a weapon difficult to aim.

A story generated by the Minstrel system.

ProtoPropp (Gervas et al. 2005) later used more general adaptation techniques. ProtoPropp used Vladimir Propp's analyses of Russian Folktales to construct a case library.



Case library for the ProtoPropp system.

The **Mexica** system (Perez y Perez and Sharples 2001) operationalizes a theory of creative writing. It alternates between two processes. The first process (engagement) operates a bit like a case-based reasoner. It compares the generated story so far to existing stories in it's library in order to produce some interesting continuations. These continuations aren't guaranteed to link up with existing story elements so a second process (reflection) uses something that looks a lot like a backward chaining planner to revise the new snippets and add new actions to link everything up.

Jaguar_knight was an inhabitant of the great Tenochtitlan. Princess was an inhabitant of the great Tenochtitlan. From the first day they met, Princess felt a special affection for Jaguar_knight. Although at the beginning Princess did not want to admit it, Princess fell in love with Jaguar_knight. Princess respected and admired Artist because Artist's heroic and intrepid behaviour during the last Flowery-war. For long time Jaguar_knight and Princess had been flirting. Now, openly they accepted the mutual attraction they felt for each other. Jaguar_knight was an ambitious person and wanted to be rich and powerful. So, Jaguar_knight kidnapped Artist and went to Chapultepec forest. Jaguar_knight's plan was to ask for an important amount of cacauatl (cacao beans) and quetzalli (quetzal) feathers to liberate Artist. *Princess had ambivalent thoughts towards Jaguar_knight. On one hand princess had strong feelings towards Jaguar_knight but on the other hand Princess abominated what Jaguar_knight did. Suddenly, the day turned into night and after seconds the sun shone again. Princess was scared. The Shaman explained to Princess that Tonatiuh (the divinity representing the sun) was demanding Princess to rescue Artist and punish the criminal. Otherwise Princess's family would die. Early in the Morning Princess went to Chapultepec forest.* Princess thoroughly observed Jaguar_knight. Then, Princess took a dagger, jumped towards Jaguar_knight and attacked Jaguar_knight. *Jaguar_knight was shocked by Princess's actions and for some seconds Jaguar_knight did not know what to do*. Suddenly, Princess and Jaguar_knight were involved in a violent fight. In a fast movement, Jaguar_knight wounded Princess. An intense haemorrhage arose which weakened Princess. Jaguar_knight felt panic and ran away. Thus, while Tlahuizcalpantecuhtli (the god who affected people's fate with his lance) observed, Princess cut the rope which bound Artist. Finally, Artist was free again! *Princess was emotionally affected and was not sure if what Princess did was right. Princess was really confused*. The injuries that Princess received were very serious. So, while praying to Mictlantecuhtli (the lord of the land of the dead) Princess died.

A story generated by the Mexica system. Regular text was generated during the engagement phase. Text in italics was generated during the reflection phase.

Some of my work attempts to further reconcile case-based reasoning and POCL planning (Riedl 2008) as well as did some closer exploration of the adaptation phase of case-based planning for stories (Li and Riedl 2010). Related to case-based reasoning is story generation via analogical reasoning (Riedl and Leon 2009).

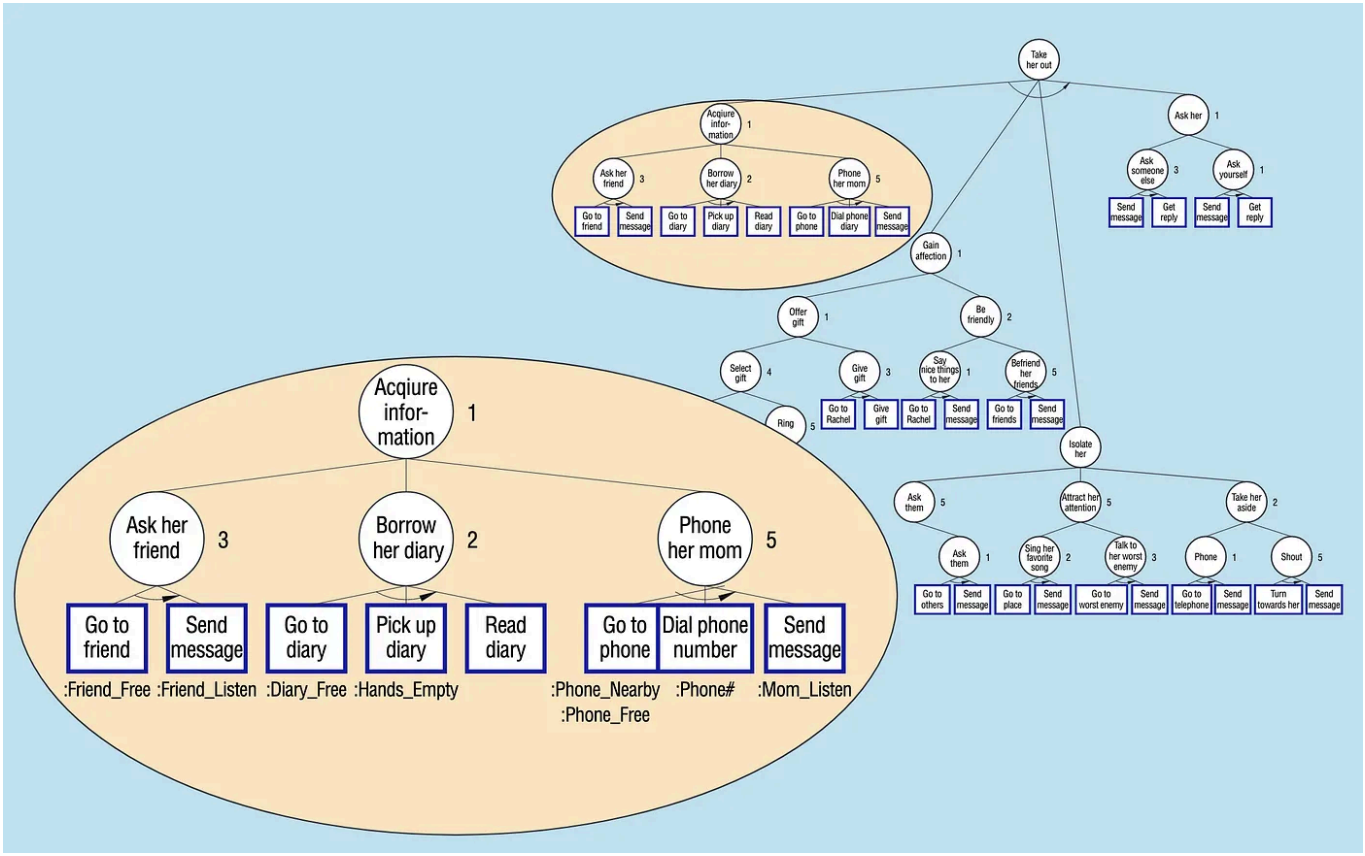
The **SayAnything** system (Swanson and Gordon 2012) is an example of *textual case-based reasoning*. Textual case-based reasoning treats a large text corpus as an unstructured case base. Most case bases are structured for easier retrieval and adaptation, so retrieving and adapting is much more challenging with unstructured text. The SayAnything system mines stories

from blogs and retrieves snippets from the blog stories in response to user-written segments of story. The system takes turns between human and computer adding to the story.

4.4. Character-Based Simulation

The above approaches can be thought of as **author-centric** (Riedl 2004) — the story generators assume the role of a singular author responsible for plotting out all the actions and events of all the characters.

There is another way to generate stories: simulate characters. In a **character-centric** simulation, each character is an autonomous agent with its own independent reason. They respond to their environment and to other characters based on their own beliefs, desires, and intentions. One example of this is the system by Cavazza, Charles, and Mead(2001), which used a modern hierarchical task network planner. Each agent had a goal and formed its own plan. Those plans may come into conflict, requiring one agent to replan (which can be done very quickly in HTN planning). This system was technically an interactive storytelling system because a user could inject new facts or beliefs into the story world, causing agents to replan in response.



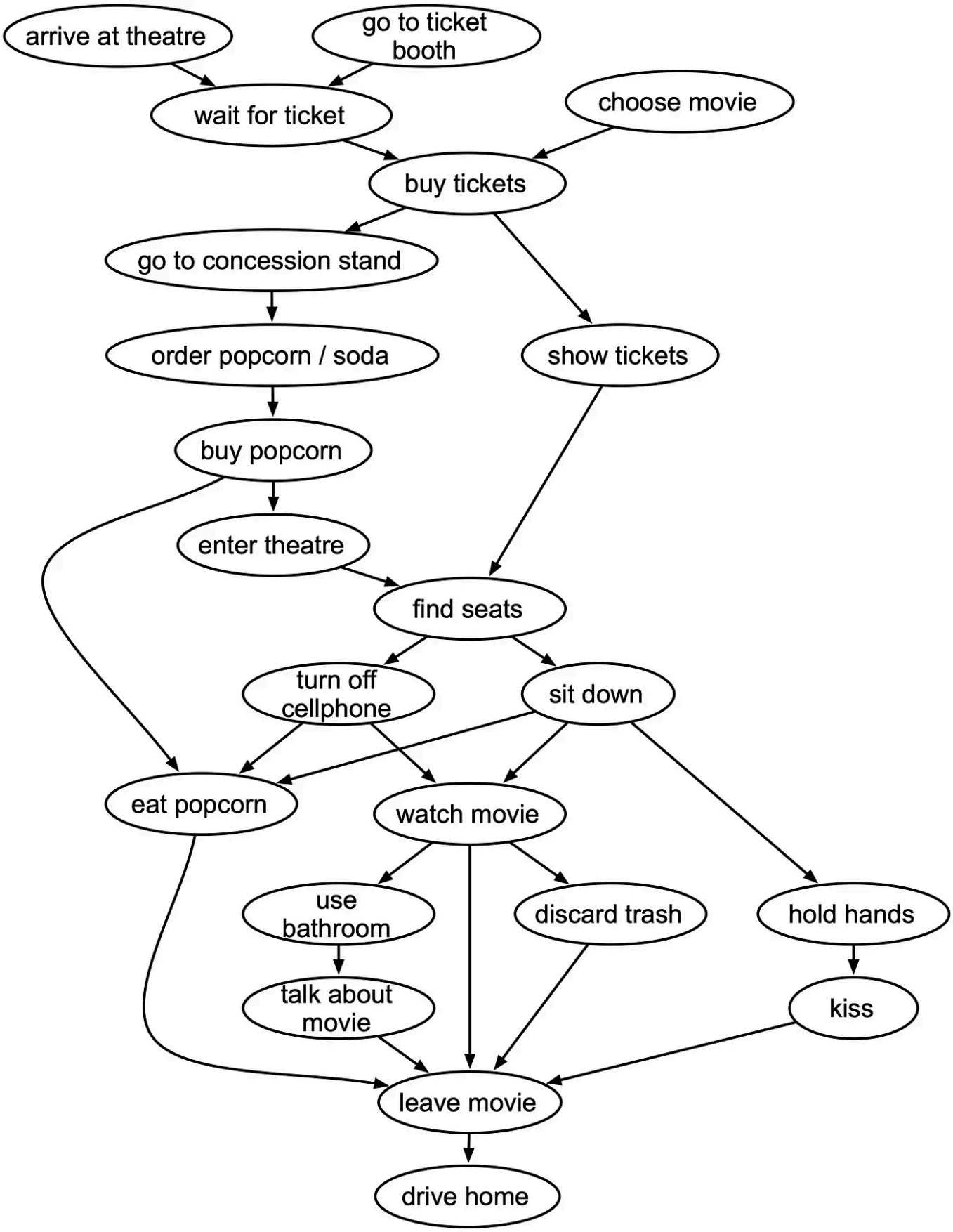
The hierarchical task network for character agents in a story simulation.

Virtually any agent technology capable of responding to changes in the environment can be applied to chacter-centric simulation based approaches to story generation. One of the limitations of the simulation approach is that there can be few guarantees about what will happen when the agents start executing. Theoretically if the agents were models of *actors* instead of characters, then they would have some ability to make decisions based on storytelling principles (Louchart and Aylett 2007) instead of being fully operating from the perspective of the character (the difference between a model of James Bond vs. building a model of Sean Connery). There has been some work in trying to model improv theatre actors (Riedl, 2010; Magerko and O'Neill 2012).

5. Machine Learning Story Generation Approaches

In this section we explore machine learning approaches that do not use neural networks.

One of the challenges of the non-learning approaches is that a vast majority of non-learning systems either require schemas encoded into a symbolic format (e.g., hierarchical plot schemas, symbolic scripts, action schema with preconditions and effects, cases stored in symbolic formats, etc.). Machine learning can be used for knowledge acquisition. The **Scheherazade** system (Li et al. 2012, 2013) maintains a memory of *plot graphs* which are partially-ordered graphs that represent the most likely ordering of events on a particular topic. For example, a plot graph for going to a restaurant might have key events such as ordering food, eating, paying, leaving, etc. The arcs are temporal order constraints, e.g., paying occurs before leaving. Plot graphs resemble scripts in spirit except (a) they are partially ordered instead of fully ordered, and (b) events are arbitrary strings. When Scheherazade is asked to tell a story about a topic but doesn't have a corresponding plot graph, it crowdsources example stories and then learns a plot graph that takes the most often mentioned events and learns the most likely ordering constraints. The system doesn't have a prescribed dictionary of what construes an "event" — an event is a cluster of semantically related strings. Once a plot graph is learned, the generation process involves selecting event nodes such that no temporal constraints are violated. Since events are are clusters of semantically related sentences from the crowdsourced corpus, the final story can be produced by selecting a linear sequence of nodes and then selecting a sentence from each node.



A plot graph learned by Scheherazade for going on a date to a movie theatre.

With sweaty palms and heart racing, John drove to Sally's house for their first date. Sally, her pretty white dress flowing in the wind, carefully entered John's car. John and Sally drove to the movie theater. John and Sally parked the car in the parking lot. Wanting to feel prepared, John had already bought tickets to the movie in advance. A pale-faced usher stood before the door; John showed the tickets and the couple entered. Sally was thirsty so John hurried to buy drinks before the movie started. John and Sally found two good seats near the back. John sat down and raised the arm rest so that he and Sally could snuggle. John paid more attention to Sally while the movie rolled and nervously sipped his drink. Finally working up the courage to do so, John extended his arm to embrace Sally. He was relieved and ecstatic to feel her move closer to him in response. Sally stood up to use the restroom during the movie, smiling coyly at John before that exit. John and Sally also held hands throughout the movie, even though John's hands were sweaty. John and Sally slowly got up from their seats. Still holding hands, John walked Sally back to his car through the maze of people all scurrying out of the theater. The bright sunshine temporarily blinded John as he opened the doors and held them for Sally as they left the dark theater and stepped back out onto the street. John let go of Sally's hand and opened the passenger side door of his car for her but instead of entering the car, she stepped forward, embraced him, and gave him a large kiss. John drove Sally back to her home.

A story generatedy by Scheherazade for the plot graph above.

6. Neural Story Generation Approaches

The past few years have seen a steady improvement in the capabilities of neural networks for text. The literature on neural network based story generation techniques is growing rapidly, which requires me to focus on just a few of the systems and works that I found notable at the time of writing.

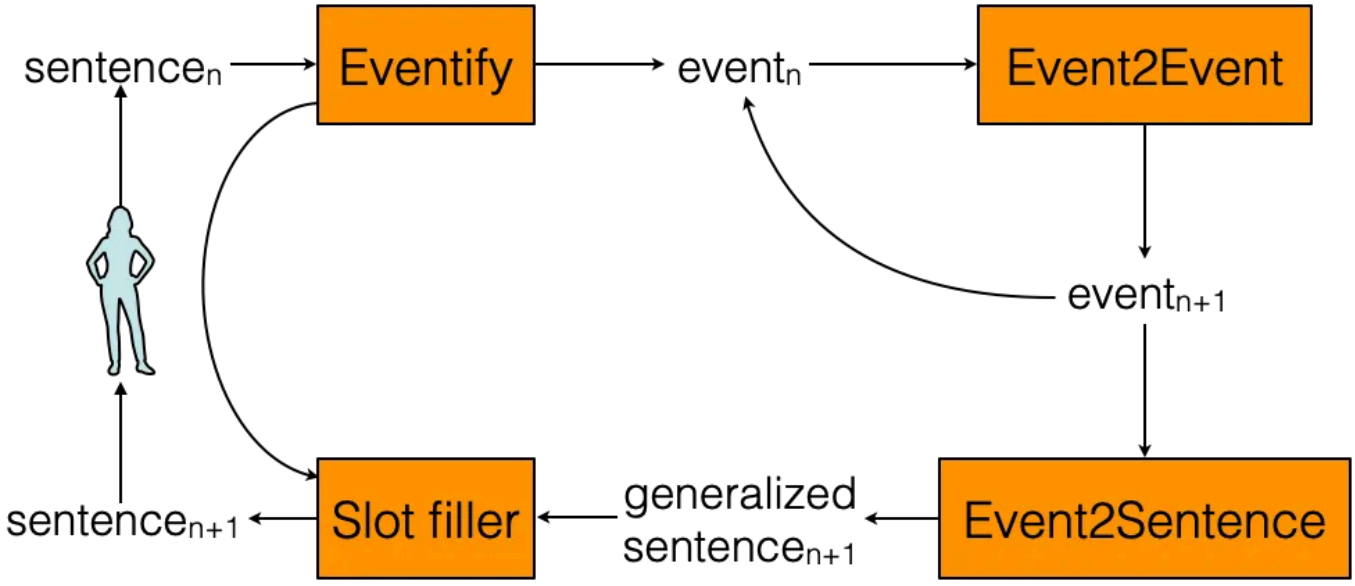
6.1. Neural Language Models

A language model learns the probability of a token (or sequence of tokens) based on a history of previously occurring tokens. The model is trained on a particular corpus of text. Text can be generated by sampling from the language model. Starting with a given prompt, the language model will provide one or more tokens that continue the text. The prompt plus the continuation can be input into the language model to get the next continuation, and so on. Training a language model on a corpus of stories means the language model will attempt to emulate what it has learned from

at corpus. Thus sampling from a language model trained on a story corpus tends to produce text that looks like a story.

Recurrent neural networks such as LSTMs and GRUs were initially explored for text generation (Roemelle and Gordon 2017, Khalifa et al 2017). Initially, if the story training corpus was too diverse, RNN-based language models would have difficulty producing sequences that didn't appear too random. This is due to sparsity — stories are written to be unique from each other and any sentence or event in a story might be unique or nearly unique. This makes learning a language model on stories difficult.

Martin et al. (2018) addressed this by trying to reduce the sparsity of story corpora by converting stories from natural language to event tuples *e*=<*s*, *v*, *o*, *p*, *m*> where *s* is a subject, *v* is a verb, *o* is a direct object of the verb, and, optionally, *p* is a preposition and *m* is an additional noun that accompanies the preposition. For example, "Sally repeatedly shot her cheating boyfriend in the bedroom" would be represented as <Sally, shot, boyfriend, in, bedroom>. These terms were further generalized by replacing them with semantic word classes from WordNet or semantic verb frame classes from VerbNet, for example <person-1, murder-42.1, male.2, in, area.5>. In theory this reduced the sparsity and it was shown that generators could learn to make better event sequences than when working with raw text. This created the problem that event tuples were hard for humans to read so they introduced a second neural network to convert event tuples back to full sentences. "Eventification" of sentences is a lossy process so this neural network was tasked with restoring information in a contextual way. This is a challenging problem tantamount to re-telling a story abstraction with natural language. An ensemble was found to work best (Ammanabrolu 2020).



The generation loop for Martin et al. (2018).

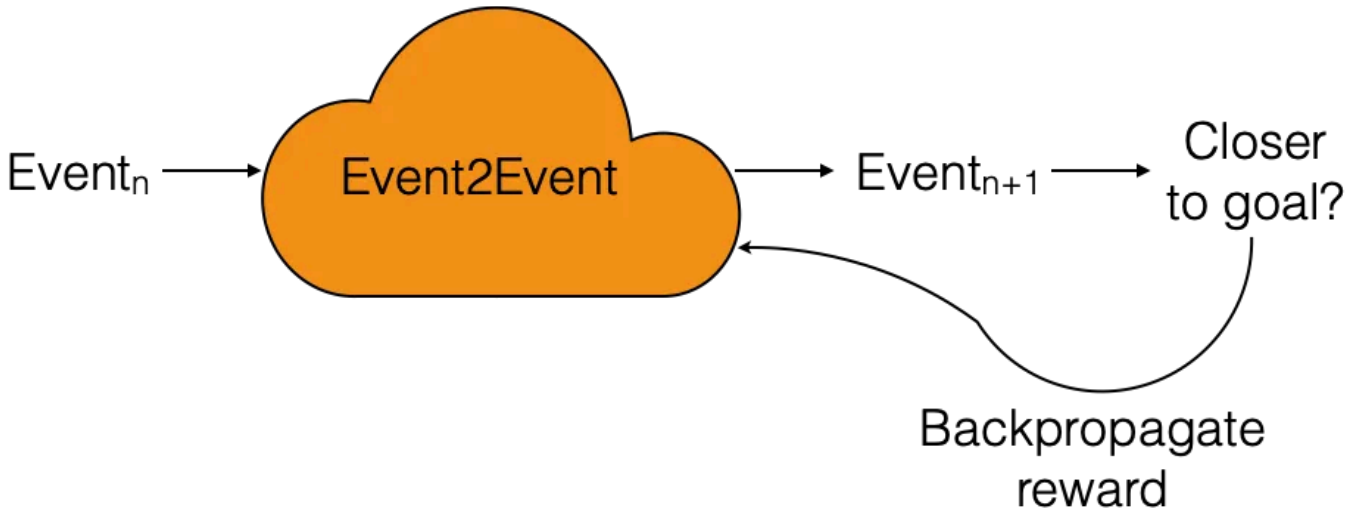
A single language model doesn't distinguish strongly between characters, which are just tokens. The work by Clark, Ji, and Smith (2018) learn to represent different entities so that stories can be generates about different characters.

6.2. Controllable Neural Story Generation

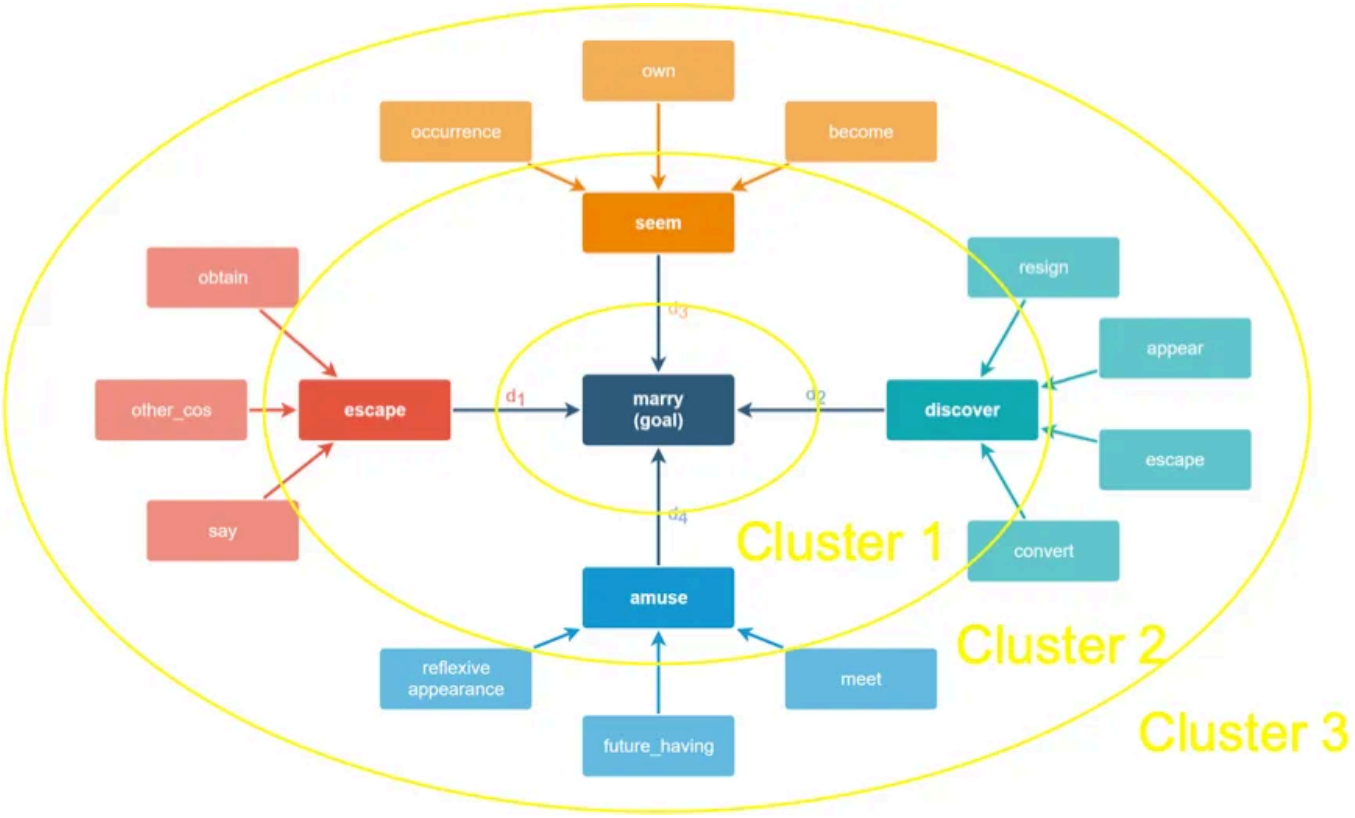
One of the main limitations of neural language models is that they generate tokens based on a sequence of previous tokens. Since they are backward-looking instead of forward-looking, there is no guarantee that the neural network will generate a text that is coherent or drives to a particular point or goal. Furthermore, as the story gets longer, the more of the earlier context is forgotten (either because it falls outside of a window of allowable history or because neural attention mechanisms prefer recency). This makes neural language model based story generation systems "fancy babblers" — the stories tend to have a stream-of-consciousness feel to them. Large-scale pre-trained transformers such as GPT-2, GPT-3, BART, and others have helped with some of the "fancy babbling" issues by allowing for larger context windows, but the problem is not completely resolved. As language models

themselves they cannot address the problem of forward-looking to ensure they are building toward something in the future, except by accident.

Tambwekar et al. (2019) use reinforcement learning to fine-tune a sequence-to-sequence language model to generate story continuations that move toward a given goal. Reinforcement learning, generally, is a technique that can be used to solve sequential decision-making problems. However the latent space of a language model is too large for true trial-and-error learning. The reinforcement learner is used as a non-differentiable loss function. But how does the reinforcement learner know whether the continuation is getting closer to a particular goal? The system extracts patterns of verbs from the story corpus, clusters them according to how far each verb is from the goal, and then rewards the language model when it produces a continuation with a verb in the next cluster closer to the goal.



Fine-tuning the event2event neural network from the Martin et al. (2018) framework.



Fine-tuning reward is calculated by analyzing how close verbs are to each other in the corpus.

While typically language models are given a prompt that is the first line of the story, neural language models can also be conditioned on plot points that need to be present in the story. This increases the probability that certain events will occur as the story generation progresses. The **hierarchical fusion model** (Fan et al. 2018) takes a one-sentence description of the story content and produces a paragraph. The **plan-and-write** technique (Yao et al. 2019) also uses a two-level approach. Instead of a single high-level sentence, the plan-and-write system learns to generate a sequence of keywords. Each of these keywords is then used to condition a language model so that it produces content about that keyword. If the keywords present a coherent progression then it stands that the lower-level elaborated content will as well. **PlotMachines** (Rashkin et al. 2020) conditions a generator on a set of concept phrases given by a user. However, instead of using this set of concept phrases as an outline to elaborate on, the system can decide for itself what order to introduce the concepts.

Example Prompt 1: Aliens start abducting humans
It has been two weeks, and the last of my kind has gone. It is only a matter of time until there will be nothing left. I'm not sure what the hell is going on... I can't think. I can hear a distant scream. I think of a strange, alien sound. I try to speak, but am interrupted by something, something that feels like a drum, I ca not tell. I mean I'm just a little bit older than an average human. But I can, and I can feel the vibrations . I hear the sound of a ship approaching. The ground quakes at the force of the impact, and a loud explosion shatters the silence.
Example Prompt 2: The scientists have discovered something terrible .
The scientist stood there, a little dazed as he stared. "What is it?" He asked. "This...this...Thing...This is a virus. A chemical that can destroy entire planet and it is a very small, complex, chemical that could destroy any planet." The scientist replied. His lab assistant looked down at the tablet. "I've just discovered it. I can't believe it. It looks like it's made of some sort of chemical that's very dangerous." " Well, there's a virus on the ground. It's very effective..." " I can't believe what it is." he said, pointing to the scientist . " We don't know what this thing is. We haven't seen anything like it . We can't even see anything like this. " Dr. Jones stared at the scientist for a moment. "What do you mean what does it do ?" "It...It 's a monster."
Example Prompt 3: The silence before the storm comes
"I'm sorry," she said "I can't do this anymore." She sat in the grass, the grass blowing behind her as she sat staring at the sky. Her eyes were still closed but she was still staring at it. Her hand was still shaking, her eyes closed as she watched the rain roll in and out of view. She could see the world through her eyelids now. Her hands were still shaking like she hadn't eaten in days. Her fingers were still clenched tight and her fingers were now shaking. She wasn't even sure if she was going to cry or if she just wanted to cry. It wasn't until she was close enough to hear the words. "I ...I just...I need to...I ..." She was crying now and then, the rain was picking up. Her lips were shaking now and she could see that the storm was getting closer and closer. Her heart stopped when she heard the thunder. She knew it was coming. The storm was

Stories generated by the hierarchical fusion model.

Dynamic	Storyline	needed → money → computer → bought → happy
	Story	John needed a computer for his birthday. He worked hard to earn money. John was able to buy his computer. He went to the store and bought a computer. John was happy with his new computer.
Static	Storyline	computer → slow → work → day → buy
	Story	I have an old computer. It was very slow. I tried to work on it but it wouldn't work. One day, I decided to buy a new one. I bought a new computer .

Stories generated by the plan-and-write system.

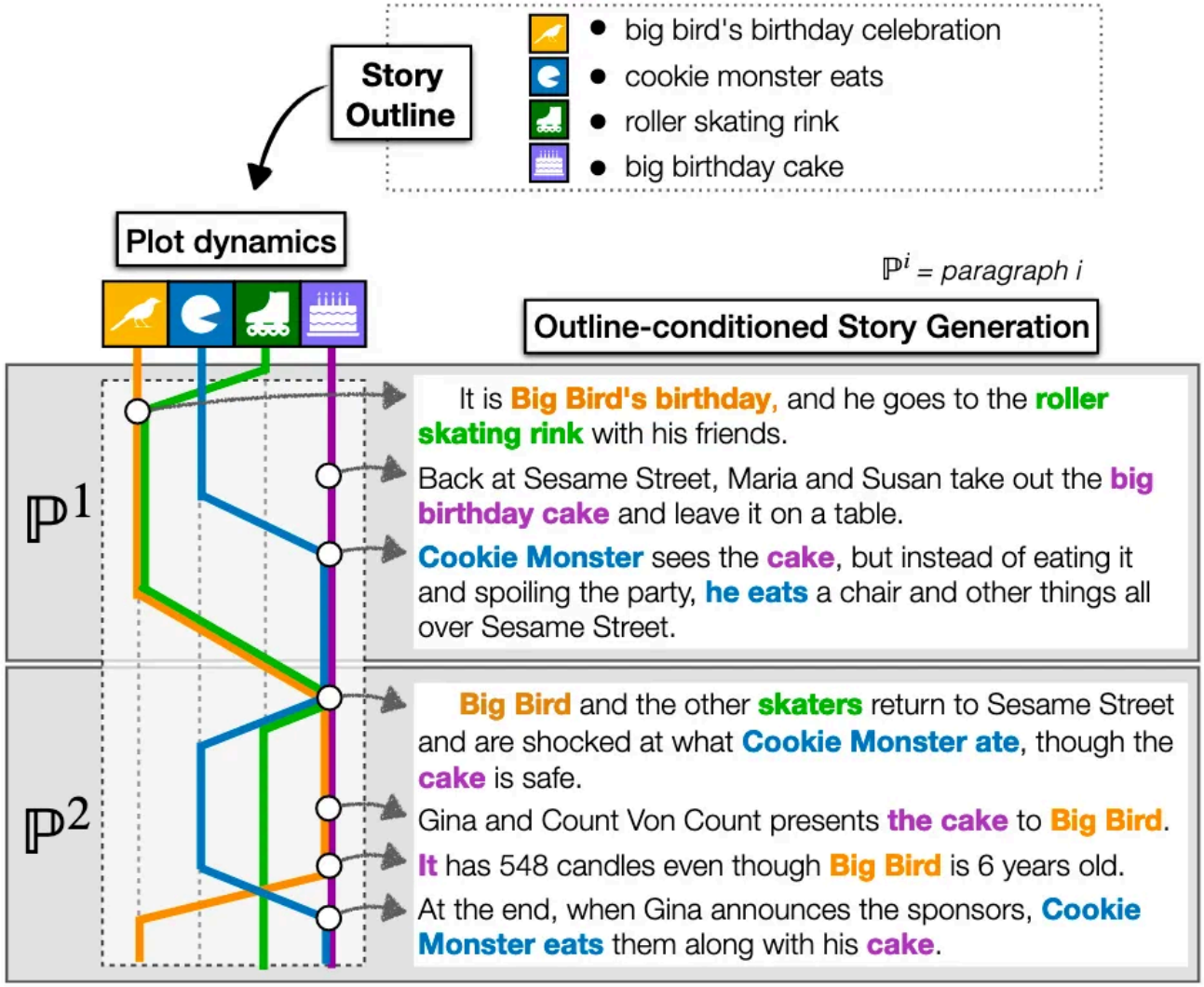


Illustration of inputs and outputs of the PlotMachines system.

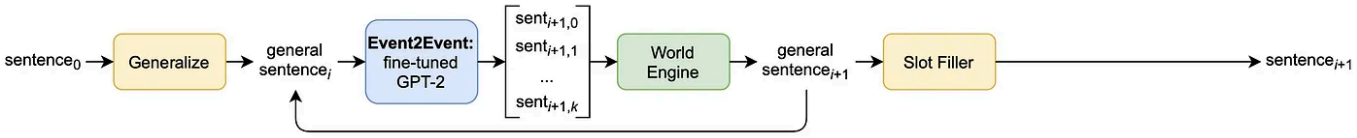
Another way of controlling the direction of a story is to provide key points in the story and in-filling. Wang, Durrett, and Erk (2020) attempt to interpolate between a given beginning and a given ending of the story. However, the language model they use doesn't know how to attend to the end prompt. Instead they generate a number of candidates starting with the beginning prompt and use a re-ranker to judge the overall coherence (beginning, middle, and end), taking the best. Ippolito et al. (2019) propose an in-filling approach where they use the given beginning and ending to a story and generate keywords that might be found in the middle of the story. The keywords are then used to condition a language model that generates the middle of the story.

6.3. Neuro-Symbolic Generation

One of the issues with neural language models is that the hidden state of the neural network (whether a recurrent neural network or a transformer) only represents what is needed to make likely word choices based on a prior context history of word tokens. The "state" of the neural network is unlikely to be the same as the mental model that a reader is constructing about the world, focusing on characters, objects, places, goals, and causes. The shift from symbolic systems to neural language models shifted the focus from the modeling of the reader to the modeling of the corpus. This makes sense because data in the form of story corpora is readily available but data in the form of the mental models readers form is not readily available. Assuming the theories about how reader mental models can be represented symbolically are correct, can we build neurosymbolic systems that take the advantages of neural language models and combine them with the advantages of symbolic models? Neural language models gave us a certain robustness to a very large space of inputs and outputs by operating in language instead of limited symbols spaces. But neural language model based story generation also resulted in a step backward from the perspective of story coherence. Symbolic systems on the other hand excelled at coherence through logical and graphical constraints but at the expense of limited symbol spaces.

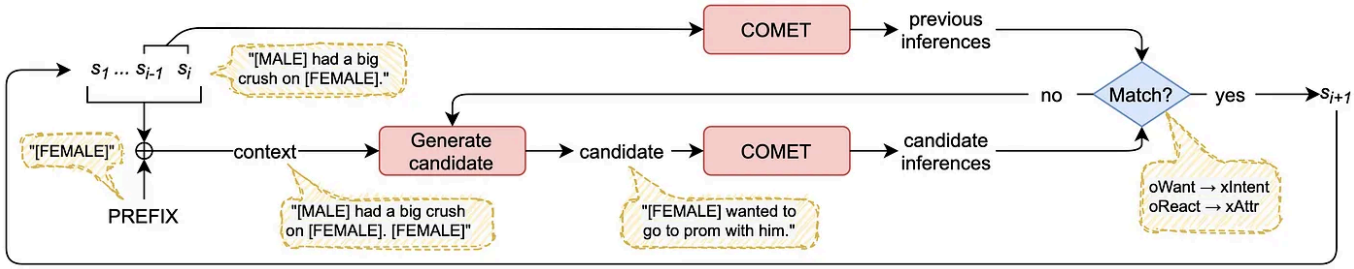
Lara Martin (Dissertation, 2021) proposes to take a neural language model such as GPT-2 and constrain it with reasoning about causality. GPT-2, as a language model, can generate story continuation probabilistically based on a history of prior word tokens. Martin's system parses the generated continuation sentence and uses VerbNet to infer what would be known to readers about the preconditions and effects of the sentence. If the preconditions of the continuation are not supported by the effects of prior sentences in the story, the continuation is rejected. If the preconditions are

supported, then the effects of the sentence are used to update a set of logic-like prepositions that describe the world. These prepositions, which are drawn from VerbNet exactly because they are based on what readers can infer from reading a sentence, can be thought of as a simple **reader model**. By tracking the hypothetical reader's model this system is less likely to make transitions from one event to another that do not make sense to the reader.



The neurosymbolic architecture by Martin. The World Engine maintains a set of propositions about the story world.

The CAST system (Peng et al. 2021) does something similar to the above neuro-symbolic generation system. CAST uses the COMET commonsense inference model as the world engine. It infers the needs and wants of the characters in the story and attempts to match against the wants and needs of the characters from previous events. Character goals and intents are another way beyond preconditions and effects that readers track the coherence of stories.



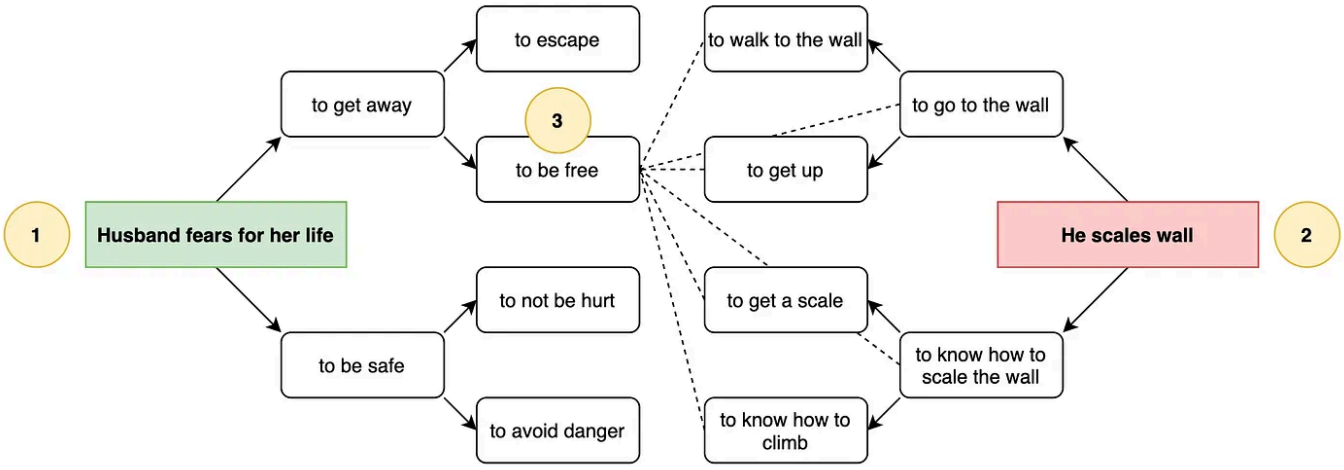
The CAST pipeline.

6.4. Other Neural Approaches

Directly sampling continuations from a language model is not the only plausible way of using neural networks to generate stories. One might

imagine search-like algorithms that use neural networks as resources for making decisions.

A lot of storytelling involves commonsense knowledge. Commonsense knowledge, or more specifically shared knowledge, is knowledge that one can assume is held by most people. To create a story that will make sense to a human reader, it might be advantageous to leverage commonsense knowledge. Neural networks such as COMET and GLUCOSE are trained to take input sentences and make inferences about what humans might also infer. The **C2PO** system (Ammanabrolu et al. 2021) takes a starting event and ending event and attempts to fill in the middle of the story. Instead of using a language model, it uses COMET to infer what most humans might believe comes next after the start and what might come before the end. This process is repeated, creating a directed acyclic graph of plausible successors and predecessors until a complete path is found from start to end.



The graph built by C2PO. 1) the initial event. 2) the final event. 3) the event found that bridges the forward and backward sub-graphs.

They live out at time seven years. Bearskin village is just came. Bearskin and family wants country. Bearskin only takes river gives valley. Bearskin gave purse of gold. Bearskin brothers also agreed. Bearskin with senior chief heads agreed. Bearskin daughter among elder sisters agrees. Bearskin promised return in three years. Bearskin they had had promised. Bearskin daughter was sisters agreed. Bearskin daughter sister married my family. **Her sisters ridiculed her.** Bearskin also always once reappeared. Bearskin who always is later left. Bearskin also becomes a. **Bearskin found devil again At end of seven years.** Bearskin says and so says. Bearskin he can sing now sings. Bearskin it has told him cries. **He fulfill his promise.** Bearskin polish grease nail nails. Bearskin polish cut boot. Bearskin helps clean burn wood cuts. **Bearskin clip his nails.** Bearskin boots and mr. Bearskin boots nails and boot. Bearskin leather leather toe boot. **He is good.** Bearskin brother had still also stood. Bearskin looked and then said. Bearskin claimed it did. **Bearskin dropped his half of ring.** Bearskin shifted and he changed. Bearskin and slowly transformed. Bearskin so gently lifted and turned. **He was her bridegroom.**

Prince calls at time night. Prince and tries again calls. Prince that still wakes. Prince then knocks and asleep. **He finds To his horror.** Prince robert francis charles george leaps. Prince charles henry louis rupert. Prince joseph john frederick maurice victor. **He leaps from tower.** Prince anthony or saint john leaves. Prince anthony nicholas edward lawrence. Prince edward nicholas james peter george. **He wanders For years.** Prince james edward thinks. Prince james john james. Prince alexander rupert james augustus george. **He hears again her voice.** Prince frederick leopold albert louis. Prince william william rupert. Prince augustus ernest william. **He leads their twins.**

Girl has clothes. Girl lives in dress. Girl know she can dress. Girl know still know she sees. **Her wear rags.** Girl not smile and talk. Girl never gets bad hair. Girl and most rarely smiles. **She do kinds of hard work.** Girl love to boy cry. Girl love girl get laugh. Girl do what will girls cry. **Girl go cry to God.**

Example stories generated by C2PO.

The earlier-mentioned work on narrative psychology established the linkage between story coherence and question-answering. Another approach to story generation is thus to treat a language model as a resource to be queried in order to answer questions about a story world — the answers to these carefully chosen questions become the content of the story itself. Castricato et al (3rd Workshop on Narrative understanding) explore this. They generate a story backwards, starting with a sentence about the end of a story. They

generate some plausible questions about how the story might have gotten to that ending. They train an explanation generation language model to answer the question and then make the resulting explanation the previous segment of the story. They then repeat this process using the last fragment as the source of questions. Theoretically, this results in more coherent stories because each segment added to the story explains what comes next.

He needs to find a way out of the house. If he wants to go to the kitchen, he must first find a way to escape from the house. This means that he must either run away or fight his way past the lock. He goes to the bathroom. The only way to get to the kitchen is through a locked door, which Hansel has never been able to do before. However, when he opens the door, it doesn't take him very long at all. **Hansel's hand still trembles as he pushes open the twice-cooked door. The last time he saw the house he was glancing back over his shoulder as he and his sister fled into the trees.**
If it does not have a pulse then it is dead. The heart rate of an animal tells them if it is alive or dead. There is no such thing as absolute death. He is able to tell the difference between something being alive and something being dead, so when he looks at the house, he feels like he's seeing things that aren't there. This makes him feel uncomfortable because he doesn't want to be in that situation. It's similar to how people can see ghosts or monsters from inside their head but they don't know what those things are. **Hansel's hand still trembles as he pushes open the twice-cooked door. The last time he saw the house he was glancing back over his shoulder as he and his sister fled into the trees.**

Stories generated via question-answering. The bold text is given input.

7. Conclusions

The field of automated story generation has gone through many phase shifts, perhaps none more significant than the phase shift from non-learning story generation systems to machine learning based story generation systems (neural networks in particular). Symbolic story generation systems were capable of generating reasonably long and coherent stories. These systems derived much of their power from well-formed knowledge bases. But these knowledge bases had to be structured by hand, which limited what the systems could generate. When we shifted to neural networks, we gained the power of neural networks to acquire and make use of knowledge from corpora. Suddenly, we could build story generation systems that could generate a larger space of stories about a greater range of topics. But we also set aside a lot of what was known about the psychology of readers and the ability to reason over rich knowledge structures to achieve story coherence. Even increasing the size of neural language models has only delayed the inevitability of coherence collapse in stories generated by neural networks.

A primer such as this one makes it easier to remember the paths that were trodden previously in case we find opportunities to avoid throwing the baby out with the bath water. This is not to say that machine learning or neural network based approaches should not be pursued. If there was a step backward it was because doing so gave us a powerful new tool with the potential to take us further ahead. The exciting thing about working on automated story generation is that we genuinely don't know the best path forward. There is a lot of room for new ideas.

Artificial Intelligence Machine Learning