Computer-Assisted Authoring of Interactive Narratives

Mubbasir Kapadia[1,3] Jessica Falk[2] Fabio Zünd[2] Marcel Marti[2] Robert W. Sumner[1,2] Markus Gross[1,2]
[1]Disney Research Zurich [2]ETH Zurich [3]Rutgers University

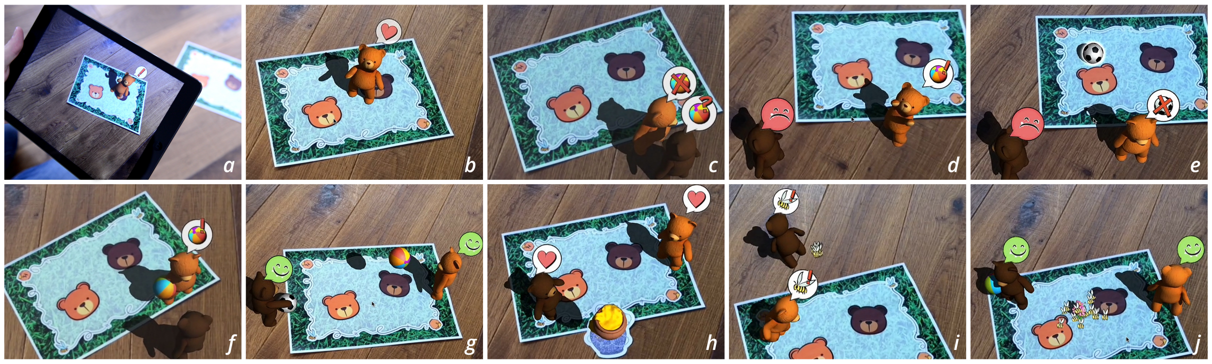


Figure 1: *We present a computer-assisted authoring tool for interactive narratives. Example narrative: (a,b) The player can freely interact with characters in the authored story to dictate story progression. (c) The bear is distracted when his friend enters the scene and asks him to play ball. (d) When no ball is to be found, the first bears turns to the player for help. (e) The bears aren't happy with the soccer ball and only want to play with the beach ball. (f, g) With the players help, the first bear gives his friend the ball so they can play a game of catch. (h) At any time, the player can use different interactions (e.g., adding a honeypot into the scene) to branch the story in a different direction. (i,j) Spawning bees wreaks havoc on the bears and the player must use flowers to distract the bees.*

Abstract

This paper explores new authoring paradigms and computer-assisted authoring tools for free-form interactive narratives. We present a new design formalism, Interactive Behavior Trees (IBT's), which decouples the monitoring of user input, the narrative, and how the user may influence the story outcome. We introduce automation tools for IBT's, to help the author detect and automatically resolve inconsistencies in the authored narrative, or conflicting user interactions that may hinder story progression. We compare IBT's to traditional story graph representations and show that our formalism better scales with the number of story arcs, and the degree and granularity of user input. The authoring time is further reduced with the help of automation, and errors are completely avoided. Our approach enables content creators to easily author complex, branching narratives with multiple story arcs in a modular, extensible fashion while empowering players with the agency to freely interact with the characters in the story and the world they inhabit.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games;

Keywords: interactive narrative, behavior trees, computer-assisted authoring

1 Introduction

Interactive narratives strive to offer immersive digital experiences in which users create or influence a dramatic storyline through their actions in interactive virtual worlds. The far-reaching goal is to immerse users in a virtual world where they become an integral part of an unfolding narrative and can significantly alter the story's outcome through their actions.

Traditional linear narratives provide little user agency to influence the outcome of the story. Computer games often use linear plots interspersed with isolated interactive segments, with all users experiencing the same plot during successive sessions. Branching narratives [Gordon et al. 2004], where the narrative outcome depends on the user's decisions provide a discrete set of choices that influence the story. The authoring complexity of these approaches grows exponentially with the number of story arcs, the number of interaction possibilities, and the granularity of interaction. Story arcs are tightly coupled and new interactions require monolithic changes where the authoring complexity is kept tractable only by severely limiting user agency to discrete choices at key points in the story. Hence, traditional interactive narrative applications such as games either provide strong narrative experiences with limited user agency or provide compelling interactive experiences with simple narrative structure.

Creating an authoring platform that enables content creators to author free-form interactive narratives with multiple story arcs where the players can influence the narrative outcome, while using automation to facilitate the authoring process and not hinder it, is the main goal of this work. There are two main challenges that we face: (1) An appropriate language for authoring interactive narratives that scales with story complexity, and freedom of interaction. (2) Integrated automation solutions to facilitate the story authoring process without sacrificing author control.

In this paper, we explore the use of Behavior Trees (BT) [Isla 2005] for authoring free-form interactive narratives. The hierarchical, graphical nature of BT's is ideal for authoring complex, branching narratives in a modular fashion. However, traditional BT formalisms are not suitable for handling state persistence and free-form user interaction, which are essential for interactive narratives. To meet these requirements, we extend the BT formalism to decouple the monitoring of user input, the narrative, and how the user may influence the story outcome. This decoupling provides users much greater freedom in their ability to interact with the characters in the story and the world they inhabit, and empowers content creators to create compelling narrative experiences with free-form user interaction. Independent of the specification language, the author has to consider all possible responses for each interaction during any point of the narrative to provide complete freedom of user interaction. To make this problem tractable, we provide a suite of automation tools to detect and resolve story inconsistencies, as well as potential conflicts where user interactions may invalidate the story plot. This empowers the author to truly focus on authoring stories, and rely on automation to make the problem of integrating interactivity tractable. Additionally, the same automation tools can be used to dynamically detect and repair stories during an actual play session, in response to unforeseen user intervention that was not detected during authoring.

We demonstrate the potential of IBT's and automation by authoring an interactive narrative for an AR application. Our approach enables content creators to easily author complex, branching narratives with multiple story arcs in a modular, extensible fashion while empowering players with the agency to freely interact with the characters in the story and the world they inhabit.

2 Related Work

We refer the readers to a comprehensive surveys on narrative authoring [Riedl and Bulitko 2013; Kapadia et al. 2013] and provide a brief review below.

Manual Authoring. Scripted approaches [Loyall 1997] describe behaviors as pre-defined sequences of actions where small changes often require far-reaching modifications of monolithic scripts. Approaches such as Improv [Perlin and Goldberg 1996] and LIVE [Menou 2001] describe behaviors as rules which govern how actors act based on certain conditions. These systems produce pre-defined behaviors corresponding to the current situation, and are not designed to generate complicated agent interactions with narrative significance. Facade [Mateas and Stern 2003] executes authored beats to manage the intensity of the story and uses a generalized scripting language [Mateas and Stern 2004] for manually authoring character interactions by encoding their preconditions for successful execution.

Story Graphs [Gordon et al. 2004] accommodate user interaction as discrete choices at key points in the authored narrative. Behavior Trees (BT's) are gaining popularity in the computer gaming industry for designing the artificial intelligence logic for non-player characters [Isla 2005]. BT's offer graphical constructs for authoring modular, extensible behaviors, which can be extended to control multiple interacting characters [Shoulson et al. 2014]. For communication between nodes, BT's rely on a blackboard [Millington and Funge 2009], which is a centralized, flat repository of data that can be accessed by nodes in the tree.

Automated Narrative. The use of domain-independent planners [Sacerdoti 1975] is a promising direction for automated narrative synthesis – but requires the specification of domain knowledge. The work in [Kapadia et al. 2011b; Kapadia et al. 2011a] synthesizes complex multi-actor interactions while conforming to narrative constraints, which cannot be dynamically changed to accommodate user input. Narrative mediation systems [Riedl and Young 2006] builds on top of traditional story graph representations by considering the ramifications of possible user interactions, and automatically synthesizing sub stories to accommodate them. This produces story graphs with high branching that are difficult to edit by humans.

Virtual directors or drama managers [Magerko et al. 2004] are responsible for steering agents towards pre-determined narrative goals [Weyhrauch 1997] while accommodating user input. Thespian [Si et al. 2005] uses decision-theoretic agents to create actors with social awareness, while PaSSAGE [Thue et al. 2007] guides a player through predefined encounters based on the system's estimation of the player's ideal experience. Event-centric planning [Shoulson et al. 2013a; Shoulson et al. 2013b] plans in the space of pre-authored narratively significant interactions, thus mitigating the combinatorial explosion of planning in the action space of individual character actions.

Comparison to Prior Work. Intelligent systems monitor the fictional world and intervene to drive the narrative forward, thus effectively replacing the human author. Since the underlying narrative still lies within traditional branching representations, it becomes intractable for the human author to iteratively build upon or replace automatically generated content. Previous work either provides complete authorial control or relies on automation to synthesize emergent stories with minimal specification. In contrast, our goal is to empower content creators to author compelling free-form interactive narratives, by leveraging automation to facilitate the creative process instead of replacing it.

3 Authoring Interactive Narratives

An interactive narrative is traditionally represented as a branching story graph where the vertices correspond to story atoms during which the user has no outcome on the narrative, and the directed edges represent a discrete set of choices, which allow the user to influence the story outcome. To provide the user a dramatic storyline in which he can heavily influence the progression and outcome of the narrative, it is important to offer many decision points and a high branching factor. However, increasing the involvement of the user also heavily increases the combinatorial complexity of authoring such a story graph. We identify three main requirements towards authoring free-form interactive narrative experiences:

1. **Modular Story Definition.** Complex interactive narratives have many interconnected story arcs that are triggered based on user input leading to widely divergent outcomes. The complexity of authoring narratives must scale linearly with the number of story arcs, which can be defined in a modular and independent fashion.

2. **User Interactions.** User interaction should be free-form, and not limited to discrete choices at key stages of the story, with far-reaching ramifications on the outcome of the narrative. Monitoring user input and story logic should be decoupled to facilitate the modification of user interactions without requiring far-reaching changes to the story definition.

3. **Persistent Stories.** The actions and interactions between the user and characters over the entire course of the narrative must persist and influence story progression.

3.1 Interactive Behavior Trees

Behavior Trees (BT's) provide a graphical paradigm for authoring complex narratives in a modular and extensible fashion. Story arcs can be independently authored as subtrees and then connected together using BT control nodes to author branching narratives. Recent extensions [Shoulson et al. 2014] facilitate the authoring of complex multi-actor interactions in a parametrizable fashion, enabling the reuse of modular plot elements, and ensures that the complexity of the narrative scales independently of the number of characters. These properties of BT's make them ideally suited for authoring complex, branching narratives (Requirement 1). However, BT's cannot easily handle free-form interactions (Requirement 2) and don't have any means of explicitly storing the past state of characters involved in the narrative (Requirement 3). These challenges are described in the supplementary document in detail.

To meet these requirements, we introduce a new BT design formalism that facilitates free-form user interaction and state persistence. Interactive Behavior Trees (IBT's), as illustrated in Fig. 2(a) are divided into 3 independent sub-trees that are connected using a Parallel control node. An IBT $\mathbf{t}_{\mathrm{IBT}} = \langle \mathbf{t}_{\mathrm{ui}}, \mathbf{t}_{\mathrm{state}}, \mathbf{t}_{\mathrm{narr}} = \{\mathbf{t}_i^{\mathrm{arc}} | \mathbf{t}_1^{\mathrm{arc}} \ldots \mathbf{t}_m^{\mathrm{arc}}\}, \beta \rangle$ where: (1) $\mathbf{t}_{\mathrm{narr}}$ is the narrative definition with modular story arcs $\{a_i\}$, each with their own independent subtree $\{\mathbf{t}_i^{\mathrm{arc}}\}$. (2) $\mathbf{t}_{\mathrm{ui}}$ processes the user interactions. Fig. 2(b) illustrates the story subtree. (3) $\mathbf{t}_{\mathrm{state}}$ monitors the state of the story to determine if the current story arc needs to be changed. Fig. 2(b) illustrates the story subtree. (4) The blackboard β stores the state of the story and its characters. (5) A fourth subtree $\mathbf{t}_{\mathrm{cr}}$ is added for conflict resolution, and will be described in § 4.

Story Definition. $\mathbf{t}_{\mathrm{narr}}$ is responsible for handling the narrative progression and is further subdivided into subtrees that represent a separate story arc. Fig. 2(b) provides an example of $\mathbf{t}_{\mathrm{narr}}$ while Fig. 2(c) illustrates each arc definition $\mathbf{t}^{\mathrm{arc}}$, which is encapsulated as a separate subtree. This introduces an assertion node, which is checked at every frame whether the current arc is still active before proceeding with its execution. This minor extension to the story arc definition allows the story to instantaneously switch arcs at any moment in response to the user's interactions.

Monitoring User Input. $\mathbf{t}_{\mathrm{ui}}$ monitors the different interactions that are available to the user and can be easily changed depending on the application or device. Once an input is detected, it sets the corresponding state in the blackboard β, which is queried by $\mathbf{t}_{\mathrm{state}}$ to determine the current state of the story, and the active story arc. Since $\mathbf{t}_{\mathrm{ui}}$ is executed in parallel with the other subtrees, we are able to immediately respond and register the interactions of the user and use it to influence the narrative outcome. Fig. 2(d) illustrates an example.

Monitoring Story State. $\mathbf{t}_{\mathrm{state}}$ contains separate subtrees for each story arc, which checks if the precondition for the particular arc is satisfied. If so, β is updated to reflect the newly activated story arc, which is used to switch the active story in $\mathbf{t}_{\mathrm{narr}}$. Fig. 2(e,f) illustrates $\mathbf{t}_{\mathrm{state}}$ and a subtree used for checking the preconditions for an example story arc. It may be possible for the preconditions of multiple story arcs to be satisfied at any instance, in which case the story arcs are activated in order of priority (the order in which they appear in $\mathbf{t}_{\mathrm{narr}}$). It is also possible for multiple story arcs to be active simultaneously if they are operating on mutually exclusive characters and objects.

Message Passing and State Persistence. The overall design of the IBT results in three subtrees that execute independently in parallel with one another. The blackboard β stores internal state variables (e.g., the current active story arc) to facilitate communication between the subtrees, and maintains state persistence. $\mathbf{t}_{\mathrm{ui}}$ updates β when any input signal is detected. Tree $\mathbf{t}_{\mathrm{state}}$ monitors β to check if the preconditions of a particular story arc are satisfied, and updates the current arc. Finally, each arc subtree in $\mathbf{t}_{\mathrm{narr}}$ checks if it is the current active arc before continuing. Also, the user input and the narrative execution can update the story and character state to influence the progression of the narrative at a later stage.

4 Automation Tools for Authoring Interactive Narratives

Independent of the specification language (story graphs, behavior trees or any other authoring paradigm), interactive narratives require the author to consider the ramifications of all possible user interactions at all points in the story, which is prohibitive for complex stories with many different interaction modalities. To address this challenge, we introduce a suite of automation tools that exploit domain knowledge to automatically identify and resolve invalid story specifications (§ 4.3), potential user actions that may invalidate story arcs (§ 4.4, § 4.5), and even automatically synthesize complete stories (§ 4.6).

4.1 Domain Knowledge

In order to use automated planning tools to facilitate the authoring process, we need to add additional domain knowledge, which can be used by an intelligent system for inference. This includes annotating semantics that characterize the attributes and relationships of objects and characters in the scene (state), different ways in which they interact (affordances), and how these affordances manipulate their state. Our framework is no different from other intelligent systems [Riedl and Bulitko 2013] in this regard. However, the cost of specifying domain knowledge is greatly mitigated by the ability of computational tools to consider all possible interactions between smart objects, how user input may change story state, and use it to detect and resolve invalid stories. Below we describe our representation of domain knowledge, which balances ease of specification and efficiency of automation.

Smart Objects. The virtual world $\mathbf{W}$ consists of smart objects with embedded information about how an actor can use the object. We define a smart object $w = \langle \mathbf{F}, s \rangle$ with a set of advertised affordances $f \in \mathbf{F}$ and a state $s = \langle \theta, R \rangle$, which comprises a set of attribute mappings θ, and a collection of pairwise relationships R with all other smart objects in $\mathbf{W}$. An attribute $\theta(i,j)$ is a bit that denotes the value of the j^{th} attribute for w_i. Attributes are used to identify immutable properties of a smart object such as its role (e.g., a ball or a bear) which never changes, or dynamic properties (e.g., `IsHappy`) which may change during the story. A specific relationship R_a is a sparse matrix of $|\mathbf{W}| \times |\mathbf{W}|$, where $R_a(i,j)$ is a bit that denotes the current value of the a^{th} relationship between w_i and w_j. For example, an `IsFriendOf` relationship indicates that w_i is a friend of w_j. Note that relationships may not be symmetric, $R_a(i,j) \neq R_a(j,i) \ \forall \ (i,j) \in |\mathbf{W}| \times |\mathbf{W}|$. Each smart object's state is stored as a bit vector encoding both attributes and relationships. The overall state of the world $\mathbf{W}$ is defined as the compound state $\mathbf{s} = \{s_1, s_2 \cdots s_{|\mathbf{W}|}\}$ of all smart objects $w \in \mathbf{W}$, which is encoded as a matrix of bit vectors. $\mathbf{s}_{\mathbf{w}}$ denotes the compound state of a set of of smart objects $\mathbf{w} \subseteq \mathbf{W}$.

Affordances. An affordance $f = \langle w_o, \mathbf{w}_u, \mathbf{\Phi}, \Omega \rangle$ is an advertised capability offered by a smart object that takes the owner of that affordance w_o and one or more smart object users $\mathbf{w}_u$, and manipulates their states. For example, a smart object such as a ball can advertise a *Throw* affordance, allowing another smart object to throw it. A precondition $\mathbf{\Phi} : \mathbf{s}_{\mathbf{w}} \leftarrow \{\mathtt{TRUE}, \mathtt{FALSE}\}$ is an expression in conjunctive normal form on the compound state $\mathbf{s}_{\mathbf{w}}$ of $\mathbf{w} : \{w_o, \mathbf{w}_u\}$ that checks if f can be executed based on their current states. A precondition is fulfilled by $\mathbf{w}$ if $\mathbf{\Phi}_f(\mathbf{w}) = \mathtt{TRUE}$. The postcondition $\Omega : \mathbf{s} \rightarrow \mathbf{s}'$ transforms the current state of all participants, $\mathbf{s}$ to $\mathbf{s}'$ by executing the effects of the affordance. When an affordance fails, $\mathbf{s}' = \mathbf{s}$. Fig. 3 describes the general definition of an affordance.

An affordance instance $\mathbf{h} = \langle f, \mathbf{w} \rangle$ includes a set of smart objects

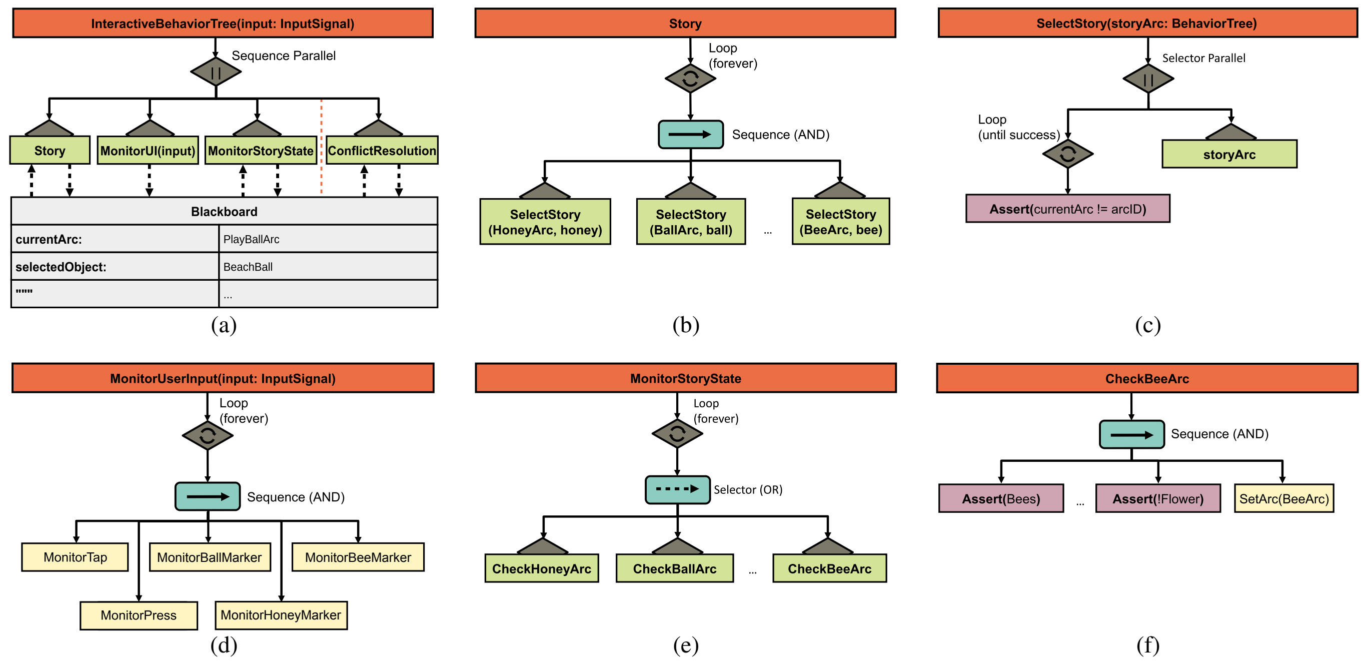


Figure 2: *(a) Design formalism of Interactive Behavior Trees (IBT's) with decoupled specification of user input, narrative definition, and the impact of user input on story state. (b) Narrative subtree with modular story arcs. (c) Each story arc definition is encapsulated in its own independent subtree, which first checks if this is the current active arc before proceeding with the narrative execution. (d) Subtree to monitor user input. (e) Subtree that changes story state based on user input, which triggers branches in story arc. (f) An example subtree from (e) which checks if all the preconditions for a particular story arc are satisfied before setting it as the current active arc.*

$\mathbf{w} \subset \mathbf{W}$ such that $\mathbf{\Phi}_f(\mathbf{s_w}) = \texttt{TRUE}$. To map affordance instances as leaf nodes of a BT, execution of an affordance returns a status that takes three possible values. It returns *Running* if the affordance is still executing. If it *succeeds*, the postconditions Ω_f are applied to the state of all smart object participants. If it *fails*, there is no change in state. This ensures that affordances are considered as atomic units.

Affordance $f\,(w_o, \mathbf{w}_u)$:
Precondition $\mathbf{\Phi}_f$**:**
CNF expression on compound state $\mathbf{s}$ of w_o and $\mathbf{w}_u$
Postcondition Ω_f**:**
Change in state of w_o and $\mathbf{w}_u$ after successfully executing f
Status γ**:**
Running: Continue executing f
Success: $\mathbf{s}' = \Omega_f(\mathbf{s})$;
Failure: $\mathbf{s}' = \mathbf{s}$;

Figure 3: *Affordance definition.*

User Interactions. We define a set of user interactions $u \in \mathbf{U}$, which define the different ways in which a user can interact with smart objects in the world $\mathbf{W}$. User interactions are treated as special kinds of affordances where the user is one of the affordance participants. This allows any underlying planning framework to accommodate user interactions during the planning process.

4.2 Problem Definition

Given the terminology defined in § 4.1, we define a general problem description $\mathbf{P} = \langle \mathbf{s}_0, \mathbf{\Phi}_g, \mathbf{A} \rangle$ consisting of an initial state $\mathbf{s}_0$, a set of preconditions to satisfy the goal state $\mathbf{\Phi}_g$, and the set of affordance instances $\mathbf{A} = \{\mathbf{h}_i\}$, which may include instances of user interactions. The problem instance $\mathbf{P}$ will be defined in a variety of ways to resolve inconsistencies in the story, or potential conflicts, described in the remainder of this section.

Causal Links. We introduce the concept of a causal link to symbolize a connection between two affordance instances such that executing the postconditions of one affordance satisfies a clause in the preconditions of the other. Causal links are represented as $\mathbf{l} = \langle \mathbf{h}_1, \phi_2^i, \mathbf{h}_2 \rangle$. ϕ_2^i defines the i^{th} clause in $\mathbf{\Phi}_2$ such that $\phi_2^i(\Omega_1(\mathbf{s})) = \texttt{TRUE}$.

We interpret $\mathbf{P}$ as a search problem in the space of possible partial plans. We define a partial plan $\pi = \langle \mathbf{H}, \mathbf{\Phi}_{open}, \mathbf{L}, \mathbf{O} \rangle$ where $\mathbf{H}$ is the set of affordance instances currently in π, $\mathbf{\Phi}_{open}$ is a set of pairs $\phi_{open} = \langle \mathbf{h}, \phi_\mathbf{h} \rangle$ where $\mathbf{h} \in \mathbf{H}$ and $\phi_\mathbf{h}$ defines one condition in the precondition expression $\mathbf{\Phi_h}$. $\mathbf{L}$ defines the set of all causal links between pairs of affordance instances in $\mathbf{H}$, and $\mathbf{O}$ defines a set of transitive and asymmetric partial orderings of affordance instances $\{\mathbf{h}_i \prec \mathbf{h}_j\}$ representing a "before" relation, where $\mathbf{h}_i, \mathbf{h}_j \in \mathbf{H}$. This means that $\mathbf{h}_i$ must occur before $\mathbf{h}_j$ in the partial order plan. A partial plan π_p is a plan that has not yet satisfied all open preconditions: $|\mathbf{\Phi}_{open}| > 0$, while a complete plan π_c has no open preconditions: $\mathbf{\Phi}_{open} = \emptyset$.

Partial Order Planning. We use a partial order planner (POP) [Sacerdoti 1975] to compute a plan $\pi_\mathrm{c} = \mathbf{Plan}(\mathbf{P})$ that generates an ordering of affordance instances from $\mathbf{s}_0$ which satisfies the preconditions $\mathbf{\Phi}_g$. While POP requires more computational power for processing a single node, it has been shown to outperform total-order planning (TOP) approaches [Pearl 1984] when dealing with goals that contain subgoals [Minton et al. 1994], allowing the planner to efficiently operate in the search space of partial plans. POP employs the Principle of Least Commitment where affordance execution is ordered only when strictly needed to ensure a consistent plan. This ensures efficiency when dealing with problems where there may exist multiple possible solutions that differ only in their ordering of affordance execution. In contrast, TOP strictly sequences actions when finding a solution. POP is also able to efficiently work for problem definitions where the goal state is partially specified – containing only the desired preconditions that must be satisfied. We provide a brief overview of the algorithm below.

At each iteration, POP selects a clause $\phi_{open} = \langle \mathbf{h}_i, \phi_i \rangle$ from the set of open preconditions $\mathbf{\Phi}_{open}$ and chooses an affordance instance $\mathbf{h}_j \in \mathbf{A}$ that satisfies ϕ_i. If $\mathbf{h}_j$ is not already present, it is inserted into the partial plan π_{p}. $\mathbf{h}_j$ must execute before $\mathbf{h}_i$, which is specified by adding a causal link $\mathbf{l} = \langle \mathbf{h}_j, \phi_i, \mathbf{h}_i \rangle$. Any instance $\mathbf{h} \in \mathbf{H}$ that contradicts ϕ_i must happen either before $\mathbf{h}_j$ or after $\mathbf{h}_i$, and is resolved by introducing additional causal links, as defined by the method **Protect**(). If $\mathbf{h}_j$ is added for the first time, its preconditions are added to $\mathbf{\Phi}_{open}$, and the process continues until all preconditions are satisfied: $\mathbf{\Phi}_{open} = \emptyset$. Alg. (1) outlines the details of the algorithm and we refer the readers to [Sacerdoti 1975] for additional details.

Algorithm 1 Partial Order Planner

```
function Plan ($\mathbf{P} = \langle \mathbf{s}_0, \mathbf{\Phi}_g, \mathbf{A} \rangle$)
  $\mathbf{\Phi}_{open} = \{\langle \mathbf{h}_{\mathbf{\Phi}_g}, \phi \rangle | \forall \phi \in \mathbf{\Phi}_g\}$
  $\mathbf{H} = \{\mathbf{h}_{\mathbf{s}_0}, \mathbf{h}_{\mathbf{\Phi}_g}\}$
  $\mathbf{O} = \{\mathbf{h}_{\mathbf{s}_0} \prec \mathbf{h}_{\mathbf{\Phi}_g}\}$
  $\mathbf{L} = \emptyset$
  while $\mathbf{\Phi}_{open} \neq \emptyset$ do
    $\langle \mathbf{h}_c, \phi_c \rangle$ = SelectAndRemoveCondition($\mathbf{\Phi}_{open}$)
    if $\phi_c(\mathbf{h}) == \texttt{FALSE} \; \forall \; \mathbf{h} \in \mathbf{H}$ then
      $\mathbf{h}_s = \exists \mathbf{h} \in \mathbf{A}$ s.t. $\phi_c(\mathbf{h}) = \texttt{TRUE}$
      $\mathbf{H} = \mathbf{H} \cup \mathbf{h}_s$
      $\mathbf{O} = \mathbf{O} \cup (\mathbf{h}_{\mathbf{s}_0} \prec \mathbf{h}_s)$
      for all $\mathbf{l} \in \mathbf{L}$ do
        $\mathbf{O}$ = Protect($\mathbf{l}, \mathbf{h}_s, \mathbf{O}$)
      $\mathbf{\Phi}_{open} = \mathbf{\Phi}_{open} \cup \{\langle \mathbf{h}_s, \phi_s \rangle | \; \forall \; \phi_s \in \mathbf{\Phi}_s\}$
    else
      $\mathbf{h} = \exists \; \mathbf{h} \in \mathbf{H}$ s.t. $\phi(\mathbf{h}) = \texttt{TRUE}$
    $\mathbf{O} = \mathbf{O} \cup (\mathbf{h}_s \prec \mathbf{h}_c)$
    $\mathbf{L} = \mathbf{L} \cup \langle \mathbf{h}_s, \phi_c, \mathbf{h}_c \rangle$
    for all $\mathbf{h} \in \mathbf{H}$ do
      $\mathbf{O}$ = Protect($\langle \mathbf{h}_s, \phi_c, \mathbf{h}_c \rangle, \mathbf{h}, \mathbf{O}$)
  $\pi = \langle \mathbf{H}, \mathbf{\Phi}_{open}, \mathbf{L}, \mathbf{O} \rangle$
  return $\pi$
```

Integrating Plan into IBT. The plan π_{c} generated by POP represents an ordering $\mathbf{O}$ of affordance instances $\mathbf{H}$, which can be easily integrated into an existing behavior tree definition by choosing appropriate control nodes that constrain the order in which affordances in the plan can be executed. Fig. 4 illustrates an example of how a plan is converted into its corresponding BT definition.

4.3 Narrative Consistency

A consistent narrative does not violate the preconditions of any affordance instances that are executed during the narrative. A story author may not consider all possible preconditions when defining a story, leading to the definition of an inconsistent story. We define an inconsistent node $\mathbf{t}$ in a story arc as an affordance instance $\mathbf{h}_t$ associated with $\mathbf{t}$ such that $\mathbf{\Phi}_t(\mathbf{s_t}) = \texttt{FALSE}$ where $\mathbf{s_t}$ is the compound state of smart objects in the scene obtained by executing all nodes in the IBT leading to $\mathbf{t}$.

Belief States. Interactive narratives authored using IBT's can branch in many directions, depending on user interaction and the execution trace of nodes in the IBT. Hence, there may be many possible states that the smart objects are in at a current node $\mathbf{t}$. Therefore, we introduce the notion of a belief state $\mathbf{b_t} = \{\mathbf{s}^1_{\mathbf{t}}, \mathbf{s}^2_{\mathbf{t}} \cdots \mathbf{s}^n_{\mathbf{t}}\}$, which represents a set of partially specified states of all smart objects that may be reached at $\mathbf{t}$ due to different possible execution traces of the IBT. A partially specified state may contain attributes whose values cannot be determined. By considering the belief state of all possible executions of the narrative that led to $\mathbf{t}$, we can determine whether the preconditions of an affordance instance might be violated by any possible execution of the story arc. The supplementary document details the static analysis of different types of nodes in an IBT definition to compute the belief state at each node.

Inconsistency Detection and Resolution. When an inconsistent node $\mathbf{t}$ is detected in the story definition $\mathbf{t}_{\mathrm{narr}}$ of an authored IBT $\mathbf{t}_{\mathrm{IBT}}$, we compute the belief state $\mathbf{b}$ of all possible states that could arise from different execution traces of $\mathbf{t}_{\mathrm{IBT}}$ up to $\mathbf{t}$. For each of these states $\mathbf{s} \in \mathbf{b}$, we define a problem instance $\mathbf{P} = \langle \mathbf{s}, \mathbf{\Phi_t}, \mathbf{A} \rangle$ and generate a plan π to add additional nodes in the tree such that $\mathbf{\Phi_t}$ is satisfied. Alg. (2) outlines the algorithm for inconsistency detection and resolution.

Algorithm 2 Inconsistency Detection and Resolution

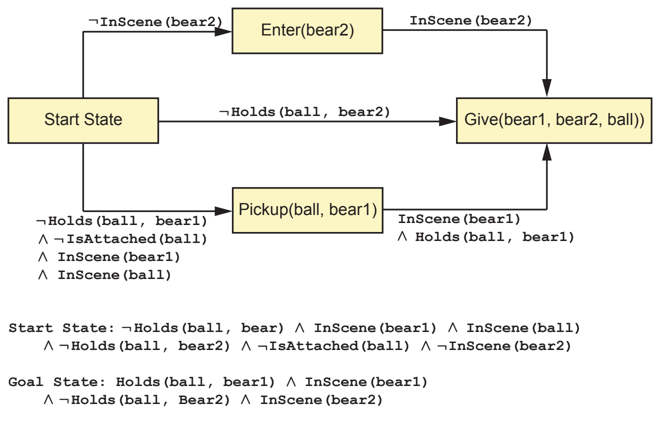
```
function DetectAndResolveInconsistencies ($\mathbf{t}_{\mathrm{IBT}}$)
  for all $\mathbf{t}^{\mathrm{arc}} \in \mathbf{t}_{\mathrm{narr}}$ do
    for all $\mathbf{t} \in \mathbf{t}^{\mathrm{arc}}$ do
      $\mathbf{b}$ = ComputeBeliefState($\mathbf{t}, \mathbf{t}_{\mathrm{IBT}}$)
      for all $\mathbf{s} \in \mathbf{b}$ do
        if $\mathbf{\Phi_t}(\mathbf{s}) == \texttt{FALSE}$ then
          $\mathbf{P} = \langle \mathbf{s}, \mathbf{\Phi_t}, \mathbf{A} \rangle$
          $\pi$ = Plan($\mathbf{P}$)
          $\mathbf{t}_{\mathrm{IBT}}$ = IntegratePlan($\pi, \mathbf{t}, \mathbf{t}_{\mathrm{IBT}}$)
  return $\mathbf{b}$
```

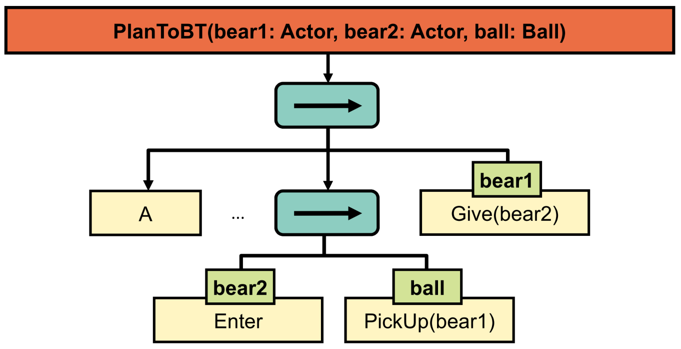
4.4 Conflicts

The players actions may invalidate the successful execution of consistent narratives, and the author must consider the ramifications of all possible interactions at all possible points in the narrative definition. In order to make this problem tractable, we present automation tools that automatically detect potential user interactions that may invalidate affordance preconditions at any stage in the narrative, and provide resolution strategies to accommodate user interference, while still ensuring that the narrative is able to proceed down the intended path.

Before we define a conflict, we first differentiate between two sets of causal links. A link $\mathbf{l} = \langle \mathbf{h}_1, \phi^i_2, \mathbf{h}_2 \rangle$ is *active* if the affordance instance $\mathbf{h_t}$ associated with the current node $\mathbf{t}$ has the following ordering: $\mathbf{h}_1 \prec \mathbf{h_t} \preceq \mathbf{h}_2$. These include all the links that are active when considering the current node $\mathbf{t}$. Keeping track of active causal links allows us to maintain a list of conditions on state attributes which may not be violated by any user interactions while executing $\mathbf{t}$. A link is *needed* to ensure the progression of the narrative at a particular node $\mathbf{t}$ in the IBT if $\mathbf{h}_1 \preceq \mathbf{h_t} \prec \mathbf{h}_2$. These include links that are active *after* the execution of the current node $\mathbf{t}$, and determine conditions on attributes which need to be met even after the execution of an user interaction, to ensure progression of the narrative.

Conflicts. This allows us to formally define a conflict $\mathbf{c}$ as a pair $\langle u, \mathbf{l} \rangle$ where $\mathbf{l} = \langle \mathbf{h}_i, \phi^i_j, \mathbf{h}_j \rangle$ is an active causal link, such that if the user performs a particular interaction $u \in \mathbf{U}$ during the execution of $\mathbf{h}_i$, ϕ^i_j may be violated. Conflicts are detected at a particular node $\mathbf{t}$ if any active causal links at $\mathbf{t}$ are violated and can be resolved by generating a plan that satisfies the conditions of all needed links.


¬InScene(bear2)
Enter(bear2)
InScene(bear2)
Start State
¬Holds(ball, bear2)
Give(bear1, bear2, ball))
Pickup(ball, bear1)
¬Holds(ball, bear1) ∧ ¬IsAttached(ball) ∧ InScene(bear1) ∧ InScene(ball)
InScene(bear1) ∧ Holds(ball, bear1)
Start State: ¬Holds(ball, bear) ∧ InScene(bear1) ∧ InScene(ball) ∧ ¬Holds(ball, bear2) ∧ ¬IsAttached(ball) ∧ ¬InScene(bear2)
Goal State: Holds(ball, bear1) ∧ InScene(bear1) ∧ ¬Holds(ball, Bear2) ∧ InScene(bear2)
(a)



(b)

Figure 4: *(a) Illustrates a sample plan constructed by POP. The edges represent the causal links between the different affordances. (b) A concrete mapping of the plan to a BT.*

Conflicts can be handled in two ways: (1) **Accommodation**. We allow the user to interfere with the narrative by successfully executing u such that $\mathbf{h}$ fails. In this case, we need to generate a conflict resolution strategy that is able to accomplish the same result, as executing $\mathbf{h}$. (2) **Interference**. The affordance instance $\mathbf{h}$ is successfully executed and u fails. No plan is needed in this case. It is up to the author to decide whether to accommodate or interfere for a particular conflict. For conflicts where no plan is possible, we are limited to interference where the user interaction is perceived to be unsuccessful.

Conflict Resolution Subtree. We add a new subtree into the IBT formalism $\mathbf{t}_{\text{cr}}$ that is automatically populated and contains the conflict resolution strategies (plans) for all potential conflicts. During narrative execution, whenever a conflict occurs, control is transferred to the corresponding subtree in $\mathbf{t}_{\text{cr}}$ that contains the plan for resolving that particular conflict.

Conflict Detection and Resolution. Alg. (3) provides algorithmic details of detecting and resolving conflicts at a particular node $\mathbf{t}$ in the IBT $\mathbf{t}_{\text{IBT}}$. We check if any interaction violates the active links at that node. For a potential conflict $\mathbf{c} = \langle u, \mathbf{l}\rangle$, we consider the belief state $\mathbf{b}$ up to the execution of the current node $\mathbf{t}$ in the IBT. For each state $\mathbf{s} \in \mathbf{b}$, we define a problem instance $\mathbf{P} = \langle \mathbf{s}_0 = \Omega_u(\mathbf{s}), \mathbf{\Phi}_g = \mathbf{\Phi}_{\text{needed}}\rangle$, where $\mathbf{\Phi}_{\text{needed}}$ are the combined conditions of all needed links. A plan π is generated for $\mathbf{P}$ and inserted into the conflict resolution subtree $\mathbf{t}_{\text{cr}}$ to accommodate u. If no plan is found, then we choose to interfere where u is said to fail. The appropriate conflict resolution strategy is added into $\mathbf{t}_{\text{cr}}$.

Dynamic Conflict Detection and Resolution. Static analysis of the IBT is not able to detect *all possible* conflicts that may occur during execution of the narrative. In particular, we cannot detect conflicts (1) that occur while executing nodes in the conflict resolution subtree, (2) due to user actions in one story arc that violate the preconditions of nodes in another story arc. These unforeseen conflicts can be handled by using a modified version of Alg. (3) during the execution of the narrative to dynamically detect and resolve conflicts. This works well in practice as only a small number of conflicts remain undetected during static analysis and the algorithm for conflict resolution is very efficient and able to instantly generate plans for reasonably complex problem domains.

Algorithm 3 Conflict Detection and Resolution

```
function DetectAndResolveConflicts ($\mathbf{t}, \mathbf{L}, \mathbf{t}_{\text{IBT}}$)
  $\mathbf{b}$ = ComputeBeliefState($\mathbf{t}, \mathbf{t}_{\text{IBT}}$)
  for all $\mathbf{s} \in \mathbf{b}$ do
    $\mathbf{\Phi}_{\text{active}} = \emptyset$
    $\mathbf{\Phi}_{\text{needed}} = \emptyset$
    for all $\mathbf{l} = \langle \mathbf{h}_i, \phi_j, \mathbf{h}_j\rangle \in \mathbf{L}$ do
      if $(\mathbf{h}_i \prec \mathbf{h}_\mathbf{t} \wedge \mathbf{h}_\mathbf{t} \preceq \mathbf{h}_j)$ == TRUE then
        $\mathbf{\Phi}_{\text{active}} = \mathbf{\Phi}_{\text{active}} \cup \phi_j$
      if $(\mathbf{h}_i \preceq \mathbf{h}_\mathbf{t} \wedge \mathbf{h}_\mathbf{t} \prec \mathbf{h}_j)$ == TRUE then
        $\mathbf{\Phi}_{\text{needed}} = \mathbf{\Phi}_{\text{needed}} \cup \phi_j$
    for all $u \in \mathbf{U}$ do
      if $\mathbf{\Phi}_{\text{active}}(\Omega_u(\mathbf{s}))$ == FALSE then
        $\mathbf{P} = \langle \Omega_u(\mathbf{s}), \mathbf{\Phi}_{\text{needed}}, \mathbf{A}\rangle$
        $\pi$ = GeneratePlan ($\mathbf{P}$)
        if $\pi \neq \emptyset$ then
          Accommodate($\mathbf{t}_{\text{cr}}, \mathbf{t}, u, \pi$)
        else
          Interfere($\mathbf{t}_{\text{cr}}, \mathbf{t}, u$)
```

4.5 User Inaction

The user may choose not to execute actions that are required to progress the narrative further. For example, a narrative may require the user to throw a ball into the scene for two bears to play catch. To account for potential user inaction, our automation framework generates contingency plans where the characters in the story may adopt alternate means to accomplish the same effect of the user interaction. For each node $\mathbf{t}$ corresponding to an interaction u, we define a problem instance $\mathbf{P} = \langle \mathbf{s}, \Omega_u(\mathbf{s}), \mathbf{A}-\mathbf{U}\rangle$ where the action space $\mathbf{A}-\mathbf{U}$ only considers affordance instances with smart objects and discounts user interactions. This is used to generate a plan that achieves the same effect as $\Omega_u(\mathbf{s})$ and is integrated into the original IBT definition, as shown in Fig. 5. During narrative execution, if the user does not perform the desired interaction within a reasonable time threshold, it is said to fail and the contingency plan is executed.

4.6 Automated Narrative Synthesis

Authors may harness the power of automation to automatically synthesize narratives which can be integrated into the IBT and edited to meet author requirements. At a given node $\mathbf{t}$ in the IBT, the author

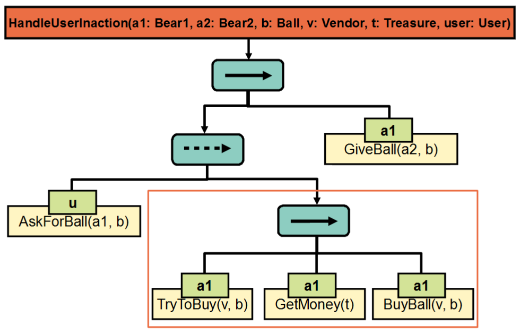

HandleUserInaction(a1: Bear1, a2: Bear2, b: Ball, v: Vendor, t: Treasure, user: User)
a1
GiveBall(a2, b)
u
AskForBall(a1, b)
a1
TryToBuy(v, b)
a1
GetMoney(t)
a1
BuyBall(v, b)

Figure 5: *Plan generation to accommodate user inaction. Our system automatically generates an alternate strategy (highlighted nodes) to accommodate potential user inactions that may hinder narrative progression.*

simply specifies a desired set of preconditions $\mathbf{\Phi}_g$. This translates into multiple problem instances $\mathbf{P} = \langle \mathbf{s}, \mathbf{\Phi}_g, \mathbf{A} \rangle$ for each state $\mathbf{s}$ in the belief state $\mathbf{b}$, obtained as a result of executing the IBT up to $\mathbf{t}$. A plan π is generated for each problem instance $\mathbf{P}$ and inserted into the IBT, to provide a narrative that accommodates author-specified preconditions $\mathbf{\Phi}_g$.

5 Application

We developed an interactive narrative authored using the tools described above and deployed it as an Augmented Reality application on mobile devices. AR applications on mobile devices with multiple sensors benefit from versatile input mechanisms and provide a strong use case for free-form interactive narratives with a host of interaction possibilities. The game application was implemented using the Unity3D game engine with a data-driven character animation system. The animation functionality is exposed to the author as affordances (e.g., `LookAt`(obj), `Reach`(target)) which can be invoked as leaf nodes in BT's. For more details of the animation system and BT implementation, please refer to [Shoulson et al. 2013c]. We used Vuforia [Qualcomm 2010] for image-based tracking and camera pose estimation. Smaller marker images were used as an additional interaction modality to trigger state changes and branch the story in different directions. Please refer to the supplementary document for additional implementation details.

5.1 Scenario and Story Definition

We author a narrative with two male bears B_1, B_2, and a female bear B_3. A shopkeeper B_4 is also present who is able to sell toys to the other bears. The characters have generic state attributes such as `InScene`, `IsHappy`, `IsPanicked`, `IsPlaying`, and relationships with other characters and smart objects such as `Knows`, `Holds`, and `Loves`. Other smart objects include a soccer ball, a beach ball (can be picked up by the bears and used to play catch), a honeypot (can be consumed to make the bears happy), bees (scare the bears away), and flowers (distract the bees). The smart objects are equipped with a variety of affordances including `Converse`, `Argue`, `ThrowBall`, and `EatHoney` which may include the user as a participant. For example, the ball has a `PickUp` affordance which the user can trigger to pick the ball from the scene. The representative affordances defined for the smart objects used to author the interactive narrative are outlined in the supplementary document for reference.

Baseline Story. The first bear B_1, enters the scene and looks up at the player with curiosity. The player can freely interact with B_1 using a host of interaction possibilities or introduce the second bear B_2 into the scene. B_2 asks B_1 for a beach ball so they can play catch. B_1 is unable to find a ball and turns to the player for help. The player may choose to give a soccer ball to the bears but they only want to play with the beach ball. Depending on where the player throws the beach ball, B_1 or B_2 may pick it up which influences future branches in the story. For example, B_1 chooses to involve the player in the game of catch if the player gave him the ball.

Additional Interactions. At any point, the player may use a honeypot image marker to trigger a honeypot in the world. The bears leave aside whatever they are doing, and make a beeline towards the honeypot, which represents one possible conclusion of the story. The player may also choose to trigger bees into the world, which chase the bears and disrupt the current arc (e.g., having a conversation, playing catch). Flowers may then be used to distract the bees and save the bears. Fig. 1 illustrates an example execution of the authored narrative.

State Persistence. The player's choices have ramifications later on in the story. For example, B_1 remembers if the player interacted him with at the beginning of the narrative, or helped him by giving him the ball, and includes him in the game of catch by periodically throwing him the ball. If the player adopts an antagonistic approach (e.g., by triggering bees), the bears are less friendly towards him.

Freedom of Interaction. Note that each of these interactions are possible at *any* point, and are not limited to discrete events at key stages in the narrative. For example, the player may choose to trigger the bees or the honeypot at the very beginning of the story, or while the bears are playing ball, and the story will naturally proceed as per the author's intentions. These different story arcs are authored as modular, independent units in the IBT and can be triggered at any stage without the need for complex connections and state checks in the story definition. The above narrative represents a very simple baseline to demonstrate the potential to author free-form interactive narrative experiences. IBT's empower the player with complete freedom of interaction where he may choose to play ball with the bears, give them honey, or simply wreak havoc by releasing a swarm bees at any point of time. The interactions elicit instantaneous and plausible interactions from the characters while staying true to the narrative intent.

Benefits of Automation. The automation tools described in § 4 facilitate the authoring process in a variety of ways.

Inconsistencies. The author specifies a story where B_1 gives the ball to B_2 without having the ball in his possession. The invalid preconditions are automatically detected and the appropriate nodes in the authored story arc are highlighted. Our system can automatically resolve inconsistencies by generating additional nodes in the story definition to satisfy the invalid preconditions. In this example, B_1 requests the player to throw the ball before proceeding to hand it to B_2.

Conflicts. Our system automatically detects potential user actions (or inactions) that may cause conflicts in the story. For example, the player may steal the ball from the bears during their game of catch thus invalidating the `PlayBall` arc, and our system automatically generates a strategy for the bears to request the player to return the ball. If player does not perform the expected interaction (e.g., does not give the bears the ball), an alternate strategy is generated where the bears find an alternative means to finding the ball. In this scenario, the bear purchases the ball from a vendor who in turn requires him to get money.

Automatic Story Synthesis. The author can simply specify desired preconditions and our system can generate story arcs that lead to the desired outcome. For example, the author may specify that the female bear B_3 must fall in love with B_2. A plan is generated whereby B_2 acquires the beach ball, which is desired by B_3 in order to win her heart.

6 Conclusion

This paper demonstrates the potential benefits of new design formalisms for authoring interactive narratives based on Behavior Trees, and the use of computer-assisted tools for reducing the authoring burden without sacrificing control. Compared to traditional story graph representations, IBT's better scale with story complexity and freedom of user interaction, and authoring stories takes lesser time with reduced number of errors. The authoring complexity is further reduced with the help of automation, and errors are completely avoided.

Limitations and Future Work. The use of computational intelligence to facilitate the authoring process requires the specification of domain knowledge, which is currently limited to domain experts. For future work, we will explore better representations and automatic acquisition of domain knowledge. While IBT's significantly reduce the authoring complexity, it has a learning curve and is not immediately accessible to end users without a computer programming background. The task of authoring interactive narratives requires expertise and takes effort and time. Improved design formalisms are thus needed to meet the growing demand of casual content creation.

The proposed automation features can in principle be applied to other formalisms as well, and future studies may use improved story graphs with hierarchical node structures to study the benefits of automation with alternate story representations. For future work, we will conduct a user study to assess the authoring complexity of the different design formalisms, and the benefits of automation during the authoring process. Additional experiments should be conducted to capture and analyze the playability of the authored narratives. Strong playability metrics that reflect the semantic quality of the authored narratives need to developed. Finally, we would like to evaluate the scalability of our approach to handle the complexity and diversity of narratives present in commercial games.

References

GORDON, A., VAN LENT, M., VELSEN, M. V., CARPENTER, P., AND JHALA, A. 2004. Branching Storylines in Virtual Reality Environments for Leadership Development. In *AAAI*, 844–851.

ISLA, D. 2005. Handling Complexity in the Halo 2 AI. In *Game Developers Conference*.

KAPADIA, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. 2011. A Behavior-Authoring Framework for Multiactor Simulations. *IEEE CGA 31*, 6 (Nov), 45–55.

KAPADIA, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. 2011. Multi-actor planning for directable simulations. In *Digital Media and Digital Content Management (DMDCM)*, 111–116.

KAPADIA, M., SHOULSON, A., DURUPINAR, F., AND BADLER, N. 2013. Authoring Multi-actor Behaviors in Crowds with Diverse Personalities. In *Modeling, Simulation and Visual Analysis of Crowds*, vol. 11. 147–180.

LOYALL, A. B. 1997. *Believable agents: building interactive personalities*. PhD thesis, Pittsburgh, PA, USA.

MAGERKO, B., LAIRD, J. E., ASSANIE, M., KERFOOT, A., AND STOKES, D. 2004. AI Characters and Directors for Interactive Computer Games. *Artificial Intelligence 1001*, 877–883.

MATEAS, M., AND STERN, A. 2003. Integrating plot , character and natural language processing in the interactive drama facade. In *TIDSE*, vol. 2.

MATEAS, M., AND STERN, A. 2004. A behavior language: Joint action and behavioral idioms. In *Life-Like Characters*. Springer, 135–161.

MENOU, E. 2001. Real-time character animation using multi-layered scripts and spacetime optimization. In *ICVS*, Springer-Verlag, London, UK, 135–144.

MILLINGTON, I., AND FUNGE, J. 2009. *Artificial Intelligence for Games, Second Edition*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

MINTON, S., BRESINA, J., AND DRUMMOND, M. 1994. Total-order and partial-order planning: A comparative analysis. *Journal of Artificial Intelligence Research 2*, 227–262.

PEARL, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of ACM SIGGRAPH*, ACM, New York, NY, USA, 205–216.

QUALCOMM, 2010. Vuforia Developer SDK.

RIEDL, M. O., AND BULITKO, V. 2013. Interactive narrative: An intelligent systems approach. *AI Magazine 34*, 1, 67–77.

RIEDL, M. O., AND YOUNG, R. M. 2006. From linear story generation to branching story graphs. *IEEE CGA 26*, 3, 23–31.

SACERDOTI, E. D. 1975. The nonlinear nature of plans. In *IJCAI*, 206–214.

SHOULSON, A., GILBERT, M. L., KAPADIA, M., AND BADLER, N. I. 2013. An event-centric planning approach for dynamic real-time narrative. In *Proceedings of Motion on Games*, ACM, New York, NY, USA, MIG '13, 99:121–99:130.

SHOULSON, A., KAPADIA, M., AND BADLER, N. 2013. PAStE: A Platform for Adaptive Storytelling with Events. In *INT VI, AIIDE Workshop*.

SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. I. 2013. Adapt: the agent development and prototyping testbed. In *ACM SIGGRAPH I3D*, 9–18.

SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. 2014. ADAPT: The Agent Developmentand Prototyping Testbed. *IEEE TVCG 20*, 7 (July), 1035–1047.

SI, M., MARSELLA, S. C., AND PYNADATH, D. V. 2005. Thespian: An architecture for interactive pedagogical drama. In *Proceeding of the 2005 Conference on Artificial Intelligence in Education*. 595–602.

THUE, D., BULITKO, V., SPETCH, M., AND WASYLISHEN, E. 2007. Interactive storytelling: A player modelling approach. In *AIIDE*.

WEYHRAUCH, P. W. 1997. *Guiding interactive drama*. PhD thesis, Pittsburgh, PA, USA. AAI9802566.