Discourse-Driven Tellability Goals for Narrative Planning*

David R. Winer and R. Michael Young
Liquid Narrative Group, Computer Science
North Carolina State University, Raleigh NC
drwiner@ncsu.edu

Abstract

Typical narrative generation systems adopt a pipeline approach in which *fabula* (i.e., plot) is generated and provided as input for generating discourse and narration (e.g. text or animation). However, stories produced independently from a communicative plan are not guaranteed to have properties which are readily tellable or worth telling. This paper presents an approach to narrative planning in which constraints for a story are discovered as part of the search for compatible story and discourse solutions. Narrative generation involves goal specification by users at multiple levels, and this approach furthers the research agenda to reduce this burden.

Introduction

Narrative intelligence is fundamental for organizing experiences, understanding our surroundings, and forming predictions about the future [Bruner, 1991; Schank, 1995]. AI planning research is a popular source of data structures and algorithms for understanding, generating, and reasoning about stories [Young et al., 2014]. Narratologists frequently distinguish the story (i.e. *fabula*) of narrative from the discourse (e.g. the narration of the story to a spectator) [Propp, 1968; Bruner, 1991], and plans have proven useful for modeling both story and discourse [Young et al., 2014]; they are effective for modeling discourse because a coherent sequence of communicative actions is plan-like [Cohen and Perrault, 1979], and plans are effective for modeling stories because stories are composed of events with cause-effect relations with characters themselves forming plans to achieve goals [Young et al., 2014]. Psychological studies have demonstrated that plans capture many of the key aspects of narratives that spectators use to understand narrative discourse [Ware et al., 2014; Radvansky et al., 2014; Cardona-Rivera et al., 2016].

A central item in the narrative planning research agenda is to adapt planners originally developed for efficient problem solving [Bonet and Geffner, 2001] to produce plans which are interesting by virtue that they have properties worth telling (i.e., they are *tellable*). However, narratologists often disagree on what properties make a story tellable [Pratt, 1977; Bruner, 1991]. Some properties such as conflict may be essential for tellability, but other properties may depend on the goals of the narrator. For example, a screenwriter may add an event to a screenplay's story level (e.g., a non-central characters falls to his death off of a narrow bridge), in order to elicit a discourse effect (e.g., the viewer believing this bridge is a dangerous obstacle for the protagonist). Other properties in a story may be tellable because they facilitate narration (e.g. a conversation in a wide room is more readily tellable through film than a conversation in a narrow hallway). In this work, we present a narrative-theoretic language in which users can specify storyworld constraints to support story tellability.

A typical approach to planning-based narra-

*These match the formatting instructions of IJCAI-07. The support of IJCAI, Inc. is acknowledged.

tive generation is a story-then-discourse pipeline approach, in which a story is produced from a story planner and passed as input to a discourse planner, which then produces a plan for telling the story [Callaway and Lester, 2002; Jhala and Young, 2010; Young et al., 2014]. As a consequence, the story is created in isolation and not tailored for the discourse plan. If there are story constraints associated with discourse actions (e.g., to film a conversation between two characters using a 2-shot, there must be sufficient free space around both characters for cameras to position themselves), an input plan that solves a story problem may not meet some set of constraints needed to solve the discourse problem (i.e., the story plan is *incompatible* with the discourse goals), even though a solution to the story problem exists and does meet those constraints.

We call a planner that generates both story and discourse plans from story and discourse problems *bipartite complete* just when the planner will find a compatible pair of story and discourse solutions when one exists. Thus, the story-then-discourse planner may not be bipartite complete. We present a discourse-driven approach to narrative generation which tailors the story to support requirements specified by the discourse. To model story plan structure at the discourse level, we define a graph structure that captures partially or incrementally refined constraints between elements of a story plan, and use the properties of this graph to add constraints and check for consistency during discourse plan construction. We introduce the BiPOCL algorithm which can interleave constructing the story graph and the discourse plan, and ensures that the completion of the constraints in the graph are both causally consistent and fulfill the requirements of the discourse plan.

Related Work

Story planning benefits from a rich history of AI research. The first story generation system to use planning is TALESPIN [Meehan, 1977], which generates stories about woodland creatures that take actions, in accordance with rules of the world, to satisfy simple needs. Another early story generation system UNIVERSE [Lebowitz, 1983] represents plot fragments as plans and selects a fragment to execute that satisfies an authorial goal. MINSTREL [Turner, 1993] uses planning to create an outline of a story and case-based reasoning to fill in details from a story library. Cavazza and colleagues use character-centric planning [Cavazza, Charles, and Mead, 2002], such that story is the emergent property of characters pursuing goals. Partial-order causal link (POCL) planning is used as a top-down approach to story generation; a user specifies a goal state of the story world and the solution space is restricted to just those story plans which are causally sound [Riedl and Young, 2010; Ware et al., 2014] (see also [Young et al., 2014] for review of story planning). CPOCL (conflict POCL) [Ware et al., 2014] is a story planner with a solution space supporting character intentionality and conflict.

In story-then-discourse systems such as Darshak [Jhala and Young, 2010], a planner generates or takes as input a story plan, and then generates a discourse plan to narrate the story. Darshak is a cinematic narrative planning system in which dramatic patterns are defined as discourse operators that have constraints describing what must be true about the story plan for the operator to instantiate as a step. Dramatic patterns decompose into camera shot patterns which convey the story steps or set of steps in a manner appropriate for the dramatic pattern.

Dramatis [O'Neill and Riedl, 2014] takes a story plan as input and measures the suspense level by generating a character plan and determining the likelihood of its success using a definition informed by a cognitive theory of suspense. Suspenser [Cheong and Young, 2015] and Prevoyant [Bae and Young, 2014] systems take as input a story plan and arrange the steps in an ordering which elicits suspense or surprise, respectively, based on cognitive-computational definitions inspired by psychological theory.

Storybook [Callaway and Lester, 2002] is an end-to-end narrative prose generation system with four parts: 1) a narrative organizer which takes as input a story plan, 2) a sentence planner which creates a proto-sentence outline, 3) a revision module which produces prose paragraphs from proto-sentences, and 4) a surface realizer which makes some grammatical edits and formats the story in a file.

Problem Formulation

The discourse-driven narrative planning approach is a search for solutions to two problems – a story problem and discourse problem – where a

solution is a plan of actions to bring an initial state to a goal state. At the story level, the solution represents the actions of characters in the storyworld, whereas at the discourse level the solution represents the communicative actions by a narrator agent to describe something about the story. In prior approaches to story and discourse generation, a story solution is passed through a pipeline model. In our approach, goals for the story solution are selected as part of the search for a discourse solution. These goals are prerequisite criteria about the story that must be true for the story and discourse plans to be a bipartite solution to the story and discourse problems.

Planning

Partial-order causal link (POCL) planning is a type of *planning as refinement search* [Kambhampati, Knoblock, and Yang, 1995], involving search through plan-space such that each child node in the search is a refinement to the (potentially flawed) plan represented at its parent node. Through an iterative process of identifying flaws in the plan and repairing them in a least-commitment manner [Weld, 1994], plans with no flaws are selected and returned as solutions to the planning task.

Definition 1 (**POCL Plan**). *A POCL plan is a tuple* $\langle S, B, O, L\rangle$ *with the following components:*

- S *is a set of steps, instantiated STRIPS-style* ***operators*** *of the form* $\langle \alpha, V, A, Pre, Eff\rangle$ *where* α *is a name distinguishing the operator type,* V *is a set of variables,* Pre *is a set of preconditions, first-order literals that must be true for the step to execute, and* Eff *is a set of effects, first-order literals that are made true by the step's execution.*
- B *is a set of* ***binding constraints*** *on variables in steps in* S *including variable assignments, codesignation, and non-codesignation constraints of the form* $\langle X, Y, =\rangle$.
- O *is a set of* ***ordering constraints*** *on steps in* S *of the form* $s_a \prec s_b$ *indicating that* s_a *executes before* s_b *for every total ordering of steps.*
- L *is a set of* ***causal links*** *of the form* $s \xrightarrow{p} t$ *indicating that* s *is a step with some effect* p*, and* t *is a step with some precondition* p*. Step* t *is referred to as the consequent of the causal link, and* p *is the dependency condition of the causal link. Step* t*'s causal antecedents are all steps* σ *such that there exists a causal link* $\sigma \xrightarrow{p} t$*. A step's causal ancestors are its causal antecedents in the transitive closure of the antecedent relation. If* $s \xrightarrow{p} t$ *is a causal link, then* p *is met when* t *executes just when there is no step* u *such that* u *has effect* $\neg p$ *and* u *may execute before* t *and after* s*.*

A plan is considered valid just when every variable of every step is assigned to a constant, and for every total ordering of steps, each step's preconditions are met when that step executes.

Definition 2 (**Problem**). *A problem is a tuple* $\langle \Lambda, C, I, G\rangle$ *where* Λ *is a set of operators,* C *is a set of constants,* I *is the initial state, a set of function-free ground literals indicating what is true about the world, and* G *is the goal state, a set of function-free ground literals indicating the goal criteria for the problem.*

The POCL algorithm solves the planning problem by resolving flaws. Initially, flaws are created for each goal condition of the goal state.

Definition 3 (**Open Precondition Flaw**). *An open precondition flaw in a plan* $P = \langle S, B, O, L\rangle$ *is a tuple* $\langle s_{need}, p\rangle$ *where* $s_{need} \in S$ *and* $\neg\exists s \in S$ *such that* $s \xrightarrow{p} s_{need} \in L$.

An open precondition flaw $\langle s_{need}, p\rangle$ is resolved by **reusing** a step s already in the plan with effect p, adding causal link $s \xrightarrow{p} s_{need}$, and adding ordering $s \prec s_{need}$ or by adding a **new step** s_{new} from an operator with effect p, adding causal link $s_{new} \xrightarrow{p} s_{need}$, adding ordering $s_{new} \prec s_{need|}$ and adding precondition flaws for each precondition of s_{need}.

Definition 4 (**Threatened Causal Link Flaw**). *A threatened causal link flaw in a plan* P *is a tuple* $\langle s \xrightarrow{p} t, s_{threat}\rangle$ *such that* $s \xrightarrow{p} t$ *is a causal link in* P *and* s_{threat} *is a step in* P *with effect* $\neg p$ *and can execute before* t *and after* s *in some total ordering of steps in* P.

Threatened causal link flaws can be resolved by adding an ordering to prevent the threatening step from executing between the antecedent and the consequent, or by adding bindings which prevent the threatening effect from unifying with the dependency condition. A plan is considered a **solution** to the problem if the plan is valid, all

orderings and bindings are consistent, and there are no flaws detected in the plan.

Element Graphs

To formalize the computational operations in our approach, it's convenient to re-conceptualize the plan as a graph, which we refer to as an element graph, composed of elements and edges between elements. An element can represent a variable, actor, literal, step, or a special root element which represents the plan in its entirety.

Definition 5 (**Element**). *An **element** is a tuple $\langle i, t, A \rangle$ where i is a unique identifier, t is a type (i.e., a variable, actor, literal, step, or special root type), and A is a dictionary of attributes associated with the type (e.g., a step-typed element has an attribute for its operator type, a literal-typed element has an attribute for its predicate name, its status as true or false, etc.)*

*Two elements are considered **consistent** just when they are the same type and for each non-empty attribute of one element, the other element's corresponding attribute is either the same value or empty. Two elements are considered **equivalent** just when they are consistent and for each non-empty attribute of one element, the other element's corresponding attribute is the same value.*

Definition 6 (**Edge**). *An **edge** between two elements a and b is a labeled, directed arc either indicating that element a and b have the relationship as specified by the arc's label (positive edge), or do not have the relationship (negative edge). Two edges are considered **consistent** just when the sources are consistent, the sinks are consistent, and they have the same label. Two edges are considered **equivalent** just when the sources are equivalent, the sinks are equivalent, and the edges have the same label.*

Definition 7 (**Element Graph**). *An **element graph** is a tuple $\langle V, E, C, \Omega \rangle$, where V is a set of elements, E is a set of positive edges between elements in V, $C = \{c_0, ..., c_n\}$ is a set of sets with positive and negative edges between elements in V, and Ω is the set of edge label options.*

The edges in sets $\{c_0, ..., c_n\} \in C$ represent **constraint sets**. A single edge may be insufficient to describe a constraint. For this reason, edges are grouped into constraint sets in order to enable constraints which specify a set of criteria which all must be in the element graph for the constraint to be considered detected. Two edge sets are considered **equivalent** just when every edge in one set is equivalent to some edge in the other, and if two edges share an endpoint, then their equivalent counterparts must also share an endpoint. Negative edges in constraint sets represent edges that if detected, disqualify the constraint from being considered detected.

Definition 8 (**Constraint Detection**). *A constraint is **detected** in an element graph $\langle V, E, C, \Omega \rangle$ just when the set of positive edges in some constraint set $c \in C$ is equivalent to some subset of E and if there is at least one negative edge in c, the set of negative edges in c is not equivalent to any subset of E.*

Definition 9 (**Internal Consistency**). *An element graph $\langle V, E, C, \Omega \rangle$ is considered **internally consistent** just when the graph is acyclic and no constraint is detected.*

Edges with label 'ordering' and 'causal-link' represent orderings and causal links. For edges with label 'ordering', the source of the edge precedes the sink.

An element such as a step element or literal element refers only to its id, type, and attributes, and not to its component arguments or preconditions and effects. The element-induced subgraph is the graph structure representing the element type and its components.

Definition 10 (**Element-Induced Subgraph**). *If G_E is an element graph and ϵ is an element in G_E, then an ϵ-induced subgraph of G_E is an element graph composed of all elements and edges reachable from ϵ by edges not labeled 'ordering' or 'causal link'.*

Definition 11 (**Instantiated Element**). *If v is an element in an element graph, $G_v = \langle V_v, E_v, C_v, L_v \rangle$ is a v-induced subgraph, and $G_O = \langle V_O, E_O, C_O, L_O \rangle$ is another element graph, then v can be instantiated by G_O just when all edges in E_v are consistent with some edge in E_O, no constraint in C_v is detected in G_O, and no constraint in C_O is detected in G_v. An element v can be considered **instantiated** as G_O just when E_v and E_O are equivalent.*

There may be multiple ways to match the edges in an element-induced subgraph to the edges in another graph, and therefore multiple ways

to instantiate an element. An assignment of an element in an element graph to an element in another element graph is called a **coupling**. The resulting set of couplings denotes how the graph was instantiated.

We now have the language needed to describe the operations of the approach.

Discourse Level

At the discourse level, literals indicate what a spectator agent believes is true and not true about the story. Discourse actions are communicative actions taken by a narrator agent to add or delete the spectator's beliefs.

Discourse operators are regular STRIPS-style operators [Fikes and Nilsson, 1972], plus a **prerequisite**: an element subgraph indicating the structural properties required for an element graph to be compatible with a discourse plan which includes that discourse action. The variables in a discourse operator are partially defined elements.

Definition 12 (**Discourse Operator**). *A **discourse operator** is a tuple $\langle \alpha, V, Pre, Eff, Req \rangle$, where α is the operator type, V is a set of elements, Pre is a set of preconditions, literals indicating what the spectator must believe for the step to execute, Eff is a set of effects, literals indicating what the spectator believes as the result of the step's execution, and Req is an element subgraph.*

The element subgraph Req indicates the properties required in an element graph representing the story for the story to be compatible.

Definition 13 (**Discourse Plan**). *A **discourse plan** is a tuple $\langle S_D, B_D, C_D, O_D, L_D, G_S \rangle$ where S_D is a set of discourse steps, B_D is a set of bindings over elements in steps in S_D, C_D is the set of couplings between elements in steps in S_D and elements in G_S, O_D is a set of orderings over steps in S_D, L_D is a set of causal links between steps in S_D, and G_S is an element graph which instantiates the prerequisites from steps in S_D. The plan is considered valid just when the plan is valid according to Definition 1, every element in steps in S_D is in a coupling with some element in G_S, all prerequisites in S_D are instantiated in G_S, and G_S is internally consistent.*

For every discourse step that is added to the discourse plan, some element in the element graph must instantiate the step's prerequisite (sometimes via the special root element). Thus, the number of possible ways to include a prerequisite in an element graph is the number of elements in the element graph which can instantiate the prerequisite times the number of ways each of those elements can be instantiated.

Definition 14 (**Open Prerequisite Flaw**). *An open prerequisite flaw in an element graph G_S is a tuple $\langle Q \rangle$ where Q is an element subgraph that is not yet instantiated by an element in G_S.*

Additionally, each step element in an element graph must become instantiated by an operator. The operators in a domain are converted into element graphs, called **operator graphs**, and the step element is instantiated by some operator graph as defined in Definition 11.

Definition 15 (**Uninstantiated Step Element Flaw**). *An uninstantiated step element flaw in an element graph G_S is a tuple $\langle \sigma \rangle$ where σ is a step element in G_S that is not yet instantiated by an operator graph.*

We can now consider planning story and discourse in the same algorithm by choosing either to resolve flaws in the story plan, which is conceptualized as an element graph, or to resolve flaws in the discourse plan.

The BiPOCL algorithm is presented in Algorithm 1. Goal planning at the discourse level (lines 3-13) includes selecting and resolving open precondition flaws, threatened causal link flaws, and open prerequisite flaws. Whenever a new discourse step is added to the discourse plan, an open prerequisite flaw is added to the set of flaws in the story plan. An open prerequisite flaw (line 13) is resolving by finding an instantiation of the prerequisite in the element graph. Goal planning at the story level (lines 14-15) includes selecting and resolving open precondition flaws, threatened causal link flaws, uninstantiated step element flaws, and other flaws beyond the scope of this work [Riedl and Young, 2010; Ware et al., 2014]. An uninstantiated step element flaw $\langle \sigma \rangle$ is resolved by selecting an operator graph which can instantiate σ in the element graph.

Example

We present example discourse operators in **Figure 1**. Imagine these two operators are instantiated as steps and there is a causal link between them. The

Algorithm 1 The BiPOCL (Bipartite Partial Order Causal Link) Algorithm

$BiPOCL\,(P_D = \langle S_D, B_D, C_D, O_D, L_D, G_S\rangle, F_D, F_S, \Lambda_D, \Lambda_S)$

Input: P_D is an empty plan with dummy initial and final steps, G_S has the special root element, F_D and F_S include open precondition flaws for discourse and story goal conditions, respectively, Λ_D is a set of discourse operators, and Λ_S is a set of operator graphs.

Termination: If either B_D, O_D, or G_S is inconsistent, fail. If $F_D \cup F_S = \emptyset$, return P_D.
Plan Refinement: Nondeterministically choose a flaw f from $F_D \cup F_S$:
    If $f \in F_D$ is an open precondition flaw $\langle s_{need}, p\rangle$, choose s_{add} with effect p in one of two ways:
        Reuse: Choose s_{add} from S_D.
        New Step: Create s_{add} from an operator in Λ_D with effect p. Let $S'_D = S_D \cup \{s_{add}\}$.
            For each precondition c of s_{add}, add new open precondition flaw $\langle s_{add}, c\rangle$ to F'_D.
            Add an open prerequisite flaw $\langle Q\rangle$ to F'_S for the prerequisite Q of s_{add}.
        Let $L'_D = L_D \cup \langle s_{add} \xrightarrow{p} s_{need}\rangle$, add bindings to B'_D, let $O'_D = O_D \cup \langle s_{add} \prec s_{need}\rangle$.
    If $f \in F_D$ is a threatened causal link flaw $\langle s \xrightarrow{p} u, t\rangle$, choose how to prevent the threat:
        Promotion: Let $O'_D = O'_D \cup \{t \prec s\}$,
        Demotion: Let $O'_D = O'_D \cup \{u \prec t\}$
        Restriction: Add bindings to B'_D which cause the threatening effect of t not to unify with p.
    If $f \in F_D$ is an open prereq flaw $\langle Q\rangle$, instantiate an element in G_S with Q, add couplings to C'_D.
    If $f \in F_S$ is not an uninstantiated step element flaw, resolve f using its associated method.
    If $f \in F_S$ is an uninstantiated step element flaw $\langle \sigma\rangle$, instantiate σ with some operator in Λ_S
Threat Detection: If any causal link (or causal link edge) λ is threatened by a step σ,
    Add new threatened causal link flaw $\langle \lambda, \sigma\rangle$ to F'_D or F'_S.
Recursive Invocation Call $BiPOCL(P'_D, F'_D, F'_S, \Lambda_D, \Lambda_S)$.

Figure 1 Example discourse operators

convey-danger-at
elements: (?death - step,
?victim - actor
?loc - variable)
precondition: (and)
effect: (bel-danger-at ?loc ?death)
prerequisite: (*edges* = {
('effect' ?death (not (alive ?victim))),
('precond' ?death (at ?victim ?loc))})

convey-in-danger
elements:(?hero ?c - actor
?dloc - variable
?move ?m2 ?dstep - step)
precondition: (bel-danger-at
?dloc ?dstep)
effect: (bel-in-danger ?hero)
prerequisite: (*edges* = {
('ordering' ?dstep ?move),
('effect' ?move (at ?hero ?dloc))},
constraints={c1={
('effect' ?m2 not (at ?c ?dloc),
('not effect' ?m2 (not (alive ?c)))
('ordering' ?m2 ?move)}})

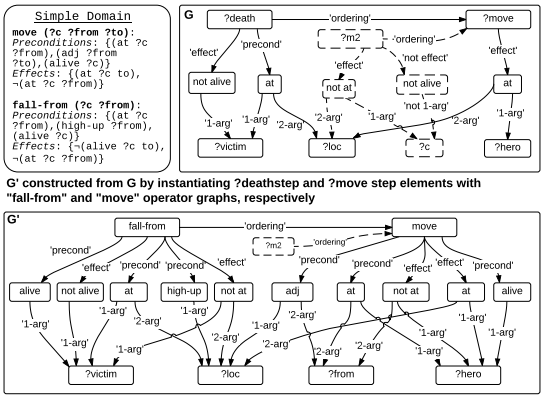
first step conveys that a location is dangerous by virtue that a character dies at this location, and the second step conveys that a character is in danger by virtue that he/she is at that dangerous location.

The top-right graph (G) of **Figure 2** shows one way to instantiate the prerequisites from those steps into an element graph. An example of another way would be that element ?victim equals element ?hero, even though this would prove problematic and ought to not be considered as early as possible. In G, the dashed arrows show constraint edges, and dashed boxes show elements which are endpoints only of constraint edges. These edges are from the constraint set in the 'convey-in-danger' discourse operator.

The element graph in G of Figure 2 specifies the following scenario: A) a step $?death$ executes with the precondition that a character $?victim$ is at a location $?loc$ and the effect is that $?victim$ is not alive, B) a step $?move$ executes after $?death$ with the effect that character $?hero$ is at location $?loc$, and C) there is no step $?m2$ which executes before $?move$ (c-1) which has the effect that some character $?c$ is not at location $?loc$ (c-2) and does not have the effect that $?c$ is not alive (c-3). The reason for part C of this scenario is that the character might not be interpreted as being in danger if the spectator observes other characters safely passing through the dangerous location.

The graph on the bottom of Figure 2 (G') shows an element graph given the instantiation of the $?death$ element with the 'fall-from' operator graph, and the instantiation of the $?move$ element with the 'move' operator graph. If the literal names "alive" and "at" were not found in the operator

Figure 2 G (top-right) shows an element graph which instantiates the prerequisites from discourse steps as described in the text. The dashed arrows show constraint edges, and dashed boxes show elements which are endpoints only of constraint edges. At the top-left is a very simple story domain with two operators. G' (bottom) shows an instantiation of the step elements in the prerequisites given operator graphs constructed from the operators.



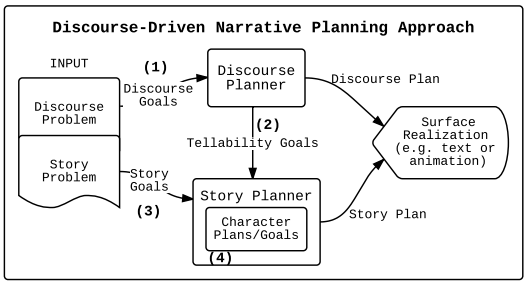
graphs, then the instantiation would not be possible. The 'fall-from' operator meets some criteria for the constraint of the discourse step 'convey-in-danger'. It is ordered to occur before 'move' = $?move$ (c-1) and has an effect that a character ($?victim$) is not at $?loc$ (c-2), but it also has the effect that $?victim$ is not alive (violating c-3); therefore, the constraint is not detected in G'. Since G' is also acyclic, G' is internally consistent.

Conclusion and Future work

The work presented in this paper describes mechanics for discourse-driven narrative planning, for the purpose of generating stories to be compatible with a discourse plan.

One of the advantages to our design is that the planning process can be interleaved such that paths in the discourse search are pruned if no internally consistent element graph is possible, and paths in the story search are pruned by considering only element graphs which are internally consistent given the prerequisites in the discourse plan. However, it is not necessary to interleave planning in this way: one can first find a valid discourse plan by constructing an element graph from prerequisites in discourse steps and then use this graph as the starting point for finding the complete set of partial plans. In future work, we will formulate and compare heuristics based on their efficiency.

Figure 3 High-level schematic showing the discourse-driven planning approach for narrative generation and 4 levels of goal management: (1) discourse goals, (2) tellability goals, (3) story goals, and (4) character goals.



Ultimately, our approach will involve goal management at 4 levels; **(1)** discourse goals specifying the final mental state of a spectator agent, **(2)** tellability goals to scaffold the story via prerequisites, **(3)** story goals specifying the final state of the story world, and **(4)** character goals which are adopted during the story but may be thwarted. **Figure 3** shows a high level schematic of the approach. Through prior work [Riedl and Young, 2010; Ware et al., 2014], the goal management at Level 4 (character goals) is encoded into the problem solving at Level 3 (i.e., $4 \subseteq 3$) by encoding plan flaws to character goal following (e.g., characters only consent to actions that are part of a plausible plan to achieve a goal), so that the story planner reasons about character goals as part of the search for story solutions. The objective of this work is to formulate tellable story goals using a meta-plan language so that the discourse planner can reason about story as part of the search for a discourse solution (i.e., $2 \subseteq 1$). A next-step ambition of our work is to generate the story planning problem goal state as part of solving tellability goals ($3 \subseteq 2$) so that a user need only specify discourse goals (i.e., $4 \subseteq 3 \subseteq 2 \subseteq 1$).

References

[Bae and Young, 2014] Bae, B.-C., and Young, R. M. 2014. A computational model of narrative generation for surprise arousal. *IEEE Transactions on Computational Intelligence and AI in Games* 6(2):131–143.

[Bonet and Geffner, 2001] Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

[Bruner, 1991] Bruner, J. 1991. The narrative construction of reality. *Critical inquiry* 1–21.

[Callaway and Lester, 2002] Callaway, C. B., and Lester, J. C. 2002. Narrative prose generation. *Artificial Intelligence* 139(2):213–252.

[Cardona-Rivera et al., 2016] Cardona-Rivera, R. E.; Price, T.; Winer, D. R.; and Young, R. M. 2016. Question answering in the context of stories generated by computers. *Advances in Cognitive Systems*. In Press.

[Cavazza, Charles, and Mead, 2002] Cavazza, M.; Charles, F.; and Mead, S. J. 2002. Character-based interactive storytelling. *IEEE Intelligent systems*.

[Cheong and Young, 2015] Cheong, Y. G., and Young, R. M. 2015. Suspenser: A story generation system for suspense. *IEEE Transactions on Computational Intelligence and AI in Games* 7(1):39–52.

[Cohen and Perrault, 1979] Cohen, P. R., and Perrault, C. R. 1979. Elements of a plan-based theory of speech acts. *Cognitive science* 3(3):177–212.

[Fikes and Nilsson, 1972] Fikes, R. E., and Nilsson, N. J. 1972. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3):189–208.

[Jhala and Young, 2010] Jhala, A., and Young, R. M. 2010. Cinematic visual discourse: Representation, generation, and evaluation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(2):69–81.

[Kambhampati, Knoblock, and Yang, 1995] Kambhampati, S.; Knoblock, C. A.; and Yang, Q. 1995. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* 76(1):167–238.

[Lebowitz, 1983] Lebowitz, M. 1983. Creating a story-telling universe. *Proceedings of the Eighth International Joint Conference on AI* 1:63–65.

[Meehan, 1977] Meehan, J. R. 1977. Tale-spin, an interactive program that writes stories. In *International Joint Conference on AI*, volume 77, 91–98.

[O'Neill and Riedl, 2014] O'Neill, B., and Riedl, M. 2014. Dramatis: A computational model of suspense. In *Association for the Advancement of AI*, volume 2, 2–2.

[Pratt, 1977] Pratt, M. L. 1977. Toward a speech act theory of literary discourse.

[Propp, 1968] Propp, V. 1968. The morphology of the folktale.

[Radvansky et al., 2014] Radvansky, G. A.; Tamplin, A. K.; Armendarez, J.; and Thompson, A. N. 2014. Different kinds of causality in event cognition. *Discourse Processes* 51(7):601–618.

[Riedl and Young, 2010] Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.

[Schank, 1995] Schank, R. C. 1995. *Tell me a story: Narrative and intelligence*. Northwestern University Press.

[Turner, 1993] Turner, S. R. 1993. Minstrel: a computer model of creativity and storytelling.

[Ware et al., 2014] Ware, S. G.; Young, R. M.; Harrison, B.; and Roberts, D. L. 2014. A computational model of plan-based narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and AI in Games* 6(3):271–288.

[Weld, 1994] Weld, D. S. 1994. An introduction to least commitment planning. *AI magazine* 15(4):27.

[Young et al., 2014] Young, R. M.; Ware, S.; Cassell, B.; and Robertson, J. 2014. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *SDV. Sprache und Datenverarbeitung.*