CPOCL: A Narrative Planner Supporting Conflict

Stephen G. Ware and R. Michael Young
sgware@ncsu.edu, young@csc.ncsu.edu
Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC 27695

Abstract

Conflict is an essential element of interesting stories, but little research in computer narrative has addressed it directly. We present a model of narrative conflict inspired by narratology research and based on Partial Order Causal Link (POCL) planning. This model informs an algorithm called CPOCL which extends previous research in story generation. Rather than eliminate all threatened causal links, CPOCL marks certain steps in a plan as non-executed in order to preserve the conflicting subplans of all characters without damaging the causal soundness of the overall story.

Introduction

Narratologists (Brooks and Warren 1979; Egri 1988; Ryan 1991; Abbott 2008) and researchers in computer story generation (Meehan 1977; Szilas 1999; Barber and Kudenko 2007) agree that conflict is an essential property of interesting stories. Abbott claims that "conflict structures narrative" (2008, p. 55), while Brooks and Warren state that "story means conflict" (1979, p. 65). Despite this universal agreement, little research has been devoted to modeling conflict in either field.

In this paper, we present a formal model of narrative conflict along with a story generation algorithm based on Partial Order Causal Link (POCL) planning. A precise understanding of this phenomenon will further empower narrative systems to generate engaging plots and adapt interactive stories.

Related Work

Historically, most conflict that occurs in commercial story-based games has been crafted by hand at design time rather than procedurally created at run-time. Most research on narrative generation that has included conflict as a component (ranging from early work by Meehan (1977) to more recent work by Thue et al. (2008)), relies on human authors to supply the conflict that drives the story. Systems like Universe (Lebowitz 1985) and Mexica (Perez ỳ Perez and Sharples 2001), for example, combine pre-scripted plot fragments (or plot grammars) to produce whole stories. However, systems which rely on pre-scripted plots or plot fragments model conflict implicitly. As a result, these systems may be challenged to reason about conflict at a the level of detail needed to ensure causal and intentional consistency.

One approach for reasoning about conflict would be to generate stories via adversarial planning or a zero sum game-playing algorithm as suggested by Smith et al (1987). This approach views antagonists as malevolent forces designed only to make trouble for the protagonist. Ideally, they would operate intentionally in pursuit of their own goals and thwart the protagonist only when required to do so.

Zambetta, Nash, and Smith (2007) model conflict in stories as a system of differential equations that simulate an arms race scenario. While this may be helpful as high-level control for the pace of a story, it cannot explain the individual motivations of the participants.

Barber and Kudenko (2007) create dramatic tension in their GADIN system with *dilemmas* — decisions the user must make which will negatively affect at least one character. GADIN detects when these dilemmas are applicable engages the user by forcing them to make difficult decisions. Similarly, Szilas (2003) annotates actions in his IDtension system with a "conflict value" that measures the extent to which a character is acting against its morals.

These methods represent progress toward encoding an understanding of conflict into the story generation process. However, dilemmas are a small subset of the conflicts available to story writers. They arise and get resolved immediately, making it is difficult to model the thematic and extended conflicts which provide important macro-structural features of the story.

POCL Planning and Intentionality

In this paper, we present definitions and descriptions that span 3 algorithms: a conventional POCL-style planner, an intentional planner, and a conflict planer. The later two planning systems build upon the definitions provided by their predecessors in this discussion. Due to space limitations, only the conflict planner is presented in pseudo code, but all the elements of its predecessors are clearly marked.

POCL plans have proven to be effective data structures for representing stories because they explicitly model the events of a story along with the casual and temporal relationships between them (Young 1999). These plan data structures can also effectively serve as proxies for the mental models formed by people reading a narrative or watch-

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Algorithm 1 The CPOCL (Conflict Partial Order Causal Link) Algorithm

CPOCL ($P = \langle S, B, O, L, I \rangle$, Λ, F)
P is a plan, initially the null plan; Λ a set of intentional operators; F a set of flaws, initially open precondition flaws for each end step precondition and unsatisfied intention frame flaws for each start step effect like (`intends` a g).

1: **Termination:** If B or O is inconsistent, fail. If $F = \emptyset$ and P has no orphans, return P. Else, fail.
2: **Plan Refinement:** Choose a flaw $f \in F$. Let $F' = F - \{f\}$.
3: **Goal Planning:** If f is open precondition flaw $f = \langle s_{need}, p \rangle$, let s_{add} be a step with an effect p.
4: Choose s_{add} in one of two ways:
5: **Reuse:** Choose s_{add} from S.
6: **New Step:** Create s_{add} from an operator in Λ with effect p. Let $S' = S + \{s_{add}\}$.
7: For each precondition c of s_{add}, add new open precondition flaw $\langle s, c \rangle$ to F'.
8: Mark s_{add} as non-executed.
9: **Link:** Create causal link $l = s_{add} \xrightarrow{p} s_{need}$. Let $L' = L + \{l\}$, $B' = B \cup \text{MGU}^1(\text{e,p})$, $O' = O + \{s_{add} < s_{need}\}$.
10: **Execution Marking:** If s_{need} is executed, mark s_{add} and all its causal ancestors as executed.
11: **New Frames:** For each effect of s_{add} like (`intends` a g):
12: Create new intention frame $r = \langle a, g, s_{add}, \emptyset, \emptyset \rangle$. Let $I' = I + \{r\}$.
13: Add new unsatisfied intention frame flaw $\langle r \rangle$ to F'.
14: **Intent Flaws:** For each intention frame $r = \langle a, g, \sigma, m, T \rangle \in I'$:
15: If $s_{add} \notin T$ and $s_{need} \in T$ and $a \in A$ for s_{add}, add new intent flaw $\langle s_{add}, r \rangle$ to F'.
16: **Threat Resolution:** If f is threatened causal link flaw $f = \left\langle s \xrightarrow{p} u, t \right\rangle$, choose how to prevent the threat:
17: **Promotion:** Let $O' = O' + \{t < s\}$.
18: **Demotion:** Let $O' = O' + \{u < t\}$.
19: **Restriction:** Add bindings to B' which cause the threatening effect of t not to unify with p.
20: **Satisfaction:** If f is unsatisfied intention frame flaw $f = \langle r = \langle a, g, m, \emptyset, T \rangle \rangle$, let s_{sat} be a step with effect g.
21: Choose s_{sat} the way s_{add} is chosen (**Reuse** or **New Step**).
22: Let $T' = T + \{s_{sat}\}$. Let $r' = \langle a, g, m, s_{sat}, T' \rangle$. Let $I' = I - \{r\} + \{r'\}$.
23: **Intent Planning:** If f is an intent flaw $f = \langle s_{orphan}, r = \langle a, g, m, \sigma, T \rangle \rangle$, choose how to handle s_{orphan}:
24: **Inclusion:** Let $T' = T + \{s_{orphan}\}$. Let $r' = \langle a, g, m, \sigma, T' \rangle$, $I' = I - \{r\} + \{r'\}$, $O' = O + \{m < s_{orphan}\}$.
25: For each causal link $s \xrightarrow{p} s_{orphan} \in L$, if $a \in A$ for s, add new intent flaw $\langle s, r' \rangle$ to F'.
26: **Exclusion:** Do nothing.
27: **Threat Detection:** If any casual link $l \in L'$ is threatened by step $\theta \in S'$, add new threatened causal link flaw $\langle l, \theta \rangle$ to F'.
28: **Recursive Invocation:** Call CPOCL ($P' = \langle S', B', O', L', I' \rangle$, F', Λ).

ing a story unfold (Christian and Young 2004). In our work, POCL plans and the algorithms that generate them also provide us with a precise semantics for conflict in stories and a well-founded method for creating it.

Plans and Their Parts Essential concepts from least commitment, partial order causal link planning are duplicated here for reference. For full details, readers are referred to Weld's overview (1994).

In our work, a plan is a sequence of steps that describes how a world transitions from its beginning, or initial state, to its end, or goal state.

Definition 1. A **state** is a single function-free ground predicate literal or a conjunction of literals describing what is true and false in a story world. The **initial state** completely describes the world before the start of a plan. The **goal state** is a literal or conjunction of literals which must be true at the end of the plan. The initial and goal states together make up the **planning problem**.

Characters, items, and places in the story are represented as logical constants. The actions which materialize between the initial and goal states in a plan are called the plan's **steps**. Actions are created from a library of action operators called a **planning domain**.

Definition 2. An **operator** is a template for an action which can occur in the world. It is a two-tuple $\langle P, E \rangle$ where P is a set of **preconditions**—literals which must be true before the action can be executed—and E is a set of **effects**—literals which are made true by the execution of the action (Fikes and Nilsson 1971). For generality, the literals in an operator's preconditions and effects can have variable terms.

Definition 3. An instance of an operator, called a **step**, represents an actual action that will take place in a story. Variable terms in a step's preconditions or effects must each be bound to a single constant.

Steps in a plan are partially ordered with respect to time.

Definition 4. An **ordering** over two steps is denoted $s < u$, where s and u are steps in the plan and s must be executed before u.

In order for a plan to be successful, each precondition of each step in the plan must be true immediately prior to the step's execution. A **causal link** in a plan indicates when one

[1] Most General Unifier of logical expressions e and p.

step establishes a condition that is needed by a later step's preconditions.

Definition 5. A **causal link** is denoted $s \xrightarrow{p} u$, where s is a step with some effect p and u is a step with some precondition p. A casual link explains how a precondition of a step is met. In other words, p is true for u because s made it so. Step u's **causal parents** are all steps s such that there exists a causal link $s \xrightarrow{p} u$. A step's **causal ancestors** are its causal parents in the transitive closure of the parent relation.

Formally, a plan is a collection of steps and the binding, temporal, and causal constraints over those steps.

Definition 6. A **plan** is a four-tuple $\langle S, B, O, L \rangle$ where S is a set of steps, B a set of variable bindings, O a set of orderings, and L a set of causal links. A **complete plan** is guaranteed to transform the world from initial state to goal. Any plan which is not complete is a **partial plan**.

The POCL Algorithm All of POCL and IPOCL* appear as parts of CPOCL. Line numbers in parenthesis refer to line numbers in the CPOCL algorithm (see Algorithm 1).

POCL planning can be viewed as refinement search (Kambhampati, Knoblock, and Yang 1995), in which a partial plan is incrementally repaired or refined until the plan is either complete (and executable) or inconsistent (and unrepairable). In this process, a partial plan is annotated with **flaws**; each flaw indicates a specific problem with the partial plan that is in need of repair.

The planning process begins by encoding the start state of the problem as a placeholder starting step and encoding the end state of the problem as a placeholder end step.

Definition 7. A **start step** has no preconditions, and effects equivalent to the initial state of the planning problem. An **end step** has no effects, and a precondition for each literal which must be true in the goal state. A **null plan** is the partial plan $P = \langle \{s, e\}, \emptyset, \emptyset, \emptyset \rangle$.

Typical POCL planning has two kinds of flaws:

Definition 8. An **open precondition flaw** indicates that some precondition of a step has not yet been satisfied by a causal link. It is a 2-tuple $\langle s_{need}, p \rangle$, where s_{need} is some step in S and p is a precondition of s_{need} such that no causal link in L has s_{need} as its head and p as its label.

Open precondition flaws are repaired by adding a new causal link to L which has s_{need} as the head (lines 3-9). The tail of the new link can be either a step already in S (line 5) or a new step created from an operator and added to S (lines 6-8). Adding new steps to S often requires adding new open precondition flaws (line 7).

Definition 9. A **threatened causal link flaw** indicates that the condition established by a casual link may be undone before it is needed. It is a 2-tuple $\left\langle s \xrightarrow{p} u, t \right\rangle$, where $s \xrightarrow{p} u$ is a causal link in L, and t is a step in S with effect $\neg p$, and $s < t < u$ is a consistent ordering given O.

Threatened causal links are fixed by preventing the ordering $s < t < u$ (lines 17-18) or by adding bindings to B which prevents the threatening effect of t from unifying with $\neg p$ (line 19).

As we will see below, certain threatened causal links encode valuable information about conflict.

Intentional Planning

Intentions and Intention Frames One limitation of typical planning approaches applied to story generation is that they produce plans in which characters fail to act on their own intentions. General purpose planning systems may construct plans in which protagonists and antagonists work together to reach the author's desired goal state regardless of personal motivation. This problem was first addressed by Riedl and Young (2010), who developed a planning system named IPOCL that creates plans in which every character's actions are clearly linked to one or more of the character's *intentions*. We adopt some of Riedl and Young's definitions here.

Definition 10. An **intention** is a modal predicate of the form `intends`(a,g) where a is an actor and g is a literal that actor a wishes to be true.

Operators in an intentional planning domain are annotated with a list of actors who must agree to carry out that action.

Definition 11. An **intentional operator** is a three-tuple $\langle P, E, A \rangle$ where P is a set of preconditions, E is a set of effects, and A is a set of **actors**, or logical terms which represent characters in the story world. The actors in A must all consent to the execution of any **intentional step**, which is an instance of an intentional operator. Any step for which $A = \emptyset$ is called a **happening**.

Happenings represent accidents or the forces of nature, which have no attributable actors. Actions with one actor in A can be done by an individual (e.g. proposing marriage), but actions with multiple actors in A require mutual consent (e.g. getting married).

Intentional planning seeks to answer *why* and *how* each actor works to achieve its goals.

Definition 12. A **motivating step** is an intentional step which causes an actor to adopt a goal. It has as one of its effects an intention—a modal predicate of the form `intends`(a,g). A **satisfying step** is an intentional step which achieves some actor goal. It must have g as one of its effects.

The steps which materialize between a motivating and satisfying step make up an intention frame.

Definition 13. An **intention frame** is a five-tuple $\langle a, g, m, \sigma, T \rangle$ where a is an actor, g is some literal that a wishes to make true, m is the motivating step with effect `intends`(a,g), σ is the satisfying step with effect g, and T is a set of intentional steps such that $\sigma \in T$, and $\forall s_i = \langle P_i, E_i, A_i \rangle \in S$, $a \in A_i$ (in other words, a must be one of the consenting actors for every step in T).

Simply put, an intention frame describes what step caused an actor to adopt a goal, the steps which that actor took to achieve the goal, and the step which finally achieved the goal. We refer to T as an actor's *subplan* to achieve a goal.

All happenings are considered to be part of a special intention frame f_{env} which has the environment as its actor.

Definition 14. An **intentional plan** is a five-tuple $P = \langle S, B, O, L, I \rangle$ where S is a set of intentional steps, B, O, and L are defined as for a POCL plan, and I is a set of intention frames.

The IPOCL* Algorithm IPOCL*, which differs slightly from Young and Riedl's IPOCL, extends POCL by managing a set of intention frames.

When a step with an effect like `intends`(a,g) is added to the plan, a new intention frame is created with that step as the motivating step (lines 11-13). IPOCL* must later choose a satisfying step to explain how the character goal gets fulfilled. This need translates into a new kind of flaw:

Definition 15. An **unsatisfied intention frame flaw** indicates that a satisfying step has not yet been chosen for an intention frame. It is a 1-tuple $\langle f \rangle$, where f is some intention frame.

After a satisfying step is chosen (lines 20-22), the frame must be populated with all the steps that the actor takes in pursuit of the goal.

When a new causal link is created, it may link a step outside an intention frame ($\notin T$) to a step inside an intention frame ($\in T$). This might indicate that the outside step was taken in pursuit of the frame's goal. If so, the outside step needs to be included in the frame.

Definition 16. An **intent flaw** occurs when a causal link $s \xrightarrow{p} u$ exists such that, for some intention frame $r = \langle a, g, m, \sigma, T \rangle$, $s \notin T$, $u \in T$, and a is an actor who must consent to s. It is a 2-tuple $\langle s, f \rangle$, where s is the step which may need to be included in frame f.

Intent flaws can be solved by either adding the step to the frame (lines 24-25) or ignoring the flaw (line 26).

It is necessary to consider ignoring the flaw to ensure that valid plans are not missed in the search process (Ridel and Young 2010), however this decision creates an orphan.

Definition 17. When, for a step $s = \langle P, E, A \rangle$, there exists and actor $a \in A$, but s is not yet a member of any intention frames for a, then a is said to be an **orphan**.

The presence of orphans makes a plan incomplete. Orphans cannot be repaired directly, only indirectly while fixing other intent flaws. As such, the presence of an orphan is not a flaw.

Conflict as Threatened Causal Links

IPOCL* forms plans in which multiple agents act intentionally to reach the story's goal state. CPOCL is an extension of IPOCL* that explicitly captures how characters can thwart one another in pursuit of their goals, which is the essence of narrative conflict (Herman 2002). CPOCL allows thwarted subplans to fail or partially fail, which addresses a key limitation of IPOCL as identified by Riedl and Young (2010).

Fortunately, POCL algorithms already contain a first-class representation of conflict: the threatened causal link flaw. In classical planning, these must be removed to ensure that a plan is causally sound. CPOCL preserves certain threats, leveraging the information they provide, without damaging the causal soundness of the plan. This is possible when certain steps in the plan are marked as non-executed.

Definition 18. An **executable step** is defined as a four-tuple $\langle P, E, A, x \rangle$, where P is a set of preconditions, E is a set of effects, A is a set of consenting actors, and x is a boolean flag. When $x = true$, the step is an **executed step** which will occur during the story. When $x = false$, the step is a **non-executed step** which will not occur during the story. A happening must be executed—that is, $A = \emptyset$ just when $x = true$.

This notion of executed and non-executed steps is not temporal; it is not related to mixed planning and acting. *Executed* means "will eventually happen during the story," whereas *non-executed* means "will never happen." A non-executed step inside a frame of intention is a step which some actor intended to take but was unable to take due to a conflict.

Because non-executed steps never actually occur, their effects cannot be used to satisfy the preconditions of executed steps. If a causal link $s \xrightarrow{p} u$ exists and s is non-executed, u must also be non-executed.

The presence of non-executed steps means that some threatened causal links are no longer problematic to the plan.

Definition 19. A **conflict** in a plan $P = \langle S, B, O, L, I \rangle$ is a four-tuple $\left\langle a_1, a_2, s \xrightarrow{p} u, t \right\rangle$ such that:

- a_1 and a_2 are actors, possibly the same
- there exists a causal link $s \xrightarrow{p} u \in L$ threatened by step t
- there exists an intention frame $f_1 = \langle a_1, g_1, m_1, \sigma_1, T_1 \rangle$ such that $u \in T_1$
- there exists an intention frame $f_2 = \langle a_2, g_2, m_2, \sigma_2, T_2 \rangle$ such that $t \in T_2$ and $f_1 \neq f_2$
- either t or u (or both) are non-executed steps

In other words, one actor forms a subplan that threatens a causal link in another actor's subplan, and one of the two subplans fails (or both fail). Conflicts are a subset of threatened causal links which are not flaws because there is no chance that they will prevent the plan from proceeding from initial state to goal.

Internal conflict occurs when $a_1 = a_2$ and a character thwarts its own plans. External conflict with other characters occurs when $a_1 \neq a_2$. Conflict with the environment occurs when $f_1 = f_{env}$ or $f_2 = f_{env}$.

The CPOCL Algorithm

No additional flaw types are required in addition to those defined by POCL and IPOCL*.

The start and end steps of the null plan are marked as executed. When a new step is added to a plan, it is initially marked as non-executed (line 8). If a causal link forms from a non-executed step to an executed step, the tail step and its causal ancestors must then be marked as executed (line 10).

Note that this means the complete plan will have only those steps marked as executed that *must* occur to achieve the goal. This follows the least commitment paradigm of POCL planners. It may be possible to mark additional steps as executed without making the plan unsound, and any system using CPOCL is free to do so.

Figure 1: Example CPOCL Problem and Domain

Initial: single(A)∧single(B)∧single(C)∧
loves(A,C)∧intends(A,married(A,C))∧
loves(B,C)∧intends(B,married(B,C))∧has(B,R)
Goal: married(A,C)

lose(?p,?i)
A: ∅
P: has(?p,?i)
E: lost(?i)∧¬has(?p,?i)

find(?p,?i)
A: ∅
P: lost(?i)
E: has(?p,?i)∧¬lost(?i)

give(?p1,?p2,?i)
A: ?p1 ?p2
P: has(?p1,?i)
E: has(?p2,?i)∧¬has(?p1,?i)

marry(?b,?g)
A: ?b ?g
P: loves(?b,?g)∧loves(?g,?b)
∧single(?b)∧single(?g)
E: married(?b,?g)∧
¬single(?b)∧¬single(?g)

propose(?b,?g)
A: ?b
P: loves(?b,?g)∧has(?b,R)
E: loves(?g,?b)∧intends(?g,married(?b,?g))

Example: A Broken Heart

A brief example of unrequited love will help to illustrate the CPOCL process and demonstrate the kinds of plans it produces.

Two men, Abe (A) and Bob (B), both wish to marry the same girl, Cat (C). Bob has already bought an engagement ring (R), but the author has decided that Abe and Cat should end up married. Figure 1 expresses these constraints as a CPOCL problem. The given domain includes 5 actions. Items can be lost, found, and given away. Note that `lose` and `find` are happenings since they do not require consent from any agents. Two people can marry, which requires mutual consent. Lastly, a person can propose, causing the other person to fall in love and intend marriage.

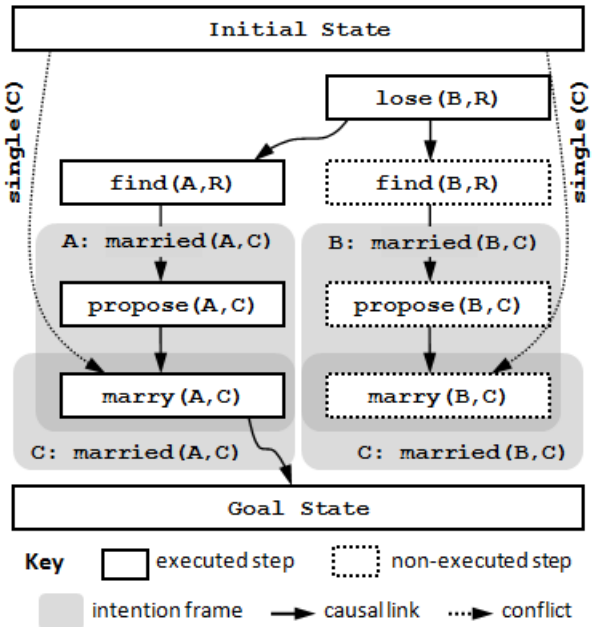
An example of one complete CPOCL solution to this problem is shown in Figure 2. Most causal links originating from the start step are not shown to avoid clutter.

Space limitations prevent us from including figures to demonstrate the construction of this solution, however, we review the process of constructing the plan here. To start, CPOCL is called with the initial plan containing just the initial step, end step, and two empty intention frames motivated by the start step which must explain how Abe and Bob will each achieve a marriage to Cat. The initial plan has three flaws: the open precondition `married(A,C)` and unsatisfied intention frame flaws for the two frames.

The open precondition is repaired by adding a causal link from a new step: `marry(A,C)`. Since the goal step is marked as executed, this step must be marked executed as well. The `marry` action requires consent from two actors, so adding this step creates two orphans. It also creates four new open precondition flaws for its preconditions. All but one are achieved by causal links from the start step. `loves(C,A)` is established by a causal link from a new step `propose(A,C)`. Adding `propose(A,C)` also motivates a new intention frame for Cat which must describe how she will achieve `married(A,C)`.

Abe cannot propose without a ring. The precondition `has(A,R)` is met by a causal link from new step `find(A,R)`, whose precondition is in turn met by a causal link from new step `lose(B,R)`.

Figure 2: Example CPOCL Plan



Three intention frames remain unsatisfied. Both Abe and Cat intend `married(A,C)`, and both of these frames are satisfied by the existing step `marry(A,C)`. This removes two orphans from the plan. It also creates an intent flaw because a causal link joins `propose(A,C)` and `marry(A,C)`. In other words, it may be that Abe proposed in order to marry Cat. The intent flaw is satisfied by including `propose(A,C)` in Abe's intention frame, which removes another orphan.

Bob's intention frame is satisfied by a new step `marry(B,C)`, which remains non-executed. Its precondition `loves(C,B)` is satisfied by a new non-executed step `propose(B,C)`, which creates a new intention frame for Cat that is subsequently satisfied by `marry(B,C)`. The step `marry(B,C)` has effect `¬single(C)` which threatens the causal link that achieves the precondition `single(C)` for the step `marry(A,C)`. This threat is a narrative conflict, not a flaw, because `marry(B,C)` is marked as non-executed.

The rest of the plan is constructed according to the CPOCL algorithm. In the end, only Abe succeeds (thanks to Bob's losing the ring).

Comparing CPOCL to POCL and IPOCL*

POCL's shortest solution to this problem would begin with Bob giving the ring to Abe. While this plan is causally sound, it does not make sense for Bob to help his rival. IPOCL* would fail to solve this problem because the goals of Abe and Bob cannot both be met. CPOCL is able to achieve a solution which guarantees causal soundness and proper character motivation.

When we consider only executed steps, the search space of plans described by CPOCL is a subset of POCL (con-

taining only plans in which characters pursue their individual goals) but a superset of IPOCL* (containing plans in which some character subplans fail). In other words, CPOCL identifies stories that IPOCL* would ignore that may be interesting—even essential, given the importance of conflict. CPOCL accomplishes this by reasoning about how and when conflict arises at the level of atomic story actions.

Although the non-executed steps are not realized in the story, they contain valuable information about the inner worlds of the characters and explain their motivations. They also represent alternate story paths, which can be helpful in story analysis or in adapting interactive plots.

Limitations and Future Work

CPOCL allows conflict to arise, but if all characters can achieve their goals without thwarting one another, these conflict-free plans will appear as part of the CPOCL search space. Since conflict is essential to stories, we may with to disregard these plans. However, doing so can violate expectations about intentionality—agents will be seen to go out of their way to create conflict where none need exist. One solution to this problem is to tailor the initial state and goal state to ensure conflict. Some work on revising story planning problems to fit more interesting plans is already underway (Ware and Young 2010b).

Another major limitation is CPOCL's inability to search for conflicts with specific properties. Our definition of conflict is broad to cover a broad narrative phenomenon, but many genres have specific requirements. We have produced initial work on classifying conflict based on 7 dimensions (participants, subject, duration, directness, intensity, balance, and resolution), and intend to use these values to guide the search process (Ware and Young 2010a).

Lastly, CPOCL does not reason about when an actor should replan. If a character's subplan fails, it may need to form a new plan for the same goal or change its goals. If it replans every time, the story may never end. If it forms the same plan over and over the story many seem less believable. Ongoing work by Fendt (2010) may address this limitation.

Conclusion

In this paper, we described a model of conflict inspired by narratology research and threatened causal links in AI planning. The model informs CPOCL, an algorithm which extends an intentional planner by strategically marking certain actions as non-executed. CPOCL represents progress toward automatically producing stories which are causally sound, reflect believable character actions, and leverage the narrative structure provided by conflict.

Acknowledgments

We wish to thank the National Science Foundation for its support of this research (award IIS-0915598).

References

Abbott, H. 2008. *The Cambridge introduction to narrative*. Cambridge U.

Barber, H., and Kudenko, D. 2007. Dynamic generation of dilemma-based interactive narratives. In *Proc. of AIIDE*.

Brooks, C., and Warren, R. 1979. *Understanding fiction*. Prentice Hall.

Christian, D., and Young, R. M. 2004. Comparing cognitive and computational models of narrative structure. In *AAAI-04*, 385–390.

Egri, L. 1988. *The art of dramatic writing*. Wildside.

Fendt, M. 2010. Dynamic social planning and intention revision in generative story planning. In *Proc. of FDG*.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ* 2(3/4):189–208.

Herman, D. 2002. *Story logic*. U. of Nebraska.

Kambhampati, S.; Knoblock, C.; and Yang, Q. 1995. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *AIJ* 76(1-2):167–238.

Lebowitz, M. 1985. Story-telling as planning and learning. *Poetics* 14(6).

Meehan, J. 1977. Tale-spin, an interactive program that writes stories. In *Proc. IJCAI*.

Perez ỳ Perez, R., and Sharples, M. 2001. Mexica: A computer model of a cognitive account of creative writing. *J. of Experimental & Theoretical AI* 13(2):119–139.

Ridel, M., and Young, R. 2010. Narrative planning: balancing plot and character. *JAIR* 39:217–268.

Ryan, M. 1991. *Possible worlds, artificial intelligence, and narrative theory*. Indiana U.

Smith, T., and Witten, I. 1987. A planning mechanism for generating story text. *Literary and Linguistic Computing* 2(2):119–126.

Szilas, N. 1999. Interactive drama on computer: beyond linear narrative. In *AAAI Fall Symp. on Narrative Intelligence*, volume 144.

Szilas, N. 2003. IDtension: a narrative engine for Interactive Drama. In *TIDSE*.

Thue, D.; Bulitko, V.; and Spetch, M. 2008. Making stories player-specific: Delayed authoring in interactive storytelling. In *ICIDS*, 230–241.

Ware, S., and Young, R. 2010a. Modeling Narrative Conflict to Generate Interesting Stories. In *Proc. AIIDE*.

Ware, S., and Young, R. 2010b. Rethinking Traditional Planning Assumptions to Facilitate Narrative Generation. In *Proc. of AAAI Fall Symp. on Comp. Models of Narrative*.

Weld, D. 1994. An introduction to least commitment planning. *AI magazine* 15(4):27–61.

Young, R. 1999. Notes on the use of plan structures in the creation of interactive plot. In *AAAI Fall Symp. on Narrative Intelligence*, 164–167.

Zambetta, F.; Nash, A.; and Smith, P. 2007. Two families: dynamical policy models in interactive storytelling. In *Proc. of ACIE*, 1–8. RMIT U.