How Authors Benefit from Linear Logic in the Authoring Process of Interactive Storyworld

Kim Dung Dang, Steve Hoffmann, Ronan Champagnat, Ulrike Spierling

How Authors Benefit from Linear Logic in the Authoring Process of Interactive Storyworlds

Kim Dung Dang[1], Steve Hoffmann[2],
Ronan Champagnat[1], Ulrike Spierling[2]

[1]University of La Rochelle - L3i, Avenue Michel Crépeau, 17042 La Rochelle, France
[2]Hochschule RheinMain, University of Applied Sciences, Faculty DCSM,
Unter den Eichen 5, 65195 Wiesbaden, Germany
[1]{kim_dung.dang,ronan.champagnat}@univ-lr.fr
[2]{steve.hoffmann,ulrike.spierling}@hs-rm.de

Abstract. We present a case study of interactive story creation, in which we applied a proof mechanism based on Linear Logic to the authoring process. After initial scenario modeling for dynamic plot generation based on planning, we used the mechanism in iterations of refinements to find possible problems within a huge possibility space of resulting discourses. We describe first results of our case study, discuss prospects and limitations and point out future work.

Keywords: Interactive Storytelling (IS), Authoring, Planning, Linear Logic, Scenario Validation, Proof Graph.

1 Introduction

In Interactive Digital Storytelling (IDS), the use of generative technologies offers to go beyond linear or branched stories. It helps to achieve higher degrees of variability in responses to user interactions, for example by dynamically influencing the sequencing of events or actions. However, generative technologies also raise new challenges for story creators. Authors not only have to conceive stories in novel non-linear ways, they also need to adapt story ideas to technological concepts underlying the used IDS systems. Mostly, abstract models of storyworlds need to be created, describing dynamic behavior through rule-like formulas to be processed by story and behavior engines [19].

On top of abstraction, another authoring challenge is that generative processes possibly create huge amounts of different plot results that are difficult to anticipate. Therefore, authoring tasks need to follow iterative cycles [20] [15]. Starting out from adding content and rules, it is then necessary to test their outcome, try to find flaws, suggest possible improvements and start again by adding or modifying elements and rules. In the following, we refer to this iterative problem solving as '*debugging'*. During debugging a complex generative storyworld, 'manual' methods of testing and keeping an overview can easily reach their limits. As solutions to this problem, automatic methods have been proposed [5] [12]. Using our automatic proof approach proposed in [5], we undertook a case study to explore prospects and limitations of

using such a proving tool within a concrete authoring process. The employed approach is based on *Linear Logic* and able to perform a so-called validation of a scenario. While it is still the responsibility of authors to look for shortcomings in the entertainment value and experiential qualities, we expect that the method provides automatic help in finding specific technical flaws, such as dead ends within marginal but possible courses of events.

In the following, we describe the first results of our case study, discuss prospects and limitations and point out future work. Thereby we bridge technical offerings with motivations of authors. We describe the initial creation of content, followed by its step-by-step transformation into representations interpretable by Linear Logic. Finally, we discuss the results of the performed proving process with our tool and its use for authors.

We use the following technical terms:

- A *discourse* is an ordered sequence of actions/events that is a possible unfolding of a story. One storyworld can be the base to let generate various discourses.
- A *scenario* is a set of all the possible discourses for a story. If we change anything in the storyworld then we will receive a new scenario.
- The *goal* of a story is the authors' desired ending. There may be one or multiple *goal state(s)* and each *goal state* corresponds with one desired possible ending.

2 Related Work

Our case study explores the use of automatic logical proofs in authoring. The case has been built upon an example story, which we initially created as educational material for introducing authors to the concepts of planning [9]. AI-based *Planning* – as described by Russell and Norvig [17] or LaValle [11] – is a prevalent method researched in IDS for drama management, such as by [4] [21] [15] [16] [18]. The role of *Planning* in IDS applications *"is to define the actions or events that must occur during the story so that the world changes from its initial state to some goal state"* [2]. This also means that planners create the order of actions dynamically.

Enabling 'debugging' can be considered a main requirement for IDS authoring tools [14]. There exist a couple of systems and tools in the context of generative IDS designed directly or indirectly with this purpose. The Virtual Storyteller system [20] lets explore debugging as an active part of 'co-creation' during the creative authoring process, in which the generative outcome may influence the authorial intent. Hence it reduces tensions between authorial intent and the partially uncontrollable outcome of story generation. The Emo-Emma authoring tool [15] allows authors to step-by-step choose among possible unfoldings of plans, visualized as a tree graph. Enigma [13] is the conceptual design of an authoring tool that solves the problem of authoring emergent narratives by letting planning-based virtual actors learn during a rehearsal mode. In that mode authors can watch and modify the outcome directly. Both previous examples are more focused on the manual testing of single discourses.

In [22], the use of Petri nets is proposed for game analysis and specification. It has been considered as a starting point for a game design method, to guarantee that the actions carried out by the player maintain the coherence during the game experience.

KANAL [12] is a tool that helps authors to create sound plans by simulating them, checking for a variety of errors and allowing to see an overview of the plan steps or timelines of objects in the plan. However, it was not designed for validating interactive storyworlds.

The current state of the art shows that the problem of managing and validating a scenario has not been efficiently handled yet, although it has been considered as important. Our goal is to offer authors an automatically generated list of all possible discourses in a scenario, (almost) complete necessary statistical information, as well as some suggestions to repair unwanted errors. We believe that our approach – in spite of some limitations – supports authors to achieve well designed results, by letting them validate scenarios more quickly and easily than with manual testing.

3 Case Study and its Initial Authoring Process

Our case study began with the creation of a simple story named 'Harold in Trouble' (explained below). We explored the transformation process of the first linear draft into a story model suitable for AI-based planning, in particular based on the STRIPS approach [6]. In order to illustrate the dynamic plot generation offered by planning, we created a physical card game based on the model to be used as a paper prototype [9]. The story was further test-implemented with an authoring tool equipped with planning software[1] [15]. We expected a greater degree of non-linearity and variation within the conceived content due to the software's replanning possibilities.

The cards were helpful as a first paper prototype in the creation process as well as a didactic method to explain the planner's search process [9] to be understood by novice authors. However, this method soon met natural limitations once the story reached a certain complexity. The planning-based authoring tool [15] then allowed to dynamically visualize more possible variations within our created story domain. However, authors still had to manually click through these dynamically created trees of possibilities, for example if we wanted to detect unwanted dead ends within some possible paths. Therefore, we further explored means to create semi-automatic answers to such questions of consistency within complex storyworlds. We translated the material into a Linear Logic representation, to then test the story structure with the SV Tool, our proof software based on Linear Logic. This translation and the verification will be shown in the next sections.

The conception and creation process of the 'Harold in Trouble' story consisted of several steps and is in detail described in [9]. Here, only a brief summary is given:

- The first written story outline consisted of scenes, characters, their goals and actions. Our story resembles a simple 'James Bond' plot in a comedy genre, in which the criminal super brain 'Silvertoe' blackmails the world. The comic character 'Harold' – a wannabe womanizer – is the clumsy assistant to the agent who negotiates with Silvertoe. The goal for our first implemented scene is to let Harold make Silvertoe so angry that he leaves the party during which the

[1] EmoEmma-AuthoringTool, http://redcap.interactive-storytelling.de/authoring-tools/emo-emma

negotiations take place. Harold, by trying to seduce female party guests, creates chaotic chain reactions that influence Silvertoe's mood negatively.

- The first step to remodel the linear outline as a planning domain was to extract possible meaningful actions and events from the draft. We identified two abstract actions at a high hierarchical level (subsuming other concrete actions) to achieve the intended story experience: 'seducing' and 'creating havoc'. This abstraction allowed for ramifications of actions that were not necessarily linear.
- Further, it was required to describe factual statements that are possible in the storyworld. Story-relevant variable attributes have been identified that would serve as such 'propositions', for example 'Cigarette is lit'. The most important emotional states in our story are the increasing anger levels of Silvertoe.
- Then, a strict order in our designed actions has been partially abandoned in that we described for each action under which circumstances ('preconditions') it can occur and how it changes the storyworld (its 'effects'). For author-intended action chains and connections we had to imply partial orders just by the design of these conditions. Table 1 in section 4 shows how the action 'Harold lights cigarette with a match' is described with a set of preconditions and matching effects.
  In the planning jargon, the elements to be described by authors are the following:
  - *Propositions* (facts): elementary and changeable situations in the storyworld, for example: 'The cigarette is lit'.
  - *Operators* (actions): possibly modify the validity of facts and hereby change the state of the world, for example: 'To light a cigarette'.
  - *Preconditions* declare under which conditions a certain action is allowed to be performed. Example precondition for 'To light a cigarette': 'The cigarette is not lit'.
  - *Effects* declare which facts are added to or deleted from the world state after the action is executed. For example, the action 'To light a cigarette' adds the fact 'The cigarette is lit' and deletes the fact 'The cigarette is not lit'.
  - *Initial state* and *goal state(s)*: The initial state is described by facts that are already in the world when the story begins. One or more goal states are described by facts that have to be in the world to let the story end.

With this information, a planner can generate sequences of actions. The process has two steps: 1) Depending on the current world state, *find* possible actions, and 2) if multiple actions are possible, *choose* the 'best' action. The choice of this 'best' next action is determined by a quality function, which is currently not described here.

After these primary steps, the first version of the paper prototype has been created as a card game (described in detail in [9]). For testing outcomes, action cards can be played after valid comparison of their preconditions with the current world state, which is then influenced by the action's effects by adding or removing proposition cards to/from the table. If more than one card can be played at once, the card game players have to decide which one would be the 'best' fitting. This process simulates the use of a quality function in a fully fledged software planner.

The design of prototypes and their testing – whether by cards or by software – is a crucial task in authoring after an initial design of a storyworld, which is not less important. During many iterations we need to keep an overview of the final outcome regarding consistency. Most importantly, we need to identify unwanted deadlocks, which would end a discourse without reaching any goal state. Initially, we largely

underestimated the amount of possible discourses even with a small set of actions and propositions. This first became obvious by the help of the automated proof tool, described in the following chapters.

4 Case Study Modeling by Means of Linear Logic

Linear Logic [7] is an executable formal model which considers propositions and actions/events as resources that are consumed and/or produced. It is employed to represent the validity of how resources are used when proving an assertion. Linear Logic is well suited to model the natural reasoning through the mechanism of sequent calculus introduced by Gentzen [10]. Besides, linguistic theory uses a subset of Linear Logic (intuitionist multiplicative and non commutative), that corresponds to Lambek calculus [1]. Consequently, Linear Logic provides a framework to model causality as well as resource allocation mechanisms.

In addition, the concept of *linear implication* in Linear Logic is also close to the logic of the concept of *narrative program*, which in Greimas' analysis [8] is an 'abstract formula' employed to express an action/event. The foregoing has inspired us to create a semantic framework of using Linear Logic to model an Interactive Storyworld and then using its reasoning to directly assist authors in managing the generated scenario. In order to reduce the complexity for readers, we do not present the complete approach, but just mention the necessary points applied within the framework of this paper (interested readers get more information in [3]).

- $\otimes$: multiplicative conjunction (times): this connective is used to express a set of propositions.
- $\multimap$: linear implication (imply): a linear implication formula is used to express the validity of transforming propositions from its left side into its right side. As a result, it is used to express an action/event in the storyworld. The transformation of an action/event from its planning representation (the action A01 in the Harold storyworld is described in Table 1 as an example) into a linear implication formula is given in Table 2.

Table 1. Example of an action and its corresponding preconditions and effects.

Preconditions	**Actions**	**Effects**	
		Add	**Delete**
P01: Bored woman is present	**A01: Harold lights cigarette with a match**	P04: Cigarette is lit	P03: Cigarette is not lit
P02: Woman has cigarette in hand			
P03: Cigarette is not lit			

Table 2. Transformation of an action/event from its planning representation into Linear Logic.

Transformation steps	Linear implication formula	
	Left side	**Right side**
Put the propositions in the *Preconditions* list in the left side of the formula; if there is (are) one (or some) proposition(s) in the *Effects/Delete* list that is (are) not included in the *Preconditions* list, continue to put it (them) in the left side of the formula; all propositions in the left side are connected by the connectives ⊗ (*Note: For A01, the proposition P03 in the Effects/Delete list is included in the Preconditions list*)	P01 ⊗ P02 ⊗ P03	
Copy all propositions in the left side of the formula to its right side	P01 ⊗ P02 ⊗ P03	P01 ⊗ P02 ⊗ P03
Add the propositions in the *Effects/Add* list to the right side of the formula	P01 ⊗ P02 ⊗ P03	P01 ⊗ P02 ⊗ P03 ⊗ P04
Delete the propositions in the right side of the formula corresponding to the ones in the *Effects/Delete* list	P01 ⊗ P02 ⊗ P03	P01 ⊗ P02 ⊗ P04

Finally, we receive the linear implication formula P01 ⊗ P02 ⊗ P03 ⊸ P01 ⊗ P02 ⊗ P04 which expresses the action A01 in the storyworld. The transformation for the remaining actions/events in the Harold storyworld is similar.

- A sequent is composed of two parts (separated by ⊢ (turnstile)): the left part includes initial proposition(s) and action(s)/event(s); the right part represents the goal of the story which includes authors' desired possibilities of ending (goal states). Proving a sequent consists in executing one of its actions/events at each step until the set of current available propositions in the left part (in the world state) contains one of the goal states in the right part of the sequent. As a proof expresses the actions/events to be executed, to reach a goal state of a sequent (may be successful or unsuccessful), it is equivalent to a discourse which is an ordered sequence of actions/events that is a possible unfolding of the story. From a sequent, we are able to build its full proof graph, therefore the sequent corresponds to a scenario which is a set of all the possible discourses for a story.

  For instance, the sequent P07, A04, A10, A11, A27, A28, A29 ⊢ P12 has the full proof graph given in Fig. 1. It expresses all the possible discourses in the scenario corresponding to this sequent, in which: P07, P10, P12, P18, P19, P37 are the propositions; P07 in the first node (in the initial state) is the initial proposition and also the current available proposition in the beginning; the propositions in the other nodes are the current available propositions (in the world state) at each step; A04, A10, A11, A27, A28, A29 are the actions/events; the sequent has one goal state including only the proposition P12; the scenario has two discourses (two proofs/branches): A10 → A28 → A27 → A04 and A11 → A29 → A27 → A04.

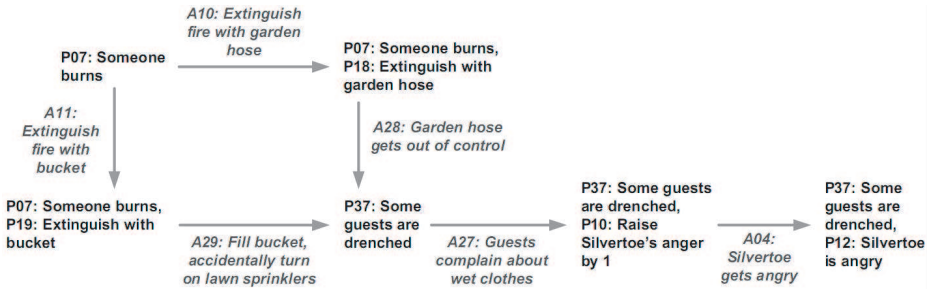

Fig. 1. Full proof graph of the sequent P07, A04, A10, A11, A27, A28, A29 ⊢ P12.

In our case study, after modeling the storyworld (propositions, initial propositions, actions/events and goal states) by means of Linear Logic, we received a sequent which represents its corresponding scenario. The scenario is verified thanks to the proof of this Linear Logic sequent. In the next sections, we will describe in detail how Linear Logic helps authors do that.

5 Scenario Validation and Debugging

As mentioned before, *debugging* means here the process of testing a storyworld and modifying and improving it if unwanted results occur. The developed tool, called Scenario Validation Tool or short SV Tool does not make use of a potential quality function which would be used by a planning system. Therefore the collected information can be seen on a more structural level because they are not the result of a higher level decision process to pick 'best' actions during the planning.

Here we provide a selection of IDS properties and requirements that are found to be important during the authoring process and which influenced the design of the tool:

- *Reachability*: It has to be ensured that every situation of a story can be reached from its initial situation. If certain actions should not be reachable they would be meaningless for the story.
- *Deadlock free*: Finding discourses that are not leading to goal states is crucial to provide stories without or at least a minimal number of deadlocks. Identifying these discourses and providing information to track the reasons is desirable.
- *Sequencing*: The discourses should be coherent regarding the logical order of certain events. It should be verifiable whether the intended order is adhered to.
- *Complexity*: The discourses should have a certain duration or length in terms of numbers of actions. This length, as a measure of complexity, should be verifiable.

Another interesting question is whether loops are existing in a scenario. That question concerns the action control mechanisms in Linear Logic. Indeed, there are two ways to control the execution of actions in a story: (1) an action is only executed once (it will be deleted from the list of actions just after it is executed); (2) an action may be executed many times (which needs more complex modeling). To reduce complexity, each action is deleted after its use, thus making loops impossible.

The SV Tool was tested on a reduced version of the Harold storyworld, which was composed of 24 actions, 46 propositions (8 initial propositions), and 1 goal state (including 1 goal proposition: to reach the anger level 2). The tool provides assistance for authors during the creation process of an interactive storyworld by offering a number of statistical values received after the proving process. These values can be distinguished in simple and parametrizable statistical information and direct debugging information. Here we present these statistics, give examples and explain how they helped us to improve the storyworld.

5.1 Simple Statistics

- **Number and list of (un-)successful discourses (deadlocks)**:
  All discourses can be shown and are distinguished in successful (reaching goal state(s)) and unsuccessful ones (not reaching any goal state(s)). By looking at the unsuccessful branches it is possible to start there to find deadlocks. Our example consists of 132 possible discourses of which 94 (71%) were successful and 38 (29%) were unsuccessful after the first implementation. So these 38 discourses were at first in the scope of further investigations to find possible deadlock reasons. Because the number is still high, we used more of the provided statistical information to reduce the possibilities.
- **Number and list of last actions in unsuccessful discourses (deadlocks)**:
  Taking a look at this list we found that most unsuccessful discourses were ending with action 15 (*'Harold extinguishes burning poodle with floor vase'*) or 30 (*'Poodle falls into pool'*); both appeared 15 times each. They seemed to be dead ends regarding the desired goal to raise the anger level to 2. That means that these two actions are a starting point to improve the story. The simplest solution was to add the proposition 10 (*'Raise Silvertoes anger by 1'*) to the effects of both actions. Running the SV tool after the changes have been made showed a success, because now all discourses were leading to the desired goal.
- **Number and list of longest/shortest (un-)successful discourses (deadlocks, complexity)**:
  Because the discourse lists can be very long, they are filtered to show the shortest and longest successful/unsuccessful discourses. That way, we can identify too short successful discourses that may contain unintended shortcuts. It is also helpful to find deadlocks in very short and very long discourses, which may be a hint for flaws in the very beginning or end of discourses. Our example had 42 successful discourses with the maximal length of 11 actions and 10 with the minimum length of 9 actions. There were also 30 unsuccessful discourses with the maximal length of 8 actions and 8 with the minimal length of 7 actions. These 8 would be a good start to begin with the debugging of the storyworld. Comparing them to the before-mentioned 'suspicious' actions 15 and 30 showed that they were among 6 of these 8 actions. This can be seen as a confirmation of the assumption that very short discourses would be a good starting point for flaw identification.
- **Number and list of successful discourses for each goal state (reachability)**:
  This list provides all successful discourses organized and depending on the goal state(s). It is possible to see how many and which discourses lead to a certain goal

state, as well as which goal states are not reachable. In our example we had only one goal state. However, if multiple goal states exist, it is for example possible that authors want to give one certain goal state a higher probability than another. This can be checked directly to lead to modifications.

- **Number and list of unused propositions (reachability)**:
  These are propositions which are not included in any action, neither in the preconditions, nor in the effects. Normally it is not intended to create propositions which are never used. This information may help to identify actions which could not be performed because of missing propositions. On the other hand, there also may be actions that do not contain appropriate propositions in the effects. In our example we found a lot of them (*'18 propositions among 46 propositions (39%) are not used in any action'*), because we removed actions from the original card game. As a result, unused propositions were left over, because it was hard to track all spare propositions manually in the first instance. In a complex authoring process with several testing iterations, such cases are likely to happen and can be easily corrected with the help of the proof.
- **Number and list of unused actions (reachability)**:
  Unused actions do not appear in any discourse. Normally it is not intended to create actions that are never used, but it may happen during processes of redesign. Identifying unused actions may help to find possible reasons for deadlocks (for instance). In a sophisticated storyworld, authors may have made the reaching of some propositions dependent upon certain actions, which in turn depend upon other true propositions. In our example we found 3 (13%) of the 24 actions were not used. All three of them could be identified as parts of other story chains which were previously taken out of the reduced version, so it was impossible to fulfill their preconditions. By removing them the issue was solved.

5.2 Parametrizable Statistics and Additional Debugging Information

The SV Tool also provides lists and information depending on parameters authors may enter. This way authors can search for certain aspects they are interested in to aid the debugging process and more easily identify possible reasons for flaws:

- **Number and list of discourses containing certain ordered or partially ordered actions (sequencing)**:
  Authors can define a list of actions which have to appear in a certain order within the scenario, because they may want to check if the intended order of some key actions occurs in general or often enough. It is possible to define that these actions have to appear in exactly the given order or just in a partial order, meaning that an action may appear any time after the previous one. In our example we wanted to proof if and how often the action A04 (*'Silvertoe notices trouble and gets angry'*) is happening right after A03 (*'Poodle burns'*) because it seems to be more believable that Silvertoe gets angry right after he notices the burning poodle. The results showed that only 15 (16%) discourses of the 94 successful discourses fit the desired order, so that this point seems to be interesting for improvements.

- **Number and list of (un-)successful discourses containing certain actions**:
  It is also possible to test if certain actions appear in general in the discourses regardless of their order. On one side this is useful if authors have some key actions in mind and want to know if and how often they are happening. On the other side it is possible to take a closer look at 'suspicious' actions which may have been identified in previous proofs. In our example we wanted to know if and how often the actions A01, A02, A03 (the beginning of one story chain in which Harold accidentally enflames the poodle) are happening together with the action A08 (in which Harold does not care about the poodle and waits). The results showed that this is very likely the case, because 78 (83%) discourses of the 94 successful discourses contain all four actions, regardless of their order.
- **Number and list of discourses with a certain length (complexity)**:
  It is not only interesting to know how many and which discourses have a maximal and minimal length, but also to set a desired length and find out how many and which discourses are below or above this length. This information is useful to get an impression of the possible durations of the discourses. In our example we found that 5 actions would be an insufficient length for the discourses. Fortunately no discourse was found that consisted of 5 or less actions. Otherwise these discourses would have been a good start to look for shortcuts or deadlocks**.**
- **Suggest actions which are the possible reason for unsuccessful discourses (deadlocks)**:
  An additional feature of the SV Tool is that it suggests 'suspicious' actions which could be the reason for unsuccessful discourses. They are computed depending on their frequency of appearance in those discourses (the higher the frequency of appearance is, the more 'suspicious' is the action). Authors can take them as suggestions and starting points to identify possible flaws.

6 Discussion

When authors use IDS technologies generating a non-linear order of events, like it is the case in planning-based approaches, they are quickly faced with the issue of keeping an overview of the storyworld structure and the possible outcomes. In our case these outcomes are a high number of possible discourses. The given example, which only used 24 of originally 40 actions in a storyworld, resulted in 132 different discourses. This already caused problems for authors to keep an overview, so that we indeed benefited from using proof software like the SV Tool.

The first question authors would have is: *"Does any of the discourses reach my desired goal state(s)?"* The result can lead to other questions, such as: *"How many of the discourses reach the goal state(s)?"* and *"How can I improve the storyworld to prevent a lot of deadlocks?"* The proposed SV Tool answers the first and second question. It further helps authors to answer the third one by giving hints where to start with a debugging process.

However, the software proof quickly reaches limitations if a storyworld gets too big, for example by rather unconstrained actions with few preconditions. As a consequence of the 'computational explosion', the process may take too long to wait

for results. It is also possible that the resulting data is too big to be explored by authors: When we tried to prove the whole set of 40 actions in the first instance, we achieved more than 200 million possible discourses. In future work we will explore using more constrained worlds, either by more preconditions or/and by quality functions to prioritize actions. Further, also the presentation of the proof results can be enhanced by visualizations and prioritization.

7 Conclusion and Future Work

Debugging is one of the main requirements for IDS authoring tools. IDS authors need to specify large numbers of rules and actions, so that finding possible flaws becomes complex and support is needed. In that context, we presented a case study of an interactive story creation process that was supported by a proof mechanism based on Linear Logic. We showed how a scenario has been modeled for dynamic plot generation based on planning. We transformed it into a Linear Logic representation to be processed by our proof tool.

The results show that such a proof tool is suitable to support authors during the authoring process, namely in the debugging of a created scenario. It helps to identify deadlocks and provides hints to find the reasons for these deadlocks and other unintentional flaws, in order to be corrected in a following iteration.

To overcome the limitations regarding the size and the resulting 'computational explosion', future work will have to find ways to reduce complexity. For example, we could select a storyworld's most important key aspects, reduce it to sub-stories or let the computation stop if certain subgoals or key actions are reached. The SV Tool can further be improved to be more accessible for authors by providing graphical representations of the results and to allow them to easily modify proof parameters.

Acknowledgments. This work has been funded (in part) by the European Commission under grant agreement IRIS (FP7-ICT-231824).

References

1. Abrusci, V. M., Ruet, P.: Non-commutative logic I: The multiplicative fragment. Annals of Pure and Applied Logic, 101(2000) 29--64 (1999)
2. Barros, L. M., Musse, S. R.: Planning algorithms for interactive storytelling. In: Computers in Entertainment, Volume 5, Number 1 (2007)
3. Champagnat, R., Prigent, A., Estraillier, P.: Scenario building based on formal methods and adaptative execution. In: ISAGA 2005 - International Simulation and Gaming Association, Atlanta, USA (2005)
4. Charles, F., Lozano, M., Mead, S. J., Bisquerra A. F., Cavazza M.: Planning Formalisms and Authoring in Interactive Storytelling. In: Technologies for Interactive Digital Storytelling and Entertainment, Proceedings of TIDSE 2003, ZGDV Computer Graphik Edition Band 9, Fraunhofer IRB Verlag, Stuttgart, pp. 216-225 (2003)

5. Dang, K. D., Champagnat, R., Augeraud, M.: Modeling of Interactive Storytelling and Validation of Scenario by Means of Linear Logic. In: Aylett, R. et al. (eds.) ICIDS 2010. LNCS, vol. 6432, pp. 153--164. Springer, Heidelberg (2010)
6. Fikes, R.,Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. In: Artificial Intelligence, 2, pp. 189-208. (1971)
7. Girard, J.-Y.: Linear Logic. Theoretical Computer Science 50(1), 1–101 (1987)
8. Hebert, L.: Tools for Text and Image Analysis: An Introduction to Applied Semiotics, Texto! (2006), http://www.revue-texto.net/Parutions/Livres-E/Hebert_AS/Hebert_Tools.html (last accessed September 13, 2011)
9. Hoffmann, S., Spierling, U., Struck, G.: A Practical Approach to Introduce Story Designers to Planning. In: Proceedings of GET 2011, IADIS International Conference Game and Entertainment Technologies, Rome (2011)
10. Indrzejczak, A.: Jaskowski and Gentzen approaches to natural deduction and related systems. The Lvov-Warsaw School and Contemporary Philosophy, Kluwer Academic Publishers, Printed in the Netherlands, 253--264 (1998)
11. LaValle, S. M.: Planning Algorithms. Cambridge, UK: Cambridge University Press, (2006)
12. Kim, J., Blythe, J.: Supporting plan authoring and analysis. In: Proceedings of the 8th international conference on Intelligent user interfaces, pp. 109-116 (2003)
13. Kriegel, M., Aylett, R., Dias, J., Paiva, A.: An Authoring Tool for an Emergent Narrative Storytelling System. In AAAI Fall, Symposium on Intelligent Narrative Technologies (2007)
14. Medler, B., Magerko, B.: Scribe: A Tool for Authoring Event Driven Interactive Drama. In: Göbel, S., Malkewitz, R., Iurgel, I. (eds.) TIDSE 2006. LNCS, vol. 4326, pp. 139–150. Springer, Heidelberg (2006)
15. Pizzi, D., Cavazza, M.: From Debugging to Authoring: Adapting Productivity Tools to Narrative Content Description. In: Spierling, U., Szilas, N. (eds.) ICIDS 2008. LNCS, vol. 5334, pp. 285--296. Springer, Heidelberg (2008)
16. Riedl, M.O.: Incorporating Authorial Intent into Generative Narrative Systems. In: Proceedings of the AAAI Spring Symposium on Intelligent Narrative Technologies II, Palo Alto, California (2009)
17. Russell, S., Norvig, P.: Artificial Intelligence – A Modern Approach. 2nd Edition, Pearson Education International, Prentice Hall, Upper Saddle River, USA (2003)
18. Skorupski, J.: Storyboard authoring of plan-based interactive dramas. In: Proceedings of the 4th International Conference on Foundations of Digital Games (FDG 2009), Orlando, USA (2009)
19. Spierling, U.: Adding Aspects of "Implicit Creation" to the Authoring Process in Interactive Storytelling. In: M. Cavazza, S. Donikian (Eds.): Virtual Storytelling. Proceedings of ICVS 2007, LNCS, vol. 4871, Springer-Verlag Berlin-Heidelberg, pp. 13-25. (2007)
20. Swartjes, I., Theune, M.: Iterative Authoring Using Story Generation Feedback: Debugging or Co-creation?. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 62-73. Springer, Heidelberg (2009)
21. Thomas, J. M., Young, R. M.: Author in the Loop: Using Mixed-Initiative Planning to Improve Interactive Narrative. In: Workshop on AI Planning for Computer Games and Synthetic Characters (ICAPS 2006), Ambleside, The English Lake District, UK (2006)
22. Vega, L., Natkin, S.: A petri net model for the analysis of the ordering of actions in computer games. In: Proceedings of 4th annual European GAME-ON Conference. London, United Kingdom (2003)