Modeling of Interactive Storytelling and Validation of Scenario by Means of Linear Logic

Kim Dung Dang, Ronan Champagnat, Michel Augeraud

Modeling of Interactive Storytelling and Validation of Scenario by means of Linear Logic

Kim Dung Dang, Ronan Champagnat, and Michel Augeraud

La Rochelle University - L3i, Avenue Michel Crépeau, 17042 La Rochelle, France
{kim_dung.dang, ronan.champagnat, michel.augeraud}@univ-lr.fr

Abstract. Research on Interactive Storytelling has mainly focused on the opposition between a discourse point of view and a character point of view for a story. In this paper, we propose an approach to model Interactive Storytelling using Linear Logic, which is a support for reasoning on both points of view. Then we give an example on an educational game that shows the possibility of validating a scenario of a story, by applying the proof graph of a Linear Logic sequent. Finally we discuss about issues which should be settled and future works to be done on the Linear Logic approach for the IS modeling.

Keywords: Linear Logic, Interactive Storytelling, formal model, validation of scenario, Interactive Storytelling controller.

1 Introduction

Everybody loves a great story, and everybody loves a great game [1]. It seems natural to bring these two very human activities together, to create a new entertainment that is different from either, but is artistically engaging and gives a lot of fun. The quest to find a way of combining storytelling and gaming took on a new life in the last years of the 20th century, when video games started to incorporate realistic environments, and then well-written stories and characters. As computers and consoles improved technically, video games became more visually interesting, their sound and music improved dramatically, and new interaction techniques appeared. As designers sought to make their games more profound and more engaging, they naturally started to incorporate stories and characters into their work, in the desire to create really great games that are blended with really great stories.

However, storytelling unfolding and player's interaction are commonly thought to be opposite [2]. The first relates to a designer's control on the game he has created as the second relates to a player's control on the game he has bought. Research on **I**nteractive **S**torytelling (IS) has mainly focused on this opposition, and now is divided into two major families: the scenario driven approach and the emergent narrative theory [3].

In the first set of families, in order to control the development of a story according to player's interaction, automated directorial control (drama management) is often applied. It aims to guarantee that player's action will keep the story in the field of the pre-defined scenario, so that the story development is both coherent and leads to the

author's desired effects. Various approaches for directorial control have been proposed. For example, [4, 5, 6] aim at comparing player's behavior with a plan, using advanced techniques for anticipating risks of disruption. And hence when a player's action deviates from the pre-computed story plan, the system either replans (gets the story back on track), or makes the player's action have no effect on the story development. [7] extends Thespian (a multi-agent framework for authoring and simulating interactive narratives) with the ability to provide proactive directorial control, using explicit user models for predicting player's behavior and estimating his experience. A director agent is designed to monitor the progress of the story, predict its future development and adjust virtual characters' behaviors and beliefs if necessary, in order to prevent violations to directorial goals. As a consequence, the scenario driven approach only gives a few sets of possibilities to the player, or only proposes alternative paths, and finally the player cannot influence narrative structure. In other words, this control is somehow presented as in contradiction with player's ability to influence the game unfolding in a persistent way.

On the opposite, the emergent narrative theory gives a complete freedom to the player, who can direct the game unfolding through his actions. This means that the story will emerge from characters' actions and its unfolding is not based on a specific narrative structure. For instance, [8] aims at breaking story bounds by proposing models in which the story is not directly written, but emerges from player's interaction with a set of autonomous virtual agents. [9] uses a set of agents designed with complex behaviors which is able to design plans and adapt them with game events. [10] explores "co-creation" role that story generation feedback may play within the creative process of interactive story authoring, in which the outcome of the story generator influences authorial intent. And hence it reduces tensions between authorial intent and the partially uncontrollable outcome of story generation. Emohawk [11] is an educational toolkit which features a narrative scenario with three characters controlled by an appraisal-driven architecture. The narrative is to some degree emergent due to this architecture, as well as due to natural non-determinism and user's interaction. At the same time, it is also partly scripted, which means that it evolves around several pre-determined plot points. Thus we can find that the emergent narrative theory offers a great space of freedom to the player, who may deeply influence the evolution of the virtual world in which he has been immersed. In other words, the player can evolve freely and autonomously. However, its foremost limit is the deriving quality, in term of consistency and pertinence, which highly depends on the player and therefore cannot be guaranteed.

We do not intend to choose between these two approaches, but would prefer to propose an IS modeling strategy that allows combining their strong points (by balancing between the discourse point of view (scenario driven) and the character point of view (emergent narrative)). As one of actual major issues is the development of formalisms supporting the IS modeling, in which these formalisms, ideally, should be computational or support a translation to traditional computational ones. So we focus on Linear Logic because it is the logic of control of events/actions and resources that is well suited to reasoning (in particular when it embeds concepts of high relevance to storytelling, such as causality), it fits narrative formalism approaches and is also close to computational aspects.

This paper begins with a presentation of how Linear Logic is applied to model IS with the discourse point of view as well as with the character one. Then we give the example on the educational game that shows the possibility of validating a scenario of a story, by using the proof graph of a Linear Logic sequent. Lastly, we discuss about the issues which should be settled and the future plans to be done on the Linear Logic approach for the IS modeling. The works mentioned in the paper are an inheritance of the ones realized in [12], in which we have improved the following essential points:

- To enhance the modeling capacity, we have ameliorated the approach by starting from the Greimas' semiotics, which has made the IS modeling more reinforced and correct. Therefore, we may take into account vaster discourse types (scenario driven and emergent) as well as increase the number of considered concepts, and so analyze much more aspects of more complex stories. This has arisen the requirement of using more connectors in comparison to the previous research (that only used $\otimes$, $\multimap$ and $\vdash$), hence we now include the !, & and $\oplus$ ones in the modeling process. As a consequence, we have to improve the translation from a Linear Logic sequent into a Petri net in order to deal with the addition of the ! connector.

- Besides, we have also expressed players' action decisions as inputs, thanks to which we can realize their role easily in the modeling.

- Finally, the contribution of the paper focuses on, first, the definition of a new proof algorithm of Linear Logic sequents and a new execution algorithm of IS models in real time that matches the above extensions; second, detailing the complete example on the validation of a scenario of a story by means of Linear Logic, which at the same time illustrates these ameliorations.

Before beginning the principal contents of the paper, we define some important notions that will be used in the next sections:

- A *story* is a set of entities, events/actions and constraints that solves a set of problems, describes an evolution concerning a set of characters and/or objects. It consists in starting from an initial state, solving the given set of problems in order to reach a final state that is an acceptable conclusion.

- A *discourse* is an ordered sequence of events/actions that is a possible unfolding of a story. Therefore a same story can generate various discourses. This consists in scheduling the events/actions corresponding to the story.

- A *scenario* is a set of all the possible discourses for a specific story. A same story may have many different scenarios, which depends on the validation phase during the game analyze/design process.

2 Linear Logic: A Formalism for the Interactive Storytelling Modeling

2.1 Overview of Linear Logic

Linear Logic has been introduced by Girard [13] as a restriction of Classical Logic (a non-standard logic) that allows taking into account the notion of events/actions and representing control of resources. For instance, the "A ⊸ B" formula can be interpreted as: "The consumption of A produces B, and so we only have either A or B", whereas in Classical Logic, the "A → B" formula can be interpreted as: "If A is true then B also, and so the both is true". Classical Logic is actually designed to model a static world, where formulas are interpreted as eternal truth that, once established, last forever and can be used again and again in derivation of other formulas.

Unlike Classical Logic, Linear Logic is not employed to determine whether an assertion is true or not, but rather the validity of how formulas are used (and then consumed) when proving an assertion. It has also been introduced as the logic of resources [14]. Indeed, Linear Logic is well suited to derive a formal model to partially ordered problems with resource sharing. In Linear Logic, formulas are treated as resources that are produced and consumed. The consequence is that every formula should be used exactly once in a derivation. Thus contrary to Classical Logic, where sequents are composed of automatically expandable sets of formulas, in Linear Logic, sequents are composed of automatically non-expandable sets of formulas.

[14] gives a complete presentation of the connectors of Linear Logic and their introduction rules. In this paper, we only mention the ones which will be used in the next sections:

- ⊸: linear implication (imply), expresses the possibility of deduction. Example: "1€ ⊸ 1kg strawberries" means that we can give 1€ to buy 1kg strawberries.
- !: of course, expresses the infinite available number of a resource or the infinite occurring number of an event/action. Example: "1€ ⊸ !(strawberries)" means that we can give 1€ to buy as much strawberries as we want.
- ⊗: multiplicative conjunction (times), expresses a set of resources (not ordered). Example: "1€ ⊸ 1kg strawberries ⊗ 1kg tomatoes" means that we can give 1€ to buy 1kg strawberries and 1kg tomatoes.
- &: additive conjunction (with), expresses an external choice to the system (for instance coming from the player). Example: "1€ ⊸ tea & coffee" means that we can choose tea or coffee when we give 1€ to an automatic machine.
- ⊕: additive disjunction (plus), expresses an internal choice to the system (for instance coming from an IS controller). Example: "1€ ⊸ tea ⊕ coffee" means that it is the automatic machine which will decide if we receive tea or coffee when we give it 1€.
- ⊢: turnstile, separates the left part (antecedent) and the right part (consequent) of a sequent. Example: "D ⊗ (D ⊸ E) ⊢ E" (or "D, (D ⊸ E) ⊢ E") means the possibility to produce a copy of "E" by consuming "D" and "D ⊸ E" (we can

substitute the ⊗ connector by the comma (,) in the left part of a sequent to be easily readable).

2.2 Proving a Linear Logic Sequent

Gentzen introduced the sequent calculus as a tool for studying natural deduction [15]. It aims at proving the consistency of formula. A sequent is a $\Gamma \vdash \Lambda$ expression, where Γ and Λ are sequences of formulas; the initial sequent is $A \vdash A$, where A is any atom.

Proving a sequent consists in rewriting the sequent, by making a substitution of its formulas in order to obtain initial sequents. In our approach, we base the proof strategy on using a Petri net (which is described in [16]). It has been shown [17, 18] that there is an equivalence between a Linear Logic sequent and a Petri net (with some restrictions). We perform the translation (inspired from [17]) of a Linear Logic sequent into a Petri net and supplement several ameliorations as follow:

- An atom is modeled by a place. The availability with any number of the atom in the left part of the sequent is expressed by the presence of the corresponding token number in the place. We add a parameter to the place to show its token number. If the token number is infinite (which means that the availability of the atom is infinite (for instance, !A)), the value of the parameter is "i".
- A linear implication formula in the left part is modeled by a transition. We add a parameter to the transition to show the occurring number of the formula. If it is infinite (for instance, !(A ⊸ B)), the value of the parameter is "i".
- The left part of the sequent involves a set of available atoms and a set of linear implication formulas. It corresponds to the initial marking and a set of transitions, respectively.
- The right part of the sequent is equivalent to the expected marking.

Thus the demonstration of a Linear Logic sequent corresponds to the firing of a sequence of transitions, starting from the initial marking and going to the expected marking. This process will be executed by the token player algorithm [19]. For example, we consider the following sequent: !A, C, D, A ⊗ C ⊸ B ⊗ C, !(D ⊸ E), E ⊗ C ⊸ F ⊗ C ⊢ B ⊗ C ⊗ F, the equivalent Petri net is given in Fig. 1 where: A, B, C, D, E, F are the atoms; A is available and infinite; C, D are available and their available number is 1; B, E, F are not available; A ⊗ C ⊸ B ⊗ C, D ⊸ E, E ⊗ C ⊸ F ⊗ C are the transitions; the occurring number of the A ⊗ C ⊸ B ⊗ C and E ⊗ C ⊸ F ⊗ C formulas is 1; the occurring number of the D ⊸ E formula is infinite. If after firing of a sequence of transitions starting from the initial marking (!A, C, D), and going to the marking corresponding to the right part of the sequent (B ⊗ C ⊗ F means that there are one token in place B, one token in place C and one token in place F), then this sequent is demonstrated.
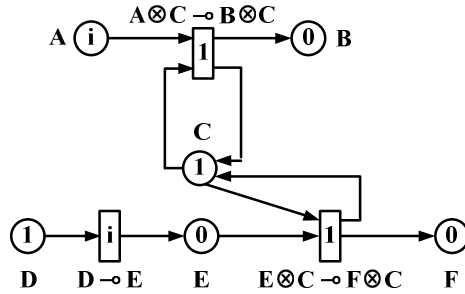
Fig. 1. Equivalent Petri net of a sequent.

2.3 Interactive Storytelling Modeling by means of Linear Logic

According to Hebert [20], Greimas developed the concept of narrative program which is an abstract formula used to represent an action. More concretely, doing (action) is defined as a temporal succession from one state to the opposite state, effected by any agent (the subject of doing) and affecting any patient (the subject of state). A state may be broken down into a subject of state and an object of state and the junction between them, which is either a conjunction (the subject is with the object) or a disjunction (the subject is without the object). The subject of doing may or may not correspond to the subject of state; in other words, what accomplishes the action may or may not be what is affected by it. A narrative program array is composed of at least two narrative programs, between which at least one temporal relation (succession, simultaneity) or one logical relation (simple or reciprocal presupposition, mutual exclusion, comparing/compared, etc.) is identified. Hence we find that a discourse (an ordered sequence of events/actions) may be described by a narrative program array.

Greimas also developed the concept of actantial model, which can be used to break an action down into six facets, or actants: (1) The subject (for example, the Prince) is what wants or does not want to be conjoined with (2) an object (the rescued Princess, for example). (3) The sender (for example, the King) is what instigates the action, while (4) the receiver (for example, the King, the Princess, the Prince) is what benefits from it. Lastly, (5) a helper (for example, the magic sword, the horse, the Prince's courage) helps to accomplish the action, while (6) an opponent (the witch, the dragon, the Prince's fatigue, a hint of terror) hinders it. Therefore this model is a tool that can theoretically be used to analyze any real or thematized action, in which we assign each element of the action being described to one of the actantial classes. An action may be analyzed by one (or a number of) actantial model(s) and so, a same action can often be seen from several different perspectives (for example, from the subject's point of view, or his rival, the anti-subject's).

Originated from those ideas, the L3i laboratory has proposed the approach to model IS by means of Linear Logic, in which we balance between the discourse point of view and the character one (by taking into account the influence of the both in the modeling process), as follow:

- Characters are modeled by atoms in the left part of the sequent. An atom corresponds to a state of a character considering a certain point of view. So a

character can be modeled by various atoms giving different points of view of the character (one for each layer). The size of the state vector of the character may vary during the unfolding of the story.

- States of the story are modeled as atoms in the left part of the sequent.
- Resources are modeled as atoms in the left part of the sequent. If the available number of a resource is infinite, it is modeled by the ! connector (for example, !A).
- The availability of the characters' states, the states of the story and resources are considered as the availability of the corresponding atoms in the left part of the sequent.
- Players' action decisions are expressed by inputs. This means that the players decide their occurrence in the unfolding of the story by entering the inputs. These inputs are modeled as atoms in the left part of the sequent and will become available after being entered into the program by the players.
- An additive conjunction formula in the left part of the sequent represents players' choices in the progress of the story. We will split this sequent into subsequents corresponding to each element of that additive conjunction formula. For instance, the A, C, $((A \multimap B) \& (D \multimap E))$, $C \multimap F \vdash F$ sequent is split into two A, C, $A \multimap B$, $C \multimap F \vdash F$ and A, C, $D \multimap E$, $C \multimap F \vdash F$ subsequents. Then we only consider the created subsequents and the received result of the original sequent will be the union of the results of all the subsequents.
- A linear implication formula in the left part of the sequent models an event/action of the story (that makes some partial states/resources evolve). If the occurring number of an event/action is infinite, it is modeled by the ! connector (for example, $!(A \otimes C \multimap B \otimes D)$).
- An outcome (goal) of the story corresponds to an atom, or a set of atoms (connected between them by the $\otimes$ connector) in the right part of the sequent. Outcomes of the story are connected between them by the $\oplus$ connector.
- A proof models a discourse of the story. As the proof is not unique, we may deduce various ones.
- The sequent gives all the possible discourses (scenario) for its corresponding story.

For instance, the A, C, D, G, $A \otimes C \multimap B \otimes F$, $D \multimap E$, $G \multimap H \vdash H \oplus (B \otimes F \otimes E)$ sequent has the proof graph given in Fig. 2 (simplified to the substitution of the linear implication formulas) where L_1: $A \otimes C \multimap B \otimes F$; L_2: $D \multimap E$; L_3: $G \multimap H$. It expresses all the possible discourses (scenario) for the story corresponding to this sequent, in which: L_1, L_2, L_3 are the events/actions; H and $B \otimes F \otimes E$ are the outcomes; an example of two of the discourses (two proofs) are the (A, C, D, G, L_1, L_2, L_3), (B, F, D, G, L_2, L_3), (B, F, E, G, L_3) and (A, C, D, G, L_1, L_2, L_3), (A, C, D, H, L_1, L_2) paths.

$$B, F, D, H, L_2 \qquad B, F, E, G, L_3$$

$$\multimap L_3 \qquad \multimap L_2 \qquad \multimap L_1$$

$$B, F, D, G, L_2, L_3 \qquad A, C, E, G, L_1, L_3 \xrightarrow{\ \multimap L_3\ } A, C, E, H, L_1$$

$$\multimap L_1 \qquad \multimap L_2$$

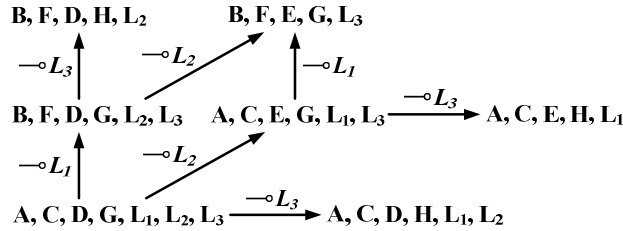$$A, C, D, G, L_1, L_2, L_3 \xrightarrow{\ \multimap L_3\ } A, C, D, H, L_1, L_2$$

Fig. 2. Proof graph of a sequent.

To apply this approach on creating interactive games, our purpose is the development of a system assuring a set of objectives: The players do not feel constrained by the game but they can determine its evolution; the virtual world must provide a coherent environment that is appropriate for players' actions; the progress of the game has to respect a structure of discourse (introduction → giving problems → solving them step by step → conclusion). To do this, we have proposed in [3] a system architecture being composed of two components: an IS controller and an IS rendering (Fig. 3). The IS controller aims to manage the unfolding of the story (discourse) by taking into account players' actions, states of the game, and game designer's goals. The execution of this process is based on reasoning of a Linear Logic sequent that models the game. Its influence is on two levels: non real-time and real-time. Firstly, for the non real-time one (the game analyze/design phase), the Linear Logic model allows validating a scenario of the game according to the discourse point of view, which will be clearly described in the next section. Secondly, for the real-time one (during the game execution), an automatic translator transforms the received Linear Logic sequent into a Petri net. Then the Petri net performs the computation of the model in real time (thanks to the token player algorithm), and gives one (or some) decision(s) by which the IS controller (with the help of an expert system), can ask the IS rendering to build suitable interfaces for the game. Thus the IS controller is able to follow the execution track and the structure of the discourse, so may detect unsatisfactory events. And hence it will operate correctly (be never in an incoherent situation), as well as assure players' freedom and their determinant role on the game, together with the reasonableness of the generated discourses.
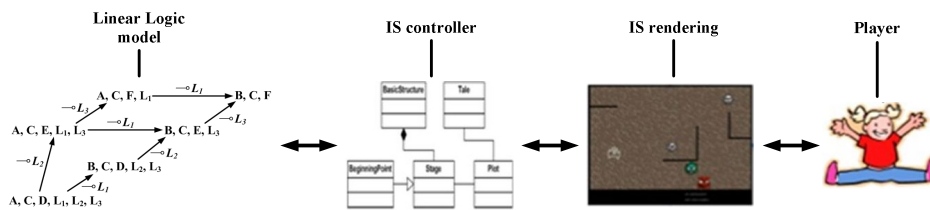

Fig. 3. Architecture of the system.

3 Validation of Scenario: An Example based on Linear Logic

In addition to the possibility of IS modeling as has been presented above, we can use the Linear Logic model in order to validate a scenario of a story. Indeed, the following example will illustrate how to apply the proof graph of a sequent to this goal.

Let us consider the educational game which warns of domestic electrical accidents, whose objective consists in causing an electric shock for the player. At first, the game designer anticipates that the player, from his initial position, will go in the kitchen, where the IS controller will start the strategy of causing the electric shock for him, by tools in there such as fridge, microwave oven, electric cooker,… However, what will happen if the player does not go in the kitchen, but has other choice, for instance, staying at the initial position to work or going in the bathroom? We model this game by means of Linear Logic as follow:

- *States of the story:* Si - Being at the initial state; Sk - Starting the strategy of causing the electric shock for the player in the kitchen; Sr - Reaching the goal (the player got the electric shock).
- *States of the player:* Pi - Being at the initial state; Pw - Working at the initial position; Pk - Being in the kitchen; Pb - Being in the bathroom; Pe - Getting the electric shock.
- *Inputs of the player (his action decisions):* Iw - Deciding to work at the initial position; Ik - Deciding to go in the kitchen; Ib - Deciding to go in the bathroom.
- *Events/actions of the story:* The player decides to work at the initial position (E1: Pi ⊗ Iw ⊸ Pw); the player decides to go from the initial position in the kitchen (E2: Pi ⊗ Ik ⊸ Pk); the player decides to go from the initial position in the bathroom (E3: Pi ⊗ Ib ⊸ Pb); the IS controller starts the strategy of causing the electric shock for the player in the kitchen (E4: Pk ⊗ Si ⊸ Pk ⊗ Sk); the player gets the electric shock in the kitchen (E5: Pk ⊗ Sk ⊸ Pe ⊗ Sr).
- Finally, we have the following sequent in which the formulas are replaced by their label: Pi, Si, (E1 & (E2, E4, E5) & E3) ⊢ Pe ⊗ Sr. It gives all the possible discourses (scenario) of the story. Because the sequent has three choices concerning the & connector, we split it into three subsequents: Pi, Si, E1 ⊢ Pe ⊗ Sr (the player decides to work at the initial position by entering Iw); Pi, Si, E2, E4, E5 ⊢ Pe ⊗ Sr (the player decides to go from the initial position in the kitchen by entering Ik); and Pi, Si, E3 ⊢ Pe ⊗ Sr (the player decides to go from the initial position in the bathroom by entering Ib). Their proof graph is represented in Fig. 4.
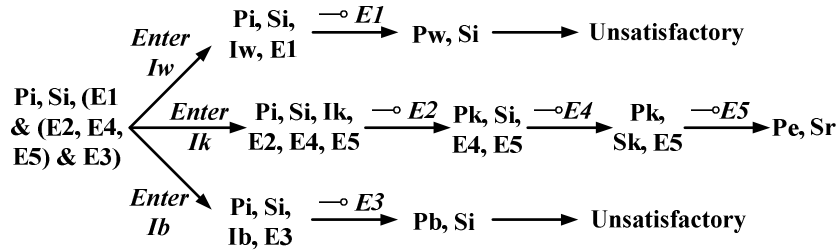
Fig. 4. Proof graph of the sequent before the validation phase.

We can see that there are two paths which lead to the unsatisfactory endings of the goal of the game (if the player decides to work at the initial position or go from the initial position in the bathroom). Therefore we have two possibilities:

- either remove the actions of the player causing the unsatisfactory endings (E1 - Working at the initial position and E3 - Going in the bathroom), but that may restrict the player's freedom, so we do not choose this possibility;
- or add two new plots to the story: If the player decides to work at the initial position, the IS controller will ask him to go in the kitchen (for example, a non-player character asks him to take an apple in the fridge); if the player decides to go in the bathroom, the IS controller will start the strategy of causing the electric shock for him in there (by tools such as hair-dryer, light bulb,...).

Thus we remodel the game as follow:

- *States of the story:* Si - Being at the initial state; Sa - Asking the player (who is working at the initial position) to go in the kitchen; Sk - Starting the strategy of causing the electric shock for the player in the kitchen; Sb - Starting the strategy of causing the electric shock for the player in the bathroom; Sr - Reaching the goal (the player got the electric shock).
- *States of the player:* Pi - Being at the initial state; Pw - Working at the initial position; Pk - Being in the kitchen; Pb - Being in the bathroom; Pe - Getting the electric shock.
- *Inputs of the player (his action decisions):* Iw - Deciding to work at the initial position; Ik - Deciding to go in the kitchen; Ib - Deciding to go in the bathroom.
- *Events/actions of the story:* The player decides to work at the initial position (E1: Pi ⊗ Iw ⊸ Pw); the player decides to go from the initial position in the kitchen (E2: Pi ⊗ Ik ⊸ Pk); the player decides to go from the initial position in the bathroom (E3: Pi ⊗ Ib ⊸ Pb); the IS controller asks the player (who is working at the initial position) to go in the kitchen (E4: Pw ⊗ Si ⊸ Pw ⊗ Sa); the player (who is working at the initial position) goes in the kitchen according to the asking of the IS controller (E5: Pw ⊗ Sa ⊸ Pk ⊗ Sa); the IS controller starts the strategy of causing the electric shock for the player in the kitchen (E61: Pk ⊗ Si ⊸ Pk ⊗ Sk; E62: Pk ⊗ Sa ⊸ Pk ⊗ Sk); the IS controller starts

the strategy of causing the electric shock for the player in the bathroom (E7: Pb ⊗ Si ⊸ Pb ⊗ Sb); the player gets the electric shock in the kitchen (E8: Pk ⊗ Sk ⊸ Pe ⊗ Sr); the player gets the electric shock in the bathroom (E9: Pb ⊗ Sb ⊸ Pe ⊗ Sr).

- Finally, we have the following sequent in which the formulas are replaced by their label: Pi, Si, ((((E1, E4, E5, E62) & (E2, E61)), E8) & (E3, E7, E9)) ⊢ Pe ⊗ Sr. It gives all the possible discourses (scenario) of the story. Because the sequent has three choices concerning the & connector, we split it into three subsequents: Pi, Si, E1, E4, E5, E62, E8 ⊢ Pe ⊗ Sr (the player decides to work at the initial position by entering Iw); Pi, Si, E2, E61, E8 ⊢ Pe ⊗ Sr (the player decides to go from the initial position in the kitchen by entering Ik); and Pi, Si, E3, E7, E9 ⊢ Pe ⊗ Sr (the player decides to go from the initial position in the bathroom by entering Ib). Their proof graph is represented in Fig. 5.
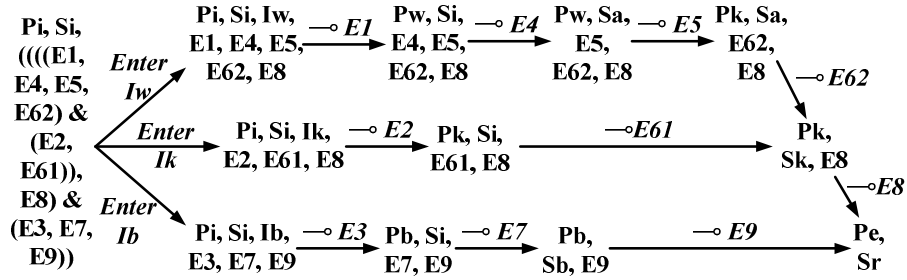

Fig. 5. Proof graph of the sequent after the validation phase.

We can find that all the paths lead to the satisfactory endings of the goal of the game (the player got the electric shock). Thus this example demonstrates the Linear Logic model is effective in determining whether the proof graph, on where we execute the planning algorithm, contains error paths. It also shows that to increase the players' freedom, we have to augment the number of their choice possibilities (by adding new plots). Besides, thanks to the use of Linear Logic model for reasoning, we are able to know the form of a scenario of a story as well as improve it easily.

4 Conclusion

In this paper, we have presented the approach for the IS modeling by means of Linear Logic, that combines the discourse point of view and the character one, in which we have supplemented several important ameliorations in comparison to the works before, therefore extended the capacity of Linear Logic in modeling IS. Then we have given the example on the educational game in order to show the possibility of validating a scenario of a story (by applying the proof graph of a Linear Logic sequent), as well as illustrate the mentioned ameliorations in the approach. We also realize that, although Linear Logic offers promising perspectives for the development of formalisms supporting the IS modeling, a number of issues should be settled.

As stated before, we base the sequent proof strategy and the execution of the model on using a Petri net, which has some restrictions (it is only possible for several connectors and so cannot model really complex systems). To overcome this inconvenience, we are implementing a proof algorithm that may apply to every case of the Linear Logic sequents. In parallel, we must also propose other method to execute the computation of the model in real time.

Concerning the scenario validation, more difficulties arose. The first, we have to build all proof graph of a sequent to validate its scenario, this meets the combinatorial explosion problem. We have implemented (and will supplement) some heuristics together with solutions based on analyzing the form of the scenario and the structure of the discourse, in order to reduce the search space. The second, we actually can validate the soundness of a scenario but do not yet validate its quality. In other words, how to show an "interesting scenario" for a game?

Another particular problem is that the geographical aspects (for example, location of characters and resources) are not yet modeled by Linear Logic. Lastly, concerning the future works to be done on the Linear Logic approach, we will incorporate the received results into the IS Engine design of the IRIS Network of Excellence [21].

References

1. Glassner, A.: Interactive Storytelling: Techniques for 21st Century Fiction. A K Peters, Ltd. Natick, Massachusetts, USA (2004)
2. Juul, J.: A Clash Between Game and Narrative. In: Digital Arts and Culture Conference. Bergen (1998)
3. Delmas, G., Champagnat, R., Augeraud, M.: From Tabletop RPG to Interactive Storytelling: Definition of a Story Manager for Videogames. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 121--126. Springer, Heidelberg (2009)
4. Magerko, B.: Story Representation and Interactive Drama. 1st Artificial Intelligence and Interactive Digital Entertainment Conference. Los Angeles, California (2005)
5. Riedl, M. O., Saretto, C. J., Young, R. M.: Managing interaction between users and agents in a multi-agent storytelling environment. In: AAMAS, pp. 741--748. ACM, New York (2003)
6. Young, R. M., Riedl, M. O., Brandy, M., Martin, J., Saretto, C. J.: An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments. Journal of Game Development, 51--70 (2004)
7. Si, M., Marsella, S. C., Pynadath, D. V.: Directorial Control in a Decision-Theoretic Framework for Interactive Narrative. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 221--233. Springer, Heidelberg (2009)
8. Aylett, R.: Narrative in Virtual Environments – Towards Emergent Narrative. In: Proceedings of the AAAI Symposium on Narrative Intelligence, pp. 83--86. AAAI Press, Menlo Park (1999)
9. Cavazza, M., Charles, F., Mead, S. J.: Character-based Interactive Storytelling. In: IEEE Intelligent Systems, special issue on AI in Interactive Entertainment, pp. 17--24 (2002)

10. 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)
11. Brom, C., Bida, M., Gemrot, J., Kadlec, R., Plch, T.: Emohawk: Searching for a "Good" Emergent Narrative. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 86--91. Springer, Heidelberg (2009)
12. 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)
13. Girard, J.-Y.: Linear Logic. Theoretical Computer Science 50(1), 1--101 (1987)
14. Lafont, Y.: Introduction to Linear Logic. In: Lecture notes from TEMPUS Summer School on Algebraic and Categorical Methods in Computer Science. Czech Republic (1993)
15. Indrzejczak: 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)
16. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, Vol. 77, No. 4, pp. 541--580 (1989)
17. Girault, F.: Using linear logic to formalize Petri nets (in French). PhD Thesis, University of Toulouse III, France (1997)
18. Küngas, P.: Using Linear Logic Planning to Make Knowledge Bases Reactive. In: Proceedings of Seventh Symposium on Programming Languages and Software Tools, pp. 135--148. Szeged, Hungary (2001)
19. Julia, S., Valette, R.: Real time scheduling of batch systems. Simulation Practice and Theory, Volume 8, Issue 5, 307--319 (2000)
20. Hebert, L.: Tools for Text and Image Analysis: An Introduction to Applied Semiotics. Texto! (2006) [online], http://www.revue-texto.net/Parutions/Livres-E/Hebert_AS/Hebert_Tools.html
21. Cavazza, M., Champagnat, R., Leonardi, R., the IRIS Consortium.: The IRIS Network of Excellence: Future Directions in Interactive Storytelling. In: Iurgel, I., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, pp. 8--13. Springer, Heidelberg (2009)