Aide à la réalisation de systèmes de pilotage de narration interactive : validation d'un scénario basée sur un modèle en logique linéaire

Kim Dung Dang

UNIVERSITÉ DE LA ROCHELLE

ÉCOLE DOCTORALE S2IM

Sciences et Ingénierie pour l'Information, Mathématiques

Laboratoire L3I

Informatique, Image et Interaction

THÈSE

présentée par :

Kim Dung DANG

kim_dung.dang@univ-lr.fr

soutenue publiquement le 30 avril 2013
pour l'obtention du grade de Docteur de l'Université de La Rochelle

discipline : Informatique et Applications

sous la direction de M. Michel AUGERAUD et M. Ronan CHAMPAGNAT

AIDE À LA RÉALISATION DE SYSTÈMES DE PILOTAGE DE NARRATION INTERACTIVE : VALIDATION D'UN SCÉNARIO BASÉE SUR UN MODÈLE EN LOGIQUE LINÉAIRE

JURY :

Jean-Marc LABAT	Professeur des universités Université Pierre et Marie Curie	*Président du jury*
Marc CAVAZZA	Professor, Assistant Dean for Research School of Computing, Teesside University	*Rapporteur*
Stéphane NATKIN	Professeur titulaire de la chaire Systèmes Multimédia CNAM Paris	*Rapporteur*
Michel AUGERAUD	Professeur des universités Université de La Rochelle	*Directeur de thèse*
Ronan CHAMPAGNAT	Maître de conférences, HDR Université de La Rochelle	*Encadrant scientifique*

Thèse réalisée au Laboratoire Informatique, Image et Interaction
Pôle Sciences & Technologies, Université de La Rochelle
Avenue Michel Crépeau
17042 La Rochelle cedex 01
Tél : +33 5 46 45 82 62
Fax : +33 5 46 45 82 42
Web : http://l3i.univ-larochelle.fr

Sous la direction de M. Michel AUGERAUD michel.augeraud@univ-lr.fr
M. Ronan CHAMPAGNAT ronan.champagnat@univ-lr.fr

Financement Cette thèse a été financée par la Commission européenne dans le cadre de la convention de subvention du réseau d'excellence recherche **I**ntegrating **R**esearch on **I**nteractive **S**torytelling (IRIS - FP7.) [19].

REMERCIEMENTS

Je tiens avant tout à remercier M. Michel AUGERAUD et M. Ronan CHAMPAGNAT, mes directeurs de thèse, pour tout : la supervision, la correction, les conseils, les discussions, les explications, l'aide, l'encouragement, le soutien et la sympathie qu'ils m'ont réservé. Grâce à eux, j'ai pu finaliser mon travail.

Je remercie sincèrement aussi M. Jean-Marc LABAT, qui a accepté de présider le jury de cette thèse, M. Marc CAVAZZA et M. Stéphane NATKIN, qui ont accepté d'en être les rapporteurs.

Je tiens particulièrement à remercier Guylain DELMAS, ancien thésard du laboratoire L3I, dont les conseils, les réponses et les documents m'ont apporté l'approfondissement de connaissances concernant la thèse.

Mes remerciements sincères vont également à Kathy THEUIL, Jennifer DE LA CORTE GOMEZ, Isabelle HIRSCH, Antoine MERCIER, Dominique LIMOUSIN et Patrice DENIS, pour leur aide grâce à laquelle je peux me familiariser avec la vie au laboratoire L3I, la vie à l'école doctorale et la vie en France (un principe vraiment important pour un étranger dont le niveau de français n'était pas trop bon comme pour moi).

Je voudrais aussi remercier les autres membres du laboratoire L3I, surtout Pascal ESTRAILLIER, Jean-Marc OGIER, Rémy MULLOT, ceux du bureau 131 (Wafa, Dounia, Bich, Antoine, Patrice, Omar, Guillaume) et ceux de l'équipe ASPIC, qui ont créé un environnement idéal et convivial où j'ai travaillé pendant les quatre années dernières.

Je remercie également fortement la Commission européenne qui, dans le cadre de la convention de subvention du réseau d'excellence recherche IRIS (FP7-ICT-231824), a financé mes travaux pour que je puisse réaliser la thèse.

Par ailleurs, je tiens à remercier l'ensemble de mes collègues dans ce projet IRIS [19], notamment Marc CAVAZZA, Fred CHARLES, Julie PORTEOUS, Ulrike SPIERLING et Steve HOFFMANN, ceux qui m'ont suggéré plusieurs bonnes idées sur la validation de scénario.

Enfin et bien sur, mes pensées vont à ma famille et à mes amis qui me soutiennent toujours et m'encouragent pendant toutes ces années d'études. Je les en remercie du fond du cœur.

Kim Dung DANG

La Rochelle, France, mars 2013

RÉSUMÉ

L'objectif de cette thèse est de fournir un modèle, une méthode et un outil d'aide à la réalisation de scénarios interactifs. Cette solution répond au problème de l'opposition entre la maîtrise du déroulement d'un jeu vidéo et son niveau d'interactivité. En d'autres termes, notre but est d'aider à réaliser des jeux vidéo dont l'évolution satisfait les intentions des auteurs tout en autorisant un déroulement influencé par les choix du joueur (exprimés aux travers de ses actions).

Pour cela, notre proposition permet à l'utilisateur de produire un modèle de scénario de jeu de bonne qualité qui est : (a) riche – le scénario fournit suffisamment d'options pertinentes aux personnages joueur/non-joueur de sorte que le joueur puisse déterminer le déroulement du jeu et sente toujours que le discours créé est intéressant, (b) valide – tous les discours possibles dans le scénario sont cohérents et répondent aux effets désirés par les auteurs, (c) opérationnel – la représentation du scénario est exécutable. Ce scénario est ensuite employé comme l'entrée d'un système de pilotage de narration interactive assurant le contrôle de la gestion du déroulement du jeu. Par conséquent, l'évolution des jeux, qui sont dirigés par un tel système de pilotage, garantit que l'exécution du jeu respecte les souhaits des auteurs, et en même temps, autorise la liberté des actions du joueur.

Pour répondre au problème exposé ci-dessus, nous appuyons notre solution sur un modèle mathématique calculable (la logique linéaire) qui offre des mécanismes de déductions rigoureux et automatiques.

Nous avons fait un tour d'horizon des approches existantes concernant le pilotage de narration interactive et la validation de scénario. Ceci nous permet d'identifier les principes nécessaires à notre solution, tels que les éléments d'architecture d'un système de pilotage ; la construction, la représentation, l'exécution de scénarios narratifs ; les propriétés de narration importantes ; l'évolution de référence des paramètres dramatiques ; la structuration de discours ; la stratégie pour la validation d'un scénario ; les informations qualitatives et statistiques nécessaires…

Nos contributions portent (1) sur la définition d'un ensemble de propriétés de narration spécifiant la qualité des scénarios de jeu ; (2) sur la proposition de modèles, algorithmes et outils pour écrire des modèles de scénario respectant ces propriétés.

Nous validons nos résultats par la réalisation de deux exemples. Le premier est un extrait d'un jeu éducatif expliquant comment appliquer notre outils en vue de produire un modèle de scénario de jeu valide, qui est exprimé par un séquent de logique linéaire dont la représentation est conforme à un métamodèle du calcul des séquents. Pour le second exemple, nous décrivons le processus de production complet d'un jeu vidéo réel basé sur l'histoire « Le Petit Chaperon rouge », mettant en ouvre un prototype de système de pilotage que nous avons proposé, ce qui permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur.

Mots-clés : jeu, jeu vidéo, histoire, discours, scénario, narration, interactivité, discours interactif, narration interactive, propriété de narration, modélisation et validation de scénario, système auteur, système de pilotage, logique linéaire, modèle formel.

Title: Towards the realization of interactive storytelling control systems: validation of a scenario based on a Linear Logic model.

Abstract: The objective of this PhD thesis is to provide a model, a method and a tool for producing interactive scenarios. Our solution solves the opposition between the controlled evolution of a video game and its interactivity level. In other words, our goal is to assist (authors) in producing video games whose unfolding satisfies authors' intentions while it is simultaneously influenced by player's choices (expressed via her/his actions).

To this purpose, our proposal is to allow users to produce a good quality game scenario model, which is: (a) rich – the scenario provides enough pertinent options for player/non-player characters so that the player can determine the evolution of the game and always feels that the created discourse is interesting, (b) valid – all the possible discourses in the scenario are consistent and meet authors' required effects, (c) operational – the representation of the scenario is executable. This scenario is then used as the input of an interactive storytelling control system to assist it in managing the unfolding of the game. As a consequence, the evolution of the video games, which are directed by such a control system, guarantees authors' requirements, while at the same time, it depends on player's actions.

In order to execute the foregoing proposal, we base our solution on a calculable mathematical model (linear logic) which provides rigorous and automatic deduction mechanisms.

We have made an overview of existing approaches concerning interactive storytelling control and scenario validation problems. This allows us to identify necessary principles for our solution, such as: architecture elements of a control system; construction, representation, implementation of narrative scenarios; important narrative properties; reference evolution of the drama parameters; discourse structuralization; scenario validation strategy; necessary statistical and qualitative information; *etc*.

Our contributions consist (1) in the definition of a set of narrative properties specifying the quality of game scenarios; (2) in the proposal of models, algorithms and tools in order to produce scenario models respecting these properties.

We validate our results by realizing two examples. The first is an extract of an educational game explaining how to apply our tools to produce a valid game scenario model, which is expressed by a linear logic sequent whose representation conforms to a metamodel of the sequent calculus. For the second example, we describe the complete production process of a real video game based on the story "Little Red Cap", implementing a prototype of control system we have proposed, which allows unfolding the game according to the produced valid scenario, so its evolution satisfies authors' intentions and simultaneously depends on player's actions.

Keywords: game, video game, story, discourse, scenario, storytelling, interactivity, interactive discourse, interactive storytelling, narrative property, scenario modeling and validation, authoring system, control system, linear logic, formal model.

TABLE DES MATIÈRES

LISTE DES ACRONYMES

ACE	**A**dvances in **C**omputer **E**ntertainment
ADI	**A**rchitecture de **D**rame **I**nteractif
AP	**A**uteur **P**rincipal
API	**A**pplication **P**rogramming **I**nterface
BPMN	**B**usiness **P**rocess **M**odeling **N**otation
EIAH	**E**nvironnement **I**nformatique pour l'**A**pprentissage **H**umain
EMF	**E**clipse **M**odeling **F**ramework
FearNot	**F**un with **e**mpathic **a**gents **r**eaching **N**ovel **o**utcomes in **t**eaching
GAO	**G**raphe **A**cyclique **O**rienté
GMF	**G**raphical **M**odeling **F**ramework
ICIDS	**I**nternational **C**onference on **I**nteractive **D**igital **S**torytelling
IDM	**I**ngénierie **D**irigée par les **M**odèles
IPG	**I**nteractive **P**lot **G**enerator
IRIS	**I**ntegrating **R**esearch in **I**nteractive **S**torytelling
JADE	**J**eu d'**A**pprentissage des **D**angers de l'**É**lectricité
L3I	**L**aboratoire **I**nformatique, **I**mage et **I**nteraction
MDE	**M**odel-**D**riven **E**ngineering
MOF	**M**eta**O**bject **F**acility
NWN2	**N**ever**W**inter **N**ights **2**
NWNX	**N**ever**W**inter **N**ights e**X**tender
OMG	**O**bject **M**anagement **G**roup
PCr	**P**etit **C**haperon **r**ouge
PDDL	**P**lanning **D**omain **D**efinition **L**anguage
PhD	**D**octor of **Ph**ilosophy
RTH	**R**éseau de **T**âches **H**iérarchisées
S2IM	**S**ciences et **I**ngénierie pour l'**I**nformation, **M**athématiques
STRIPS	**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver
XML	e**X**tensible **M**arkup **L**anguage

LISTE DES TABLEAUX ET DES FIGURES

CHAPITRE I - Introduction

Glassner dans [44] fait le constat qu'écouter (lire) une histoire et jouer à un jeu vidéo sont deux loisirs généralement très appréciés (*« Everybody loves a great story, and everybody loves a great game. »*). Les rapprocher semble naturel. La recherche d'une façon de combiner narration d'histoire et jeux a pris une nouvelle envergure dans les dernières années du 20ème siècle, au moment où les jeux vidéo ont intégré des environnements réalistes et où la volonté d'adosser une histoire au jeu s'est développée. Ces qualités ont accompagné les améliorations techniques des ordinateurs, des consoles et les nouvelles techniques d'interaction. La conséquence de cette évolution est un gain en termes de qualité médiatique des jeux vidéo. Parallèlement les concepteurs ont cherché à rendre leurs jeux plus profonds et plus attachants, travaillant sur la qualité des histoires. Ce potentiel narratif offre la capacité de répondre à des projets multiples tant dans les domaines de l'éducation que de la formation en milieu à risque et ouvre un large champ de réflexion sur les outils d'apprentissage.

À côté de leur aspect amusant, les jeux vidéo sont aussi considérés comme un moyen efficace contribuant au domaine de l'éducation et de la formation [37, 73, 84]. Dans [99], l'auteur a montré que, grâce à leur capacité d'interactivité, les jeux vidéo fournissaient un support d'apprentissage plus efficace que les médias classiques non-interactifs comme la vidéo ou le texte. En effet, les apprenants, passent d'un rôle passif (subissent un cours disponible), à un rôle actif (ils peuvent contrôler le déroulement de parcours en interagissant lors du déroulement du jeu). L'immersion des apprenants renforce leur intérêt rendant plus efficace le processus d'apprentissage (ce qui constitue un point clé pour les jeux éducatifs) [16].

Néanmoins, le déroulement d'un jeu (ou le déroulement de l'histoire intégrée dans le jeu), qui respecte les effets souhaités par ses auteurs, et le niveau d'interactivité du jeu sont généralement considérés comme des contraintes s'excluant mutuellement [54]. La façon de résoudre cette opposition, ou autrement dit, comment produire un jeu vidéo, dont l'évolution satisfait les intentions des concepteurs tout en offrant aux joueurs la possibilité d'infléchir le déroulement de l'histoire à travers la prise en compte de leurs actions, est toujours une des questions cruciales de la communauté de narration interactive. Une « bonne » réponse à ce défi est nécessaire, pour répondre aux besoins, notamment dans le domaine de l'éducation et de la formation. Les possibilités d'interactivité, rendent le processus d'apprentissage plus intéressant et plus efficace pour le joueur/l'apprenant. Dans ce cadre éducatif, la contrainte est d'améliorer l'apprentissage des connaissances/leçons transmises par les auteurs en utilisant des moyens d'interaction, et en même temps, garantir le respect du processus d'apprentissage voulu par les auteurs.

Notre objectif dans le cadre de cette thèse est de fournir une solution qui permet à un auteur d'exprimer un modèle de scénario de qualité, celui-ci est ensuite utilisé par un gestionnaire de scénario comme trame contraignant l'exécution à rester conforme au modèle. Ce dernier est en charge de contrôler le déroulement de l'exécution du jeu de façon à suivre un scénario préétablit tout en tenant compte des inflexions données par l'utilisateur.

Pour atteindre cet objectif, notre démarche consiste à employer des modèles formels calculables. Cette approche rend possible la validation de l'ensemble des choix effectués pour un contexte de jeu donné, et l'utilisation de ce modèle de jeu valide comme support pour le système de pilotage de narration.

Les travaux précédents menés au laboratoire L3I [29, 81] ont permis de définir une stratégie de modélisation de narration interactive basée sur un modèle formel. Cependant, ce modèle doit être complété en vue de prendre en compte la distinction entre les prises de décision réalisées par le système de pilotage et celles réalisées par le joueur, en utilisant les connecteurs additifs de la logique linéaire.

Nous devons, également, définir l'ensemble des propriétés caractérisant un scénario de qualité, puis en proposer une évaluation automatique. Ces réflexions nous servent de base pour déterminer un système auteur permettant à un auteur (qui n'est pas un technologue, mais un spécialiste du domaine d'application) de spécifier un scénario interactif.

Nous présentons, dans la section suivante, la problématique en exposant les verrous scientifiques à traiter, ainsi que l'approche proposée pour résoudre l'opposition entre le contrôle du déroulement d'un jeu et le niveau d'interactivité qu'il offre. Néanmoins, pour que les lecteurs comprennent mieux ce que nous décrivons dans le cadre de ce mémoire, nous donnons tout de suite une terminologie des différents concepts délimitant le cadre de cette thèse, principalement les notions de jeu/histoire, discours, scénario, narration, et interactivité.

I.1. Terminologie utilisée dans ce mémoire

Dans le cadre de ce mémoire, nous utilisons la terminologie suivante (dans laquelle quelques notions sont héritées des travaux précédents de notre laboratoire L3I [30]) :

- *Jeu/histoire* : Un jeu/une histoire (puisque un jeu est toujours intégré à une histoire dans notre approche, nous les considérons comme « identique ») est un ensemble fini d'entités (comme des personnages, des ressources…), d'événements/actions et de contraintes, qui se situe dans un cadre spatio-temporel limité, résout un ensemble de problèmes et décrit une évolution concernant un ensemble de personnages et/ou d'objets. Il consiste à partir d'une situation initiale, à résoudre un ensemble de problèmes donné afin d'atteindre une situation finale, qui correspond à une des terminaisons satisfaisant les objectifs des auteurs. Par exemple, l'histoire « Le Petit Chaperon rouge » nous raconte comment un loup tente de dévorer le petit chaperon rouge qui apporte une galette, et une bouteille de vin à sa grand-mère. Cette histoire se déroule dans la journée, à 3 lieux : la maison du petit chaperon rouge, la forêt, la maison de la grand-mère, et comporte 5 personnages : le loup, le petit chaperon rouge, la grand-mère, le chasseur et la mère.
- *Discours* : Un discours est un déroulement possible de l'histoire. Il s'agit d'une suite ordonnée d'événements/actions impliquant un ou plusieurs personnages dans un ou plusieurs lieux. Une même histoire peut ainsi donner lieu à plusieurs discours différents, dépendant de l'ordonnancement des événements/actions de l'histoire. Dans l'exemple du Petit Chaperon rouge, on peut obtenir plusieurs discours différents, tels

que le loup peut finalement réussir à manger la grand-mère et le petit chaperon rouge, ou le chasseur intervient pour les sauver.

- *Scénario* : Un scénario est un ensemble de discours possibles pour une histoire. Tout changement dans l'histoire produit un nouveau scénario. Dans notre approche, les personnes qui créent le scénario d'un jeu sont appelés ses *auteurs*. Par ailleurs, dans le cadre de cette thèse, comme un scénario est toujours représenté par un modèle, les deux termes « scénario » et « modèle de scénario » sont équivalents.
- *Ressource* : Une ressource est un élément quelconque de l'histoire qui peut être consommé et/ou produit.
- *Personnage* : Un personnage est une entité « active » dans l'histoire qui possède un comportement (dirigé soit par un système de pilotage soit par un joueur).
- *Narration* : Une narration constitue la façon dont on raconte une histoire, c'est-à-dire qu'elle touche à la façon dont le contenu et/ou la représentation de l'histoire sont donnés au spectateur. Elle peut ainsi se distinguer par l'emphase mise sur certains personnages de l'histoire, les événements/actions utilisés, le support de présentation (oral, écrit, vidéo...) et le ton employé (comédie, drame, tragédie, farce…). Par exemple, l'histoire « Le Petit Chaperon rouge » peut être narrée en utilisant de nombreux formats différents (conte oral, film, dessin animé…) avec des intrigues différentes (selon la version originale [3] ou pas où le petit chaperon rouge peut tromper/combattre le loup…), en utilisant des tons variés (comédie, tragédie…), ou en mettant l'accent sur certains personnages uniquement (c'est-à-dire qu'il est possible de raconter cette histoire en se focalisant uniquement sur le point de vue du loup, du petit chaperon rouge…).
- *Interactivité* : L'interactivité se définit par la capacité de communication et d'action réciproque de deux ou plusieurs intervenants (humains, logiciels ou matériels). Ainsi un jeu vidéo est interactif dans la mesure où le déroulement du jeu est en partie dépendant des actions du joueur, et où la stratégie employée par le joueur s'adapte en fonction des informations qui lui sont communiquées par le jeu.
- *Discours interactif* : Le discours interactif consiste de son coté à permettre à l'utilisateur d'agir sur la teneur du déroulement du discours en effectuant divers choix au cours de celui-ci. Ces choix peuvent s'exprimer de diverses façons en fonction du support. Par exemple, la fiction hypertexte permet au lecteur de naviguer d'un paragraphe à un autre en fonction de ses décisions. Sa transposée audio-visuelle, le film interactif, applique le même principe en permettant au spectateur de décider par le biais de sa télécommande de la prochaine séquence à diffuser, parmi un ensemble correspondant aux suites possibles de la séquence en cours. Dans le cadre des jeux vidéo, la proposition la plus courante est de permettre au joueur de diriger le protagoniste du jeu, et d'influencer la teneur du discours créé à travers ses actions et ses attitudes. Par exemple, si le joueur est le petit chaperon rouge alors il peut effectuer plusieurs choix, ainsi, il peut refuser à apporter la galette et du vin à la grand-mère, ou tromper le loup et donc arriver chez la grand-mère en toute sécurité.
- *Narration interactive* : À côté des moyens permettant à l'utilisateur d'agir sur la teneur du déroulement du discours comme la notion de discours interactif, la narration

interactive comprend aussi les méthodes de changement de la représentation du discours impliquant une interaction avec l'utilisateur (c'est-à-dire que celui-ci peut influer sur la représentation du discours lors du déroulement de ce même discours). Ainsi, en permettant à l'utilisateur d'agir sur la représentation du discours, ce dernier peut personnaliser l'expérience narrative et s'approprier le discours plus facilement, sans pour autant altérer le contenu du déroulement de l'histoire. Par conséquent, grâce à la narration interactive, l'utilisateur passe d'un rôle uniquement passif (il subit le discours), à un rôle actif (il en détermine la teneur du déroulement, ou tout du moins sa présentation).

Dans le cadre de cette thèse, notre objectif concerne le domaine du discours interactif. Plus concrètement, nous voulons fournir une solution permettant de réaliser les jeux vidéo, dont le contenu du déroulement est influencé par les choix (les actions) du joueur, tout en faisant qu'en même temps, l'évolution respecte les effets souhaités par les auteurs. Nous abordons, dans la section suivante, les problèmes qui se posent pour atteindre cet objectif en exposant les verrous scientifiques à traiter, ainsi que l'approche proposée en vue d'obtenir une solution calculable pour ce problème.

I.2. Problématique et approche proposée

Afin d'aider à la production de jeux vidéo, dont l'évolution satisfait aux intentions des auteurs et dont le déroulement est conjointement influencé par les choix (les actions) du joueur, notre proposition, qui fait suite aux travaux réalisés au L3I [29, 80, 81], comporte deux parties : (1) produire un scénario de jeu de bonne qualité qui est (a) riche – le scénario fournit suffisamment d'options pertinentes aux personnages joueur/non-joueur de sorte que le joueur puisse déterminer le déroulement du jeu et sente toujours que le discours créé est intéressant, (b) valide – tous les discours possibles dans le scénario sont cohérents et répondent aux effets désirés par les auteurs ; (2) construire un système de pilotage de narration interactive pour les jeux vidéo, permettant de les diriger selon les scénarios produits, qui sont l'entrée du système de pilotage.

En vue de produire un scénario de jeu de bonne qualité (notre objectif principal dans le cadre de cette thèse), il nous faut définir un ensemble de propriétés de narration caractérisant la qualité du scénario. Pour la vérification de la satisfaction du scénario par rapport à ces propriétés, notre démarche comporte deux étapes : (1) modéliser des aspects importants d'un jeu tels que ses états et ses sorties souhaitées, les états et les choix des personnages joueur/non-joueur, les événements/actions et leurs effets… ; (2) parcourir l'ensemble des discours possibles du scénario obtenu suite au processus de modélisation, et analyser leur satisfaction pour les propriétés de narration. Ainsi, on peut évaluer la qualité du scénario courant et on peut donc produire un scénario de bonne qualité. Par ailleurs, puisque le scénario produit est l'entrée du système de pilotage, sa représentation doit être opérationnelle, c'est-à-dire que l'on peut utiliser le mécanisme de déduction rigoureux et automatique qui lui est associé afin d'aider le système de pilotage à diriger le déroulement du jeu.

Les travaux précédents menés au laboratoire L3I [29, 81] ont montré que, la logique linéaire [41, 42] fournissait une solution prometteuse répondant complètement aux exigences ci-dessus. En effet, son utilisation satisfait aux exigences de modélisation d'une histoire et de

vérification des propriétés de narration au moyen de l'écriture de preuves. De plus cette logique peut être directement programmée. Cependant, la compréhension des caractéristiques de la logique linéaire est complexe et nécessite un apprentissage pour les auteurs/utilisateurs « normaux » (qui ne sont pas informaticiens, mathématiciens, logiciens, *etc*.). Par ailleurs, la vérification de la satisfaction d'un scénario pour les propriétés de narration est trop complexe si la taille du scénario modélisé est grande. Un ensemble d'outils, aidant les utilisateurs à mettre en œuvre notre solution pour la production d'un scénario de bonne qualité, même s'ils n'ont aucune connaissance en logique linéaire, est donc nécessaire.

Pour répondre au problème de la définition d'un scénario de qualité, en tenant compte des exigences énoncées précédemment, nous avons organisé nos travaux de recherche comme suit.

Dans un premier temps, nous avons dressé un bilan des travaux existants. En effet, ces dernières années ont vue de nombreux travaux sur ce thème. Nous faisons donc un tour d'horizon des principales approches existantes concernant le pilotage de narration interactive et la validation de scénario. Ces approches nous permettent d'identifier les principes nécessaires à notre solution, tels que les éléments d'architecture d'un système de pilotage ; la construction, la représentation et l'exécution de scénarios adaptatifs ; les propriétés de narration importantes ; l'évolution de référence des paramètres dramatiques ; la structuration de discours ; la stratégie utilisée pour la validation d'un scénario ; les informations qualitatives et statistiques nécessaires…

En nous appuyant sur ce constat nous devons :

- proposer un ensemble de propriétés de narration caractérisant les scénarios qui permet d'évaluer leur qualité à travers l'examen de tous les discours possibles ;
- proposer une méthode de modélisation permettant d'exprimer les aspects importants d'une histoire tels que ses états et ses sorties désirées, les états et les choix des personnages joueur/non-joueur, les événements/actions… en employant les éléments fournis par la logique linéaire ;
- proposer des algorithmes permettant d'estimer la qualité d'un scénario modélisé en logique linéaire ;
- proposer un moyen de parcourir tous les discours possibles d'un scénario et d'analyser leur validité par rapport à l'ensemble de propriétés de narration.

À partir de ces propositions et en étendant le processus de production d'une application interactive scénarisée, nous réalisons un ensemble d'outils qui constituent un système auteur. Ce dernier aide les utilisateurs à exécuter la solution proposée pour la production d'un scénario de bonne qualité, même s'ils n'ont aucune connaissance en modélisation formelle (ou en logique linéaire).

Nous finissons, pour valider notre approche, en proposant à titre d'exemple des prototypes d'applications interactives scénarisées mettant en œuvre ces outils.

Nous venons ainsi de voir quels sont les verrous scientifiques à traiter et nos propositions pour la production des jeux vidéo, dont l'évolution satisfait les intentions des auteurs et dont le déroulement est, simultanément, influencé par les choix (les actions) du joueur. Afin que

les lecteurs puissent mieux comprendre la structure d'organisation de la thèse, nous donnons ci-dessous son plan.

I.3. Plan de la thèse

Nous présentons, dans cette section, le contenu principal de chacun des chapitres suivants de ce mémoire :

- Le chapitre II – *État de l'art* constitue un tour d'horizon des principaux travaux existants, concernant le pilotage de la narration interactive et la validation d'un scénario, qui ont été proposés par la communauté et nous permettent de caractériser notre approche par la suite.
- Le chapitre III – *Validation d'un scénario* explique comment produire un scénario de bonne qualité pour un jeu. Au début, nous donnons un ensemble de propriétés de narration spécifiant la qualité des scénarios de jeu, celles-ci sont classées en deux catégories : les propriétés liées au déroulement des discours et les propriétés liées à l'effet dramatique créé par chacun des discours. Ensuite, nous décrivons la solution, utilisant la logique linéaire, mise en œuvre pour examiner la satisfaction de ces propriétés de narration.
- Le chapitre IV – *Méthodes et outils* présente la démarche et le système auteur que nous proposons, en vue de produire des scénarios de bonne qualité. Notre système auteur est composé d'un ensemble d'outils que nous avons développé, et qui implémente les algorithmes de validation définis au chapitre III. Ce système auteur permet à une personne, ne possédant pas de connaissances en modélisation formelle, de définir un modèle formel de scénario, d'évaluer sa qualité et d'améliorer son discours, afin de produire un modèle formel de scénario valide pour l'exécution par un système de pilotage des applications interactives scénarisées. Ce chapitre se termine par deux cas d'application montrant l'utilisation de notre système auteur en vue de construire une application avec narration interactive.
- Le chapitre V – *Application des méthodes et outils sur deux exemples* donne deux exemples concrets ayant pour objectifs d'illustrer ainsi que de valider les résultats obtenus dans le cadre de la thèse grâce à notre approche utilisant la logique linéaire. Le premier est un extrait d'un jeu éducatif expliquant comment appliquer notre système auteur afin de produire un scénario de jeu valide, qui est exprimé par un séquent de logique linéaire dont la représentation est conforme à un métamodèle du calcul des séquents. Pour le second exemple, nous décrivons le processus de production complet d'un jeu vidéo réel basé sur l'histoire du « Petit Chaperon rouge », mettant en œuvre le prototype de système de pilotage que nous avons proposé, ce qui permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur.
- Le chapitre VI – *Conclusions et perspectives* fait un bilan général de tous/toutes les travaux/contributions que nous avons réalisé(e)s dans le cadre de la thèse. Ensuite, nous analysons en détail les limites des résultats que nous avons obtenus jusqu'ici, et en même temps, donnons des perspectives de nos travaux dans l'avenir.

CHAPITRE II - État de l'Art

II.1. Introduction

Ce travail s'insert dans un projet plus global qui consiste à réaliser un système de pilotage pour des applications interactives scénarisées. [30] a montré que les jeux vidéo étaient la sous-catégorie des applications interactives scénarisées qui avait le plus développé d'énergie à résoudre le problème de la dualité narration interaction. Dans le cadre de cette thèse, nous portons notre attention plus particulièrement au problème de validation d'un scénario lors de la phase de conception du jeu. La production d'un modèle de scénario de bonne qualité est une étape cruciale dans le processus que nous proposons. Ce modèle est lié à l'architecture pour le pilotage d'applications interactives scénarisées que nous développons.

Pour mener notre étude bibliographique, nous avons étudié les approches de narration interactive proposées par différents groupes de recherche s'intéresant au jeu (au sens large – jeu, jeu sérieux [37]). Notre étude est menée selon deux points de vue : dans le cadre du pilotage (en cours d'exécution) et pour établir la qualité d'un scénario (en amont de l'exécution).

Tout d'abord, nous avons étudié comment ces équipes traitent de la problématique du pilotage afin de savoir sur quels concepts se base l'adaptation du discours. Pour cela, nous n'avons pas classés les travaux selon la façon de gérer le développement du discours côté pilote, comme il est habituel de le faire, en classant d'un côté les approches orientées scénario et les approches émergentes. Mais nous avons classé les approches en fonction de l'influence du joueur (ou de l'utilisateur) sur le déroulement de la partie : le joueur contrôle un personnage ; ou le joueur observe le déroulement et influence le scénario.

Puis nous avons étudié les travaux orientés sur la preuve de la qualité de scénario, que ces travaux soient basés sur une modélisation formelle ou non. Nous présentons les résultats de cette analyse suivant deux axes : les déductions faites par analyse des traces d'exécution, et les déductions faites par analyse de la structure du scénario.

II.2. Pilotage de narration interactive

Cette section mentionne les principales approches sur le pilotage de narration interactive, dont l'objectif est de résoudre le problème de l'adaptation du déroulement d'une histoire de sorte que les utilisateurs puissent influer sur celui-ci. Ces travaux sont classés selon la forme de l'interaction de l'utilisateur sur l'évolution de l'histoire :

- ***L'utilisateur contrôle un personnage joueur :*** L'utilisateur déroule l'histoire en dirigeant son avatar dans l'environnement virtuel, ce qui correspond à un personnage joueur. Ainsi, il peut influencer l'évolution de l'histoire au moyen d'actions sur son avatar. Notre approche dans le cadre de cette thèse appartient à ce groupe.
- ***L'utilisateur observe le déroulement de l'histoire et l'influence :*** Pour cette catégorie de travaux, l'utilisateur ne contrôle pas un personnage joueur, mais il observe le

déroulement de l'histoire dans l'environnement virtuel comme un spectateur, et il peut influencer ce processus en changeant l'environnement (cacher/donner une ressource…), en modifiant les paramètres des personnages virtuels, ou en leur adressant des conseils… Ainsi, l'utilisateur peut à volonté influer sur l'évolution de l'histoire à travers tous les moyens que le système met à sa disposition.

Par ailleurs, lors de la présentation de chaque approche, dans cette section, nous analysons aussi comment elle répond aux questions suivantes :

- ***Liberté d'interaction de l'utilisateur :*** Quel est le niveau de liberté dont dispose l'utilisateur pour agir sur le système (nous considérons trois modalités : faible, moyen, élevé) ?
- ***Influence de l'utilisateur sur le déroulement de l'histoire :*** Dans quelles proportions l'utilisateur peut-il influencer le déroulement de l'histoire (nous considérons trois modalités : faible, moyen, élevé) ?
- ***Contraintes prédéfinies par l'auteur :*** Est-ce que le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur ?
- ***Pertinence des discours créés :*** Les discours créés sont-ils pertinents (cohérence et intérêt) ?
- ***Modèle de l'utilisateur :*** Est-ce qu'il existe un modèle de l'utilisateur qui autorise l'enregistrement de ses traces d'activité telles que la situation actuelle, les choix précédents… ?
- ***Mémoire de l'histoire :*** Est-ce qu'il existe une mémoire de l'histoire qui contient l'enregistrement des différentes étapes ?
- ***Interaction de l'utilisateur en langage naturel :*** Est-ce que l'interaction de l'utilisateur s'effectue en langage naturel ?
- ***Possibilité de changer la présentation d'un même discours :*** Est-ce qu'un même discours est capable d'être narré/présenté de façons différentes (en changeant la musique, la voix et/ou le visage du narrateur…) ?
- ***Dissociation entre l'environnement virtuel et le pilotage :*** Est-ce que le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage ?

II.2.1. L'utilisateur contrôle un personnage joueur

Nous présentons ici les principaux travaux concernant le pilotage de narrations interactives dans lesquels, l'utilisateur déroule l'histoire en dirigeant son avatar (qui correspond à un personnage joueur) dans l'environnement virtuel.

II.2.1.1. Façade

Michael Mateas a décrit dans [67, 68, 69, 70] une scène spécifique basée sur l'intelligence artificielle – un drame interactif appelé *Façade* (voir Figure II.1), où le concept « drame interactif » signifie la construction de façon dramatique des environnements virtuels intéressants habités par des personnages contrôlés par l'ordinateur, au sein desquels le joueur déroule une histoire à partir d'une perspective à la première personne. Façade permet au joueur d'être libre de se déplacer partout dans un monde en 3D, de manipuler des objets, et

surtout, d'interagir avec d'autres personnages (à travers des dialogues en temps réel où le joueur entre des segments textuels et les autres personnages communiquent avec la parole). Toute cette activité est cohérente et avec un but précis, plus concrètement, elle oriente l'évolution de l'histoire selon un (des) arc(s) dramatique(s) prédéfini(s) (voir Figure II.2).

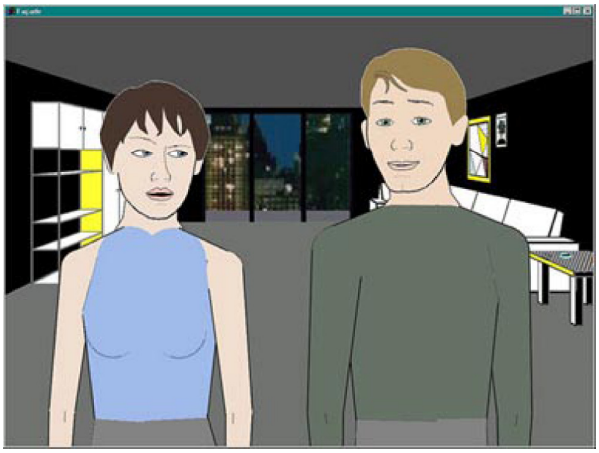


Figure II.1. Une scène dans Façade [67].

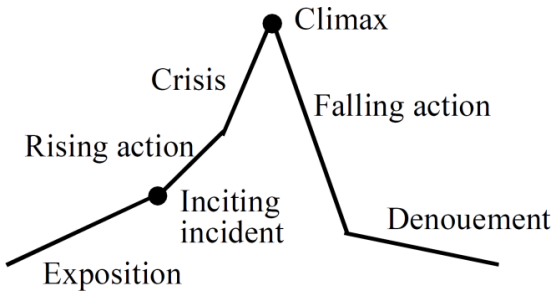


Figure II.2. Arc dramatique [67].

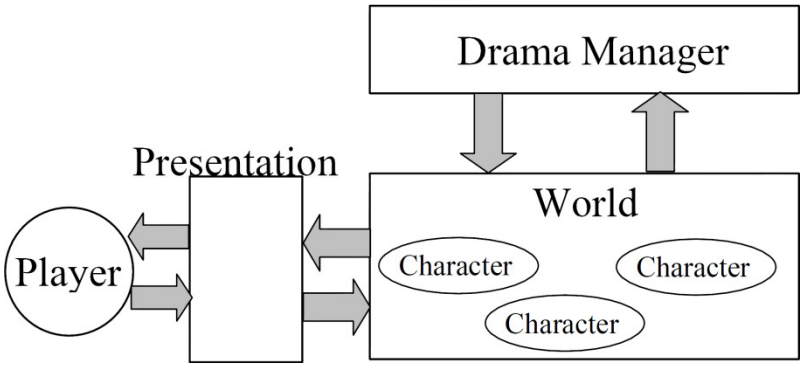


Figure II.3. Architecture de drame interactif du projet Oz [67].

L'approche « Façade » du drame interactif est directement bâtie à partir de celle du projet Oz [10] dont l'architecture est donnée dans la Figure II.3 où :

- Le monde simulé contient des agents crédibles, des personnages autonomes exprimant des personnalités, des émotions, des comportements sociaux, des motivations et des objectifs riches.

- L'utilisateur interagit avec ce monde à travers le module « Présentation ». Le module « Présentation » peut effectuer un filtrage dramatique actif, ce qui permet de changer l'angle de la caméra dans l'environnement graphique ou le style de langage utilisé dans l'environnement textuel.
- Le gestionnaire de drame peut voir tous les éléments du monde du drame interactif. Il guide l'expérience de l'utilisateur et réagit pertinemment aux entrées inappropriées et inintelligibles, afin que l'histoire se déroule « correctement » selon un (des) arc(s) dramatique(s) prédéfini(s). Cela peut impliquer des actions telles que : modifier le modèle de monde physique, inciter des personnages à poursuivre un plan d'action, ajouter ou supprimer des personnages, *etc*.

L'architecture commune ci-dessus a été concrétisée et améliorée par Mateas lors de la construction de Façade, dont les principaux composants sont (voir Figure II.4) : le monde de l'histoire, le gestionnaire de drame, la mémoire d'histoire, et le système de traitement de la langue naturelle.

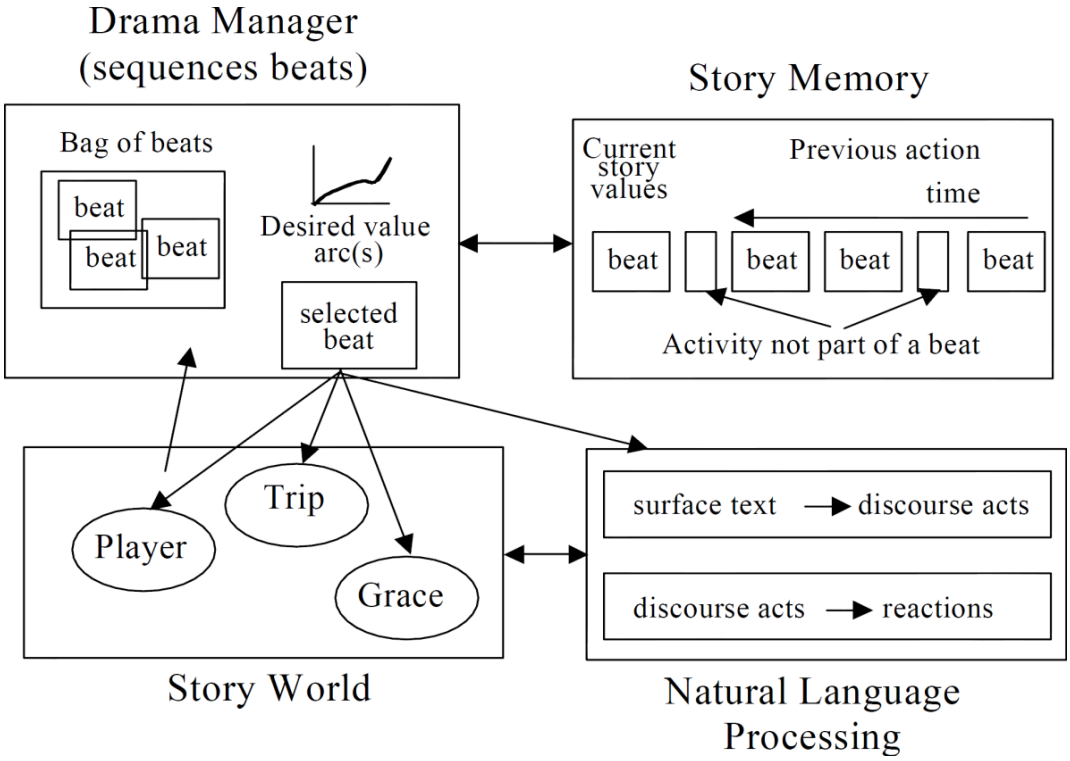


Figure II.4. Architecture de drame interactif de Façade [67].

- Monde de l'histoire : C'est un monde graphique – construit à travers un moteur de jeu – dans lequel le joueur et les agents crédibles (Trip et Grace) déroulent l'histoire. Les actions du joueur déterminent les dispositions du mariage entre Trip et Grace ainsi que les liens d'amitié des trois personnes.
- Gestionnaire de drame : Il est responsable de la sélection des *beats* – unité d'intégration de personnages/intrigues – de manière à faire avancer correctement l'histoire (dans la théorie dramatique, la plus petite unité de changement de valeur d'une histoire est appelée un *beat* et donc, aucune activité inférieure au *beat* n'est

associée au changement de valeur). Le gestionnaire de drame dispose d'un ensemble de *beats* possibles et une spécification d'arc(s) de valeurs souhaitées (qui décrit comment les valeurs de l'histoire devraient changer au cours du temps – voir Figure II.5), afin de choisir des *beats* adéquats. Le *beat* sélectionné passe ses comportements spécifiques aux personnages (donc détermine leur actions) et au système de traitement de la langue naturelle (donc l'aide à analyser les textes entrés par le joueur). Quand le gestionnaire de drame est informé qu'un *beat* est terminé (soit interrompu soit réussi), il enregistre ce *beat* dans la mémoire d'histoire, et met à jour les valeurs d'histoire impliquées par les effets du *beat*. Par ailleurs, il interprète aussi et enregistre toute l'activité du joueur dans cette mémoire (afin de prévoir ses actions plus tard). Par conséquent, ces informations permettent au gestionnaire de drame d'estimer la situation courante, les choix précédents ainsi que les réactions possibles du joueur pendant sélectionner un *beat* dans l'avenir.

- Système de traitement de la langue naturelle : Il a pour but de traiter les textes entrés par le joueur de façon convenable avec le *beat* sélectionné. Ce processus comprend 2 phases : la première cherche la correspondance entre les textes entrés et des actes de discours dans le *beat*, tandis que la deuxième cherche des réactions des personnages correspondant à ces actes de discours.

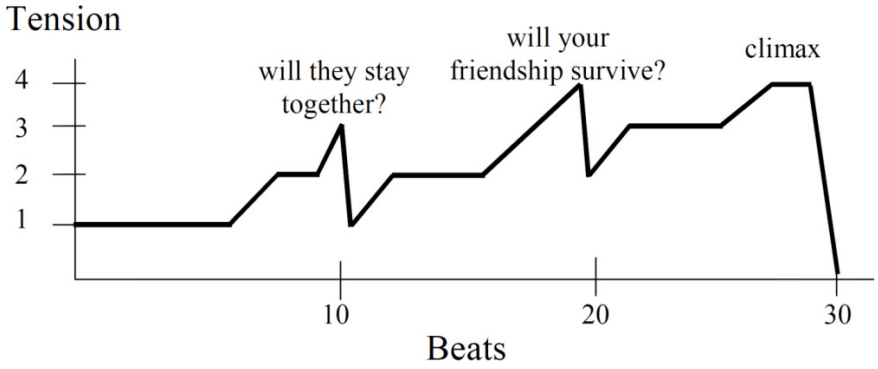


Figure II.5. Arc dramatique de Façade [67].

Ainsi, on peut trouver que, Façade répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut (se déplacer partout dans l'environnement virtuel, manipuler des objets, interagir avec d'autres personnages).
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car le dernier évolue toujours selon les arcs dramatiques prédéfinis par l'auteur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont les arcs dramatiques prédéfinis).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur.
- Il existe un modèle de l'utilisateur (dans la mémoire de l'histoire) enregistrant toute son activité (qui est utilisé afin de prévoir ses actions plus tard).

- Il existe une mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur s'effectue en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.2. IDtension

Szilas dans [92, 93, 94, 95] a présenté un système d'ordinateur baptisé *IDtension* – un moteur narratif pour le drame interactif, qui permet de combiner la narrativité et l'interactivité en s'appuyant sur des propriétés de narration (et non sur les évolutions prédéterminées). L'architecture générale du système est donnée dans la Figure II.6 et divisée en 5 modules :

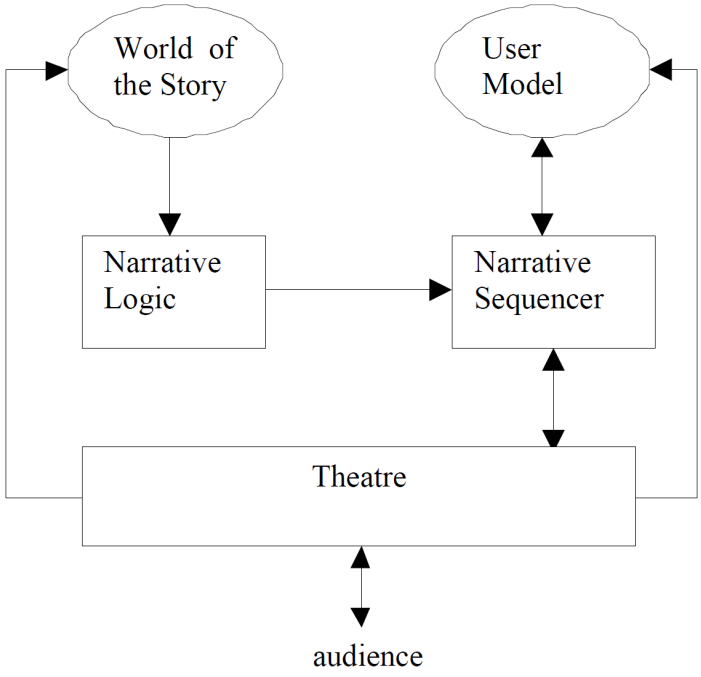


Figure II.6. Architecture générale du système [92].

- Monde de l'histoire : Il comprend des entités fondamentales dans l'histoire (personnages, objectifs, tâches, sous-tâches, obstacles), des états des personnages (exprimés par des prédicats), et des faits concernant la situation matérielle du monde de l'histoire (par exemple, le fait qu'une porte est fermée). La description du monde de l'histoire au moment initial est la mission principale des concepteurs-auteurs.
- Logique de narration : Elle calcule toutes les actions possibles des personnages en appuyant sur les paramètres des actions et sur les données du monde de l'histoire.
- Séquenceur de narration : Il filtre les actions calculées par la logique de narration, afin de les ranger de l'action la plus pertinente à celle la moins pertinente en examinant les propriétés de narration.
- Modèle de l'utilisateur : Il contient l'état de l'utilisateur au moment courant, et donc fournit, au séquenceur de narration, une estimation de l'impact de chacune des actions possibles pour l'utilisateur.

- Théâtre : Il constitue l'environnement virtuel dans lequel l'utilisateur (auditeur) voit ce qui se passe dans l'histoire et interagit avec les autres personnages.

Afin d'estimer si une action est satisfaite lors du déroulement de l'histoire, *IDtension* vérifie l'ensemble de propriétés de narration ci-dessous (la mesure de la satisfaction pour une action est une combinaison de la satisfaction de toutes ces propriétés) :

- Cohérence éthique : L'action est cohérente avec les actions précédentes du même personnage ?
- Cohérence de motivation : L'action est compatible avec les objectifs du personnage ?
- Pertinence : L'action est pertinente par rapport aux actions qui viennent d'être exécutées ?
- Caractérisation : L'action permet à l'utilisateur de comprendre les caractéristiques des personnages ?
- Obstacle : L'action soit mène directement à un obstacle soit pousse l'utilisateur vers une tâche menant à un obstacle ?

Au cours du déroulement de l'histoire, à chaque étape, le système ordonne toutes les actions possibles selon leur degré de satisfaction pour les propriétés de narration, et puis en exécute, au hasard, une parmi les actions les plus satisfaites (dont le critère de satisfaction dépasse un certain seuil). Ensuite, l'utilisateur sélectionne à volonté une action dans une liste d'actions possibles proposée, et ainsi détermine l'évolution de la narration. Par conséquent, l'histoire est déroulée de façon imprévisible, grâce à l'interaction (qui respecte toujours les effets narratifs nécessaires) entre l'utilisateur et les autres personnages dans l'environnement virtuel.

Ainsi, on peut trouver que, IDtension répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est faible car il peut seulement choisir une action dans une liste d'actions possibles proposée par le système.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé car le déroulement de l'histoire ne s'appuie pas sur les évolutions prédéterminées, mais dépend des choix de l'utilisateur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont les propriétés de narration prédéfinies).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur.
- Il existe un modèle de l'utilisateur, qui contient l'état de l'utilisateur au moment courant, et qui fournit, au séquenceur de narration, une estimation de l'impact de chacune des actions possibles pour l'utilisateur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur ne s'effectue pas en langage naturel.
- Un même discours n'est pas capable d'être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.3. Mimesis

Le *Liquid Narrative Group* du *North Carolina State University* a développé un système baptisé Mimesis [83, 100, 101, 102], qui est utilisé afin de construire des jeux interactifs, en intégrant des composants intelligents ayant des objectifs spéciaux avec des moteurs de jeu conventionnels (les auteurs appellent les moteurs de jeu *MWorld* – voir Figure II.7). À l'exécution, pendant le déroulement d'un jeu, Mimesis agit comme un générateur de comportements. Il est responsable de deux tâches : (1) générer des plans – des séquences d'actions cohérentes qui permettent d'atteindre un ensemble spécifique de buts – et (2) maintenir la cohérence de ces plans quelques soient les activités imprévues des utilisateurs.

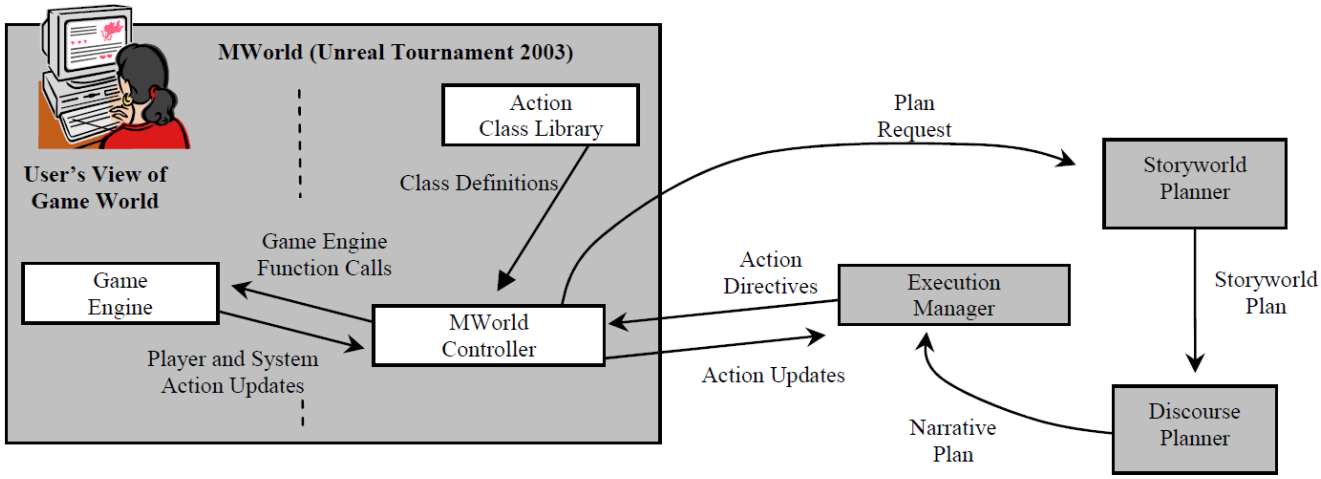


Figure II.7. Architecture du système Mimesis [102] où le moteur de jeu (*Game Engine*) *Unreal Tournament 2003* [114] est choisi comme *MWorld*.

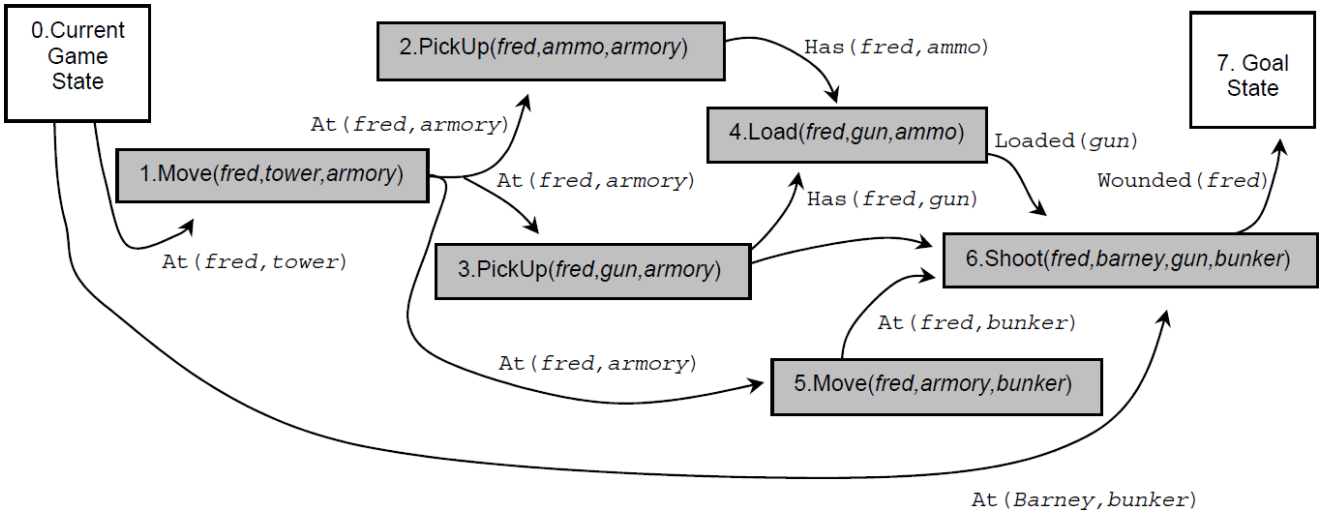
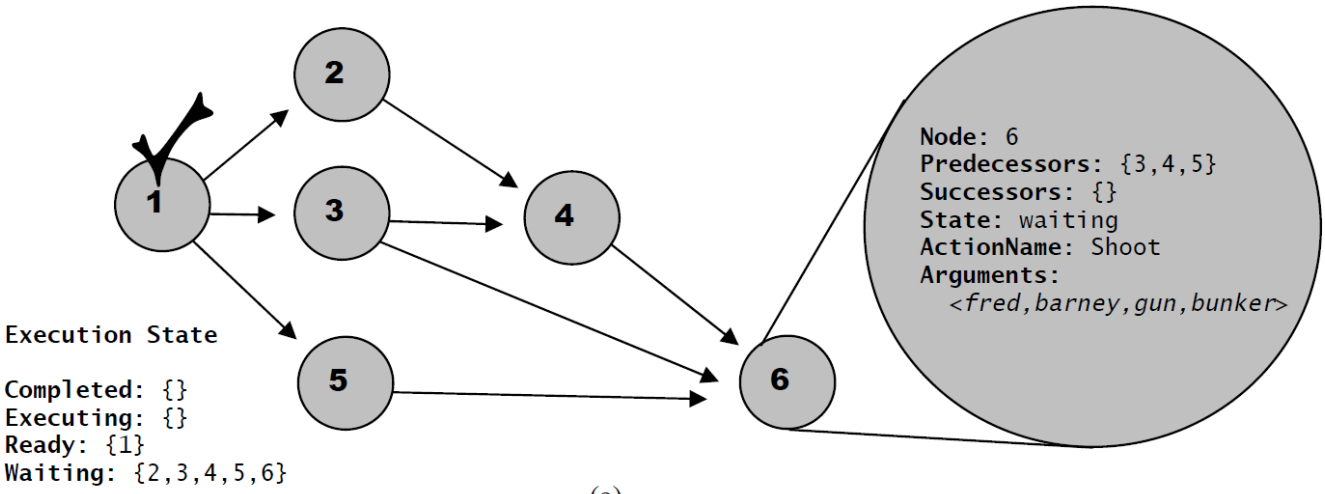


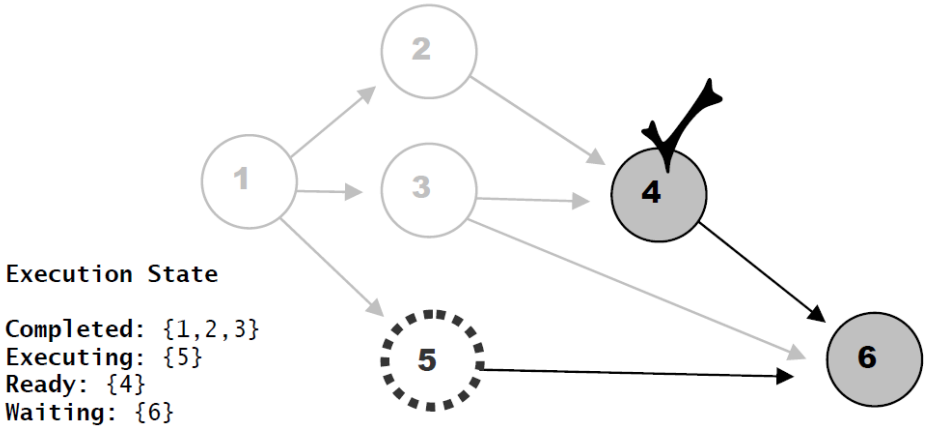
Figure II.8. Un exemple d'un *Storyworld Plan* représentant une séquence d'actions avec leurs liens de causalité [102].

Au moment initial du jeu, l'activité au sein de Mimesis commence par un message *Plan Request*, effectué par *MWorld Controller* et envoyé au composant *Storyworld Planner*. Le contenu du *Plan Request* identifie un problème spécifique de l'histoire qui a

besoin d'être traité (par exemple, quels buts doivent être atteints, quelle librairie est disponible pour construire la séquence d'actions…). Ensuite, le système (1) crée un plan qui contrôle les actions dans le jeu afin d'atteindre les buts de l'histoire, (2) exécute les actions prévues du plan, et (3) surveille leur exécution en vue de garantir les objectifs désirés.



(a)



(b)

Figure II.9. Deux exemples de l'exécution GAO utilisée par *Execution Manager* correspondant au *Storyworld Plan* dans la Figure II.8 où (a) : l'exécution du plan n'est pas encore commencée, et (b) : l'exécution GAO à l'étape 5 [102].

Pour cela, les composants de Mimesis réalisent le processus suivant : Quand *Storyworld Planner* reçoit un *Plan Request*, il produit un *Storyworld Plan* (voir Figure II.8). Ce plan est représenté par une structure de données définissant une séquence d'actions pour des personnages impliqués dans le jeu. *Storyworld Planner* passe ce plan à *Discourse Planner* – un composant responsable de construire un *Narrative Plan*, décrivant toute l'activité du système et de l'utilisateur qui sera exécutée. Le *Narrative Plan* est envoyé à *Execution Manager* qui construit un graphe acyclique orienté (GAO – voir Figure II.9), représentant les dépendances temporelles entre l'exécution de chacune des actions au sein du *Narrative Plan*. Ensuite *Execution Manager* agit comme un ordonnanceur de processus en répétant un cycle : (1) sélectionner, dans le

GAO, les actions qui sont actuellement prêtes à être exécutées, (2) envoyer des consignes à *MWorld Controller* pour exécuter ces actions, et (3) recevoir les mises à jour du jeu indiquant si les actions sont exécutées avec succès. Dans le cas où cette exécution échoue à cause des événements générés par les actions de l'utilisateur (qui mêlent la structure du plan d'histoire), le système (1) soit détourne « secrètement » ces événements pour les accorder avec le plan courant (rend l'effet des événements échoué), (2) soit adapte le plan à la nouvelle activité de l'utilisateur si la « dépense » de ce changement est faible. Ces interventions du système doivent être aussi compatibles avec sa prévision sur le plan et les buts de l'utilisateur. Cette prévision est basée sur la situation actuelle et les actions précédentes que l'utilisateur a faites. Par ailleurs, le système Mimesis fournit aussi une API (*Application Programming Interface*) aux développeurs de jeux qui peut être simplement intégrée dans un moteur de jeu typique, donc il est possible de construire facilement des jeux vidéo interactifs intelligents dont le déroulement et le scénario sont cohérents.

Ainsi, on peut trouver que, Mimesis répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut dans l'environnement virtuel au moyen d'actions sur son avatar.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car le dernier évolue toujours selon un plan généré par le système.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont les plans générés par le système).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur un plan généré par le système.
- Il existe un modèle de l'utilisateur, qui contient sa situation courante et les actions précédentes qu'il a faites.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur ne s'effectue pas en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.4. Interactive Drama Architecture

Similaire au projet Façade, les travaux réalisés par l'équipe de l'Université du Michigan concernent le concept de « drame interactif ». En effet, elle a proposé une architecture de drame interactif (ADI), qui avait pour but de trouver un équilibre entre l'interactivité du joueur et l'intention des auteurs dans le cadre des jeux orientés histoires [63, 64, 65, 66]. Comme montrée dans la Figure II.10, l'ADI est constituée des composants suivants : le joueur humain, l'auteur humain, le directeur et le monde virtuel dans lequel l'histoire se déroule. Le directeur contrôle des agents intelligents qui sont semi-autonomes parce qu'ils ont aussi la capacité d'exécuter des comportements spécifiques. Les décisions du directeur à un moment quelconque du déroulement de l'histoire sont basées sur les actions du joueur, les intrigues prédéfinies par l'auteur, l'état courant du monde, et la prévision du directeur sur le comportement du joueur dans l'avenir.

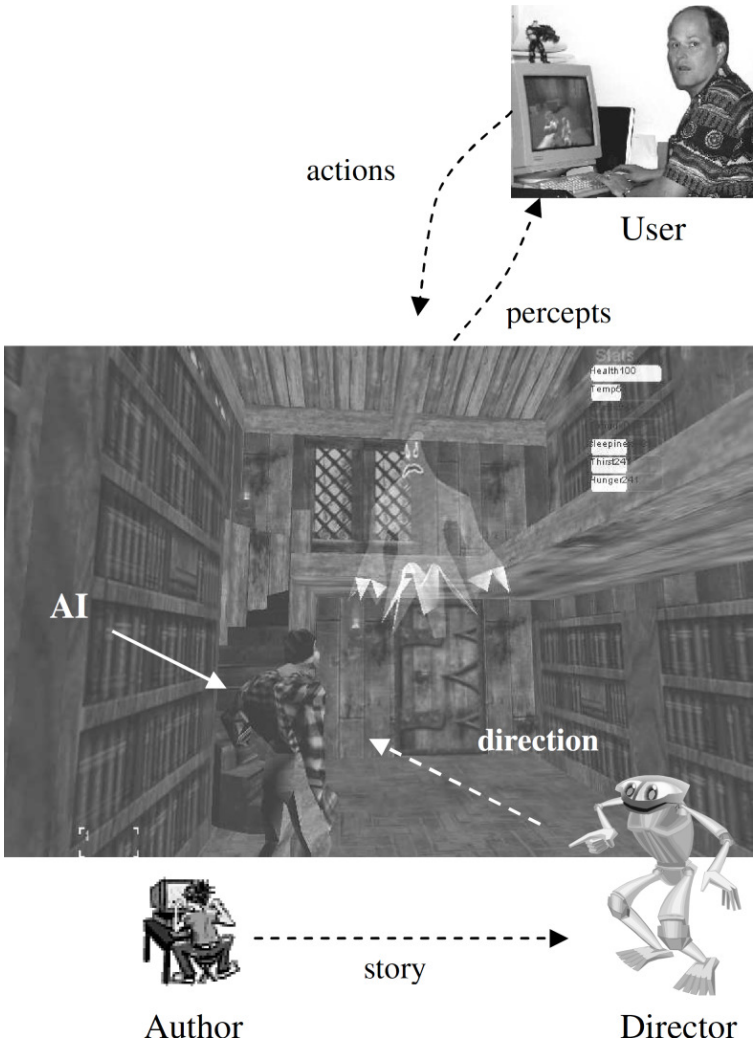


Figure II.10. Architecture de drame interactif – ADI [64].

Pour permettre à l'auteur humain de réaliser un modèle d'histoire employé dans l'ADI, [64] introduit une représentation d'histoire utilisant l'ordre partiel des points d'intrigue abstraits. Cette représentation est l'entrée du composant « directeur », qui « sait toutes les choses » dans le jeu, lors du déroulement de l'histoire. Autrement dit, le directeur travaille avec un scénario pré-écrit afin de guider le joueur. Le directeur prévoit le comportement du joueur (grâce à ses informations telles que les objectifs, la situation courante, les actions précédentes, ainsi qu'aux connaissances concernant le système que le directeur a collectées) et donne les consignes aux agents intelligents, en vue d'effectuer, si nécessaire, des éléments d'intrigue pertinents, ainsi la progression cohérente du jeu est garantie.

Un exemple de la représentation d'histoire abordée ci-dessus est donné dans la Figure II.11. Chacun des événements d'histoire est exprimé par un nœud (point d'intrigue) dans le graphe. Chaque point d'intrigue comprend un ensemble de préconditions, un ensemble d'actions qui sont exécutées si les préconditions sont satisfaites, et la contrainte temporelle liée à ce point qui décrit l'intervalle de temps pendant laquelle les préconditions doivent devenir vraies.

Ainsi, on peut trouver que, les travaux réalisés par l'équipe de l'Université du Michigan répondent aux questions citées au début de la Section II.2 comme suit :

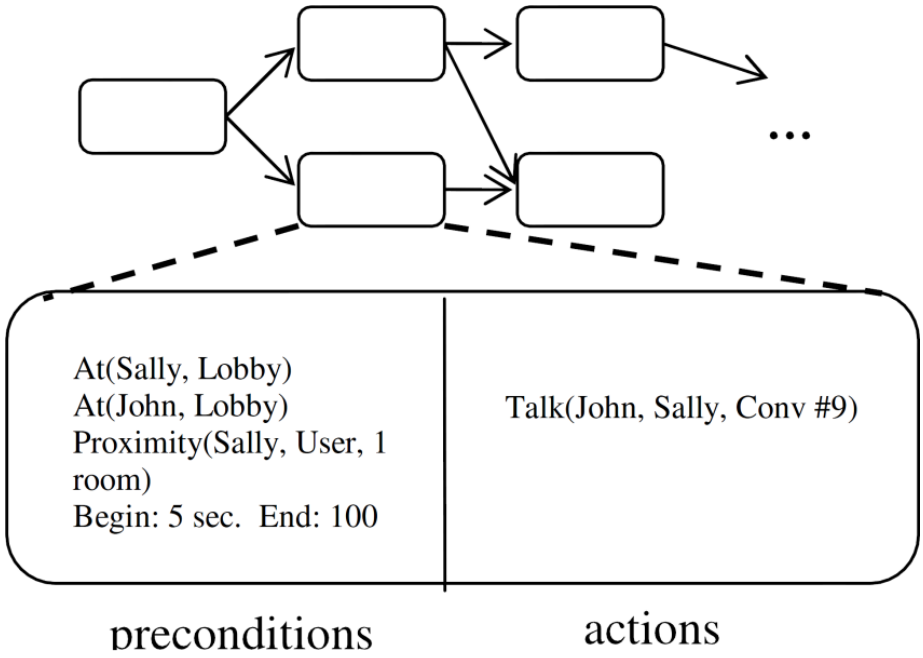


Figure II.11. Un exemple sur l'ordre partiel des points d'intrigue abstraits [64].

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut dans l'environnement virtuel.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car le dernier évolue toujours selon les intrigues prédéfinies par l'auteur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont les intrigues prédéfinies).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur les intrigues prédéfinies par l'auteur.
- Il existe un modèle de l'utilisateur, qui contient ses objectifs, sa situation courante et les actions précédentes qu'il a faites.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur ne s'effectue pas en langage naturel.
- Un même discours n'est pas capable d'être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.5. Exécution adaptative de trame narrative

Dans le cadre des travaux menés au laboratoire L3I, Prigent *et al.* ont proposé une méthode d'exécution adaptative d'un jeu de stratégie [21, 80, 81], permettant : (1) de générer « dynamiquement » les discours possibles du jeu au cours de son déroulement en examinant une fenêtre limitée d'événements (4 événements pour l'exemple utilisé par les auteurs), (2) d'analyser les chemins erronés parmi eux, et puis (3) de supprimer les discours non satisfaisants. Par conséquent, le déroulement du jeu quelque soit le choix du joueur atteint toujours une terminaison valide.

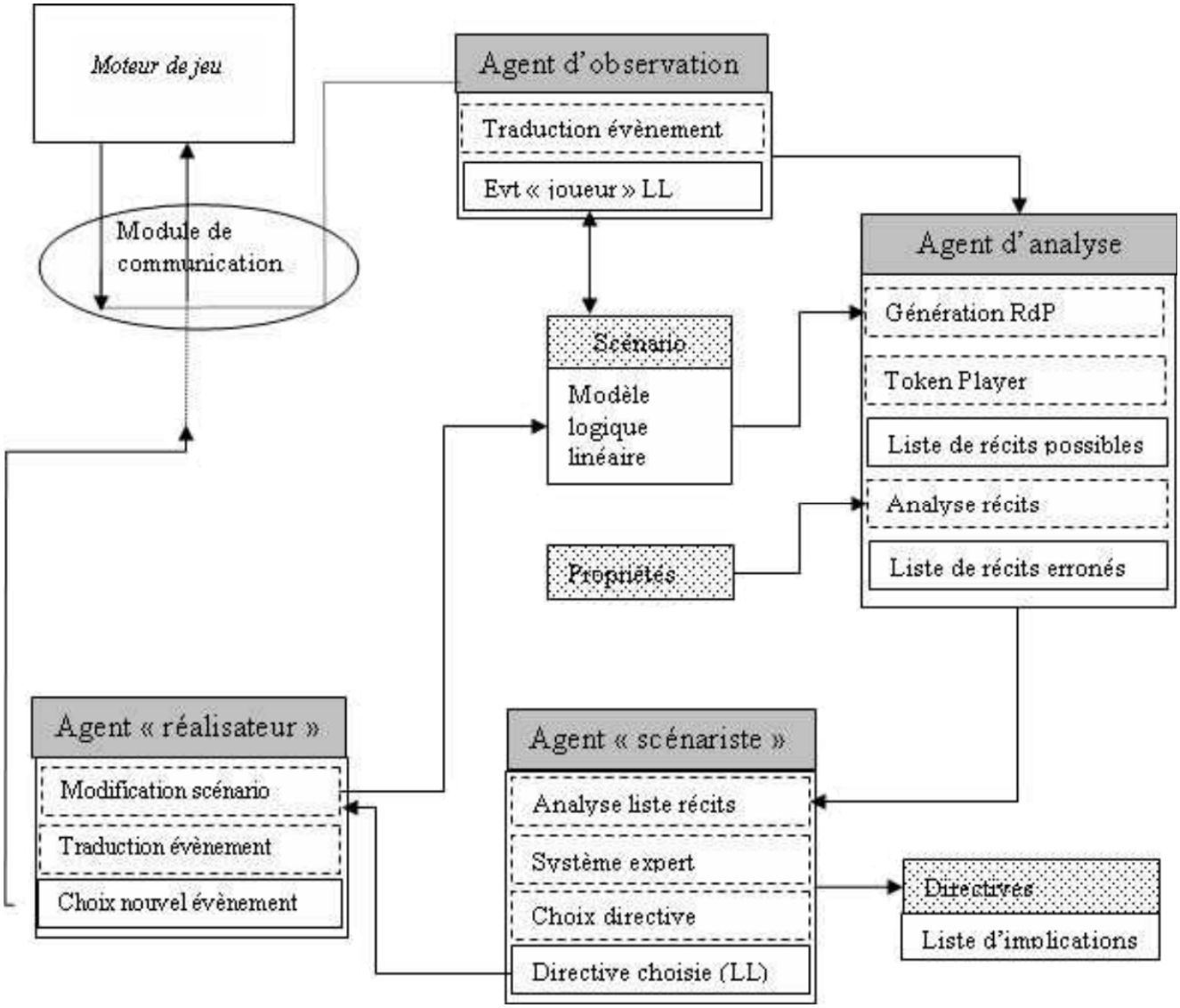


Figure II.12. Méthode d'analyse interactive du jeu [81].

Pour cela, un système adaptatif, composé d'un agent d'observation, d'un agent d'analyse, d'un agent « scénariste » et d'un agent « réalisateur », a été mise en œuvre. L'agent d'analyse effectue des analyses locales lors du déroulement du jeu, tandis que les agents scénariste et réalisateur décident et mettent en place les « modifications hors de portée du joueur » sous la forme de directives si nécessaire. La méthode d'analyse interactive du jeu est donnée dans la Figure II.12 et fonctionne de la manière suivante :

1. à la première étape, l'agent d'observation observe les actions du joueur et les traduit en événements décrits à l'aide du formalisme de la logique linéaire ;
2. l'étape suivante consiste à construire l'ensemble des discours constitués de quatre événements à partir de la situation courante du jeu. L'analyse de ces discours produit un ensemble de discours aboutissant d'une part à des blocages et d'autre part à une terminaison prématurée du jeu ;
3. l'étape suivante consiste à déterminer les ressources responsables de ces discours erronés et puis à les supprimer. Lorsque ces ressources ont été supprimées, on obtient

le nouveau modèle en logique linéaire, le jeu peut alors commencer et un réseau de Petri [72] correspondant à ce modèle de linéaire logique est construit ;

4. le simulateur du jeu déroule les transitions franchissables de ce réseau de Petri et propose les événements au joueur. Selon le choix effectué par le joueur, les événements déjà utilisés par le joueur sont retirés du modèle en logique linéaire (il est impossible qu'un joueur consomme deux fois une ressource). La première étape est effectuée à nouveau et la méthode est déroulée en considérant la situation atteinte comme situation initiale de l'évolution du jeu.

Néanmoins, la solution proposée ci-dessus comporte des inconvénients : (1) la « transformation réciproque » d'un séquent et son réseau de Petri correspondant à chaque étape du déroulement du jeu provoque un « gaspillage » de temps et de ressources machine pour leur traitement ; (2) cette solution élimine toujours des discours erronés, lors de l'exécution du jeu, juste après les avoir détecté, par conséquent, elle réduit la variété du scénario ; (3) elle ne garantit pas que le déroulement de l'histoire est toujours en succès. Par exemple, considérons le séquent A, EA1, EA2, EA3, EA4, EA5 ⊢ D où EA1 : A ⊸ B ; EA2 : A ⊸ C ; EA3 : A ⊸ D ; EA4 : B ⊸ E ; EA5 : E ⊸ F. Nous prenons, dans cet exemple, une fenêtre de taille 3 (c'est-à-dire une fenêtre de 3 événements). Après avoir vérifié l'ensemble des discours possibles composés de 3 événements, le système adaptatif détecte que, si le joueur choisit soit EA1 soit EA3, il n'y a aucun problème, mais s'il choisit EA2, le discours est erroné. C'est pourquoi, dans cette solution, EA2 est éliminé (malheureusement cela réduit la variété du scénario) et le séquent devient A, EA1, EA3, EA4, EA5 ⊢ D. Ensuite, dans le cas où le joueur exécute EA1, le séquent devient B, EA3, EA4, EA5 ⊢ D. Le système adaptatif continue à vérifier l'ensemble des discours possibles composés de 3 événements, il constate que maintenant il n'y a qu'un discours erroné. Néanmoins, à ce moment il est trop tard pour changer, et il y a donc un blocage dans cette situation de l'histoire. Par voie de conséquence, l'analyse dynamique, employant une fenêtre limitée d'événements, comme mentionnée plus haut, ne garantit pas que le déroulement de l'histoire soit toujours réussi.

Ainsi, on peut trouver que, les travaux, réalisés par Prigent *et al.* dans le cadre de notre laboratoire – L3I, répondent aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est faible car il peut seulement choisir une action dans une liste d'actions possibles proposée par le système.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé car le déroulement de l'histoire ne s'appuie pas sur les évolutions prédéterminées, mais dépend des choix de l'utilisateur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont le fait que les discours doivent toujours mener à une terminaison valide, par exemple, il n'y a ni blocages ni terminaisons prématurées du jeu…).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur.
- Il n'existe pas de modèle de l'utilisateur qui enregistre ses traces d'activité telles que la situation actuelle, les choix précédents…

- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur ne s'effectue pas en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.6. Thespian

L'*Institute for Creative Technologies* de l'*University of Southern California* a développé le projet Thespian, qui est un environnement fondé sur la technologie multi-agent, permettant d'éditer et de simuler des narrations interactives, ainsi que de fournir un contrôle directorial proactif employant des modèles explicites de l'utilisateur [85, 86, 87, 88]. Il a été utilisée en vue de développer TactLang (*Tactical Language Training System* [87, 88]), où l'utilisateur se déplace dans le monde virtuel construit en 3D par le moteur de jeu *Unreal Tournament*, et interagit, en langage naturel, avec les personnages virtuels. L'architecture de Thespian est composée de deux couches (voir Figure II.13) :

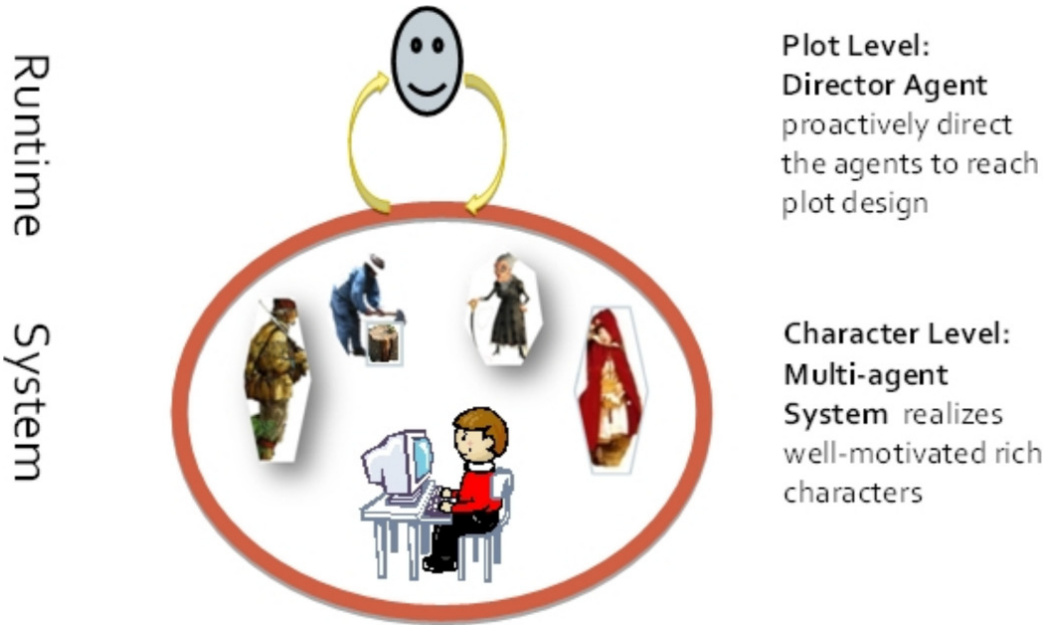


Figure II.13. Thespian : Système à deux couches pour la narration interactive [107].

- ***Character Level :*** C'est un système multi-agent comprenant des agents autonomes orientés objectifs et fonctionnant selon la théorie de la décision. Ces agents sont utilisés pour représenter les personnages de l'histoire. Un aspect clé de cette couche est la richesse de la conception des agents qui fournit, aux agents, le moyen de représenter des motivations, des émotions, des normes sociales et notamment une théorie de l'esprit. La théorie de l'esprit – voir Figure II.14 – utilisée par ce modèle, permet à un agent de raisonner sur la foi, les objectifs et les politiques des autres lorsqu'il prend une décision, assurant la correction de ses actions. Les agents interagissent de manière autonome les uns avec les autres et avec le joueur (en sélectionnant les actions à exécuter de sorte qu'ils reçoivent la plus haute récompense), et de ce fait, l'histoire progresse. En particulier, dans Thespian,

l'utilisateur est modélisé par un agent fondé sur le personnage qu'il joue, cela permet au système d'estimer l'état du joueur, de tenir compte des expériences précédentes de ce joueur, et de pronostiquer sa réaction.

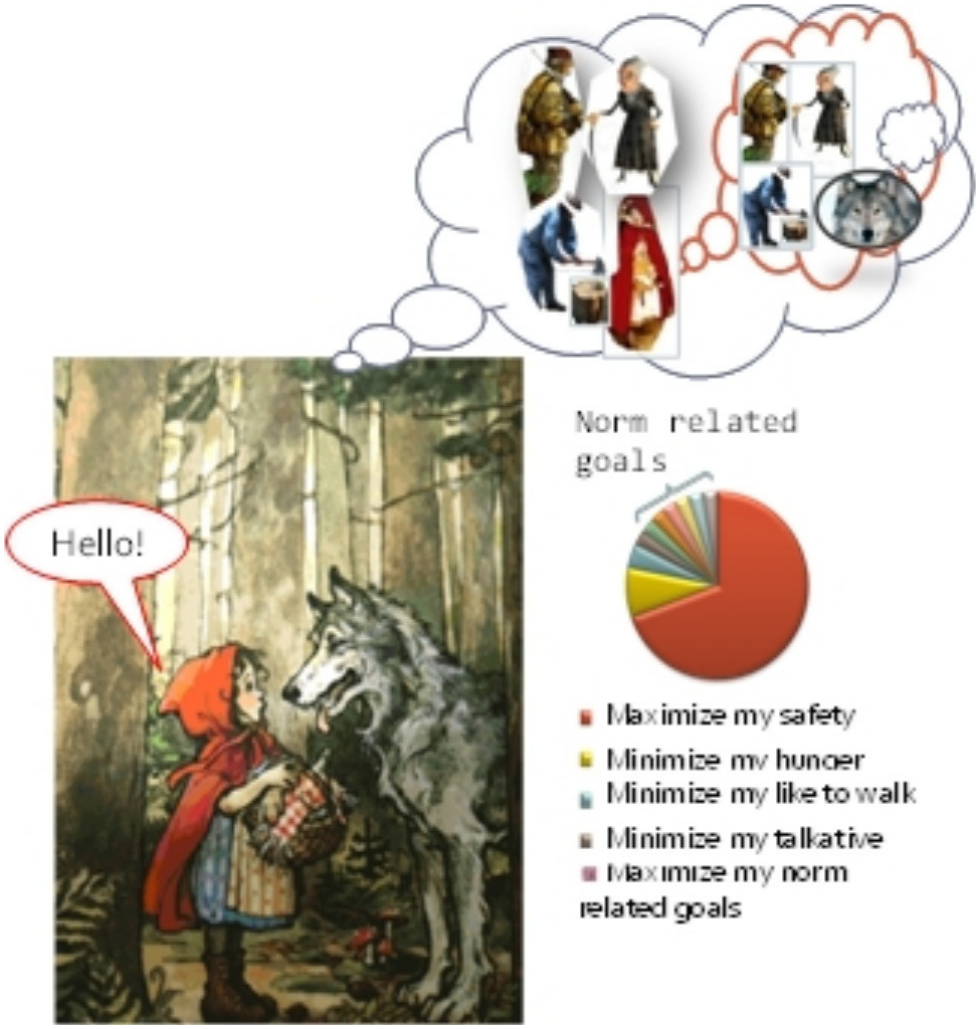


Figure II.14. Théorie de l'esprit [107].

- ***Plot Level :*** Dans cette couche, un agent directorial surveille l'évolution de l'histoire et redirige de façon proactive les personnages, quand il prévoit que le comportement futur du joueur mettra en danger la conception de l'intrigue imaginée par l'auteur (qui est exprimée comme les contraintes temporelles ou d'ordre partiel des événements clés dans l'histoire – voir Figure II.15). L'agent directorial a l'accès aux modèles des agents et du joueur, pour évaluer si les objectifs de l'histoire sont atteints. Il peut ainsi rediriger l'action des personnages si nécessaire. Dans le cas où l'agent directorial détecte une violation potentielle du déroulement prévu de l'histoire, en se fondant sur le modèle de l'utilisateur, il explore des méthodes réglant le comportement des personnages impliqués, afin de prévenir la violation et de garantir les objectifs désirés de l'auteur.

Ainsi, on peut trouver que, Thespian répond aux questions citées au début de la Section II.2 comme suit :

orders =	[*event*1,*event*2] *event*2 should happen after *event*1
earlierThan =	[*event*,*step*] *event* should happen before *step* steps of interaction
laterThan =	[*event*,*step*] *event* should happen after *step* steps of interaction
earlierThan2 =	[*event*1,*event*2,*step*] *event*2 should happen within *step* steps after *event*1 happened
laterThan2 =	[*event*1,*event*2,*step*] *event*2 should happen after *step* steps after *event*1 happened
NoObjIfLater =	[*event*,*step*] if *event* hasn't happen after *step* steps of interaction, the constraint for it to happen if exists, does not apply any more

Figure II.15. Syntaxe pour la spécification d'intrigues [85].

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut (se déplacer partout dans le monde virtuel construit en 3D, interagir avec les personnages virtuels).
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car le dernier évolue toujours selon la conception d'intrigue prédéfinie par l'auteur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont la conception d'intrigue prédéfinie).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur la conception d'intrigue prédéfinie par l'auteur.
- Il existe un modèle de l'utilisateur, qui permet au système d'estimer l'état courant de l'utilisateur ainsi que de tenir compte de ses expériences précédentes, et donc de pronostiquer sa réaction future.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur s'effectue en langage naturel.
- Un même discours n'est pas capable d'être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.7. Pilotage de discours interactifs pour les jeux vidéo

Dans le cadre de sa thèse [20, 30, 31, 32], Guylain Delmas du laboratoire L3I a défini une nouvelle approche du discours interactif pour les jeux vidéo permettant de construire, en cours d'exécution, un discours adapté aux actions du joueur. Pour cela, il a développé un système de pilotage qui était inspiré du jeu de rôle. Son objectif est de combiner soit l'interactivité et une structuration du discours (prologue, épisode, dénouement), soit l'interactivité et une évolution de la tension dramatique (voir Figure II.16). Ce système satisfait trois critères : le joueur ne doit pas être contraint par le jeu, le jeu doit proposer un environnement cohérent, et le déroulement du jeu doit suivre une structure de discours ou une évolution de la tension dramatique prédéfinie.

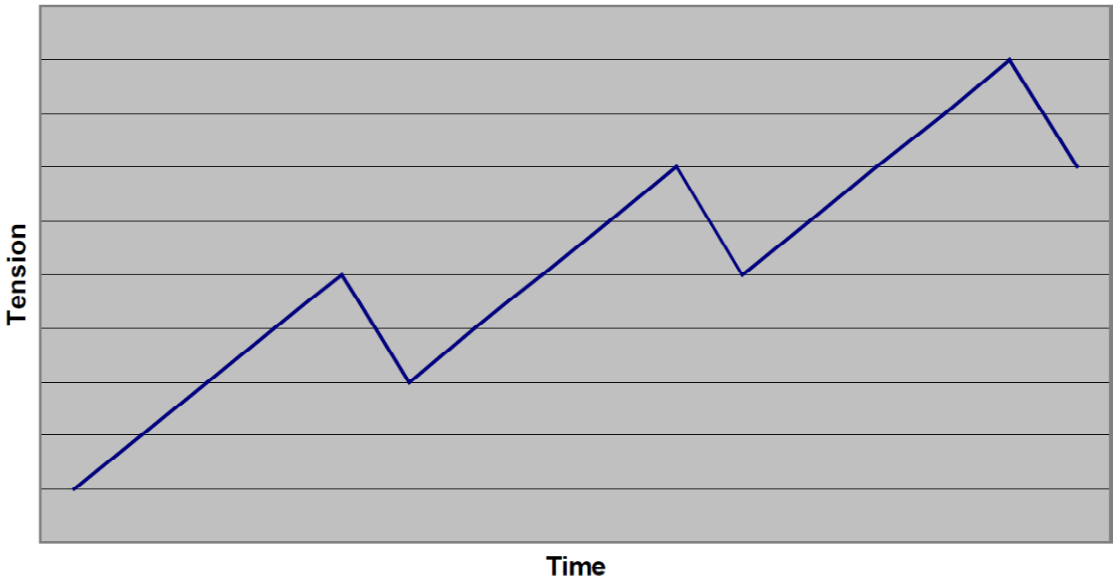


Figure II.16. Évolution typique de la tension dramatique dans un discours [30].

Le schéma d'organisation général du système est donné dans la Figure II.17 où :

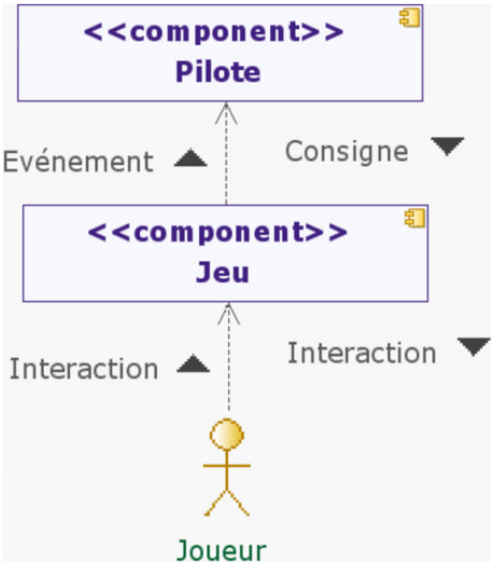


Figure II.17. Schéma d'organisation général du système [30].

- Le jeu est l'élément central, celui sur lequel agissent le pilote et le joueur. Le jeu leur renvoie des informations sur son état. Il comprend un environnement virtuel et des interfaces permettant l'interaction avec le joueur d'une part, et du pilote d'autre part.
- Le joueur dirige son avatar et observe le jeu via l'interface dédiée (par conséquent, les informations qu'il utilise sont limitées à ce que cet avatar peut observer dans le jeu).
- Le pilote emploie sa propre interface pour se renseigner sur le jeu et lui envoyer des consignes. L'information offerte au pilote, sur l'état du jeu, est exhaustive, cela l'aide à orienter le déroulement du jeu en fonction de la structure du discours ou l'évolution de la tension dramatique prédéfinie. Néanmoins, les décisions du pilote ne peuvent pas empêcher les actions du joueur.

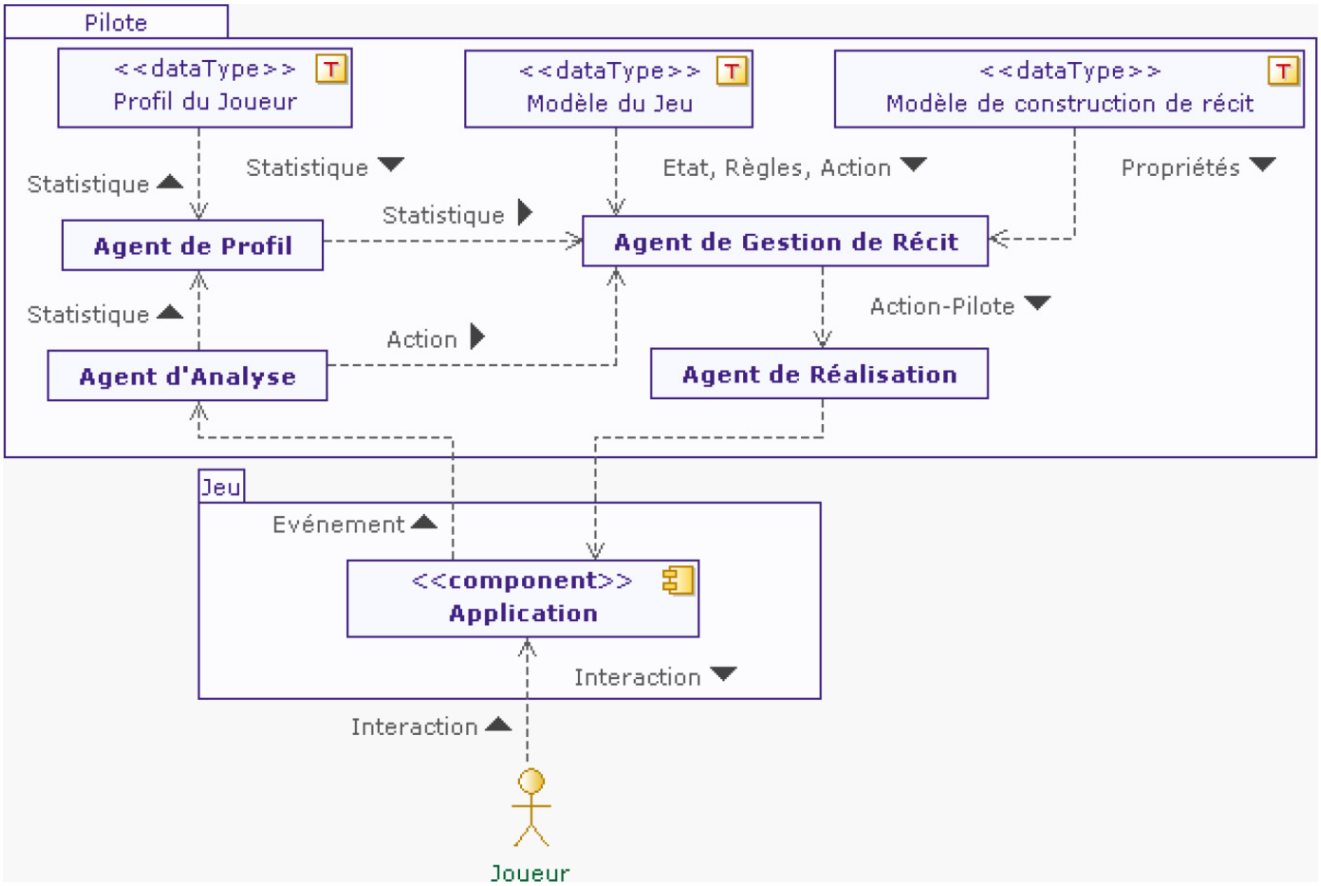


Figure II.18. Architecture du pilote [30].

Dans cette organisation, le pilote est conçu sous la forme d'un système composé de 4 agents (voir Figure II.18) :

- Agent d'Analyse : C'est l'agent en charge de la récupération des informations remontant du jeu, et de leur propagation au sein du pilote. Il travaille ainsi à partir d'EVENEMENTS (informations montantes) qu'il est responsable de traduire en STATISTIQUES (informations à l'usage de l'Agent de Profil) et en ACTIONS (informations à l'usage de l'Agent de Gestion de Récit).
- Agent de Profil : Il a pour fonction de tenir à jour le Profil du Joueur et de répondre aux requêtes le concernant. Il reçoit pour cela des STATISTIQUES (informations sur le comportement du joueur) qui lui sont transmises par l'Agent d'Analyse. A chaque STATISTIQUE reçue, il met à jour le champ correspondant dans le Profil du Joueur. L'Agent de Profil peut être interrogé par l'Agent de Gestion de Récit sur tout ou partie des informations concernant le joueur. Dans ce cas l'Agent de Profil reçoit une requête lui indiquant les champs concernés, il y répond en retournant les valeurs correspondantes.
- Agent de Gestion de Récit : Cet agent constitue le cœur du pilote. Il est en charge de l'ensemble des opérations relevant du suivi du jeu et de la prise de décision lors de l'exécution. Le suivi de l'état de jeu intervient lorsque l'agent reçoit de nouvelles ACTIONS de la part de l'Agent d'Analyse. Ensuite, il calcule les actions suivantes à exécuter (en basant sur les informations dans le Profil du Joueur, le Modèle du Jeu

ainsi que le Modèle de construction de récit – ce qui contient la structure de discours ou l'évolution de la tension dramatique prédéfinie), et les transfère à l'Agent de Réalisation.

- Agent de Réalisation : Il a pour but de faire redescendre les informations provenant de l'Agent de Gestion de Récit vers le jeu. Pour cela, il reçoit de l'Agent de Gestion de Récit une liste d'ACTIONS-PILOTE qu'il convertit en CONSIGNES afin de dérouler le jeu.

Ainsi, on peut trouver que, les travaux, réalisés par Guylain Delmas dans le cadre de notre laboratoire – L3I, répondent aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut dans l'environnement virtuel.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car le dernier évolue toujours selon une structure de discours ou une évolution de la tension dramatique prédéfinie par l'auteur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont une structure de discours ou une évolution de la tension dramatique prédéfinie).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur.
- Il existe un modèle de l'utilisateur, qui contient les informations concernant le joueur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur ne s'effectue pas en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.1.8. Bilan

Nous avons présenté, dans cette section, les principales approches concernant le pilotage de narration interactive dans lesquelles, l'utilisateur déroule l'histoire en dirigeant son avatar dans l'environnement virtuel, ce qui correspond à un personnage joueur. Par conséquent, il ne peut infléchir le discours qu'à travers les actions que son avatar peut réaliser.

Le Tableau II.1 donne un bilan des réponses de ces travaux aux questions citées au début de la Section II.2 grâce auquel, on peut avoir quelques remarques suivantes sur ceux-ci :

- Le niveau de liberté d'interaction de l'utilisateur et le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire sont toujours inverses (l'un est élevé mais l'autre est faible et vice-versa). Ainsi, aucun travail ne satisfait ces deux critères simultanément. Par ailleurs, dans les travaux mentionnés, la préférence va sur la liberté d'interaction laissée à l'utilisateur (cinq travaux sur sept).
- Le déroulement d'une histoire, dans ces travaux, est toujours basé sur le respect des contraintes prédéfinies par l'auteur. Par conséquent, le niveau de pertinence des discours créés est toujours élevé.

Note **E** – **É**levé **M** – **M**oyen **F** – **F**aible **O** – **O**ui **N** – **N**on	Façade	IDtension	Mimesis	Interactive Drama Architecture	Exécution adaptative de trame narrative	Thespian	Pilotage de discours interactifs pour les jeux vidéo
Liberté d'interaction de l'utilisateur	E	F	E	E	F	E	E
Influence de l'utilisateur sur le déroulement de l'histoire	F	E	F	F	E	F	F
Contraintes prédéfinies par l'auteur	O	O	O	O	O	O	O
Pertinence des discours créés	E	E	E	E	E	E	E
Modèle de l'utilisateur	O	O	O	O	N	O	O
Mémoire de l'histoire	O	N	N	N	N	N	N
Interaction de l'utilisateur en langage naturel	O	N	N	N	N	O	N
Possibilité de changer la présentation d'un même discours	N	N	N	N	N	N	N
Dissociation entre l'environnement virtuel et le pilotage	O	O	O	O	O	O	O

Tableau II.1. Bilan des travaux concernant le pilotage de narration interactive dans lesquels l'utilisateur contrôle un personnage joueur.

- Six travaux sur les sept analysés possèdent un modèle de l'utilisateur, cela est compréhensible car dans ces travaux, l'utilisateur contrôle un personnage joueur, et ainsi les informations le concernant sont très importantes pour le système de pilotage dans la gestion du déroulement de l'histoire selon les contraintes prédéfinies par l'auteur.
- Une seule des solutions (Façade) possède une mémoire de l'histoire qui enregistre les différentes étapes, même si ces informations sont utiles pour l'évaluation de la cohérence du déroulement de l'histoire.
- Deux travaux (Façade et Thespian) permettent une interaction de l'utilisateur en langage naturel, même si cette fonctionnalité rend l'interaction de l'utilisateur plus pratique et plus facile.
- Aucune solution ne comporte le moyen de changer la présentation d'un même discours durant l'exécution (ces travaux peuvent seulement générer les discours différents pendant le déroulement d'une histoire).
- En revanche, tous ces travaux dissocient les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage. Par conséquent, leur système de pilotage se décharge de tous les aspects graphiques et peut s'adapter à de nombreux supports différents.

Dans la section suivante, nous présentons les principales approches concernant le pilotage de narration interactive dans lesquelles, l'utilisateur ne contrôle pas un personnage joueur.

II.2.2. L'utilisateur observe le déroulement de l'histoire et l'influence

Pour les travaux appartenant à ce groupe, l'utilisateur ne contrôle pas un personnage joueur, mais il observe le déroulement de l'histoire dans l'environnement virtuel comme un spectateur, et il peut influencer ce processus. Ainsi, l'utilisateur peut à volonté influer sur l'évolution de l'histoire à travers tous les moyens offerts par le système. Nous présentons tour à tour ci-dessous les approches typiques dans ce groupe, et analysons aussi comment elles répondent aux questions citées au début de la Section II.2.

II.2.2.1. Interactive Storytelling (Teesside)

L'équipe d'*Interactive Storytelling* de l'*University of Teesside* a présenté un prototype de narration interactive orientée personnages autonomes. Il utilise des techniques de planification en temps réel s'appuyant sur des Réseaux de Tâches Hiérarchisées (RTH), et permet la génération dynamique d'histoires [17, 18, 22, 23]. Le prototype a été développé en 3D par le moteur de jeu *Unreal Tournament* et testé dans le cas du sitcom télévisé *Friends*.

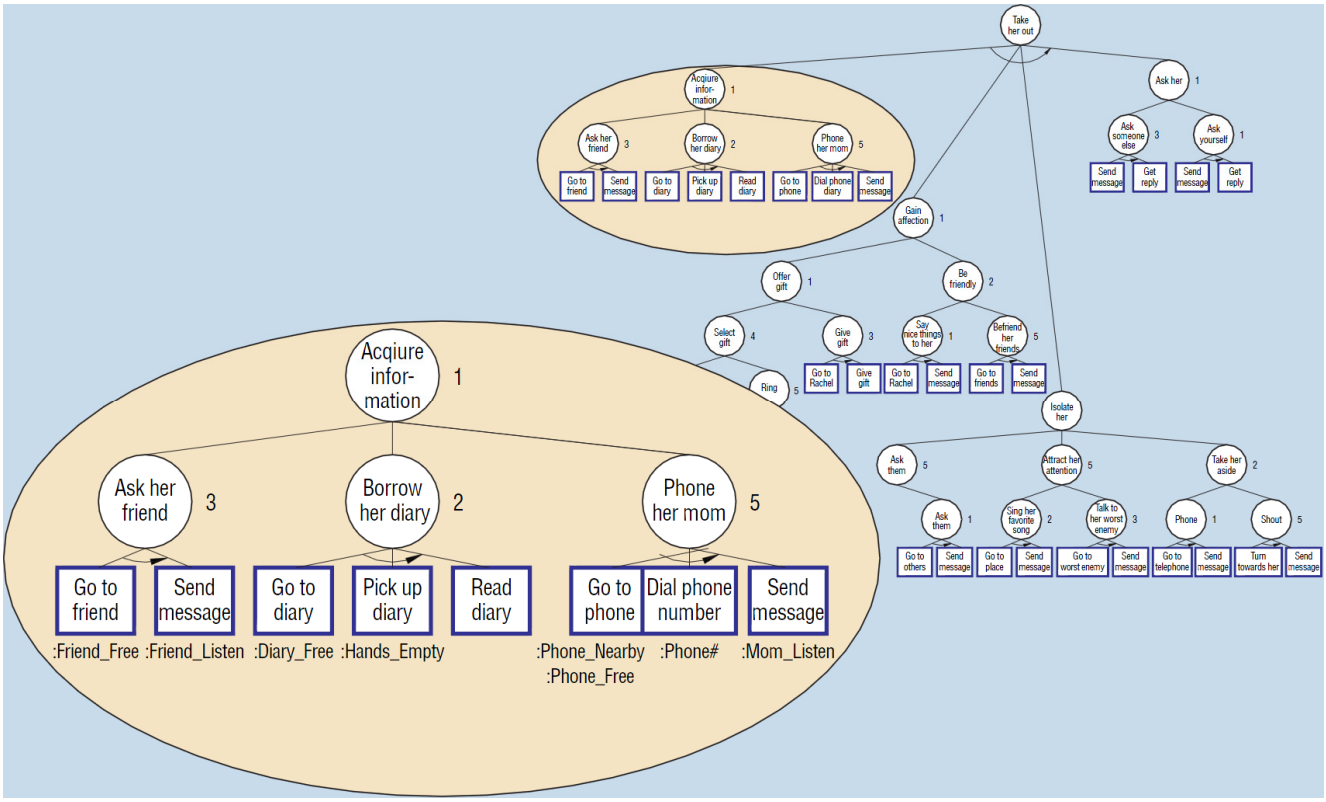


Figure II.19. Un réseau de tâches hiérarchisées pour le personnage principal – Ross du sitcom *Friends* [17].

Un RTH (voir Figure II.19) correspond à plusieurs décompositions possibles d'une tâche principale, autrement dit, on peut considérer les RTHs comme une représentation implicite de l'ensemble de solutions possibles. Par conséquent, chaque décomposition ordonnée constitue la base d'un plan pour un personnage, et chaque RTH associé à un acteur artificiel contient tous les rôles possibles pour ce personnage dans l'histoire. Les préconditions et

postconditions des diverses tâches sont liées aux nœuds correspondant aux tâches. Afin de diriger les actions d'un personnage, le système parcourt son RTH en profondeur et de gauche à droite, il exécute n'importe quelle action primitive qu'il rencontre dans ce processus. Par ailleurs, il permet de revenir en arrière et d'appliquer une autre branche (replanifier) lorsque les actions primitives ont échoué (par exemple, une ressource nécessaire de l'action est prise par un autre personnage ou cachée par une intervention de l'utilisateur).

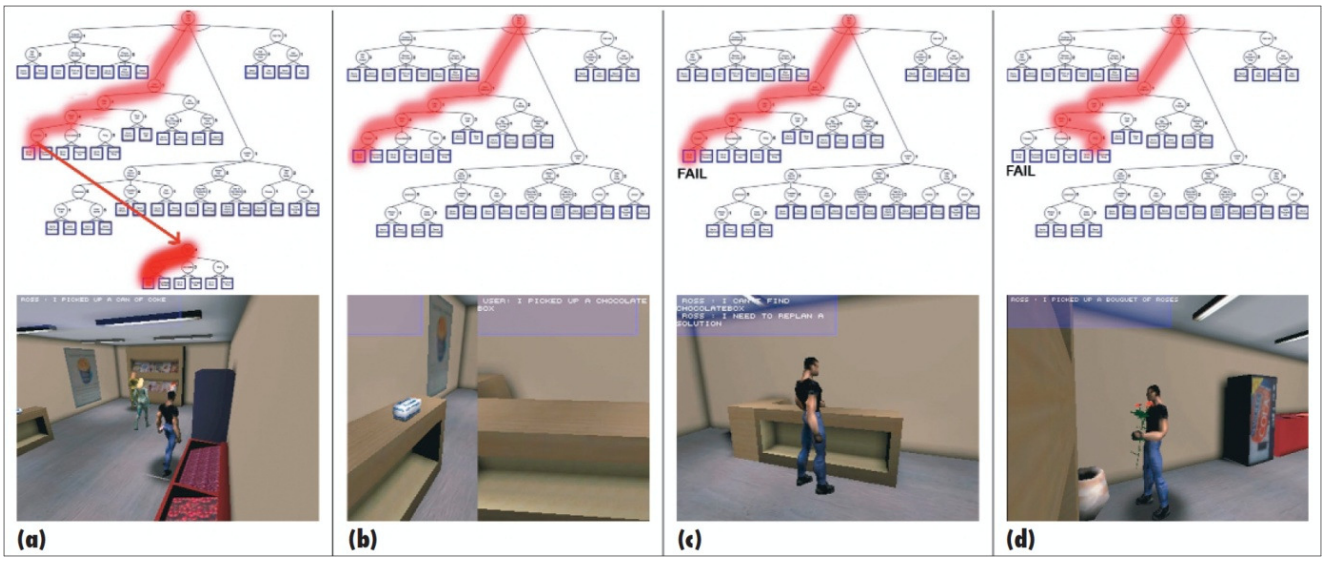


Figure II.20. Une intervention de l'utilisateur : (a) Ross veut prendre une boîte de chocolat ; (b) l'utilisateur voit cela et décide de la « voler » ; (c) Ross ne peut pas trouver la boîte de chocolat ; (d) Ross donc replanifie et prend un bouquet de fleurs [17].

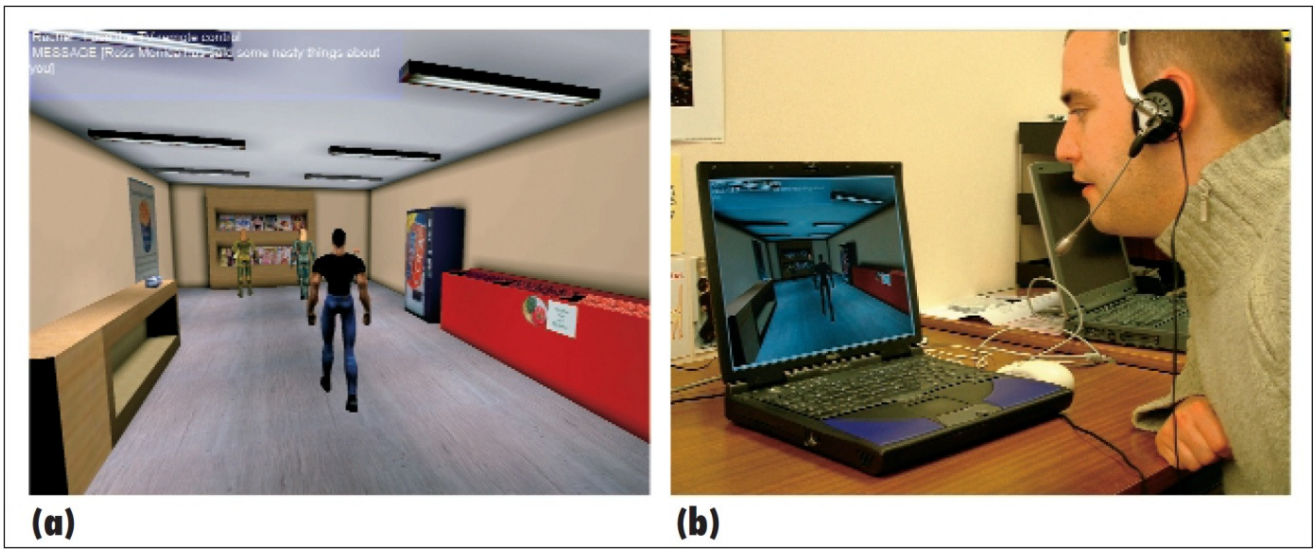


Figure II.21. Une intervention de l'utilisateur : (a) Ross va à la chambre de Rachel afin de lire son journal ; (b) l'utilisateur dit à Ross que Rachel est là [17].

On peut trouver que, dans ce système, les éléments fondamentaux des comportements des acteurs sont prédéterminés par les concepteurs au moyen des RTHs. Cependant, le déroulement du discours généré est toujours imprévisible grâce aux facteurs suivants : la

position initiale des acteurs dans l'environnement virtuel (ce qui influence fortement les ressources que les acteurs peuvent obtenir), les interactions dynamiques entre les acteurs, la sortie aléatoire au cours du déroulement d'actions, l'état émotionnel courant des personnages et surtout l'intervention de l'utilisateur. Dans ce système, l'utilisateur observe l'histoire émergente comme un spectateur, et peut régler l'évolution de la narration à un moment quelconque, en changeant l'environnement virtuel (cacher/donner une ressource – voir Figure II.20), ou en adressant des conseils sous forme vocale aux personnages (à travers un système de reconnaissance de la parole – voir Figure II.21), afin de modifier leurs comportements et de ce fait, créer de nouvelles situations dramatiques ainsi que des terminaisons différentes pour l'histoire.

Les travaux, réalisés par l'équipe d'*Interactive Storytelling* de l'*University of Teesside*, répondent aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut modifier à volonté le comportement des personnages virtuels et donc régler l'évolution de la narration (en changeant l'environnement virtuel, ou en adressant des conseils sous forme vocale aux personnages virtuels).
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé.
- Le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.
- Le niveau de pertinence des discours créés est moyen, car même si le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie, les éléments fondamentaux des comportements des acteurs virtuels sont prédéterminés par l'auteur, au moyen des RTHs.
- Il n'existe pas de modèle de l'utilisateur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur s'effectue en langage naturel.
- Un même discours n'est pas capable d'être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.2.2. LOGTELL

LOGTELL [7, 25, 26, 28] est un outil de modélisation et de simulation appuyé sur la logique pour la génération et la dramatisation interactives d'histoires cohérentes. Il emploie un modèle de logique formel afin de spécifier les événements et le comportement des personnages. En modifiant le modèle, l'utilisateur peut interagir avec l'outil à différents niveaux, en vue d'obtenir une variété d'histoires, qui correspondent à son goût, et en même temps, qui respectent les exigences de cohérence imposées.

L'idée de base de LOGTELL est de permettre de construire, par l'auteur, les caractéristiques d'une histoire à travers un modèle de logique modale temporelle [27], et puis de permettre d'exécuter, par simulation, les discours qui peuvent être créés grâce à ces caractéristiques

combinées avec l'intervention de l'utilisateur. Le modèle de logique modale temporelle est composé d'événements et de règles d'inférence de but. Les événements sont décrits au moyen d'opérations paramétrées avec des préconditions et des postconditions, afin que des algorithmes de planification puissent être utilisés pour la simulation. Les règles d'inférence de but spécifient la manière dont des situations peuvent amener de nouveaux buts pour des personnages (voir Figure II.22 comme un exemple), et donc règlent leur comportement.

$$\forall(\text{VIC},\text{T1},\text{VIL})\ e(\text{T1},\text{kidnapped}(\text{VIC},\text{VIL})) \rightarrow$$
$$\exists \text{T2}\ h(\text{T2},\neg\ (\text{kidnapped}(\text{VIC},\text{VIL}))) \wedge h(\text{T2}>\text{T1})$$

Figure II.22. Une règle d'inférence de but représentée en logique modale temporelle qui signifie : A un moment T1 quelconque, si VIC est kidnappé par VIL, alors ensuite il y a toujours un héros qui le sauve au moment T2 [28].

Pendant la génération des histoires, l'utilisateur peut intervenir soit de façon passive – en laissant les intrigues, qu'il trouve intéressantes, continuer à être engendrées, soit de manière plus active – en essayant de forcer l'exécution des événements et des situations qu'il souhaite. Néanmoins, ces interventions sont rejetées par le système s'il n'y a aucun moyen dans l'histoire de les réaliser. La dramatisation des intrigues peut être activée n'importe quand, pour exposer les intrigues partielles ou finales, où les personnages sont représentés par les acteurs dans un monde en 3D, tandis que l'utilisateur n'est pas autorisé à intervenir pendant cette étape.

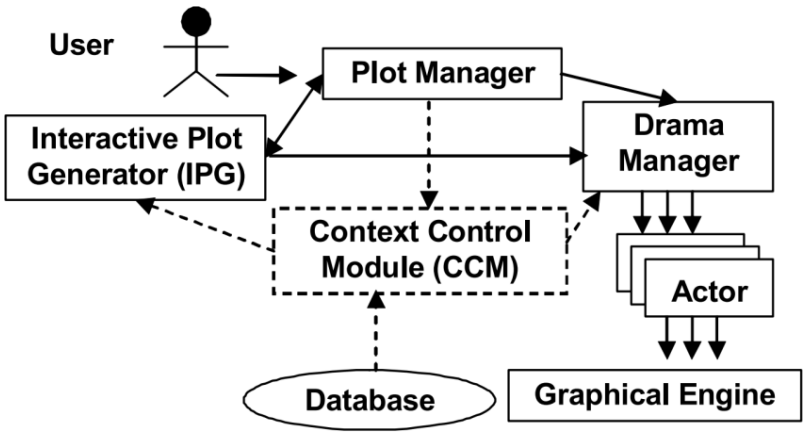


Figure II.23. Architecture modulaire de LOGTELL [28].

L'architecture modulaire de LOGTELL est donnée dans la Figure II.23. L'*Interactive Plot Generator* (IPG) a pour fonction de générer les intrigues partielles de l'histoire. Le *Plot Manager* les reçoit, permet à l'utilisateur d'intervenir sur ces intrigues (voir Figure II.24), et demande à l'IPG d'adapter les intrigues selon les interventions de l'utilisateur. Afin de visualiser la dramatisation d'une intrigue (finale ou partielle), l'utilisateur sélectionne à volonté un ensemble ordonné d'événements, et demande au *Plot Manager* d'activer le *Drama Manager*. Il est responsable du contrôle des acteurs virtuels dans un environnement en 3D (voir Figure II.25). Pendant cette étape, le *Drama Manager* consulte l'IPG afin de garder la cohérence entre les représentations logiques et graphiques. Le *Context Control Module* met

toutes les données de l'histoire dans une base de données unique qui est directement accessible par tous les modules.

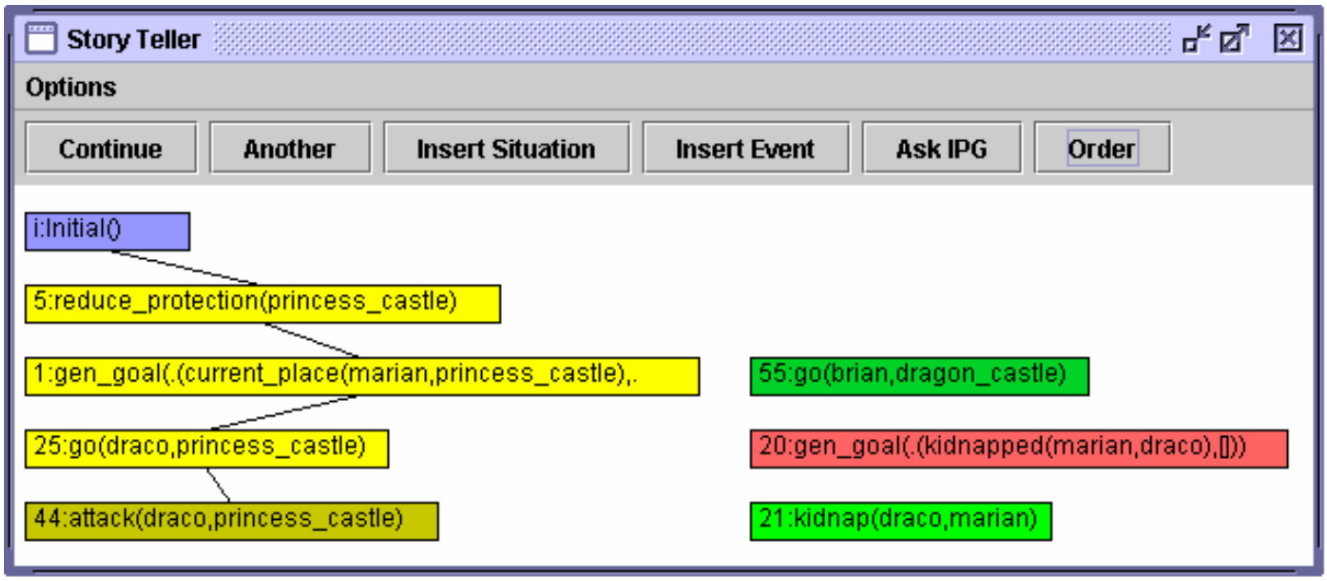


Figure II.24. Interface du *Plot Manager* permettant l'intervention de l'utilisateur sur les intrigues de l'histoire (accepter/refuser les solutions proposées par l'IPG, décrire des situations/événements qu'il souhaite voir apparaître à un moment spécifique…) [28].

Figure II.25. Étape de dramatisation de LOGTELL [28].

Ainsi, on peut trouver que, LOGTELL répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut faire tout ce qu'il veut pour influencer les intrigues de l'histoire correspondant à son goût (accepter/refuser les solutions proposées par l'IPG, décrire des situations/événements qu'il souhaite voir apparaître à un moment spécifique…).

- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est faible car ses interventions sont rejetées par le système s'il n'y a aucun moyen dans l'histoire de les réaliser.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont les caractéristiques de l'histoire construites par l'auteur, à travers un modèle de logique modale temporelle).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur.
- Il n'existe pas de modèle de l'utilisateur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur n'est pas réalisée en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.2.3. FearNot!

Le *Centre for Virtual Environments* de l'*University of Salford* a présenté et décrit les réflexions préliminaires sur le concept « narration émergente » dans [4]. Cette approche a ensuite été utilisée afin de construire *FearNot!* (**F**un with **e**mpathic **a**gents **r**eaching **N**ovel **o**utcomes in **t**eaching) [5, 6, 48, 49] – une application de drame virtuel à but pédagogique pour lutter contre l'intimidation (voir Figure II.26), dans laquelle il n'y avait ni terminaisons ni lignes d'événements en temps prédéterminées. Le développement de l'histoire est géré par tous les personnages (l'utilisateur/les agents intelligents), et dépend entièrement de leurs interactions ainsi que celles avec l'environnement.

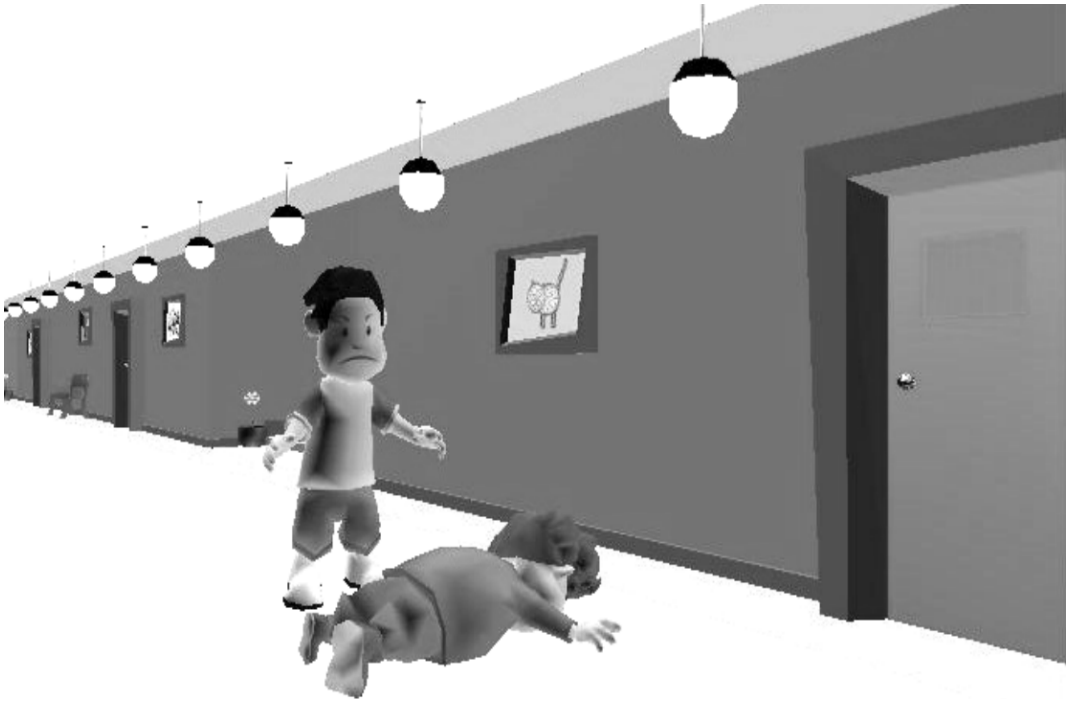


Figure II.26. Une scène dans *FearNot!* [5] qui est construite en 3D par le moteur de jeu *OGRE 3D* [112].

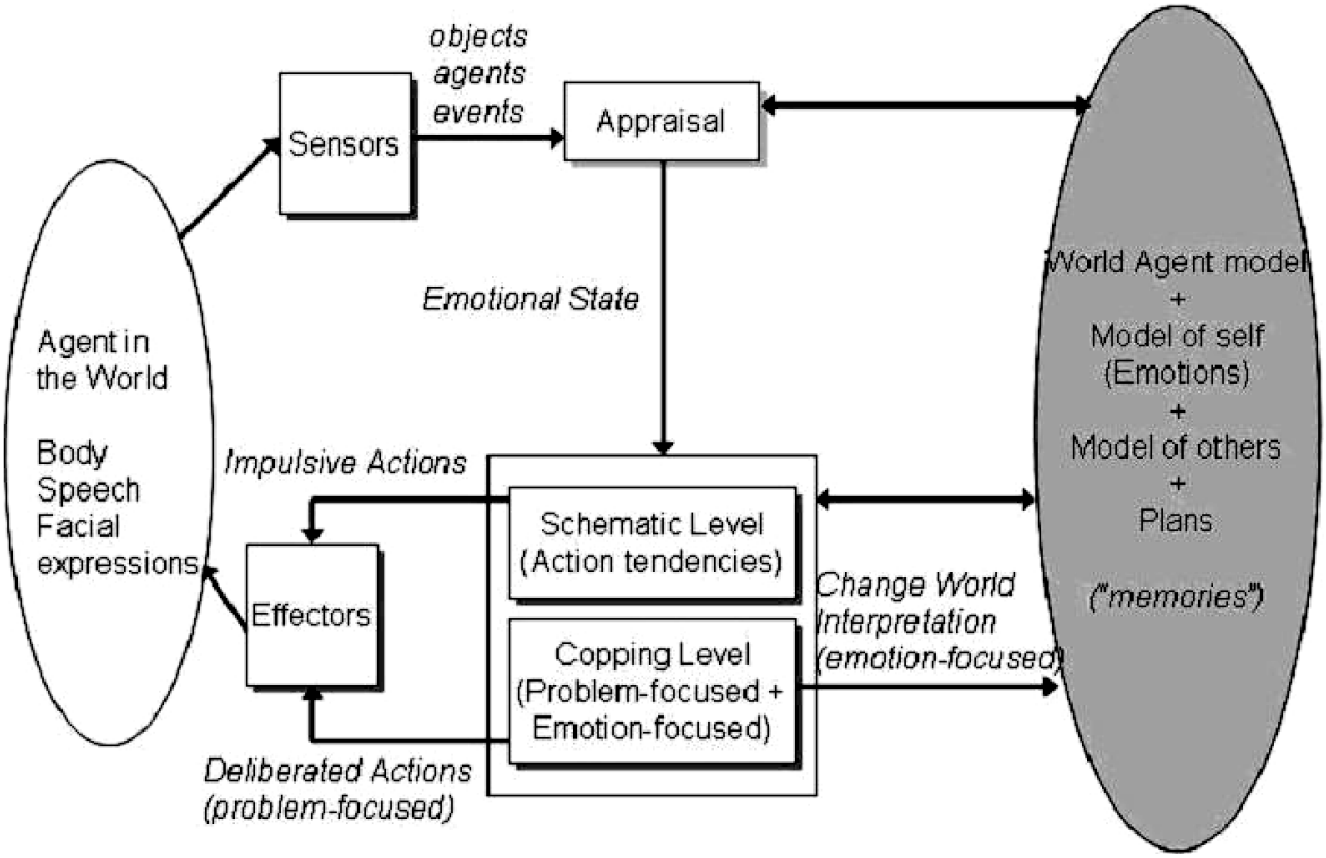


Figure II.27. Architecture orientée évaluation des agents [5].

Pour cela, une architecture orientée évaluation (voir Figure II.27) a été mise en œuvre dans *FearNot!* comme un mécanisme permettant aux agents de sélectionner de façon autonome leurs réactions, en vue de générer un discours qui émerge des intrigues. Chaque agent, dans le monde de l'histoire (qui exprime un personnage), perçoit l'environnement à travers un ensemble de capteurs (réalisant la perception des événements, des objets, *etc*. du monde), et agit dans l'environnement à travers ses effecteurs (permettant d'exécuter des actions différentes). Après avoir reçu un percept, l'agent évalue son importance (selon les objectifs, les convictions du personnage que l'agent représente), et déclenche des émotions appropriées. Ces émotions sont ensuite introduites dans un processus de sélection d'actions à deux niveaux distincts : niveau schématique et niveau réflectif. Le niveau schématique ne comprend que des réactions innées simples (par exemple, si un personnage est battu et il ne peut rien faire, alors il pleure). Le niveau réflectif comprend les réactions qui résultent des buts internes de l'agent et sont prévues. Il y a deux types de comportement à ce deuxième niveau : (1) comportement orienté problème ayant fonction de planifier et d'agir pour atteindre les buts de l'agent, (2) comportement orienté émotion ayant fonction de modifier l'interprétation de l'agent sur le monde virtuel.

La Figure II.28 montre l'interface utilisateur en vue d'interagir avec les agents intelligents dans *FearNot!*. Cet exemple décrit une situation où John (une victime d'une intimidation) demande un conseil de l'utilisateur (un enfant) afin que Luke cesse de l'intimider. L'utilisateur saisit un texte libre dans la zone de texte. L'agent reçoit ce segment textuel, l'analyse et le convertit en réactions pertinentes à travers un système de traitement de langue

fondé sur l'utilisation de patrons (*template-based language system*). Par conséquent, le déroulement de l'histoire est influencé par les actions (conseils) de l'utilisateur et ne suit aucun schéma prédéterminé par les concepteurs.

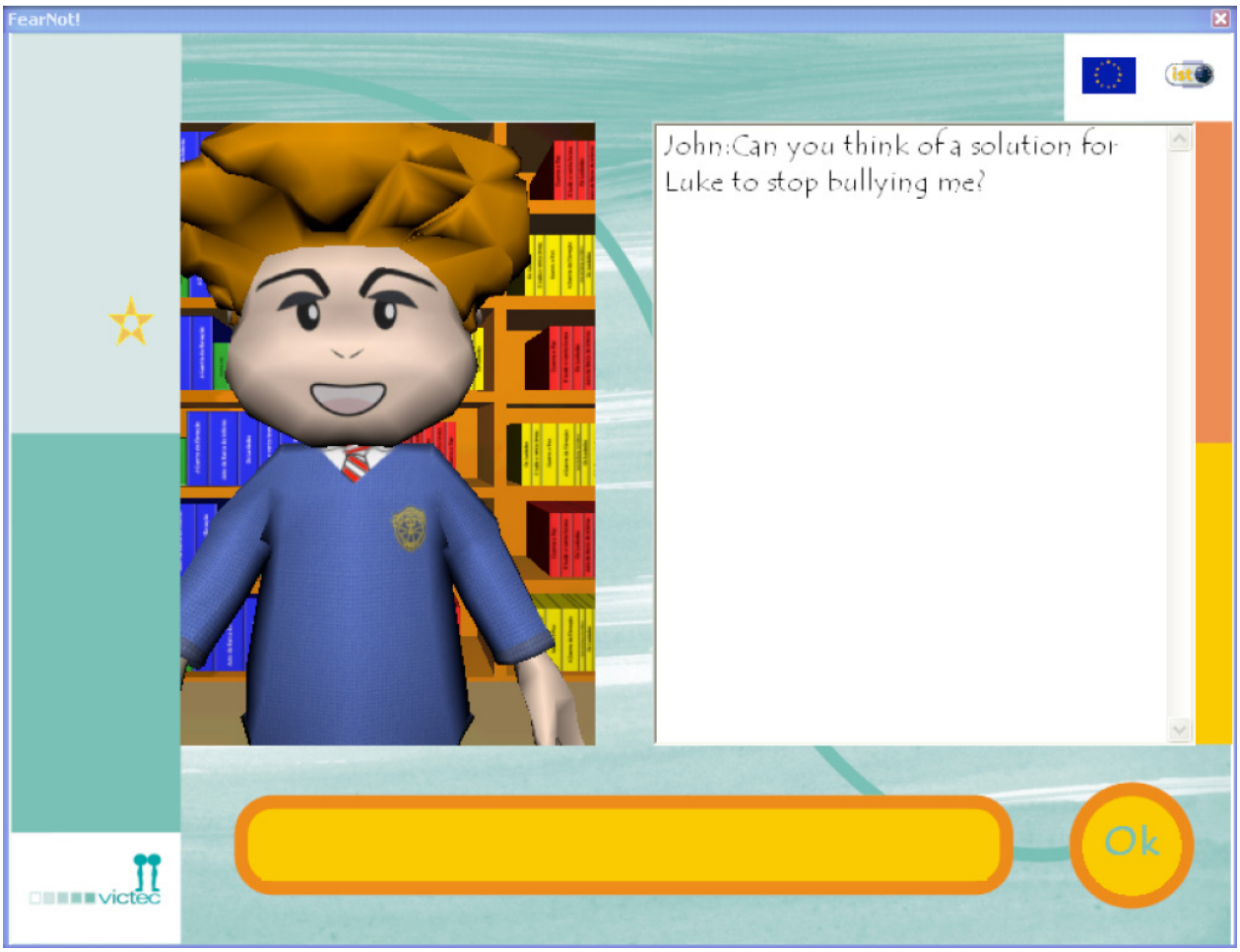


Figure II.28. Interface pour l'interaction avec l'utilisateur dans *FearNot!* [5].

FearNot! répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut donner à volonté des « conseils » aux agents intelligents et donc influencer le déroulement de l'histoire.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé.
- Le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.
- Le niveau de pertinence des discours créés est faible car le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.
- Il n'existe pas de modèle de l'utilisateur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur s'effectue en langage naturel.
- Un même discours n'a pas la possibilité d'être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.2.4. Karo

Les articles [8, 15] décrivent l'emploi de réseaux de Petri afin de représenter et dérouler des intrigues non-linéaires dans des jeux concernant des mondes virtuels associés à des environnements de grande taille (un village, une région, *etc*.). Leur système a été mis en œuvre pour un jeu sérieux *Karo* (voir Figure II.29). Il utilise une architecture de contrôle typique (voir Figure II.30) où :

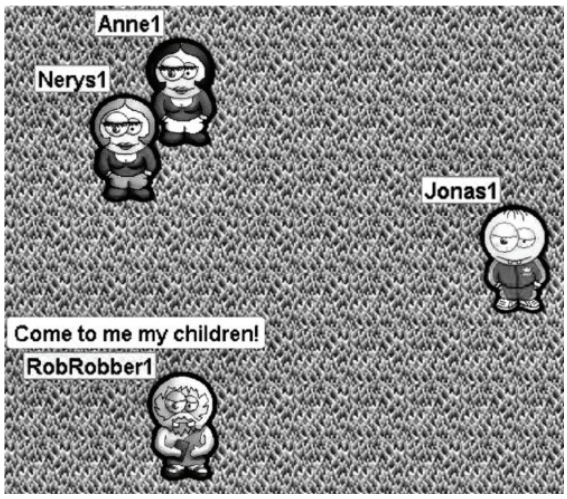


Figure II.29. Une scène dans le jeu sérieux *Karo* [8].

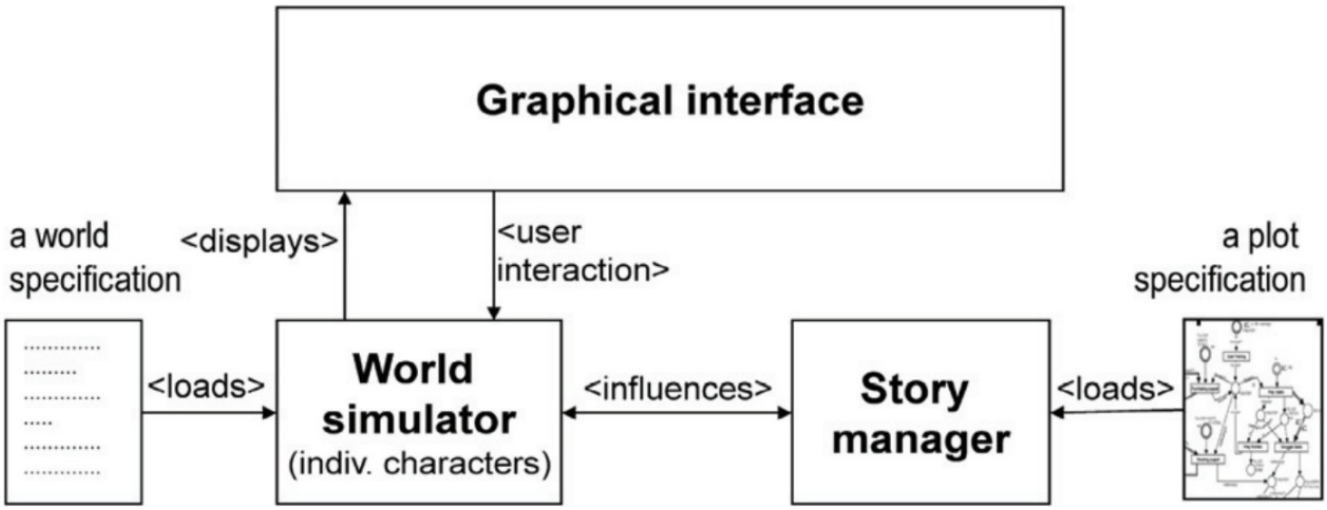


Figure II.30. Architecture de contrôle du système [8].

- Le *World simulator* définit le monde de l'histoire comprenant l'environnement et les personnages virtuels, il contrôle aussi le comportement réactif des personnages à travers des règles *If-Then* hiérarchiques.
- Le *Story manager* est responsable du déroulement des intrigues de l'histoire, qui sont représentées et gérées par les réseaux de Petri. Il modifie également, selon les buts de narration (ceux qui peuvent être changés par l'utilisateur), les objectifs de haut niveau assignés aux personnages virtuels et les états du monde de l'histoire, grâce à l'exécution des réseaux de Petri.

Dans ce système, l'utilisateur observe l'évolution de l'histoire comme un spectateur, et il peut influencer considérablement ce processus en réglant les émotions, l'humeur et les relations entre les personnages ainsi qu'en modifiant l'environnement virtuel (tel que déplacer des objets…).

Karo répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut à volonté régler les émotions, l'humeur et les relations entre les personnages virtuels ainsi que modifier l'environnement virtuel pour influer sur l'évolution de l'histoire.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé.
- Le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.
- Le niveau de pertinence des discours créés est faible car le déroulement de l'histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.
- Il n'existe pas de modèle de l'utilisateur.
- Il n'existe pas de mémoire de l'histoire qui enregistre les différentes étapes.
- L'interaction de l'utilisateur n'est pas effectuée en langage naturel.
- Un même discours ne peut être narré de façons différentes.
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.2.5. PAPOUS

Les travaux effectués dans le projet PAPOUS [76, 89, 90, 91] de l'*Intelligent Agent and Synthetic Characters Group* (Université de Lisbonne) sont un peu différents de ceux présentés dans les sections précédentes. En effet, son objectif est la construction d'un conteur virtuel crédible visualisé en 3D, qui permet d'adapter la façon dont l'histoire est narrée en réponse aux réactions de l'auditeur (un enfant). L'auditeur interagit avec le système en insérant les cartes dans celui-ci, à travers une interface tangible, pour influencer ce qui se passera plus tard.

L'architecture du système est donnée dans la Figure II.31, son fonctionnement est comme suit :

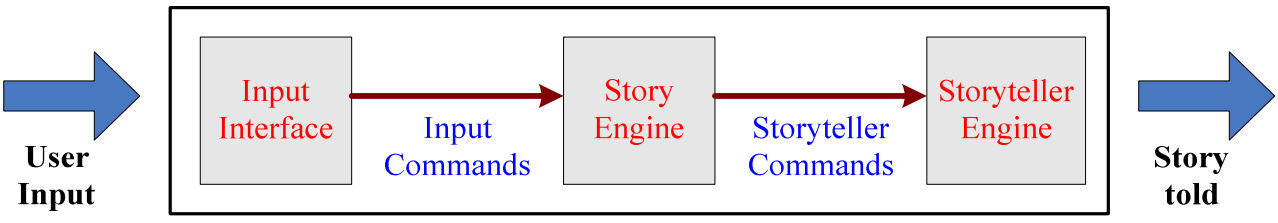


Figure II.31. Architecture du système [89].

- L'utilisateur fournit une certaine entrée au système afin que le conteur virtuel puisse décider comment raconter l'histoire.

- Le module *Input Interface* est responsable de la reception et du traitement des entrées de l'utilisateur, en organisant ces entrées en vue d'un traitement futur par le module *Story Engine*.
- Le module *Story Engine* est responsable de l'analyse de l'histoire (qui est prédéfinie par l'auteur), en l'organisant et en maintenant les informations nécessaires, afin de décider comment la narrer selon l'entrée obtenue à partir du module *Input Interface*.
- Le module *Storyteller Engine*, dépendant des décisions faites par le module *Story Engine*, est responsable du traitement des pièces d'histoire (*StoryBits*) à raconter et en même temps, de garantir que le comportement du conteur virtuel (actions, voix, visage…) est cohérent avec le déroulement de l'histoire.

Toute la structure de l'histoire, qui est composée de beaucoup de niveaux, est prédéfinie par un auteur humain (voir Figure II.32). Chaque niveau comporte un ou plusieurs *StoryBits* dont chacun possède des propriétés différentes (personnages, événements, fonctions de Propp [82]…). Quand le conteur virtuel progresse entre les niveaux, il doit sélectionner un des *StoryBits* dans chacun des niveaux selon l'entrée de l'utilisateur.

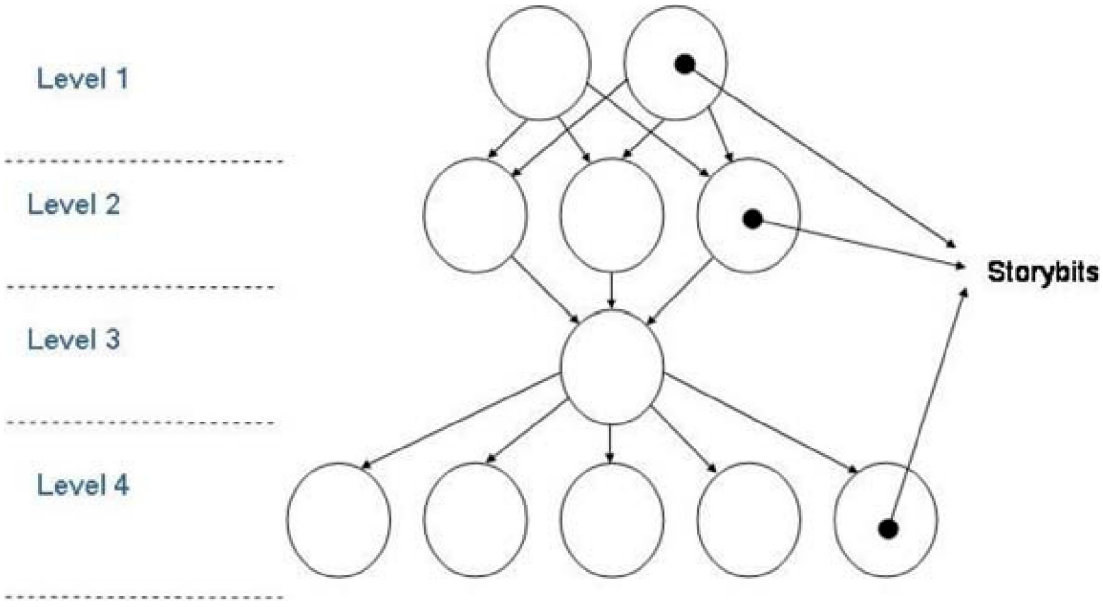


Figure II.32. Structure de l'histoire [89].

Le projet PAPOUS a été déployé. Les développeurs ont décidé d'utiliser une interface tangible (une boîte d'influence) grâce à laquelle, il suffit à l'utilisateur (un enfant) d'insérer des cartes (qui sont marquées avec des codes à barres) afin d'influencer l'histoire (voir Figure II.33). Ainsi, l'auditeur peut choisir la carte qu'il juge la plus appropriée en rapport avec une situation particulière, et puis cette carte influe sur la narration de l'histoire, qui est menée par le conteur virtuel et projetée sur un plan (tel qu'un mur…). A l'intérieur de la boîte d'influence, les cartes insérées sont identifiées, leur signification est envoyée au module *Input Interface* et traitée immédiatement. Ensuite, selon les entrées de l'utilisateur, le conteur virtuel décide de choisir le *StoryBits* le plus pertinent à raconter parmi ceux disponibles, cette décision est basée sur les propriétés de chaque *StoryBits* (émotions, personnages, événements, fonction de Propp…) et les *StoryBits* précédents. Cela s'assure que

la narration (comprenant le comportement du conteur virtuel et le déroulement de l'histoire) est cohérente et influencée par les actions de l'auditeur.

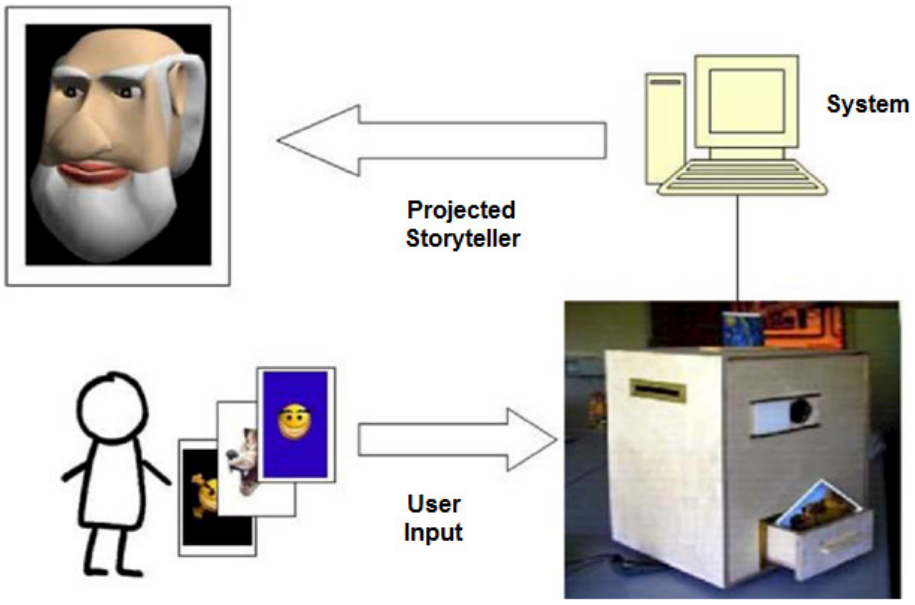


Figure II.33. Fonctionnement du système avec la boîte d'influence [89].

PAPOUS répond aux questions citées au début de la Section II.2 comme suit :

- Le niveau de liberté d'interaction de l'utilisateur est élevé car il peut choisir à voloné des cartes à insérer dans le système et donc influer sur la narration de l'histoire.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est moyen car même si l'évolution de l'histoire est modifiée par l'intervention de l'utilisateur, elle reste toujours cadrée par la structure d'histoire prédéfinie par l'auteur.
- Le déroulement de l'histoire est basé sur le respect des contraintes prédéfinies par l'auteur (les contraintes dans ce cas sont une structure d'histoire prédéfinie).
- Le niveau de pertinence des discours créés est élevé car le déroulement de l'histoire est toujours dans une structure d'histoire prédéfinie par l'auteur.
- Il n'existe pas de modèle de l'utilisateur.
- Il existe une mémoire de l'histoire qui enregistre les *StoryBits* précédents.
- L'interaction de l'utilisateur n'est pas réalisée en langage naturel.
- Un même discours est capable d'être narré de façons différentes, car en plus de permettre à l'utilisateur d'influencer le contenu des discours produits, PAPOUS permet aussi d'influer sur la façon dont ils sont narrés (à travers l'adaptation des actions, de la voix, du visage… du conteur virtuel).
- Le système de pilotage dissocie les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

II.2.2.6. Bilan

Nous avons présenté, dans cette section, les principales approches concernant le pilotage de narration interactive dans lesquelles, l'utilisateur ne contrôle pas un personnage joueur, mais

il observe le déroulement de l'histoire dans l'environnement virtuel comme un spectateur, et il peut influencer ce processus.

Note **E** – **É**levé **M** – **M**oyen **F** – **F**aible **O** – **O**ui **N** – **N**on	Interactive Storytelling (Teesside)	LOGTELL	FearNot!	Karo	PAPOUS
Liberté d'interaction de l'utilisateur	E	E	E	E	E
Influence de l'utilisateur sur le déroulement de l'histoire	E	F	E	E	M
Contraintes prédéfinies par l'auteur	N	O	N	N	O
Pertinence des discours créés	M	E	F	F	E
Modèle de l'utilisateur	N	N	N	N	N
Mémoire de l'histoire	N	N	N	N	O
Interaction de l'utilisateur en langage naturel	O	N	O	N	N
Possibilité de changer la présentation d'un même discours	N	N	N	N	O
Dissociation entre l'environnement virtuel et le pilotage	O	O	O	O	O

Tableau II.2. Bilan des travaux concernant le pilotage de narration interactive dans lesquels l'utilisateur ne contrôle pas un personnage joueur.

Le Tableau II.2 donne un bilan des réponses de ces travaux aux questions citées au début de la Section II.2. Nous pouvous en déduire la synthèse de cette étude :

- Le niveau de liberté d'interaction de l'utilisateur est toujours élevé.
- En général, le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé (sauf celui de LOGTELL et celui de PAPOUS qui sont faible et moyen respectivement).
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est élevé si le dernier n'est basé sur aucune contrainte prédéfinie par l'auteur (et vice-versa).
- Dans deux des travaux sur les cinq présentés (LOGTELL et PAPOUS), le déroulement d'une histoire est basé sur le respect des contraintes prédéfinies par l'auteur. Par conséquent, le niveau de pertinence des discours créés, dans ces deux travaux, est toujours élevé.
- Seul dans le travail *Interactive Storytelling (Teesside)*, le niveau de pertinence des discours créés est moyen (celui de *FearNot!* et celui de *Karo* sont faibles), même si le déroulement d'une histoire n'est basé sur aucune contrainte prédéfinie par l'auteur.

- Comme l'utilisateur ne contrôle pas un personnage joueur, les solutions ne possèdent pas de modèle utilisateur.
- Seul *PAPOUS* possède une mémoire de l'histoire qui enregistre les différentes étapes, même si ces informations sont utiles pour l'évaluation de la cohérence du déroulement de l'histoire.
- Il n'y a que deux travaux sur cinq (Interactive Storytelling (Teesside) et FearNot!) permettant une interaction de l'utilisateur en langage naturel, même si cette fonctionnalité rend l'interaction d'utilisateur plus pratique et plus facile.
- Seul *PAPOUS* permet de changer la présentation d'un même discours (les autres peuvent seulement générer les discours différents pendant le déroulement d'une histoire).
- En revanche, tous ces travaux dissocient les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage. Par conséquent, leur système de pilotage se décharge de tous les aspects graphiques et peut s'adapter à de nombreux supports différents.

Dans la section suivante, nous dressons un bilan général sur tous les travaux, concernant le pilotage de narration interactive, que nous avons présenté dans ce chapitre.

II.2.3. Bilan général

Dans cette section, nous avons mentionné les principales approches sur le pilotage de narration interactive, dont l'objectif est de résoudre le problème de déroulement d'une histoire de sorte que les utilisateurs puissent influer sur celui-ci. Nous avons classé ces travaux selon deux groupes : (1) l'utilisateur contrôle un personnage joueur (c'est aussi notre approche dans le cadre de cette thèse), (2) l'utilisateur observe le déroulement de l'histoire et l'influence.

Le Tableau II.3 donne un bilan des réponses de ces travaux aux questions citées au début de la Section II.2. Nous en déduisons les remarques suivantes :

- Le niveau de liberté d'interaction de l'utilisateur est élevé dans la presque totalité de ces travaux (dix travaux sur douze).
- En général, le niveau de liberté d'interaction de l'utilisateur et le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire sont inverses (l'un est élevé mais l'autre est faible et vice-versa), pour trois des travaux sur les douze seulement tous ces deux critères sont élevés.
- Le niveau d'influence de l'interaction de l'utilisateur sur le déroulement de l'histoire est de valeur inverse au respect des contraintes prédéfinies par l'auteur (cette remarque est vraie pour dix travaux sur douze).
- Le déroulement d'une histoire, dans la plupart de ces travaux (neuf travaux sur douze), est basé sur le respect des contraintes prédéfinies par l'auteur. Par conséquent, le niveau de pertinence des discours qu'ils créent est élevé.
- Il n'y a que deux travaux sur les douze présentés (Façade et PAPOUS) qui possèdent une mémoire de l'histoire permettant d'enregistrer les différentes étapes.

Note **E** – **É**levé **M** – **M**oyen **F** – **F**aible **O** – **O**ui **N** – **N**on	L'utilisateur contrôle un personnage joueur							L'utilisateur observe le déroulement de l'histoire et l'influence				
	Façade	IDtension	Mimesis	Interactive Drama Architecture	Exécution adaptative	Thespian	Pilotage de discours interactifs	IS (Teesside)	LOGTELL	FearNot!	Karo	PAPOUS
Liberté d'interaction de l'utilisateur	E	F	E	E	F	E	E	E	E	E	E	E
Influence de l'utilisateur sur le déroulement de l'histoire	F	E	F	F	E	F	F	E	F	E	E	M
Contraintes prédéfinies par l'auteur	O	O	O	O	O	O	O	N	O	N	N	O
Pertinence des discours créés	E	E	E	E	E	E	E	M	E	F	F	E
Modèle de l'utilisateur	O	O	O	O	N	O	O	N	N	N	N	N
Mémoire de l'histoire	O	N	N	N	N	N	N	N	N	N	N	O
Interaction de l'utilisateur en langage naturel	O	N	N	N	N	O	N	O	N	O	N	N
Possibilité de changer la présentation d'un même discours	N	N	N	N	N	N	N	N	N	N	N	O
Dissociation entre l'environnement virtuel et le pilotage	O	O	O	O	O	O	O	O	O	O	O	O

Tableau II.3. Bilan des principaux travaux concernant le pilotage de narration interactive.

- Il n'y a que quatre travaux sur les douze présentés (Façade, Thespian, Interactive Storytelling (Teesside) et FearNot!) qui permettent une interaction de l'utilisateur en langage naturel.

- Il n'y a qu'un travail (PAPOUS) permettant de changer la présentation d'un même discours.
- En revanche, tous ces travaux dissocient les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage.

Les approches décrites précédemment nous ont aussi permis d'identifier quelques facteurs cruciaux, comme les éléments d'architecture d'un système de pilotage ; la construction, la représentation, l'exécution de scénarios narratifs ; les propriétés de narration importantes ; l'évolution de référence des paramètres dramatiques ; et la structuration de discours, pour la réalisation de notre système de pilotage de narration interactive. Comme parmi les différents principes contribuant à cette réalisation, nous nous intéressons particulièrement au problème de validation d'un scénario lors de la phase de conception de jeu, nous présentons, dans la section suivante, un tour d'horizon des principaux travaux concernant le problème de production de scénarios valides.

II.3. Production de scénarios valides

Cette section décrit les principales approches sur le problème de production de scénarios valides, celles qui ont été proposées par la communauté en vue de prévenir, détecter et corriger les erreurs dans les scénarios créés – une des questions les plus importantes et difficiles pour garantir la qualité d'une narration interactive. Ce processus de validation, permettant aux utilisateurs de parcourir les discours possibles d'un scénario, est exécuté dans la phase de conception du scénario. Il est répété jusqu'à ce que les utilisateurs reçoivent un scénario qui répond à leurs objectifs.

Les travaux, que nous mentionnons ci-dessous, sont classés selon la manière d'exécuter l'analyse pour le scénario à valider :

- ***Approche fondée sur les traces d'exécution d'un scénario :*** Les travaux, s'appuyant sur cette manière d'analyser, permettent à l'utilisateur d'interagir directement avec un discours à chaque « séance de test », le principe consiste à observer étape par étape les traces d'exécution d'un scénario. Ainsi, le déroulement de l'histoire peut être validé et éventuellement amélioré.
- ***Analyse structurelle d'un scénario :*** Les travaux, s'appuyant sur cette technique d'analyse, permettent à l'utilisateur d'analyser au niveau structurel tous les discours possibles du scénario. Ainsi l'utilisateur peut vérifier si un scénario satisfait à un ensemble de critères/propriétés de narration prédéfini.

Dans la section suivante, nous présentons tour à tour les principaux travaux dans chacun de ces groupes, et discutons aussi comment ils aident l'utilisateur à produire des scénarios valides.

II.3.1. Approche fondée sur les traces d'exécution d'un scénario

II.3.1.1. Scribe

Scribe [38, 71] est un outil grâce auquel les auteurs peuvent organiser et créer le contenu d'histoires dans le cadre de drames interactifs orientés événements (ceux qui sont gérés par un agent directeur intelligent). Ces travaux ont été mis en œuvre pour que des stagiaires

obtiennent des connaissances expérimentalement (à travers leurs actions) dans un environnement virtuel visualisé en 3D. Scribe satisfait un ensemble de critères : généralité (il n'est pas dédié à un environnement de jeu spécifique ou à une histoire spécifique), capacité de débogage (*debugging*), facilité et efficacité de l'utilisation. Pour cela, la structure de Scribe est divisée en trois modes de création différents, chacun possède sa propre fonctionnalité :

Figure II.34. Interface du mode « Rangement d'éléments » [71].

- Rangement d'éléments : Ce mode (voir Figure II.34) permet à l'auteur de construire une carte d'environnement en 2,5D (c'est une figure en 2D – exprimée sous format XML – où la hauteur des composants est annotée). Sur cette carte, l'auteur place des éléments tangibles et configure leurs propriétés. Ces éléments sont des objets/personnages virtuels qui interagissent avec le stagiaire.
- Création d'histoire : Il permet à l'auteur de produire graphiquement la structure de l'intrigue qui décrit la relation des points d'intrigue dans l'histoire (voir Figure II.35). Dans ce mode, la configuration des éléments créés par le mode « Rangement d'éléments » est utilisée pour générer des situations d'histoire cohérentes avec la représentation visuelle au moyen de la carte d'environnement. Chaque point d'intrigue se compose de trois parties (qui sont exprimées sous format XML) :
    - Préconditions : Ce sont les états logiques qui doivent être vrais pour qu'un point d'intrigue puisse être franchi.

- Actions : Elles indiquent des changements dans l'environnement virtuel liés à l'accomplissement d'un point d'intrigue.
- Événements : Ils constituent un ensemble de déclarations dictant les changements qui se produisent au cours du temps dans l'environnement virtuel pendant que le point d'intrigue est franchi.

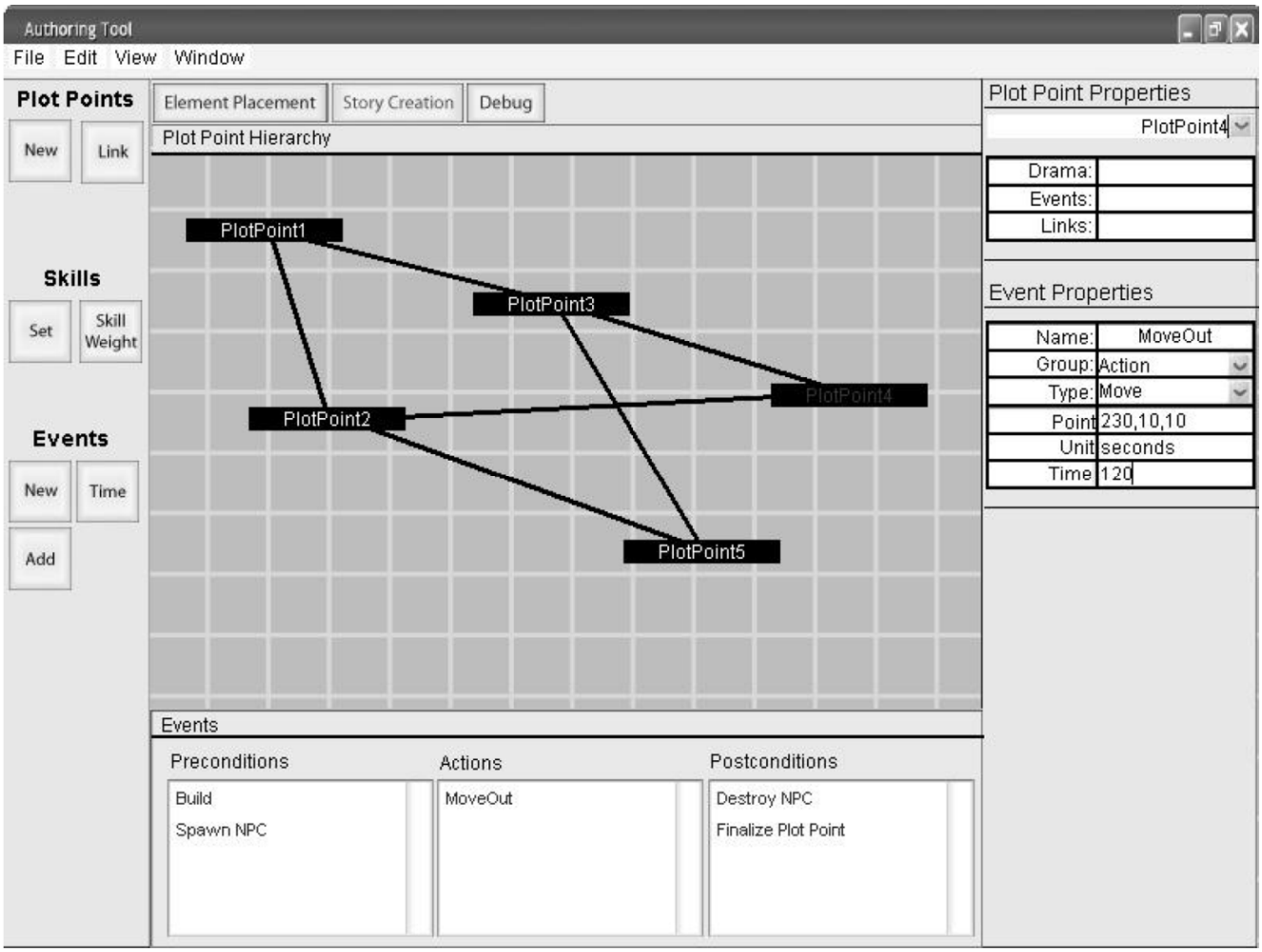


Figure II.35. Interface du mode « Création d'histoire » [71].

- Débogage : Ce mode permet à l'auteur d'interagir directement avec l'agent directeur à l'intérieur de Scribe en vue de simuler ce qui se passe dans l'environnement virtuel. Grâce à cela, l'auteur peut facilement dérouler l'histoire, et il lui est possible de savoir comment l'agent directeur gère diverses situations ainsi que la raison des décisions. Par conséquent, l'auteur peut garantir le contenu de l'histoire et le comportement de l'agent directeur (autrement dit, il peut détecter des problèmes indésirables tels que des décisions déraisonnables/non pertinentes des personnages virtuels, des situations inaccessibles, des blocages…).

Ainsi, Scribe peut aider les auteurs – non programmeurs à construire facilement et rapidement un contenu d'histoire riche et valide pour des drames interactifs, grâce à ses fonctionnalités compréhensibles et faciles d'utilisation.

II.3.1.2. EmoEmma

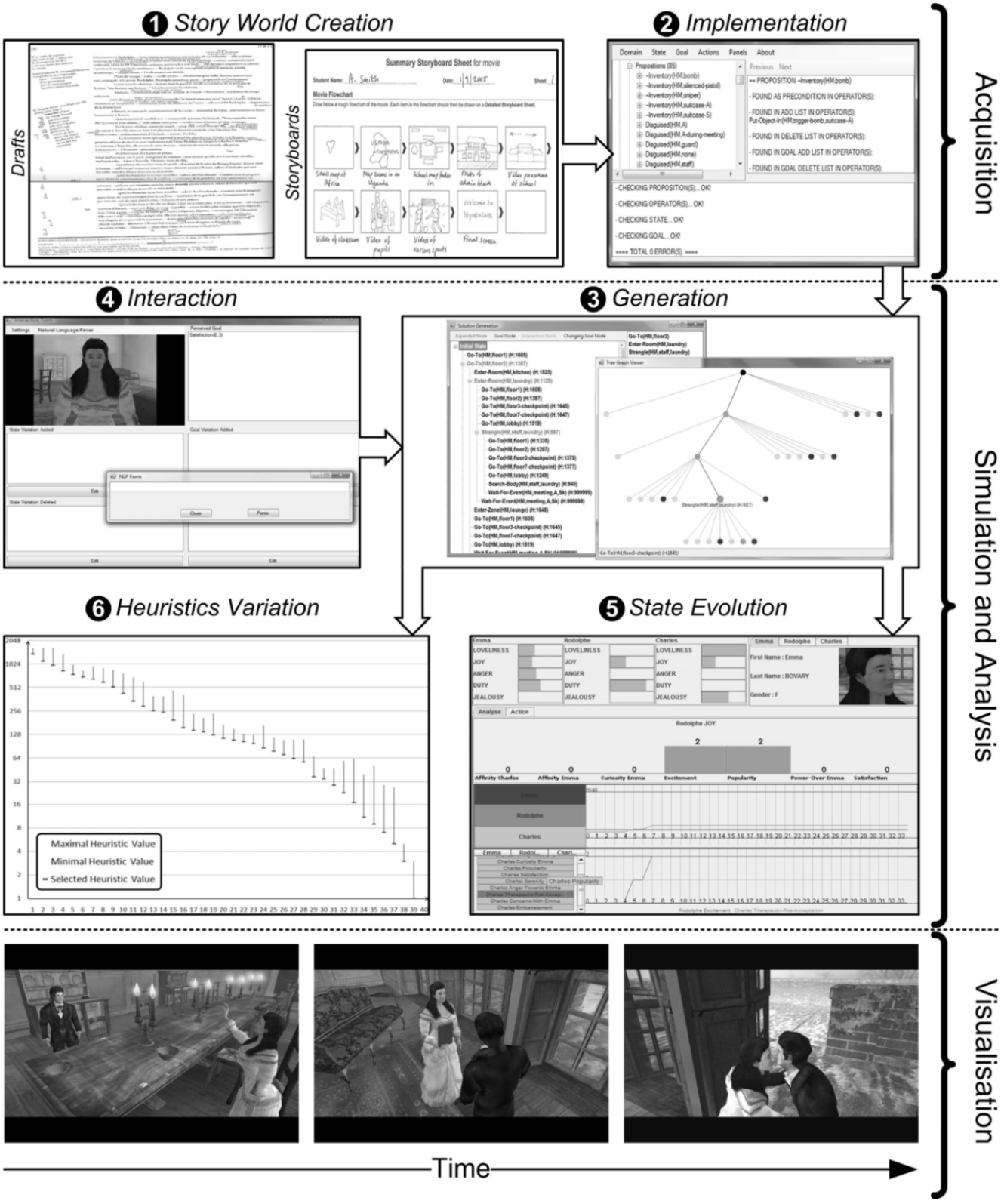


Figure II.36. Processus de création utilisant EmoEmma [78].

Dans [24, 78], les auteurs ont décrit un outil de création, baptisé EmoEmma. Il a été développé pour un système de narration interactive orienté personnages utilisant un prototype

de planification émotionnelle. Il permet de construire un domaine de planification complexe en vérifiant l'exhaustivité et la cohérence du plan généré. La Figure II.36 illustre le processus de création employant EmoEmma, qui se compose de 3 phases principales :

- Acquisition de connaissances : La production du monde de l'histoire (voir Figure II.36-1) constitue l'étape initiale où les éléments de l'histoire sont fabriqués à la main par l'auteur. Ensuite, grâce à quelques interfaces graphiques, il exécute l'étape « mise en œuvre de domaine » (voir Figure II.36-2) où chaque élément du domaine de planification (propositions, opérateurs, états…) est créé. Ici les opérateurs, qui sont exprimés sous format similaire à celui de STRIPS [39], déterminent les actions que les personnages virtuels peuvent exécuter. La description du domaine comprend également une formalisation de l'état initial et de l'état de but, ceux qui correspondent aux objectifs de la scène ou des personnages impliqués.

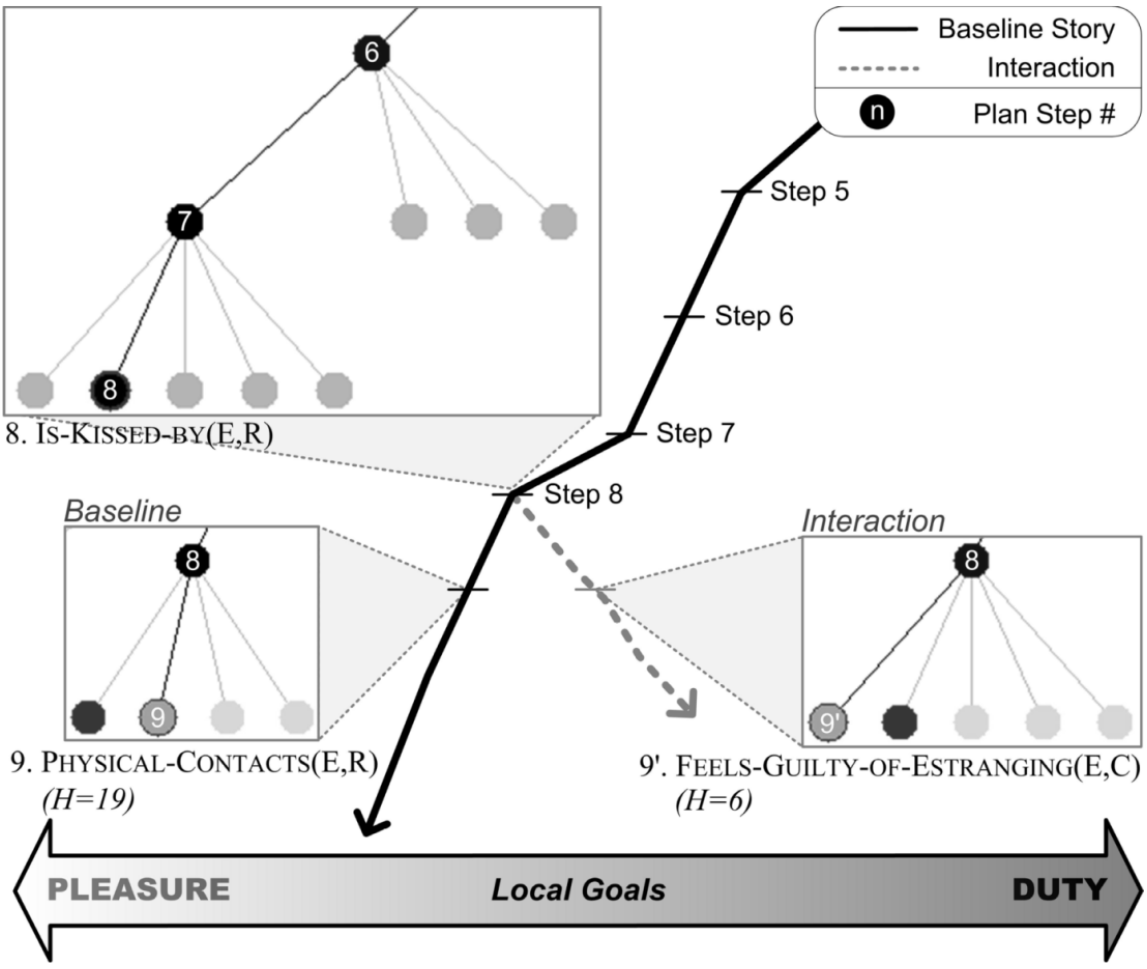


Figure II.37. Impact de l'interaction de l'utilisateur sur le plan généré : au début, l'étape 8 – *Is-Kissed-By(E,R)* – du plan original possède 4 opérateurs parmi lesquels, le système choisit l'opérateur *Physical-Contacts(E,R)* pour l'étape 9 car sa valeur heuristique est la plus grande (H = 19) ; néanmoins, supposons que l'utilisateur n'accepte pas cette solution, il emploie sa parole afin d'interagir avec l'agent virtuel (et donc d'influencer son émotion), l'effet de cette action est l'addition d'une nouvelle proposition dans la conviction de l'agent, par conséquent, un nouvel opérateur *Feels-Guilty-Of-Estranging(E,C)* est ajouté dans le plan original ; car en ce moment la valeur heuristique de cet opérateur est la plus grande, le système le sélectionne pour l'étape 9 [78].

- Simulation et analyse : Cette phase permet à l'utilisateur de voir étape par étape le plan correspondant au monde de l'histoire fabriqué lors de la phase précédente (voir Figure II.36-3), et donc lui permet de l'examiner. À partir de l'état initial, l'utilisateur peut, pour chaque étape, dérouler le plan, en employant une représentation arborescente jusqu'à ce que l'état but soit atteint. Par ailleurs, il peut interagir avec le processus de génération de solution à n'importe quel moment (voir Figure II.36-4) afin de changer le plan courant, en modifiant directement les propositions dans le monde de l'histoire en ce moment-là (voir Figure II.37). Après chaque action de l'utilisateur (accepter le choix du système, proposer un autre choix, ajouter un nouveau choix…), le système propose automatiquement une liste d'actions (suivantes) possibles ainsi que l'action qu'il sélectionne, et attend la décision de l'utilisateur, une nouvelle boucle est donc commencée. Grâce à ce mécanisme, tout l'arbre représentant le plan peut être automatiquement scanné et de ce fait, toutes les solutions possibles peuvent être listées et visualisées. Ainsi, la fonctionnalité ci-dessus du système permet un accès au domaine de planification, à l'inspection d'opérateurs ainsi qu'aux états du monde. Cette fonctionnalité crée également une structure arborescente fournissant une visualisation naturelle des actions et de leurs conséquences. EmoEmma possède aussi la capacité de simulation d'environnements virtuels (voir Figure II.36-4), permettant de simuler ce qui se passe dans le monde correspondant à l'étape courante. Les Figures II.36-5 et II.36-6 montrent les résultats d'analyse utilisés afin de valider la cohérence du plan généré, en listant l'évolution au cours du temps des éléments importants de l'histoire tels que les états des personnages, la valeur heuristique générale…
- Visualisation d'histoire : Cette phase permet aux auteurs de voir tout le déroulement de l'histoire dans l'environnement virtuel selon le plan valide engendré à la phase précédente.

Pour conclure, on constate que, le point fort d'EmoEmma est de permettre à l'utilisateur (1) de voir concrètement toutes les solutions possibles dans le plan créé et ses résultats d'analyse, (2) d'influencer facilement sur la génération du plan à n'importe quel moment. Par conséquent, l'utilisateur peut vérifier l'exhaustivité et la cohérence du contenu narratif produit. Il peut aussi corriger ce contenu si nécessaire et donc produire des plans d'exécution riches et raisonnables.

II.3.1.3. ENIGMA

ENIGMA [59, 60] est un outil de création dédié à l'architecture d'agent fondée sur la narration émergente *FatiMA*, qui est utilisée dans le cas du drame virtuel éducatif contre l'intimidation *FearNot!*. ENIGMA permet aux auteurs – non techniciens de construire automatiquement des descriptions d'objectifs et d'actions des personnages virtuels autonomes en procédant de façon itérative. Ces descriptions (qui sont exprimées au format XML par une représentation similaire à STRIPS) comprennent des préconditions, des postconditions, des règles de réaction émotionnelles, des tendances d'action et des paramètres de personnalité pour chaque agent.

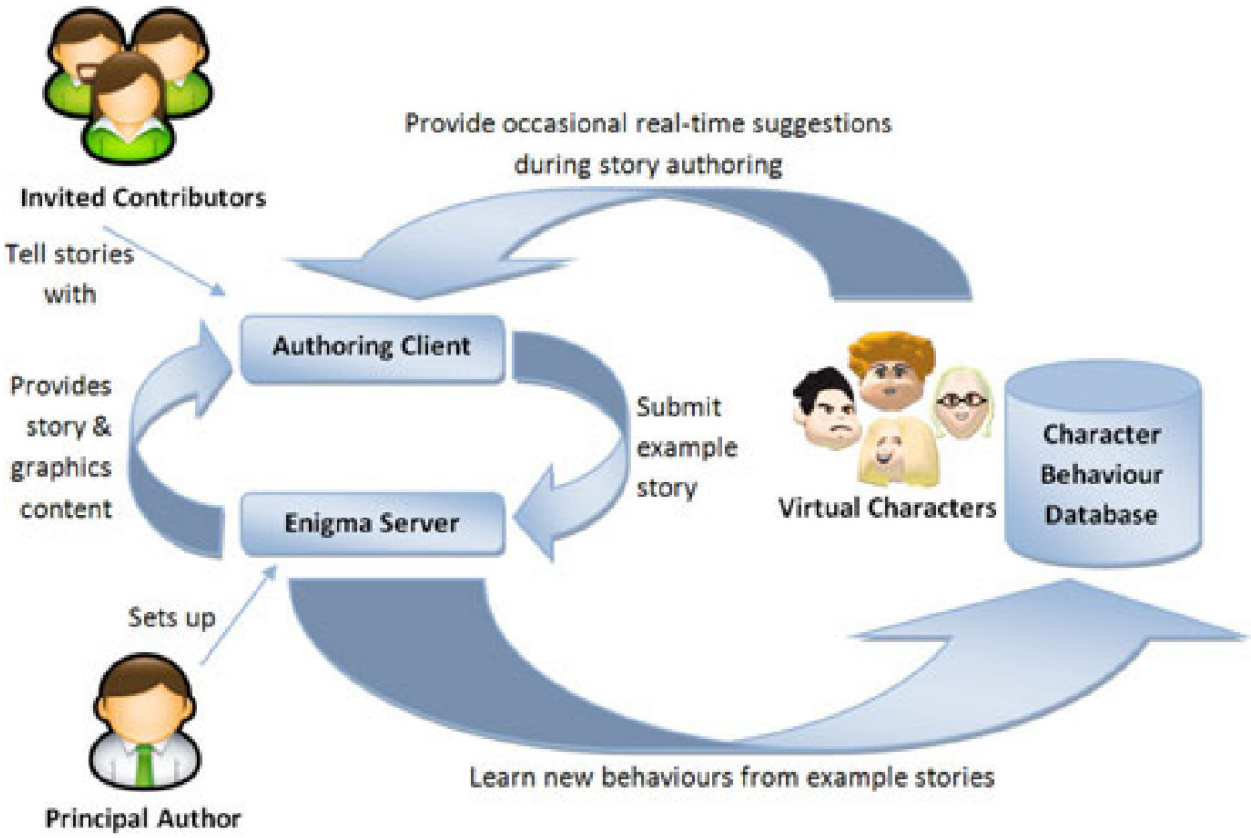


Figure II.38. Architecture du système ENIGMA [59].

La Figure II.38 donne une vue d'ensemble de l'architecture du système ENIGMA. Une application de type client, qui peut être directement exécutée à partir d'un navigateur, permet aux contributeurs, qui sont invités par un auteur principal (AP), de créer une histoire dans des limites indiquées. Ces limites, qui sont établies par l'AP, comprennent une distribution fixe de personnages, un ensemble d'accessoires, de scènes et d'instructions concernant le thème de l'histoire, des histoires précédentes des personnages, *etc*. Chaque histoire, qui est créée au sein de cette application de type client, est transmise à un serveur qui collecte l'ensemble des histoires et les traite par des algorithmes d'apprentissage automatique, pour générer des agents FAtiMA qui peuvent être utilisés comme des acteurs virtuels dans un drame interactif.

Une caractéristique d'ENIGMA réside dans le fait que les personnages virtuels peuvent fonctionner pendant le processus de création d'une histoire. Ce fonctionnement, « en mode débogage », permet d'augmenter la cohérence du comportement des personnages virtuels. Plus concrètement, l'observation des personnages virtuels peut fournir, aux contributeurs dans une session de création, des suggestions sur des événements suivants. Dans ce mode itératif, l'acceptation ou le refus des comportements obtenus ainsi que la proposition de nouveaux événements de la part des contributeurs (mode d'initiative mixte) influencent l'apprentissage des personnages virtuels (voir Figure II.39). Ce mécanisme est différent des travaux présentés dans les sections précédentes (Scribe, EmoEmma).

Les contributeurs racontent des histoires en créant et arrangeant des événements au moyen de l'interface graphique donnée dans la Figure II.40. Un événement contient un sujet, une action et ses paramètres correspondants. Chaque événement produit est aussi visualisé à travers une

bande dessinée qui est dynamiquement générée. Les contributeurs annotent les histoires qu'ils créent afin que le serveur ENIGMA puisse collecter des informations sémantiques supplémentaires nécessaires à son traitement ultérieur. Par exemple, après avoir fabriqué un nouvel événement, les contributeurs peuvent spécifier la manière dont les émotions des personnages sont changées, ainsi que les propriétés modifiées en raison de cet événement.

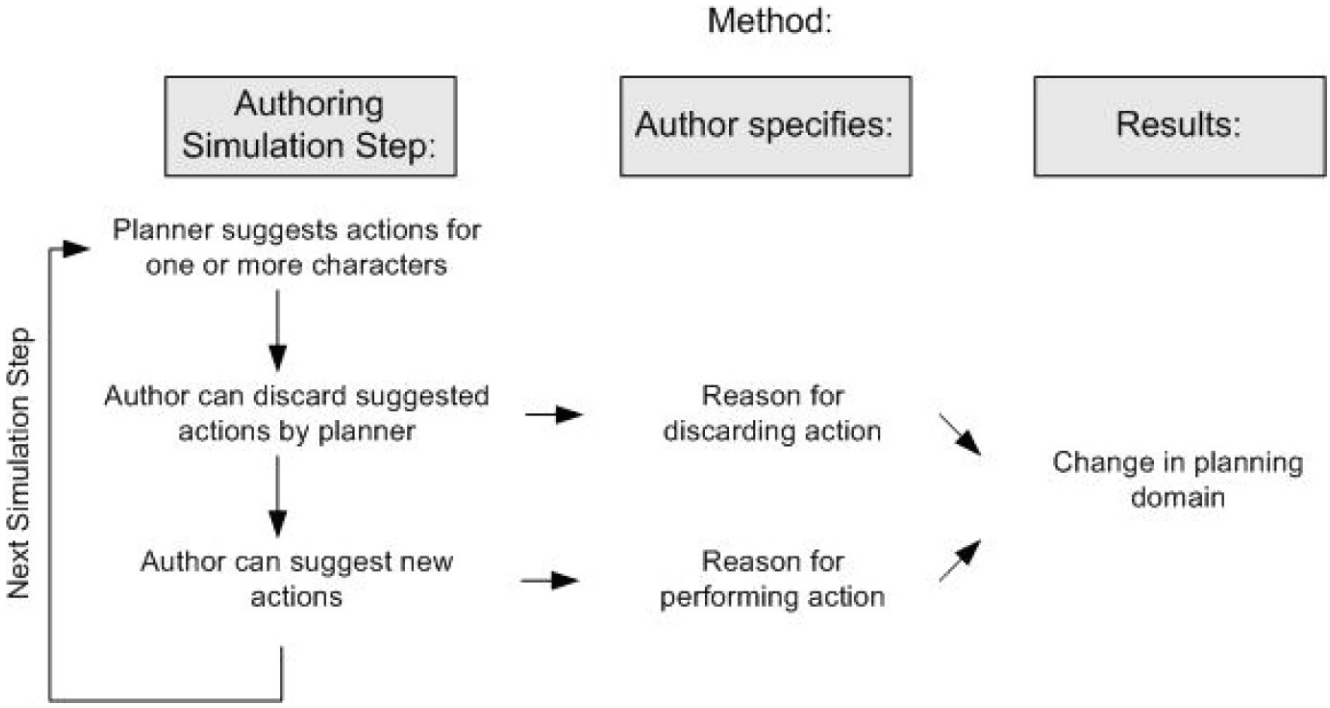


Figure II.39. Processus de création en mode d'initiative mixte par ENIGMA [60].

Figure II.40. Interface graphique principale d'ENIGMA [59].

Pour conclure, grâce à la combinaison entre l'acquisition de connaissances d'une foule de contributeurs, l'apprentissage automatique et le mode d'initiative mixte, ENIGMA permet de construire de façon itérative les comportements autonomes de personnages virtuels intelligents, et ainsi permet de créer des contenus qui peuvent être employés dans des systèmes de narration émergente.

II.3.1.4. Bilan

Nous avons présenté, dans cette section, les principales approches, concernant le problème de production de scénarios valides, qui sont fondées sur les traces d'exécution d'un scénario. Ces travaux permettent à l'utilisateur d'interagir directement avec un discours à chaque « séance de test », pour qu'il puisse voir étape par étape les traces d'exécution du scénario courant. Ainsi le déroulement de l'histoire peut être observé, et les erreurs dans le scénario peuvent être détectées et prévenues. Cette activité permet l'amélioration du scénario. Néanmoins, cette solution permet seulement une validation partielle. Plus concrètement, comme il n'y a pas de tests de couverture, la solution est par conséquent inutilisable dans le cas où le nombre de discours contenus dans le scénario est important (car le nombre de séquences de test est, aussi, proportionnellement important).

Dans la section suivante, nous présentons les travaux s'appuyant sur l'analyse structurelle d'un scénario, ce qui permet de surmonter l'obstacle énoncé ci-dessus.

II.3.2. Analyse structurelle d'un scénario

Les travaux, s'appuyant sur ce type d'analyse, permettent à l'utilisateur d'analyser au niveau structurel tous les discours possibles d'un scénario. Il peut donc vérifier si le scénario courant satisfait à un ensemble de critères/propriétés de narration prédéfini. Dans les sections suivantes, nous présentons tour à tour les principaux travaux fondés sur cette stratégie d'analyse structurelle, et discutons aussi comment ils aident l'utilisateur à produire les scénarios valides.

II.3.2.1. KANAL

KANAL [56, 57, 58] est un outil permettant à l'utilisateur de construire et analyser des plans d'exécution. L'utilisateur peut valider ces plans en étudiants les propriétés données par l'outil. Pour cela, KANAL aide l'utilisateur à spécifier des actions (ou à étendre des actions disponibles à travers le mécanisme héréditaire) d'un plan grâce à un éditeur graphique (voir Figure II.41). Ensuite, il aide l'utilisateur à simuler des plans créés (voir Figure II.42) et à les examiner. Par conséquent, KANAL peut montrer l'aperçu des plans, il est donc possible pour l'utilisateur de les corriger/améliorer plus tard.

Le résultat de l'analyse d'un plan donné par KANAL est assez riche et accessible pour l'utilisateur, il comprend :

- la transition des états à travers les étapes du plan (voir Figure II.43) ;
- des statistiques sur des erreurs/avertissements (voir Tableau II.4) ;
- les informations détaillées concernant les erreurs/avertissements (voir Figure II.44) dans lesquelles, se trouvent aussi les suggestions principales pour la réparation (filtrées selon le contexte courant). Par ailleurs, KANAL permet à l'utilisateur de

donner son feedback sur ces informations (de ce fait, les connaissances dans KANAL sont mises à jour), et il permet aussi à l'utilisateur de voir d'autres violations possibles.

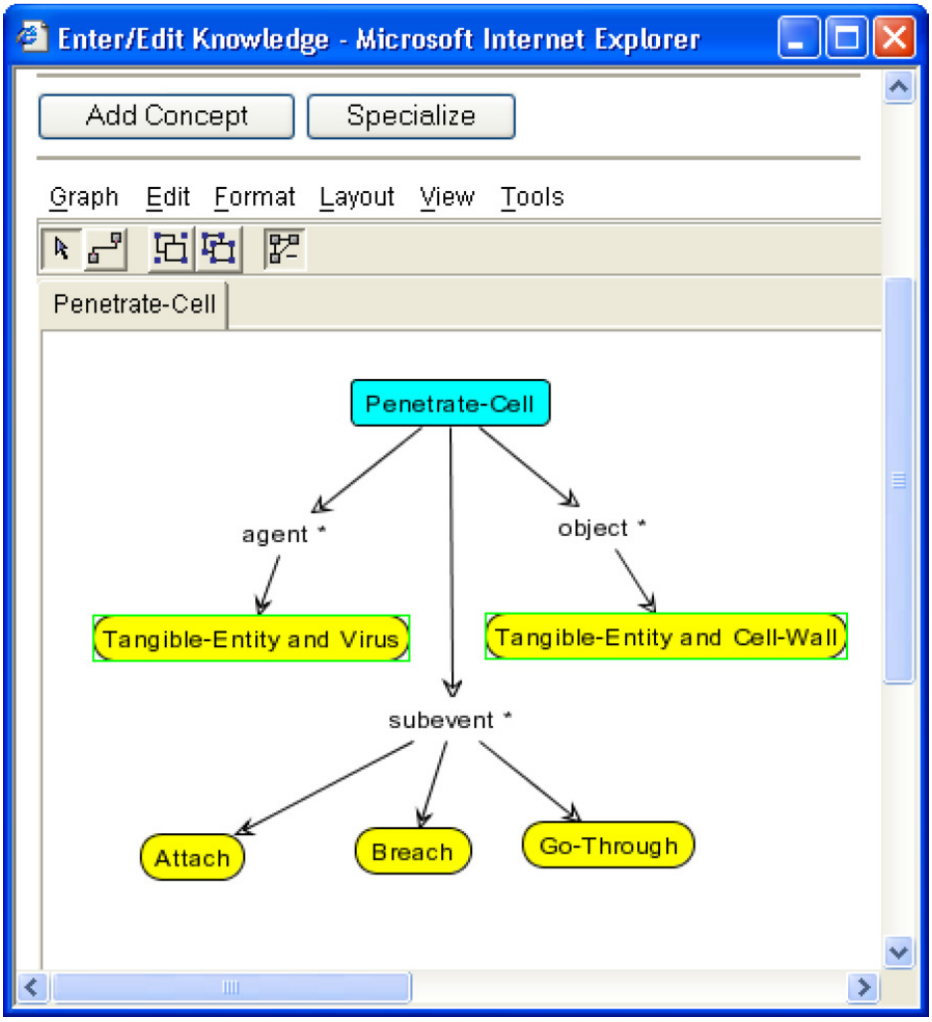


Figure II.41. Éditeur graphique pour spécifier ou étendre des actions dans KANAL [56].

Error/Warning Type	Total	Ratio
Missing first-event, subevent, next-event	37	0.26
Unreached events	55	0.38
Unnecessary ordering	105	0.73
Failed conditions	133	0.92
Failed execution of step	30	0.21
Effectless step	139	0.97
Failed expected effect	7	0.05
Loop	1	0.01

Tableau II.4. Erreurs et avertissements mis en évidence par KANAL [56].

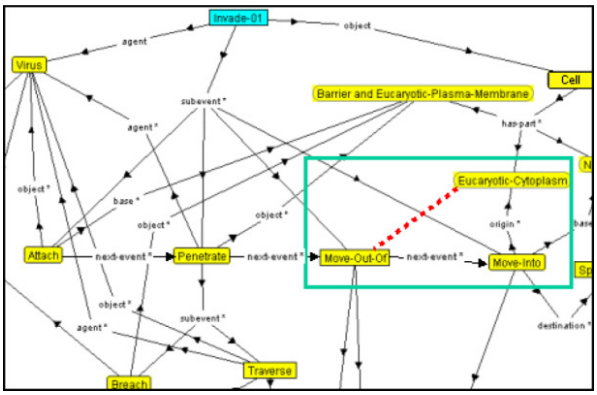
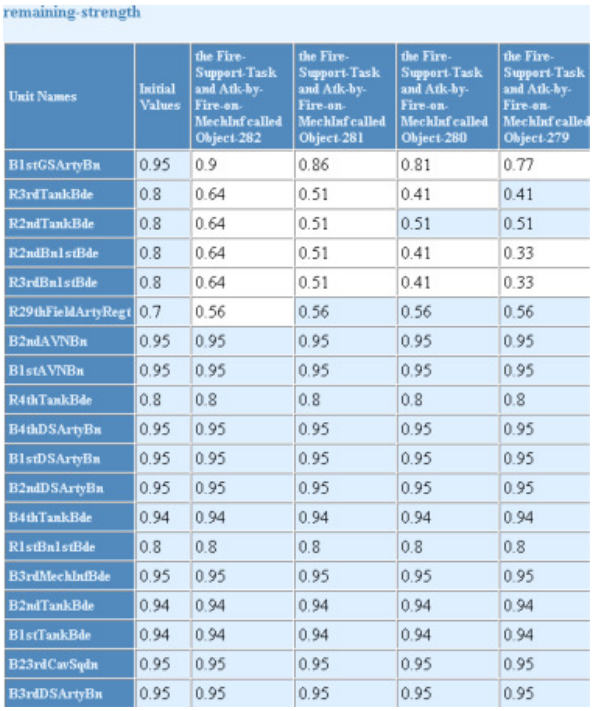


Figure II.42. Un plan construit dans KANAL [56].



remaining-strength

Unit Names	Initial Values	the Fire-Support-Task and Atk-by-Fire-on-MechInf called Object-282	the Fire-Support-Task and Atk-by-Fire-on-MechInf called Object-281	the Fire-Support-Task and Atk-by-Fire-on-MechInf called Object-280	the Fire-Support-Task and Atk-by-Fire-on-MechInf called Object-279
B1stGSArtyBn	0.95	0.9	0.86	0.81	0.77
R3rdTankBde	0.8	0.64	0.51	0.41	0.41
R2ndTankBde	0.8	0.64	0.51	0.51	0.51
R2ndBn1stBde	0.8	0.64	0.51	0.41	0.33
R3rdBn1stBde	0.8	0.64	0.51	0.41	0.33
R29thFieldArtyRegt	0.7	0.56	0.56	0.56	0.56
B2ndAVNBn	0.95	0.95	0.95	0.95	0.95
B1stAVNBn	0.95	0.95	0.95	0.95	0.95
R4thTankBde	0.8	0.8	0.8	0.8	0.8
B4thDSArtyBn	0.95	0.95	0.95	0.95	0.95
B1stDSArtyBn	0.95	0.95	0.95	0.95	0.95
B2ndDSArtyBn	0.95	0.95	0.95	0.95	0.95
B4thTankBde	0.94	0.94	0.94	0.94	0.94
R1stBn1stBde	0.8	0.8	0.8	0.8	0.8
B3rdMechInfBde	0.95	0.95	0.95	0.95	0.95
B2ndTankBde	0.94	0.94	0.94	0.94	0.94
B1stTankBde	0.94	0.94	0.94	0.94	0.94
B23rdCavSqdn	0.95	0.95	0.95	0.95	0.95
B3rdDSArtyBn	0.95	0.95	0.95	0.95	0.95

Figure II.43. La transition des états à travers les étapes d'un plan dans KANAL [56].

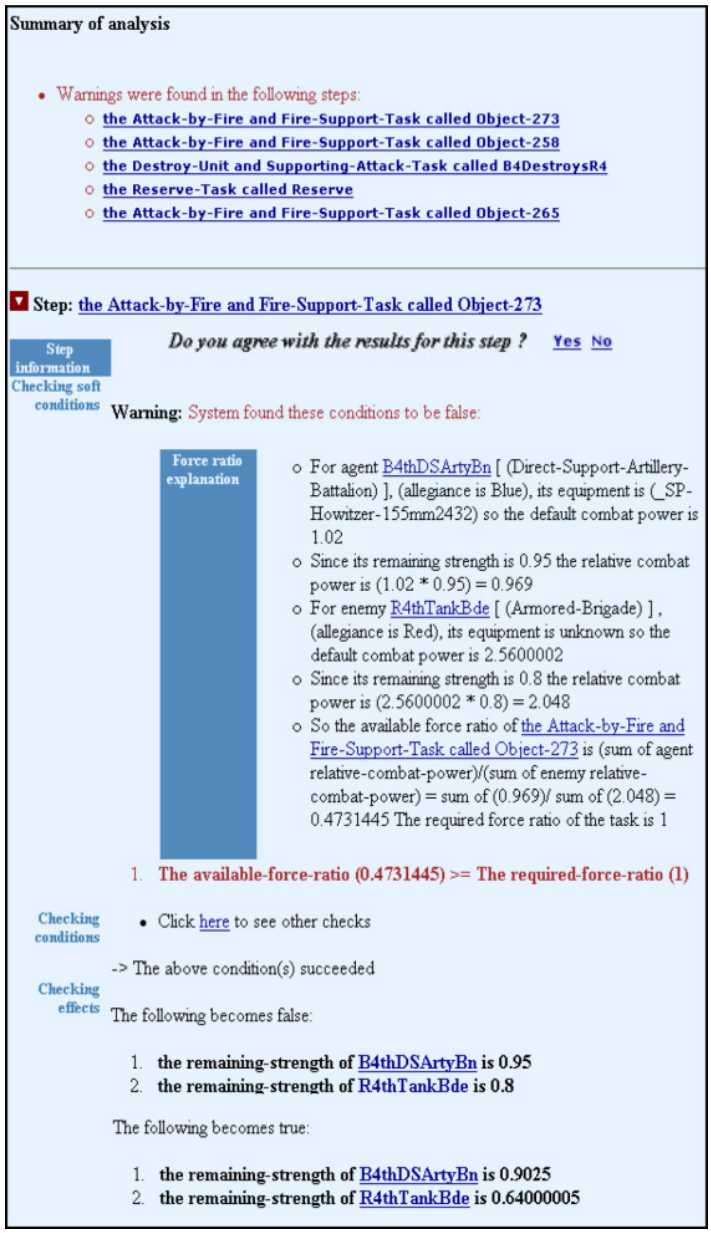


Figure II.44. Les informations détaillées d'un avertissement dans KANAL [56].

Plus concrètement, pour des erreurs/avertissements détectés, KANAL montre les états invalides, les connaissances supplémentaires qui doivent être acquises, et les connaissances existantes qui doivent être modifiées. Ces résultats sont obtenus en analysant des interdépendances (préconditions/effets) entre différents morceaux de connaissances utilisés dans la description du plan (toutes les connaissances dans KANAL sont exprimées par une représentation similaire à STRIPS).

Par conséquent, avec KANAL, l'utilisateur peut construire des plans et les vérifier. Néanmoins, comme sa conception est générale (n'est pas dédiée à un domaine particulier), plusieurs spécifications clés de la narration interactive ne sont pas abordées (telles que le modèle utilisateur, la structure d'histoire, *etc*.).

II.3.2.2. Réseau de Petri

Natkin *et al.* du laboratoire CEDRIC de CNAM dans [74, 97] ont utilisé les réseaux de Petri en vue de spécifier et analyser l'ordonnancement d'actions pour les jeux vidéo. Cette approche peut être considérée comme le point de départ d'une méthode de conception, qui garantit qu'il n'y a pas de blocage dans les jeux créés et que leur déroulement est cohérent. Pour cela, un jeu est divisé en « niveaux » (le joueur ne peut que passer un niveau s'il a passé les niveaux précédents), chacun des niveaux est associé à un sous-objectif de l'objectif général du jeu, et normalement lié à un lieu dans la carte du jeu. Un niveau comporte des épreuves qui sont composées : d'un but, d'obstacles et de solutions possibles. Les actions du joueur sont des transactions atomiques. Les réseaux de Petri sont employés pour modéliser ces transactions (voir Figure II.45) et spécifier les relations entre elles, par exemple :

- il n'y aucune relation entre les transactions (voir Figure II.46) ;
- la transaction B doit être terminée avant que la transaction A commence (voir Figure II.47) ;
- si la transaction B est exécutée, alors l'exécution de la transaction A est impossible (voir Figure II.48) ; *etc*.

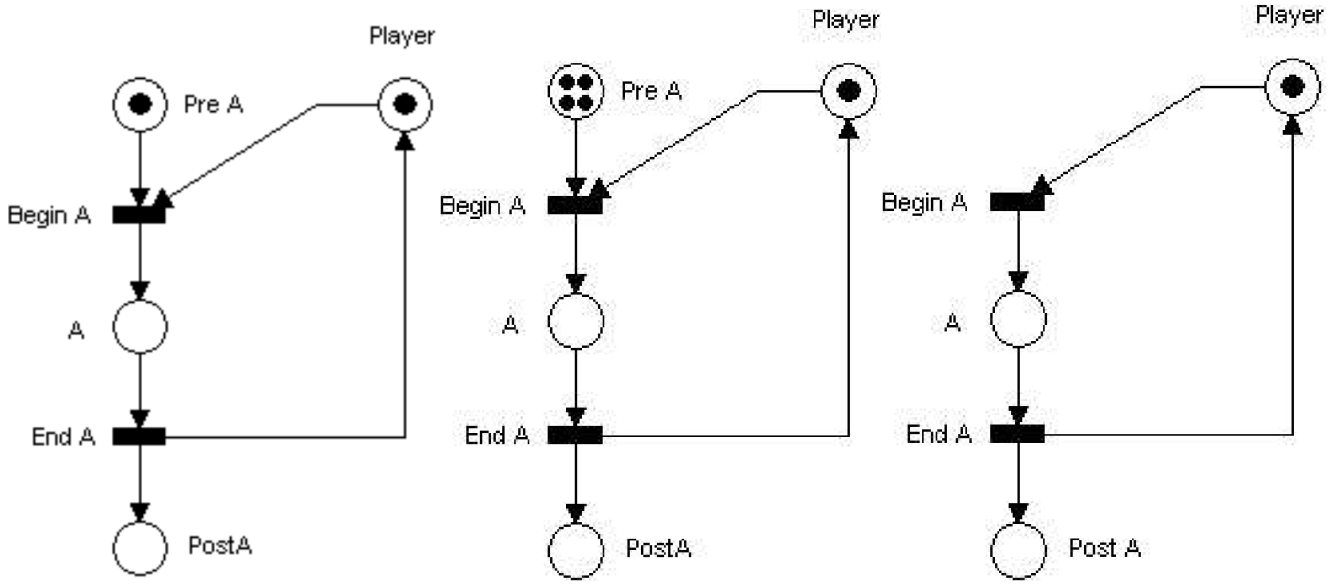


Figure II.45. Modèles de réseau de Petri de la transaction A dont le nombre d'exécution est respectivement de un, de quatre et infini [97].

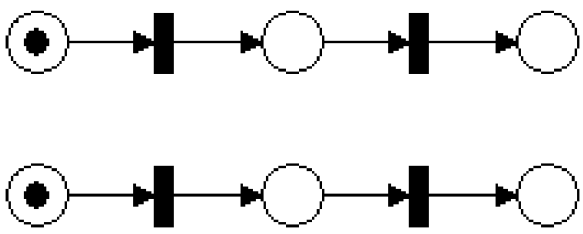


Figure II.46. Il n'y a aucune relation entre les deux transactions [74].

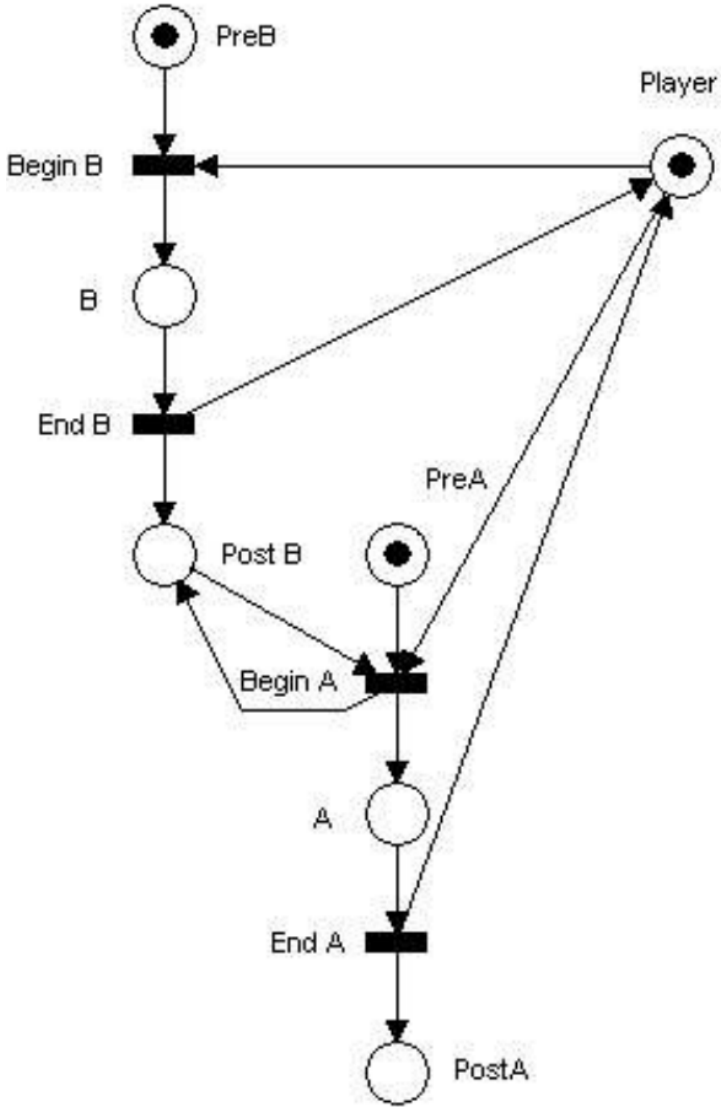


Figure II.47. La transaction B doit être terminée avant que la transaction A commence [97].

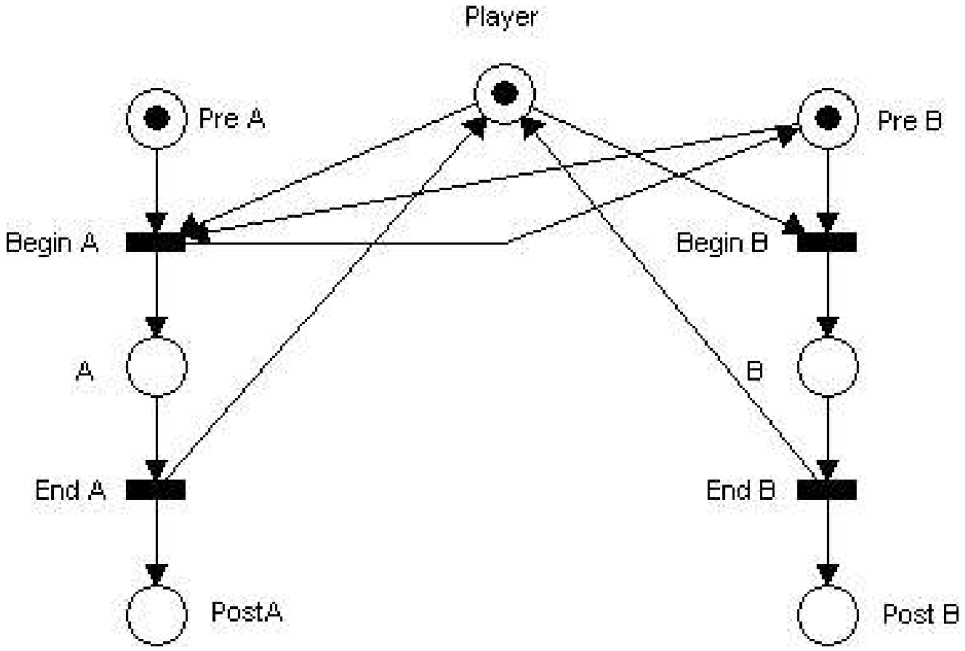


Figure II.48. Si la transaction B est exécutée, alors l'exécution de la transaction A est impossible [97].

Comme une épreuve est un ensemble de transactions ordonnées, elle est aussi modélisée par un réseau de Petri correspondant à ses transactions. Il en est de même pour les niveaux d'un jeu. Par conséquent, grâce à une telle modélisation (un procédé hiérarchique : transactions → épreuves → niveaux), les concepteurs peuvent, en appliquant les propriétés mathématiques disponibles des réseaux de Petri, analyser le problème de blocage du scénario produit d'un jeu (comprenant tous les chemins possibles), et ainsi ils peuvent le valider.

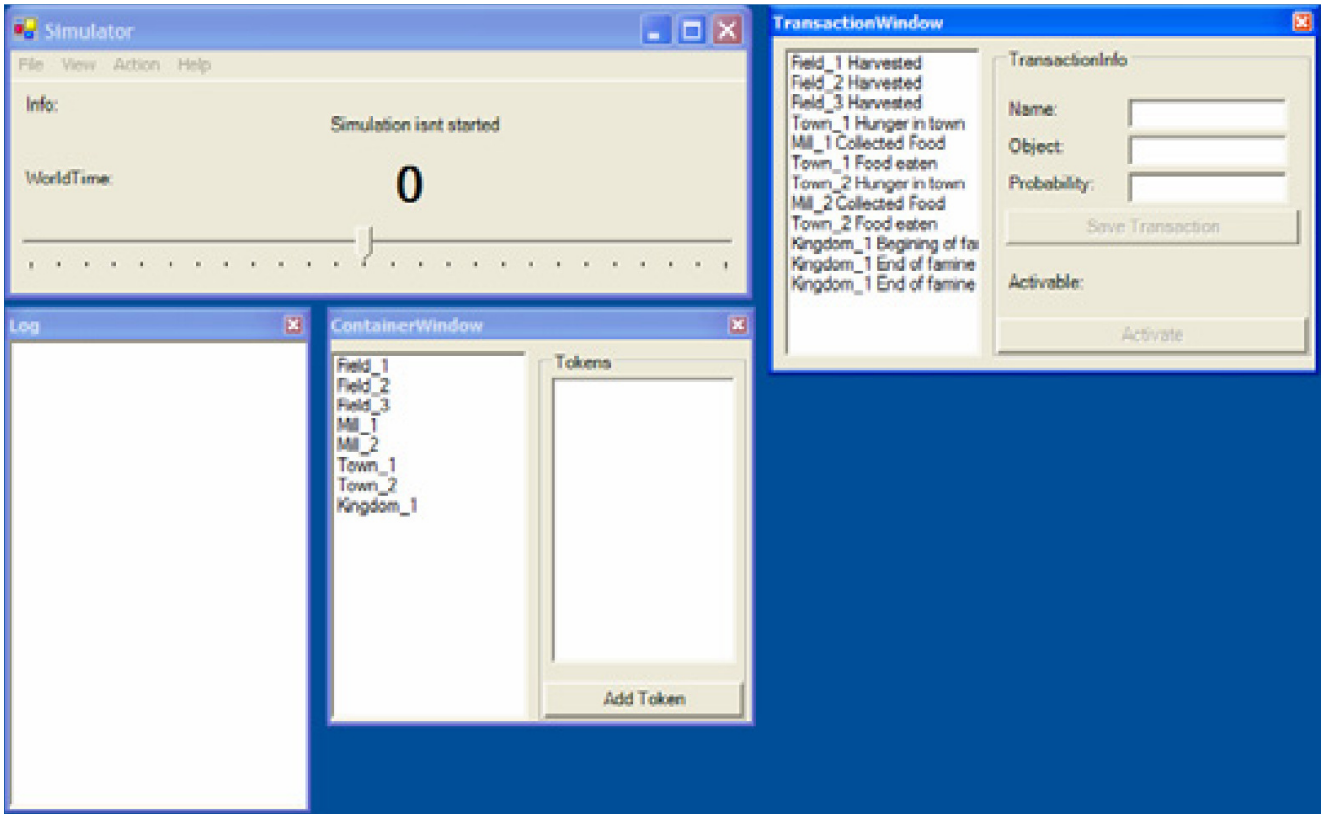


Figure II.49. Application de test [14].

Les articles [8, 15] décrivent l'emploi des réseaux de Petri afin de représenter et dérouler des intrigues non-linéaires dans des jeux concernant des mondes virtuels associés à des environnements de grande taille (un village, une région, *etc*.). Leur système a été mis en œuvre pour le jeu sérieux *Karo* que nous avons présenté en Section II.2.2.4. Nous ne rappelons pas ce que nous avons abordé, mais nous discutons ci-dessous, l'intérêt du réseau de Petri dans la vérification des scénarios produits pour ce jeu [14]. Une application de test a été développée (voir Figure II.49), son objectif est d'examiner le déroulement de l'histoire d'une manière abstraite. La démarche est la suivante : un « brouillon » du scénario est modélisé par un réseau de Petri et introduit dans l'application de test au format XML (plusieurs éléments ont été enlevés par rapport au modèle qui peut être l'entrée du *Story manager* – voir Figure II.30, Section II.2.2.4). L'application de test, grâce à la nature mathématique du réseau de Petri, peut montrer les situations inaccessibles de l'histoire. Elle permet aussi à l'utilisateur d'exécuter la simulation du scénario pendant laquelle, des événements abstraits sont démarrés selon le « brouillon », et ainsi l'utilisateur, en interagissant avec l'application de test en temps réel, peut voir ce qui se passe dans

l'environnement virtuel. Cette approche autorise l'utilisateur à corriger, si nécessaire, des défauts. Après l'exécution du processus de test, le « brouillon » validé est transformé en scénario complet final. Ce dernier est modélisé par un réseau de Petri et constitue l'entrée du *Story manager*. Ainsi, cette application de test autorise les concepteurs à vérifier la validité des intrigues et l'impact de l'intervention du joueur sur le déroulement de l'histoire.

II.3.2.3. Logique linéaire

Dans le cadre de notre laboratoire – L3I, Collé *et al.* ont proposé une méthode d'analyse d'un scénario de jeu utilisant la logique linéaire [29]. L'approche définit pour chaque scénario une liste de propriétés de narration que les discours dans le scénario doivent respecter (ces propriétés de narration ont été complétées par Prigent *et al.* [81] plus tard), telles que :

- Accessibilité (*Reachability*) : indique qu'un état quelconque du système peut être atteint ;
- Absence de blocage (*No deadlock*) : exprime que le système ne se trouve jamais dans une situation où il ne peut plus progresser ;
- Equité (*Fairness*) : tous les joueurs ont des possibilités équivalentes de remporter la partie en regard de la disposition des ressources sur le terrain ;
- Complexité (*Complexity*) : le déroulement du jeu ne contient pas d'exécutions qui mènent à la victoire trop rapidement ; *etc.*

Ensuite, les auteurs modélisent le scénario grâce au fragment multiplicatif de la logique linéaire, et puis le modèle obtenu est traduit en un réseau de Petri. Ce réseau de Petri permet de générer et parcourir tous les discours possibles du scénario modélisé, donc de les analyser selon les propriétés de narration énoncées ci-dessus, et ainsi, les auteurs peuvent valider la qualité du scénario courant.

Des travaux utilisant la logique linéaire pour valider des scénarios commencent, également, à être menés par Bosser *et al.* qui ont proposé une méthode d'analyse de la structure d'une narration en utilisant la logique linéaire [12, 13]. Pour cela, ils modélisent les ressources (comportant les conditions initiales, les actions, *etc*.) et les contraintes de terminaison d'une histoire grâce à un séquent de logique linéaire intuitionniste. Ensuite, ils emploient l'outil de preuve Coq afin de prouver ce séquent, et peuvent donc analyser le scénario modélisé et produire une analyse structurelle de tous les discours possibles dans le scénario. Par conséquent, il est possible de savoir si le séquent est prouvé, si une certaine preuve du séquent atteint une certaine sortie, si toutes les preuves du séquent sont réussies et si elles atteignent une sortie quelconque. Ainsi les auteurs peuvent évaluer la qualité du scénario courant selon ces aspects.

II.3.2.4. Bilan

Nous avons présenté, dans cette section, les principales approches, concernant le problème de production de scénarios valides, qui s'appuyent sur l'analyse structurelle d'un scénario. Ces travaux permettent à l'utilisateur d'analyser au niveau structurel tous les discours possibles d'un scénario. Il peut donc vérifier si le scénario courant satisfait à un ensemble de critères/propriétés de narration prédéfini. Par conséquent, il est possible pour l'utilisateur d'évaluer la qualité du scénario suivant ces aspects, et ainsi, il peut valider le scénario.

Néanmoins, les résultats obtenus dans les travaux mentionnés plus haut ne sont pas suffisants pour produire un scénario de bonne qualité selon nos exigences dans le cadre de cette thèse (nous décrivons en détail ces exigences dans le chapitre III). Ainsi, même si l'idée sur l'analyse structurelle d'un scénario est retenue dans notre approche, nous proposons, dans ce mémoire, d'autres solutions pour le problème de validation d'un scénario. Nous donnons ci-dessous un bilan général sur tous les travaux, concernant le problème de production de scénarios valides, que nous avons présentés dans ce chapitre.

II.3.3. Bilan général

Dans cette section, nous avons mentionné les principales approches s'intéressant au problème de production de scénarios valides. Nous les avons classifiés selon la manière d'exécuter l'analyse pour le scénario à valider :

- ***Approche fondée sur les traces d'exécution d'un scénario :*** Les travaux, s'appuyant sur cette méthode d'analyse, permettent à l'utilisateur d'interagir directement avec un discours à chaque « séance de test », pour qu'il puisse suivre étape par étape les traces d'exécution d'un scénario. Ainsi, il peut vérifier le déroulement de l'histoire et en améliorer le contenu. Néanmoins, cette solution permet seulement une validation partielle. Plus concrètement, comme il n'y a pas de tests de couverture, la solution est par conséquent d'autant plus partielle que le nombre de discours contenus dans le scénario est important.
- ***Analyse structurelle d'un scénario :*** Les travaux, s'appuyant sur cette méthode d'analyse, permettent de surmonter l'obstacle énoncé ci-dessus, en autorisant l'utilisateur à analyser au niveau structurel tous les discours possibles du scénario. L'utilisateur peut donc déterminer si le scénario courant satisfait à un ensemble de critères/propriétés de narration prédéfini. Par conséquent, il est possible pour l'utilisateur d'évaluer la qualité du scénario suivant ces aspects, et ainsi, il peut valider le scénario. Néanmoins, les résultats obtenus suite à ces travaux ne sont pas suffisants pour produire un scénario de bonne qualité selon nos exigences dans le cadre de cette thèse.

Les contributions, apportées par les travaux concernant le problème de production de scénarios valides cités plus haut et apportées par d'autres outils de création de contenus narratifs (tels que DraMachina [35], Art-E-Fact [53], Scenejo [98], StoryTec [45]), nous ont aussi permis d'identifier les principes clés, comme la stratégie de validation (surtout l'idée sur l'analyse structurelle d'un scénario), les propriétés de narration pouvant être mises en évidence, les informations qualitatives et statistiques nécessaires, l'idée sur la suggestion de causes des erreurs, sur la réutilisation des éléments de modélisation, sur la manière de représenter graphiquement un ensemble de discours et sur la modélisation d'une histoire grâce à la manipulation sous forme « glisser/déposer » d'éléments graphiques, pour la construction de notre solution afin de résoudre le problème de production et de validation d'un scénario interactif. Dans la section suivante, nous donnons le bilan général concernant l'état de l'art que nous avons abordé dans le cadre de ce chapitre.

II.4. Conclusion

L'analyse des travaux présentés dans ce chapitre nous permet de définir notre approche basée sur un modèle formel : la logique linéaire. Ce modèle est le socle que nous utilisons pour la production d'un système de pilotage de narration interactive, qui est capable de diriger le déroulement d'un jeu vidéo de sorte que le déroulement de ce jeu respecte les effets souhaités par les auteurs, et en même temps, soit influencé par les actions du joueur.

Tout d'abord, nous avons fait un tour d'horizon des principales approches concernant le pilotage de narration interactive. Nous avons présenté les travaux existants dont le but est de résoudre le problème du contrôle du déroulement d'une histoire de manière à ce que les utilisateurs puissent influer sur celui-ci. Ces travaux, qui contribuent à répondre au problème selon des approches variées, nous ont aussi permis d'identifier des facteurs important, comme les éléments d'architecture d'un système de pilotage ; la construction, la représentation, l'exécution de scénarios narratifs ; les propriétés de narration importantes ; l'évolution de référence des paramètres dramatiques ; et la structuration de discours, pour la réalisation de notre système de pilotage de narration interactive.

Ensuite, nous avons décrit les travaux existants concernant le problème de production de scénarios valides. Cette question est centrale pour garantir la qualité d'une narration interactive. Elle constitue l'objectif principal de cette thèse. Ce processus de validation est exécuté dans la phase de conception des scénarios et répété jusqu'à ce que les utilisateurs reçoivent un scénario qui répond à leurs objectifs. Les contributions, apportées par les travaux cités et apportées par d'autres outils de création de contenus narratifs (tels que DraMachina [35], Art-E-Fact [53], Scenejo [98], StoryTec [45]), nous ont permis d'identifier les principes clés sur lesquels s'appuie la construction de notre solution afin de produire et de valider un scénario interactif.

Ainsi, à partir des travaux et analyses présentés ci-dessus, même s'ils ne répondent pas complètement à nos objectifs (notamment sur l'aspect de validation de scénario), plusieurs concepts et idées ont été retenus, ceci nous permet de caractériser notre approche pour la production d'un système de pilotage de narration interactive, en particulier :

- de façon similaire aux architectures des systèmes de pilotage de narration interactive abordés en Section II.2 (Mimesis [102], Façade [67], Interactive Drama Architecture [64], *etc.*), notre système de pilotage doit dissocier les composants chargés du jeu (environnement virtuel) de ceux chargés du pilotage, afin que le système se décharge de tous les aspects graphiques et s'adapte à un large ensemble de supports différents ;
- les scénarios produits doivent respecter un ensemble de propriétés de narration en vue de garantir leur qualité ;
- l'utilisation de la logique linéaire afin de modéliser une histoire, puis s'assurer de la qualité du scénario modélisé, en parcourant l'ensemble des discours possibles et en analysant la satisfaction de ces discours au regard des propriétés de narration, grâce aux mécanismes de déduction rigoureux et automatique de la logique linéaire.

Ces concepts et idées sont améliorés et mis en œuvre de manière adaptative dans notre approche basée sur la logique linéaire, pour la production d'un système de pilotage de

narration interactive, qui dirige des jeux vidéo de sorte que le déroulement de ces jeux satisfasse les effets souhaités par leurs auteurs, et en même temps, soit influencé par les actions du joueur.

Pour cela, notre objectif dans le cadre de cette thèse est de proposer une solution consistant à : (1) permettre aux utilisateurs de produire un modèle de scénario de bonne qualité qui est riche, valide et opérationnel en définissant un système auteur, (2) déterminer l'utilisation de ce système auteur dans le cadre d'un processus de production d'un système de pilotage de narration interactive pour le jeu permettant de le diriger selon le scénario produit. Les chapitres suivants présentent ces solutions.

CHAPITRE III - Validation d'un Scénario

III.1. Introduction

La validation d'un jeu assure aux auteurs la conformité du scénario à des propriétés liées à sa qualité en tenant compte des interactions. Notre objectif principal dans cette thèse concerne la validation de scénarios de jeu. Dans notre approche, ce processus est effectué lors de la phase de conception de jeu, et non pas lors de son exécution (où le respect au scénario validé est assuré par le système de pilotage). Plus concrètement, notre but est de créer un scénario de bonne qualité, ce scénario est ensuite utilisé comme entrée du système de pilotage du jeu. Ainsi le système de pilotage dirige le déroulement du jeu selon le scénario valide produit.

Qu'est-ce qu'un scénario de bonne qualité ? Comment évaluer la qualité d'un scénario ? Ces questions sont totalement ouvertes et complexes, car elles dépendent fortement de facteurs personnels et culturels. Dans le cadre de cette thèse, nous considérons la qualité d'un scénario selon seulement deux aspects : (1) le déroulement des discours possibles dans le scénario est-il correct (c'est-à-dire si leur déroulement satisfait à des contraintes définies par les auteurs telles que l'histoire se termine-t-elle dans les conditions prévues ?, n'y-a-t-il aucune situation inaccessible ?, *etc*.) ? ; et (2) l'effet dramatique (ou les émotions) créé(es) par chacun des discours pour le joueur est-il (sont-elles) pertinent(es) ?, autrement dit, un scénario est considéré comme « de bonne qualité » (ou un scénario est considéré comme « valide ») s'il satisfait ces deux aspects.

Les questions ci-dessus sont précisées par un ensemble de propriétés de narration caractérisant un scénario. Nous les classons en deux catégories : les propriétés liées au déroulement des discours et les propriétés liées à l'effet dramatique créé par chacun des discours. Par conséquent, dans notre approche, en vue d'évaluer la qualité d'un scénario, il suffit que les utilisateurs vérifient si le scénario satisfait ces propriétés.

Pour cela, notre démarche de vérification comporte deux étapes : (1) modéliser des aspects importants d'une histoire tels que les états de l'histoire et les sorties souhaitées, les états et les choix des personnages joueur/non-joueur, les événements/actions et leurs effets dramatiques, *etc.* ; (2) analyser (dans ce travail en les parcourant) l'ensemble des discours possibles du scénario obtenu après le processus de modélisation, et s'assurer, pour les propriétés de narration abordées ci-dessus, leur satisfaction. Ainsi, on peut évaluer la qualité du scénario courant et donc le valider. Dans notre approche, le scénario qui est validé est celui qui est exécuté. C'est-à-dire qu'il constitue l'entrée d'un système de pilotage. La représentation du scénario autorise l'application d'un mécanisme de déduction automatique. De manière opérationnelle, le scénario entré aide le système de pilotage à infléchir le déroulement du jeu.

Les travaux existants concernant la logique linéaire, qui ont été présentés dans le chapitre II ([12, 13, 29, 81]), montrent que son utilisation répond aux exigences de modélisation d'une histoire et de la vérification des propriétés au moyen des preuves. Sa capacité de déduction

automatique offre le caractère opérationnel souhaité. En effet, la logique linéaire (pour plus d'information, voir Annexe A) est un modèle formel exécutable. Elle est bien adaptée à la modélisation des ressources, à la traduction de la causalité, au raisonnement automatique, à l'expression des mécanismes de consommation/production de ressources et l'énoncé des conclusions désirées dans le cadre d'une histoire. Une preuve d'un séquent indique comment sont consommées les formules d'implication linéaire afin d'atteindre une de ses conclusions. Une façon d'écrire la preuve d'un séquent correspond à un discours. C'est une suite ordonnée d'événements/actions, et l'ensemble de toutes les façons d'écrire la preuve d'un séquent correspondent à l'ensemble de tous les discours qui peuvent être générés. Par conséquent, il est possible de considérer un séquent de logique linéaire comme une représentation d'un scénario d'une histoire, et une telle représentation est évidemment opérationnelle (grâce au mécanisme de déduction automatique de séquents). Le calcul de la preuve d'un séquent donne les informations importantes sur la longueur des discours, le processus d'évolution des discours, le succès (ou l'échec) des discours ainsi que le nombre de discours possibles. Ces informations sont des éléments d'analyse qui caractérisent, pour les propriétés de narration, la conformité du scénario. Un autre point fort de la logique linéaire concerne sa capacité d'expression et particulièrement celle qui concerne la dualité action du système/action de l'utilisateur. Les connecteurs & et ⊕ peuvent être utilisés afin d'exprimer (et de distinguer) clairement les choix des personnages joueur et non-joueur (qui sont dirigés par le système de pilotage). Ainsi, les raisons citées ci-dessus montrent que la logique linéaire offre une bonne solution afin d'atteindre l'objectif posé dans le cadre de cette thèse.

En effet, grâce à la modélisation d'une histoire en logique linéaire, et ensuite la réalisation de la preuve du séquent obtenu (qui permet d'analyser/évaluer en détail la qualité du scénario correspondant), on peut produire un scénario qui est : (1) riche – le scénario fournit suffisamment d'options pertinentes aux personnages joueur/non-joueur de sorte que le joueur puisse déterminer le déroulement de l'histoire et sente toujours que le discours créé est intéressant ; (2) valide – tous les discours possibles dans le scénario sont cohérents et satisfont les intentions des auteurs ; (3) opérationnel – la représentation du scénario est exécutable, on peut donc développer un système de pilotage pour les jeux vidéo basé sur la narration interactive, ce qui garantit que le déroulement des jeux est géré par le mécanisme de déduction automatique du séquent correspondant.

L'objectif de ce chapitre est de résoudre le problème de la validation des scénarios de jeu. Pour cela, nous donnons et expliquons en détail, au début du chapitre, l'ensemble de propriétés de narration spécifiant la qualité d'un scénario selon les deux aspects abordés précédemment. Ensuite, nous décrivons notre solution, employant la logique linéaire afin d'examiner la satisfaction de ces propriétés pour un scénario. La démarche liée à la solution proposée comporte deux étapes : (1) modéliser une histoire en logique linéaire, et (2) évaluer la qualité du scénario modélisé en parcourant tous ses discours possibles et en analysant leur satisfaction pour les propriétés de narration. L'exécution de la seconde étape est divisée en deux sections correspondant aux scénarios avec les événements/actions simples ou complexes en raison de leur mécanisme de déduction différent.

III.2. Propriétés de narration d'un scénario pouvant être validées

Comme présenté auparavant, cet ensemble de propriétés de narration (qui sont héritées et développées à partir des travaux précédents de notre laboratoire L3I [81]) est classé en deux catégories répondant aux deux aspects à vérifier : d'une part, celles qui concernent le déroulement des discours qui ont pour but de s'assurer que les discours se déroulent correctement ; et, d'autre part, celles qui concernent l'effet dramatique créé par chaque discours qui ont pour but de s'assurer de l'intérêt du scénario (l'histoire n'est-elle pas monotone ?, la répartition des émotions attribuées au joueur est-elle raisonnable ?, *etc*.).

Propriétés liées au déroulement des discours

- **Accessibilité :** Garantir que toutes les situations d'une histoire peuvent être atteintes à partir de la situation initiale.
- **Absence de blocage :** Les discours ne se trouvent jamais dans des situations où ils ne peuvent plus progresser.
- **Complexité :** Cette propriété comporte deux aspects : (1) le nombre de discours possibles dans le scénario n'est pas trop petit ; (2) le déroulement de l'histoire ne contient aucune exécution qui mène à une des terminaisons souhaitées trop rapidement, autrement dit, l'histoire ne peut pas se terminer tant qu'un nombre spécifié d'événements/actions n'a pas eu lieu. Par exemple, un scénario d'un jeu est considéré comme suffisamment compliqué s'il comporte au moins 30 discours dont chacun possède au moins 10 événements/actions.
- **Satisfaction d'événements clés :** Il faut s'assurer que tous les discours possibles dans le scénario validé, à partir d'une histoire donnée, comprennent toujours tous les événements d'un ensemble prédéfini (leur ordre d'apparition n'est pas important). Par exemple, soit un jeu où le joueur doit gagner 4 épreuves principales, les 4 événements correspondant à ces résultats sont EA02, EA06, EA11 et EA21. Un scénario satisfait cette propriété si tous ses discours possibles contiennent tous les événements ci-dessus.
- **Satisfaction de la structure :** La structure de discours d'une histoire comprend plusieurs étapes (cruciales) par lesquelles tous ses discours possibles doivent passer. Ainsi il faut s'assurer que tous les discours possibles dans le scénario validé, à partir d'une histoire donnée, franchissent toutes les étapes définies dans sa structure de discours. Par exemple, la structure de discours de l'histoire « Le Petit Chaperon rouge » comprend les quatre étapes suivantes :
    - La mère demande au Petit Chaperon rouge d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Le Petit Chaperon rouge rencontre le loup dans le bois
    - La grand-mère et le Petit Chaperon rouge sont mangés par le loup
    - Le loup est mort, la grand-mère et le Petit Chaperon rouge sont vivants.

  Par conséquent, un scénario satisfait cette structure si tous ses discours possibles passent toujours toutes les étapes ci-dessus.

- **Séquencement :** Les contraintes de précédence entre les événements dans une histoire doivent être respectées. Par exemple dans un jeu, le joueur doit d'abord rencontrer un sorcier afin que celui-ci lui indique une quête à poursuivre. En suivant les instructions du sorcier, le joueur trouve et combat un dragon, puis retourne voir le sorcier et reçoit, en récompense, une clé permettant au joueur de poursuivre dans le jeu. En l'absence de cet ordre entre des événements, si le joueur tue le dragon avant d'avoir rencontré le sorcier, il ne recevra pas la récompense du sorcier et ne pourra pas poursuivre son exploration du jeu. Il faut garantir que de telles situations ne se produisent jamais.
- **Absence de boucles infinies :** Le scénario ne comporte pas de boucles dont on ne peut sortir.

Propriétés liées à l'effet dramatique créé par un discours : Avant de présenter les propriétés concernant l'effet dramatique créé par un discours dans un scénario, nous définissons les notions « événement dramatique » et « événement neutre ». Les événements dramatiques sont des événements, possédant une qualification particulière dans la liste d'événements/actions d'une histoire, et qui provoquent une modification de l'état émotionnel du joueur. Par exemple, l'événement EA10 exprimant que le joueur gagne une épreuve difficile provoquera une fierté ou un bonheur pour lui, tandis que l'événement EA15 exprimant que le joueur est sur le point de lutter contre un ennemi très fort provoquera pour lui un stress ou une peur (on peut donc lier EA10 à la fierté ou au bonheur, EA15 au stress ou à la peur). Au contraire, les événements appartenant à la liste d'événements/actions d'une histoire qui ne causent aucune émotion pour le joueur sont appelés « les événements neutres ». Nous donnons ci-dessous les propriétés liées à l'effet dramatique créé par un discours dans un scénario.

- **Fréquence des événements dramatiques :** Garantir que la fréquence d'occurrence d'événements dramatiques pendant le déroulement de chacun des discours d'un scénario est suffisamment élevée. Cette contrainte caractérise pour l'auteur une bonne immersion du joueur. Par exemple, un discours est considéré comme satisfaisant cette propriété s'il possède au moins 5 événements dramatiques (c'est-à-dire 5 événements qui provoquent une modification de l'état émotionnel du joueur).
- **Variété d'effets dramatiques :** Les événements dramatiques peuvent être organisés en catégories selon les émotions auxquelles ils sont associés : on trouve des événements liés à la peur, au stress, à la colère, à la tristesse, à la fierté, au bonheur, *etc*. Ainsi, il est important de diversifier la catégorie de ces événements au cours du discours pour mélanger les sentiments ressentis par le joueur. Par exemple, un discours est considéré comme satisfaisant cette propriété s'il provoque au moins 3 types d'émotion pour le joueur (c'est-à-dire qu'il possède au moins 3 événements dramatiques qui appartiennent à 3 catégories différentes).
- **Équilibre de l'effet dramatique :** Exprime une répartition raisonnable entre des situations générant une forte intensité dramatique et des situations calmes. Par exemple, considérons un discours possédant 10 événements dramatiques qui appartiennent à 4 catégories différentes : la peur, le stress, la colère, le bonheur. L'équilibre que recheche l'auteur est réalisé en entremêlant ces événements

dramatiques et des événements neutres. Le choix peut être effectué de sorte que : (1) il n'existe aucune longue période où le joueur se sent seulement tendu (ou le joueur se sent seulement heureux) ; et (2) il n'existe aucune longue période qui ne provoque aucune modification de l'état émotionnel du joueur.

- **Correspondance d'émotion du joueur :** Il s'agit de s'assurer que le joueur perçoit correctement les émotions attachées aux événements dramatiques. Par exemple, si l'événement où le loup mange le Petit Chaperon rouge est relié à la peur, alors nous devons garantir que le joueur ressent cette peur quand cet événement se produit.
- **Évolution de la tension dramatique :** L'évolution typique de la tension dramatique d'un discours « canonique » pour le joueur, causée par ses événements, est donnée dans la Figure II.16 – chapitre II. Elle suit une progression en dents de scie [30] : *« une alternance de montées et descentes, dessinant une élévation progressive à travers des pics locaux. La tension part ainsi d'une valeur initiale basse, progresse jusqu'à un premier pic local, redescend légèrement, puis repart à la hausse, vers un second pic local plus élevé que le premier, et alterne ainsi hausses et baisses jusqu'à atteindre en fin de discours un paroxysme final, suivi d'un épilogue ramenant la tension à la baisse »*. Par conséquent, tous les discours possibles dans le scénario doivent causer, pour le joueur, une telle évolution de la tension dramatique.

Les sections suivantes expliquent en détail la solution utilisant la logique linéaire afin de produire un scénario de bonne qualité d'une histoire. La démarche de la solution proposée comporte deux étapes : (1) modéliser l'histoire en logique linéaire, et (2) évaluer la qualité du scénario modélisé en parcourant tous ses discours possibles et en analysant leur satisfaction pour la presque totalité des propriétés de narration ci-dessus (sauf les trois propriétés « Absence de boucles infinies », « Correspondance d'émotion du joueur » et « Évolution de la tension dramatique »). L'exécution de la seconde étape est divisée en deux sections correspondant aux scénarios avec les événements/actions simples ou complexes en raison de mécanismes de déduction différents. La propriété « Absence de boucles infinies » est toujours garantie grâce au mécanisme des événements/actions utilisés dans notre approche. Cet aspect est présenté dans la section suivante. La conformité du scénario aux deux propriétés « Correspondance d'émotion du joueur » et « Évolution de la tension dramatique » fait appel à d'autres techniques comme l'utilisation de sondages sur des populations test pour mesurer la transformation émotionnelle réelle des joueurs lors de l'exécution.

III.3. Modélisation d'une histoire en logique linéaire

Nous présentons, dans cette section, la première étape de l'approche proposée dans le cadre de la thèse pour la validation d'un scénario : modéliser une histoire en logique linéaire. Ensuite, nous expliquons aussi en détail nos améliorations par rapport à la méthode de modélisation utilisée suite aux travaux précédents menés au L3I.

III.3.1. Modèle d'histoire en logique linéaire

La méthode de modélisation d'une histoire (qui a été étendue à partir des travaux précédents menés au L3I [29, 81] – voir chapitre II), exprime les aspects importants de l'histoire (tels que ses états et ses sorties souhaitées, les états et les choix des personnages joueur/non-joueur, les événements/actions…).

Notre modèle d'histoire est défini en employant les éléments fournis par la logique linéaire. Il est un triplet composé d'une liste d'atomes, d'une liste de formules et d'un séquent :

Histoire = <A, F, S>

- **A :** C'est la liste d'atomes qui représente les états de l'histoire, les états des personnages joueur/non-joueur, et les entrées du joueur.
- **F :** C'est la liste de formules composées des atomes dans la liste d'atomes et des connecteurs (⊕, &, ⊸, ⊗) de la logique linéaire. Ces formules représentent les composants suivants de l'histoire :
    - les événements/actions : exprimés par les formules ⊸ ;
    - les choix : si un choix est décidé par le système de pilotage (le joueur), alors ce choix est exprimé par une formule ⊕ (&) ;
    - les sorties :
        - si le scénario ne comporte qu'une sortie et cette sortie ne comporte qu'un état, alors cette sortie est exprimée par l'atome correspondant à cet état ;
        - si le scénario ne comporte qu'une sortie et cette sortie comporte plusieurs états, alors cette sortie est exprimée par une formule ⊗ dont les composants sont les atomes correspondant à ces états ;
        - si le scénario comporte plusieurs sorties, alors cette liste des sorties est exprimée par une formule ⊕ dont les composants sont chacune des sorties.
- **S :** C'est le séquent qui représente le scénario obtenu après le processus de modélisation de l'histoire. Ce séquent est composé des listes d'atomes et de formules mentionnées ci-dessus. Le concept de séquent employé dans le modèle est précisé dans le chapitre IV de cette thèse (voir Figure IV.3).

Notre modèle d'histoire est utilisé d'une part dans la phase de validation et d'autre part dans la phase de fonctionnement où il est exécuté par le système de pilotage. Le processus du déroulement du scénario, réalisé dans ces deux phases, est une réécriture du séquent. Suite à un événement/action, le séquent, représentant le scénario se déroulant à partir de cette situation, est réécrit en un séquent représentant le scénario se déroulant dans la nouvelle situation. Le principe des algorithmes de parcours, que nous mettons en œuvre dans les phases de validation et d'exécution du scénario, est le même, car les deux algorithmes sont basés sur les règles d'inférence du calcul des séquents [52]. Ces algorithmes sont présentés en détail dans les sections suivantes de la thèse.

Outre ces aspects opérationnels, le pouvoir d'expression du modèle est renforcé par l'ajout de marques particulières concernant les atomes ou les formules.

- Plusieurs sortes de marques peuvent être attachées aux états : la marque indiquant si l'état est un état disponible initial du scénario, la marque indiquant si l'état appartient à un cas d'une étape dans la structure de discours du scénario (le concept de structure de discours est précisé dans la section suivante – voir Figure III.3).

- La marque peut être attachée aux formules ⊕ indiquant le pourcentage attribué à chaque possibilité des choix.
- Plusieurs sortes de marques peuvent être attachées aux événements/actions : la marque indiquant une émotion que l'événement/action peut causer pour le joueur, la marque indiquant la priorité de l'événement/action.

Dans la section suivante, nous décrivons en détail la démarche de modélisation utilisant notre modèle d'histoire.

III.3.2. Description de la méthode de modélisation

La démarche de modélisation est la suivante :

- Les personnages joueurs et non-joueurs sont modélisés par des atomes dans la partie gauche du séquent. Un atome correspond à un état d'un personnage considérant un certain point de vue. Un personnage à un moment peut donc être modélisé par divers atomes qui sont placés dans un ensemble d'états correspondant à la situation du personnage à ce moment là. Ainsi, la taille et le contenu de cet ensemble d'états liés à un personnage peuvent varier au cours du déroulement de l'histoire.
- Les états de l'histoire sont modélisés comme des atomes dans la partie gauche du séquent. Ils représentent les étapes principales dans le déroulement de l'histoire. Par ailleurs, ils sont utilisés afin de définir la structure de discours du scénario modélisé dont tous les discours possibles doivent respecter (le concept de structure de discours est précisé dans la section suivante – voir Figure III.3).
- La disponibilité des états des personnages et des états de l'histoire est considérée comme la disponibilité des atomes correspondants dans la partie gauche du séquent.
- Les choix (les actions) du joueur sont exprimés par des entrées. Cela signifie que le joueur décide son impact dans le déroulement de l'histoire en sélectionnant ces entrées, de ce fait, il est libre de choisir des possibilités dans le développement du discours. Les entrées sont modélisées comme des atomes dans la partie gauche du séquent et deviennent disponibles après avoir été introduites dans l'histoire par le joueur.
- Une formule de conjonction (disjonction) additive dans la partie gauche du séquent représente un choix du joueur (système de pilotage) dans la progression de l'histoire. En utilisant les connecteurs & et ⊕ afin de modéliser une histoire, les auteurs peuvent définir les possibilités de choix dans le scénario et déterminer concrètement qui (entre le système de pilotage et le joueur) fait un choix parmi les possibilités proposées. Ainsi, les auteurs disposent d'une capacité d'expression qui tient compte de la dualité système/joueur. Ce choix peut être enrichi d'une pondération par exemple en attribuant un pourcentage à chaque possibilité de choix (sa valeur par défaut est 100/nombre de possibilité %) si le choix est décidé par le système de pilotage (les possibilités sont reliées par le connecteur ⊕).
- Un événement/action peut modifier la situation d'un (ou de plusieurs) personnage(s) et/ou l'état de l'histoire. Un événement/action est exprimé par une formule utilisant le connecteur ⊸. Le pouvoir d'expression des auteurs est renforcé par la possibilité d'attribuer aux événements/actions un ordre de priorité (la priorité de

l'événement/action est d'autant plus haute que la valeur qui lui est attribuée est petite). A l'exécution lors du calcul, s'il y a beaucoup d'événements/actions satisfaits à un même moment, le système de pilotage sélectionne en premier l'événement/action le plus prioritaire (dont l'ordre de priorité est le plus petit). Cette possibilité de pondération de l'ordre de priorité des événements/actions n'est pas appliquée aux événements/actions reliés par les formules de conjonction/disjonction additive. Par exemple, examinons le séquent A, D, A ⊸ B, D ⊸ E ⊢ B ⊗ E. Il a deux atomes disponibles A et D, deux événements/actions A ⊸ B (indication de priorité = 1) et D ⊸ E (indication de priorité = 2). Comme l'événement/action A ⊸ B est plus prioritaire que l'événement/action D ⊸ E, même si à cet instant les deux événements/actions sont valides, l'événement/action A ⊸ B est exécuté avant l'événement/action D ⊸ E. Ainsi l'ordre de transformation du séquent est : (A, D, A ⊸ B, D ⊸ E ⊢ B ⊗ E) → (B, D, D ⊸ E ⊢ B ⊗ E) → (B, E ⊢ B ⊗ E).

- Il y a deux types d'événement/action dans une histoire : si un événement/action est lié à un état émotionnel (tel que : bonheur, peur, stress…), alors c'est un événement dramatique (c'est-à-dire s'il est exécuté, le joueur passera par cet état émotionnel) ; au contraire, les événements/actions, qui ne sont liés à aucun état émotionnel, sont les événements neutres dans l'histoire.
- Tous les aspects de l'histoire (les états de l'histoire, les états des personnages, les événements/actions…) à un instant lors du déroulement de l'histoire constituent la situation de l'histoire à cet instant là.
- Une sortie (but/conclusion/conséquence) de l'histoire est une terminaison désirée des auteurs. Elle correspond à un atome ou un ensemble d'atomes (reliés entre eux par le connecteur ⊗) dans la partie droite du séquent. Les sorties de l'histoire sont reliées par le connecteur ⊕. Un discours qui abouti à une sortie sera dit « réussi », alors qu'un discours qui aboutit à une terminaison non désirée sera dit « échoué ». Nous réservons les termes « valide/non valide » pour les discours réussis pour lesquels on évalue la qualité du discours produit vis-à-vis des contraintes de l'auteur.
- Une preuve exprime les formules d'implication linéaire (événements/actions) à exécuter afin d'atteindre une des sorties d'un séquent (qui peut être réussie ou pas), elle est donc équivalente à un discours qui est une suite ordonnée d'événements/actions décrivant un déroulement possible de l'histoire.
- À partir d'un séquent, on peut construire toutes les manières d'écrire ses preuves en utilisant les règles d'inférence de la logique linéaire (toutes ces manières constituent le graphe des preuves du séquent). Par conséquent, un séquent correspond à un scénario qui est l'ensemble de tous les discours possibles pour une histoire. Pour écrire toutes les preuves d'un séquent, nous utilisons un algorithme de parcours, qui est une recherche en profondeur s'appuyant sur le calcul des séquents [52]. Cet algorithme et ses versions différentes sont expliqués en détail dans les sections suivantes de la thèse.

Remarque : Dans le cadre de cette thèse, nous n'employons pas le connecteur « ! » de la logique linéaire, un événement/action n'est donc utilisé qu'une fois pour un discours (autrement dit, il est supprimé de la liste d'événements/actions juste après son utilisation), et

de ce fait, les scénarios ne comportent pas de boucle. Ainsi, un scénario peut être représenté comme un arbre (arbre des preuves), c'est-à-dire le graphe orienté acyclique des preuves (pour être plus bref, désormais on l'appelle « graphe des preuves »). La propriété de narration « Absence de boucles infinies » est ainsi garantie par construction.

Afin d'illustrer ce qui précède, examinons le séquent A, D, EA1 ⊗ EA3 ⊕ EA2, EA4 ⊢ B ⊗ C ⊕ F où EA1 (ordre de priorité = 1) : D ⊸ E ; EA2 (ordre de priorité = 2) : A ⊗ D ⊸ B ⊗ E ; EA3 (ordre de priorité = 3) : A ⊸ B ; EA4 (ordre de priorité = 4) : B ⊗ E ⊸ B ⊗ C. Le scénario correspondant à ce séquent possède : 6 atomes/états A, B, C, D, E, F, dans lesquels A, D sont disponibles ; 4 événements/actions EA1, EA2, EA3, EA4 avec la priorité de EA1 > EA2 > EA3 > EA4 ; 2 sorties B ⊗ C (qui comprend 2 atomes/états B, C) et F (qui comprend 1 atome/état F) ; un choix effectué par le système de pilotage (choisir soit EA1 ⊗ EA3 qui signifie que le séquent devient A, D, EA1, EA3, EA4 ⊢ B ⊗ C ⊕ F, soit EA2 qui signifie que le séquent devient A, D, EA2, EA4 ⊢ B ⊗ C ⊕ F). Nous pouvons attribuer un pourcentage à chaque possibilité (tel que 40, 60 qui veut dire que la probabilité de choisir EA1 ⊗ EA3 est de 40% tandis que celle de choisir EA2 est de 60%), sinon la valeur par défaut pour les deux possibilités est de 50%. Dans le cas où le système de pilotage sélectionne EA1 ⊗ EA3 et le séquent devient donc A, D, EA1, EA3, EA4 ⊢ B ⊗ C ⊕ F, nous pouvons trouver, à cet instant, que les deux événements/actions EA1 et EA3 sont valides, néanmoins puisque la priorité de EA1 > EA3, EA1 est exécuté. Enfin, tous les discours possibles dans le scénario (deux discours/preuves/chemins/branches réussis) sont donnés dans la Figure III.1 (simplifiée à la substitution des événements/actions) : EA1 → EA3 → EA4 et EA2 → EA4 (les deux discours atteignent la sortie B ⊗ C). Par ailleurs, il est facile de constater que, la modélisation d'une histoire en logique linéaire nous permet de connaître la forme de son scénario ainsi que de la modifier simplement.

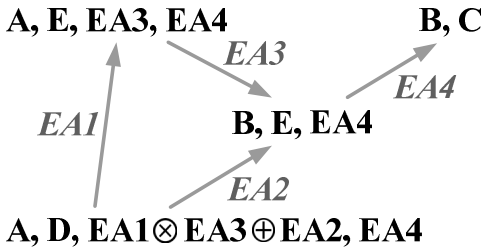


Figure III.1. Tous les discours possibles dans le scénario (graphe des preuves) correspondant au séquent A, D, EA1 ⊗ EA3 ⊕ EA2, EA4 ⊢ B ⊗ C ⊕ F où EA1 (ordre de priorité = 1) : D ⊸ E ; EA2 (ordre de priorité = 2) : A ⊗ D ⊸ B ⊗ E ; EA3 (ordre de priorité = 3) : A ⊸ B ; EA4 (ordre de priorité = 4) : B ⊗ E ⊸ B ⊗ C ; chaque nœud correspond à la partie gauche du séquent (situation de l'histoire) à un moment ; chaque arête est un événement/action (formule d'implication linéaire) exécuté.

III.3.3. Apports aux travaux précédents menés au L3I

La méthode de modélisation d'une histoire en logique linéaire que nous avons décrite ci-dessus apporte des améliorations importantes aux travaux précédents menés au L3I. En effet, dans le cadre des travaux précédents réalisés par Collé – Prigent *et al.* [29, 81], l'expression a été limitée aux formules utilisant les connecteurs & et ⊕. Le modèle n'utilisait pas d'indication de priorité pour les événements/actions. Par conséquent, tous les discours

possibles d'un scénario venaient des possibilités d'ordonnancement de ses événements/actions (s'il y avait plusieurs événements/actions satisfaits à un même moment, alors chacun d'eux serait choisi, l'un après l'autre).

Par exemple, examinons le séquent A, C, D, EA1, EA2, EA3 ⊢ B ⊗ C ⊗ F où EA1 : A ⊗ C ⊸ A ⊗ B ⊗ C ; EA2 : D ⊸ E ; EA3 : A ⊗ C ⊗ E ⊸ C ⊗ F. Le scénario correspondant à ce séquent a six atomes/états A, B, C, D, E, F, dans lesquels A, C, D sont disponibles ; trois événements/actions EA1, EA2, EA3 (leur priorité est égale) ; une sortie B ⊗ C ⊗ F comprenant trois atomes/états B, C et F. Tous les discours possibles (possibilités de choix) dans le scénario sont donnés dans la Figure III.2 (simplifiée à la substitution des événements/actions) : EA1 → EA2 → EA3, EA2 → EA1 → EA3 et EA2 → EA3 (deux discours réussis et un discours échoué).

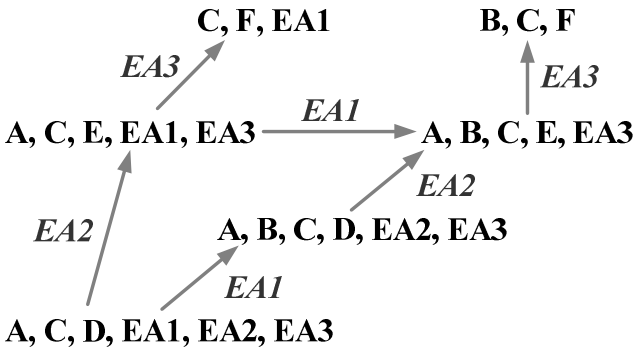


Figure III.2. Tous les discours possibles dans le scénario (graphe des preuves) correspondant au séquent A, C, D, EA1, EA2, EA3 ⊢ B ⊗ C ⊗ F où EA1 : A ⊗ C ⊸ A ⊗ B ⊗ C ; EA2 : D ⊸ E ; EA3 : A ⊗ C ⊗ E ⊸ C ⊗ F (dans le cadre des travaux précédents [29, 81]).

Cette forme de modélisation d'une histoire comporte les inconvénients suivants :

- Les auteurs ne peuvent pas distinguer les possibilités de choix correspondant à une décision du joueur, de celles décidées par le système de pilotage. Par exemple, dans la Figure III.2, au premier nœud du graphe (A, C, D, EA1, EA2, EA3), il existe deux possibilités de choix (exécuter soit EA1 soit EA2), mais nous ne pouvons pas savoir qui (le joueur ou le système de pilotage) décide ce choix. L'utilisation des connecteurs & et ⊕ dans notre méthode de modélisation permet aux auteurs de traiter cette ambiguïté.
- Le modèle ne peut pas exprimer le choix exclusif entre plusieurs actions.
- Les auteurs ne peuvent pas définir l'ordre de priorité dans la sélection des événements/actions. Cela réduit leur pouvoir de contrôle pour le déroulement du scénario. Par exemple, il peut exister d'autres possibilités dans le scénario en dehors des discours intentionnels des auteurs.

Outre l'utilisation des connecteurs &, ⊕ et l'attribution d'ordres de priorité aux événements/actions (grâce auxquelles, les inconvénients susmentionnés sont surmontés), l'apport de la méthode de modélisation comporte les ajouts suivants qui augmentent le pouvoir d'expression du modèle : donner la notion de structure de discours que tous les discours possibles dans le scénario devaient respecter, et lier les événements/actions aux états

émotionnels. Ces améliorations nous permettent de vérifier la propriété de narration « Satisfaction de la structure » ainsi que les propriétés concernant l'effet dramatique créé par chacun des discours.

La section suivante explique en détail la seconde étape de la validation d'un scénario, mise en œuvre dans l'approche proposée : évaluer la qualité du scénario obtenu après le processus de modélisation d'une histoire en logique linéaire, en parcourant tous les discours possibles et en analysant leur satisfaction pour la presque totalité de propriétés de narration (sauf les trois propriétés « Absence de boucles infinies », « Correspondance d'émotion du joueur » et « Évolution de la tension dramatique »). Néanmoins, pour que les lecteurs puissent mieux comprendre notre solution, nous présentons tout de suite une formalisation des propriétés de narration.

III.4. Formalisation des propriétés de narration

Dans cette section, nous présentons la manière d'exprimer les propriétés de narration.

Propriétés liées au déroulement des discours

- **Accessibilité :** Un scénario satisfait cette propriété si la modélisation, correspondant au scénario, ne comporte aucun état inutilisé, aucune entrée inutilisée, aucun événement/action inutilisé, aucune sortie inutilisée où :
    - les états et les entrées inutilisés sont ceux qui n'apparaissent dans aucune expression d'événement/action ;
    - les événements/actions inutilisés sont ceux qui n'apparaissent dans aucun discours ;
    - les sorties inutilisées sont celles qui ne sont la terminaison d'aucun discours.
- **Absence de blocage :** Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété. Un discours ne satisfait pas cette propriété si : (1) il existe une situation à partir de laquelle aucun calcul ne peut progresser, et (2) cette situation du discours ne correspond à aucune terminaison souhaitée de l'histoire.
- **Complexité :** Un discours satisfait cette propriété si son nombre d'événements/actions n'est pas inférieur à un nombre minimum prédéfini par l'auteur. Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété et le nombre de discours possibles dans le scénario n'est pas inférieur à un nombre minimum fixé par l'auteur. Par exemple, un scénario d'un jeu est considéré comme suffisamment compliqué s'il comporte au moins 30 discours dont chacun possède au moins 10 événements/actions.
- **Satisfaction d'événements clés :** Les événements clés d'une histoire se définissent comme un ensemble non-ordonné d'événements/actions. Cet ensemble est un sous-ensemble des événements/actions de l'histoire. Dans le processus de modélisation, l'auteur définit cet ensemble d'événements clé. Un discours satisfait cette propriété s'il comporte tous les événements appartenant à cet ensemble. Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété.

- **Satisfaction de la structure :** La structure de discours d'un scénario est définie par l'auteur comme un ensemble ordonné d'étapes. Une étape dans la structure de discours comporte un cas ou plusieurs cas distincts. Un cas d'une étape correspond à un nœud dans le graphe des preuves du scénario (voir Figure III.1). Deux cas distincts d'une même étape ne peuvent être reliés par arête (c'est-à-dire qu'ils appartiennent à deux possibilités distinctes dans le calcul de la preuve ou de manière équivalente dans le déroulement du scénario). Un nœud dans le graphe des preuves, correspondant à un cas d'une étape, doit comporter au moins un état de l'histoire. Un discours « passe une étape » s'il passe un des cas de l'étape. Un discours « passe un cas d'une étape » s'il existe un calcul (celui correspondant au discours) qui conduit des états initiaux de l'histoire aux états de ce cas. Un discours satisfait cette propriété s'il passe toutes les étapes selon l'ordre donné dans la structure de discours. Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété. Par exemple, la structure de discours de l'histoire « Le Petit Chaperon rouge » comporte 4 étapes :

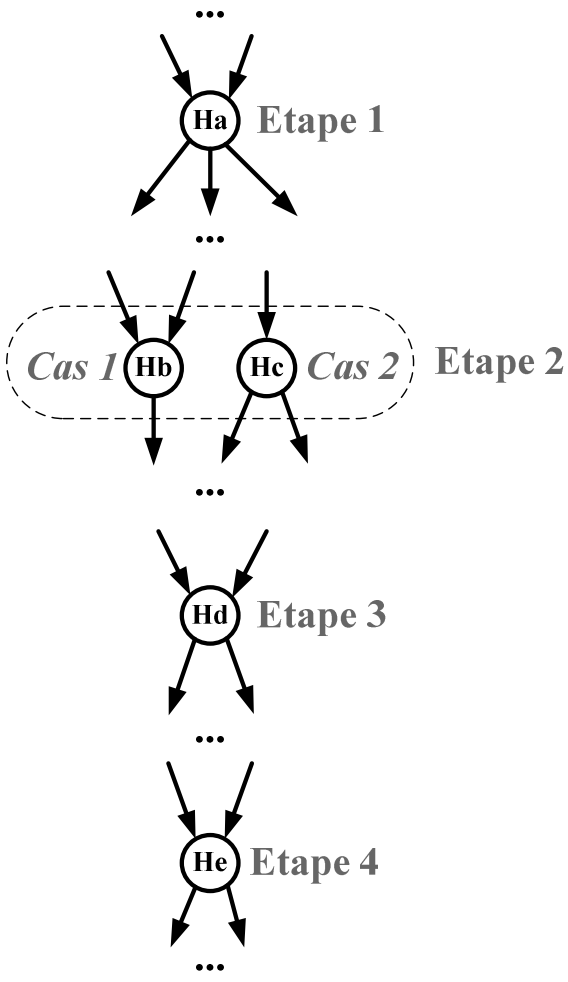


Figure III.3. Structure de discours dans le graphe des preuves de l'histoire « Le Petit Chaperon rouge ».

  - **Etape 1 :** Cette étape ne contient qu'un cas qui correspond à l'état Ha de l'histoire, cet état décrit la situation où la mère demande au Petit Chaperon rouge d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère.
  - **Etape 2 :** Cette étape comporte deux cas distincts

        - ***Cas 1 :*** Ce cas correspond à l'état Hb de l'histoire qui décrit la situation où le Petit Chaperon rouge rencontre le loup dans le bois et lui dit où est la maison de la grand-mère ;
        - ***Cas 2 :*** Ce cas correspond à l'état Hc de l'histoire qui décrit la situation où le Petit Chaperon rouge rencontre le loup dans le bois mais ne lui dit pas où est la maison de la grand-mère.
    - **Etape 3 :** Cette étape ne contient qu'un cas qui correspond à l'état Hd de l'histoire, cet état décrit la situation où la grand-mère et le Petit Chaperon rouge sont mangés par le loup.
    - **Etape 4 :** Cette étape ne contient qu'un cas qui correspond à l'état He de l'histoire, cet état décrit la situation où le loup est mort, la grand-mère et le Petit Chaperon rouge sont vivants.

  Ainsi, un discours satisfait cette structure s'il passe toutes les étapes ci-dessus selon l'ordre imposé (à l'étape 2, le discours peut passer par un des deux cas distincts). La Figure III.3 donne un scénario de l'histoire qui satisfait cette structure de discours.
- **Séquencement :** Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété. Une séquence est un ensemble ordonné des événements. Pour définir un tel ensemble, un auteur, sélectionne cette liste parmi les événements/actions du scénario modélisé. Une contrainte peut être adjointe à cet ensemble ordonné d'événements. L'auteur à la possibilité de les préciser :
    - Les événements dans la suite sont consécutifs : Dans ce cas, un discours est considéré comme « satisfaisant la propriété *Séquencement* » si tous les événements apparaissent dans l'ordre, et si l'apparition des événements dans le discours est consécutive.
    - Les événements dans la suite ne sont pas consécutifs : Dans ce cas, un discours est considéré comme « satisfaisant la propriété *Séquencement* » si tous les événements apparaissent dans l'ordre (entre deux événements il peut exister d'autres événements).

Propriétés liées à l'effet dramatique créé par un discours

- **Fréquence des événements dramatiques :** La fréquence des événements dramatiques est un nombre entier, paramètre du modèle, défini par l'auteur dans la phase de modélisation. Un discours satisfait cette propriété si le nombre d'événements dramatiques n'est pas inférieur à ce seuil minimum. Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété. Par exemple, un discours est considéré comme satisfaisant cette propriété s'il possède au moins 5 événements dramatiques.
- **Variété d'effets dramatiques :** La variété d'effets dramatiques est un nombre entier, paramètre du modèle, défini par l'auteur dans la phase de modélisation. Un discours satisfait cette propriété si le nombre de catégories des événements dramatiques, que le discours dévoile, n'est pas inférieur à ce seuil minimum. Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété. Par exemple, un

discours est considéré comme satisfaisant cette propriété s'il provoque au moins 3 types d'émotion pour le joueur (c'est-à-dire qu'il possède au moins 3 événements dramatiques qui appartiennent à 3 catégories différentes).

- **Équilibre de l'effet dramatique :** Chaque situation lors du déroulement de l'histoire peut être qualifiée par un attribut d'effet dramatique (dont la valeur est *Stress*, *Peur*, *Fierté*, *Bonheur*…) que cette situation cause pour le joueur. En l'absence de qualification explicite, la valeur de l'attribut est *Neutre*. Pour vérifier la satisfaction de cette propriété pour un discours, l'auteur définit un ensemble des motifs de catégories d'effet dramatique dont chacun est un quadruplet <C, N_min, N_max, {C_suivantes}> :
    - C : catégorie de l'effet dramatique ;
    - N_min : nombre minimum d'événements/actions consécutifs qui causent cet effet dramatique pour le joueur (la durée minimale de cet effet dramatique) ;
    - N_max : nombre maximum d'événements/actions consécutifs qui causent cet effet dramatique pour le joueur (la durée maximale de cet effet dramatique) ;
    - {C_suivantes} : ensemble de catégories d'effet dramatique suivantes (la catégorie de l'effet dramatique suivant de cet effet dramatique doit être un élément dans l'ensemble).

  Par exemple, un motif <Stress, 2, 4, {Neutre, Bonheur, Fierté}> signifie : la durée minimale de l'effet dramatique *Stress* est 2 événements/actions consécutifs, sa durée maximale est 4 événements/actions consécutifs, l'effet dramatique suivant de l'effet dramatique *Stress* doit être soit *Neutre*, soit *Bonheur*, soit *Fierté*.
  Un discours est considéré comme « satisfaisant la propriété *Équilibre de l'effet dramatique* » si la séquence de ses attributs d'effet dramatique satisfait à l'ensemble des motifs de catégories d'effet dramatique défini par l'auteur (c'est-à-dire qu'elle satisfait à tous les motifs dans l'ensemble). Un scénario satisfait cette propriété si tous ses discours possibles satisfont cette propriété.
  Par exemple, soit une histoire dont le scénario a 10 événements/actions EA01, EA02,…, EA10 dans lesquels EA02, EA03, EA05, EA07, EA08, EA10 sont les événements dramatiques, tandis que EA01, EA04, EA06 et EA09 sont les événements neutres :
    - EA02 et EA07 sont liés à l'émotion *Stress* ;
    - EA03 et EA08 sont liés à l'émotion *Peur* ;
    - EA05 est lié à l'émotion *Fierté* ;
    - EA10 est lié à l'émotion *Bonheur*.

  L'ensemble des motifs de catégories d'effet dramatique dénifi par l'auteur est :
    - Motif 1 : <Stress, 1, 2, {Neutre, Bonheur, Fierté}>
    - Motif 2 : <Peur, 1, 1, {Neutre, Bonheur, Fierté}>
    - Motif 3 : <Fierté, 1, 1, {Neutre, Stress, Peur}>
    - Motif 4 : <Bonheur, 1, 1, {Neutre, Stress, Peur}>
    - Motif 5 : <Neutre, 1, 3, {Bonheur, Fierté, Stress, Peur}>

  Supposons que ce scénario comprenne les trois discours suivants :

- Discours 1 : EA01 → EA02 → EA03 → EA04 → EA05 → EA06 → EA07 → EA08 → EA09 → EA10
  - La séquence de ses attributs d'effet dramatique est : Neutre, Stress, Peur, Neutre, Fierté, Neutre, Stress, Peur, Neutre, Bonheur
  - ➔ Ce discours ne satisfait pas à l'ensemble des motifs de catégories d'effet dramatique défini par l'auteur car il ne satisfait pas au motif 1 (l'effet dramatique *Peur* est suivant l'effet dramatique *Stress*).
- Discours 2 : EA02 → EA01 → EA04 → EA06 → EA09 → EA10
  - La séquence de ses attributs d'effet dramatique est : Stress, Neutre, Neutre, Neutre, Neutre, Bonheur
  - ➔ Ce discours ne satisfait pas à l'ensemble des motifs de catégories d'effet dramatique défini par l'auteur car il ne satisfait pas au motif 5 (la durée de l'effet dramatique *Neutre* est 4 événements/actions consécutifs).
- Discours 3 : EA01 → EA02 → EA05 → EA07 → EA10
  - La séquence de ses attributs d'effet dramatique est : Neutre, Stress, Fierté, Stress, Bonheur
  - ➔ Ce discours satisfait à l'ensemble des motifs de catégories d'effet dramatique défini par l'auteur car il satisfait à tous les motifs dans l'ensemble.

Ainsi, ce scénario ne satisfait pas la propriété *Équilibre de l'effet dramatique*.

La section suivante explique en détail la seconde étape de l'approche proposée pour la validation d'un scénario : évaluer la qualité du scénario obtenu après le processus de modélisation d'une histoire en logique linéaire, en parcourant tous les discours possibles et en analysant, grâce à la formalisation des propriétés présentée plus haut, la satisfaction des discours pour la presque totalité des propriétés de narration (sauf les trois propriétés « Absence de boucles infinies », « Correspondance d'émotion du joueur » et « Évolution de la tension dramatique »).

En s'appuyant sur les résultats d'analyse, les auteurs peuvent corriger le modèle afin de recevoir un nouveau scénario et puis refaire l'analyse si nécessaire (dans le cas où la qualité du scénario courant n'est pas suffisamment bonne). Ce processus est répété dans la phase de construction de scénario d'une histoire jusqu'à ce que les auteurs obtiennent un scénario le plus pertinent au regard leurs objectifs. Ce scénario validé est ensuite utilisé comme l'entrée d'un système de pilotage du jeu, et donc permet au système de pilotage de diriger le déroulement du jeu selon le scénario valide produit.

III.5. Évaluation de la qualité d'un scénario modélisé en logique linéaire

En vue d'évaluer la qualité du scénario obtenu après le processus de modélisation d'une histoire en logique linéaire, le principe le plus important de notre solution est de parcourir tous les discours possibles du scénario, ce qui équivaut à parcourir le graphe des preuves du séquent de logique linéaire du modèle. La preuve d'un séquent de logique linéaire est le modèle du fonctionnement dans notre approche. Nous présentons ci-dessous quelques

solutions de mise en œuvre de la preuve de séquent qui existent dans d'autres travaux. Nous nous interessons aux travaux utilisant les réseaux de Petri, Lolli, Lygon, LINK prover, Coq, *llprover*. Dans le même temps, nous montrons en quoi elles ne sont pas adéquates au regard de nos exigences.

III.5.1. Solutions existantes pour mener les preuves d'un séquent de logique linéaire

III.5.1.1. Réseaux de Petri

Les articles [43, 61, 62] montrent qu'il existe une équivalence entre un séquent associé à un sous-ensemble de la logique linéaire et un réseau de Petri. Ainsi, la preuve d'un séquent correspond au franchissement d'une séquence de transitions dans son réseau de Petri équivalent, du marquage initial au marquage prévu. Cette approche a été mise en œuvre dans les travaux antérieurs menés au laboratoire L3I dans [29, 81], en vue de générer l'ensemble des preuves possibles d'un séquent de logique linéaire. Néanmoins, une telle application des réseaux de Petri pour la preuve de séquent rencontre quelques problèmes :

- La transformation d'un séquent de logique linéaire en un réseau de Petri apporte certaines restrictions. En effet, on ne peut ni utiliser tous les connecteurs (⊗, &, ⊕, ⊸) ni représenter des formules d'implication linéaire complexes. Plus concrètement, la partie gauche du séquent ne peut comporter que les connecteurs ⊗, ⊸ ; sa partie droite n'accepte que les connecteurs ⊗, ⊕ ; et on peut seulement exprimer des formules d'implication linéaire simples correspondant à un format unique A1 ⊗ A2 ⊗ … ⊗ An ⊸ B1 ⊗ B2 ⊗ … ⊗ Bm, où Ai, Bj sont des atomes, en d'autres termes, il est impossible de représenter des formules d'implication linéaire plus complexes, telles que A ⊸ (B ⊸ C) ou (A ⊸ B) ⊸ (C ⊸ D)… Par conséquent, la transformation d'un séquent de logique linéaire en un réseau de Petri réduit le pouvoir d'expression de la logique linéaire, et donc il n'est pas possible pour nous de modéliser des systèmes complexes. Du point de vue de l'utilisateur, une autre difficulté est que cette approche ajoute le modèle des réseaux de Petri à celui de la logique linéaire.
- La technique de transformation du modèle en logique linéaire vers le modèle sous forme de réseau de Petri ne permet pas de tenir compte de priorité, réduisant le pouvoir d'expression du modèle par rapport à nos contraintes. Ainsi, parmi les possibilités de choix vérifiées lors de la preuve d'un séquent par son réseau de Petri équivalent, il peut y avoir beaucoup de preuves dont la vérification est inutile pour notre approche. Par exemple, voyons le séquent A, C, D, EA1, EA2, EA3 ⊢ B ⊗ C ⊗ F où EA1 (ordre de priorité = 1) : A ⊗ C ⊸ A ⊗ B ⊗ C ; EA2 (ordre de priorité = 2) : D ⊸ E ; EA3 (ordre de priorité = 3) : A ⊗ C ⊗ E ⊸ C ⊗ F. Si nous utilisons un réseau de Petri correspondant au séquent en vue de le prouver, ce réseau de Petri déroule entièrement les trois preuves EA1 → EA2 → EA3, EA2 → EA1 → EA3 et EA2 → EA3 (voir Figure III.2), parce que l'ordre de priorité de EA1, EA2 et EA3 n'est pas pris en considération. Toutefois, ce séquent, dans notre approche, comme la priorité de EA1 > EA2 > EA3, ne comprend qu'une preuve valide EA1 → EA2 → EA3. Par conséquent, il est inutile d'explorer les branches dans le graphe des preuves d'un séquent qui ne respectent pas l'ordre de priorité entre ses formules d'implication linéaire (telles que EA2 → EA1 → EA3 et EA2 → EA3 dans l'exemple examiné), ce

qui aboutit rapidement à une explosion combinatoire, puisque la complexité de l'algorithme d'ordonnance des formules d'implication linéaire est exponentielle.

III.5.1.2. Autres solutions pour mener les preuves

En plus des réseaux de Petri, en vue de prouver un séquent de logique linéaire, il est aussi possible pour les utilisateurs d'employer Lolli [51, 108], Lygon [50, 109], LINK prover [47], Coq [9, 13] et *llprover* [96, 103]. Cependant, ils ne répondent pas exactement à nos besoins :

- Ces outils disponibles montre si un séquent est prouvé ou non (c'est-à-dire qu'il existe au moins une preuve réussie). Ils ne permettent pas de parcourir toutes les preuves possibles du séquent, interdissant de vérifier certaines propriétés comme : combien de preuves il y a, quelles preuves sont réussies, quelles preuves sont échouées….
- De façon similaire au cas utilisant les réseaux de Petri décrit précédemment, ces outils ne prennent pas en considération l'ordre de priorité des formules d'implication linéaire. Par conséquent, parmi les choix étudiés pendant l'écriture de la preuve d'un séquent, il y a peut-être plusieurs preuves dont la vérification est inutile. D'ailleurs, dans certains cas, le résultat obtenu n'est pas correct. Par exemple, le séquent A, B, EA1, EA2 ⊢ C ⊗ D où EA1 (ordre de priorité = 1) : B ⊸ D ; EA2 (ordre de priorité = 2) : A ⊗ B ⊸ B ⊗ C, selon ces outils, est vrai car ils trouvent la branche valide EA2 → EA1, mais dans notre approche, ce séquent est faux parce qu'il ne comporte qu'une branche non valide EA1 → EA2 dans son graphe des preuves.
- S'ajoute à ce qui a été dit précédemment, le fait que LINK prover ne traite pas complètement tous les connecteurs (⊗, &, ⊕, ⊸) ; *llprover* englobe tous les cas, mais il traite le connecteur ⊕ dans la partie gauche du séquent différemment de notre approche. Plus concrètement, LINK prover s'applique uniquement au fragment multiplicatif de la logique linéaire ; pour *llprover*, il traite le connecteur ⊕ dans la partie gauche d'un séquent comme suit :

$$\frac{\Gamma 1, \Gamma 2, \Gamma 4 \vdash \Delta \qquad \Gamma 1, \Gamma 3, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \oplus \Gamma 3, \Gamma 4 \vdash \Delta} \oplus L$$

  Cela signifie que *llprover* s'intéresse à la vérité du séquent Γ1, Γ2 ⊕ Γ3, Γ4 ⊢ Δ et conclut qu'il est vrai si et seulement si les deux subséquents Γ1, Γ2, Γ4 ⊢ Δ et Γ1, Γ3, Γ4 ⊢ Δ sont vrais. S'il constate que Γ1, Γ2, Γ4 ⊢ Δ est faux, il conclut le séquent Γ1, Γ2 ⊕ Γ3, Γ4 ⊢ Δ est faux et ne prouve pas Γ1, Γ3, Γ4 ⊢ Δ. Ce mécanisme ne fourni pas l'arbre des preuves utilisé dans notre approche. En effet, nous nous intéressons seulement à la vérité des deux subséquents, pas à la vérité du séquent original. En d'autres termes, nous examinons les preuves des deux subséquents.

Ainsi, nous avons cité précédemment, pour réaliser les preuves d'un séquent de logique linéaire, les réseaux de Petri, Lolli, Lygon, LINK prover, Coq et *llprover*. Nous avons montré en quoi ces preuves n'étaient pas pertinentes au regard de nos exigences. Par conséquent, nous devons proposer, dans le cadre de cette thèse, une autre façon de parcourir le graphe des preuves d'un séquent de logique linéaire, ce qui équivaut à parcourir tous les discours

possibles du scénario correspondant. L'exécution de notre solution est divisée en deux sous-sections correspondant aux scénarios avec les événements/actions simples ou complexes en raison de leurs mécanismes de déduction différents.

III.5.2. Notre solution pour parcourir tous les discours possibles d'un scénario

III.5.2.1. Scénarios ne comportant que des événements/actions simples

En premier lieu, nous donnons la définition d'un événement/action simple dans notre approche : un événement/action *simple* est une formule d'implication linéaire de la forme A1 ⊗ A2 ⊗ … ⊗ An ⊸ B1 ⊗ B2 ⊗ … ⊗ Bm, où Ai, Bj sont les atomes. Par exemple, A ⊗ C ⊸ B est un événement/action simple, tandis que (A ⊸ B) ⊸ C n'est pas un événement/action simple. Une « version réduite » du calcul des séquents nous permet d'analyser un scénario ne contenant que des événements/actions simples. Les règles principales sont les suivantes :

- la partie droite du séquent (qui est composée de sorties reliées par les connecteurs ⊕, et dont chaque sortie comprend un ou plusieurs états reliés par les connecteurs ⊗) n'est pas transformée dans le processus de calcul ;
- pour normaliser les formules, le connecteur ⊗, entre deux atomes, entre deux formules, ou entre un atome et une formule dans la partie gauche du séquent, est substitué par une virgule « , » ;
- une formule de conjonction (disjonction) additive dans la partie gauche du séquent avec *n* composants divise le séquent en *n* sous-branches correspondant à ces composants (le calcul parcourt ensuite les sous-branches séparément), par exemple le séquent A, D, EA1 ⊗ EA3 ⊕ EA2, EA4 ⊢ B ⊗ C ⊕ F est divisé en 2 sous-branches : A, D, EA1, EA3, EA4 ⊢ B ⊗ C ⊕ F et A, D, EA2, EA4 ⊢ B ⊗ C ⊕ F ;
- une formule d'implication linéaire est exécutable si et seulement si l'ensemble des atomes de son membre de gauche est un sous-ensemble de la liste d'atomes disponibles du séquent ;
- l'exécution d'une formule d'implication linéaire consiste à réécrire le séquent en remplaçant, dans la liste des atomes disponibles du séquent, les atomes correspondant à l'ensemble des atomes de son membre de gauche, par l'ensemble des atomes de son membre droit ;
- la création d'un discours consiste à réécrire un séquent en itérant le processus (en exécutant, à chaque étape, un événement/action exécutable dont l'ordre de priorité est le plus petit) jusqu'à ce que :
    - la liste d'atomes disponibles du séquent comprenne l'ensemble d'atomes d'une de ses sorties/conclusions/conséquences → le séquent est prouvé, et donc le discours courant est réussi ;
    - il ne reste aucun événement/action exécutable → le séquent n'est pas prouvé, et donc le discours courant a échoué.

Ainsi, à travers le processus d'examen d'un séquent en appliquant un algorithme déduit de la « version réduite » du calcul des séquents ci-dessus (l'exemple donné dans la Figure III.1 est une illustration), on peut parcourir tous les discours possibles du scénario correspondant. Par conséquent, il nous est possible de connaître les informations importantes sur le scénario

telles que : les discours réussis/échoués, les événements/actions dans chaque discours, les états disponibles dans chaque discours après chaque étape, les éléments de modélisation (états, entrées, événements/actions, sorties) inutilisés, les effets dramatiques causés par chacun des discours sur le joueur… Grâce à ces informations, on peut évaluer la qualité du scénario courant en analysant et vérifiant sa satisfaction pour la presque totalité des propriétés de narration abordées plus haut (sauf les trois propriétés « Absence de boucles infinies », « Correspondance d'émotion du joueur » et « Évolution de la tension dramatique »). Nous présentons en détail, dans le chapitre suivant et avec l'aide d'un ensemble d'outils que nous avons développé, comment est faite cette évaluation.

III.5.2.2. Scénarios étendus aux événements/actions complexes

La section précédente nous permet seulement d'évaluer les scénarios avec les événements/actions simples. Il nous est impossible d'évaluer les scénarios avec les types d'événement/action plus complexes tels que A ⊸ (B ⊸ C), (A ⊸ B) ⊸ C ou (A ⊸ B) ⊸ (C ⊸ D)… Par conséquent, la puissance de modélisation de la logique linéaire est diminuée. Par exemple, dans une boutique, le coût d'un bouquet de fleurs est de 2 euros, ce qui se traduit par l'implication linéaire « 2 euros ⊸ bouquet de fleurs ». Le séquent valide « 2 euros, 2 euros ⊸ bouquet de fleurs ⊢ bouquet de fleurs » exprime que : si quelqu'un a 2 euros, et en même temps, un bouquet de fleurs est disponible dans la boutique, alors il peut acheter ce bouquet. Supposons que dans la boutique il ne reste qu'un bouquet de fleurs. Ce bouquet sera vendu à la première personne qui le demande. Si nous ne sommes pas cette personne, alors nous n'avons pas le droit de l'acheter même si nous avons 2 euros. Autrement dit, notre ressource disponible en ce moment est seulement de 2 euros (il nous manque la ressource « 2 euros ⊸ bouquet de fleurs »), c'est pourquoi il est impossible pour nous d'acheter le bouquet de fleurs. Néanmoins, si nous en avons vraiment besoin, nous pouvons demander à la personne ayant le droit d'achat de nous vendre ce droit d'achat en payant 3 euros. Supposons qu'elle soit d'accord, et que nous possédons la ressource. L'achat du droit se traduit par la formule « 3 euros ⊸ (2 euros ⊸ bouquet de fleurs) ». Nous pouvons donc acheter le bouquet de fleurs avec le montant total de 5 euros, car le scénario est représenté par le séquent « 2 euros, 3 euros, 3 euros ⊸ (2 euros ⊸ bouquet de fleurs) ⊢ bouquet de fleurs » qui est prouvé.

Nous présentons, dans cette section, notre nouvelle approche afin d'évaluer les scénarios étendus à ces types d'événement/action complexes.

III.5.2.2.1. Problème à résoudre

Nous présentons en premier lieu le processus de preuve dans le cas où le séquent n'est composé que d'événements/actions simples. C'est un calcul de séquents réduit. Un des principes essentiels de cette version réduite du calcul des séquents est : une implication linéaire (un événement/action) est exécutable si l'ensemble des atomes de son membre de gauche est un sous-ensemble de la liste d'atomes disponibles du séquent. Par exemple, soit le séquent A, B, C, A ⊗ B ⊸ D ⊢ C ⊗ D, l'événement/action A ⊗ B ⊸ D est exécutable car l'ensemble des atomes de son membre de gauche {A, B} est un sous-ensemble de la liste d'atomes disponibles du séquent {A, B, C}, ce séquent est donc prouvé. En revanche si examinons un autre séquent A, (C ⊸ D) ⊸ B, C ⊸ D ⊢ A ⊗ B, qui possède un atome

disponible A, deux événements/actions (C ⊸ D) ⊸ B et C ⊸ D. Aucun événement/action parmi ces deux événements/actions n'est exécutable selon le principe mentionné ci-dessus car :

- le membre de gauche du premier événement/action (C ⊸ D) ⊸ B est C ⊸ D qui n'est pas un ensemble d'atomes ;
- le membre de gauche du second événement/action C ⊸ D est C qui n'est pas un sous-ensemble de la liste d'atomes disponibles du séquent qui ne comprend que l'atome A.

Ainsi ce séquent n'est pas prouvable avec la version réduite du calcul des séquents. Par conséquent, une nouvelle façon pour parcourir tous les discours possibles dans des scénarios avec les événements/actions complexes est indispensable.

III.5.2.2.2. Solution pour parcourir tous les discours possibles d'un scénario étendu aux événements/actions complexes

Considérons l'exemple précédent, une réification des actions conduit à réécrire le premier événement/action (C ⊸ D) ⊸ B en EA ⊸ B et le second événement/action C ⊸ D en EA (remplacer « C ⊸ D » par EA). Dans ce cas, le séquent devient A, EA ⊸ B, EA ⊢ A ⊗ B. Le calcul peut donc être éffectué et la transformation du séquent devient : (A, EA ⊸ B, EA ⊢ A ⊗ B) → (A, B ⊢ A ⊗ B), et le séquent est donc prouvé. C'est le principe principal de notre solution pour les séquents avec les événements/actions complexes, qui est basée sur la « version complète » du calcul des séquents.

Avant de décrire la solution proposée, il est nécessaire de noter que, dans le cadre de cette thèse, nous n'avons pas l'intention de prouver un séquent de logique linéaire dans le cas général, mais nous voulons seulement utiliser le résultat du processus de preuve pour analyser un scénario d'une histoire, modélisé par un séquent selon notre approche. Les « composants » suivants dans la syntaxe de la logique linéaire ne sont pas pris en compte (car ils n'apparaissent pas dans les séquents reçus après le processus de modélisation) : la négation linéaire (par exemple $A^{\perp}$) ; le connecteur & dans la partie droite du séquent ; le connecteur ⅋ ; les constantes 1, 0, ⊤, ⊥ ; les exponentielles !, ? ; et les quantificateurs ∀, ∃.

Les règles d'inférence, utilisées dans le calcul des séquent mis en oeuvre dans le scénario modélisé avec les événements/actions complexes, comportent les suivantes :

- ⊗L : Cette règle est appliquée au connecteur ⊗ d'une formule dans la partie gauche du séquent. Elle est employée afin de représenter un séquent de manière plus brève (c'est-à-dire substituer le connecteur ⊗ entre deux atomes, entre deux formules, ou entre un atome et une formule dans la partie gauche par une virgule « , »). Son ordre de priorité est de 1 (la priorité la plus élevée), donc cette règle est toujours exécutée en priorité si possible.

$$\frac{\Gamma 1, \Gamma 2, \Gamma 3, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \otimes \Gamma 3, \Gamma 4 \vdash \Delta} \otimes L$$

- ⊕R : Cette règle est appliquée au connecteur ⊕ d'une formule dans la partie droite du séquent. Comme la partie droite exprime une liste des terminaisons que les auteurs veulent obtenir, dans le processus de preuve, nous choisissons chaque terminaison tour à tour, et puis vérifions indépendamment si la branche courante est valide ou non pour cette terminaison. Son ordre de priorité est de 2, donc cette règle est vérifiée juste après la règle ⊗L.

$$\frac{\Gamma \vdash \Delta 1}{\Gamma \vdash \Delta 1 \oplus \Delta 2} \oplus R \qquad \frac{\Gamma \vdash \Delta 2}{\Gamma \vdash \Delta 1 \oplus \Delta 2} \oplus R$$

- &L, ⊕L : Ces règles sont appliquées aux connecteurs &, ⊕ d'une formule dans la partie gauche du séquent. Il y a deux choix (dépendant du joueur ou du système de pilotage) à vérifier séparément. Les règles &L et ⊕L sont employées afin de diviser un séquent en sous-branches. Leur ordre de priorité est de 3 et 4 respectivement, donc ces règles sont vérifiées juste après la règle ⊕R.

$$\frac{\Gamma 1, \Gamma 2, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \,\&\, \Gamma 3, \Gamma 4 \vdash \Delta} \,\&L \qquad \frac{\Gamma 1, \Gamma 3, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \,\&\, \Gamma 3, \Gamma 4 \vdash \Delta} \,\&L$$

$$\frac{\Gamma 1, \Gamma 2, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \oplus \Gamma 3, \Gamma 4 \vdash \Delta} \oplus L \qquad \frac{\Gamma 1, \Gamma 3, \Gamma 4 \vdash \Delta}{\Gamma 1, \Gamma 2 \oplus \Gamma 3, \Gamma 4 \vdash \Delta} \oplus L$$

- ⊸R : Cette règle est appliquée au connecteur ⊸ d'une formule dans la partie droite du séquent. Même si ⊸R n'a pas de sens dans notre approche de modélisation, nous devons la vérifier car elle peut apparaître pendant le processus de preuve de séquent. Plus concrètement, elle peut être une conséquence de l'exécution de certaines implications linéaires complexes dans la partie gauche (telles que (A ⊸ B) ⊸ (C ⊸ D)). Son ordre de priorité est de 5, donc cette règle est vérifiée juste après la règle ⊕L.

$$\frac{\Delta 1, \Gamma \vdash \Delta 2}{\Gamma \vdash \Delta 1 \multimap \Delta 2} \multimap R$$

- ⊸L : Cette règle est appliquée au connecteur ⊸ d'une formule dans la partie gauche du séquent, ce qui a pour but d'exécuter la formule et nous donne deux séquents à démontrer en même temps. Son ordre de priorité est de 6, donc cette règle est vérifiée juste après la règle ⊸R.

$$\frac{\Gamma 1', \Gamma 4' \vdash \Gamma 2 \qquad \Gamma 1'', \Gamma 3, \Gamma 4'' \vdash \Delta}{\Gamma 1, \Gamma 2 \multimap \Gamma 3, \Gamma 4 \vdash \Delta} \multimap L$$

Afin de prouver le séquent Γ1, Γ2 ⊸ Γ3, Γ4 ⊢ Δ, on doit prouver les deux séquents Γ1', Γ4' ⊢ Γ2 et Γ1'', Γ3, Γ4'' ⊢ Δ. Ici Φ' est un sous-ensemble de Φ et Φ'' = Φ -

Φ', où Φ est Γ1 et Γ4 respectivement. Par exemple, si Γ1 = A, B (ses sous-ensembles sont {∅}, {A}, {B}, {A, B}), alors soit Γ1' = ∅ et Γ1'' = A, B ; soit Γ1' = A et Γ1'' = B ; soit Γ1' = B et Γ1'' = A ; soit Γ1' = A, B et Γ1'' = ∅. Même s'il y a plusieurs solutions à examiner à leur tour (en fonction de la combinaison entre Γ1'/Γ1'' et Γ4'/Γ4''), si le séquent original est prouvé, alors seulement une d'entre elles est réussie (les autres sont échouées). Par conséquent, on doit essayer chaque solution jusqu'à ce que : soit on trouve une solution valide (le séquent est prouvé), soit au contraire, toutes les solutions sont non valides (le séquent n'est pas prouvé). Prenons le séquent A, A ⊗ B ⊸ D, B ⊢ D, nous avons Γ1 = A, Γ2 = A ⊗ B, Γ3 = D, Γ4 = B, Δ = D, donc :

- soit Γ1' = ∅ et Γ1'' = A ; soit Γ1' = A et Γ1'' = ∅ ;
- soit Γ4' = ∅ et Γ4'' = B ; soit Γ4' = B et Γ4'' = ∅.

Ainsi, nous devons essayer les quatre solutions suivantes :

- ∅, ∅ ⊢ A ⊗ B et A, D, B ⊢ D (c'est-à-dire Γ1' = ∅, Γ4' = ∅, Γ1'' = A, Γ4'' = B) : Elle est non valide ;
- A, ∅ ⊢ A ⊗ B et ∅, D, B ⊢ D: Elle est non valide ;
- ∅, B ⊢ A ⊗ B et A, D, ∅ ⊢ D: Elle est non valide ;
- A, B ⊢ A ⊗ B et ∅, D, ∅ ⊢ D: Elle est valide.

Comme nous avons trouvé une solution valide, le séquent original A, A ⊗ B ⊸ D, B ⊢ D est prouvé.

- ⊗R : Cette règle est appliquée au connecteur ⊗ d'une formule dans la partie droite du séquent. Son exécution est similaire à celle de la règle ⊸L ci-dessus, et elle est seulement vérifiée après l'exécution de toutes les règles ⊸L possibles, donc son ordre de priorité est de 7.

$$\frac{\Gamma' \vdash \Delta 1 \qquad \Gamma'' \vdash \Delta 2}{\Gamma \vdash \Delta 1 \otimes \Delta 2} \otimes R$$

- Une preuve consiste à réécrire un séquent successivement en exécutant, à chaque étape, la règle la plus prioritaire parmi les règles décrites précédemment (si la règle ⊸L est exécutée, l'ordre de priorité de l'événement/action correspondant doit être le plus petit) jusqu'à ce que :
  - chaque feuille de l'arbre de preuve ne contienne que des séquents identité → le séquent est prouvé, et donc la preuve courante est réussie ;
  - la preuve ne puisse plus progresser → le séquent n'est pas prouvé et donc la preuve courante a échoué.

Pour que les lecteurs se représentent plus facilement la solution proposée, nous examinons à titre d'exemple le séquent A ⊗ ((C ⊸ D) ⊸ B) ⊗ (C ⊸ D) ⊢ A ⊗ B. La preuve « à la main » de ce séquent selon la version complète du calcul des séquents est donnée dans la Figure III.4. Le séquent à démontrer est écrit dessous le trait de séparation tandis que le (les) séquent(s) utilisé(s) pour cette preuve est (sont) écrit(s) au dessus. On lit la preuve de bas en haut (en partant de la conclusion et en remontant). À la première étape, même si les deux règles ⊗L et

⊗R sont applicables, car la règle ⊗L (son ordre de priorité est de 1) est plus prioritaire que la règle ⊗R (son ordre de priorité est de 7), nous appliquons la règle ⊗L. Par conséquent, la première ligne revient à substituer les connecteurs ⊗ à la partie gauche du séquent, par les virgules « , », en utilisant la règle ⊗L. Ensuite, la formule d'implication linéaire (C ⊸ D) ⊸ B se réécrit en utilisant la règle la plus prioritaire ⊸L, ce qui nous donne deux séquents à démontrer en même temps C ⊸ D ⊢ C ⊸ D et A, B ⊢ A ⊗ B. Les étapes suivantes sont expliquées de manière similaire. La preuve se termine lorsque chaque feuille de l'arbre de preuve ne contient que des séquents identité. Dans ce cas la preuve est réussie.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{C \vdash C}\,id \quad \dfrac{}{D \vdash D}\,id}{C, \mathbf{C \multimap D} \vdash D}\,\multimap L}{C \multimap D \vdash \mathbf{C \multimap D}}\,\multimap R \qquad \dfrac{\dfrac{}{A \vdash A}\,id \quad \dfrac{}{B \vdash B}\,id}{A, B \vdash \mathbf{A \otimes B}}\,\otimes R}{A, \mathbf{((C \multimap D) \multimap B)}, (C \multimap D) \vdash A \otimes B}\,\multimap L}{\mathbf{A \otimes ((C \multimap D) \multimap B) \otimes (C \multimap D)} \vdash A \otimes B}\,\otimes L$$

Figure III.4. La preuve « à la main » du séquent A ⊗ ((C ⊸ D) ⊸ B) ⊗ (C ⊸ D) ⊢ A ⊗ B, qui est basée sur la version complète du calcul des séquents.

Ainsi, à travers le processus d'examen d'un séquent avec les événements/actions complexes utilisant la version complète du calcul des séquents décrit ci-dessus, on peut parcourir tous les discours possibles du scénario correspondant. Par conséquent, il nous est possible d'avoir l'arbre des preuves ou de manière identique des exécutions du scénario. L'analyse de cet arbre peut donner les informations importantes concernant le scénario telles que : les branches réussies/échouées, les branches réussies pour chaque sortie et les situations inaccessibles. Cette analyse permet donc de vérifier deux propriétés : *Accessibilité* et *Absence de blocage*. Nous présentons en détail, comment cette vérification est effectuée avec l'aide des outils que nous avons développé, dans le chapitre suivant.

III.6. Conclusion

Nous avons présenté, dans ce chapitre, un modèle d'histoire en logique linéaire, qui autorise la validation lors de la phase de conception d'un jeu et dont l'exécution comme processus de contrôle dans la phase d'exécution. Les propriétés de narration énoncées spécifient la qualité d'un scénario de jeu. Elles sont classées en deux catégories : les propriétés liées au déroulement des discours et les propriétés liées à l'effet dramatique créé par chacun des discours pour le joueur.

Afin de vérifier la satisfaction de ces propriétés pour un scénario, nous avons donné un calcul des séquents. La phase de conception comporte deux étapes : (1) modéliser le scénario en logique linéaire, (2) parcourir et analyser tous ses discours possibles grâce au calcul des séquents.

Ces travaux apportent aux travaux précédents menés au sein du laboratoire L3I, une extension des capacités d'expression du modèle et une richesse plus importante des propriétés validées. Ainsi nous avons aussi ajouté la notion d'ordre de priorité aux événements/actions. Ces extensions nous ont conduits à développer de nouveaux algorithmes de calcul de séquent.

Le pouvoir d'expression des auteurs est renforcé par l'ajout des propriétés observables. Ainsi, à part les deux propriétés de « Correspondance d'émotion du joueur » et d'« Évolution de la tension dramatique », toutes les propriétés énoncées peuvent être validées sur le modèle proposé.

Grâce aux résultats obtenus par cette évaluation, les auteurs de scénarios peuvent corriger la modélisation afin d'obtenir un nouveau scénario et ensuite refaire l'analyse si nécessaire (dans le cas où la qualité du scénario courant n'est pas suffisamment bonne). Ce processus est répété dans la phase de construction de scénario d'une histoire jusqu'à ce que les auteurs obtiennent un scénario, qui est le plus pertinent relativement à leurs objectifs, en d'autres termes, jusqu'à ce que les auteurs trouvent sa qualité est satisfaisante (que le scénario est valide). Ce scénario validé est ensuite employé comme entrée d'un système de pilotage du jeu, et donc permet au système de pilotage de diriger le déroulement du jeu selon le scénario valide produit.

Le chapitre suivant présente la démarche et le système auteur que nous avons construit, dont l'objectif est d'aider l'utilisateur à exécuter la solution de logique linéaire proposée plus haut, pour la production d'un scénario de bonne qualité d'une histoire, même s'il n'a aucune connaissance en logique linéaire.

CHAPITRE IV - Méthodes et Outils

IV.1. Introduction

Nous présentons dans ce chapitre, après avoir précisé la notion de système auteur dans le contexte de production de scénarios de pilotage, une démarche et un ensemble d'outils que peut mettre en œuvre un auteur. Dans le cadre de l'architecture mise en œuvre au laboratoire, l'auteur produit un modèle de scénario qui définit les contraintes du système de pilotage d'une application interactive scénarisée.

Nous avons présenté au chapitre III la validation d'un scénario et un ensemble de propriétés. Le modèle validé comporte l'expression en logique linéaire du scénario. Cette formalisation autorise l'analyse du modèle. Ce chapitre, répond à la question de la mise en œuvre de cette méthodologie dans le cadre de la définition d'un système de pilotage valide par rapport à un ensemble de contraintes de qualité.

Pour cela, nous devons identifier le processus nominal de production d'un système de pilotage de narration interactive. Ce processus fait intervenir trois acteurs différents avec des compétences diverses : l'auteur, le game designer et l'informaticien. L'auteur est en charge de définir les différents scénarios, l'histoire, les rebondissements et les objectifs en termes de qualité de narration. Le game designer donne des spécifications concernant la conception du jeu telle que l'architecture, les données, l'environnement du jeu, *etc*., afin de mettre en œuvre l'histoire et le scénario défini par l'auteur. L'informaticien fournit le support technique pour réaliser/fabriquer l'application correspondante selon la conception produite par le game designer.

Ainsi, l'acteur de la phase d'écriture de scénario est l'auteur. Les outils permettent à un non spécialiste de la logique linéaire d'écrire (en logique linéaire) un modèle correspondant au scénario du jeu. Ils comportent la fonctionnalité d'analyse de sa qualité. L'utilisation itérative des possibilités d'écriture et d'analyse permettent l'amélioration du scénario. Dans ce chapitre, nous présentons le processus de développement et les outils développés pour supporter le système auteur permettant d'écrire des scénarios de bonne qualité, qui sont exprimés par les séquents de logique linéaires, et qui sont ensuite utilisés par un système de pilotage des applications interactives scénarisées.

En fin de chapitre, nous décrivons un prototype de système de pilotage que nous avons développé. Le fonctionnement utilise la réécriture selon les règles de la logique linéaire. Les scénarios/séquents valides résultant du système auteur sont les entrées du système de pilotage. Le mécanisme de déduction automatique des séquents constitue le principe de fonctionnement du système de pilotage. Par ailleurs, comme de nombreux outils de gestion de scénarios interactifs utilisent des modèles d'intelligence artificielle écrits en PDDL (Planning Domain Definition Language [40]), nous présentons une passerelle qui, grâce au système auteur que nous avons développé, produit un modèle valide d'une narration interactive exprimé en PDDL.

IV.2. Système auteur

Un système auteur dans le domaine de narration interactive est un système aidant un auteur/utilisateur à produire des contenus narratifs qui peuvent être utilisés pour des applications interactives. Plusieurs systèmes auteurs ont été présentés ces dernières années tels que Scribe [71], EmoEmma [78], ENIGMA [59, 60], DraMachina [35], Art-E-Fact [53], Scenejo [98], StoryTec [45] (voir la Section II.3, chapitre II - Production des scénarios valides). Le point commun à ces systèmes est l'utilisation de langages graphiques (interfaces, textes, nœuds, liens, interactions, *etc*.) grâce auxquels, un utilisateur peut créer des contenus narratifs.

Quelques systèmes auteurs seulement (tels que Scribe, EmoEmma, ENIGMA) fournissent une capacité de débogage (*debugging*) permettant à l'utilisateur de prévenir, détecter et corriger les erreurs dans les scénarios interactifs créés. Cependant, la capacité de débogage de ces systèmes, qui est fondée sur les traces d'exécution d'un scénario, est insuffisante pour produire une analyse exaustive dans le cas où le nombre de discours contenus dans le scénario est important.

Comme nous l'avons indiqué dans la Section II.3.2, chapitre II - Analyse structurelle d'un scénario, les travaux, s'appuyant sur l'analyse structurelle, permettent à l'utilisateur d'analyser au niveau structurel tous les discours possibles d'un scénario, et donc permettent de surmonter l'obstacle ci-dessus. Toutefois, leur usage est complexe pour l'utilisateur. Seul KANAL [56] offre un usage facile mais plusieurs spécifications clés de la narration interactive ne peuvent pas être abordées. Ainsi, un système auteur qui permet une analyse structurelle d'un scénario est nécessaire.

Dans le chapitre III, nous avons proposé un ensemble de propriétés de narration, un modèle formel d'histoire utilisant la logique linéaire et des algorithmes de validation mettant en œuvre une analyse structurelle. Leur mise en œuvre permet de produire un scénario en s'assurant de sa qualité. Ainsi, nous avons besoin de réaliser un système auteur dont l'objectif est d'aider un utilisateur à implémenter la solution proposée au chapitre III. Plus concrètement, avec l'aide du système auteur, l'utilisateur peut créer un modèle en logique linéaire exprimant le scénario d'une histoire, et puis analyser, au niveau structurel, ce modèle au regard des propriétés de narration. Cette approche vise à produire et évaluer des scénarios de qualité pour des auteurs n'ayant pas la connaissance des modèles formels utilisés en particulier n'ayant pas de connaissance en logique linéaire. Ce modèle, exprimé en logique linéaire, est ensuite employé, au cours du fonctionnement, comme l'entrée d'un système de pilotage de narration interactive assurant le contrôle de la gestion du déroulement du jeu.

Pour cela, le système auteur que nous développons répond aux contraintes suivantes :

- offrir à l'utilisateur un langage graphique approprié à la modélisation d'une histoire selon le modèle en logique linéaire proposé au chapitre III ;
- fournir à l'utilisateur un langage graphique pertinent en vue d'exprimer les contraintes de l'histoire ;
- permettre de représenter tous les éléments nécessaires de l'histoire :

    - les états de l'histoire et des personnages joueur/non-joueur, les états disponibles au moment initial de l'histoire
    - les entrées du joueur
    - les événements/actions et leur ordre de priorité
    - les choix entre les événements/actions qui sont décidés par le système de pilotage ou par le joueur, le pourcentage de chaque possibilité de choix pour les choix décidés par le système de pilotage
    - les sorties de l'histoire
    - la structure de discours de l'histoire ;
- d'une part permettre la construction automatiquement du graphe des preuves qui représente tous les discours possibles du scénario modélisé ;
- et d'autre part permettre de mener l'analyse automatique au niveau structurel pour les informations importantes concernant le scénario telles que : éléments de modélisation (états, entrées, événements/actions, sorties) inutilisés, discours réussis/échoués, événements/actions de chaque discours, états disponibles de chaque discours après chaque étape, nombre d'événements dramatiques dans chaque discours, nombre de catégories d'événements dramatiques dans chaque discours, et séquence des attributs d'effet dramatique créés pour le joueur pendant le déroulement de chaque discours.

Par ailleurs, puisque le modèle/séquent de logique linéaire exprimant le scénario valide produit est ensuite utilisé comme l'entrée du système de pilotage, sa représentation doit être exécutable. Pratiquement, notre système auteur doit permettre de générer une représentation de ce modèle automatiquement. Nous utilisons une représentation sous forme XML du séquent de logique linéaire. Pour ce faire, nous proposons un métamodèle du calcul des séquents.

Dans la section ci-dessous, nous décrivons en détail les modèles pour le système auteur que nous développons.

IV.3. Modèles pour le système auteur

Afin de répondre aux contraintes susmentionnées, notre système auteur implémente les modèles et les algorithmes suivants :

- un modèle concret d'histoire qui ne nécessite pas de connaissances en logique linéaire (pour être plus bref, désormais nous l'appelons « modèle d'histoire »),
- un modèle de contraintes exprimant les propriétés du scénario modélisé,
- un algorithme de parcours de toutes les preuves d'un modèle d'histoire,
- un modèle concret de scénario exécutable lors du fonctionnement d'un système de pilotage (pour être plus bref, désormais nous l'appelons « modèle de scénario exécutable »), et
- un algorithme de transformation d'un modèle d'histoire en un modèle de scénario exécutable.

Nous présentons tour à tour ci-dessous ces modèles (les deux algorithmes sont présentés dans la Section IV.5 de ce chapitre).

IV.3.1. Modèle d'histoire

Dans le chapitre III, nous avons donné un modèle abstrait d'histoire employant les éléments fournis par la logique linéaire. Pour qu'un utilisateur « auteur » puisse produire un modèle d'histoire sur lequel le système auteur peut opérer, nous proposons, dans cette section, un modèle concret d'histoire qui, bien qu'étant fondé sur le modèle abstrait d'histoire, ne nécessite pas, pour sa mise en œuvre, de connaissances en logique linéaire. Ce modèle est conforme au métamodèle donné dans la Figure IV.1, où un scénario est composé de six listes non-ordonnées d'éléments : états, entrées, événements/actions, choix, sorties et structure de discours (dans lesquelles celles d'états, d'événements/actions et de sorties sont obligatoires).

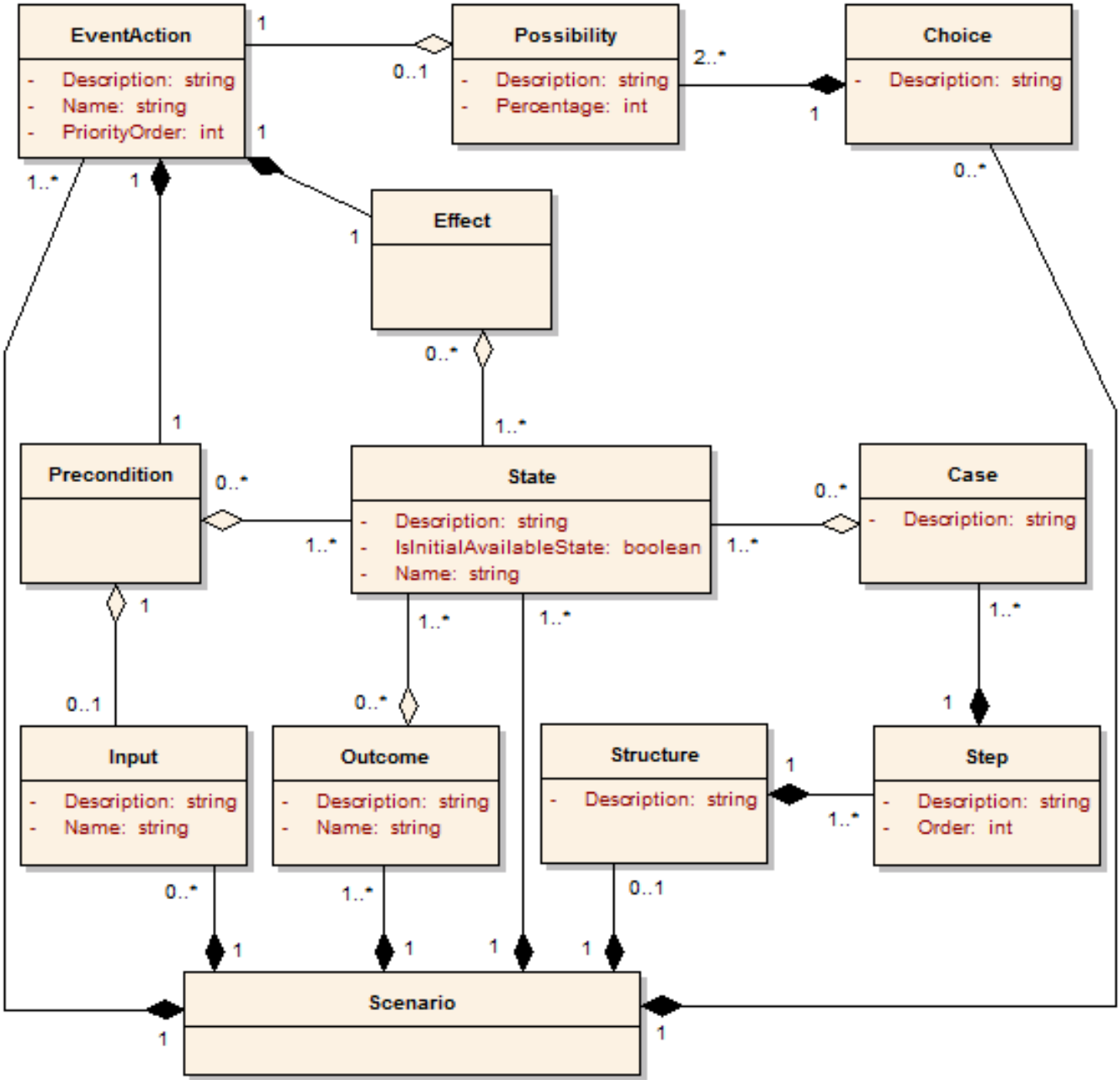


Figure IV.1. Métamodèle d'histoire qui ne nécessite pas de connaissances en logique linéaire.

Voici ci-dessous la description détaillée de ce métamodèle d'histoire (pour mieux comprendre, voir les Sections III.3 – Modélisation d'une histoire en logique linéaire et III.4 – Formalisation des propriétés de narration, chapitre III) :

- Les états, les entrées, les événements/actions et les sorties se distinguent entre eux par l'attribut *Name*, l'attribut *Description* autorise une description plus complète (l'objectif de l'attribut *Description* dans les autres éléments est similaire). Le statut d'état initial est indiqué par l'attribut *IsInitalAvailableState = True/False* (sa valeur par défaut est *False*). Une sortie est composée d'un ou plusieurs états. Un événement/action comprend une *Précondition* et un *Effet* contenant un ou plusieurs états ; si l'événement/action est exécuté par un choix du joueur, alors sa *Précondition* doit posséder une entrée. La priorité d'exécution des événements/actions, qui aide les utilisateurs à mieux contrôler le déroulement de l'histoire, est indiquée par l'attribut *PriorityOrder*.
- Un choix dans le scénario est composé d'au moins deux possibilités, chaque possibilité est reliée à un événement/action. L'utilisateur peut aussi attribuer un pourcentage à chacune des possibilités d'un choix si nécessaire.
- La structure de discours d'un scénario est définie par un ensemble ordonné des étapes. Une étape peut posséder un ou plusieurs cas. Un cas est spécifié par un ou plusieurs état(s).

Ainsi, un modèle d'histoire fondé sur ce métamodèle est caché toutes les connaissances en logique linéaire définies dans le modèle abstrait d'histoire donné dans le chapitre III. Concrètement :

- les atomes sont remplacés par les états et les entrées ;
- les formules $\otimes$ sont remplacées par les relations « Aggregate » (par exemple, la relation « Aggregate 1..* » entre *Précondition/Effet* et *État*…)
- les formules $\multimap$ sont remplacées par les événements/actions composés de *Précondition* et *Effet* ;
- les formules $\oplus$ et & dans la partie gauche du séquent sont remplacées par la liste de choix (si les événements/actions, reliés à un choix, comportent les entrées, alors ce choix est décidé par le joueur ; sinon, ce choix est décidé par le système de pilotage) ;
- la formule $\oplus$ dans la partie droite du séquent est remplacée par la relation « Compose 1..* » entre *Scénario* et *Sortie* ;
- le séquent est remplacé par *Scénario* composé de six listes d'éléments : états, entrées, événements/actions, choix, sorties et structure de discours.

IV.3.2. Modèle de contraintes

Dans le chapitre III, nous avons donné un ensemble de propriétés de narration caractérisant un scénario. Parmi celles-ci, les propriétés *Complexité*, *Satisfaction d'événements clés*, *Séquencement*, *Fréquence des événements dramatiques*, *Variété d'effets dramatiques*, et *Équilibre de l'effet dramatique* ont besoin d'être attribuées les paramètres pour pouvoir vérifier leur validité.

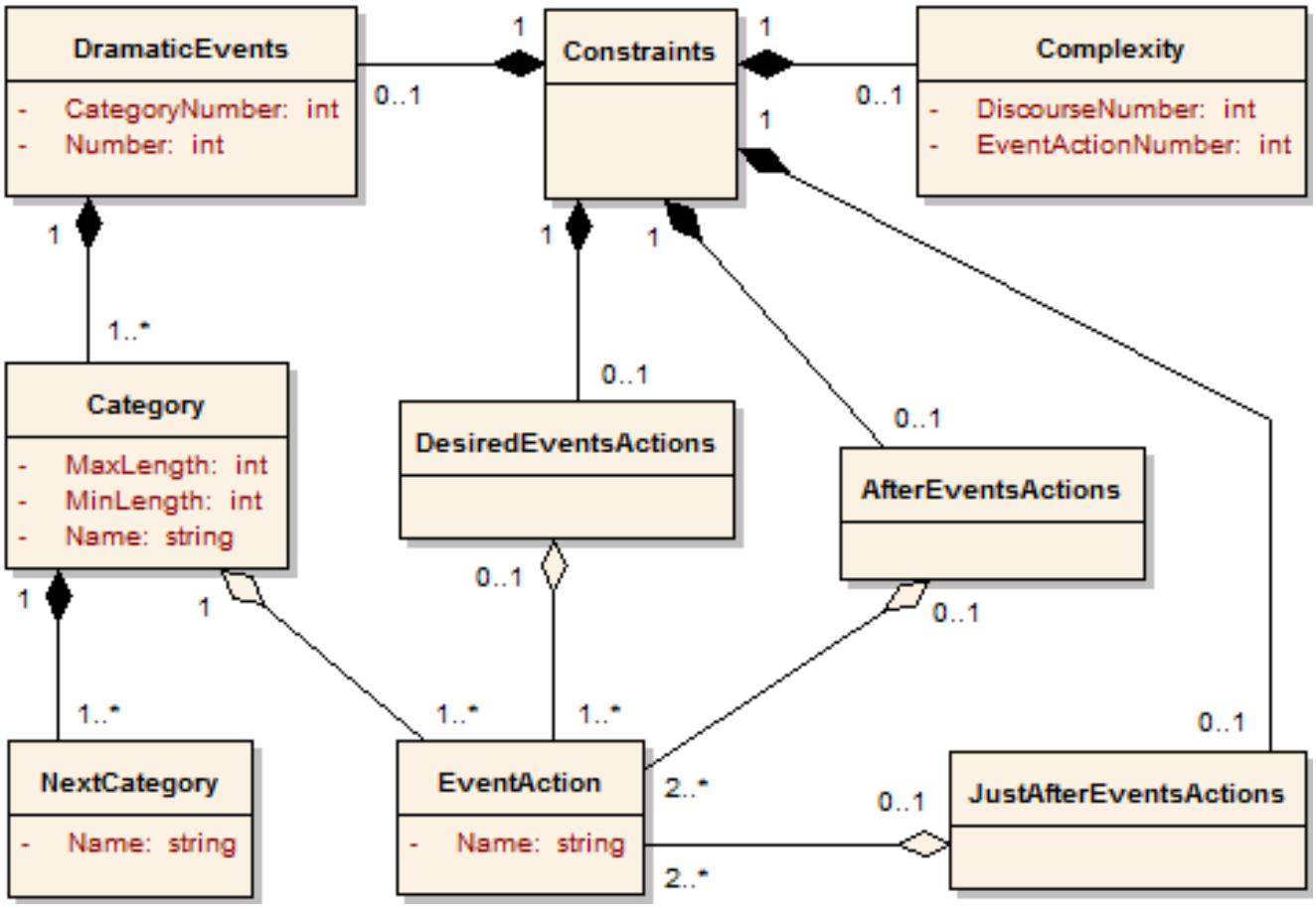


Figure IV.2. Métamodèle exprimant les contraintes d'un scénario.

Nous proposons, dans cette section, un modèle de contraintes exprimant ces paramètres. Ce modèle est conforme au métamodèle donné dans la Figure IV.2 (pour mieux comprendre, voir Section III.4, chapitre III – Formalisation des propriétés de narration), où :

- Complexity : Ce sont les paramètres utilisés afin de vérifier la propriété *Complexité*
  - DiscourseNumber : C'est le nombre minimum de discours possibles dans le scénario
  - EventActionNumber : C'est le nombre minimum d'événements/actions qu'un discours doit posséder.
- DesiredEventsActions : C'est un ensemble non-ordonné des événements/actions que tous les discours doivent comporter, ce paramètre est utilisé afin de vérifier la propriété *Satisfaction d'événements clés*.
- JustAfterEventsActions : C'est un ensemble ordonné des événements/actions consécutifs que tous les discours doivent comporter, ce paramètre est utilisé afin de vérifier la propriété *Séquencement*.
- AfterEventsActions : C'est un ensemble ordonné des événements/actions non-consécutifs que tous les discours doivent comporter, ce paramètre est utilisé afin de vérifier la propriété *Séquencement*.
- DramaticEvents : Ce sont les paramètres utilisés afin de vérifier les propriétés liées à l'effet dramatique créé par chaque discours

- Number : C'est le nombre minimum d'événements dramatiques qu'un discours doit posséder, ce paramètre est utilisé afin de vérifier la propriété *Fréquence des événements dramatiques*.
- CategoryNumber : C'est le nombre minimum de catégories d'événements dramatiques qu'un discours doit posséder, ce paramètre est utilisé afin de vérifier la propriété *Variété d'effets dramatiques*.
- Category : Ce sont les informations concernant les catégories d'effet dramatique définies dans le scénario, chaque catégorie peut être attribuée à plusieurs événements/actions. Ces paramètres sont utilisés afin de vérifier la propriété *Équilibre de l'effet dramatique*.
    - Name : Nom de la catégorie d'effet dramatique (tel que *Stress*, *Fierté*, *Bonheur*, *Neutre*...)
    - MinLength : Nombre minimum d'événements/actions consécutifs qui causent cet effet dramatique (la durée minimale de cet effet dramatique)
    - MaxLength : Nombre maximum d'événements/actions consécutifs qui causent cet effet dramatique (la durée maximale de cet effet dramatique)
    - NextCategory : Ensemble de catégories d'effet dramatique suivantes de cet effet dramatique (la catégorie de l'effet dramatique suivant de cet effet dramatique doit être un élément dans l'ensemble).

IV.3.3. Modèle de scénario exécutable

Puisque le modèle/séquent de logique linéaire exprimant le scénario valide, produit après le processus de validation de scénario, est ensuite utilisé comme l'entrée d'un système de pilotage, sa représentation doit permettre un calcul des séquents en temps réel. Notre modèle d'histoire décrit précédemment ne répond pas à cette condition, car sa notion « séquent » est implicite (qui est remplacée par *Scénario* composé de six listes d'éléments : états, entrées, événements/actions, choix, sorties et structure de discours).

Nous proposons, dans cette section, un modèle de scénario dans lequel, le séquent correspondant à un scénario d'une histoire est explicitement défini. Ce modèle est conforme à un métamodèle du calcul des séquents. Ainsi, un scénario, exprimé par un tel modèle, est exécutable lors du fonctionnement d'un système de pilotage de narration interactive grâce au mécanisme de déduction automatique du séquent.

Ce métamodèle est donné dans la Figure IV.3 selon lequel un séquent est composé de deux parties :

- La partie droite d'un séquent comporte un ensemble de sorties, chacune des sorties comprend un ou plusieurs états. Les sorties et les états se distinguent entre eux par l'attribut *Name*.
- La partie gauche d'un séquent est composée d'un ensemble d'états disponibles et d'un ensemble de formules (au moment initial du jeu, ils correspondent aux états disponibles initiaux et aux formules initiales du séquent respectivement). Ces ensembles sont modifiés à chaque étape du calcul pendant le déroulement du jeu.

Cette modification est une réécriture selon les règles d'inférence de la logique linéaire. Les formules dans la partie gauche se distinguent par leur numéro d'ordre. Celui-ci indique leur position dans la partie gauche du séquent. Chacune des formules comprend au moins trois éléments (atome, connecteur, atome), chaque élément possède 8 attributs, parmi eux seulement 3 attributs sont obligatoires : *Type, Name* et *Order* (le numéro d'ordre des éléments exprime leur position dans la formule). Afin de représenter les séquents de logique linéaire dans notre approche, nous utilisons sept types d'éléments :

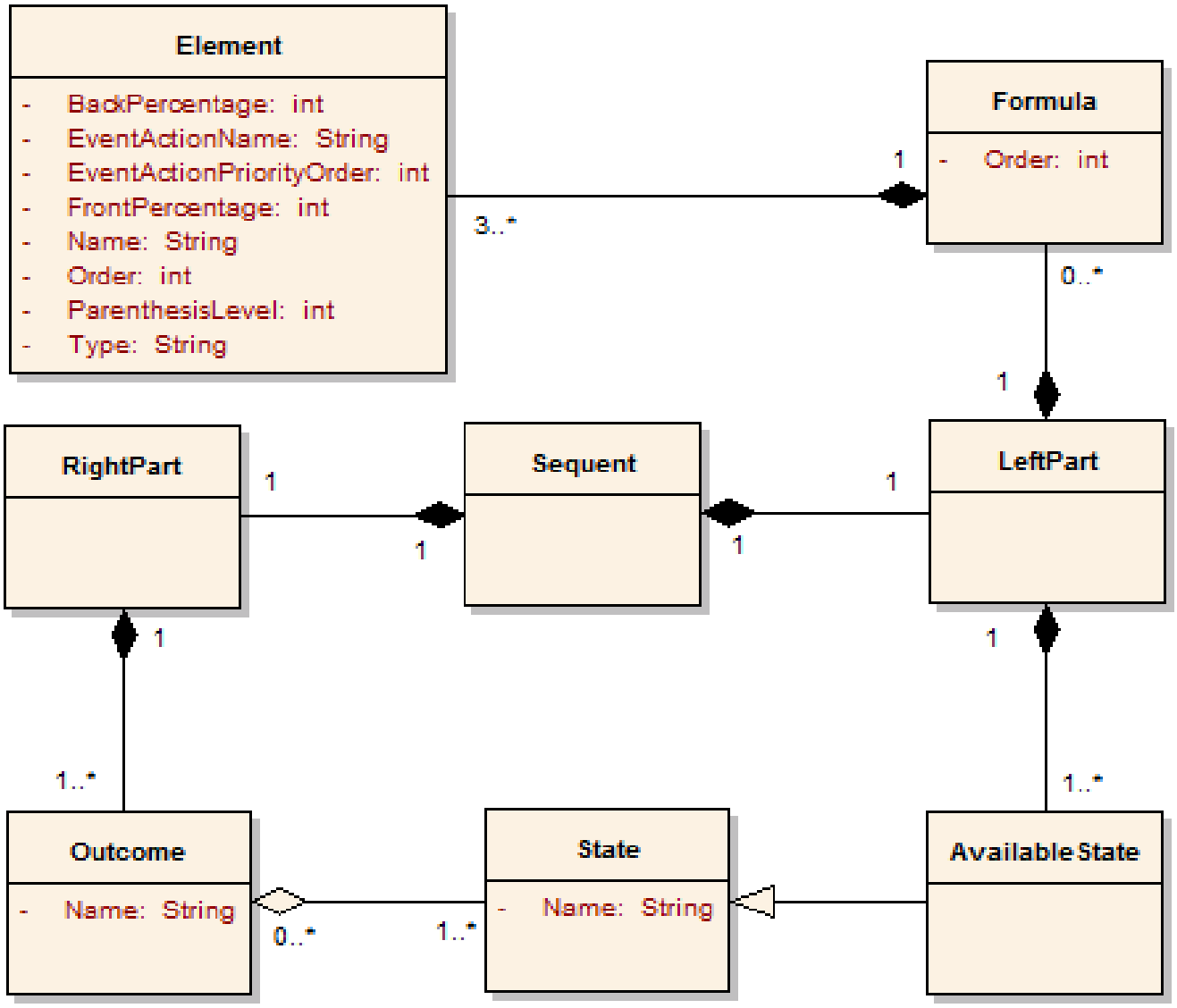


Figure IV.3. Métamodèle de scénario conforme au calcul des séquents.

- Type = "Open Parenthesis" ou "Close Parenthesis", Name = "(" ou ")", ParenthesisLevel (= 1, 2, 3…) est le niveau de la parenthèse dans la formule (le niveau des parenthèses les plus extérieures est de 1, des parenthèses qui sont les enfants directs suivants est de 2, *etc*.).

- Type = “Linear Implication”, Name = “imply” : Cet élément correspond au connecteur ⊸ d’un événement/action, ses attributs *EventActionName* et *EventActionPriorityOrder* sont le nom et l’ordre de priorité de cet événement/action dans le jeu.
- Type = “Atom” : Ces éléments expriment les états ou les entrées du jeu qui sont présents dans les formules, donc leur attribut *Name* contient le nom de ces états ou entrées.
- Type = “Multiplicative Conjunction”, Name = “times” : Cet élément correspond au connecteur ⊗, par exemple, pour Ik ⊗ (Pi ⊗ Ik ⊸ Pk), où le nom de l’événement/action Pi ⊗ Ik ⊸ Pk est EA02, son ordre de priorité est de 2, on a

```
<Element Name="Ik" Order="1" Type="Atom"/>
<Element Name="times" Order="2" Type="Multiplicative Conjunction"/>
<Element Name="(" Order="3" Type="Open Parenthesis" ParenthesisLevel="1"/>
<Element Name="Pi" Order="4" Type="Atom"/>
<Element Name="times" Order="5" Type="Multiplicative Conjunction"/>
<Element Name="Ik" Order="6" Type="Atom"/>
<Element Name="imply" Order="7" Type="Linear Implication"
EventActionName="EA02" EventActionPriorityOrder="2"/>
<Element Name="Pk" Order="8" Type="Atom"/>
<Element Name=")" Order="9" Type="Close Parenthesis" ParenthesisLevel="1"/>
```

- Type = “Additive Conjunction”, Name = “with” : Cet élément correspond au connecteur &.
- Type = “Additive Disjunction”, Name = “plus” : Cet élément correspond au connecteur ⊕. Puisqu’un élément de ce type est toujours entre deux composants d’une formule ⊕, si à ces deux composants sont attribués un pourcentage, alors les attributs *FrontPercentage* et *BackPercentage* d’un élément « plus » contiennent respectivement le pourcentage de ses composants « amont » et « aval » correspondants.

IV.3.4. Bilan

Dans cette section, nous avons présenté le modèle d’histoire, le modèle de contraintes et le modèle de scénario exécutable, qui sont implémentés par notre système auteur. Le modèle d’histoire permet à l’utilisateur de modéliser une histoire même s’il n’a aucune connaissance en logique linéaire. Le modèle de contraintes permet d’exprimer les propriétés du scénario modélisé. Le modèle de scénario exécutable permet de représenter un scénario de jeu de façon opérationnel de sorte qu’il puisse être exécutable lors du fonctionnement d’un système de pilotage.

Le système auteur emploie l’algorithme de parcours de toutes les preuves pour analyser la satisfaction du scénario modélisé selon notre modèle d’histoire par rapport aux propriétés de narration. Une fois que ce scénario est valide, le système auteur utilise l’algorithme de transformation de modèle afin de transformer le modèle d’histoire validé en un modèle de

scénario exécutable dont la représentation XML est conforme au métamodèle du calcul des séquents. Ces deux algorithmes sont présentés dans la Section IV.5 de ce chapitre.

Dans la section suivante, nous décrivons en détail la démarche grâce à laquelle un utilisateur, avec l'aide de notre système auteur, peut produire un modèle d'histoire valide, dont le scénario est exprimé par un séquent de la logique linéaire et dont la qualité, au regard des propriétés de l'histoire souhaitées par l'auteur, est garantie. Ce scénario peut être exécuté par un système de pilotage lors du fonctionnement d'une application interactive « scénarisée ». Cette démarche s'intègre dans un processus de production d'un système de pilotage de narration interactive énoncé en section *Introduction* de ce chapitre.

IV.4. Démarche utilisateur

Nous proposons, dans cette section, une démarche qu'un utilisateur peut mettre en œuvre, afin de produire un modèle valide et exécutable d'un scénario d'une histoire. Cette démarche est liée au système auteur que nous développons, ce qui implémente les modèles et les algorithmes mentionnés dans la section précédente.

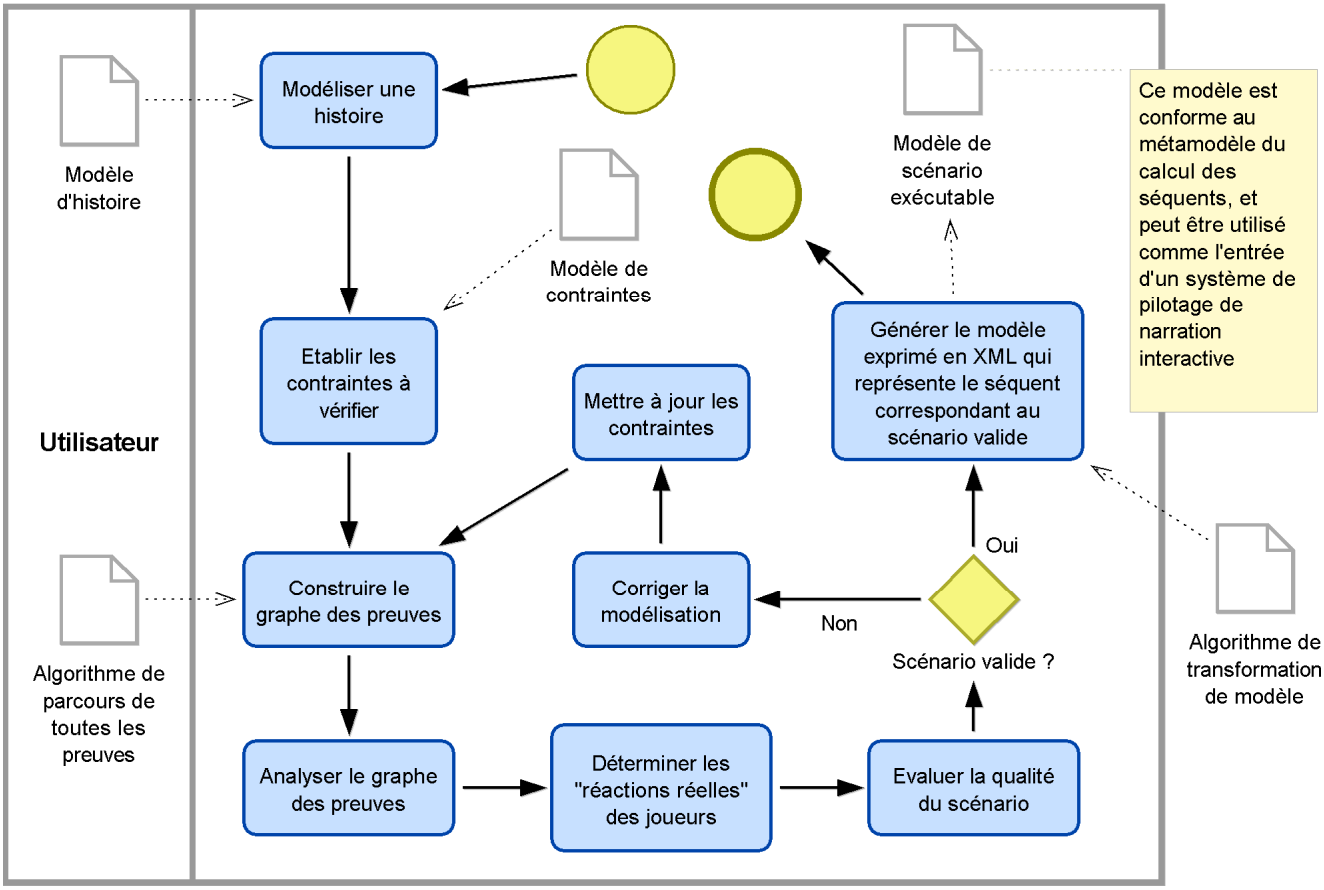


Figure IV.4. Démarche suivie pour produire un scénario de bonne qualité (construite selon la norme BPMN – Business Process Modeling Notation [104]).

Voici ci-dessous les étapes de la démarche utilisant notre système auteur (voir Figure IV.4) :

- **Etape 1 – Modéliser une histoire :** L'utilisateur (auteur, expert du domaine) emploie le système auteur en vue de modéliser une histoire (modéliser ses états et ses sorties souhaitées, les états et les choix des personnages joueur/non-joueur, les événements/actions, *etc*.) selon notre modèle d'histoire. Le résultat obtenu est un modèle exprimé en XML représentant le scénario modélisé.
- **Etape 2 – Etablir les contraintes à vérifier :** L'utilisateur emploie le système auteur en vue de préciser les paramètres pour pouvoir vérifier quelques propriétés de narration selon notre modèle de contraintes (cela correspond à l'établissement de contraintes à vérifier pour le scénario modélisé).
- **Etape 3 – Construire le graphe des preuves :** À partir du modèle exprimé en XML représentant le scénario modélisé, obtenu après l'étape 1, l'utilisateur emploie le système auteur, qui implémente l'algorithme de parcours de toutes les preuves, en vue de construire le graphe des preuves. Ce graphe est la représentation de tous les discours possibles du scénario.
- **Etape 4 – Analyser le graphe des preuves :** Afin de vérifier si le scénario satisfait aux propriétés de narration, l'utilisateur emploie le système auteur pour analyser le graphe des preuves. Le résultat se compose d'informations qualitatives et statistiques sur les aspects suivants du scénario (pour mieux comprendre, voir Section III.4, chapitre III – Formalisation des propriétés de narration) :
    - les éléments de modélisation (états, entrées, événements/actions, sorties) inutilisés : ces informations sont utilisées afin de vérifier la propriété *Accessibilité* ;
    - les discours échoués : ces informations sont utilisées afin de vérifier la propriété *Absence de blocage* ;
    - le nombre d'événements/actions dans chaque discours et le nombre de discours dans le scénario : ces informations sont utilisées afin de vérifier la propriété *Complexité* ;
    - la liste des événements/actions dans chaque discours : ces informations sont utilisées afin de vérifier la propriété *Satisfaction d'événements clés* ;
    - les états disponibles de chaque discours après chaque étape : ces informations sont utilisées afin de vérifier la propriété *Satisfaction de la structure* ;
    - l'ordre d'apparition des événements/actions dans chaque discours : ces informations sont utilisées afin de vérifier la propriété *Séquencement* ;
    - le nombre d'événements dramatiques dans chaque discours : ces informations sont utilisées afin de vérifier la propriété *Fréquence des événements dramatiques* ;
    - le nombre de catégories d'événements dramatiques dans chaque discours : ces informations sont utilisées afin de vérifier la propriété *Variété d'effets dramatiques* ;
    - la séquence des attributs d'effet dramatique créés pour le joueur pendant le déroulement de chaque discours : ces informations sont utilisées afin de vérifier la propriété *Équilibre de l'effet dramatique*.

- **Etape 5 – Déterminer les « réactions réelles » des joueurs :** En exécutant le scénario, l'utilisateur peut aussi obtenir, au moyen de sondages et automatiquement au moyen de capteurs d'émotion [77], les « réactions réelles » (ou la connaissance sensorielle) des joueurs pour le scénario, afin de répondre à d'autres questions telles que :
    - Est-ce que le joueur perçoit exactement les émotions liées aux événements dramatiques (cette question utilisée en vue de vérifier la propriété de narration *Correspondance d'émotion du joueur*) ?
    - Quelle est l'évolution de la tension dramatique, qui est causée par les discours, pour le joueur (cette question utilisée afin de vérifier la propriété de narration *Évolution de la tension dramatique*) ?
    - Est-ce que le scénario fournit suffisamment d'options pertinentes au joueur ? Est-ce que le joueur se sent contraint par le déroulement du scénario ?
    - Est-ce que le scénario est cohérent/raisonnable ?
    - Comment le joueur estime-t-il le scénario (intéressant, normal, monotone…) ?
    - Quels sont les points à améliorer du scénario ?
    - …
- **Etape 6 – Evaluer la qualité du scénario :** Grâce aux résultats obtenus après les étapes 3, 4 et 5, il est possible pour l'utilisateur d'avoir rapidement une vue globale sur le scénario courant, et donc d'évaluer sa qualité.
    - Si le scénario n'est pas encore valide (en d'autres termes, l'utilisateur trouve que sa qualité n'est pas encore satisfaisante ou le scénario courant n'est pas encore pertinent au regard de ses objectifs), l'utilisateur peut corriger la modélisation afin de produire un nouveau modèle d'histoire correspondant à un nouveau scénario (ainsi que peut mettre à jour ses contraintes à vérifier), et ensuite peut refaire les étapes 3, 4, 5, 6 de la démarche. Ce processus est répété jusqu'à ce que le scénario modélisé devienne valide.
    - Si le scénario est valide, l'utilisateur emploie le système auteur, qui implémente l'algorithme de transformation de modèle, pour générer le modèle exprimé en XML qui représente le séquent de logique linéaire correspondant au scénario valide. Ce modèle est conforme au métamodèle du calcul des séquents, il est donc exécutable lors du fonctionnement d'un système de pilotage de narration interactive grâce au mécanisme de déduction automatique du séquent.

Dans la section suivante, nous donnons une présentation modulaire du système auteur que nous avons développé. Par ailleurs, comme nous l'avons montré dans le chapitre III, pour parcourir tous les discours possibles, nous utilisons la version réduite du calcul des séquents pour les scénarios avec les événements/actions simples et la version complète du calcul des séquents pour les scénarios avec les événements/actions complexes. Ainsi le système auteur se compose de deux ensembles d'outils employant deux modèles de données différents et deux algorithmes différents. Néanmoins, car l'architecture et les principes de fonctionnement de ces deux ensembles d'outils sont similaires, nous ne présentons dans la section suivante que l'ensemble d'outils appliqué pour les scénarios avec les événements/actions simples (les

lecteurs peuvent voir la présentation de l'ensemble d'outils appliqué pour les scénarios avec les événements/actions complexes en Annexe E).

IV.5. Présentation modulaire des outils qui constituent le système auteur

L'objectif de cette thèse est d'étudier un système d'aide à la production de scénarios de bonne qualité. Pour répondre à cet objectif, nous proposon un modèle, une démarche et un outil que peut mettre en œuvre un auteur pour produire un scénario valide exprimé en logique linéaire, qui peut être exécuté par un système de pilotage d'application interactive scénarisée.

Le système auteur proposé se compose d'un ensemble d'outils qui mettent en œuvre les étapes de la démarche que nous avons donnée dans la section précédente. Il a été développé en appliquant l'approche IDM – Ingénierie Dirigée par les Modèles (les principes de base de cette approche sont présentés en Annexe B), les modules qui le compose sont les suivants :

- **Éditeur de scénario :** Son objectif est d'aider l'utilisateur à modéliser graphiquement une histoire même s'il n'a aucune connaissance en logique linéaire. Pour réaliser une interface graphique, nous avons choisi la technologie GMF (Graphical Modeling Framework) [106] et employé notre métamodèle d'histoire (voir Figure IV.1) comme modèle de domaine (modèle *ecore*). L'utilisateur (auteur), au moyen du langage graphique de l'éditeur de scénario, peut modéliser une histoire en employant des actions simples comme « glisser/déposer » d'éléments graphiques. Ainsi, il ne doit pas manipuler directement la logique linéaire lors de la modélisation. Le résultat du processus de modélisation se compose de diagrammes qui expriment les états, les entrées, les événements/actions, les sorties, les choix et la structure de discours, correspondant au scénario modélisé. Ces diagrammes sont automatiquement transformés, au moyen de la technologie GMF, dans le modèle exprimé en XML.
- **Module de prétraitement :** Ce module réalise un traitement intermédiaire dans notre chaîne de transformation de modèle. Il adapte la représentation intermédiaire XML obtenue suite à la transformation automatique GMF au modèle qui est conforme au métamodèle d'histoire donnée dans la Figure IV.1.
- **Module d'établissement des contraintes :** Son objectif est d'aider l'utilisateur à préciser les paramètres/contraintes pour pouvoir vérifier la satisfaction du scénario modélisé pour les propriétés de narration *Complexité*, *Satisfaction d'événements clés*, *Séquencement*, *Fréquence des événements dramatiques*, *Variété d'effets dramatiques*, et *Équilibre de l'effet dramatique*. Pour cela, nous avons construit ce module comme une interface graphique mettant en œuvre notre modèle de contraintes.
- **Module d'analyse :** Ce module a pour but (1) de construire automatiquement le graphe des preuves qui représente l'ensemble des discours possibles du scénario modélisé, et (2) d'analyser automatiquement ce graphe des preuves en vue de fournir à l'utilisateur les informations importantes concernant le scénario telles que : éléments de modélisation inutilisés, discours réussis/échoués, événements/actions de chaque discours, états disponibles de chaque discours après chaque étape, *etc*. Grâce à celles-ci, l'utilisateur peut évaluer rapidement la qualité du scénario courant en vérifiant sa satisfaction pour les propriétés de narration. Pour cela, le module d'analyse implémente notre algorithme de parcours de toutes les preuves. L'entrée de

l'algorithme se compose du modèle d'histoire et des contraintes du scénario à vérifier qui sont exprimés en XML.

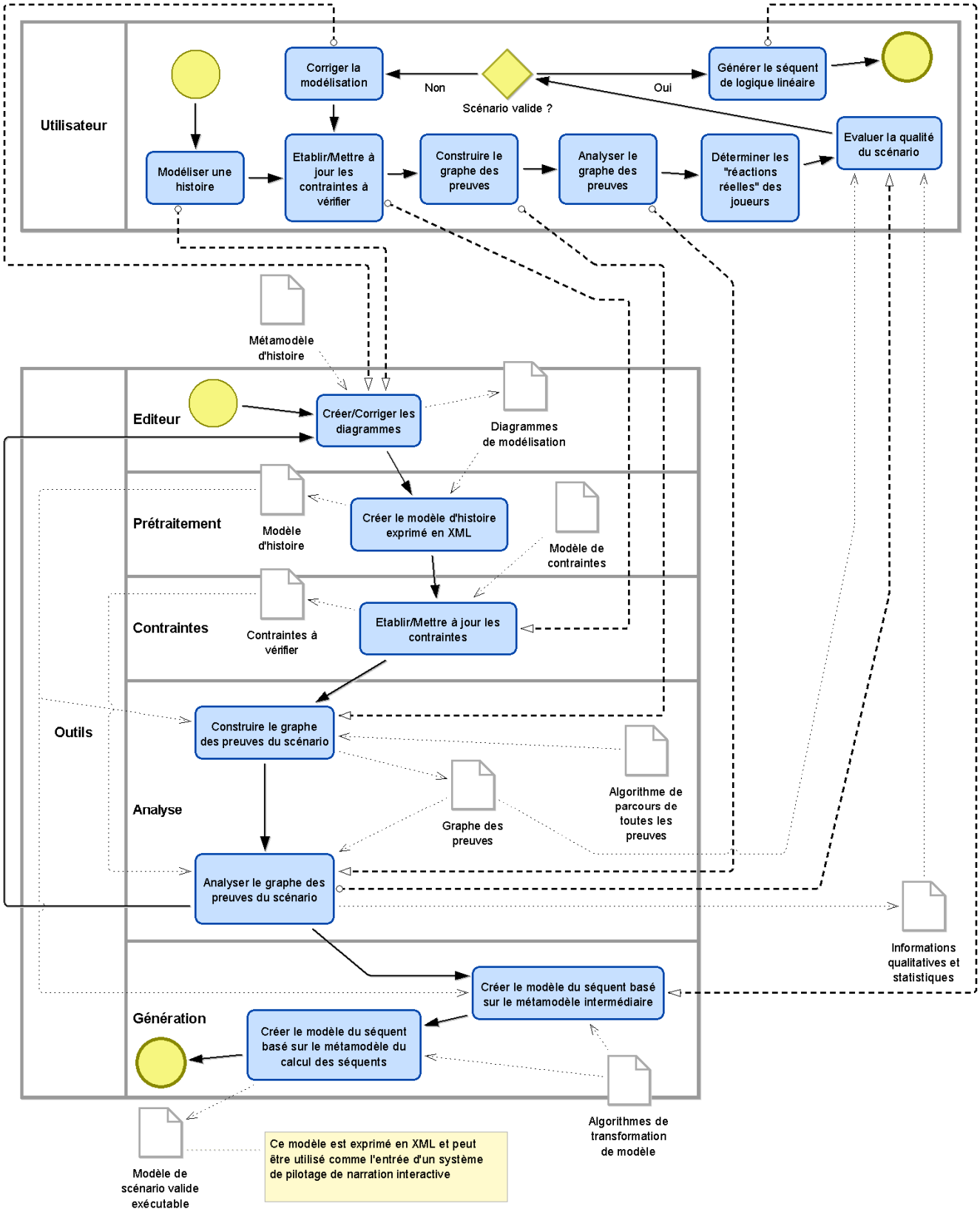


Figure IV.5. Le lien entre la démarche et les outils constituant le système auteur pour la production d'un scénario de bonne qualité modélisé en logique linéaire (construite selon la norme *BPMN* [104]).

- **Module de génération de séquent :** L'objectif de ce module est de permettre de générer automatiquement le modèle exprimé en XML du séquent de logique linéaire qui correspond au scénario validé. Ce modèle est conforme au métamodèle du calcul des séquents (voir Figure IV.3). Pour cela, le module de génération implémente notre algorithme de transformation du modèle d'histoire en le modèle de scénario exécutable. Toutefois, cette transformation de modèle directe est trop complexe, donc nous avons introduit un métamodèle pivot intermédiaire. Ainsi, le module de génération de séquent exécute deux transformations de modèle afin de créer deux représentations XML du séquent produit dans la phase d'édition :
    - la première est fondée sur un métamodèle intermédiaire pour transformer en la seconde (donc elle est implicite pour l'utilisateur) ;
    - la seconde est fondée sur le métamodèle du calcul des séquents, elle est ensuite utilisée comme l'entrée d'un système de pilotage des applications interactives scénarisées.

La Figure IV.5 montre le lien entre la démarche que nous avons donnée dans la section précédente et l'ensemble d'outils du système auteur pour la production d'un scénario de bonne qualité modélisé en logique linéaire. Dans les sections suivantes, nous présentons tour à tour chacun de l'ensemble d'outils que nous avons abordé plus haut (sauf le module de prétraitement).

IV.5.1. Éditeur de scénario

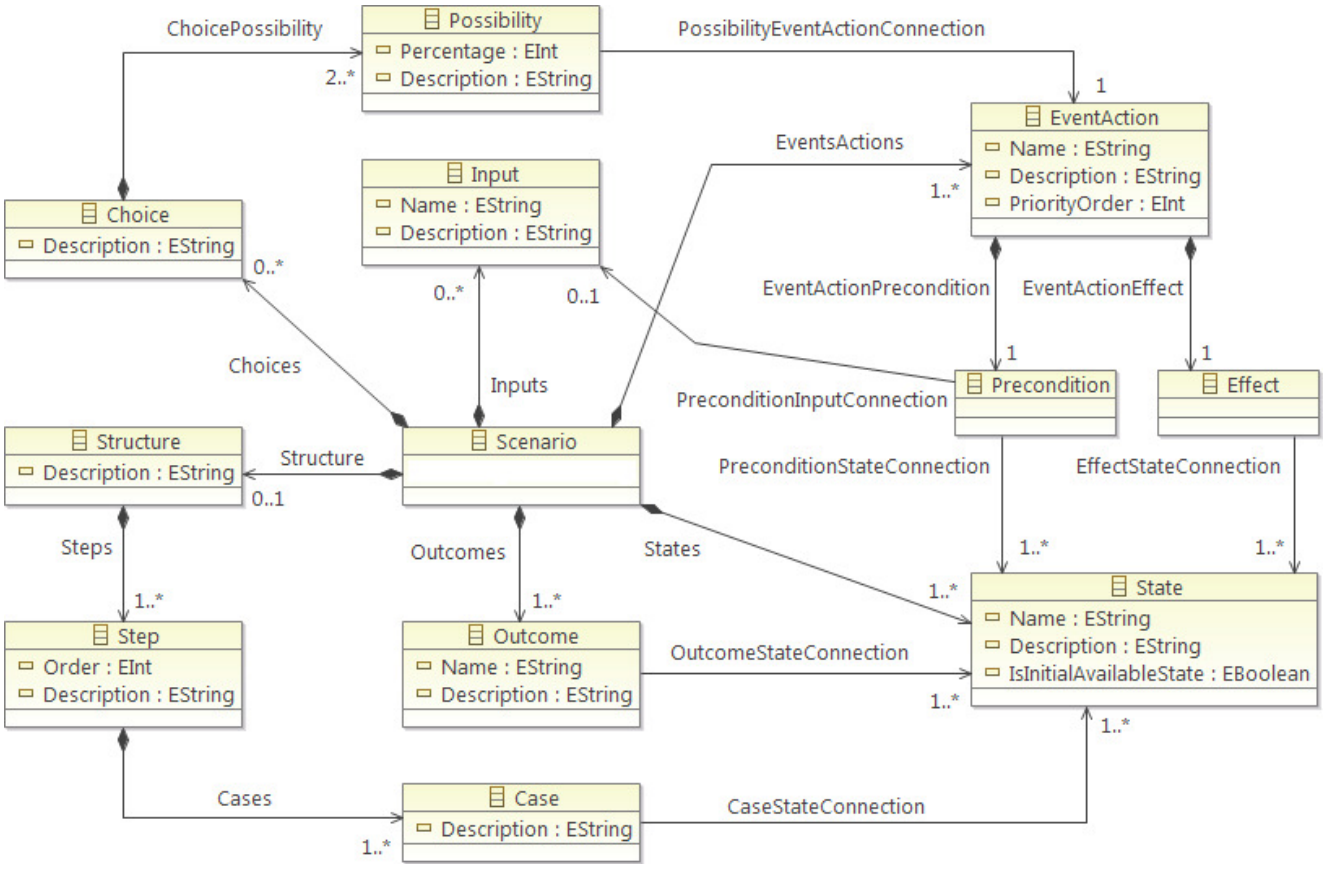


Figure IV.6. Modèle de domaine (modèle *ecore*) de l'éditeur de scénario.

Le modèle de domaine (modèle *ecore*) de l'éditeur de scénario est donné dans la Figure IV.6. Il est fondé sur notre métamodèle d'histoire (voir Figure IV.1). Le processus de fabrication de l'éditeur de scénario est décrit en détail en Annexe C. Son interface utilisateur est indiquée dans la Figure IV.7 où (pour plus d'information, voir Figure V.1 – chapitre V qui est un exemple représentant le diagramme d'états d'un jeu) :

- le **cadre 1** est la zone où les éléments graphiques sont placés afin de construire les diagrammes décrivant une histoire ;
- le **cadre 2** constitue la palette (la barre d'outils) de l'éditeur, elle est organisée en 6 groupes (*States*, *Inputs*, *Outcomes*, *Events/Actions*, *Choices* et *Structure*) qui comprennent les nœuds et les liens correspondant aux éléments graphiques ;
- le **cadre 3** montre les propriétés de l'élément graphique courant (qui est sélectionné) dans le **cadre 1**.

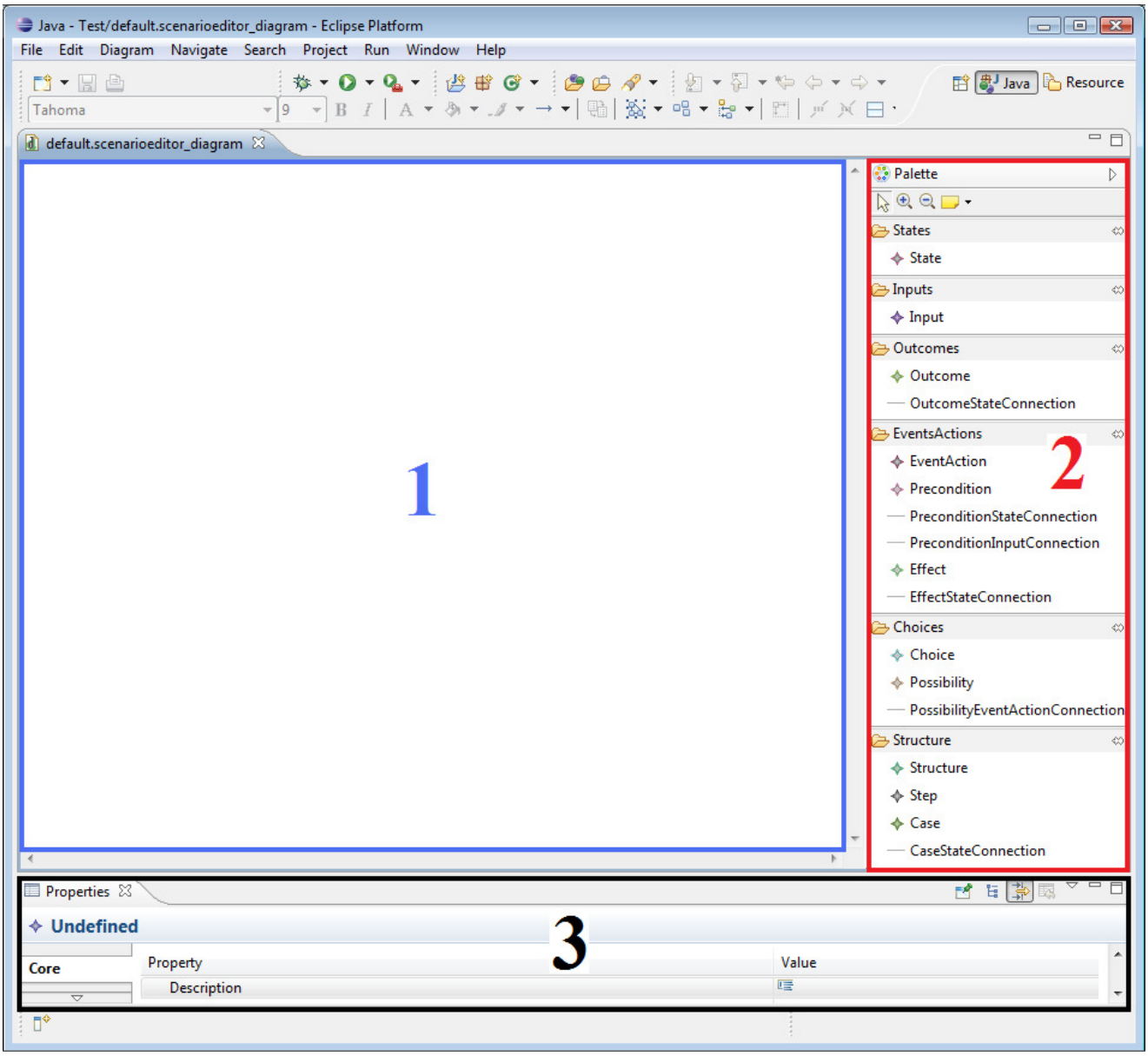


Figure IV.7. Interface utilisateur de l'éditeur de scénario.

IV.5.2. Module d'établissement des contraintes

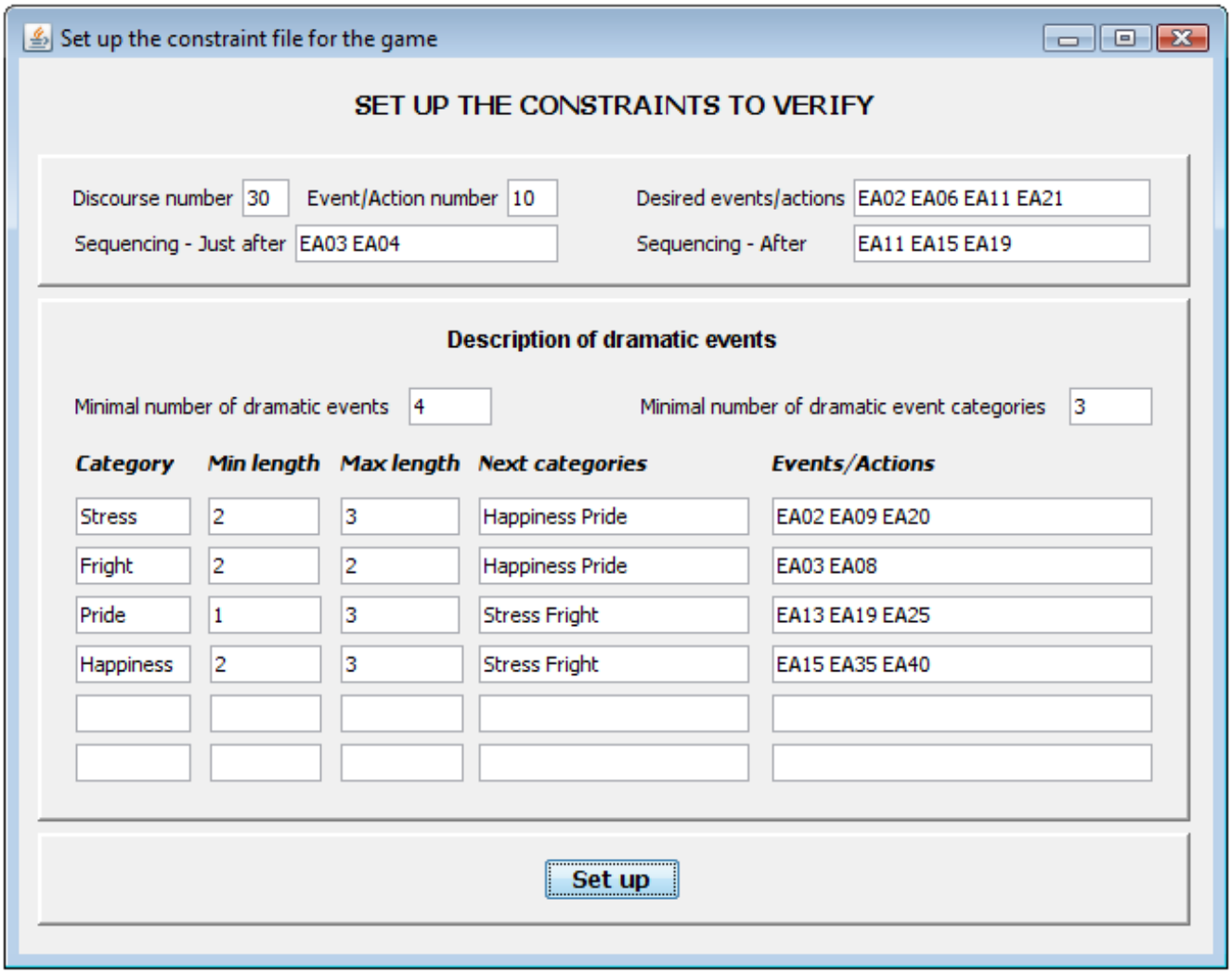


Figure IV.8. Interface utilisateur avec quelques paramètres d'exemple du module d'établissement des contraintes.

Le module d'établissement des contraintes comporte une interface graphique mettant en œuvre notre modèle de contraintes (voir Figure IV.2). La Figure IV.8 donne l'interface utilisateur avec quelques paramètres d'exemple de ce module, le résultat de l'exemple indiqué est donné sous forme XML par :

```
<Constraints>
   <Complexity DiscourseNumber="30" EventActionNumber="10"/>
   <DesiredEventsActions>
       <EventAction Name="EA02"/><EventAction Name="EA06"/>
       <EventAction Name="EA11"/><EventAction Name="EA21"/>
   </DesiredEventsActions>
```

```
        <JustAfterEventsActions>
            <EventAction Name="EA03"/><EventAction Name="EA04"/>
        </JustAfterEventsActions>
        <AfterEventsActions>
            <EventAction Name="EA11"/>
            <EventAction Name="EA15"/>
            <EventAction Name="EA19"/>
        </AfterEventsActions>
        <DramaticEvents Number="4" CategoryNumber="3">
            <Category Name="Stress" MinLength="2" MaxLength="3">
               <NextCategory Name="Happiness"/><NextCategory Name="Pride"/>
               <EventAction Name="EA02"/>
               <EventAction Name="EA09"/>
               <EventAction Name="EA20"/>
            </Category>
            <Category Name="Fright" MinLength="2" MaxLength="2">
               <NextCategory Name="Happiness"/><NextCategory Name="Pride"/>
               <EventAction Name="EA03"/><EventAction Name="EA08"/>
            </Category>
            <Category Name="Pride" MinLength="1" MaxLength="3">
               <NextCategory Name="Stress"/><NextCategory Name="Fright"/>
               <EventAction Name="EA13"/><EventAction Name="EA19"/>
               <EventAction Name="EA25"/>
            </Category>
            <Category Name="Happiness" MinLength="2" MaxLength="3">
               <NextCategory Name="Stress"/><NextCategory Name="Fright"/>
               <EventAction Name="EA15"/><EventAction Name="EA35"/>
               <EventAction Name="EA40"/>
            </Category>
        </DramaticEvents>
    </Constraints>
```

IV.5.3. Module d'analyse

Ce module a pour but (1) de construire automatiquement le graphe des preuves qui représente l'ensemble des discours possibles du scénario modélisé, et (2) d'analyser automatiquement ce graphe des preuves en vue de fournir à l'utilisateur les informations qualitatives et statistiques concernant le scénario. Ainsi, l'utilisateur peut évaluer rapidement la qualité du scénario courant en vérifiant sa satisfaction pour les propriétés de narration. Pour cela, le module

d'analyse implémente notre algorithme de parcours de toutes les preuves. Il emploie le modèle d'historie exprimé en XML (conforme au métamodèle donné dans la Figure IV.1) et le modèle de contraintes du scénario (conforme au métamodèle donné dans la Figure IV.2) comme ses entrées. Nous présentons séparément, dans les sections suivantes, l'algorithme de construction de graphe des preuves d'un scénario, et les informations sur le scénario fournies par ce module.

IV.5.3.1. Construire le graphe des preuves d'un scénario

En vue de construire le graphe des preuves d'un scénario, qui comprend tous les discours possibles (et leur résultat réussi ou échoué), nous avons proposé un algorithme, implémenté par le module d'analyse, permettant de parcourir toutes les preuves d'un modèle d'histoire. Le parcours est une recherche en profondeur. Cet algorithme est fondé sur la version réduite du calcul des séquents dont les règles principales sont décrites en Section III.5.2.1 – chapitre III. Le résultat obtenu est le graphe des preuves du scénario. Chacune de ses branches est une possibilité de choix dans le scénario (décidée soit par le système de pilotage soit par le joueur). Le fonctionnement détaillé de l'algorithme est donné dans la Figure IV.9 (pour plus d'information, les lecteurs peuvent voir Figure V.16 – chapitre V illustrant le graphe des preuves d'un scénario de jeu qui est automatiquement construit par le module d'analyse en exécutant cet algorithme).

IV.5.3.2. Analyser un scénario

En vue d'aider l'utilisateur à évaluer la qualité du scénario modélisé, le graphe des preuves du scénario est automatiquement analysé. Cette analyse donne des informations qualitatives et statistiques sur le scénario courant. Les informations fournies par ce module sont les suivantes (pour mieux comprendre, voir Section III.4, chapitre III – Formalisation des propriétés de narration) :

- Nombre, pourcentage et liste d'états, d'entrées, d'événements/actions, de sorties inutilisés : Ces informations permettent à l'utilisateur d'identifier des erreurs dans la modélisation, notamment pour celles concernant la propriété « Accessibilité ».
- Nombre, pourcentage et liste de discours réussis/échoués : Ces informations permettent à l'utilisateur de valider la propriété « Absence de blocage ».
- Nombre de discours dans le scénario : Cette information permet à l'utilisateur de valider la propriété « Complexité ».
- Nombre, pourcentage et liste de discours qui satisfont (ne satisfont pas) une certaine longueur : Ces informations permettent à l'utilisateur de valider la propriété « Complexité ».
- Nombre et liste de discours réussis/échoués les plus courts/longs (leur nombre d'événements/actions est le plus petit/grand) : Ces informations et celles sur les discours qui satisfont (ne satisfont pas) une certaine longueur fournissent à l'utilisateur une impression sur la longueur possible des discours. Par ailleurs, elles fournissent aussi à l'utilisateur quelques suggestions pour la correction du scénario courant :
    - les discours réussis les plus courts qui ne satisfont pas la propriété « Complexité » sont utilisés pour chercher des raccourcis involontaires ;

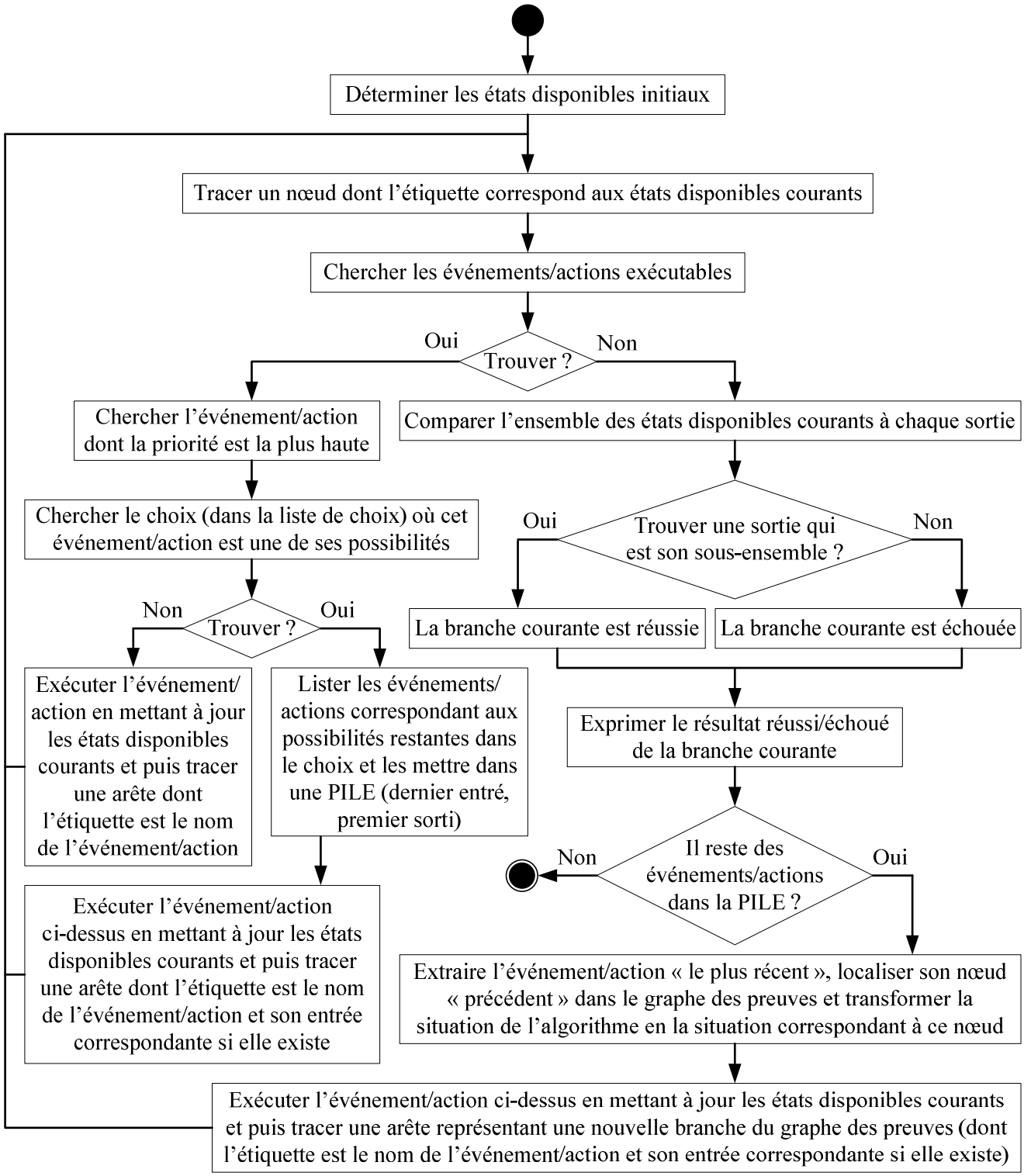


Figure IV.9. Algorithme utilisé afin de construire le graphe des preuves d'un scénario.

- les discours échoués les plus courts peuvent être le point de départ pour une recherche de blocages dans le scénario ;
- les discours échoués les plus longs (et/ou les discours échoués qui satisfont la propriété « Complexité ») suggèrent que les événements/actions à la terminaison de ces discours sont « suspects » (c'est-à-dire qu'ils peuvent être

la cause des blocages). Par exemple, le discours « EA01 → EA04 → … → EA28 → EA30 » est échoué et le plus long dans le scénario, de ce fait, les événements/actions EA28 et EA30 peuvent être la cause des blocages.

- Liste d'événements/actions qui peuvent être la cause des discours échoués : Comme montrer exactement les événements/actions, qui sont la cause des discours échoués, est très complexe, les événements/actions dans la liste établie (nous l'appelons « la liste suspecte ») sont seulement une suggestion pour l'utilisateur afin de corriger des blocages. L'algorithme exécuté par le module d'analyse pour créer cette liste est le suivant :
    - introduire dans la liste suspecte tous les événements/actions des discours échoués dont la longueur est moins de 3 ;
    - examiner tous les derniers événements/actions dans les discours échoués, introduire dans la liste suspecte les derniers événements/actions dont le nombre d'apparition est le plus fréquent, par exemple, il y a 10 discours échoués dans le scénario, leur dernier événement/action est EA07, EA08, EA11, EA07, EA011, EA12, EA11, EA07, EA11, EA07 respectivement, ainsi, le nombre d'apparition de EA07 est de 4, de EA11 est de 4, de EA08 et de EA12 est de 1, par conséquent, les deux EA07 et EA11 sont insérés dans la liste suspecte ;
    - examiner tous les événements/actions dans les discours échoués, introduire dans la liste suspecte les événements/actions dont le nombre d'apparition est le plus fréquent, par exemple, si le nombre d'apparition dans tous les discours échoués de EA05 est de 6, de EA10 est de 5,…, de EA13 est de 1, alors les deux EA05 et EA10 sont insérés dans la liste suspecte.
- Nombre, pourcentage et liste de discours qui contiennent (ne contiennent pas) un ensemble non-ordonné des événements/actions : Ces informations permettent à l'utilisateur de vérifier
    - l'apparition d'un ensemble d'événements/actions « cruciaux » dans le scénario, l'utilisateur peut donc valider la propriété « Satisfaction d'événements clés » ;
    - la fréquence d'apparition d'un ensemble d'événements/actions « suspects » dans le scénario.
- Nombre, pourcentage et liste de discours qui respectent (ne respectent pas) la structure de discours de l'histoire : Ces informations permettent à l'utilisateur de valider la propriété « Satisfaction de la structure ».
- Nombre, pourcentage et liste de discours qui contiennent (ne contiennent pas) un ensemble ordonné des événements/actions consécutifs : Ces informations permettent à l'utilisateur de valider la propriété « Séquencement ».
- Nombre, pourcentage et liste de discours qui contiennent (ne contiennent pas) un ensemble ordonné des événements/actions non-consécutifs : Ces informations permettent à l'utilisateur de valider la propriété « Séquencement ».
- Nombre, pourcentage et liste de discours réussis pour chaque sortie : Ces informations permettent à l'utilisateur de savoir quel discours conduit à quelle sortie ainsi que la probabilité d'atteindre chaque sortie. Cela peut l'aider à distribuer raisonnablement la possibilité de succès de chaque sortie.

- Nombre, pourcentage et liste de discours qui possèdent suffisamment (ne possèdent pas suffisamment) un nombre minimum d'événements dramatiques : Ces informations permettent à l'utilisateur de valider la propriété « Fréquence des événements dramatiques ».
- Nombre, pourcentage et liste de discours qui possèdent suffisamment (ne possèdent pas suffisamment) un nombre minimum de catégories d'événement dramatique : Ces informations permettent à l'utilisateur de valider la propriété « Variété d'effets dramatiques ».
- Nombre, pourcentage et liste de discours qui satisfont (ne satisfont pas) à un ensemble des motifs de catégories d'effet dramatique : Ces informations permettent à l'utilisateur de valider la propriété « Équilibre de l'effet dramatique ».

Ainsi, les informations qualitatives et statistiques ainsi que le graphe des preuves, qui correspondent au scénario modélisé et qui résultent de l'analyse, permettent à l'utilisateur d'évaluer objectivement la qualité du scénario en vérifiant les propriétés de narration. Ces résultats d'analyse aident aussi l'utilisateur à corriger/améliorer le scénario courant jusqu'à ce qu'il obtienne un scénario, qui est le plus pertinent au regard de ses objectifs (scénario valide).

IV.5.4. Module de génération de séquent

L'objectif de ce module est de générer automatiquement, à partir du modèle d'histoire obtenu après le processus de validation, la représentation XML du séquent de logique linéaire qui correspond au scénario validé. Cette représentation est conforme au métamodèle du calcul des séquents. Toutefois, comme l'algorithme pour une telle transformation de modèle est trop complexe, nous avons introduit un métamodèle « pivot » pour cette transformation. La transformation s'effectue en deux étapes. Deux algorithmes de transformation de modèle ont donc été implémentés par le module de génération, afin de créer deux modèles exprimés en XML de ce séquent de logique linéaire :

- le premier est fondé sur le métamodèle intermédiaire pour transformer en le second (donc il est implicite pour les utilisateurs), l'entrée de la transformation de modèle pour fabriquer ce modèle est le modèle d'histoire valide exprimé en XML ;
- le second est fondé sur le métamodèle du calcul des séquents donné dans la Figure IV.3, il est ensuite utilisé comme l'entrée d'un système de pilotage des applications interactives scénarisées. L'entrée de la transformation de modèle pour fabriquer ce modèle est le modèle intermédiaire exprimant en XML le résultat de la première partie de la transformation.

Nous présentons en détail, dans les sections suivantes, les deux étapes de transformation de modèle en vue de générer ces deux modèles du séquent de logique linéaire correspondant au scénario validé.

IV.5.4.1. Transformation de modèle pour la fabrication du modèle de séquent basé sur le métamodèle intermédiaire

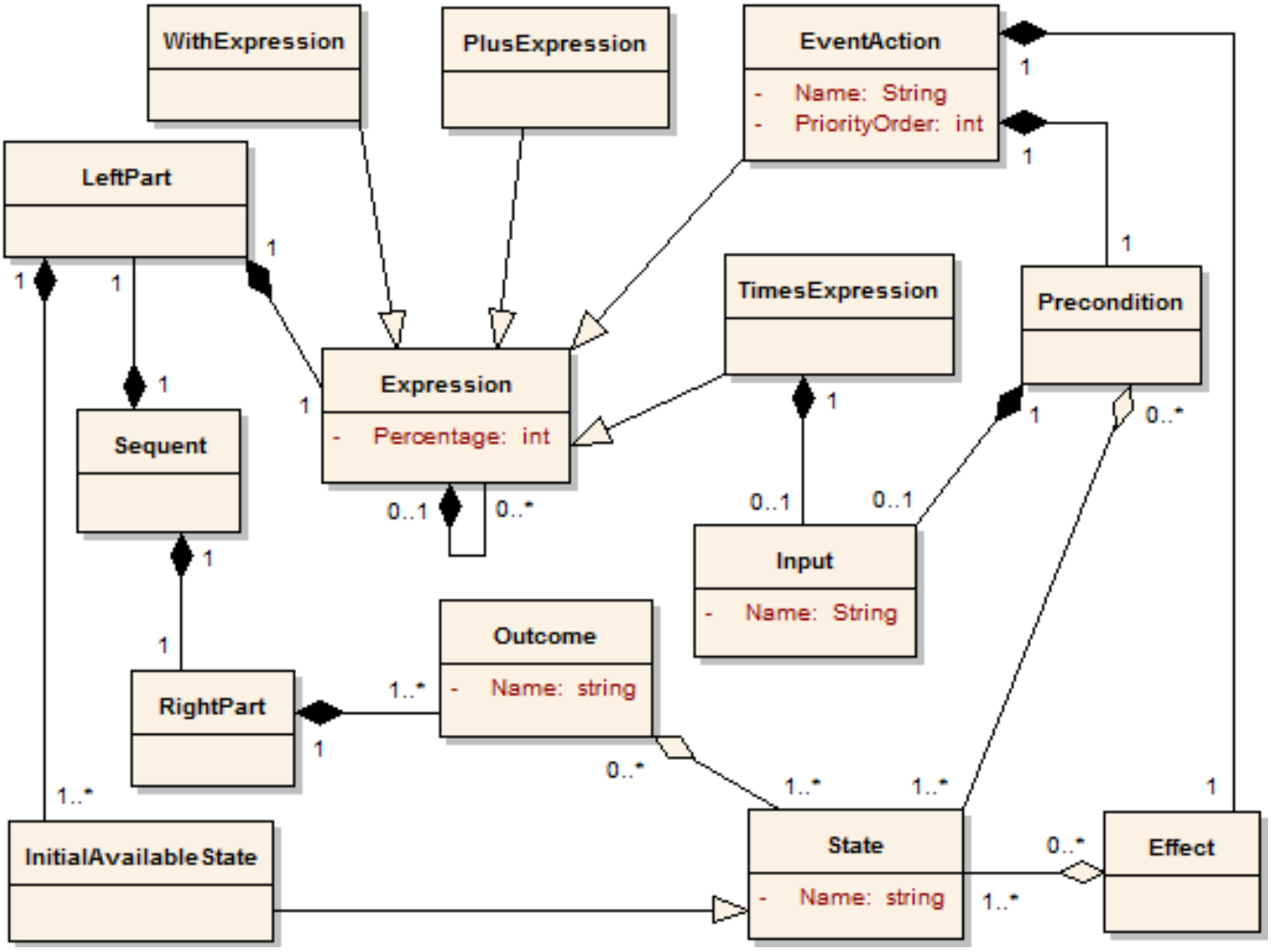


Figure IV.10. Métamodèle des séquents utilisé comme modèle intermédiaire.

La Figure IV.10 donne le métamodèle intermédiaire du séquent qui est transformé à partir du modèle d'histoire valide exprimé en XML. Ainsi, un séquent comprend 2 parties :

- La partie droite comporte un ensemble de sorties, chacune des sorties comprend un ou plusieurs états. Les sorties et les états se distinguent entre eux par l'attribut *Name*.
- La partie gauche est composée d'un ensemble d'états disponibles initiaux et d'une expression (nous l'appelons *l'expression originale*). Les états disponibles initiaux sont ceux dont l'attribut *IsInitialAvailableState* = *true* dans le modèle d'histoire exprimé en XML. L'expression originale peut contenir d'autres expressions, ces expressions peuvent également contenir leurs « expressions d'enfant », *etc*. Une expression est soit un *EventAction* soit une *TimesExpression* soit une *WithExpression* soit une *PlusExpression*.
    - Une *TimesExpression* est une expression dont les composants sont reliés par le connecteur ⊗. Si l'exécution d'un événement/action est décidée par un choix du joueur (en introduisant une entrée), alors il y a une *TimesExpression* entre l'entrée et l'événement/action. En outre, une *TimesExpression* représente

également une « succession d'expressions » (exécuter l'une après l'autre des expressions). Par exemple, si dans un scénario, on a Ik ⊗ EA02 ⊗ EA04 ⊗ EA05, cela signifie que le joueur décide d'exécuter l'événement/action EA02 en sélectionnant l'entrée Ik, ensuite les événements/actions EA04 et EA05 sont exécutés, alors cette *TimesExpression* est décrite comme suit :

```
<TimesExpression>
     <Input Name="Ik"/>
     <EventAction Name="EA02" PriorityOrder="">
          <Precondition> … <Precondition/>
          <Effect> … <Effect/>
     </EventAction>
     <EventAction Name="EA04" PriorityOrder="">
          …
     </EventAction>
     <EventAction Name="EA05" PriorityOrder="">
          …
     </EventAction>
<TimesExpression/>
```

- S'il y a un choix décidé par le joueur entre plusieurs événements/actions, alors ces événements/actions (avec leur « succession d'expressions ») sont reliés par une *WithExpression* (correspondant au connecteur &). Par exemple, si dans un scénario, le joueur possède un choix entre EA01 et EA02 ⊗ EA04 ⊗ EA05 et EA03 (en sélectionnant une des entrées Iw, Ik, Ib respectivement), alors on a la *WithExpression* Iw ⊗ EA01 & Ik ⊗ EA02 ⊗ EA04 ⊗ EA05 & Ib ⊗ EA03 représentée comme suit :

```
<WithExpression>
   <TimesExpression>
     <Input Name="Iw"/>
     <EventAction Name="EA01" PriorityOrder="">
       …
     </EventAction>
   <TimesExpression/>
   <TimesExpression>
```

```
        <Input Name="Ik"/>
        <EventAction Name="EA02" PriorityOrder="">
          …
        </EventAction>
        …
        <EventAction Name="EA05" PriorityOrder="">
          …
        </EventAction>
      <TimesExpression/>
      <TimesExpression>
        <Input Name="Ib"/>
        <EventAction Name="EA03" PriorityOrder="">
          …
        </EventAction>
      <TimesExpression/>
  <WithExpression/>
```

- S'il y a un choix décidé par le système de pilotage entre plusieurs événements/actions, alors ces événements/actions (avec leur « succession d'expressions ») sont reliés par une *PlusExpression* (correspondant au connecteur ⊕). Car on peut attribuer un pourcentage à chaque possibilité de ce choix, les composants d'une *PlusExpression* possèdent l'attribut *Percentage* afin d'exprimer leur pourcentage. Prenons un choix entre EA01 et EA02 qui est décidé par le système de pilotage avec les contraintes : EA03 sera exécuté juste après EA01 (c'est-à-dire une succession EA01 ⊗ EA03), le pourcentage pour EA01 est de 60%, celui pour EA02 est de 40%, donc on a la *PlusExpression* EA01 ⊗ EA03 ⊕ EA02 représentée comme suit :

```
  <PlusExpression>
    <TimesExpression Percentage="60">
      <EventAction Name="EA01" PriorityOrder="">
        …
      </EventAction>
```

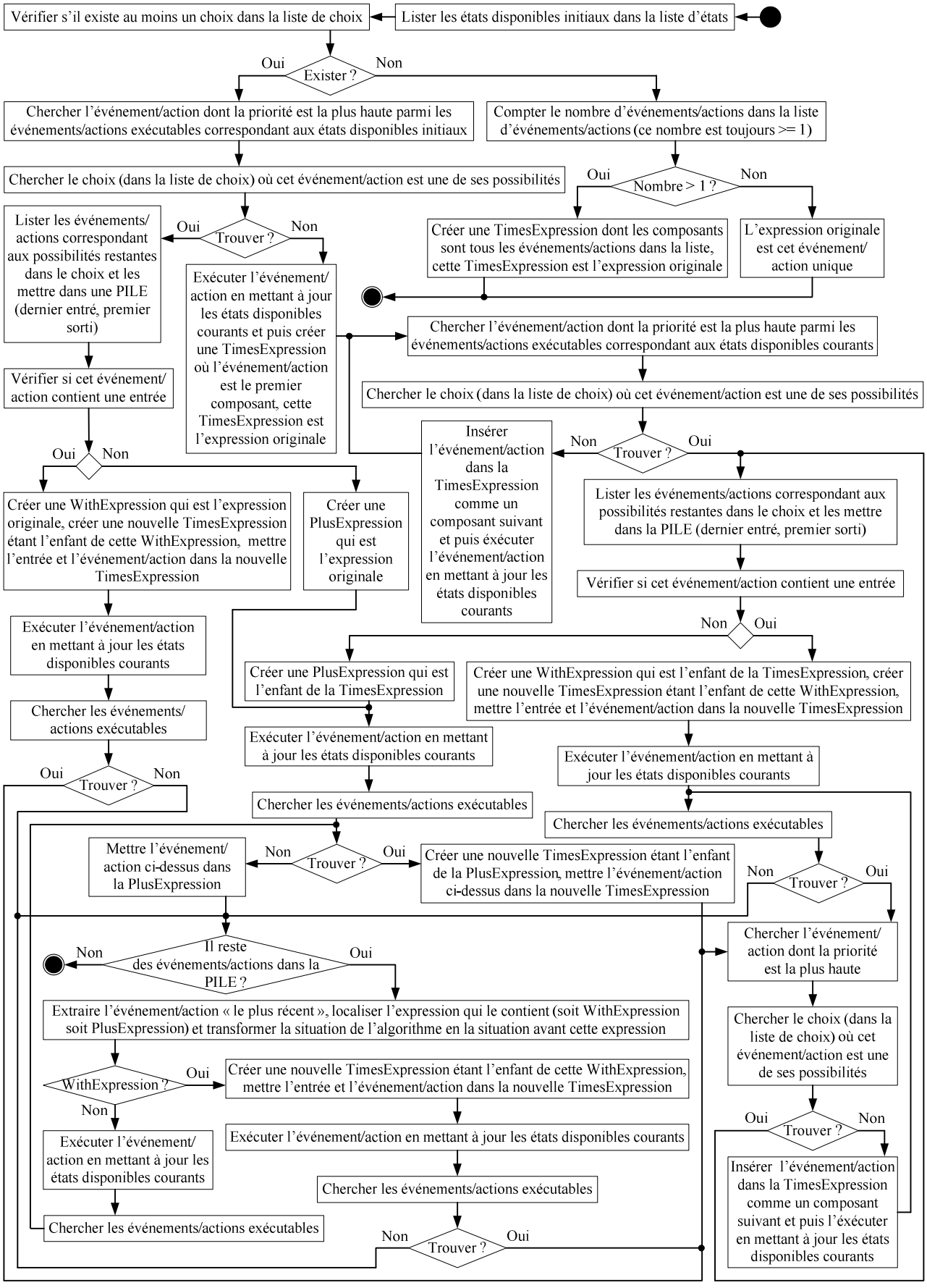


Figure IV.11. Algorithme utilisé afin de construire le modèle exprimé en XML représentant la partie gauche du séquent, qui est fondé sur le métamodèle intermédiaire et qui correspond au scénario validé.

```
        <EventAction Name="EA03" PriorityOrder="">
          …
        </EventAction>
      <TimesExpression/>
      <EventAction Name="EA02" PriorityOrder=""
      Percentage="40">
        …
      </EventAction>
    <PlusExpression/>
```

La Figure IV.11 donne l'algorithme utilisé afin de construire le modèle exprimé en XML représentant la partie gauche du séquent, qui est fondé sur le métamodèle intermédiaire indiqué dans la Figure IV.10, à partir du modèle d'histoire valide exprimé en XML (la construction de la partie droite de ce séquent n'est pas montrée ici puisqu'elle est trop simple).

IV.5.4.2. Transformation de modèle pour la fabrication du modèle de séquent basé sur le métamodèle du calcul des séquents

Le modèle du scénario validé, qui est fondé sur le métamodèle intermédiaire indiqué dans la Figure IV.10, est transformé en le modèle de scénario exécutable selon le calcul des séquents grâce à la transformation de modèle décrite dans cette section. Ce modèle exécutable est ensuite employé comme l'entrée du système de pilotage assurant le contrôle de la gestion du déroulement du jeu. Il est conforme au métamodèle donné dans la Figure IV.3.

L'algorithme, qui est implémenté pour réaliser cette transformation de modèle, fonctionne selon les règles suivantes :

- La partie droite du séquent est inchangée.
- La liste d'états disponibles au moment initial du scénario validé est la liste d'états disponibles initiaux du modèle fondé sur le métamodèle intermédiaire.
- Les éléments « imply, times, with, plus » dans les formules sont créés comme suit :
    - Type = "Linear Implication", Name = "imply" : Cet élément est ajouté entre la précondition et l'effet d'un événement/action (puisque chaque événement/action dans le jeu est décrit par une formule d'implication linéaire) ;
    - Type = "Multiplicative Conjunction", Name = "times" : Ces éléments sont ajoutés entre les composants d'une *TimesExpression*, ou entre les composants (état, entrée) dans la précondition, ou entre les états dans l'effet d'un événement/action ;
    - Type = "Additive Conjunction", Name = "with" : Ces éléments sont ajoutés entre les composants d'une WithExpression ;

- Type = “Additive Disjunction”, Name = “plus” : Ces éléments sont ajoutés entre les composants d’une PlusExpression.

- La partie gauche du séquent au moment initial du scénario validé ne comprend qu’une formule (donc son attribut *Order* = 1), qui est transformée à partir de l’expression originale du séquent fondé sur le métamodèle intermédiaire. Cette formule est construite comme suit :
  - Si l’expression originale est un événement/action, les éléments de la formule correspondent aux composants de l’événement/action, par exemple, pour l’expression originale ci-dessous

```
<EventAction Name="EA01" PriorityOrder="0">
   <Precondition>
     <State Name="A"/>
     <State Name="B"/>
   <Precondition/>
   <Effect>
     <State Name="D"/>
   <Effect/>
</EventAction>
```

    on a

```
<Formula Order="1">
    <Element Name="A" Order="1" Type="Atom"/>
    <Element Name="times" Order="2" Type="Multiplicative Conjunction"/>
    <Element Name="B" Order="3" Type="Atom"/>
    <Element Name="imply" Order="4" Type="Linear Implication"
    EventActionName="EA01" EventActionPriorityOrder="0"/>
    <Element Name="D" Order="5" Type="Atom"/>
</Formula>
```

  - Si l’expression originale est soit une WithExpression soit une PlusExpression soit une TimesExpression, les composants de la formule correspondent aux composants de l’expression originale, entre ces composants, nous ajoutons les éléments qui expriment les connecteurs &, ⊕ et ⊗ respectivement. Par exemple, si l’expression originale d’un scénario est une WithExpression composée de 3 TimesExpression, alors, dans la formule, il y a 3 composants correspondant aux 3 TimesExpression, entre eux, nous insérons deux éléments « with »

```
<Formula Order="1">
    …
    <Element Name="with" Order="…" Type="Additive Conjunction"/>
```

```
    …
    <Element Name="with" Order="…" Type="Additive Conjunction"/>
    …
</Formula>
```

Ensuite, nous considérons à leur tour les composants de l'expression originale ainsi que les composants qui sont leurs enfants, leurs petits-enfants, *etc*. avec les contraintes suivantes :

- Si le composant courant (c'est-à-dire le composant qui est en train d'être examiné) est soit une WithExpression soit une PlusExpression soit une TimesExpression, ajouter les éléments qui expriment les connecteurs &, ⊕, ⊗ respectivement entre ses composants et puis continuer à examiner à leur tour ces composants.
- Si le composant courant est soit une WithExpression soit une PlusExpression soit un événement/action, il faut ajouter une paire de parenthèses « ( ) » entourant ses composants. Par exemple, soit la TimesExpression Iw ⊗ EA01 étant l'enfant d'une expression originale WithExpression, comme cette TimesExpression possède 2 composants Iw et EA01, nous ajoutons entre eux un élément « times », comme EA01 est un événement/action, il faut une paire de parenthèses « ( ) » entourant ses composants, ainsi la WithExpression suivante

```
<WithExpression>
    <TimesExpression>
      <Input Name="Iw"/>
      <EventAction Name="EA01" PriorityOrder="0">
          <Precondition>
              <State Name="Pi"/>
              <Input Name="Iw"/>
          <Precondition/>
          <Effect>
              <State Name="Pw"/>
          <Effect/>
      </EventAction>
    <TimesExpression/>
    …
<WithExpression/>
```

correspondant à Iw ⊗ (Pi ⊗ Iw ⊸ Pw) & ..., est représentée par

```
<Formula Order="1">
     <Element Name="Iw" Order="1" Type="Atom"/>
     <Element Name="times" Order="2" Type="Multiplicative Conjunction"/>
     <Element Name="(" Order="3" Type="Open Parenthesis" ParenthesisLevel="1"/>
     <Element Name="Pi" Order="4" Type="Atom"/>
     <Element Name="times" Order="5" Type="Multiplicative Conjunction"/>
     <Element Name="Iw" Order="6" Type="Atom"/>
     <Element Name="imply" Order="7" Type="Linear Implication"
     EventActionName="EA01" EventActionPriorityOrder="0"/>
     <Element Name="Pw" Order="8" Type="Atom"/>
     <Element Name=")" Order="9" Type="Close Parenthesis" ParenthesisLevel="1"/>
     <Element Name="with" Order="10" Type="Additive Conjunction"/>
     …
</Formula>
```

IV.5.5. Bilan

Dans cette section, nous avons donné une présentation modulaire du système auteur que nous avons développé. Son objectif est d'aider un auteur/utilisateur à produire un modèle valide en logique linéaire représentant un scénario de bonne qualité d'une histoire, qui peut être exécuté par un système de pilotage des applications interactives scénarisées. Pour cela, nous avons proposé (1) un métamodèle d'histoire qui ne nécessite pas de connaissances en logique linéaire, (2) un métamodèle exprimant les contraintes d'un scénario, (3) un algorithme de parcours de toutes les preuves d'un modèle d'histoire, (4) un métamodèle des séquents utilisé comme modèle intermédiaire, (5) un métamodèle de scénario exécutable conforme au calcul des séquents, et (6) deux algorithmes de transformation de modèle pour transformer un modèle d'histoire valide en un modèle de scénario exécutable.

Avec l'aide du système auteur, qui implémente les modèles et les algorithmes mentionnés ci-dessus, un auteur/utilisateur peut :

i. modéliser graphiquement une histoire à travers les manipulations « glisser/déposer » simples ;
ii. exprimer graphiquement les contraintes à vérifier du scénario modélisé ;
iii. construire automatiquement le graphe des preuves qui représente tous les discours possibles du scénario ;
iv. faire l'analyse automatique au niveau structurel du scénario afin d'obtenir ses informations qualitatives et statistiques importantes ;
v. évaluer la qualité du scénario modélisé en vérifiant sa satisfaction par rapport aux propriétés de narration ;
vi. recevoir automatiquement un modèle de séquent de logique linéaire exécutable exprimant le scénario validé, dont la représentation est conforme au métamodèle du calcul des séquents (le modèle de séquent produit est ensuite employé comme l'entrée d'un système de pilotage assurant le contrôle de la gestion du déroulement du jeu).

Un exemple concret ayant pour objet d'illustrer le processus d'utilisation de notre système auteur est présenté au chapitre V. Dans la section suivante, nous donnons deux cas d'application du système auteur. Le premier est un prototype de système de pilotage qui

utilise le scénario/séquent valide produit par le système auteur comme son entrée. Le deuxième cas d'application décrit la manière dont on peut produire un modèle PDDL valide représentant un scénario d'une histoire, grâce à notre système auteur.

IV.6. Deux exemples d'application du système auteur

IV.6.1. Prototype de système de pilotage permettant d'exécuter un scénario valide produit par le système auteur

Dans la section précédente, nous avons présenté un système auteur permettant à l'utilisateur de produire un scénario d'un jeu. L'utilisation d'un modèle formel exécutable permet de s'assurer que le scénario produit est riche, valide et opérationnel. Ainsi, afin de réaliser des jeux vidéo fondés sur la narration interactive, dont l'évolution respecte les effets souhaités par les auteurs, et en même temps, est influencée par les actions du joueur, notre solution est de construire un système de pilotage, dont l'entrée est les scénarios de jeu de bonne qualité produits par l'utilisateur grâce au système auteur. Le mécanisme de déduction automatique des séquents de logique linéaire, correspondant aux scénarios, assiste le système de pilotage dans la gestion du déroulement des jeux, selon leurs scénarios riches et valides. Par conséquent, le déroulement des jeux créés garantit le respect des contraintes scénaristiques des auteurs tout en prenant en compte les inflexions données par le joueur.

Par ailleurs, afin de se décharger de tous les aspects graphiques et de s'adapter à de nombreux supports différents, de façon similaire aux architectures des systèmes de pilotage de narration interactive abordés en Section II.2 – chapitre II (Mimesis [102], Façade [67], Interactive Drama Architecture [64]…), dans notre système de pilotage, la dissociation entre les composants chargés du jeu (environnement virtuel) et ceux chargés du pilotage est nécessaire. Nous présentons ci-dessous un prototype de système de pilotage de narration interactive que nous avons développé, qui satisfait à toutes ces exigences.

IV.6.1.1. Architecture du système de pilotage

L'architecture de notre système de pilotage pour des jeux vidéo fondés sur la narration interactive est composée de 3 composants : un Module d'analyse de séquent, un Pilote et un Moteur de jeu (voir Figure IV.12). Le moteur de jeu est employé afin de construire à l'avance toutes les interfaces (scènes) graphiques nécessaires, et de diriger le processus de représentation du jeu (qui interagit directement avec le joueur). Le pilote a pour but de gérer le déroulement du jeu selon le scénario modélisé, et de demander au moteur de jeu de montrer les interfaces (scènes) convenables pour le jeu à chaque étape, en prenant en compte les « consignes » du module d'analyse de séquent et les choix (les actions) du joueur (transmis au pilote à travers le moteur de jeu). Ainsi, le moteur de jeu fournit toujours un environnement virtuel cohérent qui est adéquat pour chacune des situations du jeu. L'entrée du module d'analyse est un modèle exprimé en XML qui représente le séquent de logique linéaire traduisant la situation initiale du jeu. Ce modèle est conforme au métamodèle du calcul des séquents et est mis à jour après chaque étape. Il est le résultat du processus de production d'un scénario de bonne qualité utilisant le système auteur, et ainsi permet d'obtenir le modèle du scénario du jeu. Le mécanisme de déduction automatique du séquent assiste le pilote dans la gestion du déroulement du jeu. Par conséquent, le pilote est capable de suivre exactement la trace d'exécution dans le scénario. Il dirige donc correctement le

déroulement du jeu (afin qu'il n'atteigne jamais des situations incohérentes), en même temps, il peut garantir la liberté du joueur et son rôle déterminant dans la construction des discours ainsi que les objectifs indispensables fixés par les auteurs.

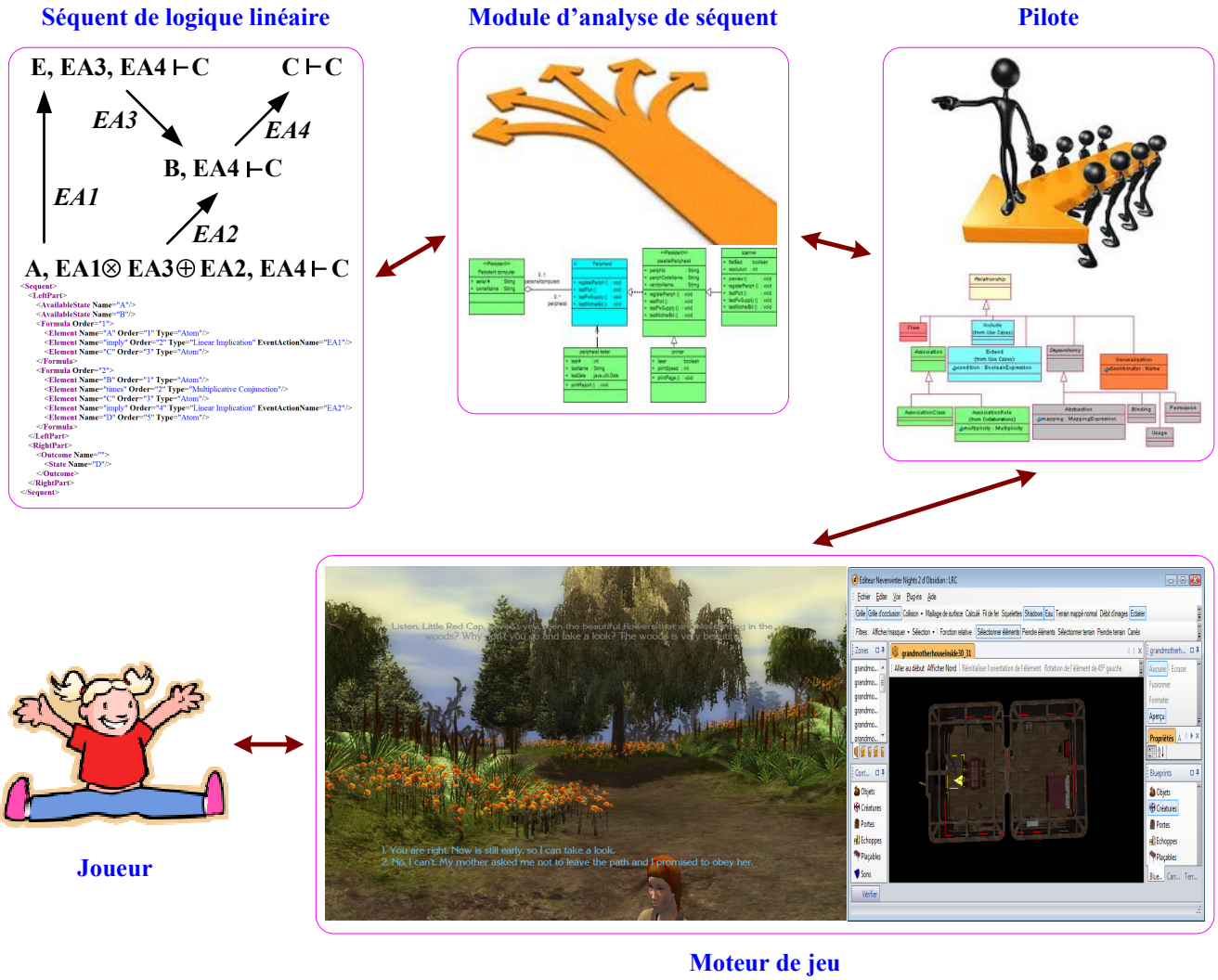


Figure IV.12. Principes constituants du système de pilotage.

IV.6.1.2. Algorithme utilisé dans le système de pilotage

La Figure IV.13 décrit en détail l'algorithme utilisé dans le système de pilotage lors du déroulement d'un jeu. Nous employons l'exemple ci-dessous pour illustrer le fonctionnement de cet algorithme.

Soit le séquent A, D, EA1 ⊗ EA3 ⊕ EA2, EA4 ⊢ B ⊗ C où :

- EA1 : D ⊸ E ;
- EA2 : A ⊗ D ⊸ B ⊗ E ;
- EA3 : A ⊗ E ⊸ B ⊗ E ;
- EA4 : B ⊗ E ⊸ B ⊗ C.

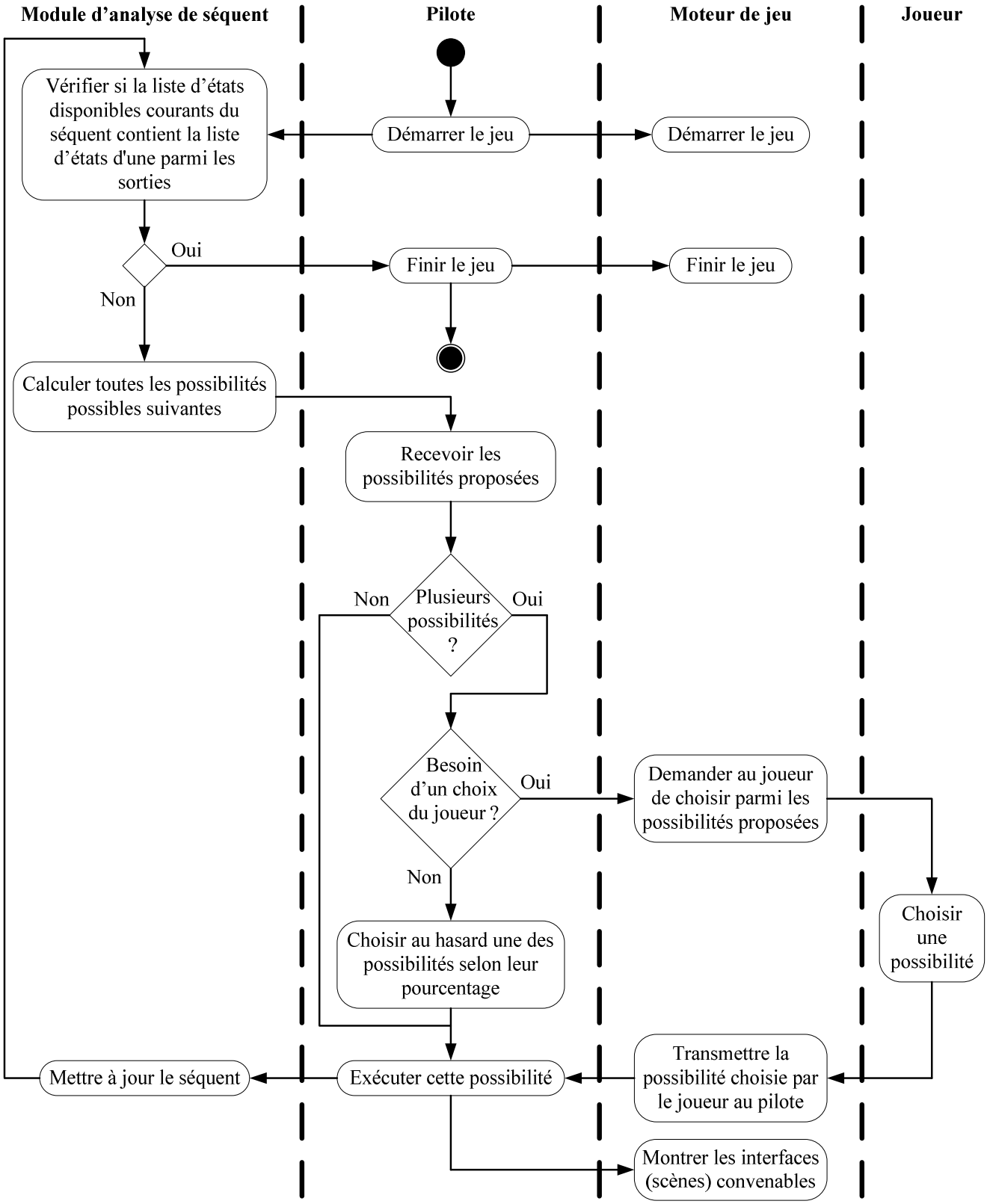


Figure IV.13. Algorithme utilisé dans le système de pilotage lors du déroulement d'un jeu.

Supposons que ce séquent exprime la situation d'un jeu à un certain moment, alors la situation comprend :

- deux états disponibles A et D ;
- deux formules (D ⊸ E) ⊗ (A ⊗ E ⊸ B ⊗ E) ⊕ (A ⊗ D ⊸ B ⊗ E) et B ⊗ E ⊸ B ⊗ C qui sont constituées par quatre événements/actions EA1, EA2, EA3 et EA4 ;
- un choix décidé par le pilote entre EA1 ⊗ EA3 et EA2 ;
- une sortie qui est composée de deux états B et C.

Le séquent est représenté par le segment exprimé sous forme XML suivant, qui est conforme au métamodèle du calcul des séquents :

```
<Sequent>
   <LeftPart>
       <AvailableState Name="A"/>
       <AvailableState Name="D"/>
        <Formula Order="1">
           <Element Name="(" Order="1" Type="Open Parenthesis" ParenthesisLevel="1"/>
           <Element Name="D" Order="2" Type="Atom"/>
           <Element Name="imply" Order="3" Type="Linear Implication" EventActionName="EA1"/>
           <Element Name="E" Order="4" Type="Atom"/>
           <Element Name=")" Order="5" Type="Close Parenthesis" ParenthesisLevel="1"/>
           <Element Name="times" Order="6" Type="Multiplicative Conjunction"/>
           <Element Name="(" Order="7" Type="Open Parenthesis" ParenthesisLevel="1"/>
           …
           <Element Name="imply" Order="11" Type="Linear Implication" EventActionName="EA3"/>
           …
           <Element Name=")" Order="15" Type="Close Parenthesis" ParenthesisLevel="1"/>
           <Element Name="plus" Order="16" Type="Additive Disjunction"/>
           <Element Name="(" Order="17" Type="Open Parenthesis" ParenthesisLevel="1"/>
           …
           <Element Name="imply" Order="21" Type="Linear Implication" EventActionName="EA2"/>
           …
           <Element Name=")" Order="25" Type="Close Parenthesis" ParenthesisLevel="1"/>
        <Formula/>
        <Formula Order="2">
           <Element Name="B" Order="1" Type="Atom"/>
           <Element Name="times" Order="2" Type="Multiplicative Conjunction"/>
           <Element Name="E" Order="3" Type="Atom"/>
           <Element Name="imply" Order="4" Type="Linear Implication" EventActionName="EA4"/>
           <Element Name="B" Order="5" Type="Atom"/>
           <Element Name="times" Order="6" Type="Multiplicative Conjunction"/>
           <Element Name="C" Order="7" Type="Atom"/>
       <Formula/>
   <LeftPart/>
```

```
    <RightPart>
        <Outcome Name="">
            <State Name="B"/>
            <State Name="C"/>
        <Outcome/>
    <RightPart/>
<Sequent/>
```

A ce point d'avancement, comme la liste des états disponibles dans la partie gauche du séquent (A et D) ne contient pas la liste des états de la sortie (B et C), la situation courante n'est pas la situation finale du jeu. Après avoir calculé toutes les possibilités possibles suivantes, le module d'analyse de séquent envoie ce résultat au pilote. Le pilote constate qu'il y a deux possibilités (soit EA1 ⊗ EA3 soit EA2 qui sont reliées par ⊕). Supposons qu'il choisisse EA2, le séquent devient A, D, EA2, EA4 ⊢ B ⊗ C. Maintenant, comme il n'y a qu'une possibilité d'exécuter l'événement/action EA2, le pilote l'exécute tout de suite, il demande au moteur de jeu de montrer les interfaces (scènes) appropriées et au module d'analyse de mettre à jour le séquent. Ainsi, le séquent devient B, E, EA4 ⊢ B ⊗ C. Cette boucle continue jusqu'à ce que le module d'analyse reçoive le séquent B, C ⊢ B ⊗ C, qui est exprimé par :

```
<Sequent>
    <LeftPart>
        <AvailableState Name="B"/>
        <AvailableState Name="C"/>
    <LeftPart/>
    <RightPart>
        <Outcome Name="">
            <State Name="B"/>
            <State Name="C"/>
        <Outcome/>
    <RightPart/>
<Sequent/>
```

A cet instant, comme la liste d'états disponibles dans la partie gauche du séquent contient la liste d'états de la sortie, le séquent courant correspond à la situation finale du jeu. Le module d'analyse informe le pilote sur cela, et puis ce dernier demande au moteur de jeu de finir le jeu.

IV.6.1.3. Bilan

Dans cette section, nous avons donné le premier exemple d'application du système auteur – un prototype de système de pilotage que nous avons développé, qui utilise les scénarios/séquents valides produits par le système auteur comme entrée. Le mécanisme de déduction automatique des séquents aide le système de pilotage à exécuter ces scénarios valides. Ainsi, le déroulement des jeux vidéo, qui sont dirigés par un tel système de pilotage, respecte les effets souhaités par les auteurs, et en même temps, est influencé par les actions du

joueur. Par ailleurs, la dissociation entre les composants chargés du pilotage et ceux chargés du jeu (environnement virtuel) permet au système de pilotage de se décharger de tous les aspects graphiques et de s'adapter à de nombreux supports différents. Dans la section suivante, nous donnons le deuxième exemple d'application, ce qui décrit la manière dont on peut produire un modèle PDDL valide représentant un scénario d'une histoire, grâce à notre système auteur.

IV.6.2. Production d'un modèle PDDL valide représentant un scénario d'une histoire grâce au système auteur

PDDL [40] est un langage employé afin de caractériser les comportements des modèles de planification. Il est inspiré par STRIPS et a d'abord été développé par Drew McDermott et ses collègues en 1998, avec l'objectif de normaliser des langages de planification utilisés en intelligence artificielle. Par conséquent, il permet une comparaison plus directe de la puissance des systèmes et des approches, ainsi qu'une réutilisation plus facile des modèles de planification. Nous présentons ci-dessous notre contribution à la production d'un modèle PDDL valide représentant un scénario d'une histoire.

Pour cela, nous (1) proposons un métamodèle d'histoire en PDDL pour modéliser une histoire ; (2) adaptons le métamodèle d'histoire en logique linéaire donné dans la Figure IV.1 aux éléments de modélisation en PDDL ; (3) proposons un algorithme de transformation de modèle pour transformer un modèle d'histoire en PDDL à un modèle d'histoire en logique linéaire, et développons un traducteur implémentant cet algorithme ; (4) adaptons notre algorithme permettant de parcourir et analyser un scénario aux changements du métamodèle d'histoire en logique linéaire, et modifions le module d'analyse du système auteur de façon correspondante à cette adaptation.

Ainsi, la démarche qu'un auteur/utilisateur, qui n'est pas un spécialiste de logique linéaire, doit faire pour produire un modèle PDDL valide représentant un scénario d'une histoire, comporte les étapes suivantes (voir la Figure IV.14) :

- **Etape 1 – Modéliser une histoire en PDDL :** L'utilisateur modélise l'histoire selon le métamodèle d'histoire en PDDL que nous proposons, et il reçoit donc un modèle en PDDL qui représente le scénario modélisé.
- **Etape 2 – Transformer le modèle obtenu en PDDL à un modèle correspondant en logique linéaire :** L'utilisateur emploie le traducteur afin de transforme le modèle obtenu en PDDL à un modèle qui est conforme à notre nouveau métamodèle d'histoire en logique linéaire. Ainsi, tous ces deux modèles expriment le scénario modélisé de l'histoire.
- **Etape 3 – Analyser le modèle en logique linéaire obtenu :** L'utilisateur emploie le module d'analyse, qui a été adapté, dans notre système auteur (voir Figure IV.5) en vue de parcourir et analyser tous les discours possibles du scénario représenté par le modèle en logique linéaire obtenu. Le module d'analyse fournit la liste de ces discours (avec leurs événements/actions), ainsi que le résultat d'analyse sur le succès/échec de chacun des discours et leur satisfaction pour les contraintes préétablies par l'utilisateur.

- **Etape 4 – Estimer la qualité du scénario courant :** Grâce aux informations fournies par le module d'analyse, il est possible pour l'utilisateur d'estimer la qualité du scénario courant. Ensuite, l'utilisateur peut corriger/améliorer directement le modèle en PDDL, et refaire les étapes ci-dessus si nécessaire (dans le cas où la qualité du scénario modélisé n'est pas suffisamment bonne). Ce processus est répété dans la phase de construction de scénario jusqu'à ce que l'utilisateur reçoive un scénario qui est le plus pertinent pour ses objectifs (que le scénario soit valide).

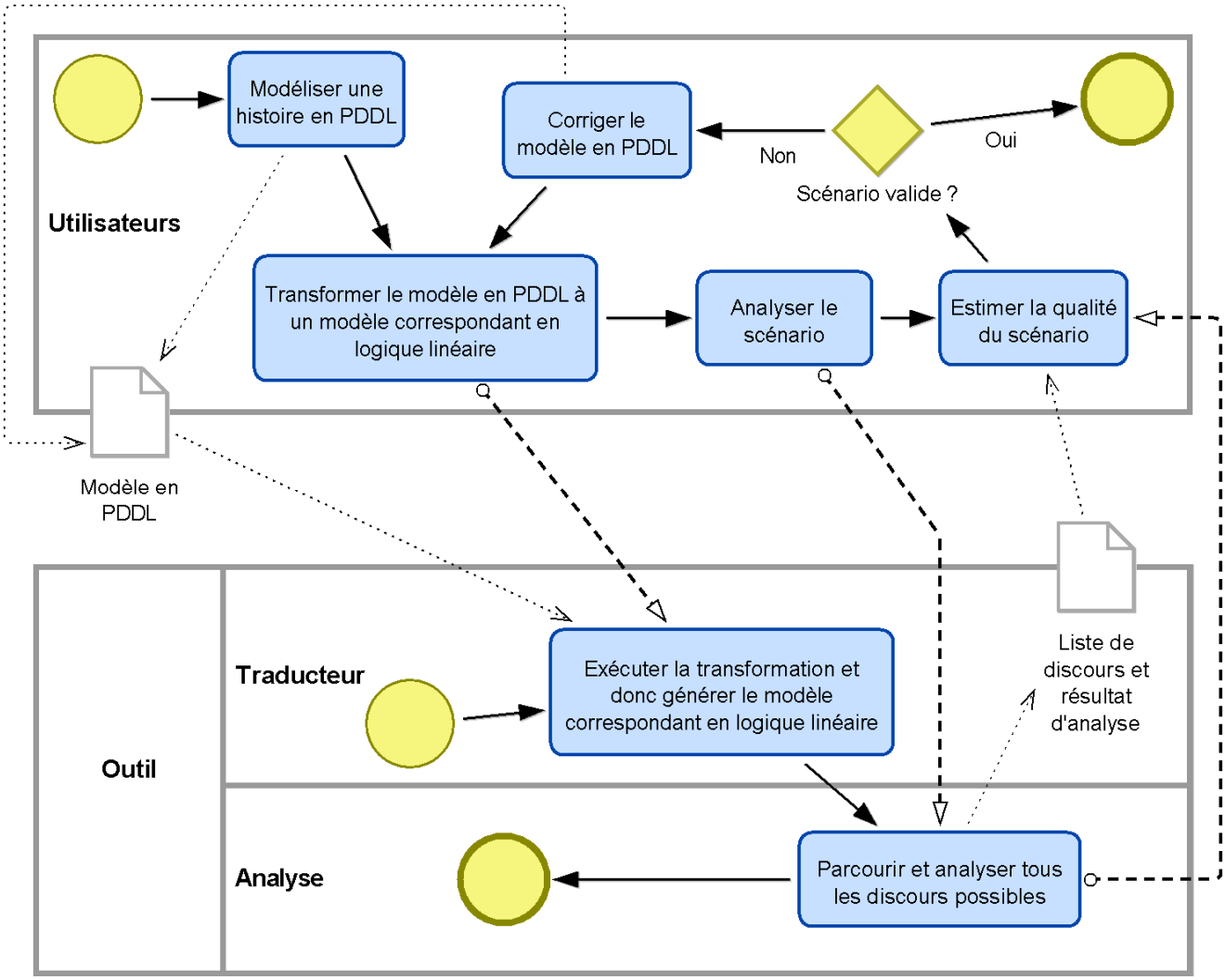


Figure IV.14. Démarche, supportée par le système auteur, pour la production d'un modèle PDDL valide représentant un scénario d'une histoire (construite selon la norme *BPMN* [104]).

Nous présentons en détail, dans les sections suivantes, nos apports pour qu'un auteur/utilisateur, en appliquant le système auteur que nous avons développé, puisse produire un modèle PDDL valide représentant un scénario d'une histoire.

IV.6.2.1. Métamodèle d'histoire en PDDL

Dans le cadre d'un exemple d'application de notre système auteur, nous n'avons pas l'intention d'exploiter toutes les fonctionnalités de PDDL. Nous ne décrivons que comment quelques composants de PDDL (tels que les prédicats, les actions paramétrables comportant

les préconditions/effets, les buts, *etc.*) sont utilisés afin de modéliser une histoire dans notre approche. Pour cela, nous avons proposé un métamodèle, contenant les éléments fournis par PDDL, qui est donné dans la Figure IV.15 où :

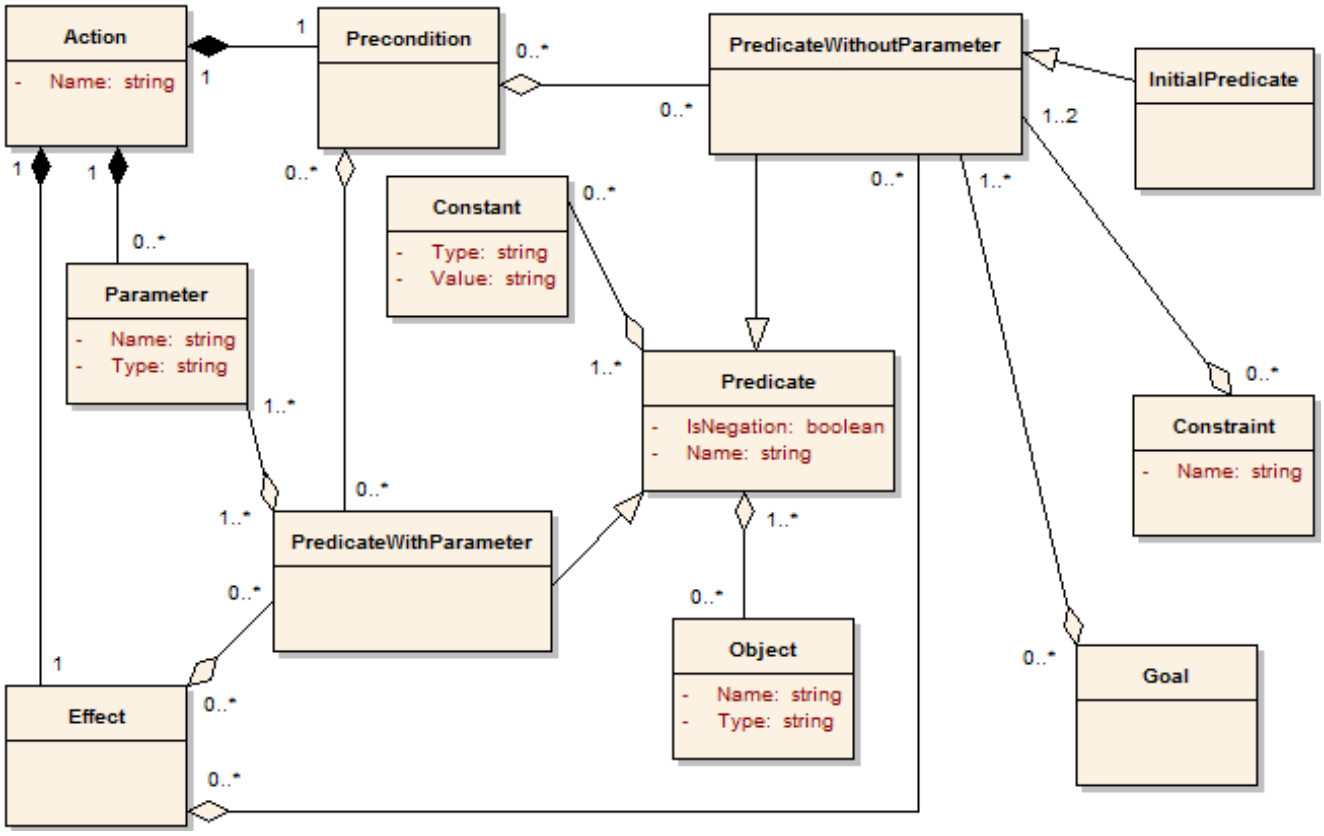


Figure IV.15. Métamodèle utilisé pour modéliser une histoire en PDDL.

- Une constante est exprimée par les attributs *Value* et *Type*, par exemple, les constantes suivantes

```
(:constants
Cake - Food
Wine - Drink
)
```

  sont représentées par le segment exprimé sous forme XML

```
<Constant Value="Cake" Type="Food"/>
<Constant Value="Wine" Type="Drink"/>
```

- Un objet est exprimé par les attributs *Name* et *Type*, par exemple, les objets suivants

```
(:objects
Little-Red-Cap - Character
Forest - Location
)
```

sont représentés par le segment exprimé sous forme XML

```
<Object Name="Little-Red-Cap" Type="Character"/>
<Object Name="Forest" Type="Location"/>
```

- Un paramètre est exprimé par les attributs *Name* et *Type*, par exemple, les paramètres suivants

```
:parameters
(?c - Character ?l - Location)
```

sont représentés par le segment exprimé sous forme XML

```
<Parameter Name="c" Type="Character"/>
<Parameter Name="l" Type="Location"/>
```

- Un prédicat est exprimé par les attributs *Name* et *IsNegation*, il peut contenir des constantes ainsi que des objets. Il y a deux catégories de prédicat : *PredicateWithoutParameter* et *PredicateWithParameter*. Un *PredicateWithParameter* doit évidemment comprendre un ou plusieurs paramètres. Par exemple, le segment exprimé sous forme XML suivant

```
<PredicateWithoutParameter Name="sick" IsNegation="false">
      <Object Name="Grandmother" Type="Character"/>
<PredicateWithoutParameter/>
```

décrit un prédicat *sick (Grandmother)*, qui exprime le fait que la grand-mère (un personnage dans l'histoire) est malade ; tandis que le segment exprimé sous forme XML suivant

```
<PredicateWithParameter Name="visit-with" IsNegation="false">
      <Parameter Name="c1" Type="Character"/>
      <Parameter Name="c2" Type="Character"/>
      <Constant Value="Cake" Type="Food"/>
<PredicateWithParameter/>
```

décrit un prédicat *visit-with (?c1 ?c2 Cake)*, qui exprime le fait que le personnage *c1* visite le personnage *c2* avec un gâteau (*c1* et *c2* sont les paramètres). Ainsi, si tous les paramètres de ce prédicat sont remplacés par les constantes/objets correspondants, il devient un *PredicateWithoutParameter*, par exemple *visit-with (Little-Red-Cap Grandmother Cake)*

```
<PredicateWithoutParameter Name="visit-with" IsNegation="false">
      <Object Name="Little-Red-Cap" Type="Character"/>
      <Object Name="Grandmother" Type="Character"/>
      <Constant Value="Cake" Type="Food"/>
<PredicateWithoutParameter/>
```

- Un prédicat initial est un *PredicateWithoutParameter* qui est vrai au moment initial de l'histoire, par exemple

```
<InitialPredicate Name="sick" IsNegation="false">
      <Object Name="Grandmother" Type="Character"/>
<InitialPredicate/>
```

- Une contrainte de l'histoire définit une exigence que tous les discours possibles dans le scénario modélisé doivent satisfaire, elle est exprimée par l'attribut *Name* et comprend un ou deux *PredicateWithoutParameter*, par exemple le segment exprimé sous forme XML suivant

```
<Constraint Name="always">
      <PredicateWithoutParameter Name="sick" IsNegation="false">
            <Object Name="Grandmother" Type="Character"/>
      <PredicateWithoutParameter/>
<Constraint/>
```

  décrit une contrainte *always (sick (Grandmother))*, qui signifie le fait que « la grand-mère est malade » doit être vrai à tous les moments pour tous les discours dans le scénario modélisé.

- Un but est une terminaison souhaitée de l'histoire, il comprend un ou plusieurs *PredicateWithoutParameter*, par exemple le segment exprimé sous forme XML suivant

```
<Goal>
    <PredicateWithoutParameter Name="alive" IsNegation="false">
        <Object Name="Grandmother" Type="Character"/>
    <PredicateWithoutParameter/>
    <PredicateWithoutParameter Name="alive" IsNegation="false">
        <Object Name="Little-Red-Cap" Type="Character"/>
    <PredicateWithoutParameter/>
    <PredicateWithoutParameter Name="alive" IsNegation="true">
        <Object Name="Wolf" Type="Character"/>
    <PredicateWithoutParameter/>
<Goal/>
```

décrit le but unique de l'histoire « Le Petit Chaperon rouge », qui signifie que tous les discours dans le scénario modélisé doivent mener à la terminaison : la grand-mère et le Petit Chaperon rouge sont vivants, tandis que le loup est mort.

- Une action permet de modifier la situation de l'histoire, elle est exprimée par l'attribut *Name* et comprend une *Précondition*, un *Effect* et des *Parameter*. La précondition et l'effet peuvent contenir des *PredicateWithParameter* ainsi que des *PredicateWithoutParameter*, les paramètres dans ces *PredicateWithParameter* sont les paramètres de l'action. Par exemple, le segment exprimé sous forme XML suivant

```
<Action Name="visit">
    <Parameter Name="c1" Type="Character"/>
    <Parameter Name="c2" Type="Character"/>
    <Parameter Name="f" Type="Food"/>
    <Precondition>
        <PredicateWithParameter Name="sick" IsNegation="false">
            <Parameter Name="c2" Type="Character"/>
        <PredicateWithParameter/>
        <PredicateWithParameter Name="grandmother" IsNegation="false">
            <Parameter Name="c2" Type="Character"/>
            <Parameter Name="c1" Type="Character"/>
        <PredicateWithParameter/>
    <Precondition/>
    <Effect>
        <PredicateWithParameter Name="visit-with" IsNegation="false">
            <Parameter Name="c1" Type="Character"/>
            <Parameter Name="c2" Type="Character"/>
            <Parameter Name="f" Type="Food"/>
        <PredicateWithParameter/>
    <Effect/>
<Action/>
```

décrit l'action *visit (?c1 ?c2 ?f)* qui signifie que : si les faits « le personnage *c2* est malade ; le personnage *c2* est la grand-mère du personnage *c1* » sont vrais, cette action est exécutable, et si elle est exécutée, le fait « le personnage *c1* visite le personnage *c2* avec la nourriture *f* » devient vrai (*c1*, *c2* et *f* sont les paramètres).

Ainsi, avec le métamodèle présenté ci-dessus, on peut modéliser une histoire en PDDL, le scénario obtenu après le processus de modélisation est un modèle en PDDL, qui est composé des listes de constantes, d'objets, de prédicats initiaux, d'actions, de buts et de contraintes décrites plus haut.

IV.6.2.2. Nouveau métamodèle d'histoire en logique linéaire et algorithme pour transformer un modèle en PDDL à un modèle correspondant en logique linéaire

Pour mettre en œuvre une démarche par transformation de modèles, nous proposons un nouveau métamodèle d'histoire en logique linéaire indiqué dans la Figure IV.16. Il a été adapté du métamodèle donné dans la Figure IV.1 pour tenir compte d'éléments de modélisation en PDDL. Pour transformer un modèle d'histoire en PDDL, conforme au métamodèle dans la Figure IV.15, en un modèle d'histoire en logique linéaire, conforme au métamodèle dans la Figure IV.16, nous avons proposé et implémenté un algorithme de transformation de modèles. Les règles principales de l'algorithme sont les suivantes :

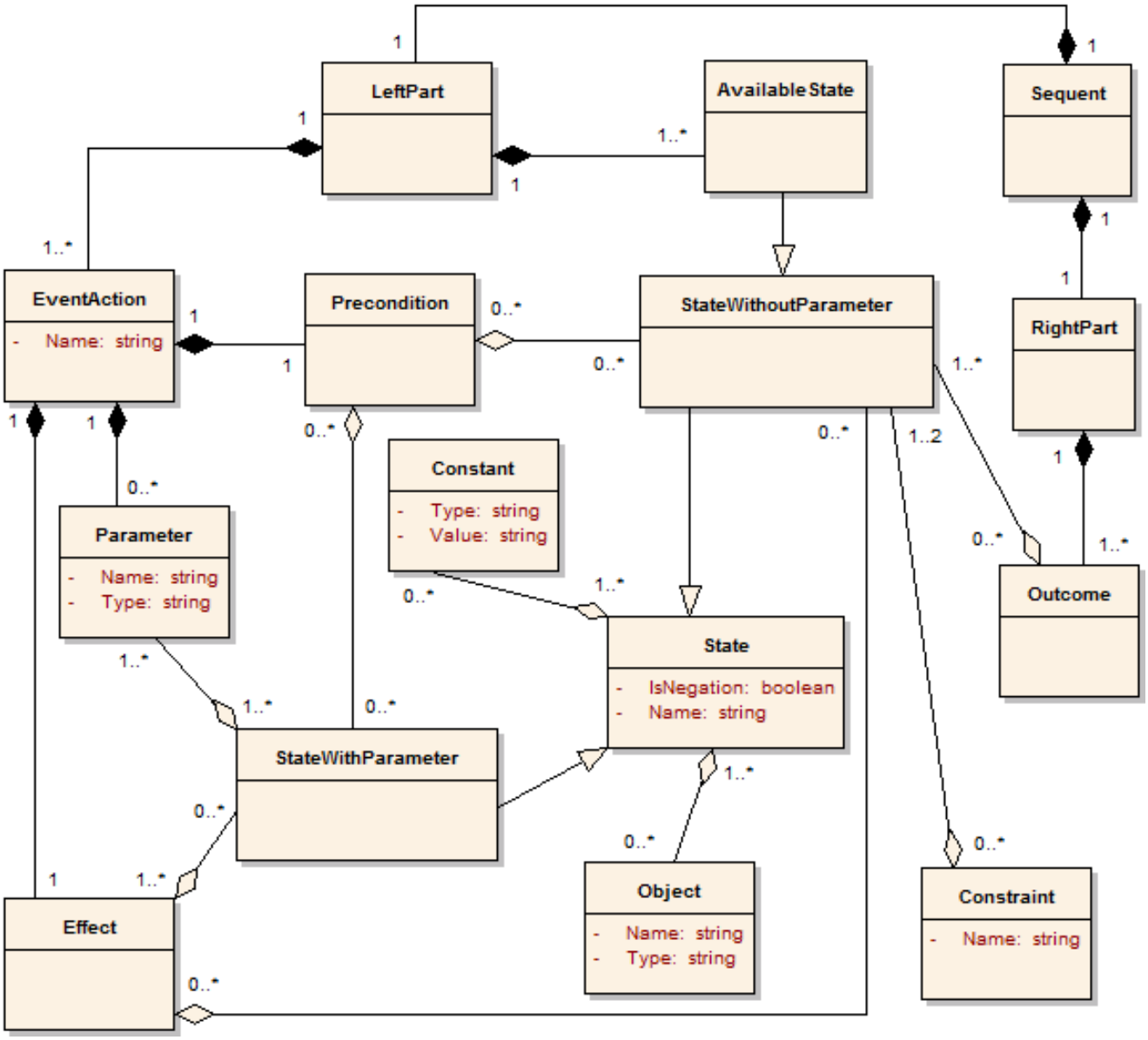


Figure IV.16. Métamodèle d'histoire en logique linéaire, qui est adapté aux éléments de modélisation en PDDL, à partir du métamodèle donné dans la Figure IV.1.

- les listes de constantes, d'objets et de contraintes du modèle en PDDL sont maintenues dans le modèle en logique linéaire ;

- les listes d'états et de sorties du modèle en logique linéaire sont transformées à partir de celles de prédicats et de buts du modèle en PDDL ;
- la liste d'états disponibles du modèle en logique linéaire contient les états qui sont disponibles au moment courant lors du déroulement de l'histoire, si le moment courant est le moment initial, cela correspond aux états disponibles initiaux, et donc équivalents aux prédicats initiaux du modèle en PDDL ;
- la liste d'événements/actions du modèle en logique linéaire est transformée à partir de celle d'actions du modèle en PDDL avec les règles suivantes :
    - les paramètres, les préconditions et les effets des actions en PDDL sont maintenus ;
    - les prédicats dans les préconditions/effets des actions en PDDL sont transformés aux états correspondants dans les préconditions/effets des événements/actions en logique linéaire ;
    - pour chacune des actions en PDDL, si un certain prédicat dans sa précondition n'a aucun prédicat négatif équivalent dans son effet, il faut ajouter un état transformé de ce prédicat dans l'effet de l'événement/action correspondant en logique linéaire.

Par exemple, l'événement/action en logique linéaire, qui est équivalent à l'action *visit* en PDDL suivante

```
<Action Name="visit">
   <Precondition>
      <PredicateWithoutParameter Name="sick" IsNegation="false">
         <Object Name="Grandmother" Type="Character"/>
      <PredicateWithoutParameter/>
      <PredicateWithoutParameter Name="at" IsNegation="true">
         <Object Name="Little-Red-Cap" Type="Character"/>
         <Object Name="Grandmother's house" Type="Location"/>
      <PredicateWithoutParameter/>
   <Precondition/>
   <Effect>
      <PredicateWithoutParameter Name="at" IsNegation="false">
         <Object Name="Little-Red-Cap" Type="Character"/>
         <Object Name="Grandmother's house" Type="Location"/>
      <PredicateWithoutParameter/>
   <Effect/>
<Action/>
```

est représenté par le segment exprimé sous forme XML ci-dessous (puisque le prédicat *sick* dans la précondition de l'action en PDDL n'a aucun prédicat négatif équivalent

dans l'effet, on doit ajouter l'état *sick* transformé de ce prédicat dans l'effet de l'événement/action en logique linéaire) :

```
<EventAction Name="visit">
   <Precondition>
      <StateWithoutParameter Name="sick" IsNegation="false">
         <Object Name="Grandmother" Type="Character"/>
      <StateWithoutParameter/>
      <StateWithoutParameter Name="at" IsNegation="true">
         <Object Name="Little-Red-Cap" Type="Character"/>
         <Object Name="Grandmother's house" Type="Location"/>
      <StateWithoutParameter/>
   <Precondition/>
   <Effect>
      <StateWithoutParameter Name="at" IsNegation="false">
         <Object Name="Little-Red-Cap" Type="Character"/>
         <Object Name="Grandmother's house" Type="Location"/>
      <StateWithoutParameter/>
      <StateWithoutParameter Name="sick" IsNegation="false">
         <Object Name="Grandmother" Type="Character"/>
      <StateWithoutParameter/>
   <Effect/>
<EventAction/>
```

Ainsi, après la transformation, exécutée automatiquement par le traducteur, d'un modèle en PDDL, qui modélise un scénario d'une histoire, à un modèle correspondant en logique linéaire, le séquent obtenu, représentant aussi ce scénario modélisé, est composé de deux parties :

- la partie gauche comprend une liste d'états disponibles initiaux, et une liste d'événements/actions dont le format unique est $A1 \otimes A2 \otimes \ldots \otimes An \multimap B1 \otimes B2 \otimes \ldots \otimes Bm$, où Ai, Bj sont les états ; ces états disponibles initiaux et événements/actions sont reliés par les connecteurs $\otimes$;
- la partie droite est une liste de sorties dont chacune contient un ou plusieurs états reliés par les connecteurs $\otimes$, ces sorties sont reliées par les connecteurs $\oplus$.

Nous décrivons, dans la section suivante, notre nouvel algorithme permettant de parcourir et analyser tous les discours possibles d'un scénario représenté par un tel séquent de logique linéaire, ce qui permet d'évaluer la qualité du scénario modélisé.

IV.6.2.3. Nouvel algorithme permettant de parcourir et analyser toutes les preuves d'un modèle qui est conforme au nouveau métamodèle d'histoire en logique linéaire

En vue de parcourir et analyser toutes les preuves d'un modèle qui est conforme à notre nouveau métamodèle d'histoire en logique linéaire (voir Figure IV.16), nous avons adapté l'algorithme exécuté par le module d'analyse du système auteur (voir Figure IV.5) aux changements de ce métamodèle d'histoire. Voici ci-dessous les règles essentielles du nouvel algorithme :

- un événement/action est exécutable si et seulement si l'ensemble des états de sa précondition (si ce sont des états avec paramètres, leurs paramètres doivent être remplacés par des constantes/objets appropriés) est un sous-ensemble de la liste d'états disponibles courants ;
- l'exécution d'un événement/action revient à remplacer tous les états dans la liste des états disponibles courants, correspondant à l'ensemble des états de sa précondition, par l'ensemble des états de son effet ;
- si à un certain moment, il y a plusieurs événements/actions exécutables, ces événements/actions créent les discours/branches différents dans le graphe des preuves ;
- afin d'éviter des boucles dans le scénario, un événement/action (avec un ensemble de constantes/objets) n'est utilisé qu'une fois pour chacun des discours possibles ;
- la création d'un discours consiste à réécrire un séquent successivement (en exécutant, à chaque étape, un événement/action exécutable) jusqu'à ce que :
    - la liste des états disponibles courants du séquent comprenne l'ensemble des états d'une de ses sorties → le séquent est prouvé, et donc le discours/branche courant est réussi ;
    - il ne reste aucun événement/action exécutable → le séquent n'est pas prouvé, et donc le discours/branche courant est échoué.

Ainsi, grâce au nouvel algorithme décrit ci-dessus, le module d'analyse de notre système auteur peut parcourir et analyser tous les discours possibles dans le scénario courant. Il peut donc donner les informations concernant les discours réussis/échoués dans le scénario et les événements/actions exécutés dans chaque discours. Par ailleurs, comme cet algorithme permet aussi de fournir la liste des états disponibles du séquent à chaque instant de chacun des discours, le module d'analyse peut vérifier la satisfaction des discours dans le scénario pour les contraintes suivantes :

- *at-end (StateWithoutParameter s)* : L'état *s* doit être dans la liste des états disponibles au moment final de tous les discours.
- *always (StateWithoutParameter s)* : L'état *s* doit être toujours dans la liste des états disponibles à tous les instants de tous les discours.
- *sometime (StateWithoutParameter s)* : L'état *s* doit être au moins une fois dans la liste des états disponibles pour tous les discours.
- *at-most-once (StateWithoutParameter s)* : Pour tous les discours, si l'état *s* est dans la liste des états disponibles et s'il sort de cette liste, alors il ne peut plus être présent dans la liste.

- *sometime-after (StateWithoutParameter s1, StateWithoutParameter s2)* : Pour tous les discours, l'état *s2* doit être au moins une fois dans la liste des états disponibles après l'arrivée de l'état *s1* dans cette liste.
- *sometime-before (StateWithoutParameter s1, StateWithoutParameter s2)* : Pour tous les discours, l'état *s2* doit être au moins une fois dans la liste des états disponibles avant l'apparition de l'état *s1* dans cette liste.

Ainsi, avec les résultats obtenus ci-dessus, qui sont fournis par le module d'analyse de notre système auteur, l'utilisateur peut évaluer la qualité du scénario modélisé et il peut donc produire un scénario valide d'une histoire.

IV.6.2.4. Bilan

Dans cette section, nous avons donné le deuxième exemple d'application décrivant la manière dont l'utilisateur peut, grâce à notre système auteur, produire un modèle PDDL valide représentant un scénario d'une histoire, même s'il n'a aucune connaissance en logique linéaire. Pour cela, nous avons (1) proposé un métamodèle d'histoire en PDDL ; (2) adapté notre métamodèle d'histoire en logique linéaire donné dans la Figure IV.1 ; (3) proposé un algorithme de transformation de modèle pour transformer un modèle d'histoire en PDDL à un modèle d'histoire en logique linéaire, et développé un traducteur implémentant cet algorithme ; (4) adapté notre algorithme, permettant de parcourir et d'analyser un scénario, aux changements du métamodèle de l'histoire décrite en logique linéaire, et modifié le module d'analyse du système auteur de façon correspondante à cette adaptation.

La démarche, qui est supportée par les outils que nous avons développé, pour produire un modèle PDDL valide, est la suivante : tout d'abord, l'utilisateur modélise l'histoire selon le métamodèle d'histoire en PDDL, et il reçoit donc un modèle en PDDL qui représente le scénario modélisé. Ensuite, l'utilisateur fait transformer automatiquement, par le traducteur, le modèle en PDDL obtenu en un modèle correspondant en logique linéaire. Enfin, il emploie le module d'analyse de notre système auteur en vue de parcourir et analyser tous les discours possibles du scénario représenté par ce modèle en logique linéaire. Le module d'analyse fournit les informations qualitatives et statistiques sur le scénario modélisé grâce auxquelles il est possible pour l'utilisateur d'estimer la qualité du scénario courant. Après cela, l'utilisateur peut corriger/améliorer directement le modèle en PDDL, et refaire les étapes ci-dessus si nécessaire (dans le cas où la qualité du scénario modélisé n'est pas suffisamment bonne). Ce processus est répété dans la phase de construction de scénario jusqu'à ce que l'utilisateur reçoive un scénario qui est le plus pertinent pour ses objectifs (que le scénario soit valide).

IV.7. Conclusion

Nous avons présenté, dans ce chapitre, la démarche et le système auteur que nous avons construits, ce qui aide l'utilisateur à produire des scénarios de bonne qualité, qui sont modélisés en logique linéaire. Nous sommes partis du constat que la production d'un système de pilotage de narration interactive fait intervenir trois différents acteurs : l'auteur qui définit le scénario du jeu, le game designer qui donne la conception du jeu en vue de mettre en œuvre ce scénario, et l'informaticien qui réalise le jeu selon la conception demandée. Ainsi, nous devons fournir des outils permettant à l'auteur, qui n'est pas un spécialiste de la modélisation

en logique linéaire, de pouvoir produire un scénario de bonne qualité, qui est modélisé en logique linéaire et qui est exécutable par un système de pilotage.

En effet, avec l'aide du système auteur que nous avons développé, mettant en œuvre la solution proposée dans cette thèse et fournissant des éléments graphiques appropriés, l'utilisateur peut, même s'il n'a aucune connaissance en logique linéaire, produire un scénario d'une histoire qui est riche, valide et opérationnel. Le séquent de logique linéaire, correspondant au scénario produit, est automatiquement généré et est exprimé par un modèle conforme au métamodèle du calcul des séquents. Il est ensuite employé comme entrée d'un système de pilotage assurant le contrôle de la gestion du déroulement du jeu selon le scénario modélisé, grâce au mécanisme de déduction automatique du séquent.

Pour cela, nous avons proposé les modèles et les algorithmes suivants, qui sont implémentés par notre système auteur : (1) un modèle d'histoire qui ne nécessite pas de connaissances en logique linéaire, (2) un modèle de contraintes, (3) un algorithme de parcours de toutes les preuves d'un modèle d'histoire, (4) un modèle de séquent utilisé comme modèle intermédiaire, (5) un modèle de scénario exécutable conforme au métamodèle du calcul des séquents, et (6) deux algorithmes de transformation de modèle pour transformer un modèle d'histoire à un modèle de scénario exécutable.

Nous avons aussi donné ensuite deux cas d'application de notre système auteur. Le premier est un prototype de système de pilotage de narration interactive que nous avons construit, qui utilise les scénarios/séquents valides produits par le système auteur en entrée. Le mécanisme de déduction automatique des séquents aide le système de pilotage à exécuter ces scénarios valides. Ainsi, l'évolution des jeux vidéo, qui sont dirigés par un tel système de pilotage, respecte les effets souhaités par les auteurs, et en même temps, est influencée par les actions du joueur. Par ailleurs, puisque les composants chargés du pilotage dans le système de pilotage proposé sont dissociés des composants chargés du jeu (environnement virtuel), notre système de pilotage ne doit pas gérer les aspects graphiques et il peut être compatible avec plusieurs supports divers.

Pour le deuxième cas d'application, nous avons décrit la manière dont l'utilisateur peut, grâce à notre système auteur, produire un modèle PDDL valide représentant un scénario d'une histoire, même s'il n'est pas un spécialiste de logique linéaire. Pour cela, nous avons proposé un métamodèle d'histoire en PDDL, et adapté notre solution utilisant la logique linéaire aux éléments de modélisation de ce métamodèle.

Dans le chapitre suivant, nous donnons deux exemples concrets ayant pour objectifs d'illustrer et également de valider l'approche de logique linéaire que nous avons proposée dans le cadre de cette thèse.

CHAPITRE V - Application des Méthodes et Outils sur Deux Exemples

V.1. Introduction

Au cours des chapitres précédents, nous avons défini une méthode de validation de scénario, qui constitue la base d'un système auteur permettant de générer un modèle formel de scénarios valides. Nous avons ensuite montré comment utiliser ce modèle pour piloter le déroulement d'une application de type jeu avec narration interactive.

Par conséquent, nous pouvons produire des jeux vidéo mettant en œuvre une narration interactive, qui sont dirigés par le système de pilotage proposé. Leur évolution respecte donc les effets souhaités par les auteurs, et en même temps, est influencée par les actions du joueur.

Dans ce chapitre, nous donnons deux exemples concrets ayant pour objectifs d'illustrer et également de valider les résultats obtenus ci-dessus. Le premier est un extrait d'un jeu éducatif (nous l'appelons « Jeu d'Apprentissage des Dangers de l'Électricité (JADE) ») expliquant comment appliquer notre système auteur afin de produire un scénario de jeu valide, qui est exprimé par un séquent de logique linéaire dont la représentation est conforme au métamodèle du calcul des séquents.

Dans le second exemple, nous décrivons le processus de production complet d'un prototype de jeu vidéo basé sur l'histoire « Le Petit Chaperon rouge », mettant en œuvre le prototype de système de pilotage que nous avons proposé. Ceci nous permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur, ce qui constitue l'objectif de nos travaux de recherche.

V.2. Jeu d'apprentissage des dangers de l'électricité

Cette section présente un exemple illustrant concrètement la production d'un scénario de jeu valide. Ce processus est composé de 4 étapes : (1) modéliser le jeu par l'éditeur de scénario, (2) générer le modèle d'histoire exprimé en XML par le module de prétraitement, (3) évaluer la qualité du scénario courant à travers le module d'analyse et après le valider, (4) générer le séquent de logique linéaire, grâce au module de génération, qui représente le scénario validé et qui est conforme au métamodèle du calcul des séquents. Par ailleurs, car l'exemple employé dans cette section n'est qu'un extrait d'un jeu simple et de complexité maîtrisable, son objectif est seulement d'aider les lecteurs à se représenter plus facilement notre solution pour la validation d'un scénario. Par conséquent, nous n'avons pas réalisé toutes les étapes de la démarche donnée dans la Figure IV.5 – chapitre IV, telles que l'établissement des contraintes à vérifier ou la création des informations qualitatives et statistiques du scénario courant (celles qui devront être complètement exécutées pour valider le scénario des jeux complexes). De façon similaire, la définition d'une structure de discours ainsi que l'attribution d'ordres de priorité aux événements/actions pour ce jeu ne sont pas nécessaires.

Notre exemple est extrait d'un jeu éducatif dont l'objectif concerne l'apprentissage des risques d'électrocution à la maison. Les auteurs/développeurs du jeu prévoient à un parcours dans la

cuisine. Le système de pilotage démarre une stratégie dans laquelle sont présentes les possibilités d'électrocution de l'avatar du joueur, liées aux appareils tels que réfrigérateur, four à micro-ondes, cuisinière électrique... Cependant, au lieu d'aller à la cuisine, d'autres choix sont possibles, par exemple, rester dans la pièce initiale pour travailler ou aller à la salle de bain. Que ce passera-t-il ?

V.2.1. Modélisation du jeu par l'éditeur de scénario

Nous illustrons ci-dessous l'application de l'éditeur en vue de décrire graphiquement un scénario du JADE, à travers les manipulations « glisser/déposer » simples. Puisqu'il n'y a pas de structure de discours pour cette histoire, son scénario est spécifié par 5 diagrammes : états, entrées, événements/actions, choix et sorties, ils sont construits tour à tour comme suit :

- Diagramme d'états : Ce diagramme (voir Figure V.1) représente les états du jeu et du joueur, dans lesquels Gi et Pi sont les états disponibles initiaux.

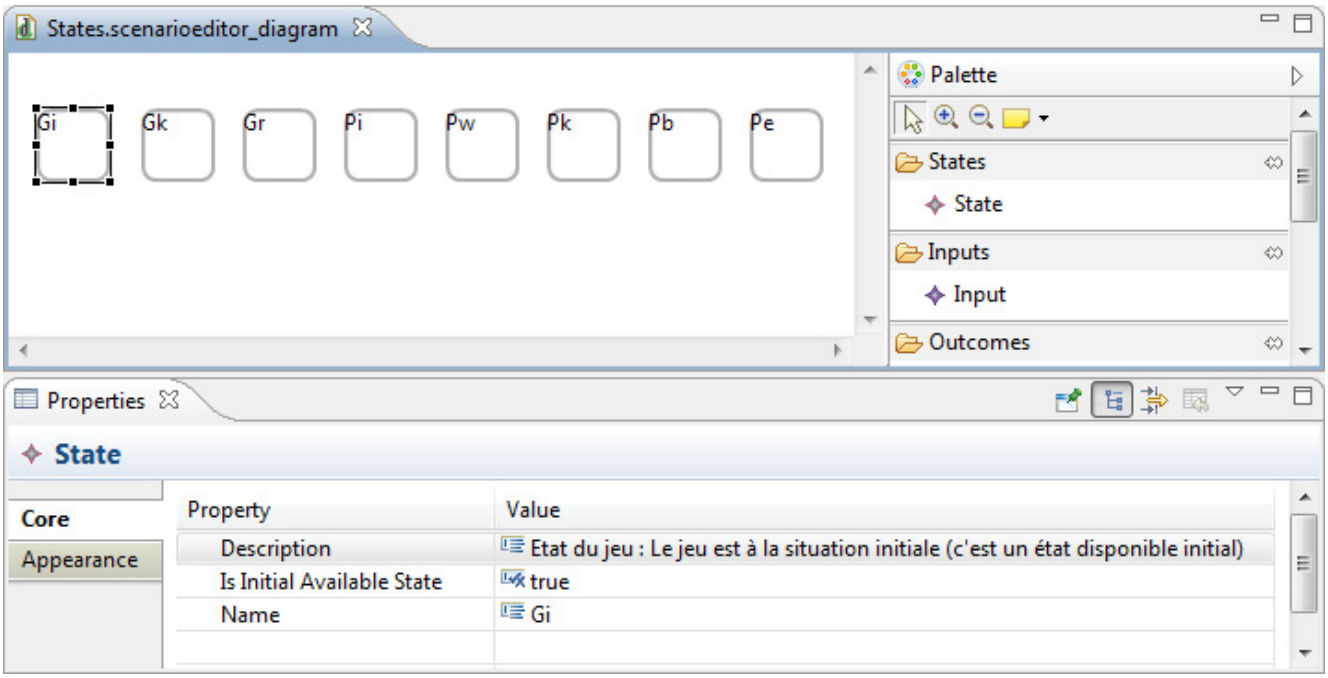


Figure V.1. Diagramme d'états du JADE dans l'éditeur de scénario, la section « Properties » décrit les propriétés de l'état Gi.

Name	IsInitial Available State	Description
Gi	True	État du jeu : Le jeu est à la situation initiale (c'est un état disponible initial)
Gk	False	État du jeu : Le système de pilotage démarre la stratégie de mise en danger d'électrocution pour le joueur dans la cuisine
Gr	False	État du jeu : Le jeu atteint l'objectif (le joueur est électrocuté)

Pi	True	État du joueur : Le joueur est dans la situation initiale (c'est un état disponible initial)
Pw	False	État du joueur : Le joueur travaille à la pièce initiale
Pk	False	État du joueur : Le joueur est dans la cuisine
Pb	False	État du joueur : Le joueur est dans la salle de bain
Pe	False	État du joueur : Le joueur reçoit une décharge électrique

- Diagramme d'entrées : Ce diagramme (voir Figure V.2) représente les entrées du joueur (ses choix d'action)

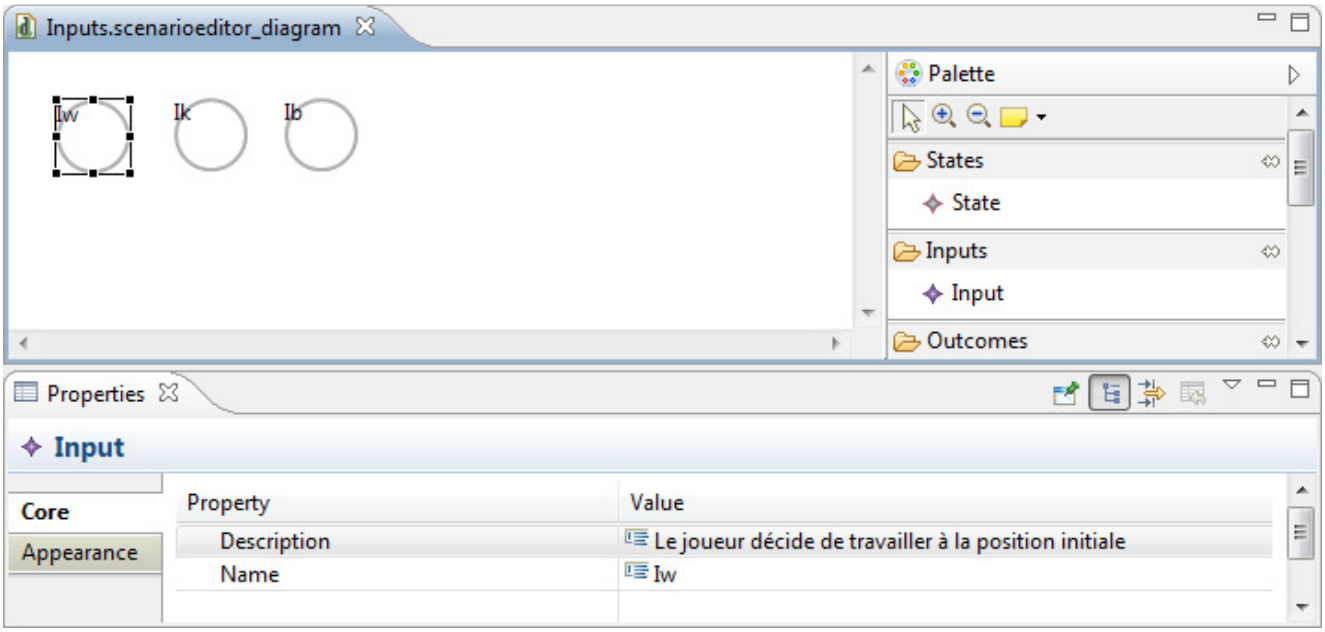


Figure V.2. Diagramme d'entrées du JADE dans l'éditeur de scénario, la section « Properties » décrit les propriétés de l'entrée Iw.

Name	**Description**
Iw	Le joueur décide de travailler dans la pièce initiale
Ik	Le joueur décide d'aller dans la cuisine
Ib	Le joueur décide d'aller dans la salle de bain

- Diagramme d'événements/actions : Ce diagramme (voir Figure V.3) représente les événements/actions du jeu (l'ordre de priorité des événements/actions n'est pas nécessaire puisque le scénario modélisé est trop simple).

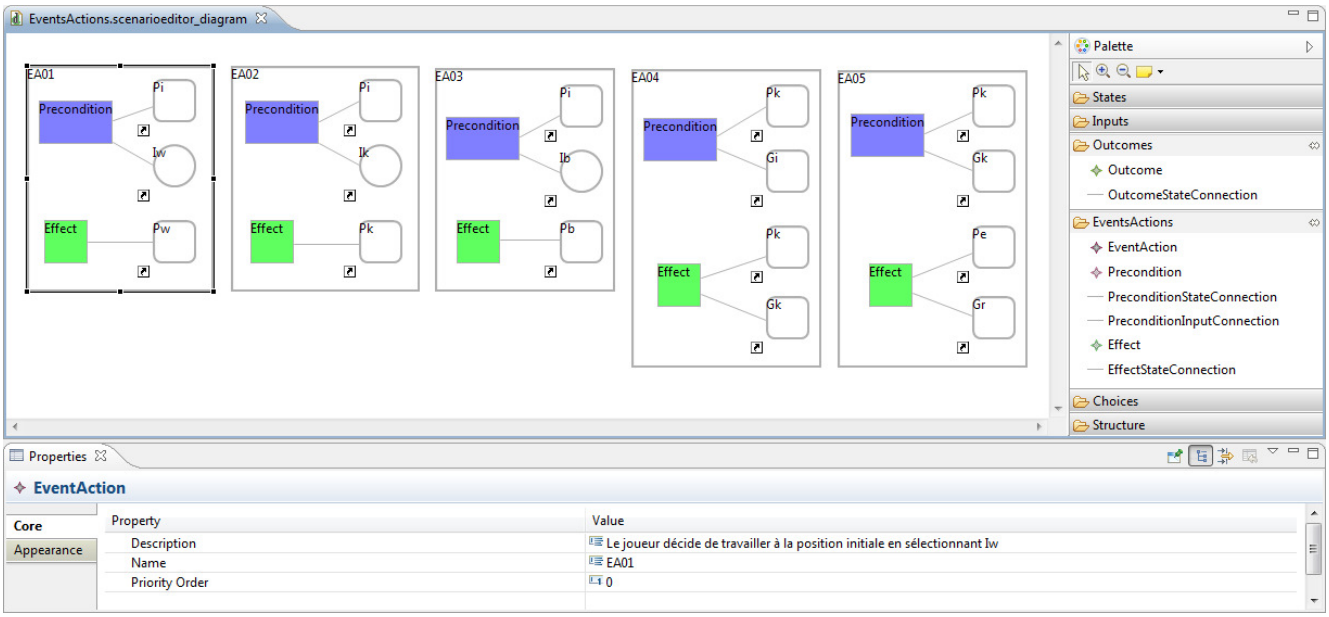


Figure V.3. Diagramme d'événements/actions du JADE dans l'éditeur de scénario, la section « Properties » décrit les propriétés de l'événement/action EA01.

Name	Description	Priority Order	Precondition	Effect
EA01	Le joueur décide de travailler dans la pièce initiale en sélectionnant Iw	0	Pi, Iw	Pw
EA02	Le joueur décide d'aller de la pièce initiale à la cuisine en sélectionnant Ik	0	Pi, Ik	Pk
EA03	Le joueur décide d'aller de la pièce initiale à la salle de bain en sélectionnant Ib	0	Pi, Ib	Pb
EA04	Le système de pilotage démarre la stratégie pour électrocuter le joueur dans la cuisine	0	Pk, Gi	Pk, Gk
EA05	Le joueur reçoit une décharge électrique	0	Pk, Gk	Pe, Gr

- Diagramme de choix : Il n'y a qu'un choix dans le JADE, c'est celui qui est décidé par le joueur (exécuter soit EA01 soit EA02 soit EA03), donc l'attribut « Percentage » de ses trois possibilités a la valeur 0. La Figure V.4 donne ce diagramme de choix.

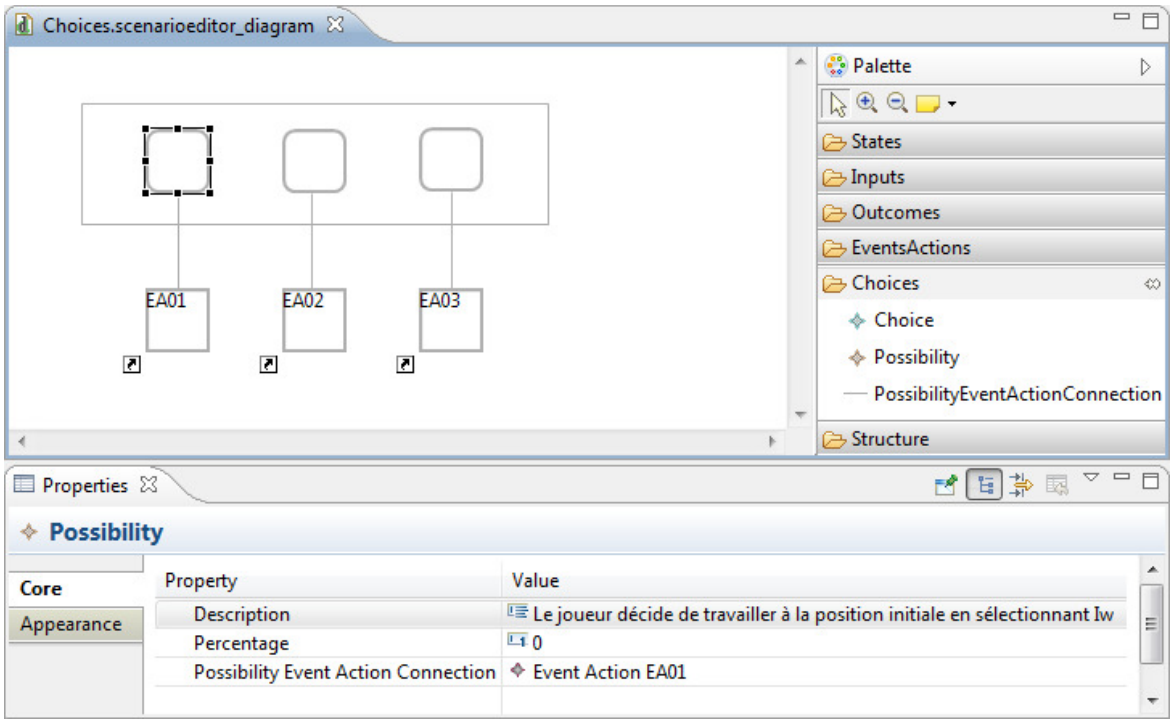


Figure V.4. Diagramme de choix du JADE dans l'éditeur de scénario (il n'y a qu'un choix), la section « Properties » décrit les propriétés de la première possibilité - EA01 de ce choix.

Description	Le joueur doit décider soit de travailler dans la pièce initiale soit d'aller de la pièce initiale à la cuisine soit d'aller de la pièce initiale à la salle de bain				
Possibility		***Possibility***		***Possibility***	
Description	*Event Action*	*Description*	*Event Action*	*Description*	*Event Action*
Le joueur décide de travailler dans la pièce initiale en sélectionnant Iw	EA01	Le joueur décide d'aller de la pièce initiale à la cuisine en sélectionnant Ik	EA02	Le joueur décide d'aller de la pièce initiale à la salle de bain en sélectionnant Ib	EA03

- Diagramme de sorties : Il n'y a qu'une sortie dans le JADE, elle est composée de deux états Pe et Gr. La Figure V.5 donne ce diagramme de sorties.

Name	**Description**	**States**
O	Le joueur reçoit le choc électrique	Pe, Gr

Ainsi, grâce à l'éditeur de scénario, les utilisateurs peuvent décrire graphiquement un scénario pour le JADE, même s'ils n'ont aucune connaissance de la logique linéaire, en créant 5 diagrammes (états, entrées, événements/actions, choix et sorties) à travers les manipulations « glisser/déposer » simples. Les sections suivantes montrent comment les

utilisateurs évaluent la qualité du scénario courant grâce au module d'analyse, et donc ils peuvent corriger la modélisation si nécessaire en vue d'obtenir un scénario valide.

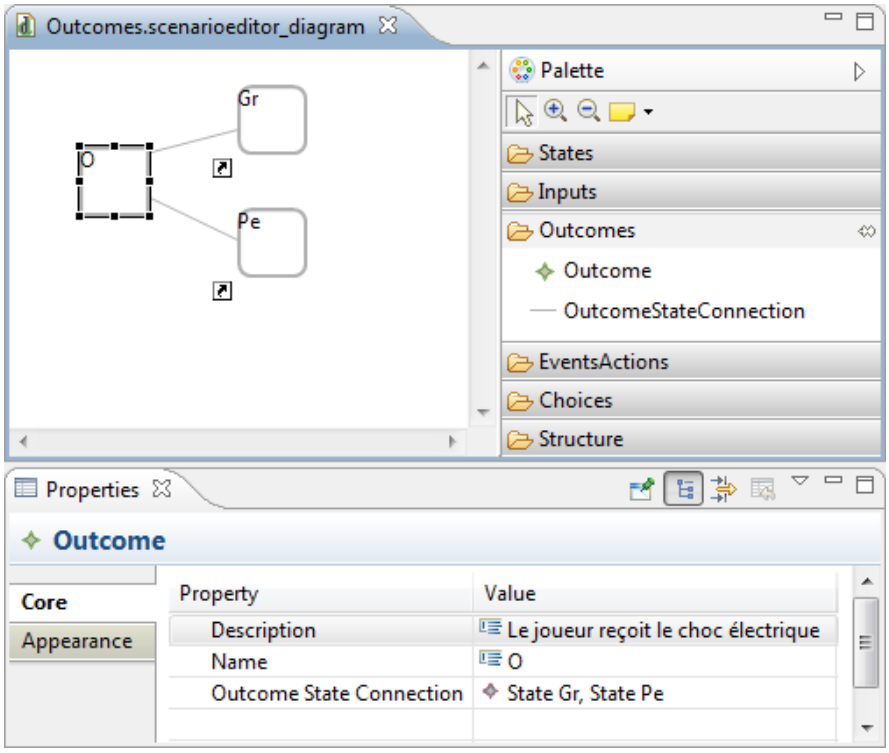


Figure V.5. Diagramme de sorties du JADE dans l'éditeur de scénario (il n'y a qu'une sortie), la section « Properties » décrit ses propriétés.

V.2.2. Création du modèle d'histoire exprimé en XML par le module de prétraitement

Néanmoins, avant d'employer le module d'analyse pour évaluer la qualité du scénario courant, nous devons générer le modèle d'histoire exprimé en XML par le module de prétraitement.

```
<scenarioeditor:Scenario xmi:version="2.0">
  <States Name="Gi" Description="Etat du jeu : Le jeu est a la situation initiale (c'est un etat disponible initial)" IsInitialAvailableState="true"/>
  <States Name="Gk" Description="Etat du jeu : Le pilote demarre la strategie de causer des chocs electriques pour le joueur dans la cuisine" IsInitialAvailableState="false"/>
  <States Name="Gr" Description="Etat du jeu : Le jeu atteint l'objectif (le joueur recoit le choc electrique)" IsInitialAvailableState="false"/>
  <States Name="Pi" Description="Etat du joueur : Le joueur est a la situation initiale (c'est un etat disponible initial)" IsInitialAvailableState="true"/>
  <States Name="Pw" Description="Etat du joueur : Le joueur travaille a la position initiale" IsInitialAvailableState="false"/>
  <States Name="Pk" Description="Etat du joueur : Le joueur est dans la cuisine" IsInitialAvailableState="false"/>
  <States Name="Pb" Description="Etat du joueur : Le joueur est dans la salle de bain" IsInitialAvailableState="false"/>
  <States Name="Pe" Description="Etat du joueur : Le joueur recoit le choc electrique" IsInitialAvailableState="false"/>
</scenarioeditor:Scenario>
```

Figure V.6. Représentation XML des états dans le modèle automatiquement créé grâce à GMF.

```
<root>
  <State Description="Etat du jeu : Le jeu est a la situation initiale (c'est un etat disponible initial)" IsInitialAvailableState="true" Name="Gi"/>
  <State Description="Etat du jeu : Le pilote demarre la strategie de causer des chocs electriques pour le joueur dans la cuisine" IsInitialAvailableState="false" Name="Gk"/>
  <State Description="Etat du jeu : Le jeu atteint l'objectif (le joueur recoit le choc electrique)" IsInitialAvailableState="false" Name="Gr"/>
  <State Description="Etat du joueur : Le joueur est a la situation initiale (c'est un etat disponible initial)" IsInitialAvailableState="true" Name="Pi"/>
  <State Description="Etat du joueur : Le joueur travaille a la position initiale" IsInitialAvailableState="false" Name="Pw"/>
  <State Description="Etat du joueur : Le joueur est dans la cuisine" IsInitialAvailableState="false" Name="Pk"/>
  <State Description="Etat du joueur : Le joueur est dans la salle de bain" IsInitialAvailableState="false" Name="Pb"/>
  <State Description="Etat du joueur : Le joueur recoit le choc electrique" IsInitialAvailableState="false" Name="Pe"/>
</root>
```

Figure V.7. Représentation XML des états dans le modèle d'histoire.

- Représentation XML des états dans le modèle d'histoire : La Figure V.6 montre la réprésentation initiale des états produite suite à la transformation automatique GMF, et la Figure V.7 indique la représentation XML des états dans le modèle d'histoire engendré par le module de prétraitement (le modèle exprimé en XML automatiquement créé grâce à GMF est conforme au modèle *ecore* de l'éditeur de scénario donné dans la Figure IV.6 – chapitre IV, le modèle d'histoire exprimé en XML est conforme au métamodèle donné dans la Figure IV.1 – chapitre IV).
- Représentation XML des entrées, des choix, des événements/actions et des sorties dans le modèle d'histoire : Voir Figures V.8 → V.15 respectivement.

```
<scenarioeditor:Scenario xmi:version="2.0">
  <Inputs Name="Iw" Description="Le joueur decide de travailler a la position initiale"/>
  <Inputs Name="Ik" Description="Le joueur decide d'aller a la cuisine"/>
  <Inputs Name="Ib" Description="Le joueur decide d'aller a la salle de bain"/>
</scenarioeditor:Scenario>
```

Figure V.8. Représentation XML des entrées dans le modèle automatiquement créé grâce à GMF.

```
<root>
  <Input Description="Le joueur decide de travailler a la position initiale" Name="Iw"/>
  <Input Description="Le joueur decide d'aller a la cuisine" Name="Ik"/>
  <Input Description="Le joueur decide d'aller a la salle de bain" Name="Ib"/>
</root>
```

Figure V.9. Représentation XML des entrées dans le modèle d'histoire.

```
<scenarioeditor:Scenario xmi:version="2.0">
  <Choices Description="Le joueur doit decider soit de travailler a la position initiale soit d'aller de la position initiale a la cuisine soit d'aller de la position initiale a la salle de bain">
    <ChoicePossibility Percentage="0" Description="Le joueur decide de travailler a la position initiale en selectionnant Iw">
      <PossibilityEventActionConnection href="EventsActions.scenarioeditor#//@EventsActions.0"/>
    </ChoicePossibility>
    <ChoicePossibility Percentage="0" Description="Le joueur decide d'aller de la position initiale a la cuisine en selectionnant Ik">
      <PossibilityEventActionConnection href="EventsActions.scenarioeditor#//@EventsActions.1"/>
    </ChoicePossibility>
    <ChoicePossibility Percentage="0" Description="Le joueur decide d'aller de la position initiale a la salle de bain en selectionnant Ib">
      <PossibilityEventActionConnection href="EventsActions.scenarioeditor#//@EventsActions.2"/>
    </ChoicePossibility>
  </Choices>
</scenarioeditor:Scenario>
```

Figure V.10. Représentation XML des choix dans le modèle automatiquement créé grâce à GMF.

```
<root>
  <Choice Description="Le joueur doit decider soit de travailler a la position initiale soit d'aller de la position initiale a la cuisine soit d'aller de la position initiale a la salle de bain">
    <Possibility Description="Le joueur decide de travailler a la position initiale en selectionnant Iw" Percentage="0">
      <EventAction Name="EA01"/>
    </Possibility>
    <Possibility Description="Le joueur decide d'aller de la position initiale a la cuisine en selectionnant Ik" Percentage="0">
      <EventAction Name="EA02"/>
    </Possibility>
    <Possibility Description="Le joueur decide d'aller de la position initiale a la salle de bain en selectionnant Ib" Percentage="0">
      <EventAction Name="EA03"/>
    </Possibility>
  </Choice>
</root>
```

Figure V.11. Représentation XML des choix dans le modèle d'histoire.

```
<scenarioeditor:Scenario xmi:version="2.0">
  <EventsActions Name="EA01" Description="Le joueur decide de travailler a la position initiale en selectionnant Iw" PriorityOrder="0">
    <EventActionPrecondition>
      <PreconditionStateConnection href="States.scenarioeditor#//@States.3"/>
      <PreconditionInputConnection href="Inputs.scenarioeditor#//@Inputs.0"/>
    </EventActionPrecondition>
    <EventActionEffect>
      <EffectStateConnection href="States.scenarioeditor#//@States.4"/>
    </EventActionEffect>
  </EventsActions>
  <EventsActions Name="EA02" Description="Le joueur decide d'aller de la position initiale a la cuisine en selectionnant Ik" PriorityOrder="0">
    <EventActionPrecondition>
      <PreconditionStateConnection href="States.scenarioeditor#//@States.3"/>
      <PreconditionInputConnection href="Inputs.scenarioeditor#//@Inputs.1"/>
    </EventActionPrecondition>
    <EventActionEffect>
      <EffectStateConnection href="States.scenarioeditor#//@States.5"/>
    </EventActionEffect>
  </EventsActions>
  <EventsActions Name="EA03" Description="Le joueur decide de d'aller de la position initiale a la salle de bain en selectionnant Ib" PriorityOrder="0">
    <EventActionPrecondition>
      <PreconditionStateConnection href="States.scenarioeditor#//@States.3"/>
      <PreconditionInputConnection href="Inputs.scenarioeditor#//@Inputs.2"/>
    </EventActionPrecondition>
    <EventActionEffect>
      <EffectStateConnection href="States.scenarioeditor#//@States.6"/>
    </EventActionEffect>
  </EventsActions>
  <EventsActions Name="EA04" Description="Le pilote demarre la strategie de causer des chocs electriques pour le joueur dans la cuisine" PriorityOrder="0"> ... </EventsActions>
  <EventsActions Name="EA05" Description="Le joueur recoit le choc electrique" PriorityOrder="0">
    <EventActionPrecondition>
      <PreconditionStateConnection href="States.scenarioeditor#//@States.5"/>
      <PreconditionStateConnection href="States.scenarioeditor#//@States.1"/>
    </EventActionPrecondition>
    <EventActionEffect>
      <EffectStateConnection href="States.scenarioeditor#//@States.7"/>
      <EffectStateConnection href="States.scenarioeditor#//@States.2"/>
    </EventActionEffect>
  </EventsActions>
</scenarioeditor:Scenario>
```

Figure V.10. Représentation XML des événements/actions dans le modèle automatiquement créé grâce à GMF.

```
<root>
  <EventAction Description="Le joueur decide de travailler a la position initiale en selectionnant Iw" Name="EA01" PriorityOrder="0">
    <Precondition>
      <State Name="Pi"/>
      <Input Name="Iw"/>
    </Precondition>
    <Effect>
      <State Name="Pw"/>
    </Effect>
  </EventAction>
  <EventAction Description="Le joueur decide d'aller de la position initiale a la cuisine en selectionnant Ik" Name="EA02" PriorityOrder="0">
    <Precondition>
      <State Name="Pi"/>
      <Input Name="Ik"/>
    </Precondition>
    <Effect>
      <State Name="Pk"/>
    </Effect>
  </EventAction>
  <EventAction Description="Le joueur decide de d'aller de la position initiale a la salle de bain en selectionnant Ib" Name="EA03" PriorityOrder="0"> ... </EventAction>
  <EventAction Description="Le pilote demarre la strategie de causer des chocs electriques pour le joueur dans la cuisine" Name="EA04" PriorityOrder="0"> ... </EventAction>
  <EventAction Description="Le joueur recoit le choc electrique" Name="EA05" PriorityOrder="0">
    <Precondition>
      <State Name="Pk"/>
      <State Name="Gk"/>
    </Precondition>
    <Effect>
      <State Name="Pe"/>
      <State Name="Gr"/>
    </Effect>
  </EventAction>
</root>
```

Figure V.11. Représentation XML des événements/actions dans le modèle d'histoire.

```
<scenarioeditor:Scenario xmi:version="2.0">
  <Outcomes Name="O" Description="Le joueur recoit le choc electrique">
    <OutcomeStateConnection href="States.scenarioeditor#//@States.2"/>
    <OutcomeStateConnection href="States.scenarioeditor#//@States.7"/>
  </Outcomes>
</scenarioeditor:Scenario>
```

Figure V.14. Représentation XML des sorties dans le modèle automatiquement créé grâce à GMF.

```
<root>
  <Outcome Description="Le joueur recoit le choc electrique" Name="O">
    <State Name="Pe"/>
    <State Name="Gr"/>
  </Outcome>
</root>
```

Figure V.15. Représentation XML des sorties dans le modèle d'histoire.

Ainsi, grâce à ce module, on obtient le modèle d'histoire exprimé en XML représentant les diagrammes d'états, d'entrées, d'événements/actions, de choix et de sorties du JADE, ceux qui ont été construits dans l'éditeur de scénario. Ce modèle peut être employé comme l'entrée du module d'analyse permettant d'évaluer la qualité du scénario modélisé, ce point est abordé dans la section suivante.

V.2.3. Validation du scénario courant en évaluant sa qualité grâce au module d'analyse

À partir du modèle d'histoire, qui exprime les diagrammes d'états, d'entrées, d'événements/actions, de choix et de sorties du JADE, le module d'analyse construit le graphe des preuves qui représente tous les discours possibles (et leur résultat réussi ou échoué) dans le scénario. De plus, il analyse ce graphe afin de fournir les informations qualitatives et statistiques sur le scénario. Dans cet exemple vu la taille du jeu, nous ne nous intéressons pas à ces informations.

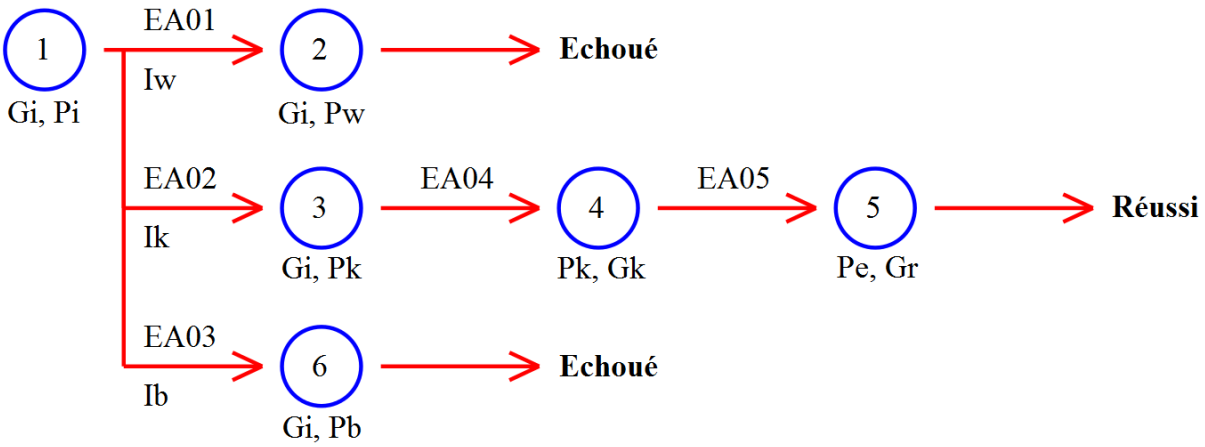


Figure V.16. Graphe des preuves automatiquement construit du JADE, où chaque nœud correspond aux états disponibles du séquent à un moment ; chaque arête est un événement/action exécuté (avec une entrée ou pas).

La Figure V.16 donne le graphe des preuves du scénario courant, où ses trois discours/branches sont les trois possibilités de choix dans le scénario (toutes sont décidées par le joueur) :

- deux échoués qui mènent aux terminaisons qui ne remplissent pas les objectifs des auteurs (le joueur reçoit le choc électrique), dans les cas où le joueur travaille dans la pièce initiale ou va de la pièce initiale à la salle de bain ;
- un réussi dans le cas où le joueur va de la pièce initiale à la cuisine.

Par conséquent, le scénario courant du jeu n'est pas valide. Afin de résoudre cela, nous avons deux options :

- soit supprimer les actions du joueur causant les terminaisons non satisfaisantes (EA01 – Travailler dans la pièce initiale et EA03 – Aller à la salle de bain), mais cela peut restreindre la liberté du joueur, donc nous ne sélectionnons pas cette possibilité ;
- soit enrichir le contenu de l'intrigue
    - si le joueur décide de travailler dans la pièce initiale, alors le système de pilotage lui demande d'aller à la cuisine (par exemple, un personnage non-joueur lui demande de prendre une pomme dans le réfrigérateur) ;
    - si le joueur décide d'aller à la salle de bain, alors le système de pilotage démarre une stratégie afin d'électrocuter le joueur dans la salle de bain (par des outils tels que le sèche-cheveux, une ampoule…).

Ainsi nous améliorons/corrigions la modélisation du jeu grâce à l'éditeur de scénario (c'est-à-dire nous modifions les diagrammes correspondants) comme suit :

- les diagrammes d'entrées, de choix et de sorties sont immuables ;
- ajouter deux nouveaux états au diagramme d'états (ce ne sont pas les états disponibles initiaux) comme montré dans la Figure V.17 ;

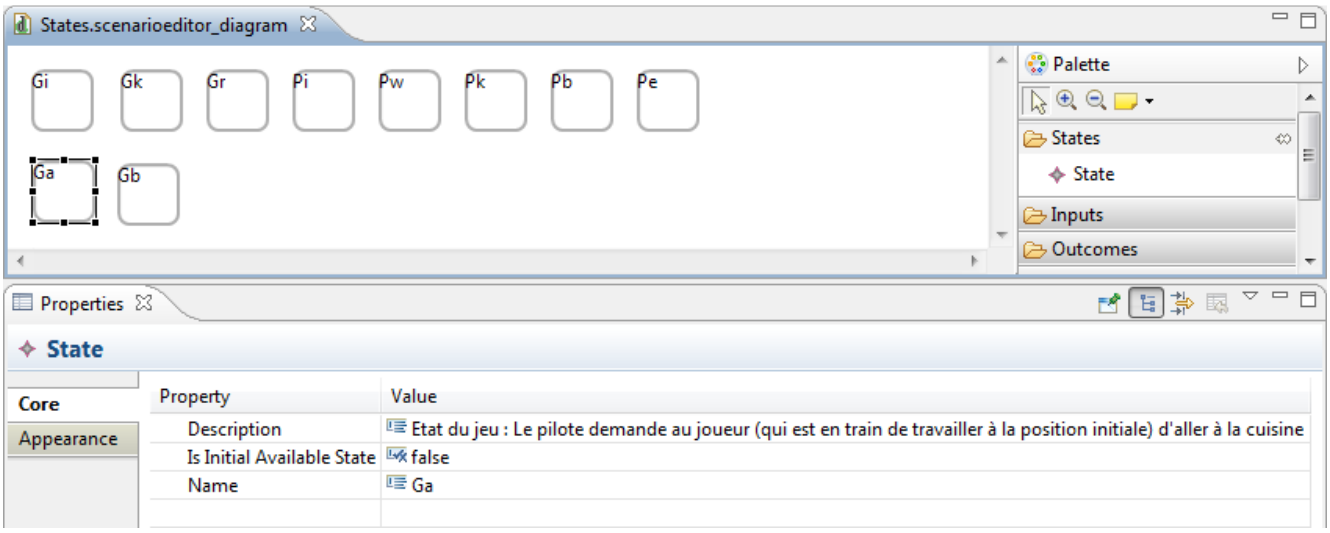


Figure V.17. Nouveau diagramme d'états du JADE dans l'éditeur de scénario, la section « Properties » décrit les propriétés du nouvel état Ga.

Name	IsInitial Available State	Description
Ga	False	État du jeu : Le système de pilotage demande au joueur (qui est en train de travailler à la pièce initiale) d'aller à la cuisine
Gb	False	État du jeu : Le système de pilotage démarre la stratégie pour électrocuter le joueur dans la salle de bain

- modifier le diagramme d'événements/actions comme montré dans la Figure V.18.

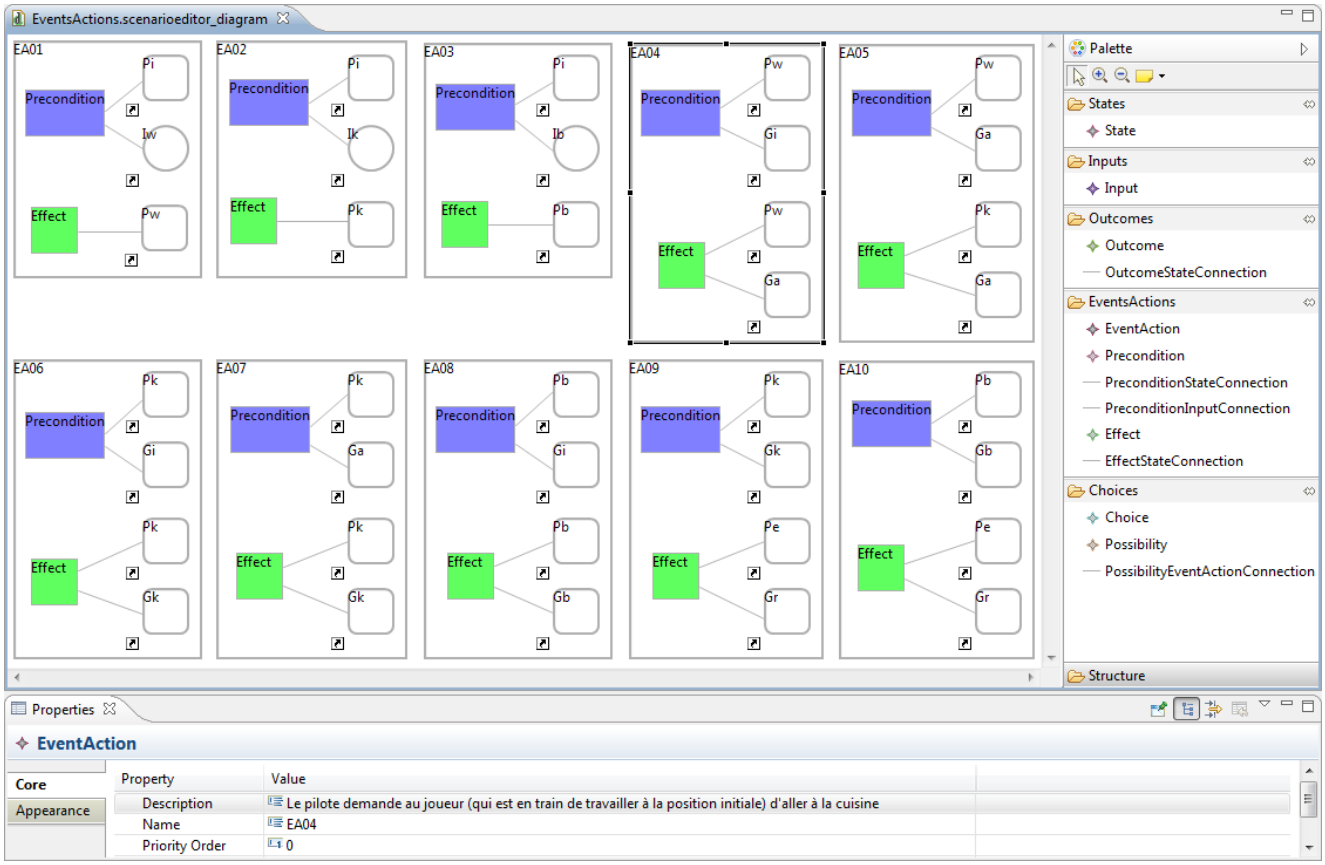


Figure V.18. Nouveau diagramme d'événements/actions du JADE dans l'éditeur de scénario, la section « Properties » décrit les propriétés du nouvel événement/action EA04.

Name	Description	Precondition	Effect
EA01	Le joueur décide de travailler dans la pièce initiale en sélectionnant Iw	Pi, Iw	Pw
EA02	Le joueur décide d'aller de la pièce initiale à la cuisine en sélectionnant Ik	Pi, Ik	Pk
EA03	Le joueur décide d'aller de la pièce initiale à la salle de bain en sélectionnant Ib	Pi, Ib	Pb

EA04	Le système de pilotage demande au joueur (qui est en train de travailler à la pièce initiale) d'aller à la cuisine	Pw, Gi	Pw, Ga
EA05	Le joueur (qui est en train de travailler à la pièce initiale) va à la cuisine selon la demande du système de pilotage	Pw, Ga	Pk, Ga
EA06	Après le joueur va de la pièce initiale à la cuisine en sélectionnant Ik, le système de pilotage démarre la stratégie afin de l'électrocuter dans cette pièce	Pk, Gi	Pk, Gk
EA07	Après le joueur va de la pièce initiale à la cuisine selon la demande du système de pilotage, le dernier démarre la stratégie afin de l'électrocuter dans cette pièce	Pk, Ga	Pk, Gk
EA08	Le système de pilotage démarre la stratégie afin d'électrocuter le joueur dans la salle de bain	Pb, Gi	Pb, Gb
EA09	Le joueur est électrocuté dans la cuisine	Pk, Gk	Pe, Gr
EA10	Le joueur est électrocuté dans la salle de bain	Pb, Gb	Pe, Gr

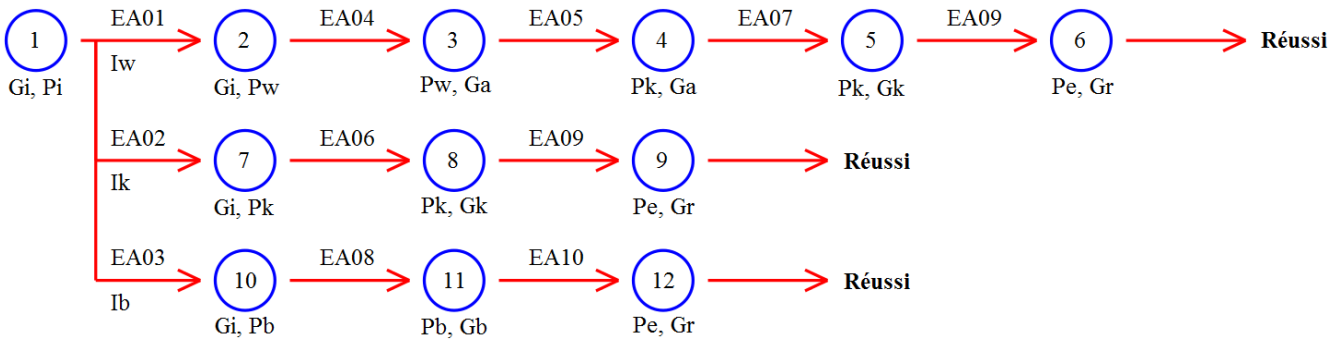


Figure V.19. Graphe des preuves correspondant au nouveau scénario du JADE après la phase de validation.

Dans la Figure V.19 présentant le graphe des preuves du jeu modifié, nous remarquons que, tous les discours/branches mènent aux terminaisons satisfaisantes (ce qui signifie que le joueur fini, dans tous les cas, par être électrocuté), et en même temps, la liberté du joueur est également garantie. Par conséquent, l'objectif des auteurs est atteint ou autrement dit, le nouveau scénario du jeu est valide.

La section suivante indique le séquent de logique linéaire exprimant ce scénario validé, ce qui est conforme au métamodèle du calcul des séquents et est automatiquement généré grâce au module de génération dans notre ensemble d'outils.

V.2.4. Génération du séquent de logique linéaire grâce au module de génération

À partir du modèle d'histoire exprimé en XML, qui est créé par le module de prétraitement et qui représente les diagrammes d'états, d'entrées, d'événements/actions, de choix, de sorties du scénario validé du JADE, le module de génération exécute consécutivement deux transformations de modèle afin de créer deux modèles du séquent de logique linéaire correspondant au scénario modélisé au moment initial du jeu :

- le premier est basé sur le métamodèle intermédiaire pour permettre la transformation dans le second (donc il est implicite pour les utilisateurs) : Gi, Pi, Iw ⊗ EA01 ⊗ EA04 ⊗ EA05 ⊗ EA07 ⊗ EA09 & Ik ⊗ EA02 ⊗ EA06 ⊗ EA09 & Ib ⊗ EA03 ⊗ EA08 ⊗ EA10 ⊢ Pe ⊗ Gr, il est exprimé par

```
<Sequent>
 <LeftPart>
   <InitialAvailableState Name="Gi"/>
   <InitialAvailableState Name="Pi"/>
   <WithExpression>
      <TimesExpression>
         <Input Name="Iw"/>
         <EventAction Name="EA01" PriorityOrder="0">
            <Precondition/>
               <State Name="Pi"/>
               <Input Name="Iw"/>
            <Precondition/>
            <Effect>
               <State Name="Pw"/>
            <Effect/>
         </EventAction>
         …
         <EventAction Name="EA09" PriorityOrder="0">
            …
         </EventAction>
      <TimesExpression/>
      <TimesExpression>
         <Input Name="Ik"/>
         <EventAction Name="EA02" PriorityOrder="0">
            …
         </EventAction>
         …
```

```
            <EventAction Name="EA09" PriorityOrder="0">
               …
            </EventAction>
         <TimesExpression/>
         <TimesExpression>
            <Input Name="Ib"/>
            <EventAction Name="EA03" PriorityOrder="0">
               …
            </EventAction>
            …
            <EventAction Name="EA10" PriorityOrder="0">
               …
            </EventAction>
         <TimesExpression/>
      <WithExpression/>
   </LeftPart>
   <RightPart>
      <Outcome Name="O">
         <State Name="Pe"/>
         <State Name="Gr"/>
      </Outcome>
    </RightPart>
  </Sequent>
```

- le second est conforme au métamodèle du calcul des séquents (l'entrée de cette transformation de modèle est la représentation XML du premier modèle ci-dessus) : Gi, Pi, Iw ⊗ (Pi ⊗ Iw ⊸ Pw) ⊗ (Pw ⊗ Gi ⊸ Pw ⊗ Ga) ⊗ (Pw ⊗ Ga ⊸ Pk ⊗ Ga) ⊗ (Pk ⊗ Ga ⊸ Pk ⊗ Gk) ⊗ (Pk ⊗ Gk ⊸ Pe ⊗ Gr) & Ik ⊗ (Pi ⊗ Ik ⊸ Pk) ⊗ (Pk ⊗ Gi ⊸ Pk ⊗ Gk) ⊗ (Pk ⊗ Gk ⊸ Pe ⊗ Gr) & Ib ⊗ (Pi ⊗ Ib ⊸ Pb) ⊗ (Pb ⊗ Gi ⊸ Pb ⊗ Gb) ⊗ (Pb ⊗ Gb ⊸ Pe ⊗ Gr) ⊢ Pe ⊗ Gr. Ce modèle est ensuite employé comme l'entrée d'un système de pilotage de narration interactive, il est représenté par le segment exprimé sous forme XML suivant :

```
<Sequent>
    <LeftPart>
        <AvailableState Name="Gi"/>
        <AvailableState Name="Pi"/>
        <Formula Order="1">
            <Element Name="Iw" Order="1" Type="Atom"/>
            <Element Name="times" Order="2" Type="Multiplicative Conjunction"/>
            <Element Name="(" Order="3" Type="Open Parenthesis" ParenthesisLevel="1"/>
            <Element Name="Pi" Order="4" Type="Atom"/>
            <Element Name="times" Order="5" Type="Multiplicative Conjunction"/>
            <Element Name="Iw" Order="6" Type="Atom"/>
            <Element Name="imply" Order="7" Type="Linear Implication"
            EventActionName="EA01" EventActionPriorityOrder="0"/>
            <Element Name="Pw" Order="8" Type="Atom"/>
            <Element Name=")" Order="9" Type="Close Parenthesis" ParenthesisLevel="1"/>
            <Element Name="times" Order="10" Type="Multiplicative Conjunction"/>
            <Element Name="(" Order="11" Type="Open Parenthesis" ParenthesisLevel="1"/>
            …
            <Element Name="imply" Order="15" Type="Linear Implication"
            EventActionName="EA04" EventActionPriorityOrder="0"/>
            …
            <Element Name=")" Order="19" Type="Close Parenthesis" ParenthesisLevel="1"/>
            <Element Name="times" Order="20" Type="Multiplicative Conjunction"/>
            …
        <Formula/>
    <LeftPart/>
    <RightPart>
        <Outcome Name="O">
            <State Name="Pe"/>
            <State Name="Gr"/>
        <Outcome/>
    <RightPart/>
<Sequent/>
```

V.2.5. Bilan

Dans cette section, nous avons présenté un exemple – Jeu d'apprentissage des dangers de l'électricité – illustrant concrètement la démarche utilisateur pour produire un scénario de jeu valide. Le processus comporte 4 étapes : (1) modéliser graphiquement le jeu par l'éditeur de scénario à travers les manipulations « glisser/déposer » simples, (2) créer le modèle d'histoire exprimé en XML au moyen du module de prétraitement, (3) évaluer la qualité du scénario courant à travers le module d'analyse et après le valider, (4) générer le séquent de logique linéaire grâce au module de génération, qui représente le scénario validé et qui est conforme au métamodèle du calcul des séquents.

Nous décrivons, dans la section suivante, le processus de production entier d'un jeu vidéo réel fondé sur l'histoire « Le Petit Chaperon rouge », mettant en œuvre le prototype de système de pilotage que nous avons proposé au chapitre précédent, ce qui permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur.

V.3. Jeu vidéo fondé sur l'histoire « Le Petit Chaperon rouge »

Afin d'illustrer toute notre approche de logique linéaire pour la production des jeux vidéo basés sur la narration interactive, nous avons construit un jeu vidéo réel fondé sur l'histoire « Le Petit Chaperon rouge » (PCr), qui a été utilisée comme une histoire partagée aux ateliers des conférences en 2009 (ICIDS 2009, Guimaraes, Portugal), 2008 (ICIDS 2008, Erfurt, Allemagne) et 2006 (TIDSE 2006, Darmstadt, Allemagne).[1] Ce jeu vidéo s'appuie sur la version de Grimms [3] à laquelle nous avons ajouté quelques options pour les personnages (joueur et non-joueur), en vue d'augmenter l'imprévisibilité ainsi que l'interactivité du jeu. En même temps, nous devons également garantir que le jeu présente des aspects éducatifs pour le joueur (en particulier des enfants), c'est pourquoi son scénario doit garantir des objectifs des auteurs. Ainsi, pendant le déroulement du jeu, dans tous les cas (le PCr dit, ou ne dit pas, au loup où est la maison de la grand-mère ; écoute, ou n'écoute pas, le loup contemplant le paysage dans le bois...), le loup mange toujours la grand-mère et le PCr. Cela peut aider le joueur à réaliser le danger, et présente un caractère pédagogique : ne jamais parler aux étrangers et ne jamais marcher seul dans des endroits dangereux.

Le schéma général du processus de production d'un jeu vidéo fondé sur la narration interactive dans notre approche est exprimé dans la Figure V.20. Le système auteur permet aux experts du domaine (auteurs) de créer le modèle du jeu et le scénario valide. Ce scénario est exprimé par un séquent de logique linéaire dont la représentation XML est conforme au métamodèle du calcul des séquents. À partir du scénario/séquent créé, les technologues (game designers et informaticiens) mettent en œuvre le prototype de système de pilotage que nous avons proposé au chapitre précédent, ce qui permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur. Les sections suivantes expliquent les points essentiels de ce processus.

V.3.1. Production d'un scénario valide du jeu

Cette section décrit la production d'un scénario valide du jeu en utilisant notre système auteur (celle qui, dans la réalité, est faite par les auteurs/experts du domaine). Car la modélisation de scénario détaillée du jeu est mise en Annexe D de la thèse, nous n'abordons ici que ses descriptions principales :

- Le jeu se déroule dans trois lieux : la maison du PCr, le bois et la maison de la grand-mère. Dans ce jeu, le joueur joue le PCr, tandis que le système de pilotage dirige les actions des quatre personnages non-joueurs : la mère, le loup, la grand-mère et le chasseur.

[1] Pour plus d'information, voir http://redcap.interactive-storytelling.de et http://www.icids.org (pages consultées le 14 février 2013)

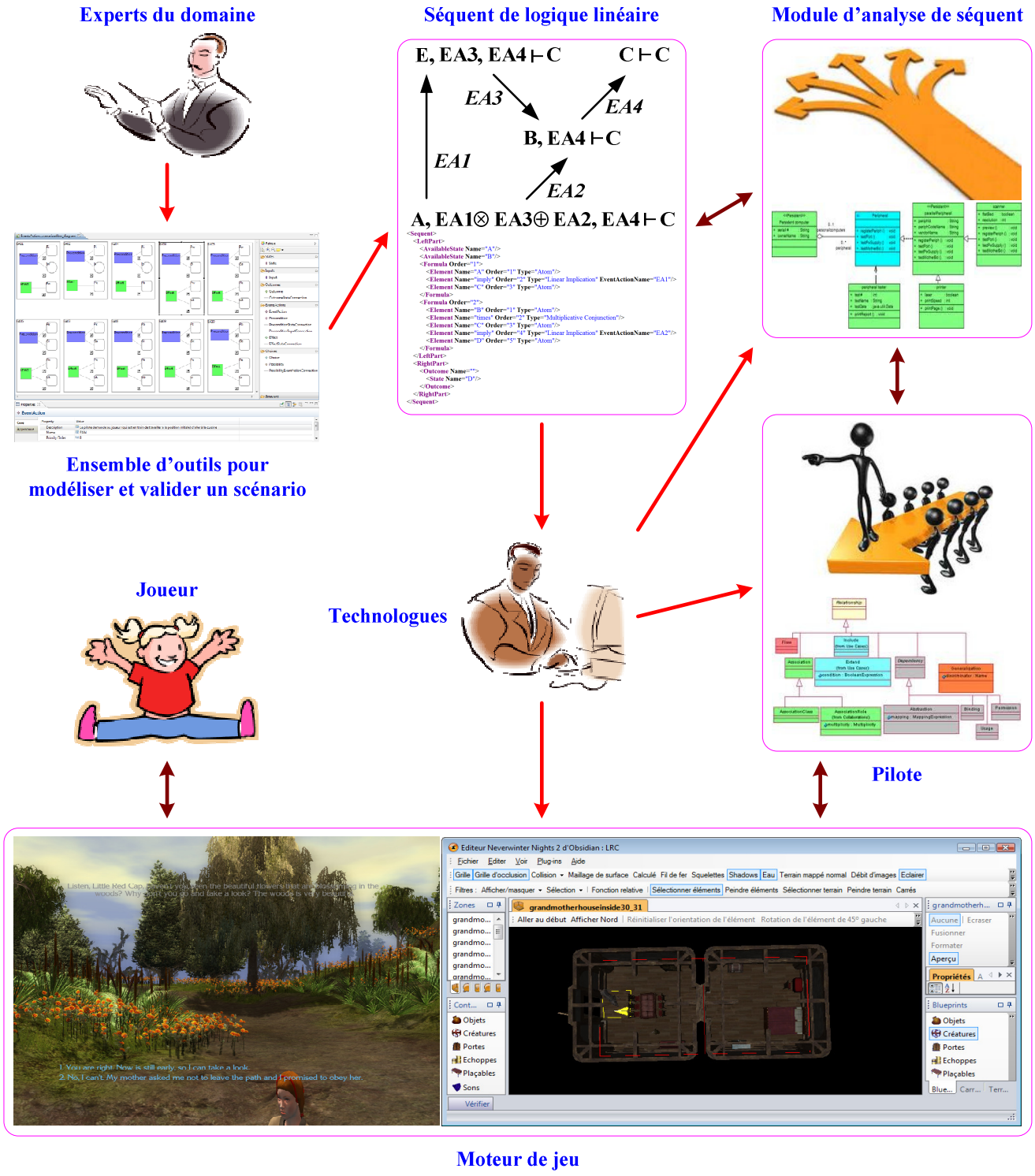


Figure V.20. Schéma général du processus de production d'un jeu vidéo basé sur la narration interactive, qui est dirigé par le mécanisme de déduction automatique d'un séquent de logique linéaire.

- L'objectif du jeu (la terminaison souhaitée des auteurs) : le loup est mort, la grand-mère et le PCr sont vivants.
- États du jeu
  - Ga : La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère

- Gm : Le PCr rencontre le loup dans le bois
- Gge : La grand-mère est mangée par le loup
- Gle : Le PCr est mangé par le loup
- Gr : Le jeu atteint l'objectif (le loup est mort, la grand-mère et le PCr sont vivants)

- Structure de discours : La structure de discours du jeu comprend quatre étapes, dans chaque étape, il n'y a qu'un cas :
  - **Etape 1** – La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère, cette étape ne comprend que l'état Ga ;
  - **Etape 2** – Le PCr rencontre le loup dans le bois, cette étape ne comprend que l'état Gm ;
  - **Etape 3** – La grand-mère et le PCr sont mangés par le loup, cette étape comprend les deux états Gge et Gle ;
  - **Etape 4** – Le loup est mort, la grand-mère et le PCr sont vivants, cette étape ne comprend que l'état Gr.

  Par conséquent, tous les discours possibles dans le scénario créé doivent passer toutes les étapes énoncées plus haut afin de répondre aux objectifs des auteurs.

L'éditeur de scénario aide les auteurs à la réalisation du modèle et à exprimer la structure de discours du jeu (voir Figure V.21). Le modèle d'histoire exprimé en XML est produit au moyen du module de prétraitement, et la qualité du scénario modélisé est analysée au moyen du module d'analyse. Une fois que le scénario obtenu est valide, le module de génération produit le séquent de logique linéaire, dont la représentation XML est conforme au métamodèle du calcul des séquents et qui représente le scénario valide au moment initial.

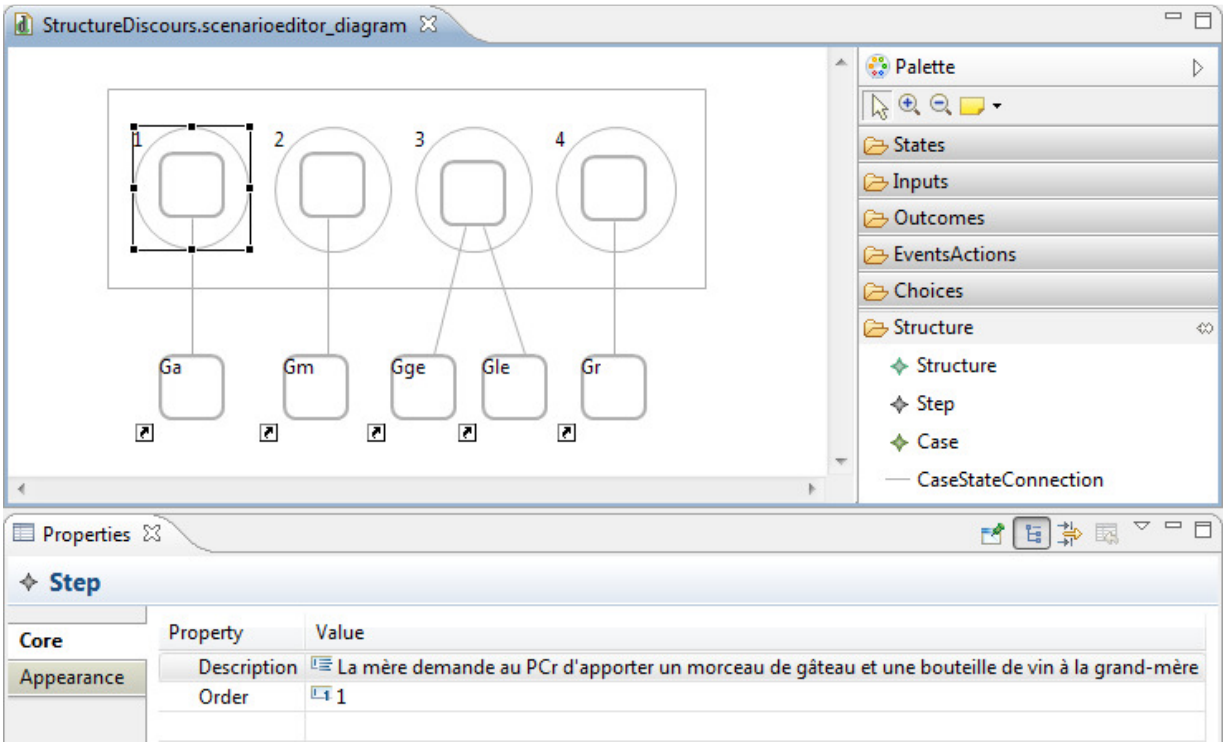


Figure V.21. Exemple de la structure de discours du jeu « Le Petit Chaperon rouge » construite en utilisant l'éditeur de scénario.

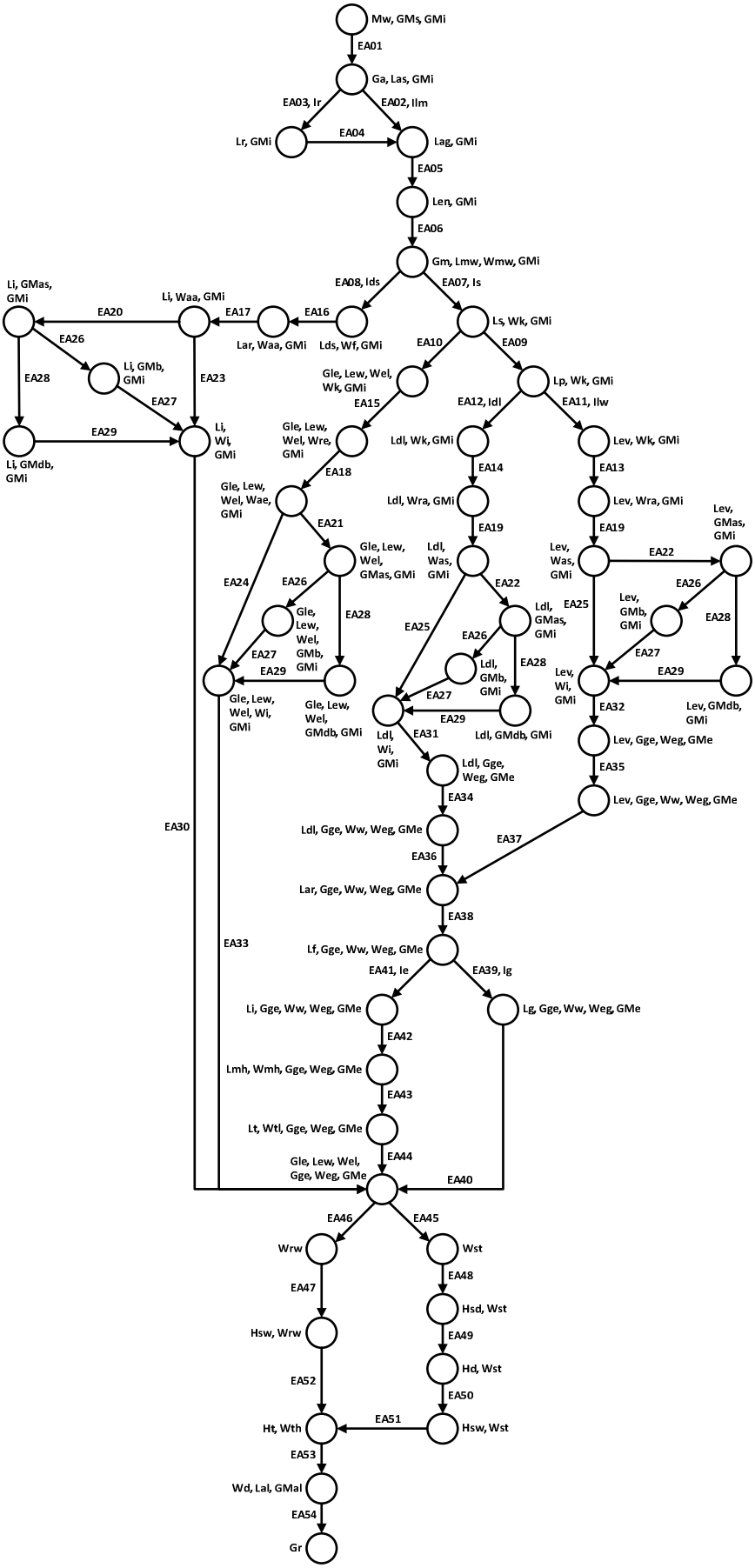


Figure V.22. Le graphe des preuves du séquent modélisant le jeu « Le Petit Chaperon rouge ».

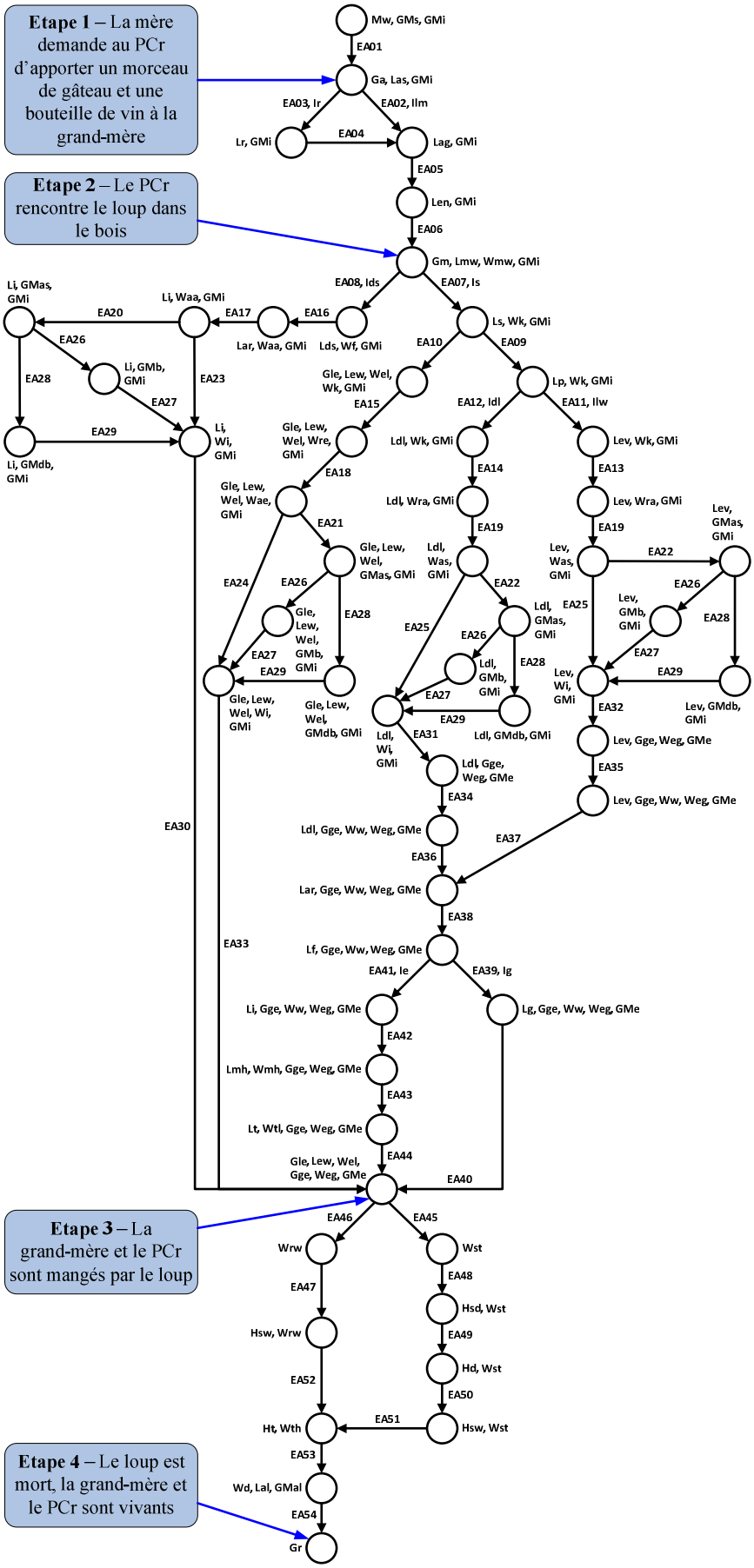


Figure V.23. La position des étapes dans la structure de discours du jeu « Le Petit Chaperon rouge ».

La Figure V.22 donne le graphe des preuves du séquent de logique linéaire créé exprimant tous les discours possibles dans le scénario courant (où un discours correspond à un chemin permettant d'atteindre le dernier nœud *Gr* à partir du premier nœud *Mw*, *GMs*, *GMi*), tandis que la Figure V.23 montre la position des étapes dans la structure de discours du jeu (comme le graphe des preuves, qui est automatiquement produit grâce au module d'analyse, est trop grand par rapport au cadre d'une feuille A4, nous l'avons retracé à la main dans les Figures V.22 et V.23). Ainsi, le scénario courant du jeu « Le Petit Chaperon rouge » comprend 72 discours (néanmoins, on peut augmenter facilement et à volonté le nombre de discours et le niveau d'interactivité du jeu à travers le processus de modélisation), dans lesquels, il y a 4 discours les plus longs avec 28 événements/actions et 1 discours le plus court avec 14 événements/actions.

À partir du scénario/séquent valide créé ci-dessus, les technologues (game designers et informaticiens) mettent en œuvre le prototype de système de pilotage que nous avons proposé au chapitre précédent, ce qui permet de dérouler le jeu selon le scénario valide produit. La section suivante décrit ce processus.

V.3.2. La mise en œuvre du prototype de système de pilotage de narration interactive et la production du jeu

Cette section décrit la mise en œuvre du prototype de système de pilotage de narration interactive que nous avons abordé auparavant, et la production du jeu vidéo « Le Petit Chaperon rouge » réel (ce travail, en réalité, est fait par les technologues). Le système de pilotage créé dirige le déroulement du jeu selon le scénario valide modélisé dans le séquent de logique linéaire. Par conséquent, son évolution garantit les objectifs des auteurs et les actions du joueur autorisent des variations dans le déroulement de l'histoire.

À titre de prototype, nous avons réalisé le module d'analyse de séquent et le pilote en utilisant le langage en Java selon leur spécification en Section IV.6.1 – chapitre IV :

- Module d'analyse de séquent : Ce composant doit exécuter deux tâches : (1) analyser le séquent de logique linéaire au moment courant (qui modélise la situation du jeu en ce moment-là) en vue d'avoir toutes les possibilités d'évolution à l'étape suivante du jeu, et puis transférer ces possibilités au pilote ; (2) mettre à jour le séquent de linéaire logique de façon correspondante à la possibilité que le pilote vient d'exécuter. Ainsi, son entrée au moment initial correspond au séquent de logique linéaire produit ci-dessus, ensuite, il analyse et met à jour ce séquent pendant le déroulement du jeu.
- Pilote : Ce composant a pour but de gérer le déroulement du jeu selon le scénario modélisé par le séquent de logique linéaire. Pour cela, à chaque étape, il recueille les possibilités d'évolution transférées par le module d'analyse, exécute une des possibilités qui est sélectionnée soit par lui-même soit par le joueur, demande au moteur de jeu de montrer les interfaces (scènes) convenables ainsi que reçoit les choix (les actions) du joueur à partir du moteur de jeu, et enfin, requiert le module d'analyse de mettre à jour le séquent de logique linéaire de façon correspondante.

Un prototype de jeu 3D a été réalisé en utilisant l'éditeur de « NeverWinter Nights 2 » (NWN2) [110] comme moteur de jeu (voir Figure V.24). *NeverWinter Nights eXtender 4* [111] permet de connecter une base de données SQLite (ou MySQL) à NWN2 et de

lire/enregistre des informations persistantes dans cette base de données, par conséquent, le pilote et NWN2 peuvent échanger des messages (voir Figure V.25). Pendant le déroulement du fonctionnement, le pilote enregistre ses demandes (pour NWN2) dans la base de données, et lit les choix (les actions) du joueur. NWN2 lit les demandes du pilote dans la base de données (puis montre les interfaces/scènes correspondantes), et enregistre les choix (les actions) du joueur dans la base de données. Ainsi, le pilote peut gérer l'évolution du jeu « Le Petit Chaperon rouge » selon le scénario exprimé dans la Figure V.22, tandis que le moteur de jeu fournit toujours un environnement virtuel cohérent qui est adéquat pour chacune des situations du jeu. La Figure V.26 donne la trace d'exécution du pilote pendant le déroulement du jeu (qui est évidemment implicite pour le joueur), les Figures V.27 → V.32 montrent quelques scènes.

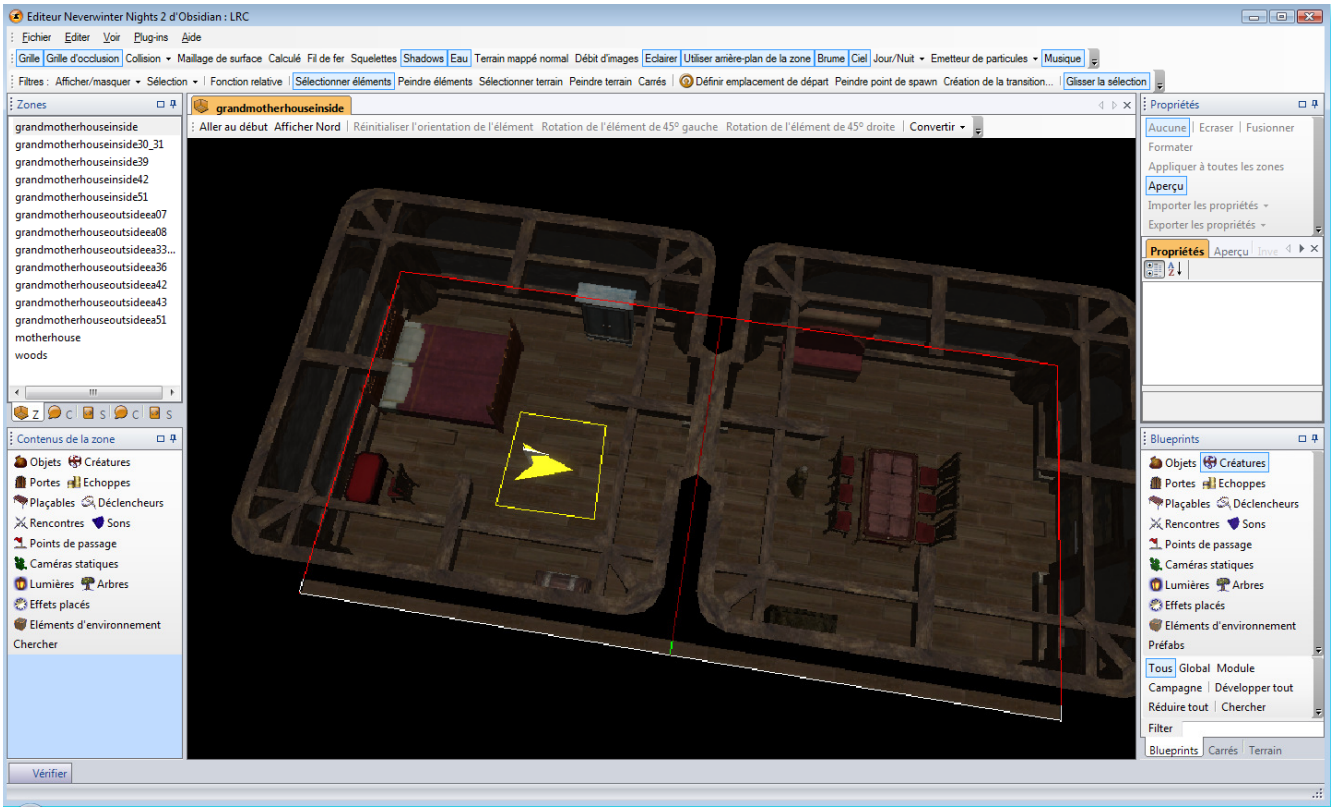


Figure V.24. Éditeur « Neverwinter Nights 2 » qui a un rôle comme le moteur de jeu dans le système de pilotage.

Cet exemple illustre, en plus de l'efficacité de la logique linéaire pour la modélisation et la validation d'un scénario d'une histoire, l'utilisation des séquents comme un modèle opérationnel. L'exemple employé montre aussi que le prototype, que nous avons proposé dans le chapitre précédent, répond bien aux exigences d'un système de pilotage pour des jeux vidéo fondés sur la narration interactive. Par conséquent, les jeux produits ne tombent jamais dans des situations incohérentes, et en même temps, leur déroulement satisfait les intentions des auteurs et répondent aux actions du joueur.

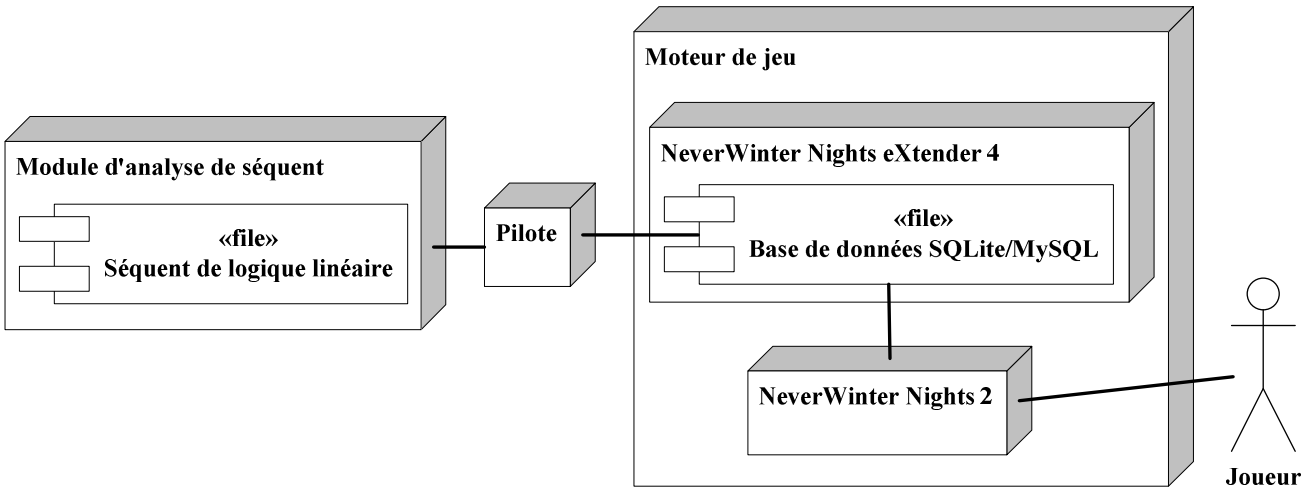


Figure V.25. Le diagramme de déploiement du système de pilotage pour le jeu vidéo « Le Petit Chaperon rouge ».

```
C:\Windows\system32\cmd.exe
C:\Users\kdang01>java -jar "C:\Users\kdang01\Documents\NetBeansProjects\ISContro
ller\dist\ISController.jar"
Execute the event/action: EA01
You have chosen the input: Ir
Execute the event/action: EA03
Execute the event/action: EA04
Execute the event/action: EA05
Execute the event/action: EA06
You have chosen the input: Is
Execute the event/action: EA07
Execute the event/action: EA09
You have chosen the input: Idl
Execute the event/action: EA12
Execute the event/action: EA14
Execute the event/action: EA19
Execute the event/action: EA25
Execute the event/action: EA31
Execute the event/action: EA34
Execute the event/action: EA36
Execute the event/action: EA38
You have chosen the input: Ie
Execute the event/action: EA41
Execute the event/action: EA42
Execute the event/action: EA43
Execute the event/action: EA44
Execute the event/action: EA46
Execute the event/action: EA47
Execute the event/action: EA52
Execute the event/action: EA53
Execute the event/action: EA54
Reach the goal successfully
List of events/actions that has been executed: EA01 EA03 EA04 EA05 EA06 EA07 EA0
9 EA12 EA14 EA19 EA25 EA31 EA34 EA36 EA38 EA41 EA42 EA43 EA44 EA46 EA47 EA52 EA5
3 EA54
```

Figure V.26. Sortie du pilote lors du contrôle du déroulement d'une partie.

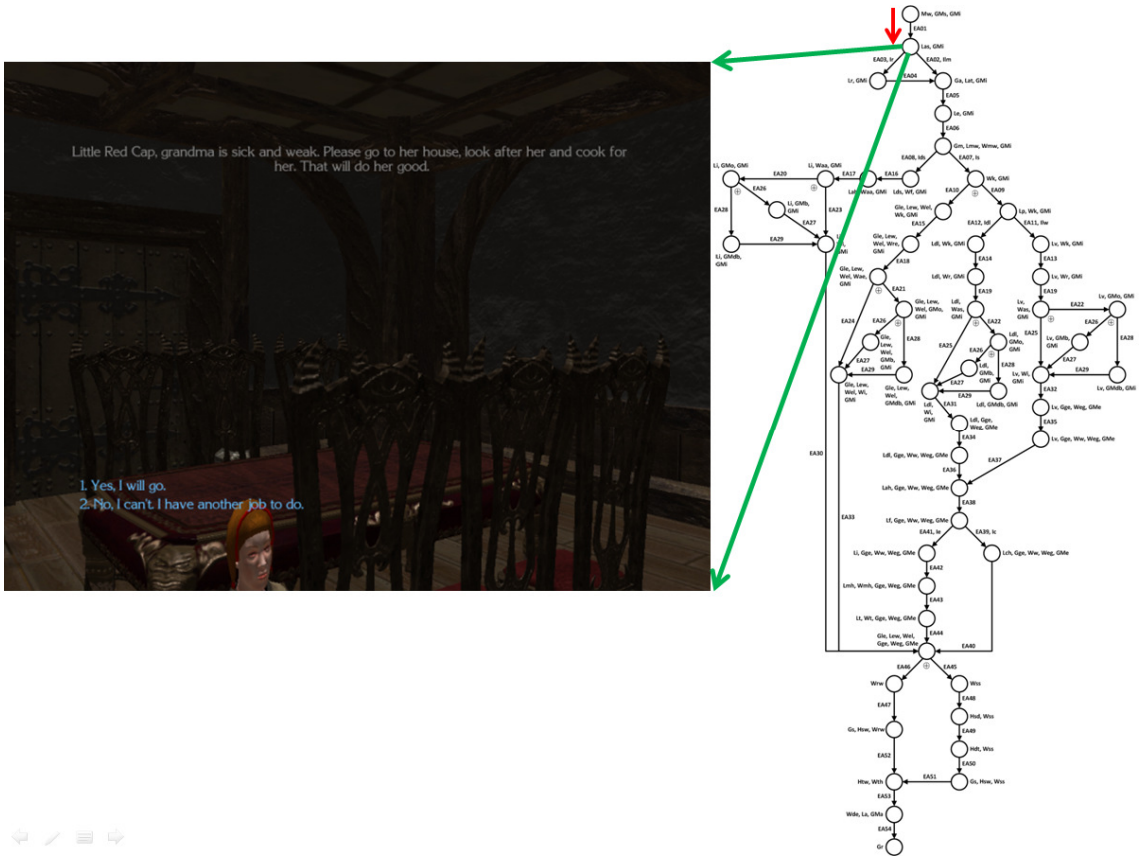


Figure V.27. La mère demande au PCr d'aller à la maison de la grand-mère qui est malade.

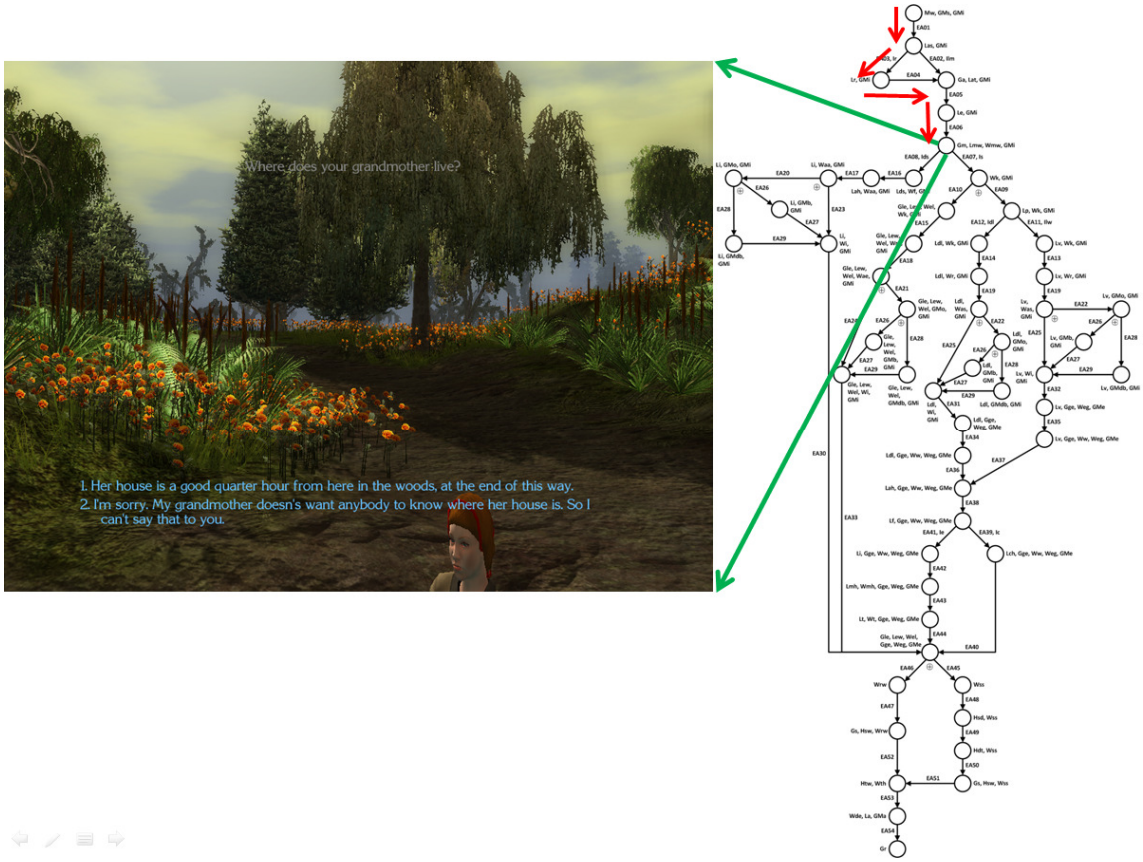


Figure V.28. Le PCr dit (ou ne dit pas) au loup où est la maison de la grand-mère, en faisant un choix par la souris.

Figure V.29. Le PCr entre dans la maison de la grand-mère (elle n'a pas dit au loup sa position).

Figure V.30. Le loup mange la grand-mère et le PCr dans la maison de la grand-mère.

Figure V.31. Le chasseur tue le loup dans la maison de la grand-mère.

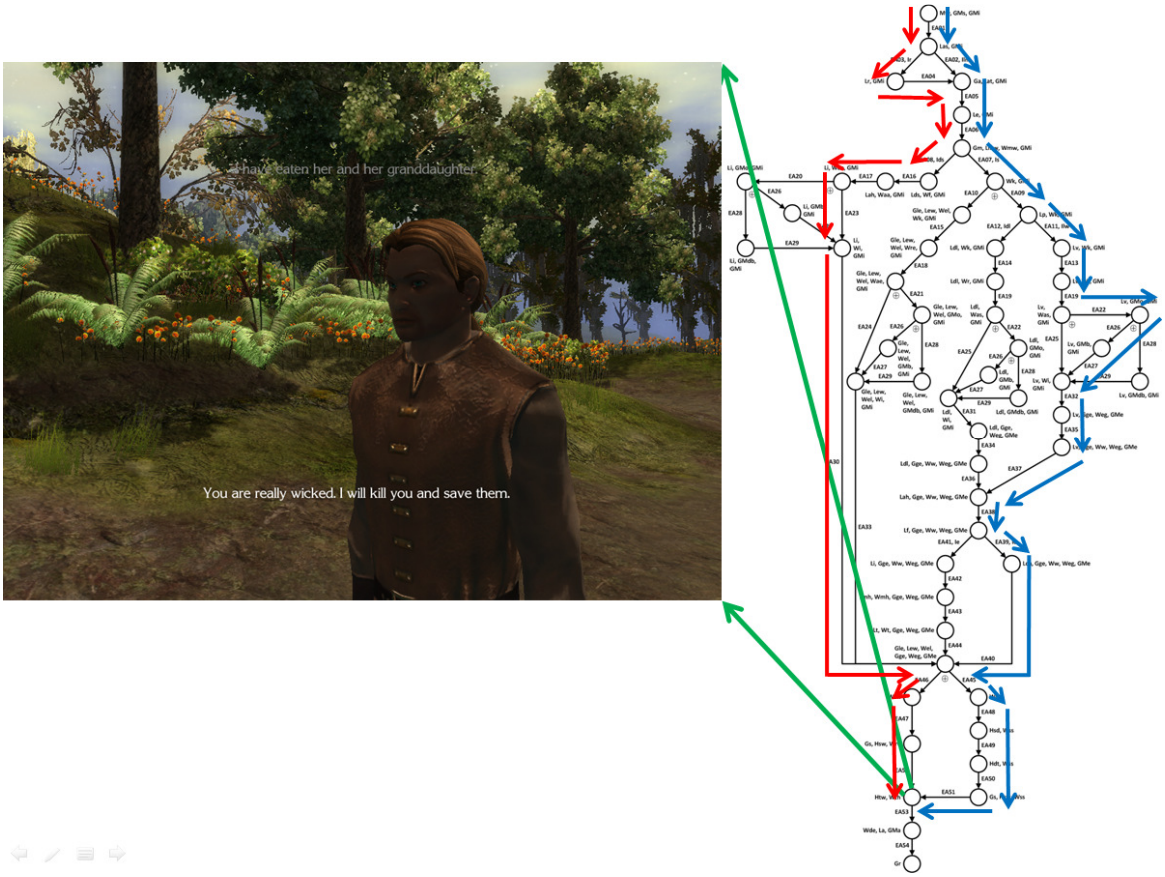


Figure V.32. Le chasseur tue le loup en dehors de la maison de la grand-mère.

V.4. Conclusion

Nous avons présenté, dans ce chapitre, deux exemples ayant pour objectifs d'illustrer et également de valider les résultats que nous avons obtenus dans le cadre de cette thèse. Le premier – Jeu d'apprentissage des dangers de l'électricité – a concrètement expliqué comment les utilisateurs (qui n'ont aucune connaissance en logique linéaire) peuvent appliquer la solution que nous proposons. De plus, il permet de décrire le système auteur que nous avons développé, afin de produire un scénario de bonne qualité d'un jeu, qui est exprimé par un séquent de logique linéaire basé sur le métamodèle du calcul des séquents.

Pour le second exemple, nous avons décrit le processus de production complet d'un jeu vidéo réel basé sur l'histoire « Le Petit Chaperon rouge », mettant en œuvre le prototype de système de pilotage que nous avons donné. Ceci nous permet de dérouler l'exécution d'un jeu selon le scénario valide produit par le système auteur. Cet exemple montre que, en plus de l'efficacité de la logique linéaire dans la modélisation et la validation d'un scénario d'une histoire, l'utilisation de la déduction automatique liée à la logique linéaire aide le système de pilotage à gérer le déroulement d'un jeu selon le scénario (séquent) prédéfini. Il a aussi montré que le prototype décrit répond bien aux exigences d'un système de pilotage pour des jeux vidéo fondés sur la narration interactive.

Cependant, comme les exemples employés sont assez simples, leur objectif est seulement d'illustrer le processus. Ainsi, ces exemples ne nous ont pas permis de valider tous les aspects abordés dans les chapitres précédents, tels que : les propriétés de narration liées à l'effet dramatique créé par un discours, les deux propriétés de narration liées au déroulement des discours *Satisfaction d'événements clés* et *Séquencement*, la validation d'un scénario avec les événements/actions complexes ou d'un scénario modélisé en PDDL, ce qui nécessiterait de développer des scénarios vraiment copieux en contenu, exigeant beaucoup de temps et donc dépassant le cadre de cette thèse.

Dans le chapitre suivant, nous donnons des conclusions de tous/toutes les travaux/contributions que nous avons réalisé(e)s dans ce mémoire ainsi que des perspectives de nos travaux futurs.

CHAPITRE VI - Conclusions et Perspectives

Dans ce chapitre, nous commençons par un bilan général de tous/toutes les travaux/contributions que nous avons réalisé(e)s dans le cadre de la thèse. Ensuite, nous analysons les limites des résultats que nous avons obtenus jusqu'ici, et en même temps, donnons des perspectives de nos travaux dans l'avenir.

VI.1. Bilan de la thèse

L'objectif de cette thèse est de fournir une solution qui d'une part permet aux auteurs de résoudre le dilemme scénario/interactivité et d'autre part donne les moyens d'une analyse du dégré d'interactivité et plus généralement de la qualité du jeu. Pour cela, nous avons proposé une méthode permettant d'étudier la qualité d'un scénario. Cette analyse vise à assurer que le scénario fournit suffisamment d'options pertinentes aux personnages joueur/non-joueur de sorte que le joueur puisse déterminer le déroulement du jeu et sente toujours que le discours créé est intéressant. En s'assurant que tous les discours possibles dans le scénario sont cohérents et répondent aux effets souhaités par les auteurs.

Ensuite nous avons étudié le processus de production d'un système de pilotage de narration interactive, et nous avons déduit un système auteur basé sur un ensemble d'outils implémentant les algorithmes permettant de déterminer la qualité du scénario. Le scénario valide, qui est produit par le système auteur et modélisé en logique linéaire, est ensuite employé comme l'entrée du système de pilotage. Son rôle est d'assister ce dernier dans la gestion du déroulement du jeu en respectant les contraintes exprimées dans le scénario modélisé. Cette capacité résulte du mécanisme de déduction rigoureux et automatique réalisé par le calcul des séquents.

Pour cela, les principaux travaux réalisés sont les suivants :

- proposer un ensemble de propriétés de narration spécifiant la qualité des scénarios de jeu, celles-ci sont classées en deux catégories : les propriétés liées au déroulement des discours et les propriétés liées à l'effet dramatique créé par chacun des discours ;
- améliorer les travaux précédemment menés au L3I concernant la modélisation d'une histoire exprimée en logique linéaire, en ajoutant les nouveautés suivantes : l'utilisation des connecteurs & et ⊕, la notion d'ordre de priorité des événements/actions, la notion de structure de discours, et l'attribution d'émotions aux événements/actions ;
- proposer des algorithmes permettant d'évaluer la qualité d'un scénario modélisé en logique linéaire ;
- proposer un moyen de parcourir tous les discours possibles d'un scénario et d'analyser leur validité par rapport à l'ensemble de propriétés de narration ;
- réaliser un ensemble d'outils constituant un système auteur, qui aide les utilisateurs à exécuter la solution proposée pour la production d'un scénario de bonne qualité, même s'ils n'ont aucune connaissance en logique linéaire ;

- réaliser deux exemples ayant pour objectifs d'illustrer ainsi que de valider les résultats obtenus dans le cadre de la thèse : le premier est un extrait d'un jeu éducatif expliquant comment appliquer notre système auteur en vue de produire un scénario de jeu valide, qui est exprimé par un séquent de logique linéaire dont la représentation est conforme au métamodèle du calcul des séquents ; pour le second exemple, nous avons décrit le processus de production complet d'un jeu vidéo réel basé sur l'histoire « Le Petit Chaperon rouge », mettant en œuvre le prototype de système de pilotage que nous avons proposé, ce qui permet de dérouler le jeu selon le scénario valide produit, donc son évolution satisfait les intentions des auteurs, et en même temps, dépend des actions du joueur.

Par ailleurs, à côté de l'emploi de la logique linéaire comme l'entrée du système de pilotage, nous avons constaté que PDDL était un modèle courant en Intelligence Artificielle. Nous avons également contribué à la réalisation des jeux vidéo utilisant PDDL (un langage d'expression de scénario utilisé dans un contexte de planification de tâches) en validant, grâce à notre système auteur, un scénario de jeu modélisé en PDDL. Notre travail permet à des utilisateurs, après avoir modélisé une histoire en PDDL selon un métamodèle d'histoire que nous avons proposé, de transformer automatiquement, ce modèle de l'histoire en PDDL en un modèle d'histoire en logique linéaire. En employant les outils de notre système auteur, une analyse du scénario peut être menée selon notre démarche et ainsi il est possible d'évaluer sa qualité, et de le valider.

Ainsi, les contributions ci-dessus que nous avons réalisées dans le cadre de la thèse nous ont permis de résoudre l'opposition entre le déroulement d'un jeu respectant les effets désirés par les auteurs et son niveau d'interactivité. Néanmoins, quelques limitations subsistent. Nous les analysons en détail dans la section suivante, où nous donnons aussi des perspectives de nos travaux futurs.

VI.2. Discussions

Les travaux présentés dans le cadre de cette thèse, permettant de produire des scénarios/jeux vidéo dont l'évolution satisfait à la fois les intentions des auteurs et les choix d'action du joueur, ont ouvert une perspective dans la construction des applications interactives offrant un équilibre entre liberté d'action du joueur et respect d'un canevas défini par un scénariste talentueux. Cette contribution concerne plusieurs domaines comme la production des jeux sérieux [37], des jeux utiles [36] ou des Environnements Informatiques pour l'Apprentissage Humain (EIAH) [46],…

Néanmoins, les résultats que nous avons obtenus jusqu'à présent comportent quelques limites. En premier lieu, nos travaux n'exploitent pas toute la puissante d'expression de la logique linéaire. En effet, dans notre approche, nous n'utilisons que 4 connecteurs ($\otimes$, $\multimap$, &, $\oplus$) afin de modéliser un scénario d'une histoire. Même si ces connecteurs permettent de représenter la presque totalité des aspects nécessaires pour la modélisation de narration interactive, il nous faudrait toujours ajouter quelques autres connecteurs en vue d'améliorer la capacité de modélisation de narration interactive de la logique linéaire, par exemple :

- l'exponentielle « ! » : ce connecteur a pour but de modéliser une ressource que l'on peut utiliser autant de fois que l'on veut ;
- la constante « 1 » : ce connecteur a pour but de modéliser un choix entre une ressource et « rien », par exemple, « café & 1 » représente un choix, décidé par le joueur, entre « café » et « rien » ;
- la négation linéaire : ce connecteur a deux aspects : (1) modéliser la négation d'une ressource, par exemple, si on modélise le fait « le joueur possède une ou plusieurs maison(s) » par l'atome A, alors on peut modéliser le fait « le joueur ne possède aucune maison » par la négation linéaire $A^{\perp}$; (2) représenter la dualité entre ce qui est décidé par le système de pilotage et ce qui est décidé par le joueur, par exemple, $(A \oplus B)^{\perp} = A^{\perp} \& B^{\perp}$.

Le système auteur que nous avons développé nécessiterait une évaluation des utilisateurs ainsi qu'une étude d'ergonomie en vue de spécifier un ensemble de modifications sur l'interface graphique, la facilité d'emploi, la représentation des produits sortis, la maturité…, pour améliorer les outils de façon correspondante.

Les résultats obtenus pour la validation des scénarios avec les événements/actions complexes ne permettent de vérifier que deux des propriétés de narration : *Accessibilité* et *Absence de blocage*. La vérification d'autres propriétés nécessite de modifier l'algorithme conçu et implémenté dans l'outil.

Pour les deux propriétés *Correspondance d'émotion du joueur* et *Évolution de la tension dramatique*, même si on peut vérifier leur satisfaction pour un scénario en faisant un sondage sur la transformation émotionnelle réelle des joueurs lors de l'exécution du jeu vidéo utilisant le scénario courant, la mise en œuvre d'une assistance automatique pour évaluer ces propriétés serait un apport intéressant ([2, 30, 77] ont donné quelques suggestions concernant ce problème).

Par ailleurs, notre stratégie d'utilisation de logique linéaire actuelle n'est pas pertinente si la taille d'histoire est trop grande (ou si le contenu d'histoire est particulièrement riche). Les raisons en sont les suivantes :

- La représentation courante du séquent modélisant une telle histoire est trop grande pour que le système de pilotage puisse bien traiter (problème de stockage, temps d'exécution), un autre métamodèle exprimant le séquent de façon plus adéquate et/ou un autre algorithme de fonctionnement appliqué pour le système de pilotage doivent donc être exécutés dans ce cas.
- Dans notre solution actuelle, même si à chaque événement/action de l'histoire est attribué un ordre de priorité (cela permet d'éliminer l'explosion combinatoire lors de l'examen du graphe des preuves d'un scénario puisque la complexité de l'algorithme d'ordonnance des événements/actions est exponentielle), la validation de scénario d'une histoire dont la taille est trop grande, en parcourant et analysant tous ses discours possibles, est soumise à des limitations :
    - due au temps d'exécution du processus d'examen de scénario qui devient trop long ;

- due aux résultats d'analyse obtenus après le processus d'examen de scénario dont la taille est trop importante pour que les utilisateurs puissent les explorer « à la main ».

Nous pouvons émettre des propositions en vue de résoudre les problèmes soulevés ci-dessus :

- permettre aux utilisateurs de sélectionner une ou quelques propriété(s) de narration dont ils ont besoin au lieu de vérifier toutes les propriétés à chaque exécution de l'analyse de scénario ;
- rechercher les symétries dans le graphe des preuves et afin de n'examiner qu'une partie et ainsi réduire l'espace de traitement ;
- diviser le graphe des preuves en plusieurs régions dont chacune est composée de nœuds et d'arêtes de sorte qu'elle ne comprenne qu'une entrée et une sortie (voir Figure VI.1). Par conséquent, il suffit que nous vérifions si chaque région du graphe des preuves contient des blocages, ainsi que si entre les régions il existe toujours une liaison à travers leurs entrées et sorties. Grâce au résultat de vérification, il est possible pour nous de connaître s'il existe des blocages dans le scénario courant.

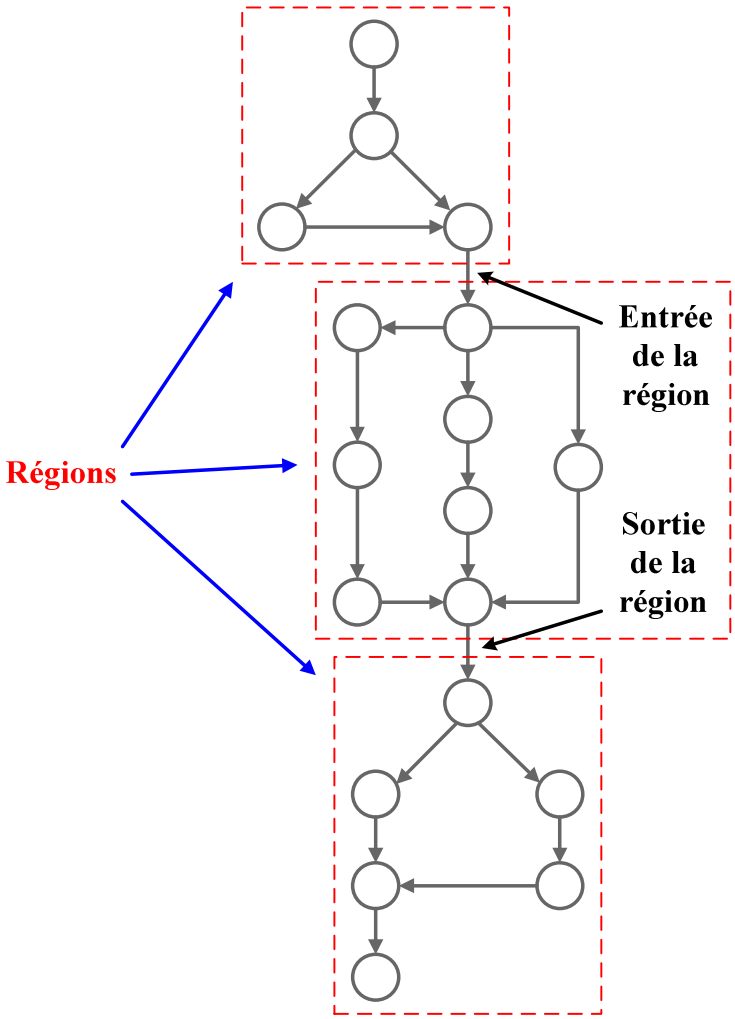


Figure VI.1. Structuration d'un scénario en régions.

À côté des problèmes cités précédemment, il convient aussi d'améliorer « l'intelligence » du système de pilotage de narration interactive que nous avons développé. Le choix, correspondant à l'opérateur additif (opérateur ⊕) effectué par le système, peut être optimisé pour répondre à des besoins concernant le déroulement, le joueur ou le contexte du jeu. Autrement dit, le système de pilotage doit jouer le rôle d'un système expert, et peut adapter le déroulement du jeu à chaque joueur/contexte. Certaines solutions de cette amélioration peuvent être les suivantes : construire un modèle utilisateur (voir [75] comme un exemple), appliquer des heuristiques afin d'analyser/évaluer les objectifs à court/long terme de l'histoire,… et ainsi, le système de pilotage peut fournir/suggérer le discours le plus pertinent/court au joueur correspondant à la situation actuelle du jeu.

Notre approche, jusqu'à présent, n'est applicable qu'à la production de jeux vidéo pour joueur unique. Pour élargir le principe à un contexte de jeu multijoueurs, le modèle de jeu en logique linéaire doit être repensé.

Enfin, la solution que nous avons proposée pour valider un scénario modélisé en PDDL ne couvre pas toute sa capacité de modélisation de narration interactive. En effet, comme ce sont simplement les exécutions préliminaires, elles ne permettent pas encore de vérifier certains composants importants en PDDL tels que les préférences ; les contraintes *within*, *always-within*, *hold-during*, *hold-after* ; les actions duratives ; *etc*. Ainsi, il nous faudrait explorer toutes les fonctionnalités de PDDL, et puis ajouter les principes correspondants à l'approche de logique linéaire courante, en vue de pouvoir satisfaire les exigences de validation d'un scénario plus complexe modélisé en PDDL.

ANNEXE A - Brève Introduction à la Logique Linéaire

Pour que les lecteurs puissent mieux comprendre la solution proposée dans le cadre de cette thèse, nous donnons ici une brève introduction à la logique linéaire, qui présente ses points forts, ses principales caractéristiques, et son adéquation, contrairement à la logique classique, à la modélisation des systèmes de consommation/production, et ainsi à la modélisation des narrations interactives.

A.1. Logique classique – les points non-concordants pour la consommation/ production

La logique classique – une formalisation du langage et du raisonnement – est largement étudiée et utilisée depuis longtemps dans beaucoup de domaines. Néanmoins, elle n'est pas adaptée à modéliser la consommation/production. Concrètement, en raison du principe de pérennité de la vérité, en logique classique, le raisonnement ci-dessous

pour deux euros j'ai un paquet de café [A $\Rightarrow$ B]

où *j'ai deux euros* [A]

j'ai un paquet de café [B]

n'est pas réaliste car les valeurs de vérité des propositions classiques ne sont pas révocables. En effet, la déduction de la conclusion B ne changerait rien à la vérité de la prémisse A. Autrement dit, obtenir un paquet de café ne coûte rien.

Par ailleurs, la logique classique possède les propriétés suivantes :

Contraction : $A \Rightarrow A \wedge A$, et
Affaiblissement : si $A \Rightarrow B$, *alors* $A \wedge C \Rightarrow B$

L'interprétation de ces règles pose problème dans le cas où les propositions représentent des ressources (la règle *Contraction* veut dire que la proportion n'est pas importante, tandis que la règle *Affaiblissement* nous permet d'ajouter un élément quelconque au membre de gauche qui n'est pas présent au membre de droite).

Reprenons l'exemple de paquet de café, et supposons que l'acheteur ait deux pièces de deux euros. La conjonction de ces propositions *j'ai deux euros* $\wedge$ *j'ai deux euros*, si elles traduisent des ressources, doit conduit à *j'ai quatre euros*, mais pas à *j'ai deux euros*.

De façon similaire, le raisonnement

pour deux euros j'ai un paquet de café [A $\Rightarrow$ B]

ne peut pas impliquer le raisonnement

pour deux euros et cinquante centimes, j'ai un paquet de café [A $\wedge$ C $\Rightarrow$ B]

dans une formalisation de ressources ayant un coût. Afin de surmonter tous les inconvénients ci-dessus, Girard a proposé la logique linéaire. Nous présentons cette notion dans la section suivante.

A.2. Logique linéaire – une amélioration de la logique classique pour la consommation/production

La logique linéaire est une logique substructuralle, proposée par Girard [41, 42] comme un modèle formel exécutable et une amélioration de la logique classique pour la consommation/production. En logique linéaire, on considère les atomes et les formules comme des ressources qui sont consommées et/ou produites, par conséquent, la présence conjointe de deux exemplaires d'un atome (ou d'une formule) est différente de celle d'un exemplaire. Contrairement à la logique classique, la logique linéaire n'est pas appliquée en vue de déterminer si une assertion est vraie ou pas, mais elle est employée pour représenter la validité de la manière dont les ressources (atomes et/ou formules) sont utilisées lors de la preuve d'une assertion. En d'autres termes, on s'intéresse à écrire la preuve et à analyser les choix de ressources faits au cours de ce processus. Par ailleurs, la logique linéaire est bien adaptée à la modélisation du raisonnement naturel à travers le mécanisme de calcul des séquents introduit par Gentzen [52]. La théorie linguistique emploie un sous-ensemble de la logique linéaire (multiplicative intuitionniste et non commutative) qui correspond au calcul Lambek [1]. On voit ainsi que la logique linéaire fournit un cadre sémantique à la modélisation de la causalité, du raisonnement automatique ainsi que des mécanismes d'allocation de ressources pour une histoire (pour des lecteurs intéressés, [33, 34] présentent comment construire des plans valides utilisant la logique linéaire, [79] est une bonne référence afin d'avoir plus d'information sur la façon de modélisation des problèmes d'allocation de ressources dans la logique linéaire, tandis que [55] donne une interprétation de calcul complète sur plusieurs fragments de la logique linéaire et établit exactement le niveau de complexité de ces fragments).

A.2.1. Connecteurs de la logique linéaire utilisés dans le cadre de la thèse

Pour modéliser une histoire dans notre approche, nous utilisons un fragment de la logique linéaire, comportant les connecteurs suivants :

- $\multimap$: *implication linéaire (imply)*, exprime la possibilité de déduction. Les ressources prises pour prémisses, sont consommées pour produire les ressources présentes dans sa conclusion. La production des conclusions entraine la consommation des prémisses. Exemple : « 1euro $\multimap$ 1 kg fraises » signifie que l'on doit donner 1euro en vue d'acheter 1kg fraises.
- $\otimes$: *conjonction multiplicative (times)*, exprime la présence conjointe d'éléments. Exemple : « 1euro $\multimap$ 1kg fraises $\otimes$ 1kg tomates » signifie que l'on peut donner 1euro afin d'acheter 1kg fraises et 1kg tomates. Dans l'exemple de paquet de café ci-dessus, la conjonction « *j'ai deux euros* $\otimes$ *j'ai deux euros* » exprime le fait qu'une pièce de deux euros est deux fois disponible.
- & : *conjonction additive (with)*, exprime un choix externe au système (par exemple venant d'un joueur) s'il est dans la partie gauche du séquent (ce connecteur n'a pas de

sens (et ainsi il n'est pas utilisé dans notre approche) s'il est dans la partie droite du séquent). Exemple : « 1euro ⊸ thé & café » signifie que l'on (le joueur) peut choisir soit thé soit café quand on donne 1euro à une machine automatique.

- ⊕ : *disjonction additive (plus)*, exprime un choix interne au système (par exemple venant d'un système de pilotage) s'il est dans la partie gauche du séquent. Exemple : « 1euro ⊸ thé ⊕ 1euro » signifie que c'est la machine automatique qui décide soit de nous donner du thé soit de nous retourner 1euro selon la disponibilité du thé dans la machine, si cette formule se trouve dans la partie gauche du séquent. Si le connecteur ⊕ est dans la partie droite du séquent, son utilisation se limite à connecter des conséquences/conclusions distinctes. Par exemple, si la partie droite d'un séquent est « thé ⊕ café », cela signifie que si le séquent est prouvable, alors à partir de la partie gauche du séquent, on peut recevoir soit du thé soit du café.

Maintenant en vue de mieux comprendre l'usage des connecteurs ci-dessus, voyons l'exemple suivant [43] : un restaurant propose un menu à 20 euros, où figurent une entrée, un plat principal et un dessert comme suit :

MENU A 20 EUROS

****

Soit Tomates soit Soupe (selon disponibilité)

Steak + Frites

Soit Fromage soit Glace (au choix)

Nous pouvons représenter ce menu par la formule d'implication linéaire ci-dessous :

20 euros ⊸ (Tomates ⊕ Soupe) ⊗ Steak ⊗ Frites ⊗ (Fromage & Glace)

où, le connecteur ⊕ sert à décrire la situation que le client rencontre avec l'entrée : il reçoit une entrée, mais le choix du plat (entre Tomates et Soupe) est fait par le restaurant. Le connecteur & indique que le client a soit un fromage soit une glace selon son choix. Le connecteur ⊸ exprime que le client dépense ce qui figure au membre de gauche pour obtenir ce qui figure au membre de droite. Enfin le connecteur ⊗ montre que le client a simultanément les objets qui figurent à sa droite et à sa gauche.

A.2.2. Séquent de logique linéaire et preuve de séquent

Un séquent est une expression $\Gamma \vdash \Delta$, où Γ et Δ sont des agrégats d'atomes et/ou de formules ; ⊢ (*turnstile*) est utilisé afin de séparer sa partie gauche (antécédents/ressources disponibles) et sa partie droite (conséquences/conclusions). Par exemple, « A ⊗ (A ⊸ B) ⊢ B » (ou « A, A ⊸ B ⊢ B ») signifie la possibilité de produire une copie de « B » en consommant les

ressources disponibles « A » et « A ⊸ B » (on peut remplacer le connecteur ⊗ entre deux atomes, entre deux formules, ou entre un atome et une formule dans la partie gauche d'un séquent par la virgule « , » pour être plus bref). À partir du contenu de la partie gauche d'un séquent, on peut aboutir à plusieurs conclusions/conséquences valides. Ce processus constitue une preuve (comment atteindre une conclusion/conséquence) qui n'est pas unique. Ceci signifie qu'il existe plusieurs façons de parvenir à une même conclusion/conséquence.

En vue de prouver un séquent (c'est-à-dire de montrer qu'il est correct), nous employons le calcul des séquents [52]. La technique consiste à réécrire le séquent, en faisant une substitution respectant les règles d'inférence de la logique linéaire jusqu'à l'obtention des séquents identité (un séquent est appelé *séquent identité* s'il est dans la forme « A ⊢ A » où A est un atome quelconque). Pour un même séquent, il peut y avoir des preuves réussies (se terminant par les séquents identité) ou bien des échecs (composée d'agrégats ne permettant plus de réécritures). La stratégie de preuve permet, au moyen de la logique linéaire, de raisonner sur la logique de discours et sur les mécanismes d'allocation de ressources lors du déroulement d'une histoire.

ANNEXE B - Principes de Base de l'Approche IDM

L'Ingénierie Dirigée par les Modèles (IDM), en anglais Model-Driven Engineering (MDE), est une démarche utilisée en vue de développer des logiciels, qui se concentre sur la création, l'exploitation et la transformation de modèles. Cette démarche permet de minimiser simultanément les coûts et le temps de production d'applications, et en facilite la maintenance et l'évolution. Nous présentons ci-dessous les notions fondamentales de cette approche (pour plus d'information, voir le document [11]) :

- Un *modèle* représente une abstraction ou une simplification d'un système ou d'un (de plusieurs) aspect(s) d'un système. La repésentation du modèle utilise les définitions d'un autre modèle nommé « métamodèle ». Par exemple, un modèle relationnel « Book », qui définit le concept d'ensemble de livres dans une bibliothèque, est donné dans la Figure B.1.

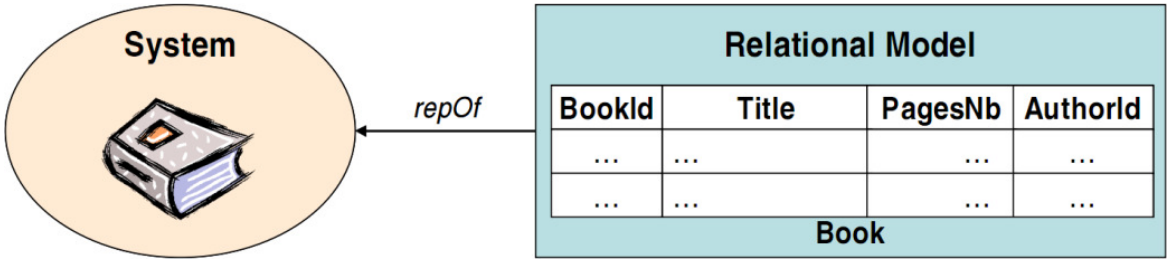


Figure B.1. La relation « représentation » entre un système et un modèle [11].

- Un *métamodèle* définit le langage permettant d'exprimer les modèles. Il décrit les différents types d'éléments du modèle, et les types de relations entre ces éléments. Il agit comme un filtre en vue d'extraire les éléments pertinents d'un système qui sont les constituants d'un modèle conforme à ce métamodèle. Par exemple, le modèle relationnel « Book » est conforme au métamodèle relationnel donné dans la Figure B.2.
- Comme les modèles, les métamodèles sont aussi composés d'éléments. Les éléments des métamodèles fournissent un schéma de typage des éléments des modèles. Ce typage est exprimé par la relation entre un élément du modèle et un élément de son métamodèle. Ainsi, un modèle est conforme à un métamodèle si et seulement si chaque élément du modèle correspond à son métaélément défini dans le métamodèle. Par exemple, la Figure B.3 montre les relations entre les éléments du modèle relationnel « Book » et les éléments de son métamodèle.
- De façon similaire, les métamodèles sont définis au moyen du langage de *métamétamodèle*. Autrement dit, un métamodèle doit être conforme à un certain métamétamodèle. Par exemple, on peut utiliser le métamétamodèle MOF (MetaObject Facility [113]) de l'OMG (Object Management Group), ou le métamétamodèle ECORE de l'EMF (Eclipse Modeling Framework [105]) pour la définition des métamodèles. Les métamétamodèles sont également composés

d'éléments. Ainsi, un métamodèle est conforme à un métamétamodèle si et seulement si chaque élément du métamodèle correspond à son metaélément défini dans le métamétamodèle. La Figure B.4 présente les relations entre les éléments du métamodèle relationnel « Book » et les éléments de son métamétamodèle MOF, tandis que la Figure B.5 exprime le schéma de relation résumé entre un système dans le monde réel et un modèle, un métamodèle, un métamétamodèle dans le monde de modélisation où le métamétamodèle doit aussi être conforme de façon récursive à lui-même.

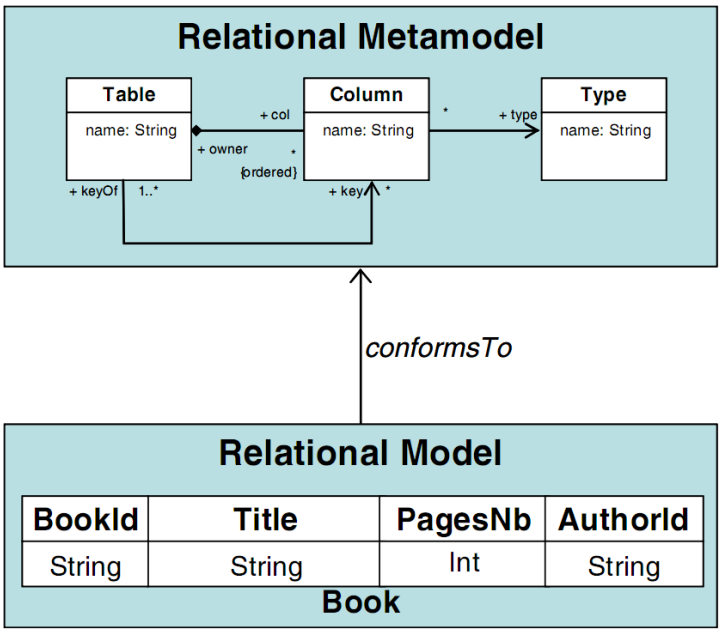


Figure B.2. La relation « conformité » entre un modèle et son métamodèle [11].

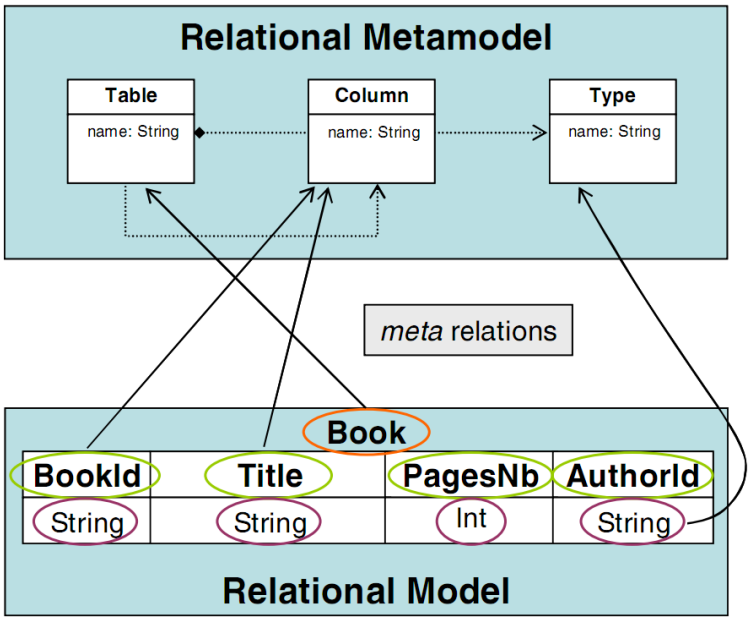


Figure B.3. Relations entre les éléments d'un modèle et les éléments de son métamodèle [11].

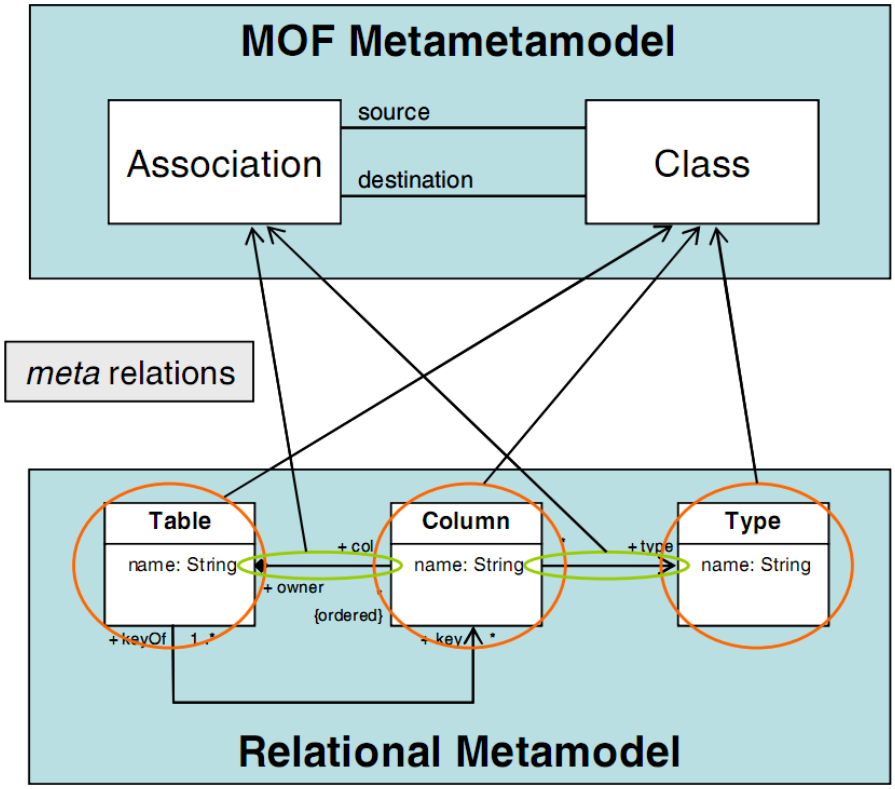


Figure B.4. Relations entre les éléments d'un métamodèle
et les éléments de son métamétamodèle MOF [11].

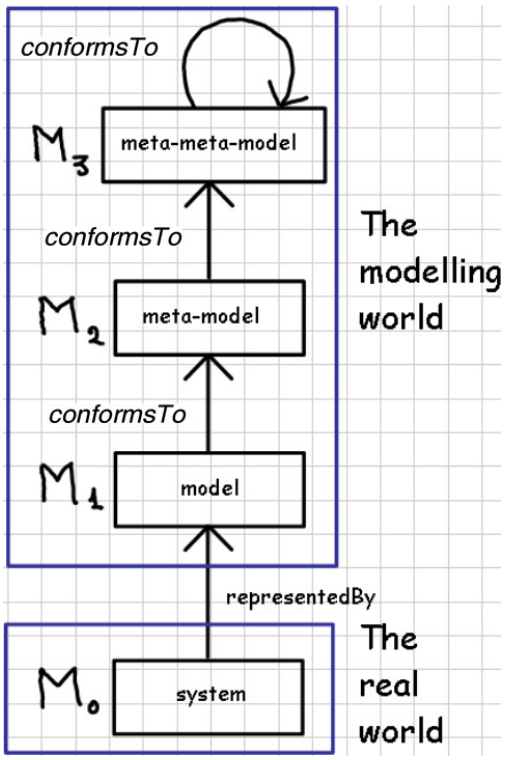


Figure B.5. Le schéma de relation résumé entre un système dans le monde réel
et un modèle, un métamodèle, un métamétamodèle dans le monde de modélisation
(où le métamétamodèle doit aussi être conforme de façon récursive à lui-même) [11].

ANNEXE C - Processus de Construction de l'Éditeur de Scénario

L'éditeur de scénario est un éditeur graphique construit selon la technologie GMF, grâce auquel les utilisateurs peuvent : (1) modéliser une histoire, même s'ils n'ont aucune connaissance en logique linéaire, en créant les diagrammes, qui expriment les états, les entrées, les événements/actions, les sorties, les choix et la structure de discours de l'histoire, par les manipulations « glisser/déposer » simples ; (2) recevoir automatiquement le modèle exprimé en XML correspondant à ces diagrammes.

Pour cela, nous avons construit l'éditeur de scénario comme un projet GMF, qui fournit un environnement de modélisation et de génération de code afin de développer des éditeurs graphiques en Eclipse à travers l'approche IDM. Le processus de production de l'éditeur de scénario est donné par la Figure C.1. Nous précisons, dans la suite, les aspects importants de ce processus.

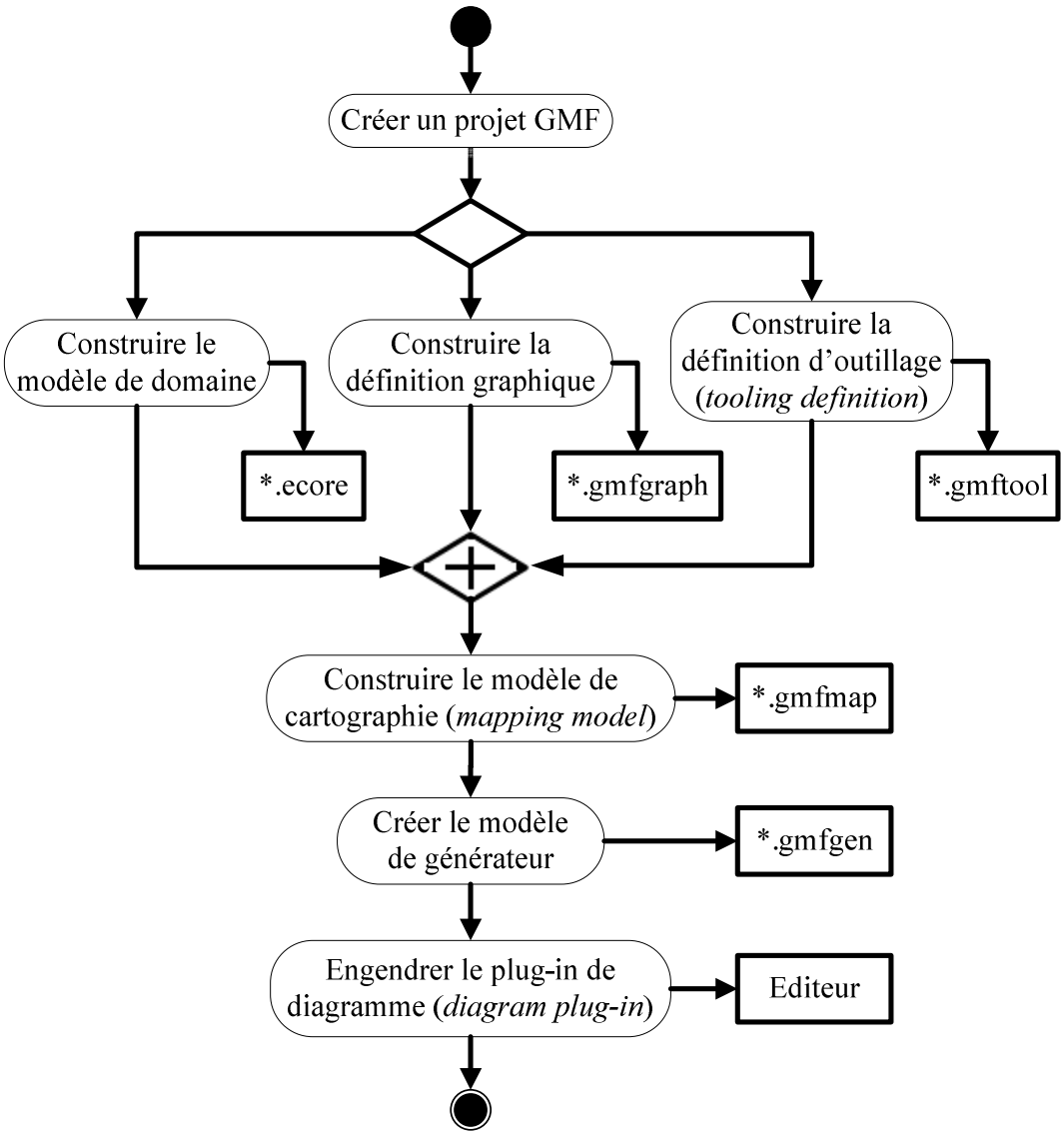


Figure C.1. Processus de construction de l'éditeur de scénario.

- Le modèle de domaine (modèle *ecore* – voir Figure IV.6) définit les éléments utilisés dans le langage de modélisation de l'éditeur de scénario, ainsi que les attributs des éléments et les relations entre ces éléments.

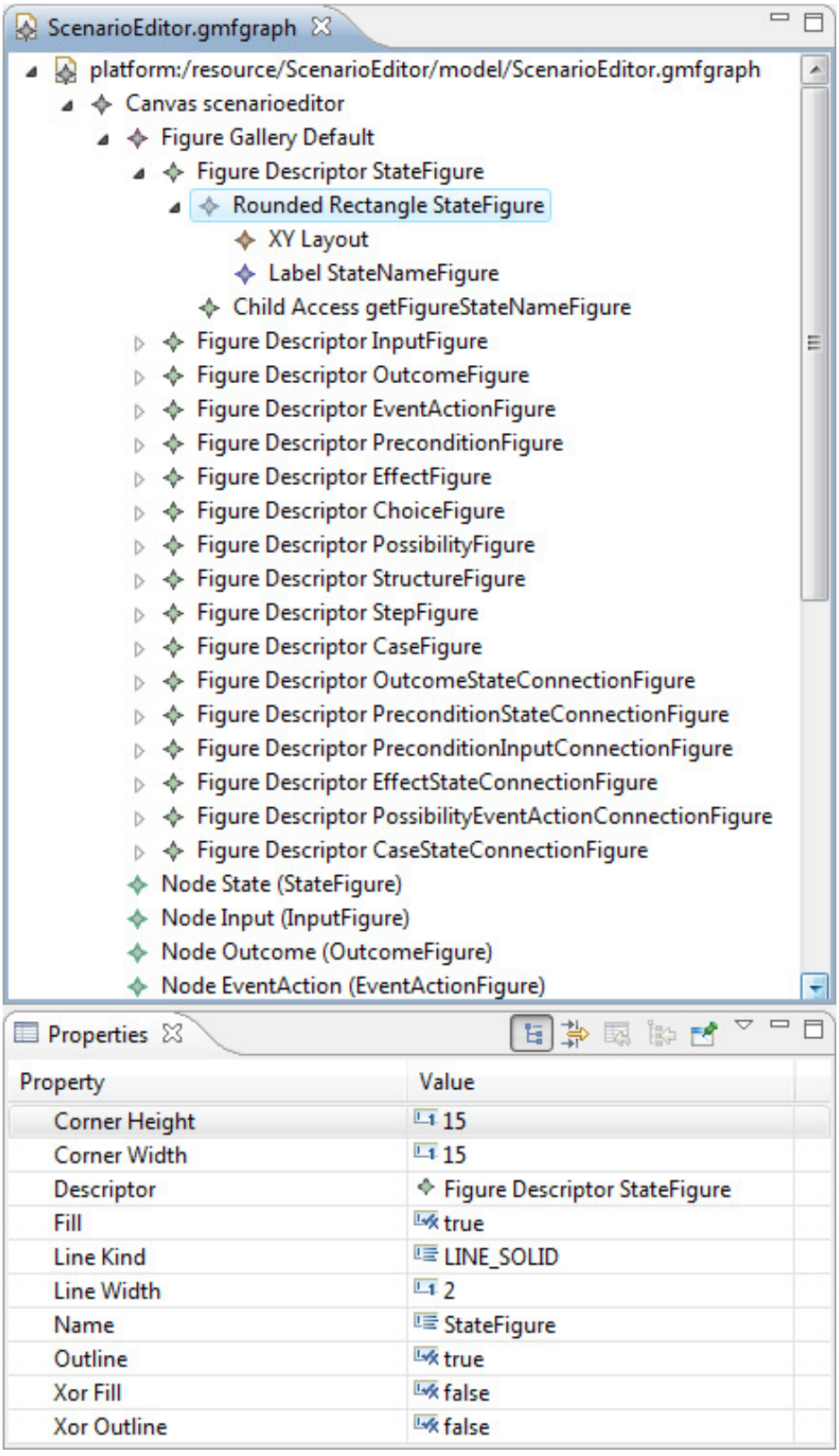


Figure C.2. Modèle *gmfgraph* de l'éditeur de scénario.

- Le modèle *gmfgraph* (voir Figure C.2) contient les informations définissant les éléments graphiques (tels que nœuds, liens, étiquettes…) qui apparaissent dans l'éditeur. Par exemple, les nœuds représentant les états sont les rectangles dont les angles sont ronds (avec « Corner Height » = 15, « Corner Width » = 15, « Line Kind » = LINE_SOLID, « Line Width » = 2), ils peuvent être mis à n'importe où dans l'éditeur (XY Layout) et possèdent une étiquette qui exprime leur nom.

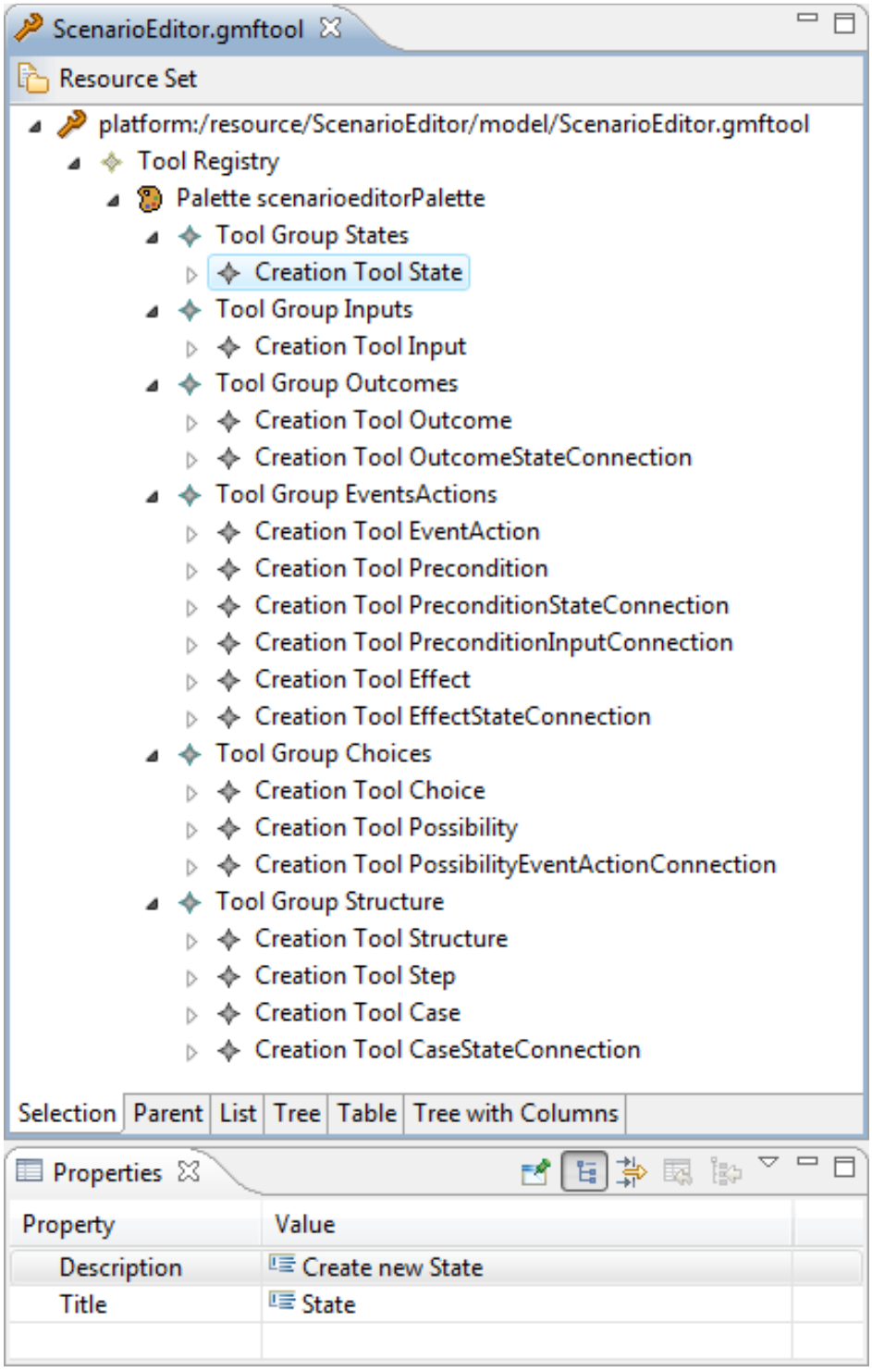


Figure C.3. Modèle *gmftool* de l'éditeur de scénario.

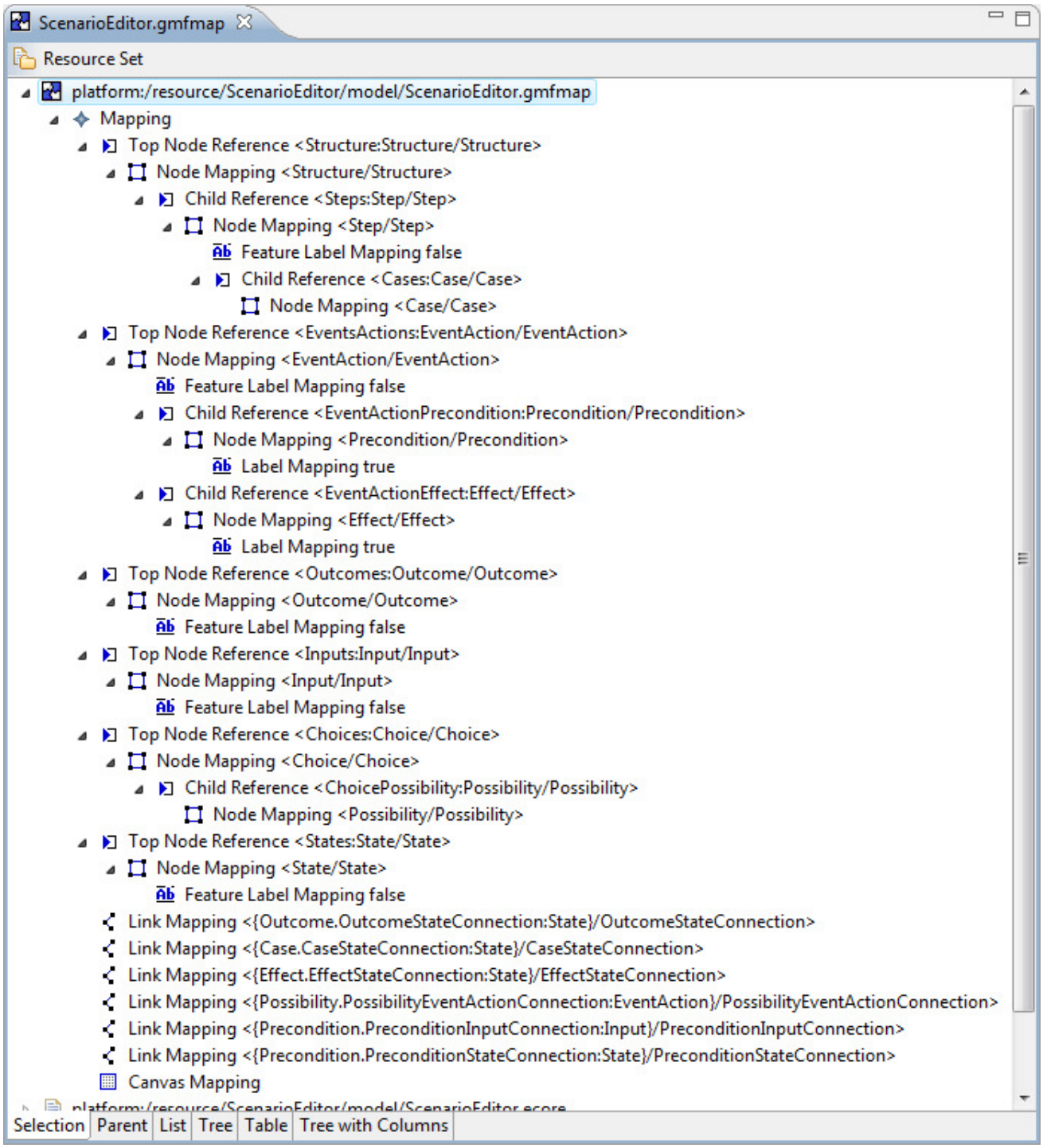


Figure C.4. Modèle *gmfmap* de l'éditeur de scénario.

- Le modèle *gmftool* (voir Figure C.3) décrit l'oganisation des termes du langage graphique dans la palette (la barre d'outils) de l'éditeur. Dans la mise en œuvre que nous avons effectuée, elle est composée de 6 groupes : (1) le groupe « États » comprend un nœud pour créer des états ; (2) le groupe « Entrées » comprend un nœud pour créer des entrées ; (3) le groupe « Sorties » comprend un nœud pour créer des sorties et un lien pour créer des liaisons d'une sortie à ses états ; (4) le groupe « Événements/Actions » comprend un nœud pour créer des événements/actions, deux nœuds pour créer des préconditions et des effets dans les événements/actions, deux

liens pour créer des liaisons d'une précondition à ses états et à son entrée, un lien pour créer des liaisons d'un effet à ses états ; (5) le groupe « Choix » comprend un nœud pour créer des choix, un nœud pour créer des possibilités dans les choix et un lien pour créer une liaison d'une possibilité à son événement/action correspondant ; (6) le groupe « Structure » comprend un nœud pour créer la structure de discours, un nœud pour créer des étapes dans la structure, un nœud pour créer des cas dans les étapes, et un lien pour créer des liaisons d'un cas à ses états.

- Le modèle *gmfmap* de l'éditeur de scénario (voir Figure C.4) est contruit en combinant les trois modèles ci-dessus. Il est utilisé pour préciser quels attributs des éléments graphiques sont apparus dans le cadre 1 de l'éditeur, quels attributs des éléments graphiques sont apparus dans le cadre 3 de l'éditeur (voir Figure IV.7), ainsi que les relations/positions des éléments graphiques dans les diagrammes construits lors de la modélisation. Par exemple, l'attribut *Name* des états, des entrées, des sorties, des événements/actions sont apparus dans le cadre 1 tandis que leurs autres attributs (tels que *Description*, *PriorityOrdre*, *IsInitialAvailableState*) sont apparus dans le cadre 3 de l'éditeur.

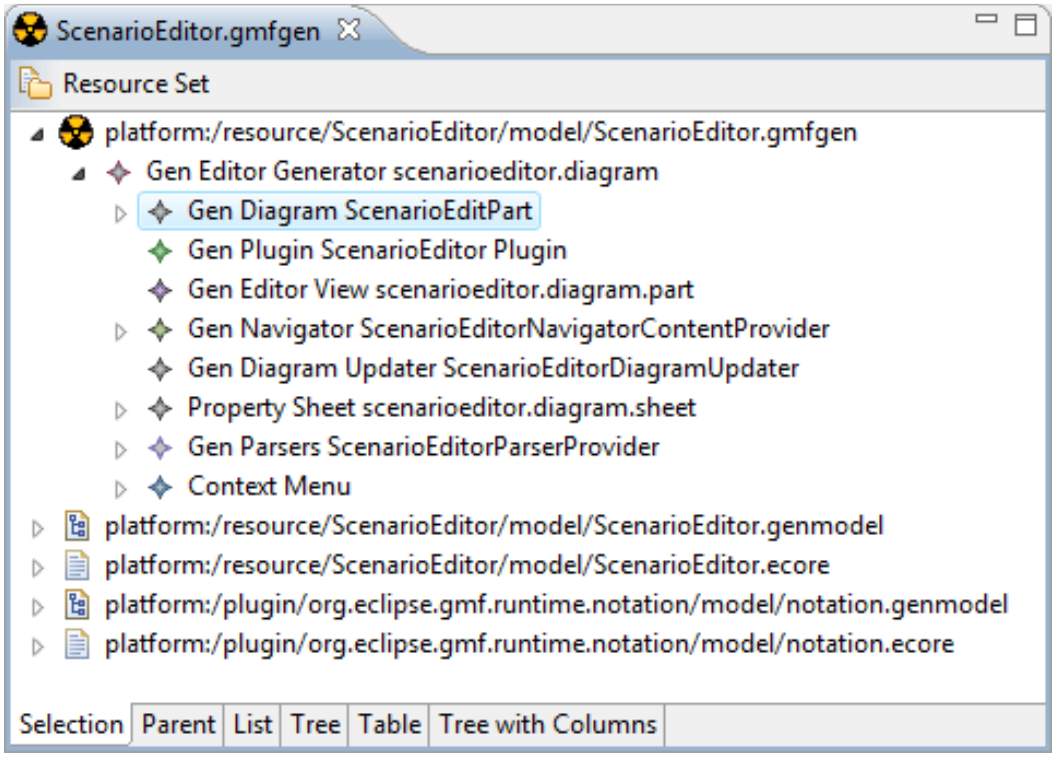


Figure C.5. Modèle *gmfgen* de l'éditeur de scénario.

- Le modèle *gmfgen* de l'éditeur de scénario (voir Figure C.5) est produit automatiquement à partir du modèle *gmfmap*. Son objectif est d'établir les propriétés pour la génération de code correspondant à l'éditeur.
- Le code de l'éditeur est engendré automatiquement sous forme d'un plug-in de diagramme (*diagram plug-in*) à partir du modèle *gmfgen*. Ensuite, nous l'exécutons comme une application Eclipse, et donc obtenons l'éditeur de scénario dont l'interface utilisateur est donnée dans la Figure IV.7.

ANNEXE D - Modélisation de Scénario du Jeu Vidéo « Le Petit Chaperon Rouge »

À titre d'exemple, nous avons construit un jeu vidéo réel basé sur l'histoire « Le Petit Chaperon rouge » (PCr). Ce jeu vidéo est appuyé sur la version de Grimms [3] à laquelle nous avons ajouté quelques options pour les personnages (joueur et non-joueur), en vue d'augmenter l'imprévisibilité ainsi que l'interactivité du jeu. Voici ci-dessous sa modélisation de scénario détaillée où : le joueur joue le PCr ; le système de pilotage dirige les actions des quatre personnages non-joueurs : la mère, le loup, la grand-mère et le chasseur ; le jeu se déroule dans trois lieux : la maison du PCr, le bois et la maison de la grand-mère ; l'objectif du jeu (la terminaison souhaitée des auteurs) : le loup est mort, la grand-mère et le PCr sont vivants.

- États du jeu
    - Ga : La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Gm : Le PCr rencontre le loup dans le bois
    - Gge : La grand-mère est mangée par le loup
    - Gle : Le PCr est mangé par le loup
    - Gr : Le jeu atteint l'objectif (le loup est mort, la grand-mère et le PCr sont vivants)
- États du joueur (états du PCr)
    - Las : La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Lag : Le PCr accepte d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Lr : Le PCr refuse à apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Len : Le PCr entre dans le bois
    - Lmw : Le PCr rencontre le loup dans le bois
    - Ls : Le PCr dit au loup où est la maison de la grand-mère
    - Lds : Le PCr ne dit pas au loup où est la maison de la grand-mère mais continue à y aller
    - Lp : Le PCr est proposé de contempler le paysage dans le bois
    - Lev : Le PCr contemple le paysage dans le bois
    - Ldl : Le PCr n'écoute pas la proposition du loup contemplant le paysage dans le bois mais continue à aller à la maison de la grand-mère
    - Lar : Le PCr arrive à la maison de la grand-mère
    - Lf : Le PCr se sent étrange et peureux parce que la porte de la maison de la grand-mère est ouverte
    - Lg : Le PCr retourne à sa maison parce qu'il a peur (quand il trouve que la porte de la maison de la grand-mère est ouverte)
    - Li : Le PCr est dans la maison de la grand-mère

- Lmh : Le PCr rencontre le loup dans la maison de la grand-mère
- Lt : Le PCr parle au loup dans la maison de la grand-mère
- Lew : Le PCr est mangé par le loup
- Lal : Le PCr est vivant et sort du ventre du loup

- État de la mère
  - Mw : La mère souhaite que la santé de la grand-mère devienne meilleure (cet état est disponible au moment initial)
- États du loup
  - Wmw : Le loup rencontre le PCr dans le bois
  - Wk : Le loup sait où est la maison de la grand-mère
  - Wf : Le loup suit le PCr secrètement afin d'aller à la maison de la grand-mère parce qu'il ne sait pas où elle est
  - Wra : Le loup court directement à la maison de la grand-mère (après avoir su où elle est) afin d'y arriver plus tôt que le PCr
  - Wre : Le loup court directement à la maison de la grand-mère après avoir mangé le PCr parce qu'il sait où elle est
  - Was : Le loup arrive à la maison de la grand-mère plus tôt que le PCr
  - Waa : Le loup arrive à la maison de la grand-mère juste après le PCr parce qu'il l'a secrètement suivi
  - Wae : Le loup arrive à la maison de la grand-mère après avoir mangé le PCr
  - Wi : Le loup est dans la maison de la grand-mère
  - Weg : Le loup mange la grand-mère
  - Ww : Le loup attend le PCr dans la maison de la grand-mère après l'avoir mangé
  - Wmh : Le loup rencontre le PCr dans la maison de la grand-mère
  - Wtl : Le loup parle au PCr dans la maison de la grand-mère
  - Wel : Le loup mange le PCr
  - Wst : Le loup reste à la maison de la grand-mère afin de dormir après avoir mangé la grand-mère et le PCr
  - Wrw : Le loup retourne au bois après avoir mangé la grand-mère et le PCr
  - Wth : Le loup parle au chasseur
  - Wd : Le loup est mort
- États de la grand-mère
  - GMs : La grand-mère est malade (cet état est disponible au moment initial)
  - GMi : La grand-mère est dans sa maison (cet état est disponible au moment initial)
  - GMas : Le loup demande à la grand-mère d'ouvrir la porte
  - GMb : La grand-mère croit le loup et donc ouvre la porte
  - GMdb : La grand-mère ne croit pas le loup en raison de sa voix et donc n'ouvre pas la porte
  - GMe : La grand-mère est mangée par le loup
  - GMal : La grand-mère est vivante et sort du ventre du loup

- États du chasseur
    - Hsd : Le chasseur trouve que la porte de la maison de la grand-mère est ouverte
    - Hd : Le chasseur décide de prendre un coup d'œil parce que la porte de la maison de la grand-mère est ouverte
    - Hsw : Le chasseur voit le loup
    - Ht : Le chasseur parle au loup
- Entrées du joueur (choix d'action du PCr)
    - Ilm : Le PCr accepte d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère
    - Ir : Le PCr refuse à apporter un morceau de gâteau et une bouteille de vin à la grand-mère parce qu'il a un autre travail à faire
    - Is : Le PCr dit au loup où est la maison de la grand-mère
    - Ids : Le PCr ne dit pas au loup où est la maison de la grand-mère mais continue à y aller
    - Ilw : Le PCr écoute la proposition du loup contemplant le paysage dans le bois
    - Idl : Le PCr n'écoute pas la proposition du loup contemplant le paysage dans le bois mais continue à aller à la maison de la grand-mère
    - Ig : Le PCr décide de retourner à sa maison parce qu'il a peur (quand il trouve que la porte de la maison de la grand-mère est ouverte)
    - Ie : Le PCr décide d'entrer dans la maison de la grand-mère même s'il se sent peureux (quand il trouve que la porte de la maison de la grand-mère est ouverte)
- États disponibles initiaux : GMs (la grand-mère est malade), GMi (la grand-mère est chez elle), Mw (la mère souhaite que la santé de la grand-mère devienne meilleure)
- Événements/actions du jeu (leur ordre de priorité augmente selon le nom des événements/actions : l'ordre de priorité de EA01 est de 1, de EA02 est de 2…)
    - EA01 – La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère parce qu'elle est malade : GMs ⊗ Mw ⊸ Ga ⊗ Las
    - EA02 – Le PCr accepte d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère : Ga ⊗ Las ⊗ Ilm ⊸ Lag
    - EA03 – Le PCr refuse à apporter un morceau de gâteau et une bouteille de vin à la grand-mère parce qu'il a un autre travail à faire : Ga ⊗ Las ⊗ Ir ⊸ Lr
    - EA04 – La mère convainc le PCr, donc il accepte d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère : Lr ⊸ Lag
    - EA05 – Le PCr entre dans le bois : Lag ⊸ Len
    - EA06 – Le PCr rencontre le loup dans le bois : Len ⊸ Gm ⊗ Lmw ⊗ Wmw
    - EA07 – Le PCr dit au loup où est la maison de la grand-mère: Gm ⊗ Lmw ⊗ Wmw ⊗ Is ⊸ Ls ⊗ Wk
    - EA08 – Le PCr ne dit pas au loup où est la maison de la grand-mère mais continue à y aller ; donc le loup le suit secrètement : Gm ⊗ Lmw ⊗ Wmw ⊗ Ids ⊸ Lds ⊗ Wf

- EA09 – Le loup propose au PCr de contempler le paysage dans le bois après avoir su où est la maison de la grand-mère : Ls ⊗ Wk ⊸ Lp ⊗ Wk
- EA10 – Le loup mange le PCr dans le bois après avoir su où est la maison de la grand-mère : Ls ⊗ Wk ⊸ Gle ⊗ Lew ⊗ Wel ⊗ Wk
- EA11 – Le PCr écoute la proposition du loup contemplant le paysage dans le bois : Lp ⊗ Ilw ⊸ Lev
- EA12 – Le PCr n'écoute pas la proposition du loup contemplant le paysage dans le bois mais continue à aller à la maison de la grand-mère : Lp ⊗ Idl ⊸ Ldl
- EA13 – Le loup court directement à la maison de la grand-mère afin d'y arriver plus tôt que le PCr parce qu'il sait où elle est (en ce moment le PCr est en train de contempler le paysage dans le bois) : Wk ⊗ Lev ⊸ Wra ⊗ Lev
- EA14 – Le loup court directement à la maison de la grand-mère afin d'y arriver plus tôt que le PCr parce qu'il sait où elle est (en ce moment le PCr est aussi en train d'y aller) : Wk ⊗ Ldl ⊸ Wra ⊗ Ldl
- EA15 – Le loup court directement à la maison de la grand-mère après avoir mangé le PCr parce qu'il sait où elle est : Wk ⊗ Gle ⊗ Lew ⊗ Wel ⊸ Wre ⊗ Gle ⊗ Lew ⊗ Wel
- EA16 – Le PCr arrive à la maison de la grand-mère ; le loup y arrive aussi juste après le PCr parce qu'il l'a secrètement suivi : Lds ⊗ Wf ⊸ Lar ⊗ Waa
- EA17 – Le PCr entre dans la maison de la grand-mère (il ne sait pas que le loup l'a secrètement suivi) : Lar ⊗ Waa ⊸ Li ⊗ Waa
- EA18 – Le loup arrive à la maison de la grand-mère après avoir mangé le PCr : Wre ⊸ Wae
- EA19 – Le loup arrive à la maison de la grand-mère plus tôt que le PCr : Wra ⊸ Was
- EA20 – Le loup demande à la grand-mère d'ouvrir la porte en disant qu'il est la mère (en ce moment le PCr est dans la maison de la grand-mère) : Waa ⊗ Li ⊗ GMi ⊸ GMas ⊗ Li ⊗ GMi
- EA21 – Le loup demande à la grand-mère d'ouvrir la porte en disant qu'il est le PCr (en ce moment il l'a mangé) : Wae ⊗ GMi ⊸ GMas ⊗ GMi
- EA22 – Le loup demande à la grand-mère d'ouvrir la porte en disant qu'il est le PCr (en ce moment le PCr n'arrive pas encore à la maison de la grand-mère) : Was ⊗ GMi ⊸ GMas ⊗ GMi
- EA23 – Le loup détruit la porte et entre dans la maison de la grand-mère sans demander à la grand-mère d'ouvrir la porte (en ce moment le PCr est dans la maison de la grand-mère) : Waa ⊗ Li ⊸ Wi ⊗ Li
- EA24 – Le loup détruit la porte et entre dans la maison de la grand-mère sans demander à la grand-mère d'ouvrir la porte (en ce moment il a mangé le PCr) : Wae ⊸ Wi
- EA25 – Le loup détruit la porte et entre dans la maison de la grand-mère sans demander à la grand-mère d'ouvrir la porte (en ce moment le PCr n'arrive pas encore à la maison de la grand-mère) : Was ⊸ Wi

- EA26 – La grand-mère croit le loup et donc ouvre la porte : GMas ⊸ GMb
- EA27 – Le loup entre dans la maison de la grand-mère parce que la porte est ouverte : GMb ⊸ Wi
- EA28 – La grand-mère ne croit pas le loup en raison de sa voix et donc n'ouvre pas la porte : GMas ⊸ GMdb
- EA29 – Le loup détruit la porte et entre dans la maison de la grand-mère parce qu'elle ne le croit pas : GMdb ⊸ Wi
- EA30 – Le loup mange la grand-mère et le PCr (en ce moment le PCr est dans la maison de la grand-mère) : Wi ⊗ Li ⊗ GMi ⊸ Gle ⊗ Lew ⊗ Wel ⊗ Gge ⊗ Weg ⊗ GMe
- EA31 – Le loup mange la grand-mère (en ce moment le PCr est en train d'aller à la maison de la grand-mère) : Wi ⊗ GMi ⊗ Ldl ⊸ Gge ⊗ Weg ⊗ GMe ⊗ Ldl
- EA32 – Le loup mange la grand-mère (en ce moment le PCr est en train de contempler le paysage dans le bois) : Wi ⊗ GMi ⊗ Lev ⊸ Gge ⊗ Weg ⊗ GMe ⊗ Lev
- EA33 – Le loup mange la grand-mère (en ce moment il a mangé le PCr) : Wi ⊗ GMi ⊗ Gle ⊗ Lew ⊗ Wel ⊸ Gge ⊗ Weg ⊗ GMe ⊗ Gle ⊗ Lew ⊗ Wel
- EA34 – Le loup attend le PCr dans la maison de la grand-mère afin de continuer à le manger (en ce moment le PCr est en train d'aller à la maison de la grand-mère) : Gge ⊗ Weg ⊗ GMe ⊗ Ldl ⊸ Ww ⊗ Gge ⊗ Weg ⊗ GMe ⊗ Ldl
- EA35 – Le loup attend le PCr dans la maison de la grand-mère afin de continuer à le manger (en ce moment le PCr est en train de contempler le paysage dans le bois) : Gge ⊗ Weg ⊗ GMe ⊗ Lev ⊸ Ww ⊗ Gge ⊗ Weg ⊗ GMe ⊗ Lev
- EA36 – Le PCr arrive à la maison de la grand-mère (il n'a pas contemplé le paysage dans le bois ; et en ce moment le loup est en train de l'attendre dans la maison de la grand-mère) : Ldl ⊗ Ww ⊸ Lar ⊗ Ww
- EA37 – Le PCr arrive à la maison de la grand-mère après avoir contemplé le paysage dans le bois (en ce moment le loup est en train de l'attendre dans la maison de la grand-mère) : Lev ⊗ Ww ⊸ Lar ⊗ Ww
- EA38 – Le PCr se sent étrange et peureux parce que la porte de la maison de la grand-mère est ouverte (en ce moment le loup est en train de l'attendre dans la maison de la grand-mère) : Lar ⊗ Ww ⊸ Lf ⊗ Ww
- EA39 – Le PCr décide de retourner à sa maison parce qu'il a peur (quand il trouve que la porte de la maison de la grand-mère est ouverte) : Lf ⊗ Ig ⊸ Lg
- EA40 – Le loup trouve que le PCr est en train de retourner à sa maison, par conséquent il le court après et le mange : Lg ⊗ Ww ⊸ Gle ⊗ Lew ⊗ Wel
- EA41 – Le PCr décide d'entrer dans la maison de la grand-mère même s'il se sent peureux (car la porte de la maison de la grand-mère est ouverte) : Lf ⊗ Ie ⊸ Li
- EA42 – Le PCr rencontre le loup dans la maison de la grand-mère : Li ⊗ Ww ⊸ Lmh ⊗ Wmh

- EA43 – Le PCr parle au loup dans la maison de la grand-mère : Lmh ⊗ Wmh ⊸ Lt ⊗ Wtl
- EA44 – Le loup mange le PCr après leur conversation dans la maison de la grand-mère : Lt ⊗ Wtl ⊸ Gle ⊗ Lew ⊗ Wel
- EA45 – Le loup reste à la maison de la grand-mère afin de dormir après avoir mangé la grand-mère et le PCr : Gge ⊗ Weg ⊗ GMe ⊗ Gle ⊗ Lew ⊗ Wel ⊸ Wst
- EA46 – Le loup retourne au bois après avoir mangé la grand-mère et le PCr : Gge ⊗ Weg ⊗ GMe ⊗ Gle ⊗ Lew ⊗ Wel ⊸ Wrw
- EA47 – Le chasseur passe par la maison de la grand-mère et voit le loup retourner au bois : Wrw ⊸ Hsw ⊗ Wrw
- EA48 – Le chasseur passe par la maison de la grand-mère et voit sa porte ouverte : Wst ⊸ Hsd ⊗ Wst
- EA49 – Le chasseur se sent étrange parce que la porte de la maison de la grand-mère est ouverte, et donc décide de prendre un coup d'œil : Hsd ⊸ Hd
- EA50 – Le chasseur voit le loup dans la maison de la grand-mère : Hd ⊗ Wst ⊸ Hsw ⊗ Wst
- EA51 – Le chasseur parle au loup dans la maison de la grand-mère : Hsw ⊗ Wst ⊸ Ht ⊗ Wth
- EA52 – Le chasseur parle au loup quand il le voit retourner au bois : Hsw ⊗ Wrw ⊸ Ht ⊗ Wth
- EA53 – Le chasseur sait que le loup vient de manger la grand-mère et le PCr, donc il le tue afin de les sauver : Ht ⊗ Wth ⊸ Wd ⊗ Lal ⊗ GMal
- EA54 – Le jeu termine en succès : Lal ⊗ GMal ⊗ Wd ⊸ Gr

- Choix du joueur (les possibilités dans chacun des choix sont reliées par le connecteur &)
  - Choisir entre EA02 et EA03 : Le joueur soit accepte d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère soit refuse à faire cela parce qu'il a un autre travail à faire
  - Choisir entre EA07 et EA08 : Le joueur soit dit au loup où est la maison de la grand-mère soit ne dit pas au loup mais continue à y aller
  - Choisir entre EA11 et EA12 : Le joueur soit écoute la proposition du loup contemplant le paysage dans le bois soit n'écoute pas la proposition du loup mais continue à aller à la maison de la grand-mère
  - Choisir entre EA39 et EA41 : Le joueur soit retourne à sa maison parce qu'il a peur (quand il trouve que la porte de la maison de la grand-mère est ouverte) soit continue à entrer dans la maison de la grand-mère
- Choix du système de pilotage (les possibilités dans chacun des choix sont reliées par le connecteur ⊕ et leur pourcentage est égal)
  - Choix du loup
    - Choisir entre EA09 et EA10 : Le loup soit propose au PCr de contempler le paysage soit mange le PCr dans le bois après avoir su où est la maison de la grand-mère

        - Choisir entre EA20 et EA23 : Le loup soit demande à la grand-mère d'ouvrir la porte en disant qu'il est la mère soit détruit la porte et entre dans la maison de la grand-mère sans lui demander d'ouvrir la porte (en ce moment le PCr est dans la maison de la grand-mère)
        - Choisir entre EA21 et EA24 : Le loup soit demande à la grand-mère d'ouvrir la porte en disant qu'il est le PCr soit détruit la porte et entre dans la maison de la grand-mère sans lui demander d'ouvrir la porte (en ce moment il a mangé le PCr dans le bois)
        - Choisir entre EA22 et EA25 : Le loup soit demande à la grand-mère d'ouvrir la porte en disant qu'il est le PCr soit détruit la porte et entre dans la maison de la grand-mère sans lui demander d'ouvrir la porte (en ce moment le PCr n'arrive pas encore à la maison de la grand-mère)
        - Choisir entre EA45 et EA46 : Le loup soit reste à la maison de la grand-mère afin de dormir soit retourne au bois après avoir mangé la grand-mère et le PCr
    - Choix de la grand-mère
        - Choisir entre EA26 et EA28 : La grand-mère soit croit le loup et donc ouvre la porte soit ne croit pas le loup en raison de sa voix et donc n'ouvre pas la porte
- Sortie du jeu : Gr (le loup est mort, la grand-mère et le PCr sont vivants)
- Structure de discours : La structure de discours du jeu comprend quatre étapes, dans chaque étape, il n'y a qu'un cas :
    - **Etape 1** – La mère demande au PCr d'apporter un morceau de gâteau et une bouteille de vin à la grand-mère, cette étape ne comprend que l'état Ga ;
    - **Etape 2** – Le PCr rencontre le loup dans le bois, cette étape ne comprend que l'état Gm ;
    - **Etape 3** – La grand-mère et le PCr sont mangés par le loup, cette étape comprend les deux états Gge et Gle ;
    - **Etape 4** – Le loup est mort, la grand-mère et le PCr sont vivants, cette étape ne comprend que l'état Gr.

  Par conséquent, tous les discours possibles dans le scénario créé doivent passer toutes les étapes énoncées plus haut afin de garantir les objectifs désirés des auteurs.
- Enfin, on obtient le séquent de logique linéaire suivant, qui représente le scénario modélisé du jeu au moment initial : Mw, GMs, GMi, EA01 ⊗ (Ilm ⊗ EA02 & Ir ⊗ EA03 ⊗ EA04) ⊗ EA05 ⊗ EA06 ⊗ (Is ⊗ EA07 ⊗ (EA09 ⊗ (Ilw ⊗ EA11 ⊗ EA13 ⊗ EA19 ⊗ (EA22 ⊗ (EA26 ⊗ EA27 ⊕ EA28 ⊗ EA29) ⊕ EA25) ⊗ EA32 ⊗ EA35 ⊗ EA37 & Idl ⊗ EA12 ⊗ EA14 ⊗ EA19 ⊗ (EA22 ⊗ (EA26 ⊗ EA27 ⊕ EA28 ⊗ EA29) ⊕ EA25) ⊗ EA31 ⊗ EA34 ⊗ EA36) ⊗ EA38 ⊗ (Ig ⊗ EA39 ⊗ EA40 & Ie ⊗ EA41 ⊗ EA42 ⊗ EA43 ⊗ EA44) ⊕ EA10 ⊗ EA15 ⊗ EA18 ⊗ (EA21 ⊗ (EA26 ⊗ EA27 ⊕ EA28 ⊗ EA29) ⊕ EA24) ⊗ EA33) & Ids ⊗ EA08 ⊗ EA16 ⊗ EA17 ⊗ (EA20 ⊗ (EA26 ⊗ EA27 ⊕ EA28 ⊗ EA29) ⊕ EA23) ⊗ EA30) ⊗ (EA45 ⊗ EA48 ⊗ EA49 ⊗ EA50 ⊗ EA51 ⊕ EA46 ⊗ EA47 ⊗ EA52) ⊗ EA53 ⊗ EA54 ⊢ Gr. Ce séquent est représenté par le segment exprimé sous forme XML ci-dessous, qui est conforme au métamodèle du calcul des séquents :

```
<Sequent>
   <LeftPart>
      <AvailableState Name="Mw"/>
      <AvailableState Name="GMs"/>
      <AvailableState Name="GMi"/>
      <Formula Order="1">
         <Element Name="(" Order="1" Type="Open Parenthesis"
         ParenthesisLevel="1"/>
         <Element Name="GMs" Order="2" Type="Atom"/>
         <Element Name="times" Order="3" Type="Multiplicative
         Conjunction"/>
         <Element Name="Mw" Order="4" Type="Atom"/>
         <Element Name="imply" Order="5" Type="Linear Implication"
         EventActionName="EA01" EventActionPriorityOrder="1"/>
         <Element Name="Ga" Order="6" Type="Atom"/>
         <Element Name="times" Order="7" Type="Multiplicative
         Conjunction"/>
         <Element Name="Las" Order="8" Type="Atom"/>
         <Element Name=")" Order="9" Type="Close Parenthesis"
         ParenthesisLevel="1"/>
         <Element Name="times" Order="10" Type="Multiplicative
         Conjunction"/>
         …
      <Formula/>
   <LeftPart/>
   <RightPart>
      <Outcome Name="">
         <State Name="Gr"/>
      <Outcome/>
   <RightPart/>
<Sequent/>
```

ANNEXE E – Système Auteur pour la Production des Scénarios Valides avec les Événements/Actions Complexes

Afin d'aider l'utilisateur à produire un scénario de bonne qualité avec les événements/actions complexes, qui est modélisé en logique linéaire et qui est exécutable par un système de pilotage, nous avons développé un outil dont l'interface utilisateur est donnée dans la Figure E.1. Pour cela, nous proposons (1) un algorithme de parcours de toutes les preuves d'un séquent de logique linéaire, (2) un modèle de données employé dans le processus d'exécuter l'algorithme, (3) un modèle de scénario exécutable qui est conforme au métamodèle du calcul des séquents, et (4) un algorithme de transformation du modèle de données en le modèle de scénario exécutable. Ces modèles et algorithmes, qui sont implémentés par l'outil, sont décrits dans les sections suivantes.

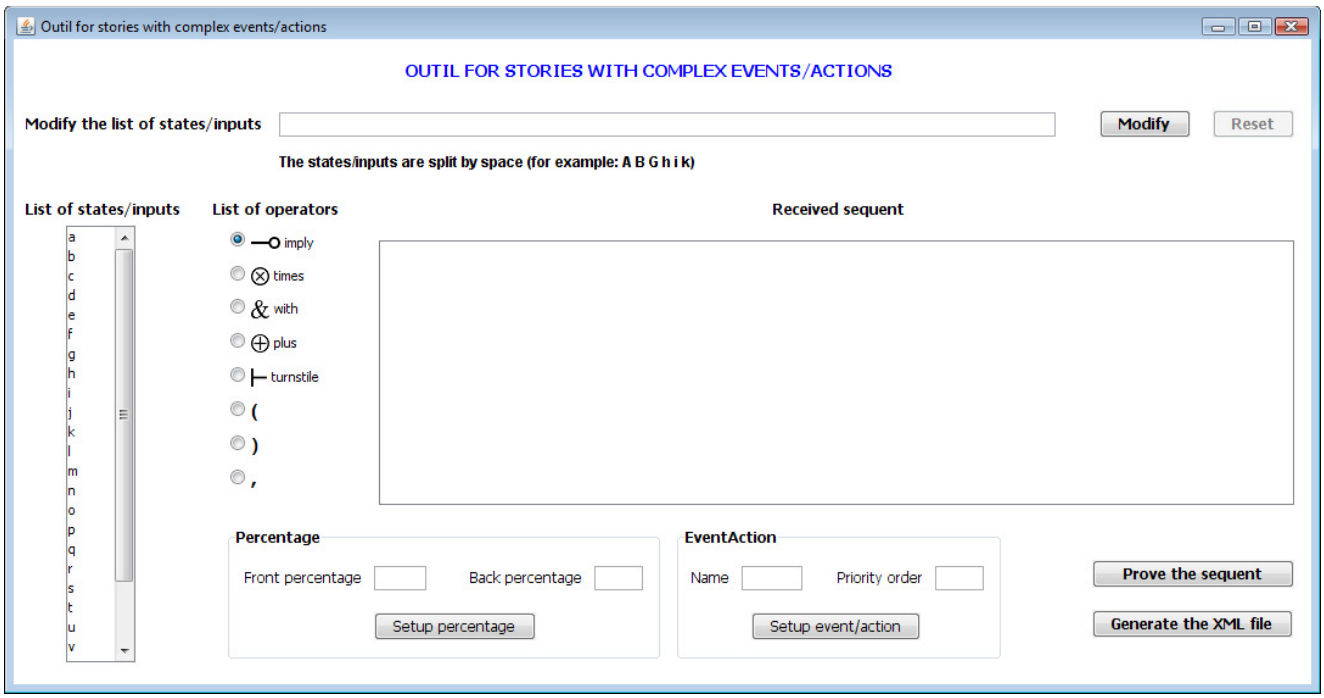


Figure E.1. Interface utilisateur de l'outil pour la production d'un scénario de bonne qualité avec les événements/actions complexes.

La démarche, qui est supportée par l'outil, pour la production des scénarios de bonne qualité avec les événements/actions complexes, est donnée dans la Figure E.2 :

- Tout d'abord, l'utilisateur (auteur, expert du domaine) modélise une histoire en logique linéaire, en exprimant ses aspects importants (tels que ses états et ses sorties souhaitées, les états et les choix des personnages joueur/non-joueur, les événements/actions, *etc.*). Le résultat obtenu est un séquent de logique linéaire représentant le scénario modélisé, par exemple A, (B ⊸ C) ⊸ D, B ⊸ C, (A ⊗ D ⊸ E) ⊕ (A ⊗ D ⊸ F) ⊢ E (pour mieux comprendre, voir Section III.3, chapitre III –

Modélisation d'une histoire en logique linéaire, un exemple de la cette méthode de modélisation est aussi donné en Annexe D).

- L'utilisateur introduit ce séquent de logique linéaire dans le modèle de données utilisé par l'outil en rédigeant le séquent au moyen de l'interface utilisateur.

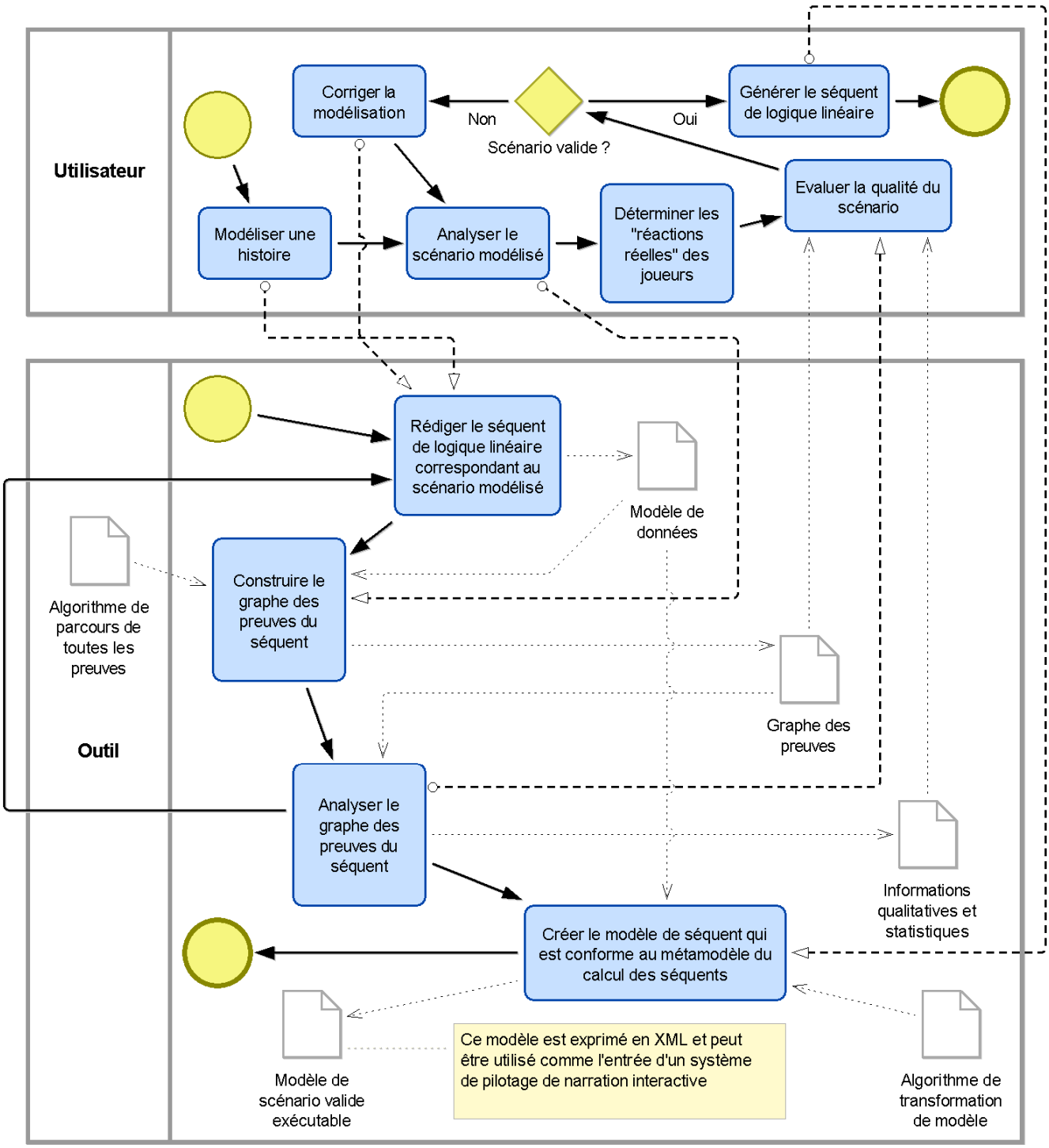


Figure E.2. La démarche, qui est supportée par l'outil, pour la production d'un scénario de bonne qualité dans le cas d'utilisation d'événements/actions complexes (construite selon la norme *BPMN* [104]).

- La preuve du séquent au moyen de l'outil utilise notre algorithme de parcours de toutes les preuves, et donc le graphe des preuves du séquent est construit. L'analyse automatique de ce graphe donne les informations qualitatives et statistiques concernant le scénario courant telles que : les branches réussies/échouées, les branches réussies pour chaque sortie et les situations inaccessibles. Ainsi, l'utilisateur peut vérifier la satisfaction du scénario par rapport aux deux propriétés de narration *Accessibilité* et *Absence de blocage* (pour mieux comprendre, voir Section III.4, chapitre III – Formalisation des propriétés de narration).
- Comme nous l'avons présenté en Section IV.4, chapitre IV – Démarche utilisateur, les « réactions réelles » (ou la connaissance sensorielle) des joueurs résultent des exécutions sur des échantillons de joueurs au cours desquelles, ces informations sont mesurées.
- Les résultats obtenus ci-dessus, donnent une évaluation de la qualité du scénario. En retour, la modélisation peut être modifiée afin de produire un nouveau scénario/séquent. L'itération de ce processus est faite jusqu'à ce que l'utilisateur obtienne un scénario, qui est le plus pertinent au regard de ses objectifs, en d'autres termes, jusqu'à ce que l'utilisateur trouve que sa qualité est satisfaisante (que le scénario soit valide).
- Enfin, l'outil transforme le modèle de données en le modèle de scénario exécutable qui est exprimé en XML. Ce modèle correspond au scénario validé et est conforme au métamodèle du calcul des séquents (voir Figure IV.3 – chapitre IV). Il constitue l'entrée d'un système de pilotage de narration interactive. Grâce aux mécanismes de déduction automatique du séquent, le système de pilotage peut exécuter ce scénario valide lors du déroulement du jeu.

Dans les sections suivantes, nous présentons tour à tour le modèle de données ainsi que l'algorithme de preuve de séquent, que nous avons construits, ceux qui sont implémentés par l'outil et lui permettent de produire et d'analyser le graphe des preuves d'un scénario avec les événements/actions complexes (nous ne décrivons pas l'algorithme de transformation de modèle, qui est aussi implémenté par l'outil, car il est trop simple).

E.1. Modèle de données utilisé par l'outil pour construire le graphe des preuves d'un scénario avec les événements/actions complexes

La Figure E.3 décrit le modèle de données utilisé par l'outil pour construire le graphe des preuves d'un scénario avec les événements/actions complexes, où :

- *Rule* : Les instances de *Rule* représentent les règles d'inférence de la logique linéaire. Le processus de preuve d'un séquent utilise septs règles suivantes (pour mieux comprendre, voir Section III.5.2.2.2 – chapitre III) :
    - ⊗L : Name = Multiplicative Conjunction ; PartType = 1 ; PriorityOrder = 1, cette règle est appliquée au connecteur ⊗ d'une formule dans la partie gauche (PartType = 1) du séquent.

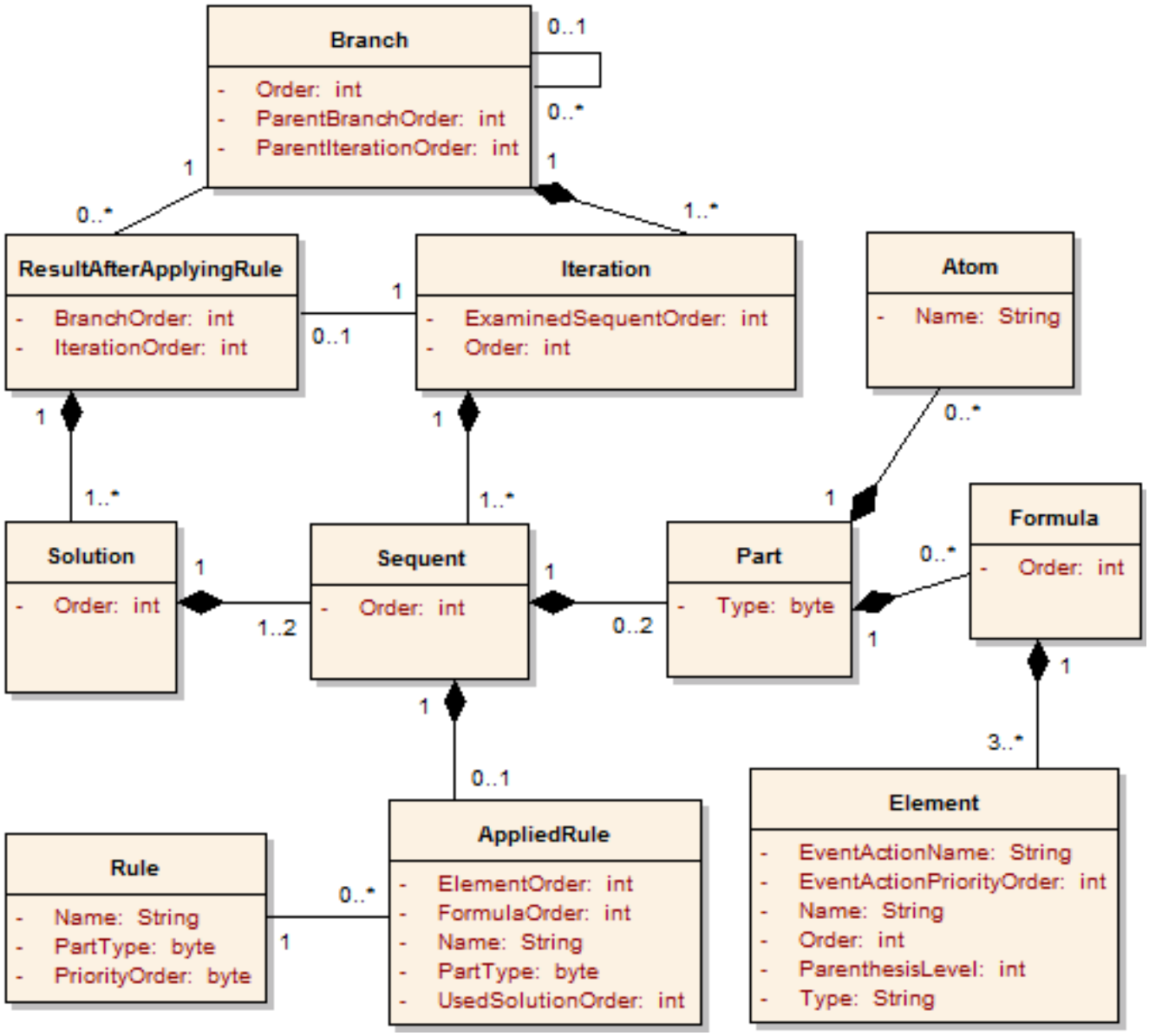


Figure E.3. Modèle de données utilisé par l'outil pour construire le graphe des preuves d'un scénario avec les événements/actions complexes.

- ⊕R : Name = Additive Disjunction ; PartType = 2 ; PriorityOrder = 2, cette règle est appliquée au connecteur ⊕ d'une formule dans la partie droite (PartType = 2) du séquent.
- &L : Name = Additive Conjunction ; PartType = 1 ; PriorityOrder = 3, cette règle est appliquée au connecteur & d'une formule dans la partie gauche du séquent.
- ⊕L : Name = Additive Disjunction ; PartType = 1 ; PriorityOrder = 4, cette règle est appliquée au connecteur ⊕ d'une formule dans la partie gauche du séquent.
- ⊸R : Name = Linear Implication ; PartType = 2 ; PriorityOrder = 5, cette règle est appliquée au connecteur ⊸ d'une formule dans la partie droite du séquent.

- ⊸L : Name = Linear Implication ; PartType = 1 ; PriorityOrder = 6, cette règle est appliquée au connecteur ⊸ d'une formule dans la partie gauche du séquent.
- ⊗R : Name = Multiplicative Conjunction ; PartType = 2 ; PriorityOrder = 7, cette règle est appliquée au connecteur ⊗ d'une formule dans la partie droite du séquent.

- Un *séquent* est composé de deux *parties* (séparées par ⊢) : Partie gauche (Type = 1) et Partie droite (Type = 2), mais chacune peut être vide. La partie gauche comprend un ensemble d'*atomes* et un ensemble de *formules*, tandis que la partie droite ne comprend que soit un atome soit une formule. Les atomes dans la partie gauche représentent les états disponibles courants du séquent, tandis que l'atome unique dans la partie droite exprime l'état unique de la sortie unique du séquent, donc le nom des atomes est celui de ces états. Les formules dans la partie gauche sont distinguées par l'attribut *Order* (= 1, 2...), qui montre le numéro d'ordre des formules dans la partie gauche, tandis que la formule unique dans la partie droite représente soit une sortie avec plusieurs états soit plusieurs sorties de l'histoire. Chaque formule est composée d'au moins trois *éléments* (atome, connecteur, atome). Les éléments sont distingués par leur numéro d'ordre (= 1, 2...) qui exprime leur position dans les formules. Il y a sept types d'éléments :
  - Type = "Open Parenthesis", Name = "(", ParenthesisLevel (= 1, 2, 3…) est le niveau de la parenthèse dans la formule ;
  - Type = "Close Parenthesis", Name = ")", ParenthesisLevel (= 1, 2, 3…) est le niveau de la parenthèse dans la formule ;
  - Type = "Additive Conjunction", Name = "with" ;
  - Type = "Additive Disjunction", Name = "plus";
  - Type = "Multiplicative Conjunction", Name = "times" ;
  - Type = "Linear Implication", Name = "imply", ses attributs *EventActionName* et *EventActionPriorityOrder* expriment le nom et l'ordre de priorité de l'événement/action correspondant dans l'histoire ;
  - Type = "Atom", ces éléments représentent les états ou les entrées de l'histoire modélisée qui sont présents dans la formule, donc leur nom est celui de ces états/entrées.

  Comme chaque formule possède toujours un connecteur principal, elle a toujours une et seulement une règle applicable correspondant à ce connecteur. Ainsi, un séquent peut posséder en même temps beaucoup de règles applicables en fonction de son nombre de formules. On doit choisir une seule règle entre elles en examinant l'ordre de priorité des règles (voir ci-dessus). S'il y a plusieurs règles avec le même ordre de priorité, la règle correspondant à la formule « la plus gauche » est sélectionnée (dans le cas où il y a plusieurs règles ⊸L applicables, la règle correspondant à l'événement/action avec la priorité la plus élevée dans l'histoire est sélectionnée). Par exemple, analysons le séquent A, B, C ⊕ D, (A ⊸ D) & ((A ⊸ E) ⊕ (B ⊸ F)) ⊢ F. La partie gauche (Type = 1) A, B, C ⊕ D, (A ⊸ D) & ((A ⊸ E) ⊕ (B ⊸ F)) a deux atomes disponibles A et B, deux formules C ⊕ D (Ordre = 1) et (A ⊸ D) & ((A ⊸ E) ⊕ (B ⊸ F)) (Ordre = 2). La première formule C ⊕ D possède trois éléments, la

seconde formule (A ⊸ D) & ((A ⊸ E) ⊕ (B ⊸ F)) possède dix-neuf éléments. Dans la seconde formule, il y a quatre parenthèses ouvertes et quatre parenthèses fermées. Le niveau des parenthèses ouvertes est de 1, 1, 2, 2 ; le niveau des parenthèses fermées est de 1, 2, 2, 1 respectivement. La partie droite (Type = 2) n'a qu'un atome F. Comme il y a deux formules C ⊕ D et (A ⊸ D) & ((A ⊸ E) ⊕ (B ⊸ F)) dans la partie gauche du séquent, nous pouvons appliquer deux règles ⊕L et &L. Néanmoins, la priorité de la règle &L est plus élevée (PriorityOrder = 3), par conséquent, elle est la règle unique utilisée pour ce séquent à la première étape de l'algorithme.

- Le graphe des preuves d'un séquent est composé d'une ou plusieurs *branches* en fonction des possibilités de choix offertes au joueur (cette possibilité est exprimée en utilisant le connecteur &) et/ou offertes au système de pilotage (cette possibilité est exprimée en utilisant le connecteur ⊕). Les branches se distinguent par leur numéro d'ordre (= 1, 2...). Chaque branche peut avoir un ou plusieurs étapes d'*itération* qui sont distinguées par leur numéro d'ordre (= 0, 1...). Chaque étape d'itération peut contenir un ou plusieurs *séquents* qui sont les subséquents générés dans le processus de preuve du séquent original. Ces séquents se distinguent par leur numéro d'ordre (= 1, 2...). Chaque étape d'itération correspond à la réécriture d'un des séquents contenus dans l'étape d'itération précédente en appliquant une règle d'inférence de la logique liénaire. Le numéro d'ordre du séquent réécrit dans une étape d'itération est placé dans son attribut *ExaminedSequentOrder* (sa valeur par défaut est de 1). La première étape d'itération (Order = 0) dans la première branche (Order = 1) du graphe des preuves contient le séquent original (Order = 1). Chaque branche dans le graphe des preuves d'un séquent possède deux attributs ayant but de garder la trace du processus de preuve : *ParentBranchOrder* et *ParentIterationOrder*. Ils sont, respectivement, les numéros d'ordre de la branche et de l'étape d'itération dans cette branche à partir des quelles la branche courante est construite (pour la première branche, *ParentBranchOrder* = 0 et *ParentIterationOrder* = 0).
- *ResultAfterApplyingRule* : Ce composant est employé afin d'enregistrer le résultat obtenu après l'application d'une règle d'inférence de la logique linéaire à un séquent à chaque étape d'itération dans le processus de preuve. Ses deux attributs *BranchOrder* et *IterationOrder* sont, respectivement, les numéros d'ordre de la branche et de l'étape d'itération dans cette branche qui contiennent le séquent auquel la rège est appliquée. Un résultat peut posséder une, deux ou plusieurs *solution(s)* qui sont distinguées par leur numéro d'ordre (= 1, 2…). Dépendant de la règle d'inférence appliquée, chaque solution peut comprendre soit un soit deux *séquent(s)*, donc leur numéro d'ordre est de 1, 2 respectivement.
- *AppliedRule* : Ce composant est employé afin de représenter la règle appliquée à un séquent à chaque étape d'itération dans le processus de preuve. Nous utilisons les attributs suivants pour la décrire :
    - FormulaOrder : C'est le numéro d'ordre de la formule dans le séquent à laquelle la règle est appliquée.
    - PartType = 1 (ou = 2) si la formule est dans la partie gauche (droite) du séquent.

- ElementOrder : C'est le numéro d'ordre de l'élément (connecteur) correspondant à la règle dans la formule.
- Name : C'est le nom de la règle correspondant au type du connecteur (Multiplicative Conjunction, Additive Conjunction, Additive Disjunction, Linear Implication).
- UsedSolutionOrder : Cet attribut indique le numéro d'ordre de la solution utilisée dans le résultat obtenu après l'exécution de la règle appliquée au séquent, son objectif est de garder la trace du processus de preuve. Comme présenté en Section III.5.2.2.2 – chapitre III, en fonction de chaque règle appliquée (⊗L, ⊕R, &L, ⊕L, ⊸R, ⊸L, ⊗R), le résultat obtenu peut être composé de : soit une solution comprenant un séquent (⊗L, ⊸R), donc *UsedSolutionOrder* est toujours de 1 ; soit deux solutions dont chacune contient un séquent (⊕R, &L, ⊕L), donc *UsedSolutionOrder* est de soit 1 soit 2 ; soit plus de deux solutions (⊸L, ⊗R) dont chacune comprend deux séquents, donc *UsedSolutionOrder* = 1, 2, 3, 4...

Par exemple, revoyons le séquent A, B, C ⊕ D, (A ⊸ D) **&** ((A ⊸ E) ⊕ (B ⊸ F)) ⊢ F. Sa règle appliquée est &L. Cette règle, utilisée pour la seconde formule dans la partie gauche, correspond au sixième élément (le connecteur &) de la formule. Le résultat obtenu après l'application de la règle est composé de deux solutions dont chacune comprend un séquent :

- Solution 1 : A, B, C ⊕ D, A ⊸ D ⊢ F. Si nous utilisons cette solution comme le résultat obtenu, la règle appliquée est représentée par le segment exprimé sous forme XML suivant :

```
<Sequent>
   <Part Type="1"> … </Part>
   <Part Type="2"> … </Part>
   <AppliedRule Name="Additive Conjunction" FormulaOrder="2"
   ElementOrder="6" PartType="1" UsedSolutionOrder="1"/>
</Sequent>
```

- Solution 2 : A, B, C ⊕ D, (A ⊸ E) ⊕ (B ⊸ F) ⊢ F. Si nous utilisons cette solution comme le résultat obtenu, la règle appliquée est représentée par le segment exprimé sous forme XML ci-dessus (seule la valeur d'un paramètre change : **`UsedSolutionOrder="2"`**).

E.2. Algorithme utilisé par l'outil pour construire le graphe des preuves d'un scénario avec les événements/actions complexes

L'algorithme utilisé par l'outil, afin de construire le graphe des preuves d'un scénario avec les événements/actions complexes, est donné dans la Figure E.4. Il met en œuvre la version complète du calcul des séquents, et emploie le modèle de données présenté ci-dessus. Le parcours, exécuté par l'algorithme, est une recherche en profondeur, qui fait attention à la priorité entre les règles ⊗L, ⊕R, &L, ⊕L, ⊸R, ⊸L, ⊗R, pour, à chaque étape d'itération du processus de preuve d'un séquent, réécrire le séquent en appliquant une de ces règles. La trace des étapes de réécriture constitue le graphe des preuves du séquent.

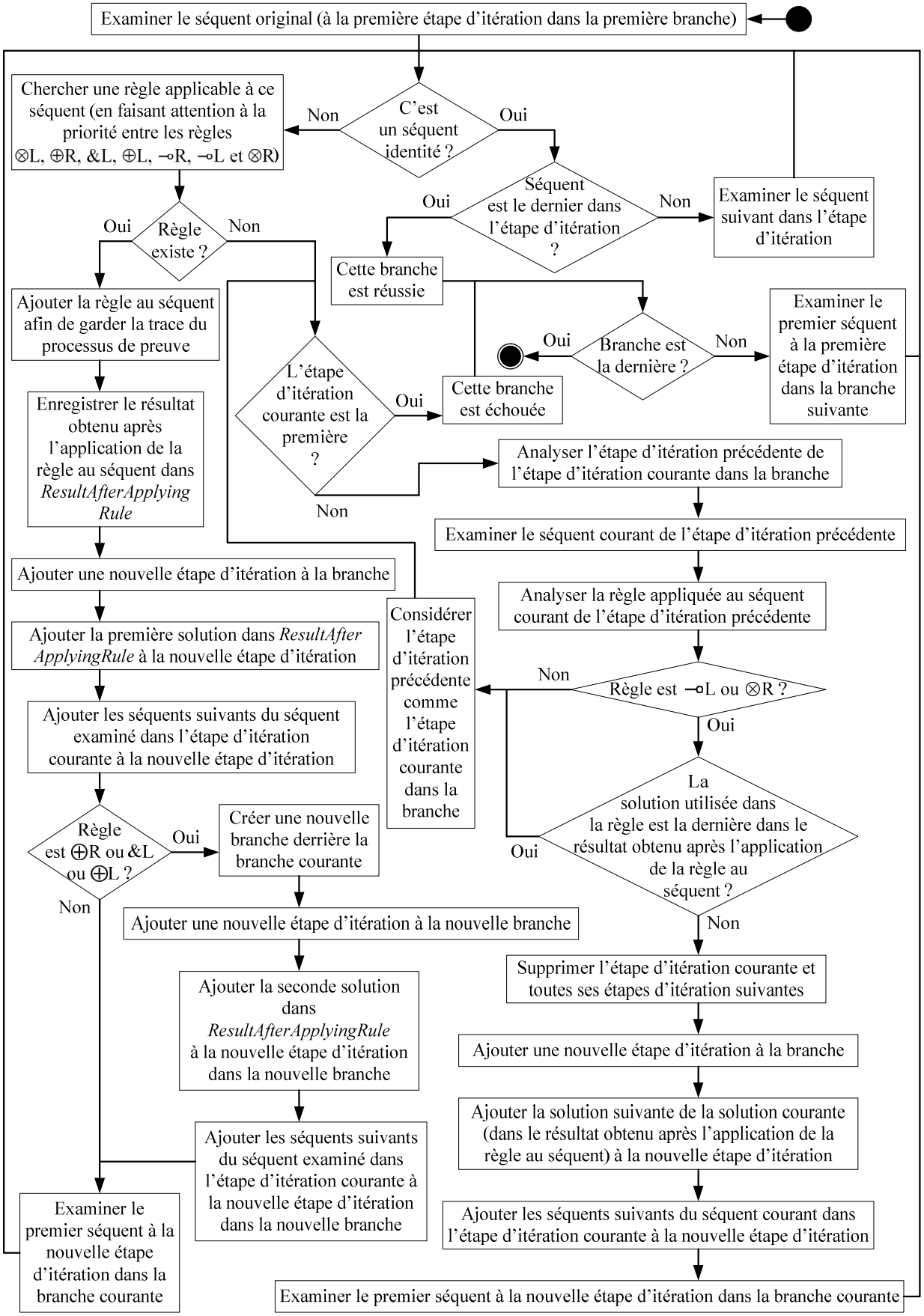


Figure E.4. Algorithme utilisé par l'outil pour construire le graphe des preuves d'un scénario avec les événements/actions complexes.

Suite à la création du graphe des preuves, son analyse automatique donne les informations qualitatives et statistiques sur le scénario modélisé.

E.3. Exemple

Nous examinons ci-dessous un exemple concret comme une illustration de la démarche, mise en œuvre au moyen de l'outil, pour la production d'un scénario de bonne qualité avec les événements/actions complexes.

Supposons que l'utilisateur modélise une histoire comme suit :

- États : A, B, C, D, E, F, G, H dans lesquels A est l'état disponible au moment initial
- Événements/actions
  - EA1 (ordre de priorité = 1) : (B ⊸ C) ⊸ D
  - EA2 (ordre de priorité = 2) : B ⊸ C
  - EA3 (ordre de priorité = 3) : A ⊗ D ⊸ E
  - EA4 (ordre de priorité = 4) : A ⊗ D ⊸ F
- 1 choix décidé par le système de pilotage entre EA3 (son pourcentage est de 40 %) et EA4 (60 %)
- Sortie : E
- Le séquent de logique linéaire représentant le scénario courant de l'histoire est : A, EA1, EA2, EA3 ⊕ EA4 ⊢ E ou A, (B ⊸ C) ⊸ D, B ⊸ C, (A ⊗ D ⊸ E) ⊕ (A ⊗ D ⊸ F) ⊢ E.

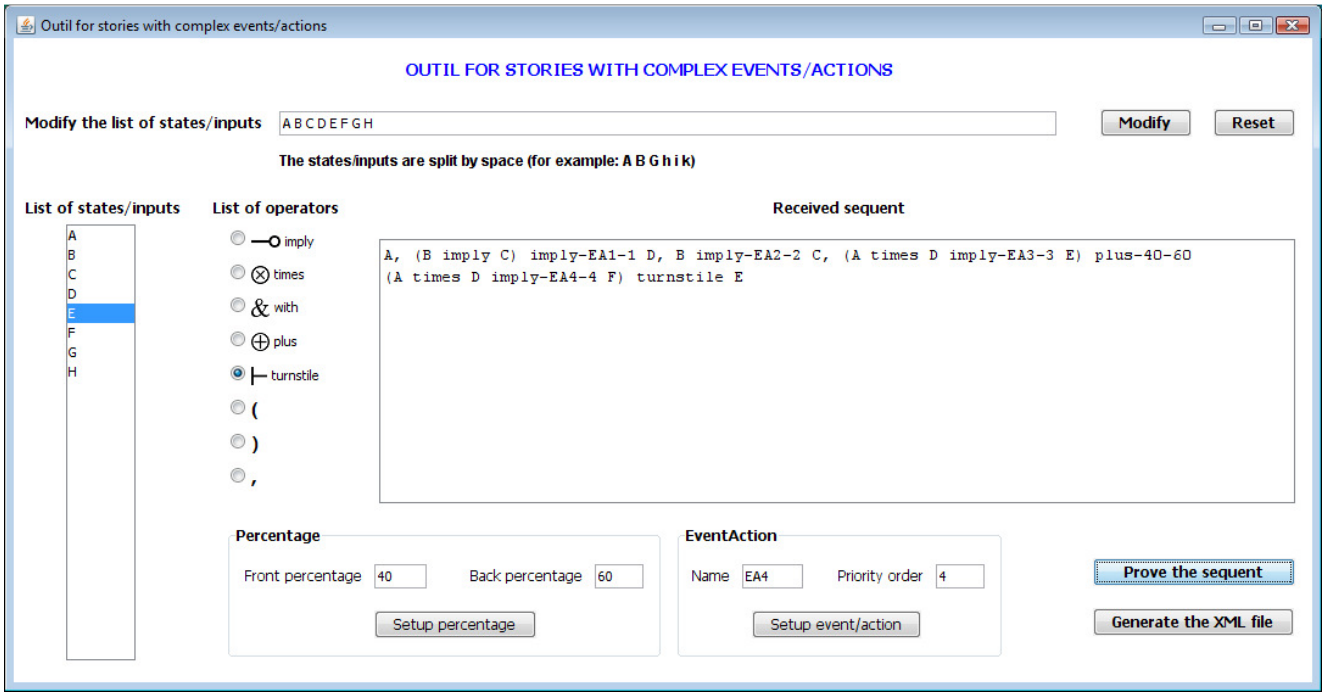


Figure E.5. Rédaction d'un séquent de logique linéaire exprimant un scénario modélisé.

L'utilisateur introduit ce séquent de logique linéaire dans l'outil en rédigeant le séquent au moyen de l'interface utilisateur comme montré dans la Figure E.5, où :

- « imply-EAi-j » signifie que l'événement/action correspondant à cette implication linéaire possède le nom « EAi » et l'ordre de priorité « j » (par exemple, *imply-EA1-1* signifie que l'événement/action correspondant à cette implication linéaire possède le nom *EA1* et l'ordre de priorité *1*) ;
- « plus-i-j » signifie que le pourcentage des possibilités « amont » et « aval » de ce connecteur ⊕ est de « i » et « j » respectivement (par exemple, *plus-40-60* signifie que le pourcentage des possibilités « amont » et « aval » de ce connecteur ⊕ est de 40% et 60% respectivement).

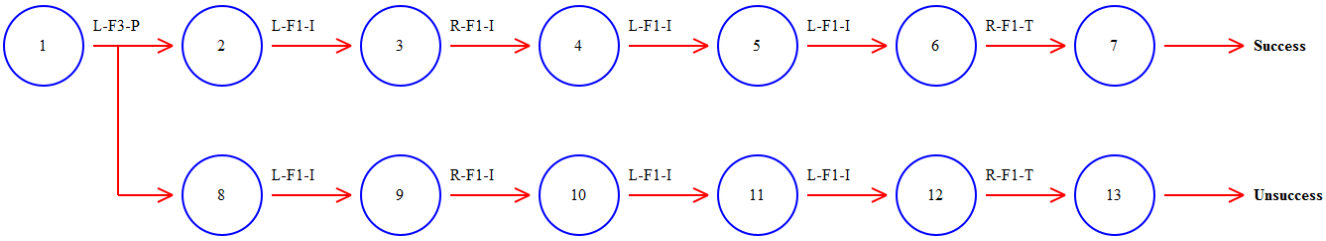


Figure E.6. Graphe des preuves du séquent A, (B ⊸ C) ⊸ D, B ⊸ C, (A ⊗ D ⊸ E) ⊕ (A ⊗ D ⊸ F) ⊢ E.

La preuve du séquent donne les résultats ci-dessous :

- Le graphe des preuves du séquent indiqué dans la Figure E.6 où
  - i (= 1, 2,…, 13) : Liste de séquents à prouver au nœud *i*
    - 1 (séquent original) : A, (B ⊸ C) ⊸ D, B ⊸ C, (A ⊗ D ⊸ E) ⊕ (A ⊗ D ⊸ F) ⊢ E
    - 2 : A, (B ⊸ C) ⊸ D, B ⊸ C, A ⊗ D ⊸ E ⊢ E
    - 3 (2 séquents) : Séquent 1 : B ⊸ C ⊢ B ⊸ C ; Séquent 2 : A, D, A ⊗ D ⊸ E ⊢ E
    - …
    - 7 (3 séquents) : Séquent 1 : A ⊢ A ; Séquent 2 : D ⊢ D ; Séquent 3 : E ⊢ E
    - 8 : A, (B ⊸ C) ⊸ D, B ⊸ C, A ⊗ D ⊸ F ⊢ E
    - …
  - La règle appliquée au séquent examiné à chaque étape d'itération :
    - L-Fi (R-Fi) : La règle est appliquée à la **F**ormule ***i*** dans la partie gauche (droite) du séquent
    - T, W, P, I : Indiquer le connecteur (**T**imes, **W**ith, **P**lus, **I**mply) correspondant à cette règle.

    Par exemple, la règle L-F3-P signifie que l'on applique la règle ⊕L à la formule 3 dans la partie gauche du séquent examiné.
- Les informations qualitatives et statistiques suivantes sur le scénario courant :
  - Il y a 2 états inutilisés (25 %) : G, H (ils n'apparaissent pas dans la partie gauche du séquent) → les situations correspondantes sont inaccessibles
  - Aucune sortie n'est inaccessible (0 %)
  - Il y a 2 branches : 1 réussie (50 %) et 1 échouée (50 %)

- Il y a 1 sortie : E (nombre de branches réussies pour cette sortie : 1, pourcentage des branches réussies pour cette sortie : 100 %).

Grâce aux résultats ci-dessus du processus de preuve, l'utilisateur peut évaluer quelques aspects sur la qualité du scénario courant (notamment il peut vérifier sa satisfaction pour les deux propriétés de narration *Accessibilité* et *Absence de blocage*). Et puis, l'utilisateur peut corriger la modélisation jusqu'à ce que le scénario réponde à son objectif (que le scénario soit valide). Enfin, l'utilisateur fait générer, par l'outil, la représentation XML du scénario validé selon le métamodèle du calcul des séquents. Ce modèle exprimé en XML est ensuite employé comme l'entrée du système de pilotage assurant le contrôle de la gestion du déroulement du jeu selon le scénario valide produit, grâce au mécanisme de déduction automatique du séquent.

Voici ci-dessous la représentation XML du séquent A, (B ⊸ C) ⊸ D, B ⊸ C, (A ⊗ D ⊸ E) ⊕ (A ⊗ D ⊸ F) ⊢ E qui est conforme au métamodèle du calcul des séquents :

```
<Sequent>
    <LeftPart>
        <AvailableState Name="A"/>
        <Formula Order="1">
            <Element Name="(" Order="1" Type="Open Parenthesis" ParenthesisLevel="1"/>
            <Element Name="B" Order="2" Type="Atom"/>
            <Element Name="imply" Order="3" Type="Linear Implication"/>
            <Element Name="C" Order="4" Type="Atom"/>
            <Element Name=")" Order="5" Type="Close Parenthesis" ParenthesisLevel="1"/>
            <Element Name="imply" Order="6" Type="Linear Implication" EventActionName="EA1"
            EventActionPriorityOrder="1"/>
            <Element Name="D" Order="7" Type="Atom"/>
        <Formula/>
        <Formula Order="2">
            …
        <Formula/>
        <Formula Order="3">
            …
            <Element Name="plus" Order="8" Type="Additive Disjunction" FrontPercentage="40"
            BackPercentage="60"/>
            …
        <Formula/>
    <LeftPart/>
    <RightPart>
        <Outcome Name="">
            <State Name="E"/>
        <Outcome/>
    <RightPart/>
<Sequent/>
```

PRINCIPALES PUBLICATIONS SCIENTIFIQUES

Revues internationales

- Dang, K. D., Champagnat, R., Augeraud, M.: A Methodology to Derive a Valid Scenario of an Interactive Storytelling. Journal *ACM Computers in Entertainment* (acceptée en 2012).

- Dang, K. D., Champagnat, R., Augeraud, M.: A Methodology to Validate Interactive Storytelling Scenarios in Linear Logic. In: Pan, Z. *et al.* (*eds*.) Journal *Transactions on Edutainment – Special Issue on Interactive Digital Storytelling*. LNCS, *vol*. 7775, pp. 53--82. Springer, Heidelberg, 2013.

Conférences internationales

- 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, 153-164. Springer, Heidelberg, 2010.

- Dang, K. D., Champagnat, R., Augeraud, M.: A Methodology to Derive a Valid Scenario of an Interactive Storytelling. In *Proceedings of the 8th international conference on Advances in Computer Entertainment technology* – ACE 2011. Lisbon, Portugal, 2011 (cet article a été choisi pour la revue internationale *ACM Computers in Entertainment* ci-dessus en tant qu'un des meilleurs articles d'ACE 2011).

- Dang, K. D., Hoffmann, S., Champagnat, R., Spierling, U.: How Authors Benefit from Linear Logic in the Authoring Process of Interactive Storyworlds. In: Si, M. *et al*. (*eds*.) ICIDS 2011. LNCS, *vol*. 7069, 249-260. Springer, Heidelberg, 2011.

BIBLIOGRAPHIE

[1] Abrusci, V. M., Ruet, P.: Non-commutative logic I: The multiplicative fragment. Annals of Pure and Applied Logic, 101(1): 29-64, 2000.

[2] Aponte, M.-V., Levieux, G., Natkin, S.: Difficulty in Videogames: An Experimental Validation of a Formal Definition. In: Proceedings of the 8th international conference on Advances in Computer Entertainment technology – ACE 2011. Lisbon, Portugal, 2011.

[3] Ashliman, D. L., 2011. Little Red Riding Hood and other tales of Aarne-Thompson-Uther type 333 (page consultée le 14 février 2013), http://www.pitt.edu/~dash/type0333.html#grimm.

[4] Aylett, R.: Narrative in Virtual Environments: Towards Emergent Narrative. Proceedings of the AAAI Fall Symposium, *Technical report FS-99-01, AAAAI Press,* Menlo Park, 83–86, 1999.

[5] Aylett, R. S., Louchart, S., Dias, J., Paiva, A., Vala, M.: FearNot! – An Experiment in Emergent Narrative. In: Panayiotopoulos, T., et al. (Eds.) IVA 2005. LNAI, vol. 3661, 305–316. Springer, Heidelberg, 2005.

[6] Aylett, R., Vala, M., Sequeira, P., Paiva, A.: FearNot! – An Emergent Narrative Approach to Virtual Dramas for Anti-bullying Education. In: Cavazza, M., Donikian, S. (Eds.): ICVS 2007, LNCS 4871, pp. 199–202. Springer, Heidelberg, 2007.

[7] Azevedo, V. C., Pozzer, C. T.: Creating a Director for an Interactive Storytelling System. In: VI Brazilian Symposium on Computer Games and Digital Entertainment, pp. 1-9, ISBN: 85-766-9154-X. Sao Leopoldo, Brazil, 2007.

[8] Balas, D., Brom, C., Abonyi, A., Gemrot, J.: Hierarchical Petri Nets for Story Plots Featuring Virtual Humans. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, 2-9, 2008.

[9] Barras, B., Boutin, S., Cornes, C., Courant, J., Filliatre, J.-C., Giménez, E., Herbelin, H., Huet, G., Muñoz, C., Murthy, C., Parent, C, Paulin-Mohring, C., Saïbi, A., Werner, B.: The Coq proof assistant reference manual: Version 6.1. Technical Report number 0203, Inria (Institut National de Recherche en Informatique et en Automatique), France, 1997.

[10] Bates, J.: Virtual Reality, Art, and Entertainment. Presence: The Journal of Teleoperators and Virtual Environments, 1(1): 133–138, MIT Press, 1992.

[11] Bézivin, J.: Model Driven Engineering: An Emerging Technical Space. In: Lämmel, R., Saraiva, J., Visser, J. (eds.) GTTSE 2005. LNCS, vol. 4143, 36–64. Springer, Heidelberg, 2006.

[12] Bosser, A.-G., Cavazza, M., Champagnat, R.: Linear Logic for Non-Linear Storytelling. In: Proceedings of ECAI 2010 - 19th European Conference on Artificial Intelligence, pp. 713--718, 2010.

[13] Bosser, A.-G., Courtieu, P., Forest, J., Cavazza, M.: Structural Analysis of Narratives with the Coq Proof Assistant. In: Proceedings of ITP11 - Interactive Theorem Proving - Second International Conference, Lecture Notes in Computer Science, Springer, 2011.

[14] Brom, C., Abonyi, A.: Petri Nets for Game Plot. In: Proc. of AISB, Vol. 3. 6–13. AISB press, 2006.
[15] Brom, C., Holan, T., Balas, D., Abonyi, A., Sisler, V., Galambos, L.: Petri Nets for Representing Story Plots in Serious Games. Journal of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour. Vol. 2, No. 1, 2010.
[16] Buche, C.: Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation. Thèse de doctorat présentée devant l'Ecole Nationale d'Ingénieurs de Brest (ENIB) et effectuée au Laboratoire d'Informatique des SYstèmes Complexes – Université de Bretagne Occidentale, France, 2005.
[17] Cavazza, M., Charles, F., Mead, S. J.: Character-Based Interactive Storytelling. IEEE Intelligent Systems, special issue on AI in Interactive Entertainment, vol. 17, no. 4, pages 17–24, 2002.
[18] Cavazza, M., Charles, F., Mead, S. J.: Interacting with Virtual Characters in Interactive Storytelling. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, AAMAS'02, pp. 318-325. Bologna, Italy, 2002.
[19] Cavazza, M., Donikian, S., Christie, M., Spierling, U., Szilas, N., Vorderer, P., Hartmann, T., Klimmt, C., André, E., Champagnat, R., Petta, P., Olivier, P.: The IRIS Network of Excellence: Integrating Research in Interactive Storytelling. In: Spierling, U., Szilas, N. (Eds.) ICIDS 2008. LNCS, vol. 5334, pp. 14--19. Springer, Heidelberg, 2008.
[20] Champagnat, R., Delmas, G., Augeraud, M.: Proposition d'une architecture de pilotage d'un jeu à récit interactif : Gestion de l'exécution adaptative du déroulement d'une application interactive. Journal Européen des Systèmes Automatisés, number 2, 2010.
[21] 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.
[22] Charles, F., Lozano, M., Mead, S. J., Bisquerra, A. F., Cavazza, M.: Planning Formalisms and Authoring in Interactive Storytelling. In: Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment. Darmstadt, Germany, 2003.
[23] Charles, F., Mead, S. J., Cavazza, M.: Character-driven Story Generation in Interactive Storytelling. In: Proceedings of the 7th International Conference on Virtual Systems and MultiMedia, VSMM'01. Berkeley, 2001.
[24] Charles, F., Pizzi, D., Cavazza, M., Vogt, T., André, E.: EmoEmma: Emotional Speech Input for Interactive Storytelling. Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Decker, Sichman, Sierra and Castelfranchi (eds.), Budapest, Hungary, pp. 1381 – 1382, 2009.
[25] Ciarlini, A. E. M., Camanho, M. M., Dória, T. R., Furtado, A., L., Pozzer, C. T., Feijó, B.: Planning and Interaction Levels for TV Storytelling. In: Spierling, U. and Szilas, N. (Eds.): ICIDS 2008, LNCS 5334, pp. 198–209. Springer, Heidelberg, 2008.

[26] Ciarlini, A. E. M., Casanova, M. A., Furtado, A., L., Veloso, P. A. S.: Modeling interactive storytelling genres as application domains. Journal of Intelligent Information Systems, Volume 35, Issue 3, pp. 347-381, 2010.

[27] Ciarlini, A., Furtado, A.: Understanding and Simulating Narratives in the Context of Information Systems. In *Proc. ER'2002 - 21st, International Conference on Conceptual Modelling*, Tampere, Finland, 2002.

[28] Ciarlini, A. E. M., Pozzer, C. T., Furtado, A., L., Feijo, B.: A logic-based tool for interactive generation and dramatization of stories. In *Proceedings of the 2005 ACM SIGCHI conference on Advances in computer entertainment technology*, 133-140, 2005.

[29] Collé, F., Champagnat, R., Prigent, A.: Scenario Analysis Based on Linear Logic. In: Advances in Computer Entertainment Technology – ACE, 2005.

[30] Delmas, G.: Pilotage de récits interactifs et mise en œuvre de formes narratives dans le contexte du jeu vidéo. Thèse de doctorat présentée devant l'école doctorale SPI&A et effectuée au Laboratoire Informatique, Image et Interaction – Université de La Rochelle, France, 2009.

[31] Delmas, G., Champagnat, R., Augeraud, M.: Narration dans les jeux vidéo : apport des jeux de rôle. Conference on Hypertext, Hypermedia, Products, Tools, Methods (H2PTM'09), pp. 227-238. Paris, France, 2009.

[32] Delmas, G., Champagnat, R., Augeraud, M.: Plot Control for Emergent Narrative: a Case Study on Tetris. International Journal for Intelligent Games and Simulation, Volume 5, number 1, pages 29-34, 2008.

[33] Dixon, L., Smaill, A., Bundy, A.: Verified Planning by Deductive Synthesis in Intuitionistic Linear Logic. ICAPS 2009 Workshop on Verification and Validation of Planning and Scheduling Systems, 2009.

[34] Dixon, L., Smaill, A., Tsang, T.: Plans, Actions and Dialogues Using Linear Logic. Journal of Logic, Language and Information, Volume 18, Number 2, 251-289, 2009.

[35] Donikian, S., Portugal, J.-N.: Writing Interactive Fiction Scenarii with DraMachina. In: Göbel, S. et al. (Eds.) TIDSE 2004. LNCS 3105, pp. 101-112. Springer, Heidelberg, 2004.

[36] Dupire, J., Labat, J.-M., Natkin, S.: Du jeu sérieux au jeu utile. In: Dupire, J., Labat, J.-M., Natkin, S. (Eds.) Revue d'Intelligence Artificielle – Jeu sérieux, révolution pédagogique ou effet de mode ?, vol. 25/2, pp. 143-146. Lavoisier, 2011.

[37] Dupire, J., Labat, J.-M., Natkin, S. (Eds.): Revue d'Intelligence Artificielle – Jeu sérieux, révolution pédagogique ou effet de mode ?, vol. 25/2. Hermes Lavoisier, 2011.

[38] Eitelman, S., Magerko, B., Holt, L. S., Stensrud, B.: Questions and Issues in Developing a Modular Narrative Learning Environment. In: Proceedings of the 2007 Artificial Intelligence in Education (AIED) Conference, Workshop on Narrative Learning Environments (NLE). Marina del Rey, California, USA, 2007.

[39] Fikes, R. E., Nilsson, N. J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, Vol. 2, No. 3-4, 189-208, 1971.

[40] Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL – The Planning Domain Definition Language. Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[41] Girard, J.-Y.: Linear Logic. Theoretical Computer Science 50(1), 1–101, 1987.

[42] Girard, J.-Y.: Linear Logic: its syntax and semantics. In: Girard, J.-Y., Lafont, Y., Regnier, L. (eds.) Advances in Linear Logic, London Mathematical Society Lecture Notes Series 222, 1–42. Cambridge University Press, 1995.

[43] Girault, F.: Formalisation en logique linéaire du fonctionnement des réseaux de Petri. Thèse préparée au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, Université de Toulouse III, France, 1997.

[44] Glassner, A.: Interactive Storytelling: Techniques for 21st Century Fiction. A K Peters, Ltd. Natick, Massachusetts, USA, 2004.

[45] Göbel, S., Salvatore, L., Konrad, R. A., Mehm, F.: StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories. In: Spierling, U., Szilas, N. (Eds.) ICIDS 2008. LNCS, vol. 5334, pp. 325-328. Springer, Heidelberg, 2008.

[46] Grandbastien, M., Labat, J.-M. (Eds.): Environnements Informatiques pour l'Apprentissage Humain. Hermes Lavoisier, 2006.

[47] Habert, L., Notin, J.-M., Galmiche, D.: LINK: A Proof Environment Based on Proof Nets. In: Egly, U., Fermuller, C.G. (eds.) TABLEAUX 2002. LNAI 2381, pp. 330-334. Springer, Heidelberg, 2002.

[48] Hall, L., Vala, M., Hall, M., Webster, M., Woods, S., Gordon, A., Aylett, R.: FearNot's Appearance: Reflecting Children's Expectations and Perspectives. In: Gratch, J. et al. (Eds.): IVA 2006, LNAI 4133, pp. 407–419. Springer, Heidelberg, 2006.

[49] Hall, L., Woods, S., Aylett, R.: FearNot! Involving Children in the Design of a Virtual Learning Environment. In: International Journal of Artificial Intelligence in Education, Volume 16, Issue 4, pp. 327-351, 2006.

[50] Harland, J., Pym, D., Winikoff, M.: Programming in Lygon: An Overview. In: Proceedings of the Fifth International Conference on Algebraic Methodology and Software Technology (AMAST), pp. 391-405, 1996.

[51] Hodas, J. S.: Lolli: An Extension of λProlog with Linear Logic Context Management. In: Proceedings of the 1992 Workshop on the λProlog Programming Language. Philadelphia, USA, 1992.

[52] 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.

[53] Iurgel, I.: From Another Point of View: Art-E-Fact. In: Göbel, S. et al. (Eds.) TIDSE 2004. LNCS 3105, pp. 26–35. Springer, Heidelberg, 2004.

[54] Juul, J.: A Clash Between Game and Narrative. In: Digital Arts and Culture Conference. Bergen, Norway, 1998.

[55] Kanovich, M. I.: Linear logic as a logic of computations. Annals of Pure and Applied Logic, 67(1-3):183–212, 1994.

[56] Kim, J., Blythe, J.: Supporting plan authoring and analysis. In: Proceedings of the 8th international conference on Intelligent user interfaces, pp. 109-116, 2003.

[57] Kim, J., Gil, Y.: Knowledge Analysis on Process Models. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001). Seattle, Washington, USA, 2001.

[58] Kim, J., Gil, Y., Spraragen, M.: Principles for Interactive Acquisition and Validation of Workflows. Journal of Experimental and Theoretical Artificial Intelligence, 2009.
[59] Kriegel, M., Aylett, R.: Crowd-Sourced AI Authoring with ENIGMA. In: Aylett, R. et al. (Eds.) ICIDS 2010. LNCS, vol. 6432, pp. 275–278. Springer, Heidelberg, 2010.
[60] 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.
[61] Küngas, P.: Linear Logic programming for AI planning. Master thesis, Institute of Cybernetics at Tallinn Technical Univesity, 2002.
[62] 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.
[63] Magerko, B. S.: Player Modeling in the Interactive Drama Architecture. PhD Thesis, Computer Science and Engineering, University of Michigan, USA, 2006.
[64] Magerko, B.: Story Representation and Interactive Drama. Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, 87–92, USA, 2005.
[65] Magerko, B., Laird, J.: Building an interactive drama architecture. In: Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03), 2003.
[66] Magerko, B., Laird, J. E.: Mediating the Tension between Plot and Interaction. AAAI Workshop Series: Challenges in Game Artificial Intelligence. San Jose, 2004.
[67] Mateas, M.: Interactive Drama, Art, and Artificial Intelligence. PhD Thesis, Technical Report CMU-CS-02-206, School of Computer Science, Computer Science Department, Carnegie Mellon University, Pittsburgh, USA, 2002.
[68] Mateas, M., Stern, A.: Architecture, Authorial Idioms and Early Observations of the Interactive Drama *Façade*. Technical report CMU-CS-02-198, School of Computer Science, Carnegie Mellon University, 2002.
[69] Mateas, M., Stern, A.: Façade: An Experiment in Building a Fully-Realized Interactive Drama. Game Developers Conference, Game Design track, 2003.
[70] Mateas, M., Stern, A.: Integrating Plot, Character and Natural Language Processing in the Interactive Drama *Façade*. In: 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE '03). Darmstadt, Germany, 2003.
[71] 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.
[72] Murata, T. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, volume 77, number 4, pp. 541–580, 1989.
[73] Natkin, S.: Du ludo-éducatif aux jeux vidéo éducatifs. In: Journal *Les Dossiers de l'Ingénierie Educative*, numéro 65, pp. 12-15, 2009.
[74] Natkin, S., Véga, L., Grünvogel, S. M.: A new methodology for spatiotemporal game design. In: Proceedings of the 5th Game-On International Conference on Computer

Games: Artificial Intelligence, Design and Education, CGAIDE'04, pp. 109-113. Reading, UK, 2004.
[75] Natkin, S., Yan, C., Jumpertz, S., Marquet, B.: Creating Multiplayer Ubiquitous Games using an adaptive narration model based on a user's model. In: Proceedings of DiGRA'07 conference, pp. 431-439. Tokyo, Japan, 2007.
[76] Paiva, A., Andersson, G., Höök, K., Mourão, D., Costa, M., Martinho, C.: SenToy in FantasyA: Designing an Affective Sympathetic Interface to a Computer Game. Journal Personal and Ubiquitous Computing, volume 6, number 5-6, pp. 378–389, 2002.
[77] Perreira Da Silva, M.: Modèle computationnel d'attention pour la vision adaptative. Thèse de doctorat présentée devant l'école doctorale S2I et effectuée au Laboratoire Informatique, Image et Interaction – Université de La Rochelle, France, 2010.
[78] 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.
[79] Porello, D., Endriss, U.: Modeling Multilateral Negotiation in Linear Logic. In: Proceedings of the 19th European Conference on Artificial Intelligence, 2010.
[80] Prigent, A., Champagnat, R., Estraillier, P.: Driving stories, benefits of properties analysis. 7th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games (CGAMES). Angoulême, France, 2005.
[81] Prigent, A., Champagnat, R., Estraillier, P.: Exécution adaptative de trame narrative. Chapitre 9 du livre Intelligence artificielle et jeux, Hermes Sciences, 2007.
[82] Propp, V.: Morphology of the folktale. Austin: University of Texas Press, 1968.
[83] Riedl, M., Saretto, C. J., Young, R. M.: Managing Interaction Between Users and Agents in a Multi-agent Storytelling Environment. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, AAMAS '03, pp. 741-748. Melbourne, Australia, 2003.
[84] Sanchez, E., Ney, M., Labat, J.-M.: Jeux sérieux et pédagogie universitaire : de la conception à l'évaluation des apprentissages. In: Revue internationale des technologies en pédagogie universitaire, 8(1-2), pp. 48-57, 2011.
[85] Si, M., Marsella, S. C., Pynadath, D. V.: Directorial Control in a Decision-Theoretic Framework for Interactive Narrative. In: Iurgel, I.A., Zagalo, N., Petta, P. (eds.) ICIDS 2009. LNCS, vol. 5915, 221–233. Springer, Heidelberg, 2009.
[86] Si, M., Marsella, S. C., Pynadath, D. V.: Thespian: An Architecture for Interactive Pedagogical Drama. International Conference on Artificial Intelligence in Education. Amsterdam, 2005.
[87] Si, M., Marsella, S. C., Pynadath, D. V.: Thespian: Modeling Socially Normative Behavior in a Decision-Theoretic Framework. In: Gratch, J., Young, M., Aylett, R.S., Ballin, D., Olivier, P. (eds.) IVA 2006. LNCS (LNAI), vol. 4133, pp. 369–382. Springer, Heidelberg, 2006.
[88] Si, M., Marsella, S. C., Pynadath, D. V.: Thespian: Using Multi-Agent Fitting to Craft Interactive Drama. International Conference on Autonomous Agents and Multi Agent Systems, AAMAS'05, pp. 21-28. Utrecht, Netherlands, 2005.

[89] Silva, A., Raimundo, G., Paiva, A.: Tell Me That Bit Again… Bringing Interactivity to a Virtual Storyteller. In: Balet, O., et al. (eds.) ICVS 2003. LNCS, vol. 2897, 146–154. Springer, Heidelberg, 2003.

[90] Silva, A., Vala, M., Paiva, A.: Papous: The Virtual Storyteller. Intelligent Virtual Agents - IVA 2001. LNCS, vol. 2190, 171-180. Madrid, Spain, 2001.

[91] Silva, A., Vala, M., Paiva, A.: The Storyteller: Building a Synthetic Character That Tells Stories. Proceedings of the workshop on Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achive Contextual Embodied Agents, at The Fifth International Conference on Autonomous Agents, pp. 53–58. Montreal, Canada, 2001.

[92] Szilas, N.: IDtension: a narrative engine for Interactive Drama. In: Gobel, S., et al. (Eds.) TIDSE 2003, pp. 187–203. Darmstadt, Germany, 2003.

[93] Szilas, N.: IDtension – Highly Interactive Drama. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2008). Stanford, California, USA, 2008.

[94] Szilas, N.: Interactive drama on computer: beyond linear narrative. AAAI Fall Symposium on Narrative Intelligence, Technical report FS-99-01, pp. 150-156, 1999.

[95] Szilas, N.: The Mutiny: an Interactive Drama on IDtension. In: Proceedings of the 3rd ACM International Conference on Digital Interactive Media in Entertainment and Arts (DIMEA 2008), pp. 539–540, Athens, Greece, 2008.

[96] Tamura, N.: User's Guide of a Linear Logic Theorem Prover (llprover). Department of Computer and Systems Engineering, Faculty of Engineering, Kobe University, Japan, 1998.

[97] 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.

[98] Weiss, S., Müller, W., Spierling, U., Steimle, F.: Scenejo – An Interactive Storytelling Platform. In: Subsol, G. (Eds.) VS 2005. LNCS, vol. 3805, pp. 77-80. Springer, Heidelberg, 2005.

[99] Wong, W. L., Shen, C., Nocera, L., Carriazo, E., Tang, F., Bugga, S., Narayanan, H., Wang, H., Ritterfeld, U.: Serious Video Game Effectiveness. ACE, Salzburg, Austria, 2007.

[100] Young, R. M.: An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment. In: Artificial Intelligence and Interactive Entertainment I, AAAI Technical Report SS-01-02, AAAI Press, pp. 77-81, 2001.

[101] Young, R. M., Riedl, M.: Towards an Architecture for Intelligent Control of Narrative in Interactive Virtual Worlds. In: IUI, 310–312. Miami, Florida, USA, 2003.

[102] 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.

[103] A Linear Logic Prover (llprover) (page consultée le 14 février 2013), http://bach.istc.kobe-u.ac.jp/llprover.

[104] BPMN Information Home (page consultée le 14 février 2013), http://www.bpmn.org.

[105] Eclipse Modeling Framework Project (page consultée le 14 février 2013), http://www.eclipse.org/modeling/emf.
[106] Graphical Modeling Framework (page consultée le 14 février 2013), http://www.eclipse.org/modeling/gmp.
[107] Little Red Riding Hood Workshop: The Authoring Process in Interactive Storytelling (page consultée le 14 février 2013), http://redcap.interactive-storytelling.de/authoring-tools/thespian.
[108] Lolli: A Linear Logic Programming Language (page consultée le 14 février 2013), http://www.lix.polytechnique.fr/~dale/lolli.
[109] Lygon: Logic Programming with Linear Logic (page consultée le 14 février 2013), http://www.cs.rmit.edu.au/lygon.
[110] Neverwinter Nights 2 (page consultée le 14 février 2013), http://nwn2.com/UK/index.php.
[111] Neverwinter Nights Extender (page consultée le 14 février 2013), http://www.nwnx.org/index.php?id=nwnx4_about.
[112] OGRE – Open Source 3D Graphics Engine (page consultée le 14 février 2013), http://www.ogre3d.org.
[113] OMG's MetaObject Facility (page consultée le 14 février 2013), http://www.omg.org/mof.
[114] Unreal Tournament (page consultée le 14 février 2013), http://www.unrealtournament.com/uk/index.html.