Astus, une plateforme pour créer et étudier les systèmes tutoriels intelligents « par traçage de modèle »

par

Jean-François Lebeau

thèse présentée au Département d'informatique en vue de l'obtention du grade de

Docteur ès sciences (Ph. D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, décembre 2015

Le 23 décembre 2015 le jury a accepté la thèse de Monsieur Jean-François Lebeau dans sa version finale.

Membres du jury

Professeur André Mayers
Directeur de recherche
Département d'informatique

Professeur Yves Bouchard
Membre interne
Département de philosophie et d'éthique appliquée

Professeur Michel C. Desmarais
Membre externe
Département de génie informatique et génie logiciel, École polytechnique Montréal

Professeur Marc Frappier
Président rapporteur
Département d'informatique

Sommaire

Cette thèse s'intéresse aux systèmes tutoriels intelligents (STI), un type d'environnement informatique pour l'apprentissage humain (EIAH) qui se distingue des autres (p. ex. les exerciseurs et les hypermédias éducatifs) en offrant un mécanisme d'évaluation plus sophistiqué. Parmi les différentes familles de STI, ce sont les STI « par traçage de modèle » (MTT) qui ont le plus fait leurs preuves.

Les MTT sont critiqués, premièrement parce qu'ils évaluent l'apprenant de façon serrée (c.-à-d. qui positionne l'action de l'apprenant par rapport à une ou plusieurs méthodes pour effectuer la tâche), ce qui n'est possible que pour des tâches bien définies. Par conséquent, on leur reproche d'encourager un apprentissage superficiel. Deuxièmement, parce que les efforts de création qu'ils requièrent sont jugés prohibitifs, ce qui a mené à l'apparition d'autres familles de STI, comme les STI « par contraintes » et les STI « par traçage d'exemples » et ceux basés sur l'apprentissage automatique.

Par cette thèse, nous voulons contribuer à renouveler l'intérêt pour les MTT en améliorant le rapport entre les efforts de création et l'efficacité potentielle des interventions, et en établissant plus clairement leur rôle pédagogique.

Pour ce faire, nous proposons la plateforme Astus qui permet d'explorer l'espace qui existe entre les MTT créés avec les plateformes existantes, et des MTT dédiés ayant recours à des connaissances didactiques sophistiquées (p. ex. des dialogues) qui exigent des efforts de création encore plus importants.

La plateforme Astus se distingue des plateformes existantes parce qu'elle génère des interventions plutôt que de recourir à des interventions prémâchées et qu'elle supporte les

tâches s'effectuant dans des environnements qui ont une dimension physique. La génération des interventions dépend :

- d'un modèle de la tâche qui s'inscrit dans le paradigme du tuteur, c'est-à-dire qui représente une abstraction et une généralisation des instructions d'un tuteur humain;
- d'un modèle de l'UI qui permet des interventions riches comme une démonstration (c.-à-d. déplacements du pointeur et simulation des clics et des saisies);
- de langages dédiés et d'outils qui réduisent les efforts de création des auteurs;
- de mécanismes d'extension qui permettent d'adapter la génération en fonction d'une stratégie pédagogique particulière.

Le paradigme du tuteur, parce qu'il favorise une communication transparente entre le système et l'apprenant, met en évidence les avantages et les désavantages de l'approche pédagogique des MTT, essentiellement une évaluation précise (c.-à-d. qui permet de produire des indices sur la prochaine étape et des rétroactions sur les erreurs), mais serrée. En s'inscrivant explicitement le paradigme du tuteur, entre autres en évitant de tirer profit de la nature de domaines particuliers ou de propriétés de tâches particulières pour assouplir l'évaluation, la plateforme Astus se démarque plus nettement des autres familles de STI que les autres MTT. Par conséquent, elle établit plus clairement le rôle pédagogique des MTT.

Cinq expérimentations (menées par Luc Paquette) à petite échelle ont été réalisées auprès d'étudiants au baccalauréat au département d'informatique (un laboratoire pour la manipulation d'arbres binaires de recherche et un pour la conversion de nombres en virgule flottante). Ces expérimentations indiquent que les interventions générées sont efficaces. Au-delà de ces résultats, c'est le processus entourant ces expérimentations, parce qu'il est comparable au processus des chercheurs potentiellement intéressés par la plateforme Astus, qui montre que la version présentée dans cette thèse est plus qu'un prototype et qu'elle peut être utilisée à l'interne dans un contexte réel.

Remerciements

Mes premiers remerciements vont à André Mayers, mon directeur de recherche, pour sa passion pour la recherche, son soutien moral et financier et toutes les discussions fort enrichissantes que nous avons eus durant ce projet.

Merci à mes collègues du laboratoire ASTUS pour les discussions qui ont orienté mon travail. En particulier, je tiens à souligner la contribution importante de Luc Paquette.

Merci à Claude Déry du département de biologie, mon collaborateur des premières heures, pour le projet « génie génétique ».

Merci à Sylvie Jetté et Mélanie Marceau de l'école des sciences infirmières pour leur collaboration au projet « soins critiques ».

Merci à Mario Paquet, analyste au département d'informatique, pour l'attention qu'il généreusement accordé aux besoins du laboratoire.

Un merci spécial à Mélanie Bergeron pour avoir bien voulu croire en moi pendant toutes ces années et pour avoir revu et corrigé cette thèse.

Finalement, je présente mes remerciements aux membres du jury, Yves Bouchard, Michel Desmarais et Marc Frappier pour avoir accepté de lire et de commenter cette thèse.

Table des matières

Liste des abréviations

ABR	Arbre binaire de recherche
CBT	« *Constraint-Based Tutors* » (STI « par contraintes »)
CRUD	« *Create, Read, Update and Delete* »
CT	Cognitive Tutors
DAG	« *directed acyclic graph* »
DL	« *description logics* » (logiques de description)
DSL	« *Domain Specific Language* » (langages dédiés)
EIAH	Environnement informatique pour l'apprentissage humain
ETT	« *Example-Tracing Tutors* » (STI « par traçage d'exemples »)
FOL	« *First-order Logic* » (logique du premier ordre)
GAT	Génération automatique de textes
GUI	« *Graphical User Interface* » (interface graphique)
IA	Intelligence artificielle
IDE	« *Integrated Development Environment* » (environnement de développement)
HTN	« *Hierarchical Task Network »* (réseaux de tâche hiérarchique)
KB	« *Knowledge Base* » (base de connaissance)

KT	« *Knowledge Tracing* »
LHS	« *Left hand side* » (partie de gauche d'une règle de production)
MGC	Module de gestion des connaissances
MVC	Modèle-vue-contrôleur (« *Model-View-Controller* »)
MTT	« *Model-Tracing Tutors* » (STI « par traçage de modèle »)
PC	Procédure complexe
PP	Procédure primitive
RHS	« *Right hand side* » (partie de droite d'une règle de production)
SABC	Système à base de connaissance
SAI	« *Selection-Action-Input* »
SDK	« *Software Development Kit* »
ST	Slide Tutor
STI	Système tutoriel intelligent
TAL	Traitement automatique du langage naturel
TDK	« *Tutor Development Kit* »
UI	« *User Interface* » (interface utilisateur)
UV	Usager virtuel
WM(E)	« *Working Memory (Element)* »

Liste des tableaux

Liste des figures

Introduction

Contexte

Les systèmes tutoriels intelligents [1] (STI) [233] constituent un type d'environnement informatique pour l'apprentissage humain[2] (EIAH) [275], au même titre que les exerciseurs, les micromondes, les hypermédias éducatifs ou encore les outils cognitifs [250]. Puisque cette thèse s'intéresse tout particulièrement aux STI, nous positionnons ces derniers par rapport aux autres types d'EIAH afin d'éclairer le lecteur. Bien que les EIAH soient généralement comparés en fonction des théories de l'apprentissage et de la pédagogie sur lesquels ils sont basés, nous les comparons en fonction de leurs fonctionnalités afin de mettre en évidence la problématique informatique. Avant de procéder à cette comparaison, nous introduisons d'abord la notion de représentation des connaissances.

La représentation des connaissances

En psychologie, en éducation et en intelligence artificielle[3] (IA), on sépare généralement les connaissances en deux catégories, soit les connaissances déclaratives et les connaissances procédurales [9, 103, 186, 213]. Dans cette thèse, nous utilisons ces notions selon la perspective de l'IA à moins d'une indication contraire. Dans cette perspective, les connaissances correspondent aux croyances d'un agent (par extension, à celles d'un humain). Les connaissances déclaratives décrivent les tâches, les domaines dont ces dernières sont tirées et les environnements dans lesquels les tâches s'effectuent. Les connaissances procédurales manipulent les connaissances déclaratives afin de produire des actions, dites internes, qui agissent sur les connaissances déclaratives ou des actions, dites externes, qui

[1] « *Intelligent Tutoring Systems* »
[2] Dans la littérature, on retrouve le concept similaire « *Intelligent Learning Environment* »
[3] Dans [216] et accessoirement dans [39], on retrouve les notions d'IA auxquelles cette thèse fait référence.

agissent sur l'environnement, qu'il soit réel ou simulé. Dans le contexte des EIAH, on s'intéresse aux habiletés cognitives, c'est-à-dire les connaissances procédurales qui ne requièrent pas d'habiletés motrices qui ne sont pas déjà acquises. Pour faire une analogie avec la programmation, on peut dire que les connaissances procédurales correspondent à des algorithmes (ou encore des euristiques, des inférences, etc.) et que les connaissances déclaratives correspondent aux données que ces algorithmes manipulent (entrées, sorties et résultats intermédiaires).

Les connaissances, déclaratives et procédurales, qu'elles soient propres à une tâche, à un domaine ou bien générales (s'appliquant dans plusieurs domaines) se représentent aisément ou non. En effet, bien que des approches de représentation particulières soient adaptées pour représenter certaines sous-catégories de connaissances, la représentation des connaissances, au sens large, est un problème IA-complet. On retrouve des connaissances représentées sous plusieurs formes : les langages naturels, les langages semi-formels (vocabulaires contrôlés, pseudocodes, etc.) et les langages formels (logiques, mathématiques, langages de programmation, automates, etc.). En particulier, parmi les outils de l'IA, on retrouve les systèmes à base de connaissance (SABC) dont le développement est étroitement lié à celui des STI. Les SABC représentent les connaissances de façon plus explicite que les langages de programmation usuels, afin de pouvoir réifier l'exécution des connaissances procédurales sous forme de connaissances déclaratives (p. ex. un système expert qui fournit les inférences qui l'ont mené à prendre une décision).

Les EIAH

Pour illustrer la comparaison des types d'EIAH, nous utilisons les connaissances du domaine de l'algèbre, en particulier celles nécessaires pour effectuer la tâche de résoudre une équation. Dans le cas d'un **exerciseur**[4], une équation à résoudre est affichée à l'apprenant et le système lui demande de choisir une réécriture particulière ou une solution (en incluant des choix

[4] Dans la littérature, on retrouve aussi les termes « *Computer-Assisted Instruction* » et « *Computer-Based Training* » [8] ou encore « enseignement assisté par ordinateur » [45].

incorrects). L'évaluation se fait grâce à un diagramme d'états prédéfini pour chacune des équations. Le diagramme est annoté pour permettre une évaluation qui doit au moins indiquer si le choix de l'apprenant est correct ou non (par exemple, un message qui explique pourquoi la réécriture est incorrecte). Par ailleurs, avant l'avènement de la micro-informatique, les exerciseurs ont eu comme précurseurs des « machines à enseigner » qui offraient un comportement essentiellement similaire [92].

Dans un **micromonde** [126], l'apprenant est placé dans un environnement où il crée des équations et explore leurs propriétés algébriques. Lorsque l'apprenant entre une équation ou applique une réécriture, le système vérifie la relation d'égalité. Par rapport aux exerciseurs, les micromondes offrent aux apprenants une grande liberté d'action et par le fait même, supposent de ces derniers un haut degré d'autonomie. En effet, même s'ils peuvent faire des suggestions (par exemple, une équation à résoudre), ils évitent autant que possible de contraindre l'apprenant. Il ne faut pas confondre les micromondes avec les simulateurs traditionnels : les premiers, contrairement aux seconds, ne supposent pas un apprentissage préalable des connaissances du domaine avant leur utilisation. En ce sens, les jeux sérieux [274] sont plus près des micromondes.

Au sein d'un **hypermédia éducatif** [46], l'apprenant ne résout pas une équation, il navigue au travers de documents contenant du texte, des figures, des animations et autres médias. On y retrouve, par exemple, la définition du concept d'équation, une explication du principe de la symétrie de l'égalité, des instructions pour résoudre une forme particulière d'équations et des exemples de résolutions. Ce type de systèmes n'inclut pas de mécanisme d'évaluation, par contre il s'adapte en fonction de l'historique de navigation de l'apprenant et de son profil.

Dans un environnement d'apprentissage doté d'**outils cognitifs** [112, 276, 277], les connaissances sont situées dans un contexte qui est le plus près possible de la réalité. Un tel système propose donc un problème qui se résout en trouvant la solution d'une équation, par exemple une équation du mouvement en mécanique classique nécessaire à la réalisation d'un projet d'un cours de science. Dans ce système, c'est l'écriture de l'équation de façon conceptuelle, puis son application à la situation et finalement l'interprétation de sa solution

qui sont au premier plan. Par rapport à l'évaluation, le système agit comme un accompagnateur, il suggère des mécanismes d'autoévaluation à l'apprenant, par exemple, évaluer sa solution en utilisant un logiciel de calcul symbolique comme Mathematica[5] ou bien comparer ses résultats (l'équation, la solution et son interprétation) à ceux de ses pairs ou encore à ceux fournis par le corps enseignant. Comme les hypermédias éducatifs, ces environnements adaptent leurs problèmes en fonction de l'apprenant.

Le comportement des **STI** est à la base un hybride de celui des exerciseurs et celui des micromondes[6], c'est-à-dire qu'un STI guide [154] l'apprenant dans un environnement généralement similaire à celui d'un micromonde. Le système choisit l'équation initiale, puis il demande à l'apprenant d'appliquer la réécriture de son choix. Le système réagit alors en fonction de son évaluation du choix de l'apprenant, par exemple en demandant à ce dernier d'appliquer une autre réécriture même si celle qu'il avait appliquée est valide[7]. Le système aide aussi l'apprenant, directement ou à la demande de ce dernier, à choisir ou à appliquer une réécriture. Le système offre également une évaluation précise lorsque la réécriture est incorrecte, par exemple, en disant à l'apprenant qu'il n'a pas appliqué sa réécriture uniformément. Comme les autres EIAH, les STI ont des mécanismes pour adapter leurs interventions à l'apprenant.

Dans les systèmes réels, les différents comportements des EIAH se chevauchent. On trouve par exemple des STI qui tentent d'intégrer des fonctionnalités propres aux micromondes ou aux outils cognitifs. Par exemple, Aplusix[8] (algèbre) s'est d'abord présenté comme un STI [179] puis comme un micromonde [180]. On trouve aussi des hypermédias ou des exerciseurs (ALEKS[9] et ASSISTments[10] [106] par exemple) auxquels on ajoute une partie du comportement d'un STI (la sélection de tâches). De plus, plusieurs combinaisons sont

[5] http://www.wolfram.com/mathematica/
[6] D'ailleurs, dans la littérature qui précède [280] on parle d'« enseignement intelligemment assisté par ordinateur » [45] ou de « *Intelligent Computer-Assisted Instruction* » [209].
[7] C'est-à-dire qu'elle préserve la relation d'égalité et n'élimine pas la variable.
[8] http://www.aplusix.com/
[9] http://www.aleks.com/
[10] http://www.assistments.org/

possibles[11], par exemple : un EIAH pour l'apprentissage en ligne formée d'un hypermédia et d'exerciseurs; un hypermédia qui est utilisé comme aide en ligne au sein d'un exerciseur ou d'un STI; un outil cognitif qui fait appel à un micromonde ou encore un STI pour aider l'apprenant à évaluer des résultats intermédiaires (ActiveMath[12] par exemple).

Les STI

Les premiers STI, apparus au courant des années 70, étaient avant tout un domaine d'application pour des travaux de recherche en psychologie cognitive et en IA [18], en particulier sur les SABC (p. ex. les systèmes de production, les réseaux bayésiens, les réseaux sémantiques et les réseaux de tâche hiérarchique[13] [HTN]). L'intérêt pour les STI en tant que tels a été suscité par des travaux de recherche sur le tutorat. Ceux-ci montraient qu'un tuteur humain pouvait offrir un gain d'apprentissage substantiel[14] ($d = 2$) [35]. Par la suite, des travaux menés avec des STI ont montré que ces derniers offraient un gain d'apprentissage équivalent à la moitié de celui offert par les tuteurs humains ($d = 1$) [123, 263]. Plus récemment, des travaux ont plutôt suggéré que les tuteurs humains et les STI offraient des gains d'apprentissage similaires ($d = 0{,}79$ et $d = 0{,}76$ respectivement) [253].

Malgré ces résultats prometteurs et des histoires de réussite (notamment, les centaines de milliers d'élèves américains qui utilisent les Cognitive Tutors de Carnegie Learning[15]), l'intérêt pour les STI est en déclin depuis le début des années 90 [223]. En effet, ils ne suscitent plus l'intérêt des chercheurs en IA [224] et n'ont jamais vraiment capté l'intérêt des chercheurs en éducation puisque ceux-ci préfèrent généralement l'approche pédagogique des hypermédias, micromondes et aux outils cognitifs [23, 184]. Toutefois, les STI occupent encore une place importante au sein de la communauté des EIAH. Par exemple, ils se

[11] Dans la littérature, on retrouve le « Computer-Based Learning » [27] et les « Virtual Learning Environments » [76] qui englobent les outils cognitifs, les hypermédias et les micromondes.
[12] http://www.activemath.org/
[13] « *Hierarchical Task Network* » [96], un terme, tiré de la planification, par lequel nous désignons les connaissances procédurales représentées sous forme de méthodes qui décomposent une tâche en sous-tâches et ce récursivement jusqu'aux actions (on retrouve aussi les « *procedural net* » [217] et les scripts [218]).
[14] Le d de Cohen qui mesure la taille d'effet ($d = 0$ indique qu'il n'y a aucun gain).
[15] http://www.carnegielearning.com/

retrouvent dans les travaux de recherche basés sur l'exploration de données (à l'aide d'entrepôts comme Datashop[16]) et dans ceux qui explorent les approches visant à faciliter et démocratiser la création des EIAH (des ouvrages récents en témoignent [272, 278]).

```
Tant que les objectifs d'apprentissage ne sont pas
atteints :
   1. Le STI choisit une tâche qu'il soumet à l'apprenant
   2. Tant que la tâche n'est pas terminée
        a. Le STI agit en prévision d'une prochaine étape
        b. L'apprenant produit une étape
        c. Le STI réagit face à l'étape produite
   3. Le STI revient sur la tâche effectuée
```

Figure 1 - Pseudocode du comportement général des STI.

Plusieurs familles de STI ont émergé des travaux de recherche effectués depuis les quarante dernières années. Pour les différencier, il faut présenter le comportement général de tous les STI. Ce comportement se décrit comme l'imbrication de deux boucles (Figure 1) : une sur les tâches (*boucle externe*) et une sur les étapes (*boucle interne*) [254]. Dans l'exemple de la tâche de résoudre une *équation*, appliquer une règle de réécriture comme « faire la même *opération* des deux côtés » correspond à une étape. L'*équation* et l'*opération* correspondent à des paramètres, ainsi « additionner 1 des deux côtés » est une instance d'une étape et « résoudre $3x - 1 = x + 2$ » est une instance d'une tâche.

Pour approfondir la description de ce comportement, il faut préciser que la plupart des STI ciblent des tâches tirées de *domaines bien définis*[17], comme plusieurs sous-domaines des sciences pures et appliquées. En contrepartie, les langues, les sciences humaines et les arts sont des exemples de *domaines mal définis*. Rédiger une dissertation à propos d'une œuvre littéraire est un exemple de tâches tirées de domaines mal définis. Dans les domaines bien

[16] https://pslcdatashop.web.cmu.edu/

[17] Dans la littérature, on trouve également les notions de problèmes bien et mal définis [234]. Les tâches bien définies font partie des premiers, les tâches mal définies dans des domaines bien définis se retrouvent dans les deux et les tâches tirées des domaines mal définis font partie des derniers.

définis, on retrouve des *tâches mal définies*, c'est-à-dire des tâches pour lesquelles il est difficile, voire impossible, ou sans intérêt de nature pratique, pédagogique ou de standardisation, de formuler une méthode (c.-à-d. une combinaison particulière d'habiletés cognitives) pour les effectuer[18]. Établir une preuve ou un algorithme, résoudre un casse-tête numérique ou logique, de même que les activités de conception (p. ex. créer un modèle UML) sont des exemples de tâches mal définies dans des domaines bien définis. Dans le cas des *tâches bien définies*, on peut formuler une ou plusieurs méthodes dignes d'intérêt pour les effectuer. Dans une échelle comportant cinq niveaux [178], les tâches bien définies correspondent aux trois premiers, c'est-à-dire des tâches les plus simples jusqu'à celles qui comportent un nombre restreint de choix qui dépendent d'euristiques ou de préférences. Les exercices classiques de mathématiques et de physique, l'opération et la maintenance d'appareils, d'instruments ou de machines en sont des exemples [138, 165].

Au niveau de la boucle externe, l'étendue du comportement global des STI se subdivise en trois cas [254] :

1. demander à l'apprenant de choisir la prochaine instance d'une tâche;

2. suivre une liste d'instances de tâches préétablie par le corps enseignant;

3. optimiser (à l'aide des outils de l'IA) l'atteinte des objectifs d'apprentissage, c'est-à-dire de construire, pour chaque apprenant, la liste d'instances de tâches optimale[19] (certains systèmes peuvent également générer des instances de tâches).

Pour préciser la notion d'objectif d'apprentissage dans le cadre des STI et par la suite considérer le comportement de la boucle interne, il faut introduire la notion d'unités de connaissance. D'un point de vue cognitif, les unités de connaissance d'un domaine [124] sont

[18] Autrement dit, ce n'est pas parce qu'il existe des programmes capables d'effectuer de telles tâches, que les connaissances qu'ils représentent plus ou moins implicitement peuvent être explicitées ou sont intéressantes à l'être (p. ex. un programme capable de résoudre tous les Sudokus).
[19] Dans la littérature, on parle parfois de « *macroadapation* » [231], dans ce cas le traitement associé à la boucle interne est nommé « *microadaptation* » [232].

les éléments de la mémoire à long terme [26] dont l'exécution produit les étapes pour effectuer une tâche. Les architectures cognitives [129, 175, 249], notamment ACT-R[20] [10], sont des SABC qui modélisent finement non seulement la mémoire à long terme, mais toutes les structures et les processus mentaux [28] (de façon limitée, puisque c'est un problème IA-complet).

En particulier, les architectures cognitives modélisent des mécanismes d'apprentissage [187] qui simulent l'acquisition d'unités de connaissance lorsqu'une tâche est effectuée. Par conséquent, en fixant les paramètres qui régissent le comportement interne de l'architecture, on obtient une simulation de l'exécution d'une tâche qui est fonction du modèle de la tâche et du modèle de l'humain qui effectue celle-ci. Le modèle de la tâche est constitué d'éléments de connaissance qui opérationnalisent les unités de connaissance[21]. Par exemple, dans ACT-R ce sont des règles de production et des « *chunks* » (l'équivalent d'un « *template* » dans un système de production ou plus généralement un enregistrement [22]) qui représentent respectivement les unités de connaissance procédurale et déclarative [11]. Le modèle de l'humain est constitué d'une part de paramètres qui, par exemple, distinguent un humain qui apprend plus rapidement qu'un autre et d'autre part d'unités de connaissance déjà acquises. Pour les STI, la complexité d'une architecture cognitive n'est pas nécessaire, c'est-à-dire que ceux-ci n'ont pas besoin de simuler finement l'apprenant [12] (ou encore son tuteur humain quoique ce soit une piste de recherche, par exemple [79]).

Dans la perspective des STI, les unités de connaissance servent avant tout à définir les objectifs d'apprentissage et à mesurer l'atteinte de ces derniers en fonction des étapes produites par l'apprenant pour effectuer les tâches. Dans notre exemple, si l'unité de connaissance « résoudre une équation linéaire » est un objectif d'apprentissage, son atteinte est fonction de l'évaluation des d'unités de connaissance sous-jacentes (p. ex. « résoudre une équation de la forme $ax + b = c$ » et « appliquer une opération des deux côtés de l'égalité »).

[20] http://act-r.psy.cmu.edu/
[21] Les unités forment le « *knowledge level* » tandis que les éléments forment le « *symbol level* » [176].
[22] « *record* » ou « *struct* ».

Dans les différentes familles de STI, la granularité des étapes varie significativement, et ce indépendamment du domaine [264]. Ainsi, pour une même tâche, il peut aussi bien y en avoir une seule que quelques dizaines. Cependant, peu importe sa granularité, une *étape* [262] représente une itération de la boucle interne sous la forme d'échanges entre l'apprenant et le système. Cet échange contient les tentatives infructueuses (actions incorrectes), la réussite de l'étape (action correcte), les demandes d'aide (action neutre) et les interventions du système qu'elles soient survenues avant ou après la réussite de l'étape. À partir de l'étape, le système détermine, de façon plus ou moins certaine, quelles sont les unités de connaissance qui ont été utilisées par l'apprenant pour produire l'étape et met à jour son évaluation des acquis (positivement ou négativement). En particulier, une action incorrecte peut ainsi être associée à une unité de connaissance qui représente une incompréhension[23], c'est-à-dire une erreur pédagogiquement pertinente [83]. Ensuite, lorsque la tâche est terminée, le système informe l'apprenant de son progrès vers l'atteinte des objectifs d'apprentissage, ce qui met fin à une itération de la boucle externe.

Pour la résolution d'une équation, un premier exemple du comportement de la boucle interne est l'apprenant qui produit la réécriture « $x + 1 = 3$ » à partir de l'équation initiale « $2x + 2 = 6$ », et le système qui lui demande plutôt d'appliquer une réécriture qui simplifie l'équation en éliminant un terme (c.-à-d. « soustraire 2 des deux côtés »). Un deuxième exemple est le système qui indique que la réécriture « $x + 2 = 3$ » à partir de l'équation initiale est invalide en précisant qu'elle n'a pas été appliquée uniformément (p. ex. « il faut diviser par 2 des deux côtés »). Un troisième exemple est que le système refuse, de la part d'un novice, la réécriture « $x = 2$ » à partir de l'équation de départ puisqu'elle résulte de la composition de deux réécritures.

L'architecture des STI

Pour préciser davantage le comportement des STI, il faut s'intéresser à leur conception. L'architecture classique d'un STI est constituée de quatre modules (Figure 2) [193, 224,

[23] Dans le cas des connaissances procédurales, la littérature parle parfois en terme de bogues [42].

271] : le module communication, le module expert, le module apprenant et le module pédagogue. Même si certains STI ne sont pas conçus selon cette approche, on retrouve généralement dans leur architecture, organisés différemment, les rôles que nous attribuons aux différents modules (des STI négligent des rôles moins essentiels).

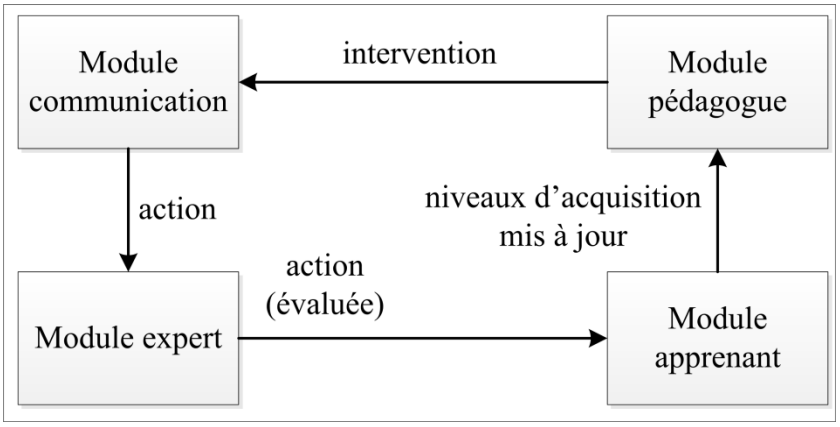

Figure 2 - L'architecture classique des STI (et le flot de données principal).

Le module **communication** est constitué d'une part de l'interface utilisateur (UI[24]) présentée à l'apprenant (idéalement en l'adaptant à ce dernier). En fait, l'UI est constituée de l'*environnement* propre à la tâche (ou au domaine) et du *tuteur* où a lieu la communication directe entre l'apprenant et le système (Figure 3). D'autre part, le module communication fait le pont entre l'UI et le module expert, et inversement, le module pédagogue et l'UI. En effet, il traduit les interactions dérivées des manipulations du clavier et de la souris dans l'UI en actions propres à l'environnement (p. ex. le choix d'une réécriture en) ou au tuteur (p. ex. une demande d'aide). Inversement, il traduit les interventions du module pédagogue en effets (p. ex. afficher un message texte et surligner un composant graphique) dans l'UI (Figure 3). En somme, le module communication joue son rôle adéquatement [19], c'est-à-dire de façon à répondre aux exigences suivantes :

1. Les actions faites par l'apprenant dans l'UI doivent être transmises au système.

[24] « *User Interface* »

2. Le système doit avoir une représentation de l'état de l'UI.

3. Le système doit pouvoir faire les actions dans l'UI.

De plus, l'UI est préférablement conçue avant le reste du système afin de circonscrire la tâche [19]. En particulier, sa conception détermine les actions possibles dans l'environnement et le tuteur, et détermine leur granularité.

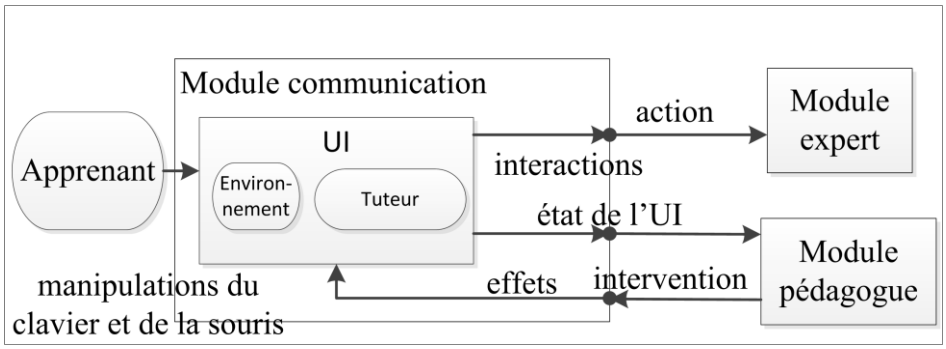

Figure 3 - Les entrées/sorties du module communication.

Un premier rôle du module **expert** est de fournir des instances de tâches. Selon la famille de STI et le domaine dont est tirée la tâche, soit le corps enseignant crée directement les instances, soit il spécifie les paramètres nécessaires pour les générer à l'aide d'outils de l'IA. Un deuxième rôle est d'évaluer les actions brutes qu'il reçoit du module communication et de transmettre son évaluation aux modules apprenant et pédagogue. Un troisième rôle est de fournir les connaissances didactiques[25] nécessaires pour adapter les interventions du module pédagogue au domaine et à la tâche[26]. Ces connaissances didactiques sont, par exemple, des gabarits de messages (qui contiennent des références aux paramètres des étapes) créés par le corps enseignant afin que le STI produise les indices et les rétroactions. Plusieurs approches sont possibles pour que le module expert remplisse ses rôles, on les distingue généralement en fonction du compromis qu'elles offrent entre les efforts requis pour créer le module expert

[25] Dans la littérature, cela correspond au « *pedagogical content knowledge* » [230].
[26] Plusieurs STI amalgament les connaissances didactiques et les connaissances pédagogiques au sein du module pédagogue. Puisque les connaissances didactiques font référence aux connaissances propres à la tâche et au domaine, dans cette thèse nous considérons qu'elles sont contenues au sein du module expert.

et la capacité de ce dernier à remplir ses rôles. Même si elle garantit un module expert capable de remplir tous ses rôles, l'approche classique, qui consiste à créer un modèle de la tâche, est de plus en plus jugée comme étant problématique parce qu'elle demande des efforts importants [276]. Par conséquent, dans les dernières années, des travaux de recherche ont étudié l'utilisation de modèles générés à partir de traces d'actions (nous y revenons plus loin dans cette section).

Au sein de l'approche classique, on distingue deux sortes de modèles [188] :

1. un modèle génératif;
2. un modèle évaluatif.

Ces deux sortes de modèles sont dits cognitifs [12] s'ils prennent en compte des contraintes psychologiques, pédagogiques ou didactiques [1]. Par le passé, on retrouvait également d'une part des modèles dits « boite de verre » [12], c'est-à-dire un modèle encodé à l'aide d'un SABC pour obtenir une évaluation précise des actions de l'apprenant, mais pas nécessairement adéquate d'un point de vue cognitif (p. ex. la première version de Guidon [57] [diagnostic médical]). D'autre part, des modèles dits « boite noire » [12], c'est-à-dire un modèle encodé sous la forme d'un programme traditionnel qui évalue grossièrement les actions de l'apprenant. Par exemple, la première version de SOPHIE [43] (dépannage de circuits électroniques) utilisait directement le simulateur SPICE[27] comme modèle.

Les *modèles génératifs* génèrent les actions nécessaires pour effectuer la tâche. Même si pour ce faire ils n'ont pas besoin d'une simulation aussi précise que celle produite par une architecture cognitive comme ACT-R, ce sont tout de même les modèles qui requièrent les efforts de création les plus importants. Pour déterminer la validité d'une action, le système vérifie si elle appartient aux actions correctes (ou à celles d'actions incorrectes propres à des incompréhensions) qu'il a générées. Guidon [56], IDEBUGGY (arithmétique) [48] et Lisp Tutor [22] en sont des exemples classiques.

[27] http://www.ngspice.com/

Les *modèles évaluatifs* évaluent les actions de l'apprenant directement (sans générer les actions comme les modèles génératifs). Pour ce faire, ils ont recours à des connaissances propres à la tâche (p. ex. des traces d'actions prédéfinies par le corps enseignant) ou au domaine (p. ex. une contrainte qui vérifie qu'une équation est valide). Selon la nature du domaine et de la tâche, ces modèles reconnaissent les actions correctes ou incorrectes (si la reconnaissance échoue, l'action est alors respectivement considérée comme incorrecte ou correcte). Proust (programmation en Pascal) [110], Cardiac Tutor (réanimation) [81] et Sherlock (diagnostic de pannes) [134] sont des exemples classiques. Project LISTEN[28] (lecture) [166] et le module SIMILE de la plateforme GIFT sont des exemples plus récents.

Le premier rôle du **module apprenant** est de fournir un modèle qui représente chacun des apprenants. Ce modèle est constitué [47] de :

- la liste des unités de connaissance acquises;
- l'historique des étapes produites et des tâches effectuées;
- le profil qui regroupe des informations utiles pour l'adaptation de l'UI : le sexe, le groupe d'âge, le bagage culturel, les aspects métacognitifs (p. ex. propension à demander de l'aide [3] ou à déjouer le système [31]) et les aspects émotifs [135, 273].

Le deuxième rôle de ce module est de mettre à jour ce modèle. Pour la liste des unités de connaissance acquises, plus l'évaluation des actions de l'apprenant par le module expert est précise, plus la mise à jour est simple [255]. Par contre, même si l'évaluation est précise, on ne peut pas totalement résoudre le problème d'incertitude sur les unités de connaissance qui ont mené l'apprenant à produire l'action (p. ex. un novice qui fait une action correcte par chance ou un habitué qui fait une erreur par mégarde) [60, 75, 83, 207, 225]. Par conséquent, les acquis sont mis à jour à l'aide d'outils de l'IA, notamment l'inférence bayésienne [62, 67, 75]. Avec ces mêmes outils, le profil est mis à jour en fonction des actions dans l'UI (p. ex. la fréquence des demandes d'aide) et parfois même à l'aide de capteurs [115].

[28] http://www.cs.cmu.edu/~listen/

Le troisième rôle du module apprenant est de fournir au module pédagogue des prévisions à partir de ce modèle (p. ex. la probabilité que l'apprenant produise la prochaine étape). En pratique, informer le module pédagogue se révèle être moins difficile que le laisse croire le problème de la mise à jour, puisqu'on peut tenir compte du fait que le nombre d'interventions possibles est restreint [226]. Ultimement, le module apprenant se comporte à l'image du modèle de l'humain dans une architecture cognitive, permettant au STI de simuler l'exécution des boucles internes et externes. Une telle simulation permet, entre autres, d'observer le comportement d'une stratégie pédagogique [140, 170, 189, 193] avant de l'expérimenter auprès d'apprenants [77, 252, 265].

Comme mentionné précédemment, le **module pédagogue** a un premier rôle au niveau de la boucle externe qui consiste à :

- faire la sélection de tâches (au besoin);
- offrir une rétroaction sur la tâche effectuée (p. ex. par rapport au progrès vers l'atteinte des objectifs d'apprentissage).

Son deuxième rôle est au niveau de la boucle interne, il intervient en amont et en aval de chacune des actions produites. Son rôle est donc de déterminer quand intervenir (ou ne pas intervenir), mais aussi quoi faire comme intervention et finalement comment appliquer cette dernière à la situation courante. Les interventions les plus communes au niveau de la boucle interne sont [254] :

- la rétroaction minimale qui indique si l'action produite est correcte ou non;
- la rétroaction sur l'erreur qui explique pourquoi l'action produite est incorrecte;
- un indice qui aide l'apprenant à produire la prochaine action;

Dans la prochaine section, nous présentons les plus importantes familles de STI qui ont émergé au cours des années en fonction des intérêts des chercheurs.

Les familles de STI

Les STI « par dialogues », qui sont basés sur un modèle évaluatif, sont des systèmes pour lesquels les actions produites dans l'UI sont principalement des questions ou des réponses (formulées à la fois par l'apprenant et le système) dans le tuteur plutôt que des manipulations dans l'environnement [100, 243]. CIRCSIM-Tutor (diagnostic médical) [84], AutoTutor (matériel informatique) [185], Why2-Atlas, Cordillera et BEETLE II (physique) [52, 80, 266] en sont des exemples. Nous n'avons pas étudié plus en détail ces systèmes puisqu'ils prennent de front le problème du traitement automatique du langage naturel (TAL); or, celui-ci requiert au mieux des efforts propres à la tâche qui sont substantiels ou, dans sa forme générale, est IA-complet.

Les STI « par traçage de modèle » (MTT pour « *Model-Tracing Tutors* ») [15, 189], qui sont basés sur un modèle génératif, produisent les actions à l'aide d'un modèle qui représente une ou plusieurs méthodes pour effectuer la tâche. Bien que les MTT soient généralement associés à une évaluation serrée [69] et à des efforts de création prohibitifs [4], ce sont les STI les plus utilisés en situation réelle d'apprentissage. Les Cognitive Tutors [19] (algèbre [211], géométrie [2], etc.), Andes (physique) [263] et Slide Tutor (diagnostic médical) [72] en sont des exemples.

Les STI « par contraintes » (CBT pour « *Constraint-Based Tutors* ») [157], qui sont basés sur un modèle évaluatif, évaluent les actions produites à l'aide de contraintes[29]. En principe, les contraintes expriment en premier lieu les connaissances générales du domaine et en second lieu les connaissances propres à la tâche [190]. En pratique, elles s'expriment aussi à partir de traces d'actions correctes fournies par le corps enseignant[30], ce qui simplifie la création du modèle [119, 120, 158]. Contrairement aux MTT, qui guident l'apprenant à partir des actions produites ou à produire, les CBT guident l'apprenant à partir des contraintes qui n'ont pas encore été validées ou qui ont été enfreintes [158]. SQL-Tutor (requêtes SQL) est le CBT le

[29] Ne pas confondre avec les « problèmes de satisfaction de contraintes » en IA; les contraintes des CBT sont prescriptives, elles ne sont pas nécessairement suffisantes pour permettre de générer une solution.
[30] Désignées dans la littérature comme « *ideal solution* »

plus abouti et le plus connu [162]. ASPIRE [161] est une plateforme qui facilite la création de CBT[31], elle a par exemple été utilisée afin de créer un CBT pour la conception d'un « modèle entité-association étendu »[32] [245].

Les STI « par traçage d'exemples » (ETT pour « Example-Tracing Tutors ») [5] sont des systèmes basés sur une approche, apparue au début des années 2000, dont l'objectif est de transférer une partie des efforts de création des auteurs (des programmeurs) au corps enseignant (des non-programmeurs). La plateforme CTAT [33] [4], offerte sous licence universitaire, en est l'exemple le plus connu (elle permet aussi la création de MTT). Inspiré de la « programmation par la démonstration » [279], cette approche consiste à générer un modèle évaluatif à partir de traces d'actions dans l'environnement et d'annotations (correcte, incorrecte, etc.) fournies par le corps enseignant [121]. CTAT offre également des outils adaptés à des non-programmeurs pour créer l'UI et généraliser le modèle qui est, à la base, propre à une instance de la tâche (p. ex. « résoudre $3x + 4 = 2$ »). Les concepteurs de CTAT admettent eux-mêmes que cette approche comporte des limitations importantes, en particulier qu'elle ne s'applique pas à toutes les tâches bien définies.

Des chercheurs poussent l'idée d'éliminer le besoin de recourir à des auteurs pour créer un modèle encore plus loin grâce à l'apprentissage automatique. Des précurseurs de cette approche sont ACM [130], qui à partir des actions et des traces des apprenants, tente de générer un modèle génératif, et PIXIE [238], qui à partir des traces des apprenants, ajoute à son modèle des éléments de connaissances qui représentent des incompréhensions. Parmi les systèmes actuels, on retrouve par exemple SimStudent, un système qui construit un modèle à l'aide de traces [143] et Deep Thought, un STI pour les preuves en logique où les actions de l'apprenant sont classifiées directement à l'aide de traces [239].

Quelques STI notoires n'entrent pas dans ces familles, d'une part les précurseurs Scholar [49, 50], Why [243] et SOPHIE avec lesquels on tentait d'obtenir un modèle génératif basé sur

[31] http://aspire.cosc.canterbury.ac.nz/
[32] « *Enhanced entity-relationship model* » [82]
[33] http://ctat.pact.cs.cmu.edu/

des inférences logiques (c'est-à-dire des connaissances procédurales indépendantes de la tâche) appliquées aux connaissances déclaratives du domaine [227]. Ces tentatives se butent tôt ou tard aux problèmes IA-complets du sens commun [151, 168] et de la résolution de problèmes [131, 235] (qui plus est, la plupart de ces systèmes utilisaient également des dialogues). D'autre part, dans des STI comme Steve [111, 210] et REACT [108] (opération et maintenance), les concepteurs voulaient éviter l'évaluation serrée des MTT puisqu'ils s'intéressaient à des environnements dynamiques. Pour ce faire, ces systèmes ont recours à des modèles génératifs qui agissent comme des planificateurs qui ont été conçus à l'aide du SABC Soar [127, 177]. Cette approche[34], bien que plus pragmatique, fait face tôt ou tard (selon la nature de la tâche et du domaine) aux mêmes problèmes IA-complets.

La pertinence des STI

L'idée même d'un tuteur (humain ou informatique) qui guide l'apprenant est contestée (des chercheurs en éducation et en psychologie se livrent une âpre bataille à ce sujet [6, 21, 118, 146, 248]). De plus, même si on admet l'importance de guider l'apprenant, la popularité des stratégies pédagogiques qui mettent l'accent sur les tâches mal définies (p. ex. l'apprentissage par problèmes) limite l'utilisation des STI. Malgré les différences significatives entre les différentes familles de STI, les critiques qui leur sont formulées mettent en lumière leurs points en commun et ce qui les différencie des autres types d'EIAH. Essentiellement, l'évaluation des actions faites par le module expert[35] n'est jamais aussi flexible qu'un tuteur humain [184] (un problème évité par les autres types d'EIAH). Même si c'est plus flagrant dans le cas des MTT qui suivent l'apprenant de façon serrée, c'est une limitation fondamentale pour tous les STI d'ici à l'avènement de l'IA forte. En fait, bien que ces critiques aient motivé les travaux sur les autres types d'EIAH (où l'évaluation est soit simple ou inexistante), ceux-ci n'ont pas, à notre connaissance, montré aussi clairement que les STI qu'ils procurent des gains d'apprentissage.

[34] Qui correspond aux « *teleo-reactive programs* » [181].
[35] Des critiques similaires ont été faites concernant les modules apprenant et pédagogue.

L'approche des MTT est particulièrement critiquée [155]; avec leur évaluation serrée, mais réputée efficace [68], et des actions de l'apprenant qui s'appliquent nécessairement à des tâches bien définies[36], on leur reproche d'encourager un apprentissage superficiel [107]. La profondeur de l'apprentissage est établie à partir de propriétés que l'on prête aux connaissances profondes, par exemple elles sont abstraites, transférables, interconnectées ou encore flexibles [32, 113, 240, 241]. Un exemple classique d'un apprentissage superficiel est l'apprenant qui applique et justifie correctement la formule de l'aire d'un parallélogramme seulement si la projection de la hauteur ne sort pas de ce dernier. Or, même s'il existe des domaines où l'acquisition de connaissances procédurales superficielles est pertinente, la question est de savoir si les MTT contribuent à l'acquisition de connaissances procédurales profondes et à l'approfondissement des connaissances déclaratives [2, 263]. Sans prétendre avoir une réponse à cette question, nous proposons dans cette thèse qu'un MTT peut éviter de renforcer inutilement des connaissances procédurales superficielles. Avant de remettre en cause la pertinence des MTT, il faut tenir compte du rôle pédagogique qu'on leur demande de remplir. D'une part, puisque leur évaluation est avant tout fonction des connaissances procédurales des apprenants, ils supposent que ces derniers ont déjà acquis les connaissances déclaratives préalables [19]. D'autre part, que les MTT soient utilisés en classe ou à la maison (devoirs), ils font généralement partie d'un curriculum qui comprend d'autres types d'activités d'apprentissage. En somme, ils sont un outil parmi d'autres que le corps enseignant peut utiliser, à sa juste valeur, comme le sont les autres familles de STI et les autres types d'EIAH.

Problématique

Tout en faisant abstraction de la pertinence pédagogique des MTT, leurs principaux représentants, les Cognitive Tutors, ont été plus particulièrement critiqués d'une part parce que les interventions qu'ils produisent proviennent d'un module pédagogue simpliste (autrement dit, ses connaissances pédagogiques sont limitées). En effet, puisque ces

[36] Des systèmes ont recours à des astuces afin de masquer cette limitation (nous y revenons au chapitre 2).

interventions prémâchées sont directement créées à partir de gabarits de message préétablis lors de la création du MTT, elles sont difficilement adaptables. D'autre part, parce que les efforts de création qu'ils nécessitent sont un obstacle important à leur diffusion [107].

Le processus de création d'un MTT comporte quatre étapes importantes. La première est l'analyse cognitive de la tâche [58] qui met en évidence non seulement les actions qui sont effectuées dans l'environnement, mais aussi les actions mentales sous-jacentes. La deuxième est la création de l'UI qui doit tenir compte en plus des besoins communs à tous les STI (énoncés dans la section précédente), des besoins particuliers des MTT, c'est-à-dire de faciliter l'évaluation serrée des actions de l'apprenant sans le contraindre inutilement [19]. La troisième est la création du modèle de la tâche, c'est-à-dire l'encodage des éléments de connaissance qui reflètent le résultat de l'analyse. La quatrième étape est de créer les instances des tâches (ou encore de spécifier les paramètres qui permettent au système de les générer). Ce processus de création requiert la collaboration entre les auteurs du modèle et le corps enseignant, comme c'est le cas pour de nombreuses autres catégories de logiciels (et ce bien au-delà des EIAH) où les programmeurs doivent collaborer avec les spécialistes du domaine d'application.

Comme nous l'avons évoqué dans la section précédente, plusieurs approches ont été poursuivies afin d'éliminer ou d'éviter des parties de ce processus (les ETT, les CBT ou encore les approches basées sur l'apprentissage automatique). Comme pour plusieurs catégories de logiciels (par exemple les jeux vidéos), il est possible de créer une plateforme[37] pour les MTT qui, d'une part, factorise les efforts qui sont indépendants de la tâche et d'autre part, qui offre des outils qui réduisent les efforts qui lui sont propres. De cette façon, le processus de création est plus efficace que si l'on crée un MTT de toute pièce. La plateforme CTAT en est un exemple, mais les MTT créés avec celle-ci sont assujettis aux limitations évoquées précédemment (d'autres sont identifiées au chapitre 1). De plus, selon notre expérience [87, 133, 197], CTAT n'est pas facilement extensible, ce qui est selon nous

[37] Non pas au sens strict d'un « *framework* », mais au sens plus large d'un « *Software Development Kit* » (SDK).

nécessaire pour qu'une plateforme satisfasse les besoins des chercheurs en éducation ou en psychologie qui font des expérimentations sur l'apprentissage et la pédagogie avec des EIAH.

La tentative la plus manifeste de surpasser ces limitations dans la famille des MTT est Ms. Lindquist (algèbre) [107]. Son approche consiste en un module pédagogue dont les interventions sont grandement enrichies par l'ajout de dialogues[38] par le corps enseignant (Andes-Atlas [214], demeuré au stade expérimental, appliquait la même approche dont GIL [Lisp] était un précurseur [205]). Or cette façon de faire ne nous apparait pas adéquate pour promouvoir le développement des MTT. D'une part, les gains d'apprentissage obtenus en offrant un dialogue en langage naturel au sein des MTT sont contestés [261]. D'autre part, les efforts supplémentaires pour créer ces dialogues (ou d'autres connaissances didactiques sophistiquées) sont déraisonnables lorsqu'on considère que ceux nécessaires pour créer un MTT, même ceux créés à l'aide d'une plateforme comme CTAT, sont jugés comme étant prohibitifs [4].

Il y a un espace inoccupé entre les MTT créés avec CTAT et ceux comme Ms. Lindquist. En fait, il y a des MTT qui occupent cet espace (par exemple Andes), mais ils ont été créés de toute pièce et non à l'aide d'une plateforme. La plateforme TOTS [208] exploite un modèle de la tâche (un HTN) simple, mais suffisant pour générer des interventions plutôt que de recourir à des interventions prémâchées. Toutefois, elle n'a pas été rendue accessible, de même qu'aucun des MTT qu'elle a pu servir à créer[39]. De plus, la simplicité de son modèle a fait en sorte qu'elle n'a été appliquée qu'à des tâches similaires à l'opération et à la maintenance, et non à des tâches tirées de domaines classiques comme les mathématiques et la physique. Autrement dit, TOTS ne peut pas remplacer CTAT.

La génération des interventions permet selon nous d'explorer cet espace puisque même si les interventions prémâchées ont l'avantage de pouvoir faire directement référence à des éléments qui ne sont pas présents dans le modèle de la tâche (connaissances générales,

[38] Originalement « *knowledge-construction dialog* » [260], mais on retrouve aussi « *micro-plan* » [148] et « *micro-step* » [55]. Des outils, tels que TuTalk [114], facilitent la création de dialogues par les auteurs.
[39] Son concepteur, J. Rickel, est décédé en 2003 des suites d'un cancer à l'âge de 40 ans.

principes du domaine, connaissances téléologiques[40], etc.), les interventions générées sont systématiques et peuvent tenir compte d'une foule d'éléments (p. ex. du modèle de l'apprenant) auxquels les auteurs n'ont pas accès lors de la création. Autrement dit, nous supposons que les interventions générées sont potentiellement plus efficaces que les interventions prémâchées.

Objectifs et méthodologie

Puisque les MTT comme Ms. Lindquist, ou ceux comme Andes nécessitent de trop grands efforts de création et que ceux créés avec CTAT sont trop limités, nous considérons que l'usage des MTT ne peut pas croitre davantage. De plus, les nombreux travaux récents qui explorent les approches qui ne sont pas basées sur un modèle de la tâche créé par des experts du domaine témoignent d'une perte d'intérêt pour les MTT dans la communauté des EIAH. Qui plus est, l'approche pédagogique des MTT est mise à mal par la littérature en éducation.

Considérant que les MTT sont délaissés par la communauté des EIAH, l'objectif plus général de cette thèse est de renouveler l'intérêt pour ces derniers. Plus particulièrement, nous faisons l'hypothèse qu'une plateforme qui satisfait les exigences suivantes peut contribuer à l'atteinte d'un tel objectif :

- mettre en évidence les avantages et les désavantages de l'approche pédagogique propre aux MTT;
- interpréter un modèle de la tâche pour générer des interventions sans exiger des efforts de création plus importants;
- produire des interventions riches grâce à un modèle de l'UI plus sophistiqué;
- supporter un maximum de tâches bien définies, peu importe le domaine;

[40] C.-à-d. une justification (formelle ou non) des connaissances procédurales à partir des principes du domaine.

- supporter différentes stratégies pédagogiques.

Plus concrètement, les travaux effectués dans le cadre de cette thèse ont permis d'atteindre les objectifs suivants :

1. établir les caractéristiques permettant à une plateforme de répondre à ces exigences en fonction des avantages recherchés et des désavantages potentiels;

2. développer une version d'une telle plateforme qui est suffisante pour mener des expérimentations, décrire ses limitations et les travaux futurs qui les repousseront;

3. exposer les avantages et les désavantages des MTT qu'une telle plateforme permet de créer par rapport aux MTT existants et aux autres familles de STI.

Pour atteindre ces objectifs, nous avons développé Astus, une plateforme inspirée des travaux passés du laboratoire ASTUS [91, 173] qui découlent eux-mêmes de l'architecture cognitive Miace [147]. Le module expert d'Astus gère un modèle de la tâche qui reflète les instructions d'un tuteur humain et qui peut être interprété pour générer des interventions [195]. Astus, comme CTAT, offre des outils pour faciliter le processus de création de façon à rendre les efforts nécessaires comparables à ceux exigés par ce dernier. Le module communication d'Astus, grâce à son modèle de l'UI, peut produire des interventions riches comme des démonstrations (déplacements du pointeur et simulation des clics et des saisies) [87]. Le module pédagogue d'Astus offre une stratégie pédagogique de base qui peut être adaptée par un tiers. Finalement, Astus offre un module apprenant minimal, car malgré l'importance de celui-ci [116, 150, 228], il n'a pas été au centre de nos travaux.

Les interventions générées par Astus ont fait l'objet d'expérimentations, à petite échelle, effectuées à l'interne[41] à l'aide d'un MTT pour la conversion de nombres en virgule flottante

[41] Avec la participation d'étudiants du baccalauréat au département d'informatique.

et d'un MTT pour la manipulation d'arbres binaires de recherche (ABR)[42] [196, 198, 199]. Les résultats de ces expérimentations montrent que les interventions sont efficaces [195].

L'atteinte des objectifs est confirmée parce qu'Astus, malgré ses limitations, a des caractéristiques qui font en sorte qu'elle répond aux exigences énoncées précédemment et que les MTT créés avec Astus ont des avantages et des désavantages qui les démarquent des MTT existants et des autres familles de STI.

Structure

- Dans le chapitre 1, on retrouve l'état de l'art des MTT.
- Dans le chapitre 2, on retrouve une synthèse des caractéristiques d'Astus qui ont émergé du développement. Cette synthèse fait abstraction de choix particuliers faits au niveau conceptuel et au niveau de l'implémentation.
- Dans le chapitre 3, on retrouve une présentation de la plateforme Astus, incluant une description des MTT créés et des expérimentations menées à l'interne.
- Dans le chapitre 4, on retrouve une analyse des caractéristiques d'Astus.
- Dans le chapitre 5, on retrouve une discussion qui positionne les MTT créés avec Astus par rapport aux autres MTT et par rapport aux autres STI.

L'état de l'art du chapitre 1, la synthèse du chapitre 2 et l'analyse du chapitre 4 sont présentés dans une structure commune qui met en évidence les éléments sur lesquels nos travaux ont porté :

1. le modèle de la tâche;
2. l'interaction avec l'environnement;

[42] L. Paquette a mené les expérimentations et a créé les MTT utilisés à cet effet.

3. le suivi de l’apprenant et la sélection de tâches;

4. le processus de création.

Chapitre 1

État de l'art des MTT

Les MTT, parce qu'ils évaluent l'apprenant de façon serrée en évitant de le contraindre inutilement, sont des STI qui nécessitent des efforts de création importants. Les MTT sont les STI qui ont le plus fait leur preuve, d'une part par le nombre d'apprenants qui les ont utilisés et d'autre part par les gains d'apprentissage obtenus lors d'expérimentations à grande échelle. Par contre, ce sont aussi les STI les plus critiqués, car leur évaluation serrée dans le cadre d'une tâche bien définie ne concorde pas avec les approches pédagogiques dominantes dans la littérature en éducation (qui sont contestées dans la littérature sur l'apprentissage en psychologie). Pour concrétiser l'approche des MTT et pour établir des points de référence avec lesquels comparer nos travaux sur Astus, nous présentons dans ce chapitre : les Cognitive Tutors (les STI les plus utilisés dans un contexte réel), Andes (le MTT le plus achevé) et Slide Tutor, un exemple dans un domaine moins traditionnel pour les MTT, le diagnostic médical.

1.1 Les Cognitive Tutors

L'historique des Cognitive Tutors (CT) remonte à Lisp Tutor (1985) [22] qui a été créé pour valider expérimentalement l'architecture cognitive ACT-R[43] [16]. Par la suite, des CT pour l'algèbre, la géométrie et d'autres sous-domaines des mathématiques des niveaux primaire et secondaire ont été créés, d'abord par les chercheurs de Carnegie Mellon University (CMU), puis par l'entreprise Carnegie Learning. En fait, sans compter les précurseurs qui ont été créés de toute pièce comme Lisp Tutor, il existe trois variantes des CT :

[43] À cette époque, c'était ACT* [14], un des précurseurs d'ACT-R

1. Les CT créés à l'aide du TDK (« *Tutor Development Kit* ») lui-même basé sur un système de production dédié (Tertl) [20]. Ces CT, initialement créés par les chercheurs de CMU, ont ensuite été repris par Carnegie Learning. À notre connaissance, le TDK n'est plus utilisé à l'interne depuis quelques années et il n'est pas accessible.

2. Les CT créés à l'aide de CTAT [1] (Figure 4), une plateforme offerte sous licence universitaire qui a été développée par CMU. Plutôt qu'utiliser un système de production dédié comme le fait le TDK, CTAT est basé sur Jess[44], un système de production offert sous licence universitaire.

3. Les CT créés à l'aide du « Cognitive Tutor SDK » (CT-SDK), une plateforme développée par Carnegie Learning, qui offre des « outils auteurs » (« *authoring tools* ») [45] [171] pour démocratiser l'encodage des modèles et qui supporte l'utilisation de logiciels existants (p. ex. Microsoft Excel) comme environnement [34]. À notre connaissance, le CT-SDK n'est pas accessible.

Le TDK offre une plus grande souplesse que CTAT, l'auteur peut modifier, plus ou moins facilement, tous les comportements par défaut à l'aide de code Lisp supplémentaire dans le système de production (nous donnons un exemple de cette souplesse plus loin dans ce chapitre). Nous ne nous prononçons pas davantage sur le CT-SDK, puisque la documentation accessible ne nous permet pas de le faire objectivement. Dans le reste de ce chapitre, nous ciblons les CT créés à l'aide de CTAT.

1.1.1 Le modèle de la tâche

À l'image des modèles d'ACT-R, les modèles des CT représentent une méthode pour effectuer une tâche[46] à l'aide d'un système de production. Comme les modèles d'ACT-R, ils

[44] Jess, the Rule Engine for the Java Platform (http://herzberg.ca.sandia.gov/)
[45] Dans la littérature, « *authoring tools* » désigne des outils pour des non-programmeurs.
[46] Dans les cas des CT, des tâches bien définies (à quelques astuces près), dans le cas d'ACT-R, on retrouve des tâches mal définies qui correspondent à des problèmes bien définis comme celui des tours de Hanoï.

contiennent deux catégories d'éléments de connaissance : des règles et des WME (« *Working Memory Element* »), l'équivalent des *chunks* d'ACT-R (Tableau 1), qui représentent respectivement les unités de connaissance procédurales et les unités de connaissance déclaratives. Contrairement à la modélisation avec ACT-R, la modélisation avec les CT se fait :

- avec une granularité moins fine [12] (p. ex. il est inutile de modéliser l'addition de deux nombres inférieurs à 10 dans un CT pour l'addition en colonnes ou encore il n'est pas toujours nécessaire d'ajouter des WME qui représentent des buts);
- sans avoir recours aux modules perceptivo-moteurs et à la simulation des processus mentaux (tampons, niveaux d'activations, etc.);
- sans mécanismes d'apprentissage (c.-à-d. la création de nouvelles règles à partir de règles existantes ou de *chunks*).

Tableau 1 - Correspondance entre la terminologie d'ACT-R, des CT et de Jess

ACT-R	**CT**	**Jess**
règle (de production)	règle (de production)	`rule`
type de *chunk*	type de WME	`template`
instance de *chunk*	instance de WME	`fact`
slot	idem	`slot`, `multislot`

De plus, dans les modèles des CT, les règles sont généralement prépondérantes par rapport aux WME puisque ces derniers ne représentent que l'environnement et les résultats des actions mentales (dans un modèle ACT-R, les *chunks* peuvent aussi représenter des principes du domaine ou encore des instructions qui sont manipulées par des règles indépendantes de la tâche ou du domaine).

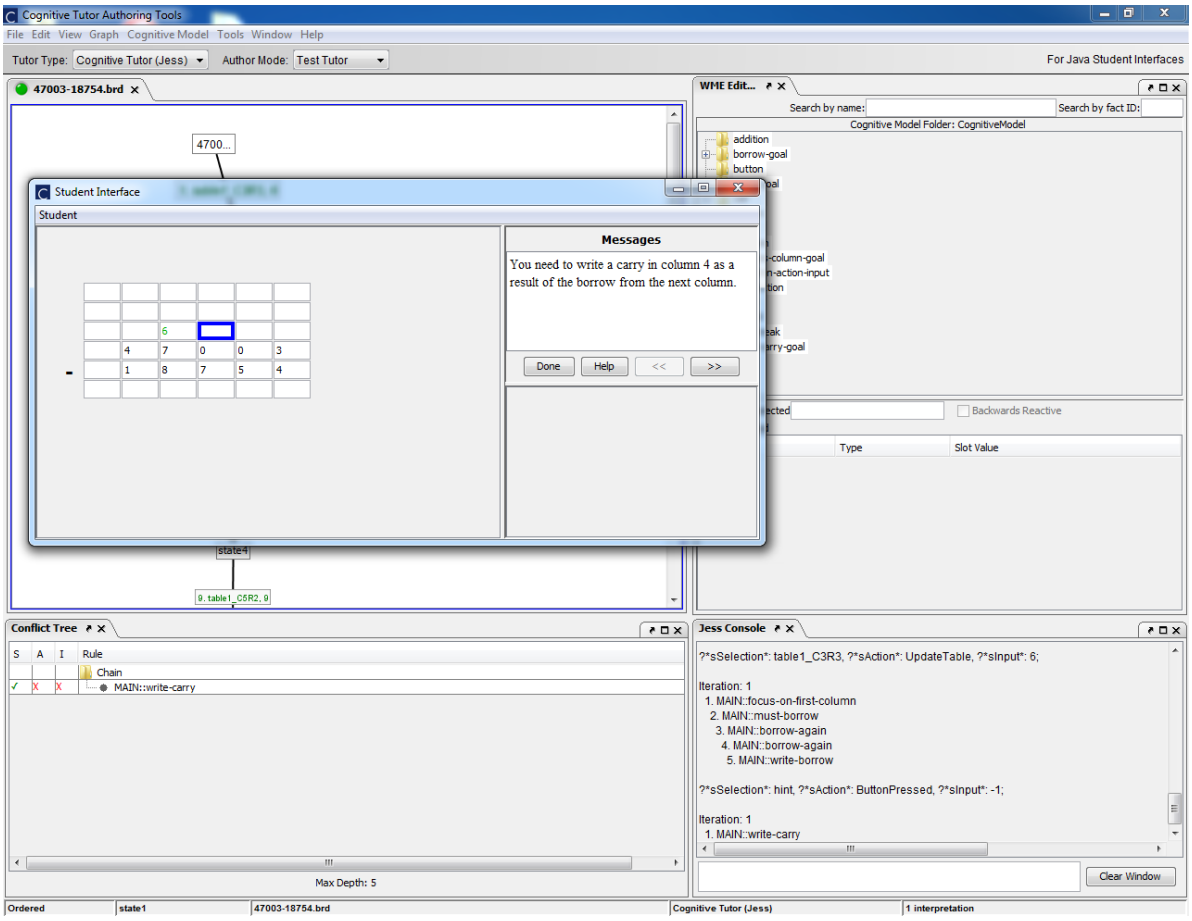

Figure 4 - Un aperçu de CTAT avec l'UI d'un CT pour la soustraction en colonnes.

```
(deftemplate table                [deftemplate cell
  (slot name)                       (slot name)
  (multislot columns)]              [slot value]
                                    (slot row-number)
(deftemplate column                 (slot column-number))
  (slot name)
  [multislot cells]               [deftemplate write-sum-goal
  (slot position)]                  [slot carry]
                                    [slot sum]
                                    [slot column])
```

Figure 5 - Exemples de WME tirés d'un CT pour l'addition en colonnes créé avec CTAT.

```
(defrule add-addends
      ?problem <- [problem (subgoals $?sg1 ?subgoal $?sg2)]
      ?subgoal <- [process-column-goal(carry ?carry)
                  (first-addend ?num1)
                  (second-addend ?num2)
                  (column ?column)
                  (sum nil)]
                  [test (or   (neq ?num1 nil)
                              (neq ?num2 nil)(neq ?carry nil))]
=>
[bind ?sum (+     (if (eq ?num1 nil) then 0 else ?num1)
                  (if (eq ?num2 nil) then 0 else ?num2))]
[bind ?new-sg (assert (write-sum-goal
                        (column ?column) (sum ?sum)(carry ?carry))))
[modify ?problem (subgoals ?new-sg $?sg1 $?sg2)]
[retract ?subgoal]
[construct-message      [You need to add the two digits in this
                        column. Adding ?num1 and ?num2 gives ?sum.]])
```

Figure 6 - Une règle tirée du même CT que la Figure 5 qui modifie seulement la WM.

Un type de WME est défini par un identificateur et des attributs possiblement multivalués qui ne sont pas typés (Figure 5). De plus, un type WME peut étendre un type existant, ce faisant, il hérite de tous les attributs de ce dernier. Les instances de WME qui forment la « *working memory* » (WM) sont généralement propres à une instance de la tâche, mais ils peuvent aussi représenter des constantes propres à la tâche ou au domaine. Chaque instance associe une ou plusieurs valeurs à chacun de ses attributs, les valeurs possibles sont les valeurs primitives traditionnelles (nombres entiers, nombres à virgules flottantes, chaines de caractères et booléens), un pointeur vers une instance, ou encore le pointeur nul.

La partie de gauche d'une règle (LHS pour « *left hand side* ») teste le contenu de la WM pour déclencher conditionnellement la partie de droite (RHS pour « *right hand side* »). Les variables Jess (Figure 6, elles commencent par un `?`) facilitent non seulement l'écriture des tests de la LHS, mais les valeurs qui leur sont appariées sont utilisées dans la RHS pour modifier la WM, en créant, modifiant ou supprimant des instances de WME (Figure 7). Ces modifications provoquent le déclenchement d'autres règles qui modifient à leur tour la WM. Cet enchainement se termine lorsque la RHS d'une règle est associée à une action dans

l'environnement[47]. Généralement, un tel enchainement se produit lorsque des règles utilisent des WME pour représenter un des sous-buts à effectuer. Pour modifier la WM, la RHS a recours à des opérateurs (p. ex. « + ») et des prédicats[48] (p. ex. « `eq` ») fournis par Jess (et d'autres provenant du langage hôte : Java pour CTAT et Lisp pour le TDK).

```
(defrule write-sum
      ?problem <- [problem (subgoals $?sg1 ?subgoal $?sg2)]
      ?subgoal <- [write-sum-goal
                        (sum ?sum&:(and (neq ?sum nil) (< ?sum 10)))
                        (column ?column)
                        (carry nil)]
      ?column <- (column (position ?pos)(cells $? ?result)]
      ?result <- (cell (name ?cell-name)]
=>
      [predict-observable-action ?cell-name "UpdateTable" ?sum]
      [modify ?result (value ?sum)]
      [modify ?problem (subgoals $?sg1 $?sg2)]
      [retract ?subgoal]
      [bind ?col-name (column-name ?pos)]
      [construct-message      [Write ?sum at the
                              bottom of the ?col-name column.]])
```

Figure 7 - Exemple d'une règle tirée du même CT qui produit ou reconnait une action

Chaque règle de la chaine contient une séquence de gabarits de message allant de l'indice délibérément vague jusqu'à une indication précise sur l'action à produire (Figure 7) en passant par une explication du principe sous-jacent[49]. Ces gabarits de message sont instanciés (les variables sont remplacées par leur valeur) lors du suivi de l'apprenant.

1.1.2 L'interaction avec l'environnement

Lorsque l'apprenant interagit avec l'environnement (bouton appuyé, saisie dans une zone texte, sélection dans une liste, etc.), une action, encodée sous la forme d'un triplet nommé SAI (« *Selection-Action-Input* »), est produite [212]. Le premier élément est le composant

[47] On reconnait une telle règle par la présence de la fonction Jess méta « `predict-observable-action` ».
[48] Au sens d'un opérateur dont le résultat est une valeur booléenne.
[49] L'approche « *hint*, *teach, bottom-out* » [254]

graphique manipulé par l'action, le deuxième est un identificateur permettant de distinguer les différentes actions associées à un même composant et le troisième, qui est optionnel, est une valeur (c.-à-d. une ou plusieurs données primitives). Par la suite, l'action est évaluée comme étant correcte ou non (ce processus est décrit dans la prochaine section).

Pour ajuster la granularité des étapes dans les CT, l'auteur doit créer des composants composites (p. ex. une « double liste de sélection » [voir Figure 36 à l'annexe A] qui regroupe deux listes et un bouton permettant de déplacer un ou plusieurs éléments d'une liste à l'autre) qui correspondent à un motif d'interactions présent dans l'environnement d'un CT. Autrement dit, leur rôle est de regrouper en un seul SAI une séquence d'interactions qui serait autrement regroupée en plusieurs SAI distincts.

Une particularité des CT est que l'état de l'environnement est indépendant de l'état de la WM. En effet, la seule forme de synchronisation automatique est la mise à jour de la WM en réponse à la production d'une action. Ce principe de conception s'explique principalement par l'intérêt, présent depuis les débuts des CT, pour les tâches où l'environnement est un logiciel existant [19]. L'absence d'un véritable mécanisme de synchronisation de l'état de l'environnement avec celui de la WM entraine des problèmes. D'abord, il est difficile (c.-à-d. que c'est possible grâce à des astuces[50]) de créer un environnement qui s'adapte aux paramètres de la tâche. Par exemple, dans un MTT pour la manipulation d'ABR, il serait difficile de représenter graphiquement des arbres de tailles différentes. Puis, lorsqu'une action incorrecte est produite dans l'environnement, CTAT rétablit l'état de la WM, mais laisse l'état de l'environnement inchangé. Par conséquent, l'environnement devient parfois bruité ou incohérent au point de rendre impossible toute action correcte (p. ex. un apprenant qui ajoute un nœud superflu ou encore qui supprime un nœud dans un ABR). Pour contourner ce problème, l'auteur peut recourir à une astuce consistant à ajouter du code Java dans la RHS des règles qui rétablit l'état de l'environnement[51].

[50] Par exemple, le composant de type grille utilisé par le CT de soustraction fourni avec CTAT.
[51] On retrouve un exemple dans le CT de logique fourni avec CTAT.

Une autre particularité des CT est de ne pas offrir à l'apprenant un moyen pour annuler une action correcte. D'ailleurs, pour éviter que l'apprenant soit tenté de le faire, CTAT désactive par défaut les composants graphiques utilisés pour produire une action correcte. Pour une tâche comme la résolution d'une équation, l'apprenant qui veut appliquer une réécriture différente (p. ex. parce qu'il en a trouvé une qui l'amène plus près de la solution) doit d'abord être en mesure de revenir à l'équation précédente. L'apprenant pourrait appliquer une réécriture qui inverse celle qu'il vient d'appliquer, mais ce n'est pas idéal du point de vue de la convivialité. Ce n'est pas un hasard si le tuteur du CT pour la résolution d'une équation créé avec le TDK (Figure 8) contient justement un bouton d'annulation. Avec CTAT, l'auteur qui veut offrir un comportement similaire doit ajouter des composants à l'environnement et des règles de production qui reconnaissent les annulations sous la forme d'actions correctes.

1.1.3 Le suivi de l'apprenant et la sélection de tâches

Le processus qui associe une action produite dans l'environnement à une chaine de règles est nommé « *Model Tracing* » (MT), on peut voir ce processus comme un problème élémentaire de « reconnaissance de plan »[52] [255]. Le MT, qui prend en argument le SAI associé à l'action, fait une fouille sur les chaines de règles pour en trouver une qui mène à une action équivalente (c.-à-d. à un SAI équivalent). Lorsque la fouille échoue (aucune chaine n'aboutit à une action équivalente), le test s'arrête et les changements dans la WM sont annulés. Le CT considère que l'action est non tracée et qu'elle correspond à une erreur non pédagogiquement pertinente. Dans ce cas, il affiche un message générique (puisqu'il ne connait pas l'erreur) accompagné d'une rétroaction négative minimale (le composant est surligné en rouge). Si la fouille réussie, le CT intervient avec une rétroaction positive minimale (le composant est surligné en vert) pour confirmer à l'apprenant que l'action est correcte. Lorsqu'une règle de la chaine qui aboutit à une action équivalente est associée à une incompréhension, le CT intervient plutôt en affichant un message associé à la règle (dite invalide) et avec une rétroaction négative minimale (le composant est surligné en rouge). Comme dans le cas de la

[52] « *plan recognition* » [220]

fouille échouée, le CT annule les changements dans la WM. Finalement, il est possible que lors du MT, il y ait plus d'une règle qui peut être déclenchée au même moment, dans ce cas, c'est la plus prioritaire qui est déclenchée[53].

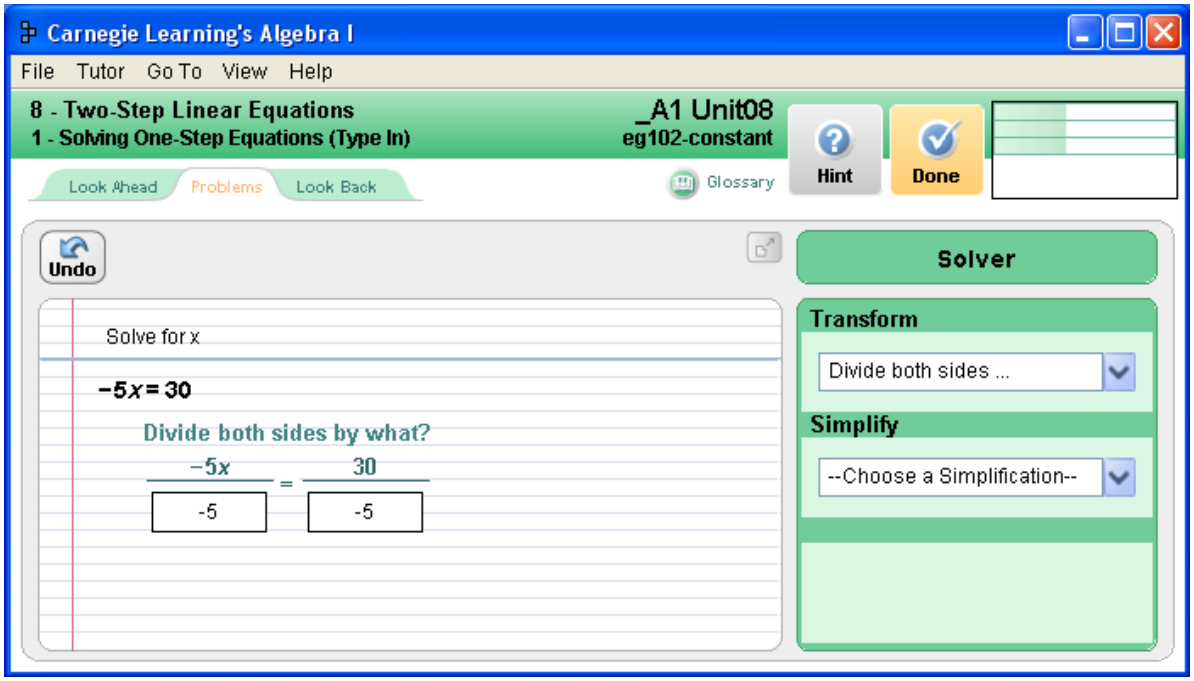

Figure 8 - La tâche de résoudre une équation dans le CT créé avec le TDK.

Puisque les CT déterminent immédiatement si une action est correcte ou incorrecte, il est difficile de modéliser des incompréhensions qui ne peuvent pas être reconnues après une seule étape. C'est-à-dire qu'il faut recourir à une astuce pour modéliser une situation ambigüe où plusieurs chaines produisent des actions équivalentes; par exemple, dans la soustraction en colonnes, plusieurs incompréhensions impliquent une telle situation [197, 256]. Notre analyse [133] d'un CT pour la création d'un nuage de points (Figure 9) créé avec le TDK [30] nous a permis de constater que ce dernier est plus souple à cet égard. Dans ce MTT, à partir d'un tableau de données, l'apprenant doit : trouver la variable dépendante et la variable indépendante, les associer à leur axe respectif, choisir une échelle appropriée pour chacun de

[53] La priorité est définie à l'aide du mécanisme de « *salience* » de Jess.

ces derniers, ajouter les étiquettes correspondantes et finalement créer les points. Or, le choix d'une échelle peut d'abord être correct, mais ensuite incorrect lorsqu'on calcule le nombre d'étiquettes devant être placées sur l'axe (il y a un minimum et un maximum à respecter).

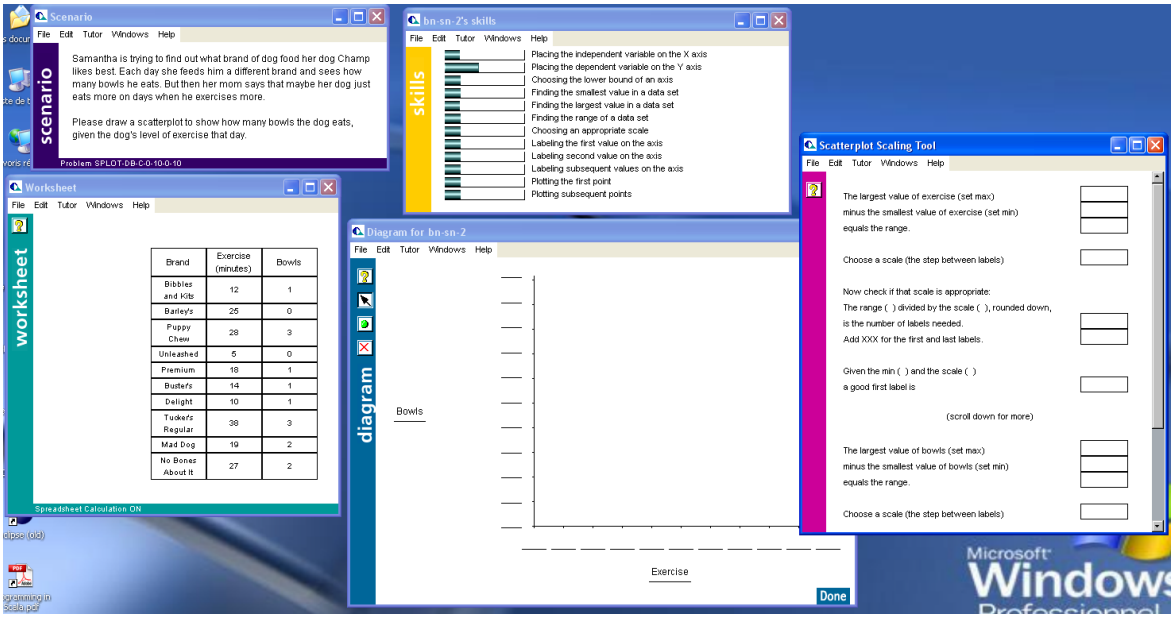

Figure 9 - Un aperçu du CT de nuages de points créé avec le TDK.

Une fois que le processus de MT est terminé pour une action donnée, un deuxième processus, nommé « *Knowledge Tracing* » (KT), est enclenché. Ce processus met à jour la liste des unités de connaissance acquises. Plus précisément, les unités correspondent à des règles qui ont été étiquetées comme objectifs d'apprentissage[54] de façon à obtenir au plus un objectif par chaine. En effet, à l'aide de l'inférence bayésienne, le KT associe à chaque objectif une probabilité (p) d'être atteint et le CT le considère comme atteint lorsque p est près de 1[55]. Par ailleurs, un graphique à barres présente ces probabilités à l'apprenant. Ces dernières sont utilisées à la fois pour le suivi (boucle interne) et pour la sélection de tâches (boucle externe). Pour le suivi, le CT utilise les probabilités pour déterminer, par exemple, s'il exécute les actions qui proviennent de chaines correspondant à des objectifs déjà atteints. Le processus

[54] Correspond aux « *skills* » dans la littérature des CT.
[55] Par défaut, lorsque $p \geq 0{,}95$.

de sélection de tâches des CT est inspiré de l'approche pédagogique nommée « *Mastery Learning* » [35], ce dernier consiste à choisir une tâche qui contient au moins un objectif non atteint, mais qui en minimise le nombre [17].

Le tuteur des CT, en plus d'afficher les messages mentionnés précédemment, offre un bouton d'aide et un bouton de fin. Lorsque l'apprenant appuie sur le premier, le système trouve la chaine de règles (celle qui est prioritaire s'il y en a plusieurs) qui aboutit à une action à partir de l'état courant de la WM, et le tuteur affiche le premier message trouvé à partir du début de la chaine. Si l'apprenant appuie de nouveau, le message suivant est affiché et ainsi de suite jusqu'au dernier. En plus des messages, le composant associé à l'action est surligné en bleu. L'apprenant appuie sur le bouton de fin lorsqu'il croit avoir terminé d'effectuer la tâche, ce que le tuteur confirme ou infirme à l'aide d'un message générique (et en surlignant en vert ou en rouge le bouton de fin). Même si le bouton de fin se retrouve dans le tuteur, il est géré comme une action de l'environnement (c'est-à-dire qu'un SAI est créé et qu'une chaine de règle doit aboutir à ce dernier pour confirmer la fin de la tâche).

1.1.4 Le processus de création

L'utilisation d'un système de production est en principe justifié par sa fidélité cognitive, mais en pratique, il l'est davantage par la flexibilité qu'il procure pour la création du modèle, de même que pour le suivi de l'apprenant et la sélection de tâches. En effet, puisque les règles dans un système de production sont indépendantes les unes des autres, l'auteur n'est pas contraint de les organiser explicitement (comme dans un HTN), il détermine simplement les conditions d'applications qui se retrouvent dans chacune des LHS.

Les modèles des CT créés avec CTAT sont encodés à l'aide de l'IDE[56] libre Eclipse[57] et d'un plugiciel pour Jess. CTAT, contrairement au TDK qui est plus une bibliothèque qu'une plateforme, offre par exemple un GUI[58] pour :

[56] « *Integrated Development Environment* »
[57] http://www.eclipse.org

- observer la mise à jour du modèle lorsqu'une action est produite;
- exécuter le modèle (c.-à-d. produire les actions découlant des chaines de règles obtenues en choisissant toujours la règle la plus prioritaire).

Plus précisément, CTAT offre des outils permettant de manipuler le modèle :

1. le « *WME Editor* » : pour visualiser et modifier directement le contenu de la WM (c.-à-d. les instances de WME);
2. le « *Conflict Tree* » : qui présente graphiquement, sous la forme d'un arbre, les chaines de règles qui sont testées;
3. le « *Why Not Window* » : qui permet pour une règle qui ne s'est pas déclenchée d'observer quels sont les tests de la LHS qui ont échoué.
4. la console Jess qui permet d'effectuer du débogage de plus bas niveau

CTAT offre également une bibliothèque de composants graphiques[59] dédiés à la création de l'environnement d'un CT compatible avec un plugiciel[60] pour Eclipse qui permet de créer visuellement un UI plutôt que d'utiliser un langage de programmation comme Java[61]. Une partie de ces composants correspondent directement à des composants traditionnels (p. ex. un bouton, une zone de texte ou une liste), leur particularité est de supporter le surlignage, les autres sont les composants composites.

Finalement, en fonction de la littérature, de la documentation et de notre expérience, le déroulement de la création d'un CT à l'aide de CTAT est le suivant :

1. créer l'environnement (à l'aide d'Eclipse);

---

[58] « *Graphical User Interface* » que nous utilisons au sens large de façon à pouvoir utiliser UI pour l'interface utilisateur propre à un STI (environnement et tuteur), autrement dit, l'UI est un GUI.
[59] Dérivés de Swing, une bibliothèque pour la création de GUI faisant partie de la bibliothèque standard de Java.
[60] WindowBuilder (http://www.eclipse.org/windowbuilder)
[61] Il y a aussi une version Flash, mais nous ne l'avons pas utilisé.

2. créer les types de WME qui représentent les paramètres de la tâche (à l'aide d'Eclipse), dans notre exemple : *table*, *column* et *cell* en font partie;

3. charger l'environnement dans le GUI de CTAT et créer au moins les instances de WME qui représente une instance de la tâche (à l'aide du « WME editor »);

4. créer (à l'aide d'Eclipse) les règles (incluant les gabarits de messages) et les WME qui représentent l'environnement et les résultats de processus mentaux (incluant les sous-buts);

5. créer suffisamment d'instances de WME représentant les instances de la tâche nécessaire l'atteinte des objectifs d'apprentissage;

6. exécuter et déboguer le modèle au besoin (à l'aide du GUI de CTAT).

1.2 Andes

Andes[62] [95] est un MTT pour la résolution de problèmes[63] de physique de niveau collégial (Figure 11). Créé en 1995, Andes est le fruit d'une coopération entre des chercheurs de University of Pittsburgh et de la US Naval Academy. La stratégie pédagogique d'Andes consiste d'une part à offrir la plus grande flexibilité possible, par exemple en laissant les apprenants saisir librement les équations qui servent à résoudre le problème. D'autre part, elle met davantage l'accent sur l'application des principes du domaine (p. ex. la 2^{e} loi de Newton) que sur la méthode globale de résolution (le nombre et l'ordre des principes appliqués). Même si cette stratégie pédagogique semble contraster avec l'approche des MTT, Andes offre, dans les faits, un suivi similaire à celui offert par les CT. Andes 3 [267], qui est en cours de test bêta, offrira encore plus de flexibilité et sera accessible sur le Web. Il fait suite à deux autres versions : Andes 1 [264] célèbre pour son réseau bayésien dynamique [62] et Andes 2 [263], le plus utilisé et celui décrit dans ce chapitre; mais aussi à deux systèmes

[62] http://www.andestutor.org/
[63] Nous parlons, dans cette section, de *problèmes* plutôt que tâches pour être cohérent avec le lexique couramment utilisé dans l'enseignement du domaine et non pas parce qu'il s'agit de tâches mal définies.

antérieurs : OLAE [142] et POLA [63], qui découlent eux-mêmes des travaux sur le modèle cognitif Cascade [265].

5. Apply Newton's second law and get a= -g*cos(200 deg)
    5.1. Draw a free-body diagram
        5.1.1. Draw the force of gravity, F_w
        5.1.2. Draw the normal force, F_n
        5.1.3. Draw the acceleration, a
    5.2. Apply the fundamental principle: m_c*a_x = F_w_x + F_n_x
    5.3. Project the vectors onto the x-axis
        5.3.1. Apply projection: F_n_x = 0
        5.3.2. Apply projection: F_w_x = F_w*cos(250 deg)
        5.3.3. Apply projection: a_x = -a
    5.4 Apply the weight law, $F_w = m_c$*g

Figure 10 - Le HTN pour la 2e loi de Newton (tirée de [263]).

1.2.1 Le modèle de la tâche

Andes a comme ancêtre le modèle cognitif Cascade qui simule l'apprentissage de la physique par la résolution de problèmes (et l'étude d'exemples), mais à un niveau d'abstraction plus élevé qu'une architecture cognitive comme ACT-R. Pour Andes, résoudre un problème signifie de déterminer la valeur, c'est-à-dire la grandeur physique (p. ex. « 3 m ») associée à une variable (p. ex. « s_x » qui représente le déplacement sur l'axe des x d'un bloc qui descend le long d'un plan incliné). Influencé par cette approche, le SABC d'Andes, combine :

- un modèle encodé directement dans le langage hôte (Lisp), plus particulièrement des HTN (Figure 10) qui reflètent les principes majeurs comme la deuxième loi de Newton (les principes mineurs s'inscrivent au sein de ces derniers sous forme de clauses de Horn[64], p. ex « près de la Terre → $g = 9{,}8\ \mathrm{m/s^2}$ »);

[64] Elles sont grosso modo des règles de production pouvant s'écrire sans disjonctions ni négations.

- un système de calcul formel[65] capable de résoudre le système d'équations associé à un problème, mais aussi à la fois de vérifier les équations entrées librement par les apprenants et les comparer à celles qui sont nécessaires pour résoudre le problème (ce qui s'est révélé être une difficulté majeure) [229].

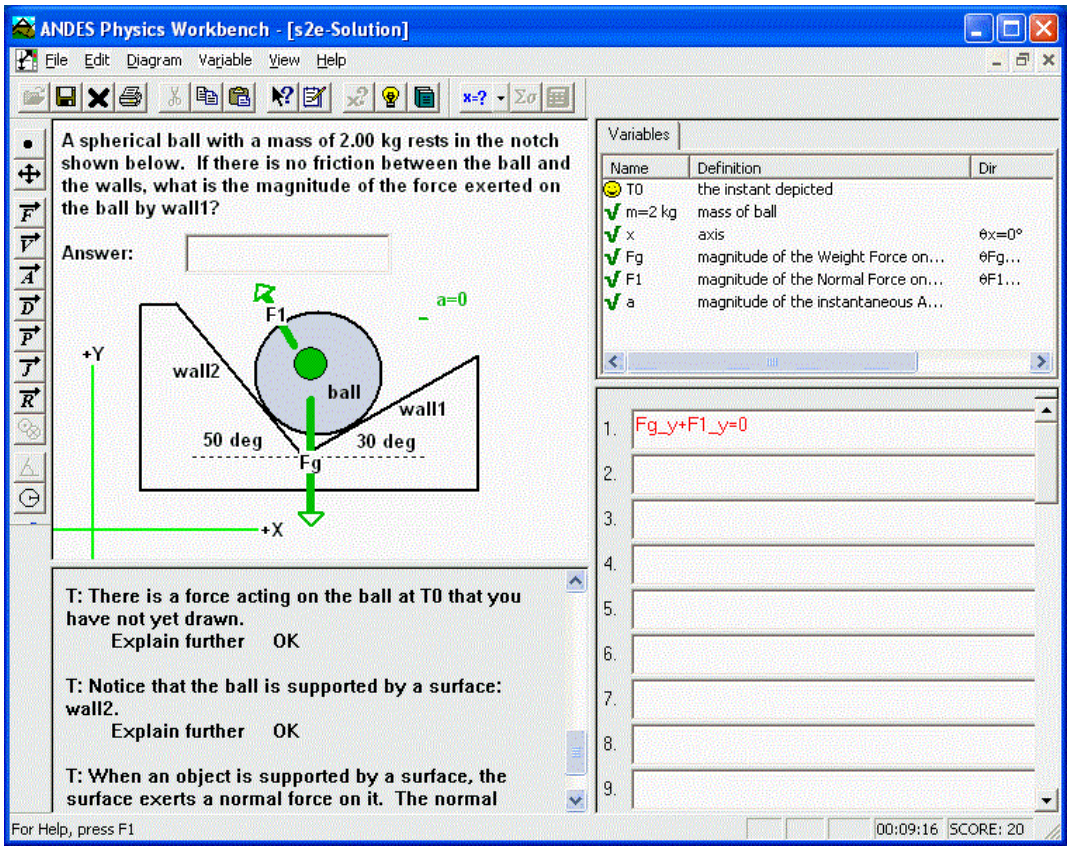

Figure 11 - Un aperçu d'Andes avec une tâche de mécanique.

À chaque HTN est associé des équations qui contiennent des variables (possiblement aussi un système d'axe et des vecteurs). Pour chaque problème, le SABC choisit un HTN dans lequel se retrouve la variable dont la valeur est recherchée, puis il répète le processus récursivement pour chacune des variables introduites par les HTN (et les clauses de Horn) dont la valeur est inconnue. Ce processus produit un système d'équations pour lequel le système de calcul formel trouve une solution, ce qui permet au SABC de construire un graphe représentant les

[65] Développé à l'interne (l'implémentation est en C++).

étapes (partiellement ordonnées) nécessaires pour résoudre le problème. Autrement dit, le graphe connecte les HTN pertinents au problème et établit les valeurs des variables (et les composantes des vecteurs et la position des axes). Du point de vue du suivi, ce graphe représente des équations à saisir, des variables à définir et puisque certaines d'entre elles font référence à des vecteurs ou des axes qui ne sont pas prédéfinis, il représente aussi des étapes qui correspondent à la définition de ces derniers. Andes préconstruit ce graphe, mais les critères de performance qui le justifiaient à l'époque sont désuets.

La fouille effectuée par le SABC est une « boite de verre », en effet, les concepteurs d'Andes ne prétendent pas qu'elle équivaut aux actions mentales d'un apprenant ou d'un expert. Par contre, puisque la stratégie pédagogique d'Andes évite sciemment de contraindre l'apprenant sur sa méthode de résolution, les HTN qui modélisent cognitivement l'application individuelle de chacun des principes sont suffisants. Les concepteurs d'Andes ont par ailleurs conçu un autre MTT, nommé Pyrenees, qui modélise une méthode particulière que l'apprenant doit suivre [265] (nous y revenons au chapitre 2).

Cette approche de représentation des connaissances, bien qu'elle soit propre à la nature de la tâche et du domaine ciblés par Andes, a quand même permis de créer plus de trois-cents problèmes pour différents sous-domaines de la physique (mécanique, électricité et magnétisme). Cette approche a aussi été appliquée dans le domaine des probabilités [53] et on peut supposer qu'elle s'applique dans plusieurs autres domaines où la tâche consiste à déterminer la valeur d'une variable en appliquant des principes à l'aide d'un système d'équations. Par exemple, la stœchiométrie (en chimie) est vraisemblablement un domaine où cette approche s'applique.

1.2.2 L'interaction avec l'environnement

Contrairement aux CT créés à l'aide de la plateforme CTAT présentés dans la section précédente, Andes est un MTT créé de toute pièce et sa gestion des interactions dans l'environnement est propre à la nature de la tâche et du domaine. De plus, son

implémentation n'est pas accessible, par conséquent sa présentation se fait à un niveau d'abstraction plus élevé que celle des CT.

L'environnement d'Andes se divise en quatre cadrans (Figure 11) :

1. la description du problème contenant :

    a. un énoncé qui décrit la variable dont la valeur est recherchée et des grandeurs physiques (sous la forme de constante « une vitesse de 3 m/s » ou de paramètres, par exemple « un angle φ ») devant être associées à des variables;

    b. une zone de texte où l'apprenant peut saisir la valeur de la variable recherchée;

2. le schéma (une image) sur lequel l'apprenant, à l'aide d'outils de dessins et de boites de dialogue, ajoute un système d'axes ou les vecteurs;

3. la liste des variables, qui, selon le problème, contient déjà une ou plusieurs entrées, l'apprenant doit préalablement ajouter, à l'aide d'une boite de dialogue, chaque variable à laquelle il veut faire référence dans une équation;

4. la liste des équations, qui est en fait est une liste de zones de texte dans lesquelles l'apprenant saisit une équation à l'aide de grandeurs physiques, d'opérations mathématiques et des variables.

Autrement dit, il y a quatre types d'actions produites par l'apprenant dans l'environnement : « ajouter un système d'axes ou un vecteur », « définir une variable », « ajouter une équation » et « indiquer la valeur de la variable recherchée ». En fait, pour les deux premiers types d'actions, on retrouve une granularité plus fine sous la forme d'une boite de dialogue où chacun des arguments de l'action est indiqué à l'aide de composants graphiques. Par exemple, lorsque l'apprenant définit une variable, il doit choisir les unités et lorsqu'il définit une variable sur un vecteur, il doit choisir un axe. En somme, ces boites de dialogue remplissent le même rôle que celui des composants composites de CTAT.

Pour indiquer la valeur de la variable recherchée, l'apprenant n'a pas besoin de faire de calcul par lui-même, Andes lui permet d'obtenir directement la solution (une grandeur physique ou un paramètre) d'une équation. Par ailleurs, si l'apprenant utilise la valeur d'une variable au lieu de cette dernière, Andes (plus précisément, le système de calcul formel) exige que celle-ci soit exprimée avec neuf chiffres significatifs si c'est dans une équation et trois chiffres significatifs si c'est la valeur de la variable recherchée. Plutôt que de considérer cette contrainte comme un problème de convivialité, les concepteurs d'Andes y voient un avantage pédagogique (les apprenants évitent d'utiliser directement les valeurs). D'ailleurs, il en est de même de l'exigence de définir les variables avant d'y faire référence dans les équations (ce qui facilite significativement la tâche du système de calcul symbolique).

Lorsque l'action produite par l'apprenant est incorrecte, l'environnement est bloqué jusqu'à ce que cette dernière soit annulée ou corrigée par une action correcte (la fenêtre contenant la variable, l'axe ou le vecteur incorrect ou encore la zone de texte contenant l'équation incorrecte ne sont pas bloqués). Par ailleurs, l'apprenant peut annuler une action correcte (supprime une variable, un axe, une équation, etc.), mais Andes ne produit pas de rétroaction et ne prend pas l'annulation en compte dans ses prochaines interventions.

1.2.3 Le suivi de l'apprenant et la sélection de tâches

Andes offre essentiellement les trois mêmes mécanismes d'interventions que les CT, soit la rétroaction minimale (c.-à-d. surligner les composants graphiques associés à l'action), la rétroaction sur l'erreur et l'indice vers la prochaine étape. Aussi, comme celui des CT, le tuteur d'Andes est constitué d'une zone de texte pour afficher un message et d'un bouton d'aide (pour obtenir un indice), mais il contient en plus un bouton d'explication (« *What's wrong ?* »). Ce dernier est nécessaire puisque la stratégie pédagogique d'Andes est minimalement invasive et par conséquent, Andes n'intervient qu'à la demande de l'apprenant.

La rétroaction minimale ne s'applique pas exactement de la même façon sur tous les types d'actions, le cas le plus simple est celui où l'apprenant produit la valeur de la variable

recherchée. En effet, dans ce cas Andes produit simplement une rétroaction positive ou négative si la valeur indiquée correspond ou non à celle fournie par le système de calcul formel (ce dernier précalcule la valeur de toutes les variables du graphe en fonction de celles connues pour un problème donné). Les cas des axes, des vecteurs et des variables sont gérés de façon similaire, Andes produit une rétroaction positive si ces dernières apparaissent intégralement (p. ex. avec les mêmes unités) dans le graphe ou négative dans le cas contraire. Par contre, dans le cas des équations, Andes est plus flexible : il produit une rétroaction négative seulement si l'équation ajoutée ne peut pas être dérivée algébriquement des équations contenues dans le graphe. Toutefois, Andes avertit l'apprenant (en plus de la rétroaction minimale) lorsque ce dernier ajoute une équation qui n'exprime pas directement l'application d'un principe. Autrement dit, bien qu'Andes tolère que l'apprenant saute par-dessus des manipulations algébriques, il l'encourage à appliquer les principes le plus explicitement possible.

La rétroaction sur l'erreur, mis à part le fait qu'elle est obtenue à la demande de l'apprenant, est en apparence similaire à celle produite par les CT. Contrairement aux CT, pour la produire Andes a recours à un mécanisme de reconnaissance des erreurs (inspiré de WEST [41] et de Proust [110] et comparable à l'approche des CBT). En effet, plutôt que d'exprimer des contraintes qui reconnaissent toutes les actions correctes, le modèle de la tâche n'exprime que les contraintes suffisantes pour reconnaitre les incompréhensions pédagogiquement pertinentes. Cette approche est plus souple que celle des CT, mais elle produit un diagnostic moins précis (ce qui est cohérent avec la stratégie pédagogique d'Andes qui tente d'éviter de surdiagnostiquer les actions mentales de l'apprenant). Une action incorrecte non reconnue par ce mécanisme équivaut à une action incorrecte hors traces dans un CT. Comme avec CTAT, des messages sont associés à chacune des incompréhensions : on retrouve d'abord une indication vague, puis une explication et finalement la description de la correction à effectuer. Ces messages font référence aux principes qui sont pertinents ainsi qu'aux variables qui en découlent.

Pour donner un indice à l'apprenant lorsqu'il appuie sur le bouton d'aide, Andes compare les étapes effectuées par celui-ci avec le graphe. Puisqu'Andes marque chacun des nœuds de ce dernier au fur et à mesure que l'apprenant produit une étape équivalente, il est en mesure de faire une fouille pour déterminer où en est l'apprenant. Si Andes trouve un sous-graphe pour lequel l'apprenant a déjà fait des étapes, il affiche les messages associés à la première étape qui n'a pas été encore effectuée, c'est-à-dire d'abord une indication vague, puis une explication du principe et finalement une description de l'action à faire. S'il ne trouve pas un tel sous-graphe, Andes demande à l'apprenant d'indiquer (à l'aide de boites de dialogue) la variable qu'il recherche et le principe qu'il veut appliquer pour la trouver. Si l'apprenant ne peut répondre, Andes prend le premier résultat de la fouille. Si l'apprenant demande de l'aide supplémentaire, Andes a déjà un sous-graphe pour lequel il y a des étapes à effectuer, il affiche les messages associés à la première d'entre elles. La stratégie pédagogique d'Andes encourage, plus ou moins explicitement, une méthode par chainage arrière qui est privilégiée puisqu'elle a permis d'augmenter les gains d'apprentissage lors d'expérimentations avec Pyrenees [54]. Cette stratégie a remplacé celle d'Andes 1 qui était basée sur un modèle de l'apprenant qui est mis à jour à l'aide d'un réseau bayésien dynamique (cette approche, contrairement au KT, permet, à la suite d'une action, d'évaluer l'atteinte de plus d'un objectif d'apprentissage).

Un tel modèle de l'apprenant a été abandonné par Andes 2 pour deux raisons, d'une part, les expérimentations ont montré que lorsqu'il était utilisé pour produire un indice (plutôt que l'approche basée sur une fouille simple et des questions posées à l'apprenant décrit dans le paragraphe précédent), il semait autant la confusion chez l'apprenant qu'il aidait ce dernier. Cette conclusion semble surprenante à première vue, mais elle s'explique par le fait que le module apprenant est victime de la stratégie pédagogique, qui en maximisant la flexibilité des actions de l'apprenant, rend beaucoup plus difficiles la « reconnaissance de plan » (déterminer la variable recherchée et le principe que l'apprenant veut appliquer) et la prédiction (déterminer la prochaine étape à faire). D'autre part, l'utilisation classique d'un tel modèle est d'optimiser la sélection de tâches pour un apprenant en particulier, mais dans le cas d'Andes, le corps enseignant a plutôt décidé qu'il préférait une liste préétablie de tâches

que tous les apprenants doivent effectuer. C'est pourquoi dans Andes 2 on retrouve simplement un résultat global qui augmente et qui diminue en fonction des actions produites et de l'aide demandée par l'apprenant.

1.2.4 Le processus de création

L'adaptation d'Andes au domaine des probabilités a montré qu'il peut être considéré comme une plateforme pour une partie des domaines où l'approche des MTT s'applique, c'est-à-dire ceux où l'on retrouve des tâches qui consistent à résoudre un problème à l'aide d'un système d'équations qui représente l'application de principes[66]. Par conséquent, il faut distinguer la création de problèmes et la création des éléments de connaissance propre à un nouveau domaine. En effet, la création d'un problème est à la portée du corps enseignant, puisqu'il suffit de définir la variable recherchée et les variables connues[67] (possiblement aussi des axes et des vecteurs).

Dans le cas d'un nouveau domaine, comme dans le cas de la création d'un CT avec CTAT, le corps enseignant doit travailler conjointement avec les auteurs pour :

- encoder les principes sous la forme de HTN et de clauses de Horn;
- modifier l'environnement (p. ex. remplacer les outils pour définir les axes et les vecteurs par ceux définissant un espace de probabilité);
- créer les messages propres au domaine (des messages indépendants du domaine offerts par Andes peuvent être réutilisés);
- créer des problèmes pour vérifier le modèle.

[66] En supposant qu'il existe un équivalent au système de calcul formel d'Andes pour ces domaines.
[67] En supposant que les HTN existants et que le système de calcul formel sont suffisants pour le résoudre.

1.3 Slide Tutor

En principe, Slide Tutor[68] (ST) [72], développé à University of Pittsburgh, est aux tâches de classification visuelle ce qu'Andes est aux problèmes résolus par un système d'équations représentant l'application de principes[69] (Figure 12). En pratique, sa conception reflète le fait qu'il a été conçu dans la perspective du domaine de la médecine. À notre connaissance, ST a été utilisé uniquement dans le cadre d'expérimentations à petite échelle [71], et son développement semble être inactif depuis 2008.

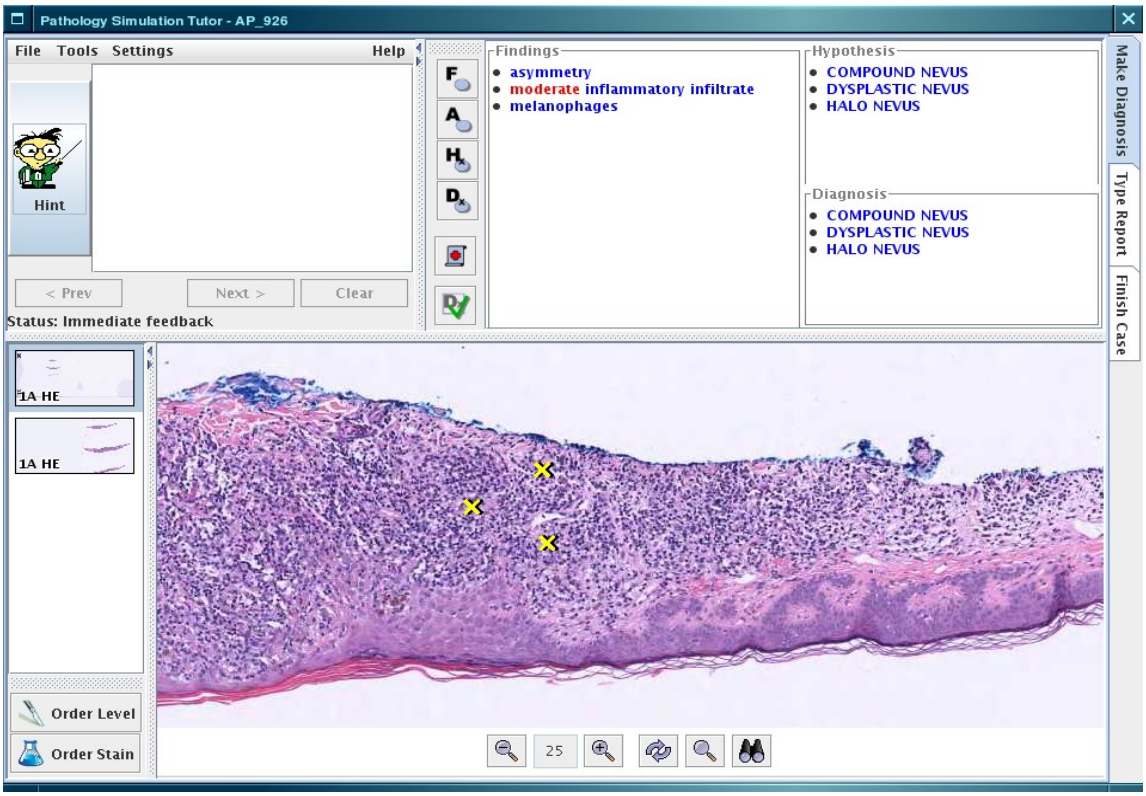

Figure 12 - Un aperçu de Slide Tutor.

[68] http://slidetutor.upmc.edu/
[69] Étonnamment, la littérature ne fait aucune comparaison avec Andes.

1.3.1 Le modèle de la tâche

Le SABC de ST est le résultat d'une analyse cognitive de la tâche consistant à établir un diagnostic à partir d'images médicales. À l'image de celui d'Andes, il a été conçu de façon à supporter plusieurs méthodes pour accomplir la tâche, en particulier afin de guider autant ceux qui procèdent par chainage avant à partir des caractéristiques de l'image et ceux qui procèdent par chainage arrière à partir d'une hypothèse [72].

Le diagnostic médical est un domaine exigeant une quantité imposante de connaissances. En effet, en plus des caractéristiques (p. ex. une masse qui peut être petite ou grosse, ronde ou irrégulière) se retrouvant dans les images, on retrouve des pathologies définies sous forme de motifs de caractéristiques (p. ex. une masse petite ou grosse et irrégulière). Pour chaque cas (une instance d'une tâche), on retrouve donc des caractéristiques particulières (p. ex. une masse petite et irrégulière). Les caractéristiques peuvent être définies qualitativement (comme dans nos exemples) ou quantitativement, dans ce cas les motifs contiennent des plages de valeur.

Pour un cas donné (une instance de la tâche), le SABC de ST construit un graphe qui contient initialement les actions qui correspondent à identifier les caractéristiques présentes sur l'image. Ce graphe est dynamique puisqu'il est mis à jour après chaque action produite par l'apprenant. Pour se faire, le SABC utilise un modèle du processus diagnostic indépendant du domaine d'application (encodé sous forme de règles Jess). Par exemple, lors de l'ajout d'une hypothèse, des nœuds correspondant à l'action d'ajouter des liens entre l'hypothèse et les caractéristiques qui la supportent sont ajoutés au graphe. Un autre exemple est que lorsqu'il y a plus d'une hypothèse et qu'il existe une caractéristique pouvant lever l'ambigüité, un nœud associé à son identification est ajouté au graphe et il est marqué comme prioritaire.

On peut remarquer que cette approche, qui consiste à mettre à jour le graphe au fur et à mesure où l'apprenant produit des actions, s'applique à plus de tâches que celle d'Andes où le graphe est préconstruit. En effet, pour certaines tâches, on évite ainsi l'explosion

combinatoire[70] (d'ailleurs la littérature indique qu'Andes 3 utilise une approche équivalente [267]).

Finalement, même si les concepteurs de ST sont optimistes quant à l'étendue des domaines d'applications possibles (en médecine et même dans d'autres domaines), ils n'ont traité que deux sous-domaines de la dermato-pathologie.

1.3.2 L'interaction avec l'environnement

Comme Andes, ST est un MTT créé de toute pièce et l'implémentation de la gestion des interactions dans l'environnement n'est pas accessible.

L'environnement de ST est divisé en deux zones, la première permet de visualiser l'image et la deuxième réifie le processus de diagnostic à l'aide des actions suivantes :

1. ajouter (et retirer) une caractéristique ou l'absence d'une caractéristique (dans le cas de l'ajout, l'apprenant doit aussi déterminer sa localisation en cliquant sur l'image);
2. raffiner une caractéristique (p. ex. si on a ajouté une « masse », on peut ensuite préciser qu'il s'agit d'une « petite masse »);
3. ajouter (et retirer) une hypothèse;
4. ajouter un lien qui supporte (ou qui réfute) une hypothèse à partir d'une caractéristique (ou son absence);
5. ajouter (et retirer) un diagnostic;

Pour toutes les actions d'ajouts (et celles permettant de raffiner une caractéristique), une boite de dialogue permet de faire une sélection. Le résultat de ces actions est représenté

[70] En fait, Andes évite l'explosion combinatoire dans plusieurs cas en regroupant les équations dans des classes d'équivalences (une approche qui n'a pas nécessairement de correspondance dans tous les domaines).

graphiquement dans l'environnement, permettant d'effectuer les autres actions en manipulant ces représentations graphiques.

Comme avec Andes, lorsque l'apprenant produit une action incorrecte, l'environnement est bloqué jusqu'à ce que cette dernière soit annulée (p. ex. retirer une caractéristique) ou corrigée par une action correcte (p. ex. raffiner différemment une caractéristique). De même, l'apprenant peut annuler une action correcte, mais ST n'en tient pas compte explicitement.

1.3.3 Le suivi de l'apprenant et la sélection de tâches

Comme dans Andes, la liste de tâches est définie par le corps enseignant. Le suivi de l'apprenant est à première vue similaire à celui d'Andes, leurs stratégies pédagogiques sont semblables, mais en pratique il s'approche de celui des CT. En effet, au-delà du fait que ST est moins achevé qu'Andes, une autre raison plus importante explique la différence dans le suivi : celui de ST est jusqu'à un certain point indépendant du domaine, c'est-à-dire que ses concepteurs souhaitent qu'il soit applicable pour tous les domaines où l'on retrouve des tâches de classification visuelle. Par conséquent, il est plus indépendant de la didactique de son principal domaine d'application (diagnostic médical) qu'Andes (physique).

En effet, d'une part les messages associés aux actions ne font référence au domaine qu'au travers des connaissances didactiques contenues dans le modèle, par exemple les descriptions en langage naturel des caractéristiques. Comme Andes, ST utilise son graphe pour choisir sur quelle action il donne un message, mais dans les cas ambigus (dans ceux où Andes communique directement avec l'apprenant), ST utilise simplement des priorités associées aux actions. D'autre part, le mécanisme de reconnaissance des erreurs est lui aussi indépendant du domaine, il reconnait des erreurs pertinentes à toutes les tâches de classification visuelle comme identifier une caractéristique qui ne se trouve pas réellement sur l'image ou encore omettre d'identifier une caractéristique qui s'y trouve. Les messages associés à ces erreurs réfèrent aussi aux connaissances didactiques, par exemple la description en langage naturel des actions.

Par ailleurs, lors de la rétroaction minimale, en plus de surligner les composants graphiques qui réifient le processus de diagnostic, ST surligne également sur l'image le « rectangle englobant »[71] de la caractéristique associée à l'action (lorsqu'il y en a une).

1.3.4 Le processus de création

La création d'un cas est facilitée par l'utilisation de l'éditeur d'ontologies Protégé[72] pour lequel on retrouve un plugiciel Jess que les concepteurs de ST ont modifié pour générer les faits (les `fact`) équivalents et pour vérifier les cas créés (c'est-à-dire vérifier qu'il y a bel et bien un diagnostic cohérent avec les caractéristiques de l'image). Selon les concepteurs de ST, le comportement par défaut du suivi (lui aussi encodé à l'aide de Jess) est facilement modifiable pour ajuster la stratégie pédagogique ou encore l'adapter à un autre domaine, mais aucun exemple n'est fourni en ce sens.

1.4 Points saillants

- Même à l'intérieur des MTT, le point de vue avec lequel l'auteur modélise de la tâche varie. Contrairement aux CT, l'auteur dans Andes/ST n'adopte pas le point de vue de l'apprenant, mais un point de vue externe. Ce point de vue plus objectif facilite un suivi de l'apprenant basé sur une évaluation plus souple.

- Dans CTAT, la WM reflète l'état de l'UI, tandis que dans Andes/ST, c'est le contraire, ce qui est cohérent avec leur point de vue respectif du modèle de la tâche.

- En tenant compte des propriétés du domaine et de la tâche, ces systèmes peuvent offrir à l'apprenant d'annuler une action grâce à une forme « d'édition sur place ».

- Le processus de création de ces systèmes est similaire, même si Andes et ST ne sont pas génériques comme les CT.

[71] « *bounding box* »
[72] http://protege.stanford.edu/

Chapitre 2

La conception d'Astus

Astus est notre interprétation[73] libre d'« Usager virtuel »[74] (UV), une spécification qui propose d'appliquer les résultats des travaux sur l'architecture cognitive Miace au contexte des STI. Les travaux antérieurs du laboratoire ASTUS, qui découlent d'UV, ont aussi influencé la conception d'Astus. En effet, bien que ces travaux se déroulaient dans un cadre plus large que celui d'une plateforme de MTT comme Astus, ils offraient une première interprétation d'UV sous la forme du MGC[75]. En fait, c'est notre perspective d'utilisateur du MGC, dans le contexte d'une initiation à la modélisation cognitive[76], qui nous a donné le recul nécessaire pour tirer d'UV plusieurs des caractéristiques d'Astus et pour se dégager de celles qui étaient pertinentes pour une architecture cognitive, mais non pour une plateforme de MTT.

Nous évitons de décrire directement Miace, UV, le MGC et une version préliminaire d'Astus développée au cours d'un projet de maitrise [132] pour éviter de confondre le lecteur. En effet, même si tous ces travaux ont un glossaire commun, les éléments de celui-ci ont des significations différentes, il serait donc difficile pour le lecteur de s'y retrouver.

Ce chapitre nous permet d'atteindre notre premier objectif, soit d'établir les caractéristiques que nous souhaitons retrouver dans Astus et celles que nous désirons éviter afin de répondre aux exigences énoncées en introduction. Ce chapitre complète également l'état de l'art, en

[73] Avec la collaboration de M. Fortin, A. Abdessemed, L. Paquette, G. Beaulieu, A. Champeau et P. Moncuquet.
[74] Mayers, A. (2001). Rapport interne non publié.
[75] Le « module de gestion des connaissances » qui devait agir comme le module expert d'une plateforme de STI répondant à des exigences de loin plus ambitieuses que celles d'Astus, mais qui n'a jamais été développée [88].
[76] Une simulation de l'interruption et de la reprise de l'exécution d'une recette d'un gâteau Tiramisu.

situant Astus par rapport aux travaux antérieurs (Miace, UV, MGC) et aux systèmes décrits au chapitre 1 (les CT[77] et Andes/ST).

La description des caractéristiques qui suit est délibérément abstraite afin d'être neutre par rapport à nos choix particuliers tant au niveau conceptuel qu'au niveau de l'implémentation; ces choix sont décrits et illustrés aux chapitres 3 et 4.

2.1 Le modèle de la tâche

Comme UV, Astus est basée sur un modèle de la tâche dont les éléments de connaissance sont divisés en trois catégories : les éléments sémantiques, procéduraux et épisodiques. Le vocable « épisodique » fait référence aux travaux en psychologie cognitive qui désigne avec ce dernier la partie autobiographique de la mémoire déclarative [251] (qui est elle-même une partie de la mémoire à long terme qui contient aussi la mémoire procédurale) [26]. En effet, comme les éléments sémantiques, les éléments épisodiques représentent des connaissances déclaratives, mais contrairement aux éléments sémantiques qui représentent des connaissances propres au domaine où à la tâche, les éléments épisodiques sont propres à une exécution particulière de la tâche.

Si l'on compare les CT à Astus, les WME correspondent aux éléments sémantiques (les WME représentant le résultat de processus mentaux comme les sous-buts correspondent aux éléments épisodiques) et les règles correspondent aux éléments procéduraux. Si l'on compare avec Andes/ST, les HTN correspondent aux éléments procéduraux, tandis que les variables, les valeurs que prennent ces dernières, les équations, les vecteurs et les axes correspondent aux éléments sémantiques. Quant au graphe qui connecte les HTN, il correspond grosso modo aux éléments épisodiques. Ces comparaisons sont inexactes, car elles ne reflètent que les similitudes entre les différents modèles; dans la suite de cette section, nous précisons les différences entre les éléments de connaissance d'Astus et ceux proposés par UV ainsi que ceux des autres MTT.

[77] Par la suite, nous désignons par CT, les MTT créés à l'aide de CTAT, tels qu'ils sont décrits au chapitre 1.

Le paradigme[78] dans lequel l'auteur s'inscrit pour créer le modèle avec Astus permet de distinguer Astus des systèmes mentionnés précédemment. Avec Andes/ST et TOTS/Steve, l'auteur s'inscrit dans le **paradigme de l'expert** (ou du domaine), c'est-à-dire que le modèle représente les connaissances le plus objectivement possible. Avec CT et UV, l'auteur s'inscrit dans le **paradigme de l'apprenant**, c'est-à-dire que le modèle représente les connaissances de l'apprenant novice à l'apprenant « idéal ». Avec Astus, l'auteur s'inscrit dans le **paradigme du tuteur** [133], c'est-à-dire que le modèle représente une abstraction et une généralisation des instructions produites par un tuteur humain. Autrement dit, un modèle dans Astus intègre des éléments didactiques qu'on retrouve normalement sous une forme implicite (p. ex. dans les gabarits de messages ou encore dans le processus de suivi) sous la forme d'éléments de connaissance pouvant être interprétés par le module pédagogue.

Le paradigme du tuteur est à la base plus près de celui de l'apprenant puisque contrairement à celui de l'expert, les deux premiers sont associés à une évaluation plus serrée. La différence entre le paradigme du tuteur et celui de l'apprenant est que le premier offre un point de référence externe pour l'évaluation (à cet égard, le paradigme du tuteur se rapproche du paradigme de l'expert). En effet, dans le paradigme du tuteur, le pédagogue peut positionner l'action de l'apprenant par rapport aux instructions qui sont reflétées par le modèle (dans le paradigme de l'expert, l'action est positionnée par rapport aux principes du domaine). Au chapitre 5, nous indiquons comment les MTT d'Astus, créés en s'inscrivant explicitement dans le paradigme du tuteur, mettent en évidence les avantages et les inconvénients des MTT.

2.1.1 Les connaissances procédurales

Comme dans UV et dans la plateforme TOTS, les éléments procéduraux d'Astus forment un HTN; Astus hérite de la nomenclature d'UV, c'est-à-dire qu'un HTN est constitué de buts (intentions[79]), de procédures complexes (plans ou méthodes[80]) et de procédures primitives

[78] Au sens d'un « *ontological commitment* » [74].
[79] Dans la littérature des agents BDI, en planification, on parle plutôt de tâches et de sous-tâches.

(actions). Il est important de ne pas confondre le sens d'un but dans les HTN, c'est-à-dire une intention, avec celui qu'on retrouve plus généralement en IA, c'est-à-dire un état[81]. Puisque satisfaire une intention (p. ex. « aller à Montréal ») peut aussi, plus ou moins directement, permettre l'atteinte d'un état (p. ex. « être à Montréal »), les buts représentant des intentions sont plus expressifs que les buts représentant des états (p. ex. il est difficile à l'aide d'états seulement d'exprimer « faire l'aller-retour Montréal – Sherbrooke ») [96]. Fait à noter, bien que nous présentons les buts avec les connaissances procédurales pour alléger le texte, ils sont en fait des éléments de connaissance sémantique (nous y revenons au chapitre 4).

Les HTN, comme les systèmes de production facilitent l'obtention d'un comportement non déterministe (c'est-à-dire la fouille de toutes les actions possibles dans l'état courant en fonction de l'état initial et des actions passées) qui est nécessaire pour le suivi de l'apprenant propre aux MTT [255].

Contrairement aux HTN d'Andes, ceux d'Astus (comme ceux de TOTS et d'UV) correspondent à la réalisation complète d'une tâche; autrement dit, ils sont autosuffisants, c'est-à-dire qu'ils peuvent être interprétés directement, sans passer par un SABC comme celui d'Andes. Ces HTN équivalent donc à des systèmes de production hiérarchiques, c'est-à-dire des systèmes de production où la LHS des règles contient un but[82]; un exemple de système de production hiérarchique est l'architecture cognitive ACT-R (ce qui n'est pas nécessairement avec les CT, comme nous l'avons indiqué au chapitre 1).

Avant d'examiner les avantages de tels HTN dans le contexte des MTT, il est important de considérer qu'il y a deux conditions à remplir pour qu'un tel HTN existe et qu'il soit suffisant pour les besoins du module expert. Premièrement, la tâche doit être bien définie, et deuxièmement, l'évaluation de l'apprenant doit être serrée, c'est-à-dire comparable à celle que l'on retrouve dans les MTT, et dans les CT en particulier (considérant qu'Andes/ST est

[80] Dans la littérature, quand on distingue méthodes et plans, c'est que les premiers génèrent les derniers en fonction d'une instance de la tâche. Or, puisqu'on retrouve également dans la littérature les notions de plans conditionnels [201] et itératifs [136], les deux termes sont, en pratique, équivalents.
[81] Dans la littérature des agents BDI, on parle respectivement de « *perform goals* » et de « *achieve goals* » [73].
[82] De plus, la RHS d'une règle ne doit pas pouvoir à la fois introduire des sous-buts et modifier la WM/KB.

un exemple de MTT qui offre une évaluation plus souple). Les travaux sur Steve, qui découlent de TOTS, ont exploré la possibilité de réduire cette limitation, nous y revenons au chapitre 5.

Pyrenees [54], contrairement à Andes, guide l'apprenant selon une méthode de résolution particulière qui correspond à un HTN tel que défini précédemment. En effet, la méthode appliquée par Pyrenees consiste à choisir le prochain principe à appliquer en observant l'équation qui en découle pour voir si elle permet de trouver la valeur d'une variable inconnue (préférablement celle qui est recherchée dans le problème) en évitant de faire apparaitre trop de nouvelles variables inconnues.

On peut remarquer que le modèle de Pyrenees semble s'inscrire dans le paradigme du tuteur. Or, il ne supporte qu'une stratégie qui consiste à choisir le prochain principe à appliquer exclusivement en fonction de connaissances procédurales supposées maitrisées (c.-à-d. résoudre un système d'équations) plutôt qu'à l'aide d'euristiques exprimées à l'aide de connaissances de la physique ou des probabilités. Par conséquent, nous considérons que Pyrenees s'inscrit minimalement et implicitement dans le paradigme du tuteur (à défaut de considérer qu'il s'inscrit dans le paradigme de l'apprenant).

Dans le cas d'Astus, on parle plutôt d'un graphe procédural d'Astus (un DAG[83]) plutôt que d'un HTN. Une telle structure permet de regrouper les tâches (autrement dit, les buts racines correspondants) qui partagent des sous-tâches (autrement dit, des sous-buts).

De façon à s'inscrire dans le paradigme du tuteur, le graphe procédural d'Astus ajoute à la structure classique des HTN des structures de contrôle explicites (séquence, sélection et itération) qui correspondent aux instructions abstraites et généralisées (Figure 13). Bien qu'UV propose d'ajouter des structures de contrôle aux scripts [84] qui définissent les procédures complexes, le rôle de ces structures n'était pas de supporter explicitement le paradigme du tuteur. Il est important de considérer d'une part que si on s'inscrit dans le

[83] « *Directed Acyclic Graph* »
[84] Exprimés dans un langage semblable au Lisp.

paradigme de l'expert, ces structures peuvent être considérées comme du sucre syntaxique. En effet, l'avantage d'une telle structure est son expressivité[85]. D'autre part, si on s'inscrit dans le paradigme de l'apprenant, ces structures peuvent être embarrassantes puisqu'elles ne correspondent pas nécessairement aux processus mentaux de l'apprenant.

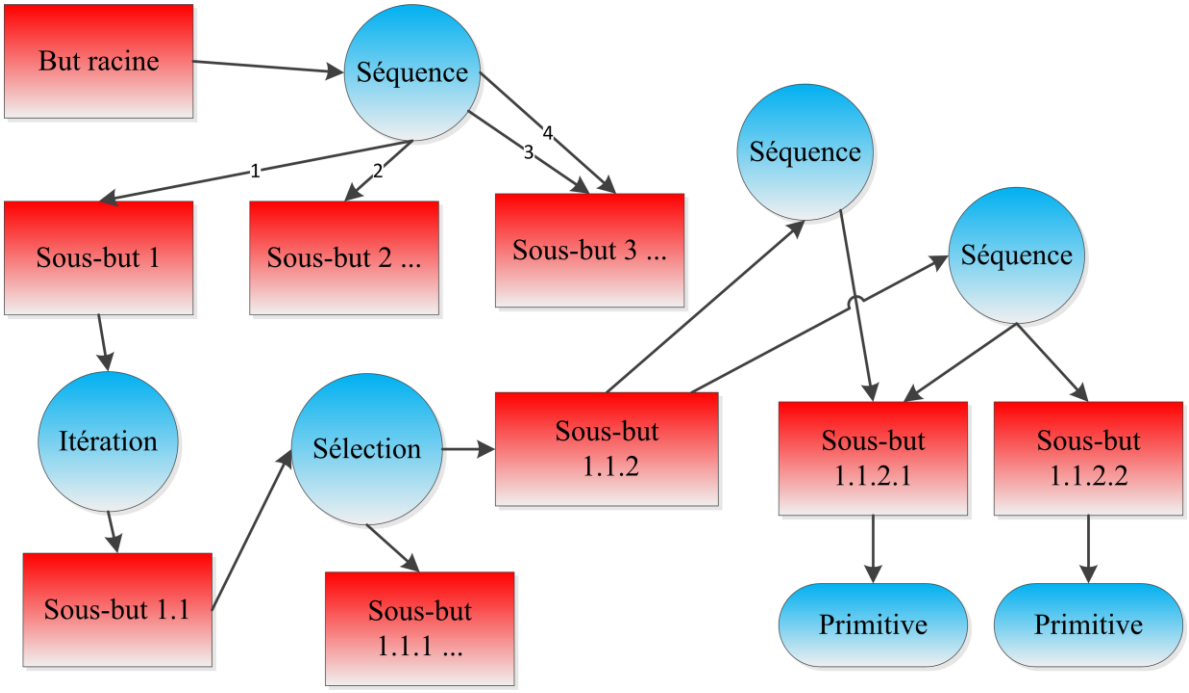

Figure 13 - Un graphe procédural abstrait (les sous-buts 2, 3 et 1.1.1 ne sont pas développés pour alléger la figure).

En effet bien que de nombreux travaux en psychologie cognitive [11, 64, 65, 66, 183, 186] appuient la hiérarchisation des connaissances procédurales (comme les HTN classiques ou les systèmes de production hiérarchiques), très peu, à l'exception de Sierra [86] [256, 257] introduisent de telles structures de contrôle (Sierra propose un modèle cognitif de la soustraction en colonnes où la présence d'une structure d'itération bornée est justifiée parce qu'elle explique des erreurs commises par des apprenants lors d'études empiriques). Par

[85] En pratique, puisqu'en principe les HTN et les systèmes de production sont équivalents (Turing-complet).
[86] Sierra est dérivé de « Repair Theory » [44] qui découle des travaux sur IDEBUGGY et ses prédécesseurs DEBUGGY et BUGGY [42, 48] et de « Step Theory » [258].

conséquent, c'est en se plaçant dans le paradigme du tuteur que l'ajout de structures de contrôle prend tout son sens. En effet, l'ajout de ces structures permet de générer des interventions plus précises (nous y revenons au chapitre 4) que la structure classique des HTN, que l'on retrouve dans TOTS, qui ont deux structures de contrôle élémentaires[87] : le « OU » (choix arbitraire d'une des procédures pour satisfaire un but) et le « ET » (plusieurs sous-buts à satisfaire pour compléter une procédure et satisfaire son but)[88].

2.1.2 Les connaissances épisodiques

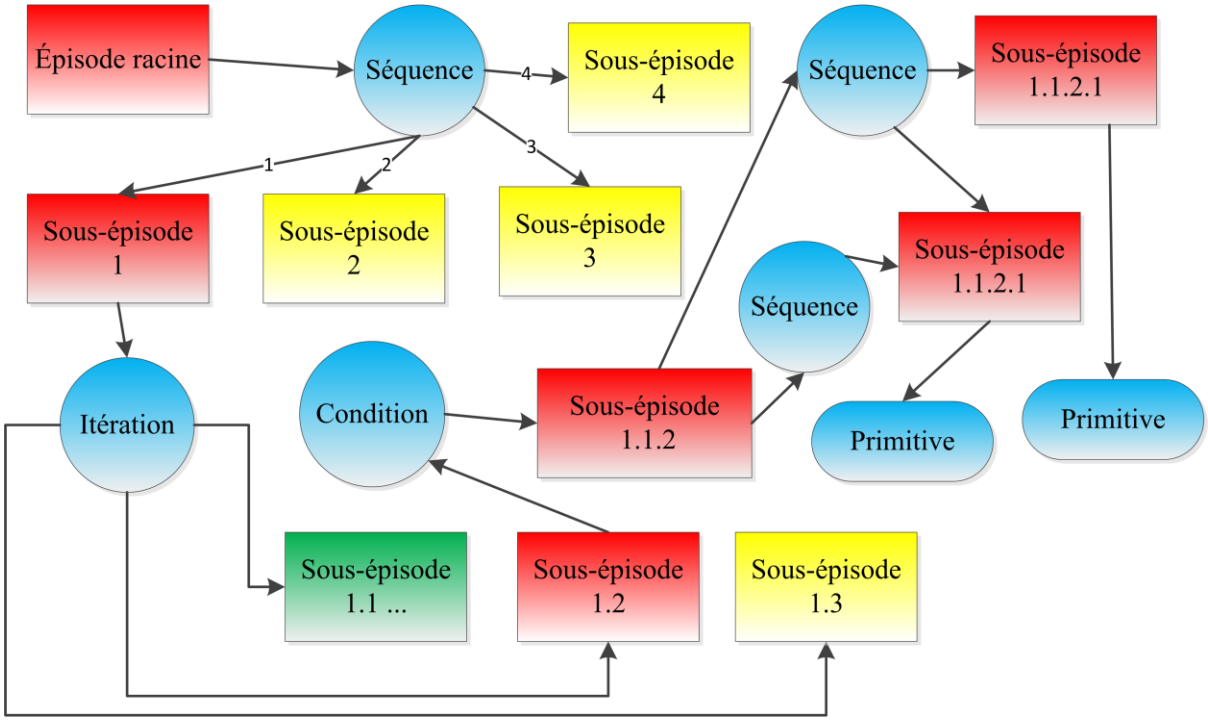

Figure 14 - Un graphe épisodique qui correspond au graphe procédural précédent (Figure 13). On peut noter qu'il y a plusieurs sous-épisodes sous l'itération et un seul sous la condition (les sous-épisodes 2, 3, 1.2 et 1.3 sont inactifs).

[87] En fait, les HTN utilisés en planification offrent également un mécanisme de préconditions qui permet de définir des contraintes d'ordre et des variables indicées qui permettent d'obtenir un comportement itératif.
[88] Ce qui équivaut à avoir plusieurs règles de production qui consomment le même sous-but dans la LHS et d'avoir une règle de production qui produit plusieurs sous-buts dans sa RHS.

L'application du graphe procédural à une instance d'une tâche donne lieu à la création d'un **graphe épisodique**[89] (un DAG). Ce dernier unifie dans une même structure trois catégories d'épisodes, où un épisode correspond à une instance de but et aux instances de procédures qui peuvent satisfaire ce dernier :

- ceux qui correspondent à la mémoire épisodique (les épisodes satisfaits, même s'ils sont par la suite annulés parce que toutes les actions sous-jacentes ont été annulées);
- ceux qui forment, selon la psychologie cognitive, la mémoire de travail (les épisodes pour lesquels une instance de procédure est amorcée) [29];
- ceux qui permettent au système d'anticiper les actions potentielles de l'apprenant (les épisodes pour lesquels aucune instance de procédure n'est encore amorcée).

Le graphe épisodique (voir Figure 14) est donc formé d'un nœud « racine » qui correspond à l'épisode contenant une instance du but racine et dont les nœuds « feuilles » correspondent aux épisodes pouvant être satisfaits par des instances des procédures primitives; entre les deux, on retrouve les nœuds correspondants aux sous-épisodes intermédiaires formés par des instances de buts pouvant être satisfaits par des instances des procédures complexes.

Contrairement à Andes et similairement à ST, le graphe épisodique est généré au fur et à mesure qu'une tâche est effectuée, ce qui constitue un prérequis pour supporter des tâches pour lesquelles la génération d'un graphe au préalable entrainerait une explosion combinatoire. En effet, puisque le graphe ne contient que les actions possibles dans l'état courant, les choix (p. ex. celui d'une procédure pour satisfaire un but) inférés à partir des actions produites permettent d'élaguer les branches du graphe qu'il n'est plus nécessaire d'explorer. Puisque le graphe épisodique est généré de façon descendante à partir des épisodes actifs plutôt que façon ascendante à partir d'une action produite, à l'image des CT, certaines tâches peuvent causer une explosion combinatoire (dans le cas où pour obtenir les

[89] Il s'agit d'un graphe et non d'un arbre lorsqu'on unifie au préalable les épisodes équivalents; omettre de le faire reviendrait à traiter les ambigüités lorsqu'une action est produite et qu'on tente de mettre l'arbre à jour.

prochaines actions il y a de nombreux choix à explorer, et ce à plusieurs niveaux différents). Au chapitre 4, nous indiquons comment ce problème peut généralement être évité.

2.1.3 Les connaissances sémantiques

De façon générale, les éléments de connaissance sémantiques (comme les WME dans les CT) remplissent deux fonctions complémentaires, d'une part décrire l'environnement dans lequel s'effectue la tâche. D'autre part, préciser comment cette dernière doit être effectuée, c'est-à-dire en ajoutant des variables qui précisent les buts (ces dernières correspondent aux variables Jess appariées aux WME dans la LHS des règles). Comme plusieurs autres types de SABC, incluant les cadres[90], le « modèle entité-association étendu » et les logiques de description (DL pour « *description logics* »), Astus a recours à :

1. un modèle contenant des éléments propres au domaine, à une tâche ou bien à une instance d'une tâche (qui se réduit à la logique du premier ordre [FOL[91]]);

2. un métamodèle qui fournit d'une part des structures comme les concepts et les objets[92] (qui sont respectivement similaires aux cadres et à leurs instances) et d'autre part des éléments de connaissance générale, c'est-à-dire indépendants de la plupart des domaines dont les tâches sont tirées (p. ex. les nombres entiers[93]);

3. un langage de requête et de manipulation (à l'image de SQL[94] ou SPARQL[95]) qui permet :

    - d'exprimer respectivement type et la valeur des variables dans les graphes procéduraux et épisodiques;

[90] Plus précisément, les « *Frame* » classiques qui précèdent les travaux sur KL-ONE [40] et dont on retrouvait une implémentation dans Protégé jusqu'en 2012 (http://protegewiki.stanford.edu/wiki/Protege-Frames).
[91] « *First-Order Logic* »
[92] À ne pas confondre avec les objets en programmation orientée objet.
[93] Si la tâche consiste à effectuer des manipulations élémentaires sur les nombres entiers, ceux-ci doivent être modélisés sous la forme d'éléments syntaxiques du domaine.
[94] Quoique le modèle d'Astus ne correspond pas tout à fait au modèle relationnel.
[95] http://www.w3.org/TR/sparql11-overview/ (quoi que le modèle n'Astus n'est pas un « *triplestore* »).

- d'accéder, au sein des différents modules, aux éléments sémantiques[96] de façon indépendante de la tâche et du domaine (p. ex. pour générer des interventions);

- de manipuler les éléments de connaissance sémantiques (commandes CRUD[97]).

Tous les systèmes mentionnés précédemment dans ce chapitre ont recours à ces trois éléments, mais Astus se distingue en les rendant plus explicites et en les développant de façon à renforcer le paradigme du tuteur. Dans plusieurs de ces systèmes, ces éléments sont plus ou moins implicites, entres autres parce qu'ils se confondent avec le langage hôte.

Dans Astus, comme dans les CT, les éléments sémantiques du métamodèle ne sont pas utilisés pour représenter des principes du domaine[98] ou des instructions. La différence est qu'avec Astus, ces éléments ont été conçus en fonction des rôles qu'ils jouent et qu'ils sont interprétés comme tel par le module pédagogue. Avec les CT, il reste une certaine ambigüité sur le rôle des WME, en effet, ceux-ci ne sont pas positionnés clairement par rapport aux chunks d'ACT-R dont ils découlent (tel que nous l'avons évoqué au chapitre 1, les *chunks* ont justement un rôle plus étendu).

Nous décrivons le rôle des éléments sémantiques et du langage de manipulation davantage dans la prochaine section (2.2), la section courante se conclut avec la description des rôles du métamodèle et du langage de requête.

Dans les CT, on retrouve un métamodèle, formé des WME et des types primitifs Jess/Java, qui est manipulé directement à l'aide de Jess/Java (p. ex. `assert`, `retract`). Dans Andes/ST, le métamodèle est plus sophistiqué que dans les CT, mais il dépend de la nature de la tâche et du domaine. UV propose un métamodèle, à la base similaire à celui des CT, c'est-à-dire une structure unique équivalente aux chunks d'ACT-R, mais il développe à partir de

[96] Et aussi les buts et leurs instances (voir chapitre 4).
[97] « *Create, Read, Update and Delete* »
[98] En fait, des principes du domaine peuvent être représentés sous forme de constantes manipulées par le graphe procédural (p. ex. un principe qui est une variable du but correspondant à l'application d'une réécriture).

cette dernière un métamodèle comportant plusieurs notions de mathématiques et de logique, comme les notions de nombres, d'ensembles, de définitions et de preuves.

Il peut paraitre étonnant qu'UV, qui s'inscrit dans le paradigme de l'apprenant, propose un tel métamodèle. En effet, ce dernier est aussi, voire plus difficile à justifier d'un point de vue cognitif, que les structures de contrôle au niveau procédural [88, 172]. Or, UV justifie son métamodèle de façon pragmatique, un métamodèle sophistiqué réduit les efforts de création pour plusieurs domaines, mais surtout, un tel métamodèle permet d'enrichir le comportement des autres modules. Cependant, UV reste flou quant à l'usage du métamodèle par ces derniers [88].

Le métamodèle d'Astus se justifie d'une part de la même façon que le fait UV. D'autre part, il se justifie pour les mêmes raisons que les structures de contrôle au niveau procédural, c'est-à-dire parce qu'elles supportent le paradigme du tuteur. Astus, par rapport à UV, élimine certaines notions du métamodèle (p. ex. celles de définitions et de preuves), mais en inclut d'autres (p. ex. celles de collections, de relations d'équivalence et de relations d'ordre) qui supportent le paradigme du tuteur. En particulier, des notions qui permettent à l'auteur d'exprimer plus directement son intention, ce qui facilite entre autres la génération des interventions par le module pédagogue. Au chapitre 4, nous indiquons comment Astus reprend la notion de structure unique telle qu'on la retrouve dans UV et les CT.

UV propose d'être pragmatique et de remplacer les buts qui correspondent à des actions mentales plutôt que des actions dans l'environnement par des expressions dans un langage de requête (ce que font aussi les CT[99] [12]). Pour supporter le paradigme du tuteur, Astus reprend cette idée et la rend plus explicite (comme le font jusqu'à un certain point Sierra et TOTS) et l'enrichit. En particulier, les primitives du langage de requête d'Astus correspondent nécessairement à des processus cognitifs élémentaires (perceptions et rappels) ou à des inférences, c'est-à-dire l'exécution mentale de connaissances procédurales générales ou propres au domaine ou à la tâche qui sont supposées maitrisées.

[99] On peut également considérer que Andes/ST le fait puisqu'ils ne modélisent explicitement pas les buts.

Par conséquent, Astus décourage[100] une astuce fréquemment utilisée dans les CT, soit de représenter des connaissances procédurales qui ne sont pas maitrisées (et qui font généralement partie des objectifs d'apprentissage) de façon « boite noire ». Un exemple classique de cette astuce est le TAL dans le cas des « problèmes écrits » (« *word problem* ») où une règle encapsule l'extraction d'un élément du texte (p. ex. extraire une équation) [19]. Autrement dit, à défaut de pouvoir offrir une solution au problème du TAL, Astus encourage les auteurs à être cohérents par rapport à l'environnement dans lequel la tâche s'effectue et les objectifs d'apprentissage que la tâche permet d'atteindre. Par exemple, dans notre MTT pour la création du nuage de points, les variables dépendante et indépendante sont déjà connues, l'objectif qui demeure est de les associer à leur axe respectif.

Même si les modèles d'Astus s'inscrivant dans le paradigme du tuteur plutôt que dans le paradigme de l'apprenant, ils sont plus près, à plusieurs égards, des modèles d'ACT-R que ceux des CT[101]. Ceci étant dit, nous ne prétendons pas pour autant que les graphes procédural et épisodique et les éléments sémantiques sont fidèles aux structures et processus de la mémoire humaine. Autrement dit, nous assumons le fait qu'Astus n'est pas une architecture cognitive, ce qui était moins clair avec UV et le MGC. De plus, même si un modèle d'Astus est plus sophistiqué que celui pour la même tâche dans les CT, en tenant compte qu'Astus offre des langages dédiés et des outils offerts (nous les présentons dans la section 2.4 et au chapitre 4), nous estimons que les efforts requis pour créer un tel modèle sont d'un ordre de grandeur comparable à ceux requis par CTAT.

Comme dans la plupart des SABC, les éléments sémantiques d'Astus forment une base de connaissances (KB pour « *knowledge base* ») où on distingue les éléments terminologiques (« TBox »), par exemple les concepts, des assertions (« ABox »), comme les objets. Dans la littérature, on désigne par ontologie les éléments de la TBox et les éléments de la ABox qui

[100] Encore une fois, même si un langage favorise ou décourage un paradigme de modélisation plus qu'un autre, il est difficile, voire impossible de contraindre l'auteur à s'y inscrire.
[101] Autrement dit, il serait plus facile de générer un modèle ACT-R à partir d'un modèle d'Astus que celui d'un modèle CT (dans les deux cas, des directives supplémentaires seraient nécessaires).

sont propres au domaine. En ce sens, la KB étend l'ontologie avec les éléments qui sont propres à une instance de la tâche (au chapitre 3, nous donnons un sens plus large à KB).

En représentation des connaissances, on distingue généralement les inférences logiques, qu'on retrouve par exemple dans OWL[102], qui doivent être conformes à un cadre logique (p. ex. celui des DL), des inférences « rationnelles »[103] qui représentent de façon générale l'exécution de connaissances procédurales menant à des actions qui mettent à jour la KB (autrement dit, des actions internes) [74]. Astus n'inclut pas d'inférences logiques pour deux raisons : d'une part, pour la même raison qu'il n'inclut pas un SABC agissant comme un planificateur au niveau procédural, c'est-à-dire parce que le raisonnement basé sur les inférences logiques est un problème IA-Complet. D'autre part, même si pour plusieurs sous-problèmes (comme celui ciblé par les DL) il existe des solutions dans les outils de l'IA (comme OWL), la correspondance entre l'expressivité de ces dernières et les besoins du paradigme du tuteur n'est pas toujours adéquate. En effet, même si certaines limitations ont été éliminées [101] depuis que nous avons évalué l'utilisation de OWL [132], des limitations importantes demeurent, puisqu'elles sont nécessaires pour que les inférences soient décidables [167]. De plus, les difficultés liées à l'usage d'un raisonneur OWL existant, ou encore celles liées à en développer un à l'interne, sont importantes[104].

Par conséquent, Astus n'inclut pas à proprement dit d'inférences[105] au niveau de la TBox (les inférences logiques facilitent grandement l'implémentation de ce type d'inférences) et se limite à inclure des inférences « rationnelles » au niveau de la ABox (à l'image du système de production SWRL[106] pour OWL). En effet, même s'il existe également des cadres logiques conçus pour raisonner au niveau de la ABox (p. ex. Datalog [51]), l'expressivité de ces derniers ne correspond pas nécessairement aux besoins du paradigme du tuteur. De plus, les

[102] Un langage de représentation des connaissances conçu pour les besoins du Web sémantique et dont le principal profil (OWL-DL) correspond aux DL (http://www.w3.org/TR/owl2-overview/).
[103] Dans le même sens qu'un « *rational agent* » en IA [216].
[104] Les travaux sur le MGC proposent d'ajouter de telles inférences à UV à l'aide de OWL [88], mais les résultats obtenus ne nous ont pas convaincus [132].
[105] Des inférences simples sont intégrées au langage de requêtes.
[106] http://www.w3.org/Submission/SWRL/

difficultés évoquées pour les raisonneurs logiques au niveau de la TBox sont également présentes pour ceux au niveau de la ABox.

Tous les systèmes mentionnés précédemment dans ce chapitre[107] supportent plus ou moins explicitement de telles inférences (p. ex. les principes mineurs sous forme de clause de Horn dans Andes). Dans les CT, les inférences correspondent au code Jess/Java dans la RHS qui produit les nouvelles instances de WME (qui ne correspondent pas à des rappels ou à des perceptions). Astus se distingue des autres systèmes en supportant explicitement à la fois des inférences « boite noire » qui sont associées à des connaissances procédurales maitrisées et des inférences interprétables qui sont est associée à des expressions dans le langage de requête.

2.2 L'interaction avec l'environnement

Pour Astus, comme pour Miace, TOTS/Steve, Sierra et Andes/ST, l'environnement reflète le contenu de la KB/WM, contrairement à UV[108] et aux CT/ACT-R, où c'est l'inverse, c'est-à-dire que la KB reflète l'état de l'environnement. Comme nous l'avons évoqué au chapitre 1, la littérature indique que la deuxième approche est adoptée pour supporter les environnements existants. Parmi les systèmes mentionnés précédemment, les plateformes (UV, CT) adoptent plutôt la deuxième approche tandis que les systèmes dédiés (Miace et Sierra[109], Steve et Andes/ST) adoptent la première approche. TOTS et Astus sont deux plateformes qui adoptent la première approche, mais Astus se distingue en la supportant explicitement. En effet, la KB dans TOTS est implicite, elle est représentée par des variables globales dans le langage hôte (Lisp) [208].

Avec Astus, nous avons retenu la première approche puisqu'elle comporte trois principaux avantages :

[107] UV ne décrit pas clairement les inférences qu'il propose de supporter [88].
[108] En effet, ni UV ni les travaux sur le MGC ne proposent de généraliser la notion d'environnement qui est propre à un domaine dans Miace (il s'agit d'un tableau noir sur lequel on fait de l'arithmétique).
[109] Son environnement est similaire à celui de Miace, mais il est propre à la tâche de soustraction en colonnes.

- elle évite les problèmes de synchronisation des CT entre la WM et l'environnement tel que décrit au chapitre 1;

- elle réduit les efforts de création en éliminant toute forme de redondance entre la KB et l'environnement;

- elle rend la sémantique de la KB (en particulier les éléments de la ABox) plus précise, c'est-à-dire qu'elle décrit explicitement les éléments communs à partir desquels l'apprenant et le module pédagogue interagissent [87].

Le dernier avantage est le plus important, puisqu'il supporte le paradigme du tuteur, c'est-à-dire que la KB doit avoir du sens à la fois dans la perspective de l'apprenant et dans celle du module pédagogue, en tenant compte que cette dernière est plus abstraite. En effet, ce dernier n'a pas accès à la sémantique de la KB qui est propre au domaine ou à la tâche, mais seulement à celle offerte par le métamodèle. Si cette approche est utilisée conjointement avec un environnement existant elle introduit une redondance qu'elle visait originellement à éviter (nous y revenons au chapitre 4). Si elle est utilisée avec un environnement créée pour les besoins du MTT, elle amène l'auteur à adopter un processus de création où la modélisation de la KB et le développement de l'environnement sont interdépendants (nous y revenons au chapitre 4).

Nous avons, avec Astus, l'ambition, de supporter des environnements qui ont une dimension physique (c.-à-d. qui simulent la manipulation d'objets concrets plutôt qu'uniquement des objets abstraits), en particulier des environnements dynamiques (c.-à-d. un environnement dont l'état évolue indépendamment des actions qui y sont produites). Par exemple, on retrouve un environnement dynamique dans Steve (Figure 15). Des exemples de domaines et de tâches qui requièrent à la fois de tels environnements et qui sont compatibles avec l'approche pédagogique d'Astus (et plus généralement celle des MTT) sont par exemple : un laboratoire simulé pour un cours de « génie génétique » (en biologie) où l'apprenant doit créer une carte de restrictions et un patient simulé pour un cours de « soins critiques » (en sciences infirmières) où l'apprenant doit appliquer un protocole bien défini. Nous revenons

sur la problématique de ces environnements et les pistes de solutions proposées par Astus au chapitre 4.

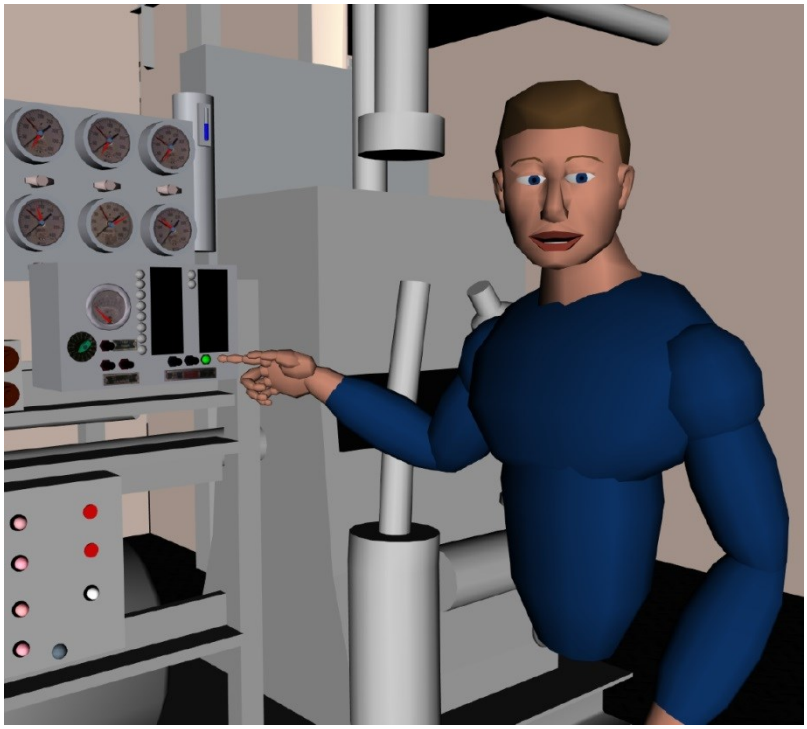

Figure 15 - Un aperçu de l'environnement de Steve (tirée de [111]).

À l'image du métamodèle qui permet au module pédagogue de manipuler la KB, Astus propose un métamodèle de l'environnement qui s'inspire du patron de conception « modèle-vue-contrôleur » (MVC) [125] où la KB joue le rôle du modèle. Ce métamodèle est constitué d'une part d'une structure qui joue le rôle de la vue en reflétant la hiérarchie des composants graphiques de l'environnement (qui avec le tuteur, forme l'UI) et d'autre part de patrons d'interactions qui sont associés aux actions [87].

Comme dans les CT une action de l'environnement est associée à un SAI, dans Astus, une action est associée à un patron d'interactions plus expressif qu'un SAI [87] (qui à l'exécution se comporte grosso modo comme un automate fini, nous y revenons au chapitre 4). Par exemple : une sélection dans une liste (qui peut être répétée) suivie d'un bouton appuyé. Lorsqu'une action est reconnue, deux processus distincts sont exécutés, et ce, avant que le

graphe procédural ne soit mis à jour. Premièrement, la KB est mise à jour en fonction d'un script écrit dans le langage de manipulation de façon à refléter la sémantique de l'action. Par exemple, dans la tâche de résoudre une équation, le script ajouterait l'objet qui représente la nouvelle équation qui résulte de l'action produite (p. ex. soustraire 2 de chaque côté de l'équation). Deuxièmement, l'environnement est mis à jour[110]. Par exemple, l'environnement afficherait l'équation qui vient d'être ajoutée. Dans les CT, la nouvelle équation serait d'abord créée dans l'environnement (qui est une « boite noire »[111]) puis serait réifiée dans la WM sous forme d'une instance de WME (par la RHS de la règle qui a produit le SAI apparié au SAI résultant de l'action produite par l'apprenante).

La structure qui joue le rôle de la vue est un arbre[112] dont la racine correspond au composant graphique qui englobe l'environnement (p. ex. une fenêtre) et dont les descendants correspondent à l'imbrication des composants graphiques qui représentent graphiquement les éléments sémantiques. À chacun de ces nœuds sont associés des composants (p. ex. les éléments d'une liste) et des signaux (p. ex. sélection dans une liste et bouton appuyé) qui sont déclenchés par des interactions dans l'environnement (p. ex. lorsqu'on sélectionne un élément dans une liste et qu'on appuie sur un bouton). Ces signaux sont acheminés au contrôleur dont le rôle est essentiellement (nous détaillons le rôle du contrôleur au chapitre 4) le suivant : lorsque les interactions produites dans l'environnement correspondent à un des patrons associés aux actions, il extrait de ces dernières les éléments qui deviennent les arguments de l'action [87] (p. ex. l'objet qui correspond à la sélection dans une liste). Lors du suivi de l'apprenant, les arguments de l'action sont comparés aux valeurs des variables des instances de procédures primitives correspondantes dans le graphe épisodique.

Cette approche basée sur le patron MVC a deux avantages, d'une part elle permet d'offrir aux auteurs un outil avec lequel ils peuvent exécuter les actions sans produire les interactions dans l'environnement. D'autre part, elle permet au module communication de faire une

[110] Certaines interactions requièrent que l'environnement soit mis à jour avant que l'action soit exécutée, nous les abordons au chapitre 4.
[111] Sous la forme d'une bibliothèque Java dans le cas particulier des CT de CTAT.
[112] Bien que la structure des objets puisse être un graphe, nous y revenons au chapitre 4.

démonstration de l'action (comme on le retrouve dans Steve) [87, 111] en générant les interactions à partir du patron et de la structure qui joue le rôle de la vue, c'est-à-dire que le module communication déplace le pointeur et simule les clics et les saisies. En ce sens, le module communication d'Astus remplit davantage son rôle tel que proposé par les concepteurs originaux des CT [19] que les CT eux-mêmes.

Comme le font des CT créés avec le TDK, le tuteur d'Astus inclut un bouton d'annulation (les autres fonctions du tuteur sont abordées dans la section suivante). Ce bouton, généralement accessible, permet à l'apprenant d'annuler la dernière action produite, qu'elle soit correcte ou incorrecte. Bien que l'usage principal de ce bouton soit d'annuler une action incorrecte, la possibilité d'annuler une action correcte est également importante afin de permettre à l'apprenant d'explorer différents chemins du graphe procédural et de faire un « retour arrière » (« *backtracking* ») lorsqu'il arrive à une impasse (c.-à-d. qu'après une action correcte donnée, il n'y a plus d'actions correctes possibles). Dans ce dernier cas, le module communication bloque également l'UI (nous précisons la notion d'impasse au chapitre 4).

Puisque l'annulation d'une action correcte est une composante essentielle de l'approche pédagogique d'Astus, il n'a pas été possible de recourir aux approches qui permettent d'annuler une action directement dans l'environnement que ce soit l'« édition sur place » comme le fait les CT et Andes/ST ou encore un patron d'interactions associé à l'annulation d'une action comme le proposent nos travaux antérieurs [87]. En effet, d'une part annuler une action dans l'environnement est problématique pour les tâches où une action fait disparaitre un élément de l'environnement, puisque l'auteur doit alors faire en sorte que ce dernier reste visuellement accessible pour permettre à l'apprenant d'effectuer l'annulation. D'autre part, l'annulation d'une action correcte dans l'environnement, dans le cas de l'« édition sur place », donne lieu à des ambigüités[113]. De plus, elle requiert l'annulation en cascade des actions produites avant celle qui est annulée, ce qui va à l'encontre du principe de moindre

[113] C'est-à-dire qu'on peut l'interpréter à la fois comme une annulation ou une nouvelle action.

surprise[114]. Par conséquent, nous proposons plutôt que le module communication bloque l'environnement (sans le masquer) après une action incorrecte. Cette approche qui s'applique pour toutes les tâches, bien que parfois moins conviviale (selon la nature de l'environnement), permet néanmoins d'assurer une cohérence dans le comportement des différents environnements supportés. De plus, cette approche évite à l'auteur d'avoir à s'assurer que l'état de la KB demeure cohérent et que l'environnement soit utilisable après une deuxième action incorrecte.

2.3 Le suivi de l'apprenant et la sélection de tâches

Les processus du suivi de l'apprenant et de la sélection de tâches d'Astus sont similaires à ceux des CT, dans les deux sous-sections suivantes, nous précisons comment ils se distinguent de ces derniers.

2.3.1 Le suivi de l'apprenant

Le suivi de l'apprenant dans Astus se distingue de celui des CT de trois façons :

1. le contenu des interventions est généré plutôt que prémâché (elles peuvent donc tenir compte de l'état du graphe épisodique, du modèle de l'apprenant, etc.);

2. le modèle de l'UI (celui de l'environnement et du tuteur) plus sophistiqué permet de bonifier les interventions (démonstrations, effets visuels, hyperliens, etc.);

3. grâce à des mécanismes d'extension[115], la génération des interventions peut être adaptée à une stratégie pédagogique particulière.

Pour assurer le suivi, l'action produite est comparée aux actions potentielles qui découlent des épisodes pouvant être satisfaits à l'aide d'une procédure primitive. Si l'action correspond à une des instances de procédures, le module communication produit une rétroaction

[114] http://en.wikipedia.org/wiki/Principle_of_least_astonishment
[115] Autrement dit, sans avoir besoin de recompiler Astus.

minimale positive (verte), sauf si cette dernière découle d'un épisode satisfait par une procédure complexe qui correspond à une incompréhension, dans ce cas, la rétroaction est négative (rouge). Autrement dit, Astus se comporte comme les CT dans ces deux cas. Astus se distingue des CT dans deux autres cas, d'abord lorsque l'action produite est associée à plus d'une procédure primitive et que celles-ci découlent à la fois de chemins valides ou invalides, c'est-à-dire des chemins comportant ou non des procédures invalides. Dans ce cas, la rétroaction minimale est neutre (bleu) tant que l'ambigüité n'est pas levée (plus précisément, quand le module expert ne peut pas trancher, parce que les épisodes sont équivalents, c'est le module apprenant qui doit le faire). Puis, lorsqu'une action ne correspond à aucune des actions potentielles, Astus tire profit de son modèle de la tâche explicite pour diagnostiquer l'erreur.

Le processus de diagnostic d'Astus crée un chemin dans le graphe épisodique entre un épisode existant et une procédure primitive dont aurait pu découler l'action. Ce processus est inspiré de Sierra, il consiste grosso modo à faire une fouille à partir des épisodes existants en introduisant des perturbations au moment d'instancier des chemins du graphe procédural [195, 200] (nous y revenons au chapitre 4). Que la fouille soit fructueuse ou non, la rétroaction minimale est négative, mais lorsqu'elle l'est, le module pédagogue peut produire une intervention plus riche qu'une rétroaction négative générique (ce que font les CT dans tous les cas).

Le tuteur (la partie de l'UI indépendante de la tâche et du domaine) d'Astus est similaire à celui des CT, en plus du bouton d'annulation introduit précédemment, il contient entre autres un bouton pour demander de l'aide, une fenêtre pour afficher les messages provenant du module pédagogue et un bouton de fin (qui contrairement à celui des CT est géré sans simuler une action de l'environnement). Le tuteur d'Astus se distingue de celui des CT, car sa fenêtre de message est interactive (document hypertexte) et permet entre autres à l'apprenant de préciser sa demande d'aide (similairement aux mécanismes offerts par Andes/ST et Steve, mais de façon indépendante du domaine) et de fournir son appréciation des interventions (nous y revenons au chapitre 4).

Astus se distingue également des CT, car le module communication peut tirer profit du modèle de l'environnement (l'arbre des représentations graphiques et les patrons d'interactions) pour bonifier les interventions du module pédagogue. Par exemple, en mettant en surbrillance la représentation graphique des éléments correspondant aux valeurs des variables d'un épisode (et non seulement à partir du SAI comme c'est le cas avec CTAT).

Nous décrivons au chapitre 4 les mécanismes de base et les mécanismes d'extensions de génération des interventions. Ces derniers tirent profit du caractère explicite et de la sémantique précise du modèle de la tâche qui découlent du paradigme du tuteur, du modèle de l'UI et d'énoncés en langage naturel [195], des éléments de connaissance didactique qui doivent être fournis par l'auteur. Syntaxiquement similaires aux gabarits des messages des CT qui sont associés aux règles, les **énoncés** sont associés aux buts et aux éléments de connaissance sémantiques (et non aux procédures). À la différence des gabarits des CT, ces énoncés, puisqu'ils sont propres à un élément de connaissance donné, peuvent être composés de façon indépendante de la tâche et du domaine pour produire un message. Par exemple, pour créer un message à l'aide du gabarit associé aux séquences, il est nécessaire d'avoir un énoncé pour chacun des sous-buts (nous revenons sur les énoncés dans la section 2.4).

2.3.2 La sélection de tâches

Comme les CT, la plupart des MTT et la majorité des STI, l'approche pédagogique d'Astus est basée sur des travaux en psychologie cognitive et en éducation [59, 94, 139, 236], en particulier ceux qui proposent d'une part de décomposer l'acquisition des connaissances procédurales en trois phases (initiale, maitrise, optimisation) et d'autre part qui soutiennent que ce sont des mécanismes d'apprentissage différents qui sont à l'œuvre durant ces trois phases [13, 187, 258, 259].

La phase initiale débute à l'extérieur du MTT, que ce soit en classe ou à l'aide d'un hypermédia éducatif, elle consiste entre autres à acquérir les connaissances déclaratives préalables à l'acquisition des connaissances procédurales (p. ex. concepts, principes du domaine, exemples et connaissances téléologiques). La phase initiale se termine lorsque

l'apprenant a été guidé une première fois par le MTT au travers d'une instance de la tâche. La phase de maitrise commence avec la fin de la phase initiale et se termine quand l'apprenant ne fait plus d'erreurs à l'exception d'erreurs d'inattention (« *slips* ») [182]. C'est normalement la phase principalement ciblée avec un MTT. Dans la phase d'optimisation, qui commence où se termine la phase de maitrise et qui n'a pas véritablement de fin, l'apprenant devient plus efficace (rapide), entre autres en sautant des actions (mentales ou dans l'environnement). Pour supporter cette phase, on peut adopter une approche minimaliste comme Andes ou ST, ou encore, dans le cas des CT et Astus, le supporter explicitement en ajoutant des éléments procéduraux qui représentent des optimisations (à l'image des éléments qui représentent des incompréhensions). En tout, on retrouve neuf mécanismes différents qui se chevauchent dans les trois phases [186] (voir Tableau 2).

Tableau 2 - Les mécanismes et les phases d'apprentissage

Mécanisme	**Phase(s)**	**Supporté explicitement par Astus**
Suivre des instructions	1, 2	Oui
Dériver les connaissances procédurales à partir des principes du domaine	1, 2	Non
Réutiliser des connaissances procédurales dans des tâches ou des domaines analogues	1, 2	Non
Utiliser des connaissances procédurales générales	1, 2	Non
Étudier des exemples, observer des démonstrations	1, 2	Oui
Tirer profit d'une rétroaction positive	2	Oui
Tirer profit d'une rétroaction négative	2	Oui
Reconnaitre des motifs dans l'environnement	3	Par extension du modèle
Mémoriser	3	Par extension du modèle

Astus supporte les trois modes de sélection de tâches décrits en introduction, c'est-à-dire la sélection par l'apprenant, la sélection basée sur une liste préétablie par le corps enseignant et la sélection effectuée par le module pédagogue en fonction des objectifs d'apprentissage[116]. En plus du mode par défaut où l'apprenant doit effectuer la tâche, Astus ajoute un mode « exemple détaillé » (« *worked example* ») où c'est le module pédagogue qui effectue

[116] Dans la version actuelle d'Astus, ce processus n'est pas optimisé.

entièrement la tâche pour l'apprenant. Astus supporte également un mode « non guidé » (non disponible par défaut) où l'apprenant peut explorer l'environnement à sa guise (une rétroaction minimale neutre est donnée pour déterminer les effets des actions)[117]. Ce mode n'est pas disponible par défaut, car il exige que l'auteur rende l'environnement plus robuste afin de permettre au module communication de ne jamais bloquer l'environnement. Pour supporter davantage la phase initiale, des travaux futurs devront intégrer Astus à un hypermédia interactif, idéalement, de façon à établir une communication bidirectionnelle entre ce dernier et le tuteur (p. ex. pour qu'un lien offert par le tuteur permette à l'apprenant d'accéder à un document de l'hypermédia interactif[118]). Aux chapitres 4 et 5, nous expliquons pourquoi il est difficile de supporter automatiquement la phase d'optimisation dans Astus.

Pour permettre au module pédagogue d'effectuer la sélection de tâches en fonction des objectifs d'apprentissage, un modèle de l'apprenant est nécessaire. Puisque la modélisation de l'apprenant, comme nous l'avons annoncé en introduction, n'a pas été au centre de nos travaux, le modèle de l'apprenant d'Astus sera développé lors de travaux futurs. Le modèle de l'apprenant minimal qui est présenté au chapitre 4 s'inspire de celui des CT, c'est-à-dire qu'il contient par défaut un objectif d'apprentissage pour chacun des chemins du graphe procédural (dans la littérature, ces chemins correspondent aux nœuds observables [75]).

2.4 Le processus de création

Astus propose un processus de création similaire à celui des CT, c'est-à-dire que le modèle est issu d'une collaboration entre le corps enseignant et des auteurs (dans la littérature, on désigne parfois ces derniers comme des « ingénieurs de connaissance »[119]). En ce sens, Astus se distingue des approches actuelles dont l'objectif est de démocratiser l'utilisation des STI en déplaçant les efforts de modélisation vers le corps enseignant. En général, ces approches ne mènent pas à la création de MTT, mais plutôt à d'autres familles de STI, comme les ETT

[117] Qui correspond en fait au mode « test de l'environnement » qui est offert aux auteurs
[118] Une approche de représentation des connaissances comme celle du langage MOT+ [194] pourrait être utilisée pour créer les liens entre les notions du domaine qu'on retrouve dans le MTT et l'hypermédia.
[119] « *Knowledge engineer* » (http://en.wikipedia.org/wiki/Knowledge_engineer)

(nous y revenons au chapitre 5). Cependant, deux approches actuelles tentent d'éviter les efforts importants liés à la création d'un MTT, tout en obtenant un comportement comparable à celui de ces derniers. La première approche, que nous avons évoquée en introduction, est d'utiliser l'apprentissage automatique. La deuxième approche est de développer des outils auteurs, c'est-à-dire des outils censés permettre au corps enseignant de créer eux-mêmes le modèle. Ces outils auteurs ont principalement recours aux des approches suivantes : la programmation visuelle (comme le langage de contrôle d'instruments de laboratoire LabVIEW[120] et le langage pour l'apprentissage de la programmation Scratch[121]) et les langages naturels contrôlés [222] (comme Attempto [93]). Par exemple, le CT-SDK propose de remplacer l'utilisation d'un système de production classique par un « *predicate tree* » qui est créé visuellement [34]. Un autre exemple est des outils auteurs permettant d'étendre le modèle d'Andes à l'aide d'un langage naturel contrôlé [244].

Avec Astus, nous nous inscrivons en faux contre ces deux dernières approches pour deux raisons. Premièrement, nous doutons que de tels outils auteurs permettent réellement au corps enseignant de créer un modèle, autrement dit, si des résultats en ce sens sont observés, c'est probablement que les sujets de l'étude sont en fait des programmeurs. Par exemple, c'était le cas avec les auteurs dans l'étude sur le Cognitive Tutors SDK [34]. Deuxièmement, si les auteurs sont en fait des programmeurs, rien ne garantit que la programmation visuelle et que les langages naturels contrôlés sont les meilleurs outils pour créer un modèle. En effet, bien que ces approches aient trouvé leur niche respective, il y a toujours un vif débat entourant leur convivialité[122]. De plus, notre expérience avec le MGC, qui proposait l'utilisation d'un tel outil auteur [174], nous a convaincus qu'il serait plus convivial d'encoder le modèle de la tâche dans un fichier source, en particulier afin d'y intégrer les éléments procéduraux « boites noires » qui sont plus aisément encodés à l'aide d'un script (programme).

[120] http://www.ni.com/download-labview/
[121] https://scratch.mit.edu/
[122] Qui, bien qu'il soit actif dans l'espace public, il est mal documenté dans la littérature à notre connaissance.

Par conséquent, dans Astus, nous avons adopté une approche visant à réduire les efforts des auteurs qui sont des programmeurs. En fait, il s'agit de leur offrir d'une part des outils tels qu'on en retrouve pour la majorité des langages de programmation (idéalement, de façon à supporter les phases d'édition, de compilation[123] et d'exécution) et d'autre part, des langages dédiés (DSL) pour faciliter l'encodage du modèle de la tâche. Des travaux futurs pourront vérifier expérimentalement si les DSL et les outils offerts permettent de contenir les efforts de modélisation à un ordre de grandeur comparable à ceux des CT, mais ce n'est pas une tâche facile. En effet, on doit tenir compte qu'Astus génère des interventions pédagogiques tandis que les CT ont recours à des interventions prémâchées (il faut donc mesurer le rapport entre les efforts de modélisation et l'efficacité des interventions produites).

De même, contrairement aux CT, Astus n'intègre pas à son processus de création l'utilisation d'un outil de création visuelle pour l'environnement. Ces outils, bien qu'ils permettent de créer rapidement des environnements simples, sont moins conviviaux qu'un DSL pour créer des environnements sophistiqués[124]. Par conséquent, comme le modèle de la tâche, le modèle de l'environnement est lui aussi créé à l'aide de DSL, en particulier un pour créer la vue (la structure des représentations graphiques) et un pour créer les patrons d'interactions.

Bien qu'Astus permette à l'auteur de vérifier son modèle à l'édition et à la compilation, la vérification se fait principalement à l'exécution. En effet, ce compromis provient de notre volonté de maintenir l'encodage des modèles d'Astus comparable à ceux des CT. Pour enrichir la vérification lors de la compilation, il aurait fallu par exemple développer un DSL statiquement typé [98] et enrichir les structures du modèle de façon à ce qu'elles permettent d'exprimer des invariants ou des axiomes. Par exemple, le DSL pour les éléments sémantiques ne permet pas d'exprimer de restrictions qui font en sorte qu'on peut vérifier automatiquement qu'un concept ou qu'un objet ajouté au modèle est cohérent avec les éléments existants (tel qu'on peut le faire avec OWL ou de façon minimale avec Protégé-

[123] Par abus de langage. On obtient un fichier qui contient les modèles, mais ce n'est pas un fichier objet.
[124] Comme pour l'utilisation de la programmation visuelle, c'est ce que laisse croire le débat dans l'espace public sans qu'il existe, à notre connaissance, une littérature qui le montre clairement.

Frames). Un exemple classique est la restriction de cardinalité, par exemple exprimer « une soustraction contient au moins 2 colonnes, mais pas plus de 5 » ou encore « dans une soustraction, le nombre représenté par les opérandes du haut doit être ≥ au nombre représenté par les opérandes du bas ». Un autre exemple est que le DSL pour les éléments procéduraux ne permet pas de définir des pré/postconditions au niveau des actions telles qu'on les retrouve en planification, ce qui fait en sorte que pour analyser le comportement des actions dans l'environnement, on doit le faire à l'aide de traces d'exécution (au chapitre 4, nous présentons le rôle que jouent les pré/postconditions dans le graphe procédural).

Plus précisément, en plus de la vérification et de l'analyse statique du modèle qui peuvent être effectuées à la compilation, l'auteur dispose d'outils de visualisation (de la KB, du graphe procédural et épisodique, etc.) et de cinq modes différents pour vérifier le modèle et faire une analyse dynamique à l'exécution :

1. le **test de l'environnement**[125] qui lui permet de manipuler l'environnement avant même d'avoir créé le graphe procédural (seules les procédures primitives sont nécessaires);

2. l'**exécution descendante** qui lui permet d'effectuer la tâche en sélectionnant les procédures qui doivent être exécutées depuis le but racine (autrement dit, l'auteur ne fait aucune manipulation dans l'environnement);

3. le mode **apprenant** où l'auteur effectue la tâche comme s'il était un apprenant, il reçoit alors les mêmes interventions pédagogiques que recevrait l'apprenant;

4. l'**exploration automatique** où Astus tente d'effectuer toutes les instances de toutes les tâches de façon à parcourir tous les chemins du graphe procédural;

5. l'**apprenant simulé** où Astus, en plus d'explorer le graphe procédural, simule aussi des erreurs (et toutes les autres actions dans le tuteur).

[125] Comme mentionné précédemment, ce mode peut être offert à l'apprenant pour des tâches particulières.

Les deux derniers modes sont supportés de façon minimale, mais suffisamment pour vérifier qu'il sera possible de les améliorer lors de travaux futurs. On peut croire que l'exploration automatique et l'apprenant simulé peuvent être exécutés sans créer et mettre à jour l'environnement, mais pour rendre efficace le processus de génération d'erreur, il faut utiliser l'état de l'environnement pour élaguer l'arbre de fouille. En particulier, si les interactions menant à une action ne peuvent pas être effectuées dans l'état courant de l'environnement, l'apprenant ne pourrait pas produire l'action. Autrement dit, il est avantageux de faire la fouille de façon bidirectionnelle, à partir des actions possibles dans l'environnement et du graphe épisodique. Pour obtenir des métadonnées sur l'état de l'environnement en fonction des actions produites, l'exploration automatique doit tenir compte de cette dernière. Par contre, une première passe peut être effectuée sans l'environnement de façon à obtenir efficacement des métadonnées de base.

On peut remarquer que l'apprenant simulé pose un problème direct tandis qu'établir le diagnostic des actions non tracées (celles qui ne découlent pas du graphe épisodique) pose un problème inverse[126]; en effet, dans les deux cas, on doit créer un chemin supplémentaire dans le graphe épisodique, mais dans le cas de l'apprenant simulé, la destination n'est pas un paramètre (c'est une variable). L'apprenant simulé permet d'observer le comportement du module pédagogue, en particulier lorsqu'on modifie la stratégie pédagogique à l'aide des mécanismes d'extensions (il permettrait également d'observer le comportement d'un modèle de l'apprenant plus sophistiqué).

L'exploration automatique permet quant à elle d'accumuler des métadonnées sur le modèle, par exemple de vérifier que les instances de tâches disponibles sont suffisantes pour couvrir ou non tous les chemins du graphe procédural et pour estimer des invariants comme les restrictions de cardinalité ou les plages de valeurs des variables du graphe procédural[127]. En effet, ceci permet d'une part de guider l'apprenant simulé et d'autre part d'obtenir une forme de vérification du modèle sans fournir d'efforts additionnels au moment de la création. Si

[126]http://en.wikipedia.org/wiki/Inverse_problem.
[127]La collecte des métadonnées est minimale, des travaux futurs viseront à l'améliorer.

l'auteur désire que ces métadonnées soient prescriptives plutôt que descriptives (c'est-à-dire qu'elles provoquent des erreurs), il doit les approuver manuellement et les ajuster au besoin.

On peut remarquer que contrairement aux invariants mentionnés plus haut, nous avons décrit les énoncés en langage naturel comme étant intégrés au modèle de la tâche et non comme des métadonnées. En fait, c'est d'abord et avant tout pour des raisons de convivialité que nous avons décidé d'encoder les connaissances didactiques à même le modèle de la tâche. Si un auteur désire supporter plusieurs langues, il est possible qu'une telle intégration ne soit plus conviviale, alors il faudra développer, lors de travaux futurs, un DSL pour supporter les langues supplémentaires.

Finalement, on peut se demander si le DSL des connaissances sémantiques, qui est indépendant du domaine, demeure convivial pour les tâches où un grand nombre d'instances sont nécessaires (ou encore les tâches où chaque instance requiert un grand nombre d'éléments de connaissance). Par exemple, nous avons constaté que dans le MTT de manipulations des ABR, la création d'instances de tâches est laborieuse (elle le serait d'ailleurs aussi avec les CT). Pour y remédier, nous suggérons aux auteurs de développer un format de données propre à la tâche (au besoin supporté par un GUI) et un script qui génère le script correspondant dans le DSL des connaissances sémantiques. Des travaux futurs pourront intégrer à Astus un mécanisme pour faciliter la génération d'instances de tâches (p. ex. en utilisant les outils de l'IA pour la satisfaction de contraintes).

Chapitre 3

La plateforme Astus

Ce chapitre présente la plateforme Astus (Figure 16), les MTT qu'elle a permis de créer et les expérimentations qu'elle a permis de mener. De cette façon, nous amorçons l'atteinte de notre deuxième objectif (qui est complété au chapitre 4), soit de montrer que la version d'Astus développée lors de nos travaux répond aux exigences énoncées en introduction.

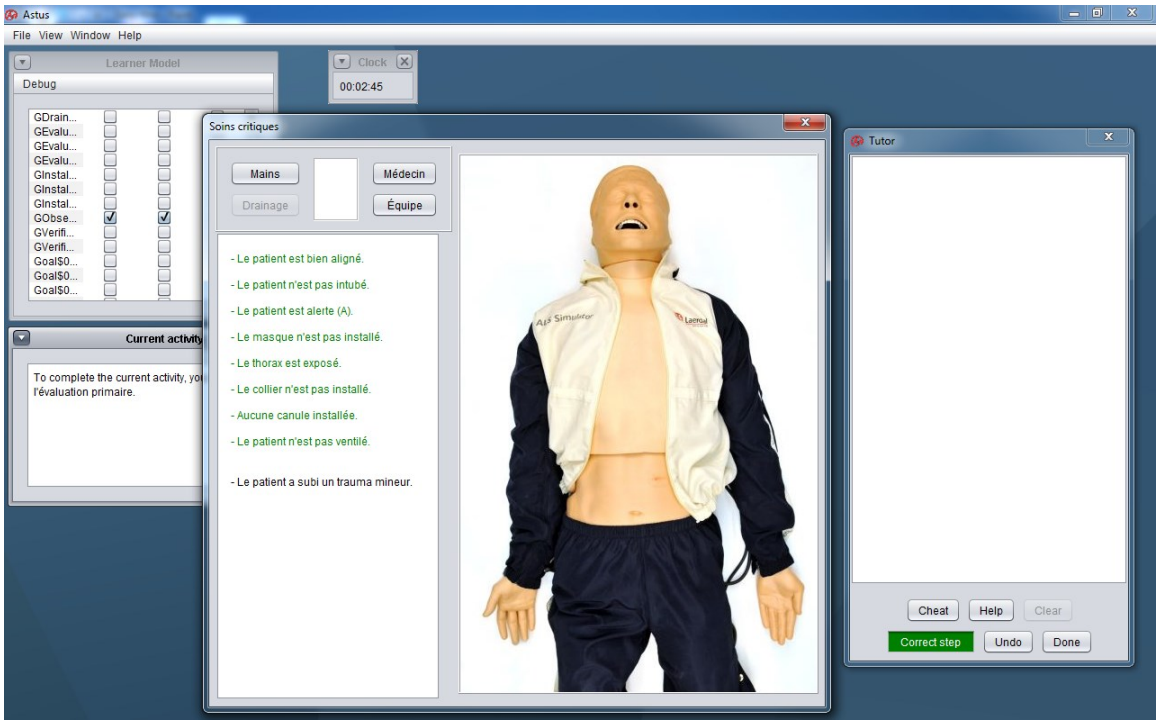

Figure 16 - Un aperçu de la plateforme Astus avec le MTT de « soins critiques ».

Implémentation de la plateforme

Nous avons implémenté[128] la plateforme Astus en Java et en Groovy[129]. Nous avons choisi Java d'une part, car plusieurs bibliothèques (notamment Jess) sont accessibles dans le cadre d'un projet universitaire[130] et d'autre part parce que Swing, une bibliothèque de création de GUI (qui fait partie de la bibliothèque standard de Java), est à la fois personnalisable et multiplateforme. Nous avons choisi Groovy, car il offre des fonctionnalités pour la création de DSL « intégrés »[131]. Nous avons préféré avoir recours à des DSL « intégrés » plutôt qu'à des DSL « externes » puisque, selon notre expérience acquise lors des travaux précédents sur les patrons d'interactions [86], les premiers sont plus flexibles que les derniers, ce qui convient davantage au contexte d'un projet de recherche. De plus, Groovy offre un DSL (« intégré ») pour la création de GUI à l'aide de Swing et une librairie de gabarits.

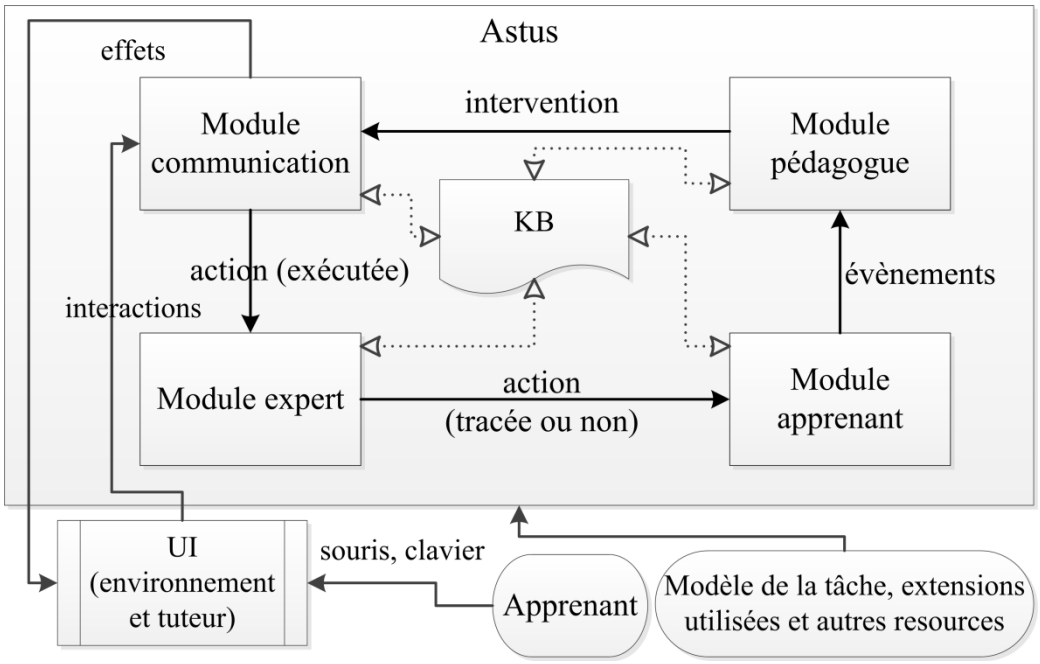

Figure 17 - La version d'Astus de l'architecture classique des STI.

[128] Avec L. Paquette.
[129] http://groovy-lang.org/
[130] P. ex. sous licences libres (p. ex. BSD, Apache, LGPL), mais non GPL; Jess est utilisé sous licence universitaire; la liste complète est sur le site Web du laboratoire ASTUS (http://astus.usherbrooke.ca).
[131] « *embedded* » c'est-à-dire par extension de la syntaxe du langage hôte (comme les macros Lisp).

Au chapitre 2, nous avons donné un sens restreint à la KB (les éléments sémantiques), mais dans ce chapitre, nous lui donnons un sens plus large. La KB (Figure 17) contient toutes les données qui sont partagées par les modules, par exemple le modèle de la tâche et de l'UI, le modèle de l'apprenant et la stratégie pédagogique. Le système de production Jess joue plusieurs rôles dans l'implémentation de la KB, nous les détaillons au chapitre 4.

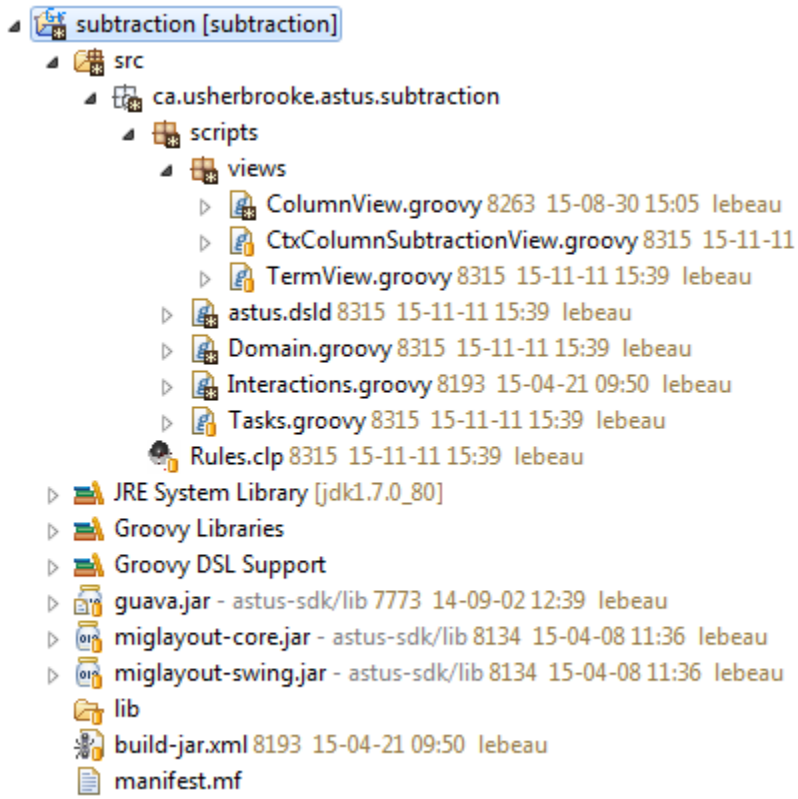

Figure 18 - La structure d'un laboratoire dans Eclipse.

Nous avons choisi l'IDE Eclipse pour la création d'un MTT entre autres pour ses plugiciels qui supportent les DSL Groovy et Jess. La création d'un MTT se fait en suivant une convention de répertoires et de fichiers qui contiennent le code source des modèles et celui propre aux mécanismes d'extension utilisés par les auteurs (Figure 18). Dans le contexte de la création d'un MTT, la plateforme Astus est utilisée comme une bibliothèque, mais lors de l'utilisation, c'est la plateforme qui agit comme une application, tandis que le code propre au MTT est utilisé comme une bibliothèque.

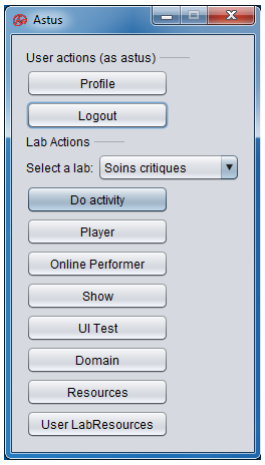

Figure 19 - La fenêtre pour un auteur (l'utilisateur *astus* dans ce cas).

Astus offre différents outils selon la catégorie de l'utilisateur[132] (Figure 19) :

- Les *administrateurs* peuvent gérer le registre des utilisateurs et des MTT, en particulier, assigner aux utilisateurs leurs droits d'accès aux MTT;

- Les *auteurs*, comme nous l'avons mentionné au chapitre 2, ont accès à des outils (Figure 20) pour créer un MTT et à plusieurs modes d'exécution pour le vérifier;

- Les *enseignants* ont accès à des outils de visualisation et ils ont accès à deux modes d'exécution qui leur sont propres[133] : rejouer la trace des actions d'une instance d'une tâche déjà effectuée et créer un « exemple détaillé » en effectuant une instance d'une tâche sans intervention du module pédagogue[134];

- Les *apprenants* peuvent étudier un « exemple détaillé », effectuer une instance d'une tâche (les mécanismes de choix disponibles sont spécifiés par l'enseignant responsable) ou revoir une instance d'une tâche déjà effectuée. Si la dernière instance effectuée n'était pas complétée, elle peut être reprise.

[132] Un drapeau permet d'activer des outils supplémentaires qui facilitent le débogage de la plateforme.
[133] Les auteurs ont aussi accès à ces deux modes.
[134] Avec une rétroaction minimale neutre.

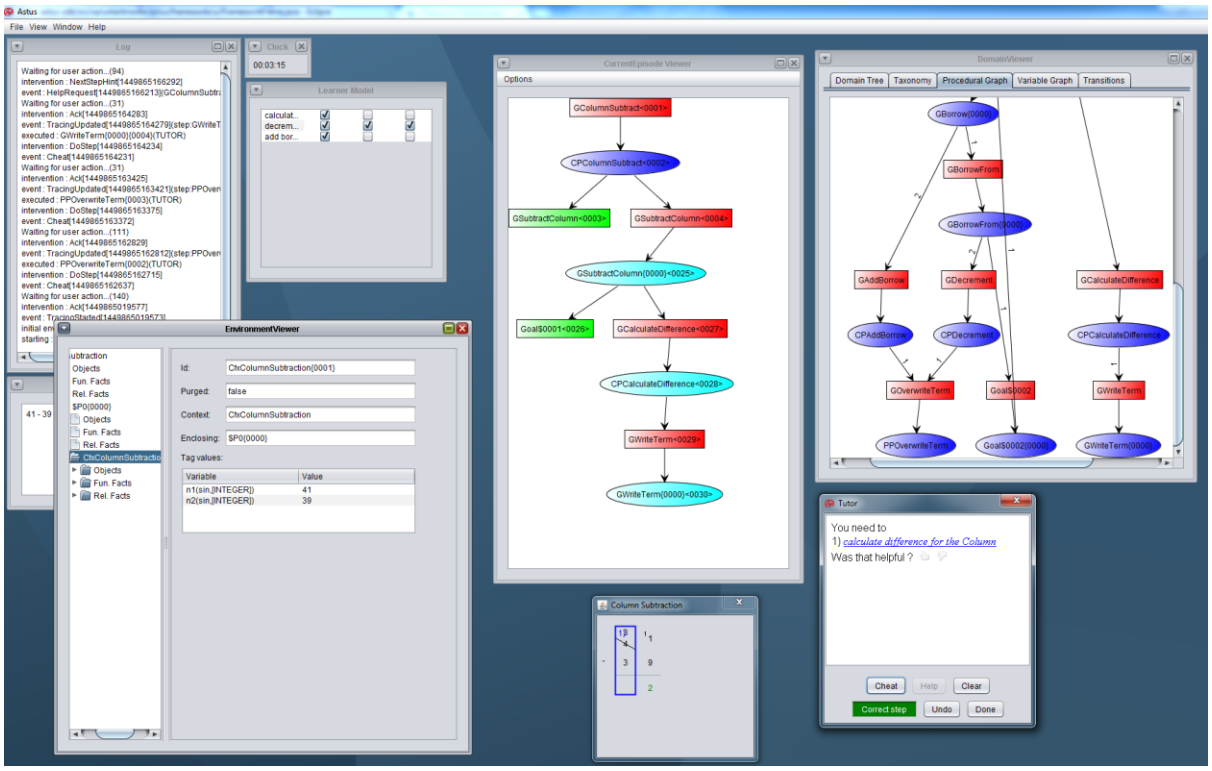

Figure 20 - Un aperçu des outils offert aux auteurs avec le MTT pour la soustraction.

Limitations de la version utilisée à l'interne

Des limitations qui sortent du cadre de nos objectifs expliquent pourquoi nous n'avons pas encore produit une version de la plateforme Astus disponible à l'externe :

- la stratégie pédagogique de base, qui a été développée en parallèle avec les MTT utilisés lors des expérimentations, ne supporte pas adéquatement la génération d'interventions pour tous les types d'éléments du métamodèle de la tâche;
- nous ne disposons pas d'une banque suffisante de MTT éprouvés qui servirait de vitrine pour convaincre la communauté d'utiliser Astus plutôt que CTAT qui est la première plateforme accessible et qui tire profit de la notoriété des CT;

- la gestion des utilisateurs, des entrées/sorties et des versions, quoique suffisante pour un usage interne, n'est pas adaptée à un usage externe (une composante client-serveur devrait être ajoutée et des fichiers XML devraient être remplacés par une BD, etc.);

- le laboratoire ASTUS ne dispose pas des ressources nécessaires pour fournir du support à des utilisateurs externes (il faudrait minimalement fournir la documentation requise pour créer un MTT et utiliser les mécanismes d'extension).

Travaux futurs en vue de faciliter l'adoption d'Astus

Bien qu'elles ne constituent pas à proprement dit des travaux de recherche et qu'elles demanderaient des ressources importantes, ces améliorations faciliteraient l'adoption d'Astus par des utilisateurs externes :

- éliminer la dépendance à Jess dont la licence est contraignante, par exemple en utilisant le système de production Drools[135];

- permettre de répartir le modèle de la tâche[136] dans plusieurs fichiers plutôt qu'un seul;

- développer un plugiciel Eclipse qui supporte explicitement les DSL d'Astus;

- remplacer Swing par JavaFX, la nouvelle bibliothèque de création de GUI de Java qui offre une allure plus moderne.

MTT créés à l'aide de la plateforme

Au cours du développement d'Astus, différents MTT ont été développés, et ce, à différents stades, soit des MTT :

- comparables aux exemples de CTAT : soustraction en colonne (le modèle est présenté à l'annexe B), addition de fractions, création d'un nuage de points;

[135] http://www.drools.org/
[136] Par contexte (voir chapitre 4) par exemple.

- utilisés lors des expérimentations : conversion de nombre en virgule flottante (Figure 21) et manipulation d'un ABR (Figure 22);
- plus ambitieux et encore en développement : « soins critiques » en sciences infirmières et « génie génétique » en biologie.

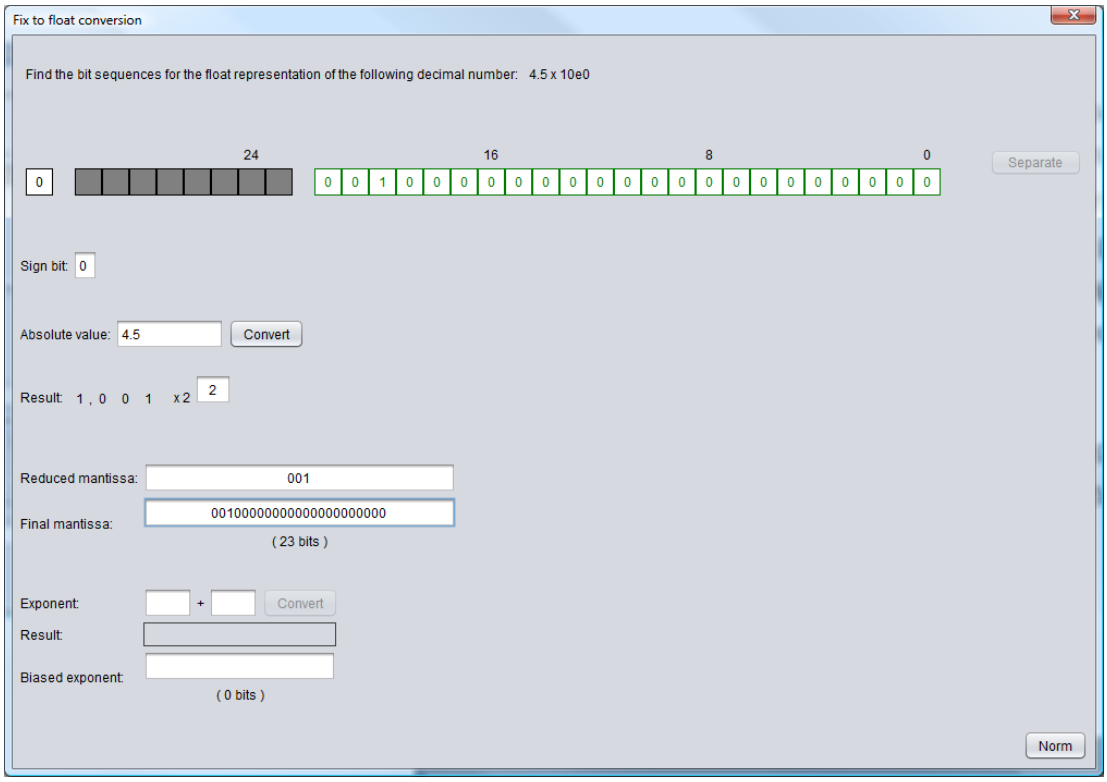

Figure 21 - Un aperçu du MTT pour la conversion de nombres à virgule flottante

De plus, pour vérifier l'implémentation d'Astus, nous avons créé des MTT de tests qui permettent de faire la couverture du code source, le profilage et les tests automatisés.

Les tâches de soustraction en colonnes et de fractions sont des classiques de la littérature des STI. Bien que nos versions soient en fait plus sophistiquées que les exemples fournis avec CTAT, ce sont des preuves de concept qui n'ont pas été testées auprès d'apprenant. L'environnement pour la soustraction s'inspire de celui de Sierra en offrant des actions d'une fine granularité (nous les décrivons au chapitre 4), ce qui permet de reproduire un grand

nombre d'incompréhensions parmi celles recensées par la littérature. Le MTT pour la création d'un nuage de points, bien qu'il soit grosso modo équivalent[137] à celui créé avec le TDK est également une preuve de concept qui n'a pas été testé auprès d'apprenants.

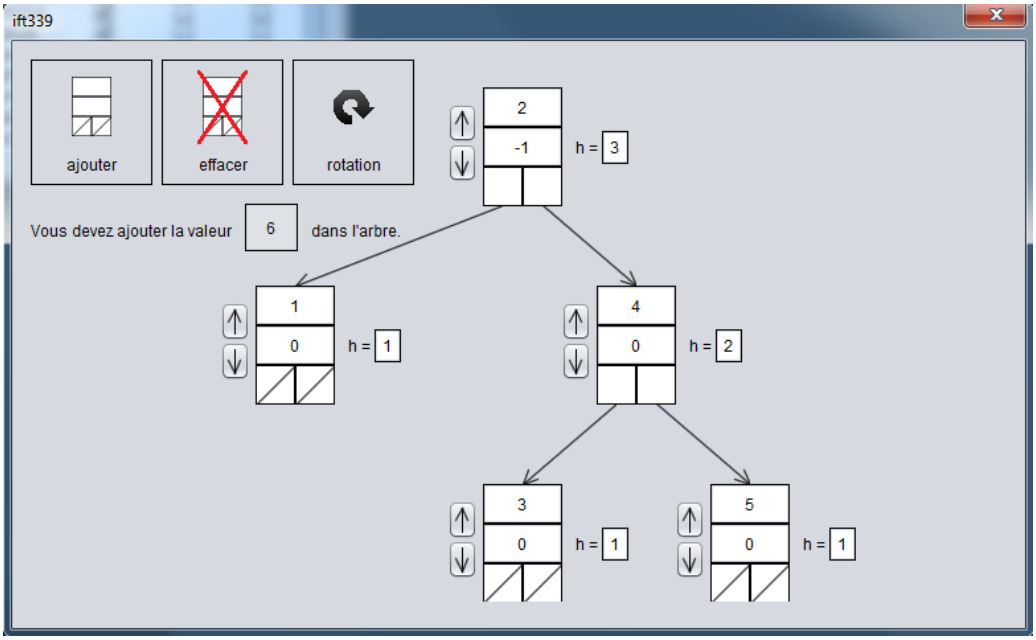

Figure 22 - Un aperçu du MTT pour la et manipulation d'un ABR

Le MTT de « soins critiques » cible la tâche correspondant à l'évaluation primaire réalisée par une infirmière lorsqu'un patient victime d'un trauma arrive aux urgences. Le protocole de l'évaluation primaire, qui est divisé en cinq grandes étapes (A-B-C-D-E), comprend plusieurs actions qui correspondent soit à des observations ou à des interventions (des observations justifient une intervention, puis de nouvelles observations sont effectuées pour vérifier les effets de l'intervention). Bien que le MTT à proprement dit n'ait pas été testé, son environnement a été testé par plus d'une centaine d'infirmières (en utilisant une stratégie pédagogique minimale qui détermine succinctement la prochaine action lors d'une demande d'aide). L'intérêt d'un tel MTT vient d'une part des limites des mannequins traditionnels et d'autre part des couts et de la logistique requise par les mannequins interactifs.

[137] En plus d'avoir éliminé l'identification des variables, nous avons aussi éliminé les questions d'analyse qui suivent la création du nuage de points pour les mêmes raisons.

Le MTT pour « génie génétique » est quant à lui encore en phase de conception, car il requiert pratiquement tous les éléments de la plateforme et par conséquent, une stratégie pédagogique complète. Toutefois, son environnement sera basé sur un laboratoire simulé que nous avons développé et testé auprès de centaines d'apprenants. Ce laboratoire simulé (voir Annexe A) permet de remplacer soit des exercices « papier et crayons » limités, soit un laboratoire réel trop couteux (étant donné le nombre d'apprenants). Ce laboratoire permet de créer une carte de restriction, une étape nécessaire à l'identification d'une molécule d'ADN inconnue. Ce procédé, qui est basé sur l'interaction entre la molécule et des enzymes de restrictions, produit une image sur laquelle les fragments, de différentes tailles, de la molécule se sont déplacés. À partir de la mesure de ces déplacements (migrations), on peut inférer la position des différents sites de coupures propres à chacune des enzymes, ce qui forme la carte de restriction.

Les expérimentations menées au département d'informatique

L'objectif des expérimentations était de mesurer l'efficacité des interventions générées, et ce, à l'aide de MTT qui sont aussi, sinon plus sophistiqués que les CT offerts en exemple par CTAT. Comme celui de « génie génétique », les MTT pour la conversion de nombres à virgule flottante et la manipulation d'ABR ont été créés en adaptant des exercices « papier et crayon ». Pour le premier, il s'agit de construire la représentation binaire (selon le standard IEEE 754**)** d'un nombre décimal ou inversement, d'obtenir un nombre décimal à partir de sa représentation binaire. Pour le deuxième, il s'agit d'ajouter un nœud dans un ABR de type AVL et de faire le(s) rotation(s) requise(s) pour rééquilibrer l'arbre à la suite de l'ajout.

Une série de cinq expérimentations à petite échelle [138] a été effectuée au département d'informatique :

[138] En moyenne, la taille des groupes (expérimentaux et contrôles) était d'une vingtaine d'étudiants.

- Une première expérimentation visait à comparer l'efficacité des indices générés (groupe expérimental) par rapport à des indices prémâchés (groupe contrôle) avec le MTT pour la conversion.

- Les deux expérimentations suivantes visaient à mesurer l'efficacité des indices générés (groupe expérimental) avec le MTT pour la conversion (le groupe contrôle n'avait pas accès aux indices).

- Les deux dernières expérimentations visaient à mesurer l'efficacité des rétroactions pour les actions non tracées (groupe expérimental) avec le MTT pour la manipulation d'ABR (le groupe contrôle n'avait pas accès aux rétroactions).

Bien que les résultats obtenus indiquent que les interventions sont efficaces [195, 196, 198, 199], des expérimentations à plus large échelle sont nécessaires pour affirmer que les MTT d'Astus, avec leurs interventions générées, sont aussi, sinon plus efficaces que les CT avec leurs interventions prémâchées. Par ailleurs, il faudra procéder à des expérimentations basées sur des tâches qui ne sont pas tirées du domaine de l'informatique. En effet, les étudiants en informatique sont susceptibles d'avoir un biais favorable envers les interventions générées puisqu'ils sont plus à même de les comprendre même si la qualité du langage naturel est parfois discutable. Il sera donc important de mener des expérimentations avec le MTT de « soins critiques » ou celui de « génie génétique », mais pour ce faire, il faut d'une part développer la stratégie pédagogique de base. En effet, plusieurs types d'éléments de connaissance utilisés dans ces MTT n'ont pas été sollicités par les expérimentations en informatique. D'autre part, il faut améliorer d'un cran la qualité du langage naturel dans les interventions. En fonction de notre expérience, nous jugeons qu'il est possible, du moins pour l'anglais, de le faire sans devoir recourir aux outils de l'IA pour la génération automatique de textes[139] (GAT).

[139] En fait, les gabarits sont une approche élémentaire de GAT, les approches plus avancées ont recours à des grammaires et à des modèles statistiques à la fois pour la planification du contenu et la réalisation de surface.

Finalement, au-delà des résultats sur l'efficacité des interventions, c'est le processus entourant les expérimentations en lui-même, qui contribue à montrer que la version d'Astus développée lors de nos travaux répond aux exigences énoncées en introduction (notre deuxième objectif). En effet, puisque le processus d'expérimentation, incluant la création des MTT, a été mené de façon relativement indépendante du développement d'Astus, il est comparable au processus de chercheurs externes qui utiliseraient Astus. D'une part, la création des MTT n'a pas requis de créer une version spécialisée de la plateforme. De plus, même si la stratégie pédagogique de base a été développée en parallèle avec la création des MTT qui ont servi aux expérimentations et qu'elle ne supporte pas tous les types d'éléments de connaissance, elle est demeurée indépendante de la didactique de ces derniers. D'autre part, le fait que la plateforme est suffisamment stable pour fonctionner adéquatement lors d'expérimentations à l'interne montre qu'elle est plus qu'un prototype, même si elle n'offre pas encore toutes les fonctionnalités nécessaires pour des expérimentations externes.

Chapitre 4

Analyse d'Astus

Ce chapitre présente une analyse des caractéristiques d'Astus. Même si dans ce chapitre nous présentons Astus plus concrètement qu'au chapitre 2, nous évitons autant que possible de faire directement référence à l'implémentation afin de maintenir l'analyse au niveau conceptuel. Cette analyse nous permet d'atteindre notre deuxième objectif, soit de montrer que la version d'Astus présentée au chapitre 3 répond aux exigences énoncées en introduction. Au chapitre 5, nous tirons profit de cette analyse pour comparer les MTT créés avec Astus avec d'autres MTT et STI afin d'atteindre notre troisième et dernier objectif.

4.1 Le modèle de la tâche

Dans cette section, nous présentons d'abord les éléments sémantiques, puis les éléments procéduraux pour terminer avec les éléments épisodiques. En plus des exemples fournis dans ce chapitre, le modèle du MTT pour la soustraction en colonnes est présenté à l'annexe B.

4.1.1 Les connaissances sémantiques

Les principales structures du métamodèle qui permettent de définir les éléments sémantiques sont : les termes[140] de la TBox (les concepts, les relations et les fonctions) et les assertions de la ABox (les objets et les faits). Par rapport aux CT, on peut considérer que les premiers sont l'équivalent des types de WME et que les derniers sont l'équivalent des instances de WME.

[140] Au sens d'élément terminologique et non au sens logique ou algébrique.

Nos choix de conception par rapport aux structures sont inspirés d'UV, de notre revue de la littérature sur la représentation des connaissances [7, 25, 36, 37, 38, 51, 70, 78, 99, 109, 137, 153, 192, 206, 269] et de notre volonté de supporter le paradigme du tuteur tout en maintenant les efforts de création dans un ordre de grandeur comparable à ceux des CT. Nous justifions ces choix de conception au fur et à mesure où il est pertinent de le faire, par exemple, certains éléments sémantiques se justifient à l'aide d'éléments procéduraux.

Les termes et leurs variables

Comme pour les types de WME, les termes contiennent des variables (« *slot* ») et peuvent être définis à partir d'un ou plusieurs termes existants (contrairement aux types de WME). À la différence des « *slot* » dans les types de WME, les variables des termes ont un type. À la différence des instances de WME qui n'instancient qu'un seul type de WME, les assertions peuvent instancier plusieurs termes. En ce sens, les termes sont similaires aux cadres, et les assertions aux instances de ces derniers. Plutôt que de définir ces structures à l'aide d'un formalisme mathématique traditionnel, nous les définissons plutôt à l'aide d'une grammaire abstraite (Figure 23 et Figure 24). Les termes, comme leurs variables, ont des propriétés (facettes) qui précisent leur sémantique.

Un terme *abstrait* est un terme qui ne peut pas être instancié directement. Par exemple, dans le MTT pour la manipulation d'ABR, le concept `Arbre` est abstrait (le concept `ArbreAVL` spécialise ce dernier)[141]. Un terme *ouvert* signifie qu'on doit tenir compte d'instances de ces termes qui ne se retrouvent pas explicitement dans la KB. Autrement dit, par défaut la KB est fermée (« *closed-world assumption* ») comme une base de données plutôt qu'être ouverte (« *open-world assumption* ») comme les ontologies OWL ou plus généralement la FOL. Les termes ouverts sont généralement utilisés pour représenter les résultats des inférences et les saisies (nous y revenons par la suite). Par exemple, dans le MTT pour l'addition de fractions, le concept `Fraction` est ouvert. L'accès aux assertions des termes ouverts est limité (p. ex.

[141] De façon à pouvoir intégrer les arbres rouges-noirs lors de travaux futurs.

le langage de requête ne permet pas d'obtenir tous les objets qui instancient un concept ouvert[142]).

```
concept ID abstrait? incomplet? ouvert? (spécialise ID+)? primitif?
ÉNONCÉS SCRIPT? {
 (attribut ID inversible? TYPE PAR_DÉFAUT ÉNONCÉS SCRIPT?)*
 (partie ID TYPE PAR_DÉFAUT ÉNONCÉS)*
 (valeur inversible? TYPE PAR_DÉFAUT SCRIPT?)?
 (ID-ATTRIBUT inversible? PAR_DÉFAUT ÉNONCÉS)*
 (redéfini ID-ATTRIBUT TYPE PAR_DÉFAUT ÉNONCÉS SCRIPT?)*
 GABARIT-PRIMITIF?
}

attribut ID TYPE PAR_DÉFAUT ÉNONCÉS

objet ID? ÉNONCÉS {
 (ID-CONCEPT [ID-VAR OBJET]*)*
 (ID-CONCEPT OBJET?)? // pour le cas où il y a seulement une valeur
}
```

Figure 23 - Grammaire pour les concepts, les attributs et les objets du domaine (les symboles non terminaux sont en majuscules et en italique, on retrouve les symboles usuels : les parenthèses pour regrouper, l'* pour répéter et le ? pour rendre optionnel).

Les objets du métamodèle

Les objets du métamodèle représentent les données primitives (nombres entiers, nombres décimaux, symboles et booléens). Ces *objets primitifs* sont utiles dans les tâches où recourir à une modélisation sous forme de concepts et d'objets du domaine n'aurait pas d'intérêt pédagogique.

Une variable peut être définie comme étant *multivaluée* afin de lui associer comme valeur un objet du métamodèle qui représente une *collection d'objets*[143]. Ces dernières peuvent contenir des objets du métamodèle ou des objets du domaine, elles peuvent être ordonnées ou non et contenir ou non des doublons. Par exemple, en « soins critiques », le concept

[142] Un drapeau permet d'obtenir les instances contenues dans la KB au moment de la requête.
[143] « *bunch* » dans [216].

`Observation` contient une variable `états` qui est une collection, non ordonnée et sans doublon, d'objets instanciant le concept `État`.

Les concepts et les objets du domaine

Les concepts sont les principaux éléments de la TBox et les objets sont les principaux éléments de la ABox. Les **concepts** agissent d'une part comme un type pour définir les variables du graphe procédural et d'autre part comme une structure pour regrouper les objets nécessaires pour décrire une abstraction qui a un intérêt pédagogique[144]. Les objets correspondent aux arguments des actions et aux valeurs des variables dans le graphe épisodique. Les objets sont généralement définis par l'intermédiaire de sous-objets auxquels ils sont associés au travers d'un concept (les objets du métamodèle sont un contrexemple). Par exemple, dans le MTT pour la soustraction en colonnes, on retrouve plusieurs objets instanciant le concept `Colonne` qui sont constitués de deux objets instanciant le concept `Opérande` et un objet instanciant le concept `Résultat`; ces deux concepts spécialisent le concept abstrait `Cellule` dont la variable `valeur` est un `entier` et l'attribut `valeurs_précédentes` est une collection ordonnée d'`entiers`.

Pour chaque terme et pour chacune des variables qui s'y retrouvent, on peut ajouter un énoncé en langage naturel. Dans l'exemple de la soustraction, pour l'anglais, on ajouterait au concept `Colonne` l'énoncé « column » et pour les variables `haut` et `bas` (de type `Opérande`), on retrouverait respectivement les énoncés « top » et « bottom ».

Dans les concepts, les variables sont désignées comme des *caractéristiques*[145] auxquelles on peut attribuer une valeur par défaut[146]. En fait, Astus distingue trois types de caractéristiques : les attributs, les parties et les valeurs. Les **attributs** sont les caractéristiques de base, leur sémantique est neutre. Si un attribut est marqué comme étant une *clé*, il y a, dans la KB, au plus un objet avec une valeur donnée pour cet attribut (autrement dit, l'attribut est inversible).

[144] Les concepts ne correspondent donc pas nécessairement à des *chunk*s au sens (psychologique) d'ACT-R.
[145] Qui ne correspondent pas nécessairement à des propriétés intrinsèques en philosophie.
[146] Elle n'a pas de sémantique dans le modèle, ce n'est que du sucre syntaxique qui réduit les efforts de création.

Si un attribut est défini globalement, il peut être ajouté une seule fois à une hiérarchie de concept; de même, un objet ne peut pas instancier des concepts issus de hiérarchies différentes qui ont des attributs globaux communs[147]. Les **parties** ont une sémantique plus précise que les attributs, elles représentent une relation « partie – tout » au sens large (c.-à-d. qu'elle s'applique autant à des objets abstraits que concrets) qui est inversible, mais qui n'est pas réflexive ni transitive. En contrepartie, le langage de requête permet de manipuler les parties, par exemple de vérifier qu'un objet fait partie d'un autre et d'obtenir tous les objets qui sont des parties d'un objet. Certaines sous-catégories d'objets, comme les objets primitifs, ne peuvent pas être associées à un objet par l'intermédiaire d'une partie. Dans l'exemple de la soustraction, les `Cellules` sont des parties d'une `Colonne`. Les parties, comme les objets primitifs et les collections ne sont pas nécessaires, mais réduisent les efforts de modélisation, d'ailleurs il en est de même pour les valeurs**.** En effet**,** ajouter une caractéristique de type **valeur** à un concept équivaut à ajouter à un concept un attribut global non typé dont l'identificateur serait `valeur`.

Un concept *incomplet* est un concept pour lequel on peut omettre de fournir la valeur d'une caractéristique lorsqu'il est instancié[148] (si la caractéristique est multivaluée, la valeur peut être partiellement fournie). Par exemple, en soustraction le concept `Résultat` est incomplet puisque sa `valeur` n'est pas fournie initialement. Lorsque le concept associé à la caractéristique n'est pas ouvert, Astus vérifie que l'objet qui est fourni comme valeur était déjà présent dans la KB[149] au moment où l'objet incomplet est créé.

Il est possible de *redéfinir* le type, la valeur par défaut et l'énoncé d'une caractéristique. Le nouveau type doit être un sous-type du type initial. Dans le MTT pour la manipulation d'ABR, le concept `ArbreAVL` redéfinit le type de la caractéristique `racine` provenant du concept `Arbre` de `Noeud` à `NoeudAVL` qui spécialise le concept `NoeudBinaire` pour

[147] Pour contourner cette limitation, il faut ajouter un concept qui unit les deux hiérarchies et qui redéfinit adéquatement le type des attributs globaux communs.
[148] C'est un objet *inconnu* du métamodèle qui prend la place de la ou des valeurs manquantes
[149] Les objets du métamodèle sont toujours considérés comme présents dans la KB

lequel la valeur par défaut des caractéristiques `arbreDeGauche` et `arbreDeDroite` est le `pointeurNul`.

Quand un script est associé à un terme, des instances sont créées automatiquement lorsque ce terme est manipulé par le langage de requête. Puisqu'un tel script est « boite noire », il doit représenter une connaissance procédurale qui est supposée maitrisée par tous les apprenants ou qui est sans intérêt pédagogique (nous donnons des exemples de tels scripts pour des fonctions dans la section suivante). Dans les cas où l'appel à la demande d'un script associé à un terme n'est pas efficace[150], il est possible de remplacer le script par une règle de production[151] qui est également « boite noire »[152]. À la différence des scripts, les règles de production ne sont pas déclenchées lorsque les termes associés sont manipulés par le langage de requête, mais plutôt lorsque l'état de la KB est modifié[153]. Dans le MTT pour la manipulation d'ABR, le concept `Feuille` est associé à une règle de production afin qu'il soit instancié par tous les objets de types `NoeudBinaire`, dont les caractéristiques `arbreDeGauche` et `arbreDeDroite` ont comme valeur le `pointeurNul` (un objet qui instancie le concept `Pointeur`). Comme nous l'avons évoqué au chapitre 2, nous décrivons dans la section des connaissances procédurales comment il est possible de définir un terme à l'aide d'une expression du langage de requête.

Dans un concept, un script (avec la même sémantique que les scripts associés aux termes) peut être associé à un attribut ou à la valeur[154]. La valeur associée à une caractéristique à laquelle on a associé un script est automatiquement calculée lorsqu'un objet qui instancie le concept est créé ou modifié.

---

[150] Un script peut, à l'aide du langage de requêtes, faire appel à d'autres scripts associés à des termes, par conséquent, l'exécution d'un script peut être considérée comme une forme de chainage arrière.
[151] Encodée directement sous forme d'une règle de production Jess.
[152] Pour qu'elles soient interprétables, il faut analyser les règles, ce qui n'est pas possible de façon générale [195] (même si on peut construire un système de production avec des règles interprétables, p. ex. ACT-R).
[153] Les règles de production peuvent donc être considérées comme une forme de chainage avant.
[154] Dans la littérature, on parle de « *procedural attachment* » ou de « *deamon* » [39].

Un concept *primitif* est un concept ouvert et complet dont les instances ont une « sémantique de valeur »[155], comme les objets du métamodèle, mais contrairement aux autres objets propres au domaine qui ont une identité indépendante des valeurs associées à leurs caractéristiques. Autrement dit, les objets primitifs du domaine sont nécessairement comparés en fonction du concept qu'ils instancient et de leurs valeurs puisqu'ils n'ont pas d'identité[156]. En « génie génétique », on retrouve par exemple le concept `Mesure` qui a une *valeur* de type `décimal` et un attribut de type `Unité` (dont les deux seules instances sont les objets `kb`[157] et `cm`). Comme les objets du métamodèle, ces objets primitifs ne sont pas nécessaires, mais ils sont utiles lorsqu'on ne veut jamais distinguer deux objets qui sont structurellement équivalents. Puisque les objets primitifs représentent généralement le résultat des inférences, l'auteur peut ajouter, aux concepts primitifs, un énoncé qui permet au module pédagogue de décrire les objets sans qu'ils soient représentés dans l'environnement (p. ex. pour décrire une `Mesure` sous la forme « 5.0 cm »).

Les relations, les fonctions et les faits

```
relation ID abstraite? ouverte? (spécialise ID+)? ÉNONCÉS {
 (place ID TYPE ÉNONCÉS)*
 (redéfini ID-VAR TYPE)*
 SCRIPT?
}

fonction ID abstraite? ouverte? (spécialise ID+)? ÉNONCÉS {
 (argument ID TYPE ÉNONCÉS)*
 (image ID TYPE ÉNONCÉS)?
 (redéfini ID-VAR TYPE ÉNONCÉS)*
 SCRIPT?
}

fait ID-FCT_REL+ { (ID-VAR OBJECT)+ }
```

Figure 24 - Grammaire pour les relations, les fonctions et les faits

[155] http://en.wikipedia.org/wiki/Value_semantics
[156] Ils correspondent donc grosso modo à des termes fermés en logique.
[157] Kilobase (mille paires de bases d'ADN).

Les relations et les fonctions[158] permettent de créer des associations supplémentaires entre des objets. Même si on peut exprimer une fonction sous la forme d'une relation, la forme fonctionnelle représente plus fidèlement l'intention de l'auteur dans plusieurs cas. Une **relation** est définie à l'aide d'au moins une place (variable); il n'y a pas de limites conceptuelles au nombre de places, mais les relations sont généralement binaires (dans certains cas, une relation à une place représente plus fidèlement l'intention de l'auteur qu'un concept). Par exemple, en soustraction, la relation `àDroite` dont les deux places sont des `Colonnes`. Les places sont typées comme les caractéristiques d'un concept et elles peuvent être multivaluées (ce qui permet de modéliser des relations dont l'arité est variable).

Quand on définit une **fonction**, on doit distinguer les variables qui correspondent aux arguments et à l'image[159] (une fonction sans arguments est une constante nommée). Une même fonction peut à la fois avoir une image multivaluée et être partielle. Par conséquent, c'est sa sémantique, qui apparait dans l'énoncé en langage naturel, qui la distingue ultimement d'une relation (p. ex. l'énoncé pour une fonction dont l'image est multivaluée est au pluriel).

Une particularité des fonctions est que leur script (ou leur règle/expression) peut également créer la valeur qui est associée à l'image, c'est-à-dire un objet qui représente le résultat d'une inférence (autrement dit, un objet qui n'est pas représenté graphiquement dans l'environnement). Des relations et des fonctions peuvent ensuite être instanciées en utilisant un tel objet comme valeur pour un(e) argument/place. En soustraction, un exemple de fonction avec un script est `différence` qui a deux arguments de type `Opérande` et une image de type `entier` (la valeur de l'image est obtenue en manipulant la valeur des `Opérandes` à l'aide du langage hôte). En addition de fractions, la fonction `réduite` dont l'argument est une `Fraction` permet d'obtenir un objet de type `Fraction` qui est créé à

[158] Ou symboles relationnels et fonctionnels (ou encore prédicat et foncteur [192]).
[159] Par abus de langage. À strictement parler, il faudrait dire résultat ou valeur.

l'aide du langage de requête (le `num` et le `dénom` de ce dernier sont obtenus en manipulant le `num`. et le `dénom` de l'argument à l'aide du langage hôte).

Contrairement aux caractéristiques des concepts, les places, les arguments et les images peuvent être anonymes (ils ont un identificateur par défaut). De plus, une relation ou une fonction qui en spécialise une autre ne peut pas ajouter des places/arguments, mais seulement les redéfinir.

Les axiomes[160] sur les termes

Par défaut, les concepts de hiérarchies distinctes sont disjoints, c'est-à-dire qu'un objet ne peut pas instancier des concepts de hiérarchies distinctes. On peut donc d'une part, **joindre** des concepts qui sont disjoints et d'autre part, **disjoindre** des concepts issus d'une même hiérarchie. En partant d'un concept abstrait, on peut donc créer une « disjonction mutuellement exclusive » de concepts. Par exemple, en soustraction, à partir du concept abstrait `Cellule`, on retrouve les concepts disjoints `Résultat` et `Opérande`.

Puisque les concepts et les relations peuvent être manipulés comme des prédicats, il est parfois judicieux de définir un concept ou une relation opposée. Un **concept opposé** signifie qu'un objet qui n'instancie pas le concept original instancie nécessairement le concept opposé et vice-versa. Une **relation opposée** signifie que soit il existe un fait pour une association « places – objets » donnée pour la relation originale, soit il en existe un pour la relation opposée. Par exemple, en addition de fractions, les concepts `FractionRéductible` et `FractionIrréductible` sont opposés.

De plus, il est possible de définir une relation ou une fonction comme l'**inverse d'une relation** (binaire) ou d'une fonction (unaire) existante. Si un fait instancie une relation qui a un inverse, cela signifie qu'il existe dans la KB un fait qui instancie celle-ci en inversant les valeurs des places (si l'inverse est une fonction, la première place correspond à l'image et la

[160] Au sens de la littérature sur les ontologies et non au sens de la logique.

deuxième à l'argument). Par exemple, en soustraction, la relation `àGauche` est définie comme l'inverse de la relation `àDroite`. Dans le cas d'une fonction, c'est l'argument et l'image qui sont inversés (l'inverse est une fonction) ou qui prennent la première et la deuxième place (l'inverse est une relation).

Les contextes et les environnements qui ont une dimension physique

La KB est subdivisée en plusieurs contextes, un **contexte** regroupe des assertions (objets et faits) qui ont la même portée et la même durée de vie. Un contexte peut contenir des étiquettes (variables) qui permettent aux scripts de retrouver des objets directement plutôt que d'utiliser le langage de requête. En plus du métacontexte qui contient les objets du métamodèle, il y a un contexte pour le domaine, un contexte propre à la tâche et un contexte pour chacun des états de l'environnement traversés lorsqu'une instance de la tâche est effectuée (un état correspond par exemple à une fenêtre principale ou à une sous-fenêtre modale[161]). Par exemple, en « génie génétique », le contexte du domaine contient toutes les `Enzymes`. Dans la manipulation d'ABR, le contexte du domaine contient le `pointeurNul`[162] et le contexte de la tâche contient l'`Arbre` et la donnée à ajouter ou à retirer (un `entier`).

La définition de la tâche spécifie le contexte initial et une instance de celui-ci est automatiquement créée avec l'instance de la tâche[163]. Par la suite, des actions de l'apprenant correspondant à des transitions de contexte peuvent créer de nouvelles instances de contextes (nous y revenons à la section 4.2).

Bien que les scripts (aussi les règles/expression) associés aux relations et aux fonctions permettent de créer des assertions, et qu'il en est de même pour les scripts associés aux actions, la plupart des faits et des objets sont généralement créés à l'aide des scripts associés à l'instanciation des contextes. Ces scripts, comme les scripts d'actions, sont créés à l'aide du

[161] Ce terme est privilégié par rapport à « boite de dialogue » puisqu'il est plus général.
[162] Les identificateurs des objets du contexte du domaine ont une portée globale.
[163] Un script d'amorce injecte des éléments dans l'instance du contexte initial depuis la définition de la tâche.

langage de manipulation (qui est décrit à la section 4.1.2). Par exemple en soustraction, les objets de type `Colonne`, `Opérande` et `Résultat`. Contrairement aux assertions du métacontexte, du contexte du domaine et de celui de la tâche, les assertions des contextes qui correspondent à des états de l'environnement peuvent être retirées (objets et faits) ou modifiées (objets). Ces changements sont effectués à l'aide du langage de manipulation dans les scripts associés aux actions.

Pour supporter les environnements qui ont une dimension physique, en particulier les environnements partiellement observables, le langage de manipulation permet d'associer à un objet du domaine, ou à une instance de contexte, des données « boite noire »[164] qui ne font pas partie de la KB, c'est-à-dire qui ne sont pas accessibles au module pédagogue (ni aux autres modules). Autrement dit, ces données forment une **couche de simulation** sous la KB, auxquels seuls les scripts, eux-mêmes « boites noires », ont accès. Par exemple, en « soins critiques », l'état du patient se trouve dans la couche de simulations, tandis que les `Observations` de l'infirmière se trouvent dans la KB (c'est la définition d'une instance de la tâche qui établit l'état du patient et qui l'associe à l'instance du contexte initial).

Remarques et travaux futurs

On peut s'étonner que le modèle de la tâche ne permette pas d'exprimer des restrictions sur les variables des termes, mais qu'il distingue les attributs des parties. Ces choix ont été faits de façon à obtenir un compromis judicieux entre la complexité du modèle et les avantages des structures ajoutées. Par exemple, les parties procurent un avantage pour le processus de diagnostic[165]. Un autre exemple est la redéfinition des types qui réduit l'usage d'opérations de conversion de type dans les expressions. Il en est de même pour les axiomes, par exemple, les « opposés » et les « inverses » qui permettent au module pédagogue de simplifier les expressions. Autrement dit, nous avons tenté d'éviter d'introduire des mécanismes dont la principale fonction est de vérifier le modèle, comme nous l'avons indiqué au chapitre 2,

[164] C'est-à-dire un `Object` Java.
[165] Elles introduisent des caractéristiques inversibles implicitement.

l'approche que nous proposons pour la vérification est basée sur les métadonnées accumulées lors de l'exploration automatique.

En plus d'avoir renoncé aux inférences logiques comme nous l'avons évoqué au chapitre 2, nous avons dû laisser de côté des mécanismes qui pourront faire l'objet de travaux futurs. En effet, il est difficile de trouver un juste équilibre entre les mécanismes de modélisation offerts et les efforts de création requis. Sans faire une énumération exhaustive, nous en évoquons quelques-uns :

- permettre de réifier les termes sous forme d'objets afin d'introduire plusieurs niveaux dans le modèle (de façon similaire aux cadres [39]), c'est-à-dire permettre qu'un terme soit à la fois instancié au niveau inférieur et qu'il soit une instance d'un terme au niveau supérieur [270];

- ajouter une couche ontologique[166] au modèle, en particulier distinguer les différentes catégories de concept (types, rôles, etc.) selon leurs propriétés ontologiques (identité, rigidité, etc.) [102];

- réifier le métamodèle, en particulier les termes, à l'aide d'une ontologie de haut niveau[167], car ils ne sont représentés au niveau de l'implémentation que sous forme de classes Java;

- en plus des termes ouverts et incomplets, supporter la disjonction et la négation classique[168]. Par exemple, un attribut `entier` dont la valeur est inconnue, mais pour lequel on sait qu'elle n'est pas 4, ou encore pour lequel on sait qu'elle est 3 ou 5.

[166] Il ne faut pas confondre celle-ci avec la couche OWL évoquée au chap. 2 et qui a été ajoutée au MGC.
[167] http://en.wikipedia.org/wiki/Upper_ontology
[168] On retrouve la négation par l'échec dans le langage de requête.

4.1.2 Les connaissances procédurales

Comme pour les éléments sémantiques, nous définissons les principaux éléments procéduraux à l'aide d'une grammaire abstraite (Figure 25) où on retrouve les éléments communs à tous les types de procédures complexes (les éléments particuliers propres à chaque type et les expressions sont décrits informellement pour alléger le texte). À la Figure 26, on retrouve un exemple de graphe procédural, c'est-à-dire celui de la soustraction.

```
But ID? abstrait? (spécialise ID+)? contextes ID+ absorbant? exclusif?
ÉNONCÉS {
 (entrée ID TYPE ÉNONCÉS)*
 (sortie ID TYPE ÉNONCÉS)?
 (redéfini ID-VAR TYPE ÉNONCÉS)*
 (satisfait CONDITION)?
 PROCEDURE*
}

TYPE_PROC ID? but ID-BUT invalide? {
 (préconditon CONDITION)?
 (variable ID OPÉRATION)*
 (sous-but ID-BUT [|ID-VAR:ID-VAR|*] |sortie ID|? exclusif? {
       |aller ID-TRANSITION [(ID-VAR:OPÉRATION)*]|?
       |retour ID-TRANSITION [(ID-VAR:OPÉRATION)*]|?
 })*
 (sous-but BUT …)*
 (postcondition CONDITION)?
 (résultat OPÉRATION)?
 GABARIT-INVALIDE?
}

primitive ID? but ID-BUT {
(préconditon CONDITION)?
SCRIPT
}

définition ID-TERME EXPRESSION

transition ID empile | dépile | succède de ID-CTXT (à ID-CTXT)?
 SCRIPT SCRIPT-RETOUR? {
 (entrée ID TYPE ÉNONCÉS) *
}
```

Figure 25 - Grammaire des principales structures procédurales.

Les expressions

Une *expression* du langage de requête combine des opérateurs/prédicats[169] qui sont appliqués à des variables[170] de façon à représenter une perception, un rappel ou une inférence. Lorsque l'expression produit une valeur booléenne, elle est nommée *condition* et lorsqu'elle produit tout autre type de valeur[171], elle est nommée *opération*. Sans faire une énumération exhaustive, le langage de requête permet :

- d'obtenir le seul ou tous les objets[172] qui instancient un concept;
- d'obtenir des objets en naviguant au travers des objets à l'aide des caractéristiques;
- d'obtenir des objets en fournissant toutes les variables, sauf une qui agit comme une variable libre, d'une relation ou fonction (autrement dit, appliquer une relation ou une fonction);
- de vérifier que de tels objets existent, autrement les prédicats qui correspondent aux opérateurs précédents;
- d'obtenir le nombre de ces objets;
- de vérifier que les objets obtenus instancient un concept;
- de comparer les objets obtenus à d'autres objets, possiblement à l'aide d'une relation d'équivalence propre au domaine (il y a une relation d'équivalence prédéfinie propre à chacune des sous-catégories d'objets) ou encore à l'aide d'une relation d'ordre propre au domaine (il y en a une relation d'ordre prédéfinie pour les types `entier` et `décimal`);

[169] Dans la littérature des agents BDI, on parle de « *query*/*test goal* » [73].
[170] D'un point de vue plus formel, on peut considérer les objets littéraux comme des opérateurs dont l'arité est 0.
[171] Lorsque les valeurs booléennes sont utilisées pour représenter un élément du domaine (p. ex. un bit dans le MTT pour la conversion de nombres en virgule flottante), on peut retrouver des opérations dont le résultat est une valeur booléenne.
[172] Pour alléger le texte, par la suite, nous écrivons simplement « des objets ».

- de trier les objets obtenus à l'aide d'une relation d'ordre (voir le point précédent);
- de filtrer/transformer les objets obtenus à l'aide d'une condition/opération;

De plus, des prédicats comme le « et », le « ou » et le « non » logiques et des opérateurs qui manipulent les collections d'objets (combiner, éliminer les doublons, etc.) permettent de composer des expressions. Par ailleurs, par défaut, la portée de ces expressions est le contexte courant, mais ils peuvent aussi cibler le contexte de la tâche ou celui du domaine, ou encore une collection d'objets déjà obtenue. Finalement, puisqu'il est impossible de fournir tous les prédicats/opérateurs qui pourraient être utiles, un mécanisme d'extension permet aux auteurs d'en ajouter.

Remarque sur les expressions

On peut remarquer que contrairement aux langages de requêtes basés sur l'algèbre relationnelle, celui d'Astus n'a pas d'opérateur de projections et ne permet d'associer des objets qu'à une seule variable libre. Par conséquent, il évite la création de tuples, autrement dit des objets qui n'instancient aucun concept du domaine défini explicitement dans le modèle. Or, selon notre expérience, de tels objets sont trop difficiles à manipuler lors de la génération des interventions.

Puisqu'il est impossible de délimiter une fois pour toutes la portée du langage de requête, il faut, au fur et à mesure où de nouvelles tâches sont modélisées, évaluer quelles sont les opérateurs et les prédicats qui sont suffisamment réutilisables pour être intégrés. Par exemple, des travaux futurs pourront ajouter des opérations arithmétiques de base sur les objets représentant des nombres entiers. Cette évaluation est d'autant plus cruciale lorsqu'on augmente l'expressivité du langage de requête, puisque le module pédagogue doit être en mesure de produire un énoncé qui décrit adéquatement les expressions qui en découlent. Par exemple, des travaux futurs pourront évaluer l'ajout d'un opérateur conditionnel ternaire (le « ? : » dans les langages de programmation dérivés du C).

Les buts

Un but est défini à l'aide de variables et il peut spécialiser un ou plusieurs buts abstraits (qui ne peuvent pas être instanciés). Un but qui spécialise un autre but peut uniquement redéfinir les variables dont il hérite. Comme nous l'avons évoqué au chapitre 2, les instances de but, avec les instances de procédures, forment les épisodes. Un épisode, peut être *actif* (par défaut), *inactif* (en présence d'une contrainte d'ordre) *satisfait* ou encore *abandonné* (nous y revenons dans la prochaine section). Parmi les *paramètres* (variables), on retrouve des *entrées* et possiblement une *sortie*[173]. Les valeurs des entrées pour une instance de but proviennent de l'instance de la tâche (but racine) ou de l'instance de la procédure parente; pour la sortie, la valeur provient de l'instance de la procédure qui a satisfait l'épisode.

Un but spécifie les contextes dans lesquelles il peut être instancié, par conséquent le but racine doit pouvoir être instancié dans le contexte initial spécifié par la tâche.

L'*énoncé* (pour une langue donnée) associé à un but doit exprimer l'intention sous la forme d'une phrase d'action. Pour préciser la phrase, l'énoncé peut contenir des références aux paramètres (@*ID-VAR*) qui sont remplacés par les valeurs des paramètres dans l'épisode faisant l'objet d'une intervention (à l'image des gabarits de message des CT). Par exemple, en soustraction, on peut associer au but `Soustraire`, dont le paramètre d'entrée `col` est une `Colonne`, l'énoncé « subtract @col » pour l'anglais.

Un but *absorbant* est un but pour lequel, lorsqu'un épisode est sur le point d'être créé, il faut d'abord vérifier s'il existe un épisode actif équivalent (même sous-but et mêmes valeurs de paramètres d'entrée). Le cas échéant, c'est l'épisode équivalent qui est ajouté à l'instance de la procédure de l'épisode parent plutôt qu'un nouvel épisode (autrement dit, comme les faits et certaines sous-catégories d'objets, les buts absorbants ont une « sémantique de valeur »).

[173] D'un point de vue plus formel, on peut considérer que les buts qui n'ont pas une sortie explicite ont une sortie dont le type est le « type unité » (http://en.wikipedia.org/wiki/Unit_type).

Un but *exclusif* est un but qui doit être satisfait sans qu'un autre épisode puisse progresser, autrement dit, un but exclusif équivaut grosso modo à un but qui nécessite une transition de contexte, sans les changements correspondants à l'environnement.

On peut associer à un but une *condition de satisfaction* qui est évaluée lorsqu'un des épisodes qui l'instancie est sur le point d'être actif. Si la condition se révèle alors être vraie, l'épisode est directement satisfait, sinon l'épisode est actif. Dans ce dernier cas, la condition est réévaluée lorsque l'épisode est satisfait et si elle se révèle être fausse, Astus considère que le graphe procédural est invalide (c.-à-d. que l'auteur doit le corriger). Par exemple, dans le MTT pour la manipulation d'ABR, le but `Équilibrer` a une condition de satisfaction qui vérifie qu'un `Arbre` est un `ArbreÉquilibré` (selon la différence des `hauteurs` des sous-arbres).

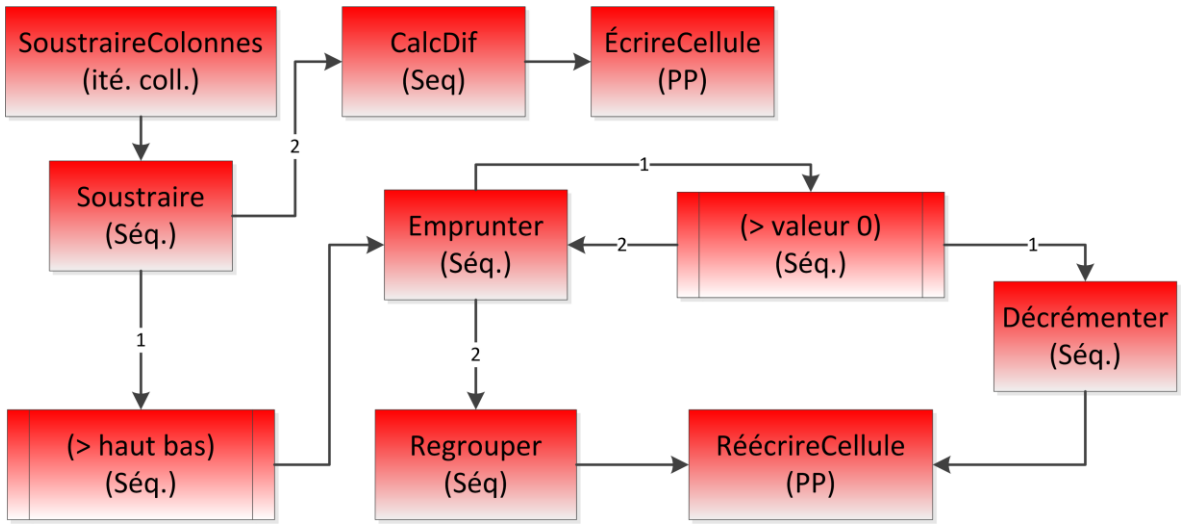

Figure 26 - Le graphe procédural pour la soustraction. Puisqu'une seule procédure par but est représentée, le type des procédures est indiqué sous les buts entre parenthèses (les buts avec les barres additionnelles sont des buts avec une condition de satisfaction et les indices indiquent des contraintes d'ordres entre les sous-buts).

Les procédures

Comme nous l'avons évoqué au chapitre 2, une procédure, complexe ou primitive, peut satisfaire un seul but, mais plusieurs procédures peuvent satisfaire le même but. Par défaut, une procédure est toujours applicable, mais on peut ajouter une **précondition** à cette dernière qui est vérifiée lorsqu'un épisode contenant une instance de but qu'elle satisfait est actif. La précondition doit se révéler vrai pour que la procédure soit instanciée afin de pouvoir satisfaire l'épisode.

Les procédures primitives

Une *procédure primitive* (PP) est définie à l'aide du but qu'elle satisfait et du script qui est exécuté lorsque l'action correspondante est produite (les mécanismes qui mènent à la production d'une action sont détaillés dans les sections suivantes). Ce script est écrit à l'aide des éléments de base du langage de requête, du langage hôte (Groovy) qui est plus expressif et plus efficace que ce dernier et du langage de manipulation. Sans faire une énumération exhaustive, le langage de manipulation permet :

- de créer un objet ou un fait ou encore retirer un objet ou un fait;
- de faire en sorte qu'un objet instancie un concept supplémentaire (ou le contraire, faire en sorte qu'un objet n'instancie plus un concept);
- de remplacer la valeur d'une caractéristique d'un objet;
- de révéler la valeur inconnue d'une caractéristique d'un objet (ou le contraire, l'oublier);
- de modifier les données de la couche de simulation associée à un objet.

Par exemple, en soustraction, le script de la PP qui satisfait le but `ÉcrireCellule` révèle l'objet associé à la valeur d'une `Cellule` tandis que la PP qui satisfait le but `RéécrireCelulle`, en plus de remplacer l'objet associé à la `valeur` d'une `Cellule`,

ajoute l'ancien objet à la collection associée à l'attribut `valeurs_précédentes`. En « soins critiques », la PP qui satisfait le but `InstallerCollier`, en plus de créer une `Observation` contenant l'`État CollierBienInstallé`, met à jour les données de la couche simulation qui représentent l'état du patient.

Comme nous l'avons évoqué au chapitre 2, en dehors du script, les effets abstraits d'une action sur la KB sont inconnus, c'est-à-dire que seul les effets concrets produits par l'exécution d'une action sont connus (nous revenons sur les avantages et les désavantages de cette approche au chapitre 5). Ces effets concrets sont d'ailleurs conservés pour faciliter l'implémentation de la rétroaction minimale et de l'annulation. Astus exige de l'auteur que l'exécution de tous les scripts soit déterministe afin d'assurer le bon fonctionnement des modes d'exécution permettant de reproduire les actions d'une instance d'une tâche déjà effectuée [212] (à la section 4.2, nous indiquons comment il est tout de même possible d'introduire un facteur aléatoire au sein des instances de tâches).

Les procédures complexes

Peu importe sa nature (séquence, sélection, itération), une **procédure complexe** (PC) est avant tout définie par ses sous-buts. Un **sous-but** est défini par un but et une association des paramètres d'entrée de ce dernier à des opérations qui manipulent les variables accessibles au sein de la PC. Ces variables sont : les paramètres d'entrée du but parent; les variables locales et les sorties des sous-buts précédents dans les procédures où il y a une notion d'ordre. Minimalement, les variables locales permettent d'éviter d'évaluer plusieurs fois la même expression (lorsque plusieurs sous-buts doivent recevoir en entrée la même valeur), mais l'auteur peut aussi leur ajouter un énoncé qui fournit une sémantique plus précise.

Peu importe le type de PC, Astus ne permet pas que l'exécution d'une instance de procédure crée deux épisodes actifs qui sont équivalents. Nous avons ajouté cette contrainte pour éviter une forme d'ambigüité qui complexifie trop le suivi par rapport à l'expressivité supplémentaire obtenue.

Si le but d'un sous-but a une sortie, on doit créer dans le sous-but, la variable à laquelle est associée la valeur produite par le *résultat* (une opération) de la PC qui satisfait l'épisode qui instancie ce sous-but. Quand il y a plusieurs procédures qui peuvent satisfaire un but ayant une sortie, on peut prédéfinir le résultat dans le but[174].

Si le but d'un sous-but doit être instancié dans un contexte différent, on doit spécifier au sein de ce dernier la transition pour l'*aller* et si nécessaire, la transition pour le *retour* au contexte courant (en fonction du type de la transition, qui peut être à sens unique, nous y revenons à la section 4.2). Par exemple, dans le MTT pour la manipulation d'ABR, la PC qui satisfait le but `ÉquilibrerÀGauche` a recours à des transitions dans le sous-but associé au but `FaireRotationÀGauche` pour entrer et sortir du `ContexteDeRotation`.

Une PC peut comporter une **postcondition** pour modéliser une procédure qui même si elle est applicable à première vue, peut échouer en cours de route. Suivant une telle impasse, un « retour arrière » est nécessaire. Un exemple simple est celui, évoqué au chapitre 1, dans le MTT pour la création d'un nuage de points, qui consiste à choisir l'échelle d'une variable de façon à ne pas avoir un trop petit nombre ou un trop grand nombre d'étiquettes à ajouter sur l'axe correspondant du graphique (nous revenons sur le rôle pédagogique des impasses au chapitre 5).

En plus des PC marquées comme *invalides*, qui représentent des incompréhensions, il existe des procédures qui représentent des optimisations et des procédures propres à une variation particulière de la tâche. Par exemple, dans l'addition de fractions, on retrouve une PC invalide qui consiste à additionner les dénominateurs ($1/2 + 1/3 \rightarrow 1/5$); dans la soustraction, on retrouve une PC qui correspond à une optimisation consistant à remplacer un zéro, sur lequel on fait un emprunt, directement par un 9 plutôt que d'abord ajouter 10, puis le décrémenter; dans le MTT sur la manipulation d'ABR, on retrouve des procédures propres aux arbres AVL. Pour les PC marquées comme invalides, l'auteur peut ajouter un gabarit de message pour la rétroaction sur l'erreur (voir section 4.3) comme dans les CT. Contrairement

[174] On doit le faire s'il y a à la fois une PP et une PC puisqu'une PP ne peut pas redéfinir le résultat.

aux règles des CT, il n'y a pas de priorité sur les procédures, plus loin dans cette section nous indiquons comment la procédure choix (une forme de sélection) remplace celle-ci (toutefois, lorsque le module pédagogue doit choisir une procédure arbitrairement, il le fait de façon déterministe, c'est-à-dire en suivant l'ordre dans lequel les procédures ont été définies par l'auteur).

Les différents types de PC

Un premier type de PC est la **séquence** qui applique des contraintes d'ordres entre des sous-buts, soit implicitement, parce que les buts ont des sorties, soit explicitement. Ces contraintes d'ordres sont rigides, c'est-à-dire que si elles ne sont pas respectées, l'action produite est non tracée. De plus, par défaut, Astus considère qu'il y a implicitement une contrainte d'ordre souple, c'est-à-dire une préférence qui guide le module pédagogue dans ses interventions, entre tous les sous-buts, selon l'ordre dans lequel ils sont définis par l'auteur. Ce dernier peut spécifier, avec le drapeau `nonOrdonnée`, que le module pédagogue doit ignorer l'ordre de définition. Un exemple de séquence avec des contraintes d'ordre explicite est la PC qui satisfait le but racine `FaireÉvaluationPrimaire` en « soins critiques », dont les sous-buts `FaireLeA`, `FaireLeB(C/D/E)` sont totalement ordonnés. En addition de fractions, un exemple d'une contrainte implicite découlant d'un but avec une sortie, est celui des deux sous-buts définis à partir du but `CalculerNumÉquivalent` qui dépendent du sous-but défini à partir du but `CalculerDénomCommun`.

Parmi les procédures d'*itération*, on retrouve, la boucle conditionnelle, la boucle sur une collection d'objets et la répétition. Une procédure d'itération n'a qu'un seul sous-but, lorsqu'elle est instanciée, elle engendre un nombre variable d'épisodes, tous au même moment, ou un à la fois. La boucle conditionnelle, dont il y a deux variantes, « **tant que »** et « **jusqu'à »**, est définie par une condition qui est vérifiée au départ, puis chaque fois qu'un épisode qui instancie son sous-but est satisfait. Dans le cas du « tant que », l'exécution se termine quand la condition se révèle être vraie, c'est l'inverse pour le cas du « jusqu'à ». La **boucle sur une collection d'objets** est définie par une opération qui produit une collection

d'objets et une variable locale qui agit comme itérateur (interne). Un épisode du sous-but est instancié pour chacun des objets, d'ailleurs Astus vérifie que le sous-but associe (directement ou non) l'itérateur à un des paramètres du but. Si la collection correspondante est non ordonnée, tous les épisodes sont actifs initialement, sinon un seul épisode est actif à la fois, créant ainsi une contrainte d'ordre[175]. La **répétition** est définie par une opération qui produit le nombre d'épisodes à créer. En soustraction, un exemple d'une boucle sur une collection d'objets est la PC qui satisfait le but racine et qui fait une itération sur la séquence des `Colonnes` obtenues en utilisant la relation `àDroite` comme relation d'ordre. Dans la conversion de nombres à virgule flottante, on retrouve des boucles conditionnelles pour faire le passage d'une base à l'autre.

Parmi les procédures de *sélection*, on retrouve la procédure conditionnelle et la procédure choix. La procédure **conditionnelle** est définie à l'aide de clauses formées d'une condition et d'un sous-but, et possiblement d'une clause `autrement`, qui contient uniquement un sous-but. Les conditions sont vérifiées une à une, dans l'ordre où les clauses sont définies, jusqu'à ce qu'une d'entre elles se révèle être vraie, ce qui détermine lequel des sous-but est instancié. Si toutes les conditions se révèlent être fausses, c'est le sous-but de la clause `autrement` qui est instancié (si cette dernière n'est pas définie, Astus indique que le graphe procédural est invalide). Dans le MTT pour la manipulation d'ABR, on retrouve une procédure conditionnelle pour le but `Équilibrer` qui a une condition de satisfaction. Si cette dernière se révèle être fausse, la conditionnelle indique s'il faut `ÉquilibrerParLaGauche` ou plutôt `ÉquilibrerParLaDroite` (selon la différence des `hauteurs`).

La procédure **choix** est d'abord définie soit comme une boucle sur une collection d'objets, soit à l'aide de plusieurs sous-buts, puis avec des euristiques. Une *euristique* est une condition associée à un ou plusieurs sous-buts. D'une part, lors de l'exécution d'une instance de la procédure, les euristiques déterminent les sous-buts qui sont instanciés, puis déterminent les épisodes qui sont abandonnés sur-le-champ (le fait de filtrer les choix en deux temps

[175] Si la collection contient des doublons, un seul épisode par objet est activé à la fois.

simplifie les conditions, en particulier en éliminant des négations). D'autre part, les euristiques définissent la relation d'ordre sur les épisodes restants. Cette relation d'ordre a préséance sur l'ordre de définition quand le module pédagogue doit déterminer un épisode à effectuer ou à suggérer à l'apprenant. En effet, contrairement aux autres procédures, même si plusieurs épisodes sont créés, un seul d'entre eux peut être satisfait, les autres sont abandonnés dès qu'il y en a un qui progresse. À la base, les euristiques sont plus expressives que les priorités des règles des CT, mais ce qui les rend encore plus expressives que ces dernières, c'est la capacité de manipuler les buts et les épisodes (nous y revenons dans la sous-section des connaissances épisodiques). Par exemple, dans le MTT pour la création d'un nuage de points, une procédure choix sur une collection (les échelles possibles) a des euristiques pour filtrer les échelles qui ont déjà été essayées et celles qui sont nécessairement trop petites ou trop grandes en fonction des essais précédents. Un autre exemple est un MTT pour la résolution d'une équation correspondant au CT créé avec le TDK, où les propriétés des règles sont remplacées par une procédure choix. Un dernier exemple est le MTT de « génie génétique », où plusieurs procédures choix sont requises (choix des digestions, choix de l'hypothèse de travail, choix de l'inférence à appliquer sur l'hypothèse, etc.).

On peut remarquer que les euristiques peuvent filtrer tous les sous-buts et les épisodes. Dans ce cas, par défaut, Astus signale que le graphe procédural est invalide. Par contre, l'auteur peut, à l'aide du drapeau `retourArrière`, faire en sorte que l'absence d'épisode actif provoque plutôt le comportement associé à une postcondition qui se révèle être fausse (c.-à-d. que l'instance de la procédure échoue et qu'un « retour arrière » est nécessaire pour sortir de l'impasse).

On peut aussi remarquer qu'en l'absence d'euristiques, le choix est arbitraire, or Astus offre un autre mécanisme, les « objets motifs », qui est généralement mieux adapté pour représenter les choix arbitraires, nous le décrivons dans la section 4.1.3.

Les définitions procédurales

Une définition procédurale permet de rendre un terme opérationnel (comme le fait un script ou une règle) en l'associant à une expression (une condition pour un concept ou une relation, une opération pour une fonction). Or, contrairement aux scripts/règles qui sont des « boites noires », les expressions sont interprétables, par conséquent le module pédagogue peut produire une intervention qui décrit le terme. En fait, une définition permet au module pédagogue de présenter d'abord le terme de façon « boite noire », c'est-à-dire en utilisant l'énoncé qui lui est associé, puis de le décrire. Par exemple, dans la manipulation d'ABR, on retrouve la fonction `nouvelleHauteur`[176] qui fait l'objet d'une définition. Si l'apprenant ignore comment obtenir l'image de cette dernière, le module pédagogue produit une intervention qui décrit l'expression qui lui est associée : `plus(hauteurCourante, 1)` [177]. Par exemple, « the *new height* is the *current height plus* one »[178].

Pour l'auteur, un bon indicateur pour déterminer quand une définition procédurale doit être utilisée plutôt qu'un script/règle, est la présence d'un prédicat ou d'un opérateur dans l'énoncé qui serait associé au terme. Par exemple, un énoncé contenant les mots « and », « or » ou « not ».

Il est important de ne pas confondre les définitions procédurales avec les définitions que l'on retrouve dans OWL. En effet, ces dernières offrent un compromis différent entre l'expressivité et la validité des inférences logiques qui en découlent par rapport à celles offertes par les définitions procédurales et les inférences « rationnelles » qui en résultent. Effectivement, les définitions procédurales sont beaucoup plus expressives, mais elles n'offrent pas de mécanismes de validation à proprement dit (quoique les axiomes peuvent être considérés comme tels, les termes opposés par exemple).

---

[176] Son argument est un `NoeudAVL` et son image un `entier`.
[177] `hauteurCourante` est une variable locale associée à la `hauteur` de l'instance de `NoeudAVL`
[178] « new height » est l'énoncé associé à la fonction `nouvelleHauteur` et « current height » est l'énoncé associé à la variable `hauteurCourante`.

Structure procédurale et buts anonymes

De façon générale, nous parlons du graphe procédural, mais pour une tâche donnée, lorsque la structure procédurale (ou une partie de celle-ci) correspond à un arbre plutôt qu'un graphe, il est possible de définir les procédures au sein des buts, de définir les sous-buts « sur place » et ce récursivement jusqu'aux PP (voir Figure 25). Cette façon de faire est particulièrement utile en présence de buts anonymes, c'est-à-dire des buts pour lesquels l'auteur ne fournit pas un énoncé. En effet, en l'absence d'un énoncé pour un but, lorsque le module pédagogue doit décrire un épisode dans une intervention, il utilise la condition de satisfaction, s'il y en a une, et les sous-buts (qui sont tirés d'une instance de procédure applicable). Par exemple, un but satisfait par une séquence, qui est utilisé dans un sous-but d'une itération ou d'une sélection n'a pas d'énoncé s'il n'y a pas d'abstraction correspondante dans le domaine (autrement dit, le but et la séquence ne font que regrouper des sous-buts qui ont des abstractions correspondantes dans le domaine). Par exemple, dans le cas de la soustraction, si le but racine est anonyme, un indice peut être généré à l'aide de la boucle sur une collection, de l'opération qu'elle contient (obtenir et trier les `Colonne`) et de son sous-but (voir Figure 26) : « For each of all the *column*s from the most *on right* to the most *on left*, *substract* it »[179].

Travaux futurs

Comme nous l'avons fait pour les éléments sémantiques, nous évoquons des travaux futurs à propos des éléments procéduraux.

- Même si grâce aux séquences partiellement ordonnées et aux boucles sur des collections non ordonnées, plusieurs épisodes peuvent être exécutés concurremment, nous souhaitons supporter plus explicitement l'exécution concurrente des PC. En effet, il faudra que des épisodes puissent se comporter comme des coroutines. C'est-à-dire, que des valeurs d'entrée soient mises à jour au fur et à mesure que les épisodes produisent des valeurs de sortie.

[179] Les énoncés sont en italiques.

- La modélisation du MTT de « soins critiques » nous a permis de constater que modéliser des variantes procédurales à l'aide de plusieurs procédures sous un même but n'est pas toujours convivial pour l'auteur (de plus, ces variantes créent des chemins ambigus qui affectent la performance du suivi). En donnant accès à l'auteur à des drapeaux qui spécifient que des sous-buts sont `optionnels` ou `répétables`, on réduirait les efforts de l'auteur (et on facilite un traitement plus efficace des chemins ambigus).

4.1.3 Les connaissances épisodiques

Par connaissances épisodiques, nous désignons les épisodes (qui sont formés d'une instance de but et d'une ou plusieurs instances de procédures) et les objets épisodiques décrits par la suite. Le Tableau 3 et le Tableau 4 récapitulent les différents états possibles des épisodes. À la Figure 27, on retrouve un exemple de graphe épisodique du MTT pour la soustraction.

Tableau 3 - États possibles des instances de but

État	**Description**
Active	État normal tant qu'aucune instance de procédure n'est terminée.
Réussie	Satisfaite à l'aide d'une PC valide.
Échouée	Satisfaite à l'aide d'une PC invalide (incompréhension).
Inactive	Pas encore active (contrainte d'ordre).
En attente	Était active, mais suspendue par une transition de contexte ou un but exclusif.
Annulée	Était active, réussie ou échouée, mais la procédure a été annulée.

Les instances de but peuvent être manipulées, à l'aide du langage de requête, comme des faits. On peut vérifier leur existence et en extraire des objets, c'est-à-dire les valeurs associées aux paramètres d'entrée et la valeur associée à la sortie s'il s'agit d'épisodes satisfaits (pour les épisodes dont le but a une sortie). Par exemple, dans le MTT pour la création d'un nuage de points, comme nous l'avons évoqué précédemment, la procédure choix rejette les épisodes correspondant au choix d'une échelle qui ont été satisfaits puis annulés.

Tableau 4 - États possibles des instances de procédures

État	**Description**
Non applicable	Les préconditions ne se sont pas avérées.
En exécution	Amorcée, mais n'est pas terminée.
Tracée	Pas encore amorcée.
Rejetée	Une autre instance de procédure a été utilisée pour satisfaire l'instance du but.
Terminée	Tous les sous-buts sont de la PC ont été satisfaits ou la PP a été appariée à une action dans l'environnement.
Annulée	Tous les sous-buts ont été annulés[180].
Suspendue	L'instance de but parent est suspendue.

Les objets épisodiques

Lorsque le but racine ou le but associé à un sous-but est instancié au sein d'un épisode, si la valeur d'un des paramètres d'entrée est un objet du contexte courant (c.-à-d. un objet qui peut être modifié et retiré), l'épisode conserve une copie superficielle de cet objet que nous désignons comme un *mémento*. Les mémentos permettent au module pédagogue de faire référence, dans une intervention, à l'état passé d'un objet modifié ou encore à un objet qui a été retiré, ce qui est particulièrement utile dans le cas d'une rétroaction suite à une action non tracée. En effet, une telle intervention est produite, puis présentée à l'apprenant alors que la KB et l'environnement ont été mis à jour suite aux effets de l'action. De plus, lorsqu'un épisode est satisfait, une seconde copie est effectuée pour obtenir des mémentos qui peuvent être comparés aux mémentos originaux pour analyser les effets sous-jacents de l'épisode dans la KB.

Pour remplacer le mécanisme par défaut de comparaison entre les valeurs des variables du graphe épisodiques et les arguments des actions (qui sont extraits des interactions sous-jacentes), on retrouve des « objets motifs ». Pour créer un « objet motif », on peut, par exemple, fournir une relation d'équivalence, ou une collection d'objets avec lesquels l'objet est comparé tour à tour ou encore les deux. Par contre, une limitation de ce mécanisme est

[180] Y compris les transitions de contexte nécessaires pour les atteindre.

qu'un « objet motif » ne peut pas être lui-même utilisé comme argument dans une expression, c'est-à-dire qu'il doit être acheminé directement vers une PP ou une transition. En effet, puisque les expressions ne sont pas nécessairement inversibles, il ne serait pas toujours possible de retrouver l'objet initial à partir de l'objet terminal lors du suivi. Par exemple, dans le MTT pour l'addition de fractions, une relation d'équivalence permet de créer un « objet motif » représentant tous les dénominateurs communs qui ne correspondent pas au `pgcd` ni au `produit` des `Dénominateurs` (p. ex. 16, 24 et 48 dans l'addition de 1/4 et 3/8). Les « objets motifs » peuvent être considérés comme du sucre syntaxique, puisque le module pédagogue les traite comme un choix arbitraire. En plus de faciliter la modélisation, les « objets motifs » sont préférables à une procédure choix équivalente puisqu'ils évitent la création d'un épisode pour chacun des objets de la collection, ce qui peut entrainer une explosion combinatoire, en particulier lorsque des procédures choix sont imbriquées.

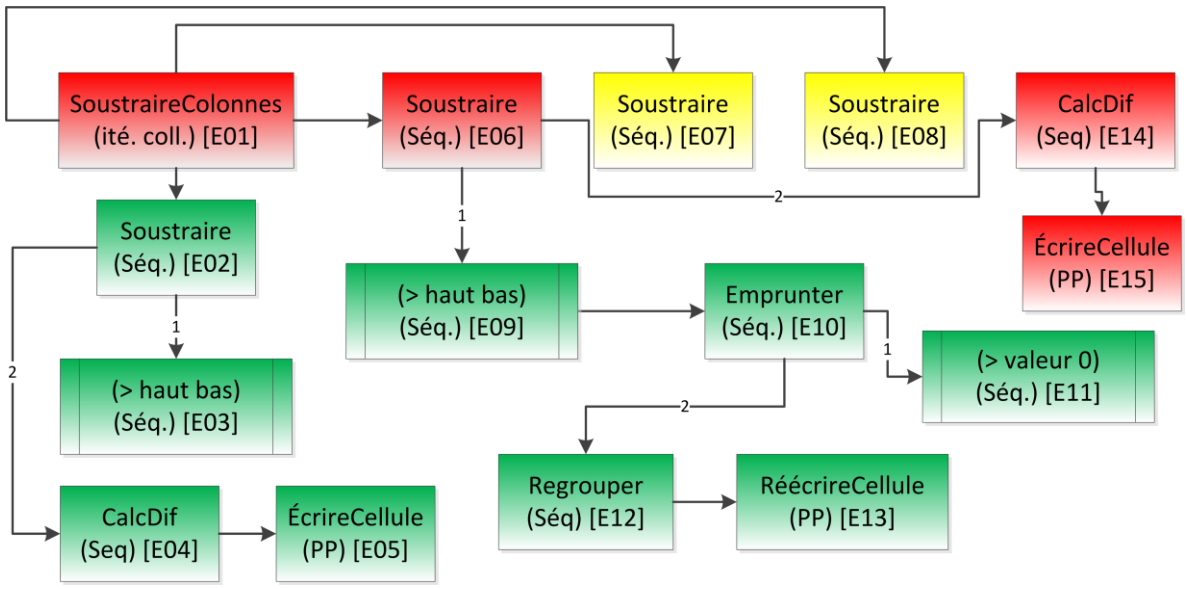

Figure 27 - Le graphe épisodique de la soustraction après que deux actions ont été produites (E05 et E13), la prochaine action correcte possible est E15.
En rouge : actif (tracée), en vert : réussi (terminée) et en jaune : inactif

Remarques sur les connaissances épisodiques

Le graphe épisodique est véritablement un graphe (DAG), et non un arbre, lorsqu'on tente d'activer un épisode et qu'un épisode actif équivalent existe déjà dans le même chemin. À ce moment, l'épisode qu'on tentait d'activer est abandonné et on partage l'épisode existant aux deux instances de procédures, transformant ainsi l'arbre en graphe (autrement dit, on connecte deux chemins). À moins d'être en présence d'un sous-but absorbant, on est donc en situation d'ambigüité. L'ambigüité peut être levée si un sous-épisode contenant un sous-but distinct est amorcé, par contre, dans plusieurs cas, l'ambigüité demeure parce que tous les sous-épisodes sont équivalents (p. ex. les cas où il y a qu'un seul sous-épisode partagé par plusieurs instances de procédures). Pour ces cas, c'est le module apprenant qui doit trancher. Le modèle de l'apprenant minimal accorde le bénéfice du doute à l'apprenant quand il y a une ambigüité entre un chemin valide et des chemins invalides. Cette capacité d'Astus de maintenir plusieurs interprétations[181] des actions produites s'est révélée particulièrement utile pour modéliser des incompréhensions en soustraction, ce qui est significativement plus difficile avec les règles des CT [197]. Un autre exemple simple, en addition de fractions, est lorsque le `produit` des `Dénominateurs` est égal au `pgcd` ou encore à l'`addition` des `Dénominateurs`.

On peut remarquer qu'il y a trois cas dans le graphe épisodique (et le graphe procédural sous-jacent) où on retrouve un nœud avec plusieurs liens sortants pour lequel seul un des chemins est conservé en fin de compte : la procédure conditionnelle, la procédure choix et lorsqu'il y a plusieurs instances de procédures dans un même épisode. Ces trois mécanismes ont toutefois une sémantique différente, c'est-à-dire qu'ils sont interprétés différemment par le module pédagogue. La conditionnelle ne représente pas un choix, il n'y a qu'un seul épisode créé. Lorsqu'il y a plusieurs procédures sous le même but, en particulier lorsqu'il y a plus d'une procédure valide, il y a effectivement une forme de choix arbitraire et implicite. En effet, le module pédagogue décrire les instances de procédures qui se trouvent dans l'épisode, mais il

[181] « *multiple interpretations* » dans la littérature [1].

n'a pas accès à un énoncé propre à chacune des PC pour les différencier ni à un critère de comparaison. C'est uniquement la procédure choix qui correspond non seulement à un choix explicite, mais aussi un choix justifié, grâce aux euristiques. Dans nos travaux précédents [132], nous avons établi l'absence d'une procédure choix comme la lacune la plus importante d'UV et du MGC.

Les travaux futurs

Comme nous l'avons fait pour les connaissances sémantiques et procédurales, nous évoquons des travaux futurs concernant les éléments épisodiques :

- La manipulation des buts et des épisodes dans les euristiques de la procédure choix n'est pas toujours conviviale et il est possible qu'elle ne soit pas suffisamment expressive. Pour améliorer la convivialité, il serait par exemple possible d'ajouter à la procédure choix un mécanisme de mémorisation explicite. Pour améliorer l'expressivité, une piste de solution est que l'auteur, au sein des scripts, marquerait des éléments sémantiques comme étant « persistants » de façon à ce qu'ils demeurent dans la KB à la suite d'une annulation.

- Même si les « objets motifs » réduisent les risques d'explosion combinatoire, l'adoption d'une approche hybride entre l'approche descendante d'Astus et l'approche ascendante des CT sera ultimement nécessaire pour éviter une explosion combinatoire pour certaines tâches. Une telle approche aurait d'abord une phase descendante qui laisse des chemins non parcourus et une phase ascendante qui fait une fouille (bidirectionnelle) une fois qu'une action est produite[182].

[182] Il est aussi possible de prolonger la phase descendante en arrière-plan. Les deux phases peuvent être optimisées en utilisant le modèle de l'apprenant et les métadonnées fournies par l'exploration automatique.

4.2 L'interaction avec l'environnement

Cette section décrit comment Astus applique le patron MVC pour gérer l'interaction avec l'environnement (le modèle correspond aux objets et aux faits dans la KB).

Les vues

Chaque contexte propre au domaine et chaque concept dont les instances sont visibles dans l'environnement doit avoir un script de vue, qui permet de créer et de mettre à jour la *vue* associée à une instance de contexte ou à un objet. Comme nous l'avons évoqué au chapitre 2, chacune des vues contient des sous-vues et des composants qui forment la représentation graphique du contexte ou du concept, et des signaux auxquels on fait référence dans les patrons d'interactions associés aux PP et aux transitions. Ainsi, à partir de la vue d'une instance de contexte, on obtient un arbre composé des vues des objets qui s'y retrouvent (il peut y avoir plus d'une vue d'un même objet au sein d'une même vue, mais une vue a toujours une seule vue parente). Les scripts de vue sont « boite noire », ont accès à la KB à l'aide d'une partie du langage de requête, peuvent accéder aux données de la couche de simulation et tirent profit du langage hôte (comme les scripts d'exécution des PP et des transitions).

Généralement, dans une vue, on retrouve un composant (une fenêtre, ou un panneau) qui est lui-même la racine d'un arbre de composants qui est arrangé par le script, mais parfois, la vue ne fait que créer des composants qui sont par la suite arrangés par la vue parente. Par exemple, en soustraction, c'est le script de vue pour le concept `Colonne` qui, pour chaque instance de ce dernier, crée les panneaux associés aux instances d'`Opérandes` et de `Résultats`, mais c'est le script du contexte qui les arrange (c.-à-d. qui les aligne horizontalement) [87]. Un *signal* est créé dans une vue à l'aide d'un gestionnaire de signal auquel on fournit un ou plusieurs composants, par exemple, en soustraction, on retrouve dans la vue d'une `Cellule` le signal `cellule_cliquée` créé à partir du gestionnaire de clics auquel on fournit un panneau. En plus des gestionnaires fournis par Astus, il est possible de créer des gestionnaires spécialisés.

Le contrôleur

Les gestionnaires, qui partagent le rôle du contrôleur avec le module communication, ajoutent une couche au-dessus des gestionnaires propres à la bibliothèque de création de GUI[183], pour les adapter aux abstractions propres à Astus, comme les objets de la KB. Par exemple, le gestionnaire pour la sélection d'un bouton d'option produit une *interaction* (instance du signal) qui permet d'obtenir l'objet associé au bouton sélectionné.

On peut remarquer que la granularité des objets représentant les données primitives, d'Astus, n'est pas alignée sur les primitives du langage hôte[184] (comme c'est le cas dans les CT), en effet, elle s'aligne plutôt sur les données obtenues des gestionnaires. Par exemple, des nombres décimaux qui peuvent être saisis dans l'environnement plutôt que des nombres à virgule flottante.

Les *patrons d'interactions* sont définis à l'aide d'une expression régulière où l'alphabet correspond aux signaux créés dans les vues. Ces expressions régulières doivent être « sans-préfixe » (« *prefix-free* ») [104], de façon à éviter d'avoir une ambigüité entre deux actions lors du traitement des interactions. En plus des opérateurs de base, Astus permet l'entrelacement (« *interleaving* ») [145], de façon à simplifier l'écriture des patrons où l'ordre entre des signaux n'est pas important. En associant une variable à des signaux[185], on accumule les interactions correspondantes, ces variables sont ensuite utilisées quand les interactions minimalement nécessaires pour satisfaire le patron sont produites par les signaux. En effet, on utilise les variables au sein d'*extracteurs* qui, à partir des interactions, obtiennent les objets devenant les arguments de l'action qui correspond à une PP ou à une transition. L'extracteur par défaut gère tous les cas de base (p. ex. extraire l'objet sélectionné dans un signal issu d'un bouton d'option, extraire l'objet correspondant à la valeur saisie dans une zone texte, etc.), mais Astus fournit aussi des extracteurs pour obtenir l'objet associé à la vue d'où l'interaction est issue, pour obtenir les objets associés à un glisser-déposer ou encore

[183] Swing
[184] Groovy/Java
[185] Certains signaux ne sont pas associés à des variables, ils servent plutôt à rendre le patron « sans préfixe ».

pour obtenir chacune des coordonnées d'un point lorsqu'il y a un clic. De plus, l'auteur peut ajouter des extracteurs spécialisés, ce qui permet d'obtenir des objets en combinant plusieurs signaux. Par exemple, dans le cas d'un patron qui correspond à l'utilisation d'une « double liste de sélection », un extracteur spécialisé combine les signaux associés à ces deux boutons et filtre les interactions correspondantes pour obtenir les objets sélectionnés. Cet exemple témoigne de la souplesse des patrons par rapport aux SAI. En effet, avec CTAT, on aurait été contraint de créer un composant composite pour encapsuler la « double liste de sélection » (ce qui serait équivalent à créer un gestionnaire spécialisé dans Astus).

Les saisies

Astus permet de définir des sous-patrons réutilisables dans différents patrons; tout comme les extracteurs, ces sous-patrons permettent d'obtenir des objets (préexistant ou créés à la demande). En particulier, un sous-patron de saisie permet d'instancier directement un concept en associant les caractéristiques de ce dernier à des extracteurs ou à des sous-patrons. Par exemple, dans l'addition de fractions, lorsque l'apprenant saisit une `Fraction` qui correspond à la réduction d'une `Fraction` existante (un signal est associé à au `numérateur` et un signal est associé au `dénominateur`). Les objets qui représentent des saisies sont généralement comparés à des objets représentant le résultat d'inférences et comme ces derniers, ils ont une « sémantique de valeur » (c.-à-d. que la comparaison se fait directement en fonction des concepts qu'ils instancient et des valeurs associées aux caractéristiques de ces derniers). Par exemple, la `Fraction` saisie par l'apprenant est comparée à l'image de la fonction `fractionRéduite` qui représente une inférence.

Finalement, pour chaque vue où peut survenir une saisie (c.-à-d. la vue qui englobe les signaux du sous-patron associé à la saisie), l'auteur doit ajouter un script qui est appelé lorsqu'une saisie est produite afin de mettre à jour la vue.

Le traitement des interactions et le déclenchement des actions

À la compilation, un automate fini est construit à partir du patron[186]. À l'exécution, tous les automates qui reconnaissent une action progressent jusqu'à ce qu'un d'entre eux atteigne son état final. Puisque les patrons ne sont jamais ambigus, lorsqu'un automate atteint son étal final, tous les automates sont remis à leur état initial (autrement dit, Astus considère qu'une seule action est effectuée à la fois).

L'auteur peut associer, dans le script de vue, un *rappel* (« *callback* ») à un signal qui met à jour la vue lorsque ce dernier produit une interaction. Par exemple, en « soins critiques », lorsque le patron d'une PP, comme `InstallerCollier`, requiert de sélectionner un `Outil`, ce dernier est affiché dès qu'il est sélectionné et non après que la PP soit exécutée. L'auteur peut également ajouter au patron un *signal d'annulation*[187] qui ramène l'automate à son état initial.

Une fois que l'automate d'un patron correspondant à une action (PP ou transition) atteint son état final, l'action correspondante est exécutée telle que nous l'avons décrit précédemment. Les actions exécutées sont conservées et sauvegardées indépendamment du graphe épisodique[188]. Une fois l'action exécutée, une copie superficielle de l'environnement (constituée d'une image et d'un squelette des vues) est conservée avant qu'il ne soit mis à jour, afin que le module pédagogue puisse y faire référence dans ses interventions. Puis, afin de mettre à jour l'environnement en fonction de l'action, le module communication extrait les vues touchées des effets de l'action et exécute les scripts de vue correspondants.

Les transitions

Lorsque l'action correspond à une transition, l'environnement est modifié en fonction du type de transition :

[186] Pour traiter efficacement les patrons avec des entrelacements, des automates hiérarchiques [156] sont utilisés.
[187] Au sens de « *cancel* », à ne pas confondre avec l'annulation (« *undo* ») d'une action.
[188] Dans certains modes d'exécutions, il n'y a pas de graphe épisodique.

- lors d'une transition de type *empile*, une nouvelle instance de contexte est créée et la vue de celle-ci contient une fenêtre modale qui est ouverte par-dessus la fenêtre du contexte courant;

- lors d'une transition de type *dépile*, la fenêtre du contexte courant est fermée (l'instance du contexte est conservée pour faciliter l'implémentation de l'annulation) de façon à ce que le contexte précédent devienne le contexte courant;

- lors d'une transition de type *succède*, la fenêtre du contexte courant est fermée (de même, l'instance du contexte est conservée) et une nouvelle instance de contexte est créée, la vue associée à cette dernière contient une fenêtre qui en s'ouvrant succède à l'ancienne fenêtre.

En plus des fenêtres associées aux vues de contextes, on retrouve deux autres types de fenêtres dans les environnements. Premièrement, des fenêtres flottantes associées à une vue de concept, qui sont utilisées quand il est difficile d'organiser la vue parente au sein d'une seule fenêtre. Deuxièmement, des fenêtres modales, dont l'ouverture et la fermeture correspondent à des signaux dans un patron d'interaction. Par exemple, en « soins critiques », une fenêtre modale qui permet à l'apprenant de sélectionner un `Outil`.

Les environnements qui ont une dimension physique

Pour supporter les environnements qui ont une dimension physique, Astus propose deux mécanismes[189] : les évènements et les interruptions. Un **évènement** est défini par des variables et un script « boite noire ». Les évènements sont instanciés lors de l'exécution des PP, mais leur script n'est pas exécuté immédiatement[190]. Le script d'une PP peut instancier un ou plusieurs évènements et pour ce faire il doit fournir les valeurs des variables de ces derniers. Une fois qu'une PP qui a créé un évènement est exécutée, l'apprenant est averti par

[189] Qui sont encore en développement.
[190] Au chapitre 5, nous précisons pourquoi déclencher des évènements en fonction du temps passé entre les actions (ou encore pendant les actions, si elles ont une durée) est une entreprise d'envergure.

une boite de dialogue qu'un évènement est survenu, c'est à ce moment que le script de l'évènement est exécuté. La KB et l'environnement sont par conséquent mis à jour une fois pour la PP et une fois pour chacun des évènements[191]. Par exemple, en « soins critiques », les évènements `MédecinArrivé` et `PerteDeConscience` qui sont déclenchés après une PP (différente selon l'instance de la tâche, mais généralement après la PP `AppelerMédecin` pour le premier). Certains évènements, comme `MédecinArrivé`, produisent des effets dans la KB qui sont pris en compte dans la suite du déroulement de la tâche. Autrement dit, ils servent uniquement à distinguer les effets directs et indirects des PP afin de préserver la sémantique de ces dernières. Les évènements qui altèrent immédiatement le déroulement de la tâche, comme `PerteDeConscience`, sont gérés à l'aide du mécanisme d'interruption.

Une **interruption** est définie au sein d'une PC à l'aide d'une condition et d'un script comparable à celui qui définit les sous-buts d'une PC. Ce script permet de modifier le comportement d'une PC lorsque la condition s'avère, c'est-à-dire en abandonnant, en annulant ou en créant de nouveaux sous-buts. La condition est vérifiée suite à chaque action où évènement tant que la PC est en exécution (des travaux futurs pourront optimiser cette vérification en ajoutant à la KB des règles équivalentes aux conditions). En « soins critiques », l'évènement `PerteDeConscience` et l'observation de l'état `AucunPouls` déclenchent des interruptions. Dans le premier cas, c'est la PC sous le but racine qui abandonne tous ses sous-buts en cours et qui les réinstancient (autrement dit, l'évaluation primaire recommence du début). Dans le deuxième cas, c'est encore une fois la PC sous le but racine qui abandonne tous ses sous-buts, mais elle instancie plutôt le but `PréparerRéanimation` qui, une fois satisfait, met immédiatement fin à l'évaluation primaire dont la réanimation ne fait pas partie. En « génie génétique », les interruptions faciliteraient la modélisation de la révision des mesures. En effet, lorsqu'une nouvelle mesure rend caduque une mesure précédente, un but `RéviserMesures` pourrait être ajouté (dans ce cas, aucun sous-but n'est abandonné).

[191] Si l'action est annulée, les évènements sont d'abord annulés dans l'ordre inverse de leur déclenchement.

On peut remarquer qu'au lieu d'avoir recours aux interruptions, on pourrait ajouter au graphe procédural un but qui a une condition de satisfaction et qui est instancié après chaque action qui peut faire en sorte que la condition s'avère. Selon notre expérience, les interruptions représentent plus fidèlement les instructions de l'enseignant lorsqu'on ne veut pas inclure dans l'intervention la condition à priori (en « soins critiques », les deux approches sont utilisées, puisqu'on veut mettre l'accent sur certaines vérifications).

Dans le cas particulier des instances de tâches où on doit introduire un facteur aléatoire. Par exemple en « génie génétique » et en « soins critiques », l'environnement doit sembler avoir un comportement stochastique. Pour ce faire, nous proposons simplement d'ajouter directement aux instances de la tâche les données suffisantes dans la couche de simulation (p. ex. un vecteur représentant des valeurs aléatoires). Par exemple, en « génie génétique », ces données peuvent représenter les constantes de migration des gels qui font en sorte que pour chaque gel, les fragments d'ADN se déplacent différemment. En « soins critiques », ces données peuvent représenter le fait qu'un évènement comme « `PerteDeConscience` » survienne ou non après une PP donnée.

Finalement, même si l'annulation est une composante essentielle de l'approche pédagogique d'Astus, on doit pouvoir limiter sa portée dans les environnements qui ont une dimension physique. Par exemple en « génie génétique », une transition vers une nouvelle instance du contexte « `ExamenDuGel` » ne peut pas être annulée afin de maintenir une contrainte de l'environnement réel (le processus réel pour obtenir un gel est long et couteux[192]). Par conséquent, des PP et des transitions peuvent être définies comme étant des *points de non-retour*.

Travaux futurs

Comme pour le modèle de la tâche, nous indiquons des travaux futurs qui visent à enrichir l'interaction avec l'environnement :

[192] http://en.wikipedia.org/wiki/Gel_electrophoresis

- Établir plus formellement les bornes de l'expressivité des patrons. En effet, il apparait difficile de montrer formellement que bien qu'une expressivité comparable à celle d'une expression régulière (ou d'un automate fini) est nécessaire, celle d'une grammaire hors contexte (ou d'un automate à pile) est trop grande.

- Supporter la définition de plusieurs scripts de vue pour un même concept et permettre au module pédagogue de contrôler lequel est utilisé à quel moment. En effet, des travaux montrent que la « représentation multiple » (« *multiple représentation* ») [149, 204], c'est-à-dire représenter un même concept de différentes façons, a une pertinence pédagogique.

- Intégrer une notion de relation spatiale (comme dans Roman Tutor, un STI pour la manipulation du bras de la station spatiale [89]) qui aurait des échos à la fois au niveau des connaissances et des vues, par exemple de façon à ce que l'affichage des `Colonnes` en soustraction soit basé sur le fait que la relation `àGaucheDe` est une relation spatiale.

- Supporter l'utilisation d'UI existants (qui sont exécutés au sein d'un processus indépendant). Pour ce faire, il faut que le module pédagogue ait accès à un modèle minimal des vues[193] qui doit être synchronisé avec l'UI. Si la couche de simulation se trouve également dans le processus indépendant, alors la synchronisation est plus difficile (plutôt que de modifier directement les données, les scripts associés aux PP et aux transitions doivent communiquer avec le processus de l'UI).

4.3 Le suivi de l'apprenant et la sélection de tâches

En plus des modèles de la tâche et de l'UI, nos travaux ont visé à mettre en place les mécanismes de base qui soutiennent l'approche pédagogique d'Astus et les mécanismes d'extension qui permettent de l'adapter en fonction des besoins des chercheurs. Ces

[193] Similaire au squelette des vues qui est conservé après qu'une action est produite.

mécanismes ont permis de développer la stratégie pédagogique de base et un processus de sélection de tâches minimal, mais ils permettront surtout à des chercheurs de développer différentes stratégies pédagogiques et différents processus de sélection de tâches. Même si les expérimentations menées à l'interne indiquent que cette stratégie pédagogique est efficace comme nous l'avons évoqué au chapitre 3, cette dernière, de même que le processus de sélection de tâches minimal, doivent avant tout être vus comme une forme de validation de ces mécanismes.

4.3.1 Le suivi de l'apprenant

Le suivi de l'apprenant, ou la boucle interne, s'articule autour d'étapes qui sont constituées d'évènements pédagogiques et d'interventions. Nous décrivons également dans cette section la composition de la stratégie pédagogique qui doit produire une intervention à partir d'un évènement pédagogique (autrement dit, on peut considérer la stratégie pédagogique comme un modèle des connaissances pédagogiques que le module pédagogue interprète).

Les évènements pédagogiques et les interventions

On retrouve dans la KB des évènements pédagogiques des types suivants :

- « début du traçage » qui est créé une fois que l'environnement est affiché et qui permet au module pédagogue d'intervenir avant la première action de l'apprenant;
- « traçage mis à jour » qui est créé soit à la suite à d'une action dans l'environnement (qu'elle soit correcte ou non) soit à la suite d'une annulation;
- « triché » qui est créé lorsque l'apprenant demande au tuteur d'obtenir la prochaine étape;
- « terminé » qui est créé lorsque l'apprenant signifie au tuteur qu'il croit avoir effectué l'instance de la tâche;

- « demande d'aide » qui est créé lorsque l'apprenant fait une demande d'aide générale (bouton) ou particulière (hyperlien);

- « délai » qui est déclenché lorsque l'apprenant n'a pas interagi depuis un certain temps avec l'environnement ou le tuteur.

Suite à ces évènements, le module pédagogue doit produire une *intervention* qui cible un élément de connaissance. Une intervention peut contenir un message (un document hypertexte), des effets visuels appliqués à l'environnement (p. ex. mettre en surbrillance une vue) et des manipulations de l'UI (environnement et tuteur) propres aux différents types d'interventions. Ces types sont :

- « ok » qui ne contient ni messages ni effets visuels et qui, exceptionnellement, ne cible pas un élément de connaissance;

- « rétroaction » qui contient un message pédagogique qui cible un élément de connaissance associé à l'étape courante;

- « indice » qui contient un message pédagogique qui cible un élément de connaissance associé à une prochaine étape;

- « démonstration » qui produit une action dans l'environnement;

- « annulation » qui annule la dernière action produite;

- « terminé » qui déclenche l'évènement du même nom;

- « fin » qui confirme à l'apprenant que l'instance de la tâche est effectuée (l'apprenant doit alors décider s'il veut effectuer une autre tâche ou non).

De plus, toutes les interventions peuvent aussi configurer l'environnement (p. ex. bloquer celui-ci) et le tuteur (p. ex. « demande d'aide » ou « triché » disponibles ou non). Comme nous l'avons évoqué au chapitre 2, peu importe l'intervention produite, après une action ou

une annulation, le module communication produit la rétroaction minimale (Figure 28 et Figure 29). Celle-ci consiste d'une part à surligner les vues ajoutées/retirées ou modifiées par les effets l'action produite/annulée, le surlignement est soit positif (vert), négatif (rouge) ou neutre (bleu) selon le résultat du traçage. D'autre part, un témoin de la même couleur est visible dans le tuteur, ce qui permet à l'apprenant de toujours voir clairement la rétroaction minimale, peu importe l'état de l'environnement.

Comme le bouton d'annulation généralisé, le bouton pour tricher découle du paradigme du tuteur, car il favorise une communication transparente entre l'apprenant et le module pédagogue. Dans la stratégie pédagogique de base, le module pédagogue produit directement une démonstration de la prochaine étape lorsque l'apprenant demande de tricher.

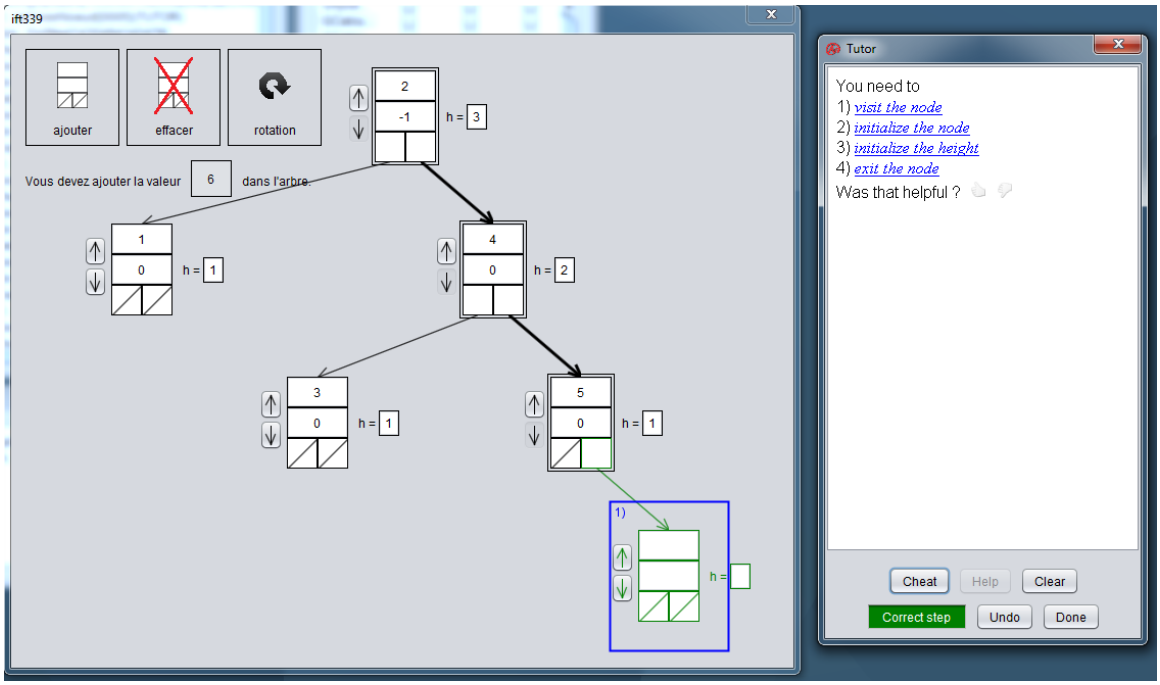

Figure 28 - Une intervention de type indice produite dans le MTT pour la manipulation d'ABR. On peut également remarquer la rétroaction minimale donnée sur la dernière action.

Après chaque intervention qui produit un message dans le tuteur, l'apprenant peut signifier au module pédagogue son appréciation de cette dernière, qu'elle soit positive ou négative (Figure 28). Puisque l'appréciation n'est pas un évènement en soi, elle ne déclenche pas

d'intervention, mais les appréciations passées peuvent être utilisées par la stratégie pédagogique.

Le module apprenant regroupe les évènements et les interventions qui forment une **étape** qui contient minimalement un « traçage mis à jour » et un « ok ». Il associe également à chaque étape une instance de but (la première au-dessus d'une instance de PC à partir de l'instance de PP ou de la transition) qui correspond à un des objectifs d'apprentissage dans le modèle de l'apprenant minimal.

Les évènements (pédagogiques) et les interventions se veulent exhaustifs, dans le sens où Astus n'offre pas de mécanismes d'extension qui permettrait aux auteurs d'en ajouter. La granularité des interventions a été choisie judicieusement de façon à minimiser leur nombre tout en répondant aux besoins de l'apprenant simulé. En effet, nous évitons ainsi que l'apprenant simulé doive interpréter le message (en ayant recours au TAL) pour tenir compte de l'intervention.

La stratégie pédagogique

Ce qu'on a désigné de façon abstraite comme la stratégie pédagogique se concrétise dans Astus sous la forme de plusieurs composantes : les situations, les métastratégies, les stratégies, les tactiques et les générateurs. Tous ces mécanismes peuvent être redéfinis par l'auteur pour comparer des stratégies pédagogiques différentes (ou encore pour tenir compte de la didactique du domaine).

Pour produire une intervention, une fois que l'évènement est créé, la KB produit une **situation** qui précise la nature de l'évènement. Astus fournit quelques situations de base, par exemple « action correcte » et « action non tracée » pour distinguer des évènements de type « traçage mis à jour ». La **métastratégie,** qui est associée à un groupe d'apprenants utilisant un MTT, fournit la stratégie qui est appliquée lors du suivi d'une instance d'une tâche; la métastratégie par défaut suppose qu'il n'y a qu'une seule stratégie disponible. Le rôle d'une **stratégie** est de fournir une tactique en réponse à une (ou plusieurs) situation. Dans la

stratégie pédagogique de base, il n'y a qu'une stratégie et celle-ci associe directement à chaque type de situation une tactique. C'est la **tactique**, à partir de la situation, qui produit l'intervention, et ce, à l'aide d'un ou plusieurs générateurs (si la tactique précédente n'est pas complétée, par défaut une tactique est complétée dès qu'elle a produit une intervention, elle produit l'intervention directement à partir de l'évènement).

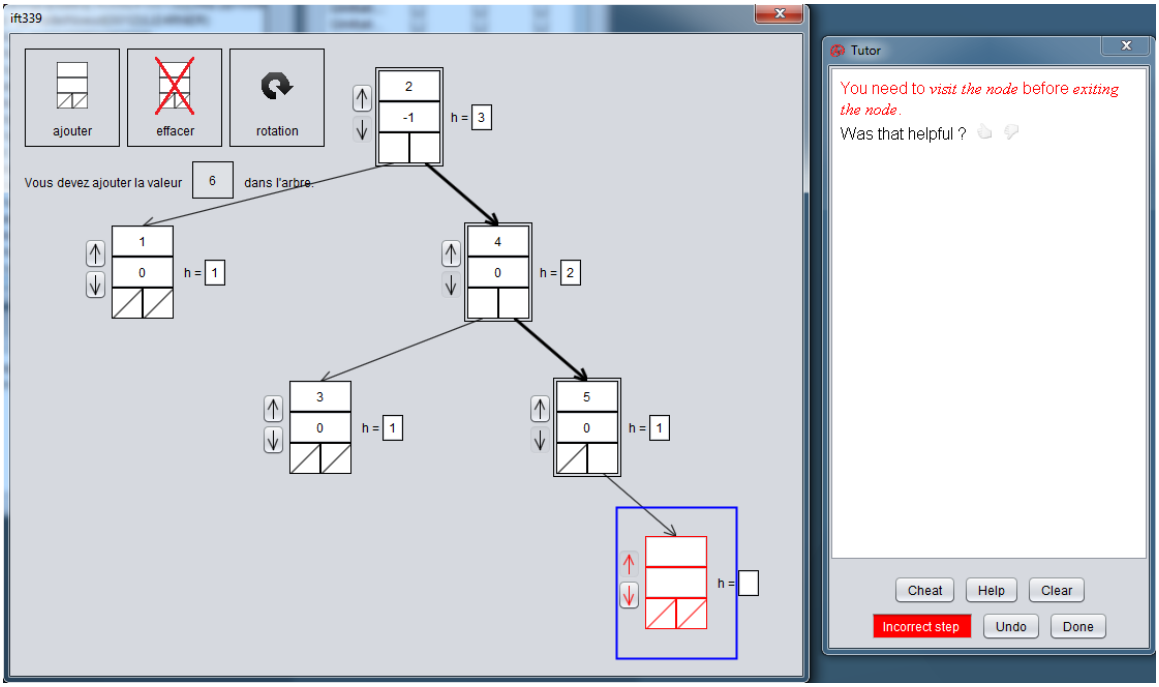

Figure 29 - Une intervention de type rétroaction (par rapport à une action non tracée).

Un **générateur** est un script « boite noire » qui reçoit en arguments des éléments de connaissance issus de la tactique et qui produit un *message pédagogique* en retour. Un message pédagogique contient un hypertexte et des effets visuels qui sont par la suite associés à une intervention. Ces scripts instancient des gabarits[194] associés aux types d'interventions et aux types des éléments du métamodèle à partir des énoncés en langage naturel associés aux éléments de connaissance (concepts, buts, etc.). Pour ce faire, les scripts manipulent l'élément de connaissance fourni par la tactique et accèdent à la KB.

[194] En pratique, ils font également un TAL minimal (p.ex. conjuguer le verbe de l'énoncé associé à un but).

Par exemple, lors d'une « action correcte », la stratégie fait appel à la tactique qui « traite les actions correctes » avec la situation en argument. Cette tactique produit soit l'intervention « ok » si l'épisode de l'étape associé à l'action n'est pas encore satisfait, soit une intervention de type « rétroaction » s'il l'est. Dans ce cas, le générateur « de rétroaction positive sur les étapes effectuées » produit le message pédagogique à l'aide de l'épisode satisfait (plus précisément, le générateur propre à la langue de l'apprenant). Par exemple, dans la manipulation d'ABR, pour l'anglais : « You have successfully *add*ed the *node* » (avec le nœud en surbrillance).

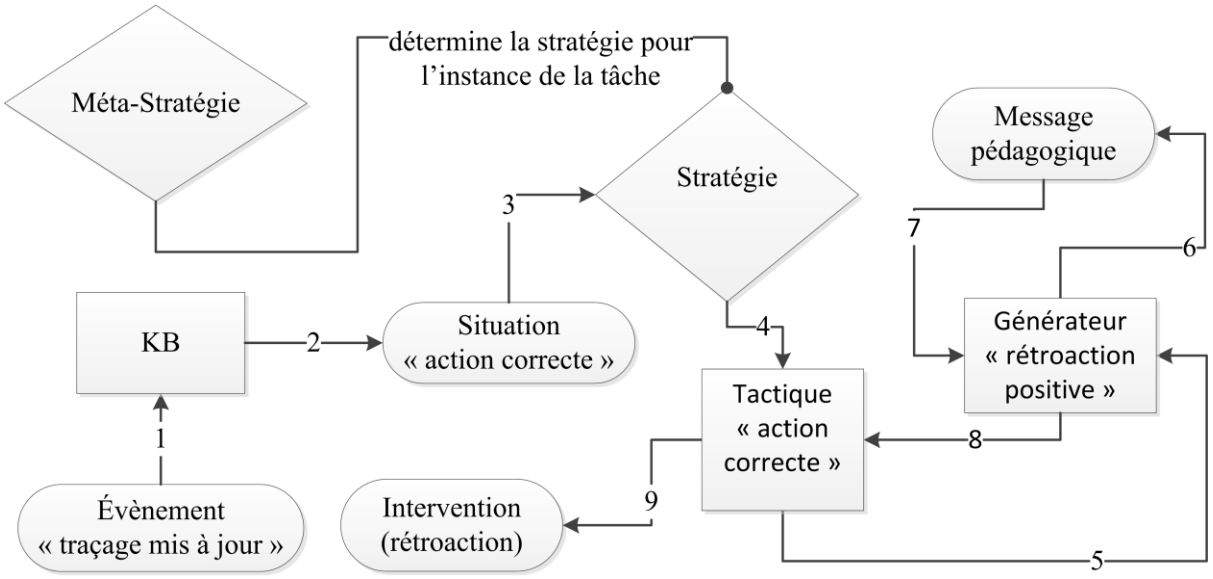

Figure 30 - Traitement d'un évènement jusqu'à la production d'une intervention par la stratégie pédagogique de base.

Remarques sur la stratégie pédagogique

L'utilisation de générateurs au sein de la stratégie pédagogique est la raison d'être du paradigme du tuteur. Bien que l'idée de base n'est pas originale à Astus, comme nous l'avons évoqué précédemment, on la retrouve dans TOTS et dans Steve qui découle de cette dernière, son application dans Astus se distingue, car elle est, entre autres, basée sur un modèle de la tâche qui s'inscrit dans le paradigme du tuteur (nous y revenons au chapitre 5). D'autres

systèmes, plus récents, génèrent également des interventions [105, 215, 237], mais celles-ci sont propres au domaine dont la tâche est tirée ou sont limitées par un métamodèle rudimentaire.

On peut remarquer qu'il y a une certaine redondance entre les situations et les tactiques, en effet la stratégie pourrait faire directement le lien entre les évènements et ces dernières. Or, nous supposons qu'il est préférable de maintenir un processus de classification[195] (situations) préalable au processus de décision (stratégies, tactiques). Des travaux futurs devront vérifier la pertinence des situations.

En observant les évènements pédagogiques et les interventions, on peut remarquer que les mécanismes classiques de suivi de l'apprenant des MT, c'est-à-dire la séquence d'indices sur la prochaine étape et la rétroaction sur l'erreur, ne sont pas prédéfinis. En effet, ils émergent du comportement du module pédagogue, qui est dicté par la stratégie pédagogique. Autrement dit, Astus n'impose pas de contraintes au niveau des interventions produites, au-delà de celles imposées par le module communication que nous avons déjà évoquées : l'environnement est bloqué lorsque le traçage n'est plus possible, et une rétroaction minimale (possiblement neutre) est toujours présente dès qu'une action est produite.

Génération d'indices sur la prochaine étape et de rétroactions

Pour produire un message, un indice sur la prochaine étape ou une rétroaction sur l'erreur, la tactique correspondante fait appel à un générateur associé au type de l'élément de connaissance qui est la cible de l'intervention. Ultimement, la cible est toujours un épisode, mais il peut y avoir une sous-cible particulière (une instance de procédure, un sous-épisode ou une expression).

Par exemple, dans la stratégie pédagogique de base, pour la rétroaction sur l'erreur, l'épisode ciblé est celui déterminé par le processus de diagnostic comme étant la source de l'erreur. De même, pour l'indice sur la prochaine étape, la cible est déterminée par une tactique, à la suite

[195] Dans l'esprit du « *Complex event processing* » (http://en.wikipedia.org/wiki/Complex_event_processing).

d'une « demande d'aide » générale (le bouton d'aide) ou suggérée par l'apprenant, à la suite d'une « demande d'aide » ciblée (un hyperlien dans le message de l'indice précédent).

Un générateur, pour produire le message pédagogique, peut faire appel à d'autres générateurs (et ce récursivement). Par exemple, le générateur d'une procédure conditionnelle fait appel au générateur pour la condition qui s'est avérée et le générateur pour l'épisode correspondant. Le générateur associé à la condition englobante fait appel aux générateurs des sous-expressions qui composent cette dernière et celui de l'épisode fait appel aux générateurs des expressions associées aux paramètres d'entrée du but instancié. Lorsque l'action produite par l'apprenant satisfait un épisode à l'aide d'une PC invalide, la tactique utilise un générateur qui produit un message à l'aide du gabarit fourni par l'auteur. On retrouve un exemple d'indice pour la prochaine étape pour la manipulation d'ABR dans la Figure 28.

Comme nous l'avons évoqué au chapitre 2, lorsque l'action produite est incorrecte et que le processus de diagnostic n'a pas réussi à reconnaitre l'erreur, la tactique fait appel à un générateur qui produit un message générique, comme dans les CT. Le processus de diagnostic, comme nous l'avons évoqué au chapitre 2, fait une fouille à partir des épisodes actifs et récemment satisfaits pour trouver un chemin qui mène à l'action incorrecte. Pour créer un chemin, le diagnostic perturbe le comportement des PC (p. ex. ignorer une contrainte d'ordre dans une séquence, ignorer une condition dans une conditionnelle ou dans une boucle conditionnelle) et celui des expressions (p. ex. utiliser une autre fonction dont les arguments sont du même type). La fouille fait un « retour arrière » lorsqu'une expression perturbée ou lorsque l'instanciation d'un but provoque une erreur interne (p. ex. obtenir l'objet qui instancie un concept et il n'est pas dans la KB). Des efforts ont été faits pour élaguer l'arbre de fouille [195], mais des travaux futurs devront formaliser le problème et explorer des pistes de solution pour rendre le diagnostic plus précis et plus efficace. On retrouve un exemple de rétroaction où le diagnostic est utilisé dans la Figure 29. Un autre exemple, dans le cas de la soustraction, est le message suivant : « Before you *calculate the difference* for *column* (1), you need to *borrow from column* (2) » où les étiquettes (1) et (2) sont affichées dans l'environnement.

Traitement des motifs procéduraux

L'idée derrière la notion de « motifs procéduraux » est que les tactiques et les générateurs ne peuvent pas produire des messages adéquats en traitant de façon isolée un élément de connaissance, c'est-à-dire qu'ils doivent tenir compte de la structure du graphe épisodique. Par exemple, reconnaitre la présence de récursivité ou des combinaisons particulières de procédures. Par exemple, en « soins critiques », on retrouve des séquences vides pour satisfaire des sous-buts d'une conditionnelle puisqu'il n'est pas pédagogiquement pertinent de créer une condition de satisfaction dans le but parent; on retrouve entre autres le cas du but `RetirerCorpsÉtranger` pour lequel il faudrait créer une condition qui vérifie que les voies respiratoires sont « `SansCorpsÉtranger` ». Un exemple qu'on retrouve dans pratiquement toutes les tâches est une séquence à un seul sous-but qui contient une expression produisant la valeur d'un paramètre du sous-but dont le but est satisfait par une PP (p. ex. dans l'addition de fractions, les PC des buts `CalculerNumÉquivalent` et `CalculerDénomCommun` ou dans la soustraction : `CalculerDifférence`). Un indice pour la séquence `CalculerDifférence` est par exemple : « *Write the difference in the column* » où l'hyperlien permet d'obtenir le sous-message : « The *difference* is the *subtraction* of the *top* and the *bottom* », puis finalement : « The *difference* is three ».

Puisque nos travaux n'ont pas été centrés sur le développement de la stratégie pédagogique, nous nous limitons à justifier le fait que le module pédagogue doit tenir compte de ces « motifs procéduraux » plutôt que de créer de nouvelles structures procédurales qui les représenteraient explicitement (ce qui n'implique pas que des travaux futurs n'ajouteront jamais de nouvelles structures procédurales). En effet, en plus des buts anonymes et séquences mentionnés précédemment, il existe plusieurs cas où les tactiques et les générateurs doivent nécessairement examiner plusieurs éléments de connaissance puisque le comportement n'est pas local (encapsulé par une PC). En voici trois : d'abord, le cas des interruptions; puis le cas des procédures choix sous une procédure de boucle conditionnelle (comme on le retrouve dans la résolution d'une équation) et finalement celui des procédures

pouvant mener à une impasse (les procédures qui ont une postcondition ou les procédures choix pour lequel le « retour arrière » est permis par l'auteur).

Effets visuels et démonstration dans l'environnement

Pour inclure un effet visuel, un générateur fait appel au module communication pour obtenir les vues qui sont associées à un élément de connaissance. Lorsque l'indice est donné au niveau d'une étape, le générateur peut aussi obtenir les signaux associés aux actions sous-jacentes (p. ex. pour mettre en surbrillance un bouton nécessaire pour déclencher l'action). Lorsque l'objet a été modifié ou a été retiré de la KB, le module communication peut ouvrir une fenêtre qui affiche l'état précédent de l'environnement et sur laquelle il peut reproduire les effets visuels (puisqu'une image de l'environnement et un squelette des vues a été conservé, comme nous l'avons mentionné précédemment).

Lorsque l'apprenant demande de l'aide alors que le dernier indice était déjà au niveau d'une étape, Astus peut faire une « démonstration ». Comme nous l'avons évoqué au chapitre 2, le module communication, à partir du patron et de l'arbre des vues, génère des interactions[196] pour satisfaire minimalement l'automate correspondant. Ces interactions sont produites de façon à produire l'action comme si l'apprenant l'avait fait par lui-même[197]. Lors d'une démonstration, les arguments de l'action sont injectés dans les gestionnaires de signaux, et ceux-ci, à l'aide de ces données et de l'état actuel du ou des composants[198] qu'il gère, manipulent l'environnement (déplacements du pointeur, clics, saisies, etc.) afin de déclencher l'interaction qui fait progresser l'automate.

Stratégie pour les « exemples détaillés »

Lorsque l'instance de la tâche est utilisée comme un « exemple détaillé », une stratégie (plus simple) est utilisée et des évènements sont interprétés différemment. Puisque l'apprenant ne

[196] Un algorithme naïf suffit puisque l'union (par rapport à la concaténation et l'étoile de Kleene) est peu utilisée dans les patrons, des travaux futurs formaliseront le problème.
[197] Des astuces sont utilisées p. ex. pour gérer l'état des composants qui utilisent des barres de défilement.
[198] Certains composants (p.ex. un bouton standard) n'ont pas d'état, dans ce cas, le gestionnaire est plus simple.

produit pas lui-même les actions dans l'environnement, « tricher » qui est alors désigné comme « suivant » dans le tuteur est la façon normale de progresser dans l'exemple. De même, « aide » produit une explication[199] à partir de l'épisode duquel découle la dernière étape produite plutôt que donner un indice sur la prochaine étape.

4.3.2 La sélection de tâches

Comme nous l'avons annoncé précédemment, la sélection de tâches, ou boucle externe, n'a pas été au centre de nos travaux. L'approche minimale proposée nous apparait toutefois suffisante pour montrer que le module apprenant et le module pédagogue peuvent remplir leurs rôles adéquatement à partir des données contenues dans la KB.

Objectifs d'apprentissage et modèle de l'apprenant minimal

Comme nous l'avons évoqué précédemment, le modèle de l'apprenant minimal contient, par défaut, un objectif d'apprentissage pour chacune des étapes. Pour chaque *objectif d'apprentissage*, on retrouve une progression en 3 étapes : *vu*, *fait* et *maitrisé*. L'objectif est considéré comme *vu* dès que l'étape est effectuée par l'apprenant ou le module pédagogue; il est considéré comme *fait* lorsque c'est l'apprenant qu'il l'a fait lui-même, peu importe si le module pédagogue l'a aidé ou non (sauf à l'aide de démonstrations) et il est considéré comme *maitrisé* lorsque l'apprenant peut faire l'étape sans être guidé. Dans le modèle minimal, le bénéfice du doute est donné immédiatement à l'apprenant, c'est-à-dire que dès que celui-ci fait une étape une première fois sans être guidé, l'objectif est considéré comme maitrisé. Le module pédagogue peut tenir compte de la progression des objectifs d'apprentissage (une fenêtre permet à l'apprenant de la visualiser, comme dans les CT), que ce soit au niveau des situations, de la stratégie ou des tactiques. L'auteur peut ajouter des objectifs d'apprentissage *de plus haut niveau*, c'est-à-dire choisir un but dans le graphe procédural pour en faire un objectif d'apprentissage. La progression de ces derniers correspond au minimum de la

[199] De nature procédurale et non par rapport aux principes du domaine où à des connaissances téléologiques.

progression des objectifs d'apprentissage de niveau inférieur (des travaux futurs devront explorer d'autres approches de calcul de la progression des objectifs de plus haut niveau).

Sélection de la tâche par le module pédagogue

Comme nous l'avons évoqué au chapitre 2, le module pédagogue peut lui-même choisir une instance d'une tâche. Puisque nos travaux se sont concentrés sur la boucle interne et non sur la boucle externe, nous n'avons pas tenté d'optimiser le mécanisme de choix. Nous nous sommes donc limités à un algorithme minimal qui s'inspire du « *Mastery Learning* » comme les CT. Le module pédagogue choisit donc la première instance d'une tâche (selon l'ordre de définition) qui comporte au moins un objectif qui n'est pas maitrisé. Tant que c'est possible, il préfère les instances de tâches qui n'ont pas déjà été effectuées. Pour déterminer si une instance d'une tâche comporte un tel objectif, il doit avoir recours aux métadonnées obtenues par l'exploration automatique.

C'est dans de l'intervention de type « fin » que le module pédagogue peut produire une rétroaction par rapport à la tâche effectuée, par exemple en indiquant quels objectifs d'apprentissage ont progressé.

Travaux futurs pour le suivi de l'apprenant et la sélection de tâches

Comme pour les sections précédentes, en plus des travaux futurs déjà mentionnés, en particulier le développement de la stratégie pédagogique de base, du processus de sélection de tâches et du modèle de l'apprenant, nous soulignons les travaux futurs suivants :

- Rendre compatibles les données qui sont issues de chaque itération de la boucle externe (les évènements pédagogiques et les interventions) avec Datashop, le principal format utilisé dans la communauté (il est possible que les types d'évènements pédagogiques et d'interventions doivent être modifiés pour permettre la comptabilité).

- Harmoniser le comportement de la boucle externe (choisir l'instance d'une tâche, décider si l'apprenant effectue cette dernière ou si elle est utilisée comme « exemple détaillé », etc.) avec Tin Can[200] (anciennement SCORM), le principal standard utilisé dans la communauté.

- Collaborer avec des chercheurs en GAT pour améliorer les générateurs, en particulier afin d'améliorer la qualité du langage naturel et faciliter l'ajout d'autres langues que l'anglais, en particulier le français.

- Enrichir les interventions pour qu'elles prennent en compte les aspects émotifs et métacognitifs (ou tout autre type d'aspects présents dans le modèle de l'apprenant).

- Étudier la possibilité d'inclure deux nouveaux types d'interventions plus sophistiquées que celles déjà supportées, l'« auto-explication » (« *self-explanation* ») et les analogies [61]. L'« auto-explication » consiste à demander à l'apprenant de réagir face à l'exemple qui lui est présenté, en particulier de justifier les actions produites. Faire une analogie consiste à faire référence, dans une intervention, à une action passée dans une autre instance de la tâche, voire dans une autre tâche.

4.4 Le processus de création

Le processus de création d'Astus est à la base similaire à celui des CT. Puisque le modèle de la tâche est plus sophistiqué et que les interventions sont générées plutôt que prémâchées, Astus offre des DSL et des outils (incluant les modes d'exécutions) pour obtenir des efforts de création d'un ordre de grandeur comparable à ceux des CT. Les principales étapes pour la création d'un MTT avec Astus sont :

1. Créer les éléments sémantiques qui sont nécessaires pour créer une version minimale de l'environnement, créer les vues correspondantes, puis faire des tests avec le mode « test de l'environnement ».

[200] http://tincanapi.com/

2. Créer les buts, les PP et les transitions qui déterminent la granularité des étapes dans l'environnement, créer les patrons correspondants, puis faire des tests avec le mode « test de l'environnement ».

3. Créer les éléments procéduraux nécessaires, en ajoutant au besoin des éléments sémantiques (et les vues correspondantes), puis faire des tests avec les modes « exécution descendante » et « apprenant ».

4. Associer les énoncés en langage naturel aux termes et aux buts, puis faire des tests avec les modes « exécution descendante » et « apprenant ».

5. Créer des instances supplémentaires de la tâche et rendre le modèle de tâche suffisamment souple (p. ex. en utilisant des « objets motifs » et en incluant des PC supplémentaires comme des PC qui représentent des incompréhensions et des optimisations) puis faire des tests avec les modes « exploration automatique » et « apprenant simulé ».

6. Ajuster le comportement du module pédagogue, en ajustant au besoin les situations, la métastratégie, les stratégies, les tactiques ou les générateurs, puis faire des tests avec les modes « exploration automatique » et « apprenant simulé ».

Précisions sur les outils

Comme nous l'avons évoqué précédemment, Astus comme CTAT, offre des outils pour observer les différents modèles. Une fois que les modèles pour une tâche sont chargés et compilés, on peut visualiser les contextes, les termes, les objets du domaine, le graphe procédural, les patrons et les instances de tâches passées (actions, épisodes, interventions). À l'exécution, on peut d'une part visualiser le reste des objets, les actions produites, le graphe épisodique, les évènements, les interventions, les automates associés aux patrons, l'état du traçage et les vues (un mode permet de superposer les rectangles englobant des vues sur l'environnement). D'autre part, on peut faire des manipulations comme exécuter des actions directement (sans produire les interactions dans l'environnement) et ajuster le modèle de

l'apprenant. Parce que les modèles d'Astus sont plus sophistiqués que ceux des CT, Astus, applique dès la compilation, puis lors de l'exécution, des vérifications pour détecter des anomalies.

Les modes « exploration automatique » et « apprenant simulé », même s'ils sont minimalement supportés, sont des composantes essentielles du processus de création, puisqu'ils facilitent significativement la vérification des modèles.

Bien que l'exploration automatique soit conceptuellement plus simple que la simulation d'un apprenant, c'est un problème fondamentalement difficile lorsqu'on rend plus fine la granularité avec laquelle on décide si deux chemins sont équivalents. En effet, si on se limite à parcourir les chemins tels qu'ils sont définis par le graphe procédural, l'exploration est proportionnelle à la taille et à la complexité du modèle (p. ex. une procédure choix amène une plus grande complexité qu'une séquence). Par contre, si l'on considère toutes les variantes que l'on rencontre dans le graphe épisodique (p. ex. les différentes valeurs que peuvent prendre les variables), l'explosion combinatoire est évidente. L'implémentation minimale effectue un nombre maximal d'instances de tâches en priorisant les chemins qu'elle n'a jamais explorés. Une instance d'une tâche, qui est effectuée un nombre maximal de fois sans ajouter de nouveaux chemins, n'est plus effectuée. Les maximums et la priorité accordée aux chemins non explorés peuvent être ajustés par l'auteur (ce qu'est pas parce qu'un chemin a été exploré, que tous les sous-chemins qu'il contient l'ont été). L'exploration automatique n'interagit pas avec le tuteur, sauf pour signaler la « fin » ou pour effectuer les annulations nécessaires suite à l'exécution d'une PC invalide ou à une impasse.

L'avantage de l'apprenant simulé par rapport à l'exploration automatique est double, d'une part il valide le modèle de façon plus exhaustive parce qu'il utilise toutes les fonctionnalités du tuteur et qu'il fait des erreurs. D'autre part, parce qu'il produit des métadonnées supplémentaires, en particulier les interventions. Comme pour l'exploration automatique, l'auteur peut ajuster les paramètres qui contrôlent l'apprenant simulé : le nombre d'instances de tâches effectuées, l'état initial des connaissances, ceux qui régissent le comportement en fonction de l'état des connaissances, comme le taux d'erreurs d'inattention (« *slip* ») et

d'actions correctes fortuites (« *guess* »), et finalement ceux qui régissent l'apprentissage en fonction de la dernière étape (c.-à-d. les changements à l'état des connaissances). Comme nous l'avons évoqué au chapitre 2, lorsque l'apprenant simulé tente d'effectuer une action incorrecte non tracée, un processus de fouille est amorcé à partir des épisodes actifs. Des épisodes sont perturbés (de la même façon que pour le diagnostic) pour trouver un chemin vers une action. Si l'action obtenue ne peut être produite dans l'environnement, c'est-à-dire que la démonstration provoque une erreur interne, la fouille fait un « retour arrière ». Après un certain nombre de « retour arrière », l'apprenant simulé abandonne la fouille et interagit différemment avec l'environnement ou le tuteur.

Remarques sur la création du modèle de la tâche

On peut remarquer que dans le graphe procédural, les préconditions et les postconditions jouent un rôle bien précis et qu'elles ne constituent pas une façon pour l'auteur de vérifier son modèle. Par contre, l'auteur peut tirer profit du fait que la modélisation à l'aide d'un DSL « interne » donne accès au langage hôte pour ajouter des vérifications au sein du modèle.

On peut aussi remarquer que même si le paradigme du tuteur guide l'auteur et même si Astus offre des outils efficaces, la création d'un modèle reste un processus difficile qui demande des efforts importants. Par exemple, le choix de la bonne granularité pour les éléments du modèle, en particulier les PP et les termes opérationnels, est crucial. Or, puisque cette thèse n'est ni un manuel de référence ni un guide de l'utilisateur pour Astus, nous ne fournissons pas une liste de bonnes et de mauvaises pratiques de création. Toutefois, ce sont les interventions, en particulier les messages issus des générateurs, qui en fin de compte guident l'auteur dans son processus de création. Dans les CT, lorsque l'auteur a obtenu le comportement de suivi (traçage) voulu et qu'il a créé ses gabarits de message, il connait précisément le comportement du MTT qu'il vient de créer. Avec Astus, le comportement du module pédagogue est émergent, par conséquent, l'auteur, à l'aide des outils offerts par la plateforme, doit s'assurer que ses intentions sont interprétées correctement par le module pédagogue.

Travaux futurs concernant le processus de création

Parmi les travaux futurs touchant le processus de création, on retrouve :

- utiliser des outils de l'IA pour améliorer l'exploration automatique et l'apprenant simulé;
- utiliser des outils de l'IA pour créer des outils de génération d'instances de la tâche à partir des métadonnées obtenues par l'exploration automatique et l'apprenant simulé;
- ajouter des mécanismes de vérification au niveau des patrons, par exemple pour vérifier qu'ils sont « sans préfixe » et qu'ils sont tous distincts;
- éliminer la séparation entre les règles et les autres éléments du modèle de la tâche, autrement dit créer un DSL qui les intègrent et qui permet de générer les règles équivalentes pour le système de production utilisé.

Chapitre 5

Discussion

Ce chapitre nous permet d'atteindre notre troisième objectif, c'est-à-dire positionner les MTT créés avec Astus, d'abord par rapport aux autres MTT, puis par rapport aux autres STI.

5.1 Astus par rapport aux autres MTT

Astus se distingue des autres MTT en s'inscrivant dans le paradigme du tuteur, le paradigme qui exige le plus explicitement que la tâche soit bien définie et que l'évaluation soit serrée. Les exemples de MTT créés avec Astus et le suivi, qui produit toujours une rétroaction minimale et qui bloque l'environnement lorsque l'action produite est incorrecte ou qu'elle mène à une impasse, en témoignent. Les MTT mentionnés précédemment, qu'ils soient le reflet du paradigme de l'apprenant ou de l'expert, peuvent apparaitre difficiles à convertir au paradigme du tuteur. Dans certains cas, comme celui d'Andes, c'est l'exigence d'une évaluation moins serrée qui est incompatible, mais dans d'autres, c'est plutôt que la tâche n'est pas suffisamment bien définie. En effet, en plus de l'exemple classique des « problèmes écrits » en algèbre où c'est le TAL qui pose problème, des tâches comme la programmation (comme un CT pour la programmation en Lisp [22]) ou les preuves (comme un CT pour les preuves par construction en géométrie [122]) ne correspondent pas à des tâches bien définies. D'ailleurs, pour les rendre suffisamment bien définies pour le paradigme de l'apprenant, les auteurs de ces CT ont eu recours à des environnements qui réifient une méthode (méthode descendante pour le Lisp, méthode basée sur les « *Diagram Configuration*

Model » pour la géométrie[201]). En effet, la structure de ces environnements (par rapport à des environnements moins structurés comme celui d'Andes/ST) fait en sorte de limiter les cas où une action de l'apprenant est traitée comme une action incorrecte même si elle pourrait être considérée comme correcte si le modèle de la tâche était complet (c'est-à-dire s'il contenait toutes les connaissances du domaine et suffisamment de connaissances générales pour interpréter ces dernières).

Pyrenees, comme nous l'avons indiqué au chapitre 2, est un MTT qui s'inscrit minimalement dans le paradigme du tuteur. En effet, la méthode qu'il demande à l'apprenant d'appliquer ne met pas l'accent sur des euristiques propres au domaine (physique, probabilités), mais utilise plutôt des connaissances supposées maitrisées (algèbre). De plus, il s'inscrit implicitement dans le paradigme du tuteur, puisque le modèle n'est pas interprétable et que des interventions prémâchées sont utilisées comme dans les autres systèmes.

On peut remarquer qu'à l'aide des mécanismes d'extensions de la stratégie pédagogique, un auteur peut recréer le comportement d'un MTT comme Ms. Lindquist, c'est-à-dire que les tactiques et les générateurs peuvent représenter des connaissances didactiques du domaine, en particulier sous la forme de dialogue. La plateforme Astus est neutre à cet égard, mais ne le supporte pas explicitement. Des travaux futurs pourraient intégrer une notion de sous-étape qui représente les questions du module pédagogue et les réponses de l'apprenant, de façon à ce que le module apprenant et le module pédagogue puissent ensuite les manipuler.

Le paradigme du tuteur reprend le raisonnement qui a mené les concepteurs des CT à créer des environnements structurés et le pousse jusqu'au bout. Le paradigme du tuteur tient pour acquis que le modèle est incomplet, c'est-à-dire que le module pédagogue considère que le graphe procédural définit une ou des méthodes qui doivent être utilisées pour effectuer la tâche. Cette affirmation peut paraitre évidente à première vue, mais dans les MTT qui ne s'inscrivent pas dans le paradigme du tuteur et qui ne génèrent pas d'interventions, on peut tirer profit d'un flou entourant le fait que le modèle est complet ou non. Par exemple, Andes

[201] Les limites de cette approche sont abordées dans [144].

tolère les actions qui ne sont pas nécessaires pour effectuer la tâche, mais qui sont valides par rapport au domaine (comme un micromonde). Un autre exemple est celui des CT qui traitent toutes les actions non tracées de la même façon, c'est-à-dire avec un message d'erreur générique.

Le principal avantage du paradigme du tuteur est que le module pédagogue, en toute transparence, peut dire à l'apprenant que l'action qu'il a effectuée est incorrecte en fonction des méthodes incluses dans le modèle par les auteurs. Or, cette propriété peut être considérée comme un inconvénient, puisqu'elle met en évidence le fait que le modèle est incomplet. Au contraire, nous croyons que le flou entretenu sur les capacités réelles du module expert dans les MTT fait probablement partie des raisons qui expliquent le rejet des MTT par les chercheurs en éducation. En effet, même si l'approche des MTT est clairement plus pragmatique que les approches utilisées dans les premiers STI, le paradigme de l'expert et celui de l'apprenant ne se dégagent pas complètement du fardeau qui consiste à obtenir un module expert avec un comportement d'IA forte.

Avec Astus, nous faisons l'hypothèse qu'une plus grande transparence de la part du module pédagogue, même si elle se traduit par une évaluation plus serrée, a un impact positif sur l'apprentissage. Plus particulièrement, qu'elle évite de renforcer l'acquisition de connaissances procédurales plus superficielles. En effet, puisque non seulement l'environnement, mais aussi le tuteur, réifient les méthodes utilisées, l'apprenant est davantage en mesure de les dissocier d'une stratégie de résolution de problèmes (c.-à-d. des éléments procéduraux indépendants du domaine). Par exemple, un MTT pour la soustraction en colonnes qui s'inscrit dans le paradigme du tuteur peut supporter explicitement deux méthodes différentes (dans le même environnement) et faire en sorte que l'apprenant soit conscient qu'il doit utiliser ces méthodes plutôt que de le laisser croire que le module pédagogue peut reconnaitre une méthode (valide) de son cru.

Comme nous l'avons énoncé précédemment, la raison d'être du paradigme du tuteur est de permettre la génération d'interventions, en particulier l'indice sur la prochaine étape et la rétroaction sur une action non tracée. Avec les MTT que nous avons modélisés, nous sommes

en mesure de faire un certain bilan sur les difficultés que pose la génération. Même pour un modèle de tâche simple, nous avons déjà souligné le fait que l'auteur doit utiliser adéquatement les types d'éléments de connaissance qui sont à sa disposition et que la stratégie pédagogique doit faire en sorte de respecter les intentions de l'auteur. Dans un tel modèle, même si la transformation en langage naturel peut parfois être un problème (l'asymétrie entre le connecteur « OU » et le « or » en anglais[202], le traitement de la négation, etc.), elle semble généralement pouvoir s'implémenter à l'aide de gabarits et d'un traitement du langage naturel minimal, du moins pour l'anglais. Dans le cas de la rétroaction sur une action non tracée, c'est le problème sous-jacent du diagnostic qui est avant tout difficile. Que ce soit pour un indice ou une rétroaction, les gabarits pour les expressions ont jusqu'à présent été plus difficiles à implémenter que les gabarits pour les indices. En particulier, une difficulté est de pouvoir générer non seulement le message correspondant à la valeur de l'expression, mais aussi celui correspondant à son type[203]. En effet, on doit pouvoir faire référence à l'expression même si elle n'a pas encore été exécutée, par exemple si elle décrit le paramètre d'entrée d'un épisode inactif. Les modèles plus complexes, avec des procédures choix, des impasses ou encore avec des interruptions ne sont pas fondamentalement différents. En effet, bien que ces « motifs procéduraux » soient plus complexes, ils sont construits à l'aide des éléments de base que l'on retrouve dans les modèles plus simples (conditions, relation d'ordre, collections, sous-buts, etc.). Même si les gabarits nécessaires pour les supporter risquent d'être plus difficiles à implémenter, nous estimons qu'il est possible d'obtenir des messages acceptables, du moins pour l'anglais, sans devoir recourir à la GAT.

L'application la plus éloquente du paradigme du tuteur dans Astus est la procédure choix. D'une part, parce qu'elle encourage l'auteur à exprimer les critères de choix de façon à pouvoir les communiquer à l'apprenant. D'autre part, parce qu'en rendant explicite la notion de choix, elle encourage une communication transparente entre le MTT et l'apprenant. En

[202] De même que pour le « ou » en français.
[203] Pas seulement la valeur ou le type du résultat final, mais aussi les valeurs et les types des résultats intermédiaires.

effet, le MTT n'a pas accès à des informations privilégiées[204] pour choisir l'un ou l'autre des épisodes et par conséquent il peut guider l'apprenant vers une impasse en appliquant les euristiques définies par l'auteur. Par exemple, un MTT pour le calcul intégral pourrait proposer à l'apprenant d'appliquer l'intégration par parties avec un certain u et un certain dv puis se rendre compte en calculant du et v qu'il vaudrait mieux faire un autre choix[205]. Ce comportement, qui peut être à première vue considéré comme un inconvénient, est pour nous un avantage du paradigme du tuteur, puisqu'il met en évidence les connaissances du domaine, plutôt qu'une stratégie générale de résolution de problèmes[206]. Des études ont montré qu'en l'absence d'interventions qui mettent en évidence la présence d'un choix, les apprenants se limitent à utiliser la méthode qui leur est la plus familière [117, 268]; or celle-ci ne leur permet pas nécessairement d'exhiber la profondeur de leurs connaissances [242]. De plus, des études indiquent qu'un apprenant qui apprend différentes méthodes pour accomplir une même tâche acquiert par le fait même des connaissances plus profondes [221]. Par conséquent, la transparence qui découle du paradigme du tuteur, avec laquelle le module pédagogue propose à l'apprenant de faire des choix différents, permet possiblement aux apprenants d'acquérir des connaissances procédurales plus profondes.

Une autre application éloquente du paradigme du tuteur est lorsque l'apprenant fait face à une impasse. En effet, dans ce cas, le module pédagogue peut offrir à l'apprenant, à l'aide d'une rétroaction (dont le message pédagogique contient un hyperlien), d'annuler directement les dernières étapes afin de pouvoir explorer un autre chemin du graphe épisodique. Autrement dit, de façon à réifier le « retour arrière » [180], ce qui favorise également une communication plus transparente entre le module pédagogue et l'apprenant. Par exemple, annuler les actions qui découlent du choix du u et du dv et celles qui découlent du calcul du du et du v dans un MTT pour le calcul intégral.

[204] À l'exception des traces d'exécution passées.
[205] $\int u\, dv = uv - \int v\, du$
[206] On peut considérer que le paradigme du tuteur induit une stratégie de résolution de problème qui correspond à la fouille exhaustive provoquée par les impasses et contrôlée par les euristiques de la procédure choix.

On peut critiquer le fait qu'à l'intérieur du paradigme du tuteur, il est difficile d'assouplir l'évaluation de façon à permettre à l'apprenant de sauter des étapes comme dans Andes/ST et jusqu'à un certain point, dans les CT. En effet, même en offrant des mécanismes pour assouplir l'évaluation (comme les « objets motifs », les PC qui représentent des optimisations et les éventuels sous-buts répétables ou optionnels), ceux-ci peuvent exiger des efforts de création importants et ils ne permettent pas d'obtenir une évaluation aussi souple que celle des MTT qui agissent partiellement comme des STI basés sur des modèles évaluatifs (Andes/ST). Parce que les règles de production des CT[207] sont plus flexibles que le graphe procédural d'Astus, mais surtout parce qu'elles ne sont pas exploitées pour générer des interventions, elles permettent d'obtenir plus facilement une évaluation qui est en principe aussi souple que celle qu'on retrouve dans Andes/ST (ce qui correspond, par exemple, à utiliser un système de calcul formel comme celui d'Andes dans les règles).

Comme nous l'avons évoqué en introduction, certains STI, comme Steve, ont tenté d'assouplir l'évaluation des MTT à l'aide d'une approche basée sur la planification. Même si leur besoin était de supporter des environnements qui ont une dimension physique, on peut appliquer le même raisonnement pour des environnements abstraits (comme ceux des CT et d'Andes/ST), mais nous avons écarté cette piste de solution (nous le justifions dans la section 5.2). Par contre, nous avons envisagé deux pistes de solutions qui peuvent atténuer ce problème. La première est au niveau de la boucle interne, il s'agit de tirer profit du modèle de l'apprenant dans la stratégie pédagogique afin d'effectuer automatiquement les actions qui découlent d'objectifs d'apprentissage qui sont maitrisés, tel que le font certains CT. La deuxième est au niveau de la boucle externe, elle consiste à tirer profit d'une sélection de tâches optimisée[208] de façon à éviter de soumettre à l'apprenant des tâches pour lesquelles les connaissances qu'il a déjà acquises font en sorte qu'il est tenté de sauter des étapes.

[207] En particulier ceux créés avec le TDK (par rapport à CTAT).
[208] Grâce à un apprenant simulé amélioré ou encore à un modèle de l'apprenant plus sophistiqué construit à l'aide de l'apprentissage automatique.

Finalement, la question la plus importante qui demeure en suspens par rapport au paradigme du tuteur, est de savoir à quel point les structures de connaissances que nous proposons reflètent l'entièreté de son potentiel. Par exemple, nous avons privilégié une approche modulaire qui consiste à combiner les structures existantes. Par contre, face au défi d'implémentation des générateurs pour les « motifs procéduraux » complexes, l'idée d'introduire des structures procédurales qui représentent explicitement des méthodes de résolution de problèmes va perdurer[209]. En fait, nous avons déjà trouvé un type de tâches pour lequel les structures existantes ne sont pas suffisantes pour supporter le paradigme du tuteur. Il s'agit des tâches qui impliquent de réifier une forme de planification[210] (comme les preuves par construction en géométrie [122]) ou de façon plus générale, qui implique de réifier des croyances[211] (comme des hypothèses en « génie génétique »). Même si nous avons proposé des travaux futurs, comme l'ajout de disjonctions et de négation dans les éléments sémantiques et l'ajout de coroutines dans les éléments procéduraux qui constituent un premier pas dans cette direction, supporter de telles tâches est un projet de recherche en soi.

5.2 Astus par rapport aux autres familles de STI

Dans cette section, nous analysons brièvement d'autres types de STI qui sont comparables aux MTT, c'est-à-dire les STI pour lesquelles la granularité des étapes fait en sorte qu'il y a généralement plusieurs itérations de la boucle interne pour chaque itération de la boucle externe. Autrement dit, nous excluons les STI qui ne produisent pas d'indices pour la prochaine étape et qui se limite à produire une rétroaction globale sur l'instance de la tâche effectuée. Généralement, ces STI sont des systèmes dédiés basés sur un modèle évaluatif, comme Proust et Sherlock, ou des systèmes supportant des environnements qui ont non

[209] Cette idée se retrouve également dans les SABC en dehors du contexte des STI. Par exemple, le formalisme UPML[85] a été proposé pour représenter des méthodes de résolution de problèmes qui peuvent être appliquées dans différents domaines. Par contre, peu d'exemples sont donnés et ils ne nous ont pas convaincus.
[210] La procédure choix permet une forme minimale de planification puisque les euristiques peuvent contenir des expressions qui représentent une planification mentale (p. ex. dans le cas du calcul intégral, on pourrait avoir une opération qui représente le calcul du du, ce qui suppose que les dérivées sont considérées maitrisées)
[211] http://en.wikipedia.org/wiki/Belief_revision

seulement une dimension physique, mais qui sont aussi temps réel [128] (nous précisons la notion d'environnement temps réel dans la section 5.2.1).

5.2.1 Les STI pour les environnements dynamiques

En ajoutant la couche de simulation, les évènements et les interruptions au modèle de la tâche, nous avons l'ambition de supporter des environnements similaires à ceux que Steve supporte [212], mais dans le cadre du paradigme du tuteur. Ces environnements sont dynamiques, mais ils ne sont pas temps réel, autrement dit, ils agissent comme « un système d'évènement discret » [213]. Le MTT de « soins critiques » en est un exemple. Un environnement temps réel, est un type d'environnement pour lequel le temps écoulé entre les actions ou encore pendant les actions, lorsqu'elles ont une durée, est important. Nous excluons de supporter ce type d'environnement avec Astus, d'une part parce qu'il privilégie l'immersion par rapport aux indices et aux rétroactions immédiates et d'autre part, parce que la concurrence inhérente à ce type d'environnements est incompatible avec un mécanisme d'annulation et de façon plus générale, est difficile à circonscrire à l'intérieur du cadre du paradigme du tuteur.

La question qui devra être étudiée lors de travaux futurs est de savoir comment les mécanismes d'Astus se comparent, en pratique, à une approche basée sur la planification comme le propose Steve[214]. En principe, un planificateur dont l'exécution est entrelacée avec les actions produites dans l'environnement et qui tire profit de pré/postconditions classiques est bien plus expressif, mais il est possible que cette expressivité ne soit pas nécessaire pour les situations où l'approche d'Astus s'applique. De plus, si les mécanismes que nous proposons ne sont pas suffisants, des travaux futurs pourront ajouter des mécanismes supplémentaires pour supporter la concurrence inhérente aux environnements dynamiques (p.

[212] Nous supposons que l'approche de Steve se généralise au-delà des exemples en maintenance industrielle.
[213] https://en.wikipedia.org/wiki/Discrete_event_dynamic_system
[214] Steve n'a jamais dépassé le stade de prototype.

ex. introduire des buts dont la condition de satisfaction doit être maintenue tant que la tâche n'a pas été effectuée complètement[215]).

Le MTT de « soins critiques » est idéal pour explorer les extensions du modèle de la tâche visant à supporter des environnements dynamiques. En particulier, il permet d'envisager de s'inspirer des travaux sur les systèmes multiagents. En ce sens, une première extension serait de supporter l'exécution d'une instance de la tâche par deux apprenants à la fois, c'est-à-dire que le même graphe épisodique est mis à jour à partir des actions produites par les deux apprenants qui ont chacun leur propre accès à l'UI. Une telle extension exigerait de représenter les ressources requises pour produire une action et de représenter les actions qui ont une durée à l'aide d'évènements. Une deuxième extension serait de réifier les autres agents impliqués dans l'exécution de la tâche (médecin, inhalothérapeute, auxiliaire, etc.) qui sont implicitement présents dans les actions et les évènements. Dans ce cas, il faudrait modéliser les actes de langage qui constituent la communication entre les agents[216] [203].

5.2.2 Les ETT

Les ETT qui sont créés directement à partir des traces d'actions produites par le corps enseignant sont limités. En effet, CTAT construit simplement un graphe (« *behavior graph* ») dont les nœuds correspondent aux états de l'environnement qui résulte de la production des actions [5] (Figure 32). Deux approches fondamentalement différentes ont été poursuivies pour les rapprocher des MTT, la première est celle de SimStudent, c'est-à-dire utiliser des outils d'apprentissage automatique (nous y revenons à la section 5.2.4). La deuxième est d'offrir des outils auteurs (Figure 31) qui permettent au corps enseignant de créer un modèle évaluatif plus souple et de générer des instances de la tâche [5]. Plus particulièrement, ces outils auteurs permettent d'ajuster la structure du graphe et d'ajouter des métadonnées. Dans l'exemple de la tâche « addition de fractions », on peut par exemple enlever la contrainte d'ordre sur la réécriture des fractions (c.-à-d. quelle fraction est réécrite en premier) et ajouter

[215] « *maintenance goals* » dans les agents BDI.
[216] Avec une telle extension, il serait également possible de simuler la deuxième infirmière.

une formule qui reconnait tous les dénominateurs communs. Même si ces outils auteurs permettent de retrouver une part importante de la flexibilité des MTT, lorsqu'ils sont utilisés à leur pleine capacité, ils requièrent des habiletés équivalentes à celles requises pour créer un modèle à l'aide d'un langage de programmation ou d'un DSL. Autrement dit, comme l'admettent leurs concepteurs [5], ils ne peuvent pas être pleinement utilisés par le corps enseignant.

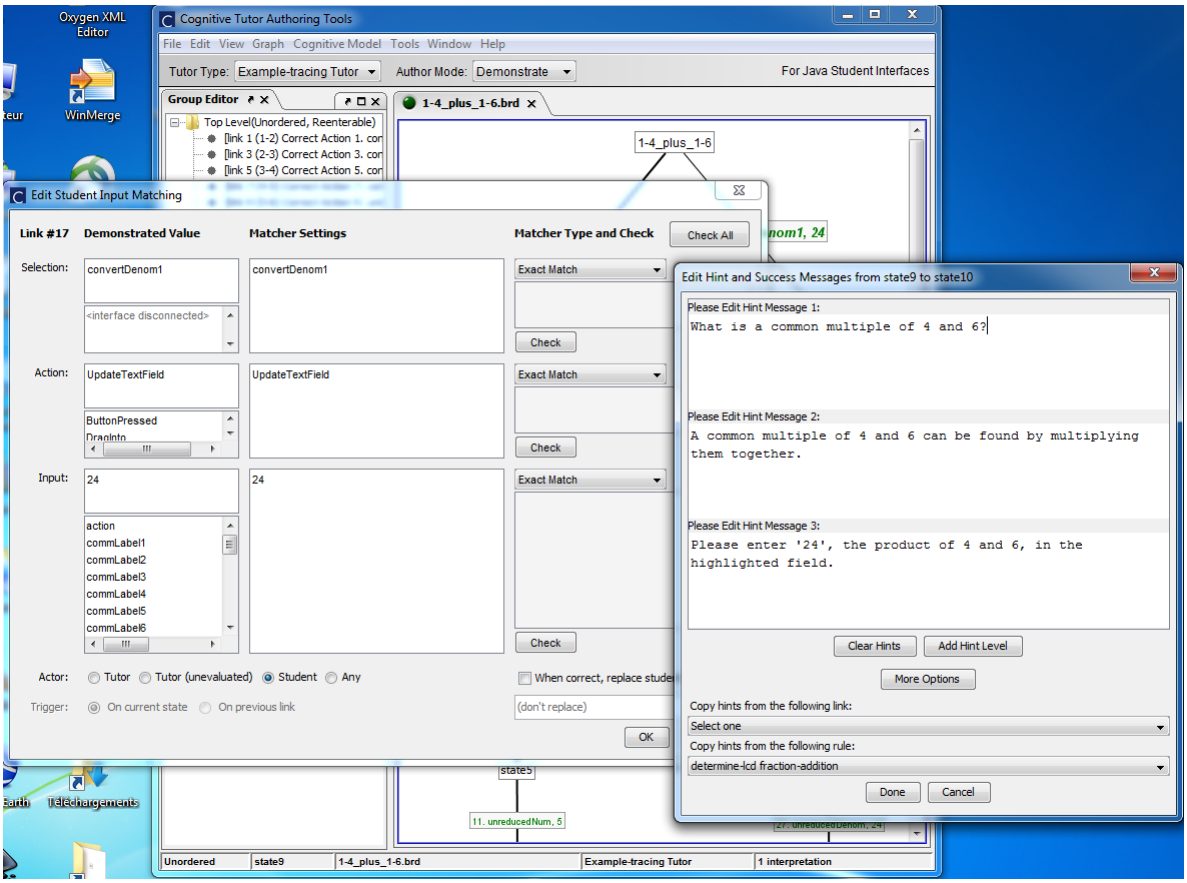

Figure 31 - À gauche, les outils auteurs qui permettent de généraliser le modèle. À droite la fenêtre qui permet de créer la séquence de messages propre à une instance de la tâche.

Comme pour la programmation visuelle, nous estimons que des auteurs ayant des habiletés de programmation sont plus efficaces lorsqu'ils utilisent un IDE qui supporte la création d'un code source qu'en utilisant des outils auteurs. Par conséquent, nous avons exclu d'introduire ce type d'outils dans Astus.

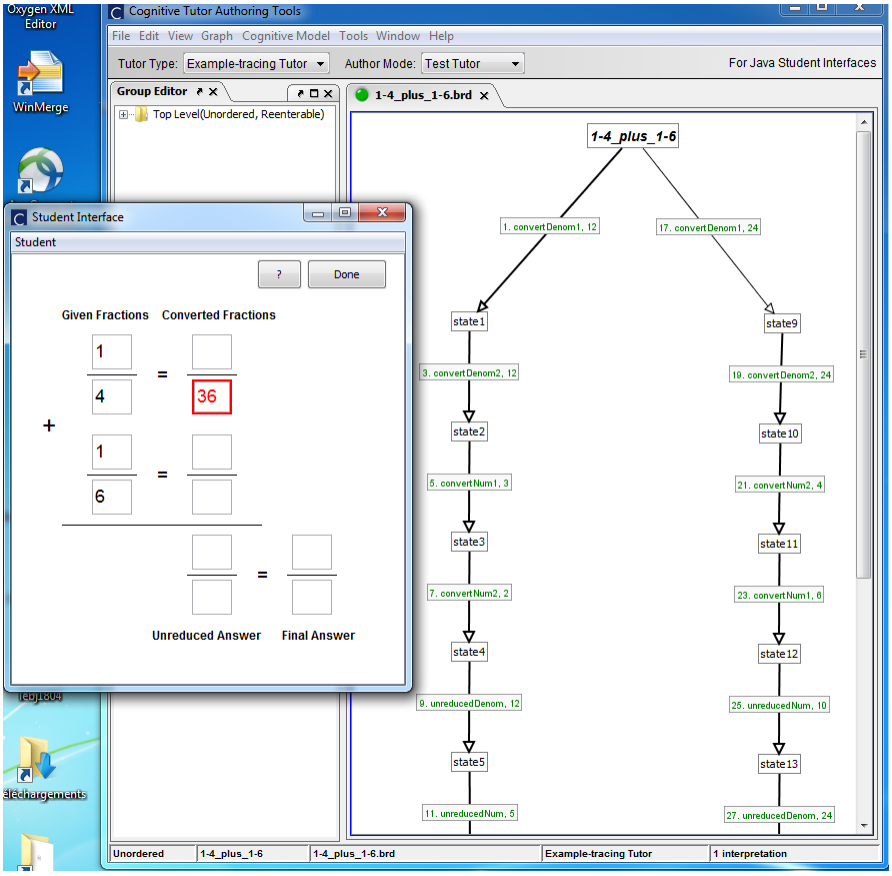

Figure 32 - Un exemple d'un ETT créé directement à l'aide de traces d'actions (ces dernières correspondent à deux façons différentes d'effectuer l'instance de la tâche « addition de fractions » où les opérandes sont 1/4 et 1/6).

La plateforme xPST [97] adopte une approche similaire aux ETT créés avec CTAT. Cette plateforme se distingue de deux façons, d'une part elle a été entièrement conçue dans le but de supporter des environnements existants (p. ex. le logiciel d'éditions d'image Paint.NET[217]). D'autre part, elle propose, pour créer l'équivalent du graphe des ETT, d'utiliser un DSL plutôt que des outils auteurs visuels. Ce DSL, puisqu'il est destiné à des auteurs n'ayant pas nécessairement d'habiletés de programmation, est plus limité que celui proposé pour représenter le modèle de la tâche dans Astus. Bien que le DSL d'xPST puisse

[217] http://www.getpaint.net/

s'avérer plus efficace que l'approche proposée par les ETT, les expérimentations rapportées dans la littérature ne montrent pas qu'il permet aux auteurs sans habiletés de programmation de créer un système équivalent à un MTT.

5.2.3 Les CBT

L'idée fondamentale derrière l'approche des CBT est de mettre l'accent sur un des mécanismes d'apprentissage, c'est-à-dire de tirer profit d'une rétroaction négative [158, 188, 190, 191]. Plus précisément, cette rétroaction négative est une forme particulière de rétroaction sur l'erreur où l'accent est mis sur l'état de l'environnement [218]:

- en fonction de l'action incorrecte produite plutôt qu'en fonction d'une action correcte qui aurait pu être produite;
- en fonction des effets de l'action produite plutôt qu'en fonction des actions mentales effectuées pour produire l'action[219].

Ce type de rétroaction nécessite des gabarits de messages (similaires à ceux des CT) associés à des contraintes qui sont vérifiées à chaque itération de la boucle interne. Les contraintes, à l'image de règles de production, s'expriment en fonction de deux composantes : la première correspond à la *condition de pertinence* et la deuxième qui n'est vérifiée que si la première s'avère, est la *condition de satisfaction*. Les contraintes sont dites *syntaxiques* lorsqu'elles représentent des connaissances propres au domaine ou à la tâche, mais indépendantes d'une instance donnée; elles sont dites *sémantiques* lorsqu'elles dépendent d'une « solution idéale » prédéfinie par le corps enseignant[220] (qui correspond à un état final de l'environnement ou de façon équivalente, à des actions). Par exemple, dans SQL-Tutor, les premières vérifient que les actions sont cohérentes avec la syntaxe du langage SQL tandis que les dernières vérifient que les actions sont cohérentes avec les tables qui se trouvent dans la « solution idéale »

[218] L'état de l'environnement reflète l'état de la KB implicite dans le langage hôte (Lisp)
[219] En fait, on peut considérer que les seules actions mentales prises en compte sont les perceptions
[220] Pour certaines tâches, la « solution idéale » peut être générée par un SABC dédié [159].

[162]. Un exemple d'une action dans SQL-Tutor est de définir la composante `Select` d'une requête SQL, cette action est produite à la suite d'une saisie dans une zone texte dans l'environnement (Figure 33).

Bien que les travaux de recherche sur SQL-Tutor et ASPIRE semblent avoir pris fin depuis quelques années (2007), SQL-Tutor est encore utilisé par les étudiants de University of Auckland et une version est accessible sur le Web[221].

Originalement, l'approche des CBT offrait trois avantages par rapport aux approches classiques, en particulier les MTT [188, 190] :

1. mettre l'accent sur un mécanisme d'apprentissage efficace pendant la phase de maitrise (tirer profit d'une rétroaction négative telle que décrite précédemment);

2. supporter une stratégie pédagogique qui est basée sur une évaluation plus souple que celle qu'on retrouve dans les MTT (les CT en particulier);

3. éviter les efforts de création prohibitifs associés aux MTT puisque les contraintes constituent un modèle plus simple que les règles de production ou les HTN.

Par la suite, un quatrième avantage a émergé du développement des premiers CBT : ces derniers seraient mieux adaptés que les MTT pour les tâches mal définies tirées de domaines bien définis [163]. Par exemple, l'écriture de requêtes SQL ou encore la conception d'une base de données à partir d'un texte en langage naturel [162, 246] (à l'origine, les tâches suggérées, comme l'arithmétique, l'écriture de structures de Lewis[222] ou la résolution d'équations, correspondaient aux tâches bien définies que l'on retrouve dans les MTT). Le développement des CBT a aussi amené leurs concepteurs à développer la plateforme ASPIRE[223] [161]. Cette dernière permet de réduire les efforts de création des CBT, par exemple en offrant des outils pour créer une application Web, et en offrant des « outils

[221] http://www.aw.com/databaseplacedemo/sqltutor.html
[222] http://en.wikipedia.org/wiki/Lewis_structure
[223] Elle intègre des travaux précédents : la plateforme WETAS [141] et les « outils auteurs » CAS [247].

auteurs » qui génèrent une partie des contraintes. En effet, des auteurs qui ont des habiletés de programmation doivent les compléter et les encoder (Figure 34).

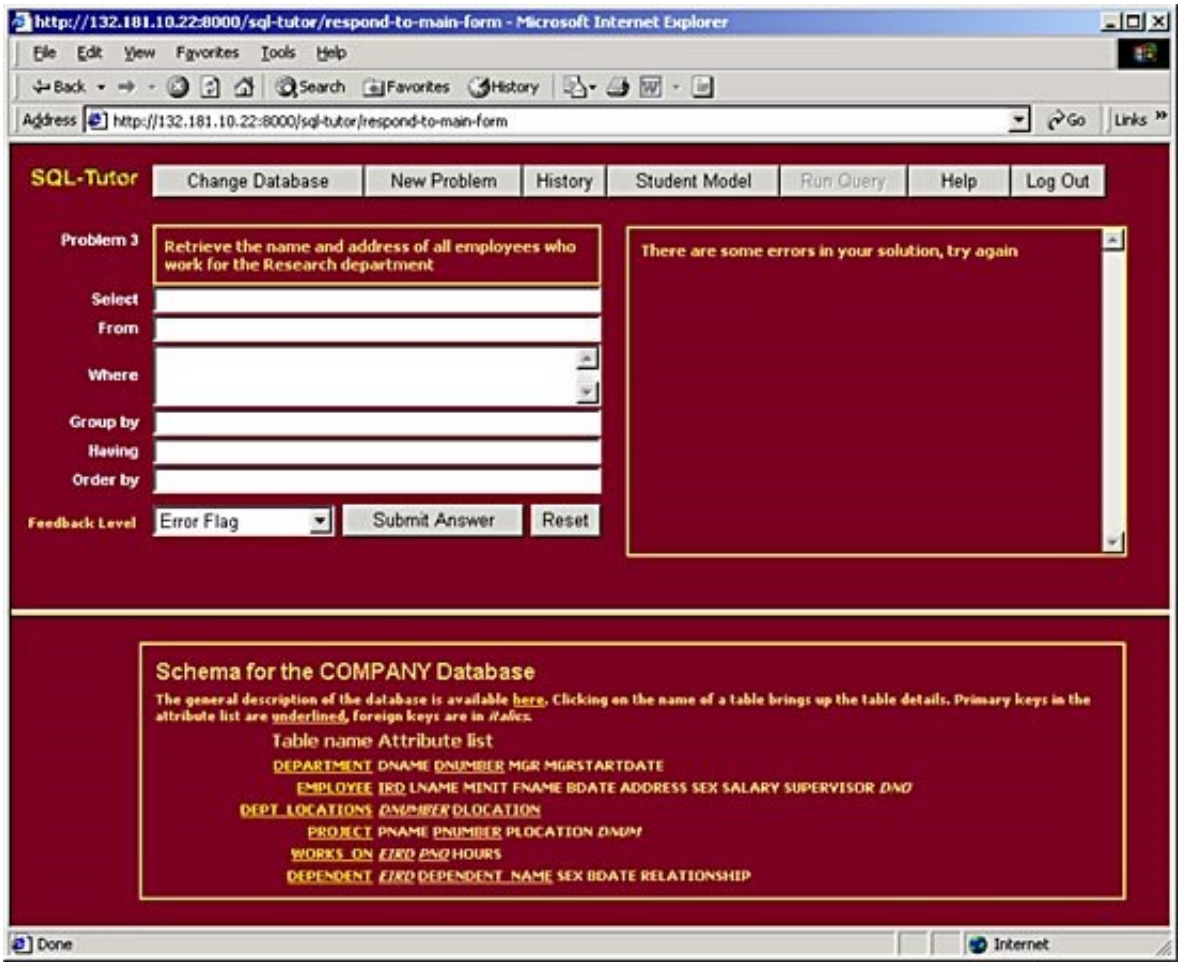

Figure 33 - Un aperçu de SQL-Tutor (tirée de http://ictg.canterbury.ac.nz/projects/sql-tutor)

En contrepartie, puisque la modélisation sous forme de contraintes se veut indépendante de la méthode utilisée, l'évaluation est moins précise et par conséquent le processus de suivi est limité dans ses interventions. Par exemple, lorsqu'il s'agit de produire un indice sur la prochaine étape ou une rétroaction positive, les CBT sont limités à extraire des contraintes des éléments qui ne sont pas encore validés ou qui viennent de l'être [158, 164]. En fait, même dans le cas de la rétroaction négative, les CBT sont limités puisqu'ils ne peuvent pas distinguer des incompréhensions qui sont des cas particuliers d'incompréhensions plus

générales. Par exemple, dans l'addition de fractions, distinguer l'addition des dénominateurs du cas plus général d'un « dénominateur commun » qui n'est pas un multiple des deux dénominateurs. Bien qu'il est possible d'ajouter des contraintes supplémentaires qui correspondent à des règles/PC représentant des incompréhensions, cela va à l'encontre des bonnes pratiques suggérées par les concepteurs des CBT [119, 163]. Dans la prochaine section, nous décrivons d'autres limitations des CBT en les comparant d'abord aux MTT classiques qui s'inscrivent dans les paradigmes de l'expert ou de l'apprenant puis à Astus.

Les CBT par rapport aux MTT

La littérature des CBT suggère que ces derniers sont en principe plus adaptés aux tâches mal définies, mais en pratique nous avons plutôt constaté que les MTT classiques peuvent supporter les mêmes tâches (les efforts de création peuvent varier). C'est d'ailleurs ce que laisse entendre une expérimentation menée conjointement par les concepteurs des CBT et des MTT [160]. En effet, même si ces deux approches sont fondamentalement différentes en principe, en pratique les astuces de modélisation utilisées font en sorte qu'elles deviennent équivalentes. Du côté des MTT, on retrouve d'une part l'utilisation de « solutions idéales », par exemple pour éviter le TAL nécessaire pour les « problèmes écrits ». D'autre part, dans des MTT comme Andes/ST, puisqu'on voulait obtenir un suivi comparable à celui offert par les CBT, on a utilisé des propriétés de la tâche et du domaine pour assouplir l'évaluation classique des MTT. En effet, dans les deux cas, l'apprenant est libre de produire les actions dans l'ordre voulu et dans le cas d'Andes, les équations sont créées à partir de saisies libres comme on en retrouve dans SQL-Tutor.

Du côté des CBT, pour plusieurs tâches, afin d'offrir une granularité raisonnablement fine au niveau de la boucle interne, il faut introduire des contraintes supplémentaires qui reflètent une méthode particulière plutôt que des principes du domaine. Par exemple, dans la soustraction en colonnes, pour suivre l'apprenant de façon plus précise que sur l'état final (la différence des deux nombres), il faut tenir compte de la méthode utilisée. Dans les environnements qui ont une dimension physique, de telles contraintes sont également nécessaires, parce que l'ordre des actions peut tenir compte d'un état interne qui n'est pas perceptible dans

l'environnement et que les actions peuvent avoir des effets indirects dans l'environnement (autrement dit, modifier l'état interne). Par exemple, en « soins critiques », bien que les actions à produire dépendent de l'état de l'environnement, l'ordre dans lequel elles sont produites est déterminé par un protocole qui a été élaboré de façon à tenir compte d'un état interne (l'état du patient ou encore l'état d'un appareil). Autrement dit, les contraintes doivent être définies de façon à refléter le protocole plutôt que de se limiter à vérifier l'évolution de l'état du patient qui est perceptible dans l'environnement.

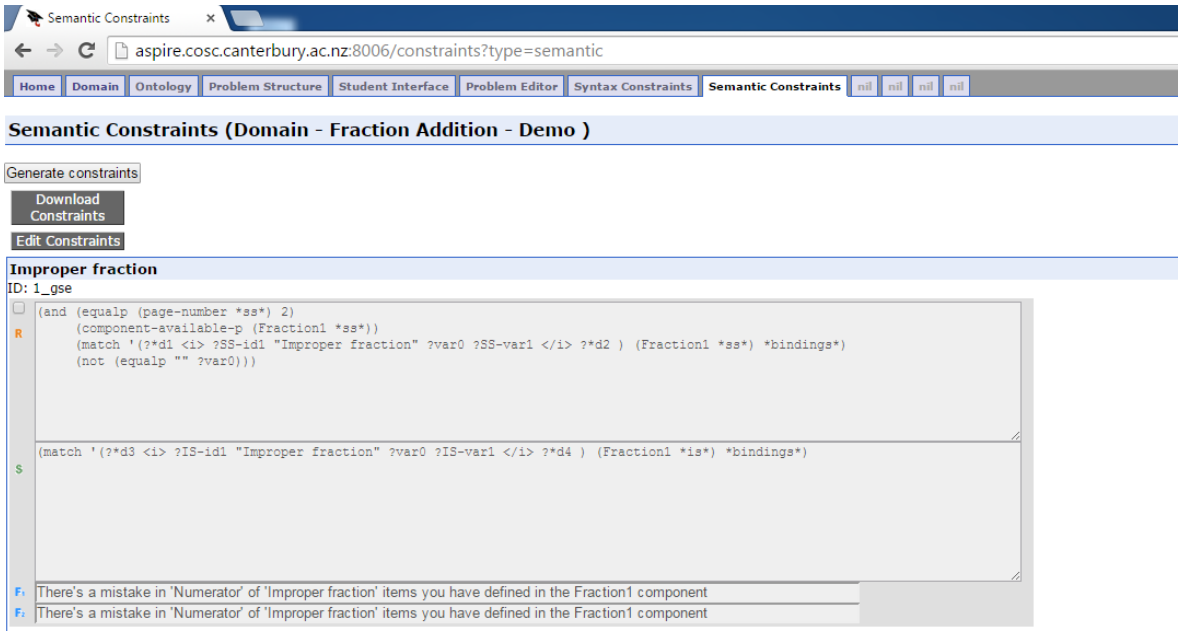

Figure 34 - Un aperçu de ASPIRE, en particulier l'application Web qui permet d'encoder les contraintes sous forme de code source Lisp. La tâche est l'addition de fractions.

Comme nous l'avons déjà évoqué, la principale limitation des MTT (Astus y compris) est la présence de « faux négatifs », c'est-à-dire des actions qui pourraient être considérées comme correctes par un tuteur humain, mais que le système considère comme incorrectes. Dans les CBT, on retrouve plutôt le problème inverse, c'est-à-dire la présence de « faux positifs ». En effet, les contraintes peuvent être poreuses, c'est-à-dire qu'elles laissent passer des actions qui seraient considérées comme incorrectes par un tuteur humain [158, 190]. En fin de compte, au-delà des propriétés de la tâche et du domaine dont elle est tirée, c'est la stratégie

pédagogique désirée qui fait en sorte que les auteurs d'un STI cherchent à minimiser soit les « faux négatifs » avec un CBT, soit les « faux positifs » avec un MTT.

Les CBT par rapport à Astus

En s'inscrivant dans le paradigme du tuteur pour créer un MTT avec Astus, on renonce à tirer profit d'une « solution idéale » et on s'engage à modéliser explicitement une (ou plusieurs) méthode qui permet d'effectuer la tâche. De plus, dans les environnements créés avec Astus, au lieu des saisies libres comme dans les CBT et dans un MTT comme Andes, on retrouve généralement une structure qui permet d'éviter les saisies (p. ex. avec du glisser-déposer) ou encore de rendre plus fine leur granularité, ce qui rend leur interprétation plus simple (p. ex. on peut ainsi éviter d'avoir recours à un système de calcul formel [180]). On peut donc croire qu'Astus rejette entièrement l'approche des CBT, ce qui n'est pas le cas.

Premièrement, Astus offre différents mécanismes pour assouplir l'évaluation, par exemple les « objets motifs » (à la différence des contraintes des CBT, ces derniers contiennent également un exemple qui peut être utilisé dans une intervention). Deuxièmement, les rétroactions négatives produites à l'aide du processus de diagnostic permettent de retrouver partiellement les rétroactions négatives des CBT. En effet, elles peuvent faire référence à l'état de l'environnement avant que l'action soit produite (si les actions mentales associées comportent des perceptions). De plus, elles ont l'avantage d'être générées plutôt que d'être prémâchées, ce qui fait qu'elles peuvent être adaptées en fonction de l'état de l'environnement. En contrepartie, elles ont certains inconvénients. D'abord, comme nous l'avons déjà souligné, les interventions prémâchées, qu'elles soient associées à des règles, des PC ou à des contraintes, ont toujours l'avantage de pouvoir faire référence à des connaissances qui ne se retrouvent pas dans le modèle. De plus, la validité cognitive du diagnostic peut être remise en cause, ce qui est partiellement contrebalancé par le fait qu'il est présenté sous forme interrogative. Finalement, le principal inconvénient est le fait que les interventions ne peuvent pas mettre en évidence la différence entre les effets dans l'environnement qui auraient dû être produits et ceux qui l'ont été par l'action diagnostiquée (ces derniers sont soulignés par la rétroaction minimale). En effet, comme nous l'avons déjà évoqué, Astus ne connait pas à l'avance les

effets d'une action qui n'a pas été exécutée. Toutefois, les interventions peuvent faire référence à l'action qui aurait dû être produite, ce qui est reconnu comme étant efficace [152].

Nous avons tenté de trouver une solution à ce problème en représentant plus formellement les effets des actions, par exemple en nous inspirant des travaux sur différents cadres logiques (p. ex. le « calcul de situations »[224]) et des langages qui en découlent comme Golog [219]. Or, il nous est apparu difficile d'intégrer un tel formalisme sans compromettre notre objectif d'améliorer le rapport entre les efforts de création et le potentiel d'efficacité des interventions générées. On peut remarquer que si on avait accès à un modèle des effets des actions, on pourrait également tenter d'inférer les effets des PC et des buts jusqu'à la racine du graphe procédural. Nous avons tenu compte de cet avantage potentiel dans notre évaluation, mais puisque ce type d'inférences demanderait d'importants travaux de recherche (la difficulté étant de généraliser et d'abstraire les effets), nous avons maintenu notre décision d'encoder les effets des actions de façon « boite noire ».

5.2.4 Les STI basés sur l'apprentissage automatique

Si les MTT souffrent de « faux négatifs » et les CBT, de « faux positifs », les STI basés sur l'apprentissage automatique souffrent des deux. Parmi les approches actuelles[225], on peut distinguer celles qui génèrèrent un modèle cognitif (celle de SimStudent) de celles qui se limitent à classifier les actions (p. ex. Deep Thought). La plateforme SimStudent [143] est indépendante du domaine, mais elle requiert un modèle de base qui décrit les actions et des prédicats propres au domaine et comme pour les CT[226], elle exige de l'auteur des interventions prémâchées. SimStudent utilise l'apprentissage automatique pour créer le modèle cognitif (c'est-à-dire les règles de production) à l'aide de traces d'actions fournies par les auteurs. Initialement, ces actions se limitaient à des actions de l'environnement, mais SimStudent tient maintenant compte d'actions qui correspondent à des interventions. Par exemple, l'auteur peut donner une rétroaction positive ou négative à une action produite par

[224] http://en.wikipedia.org/wiki/Situation_calculus
[225] Diligent [24], Demonstr8 [33] et Rides [169] les ont précédés.
[226] SimStudent est une extension de CTAT.

SimStudent. Autrement dit, SimStudent offre une façon d'obtenir un modèle cognitif en réduisant les efforts des auteurs dans le cas des tâches relativement bien définies (les tâches données en exemple sont la résolution d'une équation en algèbre élémentaire et la division en arithmétique). Puisque SimStudent se comporte en fin de compte comme un MTT, on peut le considérer davantage comme un outil qu'un type particulier de STI. Des travaux futurs pourront déterminer si un tel outil pourrait être intégré à Astus.

L'approche appliquée dans le STI pour les preuves en logique Deep Thought est complètement différente des autres approches évoquées dans cette thèse. En appliquant l'apprentissage automatique sur les traces d'actions accumulées (par des apprenants ou fournies par les auteurs), elle évalue directement les actions produites, sans passer par un modèle cognitif. Autrement dit, Deep Thought agit comme une « boite noire » entre chaque action et l'intervention qui s'en suit. L'évaluation des actions est basée sur leur similarité avec des actions issues des traces qui ont permis ou non d'accomplir la tâche. Par conséquent, les interventions sont moins précises que celles que l'on retrouve dans les MTT. Par contre, cette approche est particulièrement intéressante lorsque la tâche ou le domaine sont mal définis (p. ex. une argumentation légale [202]) puisque dans ce cas, la création d'un modèle cognitif est difficile, voire impossible [227] [90]. Ces STI dépendent des limites de l'apprentissage automatique et tant que l'IA forte reste inaccessible (et elle le sera peut-être toujours), la création d'un modèle cognitif, même si elle demande des efforts importants, permettra d'obtenir une évaluation plus précise.

5.3 Recommandations

L'approche pédagogique d'Astus parce qu'elle découle du paradigme du tuteur met en évidence les avantages et les désavantages des MTT : essentiellement, ils offrent une évaluation plus serrée, mais plus précise. En ce sens, Astus se démarque plus nettement des autres familles STI que les autres MTT et peut être considéré comme un archétype de ces derniers. Par conséquent, Astus peut servir de référence pour déterminer si l'approche des

[227] Autrement dit, c'est un problème IA-complet.

MTT doit être appliquée dans une situation donnée, c'est-à-dire en fonction d'une tâche et d'une stratégie pédagogique. Autrement dit, notre recommandation dans les situations pour lesquelles le paradigme du tuteur ne s'applique pas est qu'il est généralement préférable d'appliquer une approche propre à d'autres types de STI plutôt que celle des MTT. En principe, nous devrions recommander l'usage d'Astus par rapport à CTAT dans les situations où le paradigme du tuteur s'applique, mais en raison des limitations énoncées au chapitre 3, CTAT demeure la seule plateforme accessible en pratique.

Puisque les CBT ne sont pas, en pratique, si différents des MTT par rapport aux tâches qui sont ou ne sont pas supportées, et ce même si le paradigme du tuteur met en évidence leurs différences, nous recommandons l'approche des CBT lorsqu'une stratégie pédagogique basée sur l'évaluation serrée de l'apprenant ne convient pas au corps enseignant. Lorsque la tâche est réellement mal définie, les approches basées sur l'apprentissage automatique sont pour nous celles qui ont un plus grand potentiel.

Finalement, nous recommandons les approches basées sur l'apprentissage automatique lorsque la tâche n'est pas bien définie.

Conclusion

Contributions

Par cette thèse, nous avons voulu contribuer à renouveler l'intérêt pour les MTT en améliorant le rapport entre les efforts de création et l'efficacité potentielle des interventions, et en établissant plus clairement leur rôle pédagogique. Pour ce faire, nous avons développé la plateforme Astus qui permet d'explorer l'espace qui existe entre les MTT créés avec CTAT et les MTT qui ont recours à des connaissances didactiques sophistiquées (p. ex. des dialogues) comme Ms. Lindquist. Comme CTAT, Astus est inspirée de l'architecture classique des STI composée de quatre modules, mais Astus se distingue de CTAT en :

- supportant les tâches s'effectuant dans des environnements qui ont une dimension physique;
- en exploitant son modèle de la tâche pour générer des interventions plutôt que de recourir à des interventions prémâchées;
- en exploitant son modèle de l'UI pour produire des interventions riches comme les démonstrations.

La génération des interventions dans Astus se distingue de celle proposée par TOTS, car elle tire profit :

- d'un modèle de la tâche qui s'inscrit dans le paradigme du tuteur, c'est-à-dire qui représente une abstraction et une généralisation des instructions d'un tuteur humain;
- de DSL et d'outils comme l' « exploration automatique » et l' « apprenant simulé » qui contrebalance la complexité supplémentaire du modèle;

- de mécanismes d'extension qui permettent aux auteurs d'adapter la génération en fonction d'une stratégie pédagogique particulière.

Parce que le paradigme du tuteur, que nous avons introduit avec Astus, et la transparence qui en découle mettent en évidence les avantages et les désavantages des MTT, Astus se démarque plus nettement des autres familles de STI que les autres MTT. Astus, en tant qu'archétype, peut donc servir de référence pour déterminer si l'approche des MTT doit être appliquée dans une situation donnée, c'est-à-dire en fonction d'une tâche et d'une stratégie pédagogique. Selon les avantages et les désavantages des autres familles de STI, nous recommandons : les ETT si l'utilisation d'une plateforme comme Astus est jugée trop couteuse; les CBT si la tâche est relativement bien définie, mais qu'une évaluation serrée de l'apprenant ne convient pas et les approches basées sur l'apprentissage automatique lorsque la tâche n'est pas bien définie.

De plus, parce qu'Astus met en évidence les avantages et les désavantages des MTT, leur rôle pédagogique est plus clair pour les chercheurs en éducation ou en psychologie. Ces derniers sont donc mieux outillés pour évaluer la pertinence pédagogique des MTT.

Critique du travail

Même si nous avons atteint nos objectifs, nous sommes conscients que nous n'avons pas montré, sur une base empirique, qu'Astus est plus efficace que CTAT. Pour le faire, il faudrait procéder à des expérimentations mesurant le rapport entre les efforts de création et l'efficacité des interventions générées, et ce à l'aide de tâches qui ne sont pas tirées du domaine de l'informatique. Cependant, les ressources pour mener de telles expérimentations avec une méthodologie crédible (entre autres, en évitant les problèmes méthodologiques évoqués précédemment) ne sont pas disponibles au sein du laboratoire ASTUS.

Même si nous avons suggéré que la transparence qui découle du paradigme du tuteur permet d'éviter de différentes façons de renforcer des connaissances superficielles, nous ne pouvons pas prétendre que les MTT créés à l'aide d'Astus font en sorte, à eux seuls, que les

apprenants acquièrent des connaissances profondes. En ce sens, ils doivent non seulement être intégrés à d'autres EIAH, mais ils doivent aussi faire partie d'un curriculum qui comprend d'autres types d'activité d'apprentissage.

Travaux futurs de recherche

Tout au long de la thèse nous avons évoqué plusieurs pistes pour des travaux futurs : rendre plus sophistiqué le métamodèle de la tâche et celui de l'environnement, ou encore de tirer profit des outils de l'IA pour améliorer : le diagnostic des actions non tracées, l'exploration automatique, l'apprenant simulé, le modèle de l'apprenant minimal et la stratégie pédagogique. Une autre piste de travaux futurs est de collaborer avec d'autres chercheurs pour améliorer la génération des interventions en utilisant la GAT, en particulier pour supporter le français. Tôt ou tard, produire une version d'Astus qui permet un usage externe sera nécessaire pour tenir des expérimentations à plus grande échelle, ce qui permettra d'entreprendre des travaux de recherche en exploration de données. Une tout autre piste serait de créer un MTT pour la réduction d'expressions booléennes, une tâche qui a inspiré les travaux passés du groupe Astus (c'est une tâche qui peut faire l'objet d'expérimentations dans le cours de mathématiques discrètes en informatique).

Perspective

Parmi les EIAH, les STI, en particulier les MTT, ont un avenir incertain. En fait, même si ce sont les STI basés sur l'apprentissage automatique qui formeront le courant dominant dans les prochaines années, une plateforme comme Astus peut renouveler l'intérêt pour les MTT dans la communauté des EIAH, et ce, même si répéter l'exploit des CT, c'est-à-dire un vaste déploiement dans le milieu scolaire, est hors d'atteinte. Finalement, tant que l'IA forte reste inaccessible (et elle le sera peut-être toujours), la création d'un modèle génératif, même si elle demande des efforts importants, offre des avantages que les autres approches (celles basées sur l'apprentissage automatique, mais aussi les CBT et les STI créés à l'aide d'outils conçus pour des auteurs qui n'ont pas d'habiletés de programmation) ne peuvent pas offrir.

Annexe A

Laboratoire simulé de « génie génétique »

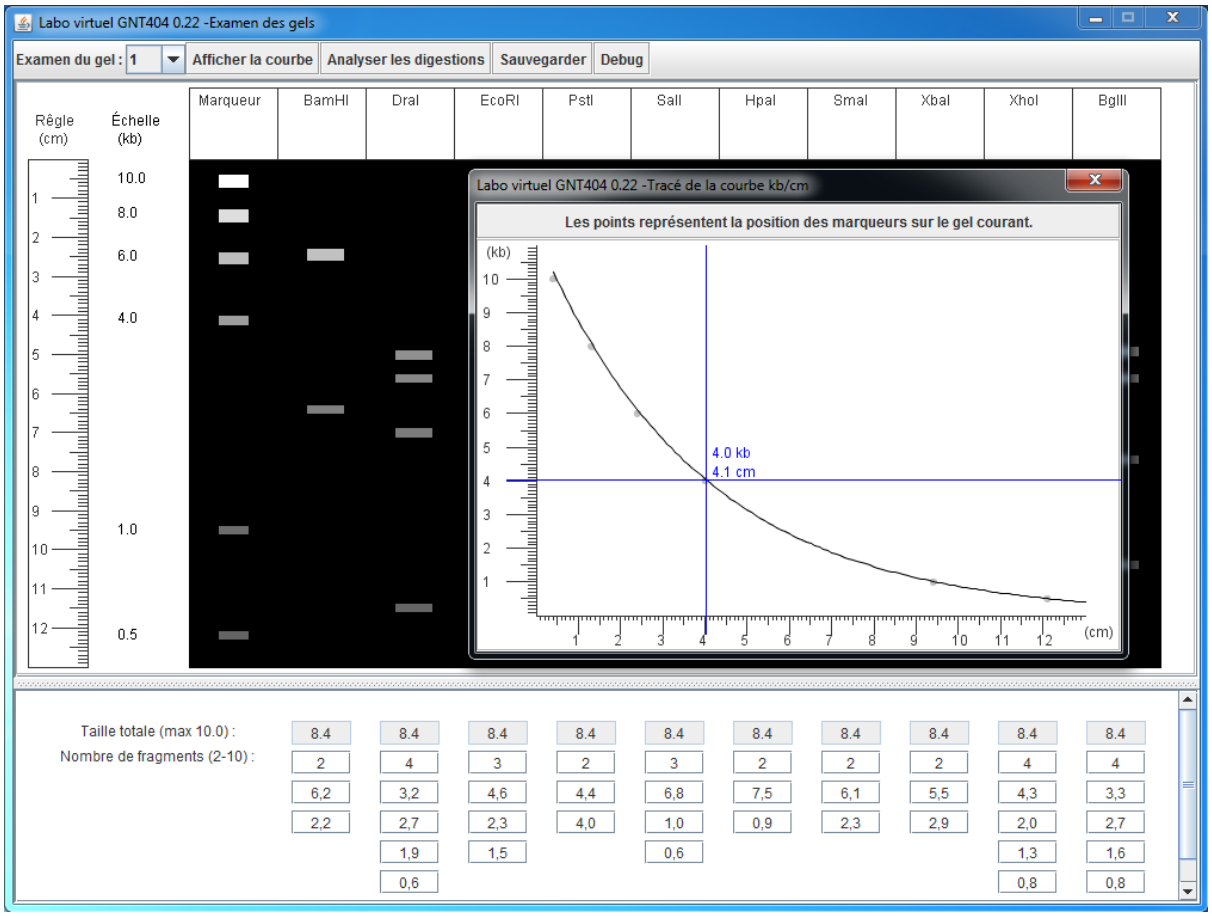

Figure 35 - La fenêtre d'examen du gel.

L'examen du gel (Figure 35) permet de mesurer les fragments issus des digestions de la molécule d'ADN par des enzymes de restriction. Dans le premier gel, il y a une digestion simple (une enzyme) pour chacune des dix enzymes choisies. Un marqueur de référence permet de construire une courbe qui fait correspondre la migration en cm et la taille en kb.

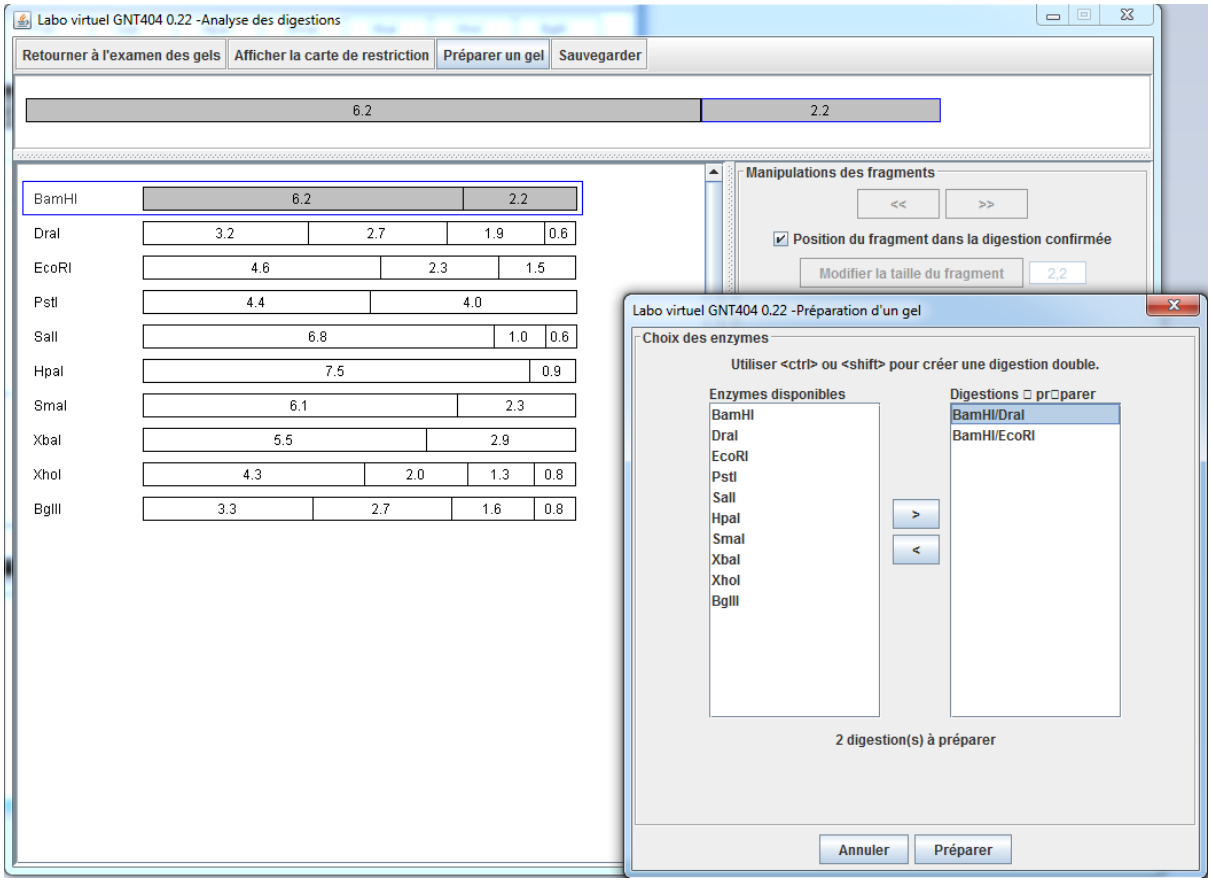

Figure 36 – La fenêtre d'analyse des digestions.

L'analyse des digestions (Figure 36) permet d'inférer la position des fragments, c'est-à-dire les sites de coupures des différents enzymes. Une enzyme avec un seul site de coupure est choisie comme référence (BamHI dans cet exemple) pour déterminer le sens de la carte de restriction (la carte miroir est tout aussi valide). Dans les gels suivants, on retrouve des digestions doubles ou triples qui sont nécessaires pour inférer la position des sites de coupures. Ces derniers sont finalement reportés sur la carte de restriction (Figure 37). Puisque certaines variations dans les tailles mesurées sont inévitables, l'évaluation de la carte est basée sur la position relative des sites.

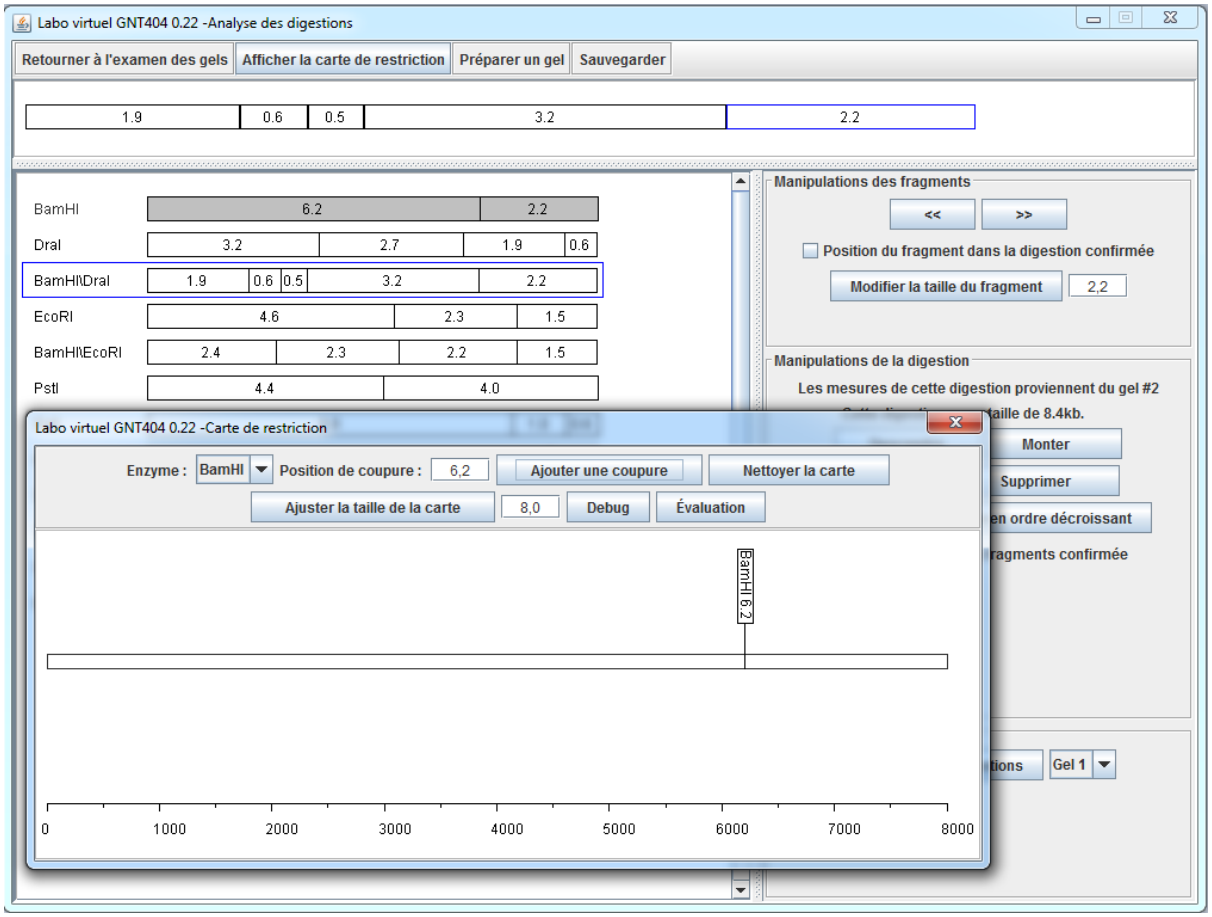

Figure 37 - La carte de restriction.

Annexe B

Le modèle de la soustraction en colonnes

```
concept('Cell', incomplete:true, readables:[en:'Cell']) {
                  value(integer, default:unknown)
                  attribute('overwritten', range:integer, type:seq)
}

concept('Difference', supers:[Cell]) {

}

concept('Column', readables:[en:'Column']) {
      part('top', range:Cell, readables:[en:'top'])
      part('bottom', range:Cell, readables:[en:'bottom'])
      part('result', range:Difference, readables:[en:'result'])
}

relation('onLeft, readables:[en:['on left']]) { // rule-bound
      place(Column, id:'c1')
      place(Column, id:'c2')
}

relation('onRight', readables:[en:['on right]])) { // rule-bound
      place(Column, id:'c1')
      place(Column, id:'c2')
}

opposite(onRight, onLeft)
inverse(onRight, onLeft)

function('fsubtraction', readables:[en:'subtraction'],
      script:{ t1.value - t2.value } ) {
      argument(Cell, id:'t1', readables:[en:'minuend'])
      argument(Cell, id:'t2', readables:[en:'subtraend'])
      image(integer)
}

function('plusTen', readables:[en:['plus ten']], script:{ t.value + 10 } ){
      argument(Cell, id:'t')
      image(integer)
}
```

```
function('minusOne', readables:[en:['minus one']], script:{ t.value - 1 } )
{
      argument(Cell, id:'t')
      image(integer)
}

context('CtxColumnSubtraction', init:{

      assert n2 > 0
      assert n1 > n2

      int minuend = n1
      int subtrahend = n2
      while (minuend != 0) {
            object(concept:Column, [
                  top:object(concept:Cell, [value:minuend % 10]),
                  bottom:object(concept:Cell, [value:subtrahend % 10]),
                  result:object(concept:Difference)])
            minuend =  minuend / 10
            subtrahend = subtrahend / 10
      }
}){
      tag(integer, id:'n1')
      tag(integer, id:'n2')
}

goal('GColumnSubtract', contexts:[CtxColumnSubtraction])

goal('GCalculateDifference', contexts:[CtxColumnSubtraction],
            readables:[en:'calculate the difference of #column']) {
      parameter('Column', id:'column')
}

goal('GBorrowFrom', contexts:[CtxColumnSubtraction],
                  readables:[en:'borrow from #column']) {
      parameter('Column', id:'column')
}

goal('GAddBorrow', contexts:[CtxColumnSubtraction],
                  readables:[en:'add borrow to #column']) {
      parameter('Column', id:'column')
}

goal('GBorrow', contexts:[CtxColumnSubtraction]) {
      parameter('Column', id:'col')
      sequence() {
            goal(GBorrowFrom, inputs:[column:least(all(concept:Column,
                  where:{exists(relation:onLeft, [$e, col])}),
                                                ord:onRight)])
            goal(GAddBorrow, inputs:[column:col])
            orderConstraints(1:2)
      }
}
```

```
goal('GDecrement', contexts:[CtxColumnSubtraction],
            readables:[en:'decrement #column']) {
      parameter('Column', id:'column')
}

goal('GEraseCell', contexts:[CtxColumnSubtraction],
                        readables:[en:'erase #c']) {
      parameter(Cell, id:'c')
}

goal('GOverwriteCell', contexts:[CtxColumnSubtraction],
                  readables:[en:'overwrite #c with #val']) {
      parameter('Cell', id:'c')
      parameter(integer, id:'val')
}

goal('GWriteCell', contexts:[CtxColumnSubtraction],
                  readables:[en:'write #val in #c']) {
      parameter(Cell, id:'c')
      parameter(integer, id:'val', readables:[en:'value'])
            primitive(script:{ reveal(t, val) }
)}

goal('GSubtractColumn',  contexts:[CtxColumnSubtraction],
                        readables:[en:'subtract #column']) {
      parameter('Column', id:'column')
      sequence() {
      goal(anon:{
            sufficient { greater(strict:false, from(column, {top.value})
                                          from(column, {bottom.value}))
                                    }
            sequence() {
                        goal(GBorrow, inputs:[col:column])
            }
      })
      goal(GCalculateDifference, inputs:[column:column])
      orderConstraints(1:2)
      }
}

forEach('CPColumnSubtract', goal:GColumnSubtract) {
      on(iterator:'c', order(all(concept:Column), ord:onRight))
      goal(GSubtractColumn, inputs:[column:c])
}

sequence('CPCalculateDifference', goal:GCalculateDifference) {
      var(id:'d', from(column, {result}))
      var(id:'v', unique(function:fsubtraction, [
            from(column, {top}),
            from(column, {bottom})
      ]))
      goal(GWriteCell, inputs:[t:d, val:v])
}
```

```
sequence('CPDecrement', goal:GDecrement) {
      var(id:'top', from(column, {top}))
      goal(GOverwriteCell, inputs:[t:top, val:unique(function:minusOne,
                                                     [top])])
}

sequence(goal:GBorrowFrom) {
      goal(inputs:[column:column], anon:{
            parameter(Column, id:'column')
            sufficient { greater(from(column, {top.value}), 0) }

            sequence() {
                  goal(GBorrow, inputs:[col:column])
            }
      })
      goal(GDecrement, inputs:[column:column])
      orderConstraints(1:2)
}


sequence('CPAddBorrow', goal:GAddBorrow) {
      var(id:'top', from(column, {top}))
      goal(GOverwriteCell, inputs:[t:top,
                                   val:unique(function:plusTen, [top])])
}

primitive('PPOverwriteCell', goal:GOverwriteCell, script:{
                  def old = t.value
                  change(t, val)
                  addTo(t, overwritten, old)
})

primitive('PPEraseCell', goal:GEraseCell, script:{ forget(t) } )


def P0n1 = 41
def P0n2 = 39

task(
      id:'P0',
      desc:[en:"$P0n1 - $P0n2"],
      context:CtxColumnSubtraction,
      goal:GColumnSubtract,
      init:{},
      inputs:{},
      boot: {
            tagged(n1:P0n1, n2:P0n2)
      }
)
```

```
(defrule onLeft
      (Column (object ?c1) (id ?id1))
      (Column (object ?c2) (id ?id2))
      (test (> (str-compare ?id1 ?id2) 0))
=>
      (fact "onLeft" ?c1 ?c2)
)

(defrule onRight
      (declare (salience -1))
      (Column (object ?c1))
      (Column (object ?c2))
      (onLeft (c1 ?c1) (c2 ?c2))
   =>
      (fact "onRight" ?c2 ?c1)
)

selection('NS', range:integer) {
      repeat(atLeastOnce:true) {
            pattern('p1', action(Cell, 'dh3'))
      }
      pattern('p2', action(Cell, 'dh4'))
      cancel(Cell, 'dh5')
      selected(extract(p1))
}

doStep(GWriteCell, parentView:Cell) {
      pattern('p1', action(Cell, 'dh1'))
      pattern('p2', action(Cell, 'dh2'))
      pattern('p3', NS)
      cancel(Cell, 'dh1')
      arg('c', owner(p1))
      arg('val', result(p3))
}

doStep(PPOverwriteCell, parentView:Cell) {
      pattern('p1', action(Cell, 'dh1'))
      pattern('p2', action(Cell, 'dh7'))
      pattern('p3', NS)
      cancel(Cell, 'dh1')
      arg('c', owner(p1))
      arg('val', result(p3))
}

doStep(PPEraseCell) {
      pattern('p1', action(Cell, 'dh1'))
      pattern('p2', action(Cell, 'dh6'))
      cancel(Cell, 'dh1')
      arg('c', owner(p1))
}
```

```
// Column View

view(

  build:{ c ->
        component(id:'top', subview(c.top, owner:$owner).container)
        component(id:'bot', subview(c.bottom, owner:$owner).container)
        component(id:'dif', subview(c.result, owner:$owner).container)
  },

  update:{ c ->  }
)

/* Des extraits de Cell View et CtxColumnSubtraction View ont été omis. */

// Cell View

view(
 build:{ c ->
  panel(id:'cp', new CellPanel())
  dialog(id:'nDlg', owner:$owner, new NumberDlg())

  handler(id:'dh1', type:ComponentClick, cp)
  handler(id:'dh6', type:ButtonPress, cp.eraseMenuItem)
  handler(id:'dh7', type:ButtonPress, cp.overwriteMenuItem)
  handler(id:'dh2', type:ButtonPress, cp.writeMenuItem)
  handler(id:'dh3', type:TextFieldValueChange, nDlg.textField)
  handler(id:'dh4', type:ButtonPress, nDlg.btn1)
  handler(id:'dh5', type:ButtonPress, nDlg.btn2)
  component(id:'container', cp)
},
  update : { c-> cp.update() })

// CtxColumnSubtraction View

view(
 build:{ cs->
   dialog(id:'sdlg', new SubstractionDlg())
   component(id:'window', sdlg)
}
```

Bibliographie

[1] Aleven, V. : Rule-Based Cognitive Modeling for Intelligent Tutoring Systems, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), Springer, 2010, p. 33‑62.

[2] Aleven, V., Koedinger, K. R., Sinclair, H. C., Snyder, J. : Combatting shallow learning in a tutor for geometry problem solving, dans *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, Goettl, B. P., Halff, H. M., Redfield, C. L., Shute, V. J. (Éd.), Springer, 1998, p. 364‑373.

[3] Aleven, V., Mclaren, B. M., Roll, I., Koedinger, K. R. : Toward Meta-cognitive Tutoring: A Model of Help-Seeking with a Cognitive Tutor, *International Journal of Artificial Intelligence in Education*, 16(1), 2006, p. 101‑130.

[4] Aleven, V., Mclaren, B. M., Sewall, J., Koedinger, K. R. : The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains, dans *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, Ikeda, M., Ashley, K. D., Chan, T.-W. (Éd.), Springer, 2006, p. 61‑70.

[5] Aleven, V., Mclaren, B. M., Sewall, J., Koedinger, K. R. : A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors, *International Journal of Artificial Intelligence in Education*, 19(2), 2009, p. 105‑154.

[6] Alfieri, L., Brooks, P., Aldrich, N., Tenenbaum, H. : Does discovery-based instruction enhance learning?, *Journal of Educational Psychology*, 103(1), 2011, p. 1‑18.

[7] Allen, J. F., Frisch, A. M. : What's in a semantic network?, dans *Proceedings of the 20th annual meeting on Association for Computational Linguistics*, Association for

Computational Linguistics, 1982, p. 19‑27.

[8] Alpert, D., Bitzer, D. L. : Advances in Computer-based Education, *Science*, 167(3925), 1970, p. 1582‑1590.

[9] Anderson, J. R. : *Language, Memory, and Thought*. Psychology Press, 1976.

[10] Anderson, J. R. : *The Atomic Components of Thought*. Psychology Press, 1998.

[11] Anderson, J. R. : *Rules of the Mind*. Lawrence Erlbaum, 1993.

[12] Anderson, J. R. : The Expert Module, dans *Foundations of Intelligent Tutoring Systems*, Polson, M. C., Richardson, J. J. (Éd.), Psychology Press, 1988, p. 21‑53.

[13] Anderson, J. R. : Skill Acquisition: Compilation of Weak-Method Problem Solutions, *Psychological Review*, 94(2), 1987, p. 192‑210.

[14] Anderson, J. R. : *The Architecture of Cognition*. Harvard University Press, 1983.

[15] Anderson, J. R., Boyle, C. F., Reiser, B. J. : Intelligent Tutoring Systems, *Science*, 228(4698), 1985, p. 456‑462.

[16] Anderson, J. R., Corbett, A. : Skill acquisition and the LISP tutor, *Cognitive Science*, 13, 1989, p. 467‑506.

[17] Anderson, J. R., Corbett, A., Boyle, C. F. : Cognitive modeling and Intelligent tutoring, *Artificial Intelligence*, 42(1), 1990, p. 7‑49.

[18] Anderson, J. R., Corbett, A., Koedinger, K. R. : Intelligent tutoring systems, dans *Handbook of Human-Computer Interaction*, 228(4698), Helander, M., Landauer, T. K., Prabhu, P. (Éd.), Elsevier Science, 1997, p. 849‑872.

[19] Anderson, J. R., Corbett, A., Koedinger, K. R., Pelletier, R. : Cognitive Tutors: Lessons Learned, *Journal of the Learning Sciences*, 4(2), 1995, p. 167‑207.

[20] Anderson, J. R., Pelletier, R. : A development system for MTT, dans *Proceedings of the International Conference of the Learning Sciences*, 1991, p. 1‑8.

[21] Anderson, J. R., Reder, L. M., Simon, H. : Radical Constructivism and Cognitive

Psychology, dans *Brooking papers on education policy*, Ravitch, D. (Éd.), Brooking Institute Press, 1998, p. 227‑278.

[22] Anderson, J. R., Reiser, B. J. : The LISP tutor, *Byte*, 10(4), 1985, p. 159‑175.

[23] Andriessen, J., Sandberg, J. A. C. : Where is education heading and how about AI?, *International Journal of Artificial Intelligence in Education*, 10(2), 1999, p. 130‑150.

[24] Angros, R., Johnson, W. L., Rickel, J., Scholer, A. : Learning domain knowledge for teaching procedural skills, dans *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM Press, 2002, p. 1372‑1378.

[25] Artale, A., Franconi, E., Guarino, N., Pazzi, L. : Part-whole relations in object-centered systems: An overview, *Data and Knowledge Engineering*, 20(3), 1996, p. 347‑383.

[26] Atkinson, R. C., Shiffrin, R. M. : Human Memory: A Proposed System and its Control Processes, *Psychology of Learning and Motivation - Advances in Research and Theory*, 2(C), 1968, p. 89‑195.

[27] Azevedo, R. : Beyond intelligent tutoring systems: Using computers as METAcognitive tools to enhance learning?, *Instructional Science*, 30(1), 2001, p. 31‑45.

[28] Baddeley, A. : *Working Memory, Thought, and Action*. Oxford University Press, 2007.

[29] Baddeley, A., Hitch, G. : Working memory, *Recent advances in learning and motivation*, 8(1), 1974, p. 47‑90.

[30] Baker, R. S. J., Corbett, A., Koedinger, K. R. : Learning to Distinguish Between Representations of Data: a Cognitive Tutor That Uses Contrasting Cases, dans *Proceedings of the International Conference of the Learning Sciences*, Kafai, Y. B., Sandoval, W. A., Enyedy, N. (Éd.), 2004, p. 58‑65.

[31] Baker, R. S. J., Corbett, A., Roll, I., Koedinger, K. R. : Developing a Generalizable Detector of When Students Game the System, *User Modeling and User-Adapted*

Interaction, 18(3), 2008, p. 287‑314.

[32] Baroody, A. J., Feil, Y., Johnson, A. R. : An Alternative Reconceptualization of Procedural and Conceptual Knowledge, *Journal for Research in Mathematics Education*, 38(2), 2007, p. 115‑131.

[33] Blessing, S. : A Programming by Demonstration Authoring Tool for Model-Tracing Tutors, *International Journal of Artificial Intelligence in Education*, 8(3), 1997, p. 233‑261.

[34] Blessing, S., Gilbert, S., Ritter, S., Ourada, S. : Authoring Model-Tracing Cognitive Tutors, *International Journal of Artificial Intelligence in Education*, 19(2), 2009, p. 189‑210.

[35] Bloom, B. : The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring, *Educational Researcher*, 13(6), 1984, p. 4‑16.

[36] Borgida, A., Brachman, R. : Conceptual Modeling with Description Logics, *The Description Logic Handbook: Theory, Implementation and Applications*, 2003, p. 359‑381.

[37] Brachman, R. : What's in a concept: structural foundations for semantic networks, *International Journal of Man-Machine Studies*, 9(2), 1977, p. 127‑152.

[38] Brachman, R. : What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks, *IEEE Computer*, 16(10), 1983, p. 30‑36.

[39] Brachman, R., Levesque, H. J. : *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2004.

[40] Brachman, R., Schmolze, J. G. : An Overview of the KL-ONE Knowledge Representation System, *Cognitive Science*, 9(2), 1985, p. 171‑216.

[41] Brown, J. S., Burton, R. : An investigation of computer coaching for informal learning activities, dans *Intelligent Tutoring Systems*, Brown, J. S., Sleeman, D. (Éd.), 1982, p. 157‑183.

[42] Brown, J. S., Burton, R. : Diagnostic Models for Procedural Bugs in Basic Mathematical Skills, *Cognitive science*, 2(2), 1978, p. 155‑192.

[43] Brown, J. S., Burton, R., De Kleer, J. : Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, dans *Intelligent Tutoring Systems*, Sleeman, D., Brown, J. S. (Éd.), 1982, p. 227‑282.

[44] Brown, J. S., Vanlehn, K. : Repair Theory: A Generative Theory of Bugs in Procedural Skills, *Cognitive Science*, 4, 1980, p. 379‑426.

[45] Bruillard, É. : *Les machines à enseigner*. Hermès, 1997.

[46] Brusilovsky, P. : Adaptive Hypermedia, *User Modeling and User-Adapted Interaction*, 11(1), 2001, p. 87‑110.

[47] Bull, S., Brna, P., Pain, H. : Extending the Scope of the Student Model, *User Modeling and User-Adapted Interaction*, 5(1), 1995, p. 45‑65.

[48] Burton, R. : Diagnosing bugs in a simple procedural skill, dans *Intelligent Tutoring Systems*, Sleeman, D., Brown, J. S. (Éd.), Erlbaum, 1982, p. 157‑183.

[49] Carbonell, J. R. : AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction, *IEEE Transactions on Man Machine Systems*, 11(4), 1970, p. 190‑202.

[50] Carbonell, J. R., Collins, A. : Natural Semantics in AI, dans *Proceedings of the 3rd international joint conference on Artificial intelligence*, Morgan Kaufmann, 1973, p. 334‑351.

[51] Ceri, S., Gottlob, G., Tanca, L. : What you always wanted to know about Datalog (and never dared to ask), *IEEE Transactions on Knowledge and Data Engineering*, 1(1), 1989, p. 146‑166.

[52] Chi, M., Jordan, P., Vanlehn, K. : When Is Tutorial Dialogue More Effective Than Step-Based Tutoring?, dans *12th International Conference on Intelligent Tutoring Systems*, Trausan-Matu, S., Boyer, K., Crosby, M., Panourigia, K. (Éd.), 2014, p. 210‑

219.

[53] Chi, M., Vanlehn, K. : Porting an Intelligent Tutoring System across Domains, dans *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, Greer, J., Lutkin, R., Koedinger, K. R. (Éd.), 2007.

[54] Chi, M., Vanlehn, K. : The Impact of Explicit Strategy Instruction on Problem-solving Behaviors across Intelligent Tutoring Systems, dans *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, McNamara, D., Trafton, J. G. (Éd.), Erlbaum, 2007, p. 167‑172.

[55] Chi, M., Vanlehn, K., Litman, D., Jordan, P. : An Evaluation of Pedagogical Tutorial Tactics for a Natural Language Tutoring System : A Reinforcement Learning, *International Journal of Artificial Intelligence in Education*, 21(1), 2011, p. 83‑113.

[56] Clancey, W. J. : From GUIDON to NEOMYCIN and HERACLES in Twenty Short Lessons: ORN Final Report 1979-1985, *AI Magazine*, 7(3), 1986, p. 40‑60.

[57] Clancey, W. J., Shortliffe, E. H., Buchanan, B. G. : Intelligent Computer-Aided Instruction for Medical Diagnosis, dans *Proc Annu Symp Comput Appl Med Care*, 75, 1979, p. 175‑183.

[58] Clark, R. E., Feldon, D. F., Yates, K., Early, S. : Cognitive Task Analysis, dans *Handbook of research on educational communications and technology (3rd ed.)*, Spector, J. M., Merill, M. D., Van Merriënboer, J. J. G., Driscoll, M. P. (Éd.), 2008.

[59] Clark, R. E., Kirschner, P. A., Sweller, J. : Putting Students on the Path to Learning The Case for Fully Guided Instruction, *American Educator*. p. 6‑11, 2012.

[60] Conati, C. : Bayesian Student Modeling, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), 2010.

[61] Conati, C. : Intelligent Tutoring Systems: New Challenges and Directions, dans *Proceedings of the 21st international jont conference on Artifical intelligence*, Kitano, H. (Éd.), Morgan Kaufmann, 2002, p. 2‑7.

[62] Conati, C., Gertner, A., Vanlehn, K. : Using Bayesian Networks to Manage Uncertainty in Student Modeling, *User Modeling and User-Adapted Interaction*, 12(1), 2002, p. 371‑417.

[63] Conati, C., Vanlehn, K. : POLA: a student modeling framework for Probabilistic On-Line Assessment of problem solving performance, dans *Proceedings of the 5th International Conference on User Modeling*, Carberry, S., Zuckerman, I. (Éd.), User Modeling Inc., 1996, p. 75‑82.

[64] Cooper, R. P., Shallice, T. : Contention Scheduling and the Control of Routine Activities, *Cognitive Neuropsychology*, 17(4), 2000, p. 297‑338.

[65] Cooper, R. P., Shallice, T. : Hierarchical schemas and goals in the control of sequential behavior, *Psychological review*, 113(4), 2006, p. 887‑916.

[66] Cooper, R. P., Shallice, T. : Structured representations in the control of behavior cannot be so easily dismissed: A reply to Botvinick and Plaut, *Psychological Review*, 113(4), 2006, p. 929‑931.

[67] Corbett, A., Anderson, J. R. : Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge, *User Modeling and User-Adapted Interaction*, 4(4), 1995, p. 253‑278.

[68] Corbett, A., Anderson, J. R. : Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate, Achievement and Attitudes, dans *CHI*, 2001, p. 245‑252.

[69] Corbett, A., Anderson, J. R., Patterson, E. : Student Modeling and Tutoring Flexibility in the Lisp Intelligent Tutoring System, dans *Intelligent tutoring systems: At the crossroads of artificial intelligence and education*, Frasson, C., Gauthier, G. (Éd.), Ablex, 1990, p. 83‑106.

[70] Corcho, O., Fernández-López, M., Gómez-Pérez, A. : Methodologies, tools and languages for building ontologies. Where is their meeting point?, *Data and Knowledge Engineering*, 46(1), 2003, p. 41‑64.

[71] Crowley, R. S., Legowski, E., Medvedeva, O. : Evaluation of an Intelligent Tutoring System in Pathology: Effects of External Representation on Performance Gains, Metacognition, and Acceptance, *J Am Med Inform Assoc*, 14(2), 2007, p. 182‑190.

[72] Crowley, R. S., Medvedeva, O. : An intelligent tutoring system for visual classification problem solving, *Artificial intelligence in Medicine*, 36(1), 2006, p. 85‑117.

[73] Dastani, M., van Riemsdijk, M. B., Winikoff, M. : Rich goal types in agent programming, dans *The 10th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2011, p. 405‑412.

[74] Davis, R., Shrobe, H., Szolovits, P. : What is a Knowledge Representation?, *AI Magazine*, 14(1), 1993, p. 17‑33.

[75] Desmarais, M., Baker, R. S. J. : A Review of Recent Advances in Learner and Skill Modeling in Intelligent Learning Environments, *User Modeling and User-Adapted Interaction*, 22(1), 2012, p. 9‑38.

[76] Dillenbourg, P., Schneider, D., Synteta, V. : Virtual Learning Environments, dans *Proceedings of the 3rd Congress on Information and Communication Technologies in education*, 2002, p. 3‑18.

[77] Dorça, F. : Implementation and use of Simulated Students for Test and Validation of new Adaptive Educational Systems: a Practical Insight, *International Journal of Artificial Intelligence in Education*, Sous press, 2015.

[78] Doyle, J., Patil, R. : Two Theses of Knowledge Representation, *Artificial Intelligence*, 48(3), 1991, p. 1‑33.

[79] Dubois, D., Gaha, M., Nkambou, R., Poirier, P. : Cognitive Tutoring System with « Consciousness », dans *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, Woolf, B. P., Aïmeur, E., Nkambou, R., Lajoie, S. (Éd.), 2008, p. 803‑806.

[80] Dzikovska, M., Steinhauser, N., Farrow, E., Moore, J. D., Campbell, G. : BEETLE II: Deep natural language understanding and automatic feedback generation for intelligent tutoring in basic electricity and electronics, *International Journal of Artificial Intelligence in Education*, 24(1), 2014, p. 284‑332.

[81] Eliot, C., Woolf, B. P. : An intelligent learning environment for advanced cardiac life support, dans *Proceedings of the AMIA Annual Fall Symposium*, Hanley & Belfus, 1996, p. 7‑11.

[82] Elmasri, R., Navathe, S. B. : *Fundamentals of Database Systems 6th Edition*. Pearson, 2010.

[83] Elsom-Cook, M. : Student modelling in intelligent tutoring systems, *Artificial Intelligence Review*, 7(3), 1993, p. 227‑240.

[84] Evens, M. W., Brandle, S., Chang, R., Glass, M., Jose, S., Michael, J. A., Rovick, A. A. : CIRCSIM-Tutor: An Intelligent Tutoring System Using Natural Language Dialogue, dans *Proceedings of the 12th Midwest AI and Cognitive Science Conference*, 2001, p. 16‑23.

[85] Fensel, D., Motta, E., Benjamins, R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Plaza, E., Schreiber, G., Studer, R., Wielinga, B. : The Unified Problem-solving Method Development Language UPML, *Knowledge and Information Systems*, 5(1), 2002, p. 83‑131.

[86] Fortin, M. : Cadre de travail pour l'implémentation d'interfaces graphiques dans ASTUS, Mémoire de maitrise, Université de Sherbrooke, 2010.

[87] Fortin, M., Lebeau, J.-F., Abdessemed, A., Courtemanche, F., Mayers, A. : A Standard Method of Developping User Interfaces for a Generic ITS, dans *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, Woolf, B. P., Aïmeur, E., Nkambou, R., Lajoie, S. (Éd.), Springer, 2008, p. 312‑322.

[88] Fournier-Viger, P. : Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents, Mémoire de maîtrise, Université de

Sherbrooke, 2005.

[89] Fournier-Viger, P., Nkambou, R., Mayers, A. : Evaluating spatial representations and skills in a simulator-based tutoring system, *IEEE Transactions on Learning Technologies*, 1(1), 2008, p. 63‑74.

[90] Fournier-Viger, P., Nkambou, R., Mephu Nguifo, E. : Building Intelligent Tutoring Systems for Ill-Defined Domains, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), Springer, 2010, p. 81‑101.

[91] Fournier-Viger, P., Nkambou, R., Najjar, M., Mayers, A. : From Black-box Learning Objects to Glass-Box Learning Objects, dans *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, Ikeda, M., Ashley, K. D., Chan, T.-W. (Éd.), Springer, 2006, p. 258‑267.

[92] Fry, E. : Teaching machine dichotomy: Skinner vs. Pressey, *Psychological Reports*, 6, 1960, p. 11‑14.

[93] Fuchs, N., Kaljurand, K., Kuhn, T. : Attempto Controlled English for Knowledge Representation, *Reasoning Web*, 2008, p. 104‑124.

[94] Gagné, R. M. : *The Conditions of Learning*. Holt, Rinehart & Winston, 1985.

[95] Gertner, A., Vanlehn, K. : Andes, a coached problem solving environment for physics, dans *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, Gauthier, G., Frasson, C., Vanlehn, K. (Éd.), Springer, 2000, p. 133‑142.

[96] Ghallab, M., Nau, D., Traverso, P. : *Automated Planning: theory and pratice*. Morgan Kaufmann, 2004.

[97] Gilbert, S., Blessing, S., Guo, E. : Authoring Effective Embedded Tutors: An Overview of the Extensible Problem Specific Tutor (xPST) System, *International Journal of Artificial Intelligence in Education*, 2015.

[98] Gill, A. : Domain-specific languages and code synthesis using Haskell, *Communications of the ACM*, 57(6), 2014, p. 42‑49.

[99] Gottlob, G., Zicari, R. : Closed World Databases Opened Through Null Values, dans *Proceedings of the 14th International Conference on Very Large Data Bases*, Bancilhon, F., DeWitt, D. J. (Éd.), Morgan Kaufmann, 1988, p. 50‑61.

[100] Graesser, A. C., Vanlehn, K., Rosé, C., Jordan, P., Harter, D. : Intelligent Tutoring Systems with Conversational Dialogue, *AI Magazine*, 22(4), 2001, p. 39‑52.

[101] Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-schneider, P., Sattler, U. : OWL 2: The Next Step for OWL, *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4), 2008, p. 309‑322.

[102] Guarino, N. : The Ontological Level: Revisiting 30 Years of Knowledge Representation, dans *Conceptual Modelling: Foundations and Applications. Essays in Honnor of John Mylopoulos*, Borgida, A., Chaudhri, V. K., Giorgini, P., Yu, E. (Éd.), Springer, 2009, p. 52‑67.

[103] Haapasalo, L. : The conflict between conceptual and procedural knowledge : Should we need to understand in order to be able to do, or vice versa ?, dans *Proceedings on the IXX symposium of the Finnish mathematics and science education research association*, Haapasalo, L., Sormunen, K. (Éd.), University of Joensuu, 2003, p. 1‑20.

[104] Han, Y.-S., Wang, Y., Wood, D. : Prefix-free regular languages and pattern matching, *Theoretical Computer Science*, 389(1), 2007, p. 307‑317.

[105] Heeren, B., Jeuring, J., Gerdes, A. : Specifying Rewrite Strategies for Interactive Exercises, *Mathematics in Computer Science*, 3(3), 2010, p. 349‑370.

[106] Heffernan, N., Heffernan, C. L. : The ASSISTments Ecosystem: Building a Platform that Brings Scientists and Teachers Together for Minimally Invasive Research on Human Learning and Teaching, *International Journal of Artificial Intelligence in Education*, 24(4), 2014, p. 470‑497.

[107] Heffernan, N., Koedinger, K. R., Razzaq, L. : Expanding the Model-Tracing Architecture: A 3rd Generation Intelligent Tutor for Algebra Symbolization,

International Journal of Artificial Intelligence in Education, 18(2), 2008, p. 153‑178.

[108] Hill, R. W., Johnson, W. L. : Situated Plan Attribution, *International Journal of Artificial Intelligence in Education*, 6(1), 1995, p. 35‑66.

[109] Hustadt, U. : Do we need the closed-world assumption in knowledge representation?, dans *Working Notes of the KI'94 Workshop: Reasoning about Structured Objects: Knowledge Representation Meets Databases*, DFKI, 1994, p. 24‑26.

[110] Johnson, W. L. : Understanding and debugging novice programs, *Artificial Intelligence*, 42(1), 1990, p. 51‑97.

[111] Johnson, W. L., Rickel, J., Lester, J. C. : Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments, *International Journal of Artificial Intelligence in Education*, 11(1), 2000, p. 47‑78.

[112] Jonassen, D. : Learning with technology: Using computers as cognitive tools, dans *Handbook of research for educational communications and technology*, Jonassen, D. (Éd.), Macmillan, 1996, p. 693‑719.

[113] de Jong, T., Ferguson-Hessler, M. G. M. : Types and Qualities of Knowledge, *Educational Psychologist*, 31(2), 1996, p. 105‑113.

[114] Jordan, P., Hall, B., Ringenberg, M. A., Cue, Y., Rosé, C. : Tools for Authoring a Dialogue Agent that Participates in Learning Studies, dans *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, 2007, p. 43‑50.

[115] Jraidi, I., Frasson, C. : Student's uncertainty modeling through a multimodal sensor-based approach, *Educational Technology and Society*, 16(1), 2013, p. 219‑230.

[116] Kay, J. : AI and education: Grand challenges, *IEEE Intelligent Systems*, 27(5), 2012, p. 66‑69.

[117] Kelly, A. E., Sleeman, D., Gilhooly, K. J. : Artificial intelligence in education: using state space search and heuristics in mathematics instruction, *International Journal of Man-Machine Studies*, 1993.

[118] Kirschner, P. A., Clark, R. E., Sweller, J. : Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching, *Educational Psychologist*, 41(2), 2006, p. 75‑86.

[119] Kodaganallur, V., Weitz, R. R., Rosenthal, D. : A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms, *International Journal of Artificial Intelligence in Education*, 15(2), 2005, p. 117‑144.

[120] Kodaganallur, V., Weitz, R. R., Rosenthal, D. : An Assessment of Constraint-Based Tutors: A Response to Mitrovic and Ohlsson's Critique of « A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms », *International Journal of Artificial Intelligence in Education*, 16(3), 2006, p. 291‑321.

[121] Koedinger, K. R., Aleven, V., Heffernan, N., Mclaren, B. M., Hockenberry, M. : Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration, dans *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, Lester, J. C., Vicari, R. M., Paraguaçu, F. (Éd.), Springer, 2004, p. 162‑174.

[122] Koedinger, K. R., Anderson, J. R. : Reifying implicit planning in geometry, dans *Computers as cognitive tools*, Lajoie, S., Derry, S. J. (Éd.), Erlbaum, 1993, p. 15‑46.

[123] Koedinger, K. R., Anderson, J. R., Hadley, W. H., Mark, M. A. : Intelligent Tutoring Goes To School in the Big City, *International Journal of Artificial Intelligence in Education*, 8(1), 1997, p. 30‑43.

[124] Koedinger, K. R., Corbett, A., Perfetti, C. : The knowledge-learning-instruction framework: bridging the science-practice chasm to enhance robust student learning, *Cognitive science*, 36(5), 2012, p. 757‑98.

[125] Krasner, G. E., Pope, S. T. : A cookbook for using the model-view controller user interface paradigm in Smalltalk-80, *Journal of Object-Oriented Programming*, 1(3), 1988, p. 26‑49.

[126] Laborde, J.-M. : Intelligent Microworlds and Learning Environments, dans *Intelligent Learning Environments: The Case of Geometry*, 1996, p. 113‑132.

[127] Laird, J. E. : Extending the Soar Cognitive Architecture, dans *Proceedings of the first conference on Artificial General Intelligence*, Wang, P., Goertze, B., Franklin, S. (Éd.), IOS Press, 2008, p. 224‑235.

[128] Lane, H. C., Johnson, W. L. : Intelligent Tutoring and Pedagogical Experience Manipulation in Virtual Learning Environments, dans *The PSI Handbook of Virtual Environments for Training and Education*, Schmorrow, D., Cohn, J. V, Nicholson, D. (Éd.), Greenwood Publishing Group, 2009, p. 393‑406.

[129] Langley, P., Choi, D. : A Unified Cognitive Architecture for Physical Agents, dans *Proceedings of the 21st National Conference on Artificial Intelligence*, AAAI Press, 2006, p. 1469‑1474.

[130] Langley, P., Ohlsson, S. : Automated Cognitive Modeling, dans *Proceedings of the 4th National Conference on Artificial Intelligence*, AAAI Press, 1984, p. 193‑197.

[131] Langley, P., Rogers, S. : An Extended Theory of Human Problem Solving, dans *Proceedings of the twenty-seventh annual meeting of the cognitive science society*, 2005.

[132] Lebeau, J.-F. : Une architecture de connaissance pour les systèmes tutoriels intelligents, Mémoire de maitrise, Université de Sherbrooke, 2007.

[133] Lebeau, J.-F., Fortin, M., Paquette, L., Mayers, A. : From Cognitive to Pedagogical Knowledge Models in Problem-Solving ITS Frameworks, dans *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, Lane, H. C., Yacef, K., Mostow, J., Pavlik, P. (Éd.), Springer, 2009, p. 731‑733.

[134] Lesgold, A., Bunzo, M., Lajoie, S., Eggan, G. : Sherlock: A coached practice environment for an electronics troubleshooting job, dans *Computer assisted instruction and intelligent tutoring systems Shared goals and complementary approaches*, Larkin, J., Chabay, R. (Éd.), Psychology Press, 1992, p. 201‑238.

[135] Lester, J. C. : Modeling self-efficacy in intelligent tutoring systems: An inductive approach, *User Modeling and User-Adapted Interaction*, 18(1), 2007, p. 81‑123.

[136] Levesque, H. J. : Planning with loops, *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005, p. 509‑515.

[137] Levesque, H. J., Brachman, R. : Expressiveness and tractability in knowledge representation and reasoning, *Computational intelligence*, 3(1), 1987, p. 78‑93.

[138] Lynch, C., Ashley, K. D., Pinkwart, N., Aleven, V. : Concepts, Structures, and Goals: Redefining Ill-Definedness, *International Journal of Artificial Intelligence in Education*, 19(3), 2009, p. 253‑266.

[139] Magliaro, S. G., Lockee, B. B., Burton, J. K. : Direct instruction revisited: A key model for instructional technology, *Educational Technology Research and Development*, 53(4), 2005, p. 41‑55.

[140] Major, N., Ainsworth, S., Wood, D. : REDEEM: Exploiting Symbiosis Between Psychology and Authoring Environments, *International Journal of Artificial Intelligence in Education*, 8(1), 1997, p. 317‑340.

[141] Martin, B., Mitrovic, A. : Authoring web-based tutoring systems with WETAS, dans *Proceedings of the International Conference on Computers in Education*, IEEE, 2002, p. 183‑187.

[142] Martin, J., Vanlehn, K. : Student assessment using Bayesian nets, *International Journal of Human-Computer Studies*, 42(6), 1995, p. 575‑591.

[143] Matsuda, N., Cohen, W. W., Koedinger, K. R. : Teaching the Teacher: Tutoring SimStudent Leads to More Effective Cognitive Tutor Authoring, *International Journal of Artificial Intelligence in Education*, 25(1), 2015, p. 1‑34.

[144] Matsuda, N., Vanlehn, K. : GRAMY: A Geometry Theorem Prover Capable of Construction, *Journal of Automated Reasoning*, 32(1), 2004, p. 3‑33.

[145] Mayer, A. J., Stockmeyer, L. J. : The complexity of word problems - this time with

interleaving, *Information and Computation*, 115(2), 1994, p. 293‑311.

[146] Mayer, R. E. : Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction, *The American psychologist*, 59(1), 2004, p. 14‑9.

[147] Mayers, A., Lefebvre, B. : La modélisation fine du processus résolution de problème dans Miace, dans *Proceedings of the 3rd Internaltional Conference on Intelligent Tutoring Systems*, Frasson, C., Gauthier, G., Mccalla, G. (Éd.), 1996, p. 148‑157.

[148] McArthur, D. : Tutoring techniques in Algebra, *Cognition and Instruction*, 7, 1990, p. 1997‑244.

[149] McArthur, D., Burdorf, C., Ormseth, T., Robyn, A., Stasz, C. : Multiple Representations of Mathematical Reasoning, dans *Proceedings of the 1st International Conference on Intelligent Tutoring Systems*, Frasson, C. (Éd.), 1988, p. 485‑490.

[150] McCalla, G. : The Central Importance of Student Modelling to Intelligent Tutoring, dans *New Directions for Intelligent Tutoring Systems Research*, Costa, E. (Éd.), Springer, 1992, p. 107‑131.

[151] McCarthy, J. : Programs with common sense, dans *Proceedings of the Symposium on the Mechanization of Thought Processes*, Her Majesty's Stationery Office, 1959.

[152] McKendree, J. : Effective Feedback Content for Tutoring Complex Skills, *Human-Computer Interaction*, 5(4), 1990, p. 381‑413.

[153] Menzel, C. : Knowledge representation, the World Wide Web, and the evolution of logic, *Synthese*, 182(2), 2009, p. 269‑295.

[154] Merrill, D. C., Reiser, B. J., Merrill, S. K., Landes, S. : Tutoring : Guided Learning by Doing, *Cognition and Instruction*, 13(3), 1995, p. 315‑372.

[155] Merrill, D. C., Reiser, B. J., Ranney, M., Trafton, J. G. : Effective tutoring techniques: A comparison of Human tutors and ITS, *The Journal of the Learning Sciences*, 2(3), 1992, p. 277‑305.

[156] Mikk, E., Lakhnech, Y., Siegel, M. : Hierarchical Automata as Model for Statecharts, dans *Proceedings of the Third Asian Computing Science Conference on Advances in Computing Science*, Shyamasundar, R. K., Ueda, K. (Éd.), Springer, 1997, p. 181‑196.

[157] Mitrovic, A. : Fifteen years of constraint-based tutors: what we have achieved and where we are going, *User Modeling and User-Adapted Interaction*, 22(1), 2011, p. 39‑72.

[158] Mitrovic, A. : Modeling Domains and Students with Constraint-Based Modeling, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), Springer, 2010, p. 63‑80.

[159] Mitrovic, A. : NORMIT: a Web-enabled tutor for database normalization, dans *Proceedings of the International Conference on Computers in Education*, IEEE, 2002, p. 1276‑1280.

[160] Mitrovic, A., Koedinger, K. R., Martin, B. : A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling, dans *Proceedings of the 9th International Conference on User Modeling*, Brusilovsky, P., Corbett, A., de Rosis, F. (Éd.), Springer, 2003, p. 313‑322.

[161] Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Mcguigan, N. : ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors, *International Journal of Artificial Intelligence in Education*, 19(2), 2009, p. 155‑188.

[162] Mitrovic, A., Ohlsson, S. : Evaluation of a Constraint-Based Tutor for a Database Language, *International Journal of Artificial Intelligence in Education*, 10(3), 1999, p. 238‑256.

[163] Mitrovic, A., Ohlsson, S. : A Critique of Kodaganallur, Weitz and Rosenthal, « A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms », *International Journal of Artificial Intelligence in Education*, 16(3), 2006, p. 227‑289.

[164] Mitrovic, A., Ohlsson, S., Barrow, D. K. : The effect of positive feedback in a constraint-based intelligent tutoring system, *Computers & Education*, 60(1), 2012, p.

264‑272.

[165] Mitrovic, A., Weerasinghe, A. : Revisiting Ill-Definedness and the Consequences for ITSs, dans *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, Dimitrova, V., Mizoguchi, R., Du Boulay, B., Graesser, A. C. (Éd.), IOS Press, 2009, p. 375‑382.

[166] Mostow, J., Aist, G. : Evaluating tutors that listen: An overview of Project LISTEN, dans *Smart Machines in Education*, Forbus, K., Feltovich, P. (Éd.), MIT Press, 2001, p. 169‑234.

[167] Motik, B., Sattler, U., Studer, R. : Query Answering for OWL-DL with Rules, *Web Semantics: Science, Services and Agents on the World Wide Web*, 2005.

[168] Mueller, E. T. : *Commonsense Reasoning*. Morgan Kaufmann, 2006.

[169] Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M., Wogulis, J. L. : Authoring Simulation-Centered Tutors with RIDES, *International Journal of Artificial Intelligence in Education*, 8(3), 1997, p. 284‑316.

[170] Murray, T. : EON: Authoring Tools for Content, Instructional Strategy, Student Model and Interface Design, dans *Authoring Tools for Advanced Technology Learning Environments*, Murray, T., Blessing, S., Ainsworth (Éd.), Springer, 2003, p. 309‑339.

[171] Murray, T. : An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art, dans *Authoring Tools for Advanced Technology Learning Environments*, Murray, T., Blessing, S., Ainsworth, S. (Éd.), Kluwer Academic Publishers, 2003, p. 493‑546.

[172] Najjar, M. : Une approche cognitive computationnelle de représentation de la connaissance au sein des environnements informatiques d'apprentissage, Thèse de doctorat, 2006.

[173] Najjar, M., Mayers, A. : AURELLIO: A Cognitive Computational Knowledge Representation Theory, *International Journal of Cognitive Informatics and Natural*

Intelligence, 1(3), 2007, p. 17‑35.

[174] Najjar, M., Fournier-Viger, P., Mayers, A., Hallé, J. : DOKGETT - An authoring tool for cognitive model-based generation of the knowledge, dans *Proceedings of the 5th International Conference on Advanced Learning Technologies,* IEEE, 2005, p. 371‑375.

[175] Newell, A. : *Unified Theories of Cognition*. Harvard University Press, 1990.

[176] Newell, A. : The knowledge level, *Artificial Intelligence*, 18(1), 1982, p. 87‑127.

[177] Newell, A., Laird, J. E., Rosenbloom, P. : Soar: an architecture for general intelligence, *Artificial Intelligence*, 33(1), 1987, p. 1‑64.

[178] Nguyen-Thinh, L., Loll, F., Pinkwart, N. : Operationalizing the Continuum between Well-Defined and Ill-Defined Problems for Educational Technology, *IEEE Transactions on Learning Technologies*, 6(3), 2013, p. 258‑270.

[179] Nicaud, J.-F. : Building ITSs to be used: Lessons Learned from the APLUSIX Project, dans *Lessons From Learning*, (I), Lewis, R. L., Mendelsohn, P. (Éd.), IFIP, 1994, p. 181‑198.

[180] Nicaud, J.-F., Bouhineau, D., Chaachoua, H. : Mixing Microworld and CAS Features in Building Computer Systems that Help Students Learn Algebra, *International Journal of Computers for Mathematical Learning*, 9(2), 2004, p. 169‑211.

[181] Nilsson, N. : Teleo-Reactive Programs for Agent Control, *Journal of Artificial Intelligence Research*, 1(1), 1994, p. 139‑158.

[182] Norman, D. A. : Categorization of Action Slips, *Psychological Review*, 1981.

[183] Norman, D. A., Shallice, T. : Attention to action: Willed and automatic control of behavior, dans *Consciousness and Self-Regulation*, Davidson, R. J., Schwartz, G. E., Shapiro, D. (Éd.), Springer, 1986, p. 1‑18.

[184] Nwana, H. : Intelligent tutoring systems: an overview, *Artificial Intelligence Review*, 4(4), 1990, p. 251‑277.

[185] Nye, B. D., Graesser, A. C., Hu, X. : AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring, *International Journal of Artificial Intelligence in Education*, 24(1), 2014, p. 427‑469.

[186] Ohlsson, S. : *Deep Learning: How the Mind Overrides Experience*. Cambridge University Press, 2011.

[187] Ohlsson, S. : Computational Models of Skill Acquisition, dans *The Cambridge Handbook of Computational Psychology*, Sun, R. (Éd.), Cambridge University Press, 2008, p. 359‑395.

[188] Ohlsson, S. : Learning from performance errors, *Psychological Review*, 103(2), 1996, p. 241‑262.

[189] Ohlsson, S. : Some principles of intelligent tutoring, *Instructional Science*, 14(3), 1986, p. 293‑326.

[190] Ohlsson, S. : Constraint-Based Student Modelling, *International Journal of Artificial Intelligence in Education*, 3(4), 1992, p. 429‑447.

[191] Ohlsson, S. : System Hacking Meets Learning Theory: Reflections on the Goals and Standards of Research in Artificial Intelligence and Education, *International Journal of Artificial Intelligence in Education*, 2(3), 1991, p. 5‑18.

[192] Oliver, A., Smiley, T. : *Plural Logic*. Oxford University Press, 2013.

[193] Orey, M. A., Nelson, W. A. : Development Principles for Intelligent Tutoring Systems: Integrating Cognitive Theory into the Development of Computer-Based Instruction, *Educational Technology Research and Development*, 41(1), 1993, p. 59‑72.

[194] Paquette, G., Lundgren-Cayrol, K., Léonard, M. : The MOT+ Visual Language for Knowledge Based Instructional Design, dans *Instructional Design: Concepts, Methodologies, Tools and Application*, IGI-Global, 2008, p. 697‑717.

[195] Paquette, L., Lebeau, J.-F., Beaulieu, G., Mayers, A. : Designing a Knowledge Representation Approach for the Generation of Pedagogical Interventions by MTTs,

International Journal of Artificial Intelligence in Education, 25(1), 2015, p. 118‑156.

[196] Paquette, L., Lebeau, J.-F., Beaulieu, G., Mayers, A. : Automating Next-Step Hints Generation Using ASTUS, dans *Proceedings of the 11th International Conference on Intelligent Tutoring Systems*, Cerri, S. A., Clancey, W. J., Papadourakis, G., Panourgia, K. (Éd.), Springer, 2012, p. 201‑211.

[197] Paquette, L., Lebeau, J.-F., Mayers, A. : Authoring Problem-Solving Tutors: A Comparison Between CTAT and ASTUS, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), Springer, 2010, p. 377‑405.

[198] Paquette, L., Lebeau, J.-F., Mayers, A. : Diagnosing Errors from Off-Path Steps in Model-Tracing Tutors, dans *Proceedings of the 16th International Conference on Artificial Intelligence in Education*, Lane, H. C., Pavlik, P., Mostow, J., Yacef, K. (Éd.), Springer, 2013, p. 611‑614.

[199] Paquette, L., Lebeau, J.-F., Mayers, A. : Generating Task-Specific Next-Step Hints Using Domain-Independant Structures, dans *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, Biswas, G., Bull, S., Kay, J., Mitrovic, A. (Éd.), Springer, 2011, p. 525‑527.

[200] Paquette, L., Lebeau, J.-F., Mayers, A. : Automating the Modeling of Learners' Erroneous Behaviors in Model-Tracing Tutors, dans *Proceedings of the 20th international conference on User Modeling, Adaptation, and Personalization*, Masthoff, J., Mobasher, B., Desmarais, M., Nkambou, R. (Éd.), Springer, 2012, p. 316‑321.

[201] Peot, M. A., Smith, D. E. : Conditional Nonlinear Planning, dans *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, 1992.

[202] Pinkwart, N., Ashley, K. D., Lynch, C., Aleven, V. : Evaluating an intelligent tutoring system for making legal arguments with hypotheticals, *International Journal of Artificial Intelligence in Education*, 19(4), 2009, p. 401‑424.

[203] Poslad, S. : Specifying protocols for multi-agent systems interaction, *ACM*

Transactions on Autonomous and Adaptive Systems, 2(4), 2007, p. 53‑77.

[204] Rau, M., Aleven, V., Rummel, N. : Intelligent tutoring systems with multiple representations and self-explanation prompts support learning of fractions, dans *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, Dimitrova, V., Mizoguchi, R., du Boulay, B. (Éd.), IOS Press, 2009, p. 441‑448.

[205] Reiser, B. J., Kimberg, D. Y., Lovett, M., Ranney, M. : Knowledge Representation and Explanation in GIL, An Intelligent Tutor for Programming, dans *Computer-Assisted instruction and Intelligent Tutoring Systems: Shared goals and complementary approaches*, Larkin, J. H., Chabay, R. W. (Éd.), Erlbaum, 1992, p. 111‑149.

[206] Reiter, R. : On Closed World Data Bases, dans *Readings in nonmonotonic reasoning*, Ginsberg, M. L. (Éd.), Morgan Kaufmann, 1987, p. 119‑140.

[207] Reye, J. : Student Modelling based on Belief Networks, *International Journal of Artificial Intelligence in Education*, 14(1), 2004, p. 1‑33.

[208] Rickel, J. : An intelligent tutoring framework for task-oriented domains, dans *Proceedings of the 1st International Conference on Intelligent Tutoring Systems*, Frasson, C. (Éd.), 1988, p. 109‑115.

[209] Rickel, J. : Intelligent Computer-Aided Instruction: A Survey Organized Around System Components, *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1), 1989, p. 40‑57.

[210] Rickel, J., Johnson, W. L. : Animated agents for procedural training in virtual reality: Perception, cognition, and motor control, *Applied Artificial Intelligence*, 13(4), 1999, p. 343‑382.

[211] Ritter, S., Anderson, J. R., Cytrynowicz, M., Medvedeva, O. : Authoring Content in the PAT Algebra Tutor, *Journal of Interactive media In Education*, 98(9), 1998, p. 1‑24.

[212] Ritter, S., Koedinger, K. R. : An architecture for plug-in tutor agents, *Journal of Artificial Intelligence in Education*, 7(3), 1996, p. 315‑347.

[213] Rittle-Johnson, B., Schneider, M. : Developing Conceptual and Procedural Knowledge of Mathematics, dans *Oxford handbook of numerical cognition*, Kadosh, R. C., Dowker, A. (Éd.), Oxford University Press, 2014.

[214] Rosé, C., Jordan, P., Vanlehn, K. : Interactive Conceptual Tutoring in Atlas-Andes, dans *Proceedings of the 10th International Conference on Artificial Intelligence in Education*, Moore, J. D., Redfield, C. L., Johnson, W. L. (Éd.), IOS Press, 2001, p. 256‑266.

[215] Rourke, E. O., Andersen, E., Gulwani, S., Popović, Z. : A Framework for Automatically Generating Interactive Instructional Scaffolding, dans *CHI*, 2015.

[216] Russell, S., Norvig, P. : *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[217] Sacerdoti, E. : The Nonlinear nature of plans, dans *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1975, p. 206‑213.

[218] Schank, R. C., Abelson, P. : *Scripts, Plans, Goals, and Understanding*. Earlbaum, 1977.

[219] Scherl, R., Levesque, H. J., Reiter, R., Lin, F., Lespérance, Y. : Golog: a Logic Programming for Dynamic Domains Language, *The Journal of Logic Programming*, 31(1), 1997, p. 59‑83.

[220] Schmidt, C. F., Sridharan, N. S., Goodson, J. L. : The plan recognition problem: An intersection of psychology and artificial intelligence, *Artificial Intelligence*, 11(1)‑(2), 1978, p. 45‑83.

[221] Schneider, M., Rittle-Johnson, B., Star, J. R. : Relations among conceptual knowledge, procedural knowledge, and procedural flexibility in two samples differing in prior

knowledge., *Developmental psychology*, 47(6), 2011, p. 1525‑38.

[222] Schwitter, R. : Controlled natural languages for knowledge representation, dans *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Huang, C.-R. (Éd.), CIPS, 2010, p. 1113‑1121.

[223] Self, J. : The defining characteristics of intelligent tutoring systems research: ITSs care, precisely, *International Journal of Artificial Intelligence in Education*, 10(3), 1999, p. 350‑364.

[224] Self, J. : Theoretical Foundations for Intelligent Tutoring Systems, *Journal of Artificial Intelligence in Education*, 1(4), 1990, p. 3‑14.

[225] Self, J. : Formal Approaches to Student Modelling, dans *Student Modelling: the key to individualized knowledge-based instruction*, 92(92), Mccalla, G., Greer, J. (Éd.), Springer, 1994, p. 295‑352.

[226] Self, J. : Bypassing the Intractable Problem of Student Modelling, dans *Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*, (41), Frasson, C., Gauthier, G. (Éd.), Ablex, 1990, p. 1‑26.

[227] Self, J. : Artificial Intelligence Techniques in CAI, *IEEE Transactions on Man Machine Systems*, 11(4), 1977, p. 190‑202.

[228] Self, J. : Student Models in Computer-aided Instruction, *International Journal of Man-Machine Studies*, 6(2), 1974, p. 261‑276.

[229] Shapiro, J. A. : An Algebra Subsystem for Diagnosing Students' Input in a Physics Tutoring System, *International Journal of Artificial Intelligence in Education*, 15(1), 2005, p. 1‑24.

[230] Shulman, L. S. : Those who understand: Knowledge growth in teaching, *Educational Researcher*, 15(2), 1986, p. 4‑31.

[231] Shute, V. J. : A Macroadaptive Approach to Tutoring, *International Journal of Artificial Intelligence in Education*, 4(1), 1993, p. 61‑93.

[232] Shute, V. J. : SMART: Student modeling approach for responsive tutoring, *User Modeling and User-Adapted Interaction*, 5(1), 1995, p. 1‑44.

[233] Shute, V. J., Psotka, J. : Intelligent Tutoring Systems: Past, Present and Future, dans *Handbook of research for educational communications and technology*, Jonassen, D. (Éd.), Macmillan, 1996, p. 570‑600.

[234] Simon, H. : The Structure of Ill Structured Problems, *Artificial Intelligence*, 4(1973), 1973, p. 181‑201.

[235] Simon, H., Newell, A. : *Human problem solving*. Prentice Hall, 1972.

[236] Skinner, B. F., Holland, J. G. : *The Analysis of Behavior: A Program for Self Instruction*. Mcgraw-Hill College, 1961.

[237] Sklavakis, D., Refanidis, I. : MATHESIS: An Intelligent Web-Based Algebra Tutoring, *International Journal of Artificial Intelligence in Education*, 22(1), 2013, p. 191‑218.

[238] Sleeman, D. : Pixie: A shell for Developing Intelligent Tutoring Systems, dans *Artificial Intelligence and education: learning environments and tutoring systems*, Lawler, R. W., Yazdani, M. (Éd.), Ablex, 1987, p. 239‑263.

[239] Stamper, J., Eagle, M., Barnes, T., Croy, M. : Experimental Evaluation of Automatic Hint Generation for a Logic Tutor, *International Journal of Artificial Intelligence in Education*, 22(1), 2012, p. 3‑18.

[240] Star, J. R. : Procedural Knowledge Reconceptualizing, *Journal for Research in Mathematics Education*, 36(5), 2005, p. 404‑411.

[241] Star, J. R. : Foregrounding Procedural Knowledge, *Journal for Research in Mathematics Education*, 38(2), 2007, p. 132‑135.

[242] Star, J. R., Rittle-Johnson, B. : Flexibility in problem solving: The case of equation solving, *Learning and Instruction*, 18(6), 2008, p. 565‑579.

[243] Stevens, A. L., Collins, A. : The Goal Structure of a Socratic Tutor, dans *Proceedings*

of the 1977 annual conference of the ACM, ACM Press, 1977, p. 256‑263.

[244] Sung-Young, J., Vanlehn, K. : Developing an intelligent tutoring system using natural language for knowledge representation, dans *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*, Aleven, V., Kay, J., Mostow, J. (Éd.), Springer, 2010, p. 355‑358.

[245] Suraweera, P., Mitrovic, A. : An Intelligent Tutoring System for Entity Relationship Modelling, *International Journal of Artificial Intelligence in Education*, 14(3), 2004, p. 375‑417.

[246] Suraweera, P., Mitrovic, A. : KERMIT: a constraint-based tutor for database modeling, dans *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, Ceri, S., Gouardères, G., Paraguaçu, F. (Éd.), Springer, 2002, p. 201‑216.

[247] Suraweera, P., Mitrovic, A., Martin, B. : Widening the Knowledge Acquisition Bottleneck for Constraint-based Tutors, *International Journal of Artificial Intelligence in Education*, 20(2), 2010, p. 137‑173.

[248] Sweller, J., Kirschner, P. A., Clark, R. E. : Why Minimally Guided Teaching Techniques Do Not Work: A Reply to Commentaries, *Educational Psychologist*, 42(2), 2007, p. 115‑121.

[249] Taatgen, N., Anderson, J. R. : The Past, Present, and Future of Cognitive Architectures, *Topics in Cognitive Science*, 2(4), 2010, p. 693‑704.

[250] Tchounikine, P. : Pour une ingénierie des Environnements Informatiques pour l'Apprentissage Humain, *Revue I3*, 2(1), 2002, p. 1‑37.

[251] Tulving, E. : Episodic and Semantic Memory, dans *Organization of Memory*, Tulving, E., Donaldson, W. (Éd.), Academic Press, 1972, p. 381–402.

[252] Ur, S., Vanlehn, K. : Steps: A Simulated, Tutorable Physics Student, *International Journal of Artificial Intelligence in Education*, 6(4), 1995, p. 1‑37.

[253] Vanlehn, K. : The Relative Effectiveness of Human Tutoring, Intelligent Tutoring

Systems, and Other Tutoring Systems, *Educational Psychologist*, 46(4), 2011, p. 197‑221.

[254] Vanlehn, K. : The Behavior of Tutoring Systems, *International Journal of Artificial Intelligence in Education*, 16(3), 2006, p. 227‑265.

[255] Vanlehn, K. : Student Modeling, dans *Foundations of Intelligent Tutoring Systems*, Polson, M., Richardson, J. (Éd.), Erlbaum, 1988, p. 55‑78.

[256] Vanlehn, K. : *Mind Bugs: The origins of Procedural Misconceptions*. MIT Press, 1990.

[257] VanLehn, K. : Learning one subprocedure per lesson, *Artificial Intelligence*, 31(1), 1987, p. 1‑40.

[258] Vanlehn, K. : Human procedural skill acquisition: theory, model and psychological validation, dans *Proceedings of the Third National Conference on Artificial Intelligence*, AAAI Press, 1983, p. 420‑423.

[259] Vanlehn, K. : Cognitive skill acquisition, *Annual review of psychology*, 47(1), 1996, p. 513‑39.

[260] Vanlehn, K., Freedman, R. K., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rosé, C., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., Wintersgill, M. : Fading and Deepening: the Next Steps for Andes and other Model-Tracing Tutors, dans *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, Gauthier, G., Frasson, C., VanLehn, K. (Éd.), Springer, 2000, p. 474‑483.

[261] Vanlehn, K., Graesser, A. C., Rosé, C. : When are tutorial dialogues more effective than reading?, *Cognitive science*, 31(1), 2007, p. 3‑62.

[262] Vanlehn, K., Koedinger, K. R., Skogsholm, A., Nwaigwe, A., Hausmann, R. G. M., Weinstein, A., Billings, B. : What's in a Step? Toward General, Abstract Representations of Tutoring System Log Data, dans *Proceedings of the 11th International Conference on User Modeling*, McCoy, K., Conati, C. (Éd.), Springer,

2007, p. 455‑459.

[263] Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M. : The Andes Physics Tutoring System: Lessons Learned, *International Journal of Artificial Intelligence in Education*, 15(3), 2005, p. 1‑47.

[264] Vanlehn, K., Niu, Z. : Bayesian Student modeling, user interfaces and feedback: A sensitivity analysis, *International Journal of Artificial Intelligence in Education*, 12(2), 2001, p. 154‑184.

[265] Vanlehn, K., Ohlsson, S., Nason, R. : Applications of Simulated Students: An Exploration, *International Journal of Artificial Intelligence in Education*, 5(2), 1993, p. 135‑175.

[266] Vanlehn, K., Rosé, C., Jordan, P. : The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing, dans *Proceedings of the 6th international conference on Intelligent Tutoring Systems*, Cerri, S., Gouardères, G., Paraguaçu, F. (Éd.), Springer, 2002, p. 158‑167.

[267] Vanlehn, K., Van De Sande, B., Shelby, R. : The Andes physics tutoring system: An experiment in freedom, dans *Advances in Intelligent Tutoring Systems*, Nkambou, R., Bourdeau, J., Mizoguchi, R. (Éd.), Springer, 2010, p. 421‑443.

[268] Waalkens, M., Aleven, V., Taatgen, N. : Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems, *Computers & Education*, 60(1), 2012, p. 159‑171.

[269] Wang, H., Tu, S., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tudorache, T., Horridge, M., Seidenberg, J. : Frames and OWL Side by Side, dans *Proceedings of the 9th International Protégé Conference*, 2006, p. 1‑4.

[270] Welty, C., Ferrucci, D. A. : What's in an Instance?, RPI Computer Science Department, 1994.

[271] Wenger, E. : *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann, 1987.

[272] Woolf, B. P. : *Building Intelligent Interactive Tutors*. Morgan Kaufmann, 2008.

[273] Woolf, B. P., Burleson, W., Arroyo, I., Picard, R. : Affect-aware tutors: recognising and responding to student affect, *International Journal of Learning Technology*, 4(3), 2009, p. 129‑164.

[274] Zyda, M. : From Visual Simulation to Virtual Reality to Games, *Computer*, 38(9), 2005, p. 25‑32.

[275] Grandbastien, M., Labat, J.-M., Éd. : *Environnements informatiques pour l'apprentissage humain*. Hermes Science Lavoisier, 2006.

[276] Lajoie, S., Derry, S. J., Éd. : *Computers as cognitive tools*. Lawrence Erlbaum, 1993.

[277] Lajoie, S., Éd. : *Computers as cognitive tools: No more walls*. Lawrence Erlbaum, 2000.

[278] Nkambou, R., Mizoguchi, R., Bourdeau, J., Éd. : *Advances in Intelligent Tutoring Systems*. Springer, 2010.

[279] Cypher, A., Éd. : *What What I Do: Programming By Demonstration*. MIT Press, 1993.

[280] Sleeman, D., Brown, J. S., Éd. : *Intelligent tutoring systems*. Academic Press, 1982.