Langage de spécification et de description de présentations multimédias

Stéphane Lo Presti

Institut National Polytechnique de Grenoble

THÈSE

pour obtenir le grade de

Docteur de l'INPG

Spécialité: Informatique: Systèmes et Communications

préparée au laboratoire **LSR-IMAG**

dans le cadre de l'**Ecole Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

présentée et soutenue publiquement par

Stéphane LO PRESTI

le 27 Novembre 2002

Langage de spécification et de description de présentations multimédias

---

Directeurs de thèse: Andrzej DUDA et Didier BERT

Composition du jury:

Brigitte Plateau	Présidente
Pierre-Yves Schobbens	Rapporteur
Serge Miguet	Rapporteur
Andrzej Duda	Directeur de thèse
Didier Bert	Co-encadrant

À la mémoire de
mon père Jean-Baptiste.

Je tiens à remercier Andrzej Duda, Professeur à l'INPG, et Didier Bert, Directeur de Recherche au CNRS, pour m'avoir encadré tout au long de ces années de thèse, ainsi que Paul Jacquet, précédemment directeur du laboratoire LSR, pour m'avoir accueilli au laboratoire LSR.

Je remercie MM. Pierre-Yves Schobbens, Professeur à l'Université de Namur (Belgique), et Serge Miguet, Professeur à l'Université Lumière (Lyon 2), pour m'avoir fait l'honneur d'être les rapporteurs de ce travail et Mme Brigitte Plateau, Professeur à l'INPG, pour celui de présider le jury de cette thèse.

Je remercie le Centre National de la Recherche Scientifique pour le soutien financier lors des trois premières années de cette thèse.

Je remercie également les nombreuses personnes dont la route a croisé la mienne: Ibtissem Hassine pour son aide généreuse, Patrice Uvietta et Christiane Plumeré pour nos nombreuses discussions intéressantes, Patrick Palmer et toute l'équipe du département Info-Com de l'I.U.T. 2 de Grenoble pour leur sympathie, François Delobel et Caroline Signol pour leur amitié, François-Xavier Marseille et Philippe et Céline Sidler pour leur complicité, Dominique Decouchant pour sa coopération, Franck Rousseau et Leyla Toumi pour leur appui, ainsi que tous les autres collègues du laboratoire LSR, en particulier ceux des équipes Drakkar et Vasco.

Des remerciements particuliers vont à Shamima Paurobally dont le soutien et l'aide inconditionnelles m'ont été des plus précieux dans mon travail.

Des mercis chaleureux à ma sœur qui m'a offert ses pensées fraternelles. Milles mercis affectueux à ma mère sans qui je n'aurai pu aller aussi loin et qui m'a offert tout son amour sans compter, ainsi qu'à mon père qui m'a donné les moyens d'aller jusqu'à cette thèse sans avoir pu en voir l'accomplissement.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Des médias au multimédia

Depuis longtemps, les informations sont diffusées au travers des *mass médias* populaires comme les journaux papiers ou les stations de radio. La télévision fut le premier moyen de communiquer via deux médias, le son et l'image animée, une véritable révolution qui commença après la seconde guerre mondiale. On peut dire qu'elle fut d'une certaine manière la première brique du multimédia et son succès s'est étendu à des industries parallèles, en sortant du "petit écran" comme dans le cas du cinéma numérique aujourd'hui. Les médias audiovisuels "de masse" ont depuis une vingtaine d'années continué leur essor en interagissant les uns les autres (cinéma à la télévision, télévision à la radio, etc.)

L'informatique est apparue en tant que la science du traitement automatique de l'information et a tissé des liens avec le monde audiovisuel ces dernières années. C'est aujourd'hui un outil privilégié, voire incontournable, dans la plupart des milieux professionnels. Il a permis de fournir des techniques et d'impulser des initiatives afin de résoudre des problèmes comme celui de la diversité des formats de données manipulées ou des méthodes de diffusion. De grandes tendances existent aujourd'hui au sein de l'informatique et le *multimédia* est l'une des plus populaires d'entre elles. Le développement du multimédia a été important depuis 1995, évoluant de problématiques technologiques de bas niveau jusqu'à un haut niveau d'abstraction.

Les données ne sont aujourd'hui plus simplement radiodiffusées (ou télédiffusées) pour des postes analogiques mais sont distribuées à travers des réseaux numériques à l'intention de plate-formes multimédias, comme l'ordinateur, les consoles de jeu ou les terminaux de télévision numérique. Le lien de l'informatique à l'industrie audiovisuelle traditionnelle a rendu possible la large diffusion de ces contenus à travers l'usage intensif des réseaux, particulièrement celui de l'Internet.

Ces technologies multimédias et leurs usages se retrouvent au cœur de ce que l'on appelle aujourd'hui la société de l'information, participant à l'élaboration de modèles tant économiques, sociaux que techniques. Le multimédia occupe une place importante au sein de cette organisation en se faisant à la fois le support des données et celui de divers concepts, comme par exemple la créativité. Le terme multimédia a été repris dans de nombreux contextes, comme le marketing où il est passé d'argument de vente à un véritable intérêt pour le public ou bien dans le contexte des créateurs de contenu, ou auteurs, où il a été à l'origine d'une multitude de travaux originaux. Cette diversité a engendré des confusions qui ont mis plusieurs années à se dissiper. Le multimédia organise aujourd'hui son domaine de recherche et de travail en dressant les grandes lignes de ses futures évolutions.

1.2 Problématiques du domaine

Les progrès technologiques de l'informatique ont porté et ont été portés par l'activité multimédia, en particulier dans la micro-informatique. Le multimédia a été initialement poussé par le développement des ordinateurs personnels dans deux directions: le matériel avec des périphériques très variés (cartes vidéo, son et accélératrice 3D; écrans plats; enceintes 5+1; etc.) qui ont permis de créer et rendre des informations monomédias de manière de plus en plus riche et évoluée; et d'autre part le logiciel avec des applications ou des systèmes qui organisent, manipulent et donnent accès à ces données. Après une courte présentation de ces deux aspects, nous nous concentrons sur les facteurs temps et espace, qui sont la source de problématiques et de conceptualisations à tous les niveaux du multimédia.

1.2.1 Construction des données monomédias

Les données monomédias sont la base de tout contenu ou système multimédia, comme les racines étymologiques en témoignent. Chaque média constitue en soi un domaine qui alimente de nombreuses recherches dans des thématiques allant de la perception des signaux jusqu'à l'interprétation des informations, en passant par la représentation et la restitution des données. Nous verrons néanmoins qu'ils ont de nombreux points communs qui offrent la possibilité de gommer tout ou partie de leur particularités, en premier lieu dans le cadre informatique.

Les données monomédias manipulées dans le monde informatique peuvent provenir de deux sources:

– Elles peuvent être numérisées à partir d'un contenu analogique. Ce processus de capture est aujourd'hui grandement facilité par la quantité, la convivialité et la performance des périphériques matériels et des logiciels de capture.

– Les données monomédias peuvent aussi être créées directement à partir d'un outil informatique, logiciel de création numérique ou bien compilateur de spécifications de documents multimédias. Ces données numériques peuvent être générées automatiquement par un programme à partir d'un corpus de connaissances et d'équipements complexes.

Ces contenus numériques bruts doivent ensuite être édités afin d'éliminer les défauts, de retoucher les détails ou de réorganiser la structure du contenu. L'ensemble des tâches mises en jeu, et même au-delà jusqu'à l'application finale de rendu, est organisé le long d'une chaîne de production qui définit le rôle des différents acteurs dans la création d'une œuvre numérique. La plupart des nœuds de la chaîne font l'objet d'applications spécifiques qui occupent un marché très compétitif où la spécialisation est poussée.

Aujourd'hui, ces données sont produites en très grande quantité et diffusées avec une facilité qui oblige à reconsidérer les solutions de stockage envisagées. Plusieurs *DVD* ainsi que des dizaines de *CD* musicaux et de *CD-ROM* informatiques représentant des giga-octets de données numériques nouvelles apparaissent chaque jour. Cette forte offre en contenu est très suivie par les utilisateurs et pousse d'importants travaux de recherche et de développement afin de structurer le contenu et ses méthodes de création et d'innover encore en matière de fonctionnalités.

1.2.2 Système multimédia, application et abstraction

Le multimédia recouvre de nombreuses applications dans des domaines aussi divers que l'enseignement à distance (*e-learning*), l'imagerie médicale, les jeux vidéos ou encore l'art numérique. Pendant longtemps, beaucoup de systèmes n'ont intégré du multimédia que la possibilité de rendre des données temporelles, comme du son ou de la vidéo, en ajoutant simplement un module de lecture de ces données. Mais aujourd'hui ce point de vue a changé et une réelle intégration des diverses données monomédias est possible.

Au fur et à mesure de l'émergence du domaine multimédia, les applications ont évolué en proposant des fonctionnalités et des modèles de données intégrant des données monomédias de manière plus structurée et cohérente. La composition des données a été la première problématique importante du multimédia "réel", s'intéressant à la définition d'objets composites à partir d'objets simples. La possibilité de répartir tout ou partie d'un contenu multimédia sur un réseau est apparue avec le *World Wide Web* qui a engendré au fil des années divers modes de distribution du contenu. La facilité avec laquelle accéder à ce contenu augmente un peu plus chaque jour.

Une des conséquences de ces évolutions est la convergence des applications multimédias qui conduit à une uniformisation des éléments intermédiaires uti-

lisés, que ce soient des langages de description multimédia ou des formats portables de données. L'abstraction des spécificités de chaque média a été accrue grâce à des descriptions communes, tout en laissant chaque application implémenter des fonctionnalités propres et utiliser les spécificités des médias qu'elle supporte.

Le multimédia s'étant étendu à la plupart des domaines créatifs (les différents arts, comme la peinture, le cinéma ou la bande dessinée), son vocabulaire a rapidement intégré la diversité des concepts adressés. Les travaux de classification du domaine se sont heurtés à cette hétérogénéité, obligeant à remettre à jour les critères de comparaison et les concepts utilisés à un rythme rapide. Des efforts sont faits aujourd'hui pour unifier les différents aspects et réunir les acteurs du domaine afin d'en dessiner la carte et convenir des grands chantiers à mettre en place.

1.2.3 Temps et espace

Le multimédia part du postulat que toute information est plongée de manière intrinsèque dans le temps et l'espace. Certaines données monomédias ne dépendent pas du temps ou en dépendent fortement, alors que d'autres n'ont qu'une représentation visuelle ou bien auditive. Mais dans tous les cas, les informations ont une vie dans le temps et l'espace. Premier facteur, le temps a été de tout âge un sujet âpre de discussion. C'est un facteur critique de notre monde car il n'est pas manipulable par l'être humain ou les machines que celui-ci crée. Il en résulte que c'est un aspect primordial des systèmes multimédias. La spécification temporelle du comportement d'un système ou d'un document multimédia repose sur la garantie qu'elle sera exécutée de manière fidèle selon le "sens" convenu pour cet outil.

L'abstraction des systèmes informatiques a poussé à proposer de nombreux langages, outils portables et abstraits, afin de définir des paradigmes de création de *scénarios*, c'est-à-dire des relations temporelles entre éléments du document multimédia. L'industrie multimédia a mis l'accent sur l'efficacité du stockage et du rendu des informations monomédias, accroissement qui s'est d'autant plus fait sentir qu'en parallèle la puissance de calcul des plate-formes informatiques augmentait rapidement. Dans l'optique moderne de la mise en réseau et de la distribution des contenus multimédias, les scénarios se sont étoffés de caractéristiques liées à cette diffusion comme la qualité de service, l'interactivité ou l'adaptation dynamique aux contraintes du réseau. Ces systèmes multimédias ont en retour modifié les protocoles réseaux en y intégrant leurs concepts et techniques.

Second facteur, l'information spatiale est sûrement mieux cernée que celle du temps, car sa maîtrise nous est plus naturelle et a fait l'objet de nombreux travaux étalés sur de longues années. C'est aussi le facteur qui véhicule le plus

d'informations et est le plus sujet à l'interprétation du spectateur humain, une des raisons pour lesquelles il est peu abordé dans le multimédia et reste l'apanage du domaine du média visuel. La plus grande partie des outils multimédias n'offrent qu'une structuration simple de l'information spatiale, concentrant leurs efforts de gestion essentiellement sur le facteur temps. L'auteur de document multimédia ou l'utilisateur de système multimédia doit donc souvent faire passer les informations spatiales dans les données monomédias.

1.3 Cadre et objectif de la thèse

Le travail de cette thèse s'intéresse à la modélisation multimédia, à la frontière entre deux domaines et deux équipes de recherche au sein du laboratoire *LSR-IMAG* de Grenoble: le multimédia pour l'équipe *Drakkar* et la modélisation pour l'équipe *VASCO*. Il s'agit donc d'apporter un travail de formalisation des concepts d'une certaine partie du multimédia.

L'objet précis que nous tentons de cerner est la *présentation multimédia* qui est un document composant un ensemble de données monomédias référencées en spécifiant leurs relations spatio-temporelles. Afin de permettre d'exprimer cette spécification, nous définirons un langage de programmation. Nous ne nous focalisons pas sur les données monomédias, dont le contenu est considéré comme créé en dehors du cadre du langage, et le cœur de notre problématique se situe dans les relations temporelles.

Le langage doit permettre la spécification des présentations multimédias à un haut niveau d'abstraction afin qu'un auteur non spécialiste en langages de programmation puisse organiser ses données monomédias de façon simple et sans se soucier des détails de mise en œuvre. Les abstractions du langage fournissent de plus une base pour construire des outils logiciels rendant cette simplicité de spécification. Le langage doit donc posséder un cadre clairement défini, qui permettra de plus d'expliciter la spécification temporelle en des termes non ambigus et d'implémenter un système d'exécution conforme au modèle du langage.

On retrouve vette problématique dans les langages multimédias *MHEG* et *SMIL*, que nous verrons plus en détail par la suite. Le standard *MHEG* de l'*ISO* permet de spécifier l'application multimédia qui exécute une présentation multimédia. Il offre un point de vue structuré sur les présentations multimédias mais trop programmatique. Dans le domaine du *World Wide Web*, le langage *SMIL* s'inscrit dans le paradigme des langages à base de balises, initié par le langage *HTML*, et ouvre la voie à de nombreux autres langages de description de présentations multimédias dans le cadre du méta-langage *XML*. SMIL, dont la première version est apparue au tout début de cette thèse, est largement supporté par les industries multimédias qui ont saisies l'opportunité que ce standard multimédia leur offrait afin d'accroître leur impact et d'étendre leurs collaborations.

1.4 Composition du mémoire de thèse

Cette introduction a succinctement survolé le domaine du multimédia afin d'en donner une vision d'ensemble en dégageant les notions importantes et le cadre de travail de cette thèse.

Le chapitre 2 dresse un panorama du domaine du multimédia en présentant tout d'abord les divers médias existant. Ensuite, nous traversons les différentes couches des systèmes informatiques du matériel aux *middleware* pour arriver aux applications, dont des exemples pertinents pour notre domaine de travail sont donnés.

Le chapitre 3 adresse plus spécifiquement la notion de présentations multimédias. Le problème de leur composition spatio-temporelle est abordé sous ses diverses formes, de la structuration aux aspects langagiers en passant par la synchronisation et la composition spatiale. Un aperçu des standards et des langages importants est donné en fin de chapitre.

Le chapitre 4 présente notre contribution, le langage *TAO* (pour *Temporal Algebraic Operators*). La syntaxe ainsi que la sémantique spatio-temporelle du langage y sont décrites, suivis d'exemples de programmes.

Le chapitre 5 s'intéresse aux propriétés, à la compilation et à l'exécution des programmes TAO. Il présente l'architecture d'exécution de programmes TAO que nous avons spécifiée.

Le chapitre 6 conclura ce mémoire en dressant un bilan de cette thèse et en présentant quelques perspectives du travail engagé.

Chapitre 2

Le multimédia

Le domaine du multimédia est très vaste, tant dans ses applications que dans ses études théoriques comme en témoignent [23, 141, 100]. Il comprend et étend les problématiques de domaines aussi divers que les réseaux [47], les bases de données [103], l'édition de documents [70], les interface homme-machine [91], la composition musicale [67], les systèmes d'exploitation [27]. Il rejoint même en certains points des domaines hors de l'informatique comme la philosophie [147] ou l'enseignement [123].

Nous tenterons dans ce chapitre de donner une vision aussi synthétique que possible de ce domaine en dressant tout d'abord une présentation générale qui permettra de situer l'historique ainsi que les contextes industriels et de recherche de ce domaine. Nous nous intéresserons ensuite à chaque média, puis déclinerons le spectre des éléments informatiques, du niveau hardware au niveau software. Cette description du domaine se terminera par quelques exemples de produits intéressants aujourd'hui.

2.1 Présentation générale

Le terme *multimédia* a pour étymologie les mots latins *multi* et *media* qui signifient figurativement "plusieurs moyens de communiquer" [105]. Diverses définitions figurées du terme existent, orientées légèrement différemment selon le domaine de travail. Après un bref historique, nous décrirons les contextes industriels et de recherche du multimédia.

2.1.1 Bref historique

L'histoire du terme et du domaine n'est pas simple à dresser car la notion de multimédia a mis du temps à émerger et à se figer. Beaucoup d'auteurs [23, 128] s'accordent à placer le système *Memex* imaginé dans son article par V. Bush

[28] après la seconde guerre mondiale à l'origine des idées qui ont conduit au multimédia.

Une première étape est franchie en 1967 lorsque le *MIT* est créé, organisé en divers laboratoires dont le *Media Lab* qui sera à l'origine de nombreuses découvertes dans le domaine multimédia. Une autre étape est franchie au laboratoire *CERN* à Genève lorsque T. Berners-Lee propose en 1989 le système hypertexte *World Wide Web*, qui a pris une nouvelle dimension au sein de l'Internet. Les évolutions [85, 87] ont continué à s'opérer, comme par exemple l'invention des supports de stockage CD et DVD ou la création des terminaux de télévision numérique.

Le boom de l'informatique au début des années 90 a propulsé ce domaine sur le devant de la scène, ce qui a à son tour accru la pénétration de l'informatique dans de nombreux autres domaines, comme les télécommunication, la télévision ou bien la médecine. Aujourd'hui, les systèmes multimédias tendent à acquérir une certaine maturité et une relative stabilité au sein de l'informatique moderne [80].

Le multimédia est aujourd'hui défini comme les techniques concernant plusieurs données médias, comme le stockage, la récupération, l'intégration, la communication ou encore le rendu [94]. Cette définition générale regroupe un grand nombre de tendances et fournit une vision large, vision que nous tenterons de structurer tout le long de ce chapitre.

2.1.2 Industrie multimédia

Le domaine du multimédia fait partie de ceux qui ont le plus bénéficié des avancées tant logicielles que matérielles en informatique. Les périphériques de rendu se généralisant, les divers médias sont devenus de plus en plus accessibles et des techniques de codage, comme celles des formats MPEG-1 layer 3 (ou mp3) ou MPEG-2 (utilisé pour les DVD et encodé via les codecs DivX), ont rendu efficaces et peu coûteux leur stockage, en parallèle avec l'avènement des CD et DVD, ouvrant la voix à un univers numérique de contenu. La forte demande des utilisateurs de l'informatique pour le multimédia a engendré une offre industrielle grandissante d'outils toujours plus puissants et évolués. Pour se convaincre de la diversité engendrée, le lecteur pourra consulter [5] sur le domaine des applications sur CD-ROM.

La diffusion de ce contenu numérique via l'Internet, médium omniprésent dans l'informatique moderne, a encore accru ce phénomène, tout en apportant des spécificités toutes propres [137]. La créativité a été grandement accrue grâce au multimédia, les utilisations classiques d'ordinateurs personnels (bureautique) faisant place à des processus créatifs originaux (cf. des sites web comme New Venue, http://www.newvenue.com, ou Dfilm, http://www.dfilm.com). Le

désavantage fut que le multimédia devint aussi un argument commercial pour vendre des ordinateurs personnels et qu'il souffrit d'une ouverture trop grande, entraînant en retour une forte demande de sécurité [111]. La sécurisation des échanges a été mise en place à la suite de l'accroissement des piratages et permet de faire respecter la propriété intellectuelle (copyright, droits d'auteurs) [161].

Au jour d'aujourd'hui, l'industrie multimédia recouvre de nombreux domaines de travail, contribuant chacun à créer un produit qui est une brique qui sera utilisée plus loin dans la chaîne de production multimédia. Parmi les principales activités, illustrées en figure 2.1, on trouve:

- la fabrication de périphériques d'entrée-sortie, comme les écrans (*Sony*), les cartes vidéo (*Matrox*) et audio (*Creative Labs*); à noter qu'une définition de l'*ordinateur personnel multimédia* (*MPC*, *Multimedia Personal Computer*) a été proposée par *Microsoft* lors de la sortie de son système d'exploitation *Windows 95* mais n'a reçu que peu de soutien car les constructeurs de matériels ont préféré construire leur propre modèle de structuration des ordinateurs *PC*;
- la création de contenus, qui comporte la numérisation des contenus analogiques (papier pour les bibliothèques, son et vidéo pour des organismes comme l'INA ou les chaînes de télévision) et la création directe de contenu numérique (studio cinématographique et d'art numérique);
- l'organisation et la gestion du contenu, comme le font les fournisseurs d'applications dédiées et de sites Internet [60];
- le service d'accès au contenu (fournisseurs de services online comme *Akamai* http://www.akamai.com), ainsi que les services associés tels que l'autorisation (commerce éléctronique) et l'organisation, par exemple au sein de moteurs de recherche ou de bases de données multimédias [64];
- l'implémentation des systèmes et outils informatiques, éventuellement multimédias, utilisés par les autres activités, dont les grands acteurs industriels sont *Adobe*, *Apple*, *Macromedia*, *Microsoft*, *Real Networks* ou encore *Sun Microsystem*.

Cette séparation des activités multimédias est généralement claire et bénéficiaire à l'ensemble de la communauté, les entreprises se concentrant sur une activité pointue et reposant sur les produits d'autres sociétés pour les services manquants. Néanmoins, cette organisation n'offrant pas un niveau de contrôle optimal, certaines entreprises ont tendance à regrouper le plus grand nombre de tâches en leur sein. C'est le cas par exemple des grandes entreprises du loisir numérique comme les *majors Sony* ou *AOL Time Warner* [89].

Il est aujourd'hui possible de concevoir des œuvres numériques intéressantes en travaillant intégralement dans le monde numérique et en n'utilisant plus aucun service analogique. Par exemple, le film *Shrek* [102] a entièrement été conçu en image de synthèse et le film *Star Wars: Episode II, Attack of the Clones*

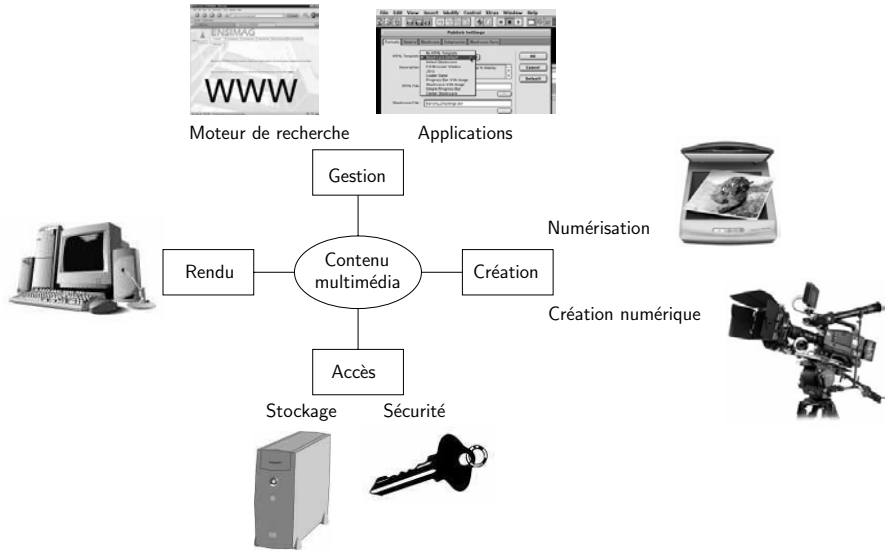


FIG. 2.1 – *Organisation des activités multimédias*

a été réalisé intégralement en technologies numériques. Le néologisme *numédia* (contraction de multimédia numérique) est parfois utilisé pour désigner cette tendance qui s'amplifiera à l'avenir.

Le domaine multimédia s'est tellement diversifié qu'un grand manque d'harmonisation s'est fait sentir, empêchant parfois même certains nouvelles solutions d'émerger. Sur la base de ce constat, le groupe de travail *MPEG* de l'ISO/IEC a mis en place en 2001 une activité nommée *MPEG-21*. Ce standard a pour but de définir une structure ouverte du domaine multimédia qui permettrait un accès transparent et évolué aux diverses ressources existantes à travers un grand nombre de réseaux et de périphériques. Le standard doit spécifier les éléments clé des systèmes et des modèles existant et définir les conditions de leur interopérabilité. Là où des manques existent, l'activité précisera quels standards doivent être créés. Nous présentons ce standard en section 2.5.2.3.

2.1.3 Recherche multimédia

La forte compétitivité et productivité de l'industrie multimédia a mené le domaine à certains conflits et à des impasses que la technique seule ne permettait pas de résoudre efficacement. En parallèle avec cette expansion, de nombreux chercheurs ont tenté d'appliquer les résultats de travaux qui n'étaient pas intrinsèquement lié au domaine multimédia et se sont attaqués aux problèmes des industriels. Les entreprises ont réalisé leur intérêt à se rejoindre sur certains points afin de pouvoir faire avancer leurs développements. Les chercheurs ont aussi joué un rôle important dans ces groupes de travail en apportant leur ex-

pertise, en faisant progresser certains pans de la recherche et en apportant des sources d'innovations aux industriels.

De nombreuses recherches ont pu bénéficier de ce champ d'application qu'est le multimédia pour être valorisées et que soient mises en œuvre des techniques qui restaient auparavant théoriques. Néanmoins, les études dans le multimédia n'ont pas toutes pu être suivies d'applications et beaucoup d'efforts se sont parfois révélés vains [85]. En effet, la grande rapidité avec laquelle les produits industriels ont évolué ont rendu difficile une vision claire du domaine, certains résultats se retrouvant même caducs au moment de leur obtention.

Aujourd'hui, la recherche dans le multimédia a vu son activité se réduire et ses problématiques se recentrer autour de quelques sujets qui ont su trouver un écho applicatif. Nous décrivons ensuite quelques unes de ces problématiques importantes.

2.1.3.1 La structuration de l'approche

L'attitude des premiers systèmes multimédias consistait à proposer toujours plus de nouveautés et de fonctionnalités en ne proposant pas de notions sous-jacentes permettant de comprendre les évolutions. C'est le contraire d'une approche structurée qui définit les notions, ou concepts, sur lesquelles repose l'approche et les relations qui les lient.

La structuration d'une approche permet avant tout de simplifier sa maîtrise par son utilisateur, en lui donnant des repères et des guides afin qu'il se familiarise avec l'approche puis construise une méthode de travail. Des exemples de structure utilisées sont les formats de données, les langages, les composants ou encore les objets.

Les structures permettent d'autre part de situer les approche entre elles. Que cela soit en des termes conceptuels, fonctionnels ou opérationnels, la structure permet de comparer les notions et les relations des différentes approches. Il est ainsi plus aisé de déterminer quelle approche est la plus adaptée à une tâche et comment passer d'une approche à une autre.

Au delà des structures individuelles des approches, on voit aujourd'hui apparaître des formes d'organisations plus générales. Le standard *MPEG-21* dans le multimédia ou bien la plate-forme *Java* dans la programmation en sont deux exemples parmi d'autres.

2.1.3.2 L'intégration des divers médias de manière uniforme

Le multimédia débute à l'intersection des domaines des divers médias, s'étend au delà de leur union et repose avant tout sur leur intégration. L'intégration n'est

pas le simple regroupement des médias entre eux, c'est aussi le fait que des liens importants existent, et sont maintenus, entre eux. Ainsi, les documents HTML assemblent divers médias (texte, image, puis son et vidéo plus tard) mais ne les intègrent que partiellement car ne précisant qu'une partie de leurs arrangements dans l'espace visuel. Cela a donné lieu à deux interprétations, celles de Netscape Navigator et Microsoft Internet Explorer dont les différences subsistent encore aujourd'hui, qui ont été sources de confusion pour les utilisateurs et un obstacle au développement de certains systèmes sur le Web. En fournissant un modèle d'intégration spatiale, puis temporelle par le biais du langage SMIL, la version 4 du langage HTML a rectifié cet écueil.

Cette intégration a été améliorée ces dernières années grâce à l'apparition de lecteurs (ou *players*) dédiés au rendu de chaque média et pouvant être intégré à des systèmes plus larges. Cela a permis de déléguer la tâche de rendu de façon uniforme et de concentrer les efforts de développement sur les relations entre ces activités de rendu.

L'uniformité des approches envisagées, dans le sens où tous les traitements de rendus sont délégués de la même façon au composant player, n'est pas seulement un aspect technique et s'exprime aussi au niveau conceptuel. Ainsi, en considérant tous les médias comme temporels, ils partagent tous un ensemble de propriétés et de comportements identiques et sont capturés par le modèle de la même façon.

2.1.3.3 La gestion des ressources systèmes

Les ressources que les systèmes informatiques sont amenés à utiliser se résument au couple (*temps*, *espace de stockage*).

Le temps comporte les délais de transfert et de traitement des données par les périphériques matériels. Le transfert peut avoir lieu localement (copie des données sur un bus de carte) ou bien depuis ou vers un ordinateur distant par le biais d'un périphérique d'entrée-sortie (typiquement un réseau). Nous verrons plus loin que le facteur temps est un des aspects primordiaux du multimédia et qu'il nécessite une attention toute particulière. L'abaissement drastique des temps de traitement grâce à des ordinateurs toujours plus puissants a permis d'implémenter des systèmes efficaces sans qu'il soit nécessaire d'utiliser des techniques temporelles de bas niveau de synchronisation, comme par exemple des horloges multiples.

L'espace de stockage est la taille qu'il faut allouer aux données sur des mémoires vives (à court terme) ou de masse (à long terme). La taille des données médias varie de façon importante selon le média, jusqu'à atteindre des giga-octets pour la vidéo, ce qui impose de la gérer efficacement afin de ne pas rendre le système inutilisable. Tout comme la ressource temps, l'espace de stockage est devenu

plus accessible, la taille des unités de stockage gravissant de nouveaux sommets chaque mois, et a rendu possible des applications impossibles il y a dix années. On estime aujourd'hui que les plus gros systèmes de stockage de données sont parmi ceux des centres de recherche météorologique et peuvent atteindre 500 Téra-octets [45]. Ceci exige des techniques de gestions spécifiques, comme la compréssion des signaux selon des codages propres à leur média, car les techniques actuelles tel que les systèmes de fichiers sont pris en défaut par de telles capacités.

Notons néanmoins que des plate-formes nouvelles, comme les téléphones mobiles, remettent en cause l'idée que les ressources deviennent de plus en plus abondantes. Il est toujours nécessaire de mettre en œuvre des techniques de bas niveau afin d'assurer dans tous les cas que les systèmes peuvent s'exécuter correctement.

2.1.3.4 L'utilisation du réseau

Les réseaux informatiques sont utilisés d'une part pour transférer les données et d'autre part pour répartir les traitements. Les réseaux peuvent être vus comme une des ressources à gérer, contraignant les temps de transfert en obligeant à gérer de multiples paramètres comme par exemple le débit, le délai ou la qualité de service [100]. Mais leur impact est si important dans les systèmes informatiques qu'ils sont souvent considérées à part.

L'échange d'informations par le réseau a été grandement simplifié par l'avènement de l'Internet, qui n'est que la partie émergée du vaste domaine de recherche dans les réseaux. Les réseaux sont maintenant présents dans la grand majorité des systèmes informatiques et sont un aspect incontournable dans le domaine du multimédia, car ils sont parfois la seule solution à des problèmes comme le stockage des données volumineuse ou leur accès rapide (des réseaux locaux pouvant parfois se révéler plus efficace que des supports de stockage traditionnels). Les codages des informations s'efforcent d'être les plus compacts possible afin de permettre la transmission de tous types de données sur des réseaux à très bas débit (vidéo sur réseaux mobiles à quelques dizaines de kbit/s).

Du point de vue du multimédia applicatif, les réseaux peuvent se ramener à une désignation particulière des entités indiquant le chemin vers les données et le moyen d'y accéder (protocoles, ports et autres paramètres de connexion). Cette seule modification à apporter à un modèle local des applications permet de rendre transparente l'utilisation des réseaux, ce qui est aujourd'hui recoupé sous le terme d'*ubiquitaire*. En retour, le multimédia a influencé le domaine des réseaux, par exemple avec l'introduction de contraintes temporelles [48], et a lancé l'idée de la technique de *streaming* (envoi de données en flux).

La distribution des systèmes informatiques constitue une des autres application du domaine des réseaux. Les systèmes distribués sont un paradigme à part entière qui proposent de nombreux services intégrables à tous les systèmes. Des normes, comme *CORBA* [152] ou *RTP/I* [153], et des modèles, comme le modèle client-serveur, ont permis à ce domaine de se répandre. Ces systèmes adaptent les traitements traditionnels aux conditions particulières des réseaux et en développent de nouveaux selon des modèles appropriés [14, 35]. Ils s'attaquent à un problème difficile dans le cas général car les garanties sur les traitements et les données sont parfois faibles, notamment quand on considère un réseau comme l'Internet.

À l'intersection de ces deux aspects se trouvent divers travaux, dont le travail coopératif, qui tentent de répartir à la fois les données et les traitements entre les utilisateurs et les systèmes coopérant. Ce domaine a été lancé il y a longtemps déjà mais son impact est encore faible, à la fois dans l'informatique générale et dans le multimédia. Il promet d'apporter de nouveaux paradigmes de programmation et de manipulation des données, innovations qui pourraient bien ouvrir la voie à de nombreux autres développement dans le cadre du multimédia réparti.

2.2 Les médias

Nous présentons ici les divers médias usuels dans le domaine informatique. Après une définition générale nous les décrivons les uns après les autres et terminons en indiquant différents critères selon lesquels les considérer.

2.2.1 Définition d'un média

La notion de *medium* signifie originellement en latin "milieu, centre" mais aussi "lieu accessible à tous, à la disposition de tous, exposé aux regards de tous" [58]. Le mot prend plus tard le sens d'intermédiaire et de moyen de communication de la pensée [105]. Il transite ensuite par le mot anglophone *mass media* qui désigne aujourd'hui l'ensemble des moyens d'information par voie papier, radiophonique et télévisée. Il est finalement raccourci en *média*.

Cette dernière acception est généralisée en informatique en considérant un média comme un moyen de transmettre, stocker ou présenter des informations [94]. Au sein de ces informations, peu sont compréhensibles par un être humain à cause de leur trop bas niveau de granularité (bit, trame réseau, page physique). Le multimédia s'intéresse aux médias véhiculant des informations conceptuellement accessibles par un utilisateur humain, c'est-à-dire à travers ses cinq sens d'observation. Les cinq médias basiques sont visuel, auditif, tactile, olfactif et gustatif. Ces médias basiques sont ensuite étendus à divers types d'informations qui possèdent des caractéristiques communes. La classification classique en multimédia regroupe les types de média suivant: texte, image, son, animation et

vidéo. Nous ajoutons à ceux-là le média programme qui englobe les données qui précisent des traitements d'autres données médias. Ne sont pas inclus dans cette classification les médias tactile, olfactif et gustatif, car ils sont de rare utilisation dans le monde informatique, mais nous glisserons un mot sur chacun d'eux dans les sections suivantes.

Les données se référant à chaque média sont appelées *monomédia*. Chaque média est l'objet d'études particulières visant à faire avancer la compréhension et la maîtrise du média et de ses données. La représentation des données monomédias dans un format informatique compréhensible par une machine est appelé *codage*. Des efforts considérables sont fait afin de définir des codages adéquats pour chaque média afin de faciliter certaines opérations [94]. En effet, les données sont inséparables de leurs traitements au sein des systèmes informatiques et des choix doivent être faits quant à ceux qui seront privilégiés.

2.2.2 Les principaux médias

2.2.2.1 Texte

C'est le média artificiel le plus ancien au monde et celui qui a fait l'objet des premiers développements en informatique. Ses informations sont conceptuellement bien intégrées dans nos modèles courants, ce qui les rend simples à modéliser au sein des systèmes informatiques.

Un texte basique est une séquence de caractères d'un alphabet. L'encodage des caractères a été un problème pendant les débuts de l'informatique personnelle, mais ne pose plus d'insurmontables problèmes aujourd'hui grâce à des standards répandus, comme l'ASCII, l'ISO-8859 et l'UNICODE. Un texte est traditionnellement découpé en mots, phrases, paragraphes, sections et chapitres au sein d'une œuvre littéraire comme un article ou un livre.

Un texte enrichi ajoute des informations de mise en forme du texte en associant à chaque caractère des paramètres de présentation, comme la police de caractère (groupe de caractères de caractéristiques de formes identiques), la casse, la graisse, l'italique, le soulignement ou bien encore la couleur. Lorsque le texte est mis en page, chaque caractère possède un alignement relatif aux caractéristiques de la page (tailles, marges, etc.).

Le texte est un flot de caractères qui possède une direction intrinsèque. Les exercices de style comme les *palindromes*, les calligrammes de G. Apollinaire ou les *ASCII art* (textes qui inspirent des dessins) ne modifient en rien cette propriété mais jouent sur des effets de forme. Cette linéarité horizontale de sens gauche à droite en occident est très différente des écritures asiatiques, en particulier pour les kanji ou idéogrammes dont l'organisation est fortement spatiale. La linéarité du texte a été d'une certaine manière brisée par l'introduction de

liens hypertextes, rendu célèbre par le web et le langage HTML, permettant au lecteur de passer d'un morceau du texte à un autre en activant le lien.

Bien que moins dense que les autres médias, le texte reste encore aujourd'hui l'élément privilégié pour exprimer le sens et expliciter la signification des autres médias grâce à son intuition et sa simplicité [57]. Sa linéarité reste toutefois une limitation qui rend nécessaire l'usage des autres médias pour créer des contenus riches et denses. La figure 2.2 illustre les différents types de textes.

texte basique	texte enrichi	calligramme	idéogramme
Texte	TeXte		絵

FIG. 2.2 – *Les catégories de textes*

2.2.2.2 Images

Les images sont des entités bi-dimensionnelles (ou *2D*) composées de points, ou pixels, et d'une taille, en nombre de pixels. Chaque pixel possède une couleur et éventuellement une transparence. Les couleurs possèdent des codages propres, comme le triplet RVB de proportions de rouge, vert et bleu ou encore un code parmi une palette de taille entre deux (bicolore codé sur un bit) et seize millions (*true color* codé sur 24 bits). Cette définition fournit la représentation bas niveau du médium par le système informatique. Un être humain percevra par contre des courbes et des formes, groupes de pixels de couleurs et de positions proches, seules interprétables par lui.

Le domaine de la génération d'image, ou *infographie*, s'est grandement développé et a produit des outils puissants comme *Adobe Photoshop*. De nombreux formats binaire d'image ont été standardisés, comme *JPEG*, *GIF* ou *PNG*. Ils permettent de stocker des informations sur les pixels de l'image et les échanger entre systèmes informatiques, chacun facilitant certaines opérations comme l'affichage progressif ou bien la compression. Suivant la mouvance *XML*, des formats textuels d'image sont apparus, comme le standard *SVG* (*Scalable Vector Graphics*). Les formats d'image continuent d'évoluer en intégrant des techniques issues d'autres domaines. Par exemple les *smart graphics* [53] peuvent découvrir et s'adapter à des conditions particulières, comme les tâches pour lesquelles elles seront utilisées ou les paramètres réseaux de transmission des images.

La représentation de bas niveau d'une image ne peut avoir que deux dimensions, on parlera d'image simple, alors que celle de plus haut niveau peut en avoir plus. Lorsqu'une image est une superposition de plusieurs images simples, en jouant sur la transparence et l'alignement pour préciser leurs positions relatives, l'image est dite en dimension 2.5. Cet approche est souvent utilisée afin de définir des assemblages de divers couches (*layers*) empilées les unes au-dessus des autres, en laissant à la charge d'un système le calcul de l'image finale qui en résulte. Les images représentent la plupart du temps des informations en trois dimensions, ou *3D*, que ce soient des images réelles (photographies) ou bien de synthèse, c'est-à-dire générées par ordinateur. Dans cette optique 3D, les images sont plutôt conçues comme des assemblages complexes de formes correspondant à des objets réels. Le domaine des images de synthèse permet de pousser très loin le détail des images créées, jusqu'à une qualité photoréaliste extrêmement proche de la réalité, grâce à des outils de modélisation physique. Des paramètres comme l'illumination ou les forces de contact peuvent être spécifiées dans la définition d'une scène dont le rendu sera calculé par un logiciel comme le *raytracer POV* (*Persistance Of Vision*).

Les *anaglyphes* sont des images formées par l'intercalage de deux images légèrement différentes chacune passée à travers un filtre coloré. Ceci permet à un utilisateur équipé de lunettes à filtres colorés de rendre chaque image sur un des yeux afin de recréer des informations de profondeur dans l'image. Cette technique repose sur l'effet stéréoscopique dont se servent nos yeux pour percevoir les informations en 3D. Certains périphériques permettent de rendre directement les deux images sur chaque œil, technique utilisée avec des lunettes de réalité virtuelle dans les cinémas *IMAX 3D* par exemple. C'est une variante de cet effet qui est utilisé dans les *stéréogrammes*, images en cachant une autre obtenue en forçant la focalisation de l'œil. Ces diverses techniques sont un des aspects des systèmes de réalité virtuelle qui permettent une immersion plus poussée de l'utilisateur dans le monde représenté par l'image en mettant en jeu les techniques visuelles utilisées par nos yeux. Les parcs d'attraction usent de périphériques évolués qui permettent de donner plus de profondeur à une simple image 2D en jouant sur la taille de l'image et sur sa disposition par rapport au spectateur (cylindre ou demi-sphère). Ces images sont souvent déformées afin de pouvoir offrir une projection correcte tenant compte de la forme de l'écran.

Des nouveaux périphériques matériels encore en développement, comme les périphériques holographiques [84, 56], vont permettre de rendre les images, ou ici *hologrammes*, en 3D et ainsi utiliser les modélisation 3D directement sans nécessiter d'artifices. La figure 2.3 illustre un exemple de périphérique holographique. Bien que nous plaçions le média holographique dans le média image, il est parfois rapproché du média tactile à travers l'idée de sensation (cf. plus bas).

FIG. 2.3 – *Le périphérique holographique Perspecta®*

2.2.2.3 Odeur et Goût

Bien que considérés encore comme exotiques, ces deux médias font l'objet d'études et de développement en informatique [32, 71]. La raison du faible intérêt que la communauté multimédia leur porte est que leur influence est faible dans la plupart des interactions sensitives [112]. Néanmoins leur importance est primordiale dans certains domaines, comme la chimie, l'œnologie ou la cuisine, la faible pénétration du multimédia dans ces domaines expliquant aussi le retard des recherches.

Les deux domaines de recherche liés à l'odeur et au goût reposent encore sur des assesseurs humains (les "nez" et les "goûteurs") en guise de capteurs fournissant les données olfactives et gustatives. Le plupart des descriptions de l'odeur et du goût font intervenir des représentations chimiques et la morphologie humaine, ce qui ne facilite pas leur étude et leur intégration dans des modèles multimédias, qui sont eux plutôt tournés vers des notions abstraites. Une décomposition possible du goût [16] est celle en quatre saveurs canoniques: sucrée, acide, salée et amère. Des saveurs particulières sont parfois ajoutées, comme les saveurs métallique ou basique. Cette décomposition est moins claire dans le domaine des odeurs, le nombre de types de récepteurs de l'appareil olfactif humain n'étant pas encore bien fixé autour de la valeur 1000 [32].

Des périphériques de rendu ont commencé à être commercialisé il y a quelques années déjà mais ne s'adressent pour l'instant qu'aux chercheurs travaillant sur ces médias. Des études sont encore à mener afin de permettre de mieux cerner quels sont les périphériques et modèles dont l'informatique a besoin afin de développer des applications intéressantes [71]. L'avenir accordera probablement

plus d'importance à ces médias qui sont indispensables à un rendu multisensoriel et réaliste des informations multimédias. C'est ce que nous pouvons entrapercevoir dans les spectacles d'immersion du spectateur telles la diffusion de films en *odorama* (Cité des Sciences, la Vilette, Paris).

2.2.2.4 Toucher

Le toucher est un sens marginal au sein du multimédia, car les périphériques associées n'ont été développé que pour des communautés particulières, comme celles des joueurs sur ordinateur [34], des utilisateurs de systèmes critiques automatisés ou bien de personnes handicapées. Tout comme les deux médias précédent, son rôle dans la perception réaliste est déterminant [119, 81].

Le toucher occupe une place intermédiaire entre les deux groupes (odeur, goût), peu répandus, et (son, vidéo), médias principaux, du point de vue de l'usage. Il est aussi désigné par l'adjectif *haptique* qui est rattaché à la notion de sensation que ce média peut générer en forte intensité. La tendance actuelle est de parler de *retour de force* (*force feedback*). Le toucher est une sensation obtenue à partir de capteurs de contact de la peau et s'oppose à la modalité kinesthésique qui est basée sur la pression exercée sur la peau et est rattachée aux notions de force et de mouvement. Hormis les modèles biologiques, aucune description du toucher n'existe.

Liard et Beghdadi [97] utilisent comme périphérique tactile une plage tactile vibrante et joignent les informations fournies à des informations sonores, afin de rendre à des utilisateurs non-voyants le contenu d'un écran. Le coût de ces outils est relativement important, en partie parce que les utilisateurs concernés sont peu nombreux. Les investissements ont surtout lieu autour des périphériques à retour de force puisqu'il sont prisés du grand public via les *manettes de jeu* (*joysticks*) utilisés avec des jeux vidéo La figure 2.4 donne une illustration d'une manette de jeu à retour d'effort, ainsi que divers autres périphériques tactiles.

2.2.2.5 Son

Le son est l'un des deux principaux médias utilisés aujourd'hui, avec la vidéo. C'est un média temporel dont l'information est représenté par un signal périodique et continu. C'est ce signal qui est émis des périphériques comme les haut-parleurs et perçu par l'oreille de l'utilisateur humain.

Les systèmes informatiques utilisent un grand nombre de représentation du son [67] que l'on peut diviser en deux familles:

- les représentations discrètes
  ces représentations assimilent le son à une séquence de notes, chaque note étant définie par divers paramètres (instrument, volume, pression sur une membrane, etc.); une telle représentation peut être fournie par saisie des notes tapées sur un instrument réel, comme cela est fait pour le format

manette de jeu
à retour d'effort

barette braille

gant CyberTouch©

FIG. 2.4 – *Exemple de périphériques tactiles*

MIDI (Musical Instrument Digital Interface); le système de rendu calcule ensuite les signaux continus à produire à partir des notes et des techniques mises en jeu (rendu simple ou combinaison des sons élémentaires d'une *wavetable*, ou table d'enregistrements fixes de sons, ou encore interpolation linéaire); la séquence correspond à la notion classique de partition en musique; ces représentations sont compactes mais produisent des sons de richesse limitée (parfois reconnaissables);

– les représentations continues
ces représentations reposent sur la nature ondulatoire des signaux auditifs et fournissent des fonctions paramétrées afin de définir de tels signaux; les signaux élémentaires sont des sinusoïdes dont les paramètres sont la fréquence et l'amplitude; la fréquence d'échantillonage d'un signal indique sa qualité; elle est de 8 kHz pour la qualité téléphonique, 22 kHz pour la qualité radiophonique, 44.1 kHz pour la qualité CD, 48 kHz pour la qualité professionnelle; les signaux réels sont des additions de sinusoïdes et servent de base à des transformations complexes, tel que l'application de filtres numériques, la modulation de fréquence ou encore des effets de champ sonore (*sound field effect*) construits à partir de réverbérations [79]; ces transformations permettent de donner des effets réalistes afin de rendre le son plus convaincant.

Le son étant une donnée temporelle qui peut évoluer vite, il requiert une grande quantité d'informations. Les formats sonores fournissent des méthodes de compression qui sont générales pour les représentations continues, qui offrent déjà un gain par rapport aux signaux physiques, ou spécifiques dans le cas discret, comme le modèle psychoacoustique du format *MPEG-1 Layer 3* ou aussi *mp3*. Leurs implémentations définissent des algorithmes de compression et sont des composants logiciels appelés *codecs*. D'autre part, le codage relatif des informations indique les valeurs en terme d'écart par rapport aux précédentes valeurs

et permet, lorsqu'il est possible, un gain additionnel. Les codages facilitent plus ou moins certains traitements des informations, parmi lesquels la décompression ou l'accès non-séquentiel.

De plus, le son peut convoyer une information spatiale correspondant à la position de sa source, éventuellement repoussée à l'infini pour des sons ambiants qui ne semblent provenir de nulle part. L'espace étant représenté par des couples d'information, le son spatialisé ne peut exister qu'à partir de deux signaux véhiculés dans ce que l'on appelle un canal. Pendant longtemps, les ordinateurs ne possédèrent qu'un seul canal, ce qui obligeait à gérer son allocation afin d'éviter les accès concurrents [69]. Lorsque les premiers périphériques sonores multicanaux furent développés, cette gestion devint amplement simplifié et la spatialisation du son fut possible.

La polyphonie, ou *son surround*, utilise plusieurs sons (et donc émetteurs) afin de simuler une source située dans un certain domaine de l'espace en jouant sur de légères différences entre chaque son dans son intensité et la phase de son signal. Pendant longtemps, seuls deux sons ont été utilisés dans le cadre de la stéréophonie, la source se positionnant sur une ligne entre les deux émetteurs de ces sons .

Les créateurs de cartes sons ont ajouté dans les dernières années des effets complexes de spatialisation du son permettant, même avec seulement deux émetteurs, de donner l'impression qu'un son provient d'un point dans l'espace autour de l'utilisateur.

De nombreux formats de sons polyphoniques ont été développés, faisant varier le nombre de canaux, leurs utilisations et leurs codages, parmi lesquels le *EAX* de *Creative Labs*, le *A3D* de *Aureal*, le *Dolby Digital AC3* ou encore le *DTS* (*Digital Theatre Sound*). C'est la raison pour laquelle la spatialisation du son reste encore cantonné dans les jeux vidéo et la cinématographie numérique et peu de modèles multimédias la prennent en compte. La figure 2.5 illustre un équipement *Home Theatre* à septs voix.

Quelques travaux intéressants réussissent à tirer partie de cette spatialisation afin d'implémenter de nouvelles fonctionnalités. Par exemple, Goose et Möller [62] créent un navigateur auditif interactif qui utilise une position 2D du son sur un cylindre à distance fixe de l'utilisateur. La structure d'un document visuel est ainsi rendue à un utilisateur non-voyant en jouant sur la position de la source et des augmentations et diminutions du volume associées à des événements particuliers. De même la souris acoustique et tactile [97] permet à un utilisateur non-voyant de percevoir le contenu d'un écran. Un périphérique tactile est ici augmenté par un son spatialisé dont la position situe le pointeur de souris à l'écran et la hauteur sonore aide à situer verticalement le pointeur.

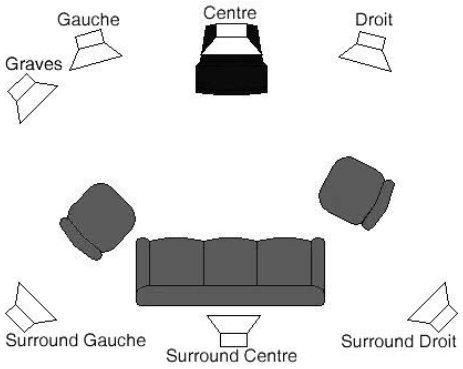


FIG. 2.5 – *Équipements Home Theatre 6+1*

L'utilisation de la spatialisation se révèle complexe mais utile, notamment afin d'augmenter d'autres médias dans un contexte multimédia. Tout comme l'image peut posseder plusieurs dimensions, le son peut constituer une dimension supplémentaire dans certains contextes, comme on le voit nettement dans certains jeux vidéo.

2.2.2.6 Animation et Vidéo

Une animation est constituée d'images, ou *trames*, présentées en séquence dans le temps. Les premières animations sont apparues en superposant de simples images et en les animant les unes après les autres. C'est cette approche qui a été initialement prise en informatique en mettant côte à côte des images, aussi appelé format *brut* et utilisé en sortie des périphériques de capture vidéo. Par contre, les animations issues du monde analogique sont constituées de trames appelées *fields* et qui sont obtenues en ne gardant qu'une ligne sur deux de l'image, ce qui a pour effet de doubler la fréquence de l'animation. L'image est ensuite reconstituée grâce à l'effet de rémanence des tubes cathodiques. La numérisation de telles animations nécessite de désentrelacer le signal, c'est-à-dire reconstruire les images complètes.

La quantité de donnée brutes d'une animation est bien trop grande puisqu'il faut de 10 à 100 Go pour stocker une heure en qualité supérieure. Elle nécessite de plus des techniques de compression adéquates pour être efficace, les techniques génériques de compression n'obtenant pas de bons résultats. Tout comme pour le son, les algorithmes de compression sont encaspulés dans des *codecs*, comme les codecs Sorenson ou encore *Divx* pour les algorithmes définis dans la norme *MPEG-2*. Tout comme pour le son, le codage différentiel des informations permet un gain supplémentaire mais la notion d'écart est différente car l'animation est une information à deux dimensions. Par exemple, dans le format MPEG, divers types de trames sont codées comme une combinaison

d'autres trames.

Les formats modernes comme *MPEG-4* (cf. section 2.5.2.3) élèvent le niveau d'abstraction de la représentation en se basant sur des notions comme les formes ou le mouvement [88], ce qui permet d'atteindre des forts taux de compression mais requiert des techniques d'encodage complexes. Les techniques d'animation ont aussi bénéficié des progrès de l'infographie jusqu'au point de rendre possible la création de films d'animation d'une qualité proche de la réalité, comme *Final Fantasy*). L'animation est aussi une composante importante des interfaces graphiques de logiciels, activité qui s'immisce dans le monde réel grâce à la réalité augmentée [126]. Comme on peut le voir actuellement dans de nombreux logiciels, les animations se répandent de façon importante, par le biais de boutons animés par exemple.

Le média vidéo

Le média vidéo est probablement le plus célèbre aujourd'hui dans le monde du multimédia et celui qui, pour certains, justifie de s'y investir. Cette importance provient aussi de celle de médias comme la télévision ou le cinéma qui fournissent un point d'entrée simple dans le monde du multimédia dans notre société actuelle.

Ce média n'en reste pas moins un parmi les autres et repose sur des notions et des traitements quasi-exclusivement issus des médias animation et son, qu'il unit en une seule entité. Bien que n'étant pas atomique, car composé à partir de deux autres médias, la vidéo est un média classique que la plus grande partie des systèmes multimédias traitent.

La représentation basique d'une vidéo comme une animation et un son joués indépendamment et en parallèle est relativement rare aujourd'hui. La vidéo requiert en effet une synchronisation fine qui oblige à regrouper ensemble les échantillons sonores et les trames d'animation qui se correspondent. On appelle aussi cette synchronisation *couplage* ou *lip-synch*, en référence à la synchronisation des lèvres avec les paroles leur correspondant.

Les formats vidéos, comme *Windows Avi* (*Audio Video Interleave*) ou *Apple Quicktime mov*, sont la plupart du temps des conteneurs dans lesquels sont indiqués les codecs audio et vidéo des données média suivi de la séquence des regroupements d'informations audio et vidéo compressées.

La technique de *streaming*, ou diffusion en flux, permet de diffuser les vidéo sur les réseaux par paquets d'informations. Elles nécessite des formats de codage adaptatifs qui permettent de s'adapter aux variations de paramètres des réseaux. Les applications qui utilisent cette technique de communication, comme celle de vidéo à la demande ou de vidéo-conférence, sont prisées et sont toujours

source d'intérêt pour les grandes sociétés, par exemple la chaîne de télévision britannique *BBC* [49].

Les techniques de codage de l'animation par formes ou objets ont ouvert la voie à des augmentations de la vidéo, comme le système de la société *Arts Video Interactive* (http://www.artsvideo.com) a pu le prouver. Ce format permet de définir des hyperliens entre objets, le clic de l'utilisateur sur l'un d'eux lançant l'affichage de la vidéo à l'endroit où se trouve le second objet. Les hypervidéos sont un type d'objet multimédia récent, à l'intersection entre le multimédia passif et l'hypertexte comme on peut en faire l'expérience sur le web [63]. D'autres développements et études sont en cours afin de faire évoluer les vidéos, notamment dans le cadre des portails vidéo [49].

2.2.2.7 Programme

Ce média n'est pas encore très conventionnel en tant que média bien qu'une grande partie de l'activité informatique consiste à créer des programmes et qu'aujourd'hui des programmes reçus comme données permettent de modifier d'autres programmes. Bien qu'imperceptible aux sens humains, ce média convoie de l'information entre systèmes informatiques, qui sont en quelque sorte les entités les percevant.

L'idée n'est pas nouvelle [86], mais sa réalisation et son intégration au sein d'autres systèmes n'est pas encore aboutie. Le standard multimédia *MHEG* [150] fut un des premiers à lancer l'idée avec la classe abstraite `Program`. Cette classe permet d'indiquer les programmes externes au système d'exécution MHEG, généralement des morceaux de codes procéduraux qui exécutent des actions spécifiques. Les classes concrètes de programme de MHEG sont:
`ResidentProgram` pour les programmes à exécuter localement (sur l'ordinateur où le système d'exécution MHEG fonctionne), `RemoteProgram` pour un appel distant de programme et `InterchangedProgram` lorsque des programmes sont transférés.

Ce média se trouve à la frontière entre le paradigme du document, où l'on indique des informations passives traitées par un système, et celui de l'application, où l'on précise à la fois les données et leurs traitements. Les programmes sont des données *actives*, dans le sens où elles ne seront résolues que par leur exécution. Par exemple, le résultat d'une requête sur une base de données ne peut être connu qu'à l'issue de son exécution par le système de gestion de la base de données [4], la résolution de la requête fournissant l'objet multimédia que la requête définit. Cette notion n'est pas à confondre avec celle de *dynamicité temporelle* qui indique le fait qu'une entité possède des instants de début et de fin ou une durée qui ne seront connues que lors de l'exécution, bien que ce que représentent ces données soit connu statiquement avant l'exécution.

On retrouve ce concept de programme comme donnée monomédia dans le langage *Java* de *Sun Microsystems* sous la forme de *sérialisation* du code. Cette fonctionnalité des objets Java permet de les stocker sous une forme particulière qui permettra leur transfert vers un autre ordinateur via le réseau ou bien leur récupération lors d'une exécution ultérieure du programme.

Les *objets actifs*, utilisés par exemple dans le système *PREMO* [75], définissent un paradigme où il est possible d'indiquer des données programmes.

Aujourd'hui, l'exemple typique de donnée programme sont les composants logiciels appelés *players* qui permettent de présenter des formats spécifiques de données, l'association entre *player* et format étant généralement faite par le système d'exploitation. On les retrouve dans le format de données multimédias *Hotmedia* d'*IBM* [90] ou bien des *plug-ins* des navigateurs Internet.

L'idée de déclarer les programmes comme des données permet d'abstraire certains traitements en les déclarant génériques dans une donnée programme qui sera instanciée à l'exécution. Au delà même réside l'idée des programmes évolutifs, modifiés par des données programmes qu'ils reçoivent et vont les modifier. Cette approche ouvre de nouvelles perspectives dans de nombreux domaines, comme dans l'infographie où les *smart graphics* [53] permettent de créer des image s'adaptant dynamiquement aux systèmes et aux utilisateurs auxquels ils seraient présentés. Elle nécessite encore des efforts intense, tant au niveau pratique vis-à-vis de la portabilité des divers systèmes (cf. section 2.4) qu'au niveau théorique à travers une interface de haut niveau capable de décrire la plus grande partie des programmes et des problèmes liés au code mobile [31].

2.2.3 Typologie des médias

La diversité des médias et de leurs caractéristiques nécessite d'en dresser une carte qui permet de saisir leur organisation générale. La classification se fait au niveau des informations même ou bien de leurs représentations. De nombreux critères peuvent être pris en compte afin de les classer, du plus technique au niveau des bits au plus conceptuels au niveau de leur interprétation par l'utilisateur. Nous donnons des critères parmi les plus utilisés dans le domaine du multimédia et qui sont des critères naturels, et non techniques, de classification des médias. La typologie en résultant est suffisante pour décrire la plupart des classes de médias utilisées aujourd'hui.

Le premier critère est la part qu'a le média dans l'interaction avec l'utilisateur, son *importance*. Cette part est en fait proportionnelle à la quantité d'information véhiculée par le média. Ce critère revient donc à classer les médias selon leur taille et ce, indépendamment d'un format de représentation particulier, ce qui donne l'ordre approché suivant, du plus important au moins important: visuel,

auditif, tactile, odorat et goût. [112] cite la proportion de 70-80 % pour le média visuel.

Une autre classification des médias est leur caractéristique temporelle ou non. Les données d'un média *non-temporel* ne dépendent pas du temps, comme pour le texte ou les images, et sont parfois appelées discrètes ou statiques. Celles d'un média *temporel* en dépend intrinsèquement, c'est-à-dire que ses données ont une durée comme l'audio ou l'animation, et sont parfois nommées continues. Une autre définition [114] des termes continus et discrets sépare respectivement les médias en analogiques et numériques. De plus, la représentation des médias temporels peut être discrète, si elle utilise des notations entières, ou continue, si elle utilise des fonctions continues du temps. Pour éviter toute confusion, nous parlerons de médias non-temporels et temporels plutôt que discrets et continus.

Une autre utilisation du terme de numérique est faite en opposition au terme de naturel et fait référence au moyen avec lequel les données médias ont été obtenues. Lorsque ces données proviennent d'une numérisation d'un objet réel, on parle de média naturel. Lorsqu'elles sont générées par des programmes informatiques, éventuellement en retouchant des données de médias naturels, les médias sont dit numériques ou encore synthétiques.

Une autre caractéristiques des données média est leur *linéarité* [9]. Les données d'un média linéaire ne peuvent être produites que depuis son début naturel et en suivant son déroulement normal. Cela signifie qu'il est impossible d'accéder à n'importe quelle partie des données d'un média linéaire sans attendre que cette partie ne soit naturellement présentée (ou bien de redémarrer la présentation de ces données depuis le début si la partie se situe dans le passé). Les données du média programme sont linéaires, alors que la plupart des autres données médias sont non-linéaires, trivialement pour les médias non-temporels. La non-linéarité requiert parfois des traitements complexes afin d'accéder à la portion désirée du média, comme par exemple pour une vidéo MPEG dont les trames peuvent être interdépendantes et nécessiter de recalculer des trames antérieures à celle voulue.

La notion de *réversibilité* est liée à celle de linéarité et indique s'il est possible de présenter les données médias dans leur ordre temporel inverse. C'est une caractéristique peu examinée et très peu de données de médias temporels possèdent aujourd'hui cette propriété.

2.2.3.1 Transcodage

La diversification des systèmes informatiques et l'extension de leurs fonctionnalités a donné naissance à une nouvelle préoccupation qui est l'adaptation. Celle-ci permet d'indiquer quelle représentation ou format de donnée choisir

pour un média donné dans une situation particulière, parmi un ensemble initialement défini. Par exemple, une vidéo peut être de choisie parmi plusieurs qualités selon le débit disponible sur le réseau pour sa diffusion, comme le langage SMIL permet déjà de le spécifier ou une image peut être codée en multirésolution [21] afin de permettre son rendu dans divers tailles.

L'adaptation est facilitée par des techniques de transcodage qui donnent le moyen de transformer les médias entre eux. Ces techniques sont coûteuses en temps de traitement et n'impliquent pas toujours des pertes d'informations, comme pour les codages fractals, mais permettent de rendre des données médias dans un plus grand nombre de conditions et ainsi économiser le stockage de plusieurs données.

Ces considérations ont été étendues à l'adaptation inter-média [19] où des informations d'un média peuvent remplacer celles d'un autre média. Cette technique pose de nouveaux problèmes liés à la spécification du contenu des médias, de façon à pouvoir assurer qu'une adaptation ne déforme pas le sens des informations. Le système d'exploitation *BeOS* [11] propose cette technique au sein d'un de ses services, celui de traduction des formats qui permet de convertir les données médias d'un format à un autre afin de permettre leur utilisation par un grand nombre d'applications.

Suivant l'élévation du niveau d'abstraction de la plupart des systèmes informatiques, de nombreux travaux s'intéressent à des descriptions de haut niveau des informations médias, comme les standards *RDF* (*Resource Description Framework*) ou bien *MPEG-7* [104]. De tels travaux tendent à remettre en cause la séparation traditionnelle entre les divers médias au profit de plus de "signification" dans les descriptions et nécessite encore des travaux de fond qui ne seront pas terminés avant plusieurs années. Les techniques qui y sont liées permettront d'automatiser les adaptations et ainsi rendre transparents ces traitements à l'utilisateur, comme le préconise le standard MPEG-21.

2.3 Évolution des matériels dédiés

Depuis longtemps déjà, l'informatique évolue au rythme des progrès des composants matériels des ordinateurs [27]. Le multimédia n'échappe pas à cette règle et dépend plus fortement encore que les autres domaines de l'informatique de ces progrès. Cette dépendance provient essentiellement de la place des périphériques d'entrée-sortie dans les systèmes multimédia, puisqu'on les retrouve aux deux bouts de la chaîne de production. Nous ne parlerons que du matériel dans le domaine grand public car c'est dans ce secteur que les évolutions ont été les plus importantes et pour l'essentiel suivies dans les autres secteurs.

Les données monomédias temporelles étant très gourmandes en ressources, l'amélioration des autres périphériques des ordinateurs (processeur, mémoires)

et la modification de l'organisation matérielle des ordinateurs ont aussi contribué à l'expansion du multimédia, comme nous allons le voir.

2.3.1 Processeurs et circuits intégrés

La capacité des processeurs continue d'augmenter jour après jour, selon la fameuse *loi de Moore*: "Le nombre de transistors intégrés dans une puce doublera tous les deux ans". Les ordinateurs personnels possèdent aujourd'hui des microprocesseurs cadencés à plusieurs GHz. Les ordinateurs multiprocesseurs commencent à se diversifier et les architectures à grappes (*cluster*) d'ordinateurs sont beaucoup utilisées pour certaines applications intensives comme le rendu photoréaliste ou la cinématographie numérique [102]. Ces accélérations entraînent des gains globaux sur tous les traitements effectués par l'ordinateur. Ces gains étant relativement modestes pour les applications multimédias, qui requièrent des manipulations rapides de données volumineuses, chaque constructeur de microprocesseurs a développé un jeu d'instructions spécifiques au multimédia (*MMX* pour *Intel*, *3DNow!* pour *AMD*). Ces jeux d'instructions ajoutent aux instructions traditionnelles la mise en parallèle (*pipelining*) d'instructions traitant des groupes de données plus important.

Ces technologies ne suffisent parfois pas pour obtenir des calculs numériques rapides, comme des transformées de Fourier par exemple. Des circuits dédiées comme les *DSP* (*Digital Signal Processor*) sont alors utilisés à la place des processeurs généraux. Les architectures matérielles tentent d'équilibrer ces divers composants entre la diversité offerte par les instructions généralisées et la rapiditité des instructions spécialisées. Par exemple la plate-forme *OMAP* (*Open Multimedia Applications Platform*) [37] combine un circuit général à un DSP, ce qui permet d'étendre le champ des applications multimédia développables en fournissant des procédures de rendu qui peuvent s'exécuter rapidement sur les plate-formes restreintes que sont les téléphones mobiles. La figure 2.6 illustre l'architecture de la plate-forme OMAP.

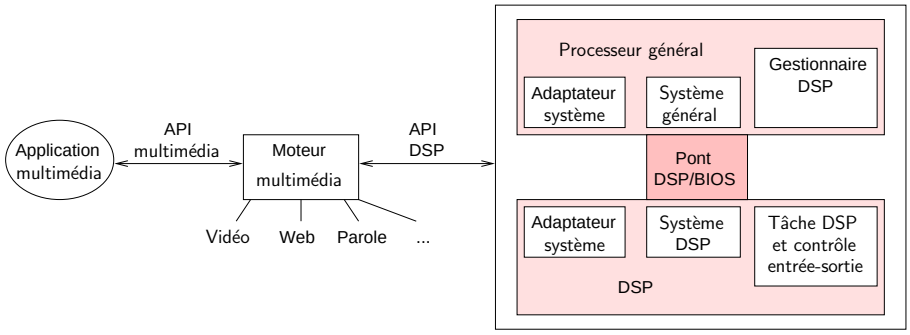


FIG. 2.6 – *Plate-forme OMAP*^{TM}

2.3.2 Périphériques de stockage

Les données multimédias sont volumineuses comme nous l'avons vu précédemment. Les divers médias étant plus ou moins demandeurs en espace de stockage, le fait d'augmenter cet espace ne fait pas qu'augmenter la quantité de données médias qu'on peut y mettre mais aussi le type des données médias. En effet, la plus petite des vidéos compressées fait plusieurs dizaines de kilo-octets, ce qui suffit à stocker un article d'une dizaine de pages ou une image de qualité moyenne sur le web. Les moyens de stocker des informations sur un ordinateur ont beaucoup évolué en quantité et qualité dans les deux dernières décennies. Les disques durs atteignent des capacités de stockage magnétique persistant importantes, qui permettent de stocker des films de plusieurs heures en très haute définition afin de les éditer.

En 1985 est apparu le *CD* (*Compact Disc* ou *disque compact*), support optique de stockage qui offre une capacité entre 650 et 800 Mo. Ce support a tout d'abord révolutionné l'industrie musicale, en offrant le moyen de stocker une heure de son de très bonne qualité à un coût de revient minime. Il s'est ensuite exporté dans le monde informatique, sous le nom de *CD-ROM*, en permettant de diffuser massivement des contenus multimédias. Ce domaine a été à la mode pendant un moment, générant des applications riches, comme des musées virtuels ou des encyclopédies multimédias, par exemple *Microsoft Encarta*, ou bien des interfaces graphiques simplement augmentées d'artifices multimédias à faible coût. Dans la même lignée, le *DVD* (*Digital Versatile Disk* ou disque vidéo numérique) est créé en 1998 et révolutionne le marché de la cinématographie. Un DVD peut contenir de 4.7 à 17 Go, ce qui permet de stocker plusieurs heures de vidéo en très haute résolution au format *MPEG-2*. L'impact du DVD sur l'industrie multimédia se limite pour l'instant aux interfaces graphiques créées pour naviguer à travers le contenu vidéo et à la capacité de stocker une plus grande qualité de données. Le DVD a engendré peu de nouveautés majeures d'un point de vue des techniques multimédias.

2.3.3 Périphériques d'entrée-sortie

Ces composants sont ceux qui ont la plus grande importance dans le multimédia car ils saisissent et rendent les informations monomédias qui sont le fondement du multimédia. Nous ferons ici un tour des périphériques d'entrée-sortie, en omettant par la suite le terme entrée-sortie, en indiquant les tendances importantes de ces technologies et sans vouloir être exhaustif.

Les périphériques visuels de sortie sont les écrans et les cartes vidéos. Les écrans ont gagnés en puissance, devenant plus grand et de meilleure résolution, mais sont aussi devenus plus ergonomiques grâce aux écrans plats qui commencent à se répandre aujourd'hui.

La cartes vidéo est le périphérique le plus développé et probablement celui qui a eu le plus fort impact dans l'informatique grand public. D'énormes progrès ont été fait grâce à des sociétés comme *Matrox*, *3Dfx* ou encore *ATI*. En parallèle avec le développement des écrans, ces cartes ont permis d'afficher des informations dans des résolutions plus importantes, allant même jusqu'à utiliser plusieurs écrans. Mais c'est le monde du jeu vidéo qui a fait faire un bond en avant aux technologies vidéos en développant des cartes accélératrices 3D afin d'afficher spécifiquement les informations en trois dimensions sur les écrans. Leur succès important a permis d'atteindre de hautes performances et ont ouvert la voie à un rendu graphique quasi-photoréaliste dans un grand nombre d'applications multimédias. De plus, les cartes vidéo intègrent de plus en plus de circuits intégrés dédiés à des tâches spécifiques, comme la décompression de vidéo au format MPEG pour le décodage de DVD.

Les périphériques visuels d'entrée sont les scanners et les caméras numériques. Les scanners sont des appareils construits pour numériser des images et les enregistrer sous un format numérique. Ils sont intensivement utilisés aujourd'hui afin de sauvegarder sur supports informatiques la plus grande partie des ouvrages papiers des grandes bibliothèques mondiales. Les caméras numériques sont la version moderne des caméras analogiques et enregistrent les vidéos capturées en informations numériques, donc transférables puis manipulables par un ordinateur. Citons aussi les webcams qui sont une version beaucoup plus légère des caméras numériques et se destinent à diffuser des animations en faible définition sur l'Internet. Ces trois périphériques permettent de joindre le monde analogique de l'image et de la vidéo au monde numérique de l'ordinateur.

Les périphériques sonores en sortie ont suivi la même évolution que les périphériques visuels. Les haut-parleurs ont un rendu de meilleure qualité et sont maintenant fournis en groupes de quatre ou six afin de rendre de façon naturelle les sons multicanaux. Ces équipements ont fait l'objet d'une spécification nommée *home theatre* afin d'assurer une haute qualité de rendu des œuvres cinématographiques et utilisée conjointement aux équipements *DVD* avec des systèmes comme le *Dolby* ou le *DTS*. Les cartes sons se sont elles contenté de rendre les sons multicanaux de très haute qualité et d'ajouter la spatialisation au son.

Les périphériques tactiles ont d'abord évolué comme périphériques d'entrée, contrairement aux périphériques visuels et sonores. En dehors des légères améliorations de la souris (qui est devenue sans fil et infrarouge), l'apparition des tablettes graphiques dans le monde de l'infographie a fournit un périphérique plus adéquat aux créateurs de contenus graphiques. Ces périphériques ont bénéficié de la croissance des jeux vidéo à travers les évolutions ergonomiques du joystick, premier périphérique tactile grand public. Par la suite, les équipements issus de la recherche en réalité virtuelle se sont diffusés, comme les lunettes et les gants, ou *handglove*, de réalité virtuelle. L'évolution suivante a été de faire

de ces équipements des périphériques d'entrée. Le joystick à retour d'effort a été utilisé dans le monde du jeu vidéo et du contrôle d'automatisme, secteurs dans lesquels le prix de revient élevé se justifie. Les écrans tactiles ont tout d'abord été utilisés au sein de bornes interactives, nécessitant des investissements relativement lourds qui ont ralenti leur introduction dans l'informatique grand public. On les retrouve néanmoins dans les ordinateurs de poche *PDA* (*Personal Digital Assistant*, *palmtops*).

Les périphériques gustatifs et olfactifs sont très spécialisés et réservés à quelques rares recherches et applications [71, 97] . Ils restent de toutes façons marginaux étant donné la place de ces médias au sein des interactions sensorielles.

Citons enfin les recherches dans le domaine des périphériques électro-physiologiques [6] qui permettent de mesurer l'état physiologique d'un être humain à travers des paramètres comme le rythme cardiaque, la température du corps ou l'activité électrique du cerveau. Ces périphériques d'un nouveau genre visent à percevoir des caractéristiques comme nos émotions, notre niveau de stress ou de conscience. Ils proviennent du milieu de la médecine et offrent de nouvelles fonctionnalités aux systèmes informatisés, comme la prise en compte de certains handicaps ou bien la capacité de *biofeedback* qu'a le système pour indiquer à son utilisateur son état suite aux actions que celui-ci a effectué.

2.3.4 Nouvelles plate-formes

Même si l'ordinateur reste l'outil informatique numéro un aujourd'hui, de nombreuses nouvelles plate-formes remettent en cause son hégémonie et offrent de nouveaux paradigmes de travail.

Les consoles de jeu vidéo ont, à l'instar des ordinateurs dont elles se sont inspirés, beaucoup évolué ces dix dernières années. La figure 2.7 en présente quelques unes. Les consoles de dernière génération se révèlent être des machines puissantes dont l'architecture rappelle en de nombreux point celle des ordinateurs avec une spécialisation poussée dans l'audiovisuel, ce qu'illustre clairement la récente console *Xbox* de *Microsoft*. Il est même possible d'y faire fonctionner des adaptation des systèmes d'exploitation usuels comme *Linux* ou *Windows CE*. Les consoles de jeu se transforment aujourd'hui en plate-forme de loisir, permettant de lire des DVD et de se connecter à l'Internet pour naviguer et lire ses émails.

La même tendance s'observe dans le monde audiovisuel qui intègre les technologies numériques grâce à des acteurs comme *TiVo* ou *Canal+*. Les terminaux de télévision numérique (*set top box*) sont de petits ordinateurs comportant des systèmes dédiés et une unité de stockage importante, afin de pouvoir fonctionner comme magnétoscope numérique. L'introduction de systèmes informatique

Nintendo GameCube | Sony PlayStation 2 | Microsoft Xbox

FIG. 2.7 – *Exemples de consoles de jeux vidéos*

a permis de rendre interactifs ces plate-formes, rendant possible la programmation personnalisée ou la diffusion de versions interactives de jeux comme par exemple les courses hippiques avec les programmes du *PMU* sur la chaîne satellite *Equidia*. Ces terminaux offrent les mêmes services réseaux ajoutés aux consoles de jeu.

Les ordinateurs de poche *PDA* ont une popularité moindre que les précédents mais apportent tout un lot de nouveautés intéressantes. La figure 2.8 montre quelques modèles de tels ordinateurs. Leur interface est tout d'abord originale, car pour palier la petite taille de l'écran un stylet est utilisé sur un écran tactile. D'autre part la place des interfaces graphiques est accrue par rapport aux systèmes des ordinateurs fixes et met l'accent sur une utilisation efficace de l'espace d'affichage. Bien qu'aujourd'hui ces PDA soient surtout réservés aux tâches d'agenda et d'organiseur, la plupart des constructeurs, comme *3COM*, *Sony* ou *Toshiba*, comptent les faire évoluer vers des mini-ordinateurs capables de diffuser des vidéos en temps-réels sur les réseaux de dernière génération.

Nokia 9210i | Utilisation d'un stylet avec un iPaq | Archos Jukebox recorder 20

FIG. 2.8 – *Exemples d'ordinateurs de poche*

Cette dernière évolution rapprochera l'ordinateur de poche du téléphone mobile, outil qui a vu un développement foudroyant dans les cinq dernières années. Le téléphone mobile est un autre exemple de la créativité qui peut être mise en œuvre afin d'apporter de nouveaux paradigmes aux utilisateurs de systèmes informatiques. Un téléphone mobile a des ressources limitées: un écran de l'ordre de six lignes de quinze caractères ou en résolution 96 x 64 pixels, un clavier d'environ quinze touches et un haut-parleur en qualité radio. Les développeurs ont pourtant rivalisé d'ingéniosité afin de créer des applications conviviales aux contenus diversifiés. Avec l'extension de leurs capacités, le téléphone mobile rend possible la diffusion de courtes vidéo à faible débits et l'introduction de formats multimédias.

2.3.5 Dernières remarques

Nous voudrions conclure par deux remarques qui soulignent que le multimédia est un domaine de l'informatique qui dépend encore largement des progrès matériels et qu'il sous-tend des évolutions qui ne cessent de modifier le paysage technologique et humain.

Dans un premier temps, en dehors de la tendance à l'augmentation de la qualité des périphériques, on peut remarquer que la proportion de périphériques d'entrée augmente au sein des périphériques. Ceci est à rapprocher de l'importance accordée au sein du multimédia à l'interactivité, qui est la prise en compte des choix de l'utilisateur et est donc fortement dépendante des possibilités offertes par les périphériques d'entrée. L'interactivité est une importante nouveauté par rapport aux médias analogiques qui ont majoritairement été conçus pour être diffusés sans canal de retour pour renvoyer des informations de l'utilisateur au système. Elle offre de nombreuses possibilités attrayantes pour l'utilisateur mais donne un travail important aux auteurs et aux développeurs de systèmes.

Dans un deuxième temps, l'informatique converge vers de nombreux secteurs industriels, comme la cinématographie, les télécommunications et le jeux vidéo, qui sont traditionnellement plongés dans un paradigme analogique. L'introduction du numérique est un grand succès et ne cesse de s'accroître, mais ces évolutions prennent des visages différents selon le domaine dans lequel elles ont lieu. Par exemple, si le média vidéo est incontournable pour une société cinématographique, un opérateur de télécommunication ne le verra pas du même œil, alors que c'est l'inverse pour le média texte. La créativité est une activité vitale pour ces secteurs technologiques et se retrouve à la fois poussée par l'introduction du multimédia et à l'origine d'évolutions que l'informatique seule n'aurait pu déclencher. De nouveaux paradigmes sont encore à trouver pour pouvoir offrir une vision uniforme au sein de cette convergence générale et le multimédia occupe une place de choix pour les fournir.

2.4 Apports des systèmes d'exploitation et middleware

Les données monomédias référencées par les présentations multimédias étant gourmandes en ressources, nous avons vu à la section précédente que des matériels efficaces permettent de rendre la plupart des applications multimédias. Cette situation n'était pas la même aux origines du multimédia et les solutions envisagées étaient alors logicielles aux niveaux les plus bas des systèmes informatiques. Cette approche perdure encore aujourd'hui mais elle occupe une place plus minime et moins centrale car l'activité logicielle s'est structurée différemment au fil des évolutions de l'informatique.

Nous décrirons tout d'abord des problématiques générales concernant la gestion du multimédia au sein des systèmes d'exploitation puis nous nous intéresserons aux middlewares.

2.4.1 Systèmes d'exploitation

Les systèmes d'exploitation sont la partie logicielle des systèmes informatiques qui est en contact direct avec l'architecture matérielle de l'ordinateur. Ces systèmes sont d'une importance majeure puisqu'ils déterminent les services disponibles à l'utilisateur et organisent leur fonctionnement. Étant donné l'importance des périphériques matériels dans le cadre du multimédia, les systèmes d'exploitation sont donc directement responsables du bon fonctionnement logiciel de l'activité multimédia.

2.4.1.1 Modularité

L'approche qui consistait à construire des système d'exploitation d'un seul bloc n'est plus de mise aujourd'hui au vu de la taille et de la complexité des systèmes. Les développements doivent se faire en équipes, souvent distribuées à une échelle mondiale, ce qui exige de découper le système en modules, ou composants, clairement définis. Cette préoccupation va même au delà dans la définition des frontières du système d'exploitation, car d'autres systèmes coopèrent avec lui pour faire fonctionner l'ordinateur et ses applications, comme nous le verrons ci-dessous pour les middlewares.

La notion de module est rattachée à celle de service, c'est-à-dire un ensemble des fonctionnalités apportées. Divers standards et méthodes de spécification de modules logiciels existent afin de définir son interface de programmation, c'est-à-dire l'ensemble des procédures et programmes qu'il comprend et qui mettent en œuvre son service.

Parmi les services généraux que l'on retrouve dans tous les systèmes d'exploitation, il y a le gestionnaire de tâches, le système de fichier et l'accès au

réseau. Cette organisation des systèmes a permis de nombreux travaux de recherche et de développement sur les divers modules, augmentant leur efficacité ou les spécialisant pour des fonctionnalités nouvelles. Un système d'exploitation comme *Linux* pousse même le développement très loin en permettant de changer certains modules alors même que le système s'exécute.

2.4.1.2 Gestion des ressources

Le système d'exploitation a pour tâche de gérer les informations stockées en mémoire afin d'en déduire les instructions à exécuter sur le processeur. Les différentes activités que le système d'exploitation doit gérer sont appelées *processus*, généralement décomposés en *tâches* élémentaires, et recouvrent des formes variées selon les systèmes d'exploitation. Les systèmes d'exploitation modernes ont apporté des méthodes de gestion des processus complexes et diversifiées. Ils sont passés d'un fonctionnement uniquement séquentiel à une exécution concurrente de plusieurs processus, ce qui a obligé à chercher des algorithmes d'ordonnancement adéquats. La préemption, ou capacité à stopper temporairement un processus pour en démarrer un autre, est venue s'ajouter afin de permettre une certain souplesse de fonctionnement, afin d'assurer notamment une certaine réactivité du système au événements utilisateurs. Les processus sont devenus légers, ou *thread*, afin de les décomposer et leur permettre de partager des ressources. Le paradigme du *multitâche* (ou *multithreading*) a fourni le moyen de construire des applications plus complexes au sein desquelles le parallélisme des traitements est indiqué. Il faut ajouter que l'exécution du parallélisme est entièrement à la charge du système d'exploitation qui doit le simuler grâce à la préemption dans la plupart des cas car les architectures matérielles ne permettent que rarement un parallélisme réel des traitements.

Dans la partie centrale des systèmes d'exploitation, appelée *noyau* ou *kernel*, le gestionnaire de tâche est devenu de plus en plus évolué au fil des générations, comme l'illustre le cas du système *Mach* [125]. Les performances de ce composant déterminent non seulement l'efficacité du système d'exploitation mais aussi son ergonomie, à travers sa réactivité aux actions de l'utilisateur et aux interactions des programmes entre eux. On retrouve cette problématique de l'ordonnancement de tâches dans de nombreux systèmes multimédias, comme MADCOW [129] ou les systèmes distribués [153]. La notion de *temps-réel* provient initialement de la contrainte qu'une tâche doit s'exécuter dans un délai minimal au delà de sa date prévue (la préemption pouvant reporter la date).

L'idée d'un système d'exploitation multimédia a occupé de nombreux travaux [141]. Il s'agissait de déplacer les problématiques du multimédia au cœur des systèmes d'exploitation en offrant un traitement des divers médias et de leur synchronisation intégré au noyau. *BeOS* [11], surnommé le *Media OS*, est une concrétisation de cette approche. BeOS a mis en place des concepts intéressants dans la programmation de bas niveau des systèmes multimédias, comme le "multitâche diffusé" (*pervasive multitasking*) qui permet au système d'exploitation de

découper les programmes en tâches de la façon la plus efficace possible. Malgré son échec commercial, les problématiques suggérées par BeOS ne cessent d'être abordées au sein des autres systèmes, *Linux* [38] par exemple.

2.4.1.3 Interface utilisateur

Bien que les premiers systèmes d'exploitation n'aient pas jugé important de fournir une interface graphique pour l'accès aux services du système d'exploitation, celle-ci se révèle indispensable dans le cadre grand public car l'utilisateur n'est pas un spécialiste. L'une des raisons pour lesquelles le multimédia occupe une place importante dans l'informatique grand public est qu'il apporte des moyens plus intuitifs d'accéder à ces services à travers d'interactions sensitives, plutôt que cognitives.

Les premières approches furent simplificatrices, offrant essentiellement plusieurs fenêtres dans lesquelles taper des commandes textuelles. Plus tard, de réels systèmes graphiques apparurent sur le marché encore jeune de l'informatique, comme *Apple MacOS*, *IBM OS/2* et *Microsoft Windows*. Ces interfaces offrent de véritables représentations graphiques des concepts des systèmes d'exploitation, comme les fichiers ou les commandes, et sont devenus des composants fondamentaux de ces systèmes.

Les évolutions des interfaces graphiques ont apporté des représentations de plus en plus conviviales, en fournissant des abstractions qui n'étaient plus simplement le reflet des aspects de bas niveau du système d'exploitation. Ces interfaces ont pioché dans les résultats de recherches du domaine de l'interface homme-machine et dans les applications graphiques, qui continuent encore d'apporter des éléments originaux.

En augmentant ainsi l'intuitivité des systèmes d'exploitation, les interfaces graphiques ont permis à de plus en plus de systèmes d'être connus et utilisés des utilisateurs d'ordinateurs. Elles ont gagné en richesse en introduisant des sons et des animations, ce qui a été rendu possible grâce à des ordinateurs et des systèmes d'exploitation plus efficaces. Ces interfaces multimédias sont encore nouvelles et mal intégrées. Des initiatives tentent de formaliser les interfaces au sein de systèmes de programmation et de langages, comme par exemple *UIML* (*User Interface Markup Language*, http://www.uiml.org). Il n'y a pas de doutes qu'elles deviendront communes dans quelques années, comme celles d'aujourd'hui le sont devenues, et qu'elles bénéficieront des progrès réalisés dans le domaine multimédia.

2.4.2 Middleware multimédia

L'approche modulaire s'étant généralisée à tous les niveaux des systèmes informatique, des modules importants ont pu être développés par des tierces parties en dehors des contextes spécifiques à chaque système d'exploitation. La

classification en couches des systèmes n'a pas été modifiée, mais une couche appelée *middleware* est venue se glisser entre celles du système d'exploitation et des applications.

Les modules du multimédia ont toujours eu une place particulière au sein de l'organisation des systèmes car ils reposent sur une multitude de services de niveau d'abstraction plus ou moins haut. La couche middleware est l'emplacement idéal pour placer les services multimédias qui se trouvent souvent à l'interface entre le bas niveau du système d'exploitation et le haut niveau des applications multimédias. Cette séparation profite aussi aux programmeurs d'applications, qui peuvent ainsi bénéficier d'outils plus fonctionnels pour créer leurs programmes. A l'image des librairies de procédures ou d'objets que l'on retrouve dans tous les langages de programmation, les middlewares regroupent des fonctionnalités qui ne font pas partie des systèmes d'exploitation mais sont communément utilisées par les applications. Ce sont en quelques sortes de grandes librairies organisées de façon thématiques.

Le standard de *CORBA* [152] de l'*OMG* (*Object Management Group*) fut parmi les premiers middlewares. Il permet de créer de multiples applications en milieu hétérogène en fournissant un système complexe et complet de spécification des applications qui abstrait les systèmes d'exploitation et de communication sous-jacents. Dans le domaine multimédia, *PREMO* [75] fait partie des propositions qui se placent le plus clairement en position de middleware multimédia en offrant un *API* (*Application Programming Interface*) complète et un mode d'organisation des applications multimédias. Parmi d'autres exemples, [27] propose un *coprocesseur multimédia* qui vient se greffer au noyau du système d'exploitation afin d'y ajouter de façon transparente une gestion distribuée de la synchronisation multimédia. Des systèmes comme *VuSystem* [146], ou *Dalì* [113] proposent divers paradigmes de programmation d'applications multimédias qui offrent des services assimilables à ceux d'un middleware multimédia.

La bibliothèque de fonctions graphiques *OpenGL* (*Open Graphic Library*) permet de spécifier des objets 2D et 3D mouvants et le cadre de leur rendu (lumière, modèle physique, etc.). Cette vaste bibliothèque implémentée sur la plupart des systèmes d'exploitation a été fortement optimisée pour la plupart des périphériques de rendu et offre une haute qualité d'images et d'animations synthétiques en temps-réel. Elle a été à l'origine de nombreux développements de logiciels dans les domaines de la conception assistée par ordinateur et des jeux vidéo.

Le middleware *DirectX* de *Microsoft*, qui est au sens strict un composant des divers systèmes d'exploitation *Windows*, fédère les services multimédias. Il comporte divers sous-composants: *DirectDraw* et *Direct3D* pour le graphique, *DirectSound* pour le son (qui comprend *DirectMusic* pour la composition musicale), *DirectPlay* pour la communication via les réseaux, *DirectInput* pour la

communication directe avec les périphériques et *DirectShow* pour la gestion des flots de données. Étant porté vers diverses plate-formes matérielles [46], ce middleware promet des développements d'applications multimédias multi-systèmes.

Le langage *Java* et sa plate-forme constituent un exemple de middleware, particulier en cela que ses fonctionnalités sont presque aussi étendues que celles d'un système d'exploitation. Java fournit des abstractions pour implémenter tous types d'applications et les exécuter sur différents systèmes d'exploitation. L'ensemble des classes multimédias de Java sont regroupées dans les interfaces de programmation (*API*) nommées *Java Media API*. Ce groupe comprend les API suivantes: *2D*, *3D*, *Image I/O* et *Advanced Imaging* pour la programmation graphique; *JMF* (pour *Java Media Framework*) qui permet de programmer des *players*, applications spécialisées dans la gestion de données multimédias; *Shared Data Toolkit* pour les applications collaboratives; et *Sound* et *Speech* pour les applications de gestion du son. Bien que le développement du multimédia en son sein a souffert des mauvaises performances des premières moutures de machines virtuelles, le succès de Java ne se dément pas et les développements se poursuivent dans le cadre de Java Media API.

2.5 Contexte applicatif

Nous présentons finalement plusieurs exemples d'applications multimédia qui sont représentatives des diverses tendances dans le domaine. Nous parlerons tout d'abord des applications industrielles avant de nous tourner vers le domaine de la recherche.

2.5.1 Applications industrielles

2.5.1.1 Flash

Flash est un format binaire de fichier créé par la société *Macromedia* et qui en est à sa sixième version, surnommée *MX* [143]. Ce format a été conçu pour délivrer des graphismes et des animations vectorielles dans le monde du Web. La spécification du format Flash est publique ce qui explique sa grande notoriété. Elle définit un niveau bas, où le programmeur code les divers éléments Flash directement dans le format binaire, et un niveau haut, où le programmeur a à sa disposition une interface de programmation à base d'objets.

La structure d'un fichier Flash est présentée en figure 2.9. Il commence par un en-tête précisant divers paramètres généraux, se poursuit par une série de *blocs balisés* (*tagged block* ou *tag*) et se termine par un bloc final. Les types de blocs peuvent être étendu par le programmeur afin qu'il intègre ses propres structures de données. Les blocs ont deux rôles: les blocs de définition, qui définissent les éléments appelés *acteurs* (*character*) qui vont être manipulés dans la vidéo, et les blocs de contrôle, qui définissent ces manipulations des acteurs.

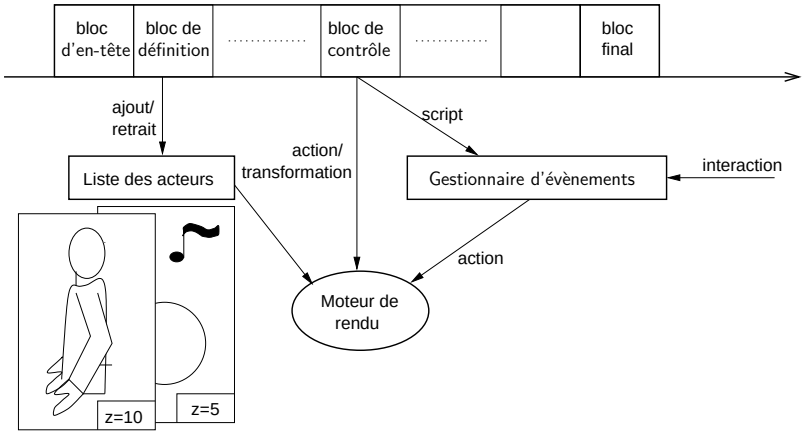


FIG. 2.9 – *Structure du format Flash*

La synchronisation temporelle des acteurs se fait tout d'abord selon un modèle timeline, d'ailleurs commun aux outils Macromedia comme *Director* et *Dream-Weaver*. Le programmeur place ses acteurs de façon absolue sur les pistes. Il peut ensuite associer un script à des événements liés aux acteurs afin d'affiner la synchronisation. Le langage de script *ActionScript* s'est inspiré de la norme *EcmaScript* (*ECMA-262* ou *ISO/IEC 16262*) qui elle-même formalise le langage JavaScript utilisé conjointement au langage HTML. ActionScript est un langage proche du langage *Lingo* fourni dans l'outil d'édition multimédia Director. Le langage ActionScript est orienté objet, mais est relativement lâche puisque certaines variables peuvent accepter tout type de donnée (entiers, réels et chaînes de caractères) indifféremment et les références aux variables peuvent être une chaîne de caractères évaluée uniquement à l'exécution de l'animation Flash. Il permet d'écrire des programmes impératifs dans les animations Flash en définissant des opérations arithmétiques et booléennes, les structures de contrôle conventionnelles de test et de boucle et des définitions de variables, de fonctions et de méthodes. Les événements et méthodes de démarrage et d'arrêt de la présentation d'un acteur permettent de programmer la synchronisation de divers acteurs. Ces scripts permettent de la même façon de gérer les interactions avec l'utilisateur aux travers d'événements utilisateur déclenchés via la souris ou une touche du clavier.

La principale force de Flash est l'interpolation automatique d'une animation à partir de deux scènes de manière efficace, ce qui est rendu possible par l'usage d'entités vectorielles. Chaque acteur est défini comme un *vecteur*, c'est-à-dire une série de coordonnées qui peut correspondre aux points d'une ligne brisée d'une courbe de Bézier quadratique. Les acteurs peuvent ainsi être manipulés de façon simple en modifiant uniquement les coordonnées de leurs points, le moteur de rendu se chargeant de retracer l'acteur si nécessaire. Le moteur de

rendu Flash gère une liste d'affichage des acteurs indiquant leur profondeur, ce qui donne un univers de rendu en 2.5D. Le système de rendu applique ensuite des transformations complexes aux objets vectoriels selon le scénario imaginé par le programmeur. Ces transformations comprennent des gradients pour les couleurs ou translations, des rotations par le biais de matrices de transformation, des homothéties, ou des morphing de formes géométriques entre elles.

2.5.1.2 QuickTime

QuickTime [144] est initialement un format de vidéo numérique créé par la société *Apple* en 1991. Au fil des années, QuickTime est devenu de plus en plus complexe jusqu'à devenir un format véritablement multimédia, même si son succès a été et reste encore dû à son format de compression vidéo.

Une présentation multimédia QuickTime est appelée film et est organisée en plusieurs pistes. Le long de chaque piste sont positionnés des éléments médias qui référencent tout ou partie d'une donnée monomédia, interne ou externe au fichier du film. Un élément média spécifie un certain type et des paramètres spécifiques aux données, comme le gestionnaire de données à utiliser ou bien les paramètres de celui-ci. Les pistes peuvent être groupées dans une même couche, l'affichage d'un film se faisant couche après couche, ou bien un groupe d'alternative, afin que seulement l'une des piste soit jouée à l'exécution en fonction de la langue ou de la puissance de la machine de rendu par exemple. Chaque piste indique les coordonnées d'un rectangle où sera affiché l'élément média, les conversions de coordonnées entre piste, élément média, film et écran se faisant par le biais d'une matrice de coordonnées 3x3.

La synchronisation temporelle est du type time-line arborescent, chaque piste ayant sa propre ligne de temps. Il est de plus possible d'attacher à un film, une piste ou un élément média une base temporelle sous la forme d'une échelle temporelle indiquant l'unité des valeurs et la vitesse à laquelle le temps s'écoule. Les temps locaux sont convertis en leurs équivalents dans la base temporelle du film lors de l'exécution du film.

La plate-forme d'exécution QuickTime est constituée d'une boite à outil qui gère l'interface de présentation et traite les données du film en les redirigeant vers des gestionnaires de son et de compression d'image. De multiples composants sont appelés lors de l'exécution du film et leur gestion dynamique est assurée par un gestionnaire de composant. Une application QuickTime est en fait un composant qui utilise les services définis dans l'interface d'autres composants dont ceux du système QuickTime.

QuickTime VR est une extension créée afin de permettre au spectateur du film de naviguer librement au sein d'un monde virtuel 3D. La figure 2.10 propose trois vues prises lors d'une navigation dans un film QuickTime VR. Ce monde

virtuel est spécifié dans le film QuickTime comme une scène composée par des nœuds représentant des objets qui sont positionnés dans le monde virtuel. En plus de pouvoir choisir dans quelle direction regarder dans le monde virtuel, l'utilisateur peut interagir via des liens afin d'effectuer certaines actions, comme changer de point de vue. L'interaction des films QuickTime comprend de plus la possibilité qu'une piste réagisse à certains événements utilisateurs en effectuant un jeu d'actions.

FIG. 2.10 – *Diverses vues de la navigation dans un film QuickTime VR*

L'organisation du format de fichier QuickTime se base sur la structure d'*atome* qui est un bloc définissant une hiérarchie de méta-données relative à une donnée, qui est ensuite référencée par une URL ou bien contenue dans le fichier. C'est ce principe qui a été retenu pour le format de fichier du standard MPEG-4 [88] qui partage comme autre point commun avec QuickTime un interfaçage avec la plate-forme Java.

2.5.1.3 HotMedia

HotMedia [90] est un système multimédia produit par *IBM* en 1999 à l'intention du web et en est à la version 3.5. Il repose sur un format de fichier binaire qui encapsule les données de la présentation et les enrichit par une structure complexe, l'ensemble étant envoyé à la demande d'un navigateur web par un serveur dédié via le protocole HTTP. La figure 2.11 représente l'architecture générale de la plate-forme Hotmedia, comprennant le client et le format de fichier.

Le format HotMedia multiplexe des données monomédias et leurs players (programmes de présentation) lorsque les formats sont spécifiques ou les nécessitent. Ceci permet au format d'être auto-suffisant en ne présupposant pas la présence de certains *plugins* installés sur le poste client. Le problème de portabilité du code est contourné en utilisant des classes Java, ce qui permet d'afficher la présentation dans un navigateur web sans aucune modification. La famille des players de l'actuelle plate-forme HotMedia est riche et supporte un grand nombre d'encodages de médias.

Les données de synchronisation et les actions que l'utilisateur peut déclencher sont spécifiées dans le format et, comme les données, envoyées progressivement

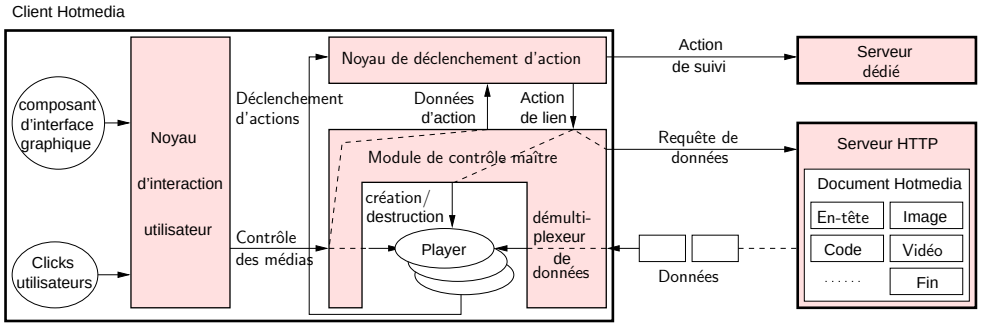


FIG. 2.11 – *Architecture de la plate-forme Hotmedia*

du serveur au client lors du déroulement de la présentation. L'auteur de la présentation doit se servir de l'outil d'assemblage d'IBM afin d'importer les contenus et préciser leur synchronisation. Pour cela, l'auteur définit des intervalles spatio-temporels (range) associés aux contenus et leur attache des actions. Ces actions sont produites par des déclencheurs qui sont liés aux interactions utilisateurs ou bien à des transitions états de la présentation ou d'un player et à une zone spatiale fixée. Elles peuvent remplacer la présentation de l'intervalle d'origine par celle d'un nouveau média ou d'une nouvelle présentation, ou bien simplement déclencher leur présentation, en ajoutant si nécessaire des effets de transition tel qu'un fondu enchaîné ou bien un zoom. La plate-forme HotMedia permet de suivre les interactions avec l'utilisateur, c'est-à-dire ses clics de souris et sa progression au sein de la présentation, en utilisant des jetons qui sont générés à certains endroits de la présentation. Une des applications de cette fonctionnalité est de déduire des modèles de comportement des utilisateurs afin de s'adapter à leurs besoins.

2.5.1.4 Les jeux vidéo

Longtemps cantonnées au rang de distractions informatiques, les applications de jeux vidéo prouvent aujourd'hui qu'elles vont au delà. Ce sont parmi les applications multimédias les plus complexes car elles utilisent un grand nombre de médias pour créer un contenu fortement multimédia, à la différence d'un simple enrichissement. Il n'est donc pas étonnant, comme nous l'avons vu auparavant, que ce domaine ait été un moteur puissant pour le développement du multimédia en informatique. La figure 2.12 présente deux images issues de jeux vidéos récents.

En tant que programmes informatiques, les jeux vidéo sont des applications très optimisées car elles sont en constante interaction avec l'utilisateur et nécessitent des temps de réaction très courts. Elles usent de traitements algorithmiques complexes, puisque les dernières générations offrent des capacités d'adaptation

Riven

Unreal tournament

FIG. 2.12 – *Exemples de jeux vidéos*

importantes aux actions du joueur en usant de formes d'intelligence artificielle issues de recherche avancées. Les mécanismes de création des scénarios de ces jeux mettent en œuvre des résultats de recherche des systèmes d'information tels que des techniques dialectiques ou déductives [124]. De nombreuses collaborations ont d'ailleurs été établies entre des sociétés de jeux vidéo et des laboratoires de recherche, servant ainsi le transfert de technologies vers le grand public. Certains jeux vidéo importants comme *Doom*, *Descent 3*, *Quake II* et *Unreal Tournament* sont des programmes ouverts qui ont permis à des chercheurs de les modifier afin de simuler certaines théories [95].

En tant qu'entité multimédia, un jeu vidéo se fonde sur une forte intégration des médias afin de pousser toujours plus loin son réalisme et améliorer l'immersion du joueur [82, 34]. Les jeux vidéo sont des œuvres dont la créativité va parfois très loin, comme dans le contexte de réalité augmentée lorsque le jeu vidéo s'immisce dans l'univers du joueur [119]. L'imagination est un aspect indispensable des jeux vidéo (et probablement du joueur) et pousse à toujours améliorer les jeux vidéos, caractéristique que l'on retrouve dans de nombreux aspects du multimédia.

2.5.2 Applications de recherche

2.5.2.1 Madeus

Madeus est un outil auteur [92] dont l'objectif est l'édition de documents interactifs multimédias. Il applique des résultats de recherche du domaine des documents structurés [127]. Le document édité par l'outil Madeus se présente selon plusieurs vues, chacune accessible dans une fenêtre spécifique dans laquelle l'auteur peut travailler librement. Le logiciel fait partie de la famille WYSIWIG

(What You See Is What You Get) dont le principe est d'afficher le document tel qu'il sera rendu et une modification de l'auteur est perçue dans toutes les vues.

Un format textuel à base de balises a été défini pour les documents Madeus, grâce au langage XML. Le logiciel a aussi été adapté afin de pouvoir éditer des documents SMIL. Un document Madeus possède une structure logique, spatiale, temporelle et hypermédia, chacune d'elle étant organisée selon une hiérarchie extensible [132]. Les éléments de base sont des objets auxquels sont attachés des attributs afin de définir les divers paramètres qui vont permettre de les présenter. Les diverses catégories d'objets sont gérées par des médiateurs qui se chargent spécifiquement d'eux au sein du gestionnaire de présentation de l'outil. Des liens peuvent être tissés entre objets afin de permettre de naviguer dans le document ou se déplacer hors de lui.

Madeus permet d'imposer des contraintes temporelles entre deux objets quelconques du document. Les relations temporelles sont inspirées de celles de *Wahl-Rothermel* [154] (cf. 3.3.2.3) et ajoutent les relations causales comme `parmin`, `parmaster` et `parmax`. Le système sous-jacent à l'interface graphique d'édition gère ces contraintes en les traduisant sur les durées des objets puis cherche une solution au système d'équations formées. Le même principe de contraintes est suivi pour la composition spatiale du document, grâce à l'introduction de relations spatiales d'alignement, de centrage, d'espacement et de décalage [30]. Le formatage des documents utilise le solveur de contraintes *DeltaBlue* qui donne la liberté à l'auteur de créer son document de façon incrémentale par ajouts successifs de contraintes entre les divers objets. Le solveur ne pouvant contrôler la cohérence du document, un algorithme de vérification doit être appliqué avant de formater le document [93].

2.5.2.2 CMIFeD et GRiNS

CMIFed [70] et *GRiNS* [25, 26] sont deux outils auteurs qui s'attaquent au même problème que Madeus sous un angle différent. Ils dérivent des travaux autour des documents structurés hypermédias qui se sont conclus par l'élaboration du format *CMIF* (*CWI Multimedia Interchange Format*) et du modèle hypermédia *Amsterdam* [68, 145]. Le format CMIF et le modèle Amsterdam reposent sur une description hiérarchique de la structure d'une présentation multimédia. Les composants atomiques référencent des données monomédias. Les composants composites comprennent des attributs dans lesquels sont indiqués les composants fils, la composition temporelle ou bien le simple regroupement, les ancres et les liens entre ces ancres.

La composition temporelle se fait tout d'abord au niveau des groupements, qui sont de deux types: séquentiels et parallèle. De plus, des arcs de synchronisation peuvent joindre un début ou une fin d'un composant à ceux d'un autre composant. Bien qu'identiques aux liens, ces arcs de synchronisation font l'objet d'un concept séparé pour des raisons de clarté et le problème de cohérence

temporelle qu'ils posent est placé hors du cadre du modèle. Tout composant est associé à un canal, qui permet d'indiquer les ressources avec lesquelles ses données seront affichées et leurs paramètres de rendu, par exemple le volume ou la position. Chaque canal sera mis en correspondance avec les ressources matérielles disponibles à l'exécution par le système qui a pour responsabilité de résoudre tous les conflits d'accès aux ressources.

CMIFed permet alors de visualiser le document spécifié dans le modèle Amsterdam selon les vues logique, temporelle et de présentation. Suite à la spécification du langage SMIL, CMIFed a donné naissance à l'outil GRiNS qui applique à SMIL les mêmes principes de vues sur le scénario.

2.5.2.3 Les formats MPEG

MPEG (*Moving Picture Coding Group*) est un groupe du comité technique commun ISO/IEC qui élabore des standards de représentation codée des images animées, des sons et de leurs combinaisons. Le premier standard MPEG-1 date de janvier 1988 et le dernier, MPEG-21, sera achevé à la fin de l'année 2003.

MPEG-1 et MPEG-2 proposent des méthodes de compression des signaux audiovisuels selon des techniques non-génériques. Dans le domaine du son, un modèle psychoacoustique de format MPEG-1 Layer 3 (ou mp3) permet d'enlever les portions du signal hors du champ fréquentiel d'un utilisateur moyen. Pour l'animation, le codage différentiel de trames permet d'éviter la redondance et d'utiliser des informations sur le mouvement afin de recalculer des trames proches de l'original. Il existe trois types de trame: les *I-frame* donnant une image complète, les *P-frame* codées par rapport aux trames précédentes et les *B-frame* codées comme une combinaison de trames *I-frame* et *P-frame* voisines. Le format MPEG-2 a une place importante puisqu'il a été choisi comme format pour les films numériques sur DVD.

MPEG-4 [88, 33] s'oriente vers des aspects plus abstraits de la représentation des signaux audiovisuels, tant naturels que synthétiques. Les fonctionnalités du standard se décomposent en quatre groupes: *DMIF* (*Delivery Multimedia Integration Framework*; distribution), *Systems* (structure générale), *Audio* et *Visual*. La structuration d'une vidéo MPEG-4 repose sur la notion d'objet média, notion orthogonale à la représentation de bas niveau des données envisagée dans les standards précédents.

La composition des objets MPEG-4 est exprimée dans le langage *BIFS* (*Binary Format for Scene Description*) qui tire en partie ses fonctionnalités du langage *VRML* que nous décrirons en section 2.5.2.4. Une spécification BIFS décrit donc des scènes organisées hiérarchiquement selon un arbre qui peut évoluer tout au long du déroulement de la vidéo. *XMT* (*Extensible MPEG-4 Textual Format*) est le format textuel utilisé pour stocker les descriptions de scènes défini en *XML*

Schema et intègre des sous-ensemble des langages SMIL et X3D. Les données XMT sont multiplexées avec les différents flux de données médias que l'auteur synchronise et peuvent être compilées en une représentation BIFS. La synchronisation des flux de données est basée sur le placement d'estampilles temporelles dans les flux. Les temps de gestion des ressources (tampons, connexions réseaux) peuvent être pris en compte dans les mécanismes d'ordonnancement des objets, qui proposent de plus des techniques de correction et de dissimulation d'erreurs. Dans la version 2 du standard MPEG-4, les objets peuvent être synchronisés dans l'arbre BIFS grâce à trois relations de haut niveau d'abstraction: `CoStart`, `CoEnd` et `Meet`. C'est la couche de synchronisation *FlexTime* qui assure le respect de ces contraintes en ralentissant, accélérant, stoppant ou décalant des flux selon les besoins.

Les informations spatiales sont indiquées sous forme de transformations spatiales associées aux nœuds BIFS et sont appliquées en cascade lors du parcours des nœuds. Le standard permet aussi la création d'animations de visages [54], ou de corps humains à partir de structures maillées dont seuls les paramètres caractéristiques sont indiqués.

MPEG-7 [104] spécifie une description du contenu, indépendamment de sa représentation codée. L'objectif est de pouvoir appliquer de nombreux traitements (semi-)automatique, dont la recherche d'informations et l'adaptation. Le standard a pour éléments de base le langage *DDL* (*Description Definition Language*) qui se décline en un format textuel défini en *XML Scheme* et une version binaire. DDL permet des créer deux entités: une description et un schéma de description. Les descriptions spécifient les aspects auditifs (instruments, voix, mélodie) et visuels (couleurs, textures, formes), mais aussi le mouvement et la navigation. Les schémas de description organisent les différentes descriptions entre elles.

MPEG-21 [20] s'attaque lui à la définition d'une structure générale du multimédia afin de rendre possible l'utilisation transparente et augmentée des ressources multimédias via un grand nombre de réseaux et de périphériques. La vision proposée par MPEG-21 s'intéresse aux divers utilisateurs de la chaîne de circulation du contenu (création - production - transport - consommation). Sept éléments clé définissent les différentes parties du standard qui traitent les divers aspects des *œuvres numériques* (*Digital Item*). La description des œuvres et de leur environnement prend en compte les préférences des utilisateurs et les caractéristiques de l'environnement afin de pouvoir adapter le contenu. MPEG-21 est une initiative ambitieuse et nécessaire qui offre au domaine multimédia d'acquérir ses lettres de noblesse.

2.5.2.4 VRML et X3D

X3D (*eXtensible 3D Graphics*) [155] est la dernière mise à jour du standard *ISO VRML* (*Virtual Reality Modelling Language* ou *ISO/IEC 14772-1:1997*) [1].

X3D ajoute à VRML une modularisation en composants, niveaux de composants et profils des diverses fonctionnalités, ces aspects offrant une plus grande ouverture et flexibilité au standard. La conformité au format XML, à travers un encodage particulier, est optionnelle et est introduite afin de faciliter l'intégration de VRML dans le Wolrd Wide Web.

VRML [101] permet de décrire un *monde virtuel* sous la forme d'un graphe de scène définissant la hiérarchie de *nœuds*. Chaque nœud possède un type qui définit l'ensemble de ses champs, les types des champs pouvant être booléen, flottant, chaîne de caractères ou encore vecteur 2D. De nombreux types de nœuds sont définis, tel que des objets médias (`AudioClip`, `MovieText`), des formes géométriques (`Sphere`, `NurbsSurface`, `Humanoid`), des transformations afin de modifier la valeur d'un champ, des capteurs, des interpolateurs ou bien encore des scripts. La relation hiérarchique entre nœuds permet d'indiquer que les fils sont les arguments du nœud père qui fournit des champs par héritage et a la charge d'appliquer certaines actions à ses fils.

Les nœuds peuvent générer des événements et réagir à certains via l'utilisation de nœuds capteurs. Le programmeur peut de plus déclarer une route que les événements doivent suivre et ainsi définir une séquence des actions à engendrer. Les nœuds médias possèdent des instants de début `starTime` et de fin `stopTime` et un indicateur de boucle `loop` afin d'indiquer leur synchronisation temporelle.

L'interactivité des mondes VRML repose sur l'usage de senseurs afin de réagir aux événements utilisateurs qu'il peut produire à partir des périphériques d'entrée de son ordinateur ou via un *avatar*, représentation d'un spectateur dans un monde virtuel libre d'évoluer dans celui-ci. C'est au navigateur VRML de calculer l'apparence des objets définis dans la scène en intégrant divers paramètres comme la luminosité, résolvant les cas de collisions entre objets après application d'interpolateurs et produire la scène sous l'angle de vue de l'avatar. Les scripts permettent d'étendre les traitements au delà de ce que permettent les interpolateurs fournis par le langage en utilisant des langages de script externe, comme *ECMAScript* ou *Java*. La langage permet la définition de nœuds génériques par le biais d'un nœud `PROTO`, si l'instanciation est définie dans le même fichier, ou `EXTERNPROTO`, si elle l'est dans un autre fichier.

Bien qu'ouvrant la voie à un paradigme nouveau et intéressant, ce standard n'a pas eu un impact important et le rapprochement du standard XML doit lui permettre de s'ouvrir aux programmeurs et utilisateurs du web. La figure 2.13 illustre la structure d'un navigateur VRML.

2.5.2.5 Alice

Alice [65] est un outil implémenté à l'université de Virginia pour le prototypage rapide d'applications de réalité virtuelle. L'outil Alice permet de définir

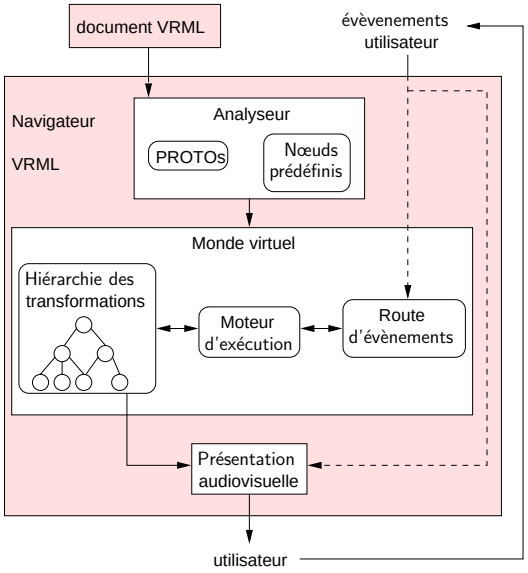


FIG. 2.13 – *Structure générale d'un navigateur VRML*

des mondes virtuels en créant des objets à partir de formes de base et en les assemblant au sein d'une hiérarchie grâce à des manipulations graphiques (glisser, déposer). Cette hiérarchie permet de construire des objets à plusieurs parties, dont chacune pourra être déplacée, tout en encapsulant ces parties dans un objet de plus haut niveau et dont le déplacement modifiera l'ensemble, en considérant les objets comme solides et en assurant qu'ils ne s'interpénètrent à aucun moment. Par exemple, l'objet `snowman` sera composé de trois objets, dont sa tête `head` qui sera composée d'un objet `nose` que l'on désignera par l'identificateur `snowman.head.nose`. Il est aussi possible d'importer des images, des sons et des vidéos et les afficher sur des zones géométriques.

L'auteur peut ensuite définir un script afin d'animer son monde virtuel en spécifiant le comportement de ses objets. Le langage de script d'Alice est proche du langage *Python*, langage orienté-objet interprété dans lequel Alice a été écrit. Ce langage est suffisamment puissant pour exprimer des traitements complexes (calculs, etc.) et de haut niveau pour ne pas rebuter le programmeur novice.

Dans Alice, le scénario est indiqué à l'aide de commandes qui permettent de créer et détruire des objets, changer leurs propriétés, les bouger et prendre en compte les actions du participant de façon dynamique. Trois commandes générales permettent d'organiser l'appel de ces commandes: `DoInOrder` pour mettre en séquence des actions, `DoTogether` pour effectuer la mise en parallèle et `loop` pour la répétition. La figure 2.14 propose un exemple de scénario et une

illustration du logiciel Alice où ce scénario est spécifié.

```
DoInOrder (
    DoTogether(
        snowman.move(forward,1),
        oven.door.turn(down,1/4)
              ),
    snowman.resize(1/4)
          )
```

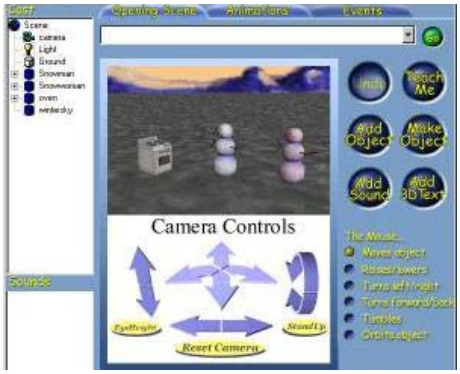


FIG. 2.14 – *Exemple de scénario dans Alice*

Le découplage entre la simulation des scènes du monde virtuel et leur rendu permet au logiciel de présenter les informations en temps-réel, malgré le fait qu'il soit interprété. La simulation calcule l'état du monde et le rendu entretient une base de données géométriques afin de calculer le point de vue courant du participant. L'optimisation de ces processus rend possible l'interaction en temps-réel qui privilégie la prise en compte de ces interactions afin d'assurer une certaine réactivité.

Alice repose sur un outil et un langage simples qui se révèle néanmoins puissant. L'intuitivité ouvre l'outil à des créateurs non-spécialistes et dont beaucoup pensent qu'il est source de créativité

2.6 En conclusion

Nous avons ici parcouru une partie relativement vaste du domaine multimédia en décrivant des aspects divers. La quantité d'objets et de problèmes recouverts par le domaine dans son ensemble est importante, même si tous les aspects examinés ne sont pas toujours raccrochés au domaine multimédia. Il en ressort également que le domaine est en perpétuelle évolution et bénéficie d'une créativité importante. Cette imagination est à la fois une source d'intérêt pour de nombreux utilisateurs et l'origine de divergences conceptuelles.

Il se dessine néanmoins aujourd'hui une convergence entre diverses plateformes matérielles, et au-delà leurs industries, autour d'aspects centraux proches du multimédia. La diversité fait alors place à une uniformité qui est relative à certaines caractéristiques du multimédia. Il faut alors assurer un certain équilibre entre les fonctionalités offertes et les difficultés, tant théoriques que pratiques, pour offrir ces mêmes services.

Chapitre 3

Composition de présentations multimédias

Les systèmes, modèles et langages informatiques apportent de nouvelles fonctionnalités ou en généralisent certaines en les regroupant à un plus haut niveau d'abstraction. Parmi les notions qui se dégagent des divers systèmes multimédias, la notion de présentation multimédia occupe une place importante. Elle joue le rôle de point pivot autour duquel on rattache divers problématiques et sert de point de référence dans les systèmes multimédias. Ce sont les entités auxquelles nous nous sommes intéressé, en examinant plus particulièrement leur composition.

Après avoir fixé la terminologie que nous emploierons dans le reste du mémoire, nous ferons le tour des différents aspects de la composition des présentations multimédias. Il s'agit ici d'en dégager les caractéristiques générales et de présenter les méthodes couramment employées dans le domaine multimédia. Nous terminons ce chapitre en présentant plusieurs langages multimédias pertinents vis-à-vis des problématiques présentées.

3.1 Présentations multimédias

Une *présentation multimédia* est un assemblage de données monomédias créée afin d'être présentée selon la sémantique spatio-temporelle définie par cet assemblage. Elle fournit la plupart du temps une abstraction des mécanismes de la mise en œuvre de sa sémantique et peut donc être interprétée par une variété d'applications différentes. La production de présentations multimédias s'organise autour d'une chaîne qui peut être diversifiée mais dans laquelle on retrouve néanmoins toujours un certains nombre de phases caractéristiques que nous décrivons ci-après, en terminant par une remarque plus générale. Les caractéristiques évoquées ici sont représentées dans la figure 3.1.

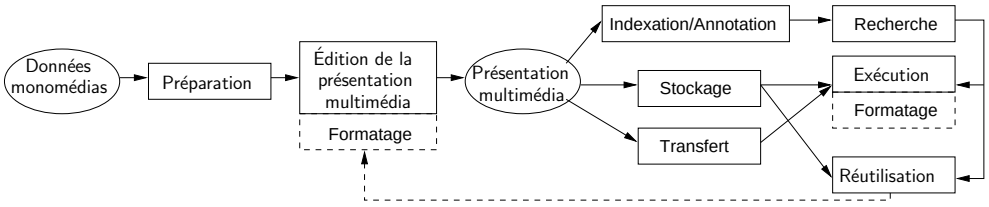


FIG. 3.1 – *Chaîne de production multimédia*

La récupération et la préparation des données monomédias

L'auteur possède initialement un ensemble de données monomédias brutes, qu'il doit éventuellement retravailler afin de pouvoir passer à la phase suivante. Certaines données médias peuvent être annotées pour faciliter leur utilisation dans des phases ultérieures, comme l'annotation du contenu multimédia ou sa réutilisation.

L'édition de la présentation

Dans cette phase, l'auteur utilise un outil multimédia, application ou langage de programmation, afin de définir l'assemblage des données monomédias dont il dispose. Ces outils facilitent le travail de spécification de l'auteur en lui fournissant des moyens techniques ou conceptuels permettant de la définir. Les applications d'édition fournissent une certaine rapidité de développement grâce à l'usage d'interface graphiques, alors que les langages exhibent des notions pertinentes dans lesquelles l'auteur pourra transcrire ses idées. Selon l'approche suivie, l'auteur aura ou non le choix du format final de sa présentation multimédia.

Le stockage, l'indexation, l'annotation et le transfert

La présentation doit être stockée de façon permanente si l'auteur désire la rejouer ou la distribuer. Selon l'approche d'édition suivie par l'auteur, il aura le choix de stocker sa présentation multimédia dans un fichier intégrant les données monomédias ou bien de n'y faire que des référence à celles-ci. Il peut ensuite indexer sa présentation multimédia afin d'utiliser les services d'une base de données et distribuer son œuvre à travers les réseaux informatiques s'il le désire.

L'exécution

Finalement, un utilisateur va restituer, ou encore exécuter, la présentation multimédia sur son ordinateur. Par souci de clarté, nous parlerons d'exécution d'une présentation multimédia. La présentation multimédia peut être dans un format directement exécutable ou bien nécessiter un système d'exécution particulier selon son format. L'exécution peut nécessiter le transfert de données, voire d'applications, via un réseau et l'interrogation d'une base de données.

Le formatage

Cette phase est particulière, en cela qu'elle peut avoir lieu à la fin de la phase d'édition ou au début de la phase de présentation, selon l'outil que l'auteur aura choisi. Elle consiste à transformer la spécification de la présentation multimédia afin de produire des données du format désiré. Cette phase peut nécessiter des traitements divers, comme le calcul d'informations manquantes ou la récupération de caractéristiques sur les données monomédias utilisées.

La réutilisation

Cette phase additionnelle est un cas particulier de la phase d'édition lorsque l'outil choisi par l'auteur lui permet de reprendre tout ou partie d'une présentation qu'il a déjà créée. La modification de cette présentation équivaut pour l'auteur à continuer la phase d'édition au lieu de passer à la phase de stockage.

Remarque: documents et applications

On observe dans le domaine du multimédia comme dans beaucoup d'autres domaines de l'informatique que les outils sont soit des documents, soit des applications. Cette séparation rejoint celle des données et des traitements dans les langages de programmation car les documents donnent le moyen à l'auteur de définir les données alors que les applications lui permettent de les manipuler. On retrouve aussi cet aspect dans le type du format final qui peut nécessiter une interprétation ou bien être exécuté et peut être textuel ou bien binaire. Le format textuel permet de créer des documents indépendamment des applications qui vont les présenter et ainsi de permettre d'implémenter diverses applications pour le même format, sacrifiant souvent l'efficacité de l'exécution au profit de la portabilité. Par contre un format binaire peut être optimisé pour un système d'exécution (système d'exploitation ou middleware) donné et garantir que des contraintes fortes sont respectées.

Cette dichotomie est importante car le choix de l'approche empêche bien souvent le programmeur/auteur de comprendre un outil ou de l'étendre et l'améliorer pour ses propres besoins. En effet, les applications multimédias sont le plus souvent des outils commerciaux, Macromedia Director étant probablement le plus connu d'entre eux. Ils offrent une multitude de fonctionnalités d'édition évoluées et la possibilité de stocker les présentations multimédias sous des formats variés mais reposent souvent sur des modèles de document pauvres. Un langage par contre offre des possibilités conceptuelles importantes puisqu'il est fondé à partir d'idées abstraites et est ouvert car il est toujours possible de l'étendre.

3.2 Structure des présentations multimédias

Les présentations multimédias abstraient le contenu brut des données qu'elles assemblent et les intègrent au sein d'une nouvelle entité. Tout comme d'autres

systèmes, elles contribuent à élever le niveau conceptuel des informations manipulées en informatique. Mais la diversité des abstractions peut être un frein à leur compréhension. Il faut donc fournir des structures qui permettent de mieux les cerner et offrir aux non-spécialistes un guide pour progressivement assimiler ces notions. Dans le domaine du multimédia, deux structurations traditionnelles comprennent les modules et les dimensions.

3.2.1 Modularité

La *modularité* impose de définir les systèmes ou documents par parties, ou modules ou encore composants. Les informations sont ainsi encapsulées dans des modules qui peuvent être hiérarchisés et manipulés indépendamment les uns des autres. Chaque module doit pouvoir être désigné de manière unique dans la hiérarchie. L'utilisation de cette désignation permet de partager un module au sein d'un même système ou bien encore de le réutiliser entre divers systèmes. Cette structure fournit un guide au programmeur pour qu'il factorise les informations en évitant la redondance, ce qui permet d'économiser des ressources et de localiser les erreurs.

Une des conséquences immédiates de la modularité au niveau des systèmes multimédias est l'*incrémentalité* du processus de création. Il est en effet possible de créer les présentations multimédias par étapes, en concevant chaque module l'un après l'autre ou bien en parallèle [18]. Cette incrémentalité est exploitée dans les systèmes d'édition multimédia, comme Madeus ou GRiNS. Ils offrent à l'auteur le moyen de définir les divers morceaux de sa présentation multimédia indépendamment les uns des autres en suivant une certaine méthodologie de création, qui vont généralement des morceaux les plus précis au moins précis ou dans la direction inverse. L'incrémentalité se révèle indispensable dans certains contextes, comme l'édition coopérative où la composition est répartie entre plusieurs endroits.

3.2.2 Dimensions du multimédia

Un aspect de la structure importante dans le domaine du multimédia réside dans les quatre dimensions usuelles selon lesquelles les informations sont structurées:

- la dimension logique
  Elle correspond à l'organisation sémantique du document [92] par le biais de relations de haut niveau conceptuel, compréhensibles uniquement par l'être humain. L'organisation de cette structure est le plus souvent hiérarchique [122], comme on le voit par exemple dans l'organisation traditionnelle en chapitres, sections et paragraphes.
- la dimension temporelle
  Toute application ou document multimédia utilise et définit des données qui évoluent dans le temps. Ces diverses données doivent donc être organisées afin de spécifier leurs liens dans le temps et la façon dont chacune

réagit à l'écoulement du temps. Nous reviendrons en détail sur cet aspect dans la section 3.3.

- les dimensions médias
  La très grande majorité des applications et documents multimédias visent à présenter des informations sur des périphériques de rendu. Ici aussi, l'organisation des données doit indiquer où présenter les informations dans l'espace, les structures utilisées pouvant être absolues ou bien relatives. La plus grande partie des systèmes multimédias n'envisagent aujourd'hui que deux dimensions médias: visuel et auditif. D'autres médias sont envisageables mais ne constituent actuellement qu'une part infime dans la proportion de périphériques de rendu, comme le goût par exemple [16].
- la dimension hypermédia
  Cette dimension regroupe les liens hypermédias qui font partie de la présentation multimédia. Cette dimension décrit les possibilités de navigation à travers le document et les formes d'interactivité permises.

Ces dimensions ne sont pas indépendantes les uns des autres et leur séparation est parfois problématique [128], puisque certaines informations se retrouvent à cheval entre deux dimensions. En effet, la structure logique d'un document peut comprendre des informations qui proviennent des structures spatiale (par exemple pour une section composée de trois paragraphes est la succession de ces trois paragraphes) et temporelle (un film est la séquence de ses scènes). La séparation de ces dimensions permet de sacrifier l'expressivité au profit de la clarté, afin de ne pas surcharger la tâche de l'auteur.

A l'exécution de la présentation multimédia, le système prend en compte les divers aspects spécifiés afin de déterminer les actions à entreprendre pour rendre la présentation conformément à l'intention de l'auteur. Comme on peut le voir dans le cadre des œuvres cinématographiques, le sens de la présentation naît de l'interaction de ces différentes dimensions. Par exemple, des films comme *Memento* ou *Run, Lola, Run* jouent à la fois sur les dimensions spatiales, par le biais des lieux et des personnages, et temporelles, par le biais de l'enchevêtrement des scènes et des répétitions de situations, d'une manière relativement évoluée.

3.3 La synchronisation temporelle

Les données monomédias sont plongées dans le temps, même si leur comportement n'en dépend pas toujours, et confèrent aux présentations multimédias la même propriété intrinsèque. Des multiples facteurs à prendre en compte dans une présentation multimédia, le temps est le plus important car c'est une ressource critique. Contrairement aux autres dimensions, il n'est pas contrôlable par les systèmes informatiques car son écoulement est une donnée intrinsèque. Par exemple, aucun équivalent à la notion de passé n'existe dans l'espace, même si des similitudes existent [83, 96].

Il est donc nécessaire d'indiquer comment sont assemblées les données monomédias dans le temps et vis-à-vis des autres données de la présentation multimédia, ce que l'on recoupe sous le nom de *synchronisation temporelle*. Certains langages multimédias n'indiquent néanmoins pas explicitement la synchronisation, qui doit alors être déduite implicitement du modèle ou, le plus souvent, de l'application de présentation.

Aujourd'hui, les présentations multimédias doivent permettent à l'auteur de préciser la synchronisation qu'il désire et d'autre part assurer que l'exécution sera conforme à ses vœux, quelle que soit l'application de présentation envisagée. Chaque outil, application ou langage, multimédia définit des constructions et offre des fonctionnalités à l'auteur pour qu'il édicte la synchronisation de sa présentation. Ces concepts définissent un modèle de synchronisation, parfois de manière implicite, qui permet de fixer leur cadre.

Nous nous attacherons ici à examiner tout d'abord les caractéristiques de la synchronisation dans les différents outils multimédias pour ensuite classer les synchronisations selon leur modèle. Nous traiterons ensuite des modèles temporels qui déterminent la vision sous-jacente du temps dans l'outil.

3.3.1 Les caractéristiques de la synchronisation

La synchronisation comporte un certain nombre d'aspects découplés que l'on peut regrouper en plusieurs familles. Trois d'entre eux sont récurrents dans les outils multimédias: l'aspect intra- ou inter-médias, les formes d'indéterminismes et le contrôle d'accès.

3.3.1.1 La synchronisation intra-média ou inter-médias

La synchronisation *intra-média* ne s'applique qu'à une seule donnée média et indique la manière de présenter ses unités d'information élémentaires, c'est-à-dire les caractères pour le texte, les échantillons pour le son et la trame pour une animation. Elle est le plus souvent indiquée au niveau du format de stockage des données médias, mais peut aussi être externe à celle-ci. Le rythme, ou vitesse, de présentation de ces unités d'information est un des premiers paramètres que l'on peut définir. Ce rythme peut être constant tout le long de la donnée ou bien varier afin d'introduire certains effets sur des portions de la donnée média ou de minimiser la taille des informations stockées. Des relations plus complexes peuvent être mises en place entre les unités d'information. C'est par exemple le cas des images MPEG vues précédemment dont certaines peuvent être codées par rapport à d'autres. Ces relations peuvent s'appuyer sur les éléments logiciels qui produisent les unités d'information, comme c'est dans le cas du standard MPEG-4 qui offre des modèles des tampons où sont stockés les données et permet de prendre en compte les temps de transfert.

La synchronisation *inter-médias* précise les relations entre plusieurs données médias. C'est celle qui définit à proprement parler le contenu de haut niveau d'abstraction, ou scénario, de la présentation multimédia. Elle peut s'exprimer à divers niveaux de détails, ou granularité, à la fois sur des entités de taille plus ou moins fine et à travers des relations qui permettent de donner plus ou moins de précision.

La synchronisation dite *fine* lie des données médias à l'échelle de leurs unités d'informations. C'est le mécanisme que l'on retrouve au cœur des vidéos sous le nom de *lip synchronisation*. A ce même niveau de détail, la synchronisation peut aussi admettre une précision beaucoup moins grande, par exemple dans un contexte distribué lorsque les données médias arrivent à des instants imprévisibles. Pour des valeurs importantes du délai entre deux unités d'information, la synchronisation n'est tolérable que pour des données textuelles ou graphiques. Il est par exemple possible de préciser des rôles maîtres et esclaves des diverses données afin de choisir le comportement à adopter lorsque la synchronisation ne peut être respectée [129].

Il est aussi possible de ne considérer dans la synchronisation que des groupes d'unités d'information, appelée synchronisation *à grains fins*. Des points de synchronisation sont alors définis sur les diverses données médias et servent de points de rendez-vous lors de la restitution des données médias. La précision exigée est généralement peu importante du fait qu'une grande quantité d'unités d'information sont présentées entre deux points de synchronisation. Ce mécanisme correspond à une resynchronisation car après un point de synchronisation les données peuvent se désynchroniser jusqu'au prochain point de synchronisation où l'écart sera rectifié.

Enfin, la résolution la moins importante est atteinte au plus haut niveau de granularité lorsque la synchronisation inter-médias n'a lieu qu'entre les points caractéristique des données médias. Le début et de fin de la donnée média sont deux points caractéristiques à toutes les données média. Mais ces points caractéristiques peuvent correspondre à d'autres informations abstraites au sein des données média, comme le début d'une scène dans un film par exemple. Ce niveau de synchronisation est plus abstrait que les précédents et permet de spécifier le comportement de la présentation multimédia d'un point de vue global. C'est cette spécification de la synchronisation qui recouvre traditionnellement le terme *scénario*.

3.3.1.2 Le traitement de l'indéterminisme

L'auteur de présentations multimédias spécifie la synchronisation en fournissant des informations qui peuvent parfois être partielles. Les diverses sources d'indéterminisme constituent à la fois un avantage, en donnant de la latitude à l'auteur dans la spécification, et un inconvénient, en obligeant le système

d'exécution à gérer des informations incomplètes. Nous allons voir que chacune de ces sources oblige à des traitements particuliers, afin de permettre de compléter les informations ultérieurement.

Instants de début et de fin indéterminés

La synchronisation s'appliquant aux données médias, la première source d'incertitude les concerne. Ces données peuvent avoir un contenu dont les points limites de début et de fin ne sont pas connus avant leurs instants d'occurrence lors de l'exécution de la présentation multimédia. On parle souvent de données de durées inconnues, pour indiquer le fait que même si la synchronisation impose un instant de début, il est impossible d'avoir des informations sur la fin avant qu'elle ne survienne. Ces occurrences dépendent d'un facteur extérieur au système d'exécution qui peut être:

- l'exécution d'un autre système
  c'est le cas par exemple des données envoyés à la volée par les systèmes de vidéo à la demande [29] ou les requêtes dont les réponses sont envoyées par des bases de données [103];
- l'interaction avec un utilisateur
  une donnée particulière encapsule généralement cette interaction, comme un bouton ou une liste déroulante.

Afin de prendre en compte cette forme d'indéterminisme, la synchronisation inter-médias doit être possible et permettre la spécification relative des données médias. Le système d'exécution peut alors maintenir cette relation à l'exécution lors des occurrences des points limites. Cette forme d'indéterminisme se résume en fait à exprimer un délai indéterminé sur l'occurrence d'un instant [41]. Cet écart peut être majoré afin de limiter la source d'indéterminisme [140] et empêcher qu'une attente infinie ne bloque l'exécution d'une présentation multimédia ou permettre une vérification statique que la synchronisation ne comporte pas de problèmes [134].

Prise en compte des délais système

Etant donné l'intérêt qui a été porté par la recherche dans les systèmes et les réseaux au multimédia, de nombreuses caractéristiques de ces systèmes ont été intégrées aux méthodes de synchronisation [140, 14]. Ces caractéristiques englobent toutes sortes de délais existant à divers niveaux de la circulation d'information à travers les systèmes rentrant en jeu dans l'exécution de la présentation multimédia.

Un grand nombre de systèmes multimédias postulent le synchronisme des actions et de leurs réponses au sein du système, en considérant comme minimes les délais de réaction. Ils fournissent alors un service au mieux des ressources disponibles [117], la puissance des ordinateurs et la rapidité des réseaux réduisant aujourd'hui ces délais à des valeurs faibles.

Les recherches dans les domaines des systèmes et des réseaux se placent plutôt sous l'hypothèse de l'asynchronisme entre actions et réponses, tentant de modéliser les écarts entre eux. Ceci permet de définir et garantir une qualité de service sur les délais d'un endroit à un autre du système. Ces garanties temps-réels correspondent à des contraintes strictes dépendantes des données médias [14]. Les deux principales catégories de délais intervenant sont:

- les délais de transmission sur le réseau de communication [135] ou bien d'accès aux unités de stockage [29];
- les délais de traitement des informations, comme ceux dûs à l'usage de tampons [88] ou liés à des procédures implantées matériellement [37].

3.3.1.3 Contrôle d'accès

L'exécution des présentations multimédias se fait par le biais d'un système sur lequel l'utilisateur garde le plus souvent le contrôle. Inspiré la plupart du temps des systèmes de rendu des vidéos, ces contrôles d'accès permettent de stopper la présentation ou bien de forcer son exécution à partir d'un endroit indiqué.

Une autre possibilité pour le contrôle d'accès est l'exécution à l'envers de la présentation, possibilité qui est soulevée par peu de travaux étant donné que peu de players implémentent cette fonctionnalité pour les médias temporels. Elle exige de préciser la spécification de la synchronisation inverse d'une présentation [99], ce qui n'est pas toujours possible. Dans le cas général, plusieurs interprétations d'une synchronisation inverse sont possibles: statique, comme si la présentation multimédia avait été enregistrée, ou bien de façon dynamique, en permettant de rejouer les interactions ou bien les opérateurs dynamiques dans le sens inverse de leur sémantique (lorsqu'elle peut être définie). Le contrôle d'accès est bien souvent laissé au système d'exécution qui doit, lorsque cela est possible, préciser l'interprétation de la synchronisation inverse. L'exécution à l'envers n'étant pas toujours possible pour tout format de média, des sémantiques de remplacement peuvent être envisagées, comme c'est le cas dans le module Time Manipulation du langage SMIL.

3.3.2 Les modèles de synchronisation

Les relations de synchronisation définissent un cadre de description de la synchronisation des présentations multimédias que l'on appelle modèle de synchronisation et qui édicte les règles la régissant. Il ne faut pas confondre ce modèle avec le modèle temporel, qui s'attache à décrire la sémantique des relations de synchronisation et donc explicite la vision du temps sous-jacente. Le modèle temporel sert à exprimer l'interprétation des présentations multimédias exprimées dans le modèle de synchronisation. Le modèle de synchronisation est orthogonal à celui des données, qui permet lui de définir les structures abstraites de représentation des informations manipulées par le système multimédia. L'entité de synchronisation du modèle de synchronisation est la partie des structures

de données sur laquelle les relations temporelles s'appliquent. Les deux entités de synchronisation les plus utilisées sont les instants et les intervalles que nous étudierons tour à tour, puis nous présenterons quelques modèles de synchronisation hybrides.

3.3.2.1 Les modèles à instants

Ces modèles manipulent la notion d'instant, ou point temporel, c'est-à-dire un morceau de temps qui ne peut être décomposé [66]. Tout instant possède une date qui indique sa position absolue dans le temps et est utilisée afin d'ordonner les instants. Les valeurs des dates peuvent être exprimées dans divers systèmes de mesure du temps, par exemple *SMPTE time* (*Society of Motion Picture and Television Engineers*) [23], qui sont tous isomorphes à l'ensemble des entiers naturels ou des réels. L'ajout de relations entre les instants oblige à s'abstraire de leur date d'occurrence et l'on parle alors plutôt de points temporels.

Timeline

Le plus simple modèle à instant est le modèle *timeline* qui consiste à placer les instants sur un axe de temps en leur affectant une date fixe. C'est le modèle qui forme généralement la base des applications de composition multimédia, comme toute la famille d'applications de la société Macromedia [143]. Il est intuitif et simple à mettre en œuvre puisque le système d'exécution peut suivre simplement la ligne de temps pour respecter la synchronisation. On retrouve ce modèle d'autre part dans le domaine des réseaux sous forme d'estampille [135]. Ce modèle ne permet pas la synchronisation relative des données médias. En effet, la coïncidence d'instants peut être voulue mais ne peut être imposée car le déplacement d'un instant n'entraîne aucune modification de l'autre.

Timeline arborescent

Le modèle timeline peut être étendu en un timeline arborescent [78]. Les branchements entre différents timeline correspondent à des choix qui sont liés à une action de l'utilisateur et permettent de rendre une certaine forme d'interactivité par l'usage d'alternatives. Dans ce modèle, les timelines possèdent des points de début et de fin. Ces points peuvent être assignés à un point de branchement indiqué par un symbole de choix, un type de lien synchrone (déclenchement immédiat) ou asynchrone (un délai d'attente de l'action utilisateur est permis) et une limite maximale sur le nombre d'occurrence de ces points. Ce modèle conserve l'intuition du modèle timeline et l'augmente en permettant un forme d'interactivité contrôlée.

Réseaux de points temporels

Dans Firefly [22], un réseau de points est construit en indiquant une des relations *simultaneous with* ou *before by X*, X étant l'écart qui doit exister entre deux points. Les points peuvent être en début ou en fin de données médias ou bien être internes. L'étendue entre le début et la fin est un arc qui peut être

contraint à travers trois valeurs, minimale, maximale et optimale. Des coûts associés à l'extension ou la réduction des durées permettent de fixer des préférences entre ces trois valeurs. Ces valeurs, tout comme l'occurrence des points, peuvent être indéterminées afin d'indiquer qu'elles dépendent de conditions qui ne seront résolues que lors de la phase de présentation. Un formattage de la spécification Firefly est nécessaire afin de déterminer des valeurs de l'ensemble des éléments qui respectent les contraintes spécifiées par le scénario. Un algorithme de programmation linéaire permet de trouver des valeurs lorsqu'il en existe, en minimisant les coûts. Les contraintes indéterministes sont mises à part et prise en compte lors de la présentation uniquement.

3.3.2.2 Les modèles à événements

Un point de vue différent sur les points temporels est apporté par la notion d'événement. C'est une notion qui est née de l'abstraction de fonctionnalités matérielles, comme les interruptions, mais qui s'en démarque aujourd'hui. Un événement est instantané et est une occurrence dans le temps [117] observable potentiellement plusieurs fois. Il peut être prédictible ou non, selon l'information à laquelle il est rattaché. Une de leur caractéristique fondamentale est qu'ils sont observables. Ils ont parfois des structures typées qui permettent de leur associer des informations additionnelles. Divers mécanismes leur sont associés, d'une part leur gestion et d'autre part dans les conséquences de leur observation. Les nombreux modèles à événements jouent sur ces différents aspects afin de fournir des relations de haut niveau d'abstraction tout en conservant l'intuitivité procurée par la notion d'événement.

Langages de scripts

Ces langages impératifs sont associés à des langages déclaratifs qui leur fournissent la notion d'événement. Ils définissent des couples événement-action (*event handler*) afin de lier les événements entre eux et spécifier la synchronisation des données du document déclaratif. Ces langages étant peu complexes, il est possible de les interpréter au fur et à mesure de l'exécution de la présentation multimédia en laissant à un gestionnaire d'événements le soin d'appliquer les relations spécifiées. Chaque langage de script peut choisir entre des relations événement-action synchrones, c'est-à-dire qu'une seule relation peut être exécutée à un instant donnée, ou bien asynchrones, plusieurs événements pouvant être observés à tout instant.

On retrouve souvent dans ces langages la même hiérarchie d'événements, les noms des événements indiqués provenant de divers langages:

- les événements liés au mécanisme de présentation
  `onLoad` (chargement d'un élément du document), `onReload` (rechargement), `onLeave` (changement d'élément), `onFocus` (l'élément gagne le focus);
- les événements liés aux actions temporelles
  `onFinish` (l'élément se termine);

– les événements liés à la structure du document
`new` (la création de la structure de donnée associée à l'élément est terminée), `change` (un champ de la structure de l'élément a changé);
– les événements utilisateurs
`onClick` (clic sur un élément), `onDrag` (déplacement d'un élément), `keyDown` (appui sur une touche).

Les langages *OpenScript* du logiciel *Asymetrix Toolbook* [23], *LINGO* du logiciel *Macromedia Director* [118] et *ActionScript* de la plate-forme *Flash* [143] sont trois exemples de langages de script:

OpenScript	LINGO	ActionScript Flash

`to handble buttonDown`
`  send stream1`
`end`
	`on new me`
	`  set ancestor =`
	`  new("example")`
	`end`
		`on (press) {`
		`  goto(200);`
		`}`

NSync

NSync [10] est un langage de définition de synchronisation qui permet de spécifier des contraintes sous la forme:

`When {expression} {action}`

`expression` est une combinaison d'expressions sur des variables temporelles (vitesse, instant courant, etc.) ou non-temporelle (nombre, booléen, etc.) grâce aux opérateurs `&&` et `||`. L'`action` est une suite de commandes du langage *Tcl*, le langage utilisé pour implémenter NSync. La contrainte signifie que lorsque l'expression passe de la valeur fausse à la valeur vraie alors l'action est invoquée. La résolution de ces contraintes est prise en charge par des composants spécifiques du système de gestion de la présentation. Le composant *Evaluator* met en place une logique prédictive à quatre valeurs, *TRUE*, *FALSE*, *WBT(t)* et *WBF(t)*, les deux dernières valeurs indiquant qu'une expression deviendra vraie ou fausse après le délai t. Des tables de vérité sont données et permettent de réévaluer les expressions lorsque le composant *Change Monitor* détecte un changement possible des valeurs. Après calcul de la nouvelle valeur, si l'expression passe de *FALSE* à *TRUE*, l'action est invoquée. Si elle vaut *WBF(t)*, l'action est programmée pour s'exécuter après un délai t. Les valeurs *FALSE* et *WBF(t)* ne provoquent rien, mais *WBF(t)* permet de prédire qu'aucune action provoquée par cet expression ne pourra avoir lieu avant le délai t.

FLIPS

Le système *FLIPS* (*FLexible Interactive Presentation Synchronization*) [136] utilise deux relations entre événements. $a \dashv b$ indique que a est une *barrière* (*barrier*) pour b et signifie que a ne peut survenir après b. $a \rightarrow b$ indique que a est un *déclencheur* (*enabler*) pour b et signifie que b doit survenir dès que a survient. La relation de barrière est indéterministe car la notion le délai entre les

deux événements ne peut être désigné. L'association des deux relations permet d'obtenir des relations déterministes:

- $a \dashv b \ \wedge \ b \dashv a$ et $a \rightarrow b \ \wedge \ b \rightarrow a$
  expriment de deux manières différentes la même relation: le premier des deux points a et b qui survient cause l'autre;
- $a \dashv b \ \wedge \ b \rightarrow a$ et $a \rightarrow b \ \wedge \ b \dashv a$
  expriment deux relations symétriques: b (respectivement a) est maître de a (respectivement b), dans le sens où s'il est le premier à survenir il déclenche l'autre, sinon rien de particulier n'est imposé.

3.3.2.3 Les modèles à intervalles

Une façon différente de concevoir la synchronisation est apparue avec les travaux de Hamblin [66] puis Allen [7]. L'entité temporelle élémentaire manipulée par la synchronisation est ici un *intervalle* qui est défini comme une étendue ou portion de temps. Il est le plus souvent représenté par un intervalle numérique, celui entre les dates des instants de début et de fin de l'intervalle. Les intervalles sont un autre paradigme de synchronisation qui reposent sur une intuition différente du temps.

Logiques d'intervalles

Les logiques d'intervalles, comme l'*interval temporal logic* [52] ou bien son extension le *duration calculus* [36], abstraient clairement la notion d'intervalle en la définissant par un axiome mathématique de décomposition:
un intervalle est une entité qui peut être décomposée en deux sous-intervalles liés par un opérateur de coupure (*chop* noté $\frown$ ou ;) qui représente leur adjacence. Ces logiques permettent d'élaborer des formules correspondant à une synchronisation désirée. Mais ces formules sont trop précises et complexes pour servir de base à une spécification de la synchronisation compréhensible par un programmeur. Ces logiques sont plutôt utilisées afin de capturer les propriétés temporelles du modèle de synchronisation, comme dans [14] et [29], ce qui permet d'utiliser des techniques de *model checking* afin de vérifier la validité des systèmes multimédias. D'autres logiques d'intervalles existent, comme *MetricInterval Temporal Logic* [74], et font l'objet d'axiomatisations différentes [138].

A noter que la décomposition des intervalles n'est en pratique plus possible au-dessous d'un certain niveau de précision, ce qui limite l'applicabilité de ces logiques. La plupart des modèles à intervalles considèrent l'intervalle correspondant à une donnée média comme entité atomique, c'est-à-dire non-décomposable dans le modèle. Une granularité plus fine n'est généralement utilisée que dans le cadre de la synchronisation fine.

Relations d'Allen

Les *relations d'Allen* [7], citées mais non définies dans [66], sont les relations sur les intervalles qui sont probablement les plus célèbres. De nombreux travaux

non-informatiques ont utilisé ces relations afin de modéliser des phénomènes naturels et l'informatique ne dément pas cette tendance [50, 55, 107, 92].

Les treize relations d'Allen décrivent tous les positionnements possibles entre deux intervalles: *before*, *meets*, *overlaps*, *during*, *starts*, *finishes*, les six relations inverses des précédentes et la relation *equal*. Ces relations permettent de spécifier la synchronisation de manière qualitative sans devoir indiquer tous ses détails, comme par exemple les valeurs des écarts entre deux intervalles dans la relation *before*. L'ensemble des disjonctions de relations d'Allen forment une algèbre dénommée algèbre d'intervalles (*Interval Algebra* ou *IA*) [151, 154]. D'autre part, Little et Ghafoor [99] généralisent à un nombre quelconque d'arguments les relations d'Allen et définissent leurs inverses.

Les relations d'Allen s'adressent particulièrement aux tâches de raisonnements dans lesquelles on cherche à déduire des informations d'une spécification de la synchronisation, comme par exemple l'ensemble de toutes les contraintes satisfaisables entre des intervalles du scénario (appelé clôture). Etant donné que les relations d'Allen modélisent toutes les relations possibles entre deux intervalles, un scénario à base de relations d'Allen peut être incohérent, c'est-à-dire qu'il n'existe pas d'intervalles satisfaisant ce scénario. La vérification de cohérence est une tâche NP-difficile [59], ce qui signifie qu'elle est plus coûteuse à résoudre que tout problème se résolvant en temps polynomial. Villain et Kautz [151] s'intéressent à la sous-algèbre SA de l'algèbre IA qui comprend les disjonctions de relations d'Allen pouvant être traduites en disjonctions de relations sur les points limites des intervalles, dont la liste est dressée dans [149]. L'algorithme de calcul de la clôture d'un ensembles de relations de SA a pour coût $O(n^2)$ [149, 148] et permet de déterminer si un scénario à base de relations d'Allen est incohérent.

Relations de Wahl-Rothermel

Les relations d'Allen sont insuffisantes pour ce qui concerne les tâches de planification mises en jeu dans l'exécution des présentations multimédias car elles ne permettent de spécifier qu'une relation qui a eu lieu dans le passée et non d'indiquer une relation entre intervalles dans l'avenir. Wahl et Rothermel [154] proposent de considérer des disjonctions de relations d'Allen afin de modéliser les incertitudes sur les relations entre intervalles. Ils dégagent un jeu de vingt-neuf disjonctions de relations d'Allen qui sont représentables en terme des relations sur les points $\{<, =, >, ?\}$, la relation ? correspondant à la disjonction des trois relations sur les points "$< \vee = \vee >$". Ces vingt-neuf relations sont modélisées par un jeu de dix opérateurs qui prennent comme arguments deux intervalles et de un à trois délais indiquant les écarts entre différents points limites des intervalles. Des contraintes de validité des opérateurs lient les connaissances des durées des intervalles aux valeurs des délais des opérateurs. L'outil Madeus [92] intègre un algorithme incrémental de vérification de la cohérence des scénario à base de relations de Wahl-Rothermel et un algorithme de formatage du scénario

afin d'obtenir des valeurs d'intervalles lui correspondant. Ces opérateurs étaient esquissés par Little et Ghafoor [99] et sont présentés en table 3.2.

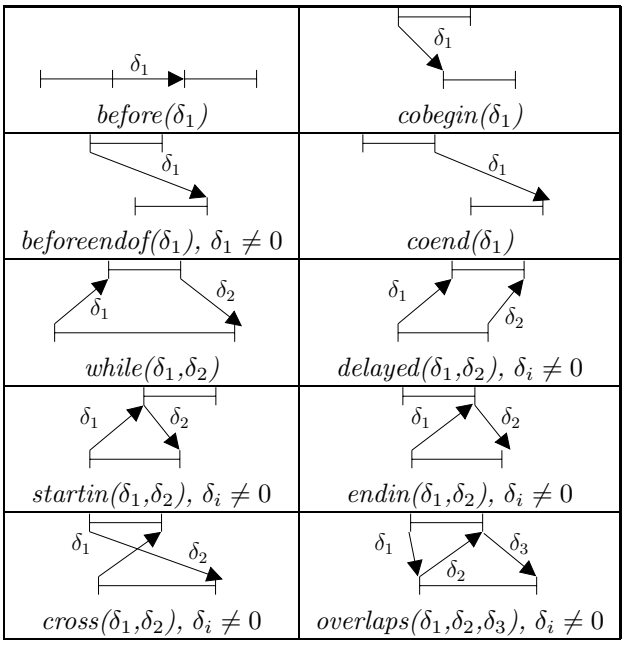


FIG. 3.2 – *Les relations de Wahl-Rothermel*

Modèles de réseaux de Petri temporels

Little et Ghafoor [98] s'inscrivent dans un modèle à intervalles en définissant *OCPN* (*Object Composition Petri Net*). Leur idée est d'utiliser une structure de réseau de Petri temporel afin de modéliser la synchronisation d'intervalles de durée fixe et connue. Les places des réseaux de Petri temporels correspondent à des intervalles ou à des délais et les transitions à des points de synchronisation. Les jetons sont actifs lors de la présentation de l'intervalle et deviennent inactifs à la fin de celui-ci. Une transition est activée lorsque toutes ses places en entrée contiennent un jeton inactif, elle enlève alors ces jetons et introduit un jeton dans chacune de ses places en sortie. OCPN permet de décrire toutes les relations d'Allen et offre un formalisme graphique de spécification de la synchronisation.

Diaz et Sénac définissent des réseaux de Petri de flux temporels *TSPN* (*Time Stream Petri Net*) [47]. Leur idée est de placer les intervalles sur les arcs et d'assigner à chaque transition une règle de déclenchement. Les intervalles, dits de validité temporelle, sont spécifiés selon leur durée minimale, nominale et maximale. Les règles de déclenchement se basent sur trois stratégies de base: au plus tôt des arcs, au plus tard des arcs ou selon les arcs maîtres. La combinaison

de ces stratégies produit neuf règles qui précisent les priorités des différents arcs dans le déclenchement de la transition. Les réseaux de Petri TSPN permettent de modéliser des scénarios indéterministes qui prennent en compte des délais sur les intervalles de manière complexe. *HTSPN* (*Hierarchical TSPN*) [158] ajoute la possibilité d'encapsuler des places dans des places composites et *I-HTSPN* (*Interpreted HTSPN*) [157] attache une sémantique particulière aux places afin de modéliser les structures conceptuelles, de contenu et de présentation des documents. Les I-HTPSN ont été adaptés aux langages MHEG et Java.

Bien que possédant une représentation graphique et des bases formelles, les réseaux de Petri temporels ne permettent pas de spécifier aisément la synchronisation des présentations multimédias. En effet, la sémantique derrière les symboles graphiques utilisés est complexe à saisir. Elle nécessite de plus des adaptations particulières afin d'intégrer des formes d'indéterminismes.

Video Algebra

La *Video Algebra* [156] définit une algèbre de synchronisation des intervalles, les intervalles correspondant ici à des segments de vidéos. La composition des intervalles est ici fonctionnelle, un opérateur de synchronisation temporelle prenant comme argument des intervalles et retournant un nouvel intervalle de type composé qui peut être à son tour synchronisé avec d'autres intervalles. La structure des présentations multimédias est isomorphe à un arbre et, contrairement aux modèles découlant des relations d'Allen, ne nécessite pas de vérification de la cohérence temporelle. La sémantique des sept opérateurs temporels est intuitive et permet de traduire le scénario en un format directement exécutable. Certains opérateurs temporels de la Video Algebra rappellent des structures de contrôle des langages de programmation classique, comme la boucle *loop* ou le test conditionnel $(test)?E_1 : E_2 : \ldots : E_n$. Des opérateurs non-temporels permettent d'annoter les vidéos et construire des requêtes sur ces annotations. Ce travail est étendu par *Menkalinan* que nous présentons en section 3.8.6.

3.3.2.4 Les modèles hybrides

Il est difficile d'unifier les notions d'instants et d'intervalles car plusieurs problèmes difficiles se posent lorsque l'on considère les intervalles comme des événements de durée non-nulle. D'une part, il est difficile de dire si les événements de durée nulle aux limites d'un événement de durée non-nulle lui appartiennent [7]. D'autre part, leurs propriétés de densité sont opposées: on peut toujours décomposer un intervalle en deux intervalles consécutifs alors qu'un événement n'est pas décomposable en plusieurs événements. La relation de composition entre événement de durée non-nulle et points complique alors la sémantique du modèle [66]. Il est néanmoins possible de manipuler à la fois des instants et des intervalles au sein du même modèle grâce à des représentations particulières.

Multimedia Representation Graph

[114] présente un modèle général de synchronisation nommé *Multimedia Re-*

presentation Graph (*MRG*). Les nœuds du graphe sont des éléments médias (*media elements*) qui peuvent contenir des données à différentes granularités, ce qui offre des possibilités de modélisation à divers niveaux. Les arêtes représentent l'ordre des éléments médias en étant à la fois une transition causale et aussi un écoulement du temps. Les arêtes peuvent être pondérées par des probabilités et former, entre autre, des chaînes de Markov. Bien qu'initialement dédié à l'organisation d'informations textuelles, ce modèle permet de manipuler instants et intervalles. De nombreuses opérations sont définies sur les MRG afin de modéliser la corrélation de diverses données médias, par exemple l'alignement du son avec ses sous-titres. Néanmoins ce modèle n'explicite pas complètement la sémantique des diverses relations, ce qui rend son usage difficile.

Connecteurs hypermédias

[109] définit un langage d'édition qui repose sur la notion de *connecteur hypermédia*. Le modèle utilise une structure de *composant* qui peut être défini de manière composite et est relié à un connecteur hypermédia par le biais d'une liaison, comme illustré en figure 3.3. Chaque événement correspond à une machine à état et possède des variables internes. Un connecteur hypermédia lie un nombre quelconque d'événements en attribuant à chacun d'eux des rôles et en indiquant leurs interactions à travers un attribut *glue*. Le rôle d'un événement peut être de deux types: condition ou action. Un rôle condition correspond à un événement en entrée du connecteur hypermédia et définit les conditions qui doivent être successivement obtenue pour que l'événement participe à l'exécution du connecteur. Un rôle action correspond à un événement en sortie du connecteur et indique quelles transitions effectuer sur cet événement. L'attribut *glue* combine les rôles conditions par des connecteurs logiques et peut appliquer un opérateur de retard $\oplus$ afin de définir les délais entre deux événements. Cet attribut combine aussi les rôles actions par deux relations temporelles, $\rightarrow$ pour la séquentialité et $|$ pour le parallélisme. Lorsque les événements en entrée sont activés, la *glue* du connecteur hypermédia indique quelles actions effectuer sur les états des événements en sortie.

La structure des connecteurs hypermédias est proche de celle des réseaux de Petri temporels, à la différence que le connecteur hypermédia est plus riche qu'une transition puisqu'il permet à la fois de composer des événements de manière causale et de définir des liens hypermédias. L'organisation des événements en machines à états ouvre de plus des perspectives de synchronisation différentes.

DAMSEL

Le langage *DAMSEL* [116, 115] s'appuie sur la notion d'événement mais il permet de les composer entre eux: ainsi, étant donné deux événements a et b, $a > b$ constitue l'événement qui surviendra lorsqu'une occurrence de a surviendra après une occurrence de b. Les événements sont regroupés en catégories: produits par le système d'exécution ou de l'utilisateur, dépendant d'une condition

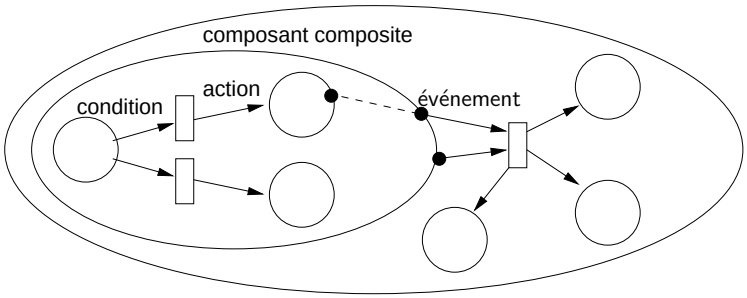


FIG. 3.3 – *Modèle de connecteurs hypermédias*

logique ou provenant d'en dehors du système DAMSEL. Les événements sont étendus à des durées non-nulles, modélisant ainsi des intervalles. Trois règles sont alors applicables sur les événements sous forme de prédicats logiques: l'*activation* d'un événement par un autre selon un comportement permettant d'indiquer à l'ordonnanceur la manière de prendre en compte cette activation; le *report* d'un événement hors d'un certain intervalle, qui réordonne un événement qui arriverait pendant cet intervalle à la fin de cet intervalle; et la synchronisation fine de deux intervalles, qui est un cas particulier d'activation. D'autre part, DAMSEL envisage d'user d'une logique temporelle (*Conditional Temporal Logic* et logique causale) afin de contraindre un ordonnancement particulier d'événements dans le scénario. La synchronisation dans DAMSEL peut amener à des conflits dont la résolution peut être indiquée dans la spécification et possède deux modes, statique (à la compilation) ou dynamique (à l'exécution).

Algèbres de processus

Les algèbres de processus (*process algebra*) constituent des modèles complémentaires des logiques d'intervalles auxquelles elles sont parfois associées dans la définition des systèmes multimédias [14]. Ces techniques de description formelle (*FDT*, *Formal Description Technique*) sont initialement prévues pour spécifier l'ordonnancement des processus entrant en jeu dans une application. *LOTOS* (*Language of Temporal Ordering Specification*) [44, 42] est une de ces algèbres de processus et est standardisée par l'organisme *ISO*. *RT-LOTOS* [41] est une extension de LOTOS aux applications temps-réel dont l'ajout principal est l'opérateur *delay(t)* qui permet de fixer une durée de manière absolue. Ce langage n'est pas utilisé directement pour spécifier la synchronisation, bien que cela soit possible mais peu intuitif. Il sert comme format intermédiaire afin de prouver de manière automatique la cohérence temporelle de présentations multimédias exprimées dans un autre langage, comme SMIL [134, 133].

Les algèbres de processus font apparaître que la synchronisation des données médias peut être envisagée comme celle des processus qui les présentent. On retrouve la dichotomie application/document évoquée en section 3.1. Les algèbres

de processus peuvent être utilisées pour spécifier des systèmes multimédias utilisant à la fois des instants et des intervalles. Par exemple dans la *Constraint Library* [40], le langage RT-LOTOS est utilisé pour fournir une bibliothèque de spécifications qui permettent d'organiser:

- des événements, selon des relations d'égalité (simultanéité ou bien écart fixe) ou d'inégalité (précédence avec une contrainte sur la valeur de l'écart qui peut être inconnue, supérieure et/ou inférieure à une valeur donnée);
- des intervalles, selon des relations de parallélisme (indiquant si la composition se termine lorsque la première, la dernière ou bien la présentation maître se termine) ou une des ving-neuf relations de Wahl-Rothermel.

3.3.3 Les modèles temporels

La référence au temps est le plus souvent cachée dans la sémantique des relations temporelles. Les concepts pour exprimer cette sémantique constituent le modèle temporel sous-jacent à la synchronisation. Le modèle temporel abstrait à son tour la représentation du temps tel que le système d'exécution l'utilise. Cette représentation est, au niveau le plus bas du système informatique, l'horloge qui cadence les instructions du processeur et ordonne séquentiellement les bits traités. A l'opposé, les systèmes multimédias se situent dans des couches logicielles généralement de haut niveau et organisent les données de manière parallèle.

De nombreux outils de raisonnement et d'organisation utilisent en fait un modèle temporel, sans spécifier de synchronisation. Ils décrivent des informations déjà présentées ou bien statiques de manière à pouvoir les analyser, comme c'est le cas dans les systèmes de gestion de bases de données [160, 2], dans des systèmes de raisonnement [13] ou bien dans des modèles musicaux [67].

3.3.3.1 Les modèles à instants

Nous avons vu que les valeurs d'instants sont des dates qui se ramènent à deux ensembles:

- les entiers naturels $\mathbb{N}$
  Le temps est alors *discret*, dans le sens où la représentation des valeurs des dates est une succession de points entre lesquels il n'y a aucun instant. Tous les systèmes de datation, comme *SMPTE time* ou *UTC* (*Coordinated Universal Time*), peuvent être convertis en des valeurs entières. Les estampilles temporelles (*timestamps*) sont aussi des dates, utilisées dans de nombreuses recherches sur les systèmes [8] et les réseaux [135, 77].
- les réels $\mathbb{R}$
  Le temps est alors *dense*, on dit aussi *continu*. La représentation des valeurs des dates est une ligne continue d'instants, de sorte qu'entre deux instants distincts il est toujours possible de trouver un nouvel instant.

Les dates sont comparées par le biais des trois opérateurs numériques usuels $<$, $=$ et $>$, qui couvrent tous les cas de figure. Ces opérateurs de comparaison font l'objet de définitions spécifiques selon l'ensemble des valeurs choisies [52]. L'addition et la soustraction de valeur numériques sont utilisées afin de définir des équations entre valeurs de dates. Afin d'indiquer des relations incomplètes, des disjonctions d'opérateurs de comparaison simples sont utilisées. Ces disjonctions de relations se résument à l'ensemble $\{\emptyset, <, \leq, =, \geq, >, \neq, ?\}$ avec:

$$\begin{array}{lcl} \emptyset & \equiv & \text{absence de relations} \\ \neq & \equiv & \{<, >\} \\ ? & \equiv & \{<, =, >\} \end{array}$$

[154] s'intéresse aux modèles cohérents, c'est-à-dire qui permettent de spécifier des situations possibles, et éliminent de ce fait la relation $\emptyset$. Les auteurs ajoutent que les systèmes multimédias peuvent tolérer de faibles imprécisions car les utilisateurs ne peuvent les percevoir. Ceci implique que les couples de relations $(<, \leq)$, $(\neq, ?)$ et $(>, \geq)$ peuvent être confondus. Finalement les relations pertinentes pour les systèmes multimédias se réduisent à l'ensemble $\{<, =, >, ?\}$.

Les modèles causaux

Ces modèles temporels reposent sur une relation particulière entre les instants. La *causalité* [108] est une relation entre points temporels qui possède une direction, c'est-à-dire qu'elle est antisymétrique et irréflexive. Le point en amont de la relation causale est appelé *cause* et celui en aval *conséquence*, ou encore *effet*. C'est une notion qui provient de considérations originellement philosophiques [147] autour des causes, des conséquences et des actions. La relation de causalité s'oppose à celle de simultanéité temporelle $=$ et implique celle de précédence temporelle $\leq$. Elle peut exprimer la cause directe ou bien son contraire, c'est-à-dire qu'un point ne peut être la cause directe d'un autre [108]. On retrouve ici les déclencheurs et les barrières de FLIPS [136].

Une relation causale peut posséder une condition qui restreint son application au moment où la cause survient, comme pour les connecteurs hypermédias [109]. L'interprétation générale de la relation de causalité est que lorsque la cause survient, si la condition est vérifiée, la conséquence doit être appliquée selon le type de causalité utilisée.

On retrouve cette notion dans les liens hypertemps [130] qui sont l'extension naturelle des liens hypertextes au domaine du temps. Ceci permet d'intégrer de manière cohérente la navigation, au sens des choix donnés à l'utilisateur d'accéder aux diverses parties de la présentation multimédia, dans la spécification temporelle.

3.3.3.2 Les modèles à intervalles

Les valeurs des intervalles peuvent être représentées de deux manières:
– deux instants, correspondants au début et à la fin

ces instants se ramènent au modèle précédent avec la contrainte inhérente à la définition de l'intervalle: le début est avant la fin;
- un instant de début et une durée
  bien que pouvant être ramenée à la précédente, cette représentation apporte un plus haut niveau d'abstraction puisque la notion de durée est intrinsèquement relative.

Comme vu pour les modèles de synchronisation, les treize relations d'Allen permettent de décrire toutes les relations entre les intervalles. Tout comme pour les modèles temporels à instants, ces relations sont généralisées aux disjonctions de relations d'Allen. [154] explicite parmi les 8192 disjonctions de relations d'Allen vingt-neuf relations pertinentes, à partir des quatre disjonctions pertinentes de relations sur les instants. Notons que les disjonctions de relations sur les intervalles ont un plus grand pouvoir d'expression que les disjonctions de relations sur les instants. En effet, l'exclusion mutuelle de deux intervalles A et B, c'est-à-dire le fait que l'un est avant l'autre, nécessite une disjonction de relations sur les instants (fin de $A <$ début de $B \ \vee$ fin de $B <$ début de A).

Dans des contextes plus spécifiques, d'autres relations entre intervalles sont utilisées, comme la comparaison de leur durée dans la logique temporelle d'intervalles [52] ou bien leur inclusion dans le cadre d'un modèle spatio-temporel [13].

3.3.3.3 Les modèles à automates

Les techniques de descriptions formelles, comme Z et $LOTOS$ [14, 44], utilisent le plus souvent comme sémantique un modèle temporel d'automate. Un automate d'états fini est un graphe d'états fini dont les transitions sont étiquetées par les lettres d'un alphabet. Les automates temporisés [43] introduisent un nombre fini d'horloges dans les états et conditionnent les transitions en fonction des valeurs des horloges. Ce formalisme est directement exécutable [136] et permet aisément d'être augmenté afin de fournir de nouvelles abstractions ou fonctionnalités [51].

3.3.4 Formatage de la spécification temporelle

La spécification temporelle exprime le scénario de la présentation multimédia. Elle est l'expression de l'intention de l'auteur et le système d'exécution de la présentation multimédia doit assurer la synchronisation qui y est exprimée. Elle n'est parfois pas exécutable directement et nécessite des traitements particuliers afin de pouvoir être exécutée, traitements regroupés sous le nom de *formatage temporel* [92]. La synchronisation est traduite dans le modèle temporel sous forme d'un ensemble de contraintes qu'il faut résoudre, la solution apportant la forme exécutable recherchée. Le formatage est constitué principalement de trois activités qui sont le contrôle de cohérence et les formatages statiques et dynamiques. Il peut se décliner de six manières, selon qu'il s'effectue intégralement

de façon statique ou dynamique ou qu'il se décompose entre ces deux modes, avec une phase de contrôle de cohérence optionnelle.

Le contrôle de cohérence

S'il n'existe pas de solution au système de contraintes correspondant à la synchronisation, le scénario est incohérent et la présentation multimédia ne pas être exécutée. Le calcul de solution pouvant être une opération coûteuse, l'étape de contrôle de la cohérence permet, lorsque cela est possible, de déterminer les scénarios incohérents à moindre coût, par exemple en détectant les cycles dans la spécification. Le contrôle de cohérence peut toutefois être coûteux, d'un ordre de complexité de $O(n^2)$ pour les relations d'Allen [148, 59, 151], et susceptible de générer une explosion du nombre d'états dans le cas des réseaux de Petri [44]. Les modèles de synchronisation utilisant les opérateurs [156] ne sont pas sources d'incohérence temporelle car on peut prouver qu'ils construisent des intervalles qui sont, par hypothèse, des entités cohérentes. Il existe deux formes d'incohérence temporelle [93]: quantitative lorsque ce sont les informations quantitatives (durées, délais) qui provoquent une incompatibilité; ou bien qualitative lorsque c'est la synchronisation même qui est responsable de l'incohérence, par exemple spécifier que A est avant B et B avant A.

Le formatage statique, qui a lieu avant l'exécution

L'objectif est d'ordonnancer l'ensemble des instants de début et de fin des données, autant que faire se peut. Le processus de formatage statique déduit de la spécification une forme exécutable totalement ou partiellement, soit en calculant une solution totale ou partielle des contraintes de la synchronisation, soit en traduisant directement la spécification dans un format compréhensible par un système d'exécution. Cette étape se charge parfois de réduire les sources d'indéterminismes qui sont permises par l'outil lors de la spécification du scénario. Par exemple, [22] définit les durées sous la forme d'un triplet, chaque valeur possédant un poids qui exprime la priorité qui lui est accordée. Le formatage statique met alors en œuvre un algorithme de programmation linéaire afin de minimiser le coût total du graphe de spécification. Lorsque les durées des données sont fixées [78], les instants de début et de fin peuvent même être calculés avant l'exécution afin de ramener le scénario à une séquence d'actions.

Le formatage dynamique, qui a lieu lors de l'exécution

Si la solution calculée lors du formatage statique est partielle, il est nécessaire de procéder à des traitements au fur et à mesure que les données sont présentées. Par exemple, Little et Ghaffor [98] propagent les contraintes en appelant un algorithme d'ordonnancement sur les divers éléments de la composition lorsque ceux-ci surviennent. Layaïda [92] propage les durées afin de réajuster le graphe d'ordonnancement des points du scénario. La solution approchée est parfois la source de conflits qui peuvent être résolus en relâchant les contraintes de la spécification afin d'appliquer des techniques de programmation linéaire [2]. Le contrôle de cohérence n'étant pas toujours possible, il est parfois nécessaire de traiter les

conflits lors de l'exécution, comme dans le contexte de systèmes distribués [153]. La résolution des conflits consiste dans cette proposition à ajouter des délais (*local lag*) et réajuster les occurrences de certains instants (*timewarp*). Lorsque les conflits ne peuvent être résolu, par exemple parce que les ressources manquent, il y a *violation temporelle d'actions* [41] et des mécanismes de récupération d'erreurs peuvent intervenir, soit spécifié par l'auteur, soit implémentés par le système.

A noter que toute forme d'interaction prise en compte dans la spécification temporelle ne peut être résolue que lors de l'exécution de la présentation multimédia. Lorsque le modèle temporel utilise des relations qualitatives, les interactions sont automatiquement prises en compte. Par exemple, les relations causales sont actives jusqu'à ce que l'interaction ait lieu, dans la fenêtre temporelle de leur validité. Par contre, lorsque le modèle temporel utilise des relations quantitatives, des recalculs de valeurs sont généralement nécessaires lorsque les interactions ont lieu. Le même problème se pose lorsque la spécification temporelle prends en compte des paramètres dynamiques qui ne peuvent être connus que lors de l'exécution, comme le débit du réseau, la rapidité de l'ordinateur ou même les préférences de l'utilisateur. Dans tous ces cas, il est parfois possible d'envisager de dégrader la qualité des données [21] ou de les transcoder [19] afin de pallier les défaillances du système.

3.4 Spécification de la composition spatiale

La plus grand partie des données médias est aussi plongée dans l'espace, si l'on excepte l'audio non-spatialisée, et nécessite donc de préciser la composition spatiale de la présentation multimédia. Cette dimension des présentations multimédias a fait l'objet de moins d'études et de propositions que la dimension temporelle dans le domaine multimédia [30]. Cette dimension n'étant pas sujette à un écoulement intrinsèque, le problème d'ordonnancement des données est ici plus simple.

3.4.1 Objets spatiaux

La composition spatiale manipule cinq types d'objets, ou *formes*:

- les *points*
  Comme les instants pour le temps, les points sont des éléments spatiaux insécables.
- les *courbes*
  Les courbes continues sont des éléments spatiaux définis par un morphisme appliqué à une ligne. Une ligne est un ensemble de points compris entre deux points le long d'un axe passant par ces deux points. Les courbes non-continues regroupent des courbes continues qui n'ont pas de points communs.

– les *surfaces*
  Ces éléments géométriques sont construits à partir des courbes continues, de la même manière que ces dernières le sont à partir des points: les surfaces continues sont des ensembles de points compris "entre" des courbes continues, le mot "entre" prenant le sens d'être sur un axe passant par les points des courbes et entre ces points sur cet axe. Tout comme les courbes, les surfaces non-continues regroupent d'autres surfaces continues.
– les *volumes*
  Ils étendent encore d'une dimension les formes, les volumes continus étant compris entre des surfaces continues et les volumes non-continus étant un groupe de volumes continus.
– les *objets composites*
  Ces objets sont composés de plusieurs formes des catégories précédentes. Ces formes représentent de la façon la plus fidèle des objets réels mais sont bien trop complexes pour être utilisés dans une présentation multimédia.
– D'autres objets spatiaux existent, définit le plus souvent à partir de modèles ou de propriétés mathématiques complexes. Les objets fractaux ou bien les *nurbs* constituent deux classes de tels objets. Ces objets étant spécifique à certains domaines, ils sont peu répandus. De plus, la complexité de leur rendu rend difficile leur utilisation.

La plus grande partie des outils multimédias n'utilisent que des lignes brisées, c'est-à-dire des ensembles de lignes connectées par leurs extrémités, des polygones (surfaces entre des lignes) et des polyèdres (volumes entre des polygones). Les systèmes informatiques modernes ne permettant pas encore de traiter en temps-réel des formes plus complexes que des formes linéaires, des artifices doivent être utilisés pour rendre ces objets.

3.4.2 Modèles spatiaux absolu et relatif

Les modèles spatiaux étant moins nombreux que les modèles temporels, ils sont plus aisés à classer. Une classification simple revient à distinguer les modèles absolus des modèles relatifs.

3.4.2.1 Modèles absolus

Ce type de spécification spatiale repose sur une description de l'espace en terme de *référentiel*, c'est-à-dire un système de coordonnées à deux ou trois axes. Les axes n'ont comme seul lien le fait d'avoir une origine commune. Chaque axe a une direction et un ensemble de points. Les points spatiaux sur les axes peuvent avoir des valeurs dans les entiers naturels ou bien dans les réels et sont ordonnés par les relations d'ordre $<$, $=$ et $>$, avec la sémantique sous-jacente de l'ensemble de valeurs.

Un point est repéré dans l'espace par un ensemble de valeurs sur chaque axe appelé *coordonnée*. Il est possible de passer d'un référentiel à un autre en

traduisant les coordonnées des points, généralement par l'usage d'une matrice de conversion.

Les lignes brisées, polygones et polyèdres sont repérées par les coordonnées de leurs points caractéristiques. Les autres formes sont généralement définies à partir d'équations telles que les coordonnées de leurs points en sont solutions. Les courbes de Bézier font parties des familles de courbes très utilisées dans le domaine du multimédia [143].

C'est ce modèle qui représente les informations graphiques au niveau du système d'exécution de la présentation multimédia, deux axes suffisant dans la très grande majorité des cas.

3.4.3 Modèles relatifs

Les modèles spatiaux relatifs utilisent des formes continues et précisent des jeux de relations entre elles. La position des objets découle des relations qui abstraient les dépendances spatiales entre objets. Un moyen simple de considérer ces objets est de les projeter sur les différents axes du référentiel. Chaque objet est alors représenté par un couple ou un triplet d'intervalles, ce qui est dénommé *ombre temporelle* dans [3], dont la composition est désignée par le biais des relations d'Allen sur chaque axes. Un modèle spatio-temporel de ces relations est proposé dans [4] dans le cadre du système de base de données *STORM* (*Structural and Temporal Object oRiented model for Multimedia data*). [50] illustre ces relations dans cas de la conception de scènes architecturales.

Les autres modèles relatifs ne séparent pas les objets mais les considèrent comme les seules entités à organiser. Ces modèles sont plus proches de l'intuition que nous nous faisons de l'espace comme un médium uniforme et non comme la combinaison de deux ou trois dimensions.

Le domaine de la méréo-topologie [110] s'intéresse aux relations de contact et de connexité entre parties et tout. Il comprend beaucoup d'ontologies dont les modèles spatiaux diffèrent par des variations fines. Parmi les catégories de modèles qui se dégagent se trouvent [96]:

- les modèles topologiques
  Ils sont fondés sur les notions de proximité et d'incidence. Le modèle *RCC-8* (*Region Connection Calculus*) [110, 159] constitue le plus utilisé d'entre eux. Il définit les huit relations qui peuvent exister entre deux surfaces continues quelconques, comme illustré en table 3.4.
  RCC-8 est la contrepartie des relations d'Allen en deux dimensions. [59] joint ces deux modèles en se limitant aux relations non-disjonctives et aux objets de taille fixe qui évoluent continûment. Tout comme pour les relations d'Allen, des disjonctions de relations RCC-8 permettent d'exprimer

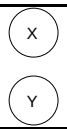
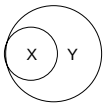
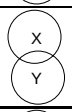
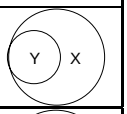
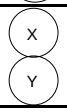
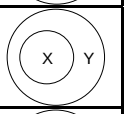
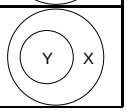
DC (disconnected) 	TPP (tangential proper part) 
PO (partial overlap) 	TPP^{-1} (TPP inverse) 
EC (externally connected) 	NTPP (non-tangential proper part) 
EQ (equal) 	$NTPP^{-1}$ (NTPP inverse) 

FIG. 3.4 – *Les relations spatiales RCC-8*

de l'indéterminisme sur la composition spatiale et sont la base de système de raisonnement sur cette composition [159].
[13] définit des relations d'inclusions hiérarchiques, c'est-à-dire sans recouvrements, entre les régions dénommées granularités.

- les modèles directionnels
  Ils sont fondés sur l'ordonnancement de l'espace, le plus simple d'entre eux divisant l'espace en ses huit directions usuelles, *north*, *northeast*, *east*, *southeast*, *south*, *southwest*, *west* et *northwest*.
- les modèles basés sur la distance
  Ils se basent sur la définition d'une distance entre objets.

3.4.3.1 Géométrie fonctionnelle

P. Henderson [73] définit un jeu de cinq opérateurs spatiaux afin de construire des assemblages d'éléments géométriques simples en deux dimensions. Ces opérateurs sont *flip* (reflet vertical), *beside* et *above* (juxtapositions verticale et horizontale), *rot* (rotation) et *overlay* (superposition). Ils s'appliquent sur des formes définies à partir d'une grille qui spécifie la taille de la forme ainsi que les coordonnées de ses points caractéristiques. Le constructeur *nil* permet de créer une grille vide.

Ce modèle très élégant permet d'obtenir une diversité de compositions tout en gardant une grande simplicité. Comme les approches temporelles à base d'opérateurs, il se suffit à lui-même, ne nécessitant pas de calculs complexes lors du formatage, et ne permet pas de spécifier des compositions incohérentes, car chaque opérateur fonctionnel renvoie une forme cohérente par construction. On retrouve une version simplifiée de cette approche dans le système des boites imbriquées où seuls les deux opérateurs *beside* et *above* sont utilisés.

3.4.4 Formatage spatial

A l'instar de la spécification temporelle, la spécification spatiale nécessite parfois des traitements afin d'être rendue, ce que l'on recoupe sous le nom de formatage spatial [30]. Afin que la spécification soit transcrite en ensembles de points des diverses formes, la résolution de système d'équations ou la génération des coordonnées des formes doit avoir lieu [50]. Les diverses phases du formatage et les problèmes d'incohérence de spécification sont identiques à ceux du formatage temporel, nous ne nous attarderons donc pas à les répéter.

3.5 Mouvement

A la frontière entre les spécifications spatiales et temporelles se trouve la spécification des mouvements des objets multimédias. Le mouvement fait l'objet d'études et de propositions spécifiques, comme le module Animation du standard *SMIL* [9] ou la théorie qualitative du mouvement de Muller [110]. On retrouve la même topologie de relations que dans les cas temporel et spatial:

- relations absolues ou quantitatives
  Les relations utilisent alors des représentations quantitatives de l'espace et du temps qu'elles lient à travers un système de contraintes ou d'équations. Par exemple dans SMIL, à une variable spatiale est attachée une fonction d'animation, sous la forme d'une série de valeurs, une durée et un mode d'interpolation des valeurs intermédiaires. Dans Flash [143], le mouvement est représenté par une matrice de transformations des coordonnées attachée à chaque objet.
  Le mouvement possède deux paramètres particuliers qui sont la *vitesse*, variation de la position par unité de temps, et l'*accélération*, variation de la vitesse par unité de temps. Une vitesse négative permet d'indiquer une exécution à l'envers de la présentation. Dans la plupart des outils multimédias, la vitesse est indiquée au niveau de la définition de chaque entité comme un attribut qui modifie le rythme de l'horloge locale de chaque entité, par exemple l'attribut `speed` de SMIL. L'accélération correspond à une animation de cet attribut de vitesse mais est le plus souvent indiquée comme un facteur fixe, étant donné que la plupart des présentations multimédias ne requièrent pas d'importantes variations de l'accélération. Par contre, les modèles musicaux usent grandement de ce facteur pour décrire des effets sonores [67].
- relations relatives ou qualitatives
  Ces relations reposent sur les relations qualitatives spatiales et temporelles. La définition du mouvement dans ce cadre postule la continuité du mouvement, c'est-à-dire qu'il n'y a pas de sauts [110]. [55] indique le graphe d'évolution continue des relations d'Allen et [110] celui des relations RCC-8. En combinant ces informations, [110] dégage six classes générale du mouvements: *leave*, *reach*, *hit*, *cross*, *internal* et *external*.

3.6 Navigation

La navigation à travers les présentations multimédias, que l'on renomme pour l'occasion hypermédia, est l'objet de nombreuses recherches [68, 83, 109]. Cette dimension des documents multimédias a pris une grande importance lors du développement du World Wide Web et à la suite de l'apparition des liens hypertextes dans le langage HTML. La navigation repose sur un mécanisme de lien, ou arc, sémantique généralement activé par l'utilisateur de la présentation multimédia, mais aussi parfois automatiquement par des systèmes informatiques. Elle est au cœur même de l'interaction avec cet utilisateur en lui offrant des choix, ou chemins, qui déterminent son parcours de la présentation. Tout lien comporte deux éléments:

- des *ancres*, ou *contextes*, de départ et de destination
  Les ancres définissent les conditions et les conséquences d'activation du lien. La ou les ancres de départ indiquent sur quels objets le lien repose, quelle fenêtre spatio-temporelle de leur présentation est valide pour ce lien et quelles autres conditions doivent être respectées, comme par exemple qu'une certaine variable numérique ait une valeur particulière [109]. Les ancres de destination indiquent les actions qui doivent être effectuées lorsque le lien est activé, comme déplacer l'instant courant de la présentation à un autre instant (passé ou futur), modifier un objet spatial ou bien détruire une information.
- une *sémantique*
  Elle permet de rendre le lien d'une manière particulière afin que l'utilisateur saisisse le choix qui lui est offert. La sémantique peut indiquer que le lien est d'ordre logique (renvoi, référence, annotation), spatio-temporel (saut) ou composite (plusieurs actions à la fois). Elle indique si le lien est bidirectionnel, ce qui permet au lien d'être traversé dans les deux sens en inversant ancres de départ et de destination.

3.7 Caractéristiques langagières

Comme indiqué en section 3.1, les modèles de composition multimédia peuvent être représentés par un langage (textuel) ou bien un format (binaire). Un langage est descriptible de manière unique, indépendamment des plate-formes sur lesquels il sera utilisé. Les langages de programmation informatiques ont fait l'objet de nombreuses études, théories et applications. Il se révèle utile d'analyser les langages multimédias sous cet angle car cela offre un éclairage différent et complémentaire sur ces langages. En effet, n'étant pas toujours formellement spécifiés, les langages multimédias souffrent parfois de défauts de conception, comme on a pu le voir pour le langage HTML dont la syntaxe a été reprise dans le cadre du méta-langage XML. Deux aspects des langages de programmation sont principalement concernés par les langages multimédias: son caractère impératif et son niveau d'abstraction.

Impérativité et déclarativité du langage

Tout langage de programmation repose sur le langage des instructions des processeurs des architectures matérielles des ordinateurs par le biais de *structures de contrôle*. On dit qu'un langage est *impératif* lorsque la séquence des instructions du langage correspond à celle des instructions qui seront exécutées par l'architecture matérielle, aux exceptions des boucles et alternatives qui ne sont qu'un moyen plus rapide de spécifier une séquence particulière d'instructions. Les langages C et Pascal sont deux langages impératifs classiques qui ont servis de base à la spécification de systèmes multimédias, par exemple dans le cadre de *Daliì* [113]. Bien que permettant d'implémenter efficacement ces systèmes, et étant parfois la seule approche possible lorsque les ressources nécessitent des optimisations particulières comme dans le cas des terminaux mobiles, cette approche n'est pas toujours satisfaisante. Les scénarios de ce type sont difficilement lisibles car leur séquentialité rend les mises en parallèle peu évidentes.

Les langages *déclaratifs* utilisent des abstractions des instructions machines, l'ordre des structures de contrôle utilisées n'étant pas celui des instructions qui seront exécutées. Les langages logico-fonctionnels, comme *Prolog* ou *Objective Caml*, fournissent le moyen d'organiser les déclarations des programmes de manière logique, soit en définissant des propositions logiques, soit en définissant des fonctions calculatoires. Les langages de contraintes permettent d'indiquer des ensembles de contraintes, indépendamment les unes des autres, l'exécution de ces programmes correspondant à la recherche de solution de ces contraintes. Les langages logiques peuvent être considérés comme les langages les plus déclaratifs qui soient, étant donné qu'il ne font pas intervenir de structure de contrôle. De nombreuses logiques temporelles ont été mises en place, comme *CTL* (*Computation Tree Logic*), *LTL* (*Linear Temporal Logic*) ou la logique modale *Tense Logic*, et certaines ont même été développées au cas par cas [116, 108, 14]. L'approche déclarative offre un haut niveau d'abstraction à la programmation, ce qui convient à la spécification multimédia. Les compilateurs ou les interpréteurs de ces langages doivent mettre en œuvre des mécanismes complexes afin d'adapter les fonctionnalités de haut niveau du langage au fonctionnement séquentiel des architectures matérielles et à leurs instructions. En plus de leur portabilité, ces approches apportent une vision conceptuelle des traitements des données, facilitant d'autant plus la compréhension des programmes, c'est-à-dire les présentations multimédias.

Le niveau d'abstraction du langage

Le deuxième aspect fondamental des langages de programmation sont les données qui sont construites et manipulées par les programmes. Les structures de données permettent la structuration du document (cf. section 3.2). Cet aspect n'est pas totalement indépendant du précédent, car souvent un haut niveau d'abstraction ne peut s'exprimer que dans un langage déclaratif. Le niveau le plus bas d'abstraction de ces données est le *bit*, qui abstrait la représentation des signaux physiques, et correspond à l'unité traitée par l'architecture matérielle.

Seuls les langages de bas niveau (assembleur, C) offrent accès aux bits.

Des structures plus intéressantes sont les entiers et caractères, généralement représentés par un ou plusieurs octet, cette représentation dépendant parfois des systèmes d'exploitation. La programmation des systèmes d'exécution des présentations multimédias optimisés requiert de travailler à ce niveau. Ensuite, les structures de données acquièrent des niveaux plus proches des données médias et encapsulent de grands groupes de données au sein d'une entité plus conceptuelle. On retrouve ici les représentations des images, des sons, des animations et des vidéos. Le langage associé au format Flash se situe à ce niveau.

Au delà, la structuration des données est forte, évoluée et placée au niveau du programmeur, qui se retrouve libre de créer ses propres structures. Ces langages permettent de créer des structures fonctions d'autres structures, certains permettant une définition générique d'une structure paramétrée par d'autres structures. On retrouve cette généricité dans des langages comme Z_YX [18]. Les langages orientés objets forment un paradigme très utilisé aujourd'hui. Un objet est généralement défini comme instanciation d'une classe, qui définit elle un ensemble de données et de méthodes liées sémantiquement. Le mécanisme de création des classes permet de les hiérarchiser en spécialisant des classes par héritage (ou implantation si certaines méthodes étaient déclarées mais non définies). La plupart des langages multimédias fournissent une hiérarchie d'objet à partir de laquelle des objets peuvent être instanciés pour créer une présentation multimédia, comme MHEG [150], d'autres fournissent aussi le moyen de créer de nouvelles classes, comme PREMO [75] . On peut noter que les programmeurs de langages de scripts comme LINGO se sont retrouvés dans le besoin de structurer leurs objets en créant des hiérarchies, mais ont dû fabriquer avec leurs propres moyens la notion d'héritage (par le biais de champs indiquant les objets père et fils par exemple) étant donné que LINGO ne fournit pas cette fonctionnalité. Il est intéressant de voir comment la spécification de données Flash amène à des problématiques de haut niveau et que le point de vue données tend tôt ou tard à rejoindre celui des traitements.

Enfin, par delà les données se trouvent les méta-données, c'est-à-dire les données qui informent sur d'autres données. Ce niveau est purement conceptuel puisque toute notion de traitement des données disparaît pour ne laisser place qu'à leur description. De nombreuses ontologies définissent ces méta-données par domaines [17] et sont définies dans des langages spécifiques, comme RDF. Il est possible d'organiser ces méta-données par le biais de constructions complexes, comme les strates interconnectées [122]. Le standard MPEG illustre l'élévation des niveaux d'abstraction: MPEG-1 et MPEG-2 définissent des données audio-visuelles et leurs encodages, MPEG-4 s'intéresse à modéliser ces données grâce à des structures à objets et enfin MPEG-7 modélise les méta-données de ces vidéos.

Un aspect des langages de programmation moins important que les précédents mais qui a un intérêt dans le multimédia est l'accès à des données dynamiques, c'est-à-dire dont la valeur ne peut être déterminée que lors de l'exécution. Pour les spécifications relatives et comprenant des sources d'indéterminisme, la compilation, c'est-à-dire le formatage, ne peut être complète et nécessite de reporter certaines évaluations à l'exécution.

Notons enfin que divers travaux ont tenté de comparer les langages multimédias à la lueur de leur pouvoir d'expression temporel [154, 117]. Cette comparaison n'est pas simple car certains concepts semblent trop orthogonaux pour être comparés, comme la causalité temporelle du modèle *Menkalinan* par rapport aux relations d'Allen [147]. Un critère de comparaison plus objectif pourrait être le coût du formatage mais aucune étude générale n'a été menée sur ce sujet.

3.8 Langages multimédias

3.8.1 HyTime

HyTime (*Hypermedia/Time-based Structuring Language*) [139] est un standard *ISO* (ISO/IEC 10744) de la couche Presentation du modèle *OSI* conçu pour la représentation des documents structurés hypermédias. Il ne spécifie pas le codage ou le format des données qui sont structurées au sein de ses documents.

Il est une extension du méta-langage *SGML* (*Standardized General Markup Language*) qui permet la description des documents structurés à base de balises. HyTime ajoute à SGML des extensions (*extended facilities*) permettant la définition de types lexicaux, de formes architecturales, d'ensembles de propriétés et d'identificateurs de systèmes formels. Un langage utilise ensuite ces fonctionnalités au sein d'une DTD HyTime. Les différentes parties du standard sont chacune regroupées dans une méta-DTD et définissent chacune diverses formes architecturales, c'est-à-dire des ensembles d'éléments, d'attributs et des notations. La méta-DTD HyTime principale est divisée en cinq modules:

- le module de base
  Il précise les notations et concepts utilisés dans les autres modules.
- l'adressage des localisations. Le standard définit trois moyens de désigner ces entités: par nom, par localisation sémantique ou par coordonnées. Il est aussi possible d'utiliser des conventions extérieures, comme celles de XML par exemple.
- les hyperliens
  Ce module définit la syntaxe et la sémantique de la représentation des liens à partir des modes d'adressage précédent. Il faut utiliser un ensemble de propriétés de HyTime afin de pouvoir mettre en place ces liens effectivement.

– l'ordonnancement
  C'est le module qui permet de définir la synchronisation des divers éléments et qui est décrit plus en détail plus bas.
– le rendu
  Ce module est une application du précédent et permet de créer de nouveaux ordonnancements en appliquant des règles de rendu à un ordonnancement.

L'ordonnancement consiste à placer les objets d'un document sur des espaces finis de coordonnées (*FCS*, *Finite Coordinate Spaces*) qui définissent un ou plusieurs axes de coordonnées, spatiales ou temporelles. Un objet est représenté sur un axe par une *dimension* et l'ensemble de ses dimensions forment un *extent*. Chaque dimension est définie par deux marqueurs d'axes, un qui positionne le début de l'objet et l'autre qui indique la taille ou la durée. Ces marqueurs peuvent utiliser des références à d'autres marqueurs, créant ainsi dans le temps une synchronisation en séquence ou en parallèle d'objets et des objets de même durée. Un *pulse* peut être rattaché aux extents afin de les répéter. Les extents doivent finalement être attachés à des événements (scheduled event) afin d'être positionnés de façon absolue dans un FCS. Ces événements sont associés à des modifications d'objets grâce à des *baguettes* (wand). Des projecteurs sont aussi utilisés afin de projeter les extents sur un autre extent au sein d'un *bâton*. Ces projections permettent par exemple de mettre en correspondance un extent aux coordonnées abstraites avec un extent aux coordonnées concrètes. Chaque application spécifiée en HyTime définit les modificateurs d'objet et les projections dont elle a besoin pour exprimer sa sémantique. La composition spatiale d'une application HyTime peut être définie par une feuille de style *DSSSL* (*Document Style Semantics and Specification Language*). Les transformations utilisées par DSSSL permettent aussi de traduire une spécification HyTime en un scénario MHEG ou encore CMIF [131].

Un exemple simple de synchronisation utilisant les définitions de HyTime est donné en table 3.1

`<timefcs>`	
`  <evsched>`	événement
`    <event exspec="extent1">`	
`    <event exspec="extent2">`	
`    <extlist id="extent1">`	définition de extent1
`      <dimspec id="dimension1"> 30 210`	début:30; durée: 210
`    <extlist id="extent2">`	définition de extent2
`      <dimspec id="dimension2">`	
`        <dimref elemref="dimension1"`	début: égal au dernier
`                selcomp="last"`	quantum de dimension1
`                flip="flip">`	
`        <dimref elemref="dimension1"`	durée: même quantité de
`                selcomp="qcnt">`	quantum que pour dimension1

TAB. 3.1 – *Exemple de présentation HyTime*

Le standard est en fin de compte ardu à maîtriser, comme en témoigne sa longueur de 450 pages [61]. Peu d'applications d'exécution pour HyTime, nommées *HyTime engine*, ont vu le jour à cause sa trop grande généralité et flexibilité. Le standard a été remis à jour en 1997 peu avant que XML soit une recommandation mais n'a plus évolué depuis.

3.8.2 PREMO

PREMO (*Presentation Environment for Multimedia Object*) [75] est un standard *ISO* (*ISO/IEC 14478*) qui se focalise sur la présentation des objets multimédias. PREMO sort du traditionnel paradigme du document pour s'inscrire dans le domaine des applications multimédias en ayant pour objectif de fournir un cadre de programmation flexible et ouvert. Il constitue une tentative pour fournir un middleware multimédia susceptible de permettre l'émergence de nombreuses applications et systèmes multimédias.

PREMO est décrit dans un cadre orienté objet. Les objets PREMO sont actifs, au sens des processus concurrents et possédent des méthodes synchrones, asynchrones ou simples (*sampled*), les demandes d'exécution de ces dernières n'étant pas stockées en file d'attente. L'héritage permet de spécialiser les classes spécifiées par le standard. Un composant correspond à un ensemble d'objets PREMO travaillant à fournir certains services. PREMO se décline en quatre parties: fondamentaux (définition des concepts utilisés, architecture); composants de base; composants de modélisation, de présentation et d'interaction et composants de services pour systèmes multimédias. Les applications PREMO spécifient la présentation des données en procédant selon cinq étapes de transformation de ces données. Ces transformation sont modélisées en environnements: construction (création d'un modèle à partir des données), virtuel (production d'une scène à partir du modèle), visionnage (génération d'une image par projection de la scène), logique (complétion des primitives d'affichage, qui ne sont pas nécessairement associées à des périphériques physiques), réalisation (affichage effectif). Les composants de service permettent de spécifier le type des informations utilisées par l'application multimédia et de spécialiser les flots des données afin de préciser leur codage, leurs protocoles de transports ou bien l'utilisation de fonctionnalités particulières des périphériques physiques.

La synchronisation inter-objets est basée sur l'utilisation d'un objet gestionnaire d'événement [76]. Les objets synchronisables possèdent des *points de référence* sur la ligne de temps qui leur est associée. Un objet PREMO s'enregistre auprès du gestionnaire d'événement d'un autre objet afin d'être informé de l'occurrence d'un certain point de référence et fournit le nom de la méthode, dite de *callback* à invoquer lorsque l'événement se produit. Lorsque l'événement survient, l'objet source informe le gestionnaire d'événement, qui transmet l'information à tout ou partie des objets enregistrés selon sa logique interne.

Les diverses fonctionnalités proposées par le standard PREMO permettent aux programmeurs d'implémenter une multitude d'applications multimédias aux politiques de synchronisation très diverses. Afin d'assurer que ces applications n'aient qu'une unique interprétation, la sémantique des divers classes d'objets PREMO et de leurs méthodes est formellement spécifiée dans le langage Object-Z [51]. Chaque objet a un état interne dont l'évolution, après appel de ses méthodes, est décrite par une machine d'état finis. Des notes sont ajoutées à la spécification Object-Z des objets PREMO et de leurs méthodes afin d'exprimer en langue naturelle des aspects fondamentaux qui ne sont pas clairement capturés en Object-Z, comme la progression du temps.

3.8.3 MHEG

MHEG (*Multimedia and Hypermedia Information Coding Expert Group*) est un standard de l'*ISO* (*ISO/IEC 3522*) qui étend les préoccupations des standards JPEG et MPEG aux données multimédias et hypermédias. Il s'agit donc de coder les informations multimédias interactives qui seront utilisées dans des environnements aussi divers que les ordinateurs multimédias, des assistants numériques *PDA*, des bornes interactives ou bien les set-top box de télévision numérique, domaine dans lequel le standard a le plus percé. Un document MHEG constitue une forme finale qu'un moteur de rendu pourra utiliser pour jouer une présentation multimédia [106].

Ce standard à évolué à travers diverses parties, passant de représentations ASN.1 à SGML puis à une syntaxe spécifique et une mise au format XML est en cours. Il spécifie le format de définition des documents MHEG et les fonctionnalités d'un programme qui doit présenter ces documents, programme appelé moteur MHEG (*MHEG engine*).

MHEG s'inscrit dans un paradigme orienté objets, avec une notion classique de classe mais la possibilité de créer deux types d'objets, les objets de contenu (*content object*) et les objets d'exécution (*virtual view* ou *runtime object*) qui permettent de reproduire les premiers. Dans les premières versions de MHEG, les objets étaient actifs, mais ce principe a été abandonné dès MHEG-5. MHEG définit une hiérarchie de classes dont la racine est `Root`. La hiérarchie des objets est présenté dans la figure 3.5. Cette hiérarchie ne sert qu'à présenter de manière structurée les classes car il est impossible de définir de nouvelles classes, la notion d'héritage étant traduite par des importations et exportations de types de données.

Une présentation multimédia MHEG correspond à un objet de classe `Application` qui est une séquence d'objets `Ingredient`. La classe `Ingredient` est la racine d'une sous-hiérarchie organisée couvrant la plupart des médias, les objets interactifs, les liens et les programmes. Ces objets programmes sont le moyen de lancer un programme exécutable qui est extérieur au moteur MHEG

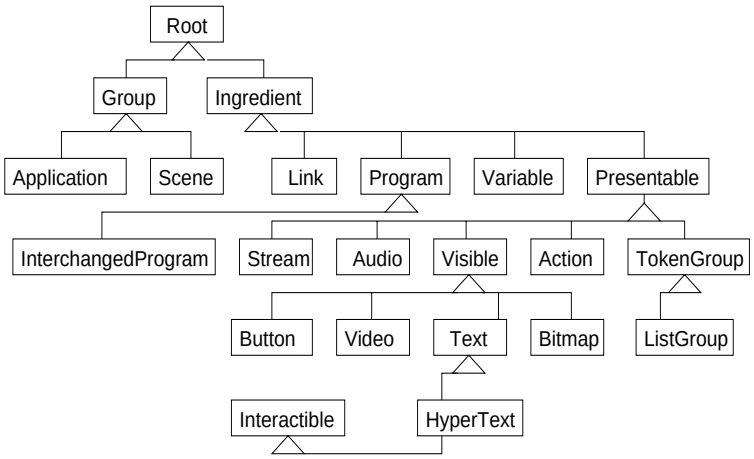


FIG. 3.5 – *Hiérarchie des objets MHEG*

et donc d'interfacer le langage avec d'autres langages de programmation. Les objets interactifs héritent de la classe abstraite `Interactible` et permettent de créer des applications dont l'interactivité est évoluée [142].

La synchronisation des objets MHEG se fait en définissant les comportements de ceux-ci. Elle est définie à deux niveaux:

- par encapsulation d'objets au sein d'objets composites de plus haut niveau, comme `Application` qui compose les objets `Scene` en séquence; les classes `List` et `TemplateGroup` permettent de spécifier un comportement générique qui est applicable à un groupe quelconque d'ingrédients;
- par utilisation d'un lien (`Link`) qui envoie une ou plusieurs actions (objet `Action`) à un ou plusieurs objets destinations lorqu'une condition est vérifiée; cette condition est basée sur un événement d'un certain type associé à un objet source; dans la partie MHEG-1, la condition est basée sur des variables associées à l'état de l'objet source; une action élémentaire est un appel de méthode de l'objet destination et certaines actions déclenchent un changement d'état des objets; une action peut être composite en encapsulant un arbre d'actions élémentaires, indiquant des mises en séquence ou en parallèle d'actions.

Les actions de synchronisation sont `Run` et `Stop` et sont adjointes à des actions de délais [145]. D'autres actions font appel à des méthodes qui ne sont pas liées directement à la présentation de contenu, impliquant un calcul ou la navigation dans le document. Ceci permet de créer des présentations complexes en MHEG, en mélangeant au code de structuration de la présentation multimédia des sélections qui peuvent être basées sur des données présentées ou sur les choix de l'utilisateur selon une logique complexe. Même s'il en rappelle certains

concepts, MHEG n'a pas la complexité d'un langage de programmation complet, puisque les structures du langage restent simples et qu'il est impossible de créer de nouvelles classes.

Un exemple de synchronisation simple en MHEG-5 est présenté dans le scénario en table 3.2.

`{:scene`	
`:object-identifier:Example`	la scène `Example` est constituée de:
`:group-items(`	
`{:bitmap`	l'image `Pic.jpg`
`  :object-identifier 1`	
`  :initially-active  true`	
`  :original-box-size (320 240)`	
`  :original-position (0  0)`	
`  :content-data`	
`    :referenced-content "Pic.jpg"`	
`}`	
`{:text`	un texte situé sur cette image
`  :object-identifier 2`	
`  :original-box-size (280 20)`	
`  :original-position (40 50)`	
`  :content-data`	
`    :included-content "Next"`	
`}`	
`{:link`	et un lien qui joue l'objet `nextPart` lorsque l'on clique
`  :event-source Example`	
`  :event-type #UserInput`	
`  :event-data #Left`	
`  :link-effect :action`	
`    :transition-to nextPart`	
`})}`	

TAB. 3.2 – *Exemple de présentation MHEG*

3.8.4 SMIL

SMIL (*Synchronized Multimedia Integration Language*) [9, 24] est un langage créé par le *W3C* qui en est à sa deuxième version. Il est le résultat de la coopération de nombreuses sociétés et de laboratoire de recherches dans le multimédia. SMIL est un langage à base de balises défini en XML (*eXtensible Markup Language*), méta-langage permettant la définition de langage à base de balises. Il permet de créer des présentations multimédias interactives grâce à l'usage de balises et d'attributs définis dans des modules. La définition modulaire permet d'intégrer les fonctionnalités apportées par SMIL dans d'autres langages basés sur le méta-langage XML. Divers profils fournissent des regroupements par défaut de modules et permettent par exemple d'adresser le cas des

plate-formes aux ressources limitées (SMIL 2.0 Basic) ou bien de s'intégrer à un langage déjà existant (XHTML+SMIL).

Un document SMIL comporte deux parties, une en-tête et un corps de document. L'en-tête est délimitée par les balises `<head>` et `</head>` et contient la définition des zones d'affichage (*layout*) et de leurs régions dans lesquelles les objets synchronisés dans la présentation sont rendus. Ces zones peuvent être définies de manière imbriquées et selon des modes de positionnement complexes. Leur affichage se fait en 2.5D, chaque région étant positionnée dans un plan dont on peut définir la hauteur. Le corps du document comporte les définitions des objets à synchroniser, de leur synchronisation, ainsi que les animations, liens et transitions sur les objets. Les données à présenter sont encapsulées dans diverses balises qui indiquent leur média, comme par exemple `animation`, `text`, `textstream` ou `video`, et sont référencées via des identificateurs *URI* (*Uniform Resource Identifier*). Les données étant référencées et non comprises dans le document SMIL, elles peuvent être partagées et réutilisées entre plusieurs documents.

La synchronisation des objets se fait à travers:

- des balises qui définissent un conteneur temporel, les fils de cet élément étant les éléments encapsulés dans le conteneur; trois balises sont définies dans SMIL 2.0:
  - `<seq>` pour la présentation en séquence;
  - `<par>` pour la présentation en parallèle;
  - `<excl>` pour la présentation en mode exclusif, un seul des fils étant présenté à la fois avec la possibilité d'indiquer des priorités;
- des attributs, définis pour la plupart des balises SMIL, qui permettent d'indiquer:
  - l'instant de début (`begin`) ou de fin (`end`); ces attributs peuvent comporter une liste de valeurs qui sera interprétée comme une liste d'occurrences parmi laquelle tous les éléments ou bien uniquement le premier d'entre eux compte; une valeur peut être un temps absolu, un événement relatif au début ou à la fin d'un autre élément, un événement extérieur (clic, appui sur une touche de clavier) ou `indefinite` (pour indiquer une activation par un hyperlien ou par une méthode externe à SMIL);
  - la durée de l'élément (`dur`, `min`);
  - les répétitions de la présentation de l'élément (`repeatCount`, `repeatDur`, `restart`);
  - le comportement de synchronisation fine (`syncBehavior`, `syncMaster`).

Le comportement d'un scénario SMIL repose sur une sémantique à base d'intervalles. Chaque élément, objet média ou bien conteneur temporel, possède deux listes indiquant les occurrences possibles des débuts et des fins, listes qui

peuvent changer dynamiquement lors de l'exécution du scénario. Un intervalle, ou durée simple, s'étend alors depuis une valeur de la liste de début jusqu'à la fin correspondante dans la liste des fins. L'ensemble des répétitions de cette durée simple constituent la durée active. Divers algorithmes sont établis dans la sémantique de SMIL afin de calculer ces durées et de construire un graphe temporel du scénario, dont les nœuds sont les éléments du scénario et les arcs des relations de synchronisation. La complexité de ces calculs provient de l'interaction des divers attributs et des conteneurs temporels sur les deux durées et de la prise en compte d'événements dont l'instant d'occurrence ne peut être connu qu'à l'exécution.

La table 3.3 présente un exemple de document SMIL. Dans cet exemple, trois régions `a`, `b` et `c` sont définies afin d'afficher trois données médias identifiées par les mêmes noms. La composition temporelle consiste en la mise en parallèle de l'audio `a` avec la séquence de l'image `b`, présentée cinq secondes, et la vidéo `c`, arrêtée par l'audio.

```
<smil>
<head>
  <layout>
    <root-layout height="350" width="600"
      background-color="#ffffff"
      title="Exemple de document SMIL"/>
    <region id="a" height="47" width="63" z-index="1"/>
    <region id="b" height="49" width="55" z-index="2"/>
    <region id="c" height="47" width="44" z-index="3"/>
  </layout>
</head>
<body>
  <par>
    <audio id="a" src="media/a.au" begin="5s" end="11s"/>
    <seq>
      <img id="b" src="media/b.jpg" region="b" dur="5s"/>
      <video id="c" src="media/c.mov" region="c" end="a.end"/>
    </seq>
   </par>
  </body>
</smil>
```

TAB. 3.3 – *Exemple de présentation SMIL*

3.8.5 MADCOW

MADCOW (*Multimedia Architecture for Distribution of Contents Over the Web*) [128] est un projet élaboré autour d'une extension temporelle au langage

HTML. Le langage développé s'organise autour de trois concepts [130]: les liens hypertemps, les bases de temps et les zones d'affichage dynamiques.

Une unique balise `object` permet de définir les objets média du document en référençant les données médias ou bien encapsulant un groupe d'objets. Les attributs de ces objets indiquent leur identificateur, la localisation des données et des paramètres comme la vitesse de présentation. Des sous-balises introduisent les points de synchronisation, les points temporels et les liens hypertemps de l'objet. Par défaut, les objets possèdent deux points temporel de début (`beg`) et de fin (`end`).

La synchronisation fine des objets est déclarée grâce aux bases de temps, au sein desquelles chaque objet possède un rôle de maître ou, par défaut, d'esclave. Tout délai intervenant lors de la présentation d'un point de synchronisation d'un objet maître est répercuté sur l'ensemble de la base de temps, alors que ceux des objets esclaves sont ignorés. La synchronisation inter-média est indiquée par les liens hypertemps, introduit par la balise `htlink`, qui sont une extension intuitive des liens hypertextes au domaine du temps. Ce lien relie un point temporel d'origine à un point temporel de destination. La sémantique temporelle d'un document MADCOW est simple et intuitive.

Cette simplicité se retrouve dans le prototype, lui aussi architecturé autour de trois entités [129]: les objets synchronisables, les événements de synchronisation et les gestionnaires de synchronisation. La correspondance avec les éléments du langage est immédiate. Les événements de synchronisation sont générés par les objets synchronisables puis envoyés au gestionnaire de synchronisation, qui peut alors appliquer la politique de synchronisation du document, en envoyant l'événement à un objet synchronisable dans le cas d'un lien hypertemps ou en vérifiant que la synchronisation fine est respectée dans le cas d'un point de synchronisation.

Les informations spatiales d'un document MADCOW consistent à définir une zone d'affichage qui est traitée comme un cas particulier d'objet. Cette zone contient un ou plusieurs cadres dans lequel est affiché le contenu d'un objet, l'imbrication de zones d'affichage étant alors possible.
La table 3.4 donne un exemple de document MADCOW.

3.8.6 Menkalinan

Le modèle de *Menkalinan* [85] permet de spécifier la synchronisation de présentations multimédias interactives. Il s'inspire de celui de la *Video Algebra* [156] et repose sur deux concepts: la stratification et les opérateurs algébriques. Le concept de strate est issu des problématiques d'annotation du contenu et est un objet abstrait qui encapsule une unité de sens, qui peut être composée afin de construire des unités d'un niveau sémantique supérieur. Les strates sont

`<timebase>`	
`  <layout>`	
`    <frame src="audio1_f" shape="0,0">`	Zone de l'audio
`    <frame src="video1_f" shape="0,0,250,300">`	Zone de la vidéo
`  </layout>`	
`  <object id="audio1" src="audios\aud1.wav">`	Définition de l'audio
`  </object>`	
`  <object id="video1" src="videos\vid1.mpg">`	Définition de la vidéo
`    <time-point id="end" value="30"`	dont la fin est
`       unit="second" type="absolute">`	redéfinie à 30 secondes
`  </object>`	
`</timebase>`	
`<htlink target="audio1.beg">`	l'audio commence
`<htlink orig="audio1.beg" target="video1.beg">`	et sa fin lance la vidéo

TAB. 3.4 – *Exemple de présentation MADCOW*

organisées de manière complexe par recouvrement afin d'offrir les moyens de modéliser des arrangements d'une certaine richesse.

La composition se fait par le biais d'opérateurs fonctionnels qui offrent une structure d'arbre au présentations Menkalinan. Les opérateurs encapsulent leurs objets arguments en fournissant des objets composés de plus haut niveau d'abstraction. Ils sont de deux types: temporels et non-temporels. Les opérateurs non-temporels permettent la fusion de structure, l'annotation et l'accès par le contenu. La fusion de structure correspond aux opérations ensemblistes usuelles sur des objets composés, permettant de ne référençer que les portions communes ou différentes des arbres qui sont fusionnés. L'annotation d'un objet permet de lui associer une information textuelle qui pourra ensuite être lue et comparée avec celles d'autres objets.

Les opérateurs temporels sont au nombre de dix. Certains reprennent des constructions bien connues (`seq`, `par-begin`) alors que d'autres désambiguïsent la relation d'Allen *equal* (`equal` et `ident`). Les opérateurs de boucle `loop` et de choix `alternative` rappellent les structures de contrôle traditionnelles des langages de programmation. La sémantique des opérateurs temporels repose sur le principe de causalité entre points temporels liés aux objets. Bien qu'augmentant le pouvoir d'expression du langage, la fusion temporelle a une sémantique qui est non-triviale. En effet, [2], [160] et [3] donnent diverses interprétations des opérations ensemblistes sur les objets de bases de données. Certains scénarios faisant intervenir la fusion prêtent à confusion, par exemple:

```
union(seq(A,B),seq(B,A))
```

En effet, ce scénario peut être interprété comme la séquence d'intervalles `A;B;A` si l'on fusionne l'intervalle `B`, la séquence d'intervalles `B;A;B` si l'on fusionne

l'intervalle `A` ou bien être considéré comme incohérent si l'on tente de fusionner les deux intervalles.

On retrouve ces *Interval Expressions* dans le modèle Z_YX [18] et dans [21].

3.9 Conclusion

Nous avons tenté de dresser dans ce chapitre un panorama aussi fidèle que possible des présentations multimédias et de leur composition. Il en ressort qu'un grand nombre de solutions existent afin de spécifier cette composition et que la diversité des approches peut être analysée en regard de certains critères simples. Les diverses caractéristiques étudiées font apparaître l'utilité de méthodes structurées et d'approches reposant sur des modèles clairement définis. Les langages sont un paradigme adapté à ces contraintes, leur syntaxe définissant la structure de l'outil et leur sémantique son modèle. Néanmoins peu de langages arrivent à couvrir un large spectre des aspects de la composition multimédia. L'exhaustivité de certaines approches se heurte à leur manque d'uniformité et ne contribue pas à clarifier les solutions apportées.

Nous avons choisi de prolonger les travaux du modèle *Menkalinan* qui offre une composition temporelle à base d'opérateurs d'un haut niveau d'abstraction. *Menkalinan* pose les bases d'une sémantique temporelle intuitive qui clarifie la structure d'une présentation multimédia. Nous avons examiné plus avant les possibilités de ce modèle et l'avons formalisé dans le cadre d'un langage multimédia que nous présentons ci-après.

Chapitre 4

TAO: Langage de spécification de présentations multimédias

Nous présentons ici notre proposition dans le cadre de la spécification de présentations multimédias sous la forme du langage *TAO*, pour *Temporal Algebraic Operators*. Ce langage doit permettre la composition d'objets monomédias constitués en référençant des données que l'auteur a à sa disposition et d'indiquer les arrangements dans le temps et l'espace de ces objets afin de traduire le scénario que l'auteur veut transcrire. Le langage TAO ainsi que ses principes ont fait l'objet de trois publications [12, 120, 121].

L'auteur manipule dans le cadre du langage des objets, décrits en section 4.1. Il se sert alors des opérateurs du langage afin de composer la synchronisation de ses objets dans le temps, comme présenté en section 4.2. Le sémantique de ces opérateurs fait l'objet d'un modèle temporel qui est exposé dans la section 4.3. La définition des propriétés spatiales des objets est introduite en section 4.4. Pour terminer ce chapitre, des exemples de programmes TAO sont donnés en section 4.5.

La programmation de la composition se fait à un haut niveau d'abstraction afin de gommer les détails d'implémentation qui permettent de mettre en œuvre la composition. Cette spécification sera ensuite raffinée afin d'exécuter à proprement parler le scénario que l'auteur a décrit, ce que nous décrivons dans le chapitre 5.

4.1 Objets multimédias

4.1.1 Langage orienté objet

Notre contribution ne portant pas sur les langages de programmation à proprement parler, le cadre du travail s'inscrit dans un paradigme très répandu aujourd'hui, celui des langages orientés objets. TAO utilise la notion d'objet dans la définition classique qu'on lui attribue aujourd'hui.

Les objets du langage sont des instances de classes, au sens où il fournissent une valeur respectant la définition de la classe, et sont identifiées par un nom. Les classes sont composées d'attributs qui sont d'autres objets et de méthodes qui définissent les opérations appliquées aux attributs et fournies par la classe. Une classe est dite abstraite lorsqu'au moins une de ses méthodes est déclarée mais sans définition fournie et interface lorsque aucune de ses méthodes n'a de définition. L'intérêt de ces classes est de fournir un modèle pour plusieurs autres classes, dites concrètes ou implémentation, qui sont créées en définissant les méthodes qui ne possèdent pas de corps.

L'héritage est un aspect essentiel de la programmation orientée objet. Une classe B hérite d'une classe A lorsque tous les membres et les méthodes de la classe A se retrouvent dans la définition de la classe B. Certains langages permettent des formes d'héritage multiple de plusieurs classes, ce qui peut amener à des conflits sur la définition d'attributs et de méthodes. Des langages comme *Java* résolvent le problème en ne permettant l'héritage multiple que de la part d'une seule classe concrète et plusieurs interfaces, évitant ainsi les conflits. Il est d'autre part possible de redéfinir une méthode héritée en la déclarant à nouveau et en fournissant une nouvelle définition qui viendra remplacer celle héritée de la classe parent. Ce mécanisme fournit une ouverture du système qui permet à tout programmeur de spécialiser des classes existantes en redéfinissant certaines de leurs méthodes. Un programmeur utilise le polymorphisme des classes lorsqu'il déclare un objet d'une certaine classe et l'instancie par une de ses classes héritées. C'est là que réside un des intérêts majeurs de l'héritage de classes qui permet de déclarer des objets d'une classe commune et les affecter avec des classes particulières. Le polymorphisme implique de définir une sémantique des appels de méthodes qui peuvent être résolus à deux moments distincts, soit à la compilation (ce que l'on appelle liaison statique), soit à l'exécution (liaison dynamique).

4.1.2 Java

Plus précisément, notre travail s'insère dans la mouvance du langage Java de *Sun Microsystems*. Java est un langage orienté objet apparu en 1995, qui est proche du langage *C++* et dont les points clé sont:

- l'utilisation de structure de données objets
  Seules les structures de données fondamentales (`int`, `char`, `boolean`) n'uti-

lisent pas la structure d'objet, concession qui a été faite pour conserver des habitudes proches du C++ et que l'on peut contourner en se servant de leurs équivalents sous forme d'objets. Le programmeur doit se servir pour l'essentiel de la structure d'objet, en créant éventuellement ses propres classes d'objets par héritage.

- de nombreuses librairies de classes
  Les librairies Java sont structurées de façon hiérarchique en leur donnant un nom comme séquence d'identificateurs séparés par le symbole "." en imbriquant des concepts du moins précis au plus précis, par exemple `javax.imageio.plugins.jpeg`. Les librairies Java sont l'une des grandes richesses de la plate-forme car elles sont nombreuses et l'on y trouve un grand nombre des fonctionnalités déjà implémentées, ce qui permet aux programmeurs de se concentrer sur leurs apports spécifiques. Les hiérarchies de classes sont parfois amenées à changer, par exemple lorsque des fonctionnalités spécialisées deviennent communes et sont remontées d'un niveau dans la hiérarchie. Il est néanmoins important de maintenir les classes dans des "niveaux" clairs afin de garder une interface relativement fixe, que tout programmeur peut retrouver.
- un mécanisme de ramasse-miettes (*garbage collector*)
  Ce composant gère la destruction des objets et de leurs références automatiquement, facilitant la tâche du programmeur qui ne se soucie plus de la destruction de ses objets.
- l'indépendance des programmes vis-à-vis de la plate-forme
  La plate-forme Java utilise un format intermédiaire, le *bytecode*, afin de séparer la compilation de l'exécution du code, qui se fait grâce à un interpréteur appelé *machine virtuelle Java*, et permettre d'implémenter ces éléments séparément sur chacune des grandes plate-formes informatiques (Unix, Linux, Windows, MacOS). Aujourd'hui cette indépendance est acquise car les années de travail des développeurs de machines virtuelles Java ont permis de porter toutes les fonctionnalités du langage vers toutes ces plate-formes propriétaires.

Java dans sa version 2.0 a un succès important dans l'industrie informatique et fournit, à l'image du multimédia pour les données, un paradigme de programmation ouvert, puissant et portable. Le langage est donc relativement bien ancré dans l'esprit des programmeurs et l'usage d'une syntaxe s'en inspirant permet de bénéficier de tout le travail de définition du langage qui n'est pas à refaire. Java fournit un socle solide sur lequel nous ajouterons des fonctionnalités propres à la spécification de présentations multimédias. Lorsque des éléments de définition du langage TAO manquent, il sera implicite qu'ils sont identiques à ceux du langage Java.

4.1.3 Objets primitifs TAO

Les objets primitifs du scénario sont ceux qui référencent des données monomédias, créées hors du cadre de TAO, et dont la classe hérite de la classe `Interval`. Ces classes médias font abstraction des formats de données, la plupart des auteurs usant de formats répandus et qui sont parfaitement supportés par Java. La référence aux données se fait grâce à un identificateur URL (*Uniform Resource Locator*), popularisé par les adresses sur le web et aujourd'hui communément utilisé dans de nombreux langages. Les méthodes communes des objets primitifs sont indiquées dans la classe `Interval` et permettent de présenter les données ou bien stopper cette présentation, leur définition pour chaque média étant surchargée dans la classe média héritée de `Interval`. Des classes plus spécialisées peuvent être écrites afin de prendre en compte des fonctionnalités plus spécifiques, comme la lecture en marche arrière ou encore un rendu optimisé de certains formats. Nous ne nous sommes pas intéressé à ce point précis, suivant le point de vue choisi dans le langage multimédia MHEG qui considére que la hiérarchie fournie suffit à la plupart des besoins des auteurs.

La hiérarchie des classes standard TAO est issue de notre analyse des divers médias et reprend des classes et une organisation similaires à celles que l'on peut trouver dans des standards comme PREMO ou MHEG. Elle est présentée en figure 4.1.

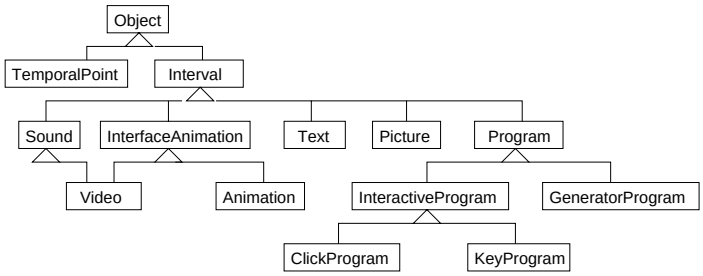


FIG. 4.1 – *Hiérarchie des objets primitifs TAO*

La classe `Video` est un cas particulier, car elle fait référence à des données qui sont rendues via deux périphériques, visuel et auditif. Elle représente des objets multimédias, et non monomédias, car elle hérite des deux classes `Sound` et `InterfaceAnimation`, cette dernière étant une interface introduite au-dessus de la classe `Animation` afin de permettre l'héritage multiple. Les vidéos sont créées par l'auteur ou bien le résultat de la mise en parallèle d'un objet animation et d'un objet son.

La classe `Program` représente des données programmes mais ne permet pas d'interfacer n'importe quel programme existant avec des présentations mul-

timédias ou bien d'implémenter des actions qui ne sont pas possibles dans le langage. Les programmes se regroupent en deux types:

- classe `InteractiveProgram`
  Ce sont des programmes interactifs, générateurs d'événements utilisateur correspondant aux interactions via les périphériques d'entrée comme le clavier, la souris, un écran tactile, un joystick ou bien une interface de reconnaissance vocale. Nous nous sommes bornés à fournir deux classe concrètes basiques, le programme identifiant le clic dans une région `ClickProgram`, qui émet les événements `onClick` et `onRelease`, et celui réagissant à l'appui d'une touche `KeyProgram`, avec les événements `keyPressed` et `keyReleased`.
- classe `GeneratorProgram`
  Ces programmes génèrent des données médias et s'exécutent en dehors du système TAO. Nous n'avons pas proposé de classes concrètes pour ce cas.

Les programmes peuvent avoir un ensemble d'événements spécifiques qui surviennent au cours de leur présentation et leur déclaration doit être faite lorsqu'on les utilise dans la présentation multimédia, grâce à la classe `TemporalPoint`. La présentation d'un objet programme est son exécution, dont le système de présentation des scénarios ne verra que les événements. Les programmes peuvent être composés au sein de programmes plus complexes et les événements peuvent alors être exportés dans l'objet composé de la manière suivante:

```
Program P1,P2,P3 ;
Point e1, e2 ;
Program P = ... /* composition des programmes P1, P2 et P3 */
            .{ e1 = P1.onClick;
               e2 = P2.keyPressed,P3.keyReleased} ;
```

La section entre accolades permet de déclarer l'exportation d'événements, les déclarations étant séparées par des points virgule. La virgule entre deux événements sert à indiquer leur disjonction, dans l'exemple précédent `e2` survient lorsque l'un des deux événements indiqués surviennent.

4.1.4 Objets composés

Les objets primitifs sont ensuite composés grâce à des *opérateurs* du langage pour définir des objets composés dont la classe est `Interval`. Ces opérateurs sont des mots clé du langage et seront décrits plus en détail en section 4.2.3. Les objets composés encapsulent les objets primitifs qui leur sont passés en arguments et une information de structure, la dimension nous intéressant ici étant le temps. Ils constituent les objets multimédias dans le sens où ils n'existent qu'à partir de plusieurs objets monomédias en les reliant par une relation structurelle.

Les objets composés peuvent eux-mêmes être arguments d'opérateurs, permettant ainsi de définir des scénarios de façon incrémentale. Les opérateurs prenant comme arguments des objets de classe `Interval`, les classe médias

doivent donc hériter de cette classe, c'est-à-dire que les objets monomédias sont des cas dégénérés d'objets multimédias. Grâce à l'incrémentalité, des scénarios entiers peuvent être composés entre eux, permettant ainsi la réutilisation des scénarios comme vu en section 3.2.1. La présentation des objets composés fait appel à la même méthode de la classe `Interval` que pour les objets primitifs mais sa définition consiste à organiser dans le temps la présentation des arguments de l'opérateur. La classe `Interval` regroupe la sémantique des opérateurs du langage et sa définition est donc complexe. L'utilisateur du langage TAO n'a pas besoin de se soucier de ces détails qui doivent être pris en compte par le mécanisme d'exécution dont nous parlerons en section 5.2.

4.2 Spécification temporelle du scénario

4.2.1 Notions préliminaires

Afin de pouvoir introduire la spécification temporelle du scénario, certaines notions des modèles temporels doivent tout d'abord être présentées. Ces notions seront reprises plus en détail en section 4.3.

Les objets multimédias sont ici considérés du point de vue du temps, en oubliant les autres aspects comme le positionnement spatial. Cette vue simplifiée des objets permet de se concentrer sur les aspects temporels afin de les interpréter plus clairement. La projection d'un objet multimédia sur l'axe temporel est appelée un *intervalle* et possède les caractéristiques temporelles suivantes:

- Chaque intervalle possède quatre points temporels, qui sont soit *observés* soit *générés* par la présentation multimédia. Les points temporels observés sont dénotés $\mathsf{beg}(A)$ et $\mathsf{end}(A)$, respectivement pour le début et la fin de l'intervalle A. De façon similaire, les points temporels générés sont $\mathsf{start}(A)$ et $\mathsf{stop}(A)$. Lorsqu'un opérateur est défini, les points temporels `beg` et `stop` sans arguments sont respectivement le début et la fin de l'intervalle retourné par l'opérateur. Le point `beg` est toujours avant le point `end` qui lui correspond.
- Les programmes peuvent générer des événements durant leur présentation. Ces événements sont spécifiés comme des points temporels.
- La durée d'un intervalle, c'est-à-dire l'écart entre sa fin et son début, peut être connue ou non au moment de sa définition.

Les intervalles sont dit primitifs lorsqu'ils correspondent à des objets primitifs et composés pour des objets composés.

4.2.2 Définition et sémantique d'un opérateur

Un opérateur est un élément syntaxique du langage qui dérive directement de celle présentée en [156, 85]. L'opérateur prend comme arguments un ou plusieurs intervalles, primitifs ou composés, et retourne un intervalle composé. L'intervalle

retourné synchronise ses arguments selon sa sémantique temporelle propre. Par la suite, les intervalles seront représentés par des rectangles, considérant que le temps s'écoule de gauche à droite, comme illustré en figure 4.2. Le rectangle englobant en traits pointillés est l'intervalle retourné par l'opérateur. Les rectangles à l'intérieur sont les intervalles arguments. Pour certains opérateurs, ces rectangles comportent des traits discontinus représentant différentes durées possibles de l'intervalle qui illustrent les cas limites de la sémantique de l'opérateur. Les points épais représentent les points temporels rattachés aux intervalles. On peut observer sur cette figure que $\mathsf{beg}(A)$ est un point temporel de A vu de l'extérieur et est donc une cause tandis que $\mathsf{start}(A)$ est vu de l'intérieur de cet intervalle A et est donc causé.

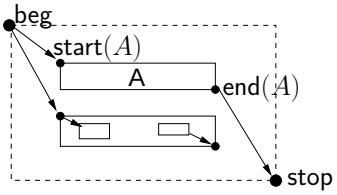


FIG. 4.2 – *Description générale d'un opérateur*

Une fois qu'un point temporel généré est crée, le système multimédia créé automatiquement le point temporel observé correspondant, start étant mis en correspondance avec beg et stop avec end, ce que nous appellerons par la suite la *mise en correspondance*. La différence entre les points temporels observés et générés provient des flèches internes qui lient des points temporels sur le schéma précédent. Cette relation entre les points temporels est appelée *relation causale* ou *relation de causalité* et est dénotée

$$p \mapsto q$$

p est un point temporel observé appelé *cause* et q un point temporel généré appelé *conséquence*. Cette relation causale signifie "quand p survient, alors q doit survenir immédiatement". La notation $p \mapsto q,r$ équivaut à $p \mapsto q$ & $p \mapsto r$, avec généralisation triviale à n'importe quel nombre de points temporels. Le symbole & est le connecteur de relation. Notons que la mise en correspondance établie par le système multimédia entre les points temporels généré et observé ne peut pas être exprimée comme une relation causale, car aucun point temporel généré ne peut être la cause d'une relation causale.

Les relations causales peuvent aussi être conditionnées. La relation causale

$$p \mapsto q \quad \textit{if} \quad c$$

signifie que la *causalité* $p \mapsto q$ n'a lieu que si c est *true*, sinon elle n'a pas lieu. Les équivalences et règles de distributivités suivantes s'appliquent:

$$\begin{array}{lcl} p \mapsto q \ \ if \ \ true & \equiv & p \mapsto q \\ p \mapsto q \ \ if \ \ false & \equiv & \text{absence de relation causale} \\ (r \ \ if \ \ c_1) \ \ if \ \ c_2 & \Leftrightarrow & r \ \ if \ \ c_1 \ \wedge \ c_2 \\ (r_1 \ \& \ r_2) \ \ if \ \ c & \Leftrightarrow & (r_1 \ \ if \ \ c) \ \& \ (r_2 \ \ if \ \ c) \end{array}$$

Ces conditions utilisent le test x while Y qui a pour valeur *true* si le point temporel x survient pendant que l'intervalle Y est présenté, *false* sinon.

Certains opérateurs ont un nombre variable d'arguments ou bien doivent identifier plusieurs occurrences de leurs arguments, ce qui nécessite d'indicer les intervalles sous la forme A_i. Cette notation des noms d'intervalle n'est qu'une convention que nous utilisons et est indépendante du mécanisme de nommage des objets que le système d'exécution des présentations multimédias peut utiliser. Les relations causales indicées ont la forme suivante:

$$\begin{array}{l} \forall i \in set, (\& \ causal \ rules \ ) \ \ if \ \ conditions \\ \forall i \in set, \ \wedge \ \ cond(i) \ (\& \ causal \ rules \ ) \ \ if \ \ conditions \end{array}$$

avec $set = I\!N$ ou $set = \{x_1 \ldots x_2\}$ pour $(x_1,x_2) \in I\!N^2$, ce dernier ensemble désignant les entiers entre x_1 et x_2; *cond* est un prédicat dépendant de i et utilisant les connecteurs de la logique propositionnelle ($\neg$, $\wedge$, $\vee$) et les fonctions usuelles sur les entiers (+, -, etc.). Dans le cas où *set* est vide, la condition est fausse. La portée de l'indice i s'étend du symbole $\forall$ à la fin de *conditions*. *conditions* exprime des comparaisons de points temporels indicés. Les mêmes indices peuvent être utilisés au début de *conditions*, de façon strictement équivalente à la précédente forme, ce qui donne:

$$\begin{array}{l} (\& \ causal \ rules \ ) \ \ if \ \ \forall i \in set, \ conditions \\ (\& \ causal \ rules \ ) \ \ if \ \ \forall i \in set, \ \wedge \ \ cond(i) \ conditions \end{array}$$

Cette forme-ci est utile lorsque les intervalles indicés n'apparaissent que dans la condition de la relation causale.

Notons que l'usage du quantificateur $\forall$ dans les conditions élève l'ordre de la logique de 0 à 1, en permettant de dénombrer des intervalles. Une expression à partir d'opérateurs et d'objets primitifs est dite *bien-formée* lorsque chaque opérateur a un nombre d'arguments correspondant à son arité et que chaque argument est un objet valide. L'expression est alors désignée comme un *scénario*.

4.2.3 Description des opérateurs

Les opérateurs prédéfinis du langage TAO sont décrits en utilisant les schémas précédemment introduits et définis grâce à l'ensemble de relations causales qu'ils imposent sur les points temporels de leurs arguments. Ils sont regroupés par type d'opérateurs, dont la sémantique partage des similitudes.

La notation pour les opérateurs est choisie pour rappeler leur sémantique et suit la convention suivante:

- pour un opérateur séquentiel
  $nom[paramètre](arguments)$ *paramètre* est un paramètre de la sémantique de l'opérateur et $arguments$ les expressions où se trouvent les intervalles à synchroniser. On notera par exemple $\mathsf{loop}[n](A)$.
- pour un opérateur parallèle
  $nom[arguments]$ On notera par exemple $\mathsf{loop}[n](A)$.
- *nom* désigne toujours le nom de l'opérateur.

4.2.3.1 Opérateur de delai

L'opérateur delay n'utilise qu'un paramètre et n'a aucun argument. Il retourne un objet multimédia qui n'a aucun contenu (dans tous les espaces de média) et a une durée égale à la valeur passée en paramètre. Le symbole $*$ est utilisé pour indiquer une durée infinie. Cette dernière valeur doit être utilisée en conjonction avec d'autres opérateurs pour exprimer l'attente d'un événement, sous peine de spécifier des éléments de la présentation multimédia qui ne se termineront pas.

C'est un opérateur identique à l'opérateur *delay* qui est un ajout de l'algèbre de processus *RT-LOTOS* par rapport à *LOTOS* [42].

Opérateur	Description
$\mathsf{delay}[d]$	delai de d unités de temps
$\mathsf{delay}[*]$	delai infini

4.2.3.2 Opérateurs séquentiels

Ces opérateurs arrangent les occurrences de leurs arguments en ordre séquentiel, les uns après les autres. L'opérateur "⨟" met en séquence ses arguments. Cette notation de l'opérateur de séquence rappelle le séparateur de déclaration d'un grand nombre de langages de programmation, qui indique effectivement pour les parties procédurales de code la séquence des instructions à exécuter. Cet opérateur est aussi utilisé dans le langage *LOTOS* [40] d'une manière similaire. L'opérateur $\mathsf{loop}(A)$ présente en boucle son unique argument et possède un paramètre optionnel qui permet d'indiquer le nombre de boucle désirées.

La séquence est généralisée à un nombre quelconque d'arguments par regroupement à gauche:

$$A \fatsemi B \fatsemi C = A \fatsemi (B \fatsemi C)$$

4.2.3.3 Opérateurs parallèles simples

Dans cet ensemble d'opérateurs binaires, les arguments des opérateurs commencent en parallèle. Des relations causales conditionnelles sont utilisées afin de déterminer quelles causalités vont avoir lieu selon les occurrences des fins des intervalles arguments. Certaines des relations causales introduites par ces

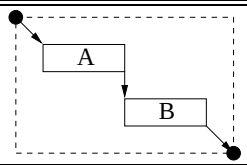
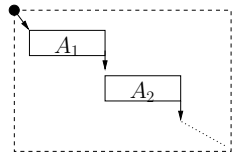
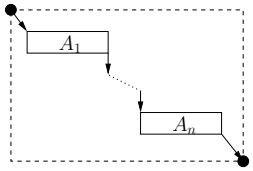
Opérateur	Schéma	Relations causales
$A \fatsemi B$		$\mathsf{beg} \mapsto \mathsf{start}(A)$ $\&\quad \mathsf{end}(A) \mapsto \mathsf{start}(B)$ $\&\quad \mathsf{end}(B) \mapsto \mathsf{stop}$
$\mathsf{loop}(A)$		$\mathsf{beg} \mapsto \mathsf{start}(A_1)$ $\forall j \in I\!N^*,$ $(\&\ \mathsf{end}(A_j) \mapsto \mathsf{start}(A_{j+1})\ )$
$\mathsf{loop}[n](A)$		$\mathsf{beg} \mapsto \mathsf{start}(A_1)$ $\forall j \in \{1 \ldots n-1\},$ $(\&\ \mathsf{end}(A_j) \mapsto \mathsf{start}(A_{j+1})\ )$ $\&\quad \mathsf{end}(A_n) \mapsto \mathsf{stop}$

FIG. 4.3 – *Opérateurs TAO séquentiels*

opérateurs créent un point généré d'arrêt stop d'un intervalle argument. Dans le cas d'un intervalle composé, cet arrêt, que nous appellerons *coupure*, possède une sémantique stricte qui sera introduite plus en détails en section 4.3.2.

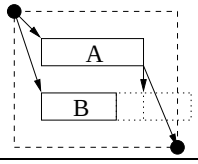
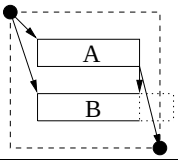
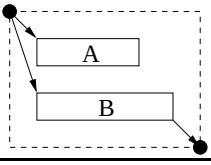
$\mathsf{master}[\ A \parallel B\ ]$		$\mathsf{beg} \mapsto \mathsf{start}(A),\mathsf{start}(B)$ $\&\quad \mathsf{end}(A) \mapsto \mathsf{stop}(B)\ \ \textit{if}\ \ \mathsf{end}(A)\ \mathsf{while}\ B$ $\&\quad \mathsf{end}(A) \mapsto \mathsf{stop}$
$\mathsf{min}[\ A \parallel B\ ]$		$\mathsf{beg} \mapsto \mathsf{start}(A),\mathsf{start}(B)$ $\&\quad \mathsf{end}(A) \mapsto \mathsf{stop}(B),\mathsf{stop}\ \ \textit{if}\ \ \mathsf{end}(A)\ \mathsf{while}\ B$ $\&\quad \mathsf{end}(B) \mapsto \mathsf{stop}(A),\mathsf{stop}\ \ \textit{if}\ \ \mathsf{end}(B)\ \mathsf{while}\ A$
$\mathsf{max}[\ A \parallel B\ ]$		$\mathsf{beg} \mapsto \mathsf{start}(A),\mathsf{start}(B)$ $\&\quad \mathsf{end}(A) \mapsto \mathsf{stop}\ \ \textit{if}\ \ \mathsf{end}(B)\ \mathsf{while}\ A$ $\&\quad \mathsf{end}(B) \mapsto \mathsf{stop}\ \ \textit{if}\ \ \mathsf{end}(A)\ \mathsf{while}\ B$

FIG. 4.4 – *Opérateurs TAO parallèles simples*

La généralisation à un nombre quelconque d'arguments est immédiate:

$$op[\ A \parallel B \parallel C\ ] = op[\ A \parallel op[\ B \parallel C\ ]\ ]$$

où *op* est un opérateur parallèle simple. Le regroupement se fait à gauche car il permet de donner un sens à la généralisation de l'opérateur master dont les deux opérandes ne jouent pas le même rôle. D'autre part, ce regroupement est identique à celui de l'opérateur de séquence.

4.2.3.4 Opérateur d'alternative

Cet opérateur possède un nombre quelconque d'arguments notés $(A_j)_{i=1\ldots n}$. Ses arguments débutent en parallèle et le premier intervalle A_i qui se termine (nous le verrons en section 4.3 qu'il ne peut y en avoir qu'un seul) stoppe les autres et démarre l'intervalle B_i d'indice correspondant. Tout comme pour les opérateurs parallèles simples, la sémantique de la coupure s'applique.

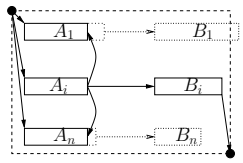
alt[$A_1 => B_1$ $\ldots$ $\parallel A_i => B_i$ $\ldots$ $\parallel A_n => B_n$]		$\forall i \in \{1\ldots n\}$, (& beg $\mapsto$ start(A_i)) $\forall i \in \{1\ldots n\}$, (& $\forall j \in \{1\ldots n\} \ \wedge \ j \neq i$, (& end$(A_i) \mapsto$ stop(A_j)) & end$(A_i) \mapsto$ start(B_i) & end$(B_i) \mapsto$ stop) *if* end(A_i) while A_j

FIG. 4.5 – *Opérateur TAO alternatif*

4.2.3.5 Opérateurs parallèle géneral et de lien

L'opérateur parallèle général par se distingue des opérateurs parallèles simples par la possibilité de nommer les diverses *branches* qui sont présentées en parallèle. Les opérateurs de lien permettent alors de stopper une branche, ou d'être stoppé par la fin d'une autre branche, du même opérateur par où est utilisé l'opérateur de lien. *ref*(b) dénote l'intervalle correspondant à la branche référencée par le nom b. L'événement v dans le scénario $A \nwarrow_{\{v\}}$ est soit la fin d'une branche b, que l'on note b.end, soit un événement e déclenché par un programme P, que l'on note $P.e$. Dans le cas d'un événement de fin de branche, la branche b doit être une branche de l'opérateur par où l'opérateur de lien apparaît, mais ne peut pas être la branche dans laquelle cet opérateur est défini. Si une expression à base d'opérateurs ne respecte pas cette condition, elle n'est pas valide.

Tout comme pour les opérateurs parallèles simples, la sémantique de la coupure s'applique. Notons que les branches et les opérateurs de lien ne permettent pas de définir n'importe quelle relation causale entre deux intervalles donnés, ce qui amènerait à des scénarios ardus à comprendre, voire temporellement incohérents. Contrairement aux autres opérateurs, les opérateurs de lien ont un

effet sur des intervalles qui ne sont pas leurs arguments. Cette action ressemble aux effets de bord des langages de programmation traditionnels.

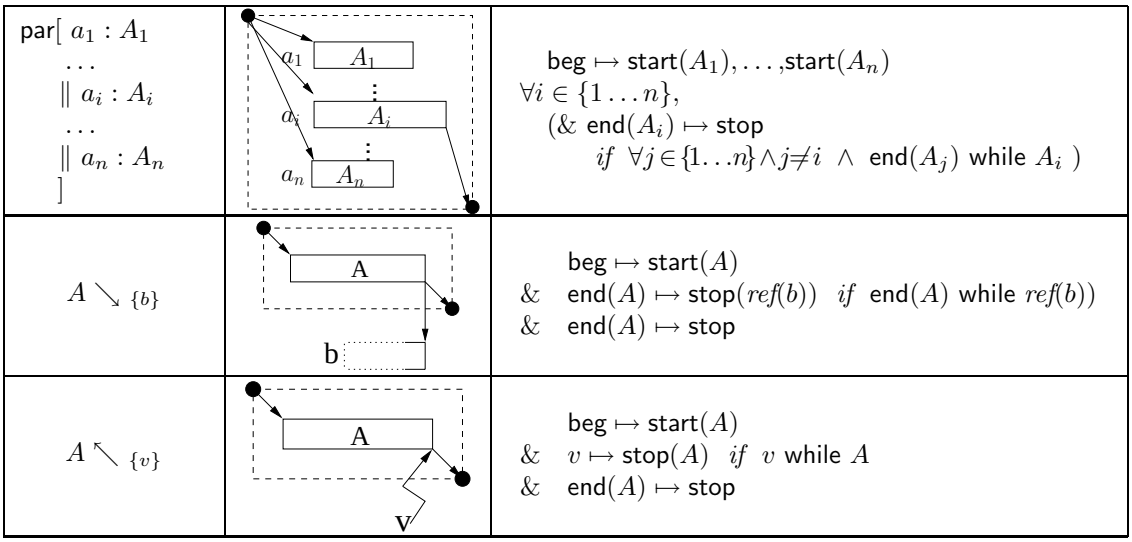


FIG. 4.6 – *Opérateurs TAO parallèle géneral et de lien*

4.3 Modèle temporel du langage

Après avoir décrit la synchronisation dans le langage TAO, nous donnons ici la définition de son modèle temporel. Après une discussion sur les entités de base formant le modèle temporel, nous expliciterons la sémantique temporelle complète des programmes TAO.

4.3.1 Les points et intervalles temporels

Nous avons déjà vu en section 4.2.1 les concepts fondamentaux du modèle temporel de TAO: des points temporels sont attachés aux intervalles et ils sont organisés par le biais de relations causales imposées par les opérateurs. Nous allons examiner plus en détail les intervalles et leurs points temporels.

4.3.1.1 Unicité spatio-temporelle

Les intervalles temporels doivent être identifiés de manière unique dans un scénario donné, ce que l'on appellera la propriété d'unicité spatio-temporelle: "Il ne peut exister qu'une seule et unique occurrence d'un objet, c'est-à-dire un intervalle, à un endroit et à un instant donné dans son étendue temporelle." Un objet peut toutefois être utilisé plusieurs fois dans le même scénario, chaque

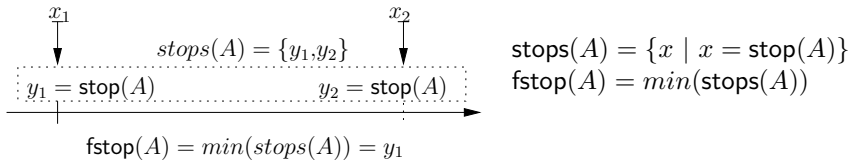
occurrence étant la répétition de la présentation de l'objet et correspondant à un intervalle distinct. La propriété d'unicité spatio-temporelle a des implications dans diverses situations. Considérons par exemple deux objets se chevauchant spatialement ou qui s'évanouissent (*fade in/out*, un effet de transition où des objets apparaissent et disparaissent graduellement). Du point de vue de l'espace de rendu, il n'y a que des points qui ont une valeur unique, une couleur pour un pixel dans le cas spatial ou une note pour un canal sonore, qui est calculée en combinant une partie des données des deux objets. Par contre à un haut niveau d'abstraction, cette situation doit être vue comme la présentation d'un objet composé qui définit le chevauchement ou l'évanouissement à partir des deux objets afin de respecter la propriété d'unicité spatio-temporelle. La propriété d'unicité spatio-temporelle implique de pouvoir désigner précisément et sans équivoque chaque occurrence, ce que le système multimédia doit assurer en assignant des noms uniques aux intervalles.

De plus, chaque intervalle a une propriété de connexité provenant de son étendue temporelle, c'est-à-dire qu'il est tel que tous les points se situant entre deux points de l'intervalle sont aussi dans l'intervalle. Autrement dit, l'intervalle ne comporte pas de trous. Cette propriété est nécessaire afin de définir un ordre total sur les intervalles, ce qui n'est pas possible lorsque les intervalles sont discontinus.

4.3.1.2 Le cas du point temporel fstop

Le point temporel $\mathsf{fstop}(A)$ est le premier point temporel $\mathsf{stop}(A)$ à survenir pendant la présentation de l'intervalle A Il est mis en correspondance avec le point $\mathsf{end}(A)$ par le système multimédia. Les autres points temporels $\mathsf{stop}(A)$ ne sont pas mis en correspondance avec le point $\mathsf{end}(A)$, c'est-à-dire qu'ils ne sont pas pris en compte pour l'arrêt de la présentation de A car ils correspondent à une intention de stopper l'intervalle qui ne peut être réalisée. La figure 4.7 illustre la définition du point $\mathsf{fstop}(A)$.

FIG. 4.7 – *Définition du point temporel* $\mathsf{fstop}(A)$



4.3.1.3 Relation d'ordre sur les points temporels

Les conditions des relations causales utilisent le prédicat while afin d'exprimer quel doit être la configuration des points temporels pour que la relation causale s'exécute. Le prédicat while doit être exprimé dans le modèle temporel

et nous définissons pour cela les relations $\prec$ et $\succ$ d'ordonnancement des points temporels comme, pour A un intervalle et p, q et x des points temporels:

- $\mathsf{start}(A) \prec \mathsf{beg}(A)$
- $\mathsf{fstop}(A) \prec \mathsf{end}(A)$
- $\forall p \mid p = \mathsf{stop}(A) \ \wedge \ p \neq \mathsf{fstop}(A), \ \ p \prec \mathsf{beg}(A) \ \vee \ \mathsf{end}(A) \prec p$
- $\forall (p,q) \mid (p,q) \neq (\mathsf{start}(A),\mathsf{beg}(A)) \ \wedge \ (p,q) \neq (\mathsf{fstop}(A),\mathsf{end}(A))$
  $\wedge \ (p,q) \neq (\mathsf{stop}(A),\mathsf{beg}(A)) \ \wedge \ (p,q) \neq (\mathsf{end}(A),\mathsf{stop}(A)),$
  $p \prec q \ \stackrel{def}{=} \ (p < q) \ \vee \ \exists x | ((p \mapsto x \ \wedge \ x \prec q) \ \vee \ (p \prec x \ \wedge \ x \mapsto q))$

La relation $\succ$ est la relation inverse de $\prec$.

Remarquons tout d'abord que la relation d'ordre $\prec$ est impliquée par la relation $<$ d'ordre naturel entre les points temporels, qui apparaît dans la quatrième formule de la définition de $\prec$. La condition est ajoutée à la relation $<$ permet d'ordonner des points à partir d'une relation d'ordre déjà connue ($x \prec q$) et d'une causalité ($p \mapsto x$). La causalité préserve ainsi l'ordre des points.

Cette relation d'ordre est l'unique partie du modèle temporel où apparaît le point fstop. Ce point sert à différencier le premier point $\mathsf{stop}(a)$ à survenir, les autres étant dénotés p dans la troisième formule de la définition de $\prec$ et se retrouvent soit avant $\mathsf{beg}(A)$ soit après $\mathsf{end}(A)$. La figure 4.8 illustre l'ordonnancement typique des divers points temporels pour un intervalle A donné.

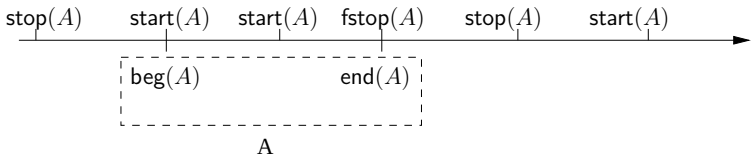


FIG. 4.8 – *Ordre des points temporels de l'intervalle A*

La condition "x while Y" correspond à l'expression $\mathsf{beg}(Y) \prec x \wedge x \prec \mathsf{end}(Y)$. Cette expression peut être simplifiée en tenant compte des autres relations causales introduites par l'opérateur, à l'exception des opérateurs de liens $\searrow$ et $\nwarrow$ qui ne permettent aucune simplification car aucune information n'est connue sur l'événement mis en jeu. Pour tous les autres opérateurs parallèles, les arguments de l'opérateur débutant au même instant, pour x la fin d'un des arguments et Y un autre argument, la condition $x \succ \mathsf{beg}(Y)$ est *true* et permet de simplifier " x while Y" en "$x \prec \mathsf{end}(Y)$". Cette simplification est utile afin d'indiquer plus clairement la signification de chaque relation causale dans le cadre d'un opérateur car l'on évite la redondance de certaines relations entre points

temporels. Elle est néanmoins inadéquate lorsque l'on désire examiner chaque relation causale indépendamment des autres, car l'information redondante que l'on élimine doit apparaître.

Les conditions des relations causales comprennent finalement des expressions de la forme $p \prec \mathsf{end}(s)$, $p \succ \mathsf{end}(s)$ et $p \succ \mathsf{beg}(A)$. Toutes les expressions sont des conjonctions ou disjonctions finies de ces comparaisons entre points temporels. Ces littéraux peuvent donc être écrits grâce à la relation $\prec$ et les trois opérateurs de la logique propositionnelle. Un tel formalisme logique a été prouvé décidable dans le domaine des réels [39]. Il est donc possible de déterminer une forme normale pour ces conditions, ce qui rend l'opération de fermeture décidable.

4.3.2 Normalisation de termes

La sémantique de chaque opérateur est l'ensemble de ses relations causales. Il faut maintenant définir quelle est la sémantique d'un programme TAO comportant plusieurs opérateurs. Un programme *TAO* est représenté par un *terme*, c'est-à-dire une expressions construite à partir d'opérateurs, de constantes (dont les intervalles primitifs) et de variables, auxquelles on peut assigner un terme. Une variable de terme dénote un sous-terme et est un terme à part entière. Cette structure de terme est isomorphe à celle d'un d'arbre, les nœuds étant les opérateurs et les feuilles les intervalles primitifs.

Afin de déterminer la sémantique du scénario, sa structure d'arbre va être transformée lors de l'application d'un processus de normalisation basé sur des règles de réécriture conditionnelles. Cette normalisation produit un ensemble de relations causales conditionnées par des comparaisons entre points temporels d'intervalles primitifs. Ce processus peut être vu comme un aplatissement de l'arbre correspondant au terme, les nœuds internes (correspondant aux opérateurs) étant éliminés et les relations causales étant remontées vers la racine de l'arbre. A la fin du processus, les relations causales ne s'appliquent plus qu'aux points temporels des feuilles de l'arbre, c'est-à-dire des intervalles primitifs.

Les règles de réécriture conditionnelles du processus de normalisation indiquent comment remonter le long des nœuds les relations causales ou modifier les ensembles de relations causales à chaque nœud. Une règle d'de réécriture conditionnelle est définie par un nom, une condition C et une substitution, composée d'une prémisse α et d'une conclusion β:

$$\text{If } C \text{ then } \frac{\alpha}{\beta}$$

Les règles de réécriture conditionnelles reposent sur un mécanisme de filtrage qui détermine si il existe une substitution des variables de la prémisse α permettant d'obtenir une partie du terme courant. Le fonctionnement est alors le suivant: si les conditions C sont satisfaites et que la prémisse α filtre un élément du

terme courant, alors cet élément peut être remplacé par la conclusion, suivant la substitution de variables qui a permis le filtrage.

Les règles de réécriture conditionnelles expriment le processus logique de normalisation. Dans l'absolu, elles sont indépendantes de leur ordre d'application et il est alors nécessaire de prouver la confluence des différentes règles de réécriture conditionnelles. Dans notre cas, chaque règle de réécriture conditionnelle opère sur des parties spécifiques des relations causales, ce qui permet d'assurer que les divers ordres d'application n'ont pas d'influence sur le résultat obtenu.

4.3.2.1 Notions préliminaires

Nous introduisons ici les notions d'opérateurs et formes fonctionnelles et de niveau de relations causales qui seront utilisées par la suite.

Opérateurs et formes fonctionnelles de termes TAO

Les termes écrits à partir d'opérateurs TAO ont une forme *syntaxique* que nous opposerons à une autre forme que nous nommerons *fonctionnelle*. Cette forme fonctionnelle décrit les termes sous la forme d'un opérateur fonctionnel appliqué à des arguments intervalles. Cet opérateur fonctionnel s'écrit sous la forme d'un nom suivi de paramètres indiqués entre crochets et qui correspondent aux paramètres de l'opérateur syntaxique ainsi que de ses arguments non-intervalles, c'est-a-dire les noms de branches et d'événements. L'opérateur fonctionnel *Id* désigne l'identité. La fonction $\Delta(\psi)$ renvoie pour un terme ψ sa forme fonctionnelle. La table 4.1 donne les opérateurs et formes fonctionnelles des termes TAO.

Forme syntaxique ψ	Opérateur fonctionnel	Forme fonctionnelle $\Delta(\psi)$
$A_1 \fatsemi \ldots \fatsemi A_n$	$\mathsf{seq}[n]$	$\mathsf{seq}[n](A_1,\ldots,A_n)$
$\mathsf{loop}(A)$	loop	$\mathsf{loop}(A)$
$\mathsf{loop}[n](A)$	$\mathsf{loop}[n]$	$\mathsf{loop}[n](A)$
$\mathsf{master}[\ A_1 \parallel \ldots \parallel A_n\ ]$	$\mathsf{master}[n]$	$\mathsf{master}[n](A_1,\ldots,A_n)$
$\mathsf{min}[\ A_1 \parallel \ldots \parallel A_n\ ]$	$\mathsf{min}[n]$	$\mathsf{min}[n](A_1,\ldots,A_n)$
$\mathsf{max}[\ A_1 \parallel \ldots \parallel A_n\ ]$	$\mathsf{max}[n]$	$\mathsf{max}[n](A_1,\ldots,A_n)$
$\mathsf{alt}[\ A_1 => B_1 \parallel \ldots \parallel A_n => B_n\ ]$	$\mathsf{alt}[n]$	$\mathsf{alt}[n](A_1,B_1\ldots,A_n,B_n)$
$\mathsf{par}[\ a_1 : A_1 \parallel \ldots \parallel a_n : A_n\ ]$	$\mathsf{par}[a_1,\ldots,a_n]$	$\mathsf{par}[a_1,\ldots,a_n](A_1,\ldots,A_n)$
$A \searrow_{\{b\}}$	$\mathsf{cut}[b]$	$\mathsf{cut}[b](A)$
$A \nwarrow_{\{v\}}$	$\mathsf{ended}[v]$	$\mathsf{ended}[v](A)$
A	Id	$Id(A)$

TAB. 4.1 – *Opérateurs et formes fonctionnelles des termes TAO*

Niveau de relations causales

Le nœud d'un arbre représentant un scénario est formalisé par la notion de

niveau de relations causales. Un niveau est dénoté par une paire $I.\rho$ où I est l'identificateur du nœud et ρ est un ensemble de relations causales attaché au nœud. L'identificateur est construit à partir d'une numérotation des nœuds de l'arbre du scénario de manière hiérarchique, comme illustré en figure 4.9. On ajoute à la suite de ce numéro, séparé par le symbole ":", la racine du terme, c'est-à-dire la valeur retournée par la fonction *rac*:

$$rac(\psi) = \begin{cases} \psi & \text{si } \psi \text{ est un terme constant} \\ \text{l'opérateur fonctionnel de } \psi & \text{si } \psi \text{ est un terme composé} \end{cases}$$

Pour l'exemple de la figure 4.9, on pourra ainsi avoir un identificateur $1.2.1 : \mathsf{seq}[2]$ pour un nœud qui correspond à un terme $A ; B$, les nœuds inférieurs ayant alors pour identificateurs $1.2.1.1 : A$ et $1.2.1.2 : B$. L'identificateur de niveau remplacera le nom de l'intervalle correspondant par la suite et sera utilisé de la même manière, en argument d'un point temporel par exemple. On parlera de niveau composé pour un niveau correspondant à un terme composé et de niveau terminal lorsqu'il correspond à un terme constant.

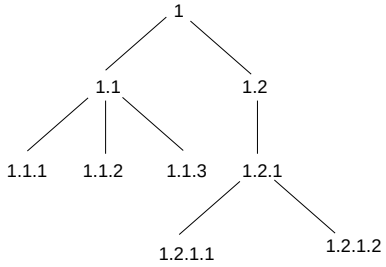


FIG. 4.9 – *Numérotation des nœuds d'un arbre*

Un niveau de relations causales $I.\rho$ est *valide* si et seulement si l'ensemble de relations causales ρ correspond à la sémantique du terme I. La définition des opérateurs donnée en section 4.2.3 correspond à des niveaux valides. Nous désignons par *sem* la fonction qui retourne cet ensemble de relations causales. Cette fonction est paramétrée par l'opérateur fonctionnel associé à l'opérateur TAO considéré et prend comme arguments des intervalles. On a par exemple:

$$\begin{aligned} &sem[\mathsf{seq}[2]](A,B) = \{\mathsf{beg} \mapsto \mathsf{start}(A) \ \&\ \mathsf{end}(A) \mapsto \mathsf{start}(B) \ \&\ \mathsf{end}(B) \mapsto \mathsf{stop}\} \\ &sem[\mathsf{loop}[n]](A) = \{\mathsf{beg} \mapsto \mathsf{start}(A_1) \\ &\quad \&\ \forall j \in \{1 \ldots n-1\}, (\&\ \mathsf{end}(A_j) \mapsto \mathsf{start}(A_{j+1})\ ) \ \&\ \mathsf{end}(A_n) \mapsto \mathsf{stop}\} \\ &sem[\mathsf{par}[a_1, \ldots, a_n]](A_1, \ldots, A_n) = \{\mathsf{beg} \mapsto \mathsf{start}(A_1), \ldots, \mathsf{start}(A_n) \\ &\quad \&\ \forall i \in \{1 \ldots n\}, (\&\ \mathsf{end}(A_i) \mapsto \mathsf{stop}\ \forall j \in \{1 \ldots n\}\ \wedge\ j \neq i\ \wedge\ \mathsf{end}(A_j)\ \mathsf{while}\ A_i\ )\} \end{aligned}$$

Les niveaux de relations causales sont hiérarchiquement organisés par la relation d'ordre provenant de la numérotation des arbres. Pour I, J et K des numéros, cette relation d'ordre est définie par:

$$I < J \stackrel{def}{=} (\exists K \mid I < K \ \wedge\ K < J) \ \vee\ (\exists l \in I\!N \mid I = J.l)$$

Pour r une relation causale et I un identificateur de niveau, la condition $r \in I$ teste si r est dans l'ensemble de relations causales du niveau désigné par I.

4.3.2.2 Structure du calcul de l'ensemble des relations causales

Pour un terme ψ, l'ensemble des relations causales lui correspondant est calculé grâce à la fonction $\mathit{Rel}(\psi)$ définie comme:

$$
\begin{array}{lcl}
\mathit{Rel}(\psi) & = & \mathit{Norm}(1.\mathit{rec}(\Delta(\psi))) \\
I.\mathit{rec}(\mathit{Id}(\psi)) & = & I : \psi.\emptyset \qquad \text{si } \psi \text{ est une feuille} \\
I.\mathit{rec}(op[p_1,\ldots,p_m](\psi_1,\ldots,\psi_n)) & = & \\
I{:}op[p_1,\ldots,p_m].sem[op[p'_1,\ldots,p'_m]](I.1{:}\mathit{rac}(\psi_1),\ldots,I.n{:}\mathit{rac}(\psi_n)) \cup \bigcup_{j\in\{1\ldots n\}} I.j.\mathit{rec}(\Delta(\psi_j)) &&
\end{array}
$$

$$
\text{avec } p'_j = \begin{cases} p_j & \text{si } p_j \in I\!N \text{ ou } \exists \texttt{Program } P \mid p'_j = P.e \\ I & \text{si } p_j = b \text{ et } \mathit{ref}(b) \text{ a pour identificateur } I \\ I.\mathit{end} & \text{si } p_j = b.\mathit{end} \text{ et } \mathit{ref}(b) \text{ a pour identificateur } I \end{cases}
$$

L'opérateur *rec* est utilisé pour calculer inductivement l'ensemble des sous-niveaux d'un terme. Il construit les identificateurs de niveaux et appelle les fonctions *sem* et Δ. La modification des paramètres p_j dans la troisième formule correspond au remplacement des références de branches b dans les opérateurs de lien par les identificateurs des nœuds correspondants.

Les niveaux construits à partir de *rec* sont ensuite modifiés par la fonction de normalisation *Norm*. Cette fonction de normalisation est définie à partir de règles de réécriture conditionnelles dont la prémisse et la conclusion sont un ensemble de niveaux. Les règles d'de réécriture conditionnelles expriment les modifications à apporter aux ensembles de relations causales des niveaux mis en jeu. La fonction *Norm* ne peut être définie inductivement car, dans la troisième formule définissant *Rel*, la sémantique de chacune des deux parties peut dépendre de celle de l'autre partie, selon les opérateurs mis en jeu. Elle doit donc s'appliquer donc à l'ensemble développé des niveaux.

4.3.2.3 Taxonomie des règles de réécriture conditionnelles

La normalisation est appliquée via des règles de réécriture conditionnelles qui sont un moyen logique plutôt qu'algorithmique de calculer l'ensemble des relations causales d'un terme. Certaines règles de réécriture conditionnelles requièrent que d'autres règles de réécriture conditionnelles aient été appliquées auparavant afin que leurs conditions soient vérifiées. Des optimisations du processus de normalisation sont possibles, par exemple pour les niveaux composés à partir d'opérateurs séquentiels car la sémantique de leurs sous-niveaux peut être calculée en parallèle, ces sous-niveaux étant indépendants les uns des autres.

Les règles de réécriture conditionnelles sont classées en deux catégories: les *règles contextuelles* et les *règles de coupure*. La règle de réécriture conditionnelle (lift) a une place particulière car elle est utilisée une fois que les autres règles de réécriture conditionnelles ont été appliquées, condition que l'on exprimera par le prédicat normal qui sera défini en section 4.3.2.5, afin de remonter les relations causales modifiées. Nous présentons ici le principe de ces règles de réécriture conditionnelles, leur expression étant donnée en annexe A.1.

Les règles contextuelles

Les règles contextuelles appliquent des transformations aux niveaux afin que les identificateurs de sous-niveaux soient remplacés par ceux des niveaux plus profonds. Elles seront appliquées plusieurs fois jusqu'à ce que seuls les noms des niveaux terminaux apparaissent. Elles sont au nombre de sept: (p-start), (p-end), (c-beg), (c-end), (p-event), (c-event) et (c-while).

Les règles contextuelles (p-start) and (p-end) transforment les points temporels $\mathsf{start}(I')$ et $\mathsf{end}(I')$ dans un niveau I en leurs valeurs données par les relations causales du sous-niveau I'. Ces valeurs sont respectivement la conséquence et la cause de la relation causale du sous-niveau I' dont les points beg et stop sont cause et conséquence. Les règles contextuelles (c-beg) et (c-end) opèrent une substitution similaire sur la condition de la relation causale.

Les règles contextuelles (p-event) and (c-event) transforment l'événement introduit par l'opérateur $\nwarrow$ en sa valeur correcte, respectivement dans la cause et la condition de la relation causale. Ceci ne peut survenir que quand l'événement est $b.\mathsf{end}$, car un événement de programme $P.e$ provient de l'extérieur du système multimédia et aucune information n'est disponible sur lui avant qu'il ne survienne. Cet événement n'est donc pas substitué.

Les règles contextuelles (c-end), (p-event) et (c-event) reposent sur le fait que tous les opérateurs TAO sont arrêtés par la fin d'un de leurs intervalles arguments et jamais par le début d'un intervalle argument, ce qui permet de considérer uniquement la relation causale $\mathsf{end}(J) \mapsto \mathsf{stop}$.

La règle contextuelle (c-while) permet de simplifier les conditions d'une relation causale lorsque deux conditions while sont exprimées sur un le même point avec un niveau et un de ses sous-niveaux.

Les règles de coupure

Les règles de coupure sont des règles de réécriture conditionnelles qui opèrent sur des ensembles de niveaux correspondant à la situation où un opérateur arrête, ou coupe, un de ses arguments qui est lui-même composé à partir d'un opérateur. Chaque règle de coupure repose donc sur une relation causale du type

$p \mapsto \mathsf{stop}(I : op)$ où op désigne l'opérateur fonctionnel qui est coupé. Ces règles de coupure divisent les opérateurs TAO en deux groupes:

- ceux qui ne sont pas à l'origine d'une coupure, $\fatsemi$, loop, loop$[n]$, max et par,
- et ceux qui le sont, master, min, alt, $\searrow$ et $\nwarrow$.

Ce critère de sélection n'est pas lié au fait que l'opérateur est séquentiel ou parallèle. Tous les opérateurs séquentiels n'engendrent pas de coupure mais parmi les opérateurs parallèles certains aussi n'en engendrent pas, comme max et par.

Les règles de coupure sont complémentaires des règles contextuelles (p-start) et (p-end) pour les relations causales où un point temporel stop d'un niveau composé apparaît. Elles transforment les conditions des relations causales et introduisent parfois de nouvelles relations causales. Une fois ces substitutions appliquées, les niveaux supérieurs ne coupent plus leurs sous-niveaux. Les relations causales modifiées peuvent alors être remontées vers un niveau supérieur via la règle de réécriture conditionnelle (lift).

4.3.2.4 La règle générique de coupure

Les règles de coupure sont déterminées en instanciant une règle générique pour chaque cas d'opérateur temporel. Cette règle générique de coupure exprime la sémantique de l'arrêt d'un opérateur quelconque. Elle est découpée en plusieurs étapes, définissant chacune une partie de la règle générique de coupure composée de substitutions simples à comprendre. La règle générique de coupure sera ensuite donnée dans son intégralité en superposant ces étapes.

L'opérateur qui est coupé est abstrait au sein de la règle générique de coupure. Des groupes de relations causales sur lesquels on applique les substitutions sont définis à partir de conditions qui seront résolues lors de l'instanciation de la règle générique de coupure à partir de la sémantique de l'opérateur instancié. Ces conditions sont plus complexes au niveau de l'opérateur générique de coupure que celles qui seront obtenues après instanciation avec un opérateur à cause de leur généricité.

Notations

Nous dénotons $a \triangleq b$ le fait que le symbole a est un raccourci syntaxique pour l'expression b dans toute expression de la règle générique de coupure, c'est-à-dire que partout où a apparaîtra, il sera substitué par le terme b.

Pour A une variable sur les points temporels utilisée dans la règle générique de coupure et b un certain point temporel, l'expression "$A \neq b$" signifie que A ne peut pas prendre la valeur b.

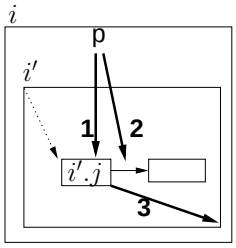
Pour I un identificateur de niveau donnée, nous utiliserons la notation $I.j$ pour désigner implicitement les nœuds fils de ce nœud, en déclarant l'indice j dans l'ensemble $\{1 \dots n\}$.

Les relations causales sont indicées au sein de la règle générique de coupure de la manière suivante:

- $\forall i \in set \mid C_i,\ r_i$
  signifie que chaque relation causale r_i remplit la condition C_i;
- $\forall r \in I \mid D,\ r$
  signifie que les relations causales r présente au niveau I remplissent les conditions D.

Ces notations sont uniquement syntaxiques et utilisées dans l'écriture de la règle générique de coupure. Lors de son instanciation pour chaque opérateur, les conditions sont résolues et déterminent l'ensemble des relations causales de l'opérateur qui entre en jeu dans la règle de coupure.

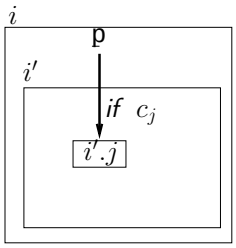
Etapes de la règle générique de coupure



La figure ci-contre illustre les trois étapes de la règle générique de coupure dans le cas où I et I' sont deux niveaux tels que $I' < I$, I contient la relation causale $p \mapsto \mathsf{stop}(I')$ et $I'.j$ est un sous-niveau de I'.

Etape 1: modification de la relation causale de coupure

$$\text{(cut-stop)}\quad \frac{I.\{\ p \mapsto \mathsf{stop}(I')\ \ \mathit{if}\ \ c\ \}}{I.\{\ \forall j \in \{1\ldots n\},\ (\&\ p \mapsto \mathsf{stop}(I'.j)\ \ \mathit{if}\ \ c_j\ )\ \}}$$



$c_j \triangleq c\ \wedge\ p$ while $I'.j$

Dans le niveau I, la relation causale ayant comme conséquence $\mathsf{stop}(I')$ est remplacée par autant de relations causales qu'il existe de sous-niveaux $I'.j$ du niveau I'. La condition c_j exprime le fait que le point temporel p ne peut stopper le niveau $I'.j$ que si cet arrêt est valide (ce qui correspond à la partie "c" de

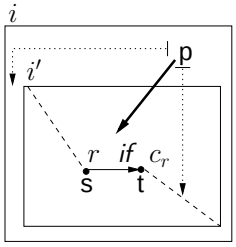
la condition) et s'il survient pendant que le niveau $I'.j$ est présenté (partie "p while $I'.j$").

Etape 2 (modification des sous-niveaux): ajout de conditions aux relations causales du niveau coupé

$$\text{(cut-sub)} \quad \frac{I'.\{\ \forall r \in I' \mid (s \neq \mathsf{beg}),\ (\&\ r\ )\ \}}{I'.\{\ \forall r \in I' \mid (s \neq \mathsf{beg}),\ (\&\ r \ \ \mathit{if}\ \ c_r\ )\ \}}$$

La condition c_r utilise une substitution de points temporels notée "[ ]" et qui permet d'écrire les comparaisons de points temporels sous une forme adéquate. Cette substitution est définie comme:

- $[\mathsf{beg}(I')]$ est substitué en la valeur du début de I' selon la définition de l'opérateur qui est instancié; lorsque cette substitution peut être appliquée plusieurs fois (ce qui est le cas pour les opérateurs parallèles), n'importe laquelle peut être choisie;
  cette substitution est équivalente à la règle contextuelle (c-beg);
- lorsque t est un point temporel $\mathsf{start}(J)$ pour J un numéro de niveau, $[t]$ est égal au point temporel $\mathsf{beg}(J)$; lorsque c'est un point temporel $\mathsf{stop}(J)$, $[t]$ est égal au point temporel $\mathsf{end}(J)$.



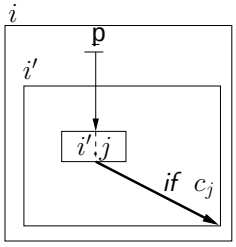
$$r \triangleq s \mapsto t \ \ \mathit{if}\ \ c'$$
$$c_r \triangleq (c\ \wedge\ (p \prec [\mathsf{beg}(I')]\ \vee\ p \succ [t]))\ \vee\ \neg c$$

Les relations causales r du niveau courant du terme I' qui n'ont pas le point temporel beg pour cause sont modifiées en ajoutant la condition c_r. Cette condition exprime le fait que r n'aura lieu que si p arrête I' (partie "c" de la condition), qu'il survient avant son début (partie "$p \prec [\mathsf{beg}(I')]$") ou bien après la relation causale r (partie "$p \succ [t]$") ou finalement qu'il ne l'arrête pas (partie "$\neg c$"). c_r peut être simplifiée en:

$$c_r \triangleq p \prec [\mathsf{beg}(I')]\ \vee\ p \succ [t]\ \vee\ \neg c$$

Etape 3 (modification des sous-niveaux): ajout de relations causales terminant le niveau coupé

(cut-end) $\dfrac{}{I'.\{\ \forall j \in \{1\ldots n\},\ (\&\ \mathsf{end}(I'.j) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_j\ )\ \}}$



Finalement, on ajoute que les niveaux $I'.j$ stoppés par p peuvent à leur tour causer la fin du niveau I'.

Etape 4: modification de la règle générique (cut-end)

La règle générique (cut-end) introduit de nouvelles relations causales dans les sous-niveaux $I'.j$ de I' lorsque ces sous-niveaux ne terminent pas le niveau I'. Ces relations causales devraient à leur tour être substituées par la règle générique (cut-sub), ce qui donnerait une relation causale dont la condition est:

$c_j\ \wedge\ (p \prec [\mathsf{beg}(I')]\ \vee\ p \succ \mathsf{end}(I'.j)\ \vee\ \neg c)$
$= (c\ \wedge\ p\ \mathsf{while}\ I'.j)\ \wedge\ (p \prec [\mathsf{beg}(I')]\ \vee\ p \succ \mathsf{end}(I'.j)\ \vee\ \neg c)$
$= c\ \wedge\ p\ \mathsf{while}\ I'.j\ \wedge\ (p \prec [\mathsf{beg}(I')]\ \vee\ p \succ \mathsf{end}(I'.j))$
$= \mathit{false}$

Ceci invaliderait l'ajout de ces relations causales. La règle générique (cut-sub) doit donc être modifiée pour ne pas traiter ces relations causales ajoutées. Sa premisse est réécrite en:

$$I'.\{\ \forall r \in I'\ |\ D_r,\ (\&\ r\ )\ \}$$
$$\text{avec}\quad D_r \triangleq s{\neq}\mathsf{beg}\ \wedge\ (\forall j \in \{1\ldots n\},\ s{\neq}\mathsf{end}(I'.j))\ \wedge\ t{\neq}\mathsf{stop}$$

Les relations causales ajoutées sont traitées par la règle générique suivante:

$$\frac{I'.\{\ \forall j \in \{1\ldots n\},\ (\&\ \mathsf{end}(I'.j) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c'_j\ )\ \}}{I'.\{\ \forall j \in \{1\ldots n\},\ (\&\ \mathsf{end}(I'.j) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c''_j\ )\ \}}$$

$$c''_j \triangleq c_j\ \vee\ (\neg c_j\ \wedge\ \textstyle\bigwedge_{E_{jk}} \neg c_k\ \wedge\ c'_j)$$
$$E_{jk} \triangleq k \in \{1\ldots\}\ \wedge\ (\neg c\ \vee\ \textstyle\bigwedge_{E_{jk}} (p \prec \mathsf{beg}(I'.k)\ \vee\ p \succ \mathsf{end}(I'.k)))\ \wedge\ c'_j)$$

La condition c''_j exprime le fait que le sous-niveau $I'.j$ peut arrêter le niveau I' aussitôt qu'il est lui-même arrêté par le point p (partie c_j) ou bien dans le cas opposé (partie $\neg c_j$) il arrête I' selon la définition des relations causales dans ce niveau (partie c'_j)) si p n'a pas arrêté un sous-niveau avant $I'.j$ (partie

$\bigwedge_{E_{jk}} \neg c_k$). La condition c''_j est simplifiée en:

$$\begin{aligned} c''_j &\triangleq c_j \ \vee \ (\neg c_j \ \wedge \ \bigwedge_{E_{jk}} \neg(c \ \wedge \ p \ \mathsf{while} \ I'.k) \ \wedge \ c'_j) \\ &\triangleq c_j \ \vee \ (\neg c_j \ \wedge \ (\neg c \ \vee \ \bigwedge_{E_{jk}} (p \prec \mathsf{beg}(I'.k) \ \vee \ p \succ \mathsf{end}(I'.k))) \ \wedge \ c'_j) \\ &= c_j \ \vee \ (\neg c_j \ \wedge \ (\neg c \ \vee \ p \prec [\mathsf{beg}(I')] \ \vee \ p \succ \mathsf{beg}(I'.j)) \ \wedge \ c'_j) \\ &= c_j \ \vee \ (\neg c \ \wedge \ (p \prec [\mathsf{beg}(I')] \ \vee \ p \succ \mathsf{beg}(I'.j)) \ \wedge \ c'_j) \end{aligned}$$

Entre les lignes 2 et 3, l'expression $\bigwedge_{E_{jk}} \neg c_k$ est simplifiée selon le schéma suivant:

- Dans le cas de deux sous-niveaux, on a:
  $$\begin{aligned} &((p \prec \mathsf{beg}(I'.1) \ \vee \ p \succ \mathsf{end}(I'.1)) \ \wedge \ (p \prec \mathsf{beg}(I'.2) \ \vee \ p \succ \mathsf{end}(I'.2))) \\ &= p \prec \mathsf{beg}(I'.1) \ \vee \ (p \succ \mathsf{end}(I'.1) \ \wedge \ p \prec \mathsf{beg}(I'.2)) \ \vee \ p \succ \mathsf{end}(I'.2) \end{aligned}$$
  Le test qui se trouve au centre a pour valeur *false* car à tout moment de la présentation d'un niveau, au moins un de ses sous-niveaux est présenté.
- En généralisant à un nombre quelconque de sous-niveaux ce raisonnement, on obtient:
  $\bigwedge_{E_{jk}} (p \prec \mathsf{beg}(I'.j) \ \vee \ p \succ \mathsf{end}(I'.j)) = p \prec \mathsf{beg}(I'.1) \ \vee \ p \succ \mathsf{end}(I'.k)$
  avec $I'.1$ un des premiers niveaux présentés au sein du niveau I' (c'est-à-dire dont le début est généré par $[\mathsf{beg}(I')]$) et $I'.k$ un des derniers niveaux valides selon la condition $\mathsf{end}(I'.k) \prec \mathsf{beg}(I'.i)$.
  On peut finalement remplacer avantageusement les points temporels $\mathsf{beg}(I'.1)$ et $\mathsf{end}(I'.k)$ respectivement par $[\mathsf{beg}(I')]$ et $\mathsf{beg}(I'.j)$ sans enfreindre les contraintes. Ceci permet d'éliminer la particularité des niveaux $I'.1$ et $I'.k$.

Enfin, tout comme à l'étape 2, on peut éliminer l'expression $\neg c$ car c_j implique c, et donc finalement:

$$c''_j \triangleq c_j \ \vee \ ((p \prec [\mathsf{beg}(I')] \ \vee \ p \succ \mathsf{beg}(I'.j)) \ \wedge \ c'_j)$$

Cette règle générique supplémentaire inclut en fait la (cut-end), qui est obtenue si l'on choisit $c'_j = \mathit{false}$. Elle peut donc remplacer cette règle générique de manière cohérente. La valeur *false* de c'_j correspond à la première application de la règle générique lors de la normalisation, lorsque ces nouvelles relations causales n'apparaissent pas et l'ensemble des relations causales est celui de la sémantique de l'opérateur du niveau. Par contre, c'_j prend d'autres valeurs lors des applications ultérieures de la règle générique, puisque la conclusion de la règle générique ajoute cette relation causale.

Remarque

Les seules relations causales qui n'ont pas été traitées dans la règle générique de coupure sont celles dont beg est la cause. Ces relations causales ne sont en fait pas modifiées car elles ne possèdent jamais de conditions. Si toutes les relations causale du niveau I' ont soit beg pour cause, soit end pour conséquence (pour les opérateurs max et par par exemple) alors la règle générique (cut-sub) n'aura aucun effet et seules les relations causales générant les points stop des sous-niveaux du niveau i' seront traités par (cut-stop) et (cut-end). Dans le cas

opposé, il existe au moins une relation causale qui sera modifiée par la règle générique (cut-sub).

Ecriture de la règle générique de coupure

Les trois étapes présentées l'ont été dans un ordre précis afin d'expliquer la modification de l'étape 3 dans le but de la rendre compatible avec l'étape 2. La règle générique de coupure est obtenue en appliquant les trois substitutions dans n'importe quel ordre, les étapes ne dépendant finalement plus les unes des autres.

$$
\begin{array}{l}
\boldsymbol{I}.\{\ p \mapsto \mathsf{stop}(I')\ \ \mathit{if}\ \ c\ \} \\
\cup \\
\boldsymbol{I'}.\{\ \forall r \in I' \mid D_r,\ (\&\ r\ ) \\
\quad \&\ \forall j \in \{1 \ldots n\},\ (\&\ \mathsf{end}(I'.j) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c'_j\ )\ \} \\
\hline
\boldsymbol{I}.\{\ \forall j \in \{1 \ldots n\},\ (\&\ p \mapsto \mathsf{stop}(I'.j)\ \ \mathit{if}\ \ c_j\ ) \\
\cup \\
\boldsymbol{I'}.\{\ \forall r \in I' \mid D_r,\ (\&\ r\ \ \mathit{if}\ \ c_r\ ) \\
\quad \&\ \forall j \in \{1 \ldots n\},\ (\&\ \mathsf{end}(I'.j) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c''_j\ )\ \} \\
\\
c_j \quad \triangleq c\ \wedge\ p\ \mathsf{while}\ I'.j \\
r \quad \triangleq s \mapsto t\ \ \mathit{if}\ \ c' \\
D_r \quad \triangleq s \neq \mathsf{beg}\ \wedge\ (\forall j \in \{1 \ldots n\},\ s \neq \mathsf{end}(I'.j))\ \wedge\ t \neq \mathsf{stop} \\
c_r \quad \triangleq p \prec [\mathsf{beg}(I')]\ \vee\ p \succ [t]\ \vee\ \neg c \\
c''_j \quad \triangleq c'_j\ \vee\ ((p \prec [\mathsf{beg}(I')]\ \vee\ p \succ \mathsf{beg}(I'.j))\ \wedge\ c'_j)
\end{array}
$$

TAB. 4.2 – *Écriture de la règle générique de coupure*

Instanciation de la règle générique de coupure

Lors de l'instanciation de la règle générique de coupure, les relations causales r sont déterminées à partir de la sémantique de l'opérateur considéré en résolvant la condition D_r. Les conditions c_j, c_r et c''_j peuvent ensuite être simplifiées. De plus pour l'opérateur min, des relations causales peuvent être regroupées, car tout ses intervalles arguments sont présentés en parallèle. Les règles de coupure sont données en annexe A.1.

4.3.2.5 Arrêt de la normalisation

Notons tout d'abord que l'examen du terme par la normalisation peut être arrêté à un certain niveau en nommant son ensemble de relations causales sans en donner la valeur. Cette non-instanciation permet d'exclure ce niveau de la

normalisation, par exemple pour indiquer que cet ensemble est inconnu au moment de la normalisation.

L'arrêt du processus de normalisation a lieu lorsque le terme à normaliser est sous une forme dite *normalisée*. Pour $I.\rho$ le niveau courant, nous dénoterons cette condition par le prédicat $\mathsf{normal}(I)$ qui a pour valeur *true* lorsque:

- I est une feuille ($\nexists J \mid J < I$) ou une variable de terme non-instanciée;
- ou $\forall J \mid J < I$:
  - $\mathsf{normal}(J)$ a pour valeur *true*;
  - $\forall r \in J$, la règle de réécriture conditionnelle (lift) de remontée des relations causales dans le niveau supérieur doit avoir été appliquée (éventuellement après d'autres règles de réécriture conditionnelles);
  - et l'identificateur J n'est pas utilisé dans les points temporels ou les conditions du niveau I.

Les premier, deuxième et quatrième points de la définition signifient que toutes les relations causales du niveau I ne s'appliquent qu'à des niveaux terminaux. Le troisième élément signifie qu'aucune relation causale n'a été perdue lors de la normalisation, leurs remontées s'étant faite grâce à une règle de réécriture conditionnelle (lift). Appliqué au niveau le plus haut, le prédicat normal représente la condition d'arrêt de la normalisation. La forme normalisée est alors obtenue en ne retenant que l'ensemble des relations causales de ce niveau:

$$\text{si } (\nexists J \mid I < J) \wedge \mathsf{normal}(I) \text{ alors } Norm(I.\rho \cup_{(J \mid J<I)} J.\rho_J) = \rho$$

En pratique, les règles contextuelles sont appliquées en premier et éliminent les références aux sous-niveaux dans les points temporels beg, end et start ainsi que dans les conditions des relations causales. Les règles de coupure sont ensuite appliquées afin d'éliminer les coupures, et donc les identificateurs des niveaux inférieurs dans les relations causales dont les conséquences comportent un point stop. Les relations causales sont enfin remontées dans les niveaux supérieurs grâce à la règle de réécriture conditionnelle (lift).

4.3.2.6 Exemple de normalisation d'un terme

Nous donnons ici le détail de la normalisation du terme $S = \mathsf{master}[\ A \parallel B \fatsemi C\ ]$.
$Rel(S) = Norm(Set)$

$$\begin{aligned}
&Set = 1.rec(\Delta(S)) = 1.rec(\mathsf{master}[2](A,\ B \fatsemi C))\\
&= 1{:}\mathsf{master}[2].sem[\mathsf{master}[2]](1.1{:}A,\ 1.2{:}\mathsf{seq}) \cup 1.1.rec(\Delta(A))\\
&\qquad \cup 1.2.rec(\Delta(B \fatsemi C))\\
&= 1{:}\mathsf{master}[2].sem[\mathsf{master}[2]](1.1{:}A,\ 1.2{:}\mathsf{seq}) \cup 1.1.rec(Id(A))\\
&\qquad \cup 1.2.rec(\mathsf{seq}[2](B,\ C))\\
&= 1{:}\mathsf{master}[2].sem[\mathsf{master}[2]](1.1{:}A,\ 1.2{:}\mathsf{seq}) \cup 1.1{:}A.\emptyset\\
&\qquad \cup 1.2{:}\mathsf{seq}[2].sem[\mathsf{seq}[2]](1.2.1{:}B,\ 1.2.2{:}C)) \cup 1.2.1{:}B.\emptyset \cup 1.2.2{:}C.\emptyset
\end{aligned}$$

$$
\begin{aligned}
Set = {} & 1{:}\mathsf{master}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.1:A),\mathsf{start}(1.2:\mathsf{seq}) \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2:\mathsf{seq})\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq} \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}\ \} \\
& \cup\ 1.2:\mathsf{seq}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.2.1:B)\ \&\ \mathsf{end}(1.2.1:B) \mapsto \mathsf{start}(1.2.2:C) \\
& \quad \&\ \mathsf{end}(1.2.2:C) \mapsto \mathsf{stop}\ \} \\
& \cup\ 1.1:A.\emptyset\ \cup\ 1.2.1:B.\emptyset\ \cup\ 1.2.2:C.\emptyset
\end{aligned}
$$

L'ensemble *Set* étant complètement développé, on commence par simplifier l'expression en appliquant la règle contextuelle (p-start), ce qui remplace le début de la séquence par celui de l'intervalle B:

$$
\begin{aligned}
Set = {} & 1{:}\mathsf{master}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.1:A),\mathsf{start}(1.2.1:B) \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2:\mathsf{seq})\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq} \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}\ \} \\
& \cup\ 1.2:\mathsf{seq}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.2.1:B)\ \&\ \mathsf{end}(1.2.1:B) \mapsto \mathsf{start}(1.2.2:C) \\
& \quad \&\ \mathsf{end}(1.2.2:C) \mapsto \mathsf{stop}\ \} \\
& \cup\ 1.1:A.\emptyset\ \cup\ 1.2.1:B.\emptyset\ \cup\ 1.2.2:C.\emptyset
\end{aligned}
$$

La normalisation applique ensuite la règle de coupure (cut-seq):

$$
\begin{aligned}
Set = {} & 1{:}\mathsf{master}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.1:A),\mathsf{start}(1.2.1:B) \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.1:B)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq} \\
& \qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.1:B \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.2:C)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq} \\
& \qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.2:C \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}\ \} \\
& \cup\ 1.2:\mathsf{seq}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.2.1:B) \\
& \quad \&\ \mathsf{end}(1.2.1:B) \mapsto \mathsf{start}(1.2.2:C)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A) \prec \mathsf{beg}(1.2.1:B) \\
& \qquad \vee\ \mathsf{end}(1.1:A) \succ \mathsf{beg}(1.2.2:C)\ \vee\ \neg(\mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq}) \\
& \quad \&\ \mathsf{end}(1.2.1:B) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.1:B \\
& \qquad\qquad\qquad\qquad \vee\ \neg(\mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq}) \\
& \quad \&\ \mathsf{end}(1.2.2:C) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (\mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq} \\
& \quad \wedge\ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.2:C)\ \vee\ (\neg(\mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq}) \\
& \qquad \wedge\ (\mathsf{end}(1.1:A) \prec \mathsf{beg}(1.2.1:B)\ \vee\ \mathsf{end}(1.1:A) \succ \mathsf{beg}(1.2.2:C)))\ \} \\
& \cup\ 1.1:A.\emptyset\ \cup\ 1.2.1:B.\emptyset\ \cup\ 1.2.2:C.\emptyset
\end{aligned}
$$

Toutes les coupures ayant été appliquées, il nous est alors possible de remonter les relations causales via la règle contextuelle (lift), seule celle liant la fin de B au début de C étant concernée dans notre cas. Après plusieurs applications de la règle contextuelle (c-while) permettent de simplifier les conditions while imbriquées, on obtient:

$$
\begin{aligned}
Set = {} & 1{:}\mathsf{master}[2].\{\ \mathsf{beg} \mapsto \mathsf{start}(1.1:A),\mathsf{start}(1.2.1:B) \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.1:B)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.1:B \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.2:C)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.2:C \\
& \quad \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop} \\
& \&\ \mathsf{end}(1.2.1:B) \mapsto \mathsf{start}(1.2.2:C)\ \ \mathit{if}\ \ \mathsf{end}(1.1:A) \prec \mathsf{beg}(1.2.1:B) \\
& \quad \vee\ \mathsf{end}(1.1:A) \succ \mathsf{beg}(1.2.2:C)\ \vee\ \neg(\mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2:\mathsf{seq})\ \}
\end{aligned}
$$

Il ne reste plus alors qu'à éliminer la dernière référence au niveau $1.2:\mathsf{seq}$ qui se trouve dans la dernière relation causale. On applique alors les règles contextuelles (c-beg) et (c-end) afin de ne plus faire apparaître que les niveaux terminaux $1.2.1:B$ et $1.2.2:C$. La règle contextuelle (c-while) permet de simplifier la condition de la dernière relation causale en:

$$\mathsf{end}(1.1:A) \prec \mathsf{beg}(1.2.1:B) \ \vee \ \mathsf{end}(1.1:A) \succ \mathsf{beg}(1.2.2:C)$$

Finalement la normalisation donne:

$$\begin{aligned} \mathit{Rel}(S) = \{\ & \mathsf{beg} \mapsto \mathsf{start}(1.1:A), \mathsf{start}(1.2.1:B) \\ & \ \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.1:B) \ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.1:B \\ & \ \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop}(1.2.2:C) \ \ \mathit{if}\ \ \mathsf{end}(1.1:A)\ \mathsf{while}\ 1.2.2:C \\ & \ \&\ \mathsf{end}(1.1:A) \mapsto \mathsf{stop} \\ \&\ & \mathsf{end}(1.2.1:B) \mapsto \mathsf{start}(1.2.2:C) \ \ \mathit{if}\ \ \mathsf{end}(1.1:A) \prec \mathsf{beg}(1.2.1:B) \\ & \ \vee\ \mathsf{end}(1.1:A) \succ \mathsf{beg}(1.2.2:C)\ \} \end{aligned}$$

4.4 Positionnement spatial

La plus grande partie du travail de création du langage de spécification de présentations multimédia TAO a porté sur la spécification temporelle. La spécification de la composition spatiale dans TAO est très classique en reprenant le modèle des fenêtres imbriquées, que l'on retrouve dans le langage Video Algebra et sous une forme différente dans le langage SMIL.

Le composition spatiale part de la définition de l'espace d'affichage comme un ensemble de pixels. Chaque pixel étant représenté par ses coordonnées sous la forme de deux arguments réels dans le sous-ensemble des réels $[0,1] \cup +\infty$. L'intervalle [0,1] correspond à la valeur d'une coordonnée d'un point dans son espace d'affichage, alors que la valeur $+\infty$ indique que cette valeur n'est pas fixée lors de la définition de la présentation multimédia. La convention suivante est prise: le point (0,0) correspondant au coin supérieur gauche de l'espace d'affichage et (1,1) au coin inférieur droit. Les points sont positionnés de manière relative à l'espace d'affichage de sorte que les données spatiales peuvent être simplement mises à l'échelle lors de la modification de l'espace d'affichage, un agrandissement de la fenêtre de rendu par exemple.

Des fenêtres sont ensuite spécifiées au sein de cet espace et sont attachées à des objets qui permettent la définition de sous-fenêtres pour les objets qui les composent temporellement. L'opérateur spatial `window` permet de construire une fenêtre et accepte comme paramètres un ou deux couples de coordonnées. Lorsque deux couples de coordonnées sont indiquées, ils sont interprétés comme des points opposés de la fenêtre, leur place dans la fenêtre étant déterminée automatiquement par leur position relative. Lorsqu'un seul couple de coordonnées est indiqué, il est interprété comme les coordonnées du point supérieur gauche de la fenêtre et le point opposé inférieur droit est laissé libre dans le cadre courant,

autrement dit ses coordonnées sont (+∞,+∞). Les données médias devant être affichées dans la fenêtre déterminent la position de ce point lors de l'exécution.

Les fenêtres sont finalement affectées aux attributs `window` des objets:

```
Image1.window = window(25,7) ;
Image2.window = window(0,0,100,200) ;
```

L'attribut `window` peut avoir une signification particulière pour certains objets comme le programme `ClickProgram` qui réagit aux clics de souris dans la zone définie par la fenêtre.

4.5 Exemples de programmes TAO

Nous donnons ici quelques exemples de programmes TAO illustrant les différents aspects du langage.

Tout d'abord trois présentations multimédias correspondant chacune à un type de bouton dont le comportement est indiqué par l'usage des opérateurs temporels et les événements auxquels il répond. Pour le premier bouton `Switch1`, `Bup` et `Bdown` sont les deux images du bouton respectivement dans ses positions hautes et basses, `BControler` est le programme de contrôle qui retourne les occurrences des points temporels `onClick` et `onRelease` lorsque le bouton gauche de la souris est cliqué et relâché dans la zone de l'écran où sont positionnées les images. Le bouton `Switch1` présente en parallèle le programme de contrôle et une boucle sur la séquence des deux images arrêtées par un événement du contrôleur:

```
Program BC = new ClickProgram() ;
  BC.window = window(0,0,50,50) ;
Picture Bup = new Image("pics/image1.bmp") ;
  Bup.window = window(0,0) ;
Picture Bdown = new Image("pics/image2.jpg") ;
  Bdown.window = window(0,0) ;
Program Switch1 ;
TemporalPoint Switch1.onClick, Switch1.onRelease ;

Switch1 = par [ BControler : BC
              || BImage : loop( Bup ↖ {BC.OnClick} ° 9
                                Bdown ↖ {BC.OnRelease}) ]
           .{ onClick = BC.onClick;
              onRelease = BC.onRelease } ;
```

Le deuxième bouton `Switch2` a un comportement légèrement différent car il ne réagit qu'à l'événement `OnClick`, se comportant ainsi comme un interrupteur:

```
Switch2 = par [ BConstroler : BC
              || BImage : loop( Bup ↖ {BC.OnClick} °
                                Bdown ↖ {BC.OnClick} ) ]
          .{ onClick = BC.onClick } ;
```

Le troisième objet `RadioSwitches`, ou bouton radio, est composé de deux boutons simples. Les boutons simples sont groupés de façon à ce qu'un seul d'entre eux soit actif à un moment donné. Le bouton radio possède deux groupes de deux images et un programme de contrôle gérant quatre événements, deux pour chaque bouton simple. Ce comportement en exclusion mutuelle est plus compliqué à spécifier car il est nécessaire d'exprimer les divers états du groupe: le bouton radio est tout d'abord présenté sans aucun bouton simple enfoncé (branches `BImages1` et `BImages2`) et lorsqu'un des deux boutons simples est cliqué, une boucle démarre alternant entre le bouton simple 1 en position haute puis le bouton simple 2 en position basse (branche `BImages3`) et la situation inverse (branche `BImages4`).

```
Program BC1 = new ClickProgram() ;
  BC1.window = window(0,0,25,25) ;
Program BC2 = new ClickProgram() ;
  BC2.window = window(25,0,50,50) ;
Point BC1.onClick, BC2.onClick, RadioSwitches.onClick ;)
Picture B1u = new Image("pics/image1up.jpg") ;
  B1u.window = window(0,0) ;
Picture D1d = new Image("pics/image1down.jpg") ;
  D1d.window = window(0,0) ;
Picture B2u = new Image("pics/image2up.jpg") ;
  B2u.window = window(0,0) ;
Picture D2d = new Image("pics/image2down.jpg") ;
  D2d.window = window(0,0) ;
Program RadioSwitches ;
TemporalPoint RadioSwitches.onClick ;

RadioSwitches = par [ BControler : par [ b1:BC1 || b2:BC2 ]
   || BImages1 : par [ b1:B1d || b2:B2d ] ↖ {BC1.onClick}
   || BImages2 : delay[*] ↖ {BC2.onClick}
   || BImages3 : delay[*] ↖ {BImages1.end} °
                 loop ( par [ a1: B1u || a2: B2d] ↖ {BC2.onClick} °
                        par [ a1: B1d || a2: B2u] ↖ {BC1.onClick})
   || BImages4 : delay[*] ↖ {BImages2.end} °
                 loop ( par [ a1: B1d || a2: B2u] ↖ {BC1.onClick} °
                        par [ a1: B1u || a2: B2d] ↖ {BC2.onClick}).
          .{ OnClick = BC1.OnClick, BC2.onClick } ;
```

A noter que dans la plupart des langages, ces boutons sont fournis par une librairie standard, encapsulant une version plus efficace à l'exécution. Leur com-

portement ne peut néanmoins pas être indiqué.

Nous illustrons enfin la composition incrémentale des programmes TAO avec l'exemple suivant:

```
Incremental = par [  a: A ↖ {b} ↖ {c}
                  || b: B
                  || c: C ] ;
```

Cette présentation multimédia synchronise trois objets `A`, `B` et `C` et peut avoir une signification différente selon ce que représentent ces objets:

- une présentation pré-orchestrée:
  A peut être une présentation de fond (image de fond et musique d'ambiance), mise en boucle jusqu'à ce que la vidéo B ou le texte C se terminent;
- une présentation interactive:
  A peut être un programme présentant une séquence de questions à l'utilisateur qui peut cliquer sur les boutons B et C pour indiquer son choix;
- un programme de contrôle du déroulement d'une présentation:
  A peut être une présentation multimédia et B et C deux moyens de la stopper, comme un bouton et un raccourci clavier par exemple.

Il apparaît ici que la richesse des présentations multimédias TAO dépend à la fois du scénario exprimé et des objets utilisés, la combinaison de ces deux éléments déterminant l'intention de l'auteur.

4.6 Conclusion

Nous avons dressé le cadre formel du langage TAO à travers sa syntaxe et sa sémantique. Les opérateurs temporels constituent le principal apport du langage et permettent de spécifier la synchronisation des objets primitifs à un haut niveau d'abstraction. Ils reposent sur un modèle temporel indiquant les causalités du scénario. Ce modèle permet de calculer la sémantique du scénario grâce à un processus de normalisation basé sur des règles de réécriture conditionnelles.

Nous avons pu voir sur quelques exemples que les programmes TAO sont intuitifs. Cette lisibilité pour l'utilisateur est indépendante de la la sémantique du scénario, dont l'expression peut se révéler complexe et peut être calculée automatiquement par un programme. Ce découplage entre syntaxe et sémantique est un point fort de TAO qui permet de regrouper dans le modèle temporel les difficultés de la spécification.

Chapitre 5

Compilation et exécution des programmes TAO

Après la définition du langage TAO, nous nous intéressons ici aux traitements applicables aux programmes du langage. Dans le domaine de la compilation, les propriétés des langages sont exploitées afin d'adapter des techniques de transformation de programmes à ce langage. Nous examinerons ainsi dans un premier temps certaines propriétés du langage qui proviennent de son modèle temporel. Nous présenterons dans un deuxième temps une machine d'exécution des programmes TAO, en la définissant tout d'abord formellement puis en indiquant la manière technique dont nous l'implantons. Les principes de cette machine d'exécution ont été introduit dans [121].

5.1 Propriétés du modèle temporel

5.1.1 Cohérence temporelle

Etant donné que les programmes TAO spécifient une synchronisation temporelle, leur cohérence est un aspect central. Spécifier des présentations multimédia qui soient exécutables est un objectif important. En effet, les présentations multimédias intéressantes possèdent le plus souvent un grand nombre d'objets et leur création est alors une tâche fastidieuse. Une incohérence dans le scénario peut alors être difficile à détecter et sa résolution ralentit l'auteur. La définition de la cohérence d'un programme TAO peut se faire à deux niveaux distincts:

- au niveau du modèle temporel, cette propriété étant vérifiée sur les points temporels,
- ou au niveau du modèle de synchronisation, cette propriété étant vérifiée sur les relations causales définies par chaque opérateur.

Au niveau du modèle temporel

A ce niveau, une incohérence apparaît si la condition de cohérence d'un in-

tervalle est brisée, c'est-à-dire si $\mathsf{beg}(A) \prec \mathsf{end}(A)$ ou si aucun point temporel $\mathsf{beg}(A)$ ne survient avant qu'un point temporel $\mathsf{end}(A)$ ne survienne, comme illustré en figure 5.1. Ceci ne peut arriver dans le modèle temporel du langage grâce à la séparation entre les points observés et ceux générés et à leur mise en correspondance. Un intervalle est défini par l'étendue temporelle entre les points observés $\mathsf{beg}(A)$ et $\mathsf{end}(A)$, qui ne peuvent être créés que par le système. Ce système ne les crée qu'après avoir respectivement reçu le premier point $\mathsf{start}(A)$ et le premier point $\mathsf{stop}(B)$ qui suit $\mathsf{beg}(A)$, point dénommé $\mathsf{fstop}(A)$. C'est cette deuxième correspondance qui assure à elle seule que les conditions de cohérence énoncées ci-dessus sont respectées. En effet, le point temporel $\mathsf{end}(A)$ ne peut être qu'après le point $\mathsf{beg}(A)$ si celui-ci survient ou ne pas survenir dans le cas contraire. Les points $\mathsf{start}(A)$ survenant après le premier d'entre eux et les points $\mathsf{stop}(A)$ survenant avant le premier point $\mathsf{start}(A)$ ou après le premier point stop sont donc ignorés. Ce sont des intentions de démarrer ou stopper un intervalle qui ne peuvent être prise en compte afin d'assurer la cohérence du système.

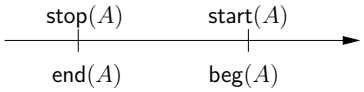


FIG. 5.1 – *Situation d'incohérence temporelle*

Notons de plus qu'un point temporel $\mathsf{start}(A)$ survenant après le point $\mathsf{end}(A)$ n'est pas interprété comme un redémarrage de l'intervalle A car ceci violerait la propriété d'unicité spatio-temporelle. Rappelons que l'intervalle A est l'occurrence d'un objet qui peut apparaître plusieurs fois dans un programme TAO, correspondant chaque fois à un intervalle différent. Ce n'est pas le point de vue pris par SMIL par exemple puisque sa sémantique calcule les redémarrages de présentations des intervalles. Une machine de présentation des programmes TAO doit respecter ce principe afin d'assurer un ordonnancement correct des événements liés aux intervalles primitifs. Cette machine doit mémoriser les début et les fins des intervalles de façon à pouvoir ignorer les points temporels survenants après l'occurrence de l'intervalle.

Au niveau du modèle de synchronisation

Le modèle de synchronisation reposant sur le modèle temporel, la cohérence est donc toujours assurée. Néanmoins, la définition des opérateurs assure cette cohérence indépendamment du modèle temporel afin que le scénario se suffise à lui-même et permette d'expliciter le comportement d'un scénario au niveau de la synchronisation. La cohérence repose sur une relation causale dont la causalité est à l'origine des règles de coupure dans le modèle temporel. Par contre ici, la condition est déterminante. La relation causale que nous nommerons RCC est la suivante:

$$RCC: \quad \mathsf{end}(A) \mapsto \mathsf{stop}(B) \quad \textit{if} \quad c \qquad \text{avec } c = (\mathsf{end}(A) \prec \mathsf{end}(B))$$

Les opérateurs mettant en jeu RCC sont ceux qui sont à l'origine d'une coupure, donc master, min, alt, $\nwarrow$ et $\searrow$. Chaque opérateur pouvant introduire une relation causale RCC, un scénario peut en contenir plusieurs. Néanmoins, grâce à la condition c, une seule d'entre elle s'appliquera et génèrera un point temporel $\mathsf{stop}(A)$.

La relation RCC peut s'exécuter de deux façons:

- Soit le point $\mathsf{end}(A)$ survient avant qu'un point temporel $\mathsf{stop}(B)$ ne soit survenu, donc avant le premier d'entre eux $\mathsf{fstop}(B)$ qui est mis en correspondance avec le point $\mathsf{end}(B)$, ce qui rend la condition c vraie conformément à la définition de la relation $\prec$; la causalité génère alors le point $\mathsf{stop}(A)$ qui se révèle être le point $\mathsf{fstop}(A)$; l'ordonnancement des points est ici $\mathsf{end}(A) \prec \mathsf{fstop}(B) \prec \mathsf{end}(B)$;
- Soit $\mathsf{end}(A)$ survient après le premier point temporel $\mathsf{stop}(B)$; la condition est alors fausse, la causalité ne s'applique pas et aucun point $\mathsf{stop}(B)$ n'est généré; l'ordonnancement des points est $\mathsf{fstop}(B) \prec \mathsf{end}(B) \prec \mathsf{end}(A)$.

La condition ajoutée après la causalité permet d'éliminer logiquement les relations causales qui génèreraient un point $\mathsf{stop}(A)$ mais ne seraient pas suivies d'une mise en correspondance avec le point temporel $\mathsf{end}(A)$. La raison de la cohérence est différente de celle du modèle temporel, où les points $\mathsf{stop}(A)$ évités ici surviennent mais ne sont pas mis en correspondance avec le point $\mathsf{end}(A)$.

Sémantiques à cohérence forte et faible

La sémantique des opérateurs a été donnée avec ce que nous appellerons une cohérence forte. Il est possible de définir une sémantique à cohérence faible en ôtant de la relation RCC la condition c et en s'appuyant sur la cohérence assurée au niveau du modèle temporel. La sémantique à cohérence faible est plus efficace, car elle élimine le test c, mais repose sur un mécanisme plus fondamental au niveau du modèle temporel. Cette écriture diminue par contre la précision de la sémantique des opérateurs et rend impossible l'utilisation de relations causales issues de la normalisation des termes, puisque les conditions seraient incomplètes et ne permettraient pas d'expliciter précisément le comportement de chaque relation causale. Ajoutons que réciproquement, le passage d'une sémantique à cohérence faible à une sémantique à cohérence forte consiste en l'ajout de la condition c aux relations causales de même causalité que RCC.

Situations particulières

Les situations suivantes n'amènent pas à des incohérences temporelles mais ont une interprétation particulière qui n'est pas immédiate.

La première situation provient des relations causales suivantes:

$$\mathsf{beg}(A) \mapsto \mathsf{start}(B) \ \& \ \mathsf{beg}(B) \mapsto \mathsf{start}(A)$$

Cette spécification semble boucler indéfiniment entre les deux relations causales. Mais au niveau du modèle temporel, les quatre points rentrant en jeu sont clairement ordonnés:

- si $\mathsf{beg}(A)$ survient avant $\mathsf{beg}(B)$, l'ordonnancement est $\mathsf{beg}(A) \prec \mathsf{start}(B) \prec \mathsf{beg}(B) \prec \mathsf{start}(A)$ et le point $\mathsf{start}(A)$ n'est pas mis en correspondance avec un point $\mathsf{beg}(A)$ puisque celui-ci est déjà survenu;
- dans le cas contraire les intervalles A et B sont inversés.

Une seule des deux causalités opposées survient donc.

La seconde situation est celle de la simultanéité et se divise en deux cas:

- au niveau de la spécification, deux relations causales ont une même cause, ce qui correspond à:
  $p \mapsto q_1$ & $p \mapsto q_2$, équivalent à $p \mapsto q_1$,q_2
- au niveau de l'exécution, deux relations causales dont les causes peuvent se produirent en même temps ont la même conséquence, c'est-à-dire:
  $p_1 \mapsto q$ & $p_2 \mapsto q$

Rien ne permet dans le modèle temporel d'indiquer laquelle des deux relations causales doit s'appliquer lorsque les causes surviennent, car elles sont logiquement simultanées. Les deux situations qui peuvent survenir seraient différentes uniquement s'il était possible d'exprimer dans la synchronisation temporelle une exclusion mutuelle. La causalité qui est responsable de l'exclusion mutuelle est: "$\mathsf{beg}(A) \mapsto \mathsf{stop}(B)$". Or, aucun opérateur TAO n'use de cette causalité, donc le problème ne se pose pas. Cette notion de simultanéité n'est pas réellement traduisibles en des termes informatiques non-ambigus, car les ordinateurs et leurs systèmes sont aujourd'hui des systèmes intrinsèquement séquentiels et le parallélisme n'est que simulé, existant parfois à un très bas niveau. Par contre tout système d'exécution est responsable de ce choix, qui peut éventuellement être arbitraire.

5.1.2 Équivalence des programmes

Une propriété importante des langages de programmation est celle d'équivalence des programmes car elle permet de cerner les redondances au sein du langage. Dans le cas de TAO, cette équivalence revient à déterminer les scénarios dont les termes sont égaux, ce qui recoupe deux définition de la notion d'égalité.

Premièrement, l'*égalité observationnelle* est dénotée par le symbole $\simeq$ et est définie par:
$A \simeq B$ si et seulement si il existe une présentation de A et une de B telle que les occurrences des points beg et end de tous leurs intervalles primitifs coïncident, quelles que soient les relations causales mises en jeu. Par exemple, si A et B sont des intervalles primitifs de même durée, alors:

$$\mathsf{par}[a : (A \fatsemi C) \parallel b : B] \simeq \mathsf{par}[a : A \parallel b\,(B \fatsemi C)]$$

Dans cette exemple, les ensembles de relations causales des deux termes sont justement différents. Pour le premier terme la fin de A lance C, alors que dans la seconde c'est la fin de B qui lance C. Néanmoins, les présentations sont les mêmes car A et B sont présentés en parallèle jusqu'à ce qu'ils s'arrêtent en même temps et C est alors présenté. Le plus souvent, deux termes seront égaux observationnellement pour certaines durées des intervalles primitifs utilisés, notamment pour les scénarios faisant intervenir des objets interactifs dépendant d'une action de l'utilisateur.

La seconde égalité est l'*égalité sémantique* qui est dénotée $=$ et qui est définie par la formule suivante:

$$A = B \;\Leftrightarrow\; Cl(Rel(A)) \approx Cl(Rel(B))$$

Cl est la clôture d'un ensemble de relations causales par les propriétés de transitivité et de triangularisation:

transitivité:	$(p \mapsto q \;\; if \;\; c_1)$ & $(q' \mapsto r \;\; if \;\; c_2) \;\Rightarrow\; p \mapsto r \;\; if \;\; c_1 \wedge c_2$
triangularisation:	$(p \mapsto q \;\; if \;\; c_1)$ & $(p \mapsto r \;\; if \;\; c_2) \;\Rightarrow\; q' \mapsto r \;\; if \;\; (c_1 \Rightarrow c_2)$
	avec $(q,q') \in \{(\mathsf{start}(X),\mathsf{beg}(X)),(\mathsf{stop}(X),\mathsf{end}(X))\}$

$\approx$ est l'égalité d'ensembles pour la commutativité, l'associativité, l'idempotence de & et l'égalité sémantique des conditions:

commutativité:	r_1 & $r_2 \;\Leftrightarrow\; r_2$ & r_1
associativité:	$(r_1$ & $r_2)$ & $r_3 \;\Leftrightarrow\; r_1$ & $(r_2$ & $r_3)$
idempotence:	r & $r \;\Leftrightarrow\; r$
égalité sémantique des conditions:	$((p \mapsto q \;\; if \;\; c_1) = (p \mapsto q \;\; if \;\; c_2)) \;\Leftrightarrow\; (c_1 \;\Leftrightarrow\; c_2)$

Un cas d'égalité intéressant qui met en jeu les propriétés de transitivité et d'équivalence sémantique des conditions est une propriété remarquable de l'opérateur "$\fatsemi$":

$$A \fatsemi B = \mathsf{par}[\; a : A \parallel b : \mathsf{delay}[*] \nwarrow_{\{a\}} \fatsemi B \;]$$

L'ensemble de relations causales du second membre est normalisé en:

1.$\mathsf{par}[a,b].\{\mathsf{beg} \mapsto \mathsf{start}(A),\mathsf{start}(\mathsf{delay})$
$\quad$ & $\mathsf{end}(\mathsf{delay}) \mapsto \mathsf{start}(B)$
$\quad$ & $\mathsf{end}(A) \mapsto \mathsf{stop}(\mathsf{delay}) \;\; if \;\; \mathsf{end}(A) \in \mathsf{while\ delay}$
$\quad$ & $\mathsf{end}(A) \mapsto \mathsf{stop} \;\; if \;\; \mathsf{end}(A) \succ \mathsf{end}(B)$
$\quad$ & $\mathsf{end}(B) \mapsto \mathsf{stop} \;\; if \;\; \mathsf{end}(B) \succ \mathsf{end}(A)$

où delay est l'intervalle primitif résultant de l'utilisation de l'opérateur $\mathsf{delay}[*]$. Le test de la troisième relation causale est éliminé grâce au fait que A et delay débutent au même moment et que l'intervalle delay n'a pas de fin (reportée à l'infinie). Alors la transitivité appliquée aux deuxième et troisième relations causales donne: $\mathsf{end}(A) \mapsto \mathsf{start}(B)$, qui élimine à son tour la quatrième relation causale et valide la condition de la cinquième. Finalement l'ensemble final de relations causales correspond bien à la sémantique de l'opérateur "$\fatsemi$".

L'égalité sémantique permet de déduire des équivalences de termes présentées en table 5.1. Ces égalités explicitent le fait que le jeu d'opérateurs de TAO n'est pas minimal, étant donné que les opérateurs master, min et max sont exprimés en fonction d'autres d'opérateurs. Ces opérateurs sont fournis afin de faciliter l'écriture des scénarios. Les équivalences les plus simples permettent d'éliminer des situations triviales dans un scénario. Les autres équivalences peuvent être utilisées par une machine d'exécution particulière afin d'intervertir deux opérateurs dans le cas où cette machine d'exécution peut tirer un avantage (gain de place ou de temps) de cet échange.

$$
\begin{array}{rcl}
A_1 \fatsemi (A_2 \fatsemi A_3) & = & (A_1 \fatsemi A_2) \fatsemi A_3 \\
A \fatsemi \mathsf{delay}[0] & = & \mathsf{delay}[0] \fatsemi A \;=\; A \\
\mathsf{loop}[0](A) & = & \mathsf{delay}[0] \\
\mathsf{loop}[n](A) & = & A \fatsemi \mathsf{loop}[n-1](A) \;=\; \mathsf{loop}[n-1](A) \fatsemi A \qquad (n>0) \\
\mathsf{loop}[n](\mathsf{loop}[m](A)) & = & \mathsf{loop}[n*m](A) \\
\mathsf{loop}[n](\mathsf{loop}(A)) & = & \mathsf{loop}(\mathsf{loop}[n](A)) \;=\; \mathsf{loop}(A) \\
\mathsf{master}[A \parallel B] & = & \mathsf{par}[a : A \searrow_{\{b\}} \parallel b : B] \\
\mathsf{min}[A \parallel B] & = & \mathsf{par}[a : A \searrow_{\{b\}} \parallel b : B \searrow_{\{a\}}] \\
\mathsf{max}[A \parallel B] & = & \mathsf{par}[a : A \parallel b : B] \\
(A_1 \fatsemi A_2) \searrow_{\{b\}} & = & A_1 \fatsemi (A_2 \searrow_{\{b\}})
\end{array}
$$

TAB. 5.1 – *Équivalences sémantiques de termes*

La différence entre les égalités observationnelle et sémantique est dans le fait que deux termes égaux sémantiquement le seront quels que soient les présentations des intervalles primitifs utilisés, alors que l'égalité observationnelle repose sur les durées de ceux-ci. L'égalité sémantique de deux termes implique leur égalité observationnelle grâce à la propriété de "simultanéïté perçue" des relations causales: un observateur percevra la causalité comme une égalité, autrement dit:

$$p \mapsto q \;\Rightarrow\; p = q$$

Ceci implique que l'egalité sémantique et la connaissance des durées des intervalles primitifs se traduit en égalité observationnelle, car les conditions des relations causales disparaissent et les propriétés des relations Cl et $\approx$ se ramènent à celles de l'égalité $=$.

Notons que le modèle temporel ne définit que la relation d'ordre $\prec$ et qu'il n'existe pas de relation d'égalité entre les points temporels. L'égalité classique $=$ est en fait découplée en plusieurs cas de figure déterminés tout d'abord par la direction de la relation causale ($\mapsto$) et par ses conditions.

5.1.3 Traduction vers d'autres systèmes

Nous présentons ici deux principes de traductions de programmes TAO vers d'autres langages. Ceci peut servir de base à un compilateur, bien que nous n'ayons ici dressé que le principe de la traduction.

5.1.3.1 Traduction vers des ensembles de relations sur des points temporels

L'approche du langage TAO consiste à concevoir les scénarios en se basant sur la synchronisation des intervalles. En calculant la sémantique d'un scénario, nous avons effacé la structure d'intervalles afin de ne plus voir que des points temporels. Les modèles temporels classiques [154] reposent aussi sur ces entités mais avec des relations différentes des relations causales. Comme nous l'avons vu à la section précédente, il est possible de ramener les relations causales et la connaissance des durées des intervalles à la relation d'égalité $=$. En enlevant la connaissance des durées et en élargissant les relations sur les points aux relations $<$, $=$ et $>$, nous allons donner une idée de la traduction des programmes TAO en relations sur les points temporels en suivant un exemple.

Les limites des intervalles sont donc considérées comme des points temporels, dénotés $\mathsf{b}S$ et $\mathsf{e}S$ respectivement pour le début et la fin de l'intervalle S. Les premiers points temporels $\mathsf{start}(S)$ et $\mathsf{stop}(S)$ (c'est-à-dire $\mathsf{fstop}(S)$) à survenir sont respectivement équivalents à $\mathsf{b}S$ et $\mathsf{e}S$, ceux survenant ultérieurement n'ayant pas d'équivalents. b et e correspondent aux début et fin de l'intervalle composé dont on considère l'expression. La condition de cohérence de l'intervalle S s'exprime comme: $\mathsf{b}S \leq \mathsf{e}S$, alors que l'on indique que cet intervalle S n'a pas de durée nulle grâce à la proposition: $\mathsf{b}S < \mathsf{e}S$.

Lorsqu'un scénario fait intervenir des opérateurs dynamiques, plusieurs cas d'exécution, ou *chemin*, sont définis selon les positionnements relatifs de certains points temporels entre eux. Un scénario sera donc traduit par une disjonction de cas d'exécution, chacun d'eux étant un chemin correspondant à une conjonction de relations sur les points temporels et correspondant à un ordonnancement fixe des points temporels.

Considérons le scénario $S = \mathsf{master}[A \parallel B ; C]$. L'ensemble de relations causales $Rel(S)$ issu du processus de normalisation est, en modifiant les notations des points temporels dans les conditions:

$$
\begin{array}{lll}
\mathsf{beg} \mapsto \mathsf{start}(A),\mathsf{start}(B) & & \\
\& \ \mathsf{end}(A) \mapsto \mathsf{stop}(B) & \textit{if} & \mathsf{b}B < \mathsf{e}A \wedge \mathsf{e}A < \mathsf{e}B \\
\& \ \mathsf{end}(A) \mapsto \mathsf{stop}(C) & \textit{if} & \mathsf{b}C < \mathsf{e}A \wedge \mathsf{e}A < \mathsf{e}C \\
\& \ \mathsf{end}(B) \mapsto \mathsf{start}(C) & \textit{if} & \mathsf{e}A < \mathsf{b}B \vee \mathsf{e}A > \mathsf{b}C \\
\& \ \mathsf{end}(A) \mapsto \mathsf{stop} & &
\end{array}
$$

Les conditions définissent trois chemins selon le positionnement relatif de $\mathsf{e}A$ par rapport aux deux autres points temporels $\mathsf{e}B$ (qui est égal à $\mathsf{b}C$ si $\mathsf{e}A$ n'est

pas survenu avant, comme c'est le cas dans la troisième relation causale) et $\mathsf{e}C$ (si C est effectivement présenté). Les conditions définissant les régions sont:

$$(\mathsf{b}B < \mathsf{e}A \wedge \mathsf{e}A < \mathsf{e}B),\ (\mathsf{b}C < \mathsf{e}A \wedge \mathsf{e}A < \mathsf{e}C) \text{ et } (\mathsf{e}A > \mathsf{e}C)$$

La première relation causale étant inconditionnelle, la relation $\mathsf{b}A = \mathsf{b}B$ tient dans tous les cas d'exécution. Mais ceci n'est pas encore pris en compte dans les relations causales suivantes, qui n'ont pu être déduites que du processus de normalisation et ne contiennent pas de simplification.

L'ensemble de relations causales est ensuite fermé grâce à l'opérateur Cl précédemment décrit. L'ensemble clos peut alors être simplifié. On regroupe les relations causales en les factorisant sur leur causalité. Un point temporel $\mathsf{start}(S)$ indique que l'intervalle S sera présenté dans le cas d'exécution considéré, ce qui permet donc d'ajouter la contrainte $\mathsf{b}S < \mathsf{e}S$ dans ce cas d'exécution. Dans le cas où le point $\mathsf{b}S$ n'apparaît pas, S n'est pas présenté.

La règle de simultanéïté perçue est ensuite appliquée en enlevant de la condition toute relation portant sur les points temporels en jeu dans la causalité. Il faut ensuite éliminer les chemins dont les relations causales sont en contradiction, qui correspondent à des cas d'exécution impossibles. Dans l'exemple précédent, le cas d'exécution suivant est incohérent:

$$(\mathsf{beg} \mapsto \mathsf{start}(A), \mathsf{start}(B)\ \&\ \mathsf{end}(B) \mapsto \mathsf{start}(C)\ \&\ \mathsf{end}(A) \mapsto \mathsf{stop})\ \ \textit{if}\ \ \mathsf{e}A < \mathsf{b}B$$

En effet, après application de la propriété de simultanéïté, on obtient:

$$\mathsf{b} = \mathsf{b}A \wedge \mathsf{b} = \mathsf{b}B \wedge \mathsf{e}B = \mathsf{b}C \wedge \mathsf{e}A = \mathsf{e}$$

Ces relations combinées avec les conditions de cohérence des intervalles $\mathsf{b}A \leq \mathsf{e}A \wedge \mathsf{b}B \leq \mathsf{e}B \wedge \mathsf{b}C \leq \mathsf{e}C$ produisent la relation $\mathsf{b}B \leq \mathsf{e}A$. Une contradiction est exhibée et donc ce cas de figure est éliminé.

La vérification de la cohérence des ensembles de points temporels est automatisable et des algorithmes en $O(n^3)$ [151], plus tard ramené à $O(n^2)$ [148], existent.

La traduction du scénario S est finalement:

$$\begin{aligned}&(\mathsf{b}A \leq \mathsf{e}A \wedge \mathsf{b}B \leq \mathsf{e}B \wedge \mathsf{e}A = \mathsf{e}B)\\ \vee\ &(\mathsf{b}A \leq \mathsf{e}A \wedge \mathsf{b}B \leq \mathsf{e}B \wedge \mathsf{e}B = \mathsf{b}C \wedge \mathsf{b}C \leq \mathsf{e}C \wedge \mathsf{b}C < \mathsf{e}A \wedge \mathsf{e}A = \mathsf{e}C)\\ \vee\ &(\mathsf{b}A \leq \mathsf{e}A \wedge \mathsf{b}B \leq \mathsf{e}B \wedge \mathsf{e}B = \mathsf{b}C \wedge \mathsf{b}C \leq \mathsf{e}C \wedge \mathsf{e}C < \mathsf{e}A)\end{aligned}$$

Nous n'avons pas fait figurer les relations sur le début ($\mathsf{b} = \mathsf{b}A \wedge \mathsf{b} = \mathsf{b}B$) et sur la fin ($\mathsf{e}A = \mathsf{e}$) du scénario mais elles sont présentes sous la même forme dans tous les cas d'exécution. Dans le premier cas d'exécution, l'intervalle C n'est pas présenté.

Il est de plus possible d'optimiser cet ensemble de relations sur les points temporels grâce aux durées de certains intervalles en éliminant certains chemins qui apparaissent contradictoires au vu de ces durées (phase d'optimisation proche de celle de propagation des constantes dans le domaine de la compilation).

Remarquons finalement que l'utilisation des propriétés de transitivité et de triangularisation amènent à introduire de nouvelles relations causales qui n'étaient pas originellement dans le scénario, tout en restant cohérent avec le modèle de synchronisation de TAO. La causalité originellement désirée par le créateur du scénario est perdue. Elle se retrouve plongée au milieu de toutes les relations qu'elle implique.

5.1.4 Traduction vers des relation de Wahl-Rothermel

Les relations de Wahl-Rothermel constituant un langage important dans le domaine, nous avons considéré la traduction de notre langage vers ces relations. Étant elles-même construites à partir des disjonctions de relations d'Allen, une traduction vers ces dernières est donc immédiate par transitivité. Les relations de Wahl-Rothermel utilisant les durées des intervalles, notamment afin de déterminer la validité des paramètres de certaines relations, nous utiliserons la fonction *dur*(A) pour indiquer la durée de l'intervalle A. Les deux langages étant de haut niveau d'abstraction, des équivalences entre opérateurs suffisent à les traduire et sont présentées en table 5.2. Aucune traduction n'est possible pour l'opérateur $\mathsf{loop}(A)$, qui nécessiterait d'écrire une infinité de fois la relation *A before(0) A*. Il n'y en a pas non plus pour le terme $A \nwarrow_{\{v\}}$ dans le cas où v est un événement de programme. Ceci est dû au fait que les relations de Wahl-Rothermel ne permettent pas de gérer les cas d'indéterminismes où l'on ne possède aucune connaissance sur les intervalles.

$A \fatsemi B$	*A before(0) B*
$\mathsf{loop}[n](A)$	(n-1) fois A *before(0) A*
$\mathsf{master}[\ A \parallel B\ ]$	if *dur*(A) < *dur*(B) then *A cobegin(0) B* else *A while(0,0) B*
$\mathsf{min}[\ A \parallel B\ ]$	*A while(0,0) B*
$\mathsf{max}[\ A \parallel B\ ]$	*A cobegin(0) B*
$\mathsf{alt}[\ A_1 => B_1 \parallel A_2 => B_2]$	if *dur*(A_1) < *dur*(A_2) then A_1 *while(0,0)* A_2 & A_1 *before(0)* B_1 else A_1 *while(0,0)* A_2 & A_2 *before(0)* B_2
$\mathsf{par}[\ a_1 : A_1 \parallel a_i : A_2]$	A_1 *cobegin(0)* A_2
$A \searrow_{\{b\}}$	*A coend(0) ref*(b)
$A \nwarrow_{\{b.end\}}$	*ref*(b) *coend(0) A*

TAB. 5.2 – *Traduction des opérateur TAO en relations de Wahl-Rothermel*

La séparation en plusieurs cas selon les durées des intervalles est identique à celle en plusieurs chemins dans la traduction vers des ensembles de relations sur des points temporels. Notons enfin que tous les programmes TAO construisant une expression, sa traduction en liste d'opérateurs de Wahl-Rothermel oblige à

utiliser des noms sur tous les intervalles de l'expression. Par exemple, le scénario $S = \mathsf{master}[A \parallel B \fatsemi C]$ de la section précédente nécessite de nommer l'intervalle $S_2 = B \fatsemi C$.

5.1.5 Calcul incrémental de propriétés

Les programmes TAO étant basés sur l'usage d'opérateurs, leur structure arborescente offre la possibilité de calculer des propriétés de façon incrémentale, c'est-à-dire en procédant à des calculs séparés sur les différentes parties du scénario. Nous examinons deux propriétés en particulier: la mise à l'échelle et la terminaison des programmes TAO.

5.1.5.1 Mise à l'échelle de termes

Une mise à l'échelle d'un scénario consiste en son raccourcissement ou son allongement selon un certain facteur. Cela revient à respectivement accélérer ou ralentir la vitesse d'écoulement du temps d'un facteur égal. Les relations de synchronisation étant invariante par cette transformation, son application se fait de façon identique sur chacun des éléments du scénario.

Cette transformation est une illustration immédiate du calcul incrémental de propriétés sur les programmes TAO et est effectuée par la fonction scale, de paramètres le facteur de transformation et définie comme:

- pour un intervalle primitif A, $\mathsf{scale}[s](A)$ est un intervalle primitif qui présente les données référencées par A à la vitesse $1/s$ si cela est possible; pour des données non-temporelles, cela revient à multiplier leurs durées par s;
  pour des données temporelles, il faut pouvoir changer la vitesse de présentation de l'objet dont l'intervalle est une occurrence; si cela n'est pas possible, la durée de l'intervalle est quand même multipliée par s, mais les données seront soit tronquées, soit complétées (par un délai ou bien la dernière donnée présentée);
- pour un intervalle composé $op(A_i)_{i\in\{1\ldots n\}}$, pour op un opérateur:

$$\mathsf{scale}[s](op(A_i)_{i\in\{1\ldots n\}}) = op(\mathsf{scale}[s](A_i))_{i\in\{1\ldots n\}}$$

Comme nous l'avons vu en section 3.3.1.3, certains média ne permettent de régler leur vitesse de présentation. Une sémantique de remplacement doit être envisagée. Elle repose le plus souvent sur une approche naïve qui consiste à jouer le scénario de manière transformée, en laissant s'exécuter normalement la présentation des portions qui ne peuvent être transformées et en appliquant une synchronisation transformée.

5.1.5.2 Terminaison

Lorsque des intervalles de durée indéfinie sont utilisés, par exemple pour un bouton ou une donnée temporelle diffusée en flux (streaming), un scénario peut se retrouver bloqué par la présentation de ces intervalles, qui peut très bien ne jamais se terminer. Par exemple, le scénario $A \fatsemi B$, où A est une animation 3D générée par un serveur sur l'Internet, ne permettra pas de voir l'intervalle B si A n'a pas de durée finie. Il est donc utile d'indiquer au créateur du scénario les endroits où de tels blocages peuvent apparaître.

Il s'agit en fait du traditionnel problème de la terminaison d'un programme. Dans le cas d'une présentation multimédia, le problème est modifié par les interactions avec l'environnement de la présentation. La terminaison d'un programme *TAO* se décompose en deux cas:

- la terminaison forte
  cette propriété assure que la présentation d'un terme s'arrête dans tous les cas de figure; l'arrêt a lieu si le terme a une durée finie, ce qui arrive si les données qu'il référence ont toutes une durée intrinsèque finie ou bien qu'il est argument d'un opérateur qui provoque sa coupure par un autre terme qui a une durée finie;
- la terminaison faible, dite interactive
  cette propriété est vérifiée si un terme est *contrôlé*, c'est-à-dire qu'il a une durée inconnue ou infinie mais peut être arrêté par une interaction avec l'utilisateur ou avec un autre système ou qu'il est coupé par un autre terme possédant la propriété de terminaison faible.

La terminaison forte implique d'une certaine manière la terminaison interactive en considérant que les interactions responsable de l'arrêt son internes au système d'exécution de TAO.

Un objet primitif qui référence des données comme une vidéo ou un son se termine de manière forte car il a une durée finie. Par contre un objet représentant un bouton ne se termine que de manière faible, car un spectateur peut toujours l'arrêter mais il est possible qu'il ne le fasse jamais. Les termes $\mathsf{delay}[*]$ et $\mathsf{loop}(A)$ ne se terminent pas d'eux-mêmes et ce sont les opérateurs dans lesquels ces termes sont inclus qui peuvent éventuellement propager la propriété de terminaison des autres arguments, comme par exemple pour le scénario $\mathsf{master}[B \parallel \mathsf{loop}(A)]$ où B se termine interactivement.

Afin de déterminer la terminaison, forte ou interactive, d'un terme ψ, nous calculons la fonction $\mathsf{iterm}(\psi)$ qui retourne un ensemble de sous-termes de ψ qui peuvent nécessiter une terminaison.

Le seul cas du calcul de iterm qui ne renvoie pas une valeur précise est celui de l'alternative:

$$\mathsf{iterm}(\mathsf{alt}[\ \|_{\ i\in\{1\ldots n\}}\ A_i => B_i]) = \bigcup_{i\in\{1\ldots n\}} (\mathsf{iterm}(A_i) \cup \mathsf{iterm}(B_i))$$

Il est impossible de savoir quel intervalle argument B_i sera présenté et donc la propriété de terminaison de tous les intervalles B_i doit être examinée, même si seul l'un d'entre eux sera effectivement présenté.

La valeur de $\mathsf{iterm}(A)$ pour un intervalle primitif A dépend la propriété de terminaison examinée. Dans tous les cas, elle a pour valeur $\emptyset$ si A a une durée finie. Si A représente une interaction, $\mathsf{iterm}(A)$ a pour valeur $\emptyset$ pour la propriété de terminaison interactive, mais $\{A\}$ pour la terminaison forte. Pour les intervalles composés, la valeur est calculée de façon incrémentale et est donnée en 5.3.

$$
\begin{array}{l}
\mathsf{iterm}(A \fatsemi B) = \mathsf{iterm}(A) \cup \mathsf{iterm}(B) \\
\mathsf{iterm}(\mathsf{loop}(A)) = \{\mathsf{loop}(A)\} \\
\mathsf{iterm}(\mathsf{loop}[n](A)) = \mathsf{iterm}(A) \\
\mathsf{iterm}(\mathsf{delay}[d]) = \emptyset \\
\mathsf{iterm}(\mathsf{delay}[*]) = \{\mathsf{delay}[*]\} \\
\mathsf{iterm}(\mathsf{master}[A \parallel B]) = \mathsf{iterm}(A) \\
\mathsf{iterm}(\mathsf{min}[A \parallel B]) = \begin{cases} \emptyset & \textit{if } \mathsf{iterm}(A) = \emptyset \ \vee \ \mathsf{iterm}(B) = \emptyset \\ \mathsf{iterm}(A) \cup \mathsf{iterm}(B) & \textit{otherwise} \end{cases} \\
\mathsf{iterm}(\mathsf{max}[A \parallel B]) = \mathsf{iterm}(A) \cup \mathsf{iterm}(B) \\
\mathsf{iterm}(\mathsf{alt}[\parallel_{i \in \{1 \ldots n\}} A_i => B_i]) = \bigcup_{i \in \{1 \ldots n\}} (\mathsf{iterm}(A_i) \cup \mathsf{iterm}(B_i)) \\
\mathsf{iterm}(\mathsf{par}[\parallel_{i \in \{1 \ldots n\}} a_i : A_i]) = \bigcup_{i \in \{1 \ldots n\}} \mathsf{iterm}(A_i) \\
\mathsf{iterm}(A \searrow_{\{b\}}) = \mathsf{iterm}(A) \\
\mathsf{iterm}(A \nwarrow_{\{b\}}) = \begin{cases} \emptyset & \textit{if } \mathsf{iterm}(A) = \emptyset \ \vee \ \mathsf{iterm}(\mathit{ref}(b)) = \emptyset \\ \mathsf{iterm}(A) & \textit{otherwise} \end{cases}
\end{array}
$$

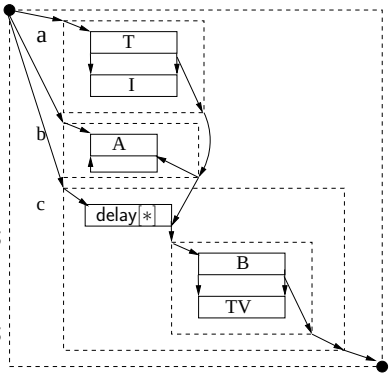
TAB. 5.3 – *Calcul de la fonction* iterm

Les sous-termes de ψ qui ne sont pas dans $\mathsf{iterm}(\psi)$ terminent, de façon forte ou interactive selon le cas considéré. Les sous-termes qui sont dans l'ensemble $\mathsf{iterm}(\psi)$ doivent par contre être contrôlés afin de garantir que le terme ψ se terminera bien. Une manière de les contrôler est d'introduire dans le scénario un opérateur additionnel parmi master, min, max ou les opérateurs de lien immédiatement au dessus de ces termes. L'autre argument de cet opérateur ajouté doit être un objet contrôlé qui permettra d'imposer une fin sur le terme incontrôlé (typiquement un objet $\mathsf{delay}[d]$ avec une durée d finie ou bien un bouton).

A titre d'exemple, considérons le scénario E où trois fenêtres sont présentées, correspondant aux branches a,b et c. Une image I est présentée dans a alors qu'un texte T est écrit. Un clip audio A est répété dans b. A la fin de la

présentation de a, b et le délai infini au début de c sont stoppés et un programme TV en ligne TV est présenté avec un bouton B de contrôle dans c. La table 5.4 présente ce scénario et le calcul de la valeur de $\mathsf{iterm}(E)$.

```
E = par[  a: Fenetre1
       || b: Fenetre2
       || c: Fenetre3] ;
Interval Fenetre1 = master[T || I] ;
Fenetre1.position = window(0,0,50,50) ;
Interval Fenetre2 = loop(A) ↖ {a} ;
Fenetre2.position = window(50,0,100,100) ;
Interval Fenetre3 = delay[*] ↖ {b} ∘
                     master[B || TV] ;
Fenetre3.position = window(50,0,200,200) ;
```



$\mathsf{iterm}(\texttt{E}) = \mathsf{iterm}(\texttt{Fenetre1}) \cup \mathsf{iterm}(\texttt{Fenetre2}) \cup \mathsf{iterm}(\texttt{Fenetre3})$
$\mathsf{iterm}(\texttt{Fenetre1}) = \mathsf{iterm}(\texttt{master[T || I]}) = \mathsf{iterm}(\texttt{T}) = \emptyset$,
$\mathsf{iterm}(\texttt{Fenetre2}) = \mathsf{iterm}(\texttt{loop(A)} \nwarrow_{\{a\}}) = \emptyset$ à cause de `Fenetre1`
$\mathsf{iterm}(\texttt{Fenetre3}) = \mathsf{iterm}(\texttt{delay[*]} \nwarrow_{\{b\}}) \cup \mathsf{iterm}(\texttt{master[B || TV]}) = \emptyset \cup \mathsf{iterm}(\texttt{B})$
à cause de `Fenetre2` et finalement $\mathsf{iterm}(\texttt{Fenetre3}) = \emptyset$.

TAB. 5.4 – *Exemple de calcul de terminaison d'un scénario TAO*

Notons finalement que la notion de contrôlabilité qui est envisagée ici est différente de celle des systèmes de formatage [93]. Dans ce cas, elle désigne la capacité du système de formatage à pouvoir librement réajuster les comportements lors de l'exécution afin de prendre en compte les événements indéterminés.

5.2 Execution des programmes TAO

Les programme étant des documents textuels compréhensibles par des êtres humains, ils ne peuvent être compris des machines, dont la tâche est de traiter des informations plus basiques. Un programme doit donc être compilé vers un format exécutable, soit par un interpréteur, soit par le système d'exploitation.

Nous spécifions tout d'abord la machine de présentation sur laquelle vont être joués les programmes en respectant la sémantique du langage. Ensuite est présenté le processus de compilation des programmes TAO vers les instructions que la machine interprète. Nous terminons par l'implémentation que nous avons faite de cette machine de présentation.

5.2.1 Structure de la machine de présentation

La machine de présentation accepte en entrée un programme TAO et, s'il est valide, le compile en instructions qu'elle joue en présentant les données de la présentation multimédia sur les différents périphériques de sortie de l'ordinateur. Elle est décomposée en trois machines plus spécifiques:

- la machine à objets
  elle contient la définition des objets, leurs attributs et leur méthodes étant ceux déclarés dans leurs classes; elle met à disposition les valeurs des attributs, dont la référence aux données des objets primitifs, et permet d'exécuter leurs méthodes; elle a à sa charge de gérer les noms des occurrences d'un objet donné, noms qui sont déterminés à la suite de la compilation du scénario; elle peut reposer sur les services offerts par une base de donnée multimédia [64];
- la machine de rendu
  elle a la charge du rendu de chaque objet primitif et est commandée par la machine d'exécution qui lui envoie les événements de démarrage $\mathsf{strt}(A)$ et d'arrêt $\mathsf{stp}(A)$ pour tout objet primitif A; inversement, lorsque la présentation des données d'un objet primitif se termine, un événement $\mathsf{stp}(A)$ est envoyé à la machine d'exécution; elle peut reposer sur les services offerts par un moteur de rendu, comme on en trouve dans le domaine des jeux vidéo [95];
- la machine d'exécution
  elle est au cœur des mécanismes de synchronisation et est décrite plus dans le détail ci-dessous.

Comme illustré en figure 5.2, la machine d'exécution est elle-même découpée en trois composants gérant chacun des éléments que nous nommerons contextes et qui seront déplacés d'un composant à un autre au cours de la présentation du scénario:

- le composant CX (pour ConteXt) entrepose les contextes définis
  c'est dans ce composant que sont stockés les définitions des contextes prêts à être joués à un instant donné; les contextes sont envoyés au composant STK à sa demande;
- le composant STK (pour STacK) joue les contextes
  c'est dans ce composant que s'exécute le noyau fonctionnel de la machine d'exécution; il retire les contextes du composant CX, les joue en exécutant leurs instructions afin d'assurer la synchronisation des divers objets primitifs, note les moments d'occurrence des événements et envoie les contextes, une fois terminés, vers le composant PE;
- le composant PE (pour Past Events) archive les contextes terminés
  ce composant ne sert qu'à stocker les contextes joués, ces contextes contenant les instants d'occurrences des événements survenus; cette trace de l'exécution du programme peut servir à:
  - interroger le scénario, par exemple afin de savoir si un intervalle a été

présenté avant un autre par exemple;
- examiner le comportement de l'utilisateur à travers ses interactions;
- enregistrer la présentation comme une vidéo, ce qui peut permettre de la rejouer dans le sens inverse ou à une vitesse différente;
- ou encore rejouer la présentation jusqu'à un certain point où l'utilisateur peut faire de nouveaux choix.

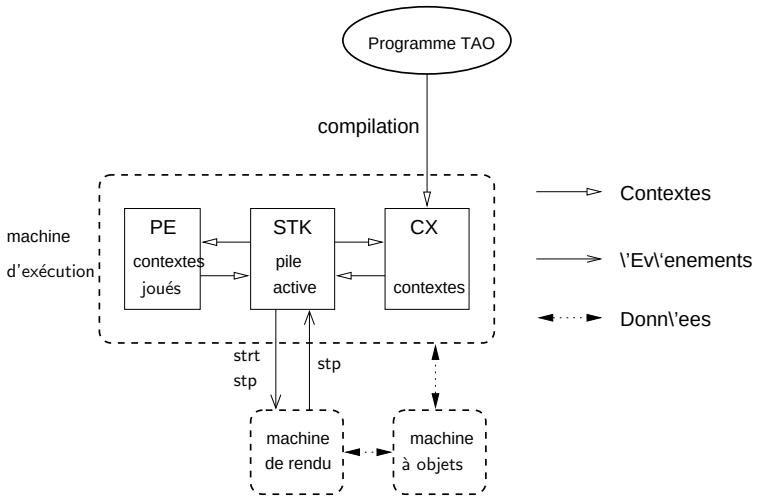


FIG. 5.2 – *Schéma de la machine de présentation*

5.2.2 Instructions et contextes

Un contexte est une suite d'instructions qui sont exécutées en séquence sauf lorsqu'une instruction de saut est rencontrée. De la même manière qu'un intervalle composé pour la sémantique des scénarios, un contexte est l'entité qui encaspule les données de synchronisation des objets primitifs qui proviennent des opérateurs utilisés dans le programme TAO à la manière des objets composés au niveau du langage. La différence provient des éléments sous-jacents à ces entités, les instructions remplaçant ici les relations causales.

Une instruction est une déclaration impérative des actions à générer en fonction des événements observés sur les objets primitifs et les contextes. Elle remplace dans la machine d'exécution une ou plusieurs relation(s) causale(s), qui spécifiai(en)t la synchronisation logique des intervalles au niveau de la sémantique du scénario. Les instructions permettent d'expliciter les chemins d'exécution possibles au sein des actions sur les objets, chemin au sens où nous

l'avons vu en section 5.1.3.1. Chaque instruction possède un numéro qui est unique dans le contexte où elle est définie. Les instructions sont de deux types:

- une instruction gardée: *Garde* $\Longrightarrow$ *Séquence d'actions*
  une telle instruction génère les *actions* dans l'ordre dans lequel elles sont déclarées (séparées par des symboles ',') lorsque la *Garde* survient;
- l'instruction $\mathsf{parEx}(I_1; \ldots; I_n)$,
  où $(I_i)_{i\in\{1\ldots n\}}$ sont des instructions gardées, exécute l'instruction gardée I_j dont la garde survient en premier. Si la machine d'exécution est capable de traiter simultanément deux événements, nous choisissons que, dans le cas où deux gardes surviennent en même temps dans le cadre d'une instruction parEx, les actions de l'instruction gardée I_j d'indice le plus petit sont exécutées. Ce choix est celui que nous avons indiqué dans la troisième situation particulière à la fin de la section 5.1.1.

Les gardes et actions sont un sous-ensemble de celui des événements, ces derniers étant:

- $\mathsf{strt}(A)$ le début de l'objet ou du contexte A;
- ou $\mathsf{stp}(A)$ la fin de l'objet ou du contexte A;
- ou strt le début du contexte où l'événement apparaît;
- ou stp la fin du contexte où l'événement apparaît.

Chaque événement contient d'autre part deux attributs:

- un champ date qui mémorise la date d'occurrence de l'événement, initialisé à 0;
- un champ sequenced qui permet d'indiquer avec quel contexte le contexte démarré par cet événement est mis en séquence, initialisé à 0.

L'ensemble des gardes est celui des événements privé de stp. La première instruction, de numero 1, est la seul gardée par l'événement strt et détermine les premières actions à exécuter lorsque le contexte est joué. Une action est un événement, sauf l'événement strt, et peut aussi être:

- une affectation de variable explicite: *nom* := *expression*
  Les variables sont de type entier ou un ensemble d'entiers .
  Des variables explicites peuvent être définies dans chaque contexte. Elles sont définies lors de la première utilisation de leur nom dans un contexte donné et n'ont de portée que dans ce contexte. Elles sont ensuite initialisées à leur valeur par défaut, 0 pour les entiers et $\emptyset$ pour les ensembles.
  Les expressions entières sont constituées de noms de variables entières, de littéraux entiers et des opérateurs arithmétiques usuels ($+$, $-$, $*$ et division entière/). Les ensembles sont construits comme suit: pour x_1 et x_2 deux expressions entières telles que $x_1 < x_2$, $S = \{x_1\}$ est le singleton contenant l'élément x_1 et $S = x_1 \ldots x_2$ correspond à l'ensemble des valeurs entières entre x_1 et x_2. Les opérateurs ensemblistes sont l'union $\cup$, la soustraction $-$ et $getElem(S)$ qui renvoie un élément au hasard dans l'ensemble S.

– un saut conditionnel:
  if c then jump(n)
  c est une comparaison de deux expressions entières grâce aux relations d'ordre $<$, $=$ et $>$; la condition c est évaluée et si elle vaut *true* alors le numéro de l'instruction courante du contexte devient n, sinon rien ne se passe;
– un saut inconditionnel jump(n) équivalente à if *true* then jump(n).

Un contexte contient aussi des variables implicites qui sont définies pour tout contexte quel qu'il soit. Certaines de ces variables sont utilisées pour la gestion interne du contexte et ne sont donc pas consultables ou manipulables dans les instructions des contextes. Pour un contexte C, les variables implicites sont:

– le numéro $C.IC$ de l'instruction courante, initialisé à 1;
– la liste $C.ED$ des entités démarrées par le contexte et non encore terminées, initialisée à $\emptyset$;
– la liste $C.ET$ des entités qui n'ont pas encore été démarrées ou bien dont la présentation est déjà terminée, initialisée à l'ensemble des noms des entités présentes dans le contexte, c'est-à-dire qui sont arguments d'un événement contenu dans une instruction du contexte.

De plus, les contextes correspondant à un opérateur loop[n], alt et par possèdent une variable implicite $C.n$ qui est le nombre de boucles pour loop et le nombre d'arguments pour alt et par. Cette variable implicite est consultable dans les instructions du contexte mais n'est pas modifiable. Au fur et à mesure que le contexte est joué, les noms des entités présentes dans le contexte sont déplacés de $C.ET$ à $C.ED$ lorsqu'elles sont démarrées puis de $C.ED$ à $C.ET$ lorsqu'elles sont terminées. La figure 5.3 illustre la définition générale d'un contexte.

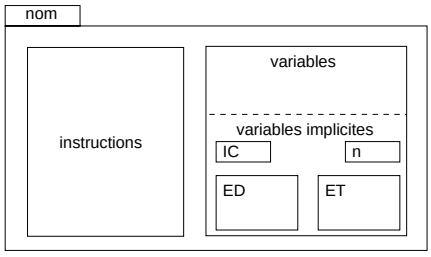


FIG. 5.3 – *Définition générale d'un contexte*

Dans ces instructions, les intervalles apparaissent sous le nom qui leur est donné après compilation. Dans certains cas, ces noms sont génériques et dépendent de variables entières. Deux notations sont alors utilisées:

– $A\{i\}$ désigne la i^{eme} répétition de A avec une variable i définie dans les instructions et qui sera substituée lors de l'exécution;

– $A(i)_{i \in set}$ et $A(i)$ sont des raccourcis syntaxiques utilisés uniquement lors de la traduction des programmes *TAO* en instructions. Ils permettent de simplifier l'écriture de cette traduction ou bien d'exprimer la valeur d'une variable explicite, comme un paramètre de l'opérateur correspondant au contexte. L'ensemble *set* des valeurs de i doit être défini à la première apparition de l'indice i, puis n'est plus nécessaire ensuite. Ces indices seront alors substitués à la compilation en dupliquant l'instruction où ils apparaissent pour chaque valeur de l'ensemble *set* ou bien en remplaçant (i) par sa valeur pour une variable explicite. Par exemple, les substitutions suivantes auront lieu:
  $\mathsf{parEx}(\mathsf{stp}(A(i)_{i \in \{1,2\}}) \Longrightarrow \mathsf{stp})$ $\quad$ $\mathsf{parEx}(\mathsf{stp}(A1) \Longrightarrow \mathsf{stp}; \mathsf{stp}(A2) \Longrightarrow \mathsf{stp})$
  $\mathsf{stp}(A(n)) \Longrightarrow \mathsf{stp}$ $\quad$ $\mathsf{strt}(A15) \Longrightarrow \mathsf{stp}$

Le contexte initial est un contexte particulier qui est différencié des autres car il correspond à l'opérateur de plus haut niveau dans le scénario et doit être joué en premier.

5.2.3 Compilation

La compilation a pour objectif de transformer l'expression du scénario à base d'opérateurs en un ensemble de contextes et leur donner des étiquettes. Elle s'effectue en plusieurs étapes:

– Lexicographie et analyse syntaxique
  Le compilateur vérifie que les divers éléments lexicographique du scénario sont valides et qu'ils sont assemblés conformément à la grammaire du langage. Le scénario est représenté à la fin de ces deux phases par un arbre abstrait syntaxique.
– Génération de code
  C'est lors de cette phase qu'est généré le code cible correspondant à l'arbre abstrait issu des phases précédentes.

Le langage cible est dans notre cas celui des contextes d'instructions. La traduction lors de la génération de code repose sur la sémantique de chaque opérateur en transformant les ensembles de relations causales en contextes d'instructions, définis à la section précédente.

Le principe de base de la traduction est que la cause d'une relation causale est transformée en garde et les conséquences en action. Plus précisément, les relations causales qui ont des conditions qui se complémentent seront traduites par plusieurs instructions de sauts exprimant cette complémentarité. Ainsi, pour l'opérateur min:

$\mathsf{end}(A) \mapsto \mathsf{stop}(B),\mathsf{stop}$ *if* $\mathsf{end}(A)$ while B
& $\mathsf{end}(B) \mapsto \mathsf{stop}(A),\mathsf{stop}$ *if* $\mathsf{end}(B)$ while A

sera traduit en:

$\mathsf{parEx}(\mathsf{stp}(A) \Longrightarrow \mathsf{stp}(B),\mathsf{stp};$
$\mathsf{stp}(B) \Longrightarrow \mathsf{stp}(A),\mathsf{stp})$

La traduction est donnée en table 5.5

$A \fatsemi B$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A)$ $2 : \mathsf{stp}(A) \Longrightarrow \mathsf{strt}(B)$ $3 : \mathsf{stp}(B) \Longrightarrow \mathsf{stp}$
$\mathsf{loop}(A)$	$1 : \mathsf{strt} \Longrightarrow i{:=}1,\mathsf{strt}(A\{i\})$ $2 : \mathsf{stp}(A\{i\}) \Longrightarrow \mathsf{strt}(A\{i{+}1\}),i{:=}i{+}1,\mathsf{jump}(2)$
$\mathsf{loop}[n](A)$	$1 : i := 1,\mathsf{strt} \Longrightarrow \mathsf{strt}(A\{i\})$ $2 : \mathsf{stp}(A\{i\}) \Longrightarrow \mathsf{strt}(A\{i{+}1\}),i{:=}i{+}1,$ $\mathsf{if}\ \ i{<}n{-}1\ \ \mathsf{then}\ \ \mathsf{jump}(2)$ $3 : \mathsf{stp}(A(n)) \Longrightarrow \mathsf{stp}$
$\mathsf{master}[\ A \parallel B\ ]$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A),\mathsf{strt}(B)$ $2 : \mathsf{stp}(A) \Longrightarrow \mathsf{stp}(B),\mathsf{stp}$
$\mathsf{min}[\ A \parallel B\ ]$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A),\mathsf{strt}(B)$ $2 : \mathsf{parEx}(\mathsf{stp}(A) \Longrightarrow \mathsf{stp}(B),\mathsf{stp};$ $\mathsf{stp}(B) \Longrightarrow \mathsf{stp}(A),\mathsf{stp})$
$\mathsf{max}[\ A \parallel B\ ]$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A),\mathsf{strt}(B)$ $2 : \mathsf{parEx}(\mathsf{stp}(A) \Longrightarrow \mathsf{jump}(3);$ $\mathsf{stp}(B) \Longrightarrow \mathsf{jump}(4))$ $3 : \mathsf{stp}(B) \Longrightarrow \mathsf{stp}$ $4 : \mathsf{stp}(A) \Longrightarrow \mathsf{stp}$
$\mathsf{alt}[\ \parallel_{i\in\{1\ldots n\}} A_i => B_i\ ]$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A(i)_{i\in\{1\ldots n\}})$ $2 : \mathsf{parEx}(\mathsf{stp}(A(i)) \Longrightarrow \mathsf{stp}(A(j)_{j\in\{1\ldots n\}\wedge j\neq i}),$ $\mathsf{strt}(B(i)),ind{:=}(i))$ $3 : \mathsf{stp}(B\{ind\}) \Longrightarrow \mathsf{stp}$
$\mathsf{par}[\ \parallel_{i\in\{1\ldots n\}} a_i : A_i\ ]$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A(i)_{i\in\{1\ldots n\}}),cpt{:=}(n),$ $S{:=}1\ldots(n)$ $2 : \mathsf{parEx}(\mathsf{stp}(A(i)) \Longrightarrow cpt{:=}cpt{-}1,S{:=}S{-}\{(i)\},$ $\mathsf{if}\ \ cpt{\neq}1\ \ \mathsf{then}\ \ \mathsf{jump}(2))$ $3 : \mathsf{stp}(A\{getElem(S)\}) \Longrightarrow \mathsf{stp}$
$A \searrow_{\{b\}}$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A)$ $2 : \mathsf{stp}(A) \Longrightarrow \mathsf{stp}(b),\mathsf{stp}$
$A \nwarrow_{\{v\}}$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A)$ $2 : \mathsf{parEx}(v' \Longrightarrow \mathsf{stp}(A),\mathsf{stp};$ $\mathsf{stp}(A) \Longrightarrow \mathsf{stp})$ $(v,v') \in \{(b.end,\mathsf{stp}(b)),(P.e,P.e)\}$

TAB. 5.5 – *Traduction des opérateurs TAO en instructions*

La traduction de alt utilise la variable *ind* pour récupérer l'indice du premier argument de l'opérateur à se terminer. La traduction par utilise elle une variable *cpt* pour détecter que $n-1$ objets se sont terminés, puis $getElem(S)$ permet d'obtenir le numéro du dernier objet en cours de présentation.

5.2.3.1 Réduction du nombre de contextes

La compilation génère par défaut un contexte par opérateur, assignant des noms aux contextes créés et définissant le contexte initial. Le nombre de contextes créés peut parfois être réduit, ce qui est une optimisation puisque le mécanisme de chargement puis de démarrage d'un contexte peut être économisé. Les contextes ainsi optimisés sont obtenus d'une manière similaire à la sémantique des scénarios, en appliquant des règles de réécriture conditionnelles sur les contextes non-optimisés.

Comme pour la normalisation, cette réduction du nombre de contextes va aplatir la structure d'arbre du scénario en remontant certaines informations de certains contextes vers d'autres. Les contextes sont isomorphes aux nœuds de l'arbre et peuvent être organisés hiérarchiquement de la même manière. La différence avec la normalisation est qu'ici tous les nœuds ne seront pas supprimés en règle générale.

Les nœuds qui correspondent à des contextes arrêtés par un événement stp seront conservés et aucune règle de réécriture conditionnelle ne leur sera appliquée. Comme vu pour la normalisation, cet arrêt intervient lorsqu'un contexte correspond à un opérateur qui est argument d'un des opérateurs master, min, alt et de liens. Les points temporels stop à l'origine de la coupure seront traduits par une action stp qui aura pour argument un contexte correspondant à l'argument de l'opérateur et mettra donc en jeu le mécanisme de coupure au niveau de la machine d'exécution (cf. plus loin en section 5.2.4). Il n'y aura donc pas ici d'équivalent aux règles de coupure. Pour l'opérateur par, les branches qui doivent être coupées par un ou plusieurs opérateurs de lien seront transformées en contexte, qui seront arrêtés ou bien arrêteront une autre branche grâce à une action stp. Les autres branches ne seront pas traduites en contextes et leurs instructions seront remontées.

Pour tous les autres contextes, les règles de réécriture conditionnelles (ou plus précisément leurs équivalents pour les instructions) s'appliqueront et remonteront leurs instructions dans les contextes supérieurs, pour s'achever par la destruction du contexte ne contenant plus de règles. Nous ne décrivons ici que le principe des règles de réécriture conditionnelles, leurs notations et expressions étant données en annexe A.2.

Les règles de réécriture conditionnelles (I_lift1) et (I_lift2) décrivent la remontée d'une instruction, respectivement gardée et parEx et qui ne sont pas celles de démarrage et d'arrêt du contexte C', d'un contexte C' vers un contexte supérieur C. La condition indep(C,C') indique que C est un contexte supérieur à C' et qu'il ne contient pas d'événement stp(C').

Les instructions de démarrage et d'arrêt du contexte C' sont traitées par les autres règles de réécriture conditionnelles, chacune correspondant à un cas complémentaire des règles de réécriture conditionnelles (I_lift1) et (I_lift2): $p \neq \mathsf{strt}$ pour (I_strt), $\mathsf{stp} \notin Q_1$ pour (I_stp1) et $\nexists p,Q_1,Q_2 \mid (p \Longrightarrow Q_1,\mathsf{stp},Q_2) \in I_1$ pour (I_stp2).

A ce stade, il manque deux règles de réécriture conditionnelles, pour compléménter les règles de réécriture conditionnelles (I_stp1) et (I_stp2) dans le cas où l'événement $\mathsf{stp}(C')$ apparaît dans le contexte C au sein d'une instruction parEx. Ces règles de réécriture conditionnelles ne seront pas développées car elle sont trop complexes et ajoutent un grand nombre d'instructions (de l'ordre de la factorielle du nombre d'objets mis en séquence dans le contexte C'). En effet, lorsque l'on substitue l'événement $\mathsf{stp}(C')$ dans ces règles de réécriture conditionnelles, on ne peut pas simplement le remplacer par celui qui génère l'événement stp dans le contexte C'. Regardons sur l'exemple du scénario $S = \mathsf{max}[A \parallel \alpha]$ avec $\alpha = \mathsf{max}[B \parallel C]$:

S	α
$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(A),\mathsf{strt}(\alpha)$	$1 : \mathsf{strt} \Longrightarrow \mathsf{strt}(B),\mathsf{strt}(C)$
$2 : \mathsf{parEx}(\mathsf{stp}(A) \Longrightarrow \mathsf{jump}(3);$	$2 : \mathsf{parEx}(\mathsf{stp}(B) \Longrightarrow \mathsf{jump}(3);$
$\mathsf{stp}(\alpha) \Longrightarrow \mathsf{jump}(4))$	$\mathsf{stp}(C) \Longrightarrow \mathsf{jump}(4))$
$3 : \mathsf{stp}(\alpha) \Longrightarrow \mathsf{stp}$	$3 : \mathsf{stp}(C) \Longrightarrow \mathsf{stp}$
$4 : \mathsf{stp}(A) \Longrightarrow \mathsf{stp}$	$4 : \mathsf{stp}(B) \Longrightarrow \mathsf{stp}$

Pour remplacer l'événement $\mathsf{stp}(\alpha)$ de l'instruction numéro 2 du contexte S, on ne peut pas simplement considérer les instructions 3 et 4 du contexte α. Il faut en effet prendre en compte l'instruction 2 du contexte α afin de mettre en séquence les instructions parEx des deux contextes. Dans le cas général, il faudrait créer de nouvelles instructions parEx qui prennent en compte toutes les séquences de présentation d'objets qui ont lieu dans le contexte inférieur C'.

La réduction du nombre de contextes ne peut se faire que lorsqu'un cheminement à travers les instructions ne rencontre pas une instruction parEx comportant l'arrêt d'un contexte, ce qui arrive pour tous les opérateurs parallèles sauf master. La réduction du nombre de contextes n'a donc lieu que pour les scénarios possédant des opérateurs séquentiels ou master et les opérateurs n'opérant que sur des objets primitifs. Bien que présentées de manière logique, les règles de réécriture conditionnelles ne sont pas appliquées dans n'importe quel ordre puisque le format des contextes d'instructions est séquentiel. C'est l'ordre des instructions du contexte substitué C' qui doit être suivi:

- tout d'abord l'instruction de démarrage du contexte C' permet de substituer l'instruction du contexte C qui a pour action $\mathsf{strt}(C')$ par la règle de réécriture conditionnelle (I_strt); cette substitution n'aura lieu qu'une fois car aucun des opérateurs prédéfinis ne démarre plusieurs fois un de ses arguments; le numéro de l'instruction i substituée dans C est mémorisé dans la variable *indice* afin de pouvoir insérer les autres instructions à la suite de celle-ci;

– ensuite les règles de réécriture conditionnelles (I_lift1) et (I_lift2) sont éventuellement appliquées afin de remonter les instructions suivantes du contexte C' vers le contexte C; le numéro d'instruction i utilisé dans les règles de réécriture conditionnelles a pour valeur celle de la variable *indice* qui est incrémenté à chacune de ses utilisations;
– et enfin les règles de réécriture conditionnelles (I_stop1) et (I_stop2) substituent les instructions ayant pour points observés $\mathsf{stp}(C')$, qui se trouvent après les substitutions précédentes au numéro d'instruction *indice*, toujours incrémenté à chaque utilisation car il peut y avoir plusieurs applications de ces règles de réécriture conditionnelles.

Lorsque toutes les instructions du contexte inférieur C' ont été traitées par une de ces règles de réécriture conditionnelles, ce contexte C' est détruit. Les instructions de C qui suivent la dernière instruction substituée doivent être décalées en fonction des renumérotations d'instructions effectuées.

A titre d'exemple, le scénario $B \fatsemi C \fatsemi D$ correspond après réduction du nombre de contexte à un unique contexte initial $CX1$ dont les instructions sont:

$$\boxed{\begin{array}{l} 1\text{: } \mathsf{strt} \Longrightarrow \mathsf{strt}(D) \\ 2\text{: } \mathsf{stp}(B) \Longrightarrow \mathsf{strt}(C) \\ 3\text{: } \mathsf{stp}(C) \Longrightarrow \mathsf{start}(D) \\ 4\text{: } \mathsf{stp}(D) \Longrightarrow \mathsf{stp} \end{array}}$$

Par contre, le scénario $\mathsf{master}[\ A \parallel B \fatsemi C \fatsemi D\ ]$ est traduit en deux contextes, le précédent $CX1$ et le contexte initial $CX2$ suivant:

$$\boxed{\begin{array}{l} 1\text{: } \mathsf{strt} \Longrightarrow \mathsf{strt}(A),\mathsf{strt}(CX1) \\ 2\text{: } \mathsf{stp}(A) \Longrightarrow \mathsf{stp}(CX1),\mathsf{stp} \end{array}}$$

5.2.4 Fonctionnement de la machine d'exécution

Une fois dressée l'architecture générale de la machine de présentation, le format intermédiaire des contextes d'instructions et le mécanisme de compilation des programmes, nous précisons ici le fonctionnement général de l'ensemble.

5.2.4.1 Liste des entités dans les divers composants

Les trois composants CX, STK et PE tiennent à jour une liste de noms des entités, contextes et objets, qu'ils gèrent, respectivement nommées $listeCX$, $listeSTK$ et $listePE$ et définies à un moment donné par:

– $listeCX$ contient le nom des contextes fournis à l'issue de la compilation et pas encore transférés vers le composant STK ainsi que l'union des variables implicites ET de ces contextes, ce qui correspond à la liste des entités pas encore présentées;
– $listeSTK$ contient des contextes dont le composant a demandé le transfert au composant CX ainsi que l'union des variables implicites ED de ces contextes, ce qui correspond à la liste des entités en cours de présentations;

– $listePE$ contient la liste des contextes que le composant STK lui a envoyé ainsi que l'union des variables implicites ET de ces contextes, ce qui correspond aux entités déjà présentées.

Le noms d'un contexte change de liste au cours de la présentation selon que le contexte est prêt à être joué, en cours de présentation ou bien terminé. Il ne peut être que dans une et une seule de ces listes à un moment donné. Le nom d'un objet primitif peut être dans une de ces listes ou bien aucune. Lorsque le nom d'un objet n'est dans aucune liste, le contexte où est présent l'objet primitif est en cours de présentation mais l'objet n'a pas encore été démarré par une instruction de ce contexte.

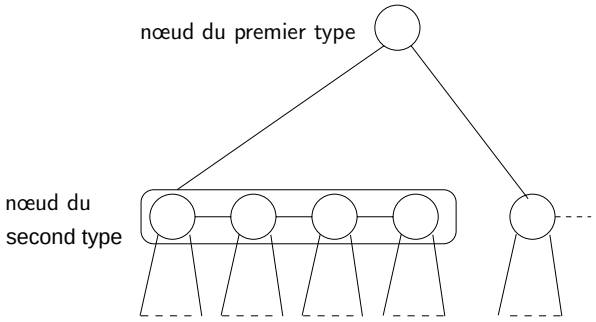
Notons finalement que les événements jouent à la fois le rôle de cause et de conséquence pour les instructions de la machine d'exécution, mêlant ainsi les notions de points observés et générés du modèle temporels de TAO. C'est à la charge du composant STK de vérifier qu'un objet n'est pas redémarré en consultant $listePE$ ou bien qu'il n'est pas présenté en consultant $listeCX$. Ce comportement de la machine d'exécution correspond à la règle du modèle temporel assurant la cohérence temporelle indiqué en section 5.1.1.

5.2.4.2 Structure de données dans les composants STK et PE

Dans le composant STK, les contextes sont organisés à la manière d'une pile cactus, ou pile saguaro, ou encore *tree structure stack* [72, 15]. Originellement utilisée en architecture des machines informatiques, cette pile admet une méthode supplémentaire d'empilement, nommée branch, qui empile une nouvelle pile sur la pile courrante où la méthode a été appellée. Les commandes push et pop sont appellées de façon transparente respectivement à travers les événements strt et stp et permet de dépiler efficacement un contexte lorsqu'un événement stp survient pour ce contexte.

Les éléments poussés sur la pile sont des contextes. Les contextes d'opérateurs séquentiels correspondent à un niveau de la pile où sont joués les sous-contextes en les empilant puis en les dépilant successivement. Les contextes d'opérateurs parallèles utilisent eux plusieurs sous-piles pour gérer le parallélisme des présentations d'arguments.

Pour le composant PE, la structure est différente de la pile cactus car elle n'est pas utilisée pour jouer les contextes mais pour stocker leur exécution, ce qui dans le cas des opérateurs séquentiels nécessite de mémoriser les séquences de contextes joués. Cette structure que nous nommerons TPE (pour Tree of Past Events) ressemble à un arbre alternant deux types de nœuds comme illustré en figure 5.2.4.2. La racine de cette arbre est un nœud du premier type qui contient un contexte et représente les exécutions en parallèle de ses fils. Chaque fils est un nœud d'un second type constitué d'une liste ordonnée de nœuds du premier type qui représente leur séquence. Un nœud du premier type peut n'avoir aucun



fils si le contexte n'a démarré aucun contexte, auquel cas il n'a démarré que des objets primitifs. Ceux du second type peuvent avoir une liste réduite à un seul élément si aucun contexte n'a été joué en séquence avec un autre en son sein. Il y a deux manières d'ajouter un nœud dans cette structure: soit en fils d'un nœud du premier type dans le cas où le contexte est joué en parallèle avec d'autres contextes déjà joués dans le contexte qui correspond à ce nœud, soit en bout d'une liste d'un nœud du deuxième type pour un contexte joué en séquence avec le dernier élément de la liste.

Pour que la structure TPE soit bien construite de la racine aux feuilles, il est nécessaire que lorsqu'un contexte est transféré dans le composant STK, son image soit créée dans la structure TPE (mais son contenu n'y est pas copié). De plus, lorsqu'un contexte est ajouté à la structure TPE, son champ sequenced permet de savoir s'il a été présenté en séquence et doit donc être ajouté à la suite du nœud de second type courant ou bien en parallèle, auquel cas il sera ajouté en nœud du premier type au nœud de second type courant.

Lorsque des actions sont créées par des instructions dans le composant STK, leurs dates d'occurrence sont notées dans leur champ date. Ceci permet d'exécuter la présentation multimédia telle qu'elle s'est déroulée et de procéder à des calculs sur les différentes durées et délais des objets de cette présentation.

5.2.4.3 Fonctionnement général

Maintenant que toutes les briques de la machine de présentation ont été présentées, nous pouvons définir son fonctionnement général en liant les différents aspects précédemment abordés.

En entrée de la machine de présentation, un scénario est tout d'abord compilé en un ensemble de contextes, dont un est marqué comme initial. Ces contextes sont chargés dans le composant CX. Les variables de ces contextes sont alors initialisées, puis la variable $listeCX$ du composant CX est initialisée avec la liste

des noms des contextes et l'union de leurs variables ET, alors que $listeSTK$ et $listePE$ sont initialisés a $\emptyset$.

Lorsque tous les services de la machine de présentation ont été initialisés, le composant STK charge le contexte initial, le composant PE créé l'image de ce contexte dans la structure TPE et le composant STK génère l'événement strt de ce contexte. Cet événement est placé dans la file d'attente des événements, nommée FA, que le composant STK utilise pour traiter les événements.

A partir de ce moment, le composant STK exécute une boucle de traitement des événements de la file d'attente FA. Il vérifie tout d'abord si l'événement doit être ignoré dans le cas où c'est un événement stp ayant lieu avant que l'intervalle correspondant ait démarré ou strt après que cet intervalle ait été présenté. Si l'événement est valide et relatif à un objet primitif, l'instruction est relayée à la machine de rendu qui démarre ou arrête la présentation de l'objet. Si l'événement est valide et relatif à un objet composé, un empilement ou un dépilement sur la pile saguaro a lieu et le contexte mis en jeu est transféré, respectivement depuis CX et vers PE. Tout au long des déplacements de contextes et des démarrages et arrêts d'objets, les listes des entités des composants ainsi que les variables internes des contextes sont mises à jour. Enfin, le composant STK interprète l'instruction dont l'événement est la garde.

La présentation est terminée lorsque le contexte initial est dépilé dans le composant STK et transféré dans le composant PE. Tous les contextes sont alors dans le composant PE, les dates d'occurrence des événements qui sont survenus étant notées.

5.2.5 Implémentation au-dessus du prototype MADCOW

Le code à base de contextes d'instructions joue en fait le rôle de code intermédiaire interprété par la machine de présentation dont nous avons posé les bases. Plutôt que d'implémenter la machine de présentation depuis le départ, nous avons voulu réutiliser et étendre le code Java de la plate-forme MADCOW (cf. section 3.8.5). Trois approches étaient alors possibles:

- traduire les programmes TAO dans le langage à base de balises défini par MADCOW; cette approche nécessite d'examiner les règles sémantiques de TAO en regard du langage et de la plate-forme MADCOW, afin d'assurer que les programmes TAO sont bien présentés en accord avec leur sémantique;
- traduire les programmes TAO dans des programmes Java utilisant les classes définies au sein de la plate-forme MADCOW, approche identique à celle du compilateur MADCOW; la traduction doit comme ci-dessus être examinée au vu du code Java généré, en reposant sur la description conceptuelle qui est donnée des classes MADCOW [128]; cette mise en correspondance des sémantiques est difficile à exhiber car, bien qu'elles

partagent toutes deux la notion d'événements, la notion de relation causale ou d'instruction d'exécution ne suit pas la même logique;
- afin de pouvoir traduire cette logique, une troisième approche nous a semblé meilleure; il s'agit de reprendre les classes Java de la plate-forme MADCOW afin d'implémenter tous les éléments de la machine de présentation que nous avons définie auparavant; à ce niveau de maîtrise du code, il nous est possible de traduire exactement les mécanismes voulus et donc assurer la sémantique du langage.

En utilisant l'héritage offert par le langage Java, nous réutilisons au mieux les classes MADCOW qui se décomposent en trois groupes:

- `SyncObject` et ses sous-classes
  ces classes définissent les objets multimédias manipulés par la plate-forme MADCOW; chaque classe d'objet multimédia comprend deux parties, celle héritée de la classe `SyncObject` qui regroupe tous les traitements généraux communs à tous les médias et celle hérité de l'interface `Synchronisable` qui déclare les traitements spécifiques à chaque média et format de données;
- `SyncEvent`
  cette classe définit de manière abstraite les événements qui sont manipulés par la plate-forme et définissent la synchronisation d'un document MADCOW; les sous-classes concrètes sont `IntraEvent` utilisé pour la synchronisation intra-média et `StartEvent` et `StopEvent` pour la synchronisation inter-média des divers objets médias;
- `SyncManager`
  c'est la classe qui se charge d'appliquer la politique de synchronisation en gérant pour un groupe d'objets les occurrences des événements selon le rôle maître ou esclave des objets auxquels ils s'appliquent; les différents gestionnaires de synchronisation s'exécutant pour la présentation reposent sur un unique ordonnanceur (classe `Scheduler`) qui a la charge de gérer les événements au sein d'une file d'attente, en calculant de manière dynamique leurs dates d'activation.

A noter que MADCOW définit la synchronisation fine des objets médias alors que TAO ne le permet pas. L'implémentation de la machine de présentation des programmes TAO ne fait pas usage des rôles de maîtres et esclaves puisque la politique de synchronisation a été redéfinie.

Chapitre 6

Conclusion

6.1 Bilan

Le multimédia a été le fil directeur de ce travail de thèse. Nous avons pu avoir un aperçu de sa diversité, due à une pénétration dans de nombreux autres domaines, techniques ou non, dont la richesse s'est communiquée au multimédia. Cette diversité est un avantage pour les utilisateurs qui entrevoient des systèmes tout en un, une maîtrise plus complète de leurs créations multimédias et la faculté de bénéficier des innovations les plus récentes. Elle se révèle par contre être une difficulté quand il s'agit de concevoir les systèmes multimédias car il faut intégrer des médias différents tout en donnant une vision claire de comment les utiliser au sein du système. L'implémentation de ces systèmes sur plusieurs plate-formes informatiques est un autre aspect difficile, puisqu'il nécessite de mettre en correspondance les fonctionnalités logiques avec les équipements matériels, qui peuvent être variables d'une plate-forme à l'autre.

Le multimédia se décompose en sous-domaines, tels que les périphériques matériels de rendu des médias, le traitement de chaque média, la synchronisation, la composition spatiale ou bien encore la diffusion des contenus. Il est donc nécessaire de clairement inscrire une contribution dans un domaine ou bien délimiter les nouvelles possibilités offertes par rapport à celles existantes. Nous avons choisi de contribuer au domaine de la modélisation multimédia qui propose d'utiliser des outils de modélisation afin d'identifier et de définir les divers composants du multimédia. C'est un domaine abstrait de recherche qui efface les dépendances vis-à-vis des détails et des spécificités d'une plate-forme particulière, et qui permet d'implémenter des résultats sur tous ces environnements. L'objet particulier de nos études était la présentation multimédia, représentation de haut niveau de l'organisation et de l'intégration de divers données médias.

Les travaux important en modélisation multimédia comportent des modèles de données multimédias ou d'organisation de ces données. Nous avons vu qu'un

des aspects centraux de cette organisation est la synchronisation qui spécifie les relations temporelles des objets médias. Les modèles de synchronisation ont fait l'objet de beaucoup de propositions qui ont délimité des paradigmes de synchronisation, comme les intervalles des relations d'Allen ou les opérateurs de la *Video Algebra*. Ces modèles ont aussi été introduits à travers des langages de spécification, qui offrent une description compréhensible par tout concepteur de système informatique et servent de base à l'implémentation de plusieurs programmes. Parmi les langages qui comptent dans le domaine, nous avons pu voir *HyTime*, *MHEG* ou plus récemment *SMIL*. SMIL constitue une étape importante dans le multimédia car il propose un point de vue relativement complet des présentations multimédias et a été largement adopté par la communauté informatique.

L'approche que nous avons choisie s'insère dans le paradigme d'opérateurs temporels de la *Video Algebra* afin de spécifier et décrire les présentations multimédias. Le haut niveau d'abstraction des opérateurs fournit un moyen conceptuel pour la définition des présentations multimédias, en comptant sur le système d'exécution du langage pour mettre en œuvre les détails d'implémentation de la sémantique de ces opérateurs.

Nos propositions ont été concrétisées par la définition du langage que nous avons nommé TAO (pour *Temporal Algebraic Operators*) dont les caractéristiques sont:

- TAO est un langage orienté objet dans lequel les objets référencent les données monomédias à présenter et les opérateurs temporels décrivent la synchronisation de ces objets. Les classes d'objets proposées permettent de décrire toute la palette des médias classique et considèrent de plus le média programmatique.
- Le modèle de synchronisation de TAO est basé sur les intervalles et permet de gérer des formes d'indéterminisme sur les objets médias, comme une durée inconnue ou bien une fin indiquée par une interaction utilisateur. Les dix opérateurs TAO permettent une spécification claire et puissante du scénario de la présentation multimédia.
- La sémantique des opérateurs est indiquée dans le modèle temporel de TAO qui est basé sur les relations causales entre points temporels associés aux objets médias et permet de définir précisément et de façon intuitive la sémantique de chaque opérateur temporel. Le modèle temporel offre un cadre pour l'expression formelle de la sémantique des programmes TAO et permet de calculer des propriétés sur ces présentations, comme leur terminaison par exemple. La traduction des programmes TAO vers d'autres langages peut être basée sur cette sémantique, comme nous l'avons vu dans le cas des relations sur les points temporels classiques.

Enfin, le multimédia étant une culture technique empreinte d'expérimentation, nous avons proposé l'architecture d'une machine d'exécution des présentations

multimédias TAO afin d'implémenter nos travaux théoriques. Cette machine d'exécution est divisée en trois parties logiques, les machines à objets, de rendu et d'exécution, que nous spécifions en mettant l'accent sur la modularité. Ainsi la machine d'exécution a été divisée en composants, dont les algorithmes de fonctionnement et les structures de données ont été précisées. Cette spécification tente d'expliciter tous les aspects important de l'implémentation afin que les résultats de cette phase de conception fournissent un guide efficace lors de l'implantation. Nous avons choisi de coder le prototype par dessus la machine d'exécution du projet MADCOW, utilisant ainsi les possibilités d'extension et de modularité offerte par le langage Java.

6.2 Perspectives

Nous avons vu tout au long de cette thèse que le multimédia est facteur de convergence entre des mondes qui jusqu'ici n'évoluaient pas dans la même direction, comme la télévision, les télécommunications ou l'industrie du jeu. Dans le nouveau monde numérique qui se dessine, le multimédia apparaît comme un plaque tournante autour de laquelle viennent se greffer un ensemble de problèmes conceptuels et techniques. Les problématiques techniques unifient des domaines séparés de l'informatique, tels que les systèmes, les bases de données ou les réseaux. De la même manière que toutes les problématiques théoriques n'ont pas été épuisées, des propositions originales continuent d'émerger des systèmes multimédias et un effort d'organisation se fait aujourd'hui sentir. Des initiatives comme le standard MPEG-21 [20] laissent entrevoir un projet plus global et construit du domaine, offrant des possibilités intéressantes pour l'avenir et conservant la diversité des fonctionnalités à la disposition des auteurs aujourd'hui.

Dans le domaine plus spécifique de la modélisation multimédia, les modèles existant semblent insuffisants en terme de conceptualisation et n'abordent pas tous les aspects nécessaires des présentations multimédias. Un langage majeur comme SMIL propose par exemple une synchronisation temporelle complète mais aussi complexe et dont la sémantique prête parfois à confusion. Nous pensons donc que des contributions comme TAO peuvent apporter des éléments cohérents permettant de clarifier la spécification des présentations multimédias, que ce soit à travers le langage lui-même ou bien des notions de ses modèles de synchronisation et temporel, notions qui peuvent être appliquées à d'autres langages. Une piste intéressante à explorer serait la comparaison des langages TAO et SMIL. Ceci permettrait d'exécuter les programmes TAO sur des players SMIL, de comparer leurs pouvoirs d'expression et d'expliciter les points sensibles et difficiles de la sémantique de SMIL.

Le jeu des opérateurs TAO peut encore être augmenté afin de rendre d'autres aspects des présentations multimédias. Le travail futur autour des opérateurs

peut s'orienter dans deux directions en particulier:

- la composition spatiale
  cette dimension de la composition des présentations multimédias est indispensable afin d'homogénéiser les spécifications; étant donné les similitudes existant entre l'approche par opérateurs et la géométrie fonctionnelle [73], nous pensons qu'une étude intéressante peut être menée dans ce cadre; nous n'avons malheureusement pas pu aborder une étude dans cette direction;
- les annotations
  bien que moins primordiale, cette dimension n'en est pas moins nécessaire au sein d'une présentation multimédia, afin de permettre de définir le sens abstrait des présentations multimédias, au sens où leurs créateurs l'entendent, et permettre l'organisation du contenu multimédia dans son ensemble afin de faciliter les recherches au sein du vaste contenu qui existe aujourd'hui; il est alors possible d'imaginer des méthodes de requêtes ad hoc ou inspirées de celles créées pour les bases de données multimédias.

Autre aspect important, la hiérarchie des classes d'objets peut être enrichie afin de couvrir un plus grand spectre de données et de formats spécifiques. Les classes abstraites définissent des traitements génériques et chaque média peut en posséder plusieurs déclinaisons, comme on l'a vu pour le son et l'animation en flux (*stream*). Les spécialisations de classes doivent donc se traduire par un effort au niveau de la machine d'exécution pour supporter un nombre important de formats de données monomédias, l'API Java Media offrant une piste intéressante à ce propos. De plus, la classe `Program` nécessite d'être détaillée à deux niveaux:

- d'un point de vue théorique, une classification des traitements permis aux programmes déclarés dans le langage TAO doit être dressée afin de pouvoir préserver la sémantique temporelle des programmes TAO; divers mécanismes peuvent entrer en jeu, selon que ces objets programmes sont spécifiés dans un langage dédié (qu'il faudrait alors définir), comme c'est le cas pour les scripts dans le pages HTML, ou bien qu'ils référencent des programmes externes au système d'exécution, comme cela est envisagé dans le middleware CORBA grâce aux déclarations d'interfaces (*stubs* et *skeletons*);
- au niveau pratique, soit la compilation est modifiée en conséquence du choix précédent si un langage dédié est envisagé, soit la machine d'exécution des programmes doit inclure un mécanisme d'appel aux programmes externes qui peut s'inspirer de l'extension *JNI* (*Java Native Interface*) de la plate-forme Java.

Les manipulations permettant de passer d'un média à un autre, ou transcoadage, sont déjà apparues et se répandront très sûrement dans les années à venir. Les problèmes qu'elles posent semblent commencer à émerger aujourd'hui et nécessitent des études plus poussées.

En conclusion, nous sommes conscient que des efforts restent à faire afin de faire avancer notre approche, tout comme TAO constitue une continuation aux travaux de la *Video Algebra* et de *Menkalinan*, et que sa place reste encore à trouver au sein du vaste univers multimédia. Nous semblons néanmoins à un nouveau tournant du multimédia car des propositions importantes comme SMIL trouvent des échos positifs dans de nombreux domaines, tant techniques que non-scientifiques, et d'autres propositions continuent d'émerger comme MPEG-21 et X3D. De même que la situation du monde et le visage de l'informatique changent, le multimédia continue de suivre et façonner des évolutions dont nous ne voyons qu'une petite partie. Si des spécialistes prévoient que des domaines comme la programmation collaborative ou les réseaux ambiants auront un rôle important à l'avenir, il leur est difficile de dire dans quelle direction évoluera le multimédia. Il semble déjà au cœur de bons nombres de systèmes de convergence entre informatique, cinématographie, téléphonie et industrie du jeu. Nul doute que la créativité qui est propre au multimédia saura ouvrir de nouvelles portes et qu'elle sera un moteur d'expansion et de maturation pour l'informatique du futur.

Bibliographie

[1] ISO JTC 1/SC 24. Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language – Part 1: Functional specification and UTF-8 encoding, 1997.

[2] S. Adali, M.L. Sapino, and V.S. Subrahmanian. An algebra for creating and querying multimedia presentations. *Multimedia Systems*, 8:212–230, 2000.

[3] Michel Adiba. *Multimedia Database System, design and implementation strategies*, chapter 3, STORM: An Object-oriented multimedia DBMS, pages 47–88. Kluwer Academic Publishers, 1996.

[4] Michel Adiba and José-Luis Zechinelli-Martini. Spatio-Temporal Multimedia Presentations as Database Objects. In *Proceedings of DEXA'99, 10th International Conference on Databases and Expert Systems Applications*, Florence, Italy, Septembre 1999.

[5] Alain Boucher and Denis Beauchemin and Denis Boucher and Jean-François Barbe and Daniel Chouinard and Monique Demers. CD-ROM et multimédia, Guide des sources d'information, 1998. http://www.sdm.qc.ca/txtdoc/tbcdrm00.html.

[6] Jennifer Allanson. Electrophysiologically Interactive Computer Systems. *Computer*, 35(3):60–65, Mars 2002.

[7] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, Novembre 1983.

[8] James F. Allen. Time and Time Again : The Many Ways to Represent Time. *International Journal of intelligent Systems*, 6(4):341–355, Juillet 1991. Wiley Computer Books.

[9] Jeff Ayars, Dick Bulterman, Aaron Cohen, and al. Synchronized Multimedia Integration Langage (SMIL 2.0). Technical report, W3C : World Wide Web Consortium, Août 2001. REC-smil20-20010807. http://www.w3.org/TR/smil20/.

[10] Brian Bailey, Joseph A? Konstan, Robert Cooley, and Moses Dejong. Nsync - A Toolkit for Building Interactive Multimedia Presentations. In *Proceedings of the 6th ACM International Multimedia Conference*, Bristol, England, 12-16 Septembre 1998.

[11] Be Inc. The Media OS, 1998. http://www.beatjapan.org/mirror/www.be.com/products/beos/mediaos.html.

[12] Didier Bert and Stéphane Lo Presti. Algebraic Specification of Operator-based Multimedia Scenarios. In Springer-Verlag, editor, *Recent Trends in Algebraic Development Techiques, WADT'99. Selected Papers*, LNCS 1827, pages 382–399, 2000.

[13] Thomas Bittner. Reasoning about Qualitative Spatio-temporal Relations at Multiple Levels of Granularity. In *Proceedings of the Conference ECAI'02*, pages 317–321, Lyon, France, 21-26 Juillet 2002.

[14] G. S. Blair, L. Blair, and J. B. Stefani. A Specification Architecture for Multimedia Systems in Open Distributed Processing. *Computer Networks and ISDN Systems*, 29(4), Mars 1997.

[15] Daniel G. Bobrow and Ben Wegbreit. A Model and Stack Implementation of Multiple Environments. *Communications of the ACM*, 16(10):591–603, Octobre 1973.

[16] Katalin Bognár. *Modélisation du goût*. PhD thesis, Université de Savoie, Mai 1997.

[17] Klemens Böhm and Thomas C. Rakow. Metadata for Multimedia Documents. *ACM SIGMOD record*, 23(4):21–26, Decembre 1994.

[18] Susanne Boll and Wolfgang Klas. Z_YX - A Multimedia Document Model for Reuse and Adaptation of Multimedia Content. *IEEE Transactions on Knowledge and Data Engineering*, 13(3):361–382, Mai/Juin 2001.

[19] Susanne Boll, Wolfgang Klas, and Jochen Wandel. A Cross-Media Adaptation Strategy for Multimedia Presentations. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[20] Jan Bormans and Keith Hill. Overview of the MPEG-21 Standard. ISO/IEC JTC1/SC29/WG 11 N4318, Juillet 2001.

[21] Marc Brelot. *Représentation orientées-objets de scènes visuelles pour la composition flexible*. PhD thesis, Institut National Polytechnique de Grenoble (INPG), Mai 1999.

[22] M. C. Buchanan and P. T. Zellweger. Automatic Temporal Layout Mechanisms. In *Proceedings of the First ACM International Conference on Multimedia*, pages 341–350, Anaheim, CA, 1-6 Août 1993.

[23] John F. Koegel Buford. *Multimedia Systems*. ACM Press, 1994.

[24] Dick C.A. Bulterman. SMIL 2.0. *IEEE Multimedia*, pages 74–84, Janvier - Mars 2002. Edited by Peiya Liu.

[25] Dick C.A. Bulterman, Lloyd Hardman, Jack Jansen, K. Sjoerd Mullender, and Lloyd Rutledge. GRiNS: A GRaphical INterface for Creating and Playing SMIL Documents. In *Proceedings of the Seventh Int. World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, Brisbane, Australia, 14-18 Avril 1998.

[26] Dick C.A. Bulterman, Lloyd Rutledge, Lynda Hardman, Jack Jansen, and K. Sjoerd Mullender. GRiNS: An Authoring Environment for Web Multimedia. In *Proceedings of the ED-MEDIA 99 - World Conference on Edu-*

cational Multimedia, Hypermedia and Educational Telecommunications, Seattle, WA, USA, 19-24 Juin 1999.

[27] Dick C.A. Bulterman and Robert van Liere. Multimedia Synchronization and UNIX. In *Proceedings of the 2nd International Workshop on Network and OS Support for Digital Audio/Video*, LNCS164, Heidelberg, Germany, Novembre 1991.

[28] Vanevar Bush. As We May Think. *Atlantic Monthly*, 1:102–108, juillet 1945.

[29] Sérgio Campos, Berthier Ribeiro-Neto, Autran Macedo, and Luciano Bertini. Formal Verification and Analysis of Multimedia Systems. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[30] Laurent Carcone. *Formatage spatial dans un environnement d'édition/présentation de documents multimédia*. PhD thesis, C.N.A.M., Decembre 1997.

[31] Luca Cardelli and Andrew D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 365–377, Boston, MA, USA, 19-21 Janvier 2000.

[32] L. Carmel, D. Harel, and D. Lancet. Estimating the Size of the Olfactory Repertoire. *Bulletin of Mathematical Biology*, 63(6):1063–1078, Novembre 2001.

[33] Franco Casalino, Guido Franceschini, and Mauro Quaglia. MPEG-4 Systems, Concepts and Implementation. In *Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, pages 504–517, Berlin, Germany, 1998.

[34] Dean Chang. Haptics: Gaming's New Sensation. *Computer*, 35(8):84–86, Août 2002.

[35] Edward Y. Chang. πDTV: a Client-based Interactive DTV Architecture. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[36] Zhou Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of duration. *Information Processing Letters*, 40(5):269–276, 1991.

[37] Jamil Chaoui, Ken Cyr, Jean-Pierre Giacalone, and al. OMAP$^{\text{TM}}$: Enabling Multimedia Applications in Third Generation (3G) Wireless Terminals. Technical Report SWPA001, Texas Instrument, Decembre 2000.

[38] Christian F.K. Schaller. Putting together a complete multimedia architecture for Unix, 27 Février 2002. http://www.linuxpower.com/display.php?id=215.

[39] P. J. Cohen. Decision procedures for real and p-adic fields. *Communications in Pure and Applied Mathematics*, 22:131–151, 1969.

[40] Jean-Pierre Courtiat and R. C. de Oliveira. Proving Temporal Consistency in a New Multimedia Synchronization Model. In *Proceedings of the ACM*

International Conference on Multimedia, Boston, MA, 18-22 Novembre 1996.

[41] Jean-Pierre Courtiat and Roberto Cruz de Oliveira. About time nondeterminism and exception handling in a temporal extension of LOTOS. In *Proceedings of the 14th International IFIP Symposium on Protocol Specification, Testing and Verification (PSTV'94)*, pages 33–49, Vancouver, Canada, 7-10 Juin 1994.

[42] J.P. Courtiat, L.F.R. Carmo, and R.C. De Oliveira. A general-purpose multimedia synchronization mechanism based on causal relations. *IEEE Journal on Selected Areas in Communications*, 14(1):185–195, Janvier 1996.

[43] J.P. Courtiat and R.C. De Oliveira. A reachability analysis of RT-LOTOS specifications. In *Proceedings of the 8th International IFIP Conference on Formal Description Techniques for Distributed Systems and Communications Protocols*, pages 101–108, Montréal, Canada, 17-20 Octobre 1995.

[44] J.P. Courtiat, C.A.S. Santos, C. Lohr, and O. Benaceu. Experience with RT-LOTOS, a temporal extension of the LOTOS formal description technique. *Computer Communications*, 23(12):1104–1123, Juillet 2000.

[45] Jean-Paul Delahaye. La mémoire de l'humanité. *Pour la science*, 299:98–103, Septembre 2002.

[46] Rod Deyo and Mike Jazayeri. DirectX 6.0 and Windows Media Technologies Bring Multimedia to Windows CE. *Microsoft Systems Journal*, 14(7), Juillet 1999. http://www.microsoft.com/msj/defaultframe.asp?page=/msj/0799/directx/di%rectx.htm.

[47] M. Diaz and P. Sénac. Time Stream Petri Nets, a Model for Timed Multimedia Information. In Robert Valette, editor, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, LNCS 815, pages 219–238, Zaragoza, Spain, Juin 1994. Springer-Verlag.

[48] Michel Diaz, David Garduno, Thierry Gayraud, and Stephane Owezarski. Multimedia Multicast Protocols based on Multimedia Models. In *Proceedings of the 8th International Conference on Multimedia Modeling (MMM'2001)*, CWI, Amsterdam, The Netherlands, 5-7 Novembre 2001.

[49] Nevenka Dimitrova, Rob Koenen, Heather Yu, Avideh Zakhor, Francis Galliano, and Charles Bouman. Video portals for the next century. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*. Electronic Proceedings, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[50] Stéphane Donikian and Gérard Hégron. A Declarative Desing Method for 3D Scene Sketch Modeling. In *Proceedings of the Eurographics'93 Conference*, volume 12(3), Barcelona, Spain, Septembre 1993.

[51] D. J. Duke, D. A. Duce, I. Herman, and G. Faconti. Specifying the PREMO Synchronization Objects. Technical Report ERCIM Research Report - ERCIM-02/97-R048, ISO/IEC, Fevrier 1997.

[52] Bruno Dutertre. On First Order Interval Temporal Logic. Technical Report CSD-TR-94-3, Royal Holloway, University of London, 7 Fevrier 1995.

[53] John Edwards and Linda Dailey Paulson. Smart Graphics: A New Approach to Meeting User Needs. *Computer*, 35(5):18–21, Mai 2002.

[54] F. Elisei, G. Bailly, M. Odisio, and P. Badin. Clones parlants 3D vidó-réalistes: Application à l'interprétation de FAP MPEG-4. In *Proceedings des 7^{emes} Journées d'études et d'échanges COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, Dijon, France, Novembre 2001.

[55] Jerome Euzenat. An Algebraic Approach for Granularity in Qualitative Space and Time Relations. In *Proceedings of the Conference IJCAI-95*, pages 894–900, 1995.

[56] FELIX 3D-Display project team. FELIX 3D-Display, 1999. http://www.vdivde-it.de/felix/.

[57] Sharon Flanck. Multimedia Technology in Context. *IEEE Multimedia*, Juillet - Septembre 2002.

[58] Felix Gaffiot. *Le grand Gaffio, Dictionaire Français-Latin*. Hachette, 2000.

[59] Alfonso Gerevini and Bernhard Nebel. Qualitative Spatio-Temporal Reasoning with RCC-8 and Allen's Interval Calculus: Computational Complexity. In *Proceedings of the Conference ECAI'02*, pages 312–316, Lyon, France, 21-26 Juillet 2002.

[60] Athula Ginige and San Murugesan. Web Engineering: An Introduction. *IEEE Multimedia*, Janvier - Mars 2001.

[61] Charles Goldfarb, Steven R. Newcomb, W. Eliot Kimber, and Peter J. Newcomb. A Reader's Guide to the HyTime Standard, Août 1997. HyTime Users' Group. http://www.hytime.org/papers/htguide.html.

[62] Stuart Goose and Carsten Möller. A 3D Audio Only Interactive Web Browser: Using Spatialisation to Convey Hypermedia Structure. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[63] R. Grigoras, V. Charvillat, and M. Douze. Contenu multimédia à mémoire. In *Proceedings des 7^{emes} Journées d'études et d'échanges COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, Dijon, France, Novembre 2001.

[64] William I. Grosky. Managing Multimedia Information in Database Systems. *Communications of the ACM*, 40(12):73–80, Decembre 1997.

[65] UVa User Interface Group. Alice: Rapid Prototyping for Virtual Reality. *IEEE Computer Graphics and Applications*, 15(3):8–11, Mai 1995.

[66] C. L. Hamblin. Instants and Intervals. In *Proceedings of the 1st Conference of the International Society for the Study of Time*, pages 324–331, New York, 1972.

[67] Peter Hanappe. *Design and implementation of an integrated environment for music composition and synthesis*. PhD thesis, Université Paris 6, Avril 1999.

[68] Lynda Hardman, Dick C.A. Bulterman, and Guido van Rossum. Links in Hypermedia: the Requirement for Context. In *Proceedings of the Fifth*

ACM Conference on Hypertext (Hypertext'93), pages 183–191, Seattle, WA, USA, 14-18 Novembre 1993. ACM Press.

[69] Lynda Hardman, Dick C.A. Bulterman, and Guido van Rossum. The Amsterdam Hypermedia Model: extending hypertext to support *real* multimedia. *Hypermedia Journal*, 5(1):47–69, Juillet 1993.

[70] Lynda Hardman, Guido van Rossum, and Dick C.A. Bulterman. Structured Multimedia Authoring. In *Proceedings of the ACM International Multimedia Conference*, pages 283–289, Anaheim, CA, USA, 1993.

[71] David Harel, Liran Carmel, and Doron Lancet. An Algorithmic Approach to Odor Communication and Reproduction. Technical Report MCS01-16, Mathematics & Computer Sience, Weizmann Institute Of Sience, Rehovot, Israel, 2001.

[72] E. A. Hauck and B. A. Dent. Burroughs' B6500/B7500 stack mechanism. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 245–251, 1968.

[73] Peter Henderson. Functional Geometry. In *Proceedings of the ACM Symposium on Lisp and Functional Programming*, pages 179–187, Pittsburg, PA, USA, 15-18 Août 1982.

[74] Thomas A. Henzinger, Jean-Franois Raskin, and Pierre-Yves Schobbens. The Regular Real-Time Languages. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, Aalborg, Denmark, 13-17 July 1998. Springer.

[75] I. Herman, C.S. Carson, J. Davy, D.A. Duce, P.J.W. ten Hagen, W.T. Hewitt, K. Kansy, B.J. Lurvey, R. Puk, G.J. Reynolds, and H. Stenzel. PREMO: An ISO Standard for a Presentation Environment for Multimedia Objects. In *Proceedings of the ACM International Conference on Multimedia*, San Francisco, CA, 15-20 Octobre 1994.

[76] I. Herman, G.J. Reynolds, and J. Van Loo. PREMO: an emerging standard for multimedia presentation. Technical report, CWI, 1995. Computer Science/Department of Interactive Systems, CS-R9554 1995.

[77] Saul Eduardo Pomares Hernandez, Jean Fanchon, Khalil Drira, and Michel Diaz. Causal Broadcast Protocol for Very Large Group Communication Systems. In *Proceedings of the 5th International Conference on Principles of DIstributed Systems (OPODIS'2001)*, Manzanillo, Mexico, 10-12 Decembre 2001.

[78] N. Hirzalla, Ben Falchuk, and Ahmad Karmouch. A Temporal Model for Interactive Multimedia Scenarios. *IEEE Multimedia*, 2(3), Fall 1995.

[79] Kim Ki Hong, Kim Poong Min, and Kim Hyun Bin. The Natural Sound Field Effect for Multimedia Contents. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[80] Andy Hopper. Preparing for the Digital Media Monsoons. In *Proceedings of the 6th ACM International Multimedia Conference*, pages 267–272, Bristol, England, 12-16 Septembre 1998.

[81] Jeffrey Jacobson and Zimmy Hwang. Unreal Tournament for Immersive Interactive Theater. *Communications of the ACM*, 45(1):39–42, Janvier 2002.

[82] R. Jain. Real reality. *IEEE Computer Graphics and Applications*, 20(1):40–41, Janvier - Février 2000.

[83] Muriel Jourdan, Nabil Layaïda, Loay Sabry-Ismaïl, Cécile Roisin, and Laurent Tardif. MADEUS: an Authoring Environment for Interactive Multimedia Documents. In *Proceedings of the 6th ACM International Multimedia Conference*, pages 267–272, Bristol, England, 12-16 Septembre 1998.

[84] Leander Kahney. 3D Projection Without the Glasses. *Wired*, 17 Novembre 2000. http://www.wired.com/news/technology/0,1282,40264,00.html.

[85] Chérif Keramane. *Spécification de présentations multimédia structurées interactives*. PhD thesis, Institut National Polytechnique de Grenoble (INPG), Octobre 1997.

[86] Chérif Keramane and Andrzej Duda. Interval Expressions – a Functional Model for Interactive Dynamic Multimedia Presentations. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 283–286, Hiroshima, Japan, Juin 1996.

[87] Rohit Khare and Adam Rifkin. The origin of (document) species. *Computer Networks and ISDN Systems*, 30:389–397, 1998.

[88] Rob Koenen. Overview of the MPEG-4 Standard. ISO/IEC JTC1/SC29/WG 11 N4030, Mars 2001.

[89] James Korris and Michael Macedonia. The End of Celluloid: Digital Cinema Emerges. *Computer*, 35(4):96–98, Avril 2002.

[90] Keeranor G. Kumar, James S. Lipscomb, Arun Ramchandra, and al. The HotMedia Architecture: Progressive and Interactive Rich Media for the Internet. Technical Report RC21519(97069)6JUL1999, IBM Internet Media Group, Juillet 1999. Aussi publié dans *IEEE transactions on Multimedia*, Vol. 3, No.2, Juin 2001, pp:253-267.

[91] John E. Laird. Research in Human-Level AI Using Computer Games. *Communications of the ACM*, 45(1):32–35, Janvier 2002.

[92] Nabil Layaïda. *Madeus: système d'édition et de présentation de documents structurés multimédia*. PhD thesis, Université Joseph Fourier - Grenoble 1, Juin 1997.

[93] Nabil Layaïda and Chérif Keramane. Maintaining Temporal Consistency of Multimedia Documents. In *Proceedings of the Effective Abstractions in Multimedia Layout, Presentation and Interaction ACM 95 Workshop*, San Francisco, CA, USA, Novembre 1995.

[94] Daniel Lecomte, Daniel Cohen, Philippe De Bellefonds, and Jean Barda. *Les normes et les standards du multimédia, 2^{eme} édition*. Dunod, 2000.

[95] Michael Lewis and Jeffrey Jacobson. Game Engines in Scientific Research. *Communications of the ACM*, 45(1):27–31, Janvier 2002.

[96] John Z. Li, M. Tamer Ozsu, and Duane Szafron. Modeling Video Spatial Relationships in an Object Model. Technical Report TR 96-06, Department of Computing Science, University of Alberta, Mars 1996.

[97] C. Liard and A. Beghdadi. Explorateur Audio-Tactile d'écran vidéo pour aveugles. In *Proceedings des 7^{emes} Journées d'études et d'échanges COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, Dijon, France, Novembre 2001.

[98] T.D.C. Little and A. Ghaffor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, 8(3), Avril 1990.

[99] T.D.C. Little and A. Ghafoor. Interval-Based Conceptual Models for Time-Dependant Multimedia Data. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551–563, Août 1993.

[100] Guojun Lu. *Communication and Computing for Distributed Multimedia Systems*. Artech House Publishers, 1996.

[101] Tuomas Lukka. An Introduction to VRML. *Linux Journal*, 57, 1999.

[102] Michael Macedonia. Linux in Hollywood: A Star is Born. *Computer*, 35(2):112–114, Février 2002.

[103] Hervé Martin, Michel Adiba, Raphael Lozano, and Françoise Mocellin. Management of Multimedia Data Using an Object-Oriented Database System. In *Proceedings of the DEXA'97 workshop*, Toulouse, France, 2-5 Septembre 1997.

[104] José M. Martínez. Overview of the MPEG-7 Standard. ISO/IEC JTC1/SC29/WG 11 N4031, Mars 2001.

[105] Patrice Maubourguet, editor. *Grand Larousse Universel*. Larousse, 1991.

[106] Thomas Meyer-Boudnik and Wolfgang Effelsberg. MHEG Explained. *IEEE Multimedia*, 2(1):26–38, Spring 1995.

[107] Françoise Mocellin. *Gestion de données et de présentations multimédias par un SGBD à objets*. PhD thesis, Université Joseph Fourier, Grenoble, Decembre 1997.

[108] Jonathan Moffett, John Hall, Andrew Coombes, and John McDermid. A Model for a Causal Logic for Requirements Engineering. *Requirements Engineering*, 1(1):27–46, 1996.

[109] Débora Christina Muchaluat-Saade and Luiz Fernando Gomes Soares. Hypermedia Spatio-Temporal Synchronization Relations also deserve First-Class Status. In *Proceedings of the 8th International Conference on Multimedia Modeling (MMM'2001)*, CWI, Amsterdam, The Netherlands, 5-7 Novembre 2001.

[110] Philippe Muller. A Qualitative Theory of Motion Based on Spatio-Temporal Primitives. In A.G. Cohn, L.K. Schubert, and S.C. Shapiro, editors, *Proceedings of the Principles of Knowledge Representation and Reasoning Conference (KR'98)*, pages 179–187, San Francisco, CA, USA, Juin 1998. Morgan Kaufmann.

[111] David H. Nguyen and Elizabeth D. Mynatt. Privacy Mirrors: Understanding and Shaping Socio-technical Ubiquitous Computing Systems. Technical Report GIT-GVU-02-16, Georgia Institute of Technology, 2002 Juin.

[112] Kohtaro Ohba, Takehito Tsukada, Tetsuo Kotoku, and Kazuo Tanie. Facial Expression Space for Smooth Tele-Communications. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition (FG'98)*, pages 378–383, Nara, Japan, 14-16, Avril 1998.

[113] Wei-Tsang Ooi, Brian Smith, Sugata Mukhopadhyay, and al. Dalì : A Multimedia Software Library. In *Proceedings of the SPIE Multimedia Computing and Networking Conference*, San Jose, CA, USA, 25-27 Janvier 1999.

[114] Charles B. Owen. Multiple Media Correlation: Theory and Applications. Technical Report PCS-TR98-335, Dartmouth College, Computer Science, Hanover, NH, Juin 1998.

[115] Paul Pazandak and Jaideep Srivastava. Experience with the DAMSEL Multimedia Specification Language. In *Proceedings of the Society for Photonics and Electronic (SPIE) East, Conference Multimedia*, Philadelphia, PA, USA, Septembre 1995.

[116] Paul Pazandak, Jaideep Srivastava, and John V. Carlis. The Temporal Component of DAMSEL. In *Proceedings of the Second Workshop on Protocols for Multimedia Systems (PROMS'95)*, Salzburg, Austria, 1995.

[117] M. Pérez-Luque and T. Little. A Temporal Reference Framework for Multimedia Synchronization. *IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication)*, 14(1):26–51, Janvier 1996.

[118] André Persidsky. *Director 5 Windows.* Eyrolles, 1997.

[119] Wayne Piekarski and Bruce Thomas. ARQuake: The Outdoor Augmented Reality Gaming System. *Communications of the ACM*, 45(1):36–38, Janvier 2002.

[120] Stéphane Lo Presti, Didier Bert, and Andrzej Duda. Composition d'objets multimédia à base d'opérateurs. In *Actes des 7^{emes} Journées d'études et d'échanges COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, Dijon, France, Novembre 2001.

[121] Stéphane Lo Presti, Didier Bert, and Andrzej Duda. TAO: Temporal Algebraic Operators for modeling Multimedia Presentations. *Journal of Network and Computer Applications*, A paraître 2002.

[122] Yannick Prié, Alain Mille, and Jean-Marie Pinon. An Approach for Modelling Audiovisual Documents with Annotations Interconnected Strata. In *Proceedings of the Conférence IC'98 , Ingénierie des connaissances*, pages 143–155, Pont-à-Mousson, Inria-Loria, Nancy, France, Mai 1998.

[123] Megan Quentin-Baxter. Hypermedia Learning Environments limit Access to Information. In *Proceedings of the Seventh International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, Brisbane, Australia, 1998.

[124] T.S. Raghu, R. Ramesh, and Andrew B. Whinston. Next Steps for Mobile Entertainment Portals. *Computer*, 35(5):63–70, Mai 2002.

[125] Richard Rashid, Robert Baron, Alessandro Forin, David Golub, Michael Jones, Daniel Julin, Douglas Orr, and Richard Sanzi. Mach: A Foundation for Open Systems. In *Proceedings of the Second Workshop on Workstation Operating Systems(WWOS2)*, Septembre 1989.

[126] Charles Robertson and John A. Robinson. Live Paper: Video Augmentation to Simulate Interactive Paper. In *Proceedings of the Seventh ACM Multimedia Conference (MM'99)*, Orlando, Florida, 30 Octobre - 5 Novembre 1999.

[127] Cécile Roisin. Documents multimédia structurés, Septembre 1999. Habilitation diriger des recherches, spcialit informatique, Institut National Polytechnique de Grenoble.

[128] Franck Rousseau. *Présentations multimédia synchronisées pour le WWW.* PhD thesis, Institut National Polytechnique de Grenoble (INPG), Janvier 1999.

[129] Franck Rousseau and Andrzej Duda. An Execution Architecture for Synchronized Multimedia Presentations. In *Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, pages 42–55, Berlin, Germany, 1998.

[130] Franck Rousseau and Andrzej Duda. Synchronized Multimedia for the WWW. In *Proceedings of the Seventh International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, pages 417–429, Brisbane, Australia, 1998.

[131] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, and Dick Bulterman. Cooperative Use of MHEG and HyTime in Hypermedia Environments. In *Actes de la* 4^{eme} *Conférence Internationale Hypertextes et Hypermédias – réalisations, outils et méthodes (H2PTM97)*, pages 57–73, Paris, France, Septembre 1997.

[132] Loay Sabry-Ismaïl and Ramzi Guetari. *L'Objet : Les représentations par ojets en conception*, chapter 2, Le modèle objet Madeus. Edition Hermes, 1998.

[133] P.N.M. Sampaio, C.A.S. Santos, and Jean-Pierre Courtiat. About the semantic verification of SMIL documents. In *Proceedings of the IEEE International Conference on Multimedia (ICME 2000)*, pages 1675–1678, New York, USA, 30 Juillet - 2 Août 2000.

[134] P.N.M. Sampaio, C.A.S. Santos, and Jean-Pierre Courtiat. Using a formal method to verify the temporal semantics of SMIL documents. In *Proceedings of the Conference on Software: Theory and Practice, IFIP World Congress*, pages 451–458, Beijing, China, 21-25 Août 2000.

[135] Harry Santoso, Laurent Dairaine, Serge Fdida, and Eric Horlait. Preserving Temporal Signature : a Way to Convey Time Constrained Flows. In *Proceedings of the IEEE GLOBECOM Conference (GLOBal telECOMmunications)*, Houston, USA, Novembre 1993.

[136] James Schepf, Joseph A. Konstan, and David Du. Doing FLIPS : Flexible Interactive Presentation Synchronization. In *Proceedings of the 1995 International Conference on Multimedia Computing and System*, Washington, DC, USA, Mai 1995.

[137] Patrick Schmitz. Multimedia Goes Corporate. *IEEE Multimedia*, Juillet - Septembre 2002.

[138] Pierre-Yves Schobbens, Jean-Franois Raskin, and Thomas A. Henzinger. Axioms for real-time logics. *Theoretical Computer Science*, 274(1–2):151–182, 2002.

[139] ISO International Standard. Information Technology Hypermedia/Time-Based Structuring Language (HyTime). ISO/IEC IS 10744:1997, Août 1992.

[140] Ralf Steinmetz. Synchronisation Properties in Multimedia Systems. *IEEE Journal on Selected Areas in Communication*, 8(3):401–412, 1990.

[141] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: Computing, Communications & Applications*. Prentice Hall, 1995.

[142] René Stolp, Angela Scheller, and Andreas Kraft. MHEG-5 Application Development. In *Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, pages 70–82, Berlin, Germany, 1998.

[143] Macromedia team. Macromedia Flash file format (SWF) Software Development Kit (SDK). Macromedia, 2001. http://www.macromedia.com/software/flash/open/licensing/fileformat/.

[144] Quicktime Team. Quicktime Concepts. Technical report, Apple, 1997. http://developer.apple.com/techpubs/quicktime/qtdevdocs/INMAC/INTROS/xx%Introductions.8.htm.

[145] Warner ten Kate, Dick Bulterman, Patrick Deunhouwer, Lynda Hardman, and Lloyd Rutledge. Presenting Multimedia on the Web and in TV Broadcast. In *Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, pages 56–69, Berlin, Germany, 1998.

[146] David L. Tennenhouse, Joel Adam, and David Carver al. A Software-Oriented Approach to the Design of Media Processing Environments. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Boston, MA, USA, Mai 1994.

[147] Michael Tooley. Causation and Temporal Relations. *The Newsletter of the Philosophy of Science Society, Japan*, Avril 1995. http://wwwsoc.nacsis.ac.jp/pssj/NL/99_4_25data/MICHAEL_TOOLEY.html.

[148] Peter van Beek. Reasoning about Qualitative Temporal Information. *Artificial Intelligence*, 58:297–326, 1992.

[149] Peter van Beek and Robin Cohen. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence*, 6:132–144, 1990.

[150] M.T.P. Vieira and M.T.P. Santos. Content-based Search on a MHEG-5 Standard-based Multimedia Database. In *Proceedings of the QPMIDS DEXA 97 workshop*, pages 341–350, Toulouse, France, Septembre 1997.

[151] Marc Vilain and Henry Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 377–382, Philadelphia, PA, USA, 1986.

[152] Steve Vinoski. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, Fevrier 1997.

[153] Jürgen Vogel and Martin Mauve. Consistency Control for Distributed Interactive Media. In *Proceedings of the Ninth ACM Multimedia Conference (MM'01)*, Ottawa, Ontario, Canada, 30 Setptembre - 5 Octobre 2001.

[154] Thomas Wahl and Kurt Rothermel. Representing Time in Multimedia Systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems.*, pages 538–543, Boston, MA, USA, Mai 1994.

[155] X3D Task Group Web3D Consortium. Information technology – Computer graphics and image processing – X3D, 2002.

[156] Ron Weiss, Andrzej Duda, and David K. Gifford. Composition and Search with a Video Algebra. *IEEE Multimedia*, 2(1):12–25, Spring 1995.

[157] Roberto Willrich, Pierre de Saqui Sannes, Pierre Senac, and Michel Diaz. *HTSPN: An Experiment in Formal Modeling of Multimedia Applications Coded in MHEG or Java*, pages 380–411. Idea Group Publishing, 2001.

[158] Roberto Willrich, Pierre de Saqui Sannes, Pierre Senac, and Michel Diaz. Multimedia Authoring with Hierarchical Timed Stream Petri Nets and Java. *Multimedia Tools and Applications*, 16(1):7–27, Janvier - Fevrier 2002.

[159] Frank Wolter and Michael Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *Proceedings of the Seventh International Conference Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 3–14, Breckenridge, Colorado, USA, 11-15 Avril 2000.

[160] Beat Wüthrich. Managing Time *and* Uncertainty. Technical Report CS95-4, The Hong Kong University of Science and Technology, Juin 1994.

[161] Hong Heather Yu, Deepa Kundur, and Ching-Yung Lin. Spies, Thieves and Lies: The Battle for Multimedia in the Digital Era. *IEEE Multimedia*, Juillet - Septembre 2001.

Annexe A

Annexe 1

Nous donnons ici les expressions des règles de réécriture conditionnelles intervenant dans les différents processus exposés dans cette thèse. La section A.1 indique les règles de réécriture conditionnelles du mécanisme de normalisation pour le calcul de la sémantique temporelle des programmes TAO, alors que la section A.2 définit celles de la réduction du nombre de contextes d'instructions à la compilation des programmes TAO.

A.1 Ecriture des règles de réécriture conditionnelles de la sémantique temporelle

Dans ces règles de réécriture conditionnelles, les variables I, I', J, K représentent des niveaux, i, j des entiers, p, q, r des points temporels et c, c' des conditions. La notation "E[x=v]" permet de substituer la variable x avec la valeur v dans l'expression E.

A.1.1 Règles contextuelles

Nom	Condition	Substitution
(lift)	$I' < I \ \wedge \ \mathsf{normal}(I')$ $\wedge \ (p \mapsto q \ \ \mathit{if} \ \ c) \in I'$ $\wedge \ p \neq \mathsf{beg} \ \wedge \ q \neq \mathsf{stop}$	$\dfrac{}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c\ \}}$
(p-start)	$I' < I \ \wedge \ (\mathsf{beg} \mapsto r) \in I'$	$\dfrac{I.\{\ p \mapsto \mathsf{start}(I') \ \ \mathit{if} \ \ c\ \}}{I.\{\ p \mapsto r \ \ \mathit{if} \ \ c\ \}}$

(p-end)	$I' < I \wedge$ $(p \mapsto \mathsf{stop} \ \ \mathit{if} \ \ c') \in I'$	$\dfrac{I.\{\ \mathsf{end}(I') \mapsto q \ \ \mathit{if} \ \ c\ \}}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ (c \ \wedge \ c')\ \}}$
(c-beg)	$I' < I \ \wedge \ (\mathsf{beg} \mapsto r) \in I'$	$\dfrac{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c\ \}}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c[\mathsf{beg}(I') = r]\ \}}$
(c-end)	$I' < I \wedge$ $(\mathsf{end}(J) \mapsto \mathsf{stop} \ \ \mathit{if} \ \ c') \in I'$	$\dfrac{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c\ \}}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ (c \ \wedge \ c')[\mathsf{end}(I') = \mathsf{end}(J)]\ \}}$
(p-event)	$I' < I \wedge$ $(\mathsf{end}(J) \mapsto \mathsf{stop} \ \ \mathit{if} \ \ c') \in I'$	$\dfrac{I.\{\ I'.\mathsf{end} \mapsto q \ \ \mathit{if} \ \ c\ \}}{I.\{\ \mathsf{end}(J) \mapsto q \ \ \mathit{if} \ \ c \ \wedge \ c'\ \}}$
(c-event)	$I' < I \wedge$ $(\mathsf{end}(J) \mapsto \mathsf{stop} \ \ \mathit{if} \ \ c') \in I'$	$\dfrac{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c\ \}}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ (c \ \wedge \ c')[I'.\mathsf{end} = \mathsf{end}(J)]\ \}}$
(c-while)	$J < K$	$\dfrac{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c \ \wedge \ p \ \mathsf{while} \ J \ \wedge \ p \ \mathsf{while} \ K\ \}}{I.\{\ p \mapsto q \ \ \mathit{if} \ \ c \ \wedge \ p \ \mathsf{while} \ J\ \}}$

A.1.2 Règles de coupure

L'écriture précise des diverses règles de coupure est obtenue en instanciant la règle générique de coupure donnée en 4.3.2.4 pour chaque opérateur. A noter que les conditions de ces règles de réécriture conditionnelles sont vides. Dans ces règles de réécriture conditionnelles, α, β, α_i et β_i désignent des racines de terme.

(cut-seq)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}seq[2])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{seq}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{start}(I'.2{:}\beta)\ \ \mathit{if}\ \ c_1\ \&\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2 \\
\qquad\quad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_3\ \}
\end{array}
$$

$$
\begin{array}{l}
\psi.\{\ p \mapsto \mathsf{stop}(I'.1{:}\alpha)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha \\
\quad \&\ p \mapsto \mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta\ \} \\
\cup \\
I'{:}\mathsf{seq}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{start}(I'.2{:}\beta)\ \ \mathit{if}\ \ c_1\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha)\ \vee\ p \succ \mathsf{beg}(I'.2{:}\beta)\ \vee\ \neg c) \\
\qquad\quad \&\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha)\ \vee\ (\neg\, c\ \wedge\ c_2) \\
\qquad\quad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta) \\
\qquad\qquad\qquad\qquad \vee\ (\neg\, c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha)\ \vee\ p \succ \mathsf{beg}(I'.2{:}\beta))\ \wedge\ c_3)\ \}
\end{array}
$$

(cut-loop)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}\mathsf{loop})\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{loop}.\{\ \forall i{\in}I\!N^*,\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{start}(I'.1{:}\alpha_{i+1})\ \ \mathit{if}\ \ c'_i\ ) \\
\qquad \&\ \forall i{\in}I\!N^*,\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c''_i\ )\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ \forall i{\in}I\!N^*,\ (\&\ p \mapsto \mathsf{stop}(I'.1{:}\alpha_i)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha_i\ )\ \} \\
\cup \\
I'{:}\mathsf{loop}.\{\ \forall i{\in}I\!N^*,\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{start}(I'.1{:}\alpha_{i+1})\ \ \mathit{if}\ \ c'_i\ \wedge\ ((c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha_1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vee\ p \succ \mathsf{beg}(I'.1{:}\alpha_{i+1}))\ \vee\ \neg c)\ ) \\
\qquad \&\ \forall i{\in}I\!N^*,\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha_i) \\
\qquad\qquad\qquad\qquad\qquad \vee\ (\neg c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha_1)\ \vee\ p \succ \mathsf{beg}(I'.1{:}\alpha_i))\ \wedge\ c''_i)\ )\ \}
\end{array}
$$

(cut-loop[n])

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}\mathsf{loop}[n])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{loop}[n].\{\ \forall i{\in}\{1\ldots n{-}1\},\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{start}(I'.1{:}\alpha_{i+1})\ \ \mathit{if}\ \ c'_i\ ) \\
\qquad\quad \&\ \forall i{\in}\{1\ldots n\},\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c''_i\ )\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ \forall i{\in}\{1\ldots n\},\ (\&\ p \mapsto \mathsf{stop}(I'.1{:}\alpha_i)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha_i\ )\ \} \\
\cup \\
I'{:}\mathsf{loop}[n].\{\ \forall i{\in}\{1\ldots n{-}1\},\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{start}(I'.1{:}\alpha_{i+1}) \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{if}\ \ c'_i\ \wedge\ ((c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha_1)\ \vee\ p \succ \mathsf{beg}(I'.1{:}\alpha_{i+1})))\ \vee\ \neg c)\ ) \\
\qquad\quad \&\ \forall i{\in}\{1\ldots n\},\ (\&\ \mathsf{end}(I'.1{:}\alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha_i) \\
\qquad\qquad\qquad\qquad\qquad \vee\ (\neg c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha_1)\ \vee\ p \succ \mathsf{beg}(I'.1{:}\alpha_i))\ \wedge\ c''_i)\ )\ \}
\end{array}
$$

(cut-master)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}\mathsf{master}[2])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{master}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ c_1\ \&\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2 \\
\qquad\qquad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_3\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'.1{:}\alpha)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha \\
\ \ \&\ p \mapsto \mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ \ c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta\ \} \\
\cup \\
I'{:}\mathsf{master}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ c_1\ \wedge\ ((c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vee\ p \succ \mathsf{end}(I'.2{:}\beta)))\ \vee\ \neg c) \\
\qquad\qquad \&\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha)\ \vee\ (\neg c\ \wedge\ c_2) \\
\qquad\qquad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta) \\
\qquad\qquad\qquad\qquad\qquad\qquad \vee\ (\neg c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha)\ \vee\ p \succ \mathsf{beg}(I'.2{:}\beta))\ \wedge\ c_3)\ \}
\end{array}
$$

(cut-min)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}\mathsf{min}[2])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{min}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_1\ \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'.1{:}\alpha),\mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha\ \} \\
\cup \\
I'{:}\mathsf{min}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha)\ \vee\ (\neg c\ \wedge\ c_1) \\
\qquad\qquad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta)\ \vee\ (\neg c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vee\ p \succ \mathsf{beg}(I'.2{:}\beta))\ \wedge\ c_2)\ \}
\end{array}
$$

(cut-max)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'{:}\mathsf{max}[2])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I'{:}\mathsf{max}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_1\ \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'.1{:}\alpha)\ \ \mathit{if}\ \ \ c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha \\
\ \ \&\ p \mapsto \mathsf{stop}(I'.2{:}\beta)\ \ \mathit{if}\ \ \ c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta\ \} \\
\cup \\
I'{:}\mathsf{max}[2].\{\ \mathsf{end}(I'.1{:}\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1{:}\alpha)\ \vee\ (\neg c\ \wedge\ c_1) \\
\qquad\qquad \&\ \mathsf{end}(I'.2{:}\beta) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.2{:}\beta) \\
\qquad\qquad\qquad\qquad\qquad\qquad \vee\ (\neg c\ \wedge\ (p \prec \mathsf{beg}(I'.1{:}\alpha)\ \vee\ p \succ \mathsf{beg}(I'.2{:}\beta))\ \wedge\ c_2)\ \}
\end{array}
$$

(cut-alt)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I' : \mathsf{alt}[n])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I' : \mathsf{alt}[n].\{\ \forall i \in \{1 \ldots n\}, (\&\ (\forall j \in \{1 \ldots n\} \wedge j \neq i,\ (\&\ \mathsf{end}(I'.(2i-1) : \alpha_i) \mapsto \mathsf{stop}(I'.(2j-1) : \alpha_j)\ ) \\
\qquad\qquad\qquad\qquad \&\ \mathsf{end}(I'.(2i-1) : \alpha_i) \mapsto \mathsf{start}(I'.(2i) : \beta_i))\ \ \mathit{if}\ \ c'_i \\
\qquad\qquad\qquad\qquad \&\ \mathsf{end}(I'.(2i) : \beta_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c''_i\ )\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ \forall i \in \{1 \ldots n\}, (\&\ p \mapsto \mathsf{stop}(I'.(2i-1) : \alpha_i)\ \ \mathit{if}\ \ c \wedge p\ \mathsf{while}\ I'.(2i-1) : \alpha_i \\
\quad \&\ p \mapsto \mathsf{stop}(I'.(2i) : \beta_i)\ \ \mathit{if}\ \ c \wedge p\ \mathsf{while}\ I'.(2i) : \beta_i\ )\ \} \\
\cup \\
I' : \mathsf{alt}[n].\{\ \forall i \in \{1 \ldots n\}, (\&\ (\forall j \in \{1 \ldots n\} \wedge j \neq i,\ (\&\ \mathsf{end}(I'.(2i-1) : \alpha_i) \mapsto \mathsf{stop}(I'.(2j-1) : \alpha_j) \\
\qquad\qquad\qquad\qquad \mathit{if}\ \ c'_i \wedge ((c \wedge (p \prec \mathsf{beg}(I'.1 : \alpha_1) \vee p \succ \mathsf{end}(I'.(2j-1) : \alpha_j))) \vee \neg c)) \\
\qquad\qquad\qquad\qquad \&\ \mathsf{end}(I'.(2i-1) : \alpha_i) \mapsto \mathsf{start}(I'.(2i-1) : \alpha_i) \\
\qquad\qquad\qquad\qquad \mathit{if}\ \ c'_i \wedge ((c \wedge (p \prec \mathsf{beg}(I'.1 : \alpha_1) \vee p \succ \mathsf{beg}(I'.(2i-1) : \alpha_i))) \vee \neg c)) \\
\qquad\qquad\qquad\qquad \&\ \mathsf{end}(I'.(2i) : \beta_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c \wedge p\ \mathsf{while}\ I'.(2i) : \beta_i) \\
\qquad\qquad\qquad\qquad \vee (\neg c \wedge (p \prec \mathsf{beg}(I'.1 : \alpha_1) \vee p \succ \mathsf{end}(I'.(2i) : \beta_i)) \wedge c''_i)\ )\ \}
\end{array}
$$

(cut-par)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I' : \mathsf{par}[a_1, \ldots, a_n])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I' : \mathsf{par}[a_1, \ldots, a_n].\{\ \forall i \in \{1 \ldots n\}, (\&\ \mathsf{end}(I'.i : \alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c'_i\ )\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ \forall i \in \{1 \ldots n\}, (\&\ p \mapsto \mathsf{stop}(I'.i : \alpha_i)\ \ \mathit{if}\ \ c \wedge p\ \mathsf{while}\ I'.i : \alpha_i\ )\ \} \\
\cup \\
I' : \mathsf{par}[a_1, \ldots, a_n].\{\ \forall i \in \{1 \ldots n\}, (\&\ \mathsf{end}(I'.i : \alpha_i) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c \wedge p\ \mathsf{while}\ I'.i : \alpha_i) \vee (\neg c \wedge c'_i)\ )\ \}
\end{array}
$$

(cut-cut)

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I' : \mathsf{cut}[J])\ \ \mathit{if}\ \ c\ \} \\
\cup \\
I' : \mathsf{cut}[J].\{\ \mathsf{end}(I'.1 : \alpha) \mapsto \mathsf{stop}(J)\ \ \mathit{if}\ \ c_1 \\
\qquad \&\ \mathsf{end}(I'.1 : \alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2\ \}
\end{array}
$$

$$
\begin{array}{l}
I.\{\ p \mapsto \mathsf{stop}(I'.1 : \alpha)\ \ \mathit{if}\ \ c \wedge p\ \mathsf{while}\ I'.1 : \alpha\ \} \\
\cup \\
I'.\mathsf{cut}[J].\{\ \mathsf{end}(I'.1 : \alpha) \mapsto \mathsf{stop}(J)\ \ \mathit{if}\ \ c_1 \wedge ((c \wedge (p \prec \mathsf{beg}(I'.1 : \alpha) \vee p \succ \mathsf{end}(J))) \vee \neg c) \\
\qquad \&\ \mathsf{end}(I'.1 : \alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c \wedge p\ \mathsf{while}\ I'.1 : \alpha) \vee (\neg c \wedge c_2)\ \}
\end{array}
$$

(cut-ended)

$I.\{\ p \mapsto \mathsf{stop}(I'\!:\!\mathsf{ended}[v])\ \ \mathit{if}\ \ c\ \}$
$\cup$
$I'\!:\!\mathsf{ended}[v].\{\ v \mapsto \mathsf{stop}(I'.1\!:\!\alpha)\ \ \mathit{if}\ \ c_1$
$\qquad\qquad \&\ \mathsf{end}(I'.1\!:\!\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ c_2\ \}$

$I.\{\ p \mapsto \mathsf{stop}(I'.1\!:\!\alpha)\ \ \mathit{if}\ \ c\ \wedge\ p\ \mathsf{while}\ I'.1\!:\!\alpha\ \}$
$\cup$
$I'\!:\!\mathsf{ended}[v].\{\ v \mapsto \mathsf{stop}(I'.1\!:\!\alpha)\ \ \mathit{if}\ \ c_1\ \wedge\ ((c\ \wedge\ (p \prec \mathsf{beg}(I'.1\!:\!\alpha)\ \vee\ p \succ \mathsf{end}(I'.1\!:\!\alpha)))\ \vee\ \neg c)$
$\qquad\qquad \&\ \mathsf{end}(I'.1\!:\!\alpha) \mapsto \mathsf{stop}\ \ \mathit{if}\ \ (c\ \wedge\ p\ \mathsf{while}\ I'.1\!:\!\alpha)\ \vee\ (\neg c\ \wedge\ c_2)\ \}$

A.2 Ecriture des règles de réécriture conditionnelles de réduction du nombre de contextes

Les notations suivantes sont introduites pour l'écriture des règles contextuelles:

- $C.\{i\!:\!I\}$ dénotera une instruction du contexte C de numéro i, qui vérifiera le test d'appartenance $I \in C$;
- On notera $p \Longrightarrow Q_1,q,Q_2$ l'instruction d'action observée p, d'action générée la séquence d'action Q_1, l'action q et la séquence d'action Q_2, les séquences d'actions pouvant être éventuellement vides; on notera qu'une action q est dans une séquence d'actions Q_1 par $q \in Q_1$;
- Comme pour la normalisation, les instructions de démarrage et fin d'un contexte ont une place particulière lors de la remontée des instructions dans le contexte supérieur; elles sont notées $\mathsf{strt} \Longrightarrow Q_1$, $p \Longrightarrow Q_1,\mathsf{stp},Q_2$ et $\mathsf{parEx}(I_1; p \Longrightarrow Q_1,\mathsf{stp},Q_2; I_2)$, Q_1 et Q_2 dénotant des séquences d'actions et I_1 et I_2 des séquences d'instructions (séparées par le symbole ";") qui peuvent être éventuellement vides;
- deux contextes C et C' seront hiérarchisés selon la relation d'ordre $<$ telle que: $C < C' \Leftrightarrow \exists p,Q_1,Q_2 \mid p \Longrightarrow Q_1,\mathsf{strt}(C'),Q_2 \in C$
- Tout comme le prédicat normal pour la normalisation, la condition $\mathsf{indep}(C,C')$, pour C et C' des contextes, autorise la remontée d'une instruction; elle équivaut au test suivant:
  $(C \leq C)'\ \wedge\ (\exists p,Q_1,Q_2 \mid (p \Longrightarrow Q_1,\mathsf{strt}(C'),Q_2) \in C)$
  $\qquad \wedge\ (\nexists p,Q_1,Q_2,I_1,I_2 \mid$
  $((p \Longrightarrow Q_1,\mathsf{stp}(C'),Q_2) \in C)\ \vee\ ((\mathsf{parEx}(I_1; p \Longrightarrow Q_1,\mathsf{stp}(C'),Q_2 \in C; I_2) \in C))$
  elle est vraie si le contexte C démarre le contexte C' et ne le stoppe pas; si elle est fausse, les instructions du contexte C' restent dans ce contexte;
- Un événement e d'une instruction I est substitué par la suite d'actions Q de la manière suivante: $I[e = Q]$; s'il e figure en action observée et que Q est une action unique, alors e est remplacé par Q, sinon aucune modification n'est apportée; si e figure en action générée, toutes ses occurrences sont remplacées par Q.

Dans ces règles de réécriture conditionnelles, C et C' sont des contextes, p et q des garde et action respectivement, Q_1, Q_2 et Q_3 des séquences d'actions générés, I une instruction, I_1, I_2 et I_3 des séquences d'instructions et i un entier. La variable *indice* est utilisée par le processus de compilation des programmes.

Nom	Condition	Substitution
(I_lift1)	$\mathsf{indep}(C,C') \ \wedge \ (p \Longrightarrow Q_1) \in C'$ $\wedge \ p \neq \mathsf{strt} \ \wedge \ \mathsf{stp} \notin Q_1$	$\dfrac{}{\boldsymbol{C.\{}\ i{:}p \Longrightarrow Q_1\ \boldsymbol{\}}}$
(I_lift2)	$\mathsf{indep}(C,C') \ \wedge \ (\mathsf{parEx}(I_1)) \in C' \wedge$ $\nexists p,Q_1,Q_2 \mid (p \Longrightarrow Q_1,\mathsf{stp},Q_2) \in I_1$	$\dfrac{}{\boldsymbol{C.\{}\ i{:}\mathsf{parEx}(I_1)\ \boldsymbol{\}}}$
(I_strt)	$\mathsf{indep}(C,C')$ $\wedge \ (\mathsf{strt} \Longrightarrow Q_1) \in C'$	$\dfrac{\boldsymbol{C.\{}\ \mathit{indice}{:}I\ \boldsymbol{\}}}{\boldsymbol{C.\{}\ \mathit{indice}{:}I[\mathsf{strt}(C') = Q_1]\ \boldsymbol{\}}}$
(I_stp1)	$\mathsf{indep}(C,C')$ $\wedge \ (p \Longrightarrow Q_1,\mathsf{stp},Q_2) \in C'$	$\dfrac{\boldsymbol{C.\{}\ i{:}\mathsf{stp}(C') \Longrightarrow Q_3\ \boldsymbol{\}}}{\boldsymbol{C.\{}\ i{:}p \Longrightarrow Q_1,Q_3,Q_2\ \boldsymbol{\}}}$
(I_stp2)	$\mathsf{indep}(C,C') \wedge$ $(\mathsf{parEx}(I_1; p \Longrightarrow Q_1,\mathsf{stp},Q_2; I_2)) \in C'$	$\dfrac{\boldsymbol{C.\{}\ i{:}\mathsf{stp}(C') \Longrightarrow Q_3\ \boldsymbol{\}}}{\boldsymbol{C.\{}\ i{:}\mathsf{parEx}(I_1; p \Longrightarrow Q_1,Q_3,Q_2; I_2)\ \boldsymbol{\}}}$