Institut
National de
l'Audiovisuel

Laboratoire
d'Informatique
de Paris 6

Jean Carrive

Classification de séquences audiovisuelles

Thèse soutenue le 12 septembre 2000 devant le jury composé de

Mme Cécile Roisin (rapporteur)

M. Roger Mohr (rapporteur)

M. Jean-François Perrot (président)

Mme Christine Golbreich (examinateur)

M. Amedeo Napoli (examinateur)

M. Philippe Joly (examinateur)

M. Rémi Ronfard (examinateur, encadrant INA)

M. François Pachet (examinateur, directeur de thèse)

Pour obtenir le grade de l'université Paris 6 (Pierre et Marie Curie)

Les travaux que nous présentons ont été effectués dans le cadre d'une convention industrielle de formation par la recherche (CIFRE) associant l'Institut national de l'audiovisuel (INA) et le Laboratoire d'informatique de Paris 6 (Lip6) sous l'égide de l'Association nationale de la recherche technique (ANRT). Ce travail s'est déroulé au Département de la recherche prospective de l'INA sous la direction de M. François Pachet (Lip6) et a été encadré à l'INA par M. Rémi Ronfard.

Remerciements

L'une des spécificités des thèses CIFRE est de bénéficier d'un double encadrement, universitaire et industriel. Je remercie François Pachet, mon directeur de thèse, qui a sans relâche suivi et orienté mon travail, qui a su déceler dans ce que je lui apportais les points difficiles et les points intéressants, qui a ouvert des portes au fond des impasses et qui, soutien moral autant que scientifique, a constamment œuvré pour que cette thèse parvienne à son terme. Je remercie Rémi Ronfard, qui a encadré mon travail à l'INA et au sein du projet DiVAN, dont les compétences scientifiques et la connaissance du domaine, l'exigence et la curiosité vivifiante m'ont tenus sur un fil tendu entre des préoccupations purement formelles et le besoins applicatifs.

Je remercie Cécile Roisin et Roger Mohr d'avoir accepté de rapporter mon travail et pour leurs remarques. Merci aux autres membres du jury, Christine Golbreich, Amedeo Napoli et Philippe Joly, et à Jean-François Perrot qui m'a fait l'honneur de le présider.

Je tiens particulièrement ici à ne pas oublier ceux qui ont fait en sorte ce que ma thèse puisse démarrer. Merci à Michel Dojat qui m'a amicalement offert de travailler avec lui en attendant que certaines démarches aboutissent. Merci à Jean-François Allouis qui, avant les réorientations des activités de recherche de l'INA, a fait en sorte qu'une thèse s'inscrive dans le projet Anim'2000, dont je salue toute l'équipe : Jean-Michel Benech, Patrick Courounet, Laurent Vinet, Gérard Iglésias, Sébastien Gruhier et Philippe Converset. Mon activité s'est depuis déroulée dans l'équipe indexation ; à tous ses membres un grand merci, et plus particulièrement à Michèle Wautelet, Vincent Brunie, Alain Perrier, Odile Phidias et, bien sûr, à mes ex-condisciples thésards, Karine Lespinasse, Emmanuel Veneau, Gwendal Auffret et Thomas Dechilly. Ma reconnaissance également à Bruno Bachimont, directeur de la recherche de l'INA, qui a permis que ma thèse continue malgré les bouleversements internes.

Par pudeur, un simple salut à Pierre Roy, qui a mis à mon entière disposition son intelligence, son temps, son travail, ses compétences et son amitié.

Mes chaleureux remerciements à tous ceux qui, de près ou de loin, m'ont aidé, porté ou supporté durant cette période, à mes parents, ma sœur, mon frère, à Lætitia et Agnès, mes enfants, à Isabelle, leur maman. Enfin, pour leur affection, merci à Mathias, Augusto, à Elizabeth et Olivier, à Bénédicte.

Résumé

Dans le contexte de l'indexation automatique de documents audiovisuels, nous nous intéressons aux *collections* d'émissions de télévision respectant une structure générale commune, telles que certains journaux télévisés ou certains magazines d'actualité ou de variétés. Nous cherchons à représenter et à reconnaître dans de telles émissions des séquences caractéristiques pouvant être annotées comme un tout par un(e) documentaliste, comme une chanson dans un magazine de variétés ou un reportage de journal télévisé. Nous proposons un langage de représentation de ces classes de séquences appelées *canevas*. La définition d'un canevas repose d'une part sur un vocabulaire d'événements primitifs directement observables, tels que les logos ou les jingles, et d'autre part sur un ensemble de relations temporelles prescrites entre ces événements. Nous proposons de représenter les événements primitifs dans un formalisme de logiques de descriptions et nous étendons l'algèbre d'intervalles temporels de Allen avec un opérateur d'itération noté « * » désignant des séquences d'événements temporellement adjacents ainsi qu'avec des contraintes spécifiques faisant intervenir une négation. Nous présentons le problème de la reconnaissance des canevas comme un problème de satisfaction de contraintes et nous proposons une méthode de résolution efficace ainsi qu'une implémentation de cette méthode dans un système de satisfaction de contraintes existant. Nous exposons enfin une série d'expérimentations conduites à l'Institut National de l'Audiovisuel (INA) sur un corpus de journaux télévisés.

Mots clé : *indexation audiovisuelle, collection d'émissions, canevas, séquences, logiques de descriptions, raisonnement temporel, satisfaction de contraintes*

Abstract

In the context of automatic indexing of audio-visual documents, we are interested in *collections* of programs that respect a common general structure, such as broadcast news or some news or variety programs. We aim at representing and recognizing in scuch programs some characteristic sequences which can be annotated as a whole by a documentalist, such as a song in a variety program or a report in a broadcast news. We propose a language that represent such classes of sequences as *templates*. The definition of a template relies on the one hand on a vocabulary of directly observable primitive events, such as logos or jingles, and on the other hand on a set of temporal relations between these events. We propose to represent primitive events in a description logic formalism and we extend Allen's temporal interval algebra with an iteration operator "*" denoting a sequence of temporally adjacent events, and with specific constraints involving a negation. We present the template recognition problem as a constraint satisfaction problem and propose an efficient solving procedure, as well as an implementation of this procedure in an existing constraint satisfaction framework. We finally expose a set of experiments conducted at the French National Institute of Audiovisual (INA) on a corpus of broadcast news.

Keywords: *audiovisual indexing, program collection, template, sequence, description logics, temporal reasoning, constraint satisfaction*

Table des Matières

Introduction

La numérisation croissante des documents audiovisuels entraîne des évolutions des usages et des besoins (Poncin, 1997). L'accès à d'énormes bases de films ou d'émissions devient maintenant possible. Les détenteurs d'archives audiovisuelles sont amenés à mettre en place des bases de données de documents numérisés. Aujourd'hui, l'Institut National de l'Audiovisuel (INA) possède 700 000 heures d'archives de télévision et emploie 200 documentalistes à plein temps pour annoter les documents à archiver. L'organisation de ces bases de données ainsi que les modes d'accès qu'elles offrent posent de nouveaux problèmes à la communauté des chercheurs. La variété des problèmes posés ainsi que la diversité des compétences à faire collaborer ont fait émerger l'*indexation des documents audiovisuels* comme nouvelle discipline. « Indexation » est un terme fortement polysémique, sur lequel il faut s'entendre. L'Association française de normalisation (AFNOR) définit l'indexation comme la représentation de résultats d'une analyse en langue naturelle ou dans un langage formalisé. Selon les disciplines, l'indexation recouvre des activités différentes. Ainsi, l'indexation automatique du flux audiovisuel – on parle alors d'indexation par le *contenu* – consiste à extraire des primitives par analyse d'image (couleurs, formes, textures) et à stocker ces primitives dans des bases de données. Dans le domaine des bases de données, l'indexation regroupe les méthodes permettant d'optimiser l'accès aux données (création d'index). Pour les documentalistes, l'indexation est l'activité de documentation. Notons que d'importants travaux ont été réalisés pour l'indexation d'images fixes. Ainsi (Schmid & Mohr, 1997), par exemple, cherche à retrouver dans une grande base une image donnée à partir d'une portion d'image en s'appuyant sur une représentation de valeurs de gris prises à différents points d'intérêt. L'indexation intéresse par conséquent des compétences très diverses venant aussi bien des sciences technologiques que des sciences humaines. Une réflexion sur ce thème est maintenant engagée ; les travaux autour de la future norme MPEG-7 s'inscrivent sur le thème de la description de contenu et mettent à jour les difficultés de la tâche. Sans cependant nous engager sur cette question de la description de contenu, nous pensons qu'aux compétences « traditionnelles » associées à l'indexation (analyse du signal, analyse d'image, bases de données, compression de données, etc.) doivent s'ajouter des techniques symboliques de représentation et de raisonnement. Des travaux tels que (Aigrain *et al.*, 1997) et (Faudemay, 1997) illustrent bien cette nécessité.

Dans le cadre général de l'archivage des œuvres audiovisuelles, et plus particulièrement d'émissions de télévision, nous cherchons à assister le travail des documentalistes par des procédés automatiques. Pour ce faire, l'idée principale est de calculer une représentation synthétique de l'émission à annoter, en faisant apparaître en particulier sa structure temporelle. Le problème de base est donc l'analyse structurelle automatique des documents. On distingue communément deux niveaux de structure dans un document audiovisuel : le *plan* et la *séquence*. Un plan est ce qui est tourné en une seule prise de caméra : c'est ce qui est filmé entre l'instant où le réalisateur dit « moteur » et l'instant où il dit « coupez ». C'est dans la tradition du cinéma classique – tradition le plus souvent respectée à la télévision – l'unité syntaxique. La séquence constitue quant à elle, dans cette tradition, l'unité sémantique. Une séquence respecte en général une certaine unité : unité de lieu, d'action ou de temps. Hormis le cas du plan-séquence, une séquence est en général composée de plusieurs plans. L'analyse structurelle d'un document consiste à en *isoler* et *caractériser* les plans et les séquences.

Le découpage d'un document en plans est un problème maintenant bien étudié, et des méthodes robustes existent – par exemple (Yeo & Liu, 1995). Le découpage en séquences – ou *macro-segmentation* – a également été abordé par des méthodes numériques (Merlino *et al.*, 1997). (Aigrain *et al.*, 1997) proposent également des règles symboliques de découpage permettant d'identifier certains changements de séquences. Ces méthodes ont en commun qu'elles sont généralistes : elles s'appliquent potentiellement sur tout document. Nous plaçons notre travail dans la continuité de ces travaux, en faisant au préalable deux remarques. La première de ces remarques est que nous pensons qu'il est très difficile de déterminer des critères permettant d'identifier une séquence dans le cas général, de même qu'il est très difficile de caractériser un plan ou une séquence. Des travaux ont toutefois tenté de donner des typologies des séquences (Metz, 1966; Bourdon, 1997), mais qui sont difficilement exploitables par des processus automatiques. Il n'existe pas à notre connaissance de typologie générale des plans. La seconde remarque est que dans le cas qui nous intéresse – les émissions de télévision –, il est fréquent que des informations complémentaires sur la structure des émissions soient disponibles. Beaucoup d'émissions périodiques respectent le même canevas d'une émission sur l'autre. Beaucoup d'émissions *composites* (le terme est employé par le documentaliste), c'est-à-dire d'émissions composées de plusieurs types de séquences, par exemple interviews et chansons, studio et reportage, sont construites sur des modèles préétablis. C'est souvent le cas des émissions de variété, des « talk-shows », des magazines culturels ou d'information, où le caractère général de l'émission doit être préservé d'une émission sur l'autre, même si le réalisateur change. On parle alors de *collection* de documents. L'Inathèque de France établit des *fiches collection* dans lesquelles sont consignées des informations portant sur la collection dans son ensemble. La figure 1 ci-dessous donne un large extrait de la fiche collection du journal de « France 2 » pour l'année 1992-1993. L'intégralité de la fiche collection figure en annexe.

Titre propre	F2 le journal 20H00
Chaîne de diffusion	France 2
Canal	Réseau 2
[…]	
Genre	Journal télévisé
Type de description	Collection d'émissions composites
Médiamétrie	Information, journal national
Générique	PRE,Masure Bruno; PRE,Bilalian Daniel
Chapeau	Deuxième édition quotidienne de l'actualité nationale et internationale de la journée, le journal de 20 heures est diffusé en direct et dure environ 40 minutes. Il est réalisé en studio et s'organise en une succession de plateaux - reportages. […]
Résumé	Le présentateur annonce lui-même le sommaire du journal. Au cours de ce sommaire, il peut y avoir un plan de l'invité présent sur le plateau ainsi qu'une séquence sur un fait brûlant de l'actualité. Bruno Masure, s'adresse aux téléspectateurs par un "Mesdames, messieurs Bonsoir" et lance son premier sujet. Les sujets vont de l'actualité nationale à l'internationale, pour reprendre sur le national et terminer en principe sur les nouvelles du sport et des arts et spectacles.

Dispositifs	Le présentateur nomme les journalistes ayant préparé le sujet annoncé au moment du plateau lancement. En fonction de l'actualité, il peut y avoir une chronique animée par un journaliste spécialisé de la rédaction. En fin d'édition le journaliste annonce la météo, […] Particularités : - La durée générale de l'édition varie en fonction de l'actualité. - L'édition du jeudi dure environ 20 minutes […] GENERIQUE .- Début Le générique est organisé en deux temps. Images de synthèse de la planète en mouvement : ton bleu foncé avec incrustation de rouge en alternance avec le titre "journal" qui s'affiche deux fois en flash sur un fond bleu ciel. Arrivée pleine écran du présentateur cadré serré en plan poitrine, zoom avant, plan rapproché et lancement du journal. La musique court tout au long du générique jusqu'à la fin de l'annonce des titres avec une baisse d'intensité du volume sur les titres. Absence de musique lorsqu'il y a une séquence d'images relative à un fait marquant de l'actualité. .- Fin […] DISPOSITIFS PLATEAU .- Dispositifs techniques Trois caméras au sol : deux faisant face au journaliste, une autre derrière lui située à sa droite ou à sa gauche. Une quatrième caméra prend les plans d'ensemble en plongées plus ou moins verticales. .- Plateau journaliste Deux valeurs de plan pour le présentateur : un plan poitrine et un plan rapproché. Vue d'ensemble en plongée pour le dispositif plateau .- Plateau journaliste - invité Mise en images caractérisée par l'alternance de plans d'ensemble, poitrine, rapproché entre le journaliste et l'invité. Les plans d'ensemble sont pris de biais avec en amorce le journaliste de dos. .- Incrustations * […] Affichage des incrustations plateau sous forme d'affiche : quasi constantes, elles servent d'introduction au sujet en donnant le type et la tonalité du sujet. Disposées en haut à droite, elles sont rectangulaires en position verticale. Dispositif général : fond blanc avec une barre rouge en haut et un encadré noir en bas en général. Présence de texte en bas et en haut avec au centre une image d'illustration. .- Décor et mobilier * […] Le décor est construit en deux demi cercle positionnés l'un derrière l'autre. Deux colonnes de moniteurs encadrent une grande vitre aux reflets bleutés sur la gauche suivi par un panneau clair en allant sur la droite. Le décor du plateau est surexposé. […] DISPOSITIFS ELEMENTS ..- Reportage * Incrustations : Affichage du logo du journal uniquement sur les reportages […]

	* Mention du nom du/des journalistes, commentateurs, sur un bandeau bleu transparent en bas de l'écran, vers la fin du sujet [...] CONSTRUCTION GENERALE .- L'enchaînement sujet - reportage se fait sur un rythme rapide par un fondu, suivi parfois par une vue d'ensemble du plateau avant de reprendre le journaliste en plan poitrine puis zoom avant plan rapproché ou bien en plan poitrine, ou directement en plan rapproché.

figure 1 : extrait d'une fiche collection du journal de « France 2 »

Nous cherchons à exploiter certaines des informations contenues dans les fiches collection dans le but de représenter et reconnaître des séquences caractéristiques d'une collection, telles qu'un reportage du journal télévisé de « France 2 », une séquence de reportage étant définie comme une séquence de plans adjacents montrée entre deux plans de studio consécutifs.

En premier lieu, nous cherchons à représenter les caractéristiques pouvant être extraites automatiquement du flux audiovisuel, comme les diverses catégories d'incrustation de texte indiquant un nom de lieu ou de personne, les logos, les génériques, éventuellement les diverses prises de vue du studio, etc. (voir la fiche collection ci-dessus). En second lieu, nous cherchons à représenter les informations d'ordre temporel : « l'enchaînement sujet-reportage se fait par un fondu », par exemple, ou « mention du nom du journaliste (...) sur un bandeau bleu transparent en bas de l'écran, vers la fin du sujet ».

Nous proposons de représenter les catégories d'objets – ou *événements* – pouvant être extraits automatiquement du flux audiovisuel dans un formalisme de logiques de descriptions. Ce type de formalisme permet d'une part de représenter les classes d'événements par un treillis de concepts organisés automatiquement en taxinomie par des liens de généralité – ou *subsomption* – et d'autre part de classifier automatiquement les événements eux-mêmes – les *individus*, selon la terminologie des logiques de descriptions – dans la taxinomie en calculant pour chaque événement la liste de ses parents les plus spécifiques dans la taxinomie. Nous proposons ensuite d'étendre le système de reconnaissance de plans[1] T-Rex de Robert Weida (Weida, 1992a), extension temporelle d'un formalisme de logiques de descriptions, pour représenter et reconnaître des classes de séquences audiovisuelles. Les séquences sont représentées par des *canevas*, des graphes dont les sommets sont associés à des classes d'événements de la taxinomie ou, récursivement, à d'autres canevas, et dont les arêtes sont des contraintes temporelles dans l'algèbre temporelle de Allen. De plus, nous introduisons dans le langage de description des canevas un opérateur d'itération « * » désignant une *séquence itérée*, c'est-à-dire une séquence d'objets de même classe et temporellement adjacents. Nous présentons alors le problème de la reconnaissance des occurrences des canevas comme un problème de satisfaction de contraintes (CSP, pour *Constraint Satisfaction Problem*) et proposons une méthode de résolution fondée sur une représentation implicite efficace des séquences itérées et sur des méthodes rapides de filtrage des contraintes temporelles portant sur de ces séquences itérées. Une implémentation de cette méthode de résolution est présentée, implémentation réalisée autour du système de résolution de contraintes BackJava (Roy *et al.*, 1999).

[1] Au sens de *plan recognition*

La suite de ce document est organisée de la manière suivante. Dans le premier chapitre, nous présentons le problème de l'indexation de documents audiovisuels à travers des travaux réalisés principalement dans le domaine de l'analyse des flux audio et vidéo, dans le domaine de la macro-segmentation et dans le domaine de la représentation des données de description de la vidéo. Dans le second chapitre, nous présentons les travaux que nous avons effectués avec le système de logiques de descriptions CLASSIC (Borgida *et al.*, 1989) et nous montrons les limites d'une telle approche pour la classification d'objets temporels. Dans le troisième chapitre, nous présentons le problème de la classification de séquences audiovisuelles comme un problème de reconnaissance de plans. Nous décrivons T-Rex, formalisme sur lequel nous nous sommes appuyés, ainsi que les extensions que nous y avons apportées – représentation des séquences itérées, ainsi qu'ajout de contraintes spécifiques interdisant l'apparition d'un événement d'une classe donnée entre deux éléments d'une séquence. Nous discutons alors des problèmes posés par ces extensions, notamment quant à la possibilité de calculer la relation de subsomption entre deux canevas. Dans le quatrième chapitre, nous décrivons le système Clavis (*Classification of Audiovisuel Sequences*) qui permet d'exprimer et reconnaître des classes de séquences audiovisuelles (les canevas). Nous y détaillerons certains aspects de l'implémentation, dont la représentation des séquences itérées et les mécanismes de filtrage des contraintes temporelles impliquant de telles séquences. Enfin, le cinquième chapitre présente des expérimentations menées dans le cadre du projet européen DiVAN (*Distributed Audio-Visual Archives Network*), projet auquel nous avons pris part. Ces expérimentations portent sur plusieurs séries de journaux télévisés.

Chapitre 1
Indexation vidéo automatique

L'indexation automatique de documents pose de nombreux problèmes difficiles, par exemple dans les domaines de la reconnaissance des formes ou de la représentation des connaissances. Un système complet d'indexation vidéo est encore à construire. (Zhang *et al.*, 1997) posent quelques principes que devraient respecter les systèmes d'information multimédia en général, et, par conséquent, les systèmes d'indexation de documents audiovisuels. Ils soulignent en particulier l'aspect *non structuré* des ressources à traiter. Selon les auteurs, l'analyse structurelle ou grammaticale (*parsing*, le terme est difficile à traduire) est la première extension nécessaire qui doit être apportée à un système d'information traditionnel, d'une part pour le stockage (ou l'*insertion*) des données, d'autre part pour que les éléments de structure résultant de cette analyse puissent être exploités par le système de requêtes (*retrieval*). De plus, une représentation explicite de la structure du document permet de construire des interfaces de navigation (*browsing*) autorisant un parcours de lecture rapide. Le sens donné au terme *parsing* diffère ici légèrement du sens qui lui est généralement conféré dans le domaine linguistique, puisqu'il est rare qu'on puisse réellement parler de *grammaire*, au sens formel du terme, dans le domaine audiovisuel ou multimédia. Zhang *et al.* regroupent sous ce vocable deux opérations différentes : d'une part la segmentation temporelle du document à indexer, d'autre part une extraction de contenu (*content extraction*) des unités issues de la segmentation. Si, toujours selon eux, des algorithmes de segmentation efficaces sont aujourd'hui disponibles, le problème de l'extraction de contenu est un problème très difficile qui requiert à la fois des techniques d'analyse de signal complexes et des techniques de représentation de connaissances, problème tellement difficile qu'ils recommandent d'orienter les recherches dans ce domaine vers des outils d'aide à l'indexation qui s'appuieraient sur des primitives de bas niveau extraites automatiquement.

1 Un schéma général de station d'indexation

En nous inspirant de (Brunelli *et al.*, 1999) et (Ahanger & Little, 1996), nous illustrons par la figure 2 ce que pourrait être dans ses grandes parties une station d'indexation assistée par ordinateur. Le schéma représente les principales catégories de données (vidéo, données de catalogage, etc.) ainsi que les principales opérations d'indexation (extraction de primitives, interprétation et documentation) avec leurs entrées et leurs sorties. Un élément important est ce que nous appelons ici « modèle documentaire » ; ce modèle contient d'une part les paramètres d'analyse pour le module d'extraction de primitives, d'autre part des informations complémentaires pour le module d'interprétation, comme des informations sur la structure temporelle du document, sur le décor, l'habillage visuel ou les présentateurs de l'émission, et enfin des grilles d'indexation pour les documentalistes. Le module d'extraction de primitives réalise les tâches d'extraction de bas niveau nécessitant un accès à la vidéo : segmentation en plans, détection de logos, discrimination parole/non-parole, par exemple.

Le module d'interprétation est chargé de construire à partir des primitives extraites et des informations du modèle documentaire une représentation structurée du document à analyser la plus complète possible, comme par exemple une représentation arborescente des scènes, séquences et sous-séquences. Pour cela, le module d'interprétation peut avoir besoin de données d'analyse supplémentaires, comme la détection de logos dans une partie du document, auquel cas il peut piloter le module d'extraction de primitives. Le module d'interprétation n'a pas besoin d'accéder directement à la vidéo. Le module de documentation, ou station d'annotation, propose au documentaliste une grille de saisie contrainte par le modèle documentaire qui peut être remplie partiellement de manière automatique par les résultats d'interprétation et fournit un accès à la vidéo pour permettre une consultation directe. Le module de documentation n'a pas besoin d'avoir accès aux primitives extraites, qui sont généralement volumineuses et peu parlantes, mais seulement aux résultats d'interprétation.

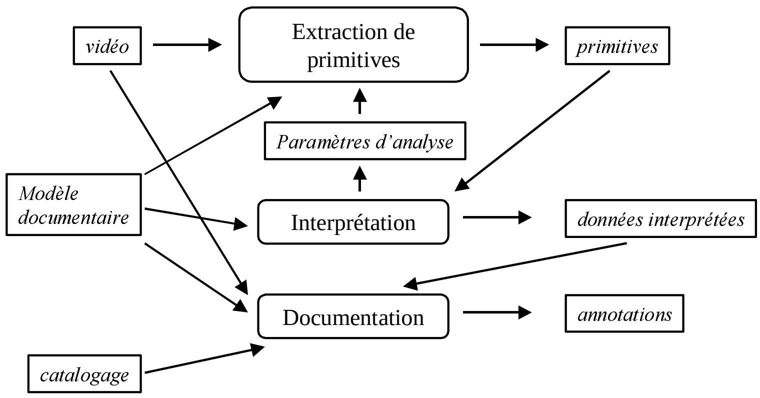


figure 2 : schéma général d'indexation

2 Segmentation en plans

Le problème de la segmentation en plans, souvent appelée jusqu'à assez récemment segmentation en scènes ou détection de scènes, est très largement traité et nous ne passerons pas en revue l'ensemble des techniques employées. (Rui *et al.*, 1999) distingue cinq types de méthodes selon qu'elles portent :

- directement sur une comparaison des pixels de chaque image (Zhang *et al.*, 1993; Hampapur *et al.*, 1994) ;
- sur des paramètres statistiques estimés de l'image (moyenne et écart-type de l'intensité lumineuse dans (Kasturi & Jain, 1991)) ;
- sur ce que les auteurs considèrent comme des transformations de l'image (*transform based techniques*) et qui sont des méthodes qui portent sur les données compressées du flux MPEG, que ce soit les coefficients DCT (Arman *et al.*, 1993) ou les vecteurs mouvements (Meng *et al.*, 1995; Yeo, 1996) ;

- sur des caractéristiques extraites de l'image. Les auteurs décrivent à ce sujet la technique proposée par (Zabih *et al.*, 1995) qui portent sur la comparaison de contours calculés sur chacune des images ;
- sur des comparaisons d'histogrammes, par exemple (Swanberg *et al.*, 1993; Zhang *et al.*, 1993).

Dans les cas les plus simples, c'est-à-dire quand il y a passage brusque d'un plan à un autre sans recouvrement d'image (les plans sont séparés par un *cut*), on peut espérer obtenir une segmentation en plan satisfaisante. Un grand nombre de méthodes de segmentation en plans sont décrites dans (Ahanger & Little, 1996; Brunelli *et al.*, 1999). D'autre part, (Brunelli *et al.*, 1999) rapportent des résultats intéressants de tests comparatifs. Il y apparaît que plusieurs méthodes affichent entre 95 et 99% de bonnes détections, pour des taux très variables de fausses alarmes. Les méthodes reposant sur des comparaisons d'histogrammes sont réputées pour être les plus robustes, notamment parce que les histogrammes de couleurs sont insensibles aux rotations et aux translations.

Lorsqu'il y a passage progressif d'un plan à un autre, on parle de « transition progressive » ou « transition graduelle » (*gradual transitions*). On parle également d' « effets d'édition » (*edit effects*), ces différentes catégories de transitions étant traditionnellement utilisées pour souligner ou ponctuer le discours cinématographique, comme l'emploi d'un fondu enchaîné lors d'un changement de séquence. Une transition entre deux plans consécutifs p_1 et p_2 est progressive 1) si p_1 se termine progressivement, si par exemple la luminosité globale diminue jusqu'au noir total (fondu vers le noir, ou au noir) ou 2) si p_2 commence progressivement. Dans de nombreux cas, les deux conditions sont vérifiées, soit qu'on observe un écran noir après la disparition progressive de p_1 et avant l'apparition progressive de p_2, soit que les deux plans se surimposent, p_2 apparaissant pendant que disparaît p_1 (fondu enchaîné). Ces catégories de transitions graduelles sont les différentes sortes de fondus, fondu du noir (*fade-in*), fondu vers le noir (*fade-out*), fondu enchaîné (*dissolve*), les différentes sortes de volets, que ce soit le passage d'une ligne horizontale ou verticale faisant apparaître le nouveau plan (*wipe*), comme une fenêtre s'ouvrant et en dévoilant une nouvelle, ou que ce soit le plan précédent qui « pousse » le plan suivant. D'autres types de transitions progressives existent, comme l'apparition du nouveau plan dans un cercle apparaissant à partir du centre de l'écran, ou une image incrustée semblant venir de derrière le présentateur du journal télévisé en effectuant quelques pirouettes avant de se stabiliser et d'occuper tout l'écran.

La présence de transitions graduelles rend beaucoup plus difficile la segmentation en plans. (Brunelli *et al.*, 1999) décrivent quelques méthodes de détections des transitions graduelles, et soulignent l'importance des hypothèses faites, notamment sur la stationnarité de l'image pendant la transition. Selon les auteurs, une voie de recherche importante devrait être l'intégration « synergique » de plusieurs techniques de segmentation.

Une fois le document découpé en plans, il est courant de représenter chaque plan par une ou plusieurs images clés (*keyframes*). Un petit nombre d'images clé permet de décrire de manière condensée certaines caractéristiques du plan telles que les couleurs dominantes et, de plus, permet de se faire une idée rapide, en les visualisant, du contenu effectif du plan, comme les personnages, les objets filmés ou le décor. En général, plus le plan est complexe, la complexité venant des mouvements de caméra ou des mouvements d'objets, plus le nombre d'images clé nécessaires à une bonne représentation est grand. Ainsi, un zoom peut être bien résumé par deux images clé, l'une au début du zoom et l'autre à la fin.

Selon (Zhang *et al.*, 1997), pour que les images clé soient vraiment représentatives du plan dont elles sont extraites, il faudrait que leur choix soit fondé sur des primitives « sémantiques », objets, actions ou événements. Malheureusement, les techniques d'analyse ne permettent pas de détecter dans l'image des éléments d'un aussi haut niveau d'abstraction, et il faut se contenter de primitives de plus bas niveau comme les histogrammes de couleurs, les mouvements de pixels dans l'image, les textures, etc.

3 Extraction de caractéristiques

Les techniques actuelles ne permettent pas d'extraire de l'image des informations de très haut niveau (Brunelli *et al.*, 1999). Cependant, certains éléments importants peuvent être détectés automatiquement, dans certaines conditions. Deux cas typiques sont les régions de l'image comportant des visages et les régions de l'image contenant du texte. Deux algorithmes spécialisés sont ainsi intégrés dans le prototype du système DiVAN que nous présentons dans le chapitre 5. (Brunelli *et al.*, 1999) recensent et comparent également plusieurs techniques permettant d'extraire des zones de visage et des zones de texte.

La présence de visages à l'écran, en particulier, peut fournir de précieuses informations sur le contenu d'un plan. Le nombre des visages, leur position et leur taille peuvent permettre dans certains cas de qualifier le plan assez précisément. Nous reviendrons sur cette question au chapitre suivant lorsqu'il sera question de classification des plans[2].

4 Analyse de l'audio

Tout comme l'analyse d'image, l'analyse du flux audio recouvre un grand nombre de disciplines et il serait difficile d'en faire une synthèse. Les progrès dans ce domaine intéressent bien évidemment ceux qui se préoccupent d'analyse de la vidéo. Les résultats les plus précieux devraient probablement provenir des techniques de reconnaissance de la parole car, dans bien des cas, une transcription automatique du texte alignée sur le flux audio peut fournir des informations extrêmement précieuses. Le projet Informedia (Watclar *et al.*, 1996)[3] est à ce titre exemplaire ; le projet Informedia débute à l'université de Carneggie Mellon, soutenu par le projet *Digital Library Initiative* fondé par la *NSF* (*National Science Foundation*) et se donne pour but d'intégrer des méthodes de reconnaissance de la parole, d'analyse d'image et de traitement de la langue naturelle pour la transcription, la segmentation et l'indexation de vidéo. Un effort particulièrement important a été mis dans ce projet sur la reconnaissance de la parole. Les techniques de reconnaissance de la parole indépendantes du locuteur, en mots connectés (c'est-à-dire sans que le locuteur soit obligé de hacher ses phrases) et à vocabulaire illimité ne sont pas encore parfaitement robustes et

---

[2] Voir « Classification de plans avec CLASSIC » page 42.

[3] Voir aussi http://www.informedia.cs.cmu.edu.

un certain nombre d'autres problèmes se posent pour la transcription de la bande son associée à la vidéo : voix fortement bruitées (reportages *live*), superposition de voix (inteviews), présence de musique, etc. Cependant, même imparfaite, la transcription peut être associée aux informations extraites de textes associés (scripts, conducteurs, sous-titres, légendes) et aux informations extraites de l'image (ruptures de plan, visages). Dans Informedia, la parole reconnue est alignée automatiquement avec les sous-titres pour consolider l'indexation et des limites de « paragraphes » sont marquées, indiquées par une faible énergie du signal ou par la ponctuation des sous-titres. Une des applications construites autour d'Informedia est un système de nouvelles à la demande (*News-On-Demand*) (Hauptmann & Witbrock, 1997) dans lequel un utilisateur peut faire une requête en langue naturelle. Les auteurs rapportent, dans différentes conditions, un taux d'erreur par mots (*word error rate*) qui totalise le nombre d'insertions, de substitutions et de suppressions de mots, qui peut dépasser 100% mais qui est considéré comme un bon indicateur. Ces conditions varient des conditions optimales de tests (8 à 12% d'erreurs) aux conditions les plus critiques (65% pour un journal télévisé d'une demi-heure, 75% pour un documentaire d'une heure et 85% pour une plage de publicités). Ces résultats, toujours d'après les auteurs, ne sont pas si accablants qu'il peut sembler : dans le système qu'ils proposent, la transcription elle-même n'est pas montrée à l'utilisateur et seule est utilisée l'information temporelle provenant de l'alignement avec les sous-titres. A cet égard, un faible taux de reconnaissance s'avère tout de même appréciable. Dans le même ordre d'idées, des expériences ont été menées au Lip6 (Laboratoire d'Informatique de Paris 6) sur l'alignement temporel du script et de la bande son. Dans (Faudemay, 1997), les auteurs rapportent une expérience effectuée sur l'alignement temporel script/bande-son de *Conte de Printemps* d'Eric Rohmer et d'un épisode de dessin animé[4], rapportent un alignement précis à la demi-seconde et proposent d'utiliser le script pour la segmentation en scènes. Il est indéniable que lorsque le script est disponible et qu'il est lui-même structuré, ce genre de techniques peut grandement faciliter la structuration de la vidéo.

Dans un contexte plus restreint, (Chang *et al.*, 1996) utilisent des techniques de reconnaissance de la parole et d'analyse de signal associées à des méthodes d'analyse d'image pour repérer les moments « forts » de retransmissions d'événements sportifs. Une première analyse cherche dans le commentaire audio un certain nombre de mots clé fixés à l'avance en appliquant une méthode de *word spotting* ; parallèlement, les acclamations de la foule, indiquant généralement un moment important du match, sont recherchées dans le signal audio par une méthode de classification relativement simple. Une fois ces éléments repérés, une méthode d'analyse d'image est appliquée au voisinage des points clé. Dans l'exemple présenté, le terme « touchdown » (touche, dans le football américain) ayant été reconnu dans la bande-son, le système recherche dans les images voisines du point auquel le terme a été reconnu une configuration de lignes similaires aux cadrages typiques représentant soit les équipes alignées face à face (deux lignes parallèles en diagonale de l'écran), soit les deux poteaux des buts. Pour cela, une méthode d'extraction de frontières utilisant le système de traitement d'image Khoros[5] est appliquée et les résultats sont comparés à un modèle géométrique. Si la similarité est reconnue, une décision finale est prise de l'occurrence de l'événement recherché. Dans cette étude, le champ d'application est volontairement limité à un domaine très spécifique (le football américain) ; les

[4] L'épisode « K » de la série *Ivanhoé* ; l'INA avait alors, d'assez loin, participé à l'expérience dans le cadre du projet européen Anim2000 de développement d'outils logiciels de pré-production de films d'animation.

[5] Voir http://www.khoral.com/khoros/

techniques proposées semblent peu généralisables en ce qui concerne l'intégration de résultats venant de l'analyse audio et de l'analyse d'images. Contrairement à (Li *et al.*, 1996) ou (Lienhart *et al.*, 1999) où des caractéristiques audio sont extraites des plans d'un document préalablement segmenté, l'idée est ici de pré-traiter le document par une recherche de mots clé dans le commentaire audio. Cette recherche préalable, ou, comme c'est plus ou moins le cas dans Informedia, une transcription préalable complète, est une approche originale qui pourrait se voir confortée dans l'avenir au fur et à mesure que les techniques de reconnaissance de la parole deviendront plus performantes et plus robustes.

Parallèlement à la reconnaissance de la parole, l'analyse auditive de scène par ordinateur (traduction approximative de *Computational Auditory Scene Analysis*) s'est développée comme activité de recherche à part entière. Les recherches dans ce domaine portent sur la compréhension de la parole, mais également sur l'extraction de caractéristiques du signal musical (rythme, mélodie, tempo, accords, par exemple) ou sur la séparation du flux selon les sources identifiées (réduction de l'effet *cocktail*, par exemple). Les techniques développées ne sont pas capables, aujourd'hui, d'extraire des informations de suffisamment haut niveau pour être directement exploitables pour la structuration ou l'indexation de vidéos, excepté dans des cas très spécifiques comme la recherche de coups de feu (Lienhart *et al.*, 1997) ou d'acclamations de foule (Chang *et al.*, 1996). Parmi ces méthodes, on peut distinguer celles qui cherchent à discriminer des segments de signal audio selon qu'ils contiennent ou non de la parole et/ou de la musique, ce type d'information pouvant être très utile pour la classification de segments vidéo. Le signal de parole étant bien modélisé, on peut espérer obtenir une bonne discrimination parole/non-parole ; quant à la musique en général, il n'est déjà pas toujours facile pour un auditeur humain d'identifier une série de sons comme étant de la musique et il nous semble qu'il faudrait soit se limiter aux musiques fortement harmoniques et rythmées, ce qui est très souvent le cas à la télévision, soit développer des classifieurs spécialisés pour différents types de musique, ce qui paraît une tâche difficile pour les résultats qu'on peut en espérer. Nous ne développerons pas plus avant cette présentation et l'on peut se reporter à (Rosenthal & Okuno, 1998) pour une sélection d'articles sur ce domaine. Quoi qu'il en soit, il est très largement admis que la piste audio est trop souvent négligée.

5 Macro-segmentation

Dans la plupart des cas, la structure temporelle d'un document audiovisuel respecte le schéma du cinéma traditionnel : un film est composé d'une suite de scènes qui sont elles-mêmes composées d'une suite de plans. Il est possible de définir formellement la notion de plan : un plan est une suite d'images enregistrées en une seule prise de caméra, un plan est une suite d'images qui n'a pas subit d'effet de montage, un plan est ce qui est enregistré sur la pellicule entre le moment où le réalisateur dit « moteur » et le moment où il dit « coupez », etc. Déterminer les limites temporelles d'un plan est par conséquent un problème généralement bien posé. Il est vrai que cette notion a tendance actuellement à être remise en question, en particulier à la télévision où les effets d'images incrustées, de duplex ou de bandes annonces défilantes partagent parfois l'écran en plusieurs régions, chacune de ces régions pouvant être considérée, du point de vue du monteur, comme temporellement indépendante. Cependant, si l'on exclut ces cas là, un plan est un plan. Définir la notion de scène est plus difficile et si l'on peut s'entendre sur une définition très générale comme « une collection de plans sémantiquement liés et temporellement adjacents » (Rui *et al.*, 1999), « subdivision d'une continuité audiovisuelle qui est définie, le plus souvent, par unité de lieu et de temps » (Lienhart *et al.*, 1999), définitions qui perpétuent la tradition théâtrale, cela ne permet pas toujours de déterminer où commencent et où finissent les scènes dans une vidéo, et encore moins de valider un tel découpage. Un découpage en scène est subjectif, comme l'est le découpage d'un texte en paragraphes. L'expérience montre cependant que certains segments seront systématiquement groupés dans la même scène : dialogues dans une scène montée en champ-contre champ, suite de plans ayant un fond musical commun par exemple (Lienhart *et al.*, 1999).

Tenter de définir des unités temporelles plus longues que le plan se justifie cependant, le découpage en plans étant insuffisant pour bien des applications, d'une part parce qu'il donne un trop grand nombre d'unités et d'autre part parce que chacune de ces unités est d'un degré d'abstraction (un « niveau sémantique ») trop faible. (Yeung *et al.*, 1996) rapportent ainsi 500 plans pour 15 minutes du film *Terminator 2* et le découpage d'un épisode de *Chapeau melon et bottes de cuir de 50 minutes* réalisé manuellement fournit quelque 1200 plans. La *macro-segmentation* d'un document consiste à extraire du flux vidéo de tels segments. Le terme de « scène » (*scene*), bien qu'usité, est fréquemment évité pour se voir préférer des expressions plus neutres : *sequence* dans de nombreux cas, ou encore *story unit* (Yeung *et al.*, 1996), *logical story unit* (Hanjalic *et al.*, 1999), *video paragraph* (Watclar *et al.*, 1996), *act* (Aoki *et al.*, 1996) par exemple. Nous présentons dans les sections suivantes quelques méthodes de macro-segmentation.

5.1 Méthodes de regroupement

Une première famille de méthodes, et peut-être la seule qui présente une certaine homogénéité, réunit des algorithmes de regroupement (*clustering*) de plans qui reposent sur une mesure de similarité entre plans. Le principe général est de regrouper en classes homogènes les plans qui « se ressemblent », c'est-à-dire qui partagent certaines caractéristiques, comme les décors, les personnages filmés, les dialogues (Lienhart *et al.*, 1999), ou, comme c'est souvent le cas, les plans qui ont des caractéristiques visuelles proches, une mesure de similarité entre plans étant généralement définie sur les images clé représentatives des plans. Cette approche repose sur l'observation que des plans dont le sujet ou le décor sont différents ont souvent des caractéristiques visuelles différentes, et que, par conséquent, en regroupant les plans visuellement proches, on a de bonnes chances de regrouper des plans ayant le même sujet ou le même décor. L'hypothèse de fond n'est pas très éloignée de celle qui sous-tend la plupart des algorithmes de segmentation en plan : une analyse de critères locaux de bas-niveau permet d'extraire des objets de plus haut niveau.

Le principal intérêt qu'on peut trouver à regrouper les plans en classes est de permettre une navigation rapide dans le document en présentant à l'utilisateur une vision condensée de chaque classe (en général une icône choisie parmi l'ensemble des images clé). Si la classification est hiérarchique, l'utilisateur est supposé pouvoir raffiner graduellement sa recherche jusqu'à obtenir l'information cherchée. D'autre part, les résultats de regroupement peuvent être utilisés pour effectuer une détection des transitions de scènes, comme c'est le cas dans (Yeung *et al.*, 1996; Hanjalic *et al.*, 1999; Rui *et al.*, 1999). Nous présentons ici quelques-unes de ces méthodes de regroupement de plans. Elles font appel à des techniques connues de classification automatique qui visent à obtenir soit une *partition* de l'ensemble des plans en classes, soit une *hiérarchie* de classes imbriquées, soit un arbre de classes dont les feuilles seront les plans. Pour plus de détail sur les principales techniques de classification automatique, on pourra se reporter à (Celeux *et al.*, 1989).

Un exemple typique est présenté dans (Zhong *et al.*, 1996). Une segmentation initiale du document fournit un ensemble de plans ainsi qu'un ensemble d'images clé associées. Une distance entre plans est définie sur des attributs calculés des images clé (histogrammes de couleur et mouvements dominants). La méthode de classification employée est une version étendue de la méthode des K-moyennes (*fuzzy K-means*) permettant de considérer des frontières de classes floues, et, ainsi, de pouvoir classer un élément proche d'une frontière dans plusieurs classes simultanément. La méthode est employée récursivement sur les classes obtenues, et permet d'obtenir une hiérarchie de classes. Dans le même esprit, Aoki *et al.* (Aoki *et al.*, 1996) cherchent à détecter des plans similaires montés en alternance de telle sorte qu'on puisse supposer l'existence d'un « dialogue » (*dialog*, le terme désignant vraisemblablement un champ-contre champ non strict).

La méthode de regroupement de plans la plus connue est sans doute la méthode du Graphe des Transitions de Scènes (*Scene Transition Graph,* ou *STG*) (Yeung & Liu, 1995; Yeo, 1996; Yeung *et al.*, 1996; Yeung & Yeo, 1997), dont beaucoup d'autres méthodes sont inspirées. Cette méthode repose sur une technique de regroupement hiérarchique et tente schématiquement de regrouper en une classe (*cluster*) les plans ayant été tournés depuis la même position de caméra en se fondant sur des caractéristiques visuelles des plans, en faisant l'hypothèse qu'à chaque position de caméra correspond des caractéristiques visuelles propres. Les scènes sont alors calculées d'après les classes obtenues.

Dans un premier temps, une segmentation du plan est effectuée, chaque plan étant associé à un ensemble d'images clé, le calcul des images clé cherchant à rendre compte des mouvements de caméra élémentaires à l'intérieur du plan. A partir d'une mesure de dissimilarité $D(f_i,f_j)$ entre les images f_i et f_j, calculée sur les histogrammes de couleur des images réduites, les auteurs définissent une mesure de similarité entre plans comme la dissimilarité minimum calculée sur le produit croisé des images de chaque plan. En pratique, pour chaque plan, seules les images clé sont prises en compte.

Pour regrouper les plans en scènes, la mesure de similarité entre plans est utilisée par une méthode itérative de regroupement hiérarchique de classes de plans, qui, à chaque étape, regroupe en une seule classe les deux classes ayant le plus de chance de correspondre à une même position de caméra. Initialement, chaque plan définit sa propre classe. A chaque itération, les deux classes les plus proches (les moins dissimilaires) sont regroupées en une nouvelle classe, à condition que la dissimilarité soit inférieure à un certain seuil δ. La mesure de similarité entre classes est définie comme le maximum de dissimilarité calculé sur le produit croisé des plans de chaque classe. On obtient alors un arbre hiérarchique binaire (un *dendogramme*) de classes dont les feuilles sont les plans. Une classe de la hiérarchie n'est pas nécessairement temporellement connexe, et pourrait, à ce stade, regrouper deux plans « similaires » utilisés dans deux scènes différentes, ce qui peut arriver en particulier lorsque des scènes similaires se répètent. Pour éviter ce problème, un seuillage temporel est effectué : la mesure de dissimilarité est considérée comme infinie si deux plans sont éloignés temporellement de plus d'une certaine valeur T, ou, autrement dit, si deux plans n'appartiennent pas à la même fenêtre temporelle de longueur T.

Cette hiérarchie est finalement utilisée pour détecter les transitions de séquences. Un graphe orienté de transitions est construit à partir de la hiérarchie des classes de plans. Les nœuds de ce graphe sont les classes de la hiérarchie, et un arc entre les nœuds correspondant aux classes C_i et C_j existe s'il existe au moins un plan de C_i qui précède immédiatement un plan de C_j. Une première utilisation de ce graphe permet une navigation hiérarchique dans le document. Les arêtes correspondant à une transition de scènes sont les *arêtes de coupure* du graphe, c'est-à-dire les arêtes dont la suppression transformerait le graphe non orienté associé au graphe de transitions en deux graphes disjoints. En d'autres termes, les scènes sont les composantes fortement connexes du graphe de transitions.

Les principaux reproches adressés à cette méthode concernent d'une part le choix du seuil permettant de choisir les « bonnes » classes dans la hiérarchie pour construire le graphe de transitions, et d'autre part l'utilisation d'une fenêtre temporelle rectangulaire. Pour éviter le problème du fenêtrage, (Rui *et al.*, 1999) remplacent cette fenêtre par une fonction d'« attraction » entre plans, fonction décroissant avec la distance temporelle et croissant avec la similarité visuelle. D'autre part, la méthode des *STG* se fait en deux passes (calcul de la hiérarchie de classes puis calcul des transitions de scènes) ce qui impose de disposer de la globalité du document avant de pouvoir fournir des résultats ; dans les situations qui nous intéressent où la segmentation peut se faire hors ligne avant un stockage en base de données, cela n'est cependant pas une restriction.

Kender et Yeo (Kender & Yeo, 1998) critiquent la méthode des *STG* et proposent de l'améliorer en utilisant une approche originale. Ils fondent leur critique sur deux constats. Le premier constat est la méthode des *STG* est fondamentalement discrète, et même binaire : deux plans sont ou ne sont pas similaires puisqu'ils appartiennent ou n'appartiennent pas à la même classe (*cluster*), d'autre part, les longueurs relatives des plans ainsi que leur éloignement temporel ne sont pas pris en compte, et, par conséquent, la décision de marquer une rupture de plans comme une rupture de scènes est à son tour binaire. Le second constat est que la méthode est sensible au bruit et aux fluctuations de classification des plans ; ainsi, supposons une série de plans *ABABCDCD*[6] dont le quatrième plan (classe *B*) serait visuellement éloigné du premier et classé dans une autre classe *X*, donnant la série *ABAXCDCD*. La méthode des *STG* donne dans ce cas trois scènes : *ABA*, *X*, et *CDCD*. L'idée est alors de modéliser un processus continu de mémoire visuelle, dans lequel la perception d'un nouveau plan « rappelle » (*recall*) des plans passés, l'intensité du rappel – du souvenir – variant avec l'éloignement temporel, la longueur et la similitude visuelle des plans. Cette modélisation permet de définir une notion de « cohérence » (*coherence*) qui indique pour un nouveau plan son degré de cohérence avec les plans passés ; à chaque transition de plan, ce degré de cohérence est calculé, qui est approximativement la somme du rappel de tous les plans passés par tous les plans à venir. Un faible degré de cohérence indique une rupture de scène. La mise en œuvre de cette idée simple et élégante est complexe et repose sur l'utilisation d'un tampon virtuel supposé suffisamment grand pour pouvoir appliquer des formules d'intégration dans un domaine continu, et sur une utilisation classique d'une mesure de similarité entre images. Il nous semble que, même si la mise en œuvre proposée n'est pas absolument convaincante, l'idée d'intégrer diverses mesures (longueurs des plans, distance temporelle, distance visuelle) dans une modélisation d'un processus de mémorisation est à retenir.

[6] Les lettres capitales représentent les classes des plans.

Partant du même constat que, dans la méthode proposée par (Yeung & Yeo, 1997), l'étiquetage en classes des plans est trop simpliste et que le choix d'une fenêtre temporelle ne rend pas bien compte de l'éloignement temporel des plans, (Hanjalic *et al.*, 1999) proposent une autre méthode de regroupement des plans en unités narratives, appelées *Logical Story Units (LSU).* La notion d'unité est centrée sur celle d'événement (*event*), un événement pouvant être un dialogue, une scène d'action, ou n'importe quel élément dramatique. Un plan est soit un plan associé à un événement (*event shot*), soit un plan descriptif venant compléter la narration (*descriptive shot*). L'hypothèse est faite qu'un film est constitué d' « épisodes » (c'est-à-dire en réalité de scènes au sens classique) se rapportant à un ou plusieurs événements, les plans se rapportant à un événement d'un épisode pouvant être imbriqués fortement au sein de l'épisode avec les plans se rapportant à un autre événement, de manière plus ou moins régulière (régularité forte dans le cas du champ-contre champ strict, par exemple). Les auteurs soulignent qu'une segmentation en épisodes n'est possible qu'en possession du script ; le calcul des *LSU* tente cependant d'approximer ces épisodes. Une hypothèse de cohérence temporelle globale du document est faite selon laquelle les plans concernant le même événement ne se succèdent pas nécessairement mais doivent probablement apparaître dans un certain intervalle de temps. De plus, un événement se rapporte à un lieu et/ou à un ou plusieurs personnages, les plans liés au même événement étant supposés présenter des similitudes visuelles.

Les auteurs proposent un algorithme pour déterminer les bornes de *LSU*s selon l'organisation des plans autour des événements. Un lien est créé entre le plan numéro k et le plan numéro p ($k<p$) si k et p se rapportent au même événement, c'est-à-dire si une certaine similarité visuelle est observée entre les plans k et p. Pour calculer cette mesure de similarité, toutes les images clé représentatives d'un plan sont composées en une seule image, chaque image constituant un *bloc*. Pour mesurer la similarité entre deux plans, on cherche un appariement entre les blocs des deux images. Ce problème étant très combinatoire, une méthode approchée est utilisée, méthode qui utilise pour chaque bloc sa couleur moyenne dans le domaine L^*u^*v. De plus, pour savoir si un lien peut être créé entre deux plans, un seuil adaptatif tenant compte de l'éloignement temporel des plans et d'une mesure globale de l'incohérence du *LSU* courant est utilisé.

Pour déterminer les bornes d'un *LSU*, les heuristiques suivantes sont employées : i) deux plans liés au même événement appartiennent au même *LSU*, ii) tous les plans compris entre deux plans liés au même événement appartiennent également au même *LSU*, iii) si le plan k ne peut être lié à aucun plan p, et si aucun des r (r est fixé) plans précédents ne peut l'être, alors k est le dernier plan d'un *LSU*. Cette méthode peut laisser des plans non rattachés à un *LSU*, ce qui, finalement, peut laisser des *LSU* dont les bornes sont indéterminées.

Cette méthode, si elle ne diffère pas fondamentalement dans sa mise en œuvre de la méthode de (Yeung & Yeo, 1997), nous semble cependant intéressante par le fait qu'elle tente de mettre en avant le fait que des plans qui se rapportent au même « événement » doivent se situer dans une même région temporelle, tout en évitant un fenêtrage trop brutal. Il nous semble, en revanche, que déterminer que deux plans se rapportent au même événement parce qu'ils ont des caractéristiques visuelles globales similaires est un peu rapide ; il serait sans doute possible d'envisager un regroupement en événements fondé sur un plus grand nombre de caractéristiques.

Dans (Hammoud & Kouam, 2000), Hammoud *et al.* utilisent également une méthode de regroupement de plans en classe selon des critères de similarité visuelle. Les classes sont alors organisées en un graphe temps-espace dont les nœuds sont les classes et dont les arcs relient les classes qui respectent l'une des quatre relations temporelles symboliques *meets*, *before*, *during* et *overlaps*, qui sont quatre relations choisies parmi les treize relations temporelles élémentaires mises en évidence par Allen. A chacune de ces relations temporelles est associée une sémantique propre. Ainsi, la relation *during* indique-t-elle qu'une classe de plans est une sous-séquence d'une séquence englobante ; les relations *meets* et *before* permettent de déterminer les limites de séquence. Le raisonnement temporel symbolique est ici certes limité, mais c'est à notre connaissance le seul exemple de son utilisation pour la macro-segmentation.

Une des caractéristiques communes des méthodes présentées ci-dessus est qu'elles cherchent à regrouper en unités narratives des plans provenant d'une segmentation préalable. L'intérêt, outre le fait que ce sont des méthodes généralistes applicables dans un grand nombre de cas sans connaissance *a priori* sur le document à traiter, réside en particulier dans le fait qu'elle sont généralisables à des critères de similarité plus complexes que ceux présentés, pouvant tenir compte des textures, des frontières d'objets, etc. La limitation se trouve sans doute dans l'utilisation d'une ligne temporelle unique de segments consécutifs – la liste des plans – ce qui interdit d'utiliser des segments provenant de segmentations différentes, et en particulier de segments provenant de la bande son.

5.2 Utilisation de connaissances structurelles

Nous présentons ici des méthodes de segmentation en séquences qui utilisent des connaissances disponibles *a priori* sur la structure des documents à traiter. Nous commencerons par présenter la méthode proposée dans (Aigrain *et al.*, 1997), dont il peut par ailleurs être discuté qu'elle entre dans cette catégorie, puis nous nous intéresserons à un type de documents très largement étudiés en raison leur structure souvent simple et régulière : les journaux télévisés.

5.2.1 Utilisation de règles de montage

La méthode de macro-segmentation par l'application de règles de production présentée dans (Aigrain *et al.*, 1997) est souvent considérée comme une méthode générale ne modélisant pas de connaissances *a priori* sur la structure du document à traiter. (Rui *et al.*, 1999), par exemple, distingue deux types de méthodes de segmentation en scènes, les méthodes générales (*general purpose approaches*) et les méthodes utilisant un « modèle » (*model-based approaches*) du document à segmenter, et range l'approche d'Aigrain *et al.* dans la première catégorie. Il est vrai que le modèle sous-jacent n'est pas aussi structuré et spécifique que peut l'être une retransmission d'un match de football, comme c'est le cas dans (Gong *et al.*, 1995), mais il nous semble cependant que les règles choisies ont pour but d'exploiter des connaissances portant sur la structure d'un certain genre de documents audiovisuels, à savoir le film de fiction traditionnel de facture hollywoodienne, et en particulier certaines règles de montage. Cette manière de filmer est devenue une norme de fait, en particulier à la télévision. Les règles proposées sont donc applicables dans un grand nombre de cas. Cependant, à nos yeux, un modèle explicite est employé. C'est peut-être cette ambiguïté qui est sous-entendue dans le terme « *medium knowledge* » employé dans le titre de l'article.

Le principe développé par Aigrain *et al.* est d'exprimer de manière déclarative sous la forme de règles de production, et en se fondant sur des primitives extraites du flux vidéo, des formes (*patterns*) typiques indiquant, par exemple, que dans certaines conditions un fondu enchaîné indique une transition entre deux séquences, que deux plans visuellement proches mais non temporellement adjacents font partie de la même séquence, etc. Les règles reposent sur un nombre relativement petit de primitives (segmentation en plans, similarité globale d'images, détection des transitions graduelles et détection de segments musicaux), et ont pour but de détecter trois types d'éléments :

a) Les limites de séquences ;
b) Les plans « importants » ;
c) Les groupes de plans faisant partie de la même séquence.

Neuf règles sont ainsi décrites, dont les trois premières, illustrées par la figure 3, examinent la succession des transitions sèches (*cuts*) et des transitions graduelles :

Si au moins trois transitions de type « cut » se succèdent, suivies d'une transition progressive, elle-même suivie d'au moins trois cuts, alors il y a probablement une rupture de séquence au moment de la transition progressive. Si cette transition est un fondu à l'ouverture (*fade out*) ou un fondu au noir, la transition marque certainement une rupture de séquence ; sinon, la transition peut également indiquer le début d'un plan important.

Si l'on observe une suite d'au moins trois cuts suivis d'exactement deux transitions progressives puis d'au moins trois cuts, alors le plan séparé par les deux transitions progressives est un plan important.

Si l'on observe une suite d'au moins trois cuts suivis d'au moins trois transitions progressives suivies d'au moins trois cuts, alors une séquence commence au plus deux plans avant la première transition progressive et se termine sur la dernière transition progressive.

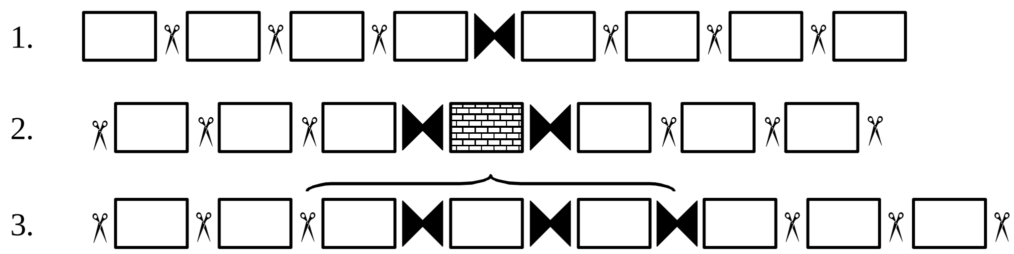


figure 3 : trois règles tirées de (Aigrain *et al.*, 1997)

Les règles 4 et 5 traitent de la similarité visuelle des plans. La règle 4 indique que deux plans similaires voisins mais non adjacents appartiennent à la même séquence, alors que la règle 5 indique que deux plans similaires adjacents marquent une rupture de séquence. La règle 6 s'appuie sur des comparaisons de couleurs et indique que si plus de trois plans similaires quant aux couleurs dominantes se succèdent, ils appartiennent à la même séquence. La règle 7 utilise un calcul de prédiction de la durée des plans pour modéliser le rythme du montage, et indique qu'un plan dont la durée est inattendue indique une rupture de séquence. La règle 8 utilise la détection de segments musicaux et indique que l'apparition de musique après au moins trente secondes de silence marque un début de séquence. La règle 9 utilise une estimation des mouvements élémentaires de caméra (zooms, panoramiques), et regroupe en séquence les successions de plus de trois plans ayant le même mouvement de caméra.

Une fois obtenus les ensembles de ruptures de séquence, de fragments de séquences et de plans importants, une série de règles de plus haut niveau s'applique pour choisir définitivement la position des séquences, d'une part en recoupant les informations obtenues, d'autre part en résolvant les conflits en appliquant une plus grande priorité aux règles considérées comme les plus fiables.

S'il est toujours possible de discuter de la validité des règles proposées – en particulier, la règle 2 nous paraît contestable –, la démarche nous semble particulièrement intéressante sur plusieurs points. Tout d'abord, il est ici question de modéliser des connaissances de haut niveau de manière relativement déclarative, une certaine déclarativité nous semblant souhaitable pour les avantages qu'on lui reconnaît habituellement, lisibilité – modularité, maintenabilité, possibilité d'enrichissement. D'autre part, les données en entrée du système sont issues ou supposées être issues d'une analyse automatique, le rôle du système étant d'interpréter ces primitives pour proposer une structuration de plus haut niveau ; il s'agit ici du passage du numérique au symbolique. C'est dans cette optique que nous avons voulu situer notre travail.

5.2.2 Indexation de journaux télévisés

Le journal télévisé a fait l'objet d'un grand nombre de travaux dans le domaine de l'indexation vidéo automatique. Les grands diffuseurs (CNN, ABC News, bien sûr, mais également nos chaînes nationales[7]) diffusent maintenant l'actualité en ligne, par l'intermédiaire d'extraits thématiques ou de la présentation de la dernière édition du bulletin d'informations. Les besoins en termes de facilité d'accès, de possibilité de recherche par le contenu et de navigation sont évidents, ne serait-ce que pour les journalistes eux-mêmes, et il est de ce point de vue compréhensible que le « JT » ait fait l'objet d'attentions particulières. D'un autre côté, la recherche en indexation vidéo trouve là un champ d'investigation bien délimité, formalisable et riche, et les possibilités d'application de méthodes d'indexation qu'on peut espérer fiables et robustes sont nombreuses : journaux à la demande (*News on Demand*), classement thématique, résumés automatiques, etc. Le journal télévisé est un genre en soi, et s'il est discutable de parler de « grammaire du film » en général, les aspects syntaxiques des journaux télévisés sont en général frappants. Ainsi, par exemple, le *6 minutes* de la chaîne M6 est constitué d'une série de reportages séparés par des « titres », un titre étant constitué d'une image fixe où figure en gros plan le logo de la chaîne et où apparaît le titre du reportage, le même jingle sonore étant entendu pendant l'affichage de l'écran titre. Cette structure particulièrement simple peut s'exprimer par l'expression régulière suivante :

générique-début (*reportage titre*)+ *reportage générique-fin*

[7] Voir les sites de France 2 : http://www.france2.fr/video/videofrm.htm,
ou France 3 : http://www.france3.fr/fr3/video/newvid.html

où *générique-début*, *générique-fin* et *titre* sont des « unités syntaxiques » facilement identifiables. La structure n'est bien sûr pas toujours aussi simple et régulière, et d'autres éléments peuvent intervenir : présence d'un ou plusieurs présentateurs, d'invités, de séquences plateaux (intervention d'un journaliste spécialisé ou interview d'un invité), de rubriques spécialisées (météo, rubriques sportives ou économiques) ainsi que l'apparition à l'écran d'informations complémentaires (sous-titres, légendes, graphiques, cartes géographiques). La plupart du temps, cependant, chacun de ces éléments est identifiable et appartient à un ensemble fixe propre à une collection. Nous présentons ici quelques travaux réalisés sur l'analyse automatique de journaux télévisés.

Le système *Broadcast News Navigator* (Maybury & Merlino, 1997; Maybury *et al.*, 1997; Merlino *et al.*, 1997) développé par le laboratoire *Multimedia Computing* de la MITRE *Corporation*[8] est un système d'archivage et de consultation dédié aux journaux télévisés dans lequel sont implémentées un certain nombre de techniques d'indexation automatique. Le but de ces techniques est de segmenter les journaux en séquences appelées « *stories* » ou « *story segments* » ; la notion de « story » n'est pas très clairement définie[9] mais semble se rapporter à ce que les documentalistes appellent un *sujet*. Le principe général de la segmentation est de détecter par une analyse automatique des sous-titres, de l'image et du son, des éléments permettant de reconnaître les états d'un automate d'états finis (*FSA*) représentant la structure du journal (Merlino *et al.*, 1997). Certains éléments sont extraits automatiquement des sous-titres, comme, par exemple, les noms des journalistes qui se trouvent dans des phrases typiques telles que : « Hello, I'm <*person*> », les termes se rapportant à la météo, certaines transitions comme les transitions du présentateur vers un journaliste intervenant (« <*person*> reports »), ainsi que certains signes conventionnels tels que « >> » qui indique que le locuteur principal a changé ou « >>> » qui indique que le sujet principal a changé. De l'audio, le système se contente d'extraire les plages de silence de plus de sept secondes. De l'image, il détecte les plans noirs, les logos, les plans montrant un unique présentateur et les plans montrant deux présentateurs. L'automate donné en exemple (journal de *CNN*) comporte vingt-deux états définis en fonction des primitives extraites, tels que *start-of-broadcast*, *start-of-story*, *advertising*, *possible-end* ou *very-possible-end*. A partir d'un état donné, par exemple *very-near-start* où le logo de *CNN* commence à apparaître, l'observation de nouvelles primitives fait passer dans un autre état ; ainsi, on passe de *very-near-start* à *start-of-highligts* par l'observation d'un écran noir. Cette démarche se rapproche de celle que nous présentons en ce qu'elle propose un modèle explicite – ici un automate d'états finis – dont les éléments de base sont une abstraction des éléments reconnus automatiquement dans le flux vidéo par des méthodes numériques.

Ide *et al.* (Ide *et al.*, 1998a; Ide & Tanaka, ; Ide *et al.*, 1998b) utilisent les textes des légendes[10] incrustées pour l'indexation de journaux de la télévision japonaise. Ils proposent une classification des légendes en huit classes :

a) Légendes indiquant le lieu ou l'organisation ;
b) Légendes indiquant une personne ;
c) Titre ;
d) Parole (résumé, traduction, etc.) ;

[8] Société d'intérêt public à but non lucratif ; voir http://www.mitre .com
[9] "distinct portions of a news broadcast where one story is discussed".
[10] Nous traduisons par « légendes » le terme *captions* qui s'oppose dans l'article à *closed-captions* qui désignent les sous-titres.

e) Légendes d'ordre temporel (dates, par exemple) ;
f) Légendes techniques ;
g) Légendes descriptives ;
h) Autres.

Les légendes des trois premiers types peuvent directement fournir des mots-clé descriptifs des plans. Dans (Ide & Tanaka,), les textes des légendes sont transcrits manuellement, mais pourraient être obtenus par des méthodes de reconnaissance de caractères, comme c'est le cas dans (Sato *et al.*, 1999) où Sato *et al.* s'intéressent également aux légendes textuelles incrustées dans les journaux télévisés. Le type de la légende est déterminé en observant certaines caractéristiques grammaticales du texte : phrase à la syntaxe simplifiée, omission du sujet, terminaisons propres au japonais indiquant une certaine modalité (estime, conjecture, etc.). D'autre part, une analyse sémantique du texte permet de discriminer les noms apparaissant dans les légendes comme nom de personne, nom de lieu ou nom commun. Dans (Ide *et al.*, 1998b), Ide *et al.* utilisent ces mêmes techniques d'analyse du texte des légendes associées à des techniques d'analyse d'image permettant de classifier les plans d'un journal télévisé en cinq classes :

a) Parole/Reportage ;
b) Présentateur ;
c) Marche ;
d) Assemblée ;
e) Effets graphiques.

La détection de visages fournit des indications permettant une première classification, selon le nombre de visages détectés (plus de deux visages indiquent une assemblée, par exemple[11]), selon que le visage montre un mouvement particulier indiquant une personne en train de marcher, etc. Une analyse particulière cherchant à détecter un mouvement des lèvres permet de rejeter les insertions d'images fixes (portraits, par exemple). Pour discriminer les plans du présentateur des plans parole/reportage, l'ensemble des plans parole/reportage est « clusterisé » selon des caractéristiques visuelles globales (histogrammes de couleur), et la classe la plus grande et la plus dense est considérée comme la classe des plans du présentateur. Les résultats de l'analyse du texte des légendes et de la classification des plans sont ensuite intégrés dans une phase de regroupement qui permet par exemple d'associer un nom de personne à un visage. Un journal télévisé est ici considéré comme ayant une structure temporelle fixe : un plan du présentateur introduit une série de reportage appelée « sujet » (*topic*). Il est alors facile d'effectuer une analyse structurelle du journal par une segmentation en sujets. Les règles de classification, comme l'association d'un nom propre et d'un visage, sont ici données manuellement. (Ide *et al.*, 1998a) suggèrent une méthode supervisée d'apprentissage de ce type de règles.

Dans (Satoh *et al.*, 1997), Satoh *et al.* proposent *Name-It*, un système qui tâche d'associer les noms extraits de transcriptions de journaux télévisés avec des visages extraits automatiquement de l'image, pour pouvoir, par exemple, retrouver dans une base de données vidéo des extraits concernant telle ou telle personnalité ; associer visages et noms peut ainsi permettre de retrouver des extraits où sont visibles mais non mentionnées des personnalités dont le nom apparaît dans d'autres extraits. Pour réaliser cette tâche, le

---

[11] Remarquons que pour une image de foule où les visages sont nombreux et petits il est fort probable qu'un algorithme ne détecterait aucun visage.

système extrait de la transcription des noms candidats, et, pour déterminer si un nom a une chance de correspondre à un visage apparaissant à l'image dans le même laps de temps, associe à chaque candidat un score en fonction de plusieurs heuristiques : le locuteur, surtout si c'est le présentateur, a peu de chances de mentionner son propre nom ; le candidat doit être l'agent d'une action dans la phrase ; il a un grand nombre de chances d'être cité avant les autres noms, pour une image donnée ; il a des chances d'être cité juste avant l'apparition de l'image. En fonction des scores obtenus, le système prend finalement une décision quant à l'association nom/visage.

5.2.3 Description de modèles de documents pour l'indexation

Les journaux télévisés, ainsi que d'autres types d'émissions, comme beaucoup de magazines de variétés ou jeux télévisés, respectent un modèle strict. Dans le cas du journal télévisé ou de retransmissions sportives, ces modèles ont été utilisés pour guider l'analyse automatique. Cependant, dans la plupart des cas, sauf peut-être pour le système *Broadcast News Navigation* (Merlino *et al.*, 1997), le modèle n'est pas donné explicitement et il est par conséquent difficile d'appliquer les méthodes d'analyse à d'autres modèles de documents. Nous pensons qu'il est important, lorsque ce modèle existe, que les algorithmes d'analyse audio et/ou vidéo soient encapsulés dans un système raisonnant à un plus haut niveau d'abstraction et possédant une représentation explicite du modèle, d'une part pour des raisons simples de modularité, en donnant par exemple la possibilité de brancher ou débrancher des algorithmes selon les besoins ou de les paramétrer en fonction d'informations données par le modèle (logos, jingles), et d'autre part pour interpréter les résultats fournis par les algorithmes en exploitant les connaissances structurelles fournies par le modèle. Le but du travail que nous présentons ici est d'une part de définir un langage permettant de représenter la structure temporelle des séquences typiques repérables dans une émission respectant un format donné (par exemple la séquence météo du journal télévisé d'une chaîne donnée) ainsi que les données d'analyse (par exemple les différents logos), et d'autre part de mettre en œuvre un processus de reconnaissance permettant d'identifier à partir des données d'analyse et du modèle d'émission des occurrences de ces séquences types dans des documents audiovisuels dont le format est connu.

6 Représentation des informations extraites de la vidéo

La description de documents audiovisuels à des fins d'archivage et de réutilisation implique plusieurs métiers : bases de données, analyse automatique, annotation et documentation. Il est donc nécessaire de pouvoir stocker, rechercher et échanger les données de description. Le schéma traditionnel des bases de données relationnelles est mal adapté à la représentation des données de description de la vidéo, en particulier les informations spatiales et temporelles. La recherche d'un formalisme de description de documents multimédias et audiovisuels est une préoccupation qui concerne aussi bien les industriels que les universitaires. Ainsi, par exemple, la société Informix[12] fournit-elle un système de gestion de bases de données intégrant des fonctionnalités spécifiques au multimédia, telles que la recherche d'informations « time-codées » au format SMPTE. (Aguierre Smith &

[12] http://www.informix.com/media

Davenport, 1992) ont proposé un modèle de *strates* permettant d'annoter un même document selon plusieurs points de vue différents et à différents niveaux de granularité. Ce modèle a connu un grand succès et a servi de base à divers travaux (voir par exemple (Prié *et al.*, 1998)). Le projet VANE (Carrer *et al.*, 1997) de station d'annotation semi-automatique de vidéo du groupe multimédia de l'université de Boston propose un modèle de représentation fondé sur SGML, avec la préoccupation de pouvoir exprimer les données bibliographiques, structurelles et de contenu (voir également (Rowe *et al.*, 1994)). Chacun de ces types de données est à son tour subdivisé, les informations structurelles se décomposant par exemple en deux catégories, les informations sur le support (*e.g.* le taux de compression) et celles concernant les aspects filmiques (*e.g.* les mouvements de caméra). Notons que les auteurs considèrent qu'une décomposition temporelle en trois niveaux (*layers*) : séquences, scènes et plans, est suffisante pour la plupart des applications.

Dans les domaines de l'hypermédia et du multimédia les travaux sont également nombreux sur les modes de représentation, la vidéo n'étant qu'un des types de documents à représenter. Ainsi, le projet MAVIS (*Microcosm Architecture for Video, Image and Sound*) (Lewis *et al.*, 1996) du groupe de recherche multimédia de l'université de Southampton cherche à étendre les hyperliens du système Microcosm (Fountains *et al.*, 1990) à différents types de média, en donnant par exemple la possibilité de créer un lien entre une région d'une image et une autre région d'une autre image. Dans le contexte du système d'édition multimédia Madeus (Jourdan *et al.*, 1998), (Roisin *et al.*, 1999) proposent un format de description hiérarchique de la vidéo reposant sur XML afin de pouvoir référencer des segments de vidéo dans un document multimédia. Notons que XML tend à s'imposer comme standard de fait pour le choix d'un langage textuel à partir duquel sont construits divers langages de description. Dans les sections suivantes, nous présentons rapidement MPEG-7, standard de description du contenu audiovisuel et multimédia en cours d'élaboration, puis le format d'échange de données de description A.E.D.I défini à l'INA.

6.1 MPEG-7

Après avoir standardisé des formats de compression pour la vidéo (MPEG-1 et MPEG-2) et un format de représentation d'objets multimédia (MPEG-4, dont la première version a été publiée en juillet 1998), le groupe MPEG (*Moving Picture Expert Group*) travaille à l'élaboration d'un standard de description du contenu audiovisuel et multimédia, MPEG-7 (Ferman *et al.*, 1998). Ce standard devrait permettre de décrire le contenu de documents audiovisuels ou multimédia intégrant divers types d'information : images fixes, vidéo, parole, musique, modèles de synthèse, etc., indépendamment de la manière dont ces informations sont codées. Le but est de spécifier un ensemble de descripteurs (*Descriptors, D*), de fournir un ensemble prédéfini de structures regroupant plusieurs descripteurs et précisant leurs relations, structures appelées « schémas de description » (*Description Schemes, DS*) et de fournir des moyens de définir de nouveaux schémas de description par l'intermédiaire d'un langage de définition de description (*Description Definition Language, DDL*).

Nous avons donc une structure hiérarchique à trois niveaux : les descripteurs sont des instances des schémas de description, ces schémas étant écrits dans un langage – ou méta-langage – de définition. Nous situons notre travail au niveau des descripteurs : il s'agit à partir de descripteurs primitifs de construire des descripteurs structurés d'un plus haut niveau d'abstraction. Nous proposons ainsi une méthode d'instanciation de descripteurs de séquences audiovisuelles. Lorsque nous avons commencé ce travail, MPEG-7 n'en était qu'à ses débuts. Nous avons donc utilisé le langage de description A.E.D.I défini à l'INA, langage que nous décrivons ci-dessous.

6.2 A.E.D.I

L'INA mène depuis 1997 une réflexion sur la définition d'un format d'échange de données descriptives de documents audiovisuels, appelé *Audiovisual Event Description Interface* (A.E.D.I), réflexion à laquelle nous avons contribué, principalement en début de thèse (Auffret *et al.*, 1999). Le travail effectué autour d'A.E.D.I relève de préoccupations similaires à celles exprimées autour de MPEG7. Le format A.E.D.I est utilisé par l'INA et ses partenaires dans plusieurs projets européens, dont DiVAN, où le format sert d'interface entre les algorithmes d'analyse automatique et le module d'interprétation[13]. Dans sa première version, A.E.D.I était doté d'une syntaxe textuelle propre. Un exemple de description A.E.D.I ainsi que la grammaire BNF de la syntaxe sont donnés en annexe. Gwendal Auffret, qui effectue sa thèse à l'INA sur la structuration des documents textuels décrivant la vidéo[14], a poursuivi cette réflexion et proposé un formalisme étendu, ainsi qu'une syntaxe fondée sur XML (Auffret & Bachimont, 1999).

En plus des travaux liés à l'élaboration du format, nous avons implémenté un outil informatique qui effectue d'une part l'analyse syntaxique et sémantique d'un flux A.E.D.I (partie *analyseur*, ou *parser*), et d'autre part la présentation des données du flux en synchronisation avec un support, tel que fichier vidéo ou fichier audio (partie *visualiseur*, ou *viewer*). L'analyseur syntaxique est utilisé dans le projet DiVAN pour interpréter les résultats des algorithmes d'analyse.

6.2.1 Le format A.E.D.I

Les contraintes auxquelles doit répondre le format d'échange sont de plusieurs ordres : dimension temporelle du document à décrire, expressivité suffisante pour prendre en compte des données disparates, simplicité d'utilisation et « streamabilité ».

Document temporel. Tout d'abord, il s'agit de décrire des documents audiovisuels ayant une dimension temporelle linéaire. Il est important que cette dimension temporelle soit prise en compte directement par le format. C'est la principale raison pour laquelle nous avons choisi de ne pas utiliser de format existant. Le format propose un mécanisme permettant d'exprimer les relations temporelles qu'entretient la description avec un ou plusieurs *supports* (fichiers vidéo, cassette vidéo, etc.). Il est ainsi par exemple possible à partir d'un découpage en plans formalisés en A.E.D.I de visualiser chacun des plans à partir d'un fichier MPEG.

[13] Voir « Le projet DiVAN » page 131.

[14] Le sujet en est « Structuration textuelle et structuration vidéo : outils linguistiques et documentaires pour structurer un document audiovisuel depuis ses descriptions textuelles. Application aux fiches documentaires de description chronologique. »

Expressivité. Il faut pouvoir transcrire aussi bien des données issues d'outils automatiques que des descriptions manuelles pouvant être de natures très diverses. Nous avons donc opté pour un *méta-langage* fondé sur la notion de *propriété*, dans lequel l'auteur du document de description peut définir ses propres propriétés. Les propriétés s'appliquent à des *entités*, qui sont les objets décrits. Le format ne définit que trois sortes d'entités, qui sont : *Unit*, entité non temporelle, qui peut être, par exemple, une couleur ou un personnage, *Event*, entité temporelle référençant un segment du document (un plan, une séquence, une réplique) et *Layer*, entité temporelle représentant un point de vue sur le document (le layer des plans, le layer des répliques, le layer des thèmes abordés, etc.). Events et layers peuvent être composés, ce qui permet de décrire par exemple des segmentations hiérarchiques (segmentation en séquences où chaque séquence est elle-même segmentée en plans) ou des points de vue imbriqués (layer « découpage son » dans lequel figurent les layers « découpage dialogues » et « découpage musique »). Une propriété est fondamentalement un couple *étiquette-valeur*, le type de la valeur pouvant être restreint. Le format propose quelques types de base, tels que nombres entiers ou flottants, chaînes de caractères, etc., ainsi que des types *références temporelles* permettant de repérer des segments de document. Ces types peuvent être combinés en types structurés (structures et listes). Les propriétés peuvent être regroupées en *modèles*. Un modèle n'est cependant pas une classe au sens de la programmation par objets. Il s'agit plutôt d'un contrat que s'engage à respecter un élément de description. Ainsi, par exemple, le modèle d'une description de plans peut définir les propriétés suivantes :

- un ou plusieurs mouvements de caméra, le mouvement de caméra par défaut étant « caméra fixe » ;
- le nombre de personnages, cette propriété étant optionnelle ;
- la couleur dominante ;
- la référence à une image représentative (keyframe).

Un modèle permet d'exprimer des restrictions de cardinalité sur les propriétés et de spécifier des valeurs par défaut. Il est possible d'associer à une entité des propriétés n'apparaissant pas dans son modèle.

Simplicité d'utilisation. Il est important que des structures de données simples puissent s'exprimer simplement. En particulier, il doit être possible pour un algorithme d'extraction de générer un fichier A.E.D.I à un coût minimum. C'est une des raisons qui ont conduit au choix d'un format texte.

Streamabilité. Aucune hypothèse n'est faite dans la définition du format sur l'origine et la destination des données de description. Un schéma possible est la génération « en continu » de données par algorithme, et leur utilisation immédiate par une unité de stockage. Il est important dans ce cas que les données puissent être présentées sous forme de *flux* (*stream*). Cette préoccupation de « streamabilité » a pour conséquence que les liens d'imbrication temporelle ou structurelle se font par l'intermédaire d'*identificateurs* et non par imbrication syntaxique.

6.2.2 Modèle temporel

Les éléments temporels (*events* et *layers*) d'une description A.E.D.I référencent des plages de temps d'un document audiovisuel. Un document audiovisuel peut être inscrit sur plusieurs *supports* physiques : bandes magnétiques, fichiers numériques, etc. On peut trouver plusieurs supports différents pour le même document, avec autant de manières différentes de se repérer temporellement sur le support : la même émission à pu être enregistrée à 24 images par seconde, diffusée à 25 images par seconde, numérisée en évitant les coupures publicitaires, etc. Il se pose alors deux questions auxquelles doit répondre le format de description : si un même document peut apparaître de plusieurs manières différentes, qu'est-ce qu'un document ? Comment repérer une même plage temporelle sur plusieurs supports différents ? A la première question, A.E.D.I apporte une réponse laconique qui tient compte du fait que la diversité des usages ne permet pas d'obtenir une définition unique de ce qu'est un document audiovisuel : pour A.E.D.I, un document est un axe temporel étiqueté. Cette définition permet de résoudre le problème de la correspondance entre une description et un ou plusieurs supports physiques en définissant un système de *projection* de la description sur le document (au sens d'A.E.D.I) d'une part, et entre le document et un support physique d'autre part. Ce système de projection est illustré par la figure 4.

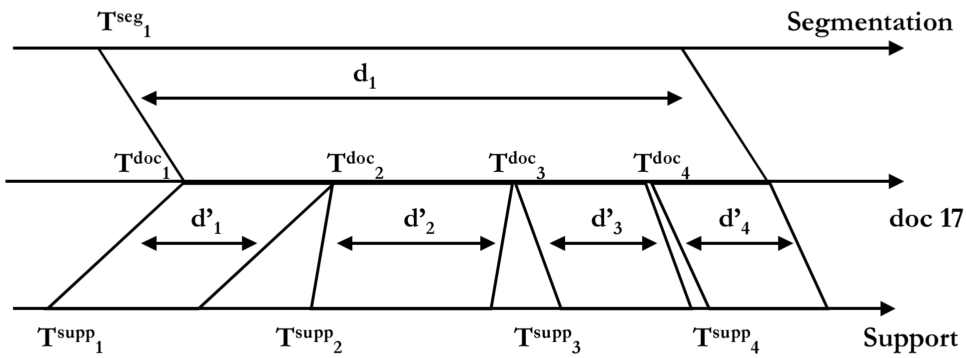


figure 4 : projections temporelles

A.E.D.I ne permet de spécifier que les projections temporelles entre la description et le document (*doc17* dans la figure), et ne spécifie pas comment indiquer les projections entre le document et les supports. Cependant, pour des raisons pratiques, un format de description des projections temporelles du document vers un support a été défini, ainsi qu'une syntaxe textuelle similaire à la syntaxe d'A.E.D.I. Enfin, notons que ce modèle temporel est inhérent au langage, ce qui ne semble pas être le choix fait, à l'heure actuelle, par les groupes de réflexion MPEG-7, où les informations temporelles semblent devoir être notées par des descripteurs n'ayant pas un statut particulier.

6.2.3 Analyseur - visualiseur A.E.D.I

Pour étayer les idées proposées pour la définition du format A.E.D.I, nous avons développé un outil permettant d'une part de vérifier la validité d'un flux A.E.D.I et d'autre part de visualiser les éléments du flux en synchronisation avec la vidéo. L'outil réalisé est écrit en Java™. L'analyseur syntaxique est écrit avec JavaCC (*Java Compiler Compiler*) qui permet de générer un analyseur syntaxique à partir d'une description d'une grammaire. L'analyseur généré est un ensemble de classes Java. La synchronisation avec la vidéo utilise l'interface de programmation (API) *Java Media Framework* (JMF) qui permet d'utiliser des fonctions de haut niveau telles que jouer, arrêter, aller à, pour des flux audio ou vidéo, indépendamment du support physique du document (fichiers audio ou vidéo de différents formats, flux ftp ou http par exemple).

La mise au point de cet outil nous a convaincu dès 1998 de la viabilité et de l'intérêt de ce format. Notons de plus qu'A.E.D.I a également été utilisé à l'INA comme format d'échange entre le logiciel d'annotation de la vidéo MediaScope et ses *plug-ins* d'analyse ou de documentation.

7 Conclusion

L'accès au contenu de documents audiovisuels par différents groupes d'utilisateurs, du grand public aux diffuseurs, est devenu une nécessité économique. La création, le stockage et l'échange de données de description du contenu audiovisuel font appel à diverses disciplines, essentiellement dans le domaine de la reconnaissance des formes et de la représentation des connaissances. Nous avons présenté dans cette section quelques travaux réalisés dans le domaine de la reconnaissance des formes ; ces travaux décrivent des méthodes d'analyse automatique telles que les méthodes de segmentation en plans, de détection de zones de texte ou de visage, les algorithmes d'analyse du flux audio – discrimination musique/non musique, par exemple –, ainsi que des méthodes de macro-segmentation visant à regrouper les segments primitifs extraits du flux en entités de plus haut niveau. Nous avons également présenté A.E.D.I, un format d'échange de données descriptives de la vidéo développé à l'INA qui répond à des préoccupations proches de celles exprimées par MPEG-7.

Dans le chapitre suivant, nous abordons le problème de l'indexation vu sous l'angle de la classification symbolique et présentons des travaux que nous avons menés en utilisant un formalisme de logiques de descriptions, formalisme logique de représentation des connaissances.

Chapitre 2
Logiques de descriptions pour l'indexation vidéo

Nous voulons raisonner sur les entités extraites de la vidéo ainsi que sur leurs modèles. Nous proposons de représenter ces informations – modèles et entités – dans un formalisme de logique de descriptions. Les logiques de descriptions forment une famille de langages de représentation de connaissances issus des idées développées autour des réseaux sémantiques et des langages de frames, idées dont le premier représentant est le système KL –ONE (Brachman, 1985). Les logiques de descriptions sont également appelées *logiques terminologiques*. On trouve dans (Napoli, 1998) une introduction très complète aux logiques de descriptions, l'ouvrage de référence restant certainement (Nebel, 1990a).

Dans une logique de descriptions, les connaissances sont représentées d'une part par des *concepts* et des *rôles* qui constituent la base terminologique (ou *TBox*) et d'autre part par les *individus* qui constituent la base des assertions (ou *ABox*). Un concept dénote un ensemble d'individus qui forment l'*extension* de ce concept tandis qu'un rôle dénote une relation binaire entre individus. Un concept peut être *primitif* ou *défini*. Les caractéristiques associées à un concept défini *C* sont nécessaires et suffisantes : tout individu appartenant à *C* possède les caractéristiques de *C* et réciproquement, de tout individu possédant ces caractéristiques il est possible d'inférer qu'il est une instance de *C*. Les concepts primitifs ne sont définis que par des caractéristiques nécessaires, c'est-à-dire que l'appartenance d'un concept à un concept primitif ne peut être déduite automatiquement. Concepts et rôles sont définis (décrits) par des *descriptions* faisant intervenir un certain nombre de *constructeurs*. Le choix de ces constructeurs permet de définir plusieurs langages.

1 Différents langages

Le langage minimal est le langage *AL*. Ce langage définit deux concepts particuliers, *Top* ($\top$) et *Bottom* ($\perp$) et quatre constructeurs. *Top* est le concept le plus général : toute instance est nécessairement instance de *Top* et réciproquement l'extension de *Top* regroupe tous les individus. *Bottom* est le concept vide, le concept le plus spécifique ; son extension ne contient aucun élément. Les constructeurs du langage *AL* sont :

- Conjonction de concepts : constructeur **and** (notation : $C \sqcap D$)
- Négation : constructeur **not**. Ce constructeur ne porte que sur les concepts primitifs (notation : $\neg C$).
- Quantification universelle : constructeur **all**. Ce constructeur permet de spécifier le co-domaine d'un rôle. Ainsi, (**all** r *C*) décrit l'ensemble des individus qui n'ont pour valeur du rôle r que des instances du concept *C* (notation : $\forall r.C$).
- Quantification existentielle non typée : constructeur **some**. Ce constructeur indique que les instances du concept (**some** r) sont mises en relation par r avec au moins un autre individu (notation : $\exists r$).

A ces constructeurs de base du langage *AL* peuvent être ajoutés d'autres constructeurs, ce qui permet d'enrichir le langage :

- Négation de concepts primitifs ou définis : constructeur **not** (notation : $\neg$C). Il s'agit alors du langage *ALC* auquel sont souvent comparés les langages plus expressifs.
- Disjonction de concepts : constructeur **or** (notation : C $\sqcup$ D).
- Quantification existentielle typée : constructeur **c-some** (notation : $\exists$r.C).
- Restriction de cardinalité sur les rôles. Constructeurs **at-least** (notation : $\geq n$ r) et **at-most** (notation: $\leq n$ r).

D'autres constructeurs peuvent encore être ajoutés, enrichissant encore le langage (conjonction de rôles, par exemple).

2 Subsomption et instanciation

Les descriptions de concepts sont organisées en hiérarchie par la relation de *subsomption* : le concept *C* subsume le concept *D* si, et seulement si, toute instance de *D* est nécessairement instance de *C*. La *classification* est l'opération permettant d'attribuer une place à un concept (ou éventuellement à un rôle) dans la hiérarchie des concepts (ou des rôles). La relation de subsomption est définie de manière extensionnelle : *C* subsume *D* si, et seulement si, toute instance de *D* est nécessairement instance de *C*. Cependant, comme on ne peut connaître l'extension réelle d'un concept, la relation de subsomption doit être établie de manière intensionnelle à partir de la définition des concepts. Selon les constructeurs dont se dotent le langage, cette opération peut être plus ou moins complexe, voire indécidable (Nebel, 1990a; Nebel, 1990b).

L'*instanciation* est l'opération qui vérifie l'appartenance d'un concept *C* à la description d'une instance *o*. Cette opération est très similaire au test de subsomption, puisqu'il s'agit schématiquement de tester si le concept *C* subsume la *description* de *o*. En pratique, l'implémentation du test de subsomption et des opérations d'instanciation repose fréquemment sur la méthode des *tableaux sémantiques* (Gochet & Gribomont, 1992).

3 Sémantique

Les différents formalismes de logique de descriptions se dotent en général d'une sémantique formelle ; c'est même une des revendications fortes de cette famille de langages que d'être « sémantiquement bien fondés ». Bernhard Nebel (Nebel, 1990a) justifie l'intérêt d'associer une sémantique formelle au langage de description principalement par le fait que la sémantique permet de se mettre d'accord sur ce que signifient les expressions du langage. Il devient alors possible de comparer les résultats réels d'un système de représentation des connaissances avec les résultats attendus et de comparer les systèmes entre eux. D'autre part, Nebel propose d'utiliser la théorie des modèles de Tarski, dans laquelle les entités manipulées sont censées dénoter des entités du monde réel[15]. Dans cette optique, les concepts sont interprétés comme des sous-ensembles d'un domaine d'interprétation et les rôles comme un ensemble de relations sur ce domaine. Soient $\mathcal{T}$ un ensemble de concepts, C et D des concepts, et r un rôle. Une *interprétation* $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est la donnée d'un ensemble non vide $\Delta^{\mathcal{I}}$ et d'une fonction $.^{\mathcal{I}}$ (fonction d'interprétation, ou fonction d'*extension*) qui associe à chaque concept un sous-ensemble de $\Delta^{\mathcal{I}}$ et à chaque rôle un sous-ensemble de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ et telle que les équations données par la figure 5 soient respectées (nous ne donnons ici que les équations correspondant aux principaux constructeurs). Si c'est le cas, on dit que $\mathcal{I}$ est un *modèle* de $\mathcal{T}$.

$$
\begin{array}{lcl}
(\top)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
(\bot)^{\mathcal{I}} & = & \varnothing \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall r.C)^{\mathcal{I}} & = & \{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 : (d_1,d_2) \in r^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\} \\
(\exists r.C)^{\mathcal{I}} & = & \{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2 : (d_1,d_2) \in r^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}
\end{array}
$$

figure 5 : sémantique extensionnelle

Comme le fait remarquer (Napoli, 1998), d'autres manières d'associer une sémantique aux différentes constructions d'une logique de descriptions ont été proposées. (Borgida, 1992), par exemple, applique la *déduction naturelle*. Dans ce formalisme, une définition consiste en un ensemble de règles d'inférence et d'axiomes qui permettent d'écrire des formules de la forme $\langle \textit{antécédent} \rangle \vdash \langle \textit{conséquent} \rangle$, appelés *séquents*. La déduction illustrée par la figure 6 exprime qu'une restriction de type **all** est d'autant plus spécifique que le concept (*d* sur la figure) qui restreint un rôle (*p* sur la figure) est plus spécifique.

$$
\frac{\vdash c \Rightarrow d}{\vdash \mathrm{all}(p,c) \Rightarrow \mathrm{all}(p,d)}
$$

figure 6 : exemple de déduction naturelle

4 Extensions des logiques de descriptions

Les logiques de descriptions présentent l'avantage d'être bien formalisées et de posséder une sémantique bien fondée. Leur pouvoir d'expression est plus ou moins grand suivant les langages et les systèmes (Heinsohn *et al.*, 1994; Kurtonina, 1997), mais il est généralement reconnu que ces langages ne peuvent prendre en charge qu'une partie limitée des besoins

[15] "expressions of a KR formalism should derive their meaning in some way from reference to external world; and the adequate tool for such a task is Tarskian model theory"

d'une application de taille réelle (Brachman *et al.*, 1991). La position la plus réaliste consiste à demander à un système terminologique de prendre en charge l'aspect terminologique de la tâche à accomplir, et de laisser le reste du système ou de l'application s'occuper des autres problèmes. C'est ce qui est fait par exemple dans le projet GALEN (Rector *et al.*, 1999). Le projet GALEN se situe dans le domaine médical où les problèmes terminologiques sont nombreux (anatomie, pathologies, traitements) et les besoins importants (diagnostic, dossiers patients, statistiques). Le but du projet GALEN est de rendre un service terminologique médical le plus complet possible. Pour cela, un modèle cohérent (*sound*) d'une terminologie médicale a été conçu ; cette terminologie est rendue disponible au reste du système par l'intermédiaire du langage terminologique GRAIL (*Galen Representation and Integration Language*). Différents modules annexes s'occupent par exemple du traitement de la langue naturelle, des aspects multilinguaux, des conversions vers d'autres langages de représentation de connaissances, etc. Dans ce cas, le système terminologique est intégré en tant que module dans une application plus complète. C'est la façon la plus simple d'ajouter des fonctionnalités à un langage terminologique.

Un langage terminologique peut être enrichi d'autres façons. Certains systèmes offrent au programmeur des facilités qui lui permette d'étendre le pouvoir d'expression du langage. Nous présentons ci-dessous certaines des extensions procédurales des systèmes CLASSIC et Loom. Une autre manière d'étendre un langage terminologique est d'intégrer les outils qu'il offre, principalement les mécanismes de classification et d'instanciation, dans un autre système de représentation de connaissances, comme c'est le cas du système CLASP (Yen *et al.*, 1991) que nous évoquerons. Enfin, beaucoup de travaux visent à étendre des formalismes de logiques de descriptions « de l'intérieur » en enrichissant le langage et en proposant une sémantique adaptée ainsi que des mécanismes d'inférence. Sur ce dernier point, nous nous attacherons spécialement à présenter quelques extensions de logiques de descriptions cherchant à permettre de représenter des connaissances temporelles et nous évoquerons le problème spécifique de la représentation des relations « partie de ». Notons que d'autres extensions que les extensions temporelles ou les extensions à des relations « partie de » ont été proposées à des formalismes de logiques de descriptions. (Koller *et al.*, 1997) proposent ainsi P-CLASSIC, une extension probabiliste du système CLASSIC dans laquelle il est possible d'exprimer le degré de recouvrement de deux concepts, et non pas seulement leur relation de subsomption.

4.1 Extensions procédurales

On souhaiterait parfois pouvoir classifier une instance en fonction de critères qui ne peuvent être exprimés dans le langage logique sur lequel repose le formalisme. C'est le cas par exemple des tests numériques. On voudrait ainsi par exemple décrire le concept de « gros plan d'un personnage » comme le concept de plan où apparaît un visage dont la superficie occupe au moins la moitié de la superficie totale de l'image (figure 7).

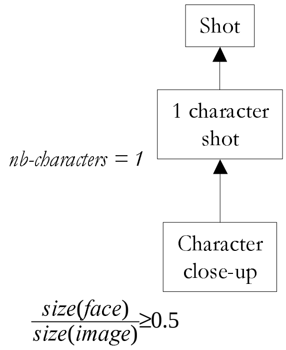


figure 7 : gros plan d'un personnage

Certains systèmes, comme CLASSIC et Loom offrent des extensions permettant de définir des critères de classification de manière procédurale.

4.1.1 CLASSIC

Le système CLASSIC offre des possibilités de règles de production en chaînage avant. Trois types de règles sont possibles : les règles simples (*simple rules*), les règles de description (*description rules*) et les règles de valeurs de rôle (*filler rules*). Schématiquement, une règle simple consiste en un concept antécédent et un concept conséquent. Dès qu'un individu est décrit par le (est instance du) concept antécédent, la règle est déclenchée et le concept conséquent est ajouté à la description de l'individu (l'individu devient instance du concept conséquent). Il s'agit donc d'exprimer une implication : le concept antécédent implique le concept conséquent, puisque la description du concept conséquent est ajoutée à tout individu instance du concept antécédent. Certaines implications comparables peuvent également être exprimées dans le système BACK (Nebel & von Luck, 1988).

Il est également possible dans une règle simple de spécifier un *filtre* qui permet de n'ajouter le concept conséquent à la description de l'individu pour lequel la règle est déclenchée qu'à la condition que le filtre soit passant. Un filtre peut être soit une description, soit un *test*, au sens de CLASSIC, c'est-à-dire une fonction lisp renvoyant *t*, *nil* ou '?, ce dernier symbole indiquant que la valeur du test n'est pas connue et peut devenir soit *t*, soit *nil* en fonction de l'évolution de la base d'assertions (CLASSIC fait l'hypothèse du monde ouvert). Nous donnons ci-dessous (figure 8) un exemple de règle simple permettant d'exprimer le concept de « gros plan d'un personnage » illustré par la figure 7.

```
;;; Définition du concept Face-Region, qui
;;; exprime une zone de l'image correspondant à un visage
;;; w : largeur de la région
;;; h : hauteur de la région
(cl-define-concept 'Face-Region `(and
   ;;(…)
   (at-least 1 w) (all w float)
   (at-least 1 h) (all h float)))

;;; Définition du concept Shot. Un plan possède
;;; également une largeur w et une hauteur h
(cl-define-concept `Shot `(and
   ;; (…)
   (at-least 1 w) (all w float)
```

```
   (at-least 1 h) (all h float)))

;;; Plan à un personnage
(cl-define-concept 'OneCharacterShot
  '(and
    Shot
    (at-least 1 has-face)
    (at-most 1 has-face)
    (all has-face Face-Region)))

;;; Gros plan à un personnage
(cl-define-primitive-concept 'OneCharacterCloseUp
'OneCharacterShot)

;;; Fonction testant le rapport de surface entre la région
;;; de l'image correspondant à un visage et l'image entière.
;;; Retourne t si ce rapport est supérieur à 0.5
(defun test-face-size(shot)
   (let* ((w-role (cl-named-role 'w))
          (h-role (cl-named-role 'h))
          (face-role (cl-named-role 'has-face))
          (face (first (cl-fillers shot face-role)))
          (face-w (first (cl-fillers face w-role)))
          (face-h (first (cl-fillers face h-role)))
          (w (first (cl-fillers shot w-role)))
          (h (first (cl-fillers shot h-role))))
      (when (> (/ (* face-w face-h) (* w h)) 0.5) t)))

;;; Déclaration de la règle CLASSIC déclenchée lors de
;;; la création d'une instance du concpet OneCharacterShot
(cl-add-rule 'CloseUpRule
  @OneCharacterShot
  'OneCharacterCloseUp
  :filter '(TEST-C test-face-size))
```

figure 8 : exemple de règle simple en CLASSIC

Les règles de description et les règles de valeurs de rôle reposent sur le même principe que les règles simples, mais offrent la possibilité pour les premières de construire dynamiquement une description de concept et pour les secondes de construire dynamiquement une liste de valeurs de rôles. L'utilisation de ces règles peut s'avérer fort utile pour exprimer des relations autres que les relations de types « est-un ». Nous avons ainsi proposé dans (Carrive *et al.*, 1998; Carrive *et al.*, 2000b) une méthode de regroupement des plans en séquences utilisant les règles CLASSIC, et nous présenterons plus loin une utilisation de ces règles permettant d'exprimer certaines relations du type « partie de » entre un plan et ses images clé[16].

Du point de vue du système d'inférence de CLASSIC, les tests sont des boîtes noires qui servent à définir des concepts de manière procédurale. Cela peut conduire à des cas d'incomplétude, CLASSIC n'étant pas en mesure d'établir un lien de généralité entre deux concepts définis par des tests. Cependant, CLASSIC offre au programmeur la possibilité de déclarer lui-même les liens de subsomption existant entre des concepts définis par des tests.

[16] Voir « Classification de plans avec CLASSIC » page 42.

4.1.2 Loom

Les extensions procédurales proposées par le système Loom se composent des *actions*, des *méthodes* et des *productions*. Une action est une opération générique qui peut être invoquée pour remplir une tâche spécifique. Les actions sont implémentées par un ensemble de méthodes associées chacune à une description de la *situation* dans laquelle cette méthode peut être exécutée. Lorsque plusieurs méthodes peuvent être appliquées, Loom choisit celles correspondant aux situations les plus spécifiques, suivant en cela une technique classique en programmation par objets. La méthode illustrée par la figure 9 ne peut être déclenchée que si son argument est une instance du concept *OneCharacterShot* (clause `:situation`). Cette méthode teste le rapport entre la superficie du visage détecté et la superficie totale de l'image (en pseudo-code sur la figure) et si ce rapport est supérieur à 50%, le concept *OneCharacterCloseUp* est ajouté à la description de l'individu paramètre de la méthode[17].

```
(defmethod isCloseup (?shot)
   :situation (OneCharacterShot ?shot)
   :response
   (when
       (> (/ (size face) (size image)) 0.5)
       (tell (OneCharacterCloseUp ?shot))))
```

figure 9 : exemple de méthode dans le système Loom

Une production est une règle qui indique une tâche à accomplir quand certaines conditions sont remplies ; une tâche est constituée d'une ou plusieurs actions. Une production en Loom comporte deux parties principales :

- le mot-clé `:when`, suivi d'une expression booléenne devant être vraie pour que la production puisse être déclenchée. A l'intérieur de cette expression, les mots clés `:detect` et `:undetect` introduisent une expression qui est vraie (resp. fausse) quand l'expression elle-même est passée de vraie à fausse (resp. de fausse à vrai) lors du dernier changement d'état de la base de faits. Ces expressions sont à rapprocher des expressions introduites par `:FAIL` dans le système BACK. Des variables peuvent être utilisées ici, qui seront liées à l'exécution à des individus de la base de faits. On dit qu'une production est « dirigée par les données » (*data-directed*).

- le mot-clé `:perform` ou `:schedule` suivi d'une méthode Loom. La méthode est exécutée (`:perform`) ou planifiée (`:schedule`) par le système lorsque l'expression de la clause `:when` est vérifiée.

La production illustrée par la figure 10 demande au système de déclencher la méthode *isCloseup* pour toutes les instances de *OneCharacterShot*. Notons que dans les exemples cités l'appartenance de l'instance en question au concept *OneCharacterShot* est vérifiée deux fois : une fois par la production, une fois par la méthode dans la clause `:situation`.

[17] Il faudrait ici demander à Loom les valeurs respectives des rôles représentant la taille de l'image et la taille de la région remplie par l'unique visage détecté dans l'image, et effectuer le test sur les valeurs ainsi obtenues.

```
(defproduction oneCharacterCloseUpProduction
  :when (OneCharacterShot ?shot)
  :perform (isCloseup ?shot))
```

figure 10 : exemple de production en Loom

4.2 Logiques de descriptions et systèmes à base de règles

Le système CLASP (Yen *et al.*, 1991) cherche à construire un système à base de règles de production qui bénéficie d'une procédure de classification des règles sémantiquement bien fondée. Dans ce système, les prémisses des règles sont représentées par une conjonction de conditions, chaque condition étant constituée d'un type et d'un prédicat. Le type d'une condition peut être :TRUE, :NOT-TRUE ou :FAIL, qui indique le passage de la valeur du prédicat de *vrai* à *faux*. L'exemple donné par les auteurs est qu'une règle ayant comme prémisse (:and (Foreigner ?x) (:FAIL (Student ?x))) sera déclenchée quand un étudiant étranger perd son statut d'étudiant. Un prédicat peut être une description de concept (prédicat unaire) ou une description de rôle (prédicat binaire). Le système CLASP intègre les procédures de classification et d'instanciation du système Loom (MacGregor, 1994)] ; ces procédures lui offre l'avantage sur un système à base de règles classique sur au moins deux points.

D'une part, Loom prend en charge la base de faits et son procédé d'*instanciation* maintient à jour pour chaque fait la liste des concepts dont il est instance. Lorsque des faits sont ajoutés dans la base, les faits existants peuvent être re-classifiés et de nouvelles règles peuvent être déclenchées.

```
(defconcept Father (:and Male (:at-least 1 child)))

(defrule R1 :when (Father x?) :then ...)

(assert-c Male Bob)
```

figure 11 : exemple de règle associée à un concept

La figure 11 donne une définition de concept, une définition de règle et une assertion. Si l'on ajoute l'assertion donnée par la figure 12, *Bob* se trouve être classifié comme instance de *Father* et la règle R_1 peut être déclenchée.

```
(assert-r Child Bob Rosa)
```

figure 12 : ajout d'une assertion à la base de faits

D'autre part, une relation de subsomption entre règles est définie qui utilise la relation de subsomption entre concepts, et les règles sont organisées en taxinomie. Il est ainsi possible pour l'algorithme de déclenchement de règles de déclencher la plus spécifique de deux règles en conflit. La spécificité d'une règle par rapport à une autre est ainsi mieux définie que par d'autres critères, comme par exemple le nombre de clauses dans les prémisses ou la simple inclusion des clauses d'une prémisse de règle dans la prémisse d'une autre règle (Andert, 1992), comme illustré par la figure 13.

```
R1: If P(x) then V(x)
R2: If P(x) and Q(x) then W(x)
```

figure 13 : spécificité de règles de production

4.3 Extensions temporelles

Plusieurs travaux proposent des extensions au formalisme des logiques de description dans le but de représenter et de raisonner sur des connaissances temporelles. Artale et Franconi (Artale & Franconi, 1999), dans un article de synthèse, proposent une étude des principaux travaux réalisés sur ce sujet.

4.3.1 Formalisme de Schmiedel

La première tentative d'extension temporelle d'un formalisme de logique de descriptions est due à Schmiedel (Schmiedel, 1990) qui propose une logique de descriptions enrichie d'une logique temporelle fondée sur les intervalles. De nouveaux constructeurs sont introduits dans une logique de descriptions existante, qui permettent de décrire des concepts temporels : le constructeur **at** permet de préciser l'intervalle temporel pendant lequel une instance est valide et les constructeurs **sometime** et **alltime** sont des quantificateurs temporels permettant d'introduire des variables temporelles. Les variables temporelles sont contraintes par des relations temporelles fondées sur l'algèbre d'intervalles de Allen étendue avec des contraintes numériques permettant de spécifier des durées absolues à différents niveaux de granularité. Schmiedel propose pour chacune des nouvelles constructions de concepts une sémantique qui préserve leur sens intuitif. Malheureusement, aucun algorithme de subsomption n'est proposé et, de plus, (Artale & Franconi, 1999) montrent que la logique ainsi définie est indécidable. Il faut cependant noter que cette logique étant très expressive et sémantiquement bien définie, elle constitue une référence à laquelle peuvent être comparées les autres extensions temporelles de logiques de descriptions.

4.3.2 Logique temporelle d'Artale & Franconi

Partant du constat que la logique de Schmiedel est indécidable et que le système T-Rex de Weida et Litman utilise deux langages différents pour définir les actions (concepts non-temporels) et les plans (réseaux de contraintes temporelles entre actions), Artale et Franconi (Artale, 1994) proposent de restreindre l'expressivité de la logique de Schmiedel. Dans le même ordre d'idées que Schmiedel, un quantificateur temporel existentiel noté ◊ permet d'introduire des variables temporelles. Des contraintes temporelles restreignant ces variables peuvent être définies dans la description de concepts temporels. Une pseudo-variable **now**, notée #, est utilisée pour désigner l'intervalle de temps pendant lequel l'action définie a lieu. Ainsi, l'expression :

$$\begin{aligned}\text{P-I-Seq} \doteq\ & \Diamond(x\ y)(x\ \mathrm{s}\ \#)(y\ \mathrm{f}\ \#)(y\ \mathrm{mi}\ x)\\ & ((*\mathrm{s1}:\mathrm{P})@x \wedge (*\mathrm{s2}:\mathrm{I})@y)\end{aligned}$$

spécifie qu'une séquence de type P-I-Seq (P pour *présentateur*, I pour *invité*) est composée d'une instance d'un plan de type P (*s1), d'un plan de type I (*s2), et que la relation entre le plan P et la séquence est *starts*, la relation entre le plan I et la séquence est *finishes* et la relation entre le plan P et le plan I est *meets*.

Artale et Franconi proposent une sémantique formelle pour leur syntaxe ainsi qu'un algorithme pour le test de subsomption. Cet algorithme se fonde sur le calcul d'une *forme normale*. Artale et Franconi soulignent qu'en présence de disjonctions de concepts, la dimension de cette forme normale est potentiellement exponentielle en fonction du nombre de concepts. Ils proposent plusieurs méthodes de calcul de forme normale en présence ou en l'absence de disjonctions. Comme dans le système T-Rex que nous verrons plus loin[18], les concepts temporels sont organisés en taxinomie. Le problème de la reconnaissance proprement dit est vu ici comme un problème d'instanciation : reconnaître le plan suivi par un agent revient à calculer le concept *décrivant* les observations. A partir des observations, la description la plus spécifique est calculée (*most specialized concept expression*). Etablir le ou les plans suivis par l'acteur revient alors à déterminer les plans qui subsument cette description.

4.3.3 Le système CLASP

Le système CLASP (*Classification of Scenarios and Plans*) (Devanbu, 1996)[19] est un système de représentation de connaissances construit au-dessus de CLASSIC (Borgida *et al.*, 1989). Ce système raisonne sur des *plans*, au sens de la reconnaissance de plans, qui sont une extension des concepts définis dans CLASSIC, et sur les *scénarios*, qui sont des instances de plan. Un plan représente une certaine configuration d'actions qu'on cherche à reconnaître parmi les actions effectivement réalisées par un agent. Si c'est le cas, on dit que le plan est *reconnu*, qu'il *satisfait* les observations ou que les observations *réalisent* le plan. Dans le cas de CLASP, un plan est défini d'une manière similaire à la façon dont sont définis les concepts CLASSIC en ce sens qu'un plan représente une classe d'instances – les scénarios. Une procédure de décision permet de déterminer qu'un ensemble d'observations réalise un plan, c'est-à-dire qu'un ensemble d'observations constitue une instance de ce plan. On dit alors, en empruntant la terminologie des logiques de descriptions, que le plan *subsume* les observations. De même que pour les concepts CLASSIC, un plan *P* subsume un plan *P'* si toute instance de *P'* est nécessairement instance de *P*.

Dans CLASP, un plan est décrit à l'aide de constructeurs empruntés aux expressions régulières, et à l'aide des concepts d'action et d'état. Les actions sont des sous-concepts du concept *Action* décrit avec CLASSIC. Le concept *Action* comporte des rôles définissant les notions d'acteur, de pré-condition, de post-condition, de but, etc. Ces définitions peuvent être restreintes par les sous-concepts du concept *Action* en utilisant les constructeurs propres à CLASSIC, dont le mécanisme de classification permet d'obtenir une taxinomie des actions élémentaires. De même, CLASSIC permet de définir une taxinomie d'états, qui expriment les états de l'agent à observer, et qui ne peuvent être que des conjonctions de concepts primitifs. Les opérateurs essentiels permettant de définir un plan expriment la disjonction, le séquencement et la répétition. La procédure du test de subsomption repose sur l'emploi de techniques issues de la théorie des automates qui permettent de comparer le

---

[18] Voir « Le système T-Rex » page 58.

[19] A ne pas confondre avec le système CLASP de (Yen *et al.*, 1991).

degré de généralité de deux expressions régulières en employant l'automate d'états finis déterministe représentant l'intersection des deux expressions, cet automate étant le résultat du produit cartésien des automates d'états finis déterministes construits à partir des deux expressions initiales. Devanbu étend le calcul du produit cartésien de deux automates pour tenir compte du fait que les transitions observées entre les états des automates sont des instances de concepts CLASSIC. Soient les deux automates (a) et (b) décrits par la figure 14, empruntée à (Devanbu, 1996) ; l'automate (a) indique qu'on passe de l'état 1 à l'état 2 lorsqu'on observe une instance du concept A. Les transitions sont associées à des concepts (A et B, sur la figure) et non à de simples lexèmes comme c'est le cas pour les expressions régulières ; il est donc possible dans l'état 1 d'observer une transition qui soit instance de A et B simultanément. Le produit cartésien des deux automates est donc beaucoup plus coûteux à calculer. L'automate (c) de la figure 14 représente le produit cartésien des automates (a) et (b).

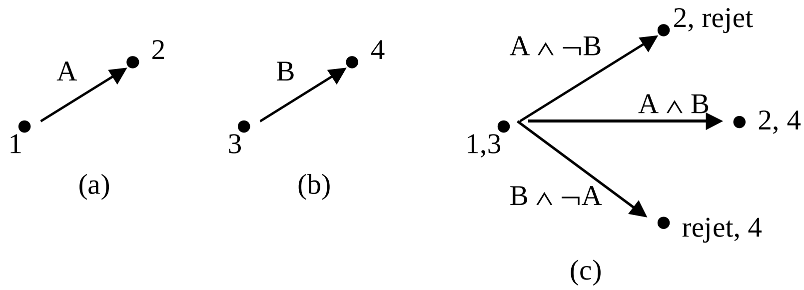


figure 14 : produit cartésien d'automates

4.3.4 Le système T-Rex

Nous présentons ici rapidement le système T-Rex [Weida 1992]. Nous avons fondé une grande partie de notre travail sur ce système et nous y reviendrons en détail au prochain chapitre[20].

Le système T-Rex, toujours dans le domaine de la reconnaissance de plans, n'est pas très éloigné dans le fond de CLASP. Un plan est défini comme un ensemble d'actions décrites dans un système de logiques de descriptions, ces actions étant contraintes temporellement par des disjonctions de relations de Allen. Pour définir la relation de subsomption, que ce soit entre un plan et les observations ou entre deux plans, T-Rex intègre raisonnement terminologique et satisfaction de contraintes, et non plus théorie des automates.

La relation de subsumption entre contraintes temporelles est définie suivant un schéma intuitif : la relation temporelle $r_1=\{meets \vee before \vee overlaps\}$ est plus spécifique que la relation temporelle $r_2=\{meets \vee overlaps\}$ puisque tous les individus mis en relation par r_2 le sont nécessairement par r_1. Une relation temporelle étant définie comme une disjonction de relations élémentaires (les 13 relations de Allen), la subsomption entre relations temporelles est définie à l'aide de la relation ensembliste d'inclusion : r_1 subsume r_2 si toutes les relations élémentaires de r_1 sont contenues dans r_2.

[20] Voir « Le système T-Rex » page 58.

Un plan est défini dans T-Rex par la donnée de *concepts d'action* organisés en une *taxinomie d'actions*, et de contraintes temporelles entre ces actions. Une relation de subsomption est définie pour les plans qui sont alors organisés en une *taxinomie de plans*. Pour pouvoir tester effectivement la subsomption entre deux plans, le système doit d'abord propager les contraintes temporelles données explicitement pour obtenir la *clôture* du réseau de contraintes. Le problème de la propagation de contraintes dans un formalisme fondé sur les intervalles temporels est connu pour être difficile. Plusieurs possibilités sont communément reconnues pour aborder cette difficulté :

- Adopter un algorithme complet mais potentiellement exponentiel.
- Restreindre la logique temporelle sous-jacente. Des sous-ensembles « calculables » de la logique temporelle de Allen ont ainsi été identifiés ((Van Beek, 1992; Nebel & Bückert, 1994))
- adopter un algorithme non complet. C'est ce que fait Allen (Allen, 1983) et c'est ce que font Weida et Litman.

La taxinomie (ou librairie) de plans est utilisée pour la reconnaissance de plans. En fonction des observations, les plans sont classés selon trois modalités : *nécessaires*, *possibles* et *impossibles*. Un plan est nécessaire s'il est réalisé quelles que soient les observations à venir, un plan est impossible s'il ne peut être réalisé quelles que soient les observations futures et un plan est possible si de nouvelles observations peuvent permettre d'affirmer que le plan est réalisé.

4.4 Hiérarchies « partie-de »

Un travail important est également réalisé dans le domaine des hiérarchies « partie-de » (*part-of* ou *part-whole hierarchies*). Nous décrivons ici succinctement quelques aspects particuliers liées à ce type de relations, en nous inspirant principalement de (Artale *et al.*, 1996b).

Les relations du partie au tout (*part-whole relations*), relations *méréologiques*[21], ne peuvent être modélisées comme des attributs ordinaires, tels que *prix* ou *couleur*, qui sont en relation *ontologique* avec leur tout. Il existe aujourd'hui une vision minimale consensuelle sur la théorie des parties et des touts : « Classical or General Extensional Mereology » (*GEM*), qui introduit la notion de relation « partie propre de » (*proper-part-of relation*) comme une relation d'ordre strict (transitive et asymétrique) satisfaisant plusieurs principes, parmi lesquels :

- *Extensionalité* : deux individus sont identiques si, et seulement si, ils ont les mêmes parties.
- *Principe de somme* : il existe toujours un individu composé de deux individus quelconques de la théorie.
- *Complémentarité* : Si un individu x est une partie propre d'un individu y, alors il existe un autre individu z qui est la partie manquante de y.

[21] Du grec *meros*, « partie ».

La relation « partie de » (*part-of*) est définie comme la disjonction de la relation « partie propre de » et de la relation d'identité. C'est donc une relation d'ordre partiel.

Ces principes sont souvent trop rigides pour s'appliquer aux cas réels, et ne permettent pas de rendre compte des individus gardant leur intégrité tout en perdant ou en acquérant des parties, ou d'individus distincts partageant les même parties.

Une des propriétés les plus discutées de ces relations est la *transitivité*. Si cette propriété est souvent vérifiée, elle ne l'est pas toujours. Artale *et al.* proposent comme contre-exemple le bras du musicien d'orchestre : le bras est une partie du musicien, le musicien est une partie de l'orchestre, mais le bras du musicien peut difficilement être considéré comme une partie de l'orchestre. La confusion vient de la polysémie de l'expression « partie de ». (Winston *et al.*, 1987) (repris dans (Odell, 1994)) proposent une classification des relations partie-de en six catégories établies selon trois critères :

- Configuration : les parties fonctionnelles (*functional parts*) sont contraintes temporellement ou spatialement par leur fonction.
- Homéomérie[22] : les parties « homéomériques » (*homeomerous parts*) sont du même type que le tout.
- Invariance : les parties séparables (*separable part*) peuvent être séparées du tout.

Les six relations « partie de » sont :

- Composant/objet intégral (*Component/Integral Object*). Les objets intégraux sont structurés, les parties sont séparables et sont fonctionnelles (les roues d'une voiture).
- Membre/collection (*Member/Collection*). Les parties n'ont pas de rôle fonctionnel par rapport au tout mais sont séparables (les arbres d'une forêt).
- Portion/masse (*Portion/Mass*). Le tout est un agrégat homogène, les parties sont similaires à lui et sont séparables (les parts d'un gâteau).
- Matériau/objet (*Stuff/Object*). Ne peut être séparé, comme le métal dont est fait une bicyclette.
- Phase/activité (*Feature/Activity*). Les parties ont un rôle mais ne sont pas séparables (le ramassage est une phase de la récolte).
- Endroit/région (*Place/Area*). Les parties sont similaires au tout, mais ne sont pas séparables (une oasis dans le désert).

La distinction entre ces six types de relations permet entre autres d'évacuer le problème de transitivité cité plus haut : la transitivité n'est vérifiée que pour un même type de relation. D'autres problèmes demeurent cependant, comme l'existence d'une relation partie-de plus générale dont les six relations décrites plus haut ne seraient que des cas particuliers. Cela a conduit d'autres auteurs à proposer des classifications différentes pour les relations partie-de. D'autres problèmes difficiles sont encore à résoudre, comme le montre (Artale *et al.*, 1996a). Citons en particulier la difficulté qu'il y a à formaliser la manière dont les caractéristiques des parties sont transmises au tout ou, inversement, dont les

[22] Le terme ne semble pas exister en français.

caractéristiques du tout sont transmises aux parties. Nous abordons également cette question dans la section suivante, où nous cherchons en particulier à faire « remonter » des informations extraites d'images clé aux plans auxquels elles appartiennent.

5 Classification de plans avec CLASSIC

Nous présentons dans cette section une expérimentation que nous avons menée avec le système CLASSIC sur la classification des plans d'une collection de journaux télévisés (Ronfard *et al.*, 2000), expérimentation qui poursuit les idées présentées dans (Ronfard, 1997). Cette classification exploite les résultats d'un algorithme de segmentation en plans et deux algorithmes numériques d'extraction de primitives développés par le laboratoire FORTH dans le cadre du projet DiVAN (Garcia & Tziritas, 1999; Garcia & Apostolidis, 2000). L'algorithme de segmentation en plans affecte à chaque plan détecté une ou plusieurs images clé représentatives du plan. Les deux autres algorithmes détectent dans une image clé pour le premier les régions de l'image qui correspondent à des visages humains et pour le second des régions de l'image qui correspondent à du texte. Dans un contexte particulier – le journal télévisé de la chaîne « M6 », par exemple –, ces simples informations nous permettent de classifier un grand nombre de plans selon des catégories significatives, telles qu'un plan d'interview ou un plan de lieu.

Ainsi, considérons les images caractéristiques des plans reproduites sur la figure 15. Ces images sont extraites de l'édition du journal de « TF1 » du corpus AIM[23]. Les plans où figure un visage filmé en « plan poitrine » (*Medium Close-Up Shot*, abbrévié en *MCU*) et où apparaît une zone de texte en bas à gauche (images (1) à (3) sur la figure) sont des plans d'interviews où le texte incrusté donne le nom de la personne filmée. Les plans où figure une zone de texte en haut à gauche sont des plans de lieu où le texte donne le nom du lieu filmé (images (4) à (6)). L'image (7) représente un plan qui est à la fois un plan de personne et un plan de lieu. L'image (8) est une exception – la seule de l'émission –, où le texte incrusté en haut à gauche n'est pas un nom de lieu (il est inscrit : « Images de France Télévision »). Il est fréquent que des règles analogues soient appliquées par les différentes chaînes, règles qui sont en général respectées assez scrupuleusement.

(1)

(2)

(3)

[23] Pour le corpus AIM, voir « Reportages des JT du corpus AIM » page 148.

(4) (5) (6)

(7) (8)

figure 15 : plans caractéristiques du JT de TF1

La liste des éléments fournis par les outils d'analyse est donnée ci-dessous :

- les plans (*shots*) ;
- les images clé (*keyframes*) représentatives des plans ;
- les régions de texte (*captions*) détectées dans une image clé ;
- les régions de visage (*faces*) détectées dans une image clé.

Chacune de ces catégories d'éléments est de manière intuitive représentée par un concept primitif, les concepts *Caption* et *Face* étant regroupés sous un concept commun dénotant une région de l'image (*Image-Region*). Ces concepts se retrouvent sur la taxinomie illustrée par la figure 17[24]. Les concepts *Caption* et *Face* sont spécialisés en plusieurs sous-concepts selon les résultats fournis par les outils d'analyse.

Dans le langage cinématographique et télévisuel, il est courant de décrire les plans où apparaissent un ou plusieurs personnages selon la distance apparente à laquelle ils se trouvent de la caméra, ce qui peut être exprimé par la taille relative du visage par rapport à la taille totale de l'image. On obtient ainsi une classification des plans qui utilise un vocabulaire professionnel établi. Les termes de ce vocabulaire sont appelés « valeurs de plan » (*shot values*). Ainsi, un *gros plan* (*Close-Up*) est un plan où un visage couvre à peu près la moitié de l'image, un *plan large* (*Long-Shot*) ou encore *plan en pied* un plan où le personnage filmé occupe approximativement la hauteur de l'écran. La terminologie en usage en France diffère légèrement de la terminologie anglo-saxonne. Dans l'expérimentation que nous présentons ici, nous employons cinq classes de valeurs de plans :

---

[24] Voir page 47

Close-Up *(CU)*	*Medium Close-Up* *(MCU)*	*Medium Shot* *(MS)*	*Medium-Long Shot* *(MLS)*	*Long Shot* *(LS)*

Ces termes sont en général définis de manière imprécise. Afin d'en fournir une définition quantitative, nous utilisons le fait que la distance subjective du sujet à la caméra varie inversement à la taille apparente de son visage sur l'écran. On calcule la valeur $d = \frac{\textit{taille du visage}}{\textit{taille de l'image}}$, et l'on définit cinq classes recouvrantes pour les cinq valeurs de plan, selon les valeurs possibles de d :

Valeurs de plans	*CU-Face*	*MCU-Face*	*MS-Face*	*MLS-Face*	*LS-Face*
Valeurs de d	$d \geq \frac{1}{4}$	$\frac{1}{6} \leq d \leq \frac{1}{2}$	$\frac{1}{8} \leq d \leq \frac{1}{4}$	$\frac{1}{10} \leq d \leq \frac{1}{8}$	$d \leq \frac{1}{8}$

Cette méthode permet de classifier les visages, un visage pouvant appartenir à une ou deux catégories. Il reste à classifier les plans. L'algorithme de segmentation en plans fournit une ou plusieurs images clé (*keyframe*) pour chaque plan. L'algorithme de détection de zones de visage est exécuté pour toutes les images clé, produisant une liste de régions de l'image. On a donc le schéma de composition suivant : des plans, qui contiennent des images clé, qui contiennent des zones de visage. Les liens de composition sont ici de type « partie-de», (*part-of* ou *part-whole*), et non de type « est-un » (*is-a*)[25]. Nous cherchons ici à propager les caractéristiques détectées sur les images clé vers les plans. Comme nous l'avons dit plus haut, la question générale de la manière dont les caractéristiques des parties se transmettent au tout est un problème difficile. Le système CLASSIC, et les logiques de descriptions en général, ne permettent pas de résoudre cette difficulté en toute généralité. Nous proposons ici une solution *ad-hoc* utilisant les fonctionnalités de CLASSIC pour exprimer au cas par cas les propriétés voulues. Pour cela, dans le cas des visages, nous représentons les relations *part-of* et les relations inverses *has-keyframe* et *has-face* par des rôles CLASSIC, comme l'illustre la figure 16.

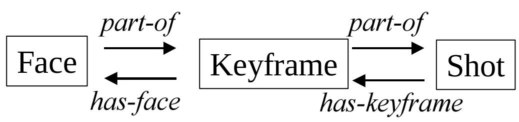


figure 16 : composition des plans, images clé et visages

Cinq classes d'images clé sont définies pour les cinq valeurs de plan : *CU-Keyframe*, *MCU-Keframe*, etc. Une image clé appartient à la classe *CU-Keyframe* (par exemple) si au moins un visage de la classe *CU-Face* a été détecté, et si tous les visages détectés dans l'image sont de la classe *CU-Face*. De la même manière, cinq classes de plans sont définies : *CU-Shot*, *MCU-Shot*, etc. Un plan est instance de *CU-Shot* si toutes ses images clé sont instances de *CU-Keyframe*.

Les définitions des concepts CLASSIC pour le cas des gros plans (*CU*) sont données ci-dessous.

---

[25] Voir « Hiérarchies « partie-de » » page 40.

```
(cl-define-concept 'CU-Face
  '(and
    Face
    (all d (min ,(/ 1 4)))))

(cl-define-concept 'CU-Keyframe
  '(and
    Keyframe
    (at-least 1 has-Face)
    (all has-Face CU-Face)))

(cl-define-concept 'CU-Shot
  '(and
    Shot
    (at-least 1 has-keyframe)
    (all has-keyframe CU-Keyframe)))
```

D'une manière similaire, des sous-concepts de *Keyframe* et *Shot* sont définis pour traiter du nombre de visages détectés. Nous distinguons quatre classes distinctes :

- Zéro visage
- Un visage
- Deux visages
- Trois visages et plus

Cela nous conduit à définir les concepts *Zero-Face-Keyframe*, *Zero-Face-Shot*, *One-Face-Keyframe*, *One-Face-Shot*, etc. Les conditions nécessaires et suffisantes d'appartenance d'une image clé à l'une de ces classes se fait cette fois-ci en restreignant la cardinalité du rôle *has-face*. Un plan à un visage est défini comme un plan dont *toutes* – qualification *universelle* – les images représentatives ne montrent qu'un unique visage. Cette définition ne prend pas en compte le cas particulier, hautement improbable dans le cadre de la télévision traditionnelle, où un même plan montrerait successivement plusieurs visages. Les concepts *One-Face-Keyframe* et *One-Face-Shot* sont définis par les expressions CLASSIC suivantes :

```
(cl-define-concept 'One-Face-Keyframe
  '(and
    Keyframe
    (at-least 1 has-face)
    (at-most 1 has-face)))

(cl-define-concept 'One-Face-Shot
  '(and
    Shot
    (all has-keyframe One-Face-Keyframe)))
```

On définit ensuite deux classes de plans (sous-concepts de *Shot*) en fonction du nombre d'images clé détectées. L'algorithme de segmentation en plans fournit une image clé par mouvement élémentaire de caméra (*pas* de mouvement, *zoom*, *traveling*, etc.). La détection d'une unique image clé indique un plan ayant un unique mouvement élémentaire de caméra. Nous appelons ces plans des plans *simples* (*Simple-Shot*) et les plans ayant au moins deux images clé des plans *composés* (*Composed-Shot*). L'expression CLASSIC suivante donne la définition du concept *SimpleShot* :

```
(cl-define-concept 'Simple-Shot
  '(and
    Shot
    (at-least 1 has-keyframe)
    (at-most 1 has-keyframe)))
```

Enfin, trois concepts primitifs sont définis pour différencier les incrustations de texte en fonction de leur position sur l'écran : en bas à gauche (*BL-caption*, pour *Bottom Left*), au centre à gauche (*CL-Caption*, pour *Center Left*) et en haut à gauche (*UL-Caption*, pour *Upper Left*). Ces concepts sont primitifs dans le sens où le test d'appartenance n'est pas réalisé au sein de CLASSIC et que les individus CLASSIC doivent être explicitement déclarés comme appartenant à l'un ou l'autre de ces concepts.

Nous cherchons alors à définir les concepts correspondant aux plans caractéristiques du journal de « TF1 » illustrés par la figure 15. Nous cherchons par exemple à définir un plan d'interview (*Interview-Shot*) comme un plan ou apparaît un seul visage (*One-Face-Shot*) en plan poitrine (*MCU-Shot*), et où une inscription de texte apparaît en bas à gauche de l'écran (*BL-Caption*). Le problème consiste ici à faire « remonter » l'information portée par les incrustations de texte de l'image clé vers le plan. Alors que les définitions précédentes étaient universelles (un gros plan est un plan dont *toutes* les images clé montrent des visages en gros plan), la définition est ici existentielle : on cherche un plan dont une image clé possède *au moins une* incrustation de texte (rôle *has-caption*) en bas à gauche de l'écran. CLASSIC ne permet pas d'exprimer directement ce type de construction. Nous contournons cette difficulté en exploitant la souplesse d'expression offerte par les règles CLASSIC[26]. On commence par déclarer les concepts *Personal-Caption*, *Personal-Keyframe* et *Personal-Shot* désignant respectivement une inscription de texte dénotant un nom de personne, une image clé contenant une telle inscription et un plan contenant une telle image clé. On définit alors les règles CLASSIC suivante :

```
(cl-add-rule
   'personal-caption-rule
   @Bottom-Caption
   'Personal-Caption)

(cl-add-description-rule
   'personal-rule
   @Personal-Caption
   'add-descriptions
   :args (list 'Personal-Keyframe 'Personal-Shot))
```

La première règle est déclenchée à l'instanciation du concept *Bottom-Caption*. L'action de cette règle est d'ajouter la description *Personal-Caption* à l'individu instancié. Cela revient donc à exprimer l'implication : *Bottom-Caption* ⇒ *Personal-Caption*.

La seconde règle est déclenchée à l'instanciation du concept *Personal-Caption*. L'action de cette règle est d'appeler la fonction lisp *add-descriptions* dont les arguments sont l'individu pour lequel la règle est déclenchée, ainsi que les descriptions de concept *Personal-Keyframe* et *Personal-Shot*. Cette fonction « remonte » de l'individu représentant l'incrustation de texte à

[26] Pour les règles CLASSIC, voir §Chapitre 24.1.1 page 33.

l'image clé, puis de l'image clé au plan, par l'intermédiaire du rôle *part-of*, et ajoute la description *Personal-Keyframe* à l'image clé et la description *Personal-Shot* au plan (voir le code de la fonction *add-descriptions* ci-dessous). Ainsi, lorsqu'une inscription de texte est instance du concept *Bottom-Caption*, elle est automatiquement classifiée en *Personal-Caption*, l'image clé à laquelle elle appartient est classifiée en *Personal-Keyframe* et le plan auquel cette image clé appartient est classifié en *Personal-Shot.*

```
defun add-descriptions (region-ind kf-desc shot-desc)
   (let* ((kf-role (cl-named-role 'part-of))
          (kf-ind  (car  (cl-ind-role-fillers  region-ind  kf-
role))))
      (when  kf-ind
         (cl-ind-add kf-ind kf-desc)
         (let* ((shot-role (cl-named-role 'part-of))
                (shot-ind   (car   (cl-ind-role-fillers   kf-ind
shot-role))))
            (when shot-ind
               (cl-ind-add shot-ind shot-desc)))))
   'Classic-thing)
```

Finalement, la définition d'un plan d'interview est la conjonction des concepts *Personal-Shot*, *MCU-Shot* et *One-Face-Shot.* La définition de ce concept est la suivante :

```
(cl-define-concept 'Interview-Shot
  '(and
    MCU-Shot
    Personal-Shot
    One-Face-Shot))
```

Ainsi, les plans auxquels appartiennent les images clé (1) à (3) de la figure 15 sont classifiés comme des plans d'interview, instances du concept *Interview-Shot.* Une méthode similaire permet de définir les plans de lieu comme étant des plans dont une image clé affiche une inscription de texte en haut et à gauche.

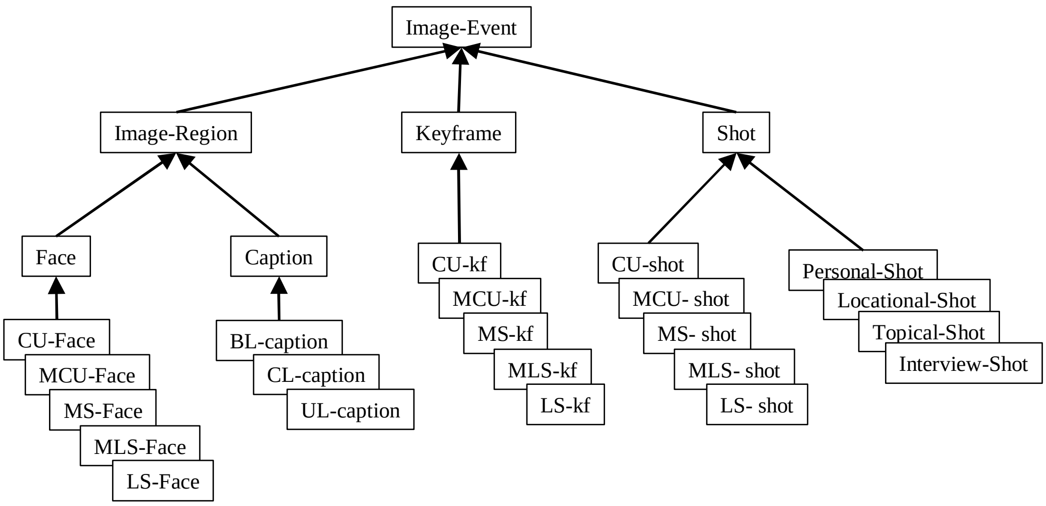


figure 17 : extrait d'une taxinomie d'événements « image »

6 Conclusion

Les logiques de descriptions forment une famille de langages bien formalisés permettant de représenter efficacement les connaissances d'ordre ontologique (relation « *est-un* »). Les systèmes existants, comme CLASSIC, offrent en outre des extensions permettant de dépasser ce cadre dans une mesure limitée. Il est possible, comme on l'a vu avec la classification des plans de journaux télévisés, d'exprimer des connaissances d'ordre méréologique (relation « *partie-de* ») et d'effectuer certaines inférences sur ces connaissances. Nous avons ainsi utilisé les règles CLASSIC pour classifier des plans de journaux télévisés. Nous exprimons par exemple que les inscriptions de texte d'une image clé ou les régions de l'image correspondant à un visage sont partagées par le plan dont l'image est extraite ; un plan d'interview peut alors être défini comme un plan dont une image clé contient un visage cadré en « plan moyen » et comportant une inscription de texte en bas à gauche de l'écran. Nous atteignons ainsi le but fixé au prix d'un effort de programmation. Cependant, les connaissances à forte composante temporelle restent hors de portée des logiques terminologiques proprement dites.

Nous cherchons alors un formalisme plus adapté à la représentation de ce type de connaissances. Dans le but de représenter et de reconnaître des classes de séquences audiovisuelles, nous proposons au chapitre suivant de reprendre et d'étendre le système T-Rex (Weida, 1992b), système réalisé dans le domaine de la reconnaissance de plans (*plan recognition*) et qui se présente comme une extension au domaine temporel d'un formalisme de logiques de descriptions.

Chapitre 3
Indexation vidéo comme problème de reconnaissance de plans

Nous nous intéressons à l'indexation de collections de documents audiovisuels partageant des caractéristiques communes identifiables, comme le ou les présentateurs, les décors, l'habillage visuel ou sonore, ou encore la structure temporelle. Beaucoup d'émissions de télévision traditionnelles appartiennent à ce qu'on appelle une collection, comme la plupart des journaux télévisés, certaines émissions d'actualité ou de variétés, des jeux, etc. A l'INA, une *fiche collection* décrit les éléments communs aux émissions de la collection, de manière relativement détaillée, et donne des consignes d'indexation aux documentalistes. Sans chercher à modéliser tous les éléments des fiches collection, nous cherchons à reconnaître dans une émission, à partir de segmentations initiales et de résultats de classification des segments obtenus, certains des éléments décrits par la fiche collection, et plus particulièrement les éléments de structure. Exprimé de cette manière, le problème que nous cherchons à résoudre s'apparente à un problème de reconnaissance de plans (au sens de *plan recognition*), où il s'agit de déterminer à partir des observations élémentaires du comportement d'un agent la ou les actions que l'agent est en train d'effectuer.

Notons qu'il est difficile dans ce domaine de ne pas introduire de confusion dans les termes employés ; certains parlent de « plan » à reconnaître (Kautz, 1991), d'autres de « scénario » (Dojat *et al.*, 1998), alors que, par exemple pour Weida, un scénario est un ensemble d'observations temporellement liées (un graphe d'observations), de même que pour (Van Beek, 1992) un scénario est une instanciation d'un graphe de contraintes temporelles. Les termes de « plan » et de « scénario » étant déjà employés dans le langage audiovisuel, nous avons choisi dans notre système d'utiliser le terme de « canevas », que nous traduisons par « template » en anglais. Nous continuerons cependant à employer les termes utilisés par les auteurs des travaux que nous citons, en tâchant de ne pas introduire d'ambiguïté. Nous noterons les observations en lettres minuscules – par exemple a_1 – et les plans à reconnaître en lettres capitales – par exemple P_1. Nous réserverons la notation T_i aux canevas.

Nous nous sommes intéressés au chapitre précédent aux logiques de descriptions pour la classification d'événements filmiques, et nous sommes arrivés à la conclusion que, schématiquement, cette famille de langages permettait de décrire et classer des plans (dans le sens filmique) mais non des séquences, pour lesquelles la dimension temporelle et les liens de type « partie de » (*part-of*) doivent pouvoir être exprimés. Le système T-Rex, proposé par Robert Weida (Weida, 1995), système dédié à la reconnaissance de plans, étend un formalisme de logique de descriptions au domaine temporel en employant des méthodes de satisfaction de contraintes. Nous reprenons ces travaux en étendant le langage de définition des plans de T-Rex. Dans ce chapitre, nous commencerons par présenter la nécessité du choix d'un formalisme de représentation et de raisonnement temporel pour la classification de séquences audiovisuelles. Après avoir décrit brièvement le contexte de la reconnaissance de plans, nous décrirons plus en détail le système T-Rex, nous en proposerons une implémentation simple utilisant le système de satisfaction de contraintes BackJava, puis nous présenterons les extensions que nous y avons apporté, ainsi que les problèmes nouveaux que ces extensions introduisent.

1 Raisonnement temporel

Il est nécessaire pour le problème que nous nous posons de prendre en compte la dimension temporelle des objets manipulés : plans, séquences, mouvements de caméra, répliques, éléments musicaux, etc. Nous cherchons de plus à prendre en compte les relations temporelles entre les éléments d'une séquence, en définissant par exemple un reportage comme étant une série de plans *commençant* par un ou plusieurs plans de studio et qui se termine juste avant le *prochain* plan de studio. Nous présentons tout d'abord un survol de travaux existants concernant la représentation du temps et le raisonnement temporel en nous inspirant pour cela de (Vila, 1994) et de (Ramaux, 1998).

1.1 Généralités

Prendre en compte une dimension temporelle est nécessaire dès lors qu'on veut raisonner sur des situations pouvant évoluer, sur des actions, sur des changements d'état, etc. Le temps est au centre de notre perception du monde, et nous manions au quotidien les concepts temporels de manière efficace et naturelle. Il s'avère cependant que la caractérisation et la formalisation du temps est une tâche difficile. De plus, il apparaît que des prises de position philosophiques aussi bien que des préoccupations formelles ou d'efficacité peuvent guider le choix d'un mode de représentation. On distingue ainsi classiquement deux statuts distincts accordés au temps, selon qu'on adopte une théorie *absolue* ou une théorie *relationnelle* du temps. Dans le premier cas, le temps a une existence propre, et les événements se produisent au cours du temps. Dans le second cas, le temps n'est que la conséquence du fait que des événements se déroulent et sont temporellement liés. Une approche *absolue modérée* opte pour une existence autonome du temps, tout en affirmant que nous ne percevons du temps que les relations temporelles entre les événements que nous observons. En Intelligence Artificielle, il semble que ce soit la théorie absolue, modérée ou non, qui soit le plus représentée (en particulier les logiques *réifiées*).

1.2 Quelques approches

Le calcul de situation (*situation calculus*) (MacCarthy, 1969) est l'une des premières tentatives de représenter le changement. Les concepts de base de ce calcul sont les *situations* et les *actions*. Une situation est définie comme la période de temps pendant laquelle une propriété est valide. Une action a pour conséquence un changements de situation : si la personne p réalise l'action A dans la situation S, il en découle une nouvelle situation, $result(p, A, S)$. Le calcul de situation se heurte au *frame problem* (MacCarthy, 1969) : comment discriminer parmi les états observables par un agent ceux qu'il doit prendre en compte ainsi que les changements d'états pertinents pour sa prise de décision. Des axiomes doivent donc être ajoutés, qui précisent en particulier les propriétés qui demeurent inchangées par la réalisation d'une action.

On oppose en général au calcul de situation le calcul d'événements (*event calculus*) (Kowalski, 1986), où les événements sont instantanés et peuvent débuter (prédicat *initiates*) ou terminer (prédicat *terminates*) une période de temps pendant laquelle une proposition est vraie. Ainsi, $initiates(e, u)$ signifie que la proposition u devient vraie à partir de l'événement e. Le prédicat *at*(*e, t*) associe l'événement *e* à l'instant *t*. Pour éviter le frame problem, une proposition est vraie jusqu'à ce qu'un événement la termine.

Les *logiques temporelles réifiées* construisent un méta-langage au-dessus d'un langage existant, les formules du langage initial devenant les propositions du nouveau langage. On peut ainsi définir un prédicat TRUE dont les arguments sont une expression atemporelle dans le langage initial et une expression dénotant un objet temporel :

TRUE<*expression atemporelle*, *expression temporelle*>

(Shoham, 1987) a défini formellement une logique réifiée où les expressions atemporelles sont réduites à des propositions atomiques. Selon l'auteur, cette logique est plus expressive que les logiques de Allen et McDermott (voir ci-dessous), mais la restriction des formules atemporelles à des formules atomiques ne permet pas d'exprimer des connaissances générales sur le temps, telles que la causalité ou la densité, par exemple.

A l'opposition calcul de situation/calcul d'événements s'ajoute une autre opposition sur l'*ontologie* temporelle, entre un temps représenté par des instants – ou points (McDermott, 1982) –, et un temps représenté par des intervalles (Allen, 1984). Pour McDermott, les points permettent de modéliser le temps comme une collection infinie et dense d'instants, ainsi que de définir une relation $\leq$ réflexive, anti-symétrique et transitive. Pour représenter la durée d'une action, un intervalle est décrit par deux points figurant ses bornes. Pour Allen, les intervalles représentent mieux notre perception du temps, en particulier pour parler de *propriétés* et *d'événements*. Allen définit un calcul d'intervalles, fondé sur treize relations temporelles correspondant aux relations possibles pouvant exister entre deux intervalles[27].

Une relation temporelle est identifiée par une disjonction de ces treize relations élémentaires. Dans l'algèbre d'intervalles complète (*full interval algebra*), c'est-à-dire dans l'algèbre qui autorise toutes les disjonctions, il n'existe pas d'algorithme complet en temps polynomial permettant de calculer la clôture d'une formule. (Van Beek, 1992) propose une algèbre d'intervalles réductibles à des points (*pointizable algebra*). Dans ce formalisme, on ne considère que les disjonctions de relations temporelles entre intervalles pouvant être exprimées par des conjonctions d'égalités et d'inégalités sur instants constituant les bornes des intervalles. Ainsi, la relation temporelle illustrée par la figure 18 :

$$A \ \{\mathit{before} \vee \mathit{meets} \vee \mathit{overlaps}\} B$$

se traduit-elle par :

$$\mathit{begin}(A) < \mathit{begin}(B) \wedge \mathit{end}(A) < \mathit{end}(B)$$

[27] Voir tableau 1 page 54.

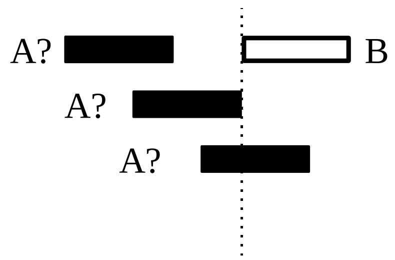


figure 18 : exemple de relation temporelle

Toutes les relations de l'algèbre complète ne sont pas représentables dans cette algèbre réduite. C'est en particulier le cas de la relation {*before* ∨ *after*}, illustrée par la figure figure 19. Il n'est pas possible d'exprimer cette disjonction de relations entre intervalles comme conjonction de relations entre les bornes de ces intervalles. Le sens intuitif de cette relation temporelle est : « l'un après l'autre », sans que l'ordre des intervalles soit précisé. Selon l'échelle à laquelle on se place – le plan ou la séquence –, le sens particulier qu'on peut donner à cette relation n'est pas le même. Il est en effet rare que l'ordre des plans soit indifférent, l'interprétation des plans étant très liée au contexte constitué des autres plans – c'est le fameux effet Koulechov. Cependant, lorsqu'on s'intéresse aux séquences, on peut être amené à vouloir exprimer ce genre de relation : « une chanson de Barbara, suivie ou précédée d'une interview ».

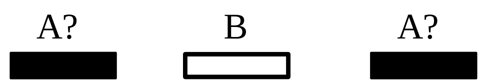


figure 19 : relation {before ∨ after}

Nebel et Bückert ont proposé dans (Nebel & Bückert, 1994) un modèle temporel calculable moins restrictif que celui de Van Beek, permettant de prendre en compte la relation {*before* ∨ *after*}.

1.3 Choix d'une structure temporelle

Nous adoptons la structure temporelle proposée par Allen (Allen, 1984). Il s'agit d'un modèle linéaire simple, sans possibilité de branchement, qui peut être figuré par un axe orienté infini. Allen propose une théorie du temps (*temporal theory*) où il existe un objet primitif : la *période*, qui correspond à un intervalle de temps, et une relation primitive : la relation *meets*[28]. Un système d'axiomes représente dans la théorie certaines intuitions communes sur la perception du temps, comme le fait que si deux périodes sont dans la relation *meets*, il existe une période qui est la concaténation des deux premières. Les deux premiers axiomes de la théorie de Allen sont décrits ci-dessous.

[28] L'usage est de conserver les termes anglais pour les relations temporelles de Allen (*meets*, *before*, *overlaps*, etc.).

1) Le temps n'est pas borné et il n'existe pas de période infinie ; en d'autres termes, il existe pour chaque période une période qui la précède au sens de la relation *meets*, et une période qui lui succède :

$$\forall i, \exists j, k \mid meets(j, i) \wedge meets(i, k) \tag{1}$$

2) Les périodes peuvent être composées. En particulier, deux périodes qui se succèdent peuvent être concaténées :

$$\forall i, j, k, l, meets(j, i) \wedge meets(j, k) \wedge meets(k, l) \Rightarrow \exists m \mid meets(i, m) \wedge meets(m, l) \tag{2}$$

Ces deux axiomes confèrent à cette structure du temps une sorte de propriété de *fermeture*. Quelles que soient les périodes de temps considérées, il existe une période qui est le résultat de leur composition. Cette propriété nous sera utile pour la définition des canevas.

Les deux axiomes suivants établissent que la relation *meets* définit des classes d'équivalence sur l'ensemble des périodes et que ces classes d'équivalence définissent à leur tour les périodes de manière unique : en particulier, si deux périodes *m* et *n* succèdent à la même période *i* et qu'une période *j* leur succède à toutes deux, alors *m* et *n* sont égales (voir figure 20).

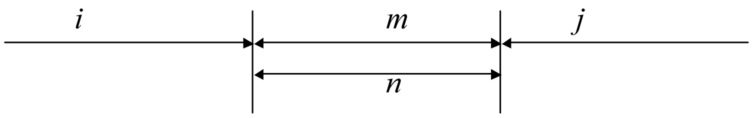


figure 20 : égalité de deux périodes

Les autres relations temporelles élémentaires (*before*, *overlaps*, …) sont définies à l'aide de ce système. Par exemple, les relations *before* et *starts* sont définies de la manière suivante (voir également la figure 21) :

$$before(i, j) \stackrel{\text{def}}{=} \exists m \mid meets(i, m) \wedge meets(m, j) \tag{3}$$

$$starts(i, j) \stackrel{\text{def}}{=} \exists l, m, n \mid meets(l, j) \wedge meets(l, i) \wedge meets(i, n) \wedge meets(j, m) \wedge meets(n, m) \tag{4}$$

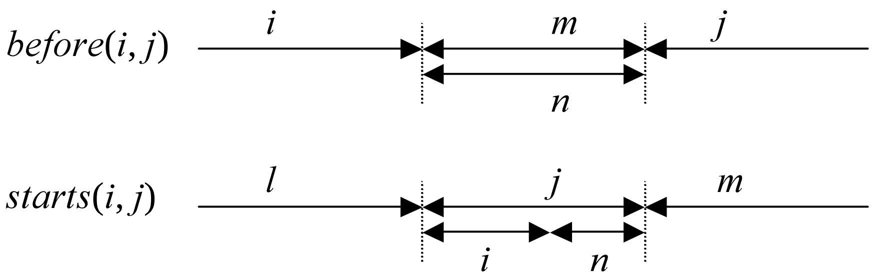


figure 21 : définition de *before* et *starts*

On définit ainsi les relations *before*, *during*, *starts*, *overlaps*, *finishes*. Les relations inverses sont également définies, ainsi que la relation d'égalité *equals*, ce qui permet d'obtenir les treize relations temporelles élémentaires correspondant aux treize positions relatives distinctes de deux intervalles sur un axe. Ces treize relations temporelles élémentaires sont illustrées par le tableau 1.

x, y	*x equals y*	[=, ou e]
x, y	*x before y* *y after* x	[b] [a]
x, y	*x meets y* *y is-met x*	[m] [mi]
x, y	*x during y* *y is-during x*	[d] [di]
y, x	*x overlaps y* *y is-overlapped x*	[o] [oi]
y, x	*x starts y* *y is-started x*	[s] [si]
y, x	*x finishes y* *y is-finished x*	[f] [fi]

tableau 1 : les treize relations temporelles élémentaires

1.3.1 Implémentation des relations temporelles

Une implémentation efficace des relations temporelles de Allen consiste à coder une disjonction de relations élémentaires par un entier non signé sur 13 bits en choisissant une position par relation élémentaire et en adoptant la convention que le $i^{ème}$ bit est positionné à 1 si la relation élémentaire est respectée, et à 0 sinon. Cette représentation offre l'avantage, outre le faible encombrement mémoire (deux octets par relation), de permettre d'effectuer certaines opérations algébriques en utilisant les opérateurs binaires. Les calculs se font ainsi très rapidement, et en temps constant. On note $\varnothing$ la relation la plus générale, disjonction de toutes les relations élémentaires. Ces opérations élémentaires sont regroupées dans le tableau 2.

Opération	**Notation**	**Implémentation**
Intersection	$r_1 \cap r_2$	« et » binaire (r_1 & r_2)
Union	$r_1 \cup r_2$	« ou » binaire (r_1 | r_2)
Compatibilité	$r_1 \sim r_2$	$(r_1 \,\&\, r_2) \neq 0$
Complément	$\neg r_1$	« xor » binaire (r_1 ^ ∅)
Subsomption	$r_1 \supset r_2$	$(r_1 \,\&\, r_2) = r_2$

tableau 2 : opérations booléennes pour les relations temporelles

Nous choisissons la convention suivante pour la représentation sous forme d'un vecteur de bits des relations élémentaires de Allen (tableau 3) :

Relation	**Représentation décimale**	**Représentation binaire**
equals	1	0000000000001
before	2	0000000000010
after	4	0000000000100
during	8	0000000001000
is-during	16	0000000010000
overlaps	32	0000000100000
is-overlapped	64	0000001000000
meets	128	0000010000000
is-met	256	0000100000000
starts	512	0001000000000
is-started	1024	0010000000000
finishes	2048	0100000000000
is-finished	4096	1000000000000

tableau 3 : convention de représentation des relations de Allen élémentaires

Un graphe de contraintes temporelles de N sommets peut être représenté par une matrice $N \times N$ d'entiers courts. Cette matrice est antisymétrique ; on a pour tous les sommets i et j :

- $r_{i,j} = \neg r_{j,i}$
- $r_{i,i} = equals$

Pour économiser de l'espace mémoire, on peut ne représenter explicitement qu'une des parties (haute ou basse) de la matrice, en renvoyant la relation symétrique $\neg r_{j,i}$ lorsque $r_{i,j}$ n'est pas représentée.

1.3.2 Calcul de la couverture minimale d'un graphe de contraintes

Le calcul de cohérence d'un graphe de contraintes temporelles est NP-complet (Vilain *et al.*, 1989). Allen propose dans (Allen, 1983) un algorithme de 3-cohérence (assurant la cohérence des chemins de longueur 3 dans le graphe de contraintes), algorithme cohérent mais non complet de minimisation du graphe de contraintes. Cet algorithme réduit les disjonctions de contraintes en supprimant certaines contraintes incohérentes, et par là-même détecte certaines configurations incohérentes du graphe. L'algorithme utilise une table de transition donnant pour deux relations r_1 et r_2 distinctes de *equals* la relation $r_1 \circ r_2$. Ainsi *before* o *before* = *before*, *overlaps* o *during* = $\{$*overlaps* $\vee$ *during* $\vee$ *starts*$\}$, *before* o *after* = $\varnothing$, etc.

L'algorithme de Allen est simple et largement utilisé. Bien que (Van Beek, 1992) propose un algorithme plus performant de 4-cohérence, nous utiliserons l'algorithme de Allen lorsque la propagation de contraintes temporelles sera nécessaire.

2 Reconnaissance de plans

Le problème de la reconnaissance de séquences caractéristiques dans un flux audiovisuel se pose comme un problème de reconnaissance de plans. La reconnaissance de plans – ou reconnaissance de scénarios ou de chroniques, selon les auteurs – vise à interpréter les actions élémentaires d'un agent en situation à partir d'une description partielle disponible *a priori* des comportements possibles de cet agent. Les champs d'application des techniques issues de la reconnaissance de plans sont entre autres la compréhension de dialogues en langue naturelle (Cohen & Perrault, 1986), la surveillance de patient assistée par ordinateur (Dojat *et al.*, 1998) ou la supervision de systèmes dynamiques (Ramaux, 1998).

(Dousson, 1994) présente le problème de la reconnaissance de plans de la manière suivante. Soient :

a) un ensemble *d'événements* datés, qui sont les résultats de l'observation de l'agent ;
b) un ensemble de contraintes sur le domaine ;
c) un ensemble de schémas temporels[29] définissant comment l'association de plusieurs événements peut produire un nouvel élément.

Reconnaître, c'est expliquer les événements a) à l'aide de l'ensemble des schémas c) tout en respectant les contraintes du domaine b).

Dominique Fontaine (Fontaine, 1996) distingue quatre étapes principales dans une tâche de reconnaissance de plans – de *scénarios*, selon la terminologie de l'auteur :

- *Opportunité*. Il s'agit de choisir le moment où confronter la base de scénarios aux observations ;
- *Filtrage*. Il s'agit d'une première sélection des scénarios à activer, qui permet de rejeter *a priori* les scénarios non pertinents ;

[29] Les plans à reconnaître

- *Activation.* Les scénarios filtrés deviennent *actifs*, c'est-à-dire susceptibles d'être reconnus. Ils redeviennent inactifs après la phase de discrimination ;
- *Discrimination.* C'est la phase de reconnaissance proprement dite. A la fin de cette phase, un scénario est soit *reconnu*, soit *compatible*, soit *rejeté*. Nous retrouverons ces trois notions lorsque nous parlerons du système T-Rex.

On distingue communément trois types de problèmes de reconnaissance de plans (Kautz, 1991) selon la conscience qu'a l'agent du plan qu'il est en train de suivre :

- l'agent connaît le plan qu'il suit et tente de s'y conformer (*intended plan recogntion*)
- l'agent ne le connaît pas ou y est indifférent (*keyhole plan recognition*)
- l'agent le connaît et tente de dissimuler ses intentions à l'observateur (*obstructed plan recognition*).

Lorsque les actions observées sont celles d'un système physique – ventilateur, système dynamique –, c'est en général la seconde hypothèse (*keyhole plan recognition*) qui prévaut. Nous retiendrons également cette hypothèse pour la reconnaissance de séquences vidéo.

Afin de pouvoir raisonner sur les actions d'un agent, un certain nombre d'hypothèses doivent être explicitées sur la nature du monde dans lequel l'agent évolue. Ainsi Kautz, dans sa formalisation logique du problème de la reconnaissance de plans (Kautz, 1991), fait les hypothèses suivantes :

1. Hypothèse d'exhaustivité (*exhaustiveness assumption*) : toutes les actions possibles de l'agent sont connues. Autrement dit, la bibliothèque de plans est complète.

2. Hypothèse de disjonction (*disjointness assumption*) : deux types d'événement sont disjoints (*i.e.* ils ont une intersection vide), à moins que l'un spécialise l'autre ou qu'il existe un type d'événement spécialisé par les deux.

3. Hypothèse de composition/utilisation (*composition/use assumption*) : la réalisation d'une action ou d'un plan implique la réalisation d'au moins un des plans ayant cette action ou ce plan comme composant.

4. Hypothèse de cardinalité minimum (*minimum cardinality assumption*) : lorsqu'il existe plusieurs manières d'interpréter les actions d'un agent, la manière la plus probable est celle faisant intervenir le moins de plans *finaux* (*i.e.* les plans qu'on cherche à reconnaître). Cette hypothèse part de la supposition que les actions d'un agent ont de fortes chances d'être corrélées.

Ces hypothèses constituent des hypothèses de monde clos qu'il est souvent nécessaire de faire pour autoriser certaines inférences, en particulier les inférences faisant intervenir une négation. Le système T-Rex que nous présentons plus loin reprend les mêmes hypothèses.

Un autre aspect important en reconnaissance de plans concerne la nature temporelle des événements et des actions. Il s'agit de choisir une ontologie des objets temporels. Dans le système IxTet (Dousson, 1994), les observations sont des *événements* au sens de *l'event calculus*

(Kowalski, 1986) qui indiquent le changement d'état d'un *attribut* prenant ses valeurs parmi un ensemble discret (comme {*on*, *off*} ou {*chaud*, *froid*, *tiède*}). Les événements sont :

- *Instantanés*. Ils correspondent à un changement d'état et ont une durée nulle ;
- *Datés*. Ils peuvent toutefois être datés de manière imprécise (« entre 13h et 14h ») ;
- L'ensemble des événements est *complet*. Tous les changements d'état ont été observés et enregistrés. Cette hypothèse de complétude est forte mais souvent nécessaire pour éviter une explosion combinatoire ;
- L'ensemble des événements est *cohérent*, c'est-à-dire qu'il vérifie les contraintes propres au domaine.

Le système SADE (Ramaux, 1998) reprend également les concepts de l'*event calculus* auquel est ajouté le concept de *mesure*, notion plus faible que l'événement dans le sens que l'hypothèse de persistance ne tient pas – la valeur d'une propriété ne peut être considérée comme stable entre deux mesures. Dans le système T-Rex que nous présentons dans la section suivante, la logique temporelle sous-jacente est la logique temporelle complète d'intervalles.

3 Le système T-Rex

Nous présentons ici le système T-Rex (Weida, 1992b; Weida, 1995) proposé par Robert Weida et sur lequel nous avons en grande partie fondé notre travail. Le champ d'application est ici celui de la reconnaissance de plans et Weida s'inspire des premiers travaux de Henry Kautz dans ce domaine (Kautz, 1991). La reconnaissance de plan (*plan recognition*), dans ce cadre, consiste à tenter de déterminer parmi un ensemble de descriptions génériques d'actions, appelées *plans*, celui ou ceux qui sont effectivement réalisés par un agent dont on observe les actions.

3.1 Description

T-Rex intègre raisonnement terminologique et contraintes temporelles pour construire une bibliothèque de plans organisée en taxinomie. Le principe y est de considérer les plans comme des concepts au sens des logiques de descriptions et les observations comme des instances. Les observations, ou actions élémentaires, sont décrites par une taxinomie de concepts d'actions représentée dans le système de logiques de descriptions K-Rep (Weida *et al.*, 1996). Un *scénario* est une série d'observations ; reconnaître qu'un scénario s réalise un plan P est interprété comme vérifier que s est une instance de P. Un scénario est ainsi un ensemble d'instances de concepts d'action défini dans K-Rep, et chaque instance peut être classée individuellement par le seul mécanisme de classification terminologique de K-Rep. Un plan est représenté dans T-Rex par un graphe de contraintes temporelles dont chaque nœud est associé à un concept d'action élémentaire et dont les arêtes sont des disjonctions de contraintes temporelles de Allen. On appellera généralement *composant* les étapes devant être respectées pour qu'un plan donné soit reconnu, lesquels composants induisent chacun un sommet (ou nœud) du graphe de contraintes.

L'exemple qui suit définit un plan[30] d'interview du journal télévisé « Soir 3 » de la chaîne « France 3 ». Un plan d'interview est un plan fixe d'un unique personnage cadré en plan moyen (*medium close-up*) *pendant* lequel apparaît en incrustation le nom du personnage filmé et *pendant* lequel la voix de l'intervenant est entendue. Une image représentative d'un tel plan est donnée par la figure 22.

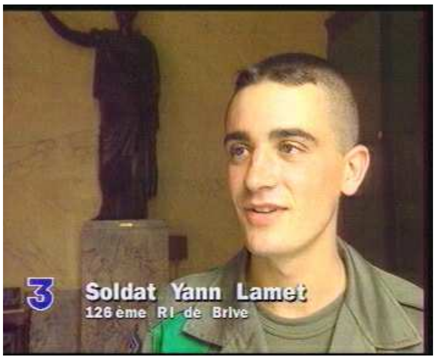


figure 22 : image représentative d'un plan d'interview

Pour les besoins de l'exemple, nous supposons que les processus suivants ont été mis en œuvre :

1. segmentation en plans ;
2. détection de zones de visages sur les images représentatives des plans ;
3. détection de zones de texte sur le flux vidéo intégral ;
4. détection de segments de parole.

Ces processus permettent d'instancier les concepts élémentaires de la taxinomie illustrée par la figure 23. Le processus de segmentation en plans crée des instances de la classe *Shot*, instances qui peuvent le cas échéant être classifiées comme instances de *MCU Shot* par le processus de détection de zones de visage[31]. Le processus de détection de zones de texte crée des instances de *Text Inscription* et le processus de détection de segments de parole crée des instances de *Speech*. Finalement, nous obtenons donc trois strates d'instances temporellement indépendantes : les plans (*Shot*), qui peuvent être ou non spécialisés en *MCU Shot*, les zones de texte (*Text Inscription*) et les segments de parole (*Speech*).

---

[30] « Plan » au sens cinématographique

[31] Voir « Classification de plans avec CLASSIC » page 42 pour la définition de ces concepts au sein d'un système de logique de descriptions.

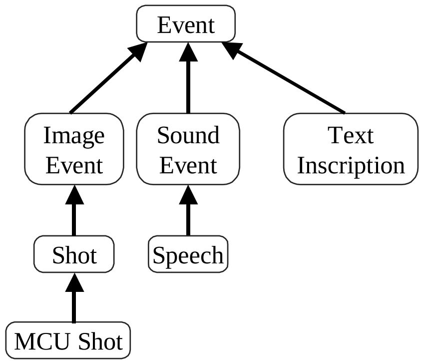


figure 23 : taxinomie utilisée pour définir un plan d'interview

Un plan d'interview (*Interview Shot*) est alors défini comme un graphe temporel comportant trois nœuds, le premier associé au concept *MCU Shot*, le second au concept *Text Inscription* et le dernier au concept *Speech* (voir figure 24). Les contraintes temporelles spécifient que la zone de texte doit être *pendant* la durée du plan, au sens large, c'est-à-dire au début (*starts*), à la fin (*finishes*), pendant au sens strict (*during*) ou sur toute sa durée (*equals*), et que le segment de parole doit avoir une intersection temporelle non vide avec le plan, ce qui se traduit par la disjonction de relations de Allen {*equals* ∨ *starts* ∨ *is-started* ∨ *finishes* ∨ *is-finished* ∨ *overlaps* ∨ *is-overlapped* ∨ *during is-during*}.

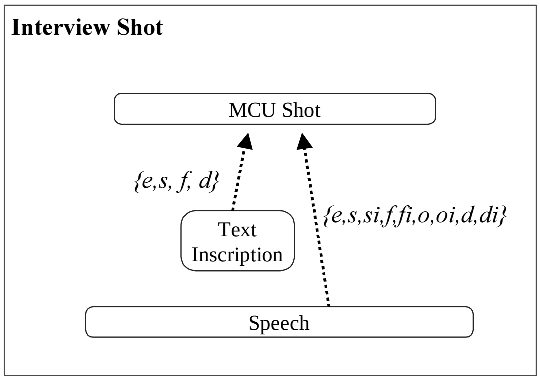


figure 24 : définition d'un plan d'inteview

Dans la syntaxe lispienne utilisée par Weida, un plan d'interview se définit par l'expression suivante :

```
(defplan interview-shot
   ((s1 MCU-SHOT)
    (s2 TEXT-INSCRIPTION)
    (s3 SPEECH))
   ((s2 (equals starts finishes during) s1)
    (s3 (equals starts etc.) s1))
```

Notons que, en plus de la spécification de contraintes temporelles entre les nœuds des graphes temporels représentatifs des plans, le système T-Rex offre une facilité d'écriture (macro) en autorisant l'imbrication de plans, c'est-à-dire en permettant d'associer à un nœud du graphe représentant le plan non un concept d'action mais un autre plan, et de spécifier des contraintes temporelles portant sur les nœuds terminaux du sous-plan. Ainsi, sur les plans illustrés par la figure 25 et tirés de (Weida, 1992a), qui reprend l'exemple maintenant classique de la cuisson des pâtes proposé par (Kautz, 1991), la définition du plan ASSEMBLE-SPAGHETTI-MARINARA spécifie une contrainte {*before* ∨ *meets*} entre un composant terminal C-BOIL et le composant terminal C-BOIL d'un plan imbriqué BOIL-SPAGHETTI.

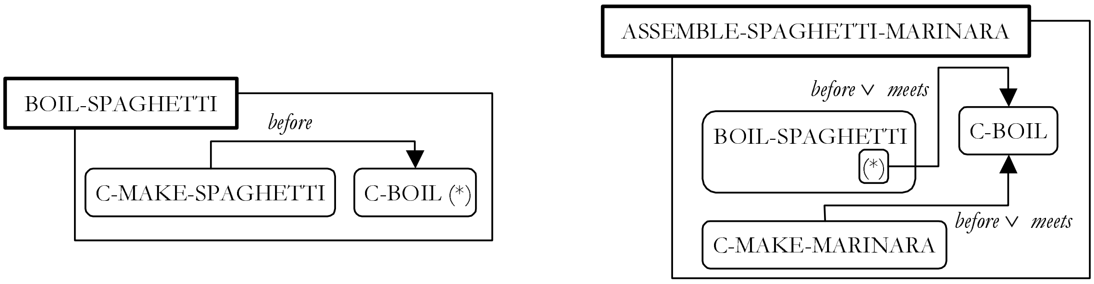


figure 25 : imbrication de plans dans T-Rex

Pour pouvoir organiser les plans en taxinomie selon leurs liens de généralité, la relation de subsomption est étendue aux plans. Cette relation repose d'une part sur la relation de subsomption définie par K-Rep entre les concepts d'action et d'autre part sur une relation de subsomption définie entre les relations temporelles. Cette dernière relation est simplement définie comme l'inclusion ensembliste. Soient r_1 et r_2 deux relations temporelles, disjonctions de relations temporelles élémentaires ; r_1 subsume temporellement r_2 si, et seulement si, l'ensemble des relations élémentaires de r_2 est un sous-ensemble de l'ensemble des relations élémentaires de r_1. Cette définition respecte l'intuition : la relation temporelle r_1 = {*meets* ∨ *before* ∨ *overlaps*} est plus spécifique que la relation temporelle r_2 = {*meets* ∨ *overlaps*} puisque tous les individus mis en relation par r_2 le sont nécessairement par r_1. A partir de ces deux relations de subsomption terminologique et de subsomption temporelle, une relation de subsomption entre plans est définie classiquement de manière extensionnelle : P_1 subsume P_2 si, et seulement si, toute instance de P_2 est nécessairement instance de P_1. D'une manière analogue à la subsomption terminologique, la définition est extensionnelle mais la procédure de vérification est intensionnelle, c'est-à-dire qu'elle repose sur la structure des plans. Cette procédure repose sur la possibilité de trouver un appariement (*matching*) entre le graphe de contraintes représentant le plan subsumant P_1 et le graphe représentant le plan subsumé P_2. Un tel appariement, appelé appariement de subsomption (*subsumption matching*) associe à chaque nœud M_i de P_1 un nœud N_j de P_2 tel que :

- l'appariement est injectif, c'est-à-dire que tous les nœuds de P_1 sont appariés à des nœuds distincts de P_2 ;
- le concept associé à un nœud de P_1 subsume « terminologiquement » le concept associé au nœud apparié dans P_2 ;
- la relation temporelle imposée entre deux nœuds de P_1 subsume temporellement la relation temporelle imposée entre les nœuds appariés de P_2.

Finalement, le plan P_1 subsume le plan P_2 si, et seulement si, il existe un appariement de subsomption de P_1 vers P_2. La procédure du calcul de subsomption, telle qu'elle est présentée par Weida, comporte quatre étapes :

- développement récursif des plans imbriqués (macros) et propagation des contraintes temporelles portant sur les nœuds des sous-plans ;
- Clôture du graphe de contraintes temporelles ;
- Analyse préliminaire : associer à chaque nœud N du graphe subsumant la liste des nœuds M_i du graphe subsumé tels que le concept associé à chacun des nœuds M_i soit subsumé par le concept associé au nœud N. Si l'une de ces listes est vide, c'est qu'il n'y a pas d'appariement possible et que, par conséquent, la subsomption n'est pas vérifiée ;
- Appariement par énumération (*backtracking*).

Le processus de reconnaissance est fondé sur le même mécanisme : un plan est réalisé si, et seulement si, il subsume les observations. Dans T-Rex, les relations temporelles entre les observations sont symboliques, comme le sont les relations entre les composants d'un plan ; ces relations peuvent également être des disjonctions de relations temporelles, de manière à pouvoir exprimer une incertitude quant aux dates des observations. La représentation d'un plan et la représentation d'un graphe d'observations sont similaires, et le mécanisme permettant de tester qu'un plan est réalisé, c'est-à-dire qu'il subsume les observations, est le même que le mécanisme permettant de tester la subsomption entre deux plans. Notons que, comme le fait remarquer Weida, cette procédure est complète dans la mesure où la clôture du graphe de contraintes temporelles est complète. De plus, Weida montre que ce problème d'appariement de graphe est NP-Complet.

3.2 Processus de reconnaissance de T-Rex

Les *scénarios*, c'est-à-dire les instances observées des actions élémentaires ainsi que leurs relations temporelles, sont représentés dans T-Rex par des graphes d'observation (*observation networks*), d'une manière similaire à la façon dont sont représentés les plans, chaque nœud du graphe étant associé à une instance et non à un concept, les arêtes du graphe étant les relations temporelles observées entre les instances des actions élémentaires.

Le processus de reconnaissance du système T-Rex consiste à assigner à chacun des plans de la taxinomie de plans l'une des modalités *nécessaire*, *possible* ou *impossible*. Un plan est nécessaire si, et seulement si, il subsume les observations. Un plan est possible si de nouvelles observations peuvent le conduire, sous certaines hypothèses, à subsumer les observations ; on parle alors de subsomption « potentielle » (*potential subsumption*). Un plan est impossible si il n'est ni nécessaire ni possible, c'est-à-dire si il est en contradiction avec les observations, sous les mêmes hypothèses.

Dans un premier temps, l'hypothèse est faite que les observations sont parfaites (*perfect observations*). Cela signifie d'une part que les observations sont toujours instances de concepts d'action terminaux, soit, en d'autres termes, qu'une observation ne peut pas être spécialisée par de nouvelles observations et d'autre part que les relations temporelles entre les observations ne peuvent elles non plus être spécialisées après coup. Le réseau d'observations n'évolue alors que par l'adjonction de nouvelles observations, elles aussi parfaites. Sous cette hypothèse, un appariement inverse de subsomption (*inverse subsumption mapping*) est défini. La définition de ce nouvel appariement est identique à celle de l'appariement de subsomption à ceci près qu'il s'agit d'un appariement du plan subsumé vers le plan subsumant, et non du plan subsumant vers le plan subsumé. Un tel appariement, associe à chaque nœud N_i du graphe d'observations O un nœud M_j du graphe de contraintes associé au plan P tel que :

- l'appariement est injectif, c'est-à-dire que tous les nœuds de O sont appariés à des nœuds distincts de P ;
- le concept associé à un nœud de O est subsumé terminologiquement par le concept associé au nœud apparié dans P ;
- la relation temporelle observée entre deux nœuds de O est subsumée temporellement par la relation temporelle imposée entre les nœuds appariés de P.

Les deux types d'appariement sont illustrés par la figure 26, où les nœuds du graphe de contraintes associés au plan sont notés M_i et les nœuds du graphe d'observation N_i. S'il existe un appariement de subsomption inverse d'un plan vers le graphe d'observations, l'ajout de nouvelles observations pourrait permettre de trouver un appariement de subsomption du plan vers les observations. Dans ce cas, le plan P est dit « directement optionnel » (*directly optional*). Enfin, un plan est dit « indirectement optionnel » si il subsume un plan directement optionnel. Il est utile de traiter ce cas à part, car il se peut qu'il n'existe pas d'appariement inverse de subsomption entre les observations et un tel plan. Un plan P est donc possible sous l'hypothèse de perfection des observations si, et seulement si, 1) il subsume les observations, ou 2) il existe un appariement inverse de subsomption des observations vers P, ou 3) P subsume un plan P' qui est lui-même possible.

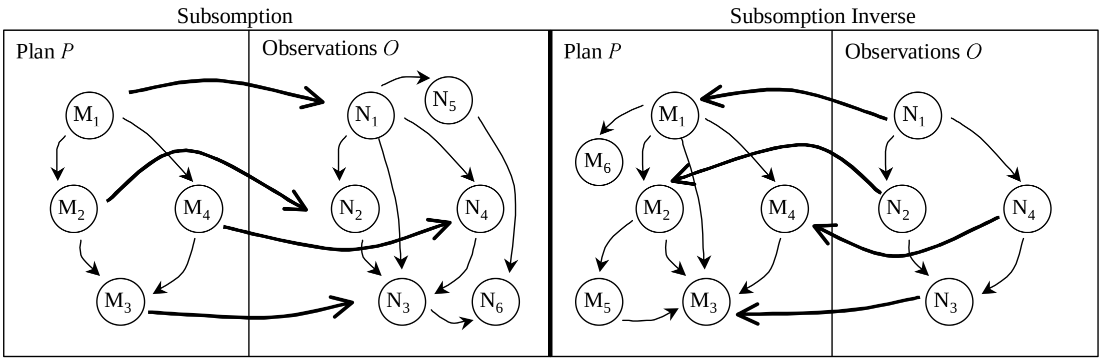


figure 26 : subsomption et subsomption inverse

Dans un second temps, l'hypothèse faite quant à la perfection des observations est relâchée au profit d'une hypothèse moins forte de monotonie des observations : les observations et les relations temporelles entre les observations peuvent alors être spécialisées après avoir été observées. Les observations faites ne sont donc jamais remises en cause, mais elles peuvent être raffinées. La relation qui entre alors principalement en jeu n'est plus la relation de subsomption mais la relation de compatibilité : deux concepts C_1 et C_2 sont compatibles s'il existe un concept C_3 subsumé à la fois par C_1 et par C_2. De même, deux relations temporelles r_1 et r_2 sont compatibles s'il existe une relation temporelle r_3 subsumée à la fois par r_1 et r_2 soit, autrement dit, si l'intersection de r_1 et r_2 n'est pas vide. De manière analogue à la définition d'un appariement inverse de subsomption, un appariement de compatibilité est défini entre les observations et un plan. Un tel appariement associe à chaque nœud N_i du graphe d'observations O un nœud M_j du graphe de contraintes associé au plan P tel que :

- l'appariement est injectif, c'est-à-dire que tous les nœuds de O sont appariés à des nœuds distincts de P ;
- le concept associé à un nœud de O et le concept associé au nœud apparié dans P sont compatibles ;
- la relation temporelle observée entre deux nœuds de O est compatbile avec la relation temporelle imposée entre les nœuds appariés de P.

Un plan est alors possible sous l'hypothèse de monotonie des observations si, et seulement si, 1) il subsume les observations, ou 2) il existe un appariement de compatibilité des observations vers P, ou 3) P subsume un plan P' qui est lui-même possible.

3.3 Une sémantique pour T-Rex

Le système T-Rex propose une extension temporelle d'un système de logique de descriptions par l'ajout d'une construction syntaxique permettant de décrire des plans comme des graphes de contraintes temporelles, ainsi qu'une procédure de calcul de la subsomption entre plans et de la réalisation d'un plan par des actions observées. Cependant, comme le font remarquer (Artale & Franconi, 1999) et comme le note Weida dans (Weida, 1992a), aucune sémantique formelle n'est associée à la description des plans.

Nous proposons dans les lignes qui suivent une piste pour définir une sémantique permettant une interprétation plus formelle de ce que représente un plan dans le système T-Rex. Notre propos n'est pas d'utiliser cette sémantique pour démontrer des propriétés théoriques de T-Rex ou pour en souligner les limitations, mais plutôt de présenter un cadre formel sur lequel nous pourrons asseoir les extensions et les modifications que nous avons apportées à T-Rex, modifications et extensions que nous présenterons plus loin[32]. La sémantique que nous décrivons ne fait que décrire de manière formelle les intuitions qu'on peut avoir sur ce que représente un plan et sur ce que signifie la subsomption entre deux plans.

De manière classique, et comme c'est le cas pour les logiques de descriptions[33], nous proposons une sémantique extensionnelle fondée sur la donnée d'une interprétation constituée d'un certain nombre d'ensembles, et de fonctions définies sur ces ensembles. Nous distinguons trois parties : la structure temporelle, le langage terminologique sous-

[32] Voir en particulier « T-Rex revisité » page 75.

[33] Voir « Sémantique » page 30.

jacent, qui possède sa sémantique propre et que nous ne détaillerons pas, et la définition des plans.

La structure temporelle est identique à celle proposée par Allen (Allen & Ferguson, 1994)[34]. Nous adoptons la même notation $T^*_\leq$ que Schmiedel (Schmiedel, 1990) pour cette structure. $T^*_\leq$ est un ensemble de périodes (ou intervalles de temps). On note A l'ensemble des symboles correspondant aux 13 relations de Allen élémentaires {*equals*, *before*, *after*, etc.}. Les relations temporelles sont définies par extension comme des ensembles de couples d'éléments de $T^*_\leq$. On note $.^{\alpha^*}$ la fonction qui associe à chaque élément de A un ensemble de couples éléments de $T^*_\leq \times T^*_\leq$, tel que les propriétés des relations soient respectées. Ainsi, par exemple, si $\{x,y\} \in meets^{\alpha^*}$ et $\{y,z\} \in meets^{\alpha^*}$, alors $\{x,z\} \in before^{\alpha^*}$. La fonction $.^{\alpha}$ associe quant à elle à un ensemble d'éléments de A représentant une disjonction de relations temporelles élémentaires l'union des images des éléments de la disjonction par $.^{\alpha^*}$. Ces éléments sont repris dans le tableau 4.

$T^*_\leq$	Structure temporelle
A	Ensemble des symboles des relations de Allen élémentaires : {*equals, before, after, meets, is-met, during, is-during, ovlerlaps, is-overlapped, starts, is-started*}.
$.^{\alpha*} : A \to 2^{T^*_\leq \times T^*_\leq}$	Fonction d'interprétation temporelle élémentaire, qui associe à chaque relation temporelle élémentaire un ensemble de couples de périodes tel que les propriétés des relations soient respectées.
$.^{\alpha} : 2^A \to 2^{T^*_\leq \times T^*_\leq}$ $\bigcup_{i=1}^{n} r_i \mapsto \bigcup_{i=1}^{n} r_i^{\alpha*}$	Fonction temporelle, qui associe à chaque disjonction de relations temporelles élémentaires un ensemble de couples de périodes, union des images des éléments de la disjonction par $.^{\alpha^*}$.

tableau 4 : interprétation de la structure temporelle

Le langage terminologique permettant de décrire les concepts d'action et d'instancier ces concepts est une logique de descriptions munie de sa propre sémantique, selon les constructeurs choisis. Nous considérons l'ensemble C des définitions de concepts de ce langage et une interprétation $(\Delta^I, .^I)$ du langage terminologique, où Δ^I est le domaine d'interprétation et $.^I$ la fonction d'interprétation qui associe à chaque description de C un sous-ensemble de Δ^I . Ces éléments sont repris dans le tableau 5.

C	Ensemble des définitions de concepts du langage terminologique.

[34] Voir aussi « Choix d'une structure temporelle » page 52.

Δ_I	Domaine d'interprétation
$.^I : C \rightarrow 2^{\Delta_I}$	Fonction d'interprétation du langage terminologique sous-jacent qui à chaque élément de *C* associe un sous-ensemble du domaine d'interprétation

tableau 5 : interprétation du langage terminologique

Les actions élémentaires – ou observations – sont des objets temporels, c'est-à-dire des éléments du domaine d'interprétation temporellement liés entre eux. Dans T-Rex, les relations temporelles entre les observations peuvent être des disjonctions de relations élémentaires. Chaque observation est par conséquent associée à un ensemble de périodes temporelles, et non à une période unique, ce qui serait le cas si seules les relations élémentaires étaient autorisées – ou si les observations étaient précisément datées, ce qui revient au même. La fonction γ associe à chaque instance des concepts d'actions élémentaires un sous-ensemble de $T^*_\leq$ qui sont les périodes temporelles possibles pour l'instance considérée.

Un plan *P* est représenté par la donnée d'un ensemble *X* de variables $x_1, \ldots, x_n$ représentant les nœuds du graphe de contraintes associé à *P*, par la donnée d'une fonction f_c qui associe à chaque variable de *X* un élément de *C*, ensemble des descriptions des concepts d'actions élémentaires dans le langage terminologique et par la donnée d'une fonction f_μ qui représente les contraintes temporelles du plan, et qui associe à chaque couple de variables de *X* un sous-ensemble de A, ensemble des symboles représentant les relations temporelles élémentaires. Ces éléments de définition des plans sont repris ci-dessous dans le tableau 6.

$\gamma : \Delta_I \rightarrow 2^{T^*_\leq}$	Fonction qui à chaque instance des concepts d'action élémentaire associe un ensemble de périodes de la structure temporelle $T^*_\leq$.
X	Ensemble de *n* variables $x_1, \ldots, x_n$ qui représentent les sommets du graphe associé au plan.
$f_c : X \rightarrow C$	Fonction qui à chaque variable de *X* associe une description dans le langage terminologique.
$f_\mu : X \times X \rightarrow 2^A$	Fonction qui associe une disjonction de relations temporelles élémentaires à tous les couples de variables. Ces relations sont les contraintes temporelles imposées entre les composants du plan.

tableau 6 : définition d'un plan

De manière classique, on associe une sémantique aux définitions de plans en interprétant chacun des plans comme un sous-ensemble du domaine d'interprétation Δ_I. Une interprétation consiste à définir l'extension d'un plan comme l'ensemble des éléments du domaine d'interprétation qui sont tels que les contraintes imposées par les fonctions

f_c et f_μ soient respectées. De manière informelle, c'est l'ensemble des éléments du domaine d'interprétation tels que :

- Tous les composants sont distincts (toutes les variables libres du plan sont associées à des individus distincts du domaine d'interprétation) ;
- Chaque composant du plan est instance du concept imposé par la fonction f_c ;
- Les contraintes temporelles entre les composants du plan sont respectées.

De manière plus formelle, une interprétation ε est la donnée d'un domaine d'interprétation Δ_I, d'un ensemble de définitions de concepts terminologiques C, d'une structure temporelle $T^*_\leq$, de l'ensemble de relations temporelles $\boldsymbol{A}$, ainsi que des fonctions $.^{\alpha*}$, $.^{\alpha}$, $.^{I}$ définies ci-dessus, l'interprétation $P\langle X, f_c, f_\mu\rangle^\varepsilon$ d'un plan $P\langle X, f_c, f_\mu\rangle$ étant définie par :

$$
\begin{aligned}
P\langle X, f_c, f_\mu\rangle^\varepsilon = \{\ a \in \Delta_I \mid \\
& x_i^\varepsilon \in \Delta_I, \forall x_i \in X \\
& x_i \neq x_j \Rightarrow x_i^\varepsilon \neq x_j^\varepsilon, \forall x_i, x_j \in X^2 \\
& x_i^\varepsilon \in f_c(x_i)^I, \forall x_i \in X \\
& (\gamma(x_i^\varepsilon), \gamma(x_j^\varepsilon)) \subseteq f_\mu(x_i, x_j)^\alpha, \forall x_i, x_j \in X^2 \}
\end{aligned}
$$

3.4 Subsomption

On définit la subsomption entre deux plans de manière extensionnelle, comme elle est définie pour le langage terminologique utilisé : le plan *P* *subsume* le plan *P'* (noté $P \succ P'$) si et seulement toute instance de *P'* est nécessairement instance de *P*, ou, en d'autres termes, si l'extension de *P* contient nécessairement l'extension de *P'*. Plus formellement, *P* subsume *P'* si, et seulement si, $P'^\varepsilon \subseteq P^\varepsilon$ pour toute interprétation ε. La relation de subsomption permet d'organiser les définitions de plans selon leur degré de généralité. On obtient ainsi une taxinomie de définitions de plans.

En reprenant Weida, on définit formellement la subsomption de plans à l'aide d'une fonction d'appariement de subsomption (*subsumption matching*). L'appariement de subsomption ***M*** d'un plan $P\langle X, f_c, f_\mu\rangle$ vers un plan $P'\langle X', f'_c, f'_\mu\rangle$ est une fonction de l'ensemble *X* des variables libres de *P* vers l'ensemble *X'* des variables libres de *P'*, telle que :

$M : X \rightarrow X'$	La fonction *M* est un appariement de subsomption si, et seulement si, • *M* est injective (chaque élément de X a un correspondant dans X' et chacun de ces

	correspondants n'a qu'un seul antécédent dans X). • Le concept terminologique associé au composant x subsume le concept associé au composant de P' correspondant. soit : $\forall x \in X, f_c(x) \succ f_c(M(x))$ • La contrainte temporelle entre les composants x_1 et x_2 est plus générale que la contrainte temporelle entre les composants correspondants de P' soit : $\forall x_1, x_2 \in X^2, f_\mu(x_1, x_2) \sqsupseteq f_\mu(M(x_1), M(x_2))$

Finalement, un plan $P\langle X, f_c, f_\mu \rangle$ subsume un plan $P'\langle X', f_c', f_\mu' \rangle$ si, et seulement si, il existe un appariement de subsomption de *P* vers *P'*.

La démonstration de cette proposition reprend celle proposée par (Weida, 1992a).

Dans le sens **si** : la proposition est évidente, par construction. S'il existe un appariement de *P* vers *P'*, alors toute instance de *P'* est instance de *P*.

Dans le sens **seulement si** : on cherche à montrer que si *P* subsume *P'*, alors il existe un appariement de subsomption de *P* vers *P'*. Pour cela, on raisonne par l'absurde et l'on montre que s'il n'existe pas d'appariement de subsomption de *P* vers *P'*, alors *P* ne subsume pas *P'*, c'est-à-dire qu'il est possible de construire une interprétation où il existe une instance de *P'* qui n'est pas instance de *P*. En effet, s'il n'existe pas d'appariement de *P* vers *P'*, c'est que 1°) soit il existe un composant de *P* qui n'a pas de correspondant dans *P'*, 2°) soit chaque sommet du graphe de contraintes temporelles *G* associé à *P* possède bien un correspondant dans *G'*, graphe de contraintes temporelles associé à *P'*, mais il existe au moins un arc *a* de *G* tel que la contrainte de l'arc de *G'* correspondant à *a* ne soit pas associée à une contrainte temporelle plus spécifique que la contrainte associée à *a*. Dans les deux cas, il est possible de construire une instance de *P'* qui n'est pas instance de *P*.

Nous reviendrons à ces éléments de sémantique lorsque nous serons amenés à faire évoluer la définition des plans proposée par Weida. Avant de présenter ces évolutions, nous présentons une implémentation fidèle du système T-Rex réalisée avec l'aide du système de satisfaction de contraintes BackJava.

3.5 Implémentation de T-Rex avec BackJava

Nous donnons dans cette partie le détail de l'implémentation que nous avons faite du système T-Rex en utilisant le système de satisfaction de contraintes BackJava (Roy *et al.*, 1999). Nous commencerons par poser les différents problèmes posés par T-Rex – subsomption, subsomption inverse et compatibilité entre plans – en termes de problème de satisfaction de contraintes (*CSP*), puis nous présenterons succinctement BackJava. Nous donnerons enfin les détails d'implémentation liés à BackJava, d'une part pour d'illustrer l'utilisation de ce système, et d'autre part afin de justifier le choix de ce cadre de travail par la simplicité de l'implémentation.

3.5.1 Formulation en termes de CSP

Le problème de savoir si le plan *P* subsume ou non le plan *P'* est un problème de satisfaction de contraintes que nous exprimons ici comme tel. Un problème de satisfaction de contraintes est défini par un ensemble de *variables* dénotant les inconnues du problème, ces variables prenant leurs valeurs parmi un ensemble fini de valeurs donné *a priori*, ensemble appelé *domaine*. Les solutions du problème sont décrites par des relations entre les variables appelées *contraintes*. Une solution du problème est une instanciation de chacune des variables par une valeur choisie dans son domaine et telle que les contraintes soient satisfaites[35].

Soit *S* (respectivement *S'*) l'ensemble des sommets de *P* (respectivement *P'*). L'ensemble des variables du problème comporte autant d'éléments qu'il y a d'éléments dans *S*, chaque variable V_s étant associée à un sommet *s* distinct de *P*. On note S(V) le sommet de *P* associé à la variable V. On note C(s) le concept associé au sommet *s* et ct(s_1, s_2) la contrainte temporelle imposée entre les sommets s_1 et s_2. On note indifféremment $\succ$ la relation de subsomption entre concepts et entre contraintes temporelles.

Le domaine dom(V) de chaque variable est l'ensemble *S'* des sommets de *P'*. On note val(V) la valeur de la variable V lorsque celle-ci est instanciée. Quelle que soit V, val(V) est donc un sommet de *P'*.

Les contraintes portant sur les variables sont de deux sortes. D'une part, pour chaque variable V, une contrainte unaire impose que le concept associé au sommet de *P* auquel est associée V subsume le concept associé à la valeur de V, la valeur de V étant un sommet de *P'*. Cette contrainte est exprimée par l'équation (5) ci-dessous. D'autre part, pour chaque couple de variable (V_1, V_2), une contrainte binaire impose que la contrainte temporelle prescrite entre les sommets de *P* associés aux variables V_1 et V_2 subsume la contrainte temporelle prescrite dans *P'* entre les valeurs de ces variables. Cette contrainte est exprimée par l'équation (6) ci-dessous.

$$\forall V, \mathrm{C}(\mathrm{S}(V)) \succ \mathrm{C}(\mathrm{val}(V)) \tag{5}$$

$$\forall V_1, V_2, \mathrm{ct}(\mathrm{S}(V_1), \mathrm{S}(V_2)) \succ \mathrm{ct}(\mathrm{val}(V_1), \mathrm{val}(V_2)) \tag{6}$$

Le problème de savoir si un plan *P* subsume potentiellement un graphe d'observations *O* dans les deux hypothèses d'une part d'observations parfaites et d'autre part d'observations monotones s'exprime d'une manière similaire, une variable étant cette fois associée à chaque nœud du graphe d'observations, et la relation de subsomption étant remplacée par la relation de compatibilité dans le cas de l'hypothèse d'observations monotones.

3.5.2 Implémentation

Nous décrivons ici l'implémentation que nous avons réalisée de T-Rex en utilisant le système BackJava, que nous présenterons plus en détail au prochain chapitre. Il nous suffit

---

[35] Pour plus de détails, voir « La satisfaction de contraintes » page 105.

pour l'instant d'indiquer que BackJava, dans une approche objet, permet à partir de classes existantes de définir par héritage et polymorphisme de nouvelles classes de variables et de contraintes.

Avant de décrire l'implémentation dans sa partie proprement liée à BackJava, il nous faut d'abord décrire succinctement la manière dont sont représentés les plans. Nous avons pour cela masqué l'implémentation réelle, plus complexe, derrière quelques interfaces Java qui nous permettrons de clarifier la présentation de cette implémentation. Un plan est un graphe de contraintes temporelles dont les sommets sont associés à un concept dans un système de raisonnement terminologique et dont les arêtes sont des contraintes temporelles de Allen. Nous définissons l'interface TxNode[36] qui représente un sommet du graphe, et l'interface TxNetwork qui représente le graphe lui-même. L'interface TxNode ne définit que deux méthodes booléennes, l'une testant la subsomption et l'autre la compatibilité entre deux sommets. Pour des raisons de simplicité, nous avons identifié ici un sommet du graphe et le concept qui lui est associé. Nous ne détaillerons pas ici l'implémentation réelle des concepts associés, qui permet soit d'interfacer le système de raisonnement terminologique CLASSIC, soit de travailler sur une représentation interne de ces concepts représentés par des objets Java.

```
public interface TxNode
{
   public abstract boolean txCompatibleWith(TxNode n);
   public abstract boolean txSubsumes(TxNode n);
}
```

L'interface TxNetwork définit deux méthodes, l'une retournant la liste des sommets du graphe, et l'autre retournant la relation temporelle définie entre deux nœuds, sous la forme d'un entier court[37]. Nous ne détaillerons pas ici la création d'une instance de TxNetwork, et en particulier le calcul de la clôture du graphe de contraintes, que nous supposerons ici minimal.

```
public interface TxNetwork
{
   public abstract java.util.Enumeration getTxNodes();
   public abstract short txConstraint(TxNode n1, TxNode n2);
}
```

Le principe de l'implémentation consiste à définir une hiérarchie de solveurs sous-classes de la classe BJSolveur de BackJava pour chacun des problèmes de subsomption et de subsomption potentielle. Pour chaque problème, il convient de spécifier quel est le graphe d'« origine », c'est-à-dire le graphe dont les sommets seront associés aux variables du CSP, et quel est le graphe subsumant dans le cas du test de subsomption et le graphe d'observations dans le cas du test de subsomption potentielle, ainsi que le graphe de « destination », c'est-à-dire le graphe dans lequel seront prises les valeurs des variables.

On ne définit qu'une classe de variables, utilisée dans tous les types de problèmes, et autant de classes de contraintes unaires et de contraintes binaires qu'il y a de types de problèmes.

[36] Les classes et méthodes de BackJava sont en général préfixées par « BJ » et les classes et méthodes définies pour l'implémentation de T-Rex sont en général préfixées par « Tx ».
[37] Voir « Implémentation des relations temporelles » page 54.

Les variables sont toutes de même type, quel que soit le type de CSP, et sont représentées par une classe Java spécialisant une classe générique de variables de BackJava. Chaque variable, de plus, « connaît » le sommet du graphe auquel elle est associée, le graphe d'origine et le graphe de destination. On définit ainsi la classe TxVar, sous-classe de la classe générique BJObjVar définie par BackJava. La classe TxVar définit trois variables d'instances :

- *txNode*, de la classe TxNode, qui représente le sommet auquel est associée la variable ;
- *fromNetwork*, de la classe TxNetwork, qui représente le graphe d'origine ;
- *toNetwork*, de la classe TxNetwork, qui représente le graphe de destination.

Les contraintes unaires de subsomption des concepts portés par les sommets des graphes associés aux plans sont implémentées par des sous-classes de la classe BJUnaryCt, selon le problème à résoudre :

- TxSubsomptionPrimAnalysisCt pour le problème de la subsomption ;
- TxInverseSubsomptionPrimAnalysisCt pour le problème de la subsomption inverse ;
- TxCompatibilityPrimAnalysisCt pour le problème de la compatibilité.

Pour implémenter le filtrage, on se contente de surcharger la méthode *isSatisfied* définie comme méthode abstraite dans la classe BJUnaryCt. Cette méthode prend comme paramètre une instance de la classe Object – la valeur de la variable – et retourne un booléen qui est *true* si la contrainte est satisfaite et *false* sinon. C'est le processus de résolution qui se chargera d'appeler la méthode *isSatisfied* de chacune des contraintes définies pour effectuer le filtrage. Le code de chacune de ces méthodes est donné figure 27. x représente la variable sur laquelle porte la contrainte ; x est définie comme variable d'instance de la classe BJUnaryCt, de classe BJVar.

```
class TxSubsomptionPrimAnalysisCt :
public boolean isSatisfied(Object a)
{
   TxVar var = (TxVar) x;
   TxNode n = (TxNode) a;
   return var.getTxNode().txSubsumes(n);
}

class TxInverseSubsomptionPrimAnalysisCt :
public boolean isSatisfied(Object a)
{
   TxVar var = (TxVar) x;
   TxNode n = (TxNode) a;
   return n.txSubsumes(var.getTxNode());
}

class TxCompatibilityPrimAnalysisCt :
public boolean isSatisfied(Object a)
{
   TxVar var = (TxVar) x;
   TxNode n = (TxNode) a;
   return var.getTxNode().txCompatibleWith(n);
}
```

figure 27 : méthode isSatisfied pour les contraintes unaires

Les contraintes binaires représentant les contraintes temporelles pour chacun de trois types de problèmes s'implémentent d'une manière analogue en définissant trois sous-classes de la classe générique BJGenericBinaryCt : les classes TxSubsumptionMatchingCt, TxInverseSumptionMatchingCt et TxCompatibilityMatchingCt, et en surchargeant la méthode *isSatisfied* qui prend cette fois-ci deux paramètres qui sont les valeurs des deux variables *x* et *y*. Pour le test de subsomption, la méthode renvoie *true* si la contrainte temporelle imposée entre les sommets du plan subsumant (graphe d'origine) représentés par les deux variables *x* et *y* est plus générale que la contrainte imposée dans le plan subsumé (graphe de destination) entre les sommets correspondants, c'est-à-dire entre les valeurs des variables. De manière analogue, pour le test de subsomption inverse, la méthode renvoie *true* si la contrainte imposée dans le plan (graphe de destination) est plus générale que la contrainte observée (graphe d'origine), et, pour le test de compatibilité, si la contrainte imposée dans le plan et la contrainte observée sont compatibles. Le code de ces méthodes est donné figure 28. Les méthodes *subsumes* et *compatible*, méthodes statiques de la classe Allen, qui figurent dans le code donné ci-dessous et qui prennent deux paramètres entiers représentant les contraintes temporelles utilisent le « et » binaire et sont illustrées figure 29.

```
class TxSubsumptionMatchingCt:
public boolean isSatisfied(Object a, Object b)
{
   TxVar v1 = (TxVar) x;
   TxVar v2 = (TxVar) y;
   TxNode n1 = (TxNode) a;
   TxNode n2 = (TxNode) b;
   short fromCt =
      v1.getFromNetwork().txConstraint(v1.getTxNode(),
      v2.getTxNode());
   short toCt = v1.getToNetwork().txConstraint(n1, n2);
   return Allen.subsumes(fromCt, toCt);
}

class TxInverseSumptionMatchingCt :
public boolean isSatisfied(Object a, Object b)
{
   TxVar v1 = (TxVar) x;
   TxVar v2 = (TxVar) y;
   TxNode n1 = (TxNode) a;
   TxNode n2 = (TxNode) b;
   short fromCt =
      v1.getFromNetwork().txConstraint(v1.getTxNode(),
      v2.getTxNode());
   short toCt = v1.getToNetwork().txConstraint(n1, n2);
   return Allen.subsumes(toCt, fromCt);
}
```

```
class TxCompatibilityMatchingCt:
public boolean isSatisfied(Object a, Object b)
{
   TxVar v1 = (TxVar) x;
   TxVar v2 = (TxVar) y;
   TxNode n1 = (TxNode) a;
   TxNode n2 = (TxNode) b;
   short fromCt =
      v1.getFromNetwork().txConstraint(v1.getTxNode(),
      v2.getTxNode());
   short toCt = v1.getToNetwork().txConstraint(n1, n2);
   return Allen.compatible(toCt, fromCt);
}
```

figure 28 : méthode isSatisfied pour les contraintes binaires

```
public static boolean subsumes(short rel1, short rel2)
{
   return (rel1 & rel2) == rel2;
}

public final static boolean compatible(short rel1, short rel2)
{
   return (rel1 & rel2) != 0;
}
```

figure 29 : tests de subsomption et de compatibilité pour les contraintes temporelles

Il ne reste plus qu'à instancier les variables et à créer les contraintes pour chacun des trois types de problèmes. On définit pour cela une hiérarchie de solveurs, en créant une classe abstraite TxSolver sous-classe de BJSolver qui implémentera les opérations communes aux trois types de problèmes, et trois sous-classes de TxSolver implémentant les tâches spécifiques : TxSumptionSolver, TxInverseSumptionSolver et TxCompatibilitySolver.

La classe TxSolver définit deux variables d'instances de la classe TxNetwork pour les graphes d'origine et de destination : *fromNetwork* et *toNetwork*. Deux méthodes abstraites sont définies, qui seront implémentées par les sous-classes de TxSolver, qui permettent de choisir les bons types de contraintes pour chacun des problèmes. Ainsi, la méthode *setMatchingCt* qui prend comme paramètre deux variables, de type TxVar, instancie une contrainte temporelle entre deux variables, la classe TxSubsumptionSolver instanciant une contrainte de type TxSubsumptionMatchingCt, la classe TxInverseSubsumptionSolver une contrainte de type TxInverseSubsumptionMatchingCt et la classe TxCompatibilitySolver une contrainte de type TxCompatibilityMatchingCt. Il en va de même pour la méthode abstraite *setPrimAnalysisCt* qui instancie pour une variable de type TxVar la contrainte unaire portant sur le concept associé à la variable. La création des contraintes est donnée pour la classe TxSubsuptionSolver dans la figure 30. Ce sont les deux seules méthodes implémentées par les sous-classes de TxSolver.

```
public void setMatchingCt(TxVar var1, TxVar var2) throws
BJWipedOutException
{
   new TxSubsumptionMatchingCt(var1, var2);
}

public void setPrimAnalysisCt(TxVar var) throws
BJWipedOutException
{
   new TxSubsomptionPremAnalysisCt(var);
}
```

figure 30 : instanciation des contraintes dans la classe TxSubsumptionSolver

Dans la classe TxSolver, il ne reste plus qu'à créer les variables et les contraintes. Une instance de TxVar est créée pour chaque sommet du graphe d'origine, toutes les variables partageant le même domaine qui est l'ensemble des sommets du graphe de destination. Pour garantir que l'appariement est injectif, la méthode *allDiff*, définie dans la classe VariableVector de BackJava, qui représente un vecteur d'instances de la classe BJVar, est appelée sur l'ensemble des variables instanciées. Cette méthode assure que chacune des variables est associée à une valeur distincte de la valeur des autres variables, ce qui assure bien, ici, que l'appariement est injectif. Une contrainte unaire est créée pour chaque variable, par l'appel de la méthode *setPrimAnalysisCt* et une contrainte temporelle est créée pour chaque couple de variables par l'appel à la méthode *setMatchingCt*. Le code créant toutes les contraintes du problème est donné par la figure 31.

```
protected void initializeConstraints(VariableVector vars)
throws BJWipedOutException
{
   vars.allDiff();
   for (int i = 0; i < vars.size(); i++)
   {
      TxVar v = (TxVar) vars.elementAt(i);
      setPrimAnalysisCt(v);
   }
   for (int i = 0; i < vars.size(); i++)
   {
      TxVar v1 = (TxVar) vars.elementAt(i);
      for (int j = i + 1; j < vars.size(); j++)
      {
         TxVar v2 = (TxVar) vars.elementAt(j);
         setMatchingCt(v1, v2);
      }
   }
}
```

figure 31 : création des contraintes du problème

Finalement, la réponse au test de subsomption, de subsomption inverse ou de compatibilité est donné par l'appel à la méthode *firstSolution* définie dans la classe BJSolver : s'il existe une solution, un appariement existe et la réponse au test de subsomption est positive.

Nous avons décrit de manière assez détaillée l'implémentation du système T-Rex dans le cadre de BackJava, du moins dans sa partie satisfaction de contraintes. Cette implémentation illustre la facilité avec laquelle il est possible de définir un nouveau problème de satisfaction de contraintes dans un tel cadre : la spécialisation de quelques classes et la surcharge de quelques méthodes permettent d'implémenter rapidement un système relativement complexe tel que T-Rex. Nous avons validé cette implémentation en reprenant intégralement l'exemple académique de la cuisson des pâtes initialement proposé par (Kautz, 1991) et repris dans (Weida, 1992b).

4 T-Rex revisité

Nous présentons à présent les extensions et modifications que nous avons apportées au formalisme de T-Rex tel qu'il est proposé par Weida. Nous parlerons à partir d'ici de « canevas », descriptions de classes de séquences audiovisuelles, et non plus de « plans » comme c'était le cas pour le système T-Rex. Nous nous attacherons plus au problème de la reconnaissance de canevas qu'au problème de la subsomption entre canevas, sur lequel nous reviendrons plus loin. Nous adaptons légèrement la syntaxe lispienne originelle pour qu'elle soit plus lisible[38].

4.1 Observations élémentaires datées

Dans le système T-Rex, les relations temporelles entre les observations peuvent être de nature disjonctive. Nous nous plaçons dans un cadre plus restreint où les observations sont des résultats d'algorithmes d'analyse et de segmentation, ou bien le produit d'une annotation manuelle, et où ces observations sont précisément datées par rapport au flux audiovisuel. Exprimé d'une autre manière, plus proche de la définition formelle que nous avons donnée ci-dessus[39], cela revient à dire que les observations sont associées à une période unique de la structure temporelle, et non plus à un ensemble de périodes. La fonction γ qui associait un ensemble de périodes à chaque instance des concepts élémentaires d'action devient :

$\gamma : \Delta_I \rightarrow T^*_\leq$	Fonction qui à chaque instance des concepts d'événements élémentaires associe une période de la structure temporelle $T^*_\leq$

Cette restriction sur l'extension temporelle des événements élémentaires nous sera surtout utile ultérieurement, en particulier lorsque nous définirons les séquences itérées[40].

---

[38] La grammaire BNF de cette syntaxe se trouve en annexe.

[39] Voir « Une sémantique pour T-Rex » page 64.

[40] Voir « Séquences itérées » page 82.

4.2 Imbrication de plans

Dans le système T-Rex, faire référence à un plan dans la définition d'un autre plan n'est qu'une facilité d'écriture (une *macro*, comme l'appelle Weida). Weida donne en exemple la définition plan ASSEMBLE-SPAGHETTI-MARINARA[41] (voir figure 25). Le plan imbriqué apparaît bien comme étant associé à un nœud du plan englobant, mais il n'apparaît pas en tant que tel dans la définition des contraintes temporelles portant sur les nœuds du graphe, et seuls les nœuds terminaux, c'est-à-dire les nœuds associés à des concepts élémentaires d'actions, peuvent être temporellement contraints. De notre point de vue, ceci est une limitation du système ; dans T-Rex, un plan est reconnu lorsque les actions qui le composent ont été observées et que ces actions respectent certaines contraintes, mais le plan reconnu ne devient pas à son tour instance d'un concept d'action de plus haut niveau. On peut dire que les plans ne sont pas « réifiés ». Pour qu'un plan puisse être composant à part entière d'un autre plan, il faut qu'il ait une dimension temporelle, ce qui revient à dire qu'on doit pouvoir associer à une instance reconnue d'un plan une période ou un ensemble de périodes de la structure temporelle sous-jacente, comme c'est le cas pour les instances de concepts d'action élémentaire. Nous donnons sur la figure 32 trois exemples de canevas en indiquant sur chaque schéma par une double flèche l'*extension temporelle* associée à l'instance du canevas reconnu, c'est-à-dire la période temporelle associée à cette instance. Sur les schémas (A) et (B), il s'agit de cas particulier de plans (au sens filmique) ; le schéma (A) illustre un plan pendant lequel apparaît une incrustation de texte, comme par exemple le nom du reporter, et le schéma (B) illustre un plan se déroulant pendant une séquence musicale. Dans les deux cas, on veut d'une certaine manière spécialiser le concept de *Plan* ; l'extension temporelle d'une instance de ces canevas doit alors naturellement être la même que l'extension temporelle du plan. Le schéma (C) illustre de manière simpliste un reportage comme étant ce qui est diffusé entre deux jingles, le reportage lui-même commençant à la fin du premier jingle et se terminant au début du second.

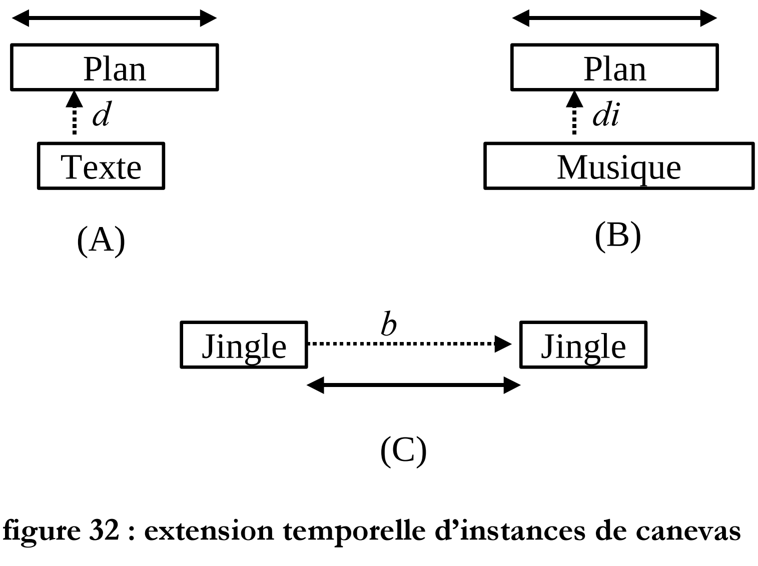


figure 32 : extension temporelle d'instances de canevas

---

[41] Voir figure 25 page 61.

Nous proposons d'exprimer l'extension temporelle d'une instance de canevas par le biais de contraintes temporelles imposées entre l'instance elle-même et ses composants. Ainsi, pour les canevas (A) et (B) de la figure 32, une contrainte d'égalité temporelle (relation de Allen *equals*) est imposée entre le composant associé au concept *Plan* et l'instance du canevas en cours de définition. Le mot-clé *this* est réservé pour figurer l'instance du canevas dans la partie de la définition où sont déclarées les contraintes temporelles. La définition textuelle des canevas illustrés par la figure 32 est donnée par la figure 33. Par la suite, nous parlerons pour abréger de contraintes temporelles entre le canevas et ses composants.

Plan-avec-Texte nodes s_1 [*Plan*] s_2 [*Texte*] constraints s_2 *during* s_1 s_1 *equals* this (A)	**Plan-Musical** nodes s_1 [*Plan*] s_2 [*Musique*] constraints s_1 *is-during* s2 s_1 *equals* this (B)	**Reportage** nodes s_1 [*Jingle*] s_2 [*Jingle*] constraints s_1 *before* s_2 s_1 *meets* this this *meets* s_2 (C)

figure 33 : définitions des canevas de la figure 32

Cette manière de procéder présente l'avantage de ne pas introduire de nouveau formalisme pour la définition de l'extension temporelle des canevas. Le graphe de contraintes associé à un canevas contient un sommet de plus que le graphe associé à un plan dans le système T-Rex mais le calcul de la couverture minimale reste le même. D'autre part, comme on le verra plus loin, il est possible dans certains cas, lorsque les observations sont datées et que les contraintes temporelles entre l'instance du canevas et les éléments observés sont élémentaires (c'est-à-dire non disjonctives) de calculer à partir du graphe de contraintes les bornes exactes des instances des canevas reconnus.

Il arrive fréquemment qu'on souhaite définir la période associée à un canevas reconnu comme étant l'union des périodes associées aux composants de ce canevas. L'union de deux périodes temporelles est prise ici dans le sens de la plus « petite » période recouvrant les deux périodes initiales. Le mot-clé *reduce* sert dans la syntaxe de définition des canevas à demander au système, lorsque c'est possible, de déterminer les contraintes temporelles qui doivent être imposées entre les composants du canevas et le canevas de manière à ce que la période temporelle associée à l'instance du canevas représente l'union temporelle de ses composants. Le principe de « réduction » du graphe de contraintes consiste dans un premier temps, après calcul de la couverture minimale du graphe, à chercher tous les composants du canevas qui sont dans une relation temporelle plus générale que – *i.e.* contenant – *equals*, *starts* ou *finishes* avec le canevas lui-même, et à stocker ces composants dans trois vecteurs distincts. Pour chacun des composants c pouvant éventuellement être dans la relation temporelle *equals* avec le canevas, on cherche parmi tous les composants s'il en existe un qui peut « déborder » temporellement c, c'est-à-dire qui est dans une relation temporelle avec c compatible[42] avec la relation temporelle {*before* ∨ *after* ∨ *is-during* ∨ *overlaps*

[42] Rappelons que deux relations temporelles sont compatibles si leur intersection est non vide.

$\vee$ *is-overlapped* $\vee$ *meets* $\vee$ *is-met* $\vee$ *is-started* $\vee$ *is-finished*}[43]. Si aucun composant vérifiant cette condition n'est trouvé, la relation temporelle entre *c* et le canevas est forcée à *equals*. On procède d'une manière similaire pour chacun des composants *c* pouvant éventuellement être dans la relation *starts* avec le canevas : si aucun autre composant n'est dans une relation compatible avec {*before* $\vee$ *is-during* $\vee$ *overlaps* $\vee$ *meets* $\vee$ *is-finished*} avec *c*, la relation entre *c* et le canevas est forcée à {*equals* $\vee$ *starts*} si la relation existante était compatible avec *equals*, et forcée à *starts* sinon. De la même manière, pour chacun des composants *c* pouvant être dans la relation *finishes* avec le canevas, si aucun autre composant n'est dans une relation compatible avec {*after* $\vee$ *is-during* $\vee$ *is-overlapped* $\vee$ *is-met* $\vee$ *is-started*} la relation entre *c* et le canevas est forcée à {*equals* $\vee$ *finishes*} ou *finishes*. Finalement, la couverture minimale du graphe est à nouveau calculée pour propager les nouvelles contraintes.

Sur le plan formel, la définition d'un canevas reprend la définition des plans[44] en y ajoutant une fonction qui représente les contraintes temporelles entre l'instance et les composants du canevas :

$f_v : X \rightarrow 2^A$	Fonction qui associe une disjonction de relations temporelles élémentaires à toutes les variables. Ces relations sont les contraintes temporelles imposées entre les périodes associées à chacun des composants du canevas et les périodes associées au canevas lui-même

Le calcul de subsomption est également légèrement modifié pour tenir compte de ces nouvelles contraintes temporelles. Le principe de l'appariement est encore valide, à condition que les contraintes données par f_V dans le graphe subsumant soient plus générales que les contraintes imposées entre les nœuds appariés du graphe subsumé. Il en va de même pour la subsomption potentielle et la compatibilité entre canevas. Pour poser le problème de la subsomption comme un problème de satisfaction de contraintes, il faut ajouter aux contraintes précédemment définies[45] une contrainte unaire portant sur chacune des variables associées aux composants du canevas subsumant. Cette contrainte est satisfaite si la relation temporelle imposée entre le composant du canevas subsumant associé à la variable et le canevas lui-même est plus générale que la contrainte imposée entre la valeur de la variable, c'est-à-dire un des composants du canevas subsumé et le canevas subsumé lui-même.

L'introduction de ces nouvelles contraintes entraîne une différence entre le calcul de subsomption et le processus de reconnaissance. En effet, lorsqu'on cherche à reconnaître l'occurrence d'un canevas parmi un ensemble d'observations, les contraintes portant sur l'instance du canevas servent à donner des indications sur l'extension temporelle du canevas reconnu, alors que ces contraintes interviennent directement dans le calcul de subsomption entre canevas.

En associant une dimension temporelle aux canevas « imbriqués » on obtient une définition récursive des canevas, où les composants d'un canevas peuvent être soit des concepts

[43] Soit la relation temporelle complémentaire de {*equals* $\vee$ *starts* $\vee$ *finishes* $\vee$ *during*}

[44] Voir « Une sémantique pour T-Rex » page 64.

[45] Voir « Formulation en termes de CSP » page 69.

d'événements élémentaires, soit d'autres canevas. On peut alors opter pour deux représentations différentes des canevas. Ces deux représentations sont illustrées par la figure 34, où un canevas représentant la fin d'un journal télévisé est composé de deux sous-canevas consécutifs, le premier représentant un reportage, un reportage commençant par un plan du présentateur et finissant par un volet, le second représentant la conclusion du journal, cette conclusion étant composée d'un plan du présentateur suivi du générique de fin. L'extension temporelle des instances des canevas illustrés est à chaque fois supposée être définie comme l'union des extensions temporelles de leurs composants.

La première représentation est la plus proche de la définition et consiste à représenter le canevas par un graphe de contraintes contenant autant de nœuds que le canevas contient de composants, un nœud pouvant être associé à un autre graphe dans le cas des canevas imbriqués, et ainsi de suite, récursivement. On ne s'occupe alors pas des contraintes pouvant être déduites, contraintes portant par exemple sur des composants de deux sous-canevas différents, et on laisse le soin au processus de reconnaissance de vérifier que les observations vérifient bien toutes les contraintes. Ce processus peut par exemple consister à chercher récursivement les occurrences de tous les canevas en commençant par les canevas les plus « profonds ». La seconde représentation, plus proche de la représentation des plans du système T-Rex consiste à développer les canevas imbriqués. Il faut alors prendre soin de tenir compte de toutes les contraintes temporelles : contraintes portant sur les composants de chaque sous-canevas, entre composants, entre les composants et le canevas père, etc., pour ne pas perdre d'information. Les contraintes temporelles portant directement sur les canevas ne servent alors qu'à être propagées à leurs composants. Dans la mesure où le calcul de la couverture minimale du graphe de contraintes est complet, les deux représentations sont équivalentes.

Lorsqu'on cherche à reconnaître parmi un ensemble d'observations élémentaires, c'est-à-dire d'instances des concepts d'événements élémentaires, les occurrences d'un canevas, la deuxième représentation est la plus économique. En effet, les observations n'étant composées que d'observations élémentaires, un canevas imbriqué ne peut-être reconnu que si ses composants l'ont été ; seuls les composants élémentaires interviennent dans la reconnaissance. Il convient cependant de garder une trace de l'imbrication des canevas, ne serait-ce que pour la présentation des résultats. Nous choisirons cette représentation, bien que nous soyons amenés dans certains cas à opter pour une représentation mixte.

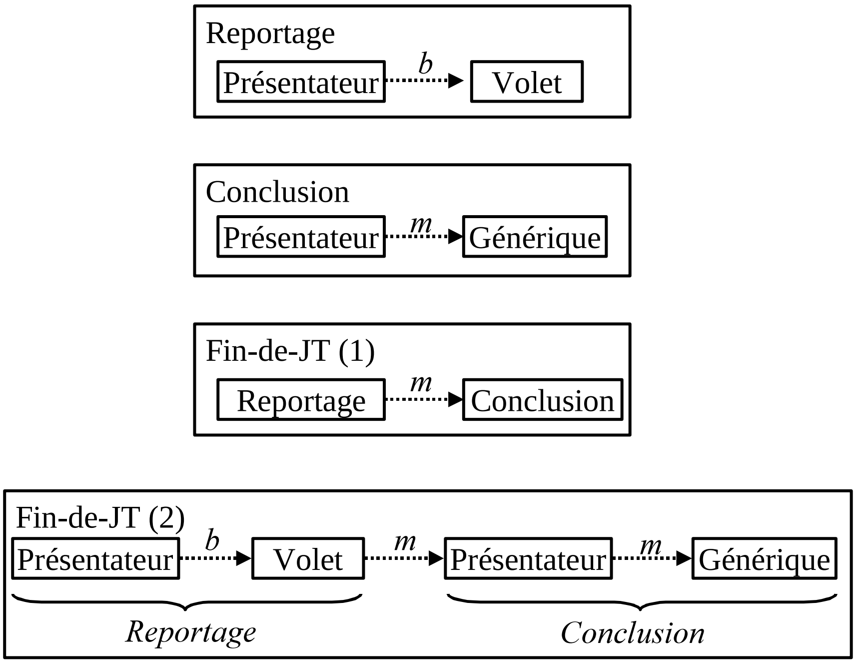


figure 34 : imbrication de canevas

4.3 Appariement non injectif

Dans le système T-Rex, pour qu'un plan soit reconnu, il faut que les composants du plan soit associés à des observations deux à deux distinctes ; l'appariement doit être injectif. Nous avons été amenés à lever cette restriction pour pouvoir traiter certains cas. La figure 35 illustre ainsi un exemple où un appariement injectif ne donnerait pas les résultats escomptés. On cherche à représenter un « plan musical » comme un plan *pendant lequel* est entendu de la musique. On définit donc le canevas *Plan-Musical* avec deux composants ayant une intersection temporelle commune – la relation temporelle correspondante est donnée sur la figure –, l'un associé à *Plan*, l'autre associé à *Musique*, qui sont supposés être des concepts d'événements élémentaires. On cherche à reconnaître les occurrences de *Plan-Musical* parmi les observations données sur la figure. Les plans p_1, p_2, p_3 et p_5 peuvent participer à une occurrence de *Plan-Musical*, chacun de ces plans ayant une intersection commune avec un segment de musique, mais, dans le cas d'un appariement injectif, les plans p_1, p_2 et p_3 ne peuvent participer *simultanément* à la reconnaissance. Ainsi, le canevas représentant deux plans musicaux consécutifs ne serait pas reconnu. Dans le cas d'un appariement non injectif, le segment de musique m_1 peut être associé simultanément aux plans p_1, p_2 et p_3 pour former des instances de *Plan-Musical*. Un même événement peut par conséquent jouer plusieurs rôles en même temps. Un autre exemple typique est celui des plans de transition ou des transitions progressives (fondus, volets, etc.) qui concluent une séquence pour en introduire une autre. C'est là la première raison qui nous fait choisir un appariement non injectif.

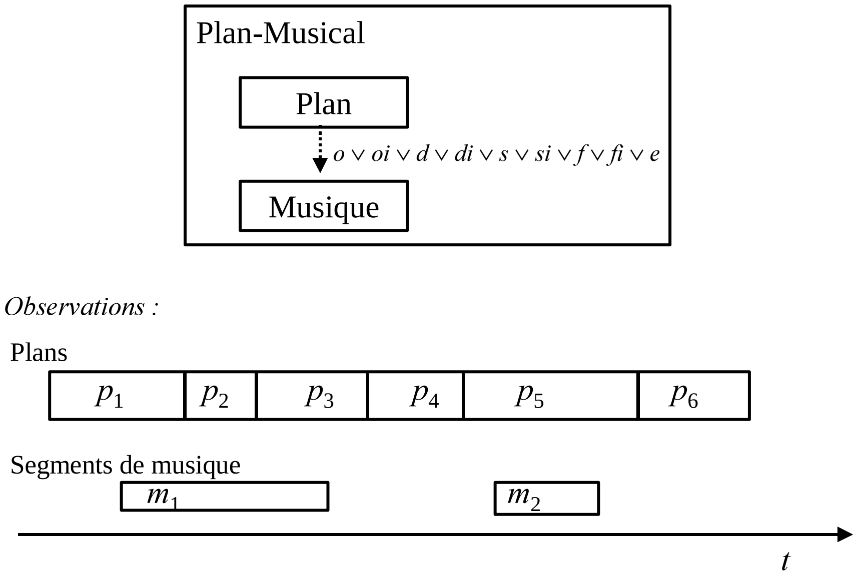


figure 35 : exemple d'appariement non injectif

La seconde raison de choisir un appariement non injectif, moins profonde peut-être, est liée à l'origine des observations initiales. Considérons le canevas illustré par la figure 36, qui représente un plan comportant une inscription de texte, le texte pouvant apparaître pendant la totalité du plan ou seulement pendant un fragment. Le processus de détection des zones de texte peut procéder de deux manières principales : il peut soit marquer les plans comportant du texte comme instances du concept *Texte*, auquel cas ces plans sont instances de la conjonction *Plan* $\cap$ *Texte*, ou bien il peut détecter des segments temporels n'ayant pas la même extension temporelle que les plans, soit qu'il analyse directement le flux vidéo, soit qu'il classifie des segments plus petits que les plans, s'il analyse par exemple des images clé correspondant à des micro-segments des plans. Dans le premier cas, les observations sont constituées d'une strate d'instances, les plans, qui peuvent être ou non des instances de *Texte*. Dans le second cas, les observations sont constituées de deux strates distinctes, l'une pour les plans, l'autre pour les segments comportant une zone de texte. Utiliser un appariement non injectif permet de traiter ces deux cas indifféremment. On peut ainsi, d'une certaine manière, exprimer par un canevas une notion proche de la conjonction de concepts. Le concept $C \cap D$ et le canevas $\{s_1\ C,\ s_2\ D,\ s_1\ \textit{equals}\ s_2\}$ ne sont pas identiques, ne serait-ce que parce que dans le premier cas la relation liant le concept à C et à D est une relation ontologique de type *is-a* et dans l'autre la relation liant les composants au canevas un lien méréologique de type *part-of*, mais ils représentent des notions voisines.

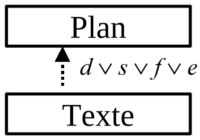


figure 36 : plan comportant une incrustation de texte

5 Séquences itérées

La *séquence* est une notion importante du langage audiovisuel. Une séquence est en général considérée comme une suite de plans présentant une certaine unité (unité d'action, unité de lieu) ; on parlera ainsi de « séquence plateau » ou de « séquence reportage ». Plus généralement, la notion de séquence désigne une série d'événements consécutifs dans le temps et présentant des caractéristiques communes ; il sera par exemple question d'une séquence de brèves[46]. Nous introduisons cette notion de séquence dans notre langage de description des canevas par l'ajout de l'opérateur « * », que nous appelons *opérateur d'itération* ; nous parlons alors et de « canevas itéré », dont les instances sont des *séquences itérées*. Par opposition, un « événement simple » est un événement non itéré. Par extension, on parlera de « composant simple » ou de « composant itéré » pour désigner les composants d'un canevas associés à un canevas simple ou à un canevas itéré.

Cet opérateur d'itération peut être comparé d'une certaine manière à l'opérateur « + » des expressions régulières. On peut ainsi définir une « séquence musicale », illustrée par la figure 37, comme une séquence de plans musicaux, la définition d'un plan musical étant donnée par le canevas de la figure 35.

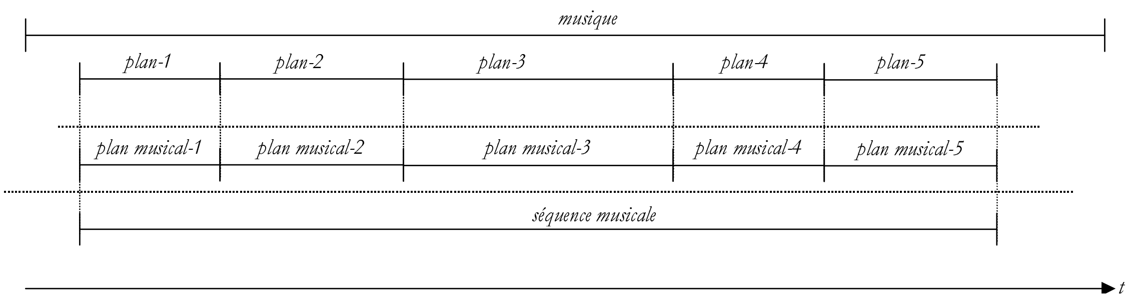


figure 37 : séquence musicale

Par définition, un canevas itéré T* représente l'ensemble des séquences itérées composées d'instances de T, c'est-à-dire les successions d'instances du canevas T dont les éléments sont deux à deux dans la relation temporelle *meets*. Une séquence itérée comporte au moins un élément, et l'extension temporelle d'une séquence itérée est nécessairement l'union temporelle des extensions temporelles de ses éléments.

Tels que nous avons défini les canevas, les concepts d'événements élémentaires ne peuvent être itérés. De manière similaire aux canevas itérés, si C est un concept d'événement

---

[46] Une brève est en général un reportage court où le journaliste n'apparaît pas à l'écran.

élémentaire, nous notons C^* le canevas itéré représentant les séquences itérées dont les éléments sont des instances de C. La notation reste ainsi homogène. Par la suite, nous aurons parfois besoin de faire la distinction entre les canevas itérés se référant à un canevas des canevas itérés se référant à un concept d'événement élémentaire.

Une période temporelle étant associée à une séquence itérée – l'union temporelle des éléments de la séquence –, il est possible d'imposer des contraintes temporelles de la même manière entre tous les composants d'un canevas, qu'ils soient itérés ou non. L'introduction de l'opérateur d'itération n'apporte pas de changement à la définition des contraintes temporelles.

5.1 Canevas bornés

Seuls les canevas que nous appelons « bornés » (*bounded*) peuvent être itérés ; un canevas est borné s'il est possible, dans le cas où chaque observation initiale est associée à une période temporelle unique – observations datées par exemple –, d'attribuer une période temporelle unique à toute instance reconnue de ce canevas. Soit, en d'autres termes, si l'on sait toujours « où commence » et « où finit » toute instance de ce canevas. Prenons l'exemple du canevas *Musical* illustré par la figure 38, qui représente simplement un segment temporel pendant lequel est entendu un fragment de musique. Ce canevas est non borné. Selon la définition que nous avons donnée des canevas itérés, il ne peut donc exister de séquence itérée d'instances de ce canevas. En effet, deux instances de ce canevas ne peuvent être dans la relation temporelle *meets*. Soient m_1 et m_2 deux segments de musique. m_1 et m_2 permettent de reconnaître deux instances M_1 et M_2 du canevas *Musical*. Quelle que soit la relation temporelle entre m_1 et m_2, la relation temporelle déduite par transitivité entre M_1 et M_2 est nécessairement une disjonction de relations temporelles, et ne peut en aucun cas être la relation *meets*.

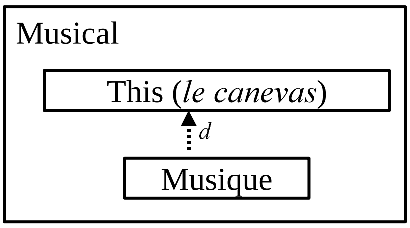


figure 38 : exemple de canevas non borné

On l'a vu, les observations sont associées à une période temporelle unique[47] ; les concepts d'événements élémentaires sont donc bornés. Pour déterminer si un canevas est borné, on examine les relations temporelles qui le relient à ceux de ses composants qui sont eux-mêmes bornés. Si, par exemple, le canevas est dans la relation *equals* avec un de ses composants bornés, il est lui-même borné. Un canevas est borné si, et seulement si, les deux propositions suivantes sont conjointement vérifiées :

[47] Voir « Observations élémentaires datées » page 75.

- **Proposition 1** : le canevas est lui-même dans une relation temporelle plus spécifique que {*starts* ∨ *equals* ∨ *is-started* } avec un de ses composants bornés, ou bien il est dans la relation temporelle *meets* avec un de ces composants bornés.

- **Proposition 2** : le canevas est lui-même dans une relation temporelle plus spécifique que {*finishes* ∨ *equals* ∨ *is-finished*} avec un de ses composants bornés, ou bien il est dans la relation temporelle *is-met* avec un de ces composants bornés.

Un canevas borné peut contenir des disjonctions de relations temporelles. Ainsi, le canevas illustré par la figure 39 est un canevas borné. Ce canevas *Introduction* indique qu'un plan introductif du présentateur commence par un jingle sonore, ce jingle pouvant commencer avant le début du plan du présentateur ou juste en même temps. Les contraintes temporelles entre les composants et le canevas lui-même indiquent que le jingle fait partie de l'introduction. Sur la figure, on voit qu'on peut associer une période temporelle à l'instance du canevas reconnue dans les deux cas de figure d'un jingle commençant avant ou d'un jingle commençant en même temps que le plan du présentateur.

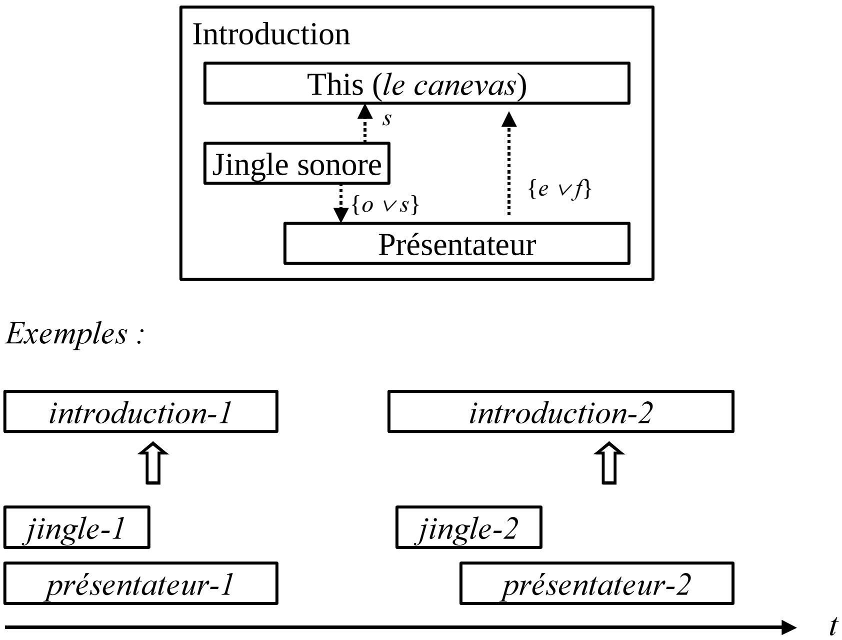


figure 39 : exemple de canevas borné

5.2 Formalisation

Intuitivement, une interprétation du canevas itéré T^* est simplement l'ensemble des séquences d'éléments du domaine d'interprétation dont la description respecte T et dont les périodes associées se « suivent » deux à deux. Plus formellement, en notant perm(E) l'ensemble des permutations des éléments de l'ensemble E et en employant les ensembles et les fonctions définies pour les canevas simples[48], on a :

$$T^{*\varepsilon} = \{ s \in perm(\Delta_I) \mid \forall a_i \in s,$$
$$a_i \in T^{\varepsilon},$$
$$si\ |s| > 1, 1 < i < |s| \Rightarrow (\gamma(a_{i-1}), \gamma(a_i)) \in \{meets\}^{\alpha} \wedge (\gamma(a_i), \gamma(a_{i+1})) \in \{meets\}^{\alpha} \}$$

5.3 Processus de reconnaissance

Les séquences itérées d'un canevas *T** forment un sous-ensemble de l'ensemble des permutations construites sur l'ensemble des observations. Lorsqu'un canevas comporte un composant itéré, le processus de reconnaissance doit donc apparier non plus un nœud du graphe associé au canevas à reconnaître à un nœud du graphe d'observations, mais un nœud à un sous-graphe du graphe d'observations. Nous donnerons dans le chapitre 4 une implémentation de ce processus d'appariement qui permet d'éviter une explosion combinatoire dans le type de cas que nous cherchons à traiter.

La reconnaissance de canevas est un problème de satisfaction de contraintes (*CSP*) dont formulation a été donnée plus haut pour les canevas ne comportant pas de composant itéré[49]. A chaque composant du canevas est associée une variable du CSP, dont le domaine est l'ensemble des observations. Une contrainte unaire porte sur chaque variable, contrainte qui impose que la valeur de la variable, c'est-à-dire une observation, soit subsumée par le concept associé à la variable, et une contrainte temporelle binaire porte sur chaque couple de variables. Le problème peut se formaliser d'une manière similaire pour les canevas comportant des composants itérés, bien que, pour des raisons d'efficacité, l'implémentation ne suive pas cette formulation. A chaque composant itéré est associée une variable dont le domaine est l'ensemble des permutations calculées sur les observations. Une contrainte unaire impose que la valeur de cette variable itérée soit une séquence itérée – succession d'observations consécutives – dont chaque élément est subsumé par le concept de base du concept associé à la variable (soit le concept *C* pour le concept itéré *C**). Une contrainte temporelle binaire porte sur chaque couple de variables, itérées et non itérées.

Schématiquement, l'idée de l'implémentation est la suivante. Pour les canevas itérés se référant à un concept itéré *C**, on cherche dans un premier temps dans le graphe d'observations les *séquences maximales* d'instances de *C*. Schématiquement une séquence maximale d'instances de *C* est une séquence itérée d'instances de *C* qui n'est sous-séquence d'aucune autre séquence itérée d'instances de *C*. Autrement dit, une séquence maximale d'instances de *C* est une séquence itérée d'instances de *C* telles qu'il n'existe pas dans le graphe d'observation d'instances de *C* qui soit dans la relation *meets* avec son premier élément ou dans la relation *is-met* avec son dernier élément. Dans un second temps, l'ensemble des séquences itérées est représenté comme l'ensemble des sous-séquences de ces séquences maximales. Une séquence itérée est alors représentée par la séquence maximale dont elle est sous-séquence, par l'indice de son premier élément dans cette séquence maximale et par sa longueur. On obtient ainsi une représentation implicite économique des séquences itérées.

---

[48] Voir « Une sémantique pour T-Rex » page 64.

[49] Voir « Formulation en termes de CSP » page 69

Pour chaque composant du canevas à reconnaître une variable est créée, variable dite « simple » pour les composants non itérés et variable « itérée » pour les composants itérés. Le domaine de chaque variable itérée V_{C^*} est l'ensemble de toutes les sous-séquences des séquences maximales d'instances de *C*. Nous avons implémenté des méthodes efficaces de réduction des domaines qui permettent de filtrer rapidement les contraintes temporelles portant sur des variables itérées.

Nous n'avons donné ici qu'une idée très générale du processus de reconnaissance, en décrivant succinctement les diverses étapes de création et de résolution du problème de satisfaction de contraintes représentant le problème à résoudre. Ce processus sera décrit de manière plus précise au chapitre 4.

5.4 Problème de la subsomption de canevas

L'introduction de l'opérateur « * » d'itération rend le calcul de la subsomption de canevas extrêmement difficile. Nous ne pouvons malheureusement pas proposer d'algorithme complet permettant de calculer la subsomption entre canevas, et nous nous contenterons ici d'indiquer quelques pistes de réflexion.

5.4.1 Vers un algorithme de subsomption

Définir la subsomption pour des canevas ayant des composants itérés ne pose pas de problème particulier, et se fait de manière extensionnelle, comme pour les canevas ne comportant que des composants simples : un canevas *T* subsume un canevas *T'* si, et seulement si, tout ensemble d'observations reconnu comme formant une instance de *T'* est aussi reconnu comme formant une instance de *T*. La difficulté que représente le *calcul* de la subsomption entre canevas provient principalement du fait qu'il ne suffit pas d'apparier les graphes de contraintes nœuds par nœuds, ce qui est déjà extrêmement coûteux[50], ou même de chercher à apparier un nœud du graphe subsumant à un sous-graphe du graphe subsumant, comme c'est le cas pour le processus de reconnaissance, ce qui est encore plus coûteux. Pour être complet, le test de subsomption devrait chercher à apparier les sous-graphes du graphe subsumant à des sous-graphes du graphe subsumé.

Remarquons tout d'abord que, étant donné les définitions précédentes, un canevas T_1 ayant un unique composant *A* et un canevas T_2 ayant un unique composant *A** sont équivalents. En effet il faut et il suffit qu'il existe parmi l'ensemble des observations au moins une observation qui appartienne à la classe *A* pour que les deux canevas soient reconnus. Si l'ensemble des observations contient une ou plusieurs séquences itérées d'instances de *A*, il existera un plus grand nombre de manière de reconnaître le canevas T_2 que le canevas T_1, mais les deux canevas seront reconnus. Ainsi, par exemple, pour les observations a_1 *m* a_2 *m* a_3 (une séquence itérée de trois instances de *A*), il existe trois façons de reconnaître T_1 et six façons de reconnaître T_2. Cette remarque nous conduit à identifier une observation de la classe *A* et une séquence itérée de longueur un d'instances de *A*.

Soient les canevas T_1 et T_2 illustrés par la figure 40. L'ensemble d'observations 1 illustré par la figure est reconnue par le canevas T_1 alors que l'ensemble d'observations 2 ne l'est pas.

[50] On l'a dit, ce problème est NP-Complet.

En revanche, les deux ensembles d'observations sont reconnus par T_2. Plus généralement, tout ensemble d'observations reconnu par T_2 l'est également par T_1. T_2 subsume donc T_1.

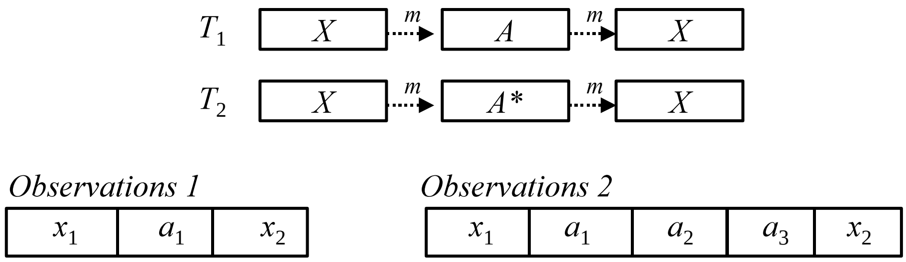


figure 40 : exemple de subsomption de canevas

La figure 41 illustre la définition de trois canevas équivalents. T_1, T_2 et T_3 représentent tous trois une séquence d'au moins deux plans. Toute séquence d'au moins trois plans pouvant être reconnue par chacun des trois canevas, les trois canevas sont équivalents. En d'autres termes, T_1 subsume T_2 qui subsume T_3 qui subsume à son tour T_1.

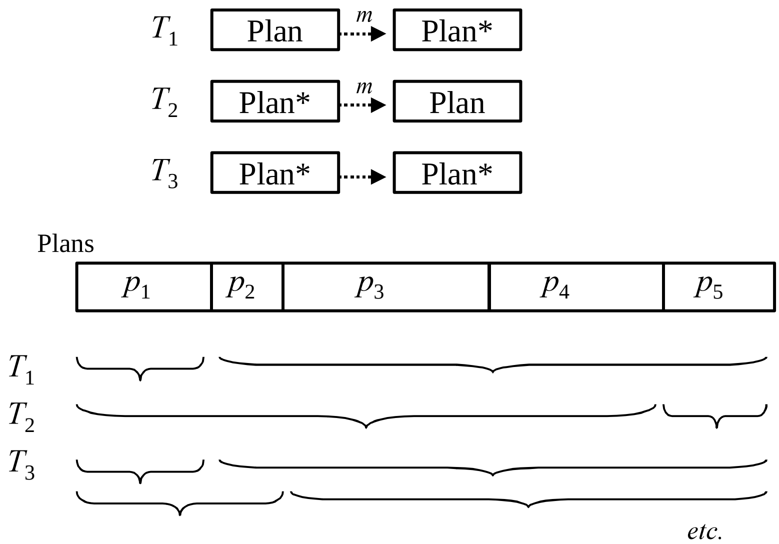


figure 41 : exemple de canevas équivalents

Si l'on identifie la relation *meets* avec un opérateur de concaténation, les définitions des trois canevas T_1, T_2 et T_3 sont trois expressions régulières. L'équivalence et la subsomption d'expressions régulières est connue, et il en a déjà été question lorsque nous avons décrit le système CLASP[51]. Une expression régulière e_1 subsume une expression régulière e_2 si, et seulement si, tout mot engendré par e_2 peut également être engendré par e_1, ou, autrement dit, si le langage représenté par e_2 est inclus dans le langage engendré par e_1. Schématiquement, pour calculer la subsomption entre e_1 et e_2, on calcule l'automate d'états finis déterministe (DFSA) produit cartésien des DFSA associés à e_1 et e_2. Dans un tel automate, les états sont de la forme <*état-dans-e_1*, *état-dans-e_2*>. Si, dans tous les états de la forme <*état-dans-e_1*, *état-final-dans-e_2*>, *état-dans-e_1* est un état final, alors à tout état final

[51] Voir « Extensions temporelles » page 38.

accessible à partir de l'état initial dans e_2 correspond un état final accessible à partir de l'état initial dans e_1, ce qui signifie que toute expression reconnue par e_2 l'est également par e_1, ou, en d'autres termes, e_2 engendre un sous-langage du langage engendré par e_1. On peut alors dire que e_1 subsume e_2. On trouvera plus de précision sur ces questions dans (Hopcroft & Ullman, 1979).

Cependant, un canevas ne peut s'exprimer sous la forme d'une expression régulière. Considérons les canevas T_1 et T_2 illustrés par la figure 42. T_1 représente une séquence d'au moins deux plans consécutifs avec un jingle entendu pendant (relation *during*) le premier plan de la séquence. Le canevas T_2 représente également une séquence d'au moins deux plans consécutifs, le jingle étant entendu cette fois pendant la première partie de la séquence, représentée par le composant associé à *Plan**. Comme le suggèrent les deux séries d'observations représentées sur la figure, toute série d'observations reconnue par le canevas T_1 l'est également par le canevas T_2, alors que la réciproque n'est pas vraie. Par conséquent, T_2 subsume T_1. Les canevas T_1 et T_2 ne peuvent être représentés par des expressions régulières et les techniques issues de la théorie des automates ne peuvent donc pas s'appliquer.

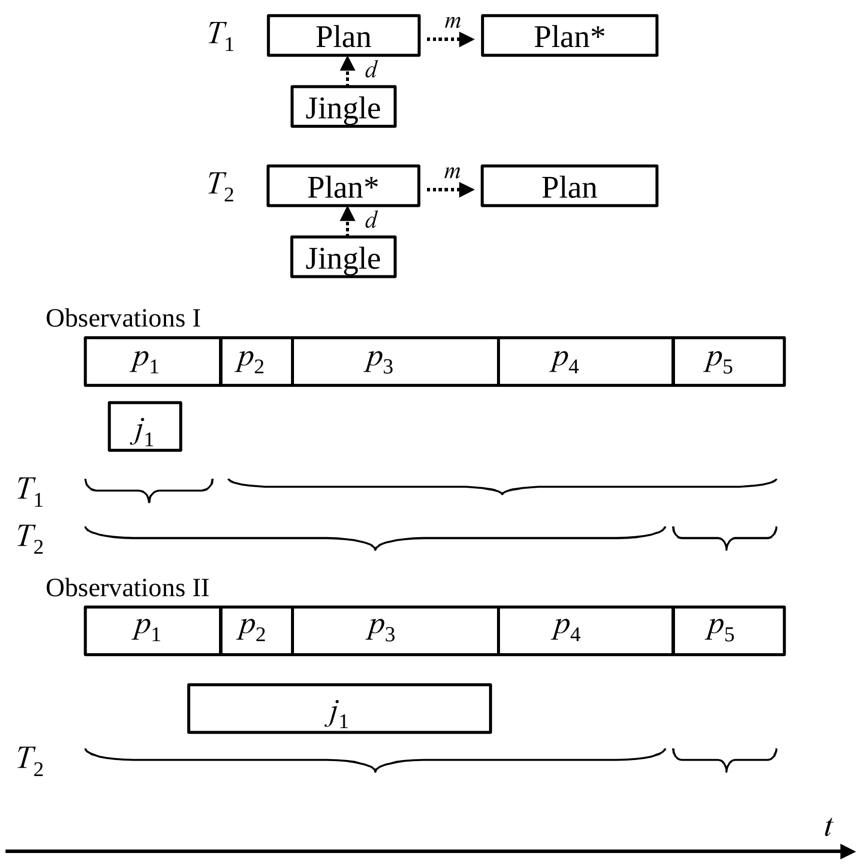


figure 42 : subsomption de canevas

La difficulté inhérente au calcul de la subsomption entre canevas provient du fait que des composants itérés figurent à la fois dans le canevas subsumant et dans le canevas subsumé.

Dans le cas où le canevas subsumé ne possède pas de composant itéré, la technique utilisée pour la reconnaissance décrite brièvement ci-dessus[52] et qui sera détaillée dans le chapitre 4 peut s'appliquer : les composants non itérés sont appariés à des composants du graphe subsumé, et les composants itérés sont appariés à des séquences itérées du graphe subsumé. Une idée d'algorithme pour le calcul de la subsomption entre canevas comportant des composants itérés consiste à remplacer dans le canevas subsumé chaque composant itéré *T** successivement par une séquence itérée de un élément associé à *T*, puis deux, jusqu'à un majorant *M* dépendant du nombre *N* de composants du graphe subsumant, *M* étant nécessairement inférieur ou égal à *N*. Si un appariement existe dans tous les cas, alors la subsomption est vérifiée. La figure 43 illustres cette procédure dans un cas simple, où le canevas subsumant T_2 ne comporte qu'un seul composant itéré. Ce composant itéré associé à *Plan** est remplacé successivement par une séquence itérée de un, deux, trois puis quatre plans consécutifs. Ces séquences sont entourées d'un rectangle sur la figure ; les appariements sont indiqués par une flèche courbe.

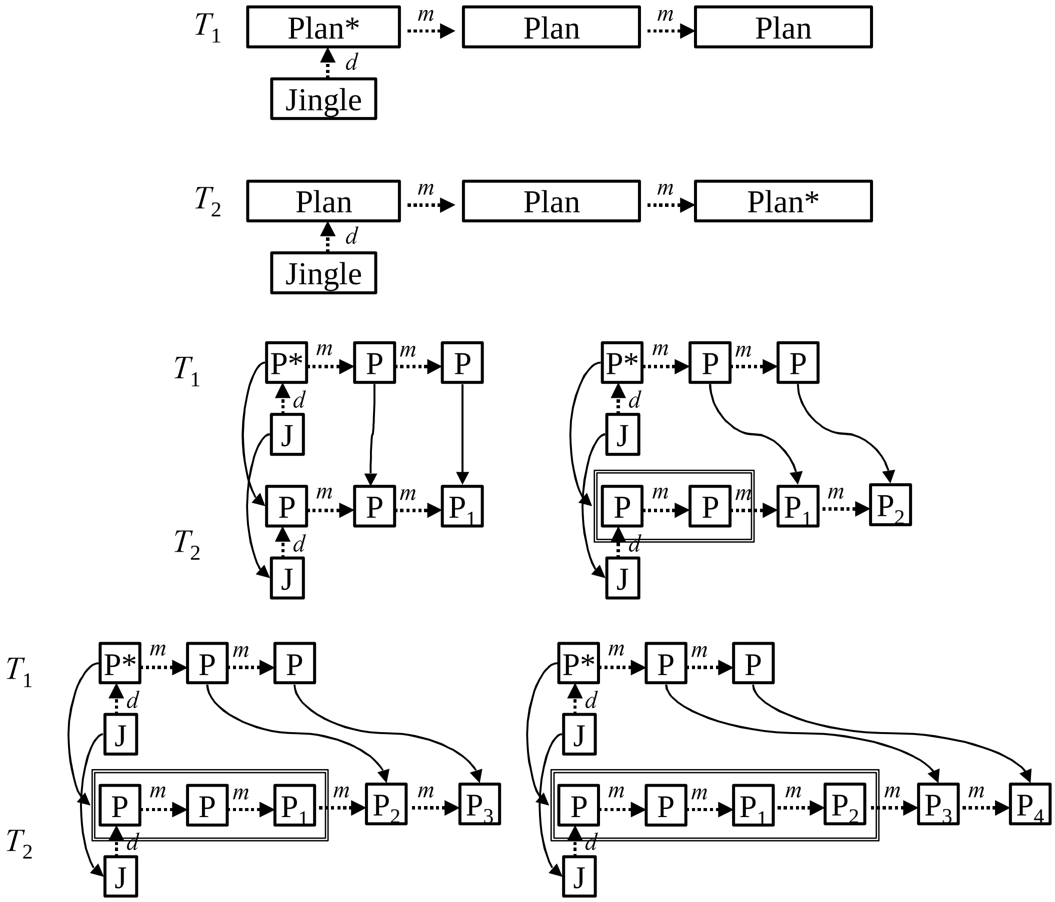


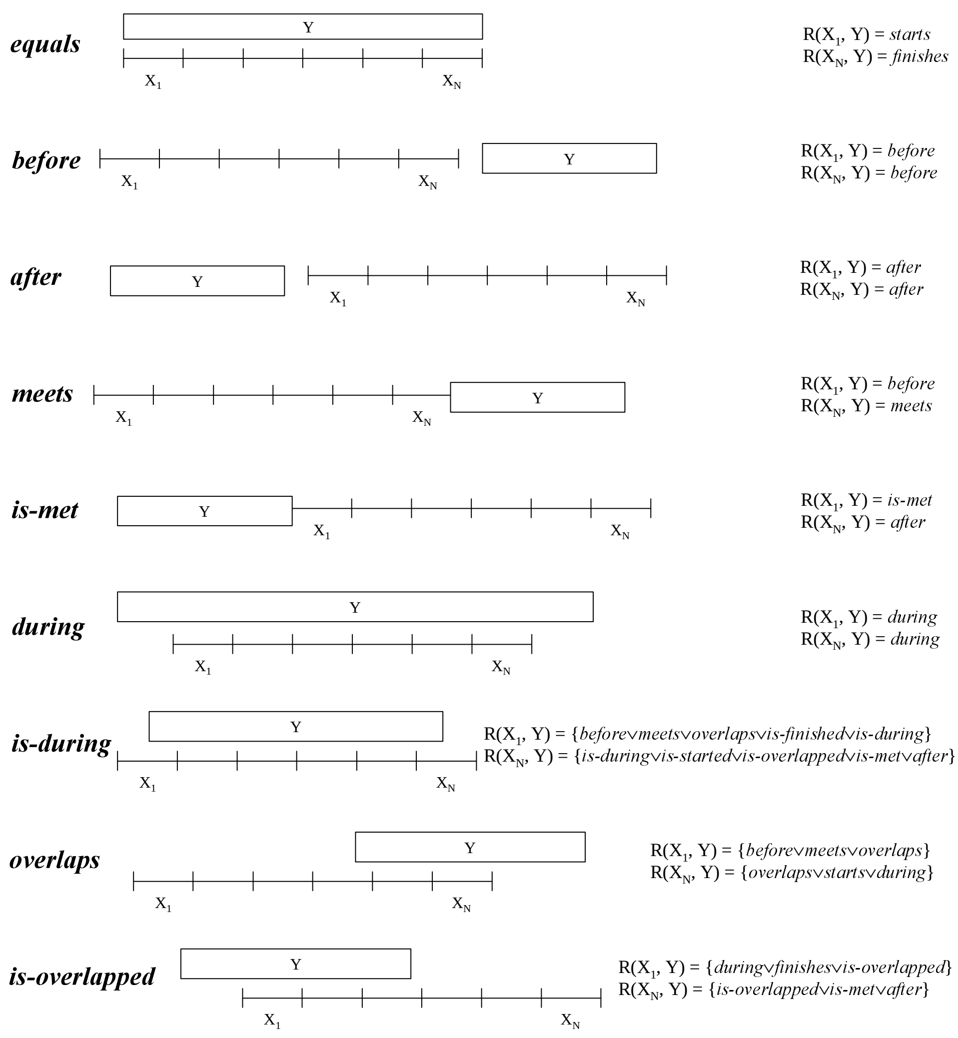
figure 43 : développement des composants itérés pour le calcul de subsomption

Lorsqu'on transforme dans le canevas subsumé un composant itéré en une séquence itérée d'une certaine longueur, il faut calculer les relations temporelles qui doivent être imposées entre les éléments de la nouvelle séquence et les autres composants. Lorsque le composant

52 Voir « Processus de reconnaissance » page 85.

itéré est développé en une séquence d'un seul élément, les relations temporelles demeurent inchangées. Lorsque la séquence comporte au moins deux éléments, nous proposons d'appliquer la méthode suivante : les relations temporelles entre le premier élément de la séquence itérée et les autres composants, ainsi que les relations entre le dernier élément de la séquence itérée et les autres composants, sont déduites des relations entre le composant itéré d'origine et les autres composants, à l'aide d'une table d'association. Les relations temporelles entre les éléments de la séquence sont quant à elles connues : les éléments adjacents de la séquence sont dans la relation *meets*. Par transitivité, on peut alors déterminer les relations temporelles manquantes, c'est-à-dire les relations temporelles entre les éléments internes de la séquence et les autres composants, en calculant par exemple la fermeture transitive du graphe de contraintes par la méthode de 3-cohérence déjà citée.

Pour déterminer les relations entre les éléments délimitant la séquence itérée et les autres composants, on utilise la table d'association illustrée par la figure 44. Ainsi, si la relation entre le canevas itéré et un autre composant est *before*, la relation entre chacune des extrémités de la séquence et cet autre composant est également *before*. La table donne ainsi pour chacune des relations élémentaires de Allen les relations correspondantes pour les deux extrémités de la séquence. Si la relation d'origine est une disjonction de relations de Allen, les relations à calculer sont les disjonctions des relations obtenues pour chacune des relations élémentaires de la relation d'origine.



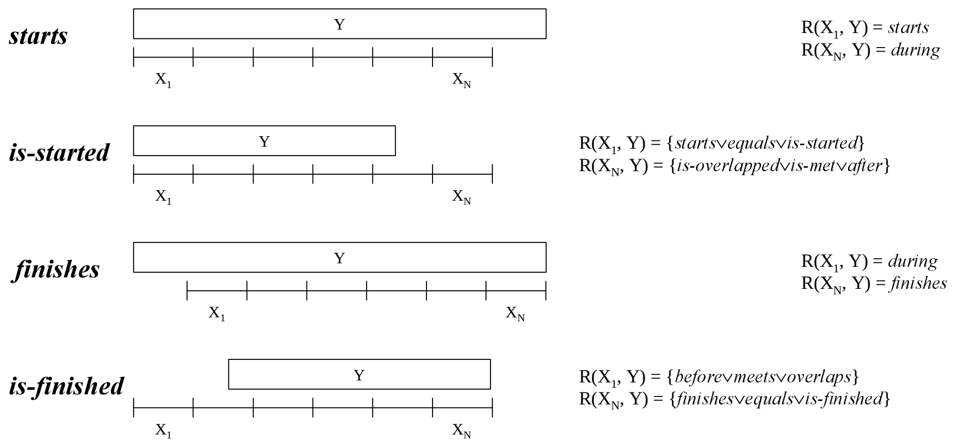


figure 44 : relations temporelles des canevas itérés développés

Des problèmes demeurent cependant. Il faudrait tout d'abord montrer que le majorant M existe. Ensuite, il faudrait pouvoir calculer un majorant optimum. Sur l'exemple illustré par la figure 43, on voit qu'il suffirait de remplacer le composant itéré de T_2 par une séquence itérée de un à trois éléments pour que la subsomption soit vérifiée. Enfin, le nombre d'appariements à vérifier est extrêmement grand, dépendant du nombre de composants du canevas subsumant et du nombre de composants du canevas subsumé.

L'introduction de l'opérateur d'itération introduit une grande différence entre la difficulté que représente la reconnaissance des occurrences d'un canevas parmi un ensemble d'observations et la difficulté que représente le calcul de la subsomption entre canevas. Nous abordons dans la section suivante quelques questions sur l'intérêt de pouvoir organiser les canevas en taxinomie selon leurs relations de subsomption.

5.4.2 Discussion

La relation de subsomption entre canevas, lorsqu'elle peut être calculée, permet d'organiser les canevas en taxinomie. On l'a vu, le calcul de cette relation est difficile et coûteux. Schématiquement, plus on cherche à rendre les canevas expressifs, plus le calcul de la subsomption est difficile, phénomène bien connu qui se rencontre par exemple dans le domaine des logiques de descriptions où l'ajout d'un nouveau constructeur syntaxique rend presque toujours le calcul de subsomption plus complexe, voire indécidable (Nebel, 1990b). Nous posons ici la question de l'intérêt de conserver une taxinomie de canevas et des conséquences de l'abandon de cette organisation.

Tout d'abord, organiser les canevas en taxinomie présente les mêmes avantages que l'organisation de concepts ontologiques en taxinomie. Cela permet d'avoir une connaissance plus approfondie de la bibliothèque de canevas, d'en avoir une meilleure gestion et de lui assurer une plus grande évolutivité. En d'autres termes, il est toujours bon de pouvoir organiser les connaissances qu'on met en œuvre, et les arguments invoqués par (Brachman *et al.*, 1991) en faveur du système CLASSIC s'appliquent ici : représentation sous la forme d'objets bien formés, descriptions partielles ou incomplètes, etc.

D'un point de vue plus pratique, une question importante concernant la nécessité ou non d'être en mesure de calculer la subsomption entre canevas est de savoir si des assertions peuvent être faites qui portent directement sur les canevas, et non pas seulement sur les concepts d'événements élémentaires, c'est-à-dire si des instances d'un canevas *T* peuvent être définies par l'utilisateur sans qu'il se préoccupe des objets – ou événements – qui composent ces instances. Si c'est le cas, l'absence de relation de subsomption explicite entre canevas peut conduire à une forme d'incomplétude. Soient deux canevas *T* et *T'*, avec *T* subsumant *T'* ($T \succ T'$). Toute instance déclarée de *T'* ne sera pas prise en compte dans le raisonnement comme une instance de *T*, et, par conséquent, les inférences portant sur des instances de *T* ne seront pas faites ; le raisonnement sera donc incomplet. En revanche, si les observations ne sont faites que sur les concepts d'événements élémentaires, et si la procédure de reconnaissance est complète, alors tout ensemble d'observations reconnu comme instance de *T'* sera également reconnu comme instance de *T*. Dans le cas qui nous occupe, les observations sont des résultats d'algorithmes d'analyse de relativement bas niveau, comme des plans, des occurrences de jingle sonores ou de logos visuels, des régions de l'image correspondant à des visages ou des régions de l'image correspondant à du texte, etc. Toutes ces informations peuvent être facilement représentées dans un formalisme de logiques de descriptions tel que nous l'avons décrit, sans qu'il soit nécessaire de faire appel à des relations temporelles complexes. Du point de vue de la complétude du processus de reconnaissance de canevas, nous pouvons donc nous dispenser d'organiser les canevas en taxinomie selon leurs liens de subsomption.

L'inconvénient réside alors dans le coût de calcul, puisqu'il faut effectuer la reconnaissance pour tous les canevas. Lorsque les relations de subsomption ont été calculées, elles sont établies une fois pour toutes. D'autre part, si un canevas *T* a été reconnu, on sait que tous les canevas subsumant *T* le seront également, par définition de la subsomption, et il est alors inutile d'effectuer la reconnaissance sur ces canevas. Symétriquement, si un canevas *T* n'est pas reconnu, les canevas subsumés par *T* ne le seront pas non plus. Il est alors possible d'éviter d'effectuer la reconnaissance pour un certain nombre de canevas. Si l'on dispose d'une heuristique permettant de choisir judicieusement les canevas à reconnaître en premier lieu – les plus généraux ou les plus spécifiques d'abord, par exemple –, le processus de reconnaissance peut être sensiblement optimisé.

Enfin, l'organisation des canevas en taxinomie permet une définition modulaire des canevas. Soit le canevas illustré par la figure 45, qui définit la structure d'un journal télévisé par la succession d'un générique de début, d'une séquence de reportages et d'un générique de fin. Cette définition très générale peut s'appliquer à bon nombre de journaux différents. En revanche, les définitions des génériques et des reportages ont de fortes chances de varier d'une chaîne à l'autre, et la définition d'un reportage de « France 2 », *F2-Report*, sera différente de la définition d'un reportage de « France 3 », *F3-Report*. Si *F2-Report* et *F3-Report* sont tous deux subsumés par *Report*, le canevas de la figure sera valide aussi bien pour « France 2 » que pour « France 3 » et pourra être reconnu pour les deux types de reportage. Autrement, autant de canevas de journal télévisé devront être définis qu'il y a de types de reportages.

figure 45 : canevas d'un JT simplifié

Il serait possible de contourner cette limitation d'une part en introduisant des canevas *primitifs*, au sens des logiques de descriptions, c'est-à-dire des canevas définis par des conditions nécessaires mais non suffisantes, et d'autre part en définissant les liens de subsomption de manière explicite. *Report* peut ainsi être défini comme un canevas primitif, et *F2-Report* et *F3-Report* comme sous-canevas de *Report*. La définition d'un journal reste ainsi valable dans les deux cas. Cela ne constitue cependant qu'une facilité d'écriture et ne préjuge pas de la manière dont le processus de reconnaissance, cherchant à reconnaître des instances du canevas *Report*, devra chercher toutes les instances de *F2-Report* et de *F3-Report*.

En fin de compte, il est certes regrettable de ne pas être capable de calculer la subsomption entre deux canevas, en particulier en ce qui concerne le dernier point évoqué ci–dessus – la possibilité d'exprimer un canevas « abstrait », un reportage par exemple, subsumant plusieurs autres canevas. Cette lacune constitue le prix à payer d'un formalisme suffisamment expressif et n'en représente pas une limitation fondamentale, le principal problème restant le celui de la reconnaissance.

6 Enrichir le langage par de nouvelles contraintes

L'introduction de l'opérateur d'itération « * » dans la définition des canevas nous conduit à renoncer à la possibilité de calculer la subsomption entre deux canevas. La bibliothèque de canevas devient donc en quelque sorte un catalogue de formes à reconnaître. On l'a vu, les canevas sont des descriptions de classes de séquences vidéo et le problème de la reconnaissance des canevas s'exprime comme un problème de satisfaction de contraintes. Si l'on s'affranchit de la nécessité de calculer la subsomption entre canevas, enrichir le langage de description revient à chercher de nouveaux opérateurs qui soient à la fois pertinents et tels que le problème de la reconnaissance puisse continuer à s'exprimer comme un CSP.

La figure 46 illustre un exemple simplifié de journal télévisé où les reportages sont des séquences de plans (P_i) délimitées par des fondus (D_i, pour *dissolve*).

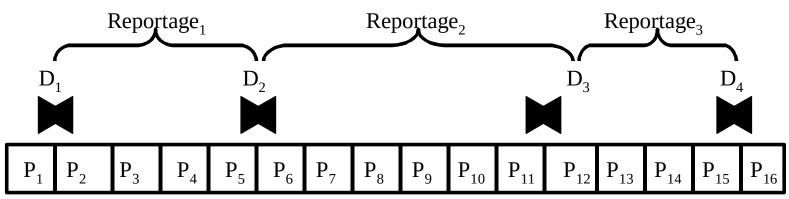
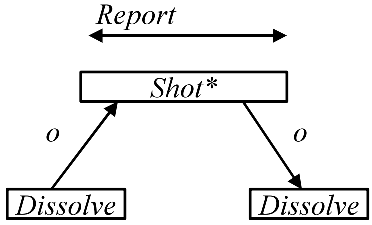


figure 46 : exemple simplifié de journal télévisé

Le canevas illustré par la figure 47, définit une séquence de reportage comme une séquence de plans délimitée par des fondus enchaînés.

Report
nodes
s_1 [*Shot**]
s_2 [*Dissolve*]
s_3 [*Dissolve*]
constraints
s_2 *overlaps* s1
s_1 *overlaps* s3



s_1 *equals* this

figure 47 : définition d'un reportage

Ce canevas ne correspond cependant pas exactement à la définition souhaitée. En effet, l'application de ce canevas à l'exemple de la figure 46 donne six reportages au lieu des trois escomptés. L'ensemble de ces six reportages est illustré par la figure 48.

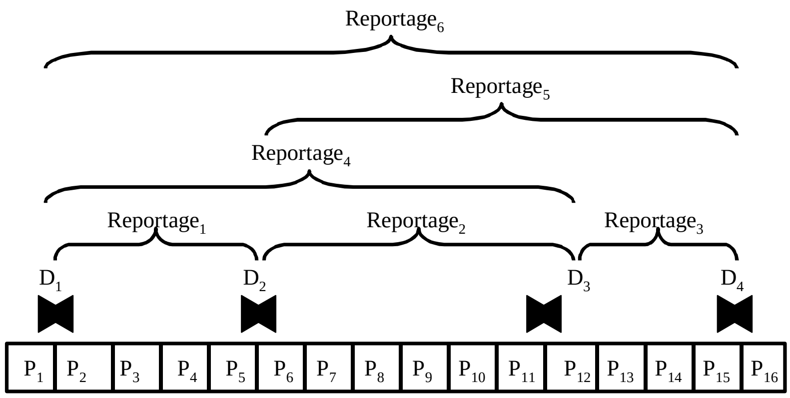


figure 48 : trop de reportages

Schématiquement, le canevas ne donne d'indication que sur les extrémités d'un reportage que constituent les fondus enchaînés. D'une certaine manière, la séquence de plans n'est utile que pour déterminer l'extension temporelle des reportages. La contrainte temporelle liant les deux fondus enchaînés, obtenue après propagation des contraintes, est la relation temporelle *before*, qui est transitive. Intuitivement, la relation temporelle qu'on souhaiterait exprimer est une sorte de relation de succession temporelle, relation qui peut être vue comme une relation *before* non transitive. En d'autres termes, un reportage peut être identifié comme : « une séquence de plans délimitée par deux fondus enchaînés successifs, avec un jingle pendant le premier plan ».

Cette notion de succession implique une négation : deux fondus enchaînés successifs sont deux fondus enchaînés f_1 et f_2 mis en relation par *before* tels qu'il n'y ait pas d'autre fondu enchaîné qui soit avant (*before*) f_2 et après (*after*) f_1. Nous généralisons la notion de succession en introduisant un type de contrainte que nous appelons « no … between », qui interdit qu'une instance d'un concept C – ou d'un canevas T – se trouve temporellement entre deux événements donnés.

La forme syntaxique de cette contrainte est :

« no » C « between » s_i s_j

où s_i et s_j sont des étiquettes identifiant deux composants du canevas, et C un nom de concept d'événement élémentaire ou un nom de canevas borné[53]. La prise en compte par le solveur de contraintes de ces nouvelles contraintes sera détaillée au chapitre suivant[54].

---

[53] Voir « Canevas bornés » page 83.
[54] Voir « Enrichir le langage par de nouvelles contraintes » page 93.

Schématiquement, lorsque les deux variables associées à s_i et s_j sont instanciées, il suffit de chercher dans le graphe d'observation si une instance de *C* ne viole pas la contrainte.

La définition d'un reportage devient alors :

Report
nodes
s_1 [*Shot**]
s_2 [*Dissolve*]
s_3 [*Dissolve*]
constraints
s_2 *overlaps* s_1
s_1 *overlaps* s_3
s_1 *equals* this
no *Dissolve* between s_2 s_3

Introduire dans le langage de définition des canevas ce nouveau type de contraintes « no *C* between s_i s_j » implique qu'au moment de la reconnaissance toutes les instances du concept *C* soient connues. Il s'agit là d'une hypothèse de monde clos (*Closed-World Assumption*) qu'il est nécessaire de faire lorsqu'on veut vérifier la validité de la négation d'une formule logique. Ceci est à rapprocher de la clôture des rôles en CLASSIC. Ainsi, l'expression classique suivante décrit un plan du (ou des) présentateur(s) comme un plan de studio qui montre un ou deux personnages :

```
(cl-define-concept 'Anchor-Person-Shot
  '(and
    Studio-Shot
    (all has-character Person)
    (at-least 1 has-character)
    (at-most 2 has-character)))
```

Soit maintenant l'expression qui crée un plan de studio où apparaissent deux personnages, Claire et Patrick :

```
(cl-create-ind 'shot-17
  '(and
    Studio-Shot
    (fills has-character Patrick)
    (fills has-character Claire)))
```

Le plan shot-17 n'est pas classifié comme étant un plan de présentateur, car CLASSIC supposant que le monde est ouvert (*Open World Assumption*), le système peut s'attendre à ce que d'autres personnages soient ajoutés au plan. Il est possible de clore le rôle has-character, indiquant par-là qu'aucun personnage ne sera ajouté par la suite :

```
(cl-ind-close-role @shot-1 @has-character)
```

Le plan shot-17 est alors classifié comme plan de présentateur, c'est-à-dire comme instance du concept Anchor-Person-Shot. De la même manière, il faut pour vérifier une contrainte

« no *C* between s_1 s_2 » faire l'hypothèse que le monde est clos, tout au moins quant au concept *C*.

D'autres types de contraintes pourraient facilement être ajoutés, comme par exemple des contraintes de cardinalité sur le nombre d'éléments des séquences itérées. Il serait également possible d'introduire de la même manière des contraintes numériques sur les bornes temporelles des événements, ce qui permettrait par exemple d'identifier un « segment musical commençant moins de trente secondes avant un changement de plan » pour reprendre une des règles proposées par (Aigrain *et al.*, 1997)[55]. Il convient toutefois de remarquer que, à moins de changer la logique temporelle sous-jacente, ces contraintes ne seraient pas prises en compte au moment du calcul de la couverture minimale du graphe de contraintes temporelles et que par conséquent d'éventuelles incohérences ne seraient pas détectées. Si le cas se produisait, le processus de reconnaissance parcourrait dans son ensemble l'espace de recherche avant de répondre que le problème n'admet pas de solution.

7 Conclusion

Nous avons présenté le problème de la classification de séquences vidéo caractéristiques d'une collection de programmes de télévision comme un problème de reconnaissance de plans (au sens de *plan recognition*). Nous avons étendu l'expressivité du système T-Rex, système de reconnaissances de plan construit au-dessus d'un formalisme de logique de descriptions. Comme dans T-Trex, les événements élémentaires – les segments audio ou vidéo résultats d'algorithme d'analyse – sont représentés par des individus d'une logique de descriptions. Les classes de séquences à reconnaître sont représentées par des *canevas* qui décrivent un ensemble de contraintes devant être respectées par les segments observés. Outre les contraintes temporelles de Allen, sur lesquelles reposent les définitions de plans dans T-Rex, les définitions de canevas font intervenir un opérateur d'itération noté « * » désignant une séquence itérée d'événements temporellement conjoints, ainsi qu'une contrainte négative notée « no … between » interdisant l'occurrence d'événements d'un certain type entre deux composants d'un canevas. Ce faisant, nous avons constaté la difficulté de maintenir une taxinomie de canevas en raison de la difficulté qu'il y a à calculer la subsomption entre deux canevas. Cette limitation n'entraîne cependant que des restrictions mineures aux possibilités du système.

Le problème de la reconnaissance de canevas s'exprime comme un problème de satisfaction de contraintes. Dans le chapitre suivant, nous décrivons le système Clavis (*Classification of Audiovisual Sequences*), construit autour du système de satisfaction de contraintes BackJava, qui met en œuvre les idées que nous venons de décrire.

[55] Voir « Utilisation de règles de montage » page 17.

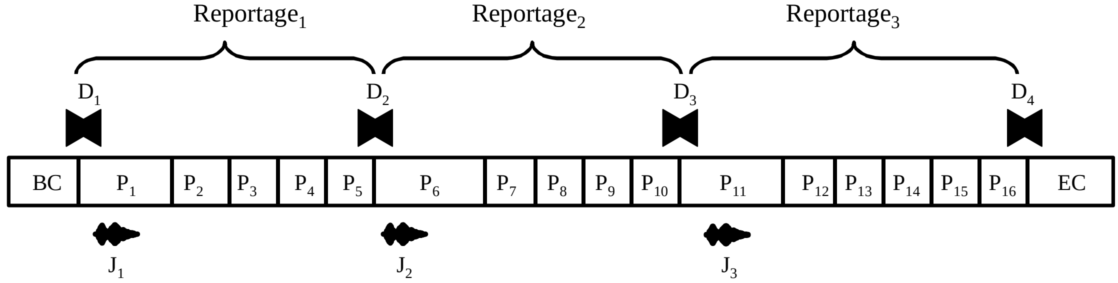
Chapitre 4
Le système Clavis

Le système Clavis (*Classification of Audiovisual Sequences*) implémente une méthode efficace de reconnaissance des canevas tels que nous les avons présentés au chapitre précédent. Clavis est construit autour du système de satisfaction de contraintes BackJava (Roy *et al.*, 1999). Nous décrirons tout d'abord les aspects généraux du système – le langage de description des canevas, les étapes de la reconnaissance, la communication avec le système de logique de descriptions CLASSIC et la représentation interne des canevas – avant de détailler les éléments du système proprement liés à la satisfaction de contraintes, c'est-à-dire principalement la représentation des séquences itérées et les méthodes de filtrage des contraintes temporelles impliquant de telles séquences (Carrive *et al.*, 2000c).

1 Un langage de description des canevas

Les classes de séquences-type d'une collection de programmes de télévision sont décrites par des canevas. Les canevas ont été définis dans le chapitre précédent. Nous illustrons ici par un exemple simple et idéalisé la manière dont les canevas permettent de décrire ces classes de séquences. Cet exemple nous permettra de décrire tous les éléments pouvant entrer dans la définition d'un canevas. Les définitions de canevas sont présentées dans la syntaxe textuelle utilisée dans le système Clavis, et sont le plus souvent accompagnées d'un schéma. Les abréviations utilisées pour les contraintes temporelles sont les abréviations usuelles et sont illustrées par le tableau 1[56].

La figure 49 reprend l'exemple du chapitre précédent[57] en le complexifiant légèrement. Un journal télévisé simplifié est composé d'un générique de début (BC, sur la figure, pour *Begin Credits*), d'une séquence de reportages et d'un générique de fin (EC, pour *End Credits*). Un reportage est une séquence de plans P_i délimitée par deux fondus enchaînés successifs – au sens de la succession temporelle définie plus haut[58] –, et un jingle sonore est entendu pendant le premier plan.



[56] Voir page 54.
[57] Voir figure 46 page 93
[58] Voir « Enrichir le langage par de nouvelles contraintes » page 93.

figure 49 : structure simplifiée d'un journal télévisé

Le canevas décrivant un reportage est illustré par la figure 50 ci-dessous.

Report
nodes
s_1 [*Shot*]
s_2 [*Shot**]
s_3 [*Jingle*]
s_4 [*Dissolve*]
s_5 [*Dissolve*]
constraints
s_1 *meets* s_2
s_3 {*starts during*} s_1
s_4 *overlaps* s1
s_2 *overlaps* s5
s_1 *starts* this
s_2 *finishes* this
no *Dissolve* between s_4 s_5

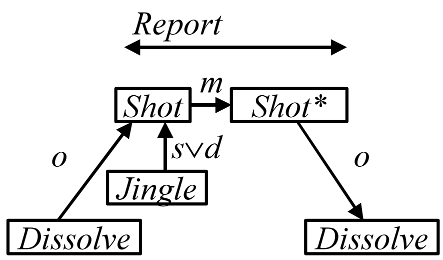


figure 50 : canevas d'un reportage

La définition complète des canevas a été donnée au chapitre précédent. En résumé, la première partie de la définition identifie par une étiquette s_i les composants du canevas, lesquels composants induisent les sommets d'un graphe de contraintes, et associe à chacun de ces composants un concept *C* – ou un autre canevas *T*. L'opérateur d'itération « * » indique une séquence itérée d'instances de *C* – ou *T*. Pour être reconnu, chaque sommet simple – non itéré – devra être associé à une instance de *C* – ou de *T* – et chaque sommet itéré devra être associé à une séquence itérée d'instances de *C* – ou de *T*. La seconde partie donne les contraintes temporelles, ainsi que les contraintes « no … between » devant être respectées par les observations associées au sommet du graphe. Le mot clé « this » représente une instance du canevas en cours de définition, et les contraintes temporelles impliquant « this » constituent un moyen de spécifier l'extension temporelle des instances du canevas.

Pour finir de décrire l'exemple illustré par la figure 49, nous donnons la définition du canevas qui décrit la structure temporelle du journal télévisé dans son ensemble. Un journal est simplement constitué d'un générique de début (*BeginCredits*) suivi d'une séquence itérée de reportages (*Report*) et terminé par un générique de fin (*EndCredits*). La figure 51 illustre ce canevas.

Newscast
nodes
s_1 [*BeginCredits*]
s_2 [*Report**]
s_3 [*EndCredits*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_1 *starts* this
s_3 *finishes* this

figure 51 : canevas du journal télévisé

La reconnaissance du canevas *Newscast* illustré ci-dessus se fait récursivement en commençant par les canevas itérés les plus « profonds ». Une première phase de reconnaissance extrait les reportages et les ajoute aux observations, avant qu'une seconde phase de reconnaissance cherche à reconnaître la structure complète du journal. La section suivante décrit les principales étapes de la reconnaissance, étapes qui seront détaillées par la suite.

Dans la suite de ce chapitre, nous ferons l'hypothèse que les observations sont précisément datées, c'est à dire que chaque observation est associée à une période temporelle unique, et que par conséquent les relations temporelles entre les observations sont toujours des relations temporelles élémentaires et non des disjonctions de relations temporelles. Cela correspond d'une part aux cas réels que nous avons eu à traiter, où les algorithmes d'analyse datent précisément les segments qu'ils détectent, et cela nous permet d'autre part de présenter plus simplement les algorithmes mis en œuvre. Cependant, la plupart des mécanismes de résolution du système Clavis ont été conçus pour traiter les disjonctions de relations temporelles entre les observations. Nous reviendrons sur cette question en fin de chapitre[59].

2 Etapes de la reconnaissance

Les principales étapes de la reconnaissance d'un canevas sont reportées sur la figure 52. Pour reconnaître les occurrences d'un canevas dans un graphe d'observations, il faut tout d'abord disposer de la taxinomie des concepts d'événements élémentaires. La gestion de cette taxinomie est du ressort du système CLASSIC ; la taxinomie doit être communiquée à Clavis. Plusieurs moyens de communication entre CLASSIC et Clavis ont été mis en place, selon que la taxonomie complète, c'est-à-dire concepts et liens de subsomption, sont communiqués à Clavis, ou selon que seuls les concepts sont connus de Clavis, les relations entre concepts restant du ressort de CLASSIC. De plus, une bibliothèque de classes Java a été développée pour représenter une taxinomie de concepts indépendamment de CLASSIC. Les canevas à reconnaître sont ensuite mis en mémoire à partir de leur description textuelle ; la représentation interne sera présentée plus loin. Le graphe d'observations est alors constitué, par exemple à partir de résultats d'algorithmes d'analyse

[59] Voir « Observations non précisément datées » page 128.

fournis à Clavis sous la forme de descriptions AEDI[60]. Le problème de satisfaction de contraintes (CSP) correspondant à la reconnaissance d'un canevas *T* est résolu en plusieurs étapes. En premier lieu, les occurrences des canevas imbriqués de *T* qui n'ont pas été expansés, en particulier les canevas itérés, sont recherchées dans le graphe d'observation par la résolution du CSP correspondant. La reconnaissance d'un canevas se fait donc de manière récursive, en commençant par les canevas les plus « profonds ». Les instances reconnues sont alors ajoutées au graphe d'observation. La deuxième étape du processus de reconnaissance est la création des variables du CSP, et en particulier la création des variables correspondant aux composants itérés du canevas pour lesquelles le calcul du domaine est complexe. La troisième étape est la création des contraintes : 1) contraintes de subsomption imposant que chaque composant simple du canevas soit associé à une instance du concept associé au composant ; 2) contraintes temporelles, entre deux variables simples, entre une variable simple et une variable itérée, et entre deux variables itérées ; 3) contraintes « no … between ». La dernière étape de la reconnaissance consiste à chercher et interpréter les solutions du CSP.

1. Lecture de la taxinomie

2. Instanciation du canevas

3. Instanciation du graphe d'observation

4. Constitution du CSP

4.1 Reconnaissance des canevas non expansés (nouveau CSP) et ajout des résultats dans le graphe d'observation

4.2 Création des variables

4.2.1 Variables simples

4.2.2 Variables itérées

4.3 Création des contraintes

4.3.1 Contraintes de subsomption

4.3.2 Contraintes temporelles entre deux variables simples

4.3.4 Contraintes temporelles entre une variable simple et une variable itérée

4.3.4 Contraintes temporelles entre deux variables itérées

4.3.5 Contraintes « no … between »

4.4 Interprétation des résultats

figure 52 : principales étapes de la reconnaissance

[60] Voir « A.E.D.I » page 24.

3 *Communication CLASSIC/Clavis*

La taxinomie des concepts d'événements élémentaires tels que *Shot*, *Jingle*, *Face Region*, etc., est représentée au sein du système CLASSIC. Chacun de ces concepts peut être associé à un ou plusieurs composants d'un canevas, composant simple ou itéré. Une fois les observations élémentaires classifiées, opération normalement réalisée par CLASSIC, leur rôle ne consiste plus qu'à discriminer les observations : un composant simple associé au concept *C* ne pourra être apparié qu'à une instance de *C*. Schématiquement, le rôle d'un concept n'est plus alors que de permettre de répondre à la question : « cette observation est-elle ou non une instance de ce concept ? ».

Une difficulté technique réside dans le fait que les systèmes CLASSIC et Clavis sont écrits dans deux langages différents (Lisp et Java). Les deux systèmes doivent donc communiquer. Les besoins d'échange, illustrés par la figure 53, sont les suivants :

- Les concepts d'événements élémentaires, représentés par une taxinomie CLASSIC (*TBox*), doivent être connus de Clavis ;
- Les observations venant de l'analyse, représentées par des individus CLASSIC (*ABox*), doivent être connues de Clavis ;
- La réponse au test d'appartenance d'une observation *o* à l'extension d'un concept *C* – test d'instanciation – doit être connue de Clavis.

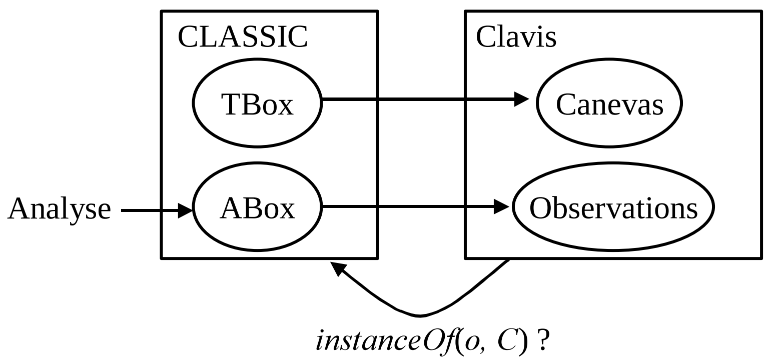


figure 53 : communication entre CLASSIC et Clavis

Le choix d'un moyen de communication entre les deux systèmes repose sur le choix d'un compromis entre la taille des données transférées de CLASSIC vers Clavis et l'importance des traitements laissés à CLASSIC. Plus simplement, si suffisamment d'information est transférée de CLASSIC à Clavis, le test d'instanciation peut être effectué au sein de Clavis. Ces deux principes sont illustrés par la figure 54 : (1) Clavis ne possède qu'une référence vers les concepts et les observations ; (2) Clavis possède également les liens de subsomption entre concepts ainsi que pour chaque observation la liste de ses parents les plus spécifiques.

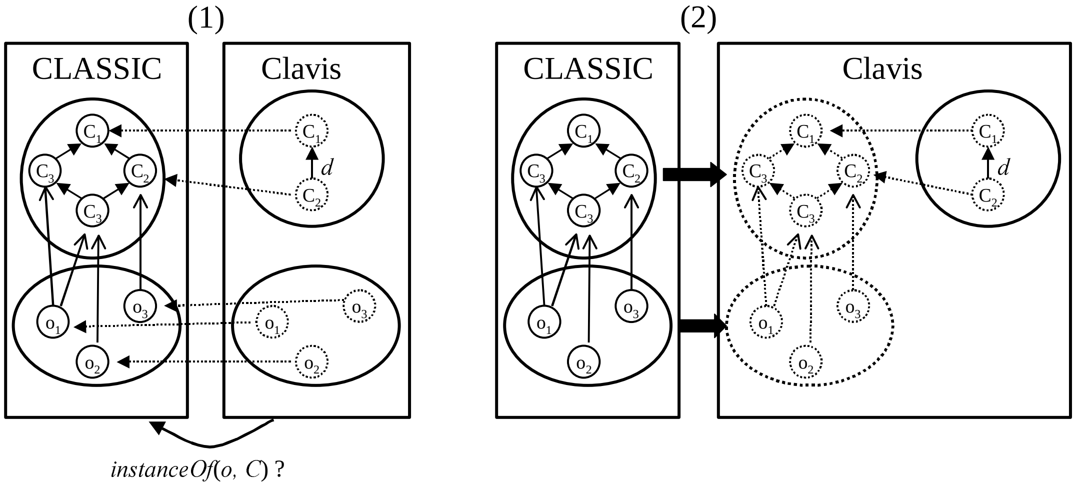


figure 54 : choix d'implémentation pour la communication CLASSIC/Clavis

La solution (1) est plus légère à mettre en œuvre mais présente l'inconvénient de faire appel à CLASSIC pour chaque test d'instanciation, cet appel étant coûteux puisque se faisant entre deux systèmes hétérogènes. La solution (2) requiert une mise en œuvre plus importante mais présente plusieurs avantages. D'une part, les liens de subsomption et les liens d'instanciation sont lus en même temps que les concepts et les observations, dans une phase d'initialisation. La quantité de données à transférer n'est guère plus importante que dans la solution (1) et le test d'instanciation peut se faire indépendamment de CLASSIC. D'autre part, il est ainsi possible en cours de développement ou pour des cas simples d'utiliser des concepts et des observations ne provenant pas de CLASSIC, ce qui autorise une version « légère » du système totalement indépendante de CLASSIC.

Pour implémenter ces deux solutions, nous avons développé :

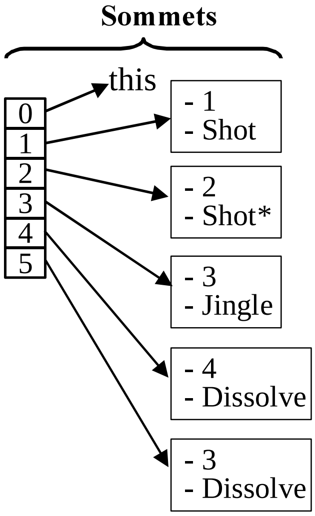
- une bibliothèque de classes Java permettant la représentation de concepts terminologiques, que ce soit dans le cas d'une simple référence vers un concept CLASSIC ou dans le cas d'une taxinomie explicite incluant les liens de subsomption ;
- un protocole de communication entre CLASSIC et Clavis, utilisant les bibliothèques de communication réseau par *sockets* disponibles dans les deux langages et permettant par exemple d'interroger CLASSIC sur l'instanciation d'une observation par un concept, sur les parents directs d'un concept, sur la subsomption entre deux concepts, etc.

4 Représentation des canevas

La représentation interne des canevas est dans son principe assez simple. Un canevas est schématiquement un graphe de contraintes temporelles. La figure 55 illustre de manière simplifiée la représentation interne du canevas de la figure 50[61]. Les sommets du graphe sont numérotés et représentés par un vecteur de sommets. L'indice 0 est réservé pour le

[61] Voir page 98.

canevas lui-même, afin de pouvoir spécifier de manière homogène les contraintes imposées entre les instances du canevas et leurs composants. D'autre part, chaque sommet indique un concept, qui peut être soit un concept d'événement élémentaire, soit un autre canevas. Les différents types de sommets – sommets simples élémentaires, sommets simples composés, sommets itérés élémentaires et sommets itérés composés – sont représentés par une hiérarchie de classes Java. Chaque contrainte temporelle est représentée par un vecteur de 13 bits contenu dans un entier court, chaque élément du vecteur dénotant une des 13 relations élémentaires de Allen selon une convention arbitraire[62]. L'ensemble des contraintes est représenté par une matrice 2×2 d'entiers courts. La matrice étant anti-symétrique, on peut se contenter de ne représenter explicitement qu'une moitié de la matrice. Les nombres indiqués sur la figure indiquent les relations temporelles du canevas en notation hexadécimale. Notons que les relations temporelles indiquées sur la figure sont les relations temporelles obtenues après complétion et minimalisation du graphe de contraintes.



	0	1	2	3	4	5
0	1					
1	200	1				
2	800	100	1			
3	208	208	2	1		
4	20	20	2	10B2	1	
5	40	4	40	4	4	1

figure 55 : représentation interne simplifiée d'un canevas

Ainsi, la relation temporelle entre les sommets 4 (*Dissolve*) et 3 (*Jingle*) du canevas est représentée par l'entier 4274 – 10B2 en hexadécimal et 1000010110010 en binaire – et représente la relation $\{$*before* $\vee$ *is-during* $\vee$ *overlaps* $\vee$ *meets* $\vee$ *is-finished*$\}$.

4.1 Imbrication de canevas

Comme on l'a vu au chapitre précédent, les canevas imbriqués peuvent être expansés[63], c'est-à-dire que les composants d'un sous-canevas *T'* d'un canevas *T* peuvent être « remontés » dans la définition de *T*. Le principe d'expansion consiste simplement à intégrer dans le graphe de contraintes temporelles les composants terminaux des sous-canevas et à propager les contraintes imposées entre les sous-canevas et leurs composants.

---

[62] Voir « Implémentation des relations temporelles » page 54.
[63] Voir « Imbrication de plans » page 76.

La figure 56 reprend l'exemple donné pour l'imbrication de canevas dans le chapitre précédent[64], où la fin d'un journal télévisé (*Fin-de-JT*) est définie comme un *Reportage* suivi d'une *Conclusion*, un reportage étant défini par un plan du *Présentateur* précédant un *Volet* et la conclusion par un plan du présentateur suivi du *Générique*. Les composants non terminaux sont signalés sur la figure par des rectangles pointillés et les contraintes temporelles déduites par des flèches pointillées.

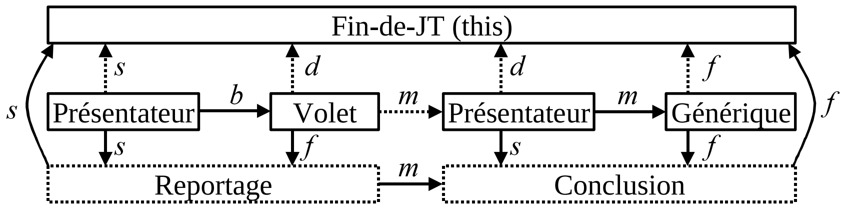


figure 56 : expansion des sous-canevas et propagation des contraintes

Par composants *terminaux*, nous entendons les composants qui ne peuvent être expansés. Il s'agit :

- des composants associés à des concepts d'événements élémentaires ;
- des composants itérés ;
- des composants sur lesquels porte une contrainte « no … between ».

La propagation des contraintes se fait en fusionnant les différents graphes de contraintes – celui du canevas et ceux des sous-canevas expansés – puis en calculant la couverture minimale du graphe ainsi obtenu par l'algorithme de 3-cohérence déjà mentionné[65]. Cette méthode a été choisie essentiellement pour des raisons de simplicité de mise en œuvre. Elle présente d'une part l'inconvénient de ne pas être complète dans tous les cas, et d'autre part celui d'induire des contraintes redondantes. Il serait certainement possible de ne propager que certaines contraintes – celles définies explicitement – et ainsi de minimiser le nombre de contraintes redondantes.

4.2 Autres aspects

Outre les composants et les contraintes temporelles, la représentation interne des canevas comporte :

- Une trace de la structure des canevas expansés. La reconnaissance ne porte que sur les canevas non expansés, et il est nécessaire de pouvoir reconstituer après la reconnaissance les instances des canevas intermédiaires. Ainsi, sur l'exemple illustré par la figure 56, la reconnaissance ne portera que sur les instances de *Présentateur*, *Volet* et

[64] Voir figure 34 page 80.
[65] Voir « Implémentation des relations temporelles » page 54.

Générique. Pour pouvoir reconstituer les instances de *Reportage* et de *Conclusion*, il est nécessaire de conserver la structure d'emboîtement des canevas ;

- Une liste des contraintes « no … between » définies pour le canevas.

4.3 Représentation des observations

Les observations sont schématiquement des objets datés placés dans une taxinomie de concepts. De même que pour les concepts d'événements élémentaires, plusieurs classes Java permettent de représenter les observations de manière uniforme dans plusieurs contextes, selon que les observations sont de simples références à des individus CLASSIC ou selon qu'elles sont intégralement représentées au sein de Clavis[66]. Les observations proviennent principalement soit d'un ensemble d'individus CLASSIC soit d'un ensemble de descriptions AEDI.

5 BackJava

Nous donnons ici un aperçu de BackJava, système de satisfaction de contraintes écrit en Java, sur lequel repose l'implémentation de Clavis. Nous commencerons par décrire le contexte général de la satisfaction de contraintes avant de donner quelques indications plus précises sur BackJava. Ce système est décrit en détail dans (Roy, 1999).

5.1 La satisfaction de contraintes

La satisfaction de contraintes est un paradigme qui permet d'exprimer et de résoudre des problèmes combinatoires difficiles, particulièrement dans les domaines finis. Un problème de satisfaction de contrainte (CSP, pour *Constraint Satisfaction Problem*) est représenté par un ensemble fini de *variables* qui représentent les inconnues du problème, ces variables prenant leurs valeurs dans un ensemble fini donné *a priori*, leur *domaine*. Les propriétés d'une solution du problème sont données par des relations entre les valeurs, nommées *contraintes.* Une *solution* du problème est une instanciation de chacune des variables avec une valeur choisie dans son domaine, toutes les contraintes étant respectées (Mackworth, 1977).

Tout CSP peut théoriquement être résolu par l'énumération explicite de toutes les valeurs de toutes les variables. La complexité de cette méthode la rend inexploitable autrement que pour des problèmes de petite taille. L'idée introduite par Mackworth pour améliorer la résolution est de prendre en compte la décomposition explicite du problème en contraintes. Il définit pour cela la notion de *cohérence* qui permet une approche combinatoire systématique à un niveau plus local que l'énumération explicite : au niveau d'une variable pour la *cohérence de nœuds*, au niveau d'une contrainte pour la *cohérence d'arc* et au niveau d'un chemin de contraintes pour la *cohérence de chemins.* L'idée essentielle de Mackworth est d'utiliser ces propriétés locales pour supprimer, *a priori* ou durant la résolution, des valeurs

[66] Voir « Communication CLASSIC/Clavis » page 101.

ne pouvant pas faire partie d'une solution, ce qui conduit à une réduction de la taille du problème et donc de la complexité de la résolution.

Plus précisément, la cohérence de nœuds consiste à supprimer *avant* le début de l'énumération les valeurs des domaines des variables qui violent une contrainte unaire. Ainsi, pour la contrainte $x > 0$, on peut supprimer toutes les valeurs négatives ou nulles du domaine de x. Une contrainte C est arc-cohérente si, et seulement si, il existe pour chaque valeur du domaine de chaque variable de C une valeur dans le domaine de chacune des autres variables de C de telle que C est satisfaite par ces valeurs. Le principe pour obtenir la cohérence d'arc est d'éliminer en cours de résolution les valeurs qui ne peuvent pas satisfaire une contrainte donnée, quelles que soient les valeurs des autres variables. Ainsi, si une contrainte requiert $x = y$, et si le domaine de x contient l'élément 1, alors que le domaine d'y ne le contient pas, on peut supprimer 1 du domaine de x. Enfin, la cohérence de chemin est une extension de la cohérence d'arc au traitement simultané de plusieurs contraintes. Ce sont les méthodes reposant sur la cohérence d'arc qui ont jusqu'à présent obtenu le plus de succès, la cohérence de nœuds, triviale, ne permettant pas d'approche intéressante et les algorithmes de cohérence de chemins étant coûteux et difficile à mettre en œuvre. De plus, la cohérence d'arc est une notion universelle car elle est applicable à toutes les contraintes indépendamment du problème à résoudre.

L'idée principale de Mackworth est qu'il est toujours d'obtenir la cohérence d'arc en supprimant des valeurs des domaines des variables sans perdre de solution. Ainsi, pour la contrainte $x = y$, avec comme domaine de x : {1, 2, 3} et comme domaine de y : {1, 2}, il suffit de supprimer 3 du domaine de x pour obtenir la cohérence d'arc. De plus, la cohérence d'arc définit la réduction maximale des domaines qu'on peut réaliser en prenant en compte les contraintes une par une sans perdre de solution. Macworth présente les algorithmes AC-1 et AC-3 – une amélioration de AC-1 – pour obenir la cohérence d'arc (Mackworth, 1977). Depuis, plusieurs algorithmes réalisant la cohérence d'arc ont été proposés. AC-4 (Mohr & Henderson, 1986), l'un des algorithmes les plus utilisés, offre une complexité théorique au pire des cas optimum, par l'emploi de structures de données sophistiquées. Citons également AC-6 (Bessière, 1994) et AC-7 (Bessière *et al.*, 1995).

Une des limitations de l'approche de Mackworth est de ne pas prendre en compte la nature des contraintes. L'idée de traiter différemment les contraintes selon leur nature, et ainsi d'améliorer l'efficacité du traitement local, est à la base du *filtrage*. Ainsi, une contrainte d'égalité $x = y$ ne se traite pas de la même manière qu'une contrainte d'inégalité $x \neq y$. Soit {1, 2, 3} le domaine initial de x et y. Pour la contrainte $x = y$, si, en cours de résolution, 3 a été supprimé du domaine de x, il peut également être supprimé du domaine de y. Pour la contrainte $x \neq y$, si, en cours de résolution 3 est la seule valeur restant dans le domaine de x (x est alors instanciée par 3), alors 3 peut être supprimé du domaine de y. Dans ces exemples, le filtrage de la contrainte réalise la cohérence d'arc. En pratique, il est des cas où il est intéressant de définir des méthodes de filtrage réalisant *moins* que la cohérence d'arc, *i.e.* qui suppriment moins de valeurs des domaines des variables (tout en laissant intact l'espace des solutions). Ainsi, pour les contraintes arithmétiques linéaires, le filtrage généralement utilisé consiste à ne considérer que les valeurs minimales et maximales du domaine de chaque variable. Ainsi, soient deux variables U et V dont le domaine est :

{1, …, 99.999, 100.001, …, 1.000.000}

et qui sont liées par la contrainte $U - 3.V = 0$. En considérant tour à tour les formules suivantes : $V = U/3$ et $U = 3.V$, on déduit dans un premier temps que $V \leq 1.000.000/3$ et donc que $V \leq 333.333$. dans un second temps on en déduit que $U \leq 999.999$. Cette méthode ne réalise pas la cohérence d'arc car la valeur 300.000 n'est pas cohérente. La cohérence d'arc serait trop coûteuse à calculer, cette méthode permet tout de même de considérablement réduire les domaines des variables.

Dans BackJava, le filtrage est implémenté à l'aide de *démons*. Techniquement, l'idée est de définir pour une même contrainte plusieurs méthodes de filtrage adaptées aux différentes manières de réduire les domaines. Par exemple, pour la contrainte d'égalité $x = y$, si une valeur est supprimée du domaine de x, le démon *valueRemoved* déclenche la suppression de la même valeur du domaine de y. Différentes classes de démons sont implémentées dans BackJava, pour la suppression d'une valeur, pour l'instanciation d'une variable, pour la modification des bornes d'un domaine ordonné, etc. On cherche donc à définir pour chaque type de contraintes un ensemble spécifique de méthodes de filtrage spécifique.

5.2 Un framework objet

BackJava est un framework au sens de (Johnson & Foote, 1988; Fayad & Schmidt, 1997), c'est-à-dire une application semi-complète, définie par des hiérarchies de classes collaborant entre elles selon un schéma prédéfini de façon à offrir une architecture réutilisable pour une famille d'applications proches. En BackJava, les principales notions de la satisfaction de contraintes sont représentées par de telles hiérarchies : 1) les domaines et les variables, 2) les contraintes, 3) les problèmes et les algorithmes de résolution et 4) les heuristiques de résolution.

La représentation des domaines, en particulier, doit permettre des modifications rapides : suppression d'une ou plusieurs valeurs, ainsi que modification des bornes pour les domaines ordonnés. La représentation des domaines quelconques repose dans BackJava sur la représentation des domaines entiers. Le choix est fait de représenter les domaines entiers par une liste d'intervalles. Ainsi, par exemple, l'ensemble {1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15} est représenté par la liste {[1-6], [8], [10-15]}. Cette représentation est plus compacte qu'une représentation explicite, excepté pour les domaines très dispersés. Elle permet également des opérations de modification efficaces, particulièrement pour les domaines denses. Les domaines arbitraires contenant des objets et non des entiers sont représentés par un domaine entier où chaque objet est apparié à un index. Par exemple, le domaine {A, B, C, D} sera représenté par le domaine entier {[1-4]}, avec les associations : A↔1, B↔2, C↔3 et D↔4. Cette représentation est plus efficace que la représentation explicite de la liste des objets. En effet, les domaines sont sauvegardés à chaque étape de la résolution et cette représentation permet de ne sauvegarder que les domaines entiers, la liste des objets et les associations demeurant inchangées. Les opérations sur les domaines arbitraires sont alors les mêmes que pour les domaines entiers et, pour supprimer l'élément A, le système supprime simplement l'index de A du domaine entier.

BackJava offre une hiérarchie de classes pour les variables relativement simple – variables entières, variables quelconques et variables ordonnées, principalement. La hiérarchie des classes de contraintes est en revanche plus complexe, le principe étant, comme nous l'avons vu, qu'à un type de contraintes correspond un ensemble spécifique de méthodes de filtrage.

Parmi ces classes de contraintes, on peut citer les contraintes permettant d'exprimer des combinaisons linaires, BJLinearCombinationCt, les contraintes permettant d'exprimer que toutes les variables doivent avoir des valeurs différentes, BJAllDiffCt ou les contraintes de cardinalité, BJCardinalityCt.

De nouvelles classes de contraintes peuvent être facilement définies à partir de contraintes existant en utilisant l'héritage et le polymorphisme (surcharge de méthodes). Nous verrons par la suite de nouvelles contraintes binaires définies par des classes héritant de la classe BJGenericBinaryCt qui représente une contrainte binaire entre deux variables quelconques.

Nous ne détaillerons pas les autres hiérarchies de classes – algorithmes de résolution et heuristiques de choix des contraintes et des variables – puisque nous avons utilisé les classes existantes. Mentionnons simplement que les démons de BackJava sont également réifiés, ce qui permet de définir de nouveaux types de démons. Nous reviendrons sur ce sujet plus loin.

6 Solveur de contraintes

Le processus général de reconnaissance – la boucle principale – est pris en charge par un *solveur* de contraintes, implémenté par la classe CLSolver, sous-classe de la classe générique BJSolver de BackJava. Les tâches réalisées par le solveur sont les suivantes :

- Effectuer la reconnaissance des sous-canevas non expansés. Cette étape consiste à instancier récursivement un nouveau solveur, puis à ajouter les instances reconnues du sous-canevas au graphe d'observations. Une seule phase de reconnaissance est bien entendu réalisée pour un sous-canevas donné, même si ce sous-canevas apparaît plusieurs fois dans une définition ;

- Créer une variable pour chaque composant du canevas à reconnaître, et calculer leur domaine. La création des variables est détaillée plus loin ;

- Créer les contraintes : contraintes temporelles et contraintes « no … between ». La création des contraintes est détaillée plus loin ;

- Exécuter le processus de reconnaissance. Cette étape consiste simplement à demander toutes les solutions du problème en faisant appel aux mécanismes de résolution de BackJava ;

- Interpréter les résultats. Il peut s'agir en particulier de calculer les références temporelles – les *dates* – des canevas reconnus en fonction des références temporelles de leurs composants.

7 Les variables

Chaque composant d'un canevas à reconnaître induit une variable du CSP. Nous distinguons les variables associées aux composants itérés, que nous appelons « variables itérées », des composants associés aux composants simples – non itérés –, que nous appelons « variables simples ». Ces deux types de variables sont implémentés par deux classes Java spécialisant des classes de variables de BackJava, CLVar et CLStarVar, ces deux classes implémentant la même interface CLVarInterface qui définit les méthodes communes à CLVar et CLStarVar. Deux de ces méthodes permettent simplement d'accéder au canevas à reconnaître (*getOriginTemplate*) et au graphe d'observations (*getObservationGraph*). Deux autres méthodes, *firstNode* et *lastNode*, permettent dans certains cas de traiter la valeur des variables comme des séquences, que la variable soit simple ou itérée, en considérant que la valeur d'une variable simple est une séquence de longueur un. Les méthodes *firstNode* et *lastNode* de la classe CLVar retournent évidemment la même valeur, qui est le nœud du graphe d'observation affecté à la variable.

```
public interface CLVarInterface
{
   public Template getOriginTemplate() ;
   public ObservationGraph getObservationGraph() ;
   public Node firstNode() ;
   public Node lastNode() ;
}
```

Le domaine de chaque variable simple est l'ensemble des nœuds du graphe d'observation. Ces nœuds sont des objets complexes ; la classe CLVar spécialise par conséquent la classe BJObjVar de BackJava, qui représente une variable dont les valeurs peuvent être n'importe quel objet Java. Le sommet du canevas auquel la variable est associée, ainsi que le canevas et le graphe d'observations, sont mémorisés par des variables d'instances.

```
public class CLVar extends CLObjVar implements CLVarInterface
{
   protected TemplateNode originNode ;
   protected Template originTemplate ;
   protected ObservationGraph observationGraph ;
}
```

Le domaine des variables instances de la classe BJObjVar est implémenté de manière indirecte, à la manière des variables entières, par un tableau d'intervalles indiquant les indices des objets restant dans le domaine. Le vecteur d'objets – les observations – peut donc être partagé par toutes les variables simples.

Le domaine d'une variable itérée associée à un concept *C**, est l'ensemble des séquences itérées du graphe d'observation dont les éléments sont des instances de *C*. La représentation interne de ces domaines et la manière dont ils sont calculés sont assez complexes et font l'objet de la prochaine section.

8 Domaine des variables itérées

Le domaine d'une variable itérée *v*, c'est-à-dire d'une variable représentant un composant associé à un canevas itéré T^*, est l'ensemble de toutes les séquences itérées d'événements associés à T dans le graphe d'observation. Ainsi, dans l'exemple illustré par la figure 57 existent-il 13 séquences itérées différentes d'événements associés au concept X: $\{X_1\}$, $\{X_2\}$, $\{X_3\}$, $\{X_4\}$, $\{X_5\}$, $\{X_1\,X_2\}$, $\{X_2\,X_3\}$, $\{X_2X_4\}$, $\{X_4\,X_5\}$, $\{X_1\,X_2\,X_3\}$, $\{X_1\,X_2\,X_4\}$, $\{X_2\,X_4\,X_5\}$, $\{X_1\,X_2\,X_4\,X_5\}$.

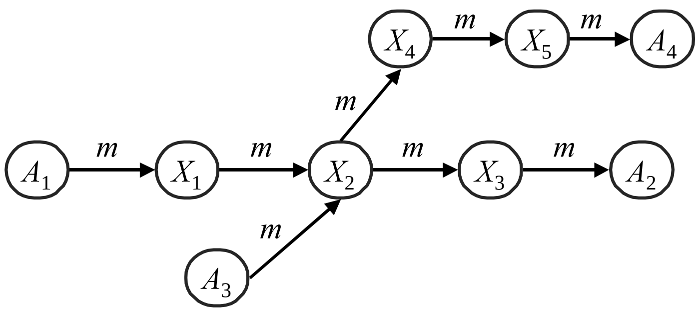


figure 57 : représentation des séquences

Le nombre de séquences itérées peut croître très rapidement avec le nombre des observations, selon la manière dont les observations sont disposées temporellement. Le pire des cas, heureusement très improbable dans les problèmes pratiques que nous avons à traiter, est un graphe d'observations ayant N^2 éléments disposés comme l'illustre la figure 58, c'est à dire un graphe dont les N^2 éléments sont disposés en N séquences parallèles, chaque élément étant en relation *equals* avec N-1 autres éléments, et chaque élément étant associé au même concept X, ou à un concept subsumé par X. Dans ce cas le nombre de séquences de longueur k est de $(N-k+1)\times N^k$, ce qui donne par exemple N séquences de longueur 1, et N^N séquences de longueur N.

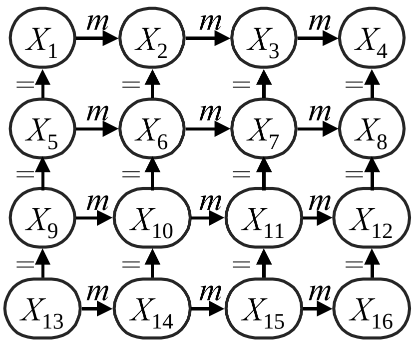


figure 58 : pire des cas pour 16 composants

Ainsi, pour une séquence itérée de N observations associées au concept X (figure 59), il y a N séquences de longueur 1, (N-1) séquences de longueur 2, …, $N-p+1$ séquences de longueur p, …, et une seule séquence de longueur N. Le nombre total des séquences possibles est donc :

$$\sum_{i=1}^{N}(N-i+1)=\frac{N(N+1)}{2}$$

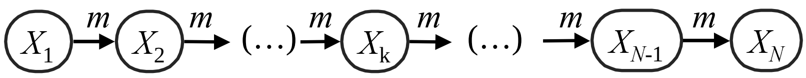


figure 59 : séquence de *N* composants

On remarque sur cet exemple que toutes les séquences itérées de X sont des *sous-séquences* de la séquence $\{X_1,\ldots, X_N\}$. On appelle *séquence maximale* associée au concept X toute séquence itérée d'instances de X dans un graphe d'observations qui n'est pas sous-séquence d'une autre séquence itérée. Pour ne pas avoir besoin de représenter explicitement toutes les séquences d'un graphe d'observation, nous calculons dans un premier temps l'ensemble des séquences maximales, et nous représentons la totalité des séquences en fonction de ces séquences maximales. Dans l'exemple de la figure 57, il y a deux séquences maximales : $\{X_1\ X_2\ X_3\}$ et $\{X_1\ X_2\ X_4\ X_5\}$. Dans certains cas, le nombre de séquences maximales peut également être très important. Dans la configuration décrite ci-dessus et illustrée par la figure 58 (N séquences itérées parallèles de N éléments chacune), le nombre de séquences maximales est le nombre de séquences de taille N, soit N^N.

8.1 Indexation des séquences sur les séquences maximales

L'ensemble des séquences itérées associées à X est l'ensemble de toutes les sous-séquences des séquences maximales associées à X. Une séquence quelconque peut être identifiée par la séquence maximale dont elle est sous-séquence, par son indice de début dans la séquence maximale, et par sa longueur (figure 60).

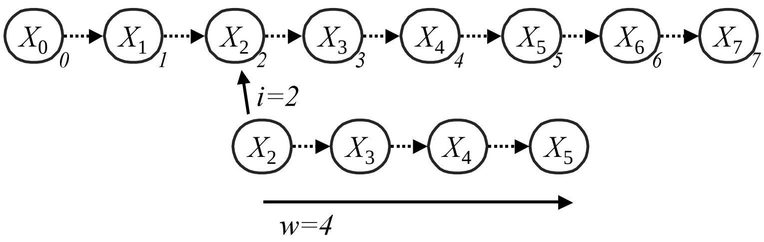


figure 60 : sous-séquence d'une séquence maximale

On cherche à représenter le domaine des variables itérées de la manière la plus simple et la plus compacte possible en associant à chaque séquence itérée un nombre entier la représentant. Pour cela, on numérote les sous-séquences d'une séquence maximale donnée en les triant par ordre de longueur croissante, puis pour une longueur donnée, par l'indice de leur premier élément dans la séquence maximale. Comme il a été vu plus haut, pour une séquence maximale de longueur N, il y a N sous-séquences de longueur 1, N-1 sous-séquences de longueur 2, $N-w+1$ sous-séquences de longueur w, et une sous-séquence de longueur N. En numérotant à partir de 0 les sous-séquences ainsi triées, on obtient les index suivants :

- de 0 à N-1 : sous-séquences de longueur 1 ;
- de N à 2N-2 : sous-séquences de longueur 2 ;
- (…)
- $\frac{N(N+1)}{2} - 1$: sous-séquence de longueur N.

Soit la fonction d'indexation $F_I(w, i, N)$ qui donne l'index de la sous-séquence de longueur w et dont le premier élément est le i$^{\text{ème}}$ élément d'une séquence maximale de longueur N dont les composants sont numérotés de 0 à N-1. La première sous-séquence de longueur w, pour $w \geq 2$, a pour index $F_I(w,0,N)$, qui est également le nombre de sous-séquences de longueur inférieure à w, soit :

$$F_I(w,0,N) = \sum_{j=1}^{w-1}(N-j+1) = N(w-1) - \frac{(w-1)(w-2)}{2} \qquad (7)$$

Par la suite, les sous-séquences de même longueur étant triées selon l'index de leur premier élément dans la séquence maximale, on obtient :

$$F_I(w,i,N) = F_I(w,0,N) + i \qquad (8)$$

Lorsqu'il y a plusieurs séquences maximales, comme c'est le cas sur la figure 57, les séquences maximales sont elles-mêmes numérotées, et les index correspondant aux sous-séquences d'une séquence maximale donnée sont décalés du nombre de sous-séquences correspondant aux séquences maximales précédentes, et le décalage à apporter pour chacune des séquences maximales est mémorisé dans un tableau *offset*.

8.2 Calcul des séquences maximales

Nous cherchons à calculer dans un graphe d'observations l'ensemble des séquences maximales dont les éléments sont associés à un concept X. La méthode de calcul des séquences maximales que nous présentons ici s'applique au cas général d'observations qui ne sont pas précisément datées, c'est-à-dire à un ensemble d'observations liées par des relations temporelles quelconques. La difficulté réside dans le fait qu'on ne connaît pas *a priori* les débuts de séquences.

Le principe de l'algorithme est simple : il consiste à chercher les débuts de séquences, puis à « remonter » les séquences jusqu'à leur fin. L'algorithme doit tenir compte du fait que chaque observation peut appartenir à plusieurs séquences maximales.

On calcule tout d'abord l'ensemble R des couples (e_i, e_j) d'observations tels que les concepts associés à e_i et e_j soient subsumés par X, et tels que la contrainte temporelle entre e_i et e_j soit *meets*. Ensuite, le principe de l'algorithme consiste à choisir un couple (e_i, e_j) dans R, et à

« remonter » toutes les séquences contenant e_i et e_j en cherchant dans R les couples (e_k e_i). Le même principe est appliqué « dans l'autre sens » une fois trouvé un début de séquence. La version la plus simple (récursive) de l'algorithme est donnée en pseudo-code ci-dessous. Dans la pratique, nous avons « dérécursivé » l'algorithme pour des raisons d'efficacité, en utilisant deux piles explicites.

```
Procédure SéquencesMax
   Tant qu'il existe un couple (ei, ej) non marqué dans R
      Arrière(ei, ej)

Procédure Arrière(ei, ej)
   Booléen trouvé = faux
   Pour tous les (ek, ei) de R /* ek meets ei */
      trouvé = vrai
      Arrière(ek, ei)

   Si non trouvé // (ei, ej) est un début de séquence maximale
Marquer (ei, ej)
      s = nouvelle séquence commençant par (ei, ej)
      Avant(s)

Procédure Avant(s)
   e = dernier élément de s
   Booléen trouvé = faux
   Pour tous (e, ei) de R /* ei meets e */
      trouvé = vrai
      s' = copie de s
      ajouter ei à la fin de s'
      Avant(s')
   Si non trouvé
      s' est une séquence maximale
```

figure 61 : calcul des séquences maximales

Cet algorithme peut évidemment être optimisé dans le cas d'observations précisément datées, ce qui ne s'est pas avéré nécessaire en pratique.

8.3 Variables entières pour les canevas itérés

Les domaines des variables associées à un canevas itéré X^* sont donc représentés d'une part par l'ensemble des séquences maximales associées à X, et d'autre part par l'ensemble des index représentant toutes les sous-séquences de ces séquences maximales. L'ensemble des valeurs possibles est donc parfaitement déterminé par un ensemble de valeurs entières. Les variables à valeurs entières sont représentées de manière efficace dans BackJava, les domaines entiers étant représentés comme un ensemble d'intervalles, ce qui est particulièrement efficace pour les domaines de grande taille et assez « denses » (Roy, 1999)[67].

[67] Voir « Un framework objet » page 107.

De même que pour les variables simples instances de la classe CLVar[68], le sommet du canevas auquel est associée la variable, le canevas d'origine et le graphe d'observations sont mémorisés par des variables d'instances. Les séquences maximales ainsi qu'un tableau d'entiers *offset* indiquant les décalages utilisés par la fonction d'indexation des sous-séquences, sont également représentés par des variables d'instance. La définition de la classe CLVar est ainsi :

```
public class CLVar extends CLObjVar implements CLVarInterface
{
   protected TemplateNode originNode ;
   protected Template originTemplate ;
   protected ObservationGraph observationGraph ;
   protected Sequence[] maximalSequences ;
   protected int[] offsets ;
}
```

8.4 Reconstruction des séquences

Nous décrivons dans cette section la manière dont les séquences sont reconstruites à partir de leur index dans le domaine des variables itérées. On définit la fonction $f_g(s,i,w)$ qui donne l'index de la séquence de taille w commençant à l'indice i dans la $s^{\text{ème}}$ séquence maximale. Si T_s est la taille de la $s^{\text{ème}}$ séquence maximale, et *offset* le tableau mémorisant pour chaque séquence maximale le décalage à apporter, on a :

$$f_g(s,i,w) = F_I(w,i,T_s) + offset[s] \tag{9}$$

Les séquences n'étant représentées que par leur index, on veut pouvoir reconstruire explicitement une séquence à partir de son index. Etant donné un index k, on cherche s, i et w tels que $f_g(s,i,w)=k$. Pour trouver s, on se contente de parcourir le tableau *offset*[69]. On cherche alors i et w tels que $F_I(w,i,T_s)=k'$, avec $k'=k-offset[s]$. On cherche tout d'abord à calculer w. Pour cela, on sait que la fonction $F_I(w,0,N)$ qui donne l'index de la « première » séquence de longueur w dans une séquence maximale de longueur N est un polynôme du second degré en w, strictement croissant pour $w \le N$:

$$F_I(w,0,N) = N(w-1) - \frac{(w-1)(w-2)}{2} = -w^2 + (2N-3)w - 2(N+1) \tag{10}$$

[68] Voir « Les variables » page 109.

[69] Recherche séquentielle ou recherche dichotomique suivant la taille du tableau *decal*.

La figure 62 illustre F_I pour N=8.

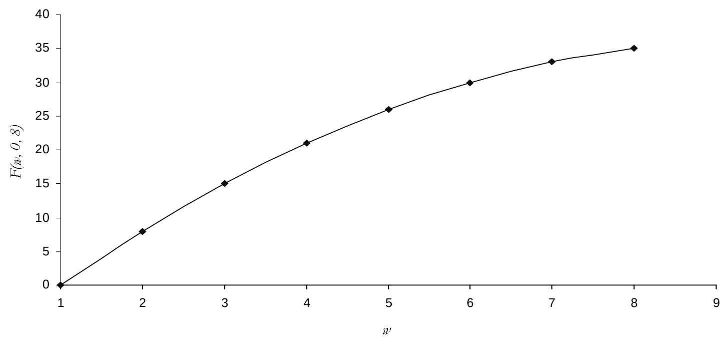


figure 62 : $F_I(w,0,8)$, index de la première séquence de longueur *w*

Par conséquent, pour obtenir la longueur w de la séquence d'index k' dans une séquence maximale de longueur N, il suffit de prendre la partie entière de la première racine de l'équation $F_I(w,0,N) = k'$, soit :

$$w = E\left(\frac{2N-3-\sqrt{(2N-3)^2-8(N+k'+1)}}{2}\right) \quad (11)$$

Une fois calculé w, on calcule l'indice i du premier élément de la séquence par :

$$i = k' - F_I(w,0,N) \quad (12)$$

Connaissant sa longueur w et l'indice i de son premier élément dans la séquence maximale d'indice s, il est facile de représenter explicitement la séquence d'index k.

8.5 Suppression des doublons

Le graphe illustré par la figure 63 comporte six séquences maximales associées au concept X : $\{X_1\ X_2\ X_3\ X_4\}$, $\{X_1\ X_2\ X_3\ X_8\}$, $\{X_1\ X_2\ X_6\ X_7\}$, $\{X_5\ X_2\ X_3\ X_4\}$, $\{X_5\ X_2\ X_3\ X_8\}$ et $\{X_5\ X_2\ X_6\ X_7\}$. Certaines séquences sont sous-séquences de plusieurs séquences maximales ; la séquence $\{X_2\ X_3\}$ est ainsi sous-séquence de quatre séquences maximales différentes.

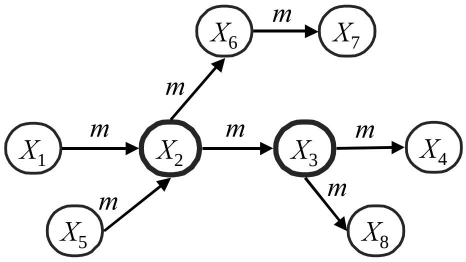


figure 63 : sous-séquences communes

Ainsi, si l'on se contente de représenter le domaine d'une variable itérée par l'intervalle [0 – Indice$_{max}$], il se peut que certaines séquences apparaissent plusieurs fois, ce qui peut conduire le solveur à trouver plusieurs solutions équivalentes. Pour résoudre ce problème, on supprime du domaine de la variable les séquences définies plusieurs fois. Le principe de l'algorithme de suppression est simple : on compare deux à deux les séquences maximales en déplaçant deux pointeurs k et l sur chacune des deux séquences maximales s_i et s_j pour chercher les séquences communes les plus longues (sous-séquences communes maximales). L'algorithme prend soin de déplacer les pointeurs le plus rapidement possible : lorsqu'une séquence commune s_c a été trouvée, la recherche peut se poursuivre à partir de l'élément suivant le dernier élément de s_c. Toutes les sous-séquences de s_c sont finalement supprimées une fois du domaine de la variable. Le domaine étant une liste d'entiers représentant les index des séquences triées par taille, il est plus rapide de supprimer les sous-séquences de s_c par ordre de taille : toutes les sous-séquences de longueur 1, puis toutes les sous-séquences de longueur 2, etc. L'instruction de suppression fait appel à la fonction f_g défini par l'équation (9)[70]. L'algorithme est donné ci-dessous par la figure 64.

```
Pour i =0 ; i < Nbre de séquences maximales – 1 ; i++
   Si = ième séquence maximale
   Pour j = i+1 < Nbre de séquences maximales ; j++
       Sj = jème séquence maximale
   garder_d = 0
   k = 0
   Tant que k < taille(Si)
       N = Si[k]
       d = indice de N dans Sj, cherché à partir de garder_d
       k++
       Si N a été trouvé dans Sj
           l = d + 1
       Tant que k < taille(Si) Et l < taille(Sj) Et si[k]=Sj[k]
           k++
           l++
       garder_d = l
       Pour w = 1 ; w <= l-d ; w++
           supprimer l'intervalle : [fg(j,d,w) ; fg(j,l-w,w)]
```

figure 64 : algorithme de suppression des sous-séquences communes

Ainsi, pour les séquences maximales s_1 à s_6 du graphe illustré par la figure 63 :

[70] Voir page 114.

$s_1 \quad = \{X_1\ X_2\ X_3\ X_4\}$
$s_2 \quad = \{X_1\ X_2\ X_3\ X_8\}$
$s_3 \quad = \{X_1\ X_2\ X_6\ X_7\}$
$s_4 \quad = \{X_5\ X_2\ X_3\ X_4\}$
$s_5 \quad = \{X_5\ X_2\ X_3\ X_8\}$
$s_6 \quad = \{X_5\ X_2\ X_6\ X_7\}$

les sous-séquences communes maximales détectées sont, par ordre de détection :

$\{X_1\ X_2\ X_3\}$ commune à s_1 et s_2
$\{X_1\ X_2\}$ commune à s_1 et s_3
$\{X_2\ X_3\ X_4\}$ commune à s_1 et s_4
$\{X_2\ X_3\}$ commune à s_1 et s_5
$\{X_2\}$ commune à s_1 et s_6
$\{X_1\ X_2\}$ commune à s_2 et s_3
$\{X_2\ X_3\}$ commune à s_2 et s_4
$\{X_2\ X_3\ X_8\}$ commune à s_2 et s_5
$\{X_2\}$ commune à s_2 et s_6
$\{X_2\}$ commune à s_3 et s_4
$\{X_2\}$ commune à s_3 et s_5
$\{X_2\ X_6\ X_7\}$ commune à s_3 et s_6
$\{X_5\ X_2\ X_3\}$ commune à s_4 et s_5
$\{X_5\ X_2\}$ commune à s_4 et s_6
$\{X_5\ X_2\}$ commune à s_5 et s_6

9 Les contraintes

Un canevas décrit une classe de séquences vidéo par certaines contraintes que doivent vérifier les observations. Les contraintes de subsomption, qui imposent que chaque composant associé à un concept élémentaire *C* ait pour valeur une observation instance de *C*, ainsi que les contraintes temporelles entre variables simples, sont similaires aux contraintes déjà décrites pour l'implémentation du système T-Rex dans le framework BackJava[71]. Ces contraintes seront décrites brièvement. Le filtrage des contraintes temporelles impliquant une ou deux variables itérées est plus complexe, et sera étudié plus en détail. Enfin, nous décrirons le filtrage des contraintes de type « no … between ».

9.1 Contraintes de subsomption

La valeur d'une variable itérée V_C associée à un concept *C* doit être une instance du concept *C*. On peut donc définir le domaine de la variable V_C comme l'ensemble des observations qui sont instances de *C*. Une autre manière de procéder consiste 1) à définir le domaine de V_C comme étant l'ensemble de toutes les observations et 2) à définir une contrainte unaire portant sur V_C imposant que la valeur de V_C soit une instance de *C*. La représentation des domaines en BackJava plaide pour cette seconde solution. En effet, les

[71] Voir l'implémentation de T-Rex page 69.

domaines d'objets quelconques sont représentés d'une part par un vecteur contenant l'ensemble des objets possibles pour une variable, et d'autre part par un ensemble d'indices représentant chacun de ces objets, ces index étant représentés par un ensemble d'intervalles entiers. C'est seulement sur ces index que portent les opérations de mise à jour des domaines – réduction et restitution. Le vecteur d'objets peut donc être partagé par plusieurs variables. Ainsi, toutes les variables associées aux composants non itérés d'un canevas partagent-elles le même domaine constitué de l'ensemble de toutes les observations. La déclaration d'une contrainte de subsomption pour chacune des variables V_C entraîne la suppression du domaine de V_C de toutes les valeurs qui ne sont pas instances de C, ou plus exactement de tous les indices des valeurs qui ne sont pas instances de C. Cette opération de complexité linéaire est réalisée une fois pour toutes au début de la phase de résolution et n'entraîne donc qu'un surcoût modéré.

La définition de ces contraintes se fait simplement par la définition de la classe CLUnarySubsumeCt, sous-classe de la classe BJUnaryCt, dont seule la méthode *isSatisfied* est redéfinie. Cette méthode se contente de vérifier que la valeur passée en argument est une instance du concept associé à la variable. Le mécanisme de filtrage des contraintes de BackJava se charge d'appeler cette méthode pour toutes les valeurs du domaine de chacune des variables, et supprime du domaine les valeurs pour lesquelles le test n'est pas vérifié.

9.2 Contraintes temporelles entre deux variables simples

Le filtrage des contraintes temporelles portant sur deux variables simples se fait de la même manière que dans l'implémentation du système T-Rex que nous avons présentée au chapitre précédent[72]. Du point de vue de l'implémentation, ce filtrage est réalisé par une sous-classe de la classe BJGenericBinaryCt, CLTemporalCt, dont la méthode *isSatisfied* est redéfinie. Cette méthode comporte deux paramètres qui sont les valeurs des deux variables lorsque toutes deux sont instanciées, et retourne *vrai* si la contrainte temporelle imposée entre les variables est respectée.

L'implémentation de ces contraintes binaires est réalisée d'une manière similaire à celle des contraintes de subsomption, en déclarant simplement – par le biais de la méthode *isSatisfied* – la condition que doivent respecter les valeurs des variables. Le mécanisme de résolution utilisé par BackJava est ici cependant différent, et s'appuie sur le principe des démons[73]. Ainsi, par exemple, lorsqu'une des variables de la contrainte est instanciée, le démon *valueChanged* déclenche la méthode de filtrage de la contrainte qui réalise la cohérence d'arc en supprimant du domaine de l'autre variable toutes les valeurs pour lesquelles la méthode *isSatisfied* retourne *faux*.

9.3 Contraintes temporelles impliquant une variable itérée

Le principe de base du filtrage des contraintes temporelles $Ct_R(V^*, V)$, où V^* est une variable itérée, et V une variable quelconque, itérée ou non, est le suivant. Quand la

[72] *ib.*

[73] Pour les démons, voir « Un framework objet » page 107.

variable *V* est instanciée, un démon déclenche la méthode *valueChanged*. La valeur de *V* est un événement élémentaire si *V* est une variable simple, et une séquence itérée si *V* est une variable itérée. Dans les deux cas, la valeur de *V* est associée à une période temporelle unique, et les méthodes *firstNode* et *lastNode* décrites ci-dessus[74] permettent de manipuler cette valeur de façon homogène par le biais de l'interface CLVarInterface. Soit *R* la relation temporelle imposée dans le canevas entre les composants correspondant aux variables *V* et *V**, c'est-à-dire la relation temporelle devant être respectée par les valeurs de *V* et *V**. Si *R* est une des relations *equals*, *before*, *after*, *meets* ou *is-met*, un filtrage spécifique est déclenché. Ces relations correspondent à des cas identifiés comme fréquents, et pour lesquels des méthodes de filtrage rapides ont été développées. Sinon, une méthode de filtrage générale est déclenchée, qui consiste pour toutes les relations temporelles élémentaires *r* qui ne sont pas contenues dans *R*, à supprimer du domaine de *V** toutes les séquences *s* telles que *s* est dans la relation *r* avec la valeur de *V*. Ce principe de filtrage est résumé ci-dessous :

```
valueChanged(CLStarVar v1, CLVarInterface v2, short R)
   throws BJWipedOutException
{

   switch(R)
   {
      case equals :
         v1.doEqualsAllen(v2.firstNode(), v2.firstNode()) ;
         return ;
      case before :
         v1.doBeforeAllen(v2.firstNode()) ;
         return ;
      //etc. Idem avec after, meets et is-met
   }

   if (!Allen.compatible(R, equals))
   {
      v1.notEqualsAllen(v2.firstNode(), v2.lastNode());
   }
   if (!Allen.compatible(R, before))
   {
      v1.notBeforeAllen(v2.firstNode()) ;
   }
   //etc. Idem avec les 11 autres relations élémentaires
}
```

9.3.1 Procédures de suppression des séquences

Deux procédures primitives de suppression de séquences du domaine d'une variable itérée ont été définies.

La première méthode supprime la séquence de longueur *w* dont le premier élément est le i[ème] élément de la s[ème] séquence maximale. Cette méthode se contente de supprimer du domaine l'entier correspondant à l'index de cette séquence, soit $f_g(s,i,w)$[75].

[74] Voir « Les variables » page 109.

[75] Voir « Reconstruction des séquences » page 114 pour la définition de la fonction f_g.

```
removeOneSequence(s, w, i)
   index = fg(s, b, w)
   supprimer index
```

La seconde méthode supprime l'ensemble des séquences de longueur *w* dont le premier élément varie entre le b^{ème} (b pour *begin*) et le e^{ème} (e pour *end*) élément de la s^{ème} séquence maximale. Les index de ces séquences sont consécutifs dans le domaine et peuvent donc être supprimés par un unique appel à la fonction de suppression d'un intervalle.

```
removeSequences(s, w, b, e)
   index1 = fg(s, w, b)
   index2 = fg(s, w, e)
   supprimer [index1 ; index2]
```

Ainsi, si la s^{ème} séquence maximale est la séquence illustrée par la figure 65, l'appel à la fonction `removeSequences` avec w=4, b=2 et e=4 supprimera les séquences $\{X_2X_3X_4X_5\}$, $\{X_3X_4X_5X_6\}$ et $\{X_4X_5X_6X_7\}$.

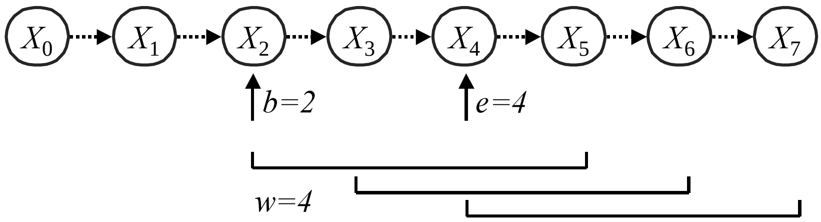
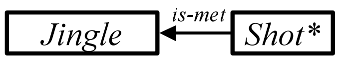


figure 65 : suppression d'un ensemble de séquences

9.3.2 Méthodes de filtrage spécifiques

Considérons l'exemple illustré par la figure 66, où l'on cherche un jingle précédant immédiatement une séquence de plan. Si, au cours de la résolution, la variable associée au composant du canevas représentant le jingle se voit affecter l'observation *jingle-1*, il est possible de supprimer du domaine de la variable représentant la séquence de plans toutes les séquences qui ne commencent pas par le plan numéro 6.

Canevas :



Observations :

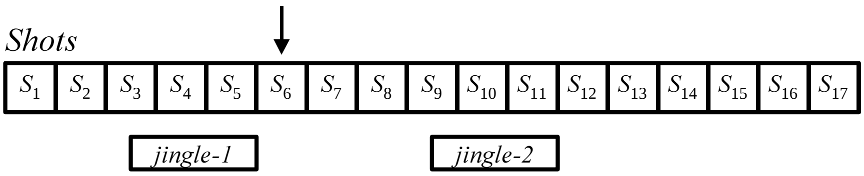


figure 66 : filtrage pour la contrainte *is-met*

La méthode de filtrage de la relation *is-met* correspondant à cet exemple est donnée ci-dessous. Les autres méthodes spécifiques de filtrage sont construites sur le même principe. Il se peut, contrairement à l'exemple donné ci-dessus (voir figure 66) que plusieurs observations différentes puissent être dans la relation *meets* avec la valeur de la variable *V*. Si ce n'est pas le cas, c'est-à-dire si une observation constitue le seul début de séquence possible, la méthode de filtrage signale ce cas de figure en armant un démon spécifique, dont il sera question plus loin.

```
doIsMetAllen(Node N)
{
   nSeqMax = nombre de séquences maximales
   firstNodeTab = tableau de Node, de taille nSeqMax
   Node firstNode = null
   boolean armDemon = true

   Pour i = 0 ; i<nSeqMax ; i++
      Si = ième séquence maximale
      miN = getFirstNodeSuchThat(Si, N, obsGraph, is-met)
      //miN = nœud du graphe d'observation tq : on is-met N
      si miN a été trouvé
         firstNodeTab[i] = miN
         Si firstNode = null
            firstNode = miN
         Sinon, si firstNode != miN //plusieurs débuts
            armDemon = false

   Si firstNode = null
      Vider le domaine, il n'y a pas de solution

   Si armDemon
      doStartWith(firstNode)
   Sinon
      pour i = 0 ; i<i<nSeqMax ; i++
         doStartWithInSequence(firstNodeTab[i], i)
}
```

Cette méthode de filtrage fait appel à plusieurs méthodes. La méthode *getFirstNodeSuchThat*(S, N, O, R) cherche dans la chaîne maximale S, le premier nœud en partant du début de S qui soit dans la relation temporelle R avec une observation N du graphe d'observations O. Des méthodes similaires ont été développées pour le premier nœud en partant de la fin, le dernier nœud en partant du début, etc., ainsi que des méthodes similaires cherchant le premier nœud qui soit dans une relation temporelle *compatible* avec l'observation N. La méthode *doStartWithInSequence*(i, N) supprime du domaine de la variable itérée toutes les sous-séquences de la $i^{ème}$ séquence maximale qui ne commencent pas par l'observation N, en supprimant les sous-chaînes par ordre de taille, de manière à effectuer le moins possible d'appels à la fonction de suppression d'un intervalle dans la liste des intervalles représentant le domaine de la variable. La méthode *doStartWith*(N) supprime du domaine toutes les séquences ne *commençant* par l'observation N en appelant *doStartWithInSequence* pour toutes les séquences maximales, et arme un démon qui indique que toutes les valeurs possibles de la variable sont des séquences qui commencent par le même nœud. Les méthodes *doFinishesWithInSequence* et *doFinishes* permettent d'imposer à une variable itérée que ses valeurs se terminent par la même observation. La méthode *doStartWithInSequence* est donné ci-dessous :

```
doStartWithInSequence(MaximalSequence MS, Node N)
   index = index de N dans MS
   si N n'a pas été trouvé
      Supprimer toutes les sous-séquences de MS
   Sinon
      NS = length(MS)
   for w = 1 to NS
      ind1 = 0
      ind2 = min(i-1, NS-w)
      ind3 = i+1
      ind4 = NS-w
      if ind2 ≥ ind1
         removeSequences(s, w, ind1, ind2)
      if ind4 ≥ ind3
            removeSequences(s, w, ind3, ind4)
```

9.3.3 Méthodes de filtrage générales

Lorsque la relation temporelle *R* d'une contrainte $Ct_R(V^*, V)$ est une relation temporelle quelconque, on applique une méthode de filtrage générale. Cette méthode consiste pour toutes les relations temporelles élémentaires *r* non contenues dans *R* à supprimer du domaine de V^* toutes les séquences qui sont dans la relation temporelle *r* avec la valeur de *V*.

L'exemple de la figure 67 illustre un canevas décrivant une séquence de plans apparaissant *pendant* un segment de musique, « pendant » étant pris au sens large et se traduisant par la disjonction de relations temporelles {*starts* ∨ *finishes* ∨ *equals* ∨ *during*}. *V* est la variable représentant le composant du canevas associé à un segment de musique et V^* représente le composant associé à la séquence itérée de plans. Lorsque *V* est instanciée, par exemple avec l'événement *music-1*, le filtrage de la variable V^* se fait en supprimant de son domaine toutes les séquences qui ne sont pas dans la relation *before* avec *music-1*, puis toutes les séquences qui ne sont pas dans la relation *after*, et ainsi de suite avec les relations *before*, *after*, *meets*, *is-met*, *overlaps*, *is-overlapped*, *is-started*, *is-finished* et *is-during*. Ainsi, la méthode *notIsMetAllen* supprime du domaine de V^* toutes les séquences commençant par le plan S_7 lorsque *V* est instanciée avec *music-1*.

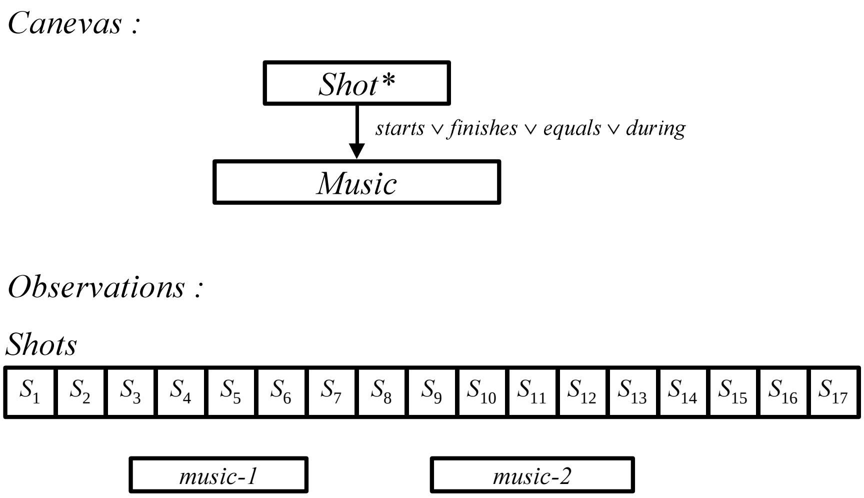


figure 67 : filtrage d'une contrainte temporelle quelconque

Le principe de la méthode *notIsMetAllen* est simple, dans le cas où les observations sont associées à une période temporelle unique, c'est-à-dire lorsque les relations temporelles entre les observations sont non disjonctives. Il consiste à chercher pour chaque séquence maximale du domaine de V^* l'élément qui est dans la relation *is-met* avec la valeur de V, et à supprimer du domaine de V^* tous les éléments commençant par cet élément. Dans l'algorithme donné ci-dessous, le paramètre N représente la valeur de la variable V.

```
notIsMetAllen(Node N)
{
   Pour i = 0 ; i<nSeqMax ; i++
       Si = ième séquence maximale
       index = entier tel que : Si[index] is-met N
       Si index est trouvé
       Pour w = 1 ; w < taille(Si)-index+1 ; w++
           removeOneSequence(i, w, index)
}
```

Dans le cas où les observations ne sont pas précisément datées – segmentation de la bande son en locuteurs, par exemple –, c'est-à-dire lorsque les relations temporelles entre les observations peuvent être des disjonctions des relations temporelles élémentaires, l'algorithme se complexifie, puisque plusieurs éléments d'une même séquence maximale peuvent être dans la relation *is-met* avec la valeur de la variable V. L'algorithme devient alors :

```
notIsMetAllen(Node N)
{
   Pour i = 0 ; i<nSeqMax ; i++
       Si = ième séquence maximale
       index = premier indice de Si tel que la relation
           temporelle entre Si[index] et N est compatible
           avec is-met
       relation = is-met
       Si index est trouvé
           nodeInSi = N
           Tant que index < taille(Si)
               Et relation compatible avec is-met
               Pour w = 1 ; w < taille(Si)-index+1 ; w++
```

```
          removeOneSeq(i, w, index)
      index++
      Si index < taille(Si)
          nodeInSi = Si[index]
          relation = relation entre nodeInSi et N
```

Des méthodes similaires ont été implémentées pour les douze autres relations temporelles élémentaires. Chacune de ces méthodes a été conçue de façon à exploiter le plus possible la représentation interne du domaine des variables itérées sous la forme d'une liste d'intervalles. Ainsi, lorsque c'est possible, les suppressions se font par ordre de taille des séquences, de manière à supprimer plusieurs séquences par un seul appel à la fonction de suppression d'un intervalle.

Ainsi, sur l'exemple illustré par la figure 67, une fois réalisé le filtrage de la variable V^* associée à la séquence de plans et si la variable V associée au segment musical a été instanciée avec le segment *music-1*, les valeurs possibles pour V^* sont les séquences de plan $\{S_4\}$, $\{S_5\}$, $\{S_6\}$, $\{S_4S_5\}$, $\{S_5S_6\}$ et $\{S_4S_5S_6\}$. Si V avait été instanciée avec le segment *music-2*, les séquences possibles seraient $\{S_{10}\}$, $\{S_{11}\}$, $\{S_{12}\}$, $\{S_{10}S_{11}\}$, $\{S_{11}S_{12}\}$ et $\{S_{10}S_{11}S_{12}\}$.

9.3.4 Filtrage des contraintes impliquant une seule variable itérée

Nous avons vu jusqu'ici la manière dont est réalisé le filtrage des domaines des variables pour les contraintes temporelles impliquant deux variables simples, et pour les contraintes impliquant au moins une variable itérée lorsque l'autre des variables est instanciée. Afin que le filtrage soit complet, il reste à filtrer le domaine des variables itérées pour les contraintes impliquant une variable simple et une variable itérée, lorsque c'est la variable itérée qui est instanciée. Soit $CT_R(V, V^*)$ une contrainte temporelle entre une variable simple V et une variable itérée V^*, R étant la relation temporelle de la contrainte. Le principe du filtrage consiste, lorsque V^* est instanciée avec une séquence itérée s, à supprimer du domaine de V tous les éléments e ne respectant pas la contrainte, c'est-à-dire les événements qui sont tels que la relation temporelle $R_{e,s}$ observée entre e et s n'est pas incluse – n'est pas subsumée par – R. Pour cela, il faut calculer $R_{e,s}$. Ce calcul est fait en examinant la relation existant entre l'événement e et les bornes de la séquence s. Ainsi, sur l'exemple illustré par la figure 68, les relations temporelles entre l'événement e et les bornes de s permettent de déduire que la relation temporelle l'événement e et la séquence s est nécessairement $\{$*equals* $\vee$ *finishes*$\}$.

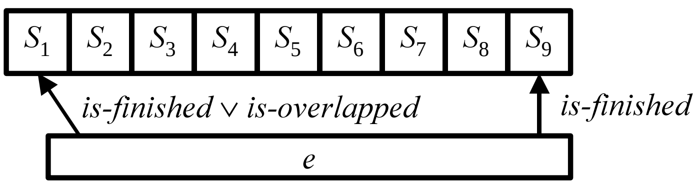


figure 68 : relation temporelle entre une séquence itérée et un événement simple

La figure 69 donne, pour chaque relation temporelle élémentaire observée entre un événement e et le premier élément d'une séquence itérée s (2ème colonne) ou entre e et le dernier élément de s (3ème colonne), la disjonction de relations temporelles élémentaires constituant la relation temporelle possible entre l'événement e et la séquence s entière. Cette méthode est à rapprocher de la méthode qui permettait de calculer les relations entre les bornes d'une séquence *s* et un événement *e* en fonction de la relation entre la séquence *s* entière et l'événement *e*[76]. La table de correspondance illustrée par la figure s'applique aussi bien aux séquences de longueur un qu'aux séquences de longueur supérieure à un.

Relation	**Avec le premier élément de *s***	**Avec le dernier élément de *s***
equals	1 2 3 4 $e \vee s$	1 2 3 4 $e \vee f$
before	1 2 3 4 b	1 2 3 4 $b \vee m \vee o \vee s \vee d$
after	1 2 3 4 $d \vee f \vee oi \vee mi \vee a$	1 2 3 4 a
meets	1 2 3 4 m	1 2 3 4 $m \vee o \vee s \vee d$
is-met	1 2 3 4 $d \vee mi \vee f \vee oi$	1 2 3 4 mi
during	1 2 3 4 d	1 2 3 4 d
is-during	1 2 3 4 $di \vee f \vee o$	1 2 3 4 $di \vee si \vee oi$
overlaps	1 2 3 4 o	1 2 3 4 $o \vee s \vee d$
is-overlapped	1 2 3 4 $oi \vee f \vee d$	1 2 3 4 oi
starts	1 2 3 4 s	1 2 3 4 $s \vee d$
is-started	1 2 3 4 $si \vee e \vee s$	1 2 3 4 $si \vee oi$
finishes	1 2 3 4 $f \vee d$	1 2 3 4 f
is-finished	1 2 3 4 $fi \vee o$	1 2 3 4 $fi \vee e \vee f$

figure 69 : relation temporelle entre les bornes d'une séquence et un événement simple

[76] Voir « Vers un algorithme de subsomption » page 86.

On détermine alors deux relations temporelles B et E, B étant la relation déterminée à partir du premier élément de la séquence et E la relation déterminée à partir du dernier élément. Soit R_B – respectivement R_E – la relation observée entre l'événement *e* et le premier – resp. le dernier – élément de *s*. B – resp. E – est l'union des relations obtenues en appliquant la table de correspondance illustrée ci-dessus sur toutes les relations élémentaires contenues dans R_B – resp. R_E. Finalement, la relation temporelle R entre l'événement *e* et la séquence *s* est l'intersection de B et de E.

Nous l'avons dit, la table de correspondance ne distingue pas les séquences de longueur un des séquences de longueur supérieure à un, comme c'était le cas pour le calcul des relations temporelles entre un événement et les bornes d'une séquence à partir de la relation temporelle entre l'événement et la séquence entière. En effet, les disjonctions indiquées sur la figure prennent en compte les deux cas. Une autre solution aurait évidemment consisté à considérer séparément les séquences de longueur un. Le premier et le dernier élément de la séquence étant confondus, on a nécessairement $R_B=R_E=B=E=R$.

9.3.5 Démons pour les débuts et fins de séquences

Le filtrage des contraintes que nous avons décrites jusqu'ici était déclenché par le démon *valueDemon*. Nous l'avons vu, BackJava permet de définir de nouvelles classes de démons[77]. Deux démons sont définis pour traiter des cas où une variable itérée a pour domaine un ensemble de séquences commençant – ou finissant – par le même élément. Ces démons sont armés par certaines des méthodes spécifiques de filtrage décrites ci-dessus[78]. Lorsque toutes les valeurs possibles d'une variable itérée sont des séquences qui commencent par le même élément, il est possible dans certains cas de réduire le domaine d'autres variables. Ainsi, considérons la relation temporelle *is-met*, illustrée par la figure 70[79]. Si le domaine de la variable associée à la séquence de plans n'est constitué que de séquences commençant par le plan numéro 6, il est possible de supprimer l'événement *jingle-2* du domaine de la variable associée au jingle.

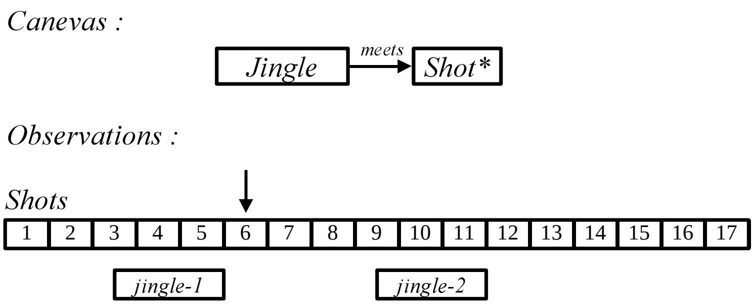


figure 70 : démon pour le filtrage de la contrainte *meets*

Nous ne décrirons pas plus précisément ces classes de démon, car leur utilité ne s'est pas avérée flagrante dans les cas que nous avons eu à traiter. Le mécanisme général nous semble cependant intéressant, qui consiste à identifier des configurations fréquentes des

[77] Voir « Un framework objet » page 107.

[78] Voir « Procédures de suppression des séquences » page 119.

[79] Cette figure reprend la figure 66 page 121.

valeurs des variables pour lesquelles il est possible de mettre en œuvre des méthodes de filtrage performantes.

9.4 Contraintes « no … between »

Les contraintes spécifiques du type « no … between » permettent en particulier d'exprimer la succession temporelle[80]. La forme de ces contraintes est : « no *C* between n_1 n_2 » où *C* est un concept d'événement élémentaire ou un canevas borné et n_1 et n_2 des composants quelconques – simples ou itérés – du canevas. Une telle contrainte interdit qu'il existe une instance de *C* dans le graphe d'observations qui soit *after* n_1 et *before* n_2. De manière analogue aux contraintes temporelles entre variables simples, ces contraintes binaires sont implémentées par la classe CLNoBetweenCt, une classe de contraintes héritant de la classe BJGenericBinaryCt de BackJava. Le concept *C* apparaissant dans la définition de la contrainte est représenté par une variable d'instance de CLNoBetweenCt. Le filtrage de la contrainte est réalisé par l'implémentation de la méthode *isSatisfied* qui est appelée lorsque les deux variables de la contrainte sont instanciées. Cette méthode retourne *vrai* si aucune instance de *C* n'apparaît dans le graphe d'observations entre les valeurs des variables.

```
public boolean isSatisfied(Object a, Object b)
{
   Node n1 = (Node) a;
   Node n2 = (Node) b;

   return !observationGraph.isAnInstanceOfBetween(concept, n1, n2);
}
```

Cette méthode implique une recherche dans le graphe d'observations dont la complexité est linéaire en fonction du nombre d'observations. Lorsque les observations sont datées, cette recherche peut bien sûr être faite de manière dichotomique.

10 Interprétation des résultats

Après la phase de reconnaissance des occurrences d'un canevas *T* dans le graphe d'observations se déroule la phase d'interprétation des résultats. Il faut alors distinguer deux cas, selon que la phase de reconnaissance est terminale où selon qu'elle intervient comme étape intermédiaire de la reconnaissance d'un canevas de niveau supérieur. Dans ce dernier cas, l'interprétation consiste simplement à ajouter les occurrences reconnues au graphe d'observations. Les observations étant datées, il faut alors calculer les dates des occurrences reconnues. Seuls les canevas « bornés »[81] pouvant faire l'objet d'une reconnaissance intermédiaire, nous sommes assurés que ce calcul est possible. Le principe de ce calcul consiste simplement à affecter comme date de début d'une occurrence *t* soit la date de début d'un composant repéré comme étant dans la relation *equals*, *starts* ou *is-started*

[80] Voir « Enrichir le langage par de nouvelles contraintes » page 93.
[81] voir « Canevas bornés » page 83.

avec *t*, soit la date de fin d'un composant repéré comme étant *meets* avec *t*. Le calcul est effectué de manière symétrique pour la date de fin de *t*.

Dans le cas où le calcul est terminal, c'est à l'application d'interpréter les résultats. Dans notre cas, nous avons rencontré principalement deux types d'utilisation possible des résultats : 1) stockage en base de données en vue d'une annotation ultérieure, comme c'est le cas dans le système DiVAN et 2) présentation à l'écran. Dans le premier cas, l'interprétation consiste à traduire les informations obtenues dans le formalisme permettant de manipuler le modèle de données de DiVAN. Dans le second cas, nous avons par exemple généré des pages au format AEDI v.3.0[82], ce qui permet de naviguer dans les occurrences des canevas par l'intermédiaire d'un navigateur HTML, les éléments temporels affichés – instances des concepts élémentaires d'événement et instances de canevas – étant synchronisés avec un *player* vidéo.

11 Discussion

Nous abordons dans cette partie quelques questions que nous n'avons pas poussées jusqu'à leur terme, principalement pour des raisons de temps, et qui demanderaient des efforts supplémentaires.

11.1 Complexité

Le principe général de la phase de reconnaissance consiste à filtrer les contraintes lorsque l'une des variables de la contrainte est instanciée. Les procédures de filtrage assurent alors la cohérence d'arc de la contrainte. La méthode générale de cohérence d'arc, telle qu'elle est présentée dans (Mackworth, 1977), est linéaire en fonction de la taille du domaine de la variable non instanciée. Plus précisément, pour chaque séquence maximale de longueur N du domaine de la variable non instanciée, cette méthode est linéaire en fonction du nombre total de sous-séquences, soit $\frac{N.(N+1)}{2}$. La représentation des domaines des variables itérées que nous avons choisie nous permet d'améliorer sensiblement cette complexité pour les contraintes impliquant une variable itérée. Ainsi, dans le meilleur des cas, par exemple pour une contrainte temporelle *before* entre une variable itérée et une variable quelconque – itérée ou non –, la complexité est linéaire pour chaque séquence maximale de taille N en fonction de N. Rappelons que le principe de filtrage des contraintes temporelles impliquant une variable itérée consiste pour certaines relations (*meets*, *before*, par exemple) à appeler des méthodes de filtrage spécifiques, et pour le cas général d'une disjonction quelconque, à appeler une méthode de filtrage pour chaque relation temporelle ne faisant pas partie de la disjonction[83]. La complexité de ces méthodes varie entre $O(N)$ et $O(N^2)$ pour chaque séquence maximale de longueur N.

---

[82] Voir « A.E.D.I » page 24.

[83] Voir « Contraintes temporelles impliquant une variable itérée » page 118.

11.2 Observations non précisément datées

Nous avons pour chaque point présenté ci-dessus fait l'hypothèse que les observations étaient précisément datées. Cependant, un certain nombre de méthodes ont été implémentées pour le cas plus général d'observations non précisément datées, c'est-à-dire pour le cas où les relations temporelles observées sont de nature disjonctive. C'est le cas par exemple des méthodes de filtrage des contraintes temporelles. Ca n'est malheureusement pas le cas de toutes les méthodes. Ainsi, par exemple, la méthode permettant de déterminer si un canevas est temporellement borné fait l'hypothèse que les concepts d'événements élémentaires sont bornés, soit, autrement dit, que les observations initiales sont précisément datées. Cette méthode devrait être modifiée pour tenir compte d'observations non précisément datées.

11.3 Propagation des contraintes temporelles

Dans le cas de canevas imbriqués[84], certaines des contraintes temporelles portant sur les sous-canevas doivent être propagées. La méthode de propagation que nous avons choisie consiste à calculer la couverture minimale du graphe de contraintes en appliquant la méthode de 3-cohérence proposée par Allen, méthode qui est potentiellement incomplète. Cette méthode a pour effet d'imposer une contrainte temporelle pour chaque couple de sommet du graphe de contraintes, ce qui entraîne dans le CSP une contrainte temporelle pour chaque couple de variables, certaines de ces contraintes pouvant être redondantes. Cela n'affecte pas le résultat mais peut avoir des incidences sur la rapidité de la résolution. Une méthode de propagation plus « locale » permettrait de résoudre ce problème.

12 Conclusion

Nous avons présenté dans ce chapitre une formulation du problème de la reconnaissance de canevas comme problème de satisfaction de contraintes. Le principe de résolution de ce problème repose essentiellement d'une part sur une représentation implicite efficace des domaines des variables itérées représentant des séquences itérées d'observations de même classe et d'autre part sur des méthodes rapides de filtrage des contraintes temporelles impliquant de telles variables. Nous avons également présenté le système Clavis (*Classification of Audiovisual Sequences*), une implémentation de cette méthode de reconnaissance de canevas. Clavis est construit comme une spécialisation du système de satisfaction de contraintes BackJava. Nous avons enfin souligné quelques uns des problèmes auxquels nous avons été confrontés, en particulier concernant la gestion d'observations non précisément datées.

---

[84] Voir « Imbrication de canevas » page 103.

Chapitre 5
Expérimentations

La série d'expérimentation que nous présentons ici a été réalisée dans le cadre du projet DiVAN (*Distributed Audio-Visual Archives Network*)[85] ; certaines de ces expérimentations ont été présentées dans (Carrive *et al.*, 2000a).

1 Le projet DiVAN

Le projet DiVAN est un projet européen Esprit (n°24956) qui vise à construire une base de données distribuées de contenus audiovisuels indexés. Le consortium DiVAN réunit des industriels (Intracom en Grèce, Tecmath en Allemagne), des laboratoires de recherche (ICS-Forth en Crète, l'INRIA-IRISA en France, l'Université de Darmstadt en Allemagne) et des centres d'archives audiovisuelles (la Télévision Nationale Grecque – ERT, la Radio-télévision Nationale Italienne – RAI et l'Institut National de l'Audiovisuel – INA).

L'architecture générale du projet est illustrée par la figure 71, reprise de la documentation du projet. Les principaux composants du système sont :

- *Material Feeding Station* : sert à alimenter le serveur video ;
- *Documentation Station* : station d'annotation ;
- *Storage Server* : serveur vidéo ;
- *Metadata Server* : serveur de « métadonnées » - ou descriptions, ou encore données d'indexation ;
- *Internal User Station* : utilisée pour l'administration ;
- *External User Station* : utilisée pour les requêtes, la consultation et la navigation.

DiVAN se veut un système complet d'archivage de la vidéo et aborde par conséquent beaucoup d'aspects différents liés à cette problématique : alimentation de la base vidéo (numérisation, compression, documentation de catalogage), indexation automatique, annotation manuelle, moteur de requêtes, serveur vidéo et serveur de données de description, interfaces de navigation et de visualisation, etc.

---

[85] http://divan.intranet.gr

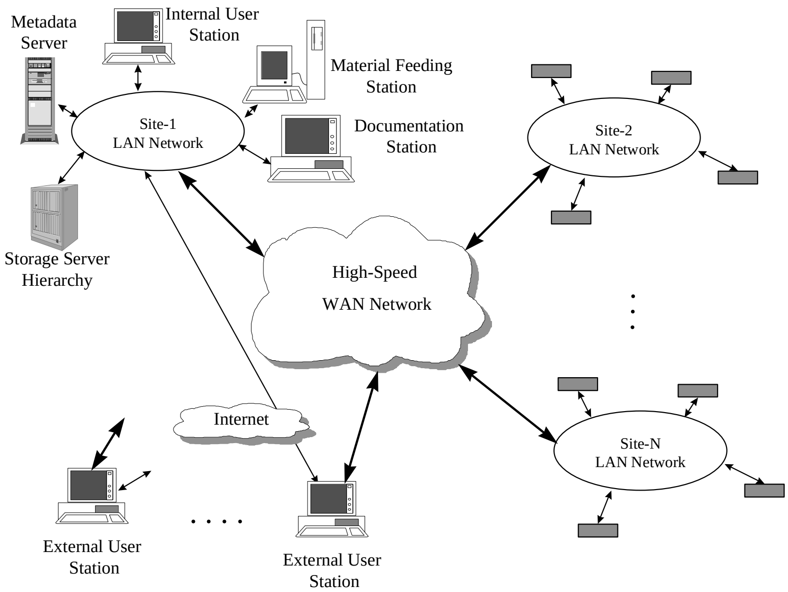


figure 71 : architecture générale du projet DiVAN

L'indexation automatique comporte deux phases d'analyse : une première phase d'analyse du flux audio/vidéo, et une seconde phase de regroupement des résultats de la première phase. Les outils d'analyse constituant la première phase d'analyse sont les suivants (voir aussi (Bouthemy *et al.*, 1999; Garcia & Tziritas, 1999; Garcia & Apostolidis, 2000)) :

- Segmentation en plans et détection de transitions progressives ;
- Recherche d'images représentatives des plans (*keyframes*) ;
- Détection de régions correspondant à un visage dans les images représentatives ;
- Détection de régions correspondant à du texte dans les images représentatives ;
- Discrimination parole/musique/bruit ;
- Détection de logos ;
- Détection de jingles sonores.

Les résultats de ces outils d'analyse sont des instances de classes organisées en taxinomie. La figure 72 donne un aperçu légèrement simplifié de la taxinomie employée dans DiVAN. Les cases ombrées sur la figure indiquent des classes spécifiques à une collection donnée (le journal de la chaîne « M6 », par exemple).

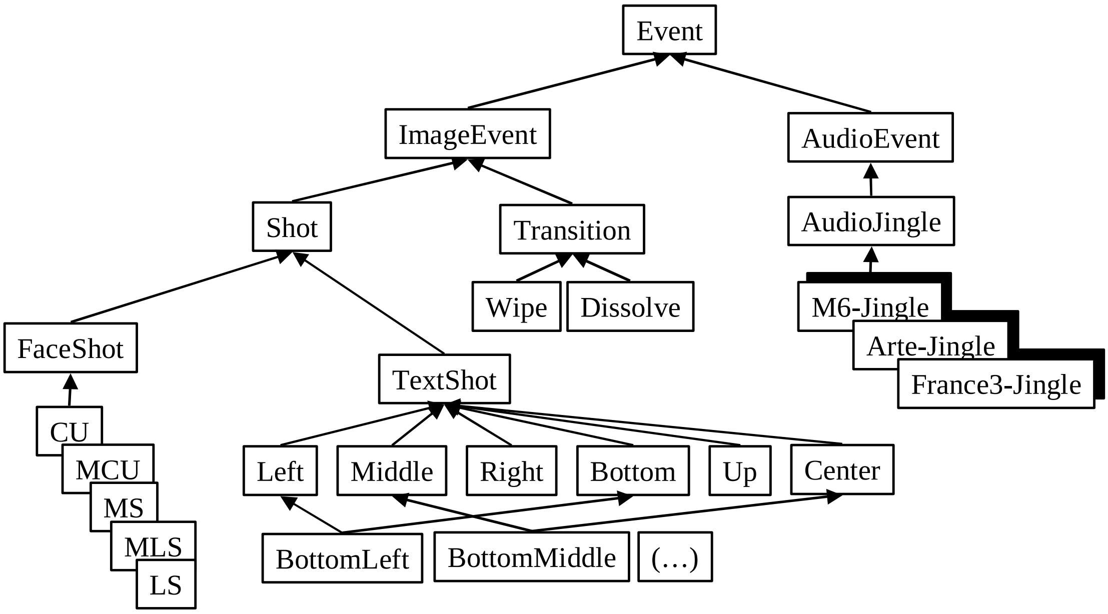


figure 72 : aperçu de la taxinomie d'événements du système DiVAN

La seconde phase d'analyse est appelée phase *d'interprétation* et son but est d'obtenir une macro-segmentation du document analysé. Deux méthodes d'interprétation sont intégrées dans le système. La première de ces méthodes, employée lorsque aucune bibliothèque de canevas n'est disponible, est une extension de la méthode non supervisée de regroupement – clustering – proposée par (Yeung *et al.*, 1996)[86] mise en œuvre à l'INA par Emmanuel Veneau (Veneau *et al.*, 2000). La figure 73 illustre de manière simplifiée le processus de reconnaissance par canevas tel qu'il est intégré dans le prototype DiVAN. L'étape de classification des segments fournis par les outils d'analyse devrait être prise en charge par le système CLASSIC, comme nous l'avons présenté au chapitre précédent[87]. Pour des raisons techniques, le système CLASSIC n'a pas été encore intégré dans le prototype courant, et la classification des segments se fait simplement en affectant à chaque segment une ou plusieurs classes de la taxinomie en fonction de l'algorithme d'analyse : la classe *ShotEvent*, par exemple, pour l'algorithme de segmentation en plans, *Wipe* ou *Dissolve* pour la détection de transitions progressives, *M6AudioJingle* pour l'outil de détection de jingles audio paramétré pour la reconnaissance du jingle de la chaîne « M6 », etc.

[86] Voir également « Méthodes de regroupement » page 13.
[87] Voir « Communication CLASSIC/Clavis » page 101.

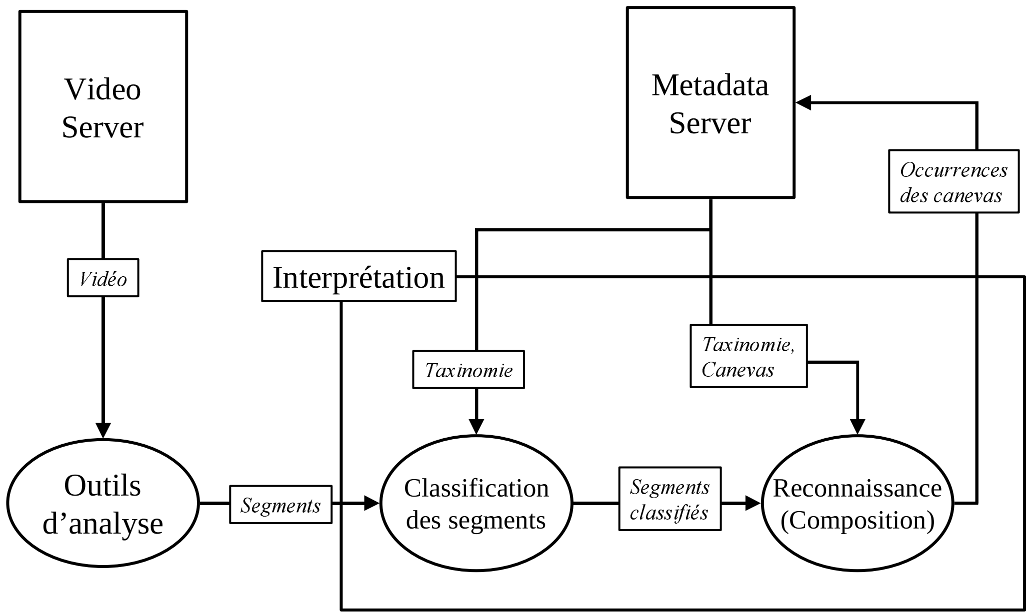


figure 73 : analyse et Interprétation dans le système DiVAN

2 Etude des journaux télévisés du corpus AIM

L'Action « Indexation Multimédia » (AIM), initialement action conjointe des GDR CHM et ISIS, et maintenant GT10 du GDR PRC ISIS, s'est constitué un corpus de vidéos numérisées pour permettre de tester et de confronter les résultats des travaux effectués par le groupe : segmentation en plans, sélection d'images représentatives, discrimination bruit/parole/musique, etc. Ce corpus d'environ 3h40, élaboré à l'INA, est composé de :

- 30 minutes de publicités et de bandes annonces ;
- Un documentaire : édition d'« Envoyé spécial » du 23 mai 1996, *Histoire d'eau* ;
- Une fiction : un épisode de *Chapeau melon et bottes de cuir* ;
- Journaux d'actualité : éditions de début de soirée (20h, 20h30) du journal télévisé des six chaînes nationales, TF1, France 2, France 3, Arte, M6, et Canal+, du 13 juillet 1996.

La segmentation en plans et en transitions graduelles a été effectuée manuellement sur tout le corpus afin de servir de référence aux algorithmes de segmentation (Ruiloba *et al.*, 1999).

Nous nous intéressons ici aux six journaux du corpus, et plus particulièrement à la macro-segmentation des journaux en reportages. Le tableau 7 recense les durées, le nombre de plans et le nombre de transitions graduelles de chacun des journaux du corpus.

Journal	Durée	# Plans	# Transitions
France 2	14 mn 48 s	157	15
France 3	15 mn 15 s	216	30
TF1	14 mn 49 s	184	24
Arte	09 mn 57 s	161	16
M6	06 mn 35 s	91	0
Canal+	05 mn 47 s	61	3

tableau 7 : durée, plans et transitions graduelles des journaux d'AIM

Sur deux des journaux, Arte et M6, nous avons détecté les occurrences des jingles audio. Nous avons enfin effectué une annotation manuelle des plans en leur associant un ou plusieurs concepts d'une taxinomie d'événements élémentaires.

2.1 Journal de TF1

Nous donnons ci-dessous un extrait de la fiche collection correspondant à l'édition du journal de « TF2 » du corpus AIM en l'illustrant par des images extraites de cette même édition.

Chapeau	Deuxième édition quotidienne de l'actualité nationale et internationale de la journée, le journal de 20 heures est diffusé en direct et dure environ 40 minutes. Il est réalisé en studio et s'organise en une succession de plateaux - reportages. Le plateau est animé en alternance par Patrick Poivre d'Arvor du lundi au jeudi et Claire Chazal du vendredi au dimanche.
Résumé	Le présentateur annonce lui-même le sommaire du journal. Au cours de ce sommaire il peut y avoir de brèves images, présentant les personnes invitées durant l'édition. L'organisation de l'information n'est pas structurée de façon rigide. Elle suit les temps forts de l'actualité. Les premiers sujets vont de l'actualité nationale à l'actualité internationale sans ordre bien précis. Le journal se termine en principe par les nouvelles du sport et des arts et spectacles. Selon l'actualité, il peut y avoir une chronique animée par un journaliste spécialisé de la rédaction. Le présentateur nomme les journalistes ayant préparé le sujet à venir au moment du plateau lancement. En fin d'édition, le présentateur fait l'annonce des programmes de la soirée. A partir du 1er avril 1996, le présentateur rappelle en outre en fin d'édition les principaux événements abordés pendant le journal. Particularité : La durée du journal peut varier en fonction de l'actualité.
Dispositifs	Générique début : Le générique est organisé en deux temps. Une série d'images de la rédaction aux couleurs sombres avec formation du titre TF1 20 heures sur une carte du monde, puis un encadré avec le présentateur en plan rapproché qui annonce le sommaire. La musique court sur toute la durée du générique avec une intensité de volume allant du fort au faible sur l'annonce des titres ; puis zoom avant sur le présentateur qui apparaît plein écran pour le lancement du premier sujet.

La figure 74 illustre le générique de début. La transition entre le plan du présentateur et les titres, puis entre les titres et le présentateur est soulignée par un volet. Les titres sont également séparés par des volets.

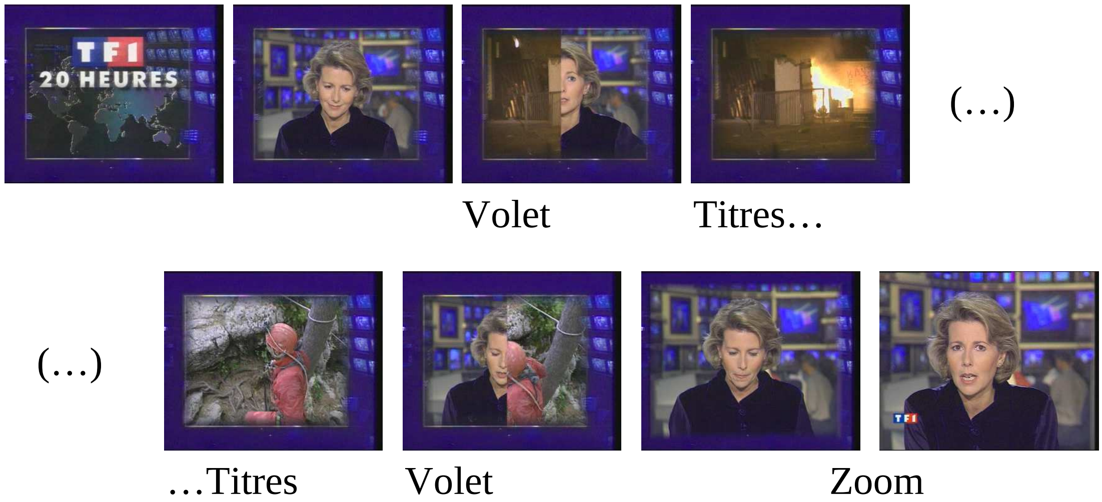


figure 74 : TF1, générique de début

	Générique fin : vue d'ensemble du studio. Fondu et départ du générique dans un mouvement descendant par panneaux sur la musique du générique de début. Il peut y avoir une annonce concernant les services télématiques proposés par la chaîne. Dispositifs plateau : dispositifs techniques multicaméras : une caméra pour prendre le journaliste de face, une autre pour prendre les plans d'ensemble du plateau.

Ces deux types de plan de plateau sont illustrés par la figure 75.

figure 75 : TF1, plans de plateau

	Incrustations : il existe plusieurs types d'incrustations : Affichage du logo de la chaîne : le logo de la chaîne est présent durant toute l'édition aussi bien sur les plateaux que sur les

reportages. Il est situé en bas à gauche de l'écran.

Affichage du nom du présentateur : il n'apparaît qu'une fois, dès l'apparition du journaliste plein cadre.

Affichage des incrustations plateau sous forme d'icônes : elles servent d'illustration sur certains sujets. Disposées en haut à droite, elles ont deux formes : un carré pour les cartes et les personnes, un losange pour les symboles. Au bas de l'icône, présence d'une barre horizontale et d'un mot clé thématique relatif au sujet.

Ces trois types d'incrustations sont illustrés par la figure 76.

Présentateur Symboles Cartes et personnes

figure 76 : TF1, incrustations plateau

[…]

Décor et mobilier : Le mobilier n'apparaît que sur les vues d'ensemble du plateau.

Identification de trois "toiles de fond" : 1) Un panneau de moniteurs de tailles diverses. L'ensemble est dans des teintes foncées de bleu, gris. Ce fond est utilisé sur toute la durée du journal sauf lorsqu'il y a un invité plateau. 2) Un panneau avec une colonne de moniteurs à droite affichant le logo de TF1 sur leurs écrans. Il est utilisé sur les séquences plateau invité comme toile de fond pour le journaliste. 3) Un panneau sombre avec le logo TF1 en biais et en grand format. Ce panneau est utilisé en toile de fond pour l'invité plateau.

Présentateur : L'habillement des présentateurs est de facture classique. Costume cravate aux couleurs foncées pour Patrick Poivre d'Arvor ; tenues aux tons pastel pour Claire Chazal.

Présence d'incrustations dans les reportages pour :

Mentionner le nom du/des journalistes, commentateurs, opérateurs de prise de vues : en bas à gauche en début de sujet. Les couleurs rouge et bleue de la chaîne sont rappelées par deux petits rectangles en début d'incrustation.

	Mentionner des lieux, ou des collaborations avec un journal local : en haut à gauche. Mentionner les noms et le statut des personnes interrogées : en bas à droite.

Ces différentes catégories d'incrustation de texte dans les reportages sont illustrées par la figure 77. Dans cet exemple, les noms de personne incrustés sont toujours situés en bas à gauche qu'il s'agisse d'une personne interviewée ou du reporter.

figure 77 : incrustations de texte dans les reportages de TF1

	Les reportages peuvent être illustrés par des écrans synthés (cartes, graphiques, maquettes...) […] Le journaliste est souvent présent à l'image à la fin de son reportage. Construction générale : Il n'y a pas de changement de caméra visible à l'écran. Les changements ne sont font que lorsqu'il y a un invité plateau. Ils sont identifiables par un changement de toile de fond. De façon générale, le rythme de l'édition est coulant.

	Cadrage et valeurs de plans : cadrage serré pour le journaliste pris en plan rapproché par caméra fixe de face. Enchaînement plateau - reportage : il se fait soit par un fondu soit par un volet allant de gauche à droite.

Des exemples d'enchaînement reportage-plateau sont donnés par la figure 78. Notons que les transitions progressives (fondu ou volet) sont plus fréquentes entre un reportage et un plateau qu'entre un plateau et un reportage. La figure 79 illustre les plans du présentateur et les transitions progressives de l'édition AIM du journal de TF1.

Volet de gauche à droite

Fondu enchaîné

figure 78 : TF1, enchaînements reportage – plateau

La figure 79 illustre les transitions progressives et les plans du présentateur de l'édition du corpus AIM du journal de France 2. On peut remarquer sur la figure que chaque plan du présentateur est introduit et/ou conclu par une transition progressive.

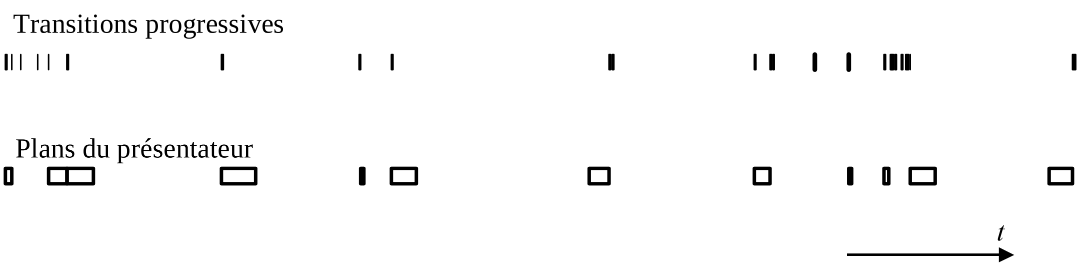


figure 79 : TF1, transitions progressives et plans du présentateur

2.2 Journal de France 2

L'INA ne possède pas à l'heure actuelle de fiche collection correspondant à l'édition du journal du soir de « France 2 » présent dans le corpus AIM. Le journal de « France 2 » respecte une alternance stricte plateaux-reportages. Les plans de plateau sont de deux sortes : le présentateur en plan poitrine ou bien l'ensemble du studio en plongée (voir figure 80).

figure 80 : plans de studio du journal de « France 2 »

Les incrustations de texte au cours d'un reportage sont de deux sortes (voir figure 81). Une zone de texte en bas de l'écran placée juste à gauche du logo de la chaîne indique le nom du lieu où se déroule l'action. Une zone de texte placée en bas et à gauche de l'écran, dans un bandeau bleu, indique le du personnage présent à l'écran, journaliste ou personne interviewée.

Nom de lieu

Nom de personne

figure 81 : incrustations de texte dans les reportages de France 2

Les enchaînements plateau-reportage sont en général soutenus par un fondu enchaîné. La figure 82 illustre les transitions progressives et les plans du présentateur de l'édition du corpus AIM du journal de « France 2 ». On peut remarquer sur la figure que chaque plan du présentateur est introduit et/ou conclu par une transition progressive.

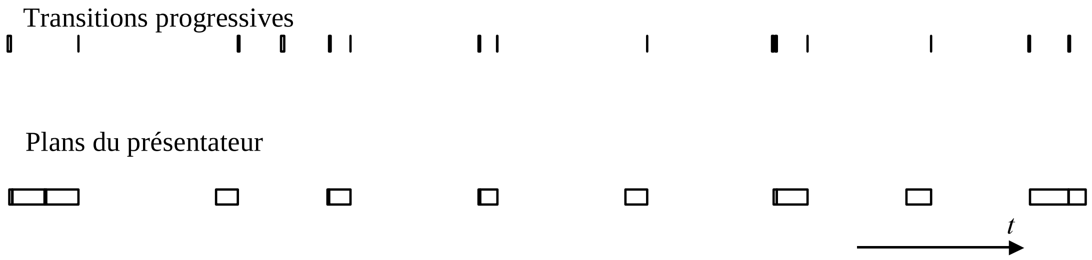


figure 82 : France 2, transitions progressives et plans de studio

2.3 Journal d'Arte

Des extraits de la fiche collection du journal « 8 ½ » de la chaîne « Arte » est donnée et illustrée par des images ci-dessous.

Chapeau	Conçu à Strasbourg par une équipe franco-allemande, ce journal télévisé de format court traite l'information sans commentaires ni éditoriaux. Cette forme stricte du traitement de l'actualité trouve son identité grâce à une présentation et un habillage recherchés.
Résumé	Le 8 1/2 profite de son statut supranational pour délivrer chaque soir une vision européenne, politique et culturelle, de l'actualité. Les 7 ou 8 sujets (en moyenne), présentés en voix OFF par les journalistes, s'enchaînent selon un ordre défini : les premiers reportages, le plus fréquemment montages d'images puisées dans la banque des EVN auxquels peuvent être intégrés des interviews fait par l'équipe du journal, sont consacrés à l'actualité internationale ; suivent des brèves, intitulées "FAX", qui viennent renforcer rapidement les nouvelles du jour ; enfin 1 ou 2 sujets, souvent des reportages "maison", traite(nt) de l'actualité culturelle (spectacles, expositions, concerts ...) en France et/ou en Allemagne.
Dispositifs	Générique début : ayant recours à la palette graphique, il est construit sur une animation à partir du titre 8 1/2 en surimpression d'un montage d'images couleur évoquant l'actualité : ces images ont été coupées en deux par leur milieu, leur partie basse apparaissant en négatif et leur partie haute en positif. Les caractères du titre se dessinent en un effet tournant et grossissant. La musique électronique est en synchronisation avec le montage et l'animation. Le principe du générique et la musique sont réutilisés pour les panneaux de transition et de lancement des sujets et

	reportages.

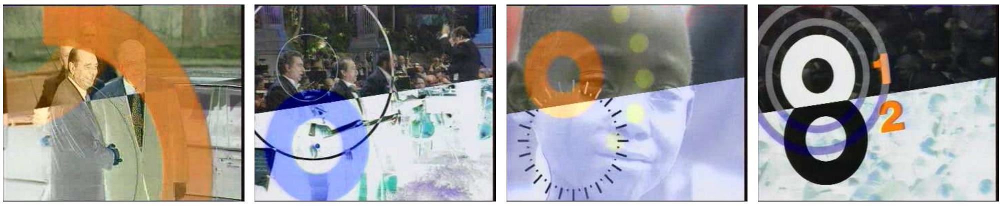


figure 83 : Arte, générique de début

Dispositifs généraux	Générique fin : réutilisation des panneaux de lancement des sujets (même principe que le générique début mais dans les tons de gris), même musique, générique de l'émission (noms des concepteurs et journalistes). Tous les reportages sont commentés en voix OFF Chaque sujet est lancé par un panneau de transition sur lequel est imprimé son titre.

figure 84 : Arte, panneaux de transition, et panneau « FAX »

Dispositifs éléments	Les noms des personnes interviewés ainsi que les adresses des lieux de spectacle ou d'exposition apparaissent à l'écran (nom en lettre banche sur fond bleu dans un bandeau au bas de l'image).

Sur l'édition d'AIM du « 8 ½ », les noms de lieu sont indiqués à droite de l'écran, en haut en français et en bas en Allemand ou au milieu lorsque le terme est identique dans les deux langues.

Noms de personne

Noms de lieu

figure 85 : Arte, noms de personne et noms de lieu

Construction générale	L'ensemble du 8 1/2 se veut informatif, didactique et rythmé. Les images des reportages sont des images d'illustration (montage descriptif) Montage-son : le commentaire est informatif et narratif.

2.4 Journal de M6

Des extraits de la fiche collection illustrée du journal « 6 minutes » de la chaîne « M6 » est donnée ci-dessous.

Chapeau	Le « 6 minutes » de M6 est un journal quotidien d'actualités nationales et internationales, tout en images d'une durée légèrement supérieure à 6 minutes. Les reportages sont commentés par le journaliste Jean-Francois RICHARD en voix off.
Résumé	Ce journal tout en images se compose uniquement de reportages courts (environ une minute), au nombre d'une dizaine par édition. Les reportages commencent souvent par une interview entrant directement dans le vif du sujet, le journaliste développant ensuite la problématique. La place des interviews est importante proportionnellement à la durée totale des reportages. Les reportages sont lancés par une accroche incrustée qui sert de titre et qui est reprise dans la première phrase du commentaire. Le bandeau "VITE" annonce les Brèves. Cette rubrique, constante dans toutes les éditions, se situe en fin d'édition. Les autres accroches sont puisées dans un 'catalogue' de mots et peuvent être repris d'une édition à l'autre suivant les sujets traités (ex : l'affaire, la guerre, zinzin, attention...).

figure 86 : M6, panneaux d'accroche

	Le ton du reportage et du commentaire est donné par l'accroche choisie.
Dispositifs	Générique début : Sur fond blanc le logo M6 est plein écran. Un bandeau noir se superpose au milieu de l'écran sur lequel est écrit en blanc '6 minutes', le logo de M6, rouge et noir, clignote à droite. Une deuxième ligne blanche horizontale traverse relativement lentement l'écran de bas en haut, suivie d'une ligne verticale. Le petit logo M6 se trouve cette fois accolé dans l'angle droit

	supérieur de leur intersection en haut à gauche de l'écran. Une large bande rouge est alors amenée par un nouveau trait vertical blanc se déplaçant latéralement de gauche à droite. Puis le titre du journal apparaît de droite à gauche sur l'écran en caractères blancs minuscules.

Le générique de l'édition d'AIM du journal est différent du générique décrit dans la fiche collection.

	Générique sur une musique inspirée du morse. Générique fin : Ecran blanc avec un bandeau noir en haut et un bandeau noir en bas de l'écran sur lequel est inscrit le nom du chef d'édition et celui du monteur. Il est suivi d'un écran identique portant les mentions des autres membres du générique.

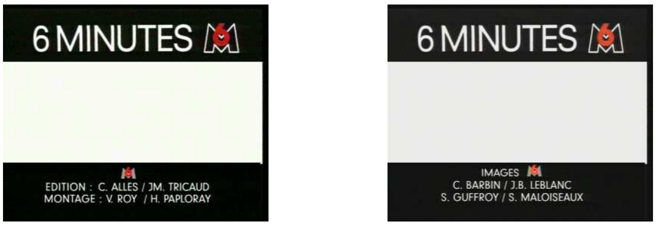


figure 87 : M6, générique de fin

	Reprise du générique du début sans le sigle M6 plein écran, en musique.
Dispositifs généraux	Toutes les couleurs utilisées pour les incrustations rappellent l'habillage de la chaîne (rouge, vert, blanc, noir).
Dispositifs éléments	Chaque sujet est annoncé par un panneau : deux bandeaux noirs encadrant un rectangle blanc. Le bandeau supérieur reprend le titre générique "6 minutes". L'accroche du sujet est incrustée sur le bandeau inférieur. Le lieu de tournage est indiqué dans chaque sujet par une incrustation latérale.

figure 88 : M6, incrustation des noms de lieu

Le nom et la qualité des personnes interviewées s'inscrit dans un bandeau noir en bas de l'écran.

figure 89 : M6, incrustation des noms de personne

Les sujets sont courts mais ne font pas l'objet d'un montage cut.

2.5 Journal de Canal +

Aucune fiche collection n'a été définie à l'INA correspond à l'édition du corpus AIM du journal du soir de « Canal + ». Ce journal est d'un format court (moins de 6 minutes) et respecte une alternance stricte plateaux/reportages. Les plans de studio montrent le présentateur sur un fond fixe (voir figure 90).

figure 90 : Canal +, plan du présentateur

Les plans de reportage sont nus, sans inscription de texte. Le montage est un montage « cut », sans effet d'édition.

2.6 Reportages des JT du corpus AIM

Nous nous intéressons ici à la définition et à la reconnaissance des reportages des journaux télévisés du corpus AIM. Ces journaux, ainsi que la plupart des journaux télévisés, se répartissent globalement en deux catégories en fonction de la présence ou l'absence d'un (ou plusieurs) présentateurs à l'écran. Les journaux du corpus AIM avec présentateur (TF1, France 2, France 3 et Canal+) n'emploient qu'un seul présentateur. La figure 91 illustre la répartition des journaux AIM dans ces deux catégories.

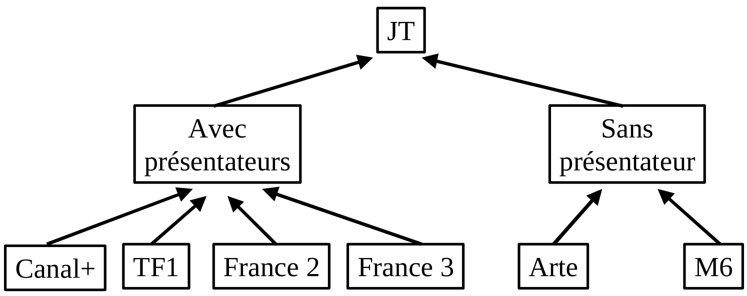


figure 91 : deux catégories de JT, selon la présence de présentateurs

La structure temporelle est schématiquement constituée d'une alternance plateaux-reportages pour les journaux avec présentateurs et de reportages courts entrecoupés de panneaux de transitions sur fond musical (jingles) pour les journaux sans présentateur.

2.6.1 Reportages des journaux avec présentateurs

Les reportages des journaux avec présentateur sont tout d'abord définis de manière très simple comme la séquence de plans se déroulant entre deux plans successifs du présentateur :

ReportFromAnchor
nodes
  s_1 [*AnchorShot*]
  s_2 [*Shot**]
  s_3 [*AnchorShot*]
constraints
  s_1 *meets* s_2
  s_2 *meets* s_3
  s_1 *starts* this
  s_2 *finishes* this
  no *AnchorShot* between s_1 s_3

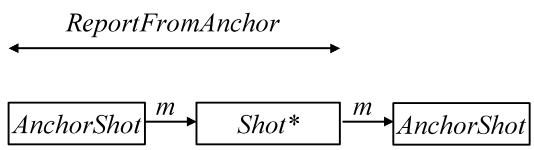
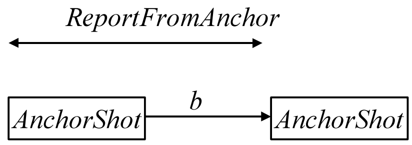


figure 92 : définition d'un reportage d'après deux plans du présentateur

Notons que le nœud s_2 associé à Shot* n'est pas nécessaire à la définition du canevas, qui peut être remplacée par la définition suivante :

ReportFromAnchor
nodes
s_1 [*AnchorShot*]
s_2 [*AnchorShot*]
constraints
s_1 *before* s_2
s_1 *starts* this
s_2 *is-met* this
no *AnchorShot* between s_1 s_2



Cette dernière définition présente toutefois l'inconvénient de ne pas faire apparaître les composants constitutifs du reportage – en l'occurrence la séquence de plans –, ce manque d'information pouvant être gênant, dans le cas par exemple d'une présentation hiérarchique des canevas reconnus.

L'extension temporelle d'un reportage inclut le premier plan du présentateur, qui est en général un plan d'introduction au reportage et « s'arrête juste avant » (*meets*) le plan du reporter suivant. Lorsqu'un seul plan du reporter sépare les reportages, l'itération du canevas *ShotFromAnchor** permet d'obtenir des séquences itérées de reportages, comme l'illustre la figure 93.

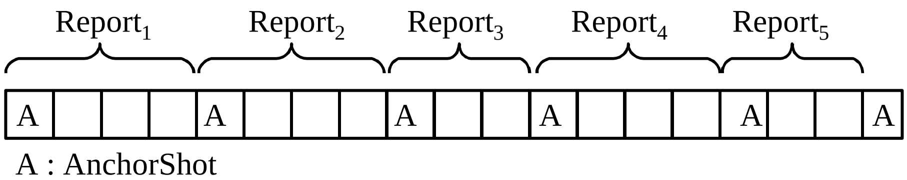
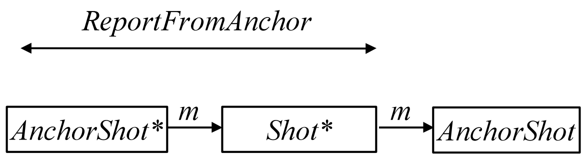


figure 93 : itération des reportages

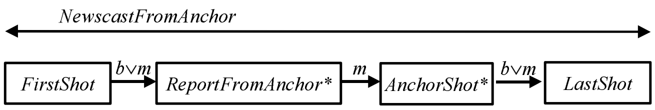
Le canevas doit être adapté lorsque plusieurs plans du présentateur se suivent, ce qui est fréquent. Afin de traiter par un seul canevas les cas où un seul plan du présentateur sépare deux reportages et les cas où plusieurs plans du présentateur se succèdent, nous agrégeons tous les plans du présentateur en début de reportage, et définissons ainsi le canevas suivant :

ReportFromAnchorSeq
nodes
s_1 [*AnchorShot**]
s_2 [*Shot**]
s_3 [*AnchorShot*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_1 *starts* this
s_2 *finishes* this
no *AnchorShot* between s_1 s_3



On peut alors définir le découpage complet d'un journal en reportages par le canevas défini ci-dessous, qui utilise les concepts *FirstShot* et *LastShot* en plus des concepts *AnchorShot* et *Shot* :

NewscastFromAnchor
nodes
 s_1 [*FirstShot*]
 s_2 [*ReportFromAnchor**]
 s_3 [*AnchorShot**]
 s_4 [*LastShot*]
constraints
 s_1 {*before meets*} s_2
 s_2 *meets* s_3
 s_3 {*before meets*} s_4
 no *AnchorShot* between s_1 s_2
 no *AnchorShot* between s_3 s_4
 reduce[88]



La relation *before* ∨ *meets* entre le premier plan (s_1 [*FirstShot*]) et la séquence de reportage (s_2 [*ReportFromAnchor**]), ainsi que la relation *before* ∨ *meets* entre la dernière séquence de plans du présentateur (s_3 [*AnchorShot**]) et le dernier plan (s_4 [*LastShot*]) permet de traiter les cas où les génériques de début et de fin sont constitués soit d'un seul plan (France 2 et TF1) soit de plusieurs plans (France 3 et Canal+).

Les figures ci-dessous illustrent les découpages des quatre journaux avec présentateur du corpus AIM. Les plans sont figurés par des rectangles qui se situent sur diverses lignes selon la classe à laquelle ils appartiennent. La disposition des rectangles sur l'axe horizontal et leur longueur sont proportionnelles à la date de début et à la durée des plans dans le journal. Les journaux n'ayant pas tous la même durée, l'échelle n'est pas constante.

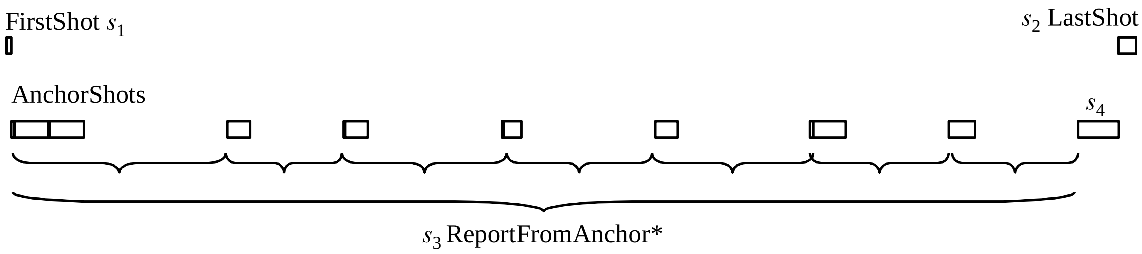


figure 94 : découpage du JT de France 2

[88] Le mot clé *reduce* indique que l'extension temporelle du canevas est l'union des extensions temporelles de ses composants. Voir également « Canevas bornés » page 83.

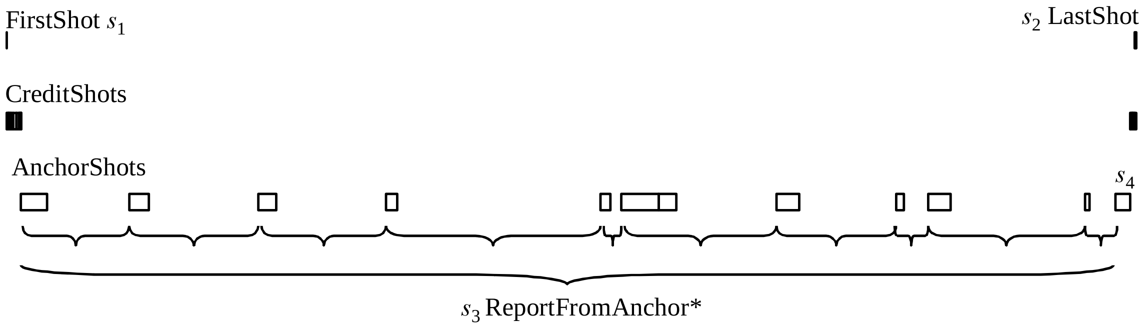


figure 95 : découpage du JT de France 3

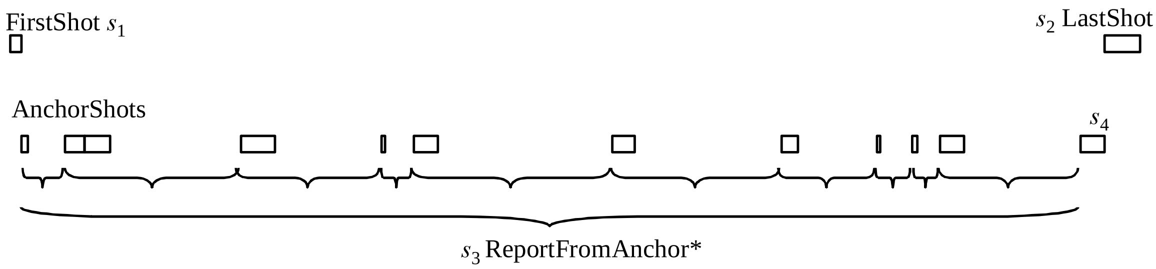


figure 96 : découpage du JT de TF1

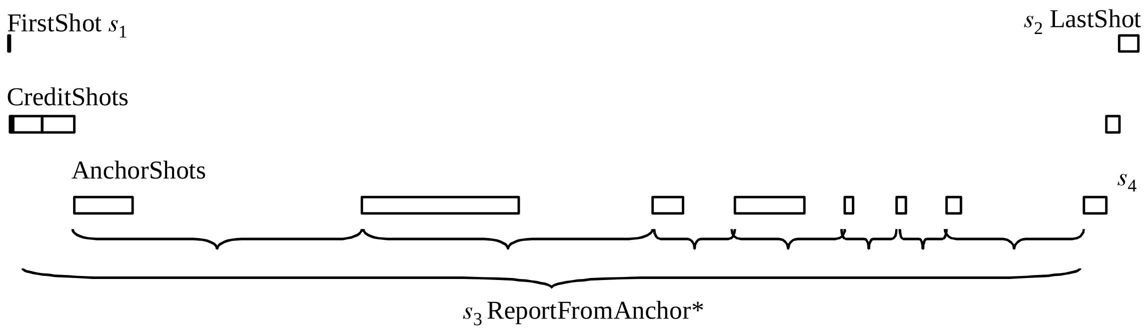


figure 97 : découpage du JT de Canal+

Les canevas utilisés pour segmenter un journal télévisé en reportages sont relativement simples et pourraient être remplacés par une grammaire régulière, qui fournirait les mêmes résultats de manière sans doute plus efficace. Les temps de calcul sont cependant tout à fait acceptables, puisqu'ils sont de l'ordre de deux à trois secondes pour les journaux les plus longs (« France 3 ») sur un ordinateur de bureau grand public (PC Pentium II à 400 Mhz).

L'algorithme de reconnaissance, implémenté par un ensemble de méthodes de filtrage de contraintes, étant complet, les segmentations obtenues correspondent aux résultats attendus. Pour chaque journal, une et une seule instance de *ReportFromAnchorSeq* est reconnue ; pour les canevas définis, le découpage est ainsi optimum.

La question se pose alors de la pertinence du découpage en reportages d'après les plans du présentateur. Pour le journal de Canal+, de format court, ainsi que pour le journal de France 2, de format plus long, la réponse est sans aucun doute positive : ces journaux sont constitués d'une alternance stricte plateaux-reportages. Pour le journal de TF1, la réponse est plus mitigée. En particulier, les titres présentés en début de journal sont considérés comme un seul reportage. Le journal de France 3, quant à lui, contient une section de « brèves », reportages courts tout en image qui ne sont pas interrompus par des plans du présentateur. Nous reviendrons sur cette question plus loin[89].

2.6.2 Reportages des journaux sans présentateur

Les journaux sans présentateurs (Arte et M6) sont de format court et sont constitués d'une série de reportages annoncés par des panneaux graphiques. Pendant l'apparition de ces panneaux, une musique caractéristique (jingle) est diffusée. Nous nous intéressons ici à la détection des reportages à partir des occurrences de ces jingles (voir figure 98). Détecter les reportages à partir des panneaux graphiques se fait d'une manière semblable à la détection des reportages à partir des plans de présentateur, comme cela a été illustré précédemment. Les jingles sont entendus « pendant » les écrans graphiques, la relation temporelle entre le jingle et l'écran graphique étant imprécise. En réalité, les jingles peuvent commencer ou finir légèrement avant ou après le début ou la fin de l'écran graphique. Un écran graphique pouvant être composé de plusieurs plans, les dispositions temporelles relatives des écrans graphiques et des jingles sont multiples. Sur le corpus AIM, la segmentation effectuée n'a retenu qu'un seul plan par écran. Nous avons également testé les canevas de reportage que nous présentons ici sur des segmentations où les écrans graphiques ont été arbitrairement sur-segmentés.

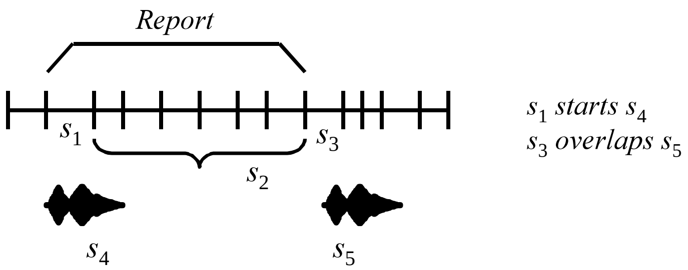


figure 98 : exemple d'un reportage reconnu d'après deux occurrences de jingles

Les figures ci-dessous illustrent les occurrences des plans graphiques et des jingles audio des éditions du corpus AIM des journaux de « M6 » et de « Arte ». Les plans graphiques et les jingles ne coïncident pas temporellement de manière exacte, ce qui est difficile à voir sur les figures étant donné leur échelle.

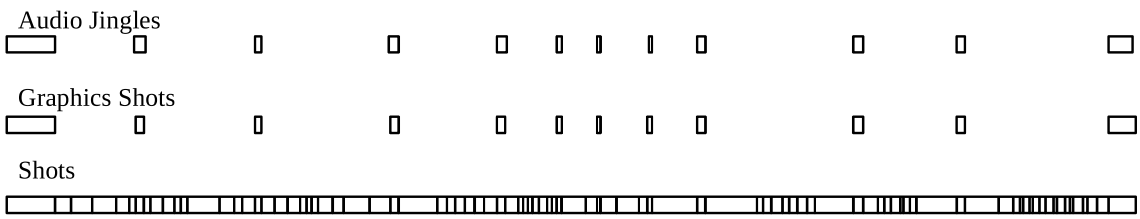


figure 99 : jingles et plans graphiques du JT de M6

[89] Voir « Etude de Soir 3 » page 156.

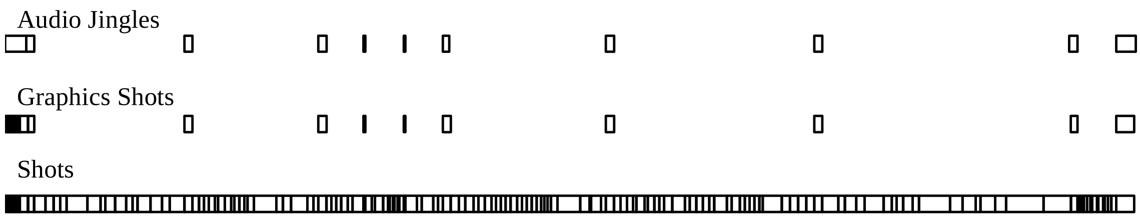


figure 100 : jingles et plans graphiques du JT de Arte

Nous définissons dans un premier temps un reportage comme une séquence de plans apparaissant « entre » deux jingles successifs, ce qui entraîne la définition du canevas défini ci-dessous (voir figure 101). Dans la définition, les relations temporelles sont définies de manière à ce que le reportage contienne temporellement les plans montrés « pendant » le premier jingle et s'arrête « juste avant » le second.

NaiveReport
nodes
  s_1 [*Shot**]
  s_2 [*Jingle*]
  s_3 [*Jingle*]
constraints:
  s_1 {*is-started is-overlapped*} s_2
  s_1 {*meets overlaps*} s_3
  s_1 *equals* this
  no *Jingle* between s_2 s_3

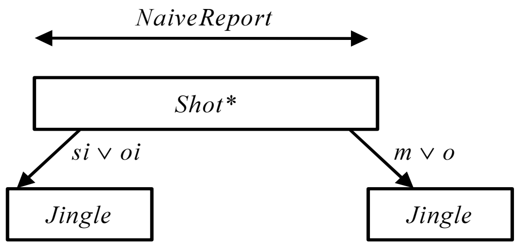


figure 101 : définition naïve d'un reportage

Considérons maintenant les plans et les jingles illustrés par la figure 102. Sur la figure, les plans pouvant être reconnus comme le premier plan de la séquence itérée associée à s_1 sont indiqués par une étoile et les plans pouvant être reconnus comme le dernier plan de cette séquence sont indiqués par un losange. La reconnaissance du canevas *NaiveReport* appliquée à cet exemple donne donc neuf solutions. Ces neuf solutions correspondent au *même* reportage, ou, plus exactement, on obtient neuf reportages différents là où l'on n'en voudrait qu'un.

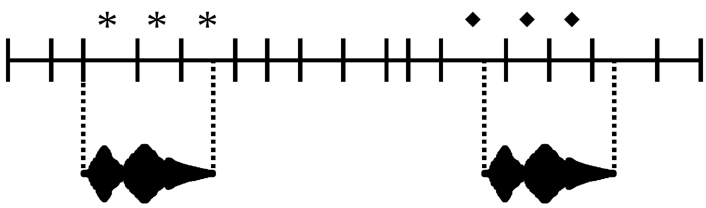


figure 102 : occurrences multiples de *NaiveReport*

Pour pallier ce problème, nous définissons un canevas où les plans situés aux bornes du reportage sont explicites, ce qui nous conduit à la définition suivante :

ReportFromJingle

nodes

s_1 [*Shot*]

s_2 [*Shot**]

s_3 [*Shot*]

s_4 [*Jingle*]

s_5 [*Jingle*]

constraints

s_1 *meets* s_2

s_2 *meets* s_3

s_1 {*overlaps starts is-started is-during*} s_4

s_3 {*overlaps starts is-started is-during*} s_5

s_1 *starts* this

s_2 *finishes* this

no *Jingle* between s4 s5

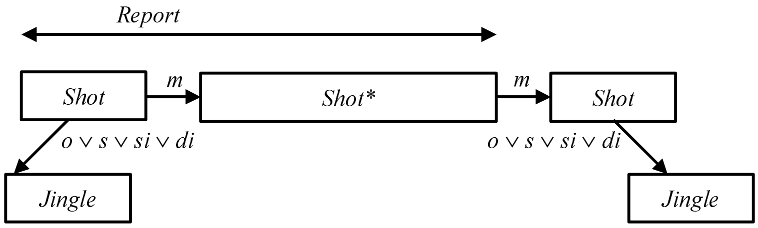


figure 103 : définition d'un reportage d'après deux jingles audio

Ce nouveau canevas résout le problème de reconnaissances multiples d'un « même » reportage. Il pose cependant un nouveau problème : pour qu'un reportage soit reconnu, il faut qu'il y ait au moins un plan du reportage qui soit après – *after*, au sens strict de Allen – le premier jingle et qu'il y ait au moins un plan avant – *before* – le second. Si le reportage est constitué d'au moins deux plans, ces conditions sont nécessairement vérifiées. Si le reportage n'est constitué que d'un seul plan, il se peut qu'elles ne le soient pas. Ce cas ne s'observe qu'une seule fois sur la durée des deux journaux « M6 » et « Arte » (voir figure 104). Il s'agit d'un reportage de M6 sur des résultats de Formule 1. La bande image est constituée d'un unique panneau illustrant le résultat de la course. D'autre part, les deux jingles débordent temporellement de quelques échantillons les plans graphiques inter-reportages. Ce cas est heureusement relativement rare.

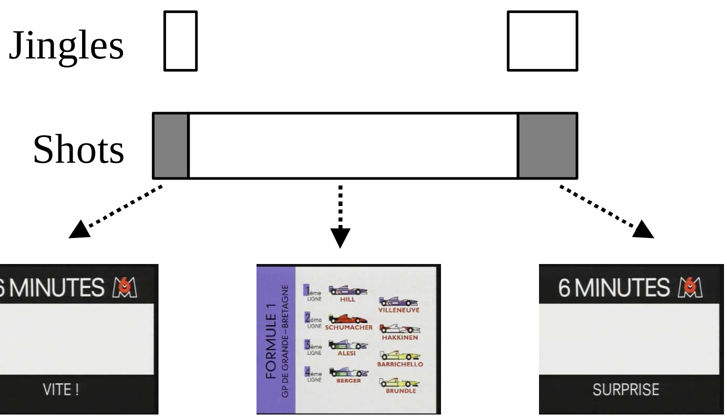
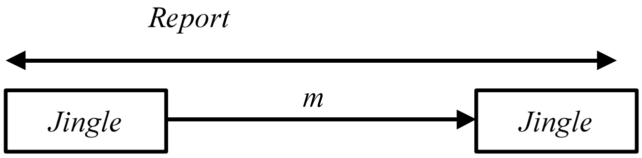


figure 104 : M6, reportage non reconnu

Finalement, en n'utilisant qu'une segmentation en plans et une détection des jingles audio, on peut reconnaître avec le même canevas les neuf reportages du journal d'Arte et dix des onze reportages du journal de M6. Notons que le canevas employé permet de délimiter les bornes temporelles d'un reportage par deux plans, et non par deux jingles audio, ce qui est plus pertinent en termes télévisuels. Il aurait été possible de délimiter temporellement un reportage en fonction des bornes temporelles des jingles, ce qui d'une part ne fait une différence que de quelques échantillons sonores pour les exemples étudiés, et ce qui d'autre part n'induit pas d'oubli. On aurait alors la définition suivante :

Report
nodes
  s_1 [*Jingle*]
  s_2 [*Jingle*]
constraints
  s_1 *before* s_2
  s_1 *starts* this
  this *meets* s_2
  no *Jingle* between s_1 s_2



Cette définition ne fait pas apparaître les plans, ce qui peut présenter un inconvénient majeur lors de l'exploitation des résultats – stockage en base de données des segments constitutifs du reportage, par exemple.

2.6.3 Spécialisation des reportages

Les reportages obtenus peuvent être spécialisés, par exemple en notant les reportages qui contiennent des plans de lieu ou des plans de personnes nommées, ces plans ayant été repérés par la présence de zones de l'image correspondant à un visage ou à une zone de texte[90]. Ainsi, un reportage contenant un plan d'une personne nommée peut se définir par le canevas suivant :

ReportWithNamedPerson
nodes
  s_1 [*Report*]
  s_2 [*NamedPersonShot*]
constraints
  s_1 {*starts finishes during equals*} s_2
  s_1 *equals* this

La relation temporelle {*starts* ∨ *finishes* ∨ *during* ∨ *equals*}indique que le plan d'une personne nommée peut-être à n'importe quel endroit de la séquence et qu'il peut également couvrir l'intégralité de la séquence, ce qui pourrait être le cas par exemple avec une séquence composée d'un seul plan (correspondant à l'étranger commentant un événement, par exemple). Détecter ainsi certains plans spécifiques lors d'un reportage peut être fort utile, d'une part pour effectuer un découpage structurel du reportage, d'autre part pour choisir des images caractéristiques représentatives du reportage.

[90] Voir « Classification de plans avec CLASSIC »page 42.

2.7 Etude de Soir 3

« Soir 3 » est journal télévisé de la fin de soirée de la chaîne « France 3 ». Le corpus AIM en comporte une édition et le corpus DiVAN en comporte quatre. Nous présentons tout d'abord une particularité de ce journal, la séquence de brèves, étudiée sur l'édition du corpus AIM, puis une comparaison entre deux méthodes de définition des reportages, étudiées sur l'ensemble des cinq éditions.

2.7.1 Séquences de brèves de Soir 3

Le journal « Soir 3 » est schématiquement constitué d'une alternance plateaux/reportages, comme les journaux de TF1 et France 2 – ainsi que celui de Canal +, plus court – étudiés précédemment. Cependant, la définition appliquée à ces derniers est trop générale. En effet, l'une des séquences de reportage est une séquence de *brèves*, c'est-à-dire une séquence de reportages courts commentés en voix *off*. Ces reportages sont séparés par un écran graphique faisait apparaître une succession rapide d'images (figure 105).

figure 105 : Soir 3, extraits d'écrans graphiques séparant les séquences de brèves

Pendant les plans graphiques, un jingle audio caractéristique est entendu. La séquence dans son ensemble est donc comparable à une édition complète des journaux de « M6 » et « Arte ». Enfin, le titre de la brève est donné en début de reportage par une incrustation de texte en haut à gauche, entourée de traits horizontaux (figure 106).

figure 106 : Soir 3, incrustation du titre des brèves

Une brève est définie comme un reportage délimité par deux jingles audio, en appliquant le canevas *ReportFromJingle*[91]. La première et la dernière brève ayant des définitions spécifiques. La première brève est délimitée par un ou plusieurs plans du présentateur et le premier

[91] Voir figure 103 page 154.

jingle, la dernière brève est délimitée par le dernier jingle audio et un plan du présentateur. Les définitions des canevas correspondants sont les suivantes :

FirstF3BriefReport
nodes
s_1 [*AnchorShot*]
s_2 [*Shot*]*
s_3 [*Shot*]
s_4 [*Jingle*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_3 {*overlaps starts is-started is-during*} s_4
s_1 *starts* this
s_2 *finishes* this
no *Jingle* between s_1 s_3
no *AnchorShot* between s_1 s_3

LastF3BriefReport
nodes
s_1 [*Shot*]
s_2 [*Shot**]
s_3 [*AnchorShot*]
s_4 [*Jingle*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_1 {*overlaps starts is-started is-during*} s_4
s_1 *starts* this
s_2 *finishes* this
no *Jingle* between s_1 s_3
no *AnchorShot* between s_1 s_3

La séquence de brèves est définie comme la succession d'une première brève (*FirstF3NewsInBrief*), d'une séquence itérée de brèves – c'est-à-dire de reportages reconnus d'après les jingles – (*ReportFromJingle**) et d'une dernière brève. Finalement, on définit la structure globale du journal « Soir 3 » comme la succession d'un générique de début (*BeginCreditShot**), d'une séquence itérée de reportages reconnus d'après les présentateurs (*ReportFromAnchor**)[92], d'une séquence de brèves (*F3BriefReportSequence*), d'une seconde séquence itérée de reportages (*ReportFromAnchor**) et d'un générique de fin (*EndCrediShot**). Afin de délimiter les bornes du journal, le premier et le dernier plan sont marqués (*FirstShot* et *LastShot*).

[92] Voir figure 92 page 148.

F3BriefReportSequence
nodes
s_1 [*FirstF3BriefReport*]
s_2 [*ReportFromJingle**]
s_3 [*LastF3BriefReport*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_1 *starts* this
s_3 *finishes* this

F3Newscast
nodes
s_1 [*FirstShot*]
s_2 [*BeginCreditsShot**]
s_3 [*ReportFromAnchor**]
s_4 [*F3BriefReportSequence*]
s_5 [*ReportFromAnchor**]
s_6 [*AnchorShot**]
s_7 [*EndCreditsShot**]
s_8 [*LastShot*]
constraints
s_1 *meets* s_2
s_2 *meets* s_3
s_3 *meets* s_4
s_4 *meets* s_5
s_5 *meets* s_6
s_6 *meets* s_7
s_7 *meets* s_8
s_1 *starts* this
s_8 *finishes* this

On obtient ainsi un découpage complet du journal « Soir 3 » à partir des résultats de trois algorithmes d'analyse : segmentation en plans, détection des plans de studio et détection des jingles audio indiquant les brèves. La reconnaissance du canevas *F3Newscast* sur l'édition de « Soir 3 » du corpus AIM, qui comporte 216 plans, dont 12 plans du présentateur, ainsi que 4 jingles audio de brèves, se fait en quelque 3 secondes sur un ordinateur de bureau (Pentium II à 400 Mhz). Le processus de reconnaissance donne une unique solution, partiellement illustrée par la figure 107.

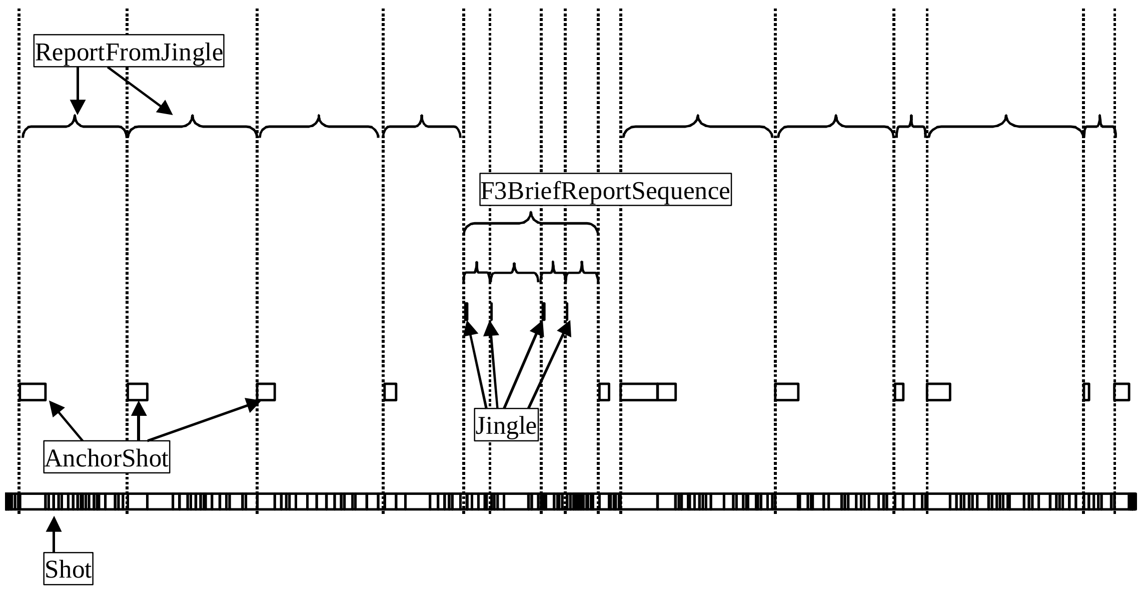


figure 107 : structure du « Soir 3 » du corpus AIM

La reconnaissance du canevas représentant le découpage complet d'une édition de « Soir 3 » en reportages et séquence de brèves donne les résultats escomptés et s'effectue en un temps qui est, comparé aux temps de calcul requis par les algorithmes d'analyse, tout à fait raisonnable.

2.7.2 Reconnaissance des reportages d'après les transitions graduelles

Les enchaînements plateau-reportage – ou reportage-plateau – sont généralement soulignés par une transition graduelle, fondu enchaîné ou volet. Les transitions progressives ont souvent été utilisées pour découper de manière automatique un journal télévisé en reportages (Demarty, 2000). Nous comparons dans cette expérimentation le découpage du journal « Soir 3 » en reportages effectué d'une part d'après les plans du présentateur, ce découpage servant de référence, et d'autre part d'après les transitions progressives.

Un reportage de référence est défini par le canevas *RealReport* comme une séquence de plans entre deux plans du présentateur. Une séquence de plans ayant lieu entre deux transitions est définie par le canevas *ReportFromTrans*. Les définitions de ces canevas sont données ci-dessous. Comme nous ne cherchons pas ici à exploiter les instances de ces canevas, par exemple pour les stocker en base de données, nous ne faisons pas apparaître explicitement dans les définitions les plans intermédiaires, mais seulement les événements – plans du présentateur ou transitions – marquant les limites des instances de canevas.

RealReport
nodes
    s_1 [*AnchorShot*]
    s_2 [*AnchorShot*]
constraints
    s_1 before s_2
    no *AnchorShot* between s_1 s_2

ReportFromTransitions
nodes
    s_1 [*GradualTransition*]
    s_2 [*GradualTransition*]
constraints
    s_1 before s_2
    no *GradualTransition* between s_1 s_2

Les figures suivantes illustrent les plans du présentateur et les transitions progressives des quatre éditions de « Soir 3 » du corpus DiVAN et de l'édition du corpus AIM.

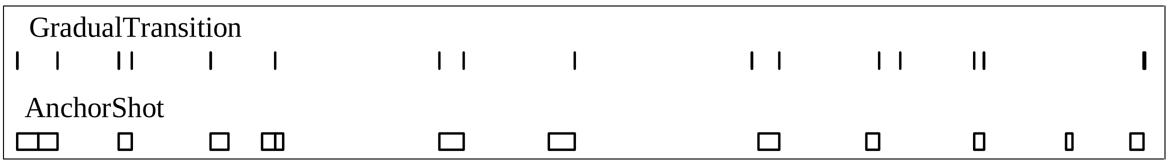


figure 108 : Soir 3 (13 juillet 1995), transitions progressives et plans du présentateur

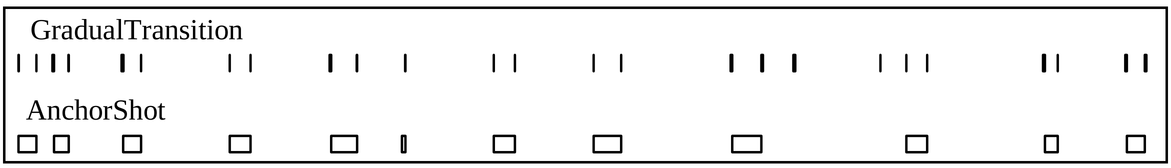


figure 109 : Soir 3 (16 juillet 1995), transitions progressives et plans du présentateur

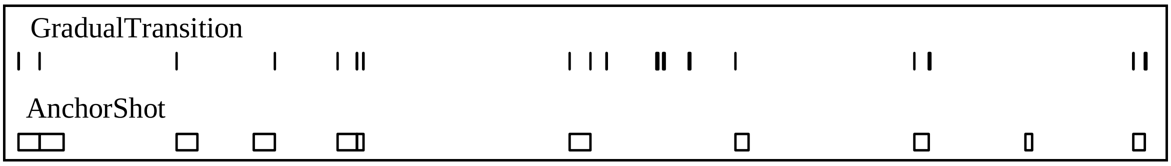


figure 110 : Soir 3 (17 février 1995), transitions progressives et plans du présentateur

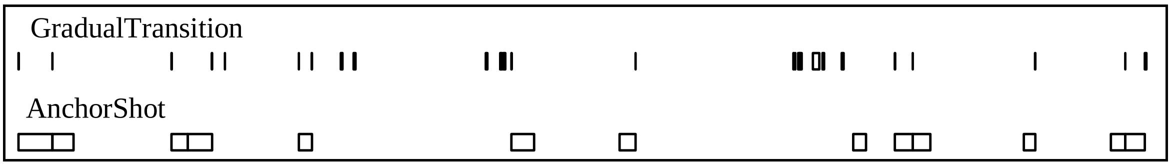


figure 111 : Soir 3 (20 juin 1995), transitions progressives et plans du présentateur

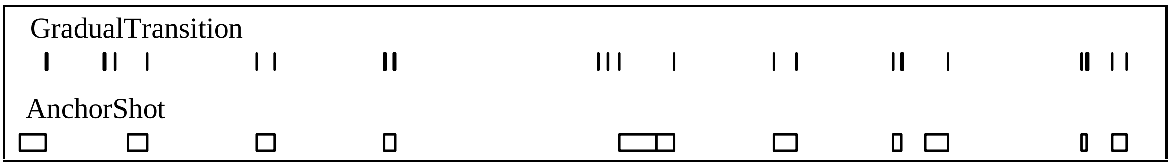


figure 112 : Soir 3 (édition d'AIM), transitions progressives et plans du présentateur

On constate sur les figures ci-dessus que, si les transitions graduelles soulignent effectivement fréquemment les enchaînements plateau-reportage, cette convention n'est pas systématiquement respectée. D'autre part, des transitions graduelles peuvent apparaître en cours de reportage. La segmentation en reportages selon les transitions progressives comporte ainsi plusieurs défauts :

- Les transitions progressives apparaissant en cours de reportage entraînent une sur-segmentation.

- Les séquences de studio – un ou plusieurs plans du présentateur – ne commençant ni ne finissant par une transition progressive ne sont pas repérées comme limites de reportage. Dans ce cas, les deux reportages situés de part et d'autre de cette séquence sont considérés comme un seul reportage.

- Les séquences de studio qui à la fois commencent et finissent par une transition progressive sont reconnus comme des reportages.

La comparaison des deux modes de reconnaissance des reportages appliquées aux cinq éditions de « Soir 3 » est résumée dans le tableau 8.

Edition	Référence (reportages d'après les plans du présentateur)	Reportages (d'après les transitions progressives)	Séquences plateau reconnues comme reportage	Séquences de studio sans transition progressive
13/07/95	10	15	5	1
16/07/95	11	24	11	0
17/02/95	8	17	6	1
20/06/95	8	26	5	0
AIM	8	20	7	0

tableau 8 : comparaison de deux modes de reconnaissance des reportages

Notons que le nombre de séquences plateau reconnues comme reportages a pu être établi automatiquement en appliquant un canevas définissant « une séquence de plans du présentateur précédée et suivie par deux transitions progressives » et que le nombre de séquences de studio sans transition progressive a pu être établi en appliquant un canevas définissant une séquence de plans du présentateur pendant (*during*) une instance de *ReportFomTransition*.

On constate que la sur-segmentation est non négligeable, puisqu'on obtient en moyenne environ deux fois trop de reportages. Ceci est à opposer au faible nombre d'oublis, puisque sur la totalité des cinq journaux, seules deux séquence de studio ne sont ni introduites ni conclues par une transition progressive. Finalement, la segmentation en séquences de plans séparées par deux transitions progressives consécutives est utile pour l'annotation dans la mesure où elle représente une amélioration par rapport à une simple segmentation en plans, à la condition de négliger les séquences de plateau oubliées.

3 Conclusion

Nous avons présenté dans ce chapitre plusieurs expérimentations menées dans le cadre du projet DiVAN. Ces expérimentations concernent essentiellement le découpage de journaux télévisés – journaux du corpus DiVAN et journaux du corpus AIM. Les journaux télévisés ont en effet fréquemment une structure figée et formalisable, ce qui explique qu'ils constituent souvent un terrain d'investigation privilégié. Nous avons en particulier montré comment définir et reconnaître les reportages dans différents types de journaux télévisés, en définissant par exemple un reportage comme une séquence de plans apparaissant entre deux plans de studio consécutifs pour un type de journal, ou comme une séquence de plans bornée par deux jingles audio caractéristiques pour un autre type.

D'autres types d'émission peuvent bénéficier de l'expérience acquise. Ainsi, par exemple, le corpus DiVAN contient plusieurs éditions de l'émission « Top A », émissions de variété des années 70 où l'on voit entre autres Serge Gainsbourg, Juliette Gréco ou Jacques

Dutronc. Ces émissions en public sans présentateur sont constituées d'une succession de numéros, chanson, sketch ou danse. A la fin de chaque numéro, le public applaudit. Un simple découpage en séquences de plans séparées par des applaudissements permet d'isoler les numéros. Le canevas permettant ce découpage est identique au canevas permettant d'isoler les reportages d'un journal télévisé d'après les occurrences d'un jingle audio.

Plus généralement, nous pensons qu'un petit nombre de canevas sont génériques et peuvent être adaptés à des situations différentes. C'est le cas, par exemple, du canevas isolant une séquence de plans commençant et finissant par des plans de type particulier – *e.g.* plans du présentateur –, ou commençant et finissant – dans un sens temporel légèrement différent – par des événements sonores de type particulier – *e.g.* un jingle.

D'autre part, le formalisme que nous proposons est suffisamment général pour s'appliquer à d'autres domaines. Nous avons ainsi effectué quelques expérimentations dans le domaine musical, en cherchant à reconnaître dans des fichiers MIDI des séquences rythmiques caractéristiques. Nous avons pour cela constitué une taxinomie des sons percussifs de la banque d'échantillons « General Midi ». La figure 113 illustre quelques-uns des 128 sons percussifs (en italique) ainsi que les concepts abstraits qui les subsument. La taxinomie comporte également d'autres concepts indiquant par exemple les temps forts et les temps faibles des mesures à deux, trois et quatre temps. On définit ensuite des canevas représentant des structures rythmiques caractéristiques, un exemple simple étant un rythme de rock avec un son de type *Bass* sur les temps forts (1 et 3) et un son de type *Snare* sur les temps faibles (2 et 4) d'une mesure à quatre temps. On cherche alors à reconnaître les occurrences de ce canevas dans un fichier MIDI converti en graphe d'observations. Sans que nous ayons eu le loisir de pousser très loin ces expérimentations, il nous a semblé qu'il était possible de cette manière de reconnaître automatiquement certains styles musicaux – rock ou salsa, par exemple.

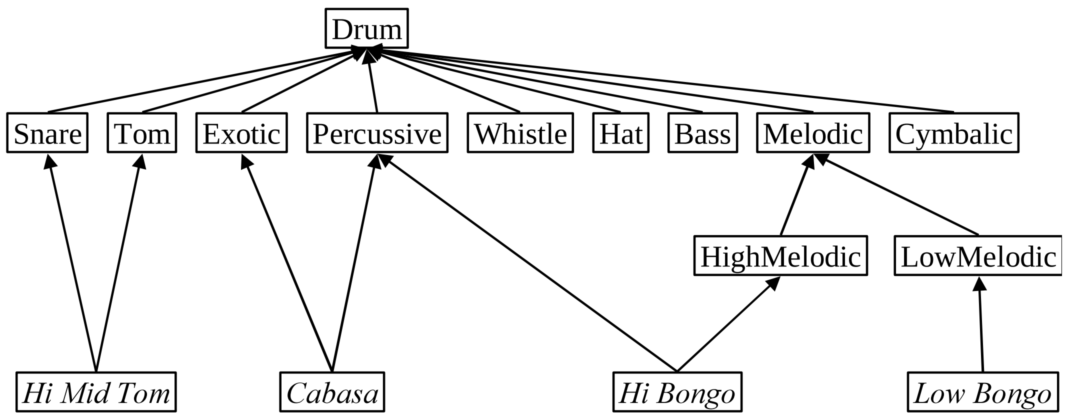


figure 113 : extrait d'une taxinomie de sons percussifs

Conclusion et perspectives

Dans le contexte de l'indexation de documents audiovisuels, et plus particulièrement de l'indexation de *collections* d'émissions de télévision partageant des caractéristiques communes, telles que les décors, les habillages de l'image – logos, incrustations de texte –, l'habillage audio – jingles – ou la structure temporelle, nous avons présenté une méthode de représentation et de classification de séquences audiovisuelles caractéristiques d'une collection. Nous exploitons pour cela les informations extraites de *fiches collection*, fiches détaillées élaborées à l'Inathèque de France décrivant les caractéristiques de l'ensemble des émissions d'une collection.

La description de ces classes de séquences, que nous appelons *canevas*, s'appuie sur une taxinomie préalable d'événements filmiques, tels que les plans, les transitions graduelles ou les jingles audio. Nous nous sommes particulièrement intéressés aux classes d'événements pouvant être extraits automatiquement du flux audiovisuel par des algorithmes de traitement du signal vidéo ou audio. Nous avons proposé de représenter ces classes d'événements par une taxinomie de concepts dans un formalisme de logiques de descriptions ; ce type de formalisme, de par ses mécanismes de classification et d'instanciation reposant sur le calcul de la relation de subsomption entre descriptions – descriptions de concepts et descriptions d'individus –, permet d'une part une gestion intelligente des classes d'événements et d'autre part une classification fine des événements extraits du flux audiovisuel. Les composants d'un canevas sont associés à une classe d'événements, ou, récursivement à un autre canevas, la dimension temporelle étant représentée par un ensemble de contraintes de Allen imposées entre les composants. Nous avons ensuite défini un opérateur d'itération noté « * » désignant une séquence itérée, c'est-à-dire une séquence d'événements temporellement adjacents. Enfin, nous avons introduit dans le langage de définition des canevas des contraintes spécifiques impliquant une négation, contraintes qui permettent en particulier d'exprimer la succession temporelle d'événements non adjacents.

Nous avons formalisé le problème de la reconnaissance des occurrences d'un canevas comme un problème de satisfaction de contraintes (CSP) et nous avons proposé une méthode efficace de résolution de ce problème ainsi qu'une implémentation de cette méthode reposant sur un système de satisfaction de contraintes existant. Au cœur de cette méthode se trouvent une représentation compacte et implicite des séquences itérées ainsi que des mécanismes efficaces de filtrage des contraintes temporelles impliquant des variables itérées. Des expérimentations sur des cas réels ont montré que, en pratique, les temps de résolution sont tout à fait satisfaisants.

Le langage de description des canevas est une extension du langage de description des *plans* – au sens de la reconnaissance de plans – du système T-Rex. Alors que dans T-Rex le problème du calcul de la subsomption entre plans est résolu, nous avons vu que l'introduction de l'opérateur d'itération « * » rendait le problème du calcul de la subsomption entre canevas extrêmement difficile. Dès lors, deux voies de recherche se dégagent. La première consiste à trouver un compromis acceptable entre expressivité et complexité, de manière à maintenir calculable la subsomption. C'est une voie de recherche théorique difficile pour laquelle nous avons posé le problème à résoudre mais n'avons fait qu'esquisser une méthode de résolution. L'autre voie laisse de côté le problème de la subsomption et prend son parti de la perte des avantages qu'elle apporte, principalement en matière de modularité et de réutilisabilité. C'est cette approche plus pragmatique que nous avons choisie en continuant à enrichir le langage de description des canevas avec un nouveau type de contraintes. Cette voie nous semble pouvoir être poursuivie, dans une optique de recherche appliquée, le but devenant de construire un langage de description des canevas fondé sur le formalisme des CSP, les caractéristiques des séquences à reconnaître étant exprimées par des contraintes qui soient à la fois pertinentes et pour lesquelles des mécanismes de résolution efficaces puissent être implémentés. Certaines contraintes peuvent très facilement être ajoutées au langage, comme des contraintes de cardinalité sur la longueur des séquences itérées – « au moins trois plans consécutifs du présentateur ». Un autre type de contraintes peut s'avérer très utile, qui permettrait d'exprimer des relations de similarité entre les éléments d'une séquence, comme des relations de similarité de couleurs entre images – « une séquence itérée de plans de couleurs similaires », ou « deux plans consécutifs dont la distance des images représentatives dans l'espace RGB des couleurs est inférieure à 0.2 ». Ce type de contraintes demande un calcul numérique qui peut être coûteux. Plus généralement, on peut chercher à intégrer le déclenchement d'algorithmes d'analyse au processus de résolution. Un système de satisfaction de contraintes tel que BackJava permet de spécifier une heuristique du choix de la contrainte à filtrer, heuristique qui peut être définie de façon à ne choisir les contraintes faisant appel à un algorithme coûteux qu'en fin de résolution, après que le filtrage des autres contraintes. Cette méthode constituerait une manière de piloter les algorithmes d'analyse.

Dans le domaine de la reconnaissance de plans, il est courant de distinguer les observations qui *réalisent* un plan des observations qui sont *compatibles* avec un plan. Nous nous sommes principalement intéressés aux plans – canevas – réalisés. La notion de compatibilité est importante dans la mesure où elle permet de raisonner sur des observations incomplètes. En toute généralité, un ensemble d'observations est compatible avec un canevas si la spécialisation des observations ou l'ajout de nouvelles observations permet de réaliser – ou reconnaître – ce canevas. Le calcul de compatibilité des canevas tels que nous les avons définis est difficile. En effet, si l'on cherche à reconnaître un canevas représentant une séquence itérée de segments de classe A – plans du présentateur, par exemple –, n'importe quel couple de segments (a_1, a_2) de classe A, tel que <a_1 *before* a_2> est compatible avec ce canevas, puisqu'il suffit d'ajouter une observation a_3 telle que <a_1 *meets* a_2> et <a_2 *meets* a_3> pour que le canevas soit réalisé (voir figure 114). D'autre part, le fait que des observations puissent être ajoutées contredit l'hypothèse de monde clos que nous avons introduite pour exprimer des contraintes faisant intervenir une négation. Il conviendrait par conséquent de prendre des précautions supplémentaires, par exemple en posant des hypothèses de monde clos plus restreintes, telles que « aucun plan ne sera ajouté » ou « tous les plans du présentateur ont été reconnus ».

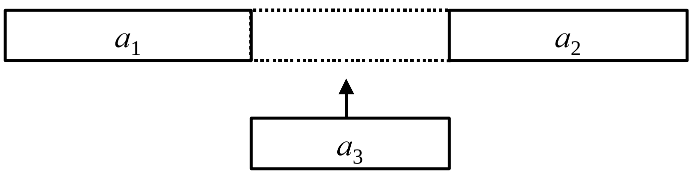


figure 114 : observations compatibles avec un canevas

Une hypothèse forte a été faite quant à la perfection des résultats provenant des algorithmes d'analyse. Cette hypothèse n'est pas réaliste. Introduire la notion de compatibilité est une manière de répondre partiellement à la question de savoir comment prendre en compte les erreurs potentielles des outils d'analyse, en considérant par exemple que les oublis d'un algorithme de classification sont des informations manquantes. Il est beaucoup plus difficile en revanche de prendre en compte les fausses alarmes de ce même algorithme.

Enfin se pose la question de la manière dont sont obtenus les canevas. Nous avons déjà évoqué la possibilité de procéder par spécialisation de canevas généraux. Ainsi, le canevas définissant une séquence de plans délimitée par deux occurrences d'un type particulier de segments sonores peut-être spécialisé, pour un journal télévisé en *reportages* si le segment sonore est un jingle, et pour un magazine de variété en *chansons* s'il s'agit d'applaudissements. Il nous semble que dans des cas aussi simples que celui-ci, il doit être possible, en restreignant peut-être la logique temporelle sous-jacente à un petit nombre de relations, d'induire automatiquement les canevas à partir d'une base d'exemples annotés. Le problème posé est alors celui de l'apprentissage automatique des canevas.

Bibliographie

Aguierre Smith, T. G., Davenport, G. (1992), « The Stratification System: A Design Environment for Random Access Video », in proceedings of the ACM Workshop on Networking and Operating Systems Support for Digital Audio and Video (NOSSDAV), La Jolla, California, USA, November 12-13, Venkat Rangan, P. (éd.), pp. 262-271.

Ahanger, G., Little, T. D. C. (1996), « A survey of technologies for parsing and indexing digital video », Journal of Visual Communication and Image Representation, 7(1), pp. 28-43.

Aigrain, P., Joly, P., Longueville, V. (1997), « Medium Knowledge-Based Macro-Segmentation of Video into Sequences », in Intelligent Multimedia Information Retrieval, Maybury, M. (éd.), AAAI Press, MIT Press, pp. 159-173.

Allen, J. F. (1983), « Maintaining knowledge about temporal intervals », Communications of the ACM, 26, pp. 832-843.

Allen, J. F. (1984), « Towards a general theory of action and time », Artificial Intelligence, 23(2), pp. 123-154.

Allen, J. F., Ferguson, G. (1994), « Actions and events in Interval Temporal Logic », Journal of Logic and Computation, 4(5), pp. 531-579.

Andert, E. P. (1992), « Automated Knowledge-Base Validation », in proceedings of the AAAI Workshop on Verification and Validation of Expert Systems, July 1992, pp. 122-127.

Aoki, H., Shimotsuji, S., Hori, O. (1996), « A shot classification method of selecting effective key frames for video browsing », in proceedings of the ACM Multimedia 1996, Boston, MA, USA, November 18-22, pp. 1-10.

Arman, F., Hsu, A., Chiu, M.-Y. (1993), « Feature management for large video databases », in proceedings of the Storage and Retrieval for Image and Video Databases (SPIE'93), San Jose, CA, USA, 31 January - 5 February, Niblack, W. (éd.), pp. 2-12.

Artale, A., Franconi, E. (1994), « A Computational Account for Description Logic of Time and Action », in proceedings of the 4th International Conference on Principles in Knowledge Representation and Reasoning (KR94), Bonn, Germany, May 24-27, pp. 3-14.

Artale, A., Franconi, E. (1999), « Temporal Description Logics », in Handbook of Time and Temporal Reasoning in Artificial Intelligence (to appear), Vila, L., van Beek, P., Fisher, M., Galton, A., Morris, R. (éd.), MIT Press.

Artale, A., Franconi, E., Guarino, N. (1996a), « Open Problems with Part-Whole Relations », in proceedings of the 1996 Workshop on Description Logics - DL'96, Cambridge, MA, USA, November 2-4, pp. 70-73.

Artale, A., Franconi, E., Guarino, N., Pazzi, L. (1996b), « Part-Whole Relations in Object-Centered Systems: An Overview », Data & Knowledge Engineering, 20, pp. 347-383.

Auffret, G., Bachimont, B. (1999), « Audiovisual Cultural Heritage: from TV and radio archiving to hypermedia publishing », in proceedings of the European Conference on Research and Advanced Technology for Digital Libraries, Paris, France, Abiteboul, S., Vercoustre, A.-M. (éd.), Springer-Verlag, pp. 58-75.

Auffret, G., Carrive, J., Chevet, O., Dechilly, T., Ronfard, R., Bachimont, B. (1999), « Audiovisual-based hypermedia authoring: using structured representations for efficient manipulation of AV documents », in proceedings of the ACM conference on Hypertext and Hypermedia 99, february 99, Darmstadt, Germany, February, ACM Press, pp. 169-178.

Bessière, C. (1994), « Arc Consistency and Arc Consistency Again », Artificial Intelligence, 65, pp. 179-190.

Bessière, C., Freuder, E., Régin, J.-C. (1995), « Using Inference to Reduce Arc Consistency Computation », in proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI-95), Montréal, Canada.

Borgida, A. (1992), « From type systems to knowledge representation: natural semantics specifications for description logics », International Journal of Intelligent and Cooperative Systems, 1(1), pp. 93-126.

Borgida, A., Brachman, R. J., McGuiness, D. L., Resnick, L. A. (1989), « CLASSIC: A Structural Data Model for Objects », in proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Porland, Oregon, USA, May 31 - June 2, Clifford, J., Lindsay, B. G., Maier, D. (éd.), pp. 59-67.

Bourdon, J. (1997), Le direct : Une politique de la voix ou la télévision comme promesse inaccomplie, Réseaux (Magazine), 81, pp. 61-78.

Bouthemy, P., Garcia, C., Ronfard, R., Tziritas, G., Veneau, E., Zugaj, D. (1999), « Scene segmentation and image feature extraction for video indexing and retrieval », in proceedings of the Third International Conference on Visual Information Systems (VISUAL'99), Amsterdam, June 1999, Huijsmans, D. P., Smeulders, A. W. M. (éd.), Springer Press, pp. 245-252.

Brachman, R. J., Schmolze, J.G. (1985), « An overview of the KL-ONE knowledge representation system », Cognitive Science, 9(2), pp. 171-216.

Brachman, R. J., McGuiness, D. L., Patel-Schneider, P. F., Resnick, L. A. (1991), « Living with CLASSIC: When and how to use a KL-ONE like language », in Principles of Semantic Networks, Sowa, J. (éd.), San Mateo, CA, USA, Morgan Kaufmann, pp. 401-456.

Brunelli, R., Mich, O., Modena, C. M. (1999), « A Survey on the Automatic Indexing of Video Data », Journal of Visual and Image Representation, 10, pp. 78-112.

Carrer, M., Ligresti, L., Ahanger, G., Little, T. D. C. (1997), « An Annotation Engine for Supporting Video Database Population », Multimedia Tools and Appications, 5(3), pp. 233-258.

Carrive, J., Pachet, F., Ronfard, R. (1998), « Using Description Logics for Indexing Audiovisual Documents », in proceedings of the 1998 Workshop on Description Logics (DL'98), Trento, Italy, June 2-5, Franconi, E., De Giacomo, G., MacGregor, R. M., Nutt, W., Welty, C. A. (éd.).

Carrive, J., Pachet, F., Ronfard, R. (2000a), « Clavis: a temporal reasoning system for classification of audiovisual sequences », in proceedings of the Content-Based Multimedia Access (RIAO'2000), Paris, France, April 12-14, pp. 1400-1415.

Carrive, J., Pachet, F., Ronfard, R. (2000b), « Logiques de descriptions pour l'analyse structurelle de film », in Ingéniérie des Connaissances, évolutions récentes et nouveaux défis, Charlet, J., Zacklad, M., Kassel, G., Bourigault, D. (éd.), Eyrolles, pp. 423-438.

Carrive, J., Roy, P., Pachet, F., Ronfard, R. (2000c), « A language for Audiovisual Template Recognition », in proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming (to appear), Singapore, September 18-22, 2000.

Celeux, G., Diday, E., Govaert, G., Lechevallier, Y., Ralambondrainy, H. (1989), Classification automatique des données, Dunod.

Chang, Y.-L., Zeng, W., Kamel, I., Alonso, R. (1996), « Integrated Image and Speech Analysis for Content-Based Video Indexing », in proceedings of the 1996 International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan, pp. 306-313.

Cohen, P. R., Perrault, C. R. (1986), « Elements of a Plan-based Theory of Speech Acts », Readings in Natural Language Processing, pp. 423-439.

Demarty, C.-M. (2000), Segmentation et Structuration d'un Document Vidéo pour la Caractérisation et l'Indexation de son Contenu Sémantique, Ecole Nationale Supérieure des Mines de Paris.

Devanbu, P. T., Litman, D. (1996), « Taxonomic Plan Reasoning », Artificial Intelligence, 84, pp. 1-35.

Dojat, M., Ramaux, N., Fontaine, D. (1998), « Scenario recognition for temporal reasoning in medical domains », Artificial Intelligence in Medicine, 14, pp. 139-155.

Dousson, C. (1994), Suivi d'évolutions et reconnaissance de chroniques, Thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, Université Paul Sabatier.

Faudemay, P., Chen, L., Montacié, C., Caraty, M.-J., Tu, X. (1997), « Segmentation multi-canaux de vidéos en séquences », in proceedings of the Troisièmes Journées d'Etudes et d'Echanges Compresion et Représentation des Signaux Audiovisuels CORESA'97, Issy-les-Moulineaux, France, March 26-27.

Fayad, E. M., Schmidt, D. C. (1997), « Object-Oriented Application Frameworks », Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, 40(10).

Ferman, A. M., Tekalp, A. M., Mehrota, R. (1998), « Effective Content Representation for Video », in proceedings of the IEEE International Conference on Image Processing (ICIP'98), Chicago, IL, USA, October 1998.

Fontaine, D. (1996), « Une approche par graphes pour la reconnaissance de scénarios temporels », Revue d'Intelligence Artificielle, 10(4), pp. 439-468.

Fountains, A., Hall, W., Heath, I., Davis, H. (1990), « Microcosm: an open model with dynamic linking », in proceedings of the Hypertext: Concepts, Systems and Applications, INRIA, France, November 1990, Rizk, A., Streitz, N. A., André, J. (éd.), pp. 198-311.

Garcia, C., Apostolidis, X. (2000), « Text Detection and Segmentation in Complex Color Images », in proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'2000), Istambul, Turkey, June 5-9.

Garcia, C., Tziritas, G. (1999), « Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis. », IEEE Transactions on Multimedia, 1(3), pp. 264-277.

Gochet, P., Gribomont, P. (1992), Logique. Méthodes pour l'informatique fondamentale, Hermès.

Gong, Y., Sim, L. T., Chuan, C. H., Zhang, H., Sakauchi, M. (1995), « Automatic parsing of tv soccer », in proceedings of the IEEE Conference on Multimedia Computing and Systems (ICMCS'95), pp. 167-174.

Hammoud, R., Kouam, D. G. (2000), « A mixed classification approach of shots for constructing scene structure for movie fims », in proceedings of the Irish Machine Vision and Image Processing Conference, Belfast, Northern Ireland, 31 August - 2 September.

Hampapur, A., Jain, R., Weymouth, T. (1994), « Digital Video Segmentation », in proceedings of the 1994 ACM Conference on Multimedia, San Francisco, California, USA, pp. 357-364.

Hanjalic, A., Lagendijk, R. L., Biemond, J. (1999), « Automatic Segmenting Movies into Logical Story Units », in proceedings of the Visual Information and and Information Systems, Third International Conference, VISUAL'99, LNCS 1614, Amsterdam, The Nederlands, June 1999, Huijsmans, D. P., Smeulers, A. W. M. (éd.), pp. 229-244.

Hauptmann, A. G., Witbrock, M. J. (1997), « Informedia: News-on-Demand Multimedia Information Acquistion and Retrieval », in Intelligent Multimedia Information Retrieval, Maybury, M. T. (éd.), AAAI Press, pp. 213-239.

Heinsohn, J., Kudenko, D., Nebel, B., Profitlich, H.-J. (1994), « An Empirical Analysis of Terminological Representation Systems », Artificial Intelligence, 68(2), pp. 367-397.

Hopcroft, J. E., Ullman, J. D. (1979), Introduction to Automata Theory, Languages and Computation, Addison Wesley.

Ide, I., Hamada, R., Tanaka, H., Shuichi, Sakai (1998a), « News Video Classification based on Semantic Attributes of Captions », in proceedings of the ACM International Multimedia Conference - Art-Demos-Technical Demos-Poster Papers, Sept. 1998, pp. 60-61.

Ide, I., Tanaka, H. (1998), « Automatic Semantic Analysis of Television News Captions », in proceedings of the Third International Workshop on Information Retrieval with Asian Languages, pp. 56-60.

Ide, I., Yamamoto, K., Tanaka, H. (1998b), « Automatic Video Indexing Based on Shot Classification », in proceedings of the First International Conference on Advanced Multimedia Content Processing (AMCP'98), Osaka, Japan, November, 9-11, Nishio, S., Kishino, F. (éd.), pp. 87-102.

Johnson, R. E., Foote, B. (1988), « Designing Reusable Classes », The Journal of Object-Oriented Programming, 1(2), pp. 22-35.

Jourdan, M., Layaïda, N., Roisin, C., Sabri-Ismaïl, L., Tardif, L. (1998), « Madeus, an Authoring Environment for Interactive Multimedia Documents », in proceedings of the Sixth ACM International Multimedia Conference, Bristol, UK, september 98, pp. 267-272.

Kasturi, R., Jain, R. (1991), « Dynamic Vision », Computer Vision: Principles.

Kautz, H. A. (1991), « A Formal Theory of Plan Recognition and its Implementation », in Reasoning about Plans, Allen, J. F., Kautz, H. A., Pelavin, R. N., Tenenberg, J. D. (éd.), Morgan Kaufman Plubishers, Inc.

Kender, J. R., Yeo, B.-L. (1998), « video scene segmentation via continuous video coherence », in proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, pp. 367-373.

Koller, D., Levy, A., Pfeffer, A. (1997), « P-CLASSIC: A tractable probabilistic description logic », in proceedings of the 14th National Conference on Artificial Intelligence, Rhode Island, pp. 390-397.

Kowalski, R. A., Sergot, M. (1986), « A logic-base calculus of events », New Generation Computing, 4(4), pp. 319-340.

Kurtonina, N. (1997), « Classifying Description Logics », in proceedings of the 1997 International Workshop on description Logics (DL'97), Gif-sur-Yvettes (France).

Lewis, P. H., Davis, H. C., Griffiths, S. R., Hall, W., Wilkins, R. J. (1996), « Media based navigation with generic links », in proceedings of the Seventh ACM Conference on Hypertext, Washington, DC, USA, March 16-20, pp. 215-223.

Li, W., Gauch, S., Gauch, J., Pus, K. M. (1996), « VISION: A Digital Video Library », in proceedings of the First ACM International Conference on Digital Libraries, Bethesda, MD, USA, 20 March - 3 April, pp. 19-27.

Lienhart, R., Pfeiffer, S., Effelsberg, W. (1997), « Video Abstracting », Communications of the ACM, 40(12), pp. 55-63.

Lienhart, R., Pfeiffer, S., Effelsberg, W. (1999), « Scene Determination based on Video and Audio Features », in proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), Florence, Italy, June 7-11, pp. 685-690.

MacCarthy, J., Hayes, P. (1969), « Some Philosophical Problems from the Standpoint of Artificial Intelligence », Machine Intelligence, 4, pp. 463-502.

MacGregor, R. (1994), « A Description Classifier for the Predicate Calculus », in proceedings of the AAAI'94, pp. 213-220.

Mackworth, A. (1977), « Consistency in Networks of Relations », Artificial Intelligence, 8(1), pp. 99-118.

Maybury, M., Merlino, A. (1997), « Multimedia Summaries of Broadcast News », in proceedings of the International Conference on Intelligent Information Systems (IIS'97), Grand Bahama Island, Bahamas, December 8-10.

Maybury, M., Merlino, A., Rayson, J. (1997), « Segmentation, Content Extraction and Visualization of Broadcast News Video using Multistream Analysis », in proceedings of the AAAI Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora, pp. 102-112.

McDermott (1982), « A temporal logic for reasoning about processes and plans », Cognitive Science, 6, pp. 101-155.

Meng, J., Juan, Y., Chang, S.-F. (1995), « Scene change detection in a mpeg compressed video sequence », in proceedings of the S&T/SPIE Symposium on Electronic Imaging: Science & Technology, San Jose, CA, USA, February, pp. 14-25.

Merlino, A., Morey, D., Maybury, M. (1997), « Broadcast News Navigation using Story Segmentation », in proceedings of the Fifth ACM International Multimedia Conference, Seattle, Washington, USA, pp. 381-389.

Metz, C. (1966), « La grande syntagmatique du film narratif », Communications n° 8 - Numéro spécial «L'analyse structurale du récit». — Numéro republié à Paris aux Éditions du Seuil en 1981, dans la collection «Points», no 129(8).

Mohr, R., Henderson, T. (1986), « Arc and Path Consistency Revisited », Artficial Intelligence, 28, pp. 225-233.

Napoli, A. (1998), « Une introduction aux logiques de descriptions », in Langages et Modèles à Objets, Etat des recherches et perspectives, Ducournau, R., Euzenat, J., Masini, G., Napoli, A. (éd.), Le Chesnay, INRIA, pp. 321-350.

Nebel, B. (1990a), « Reasoning and Revision in Hybrid Representation Systems », Lecture Notes in Artificial Intelligence, 422.

Nebel, B. (1990b), « Terminological Reasoning is Inherently Intractable », Artificial Intelligence, 43(1), pp. 235-249.

Nebel, B., Bückert, H. J. (1994), « Reasoning about Temporal Relations: a Maximal Tractable Subclass of Allen's Interval Algebra », in proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94), Seattle, Washington, USA, pp. 356-361.

Nebel, B., von Luck, K. (1988), « Hybrid Reasoning in BACK », in proceedings of the International Syposium on Methodologies for Intelligent Systems

(ISMIS'88), Turin, Italy, October 12-15, pp. 260-269.

Odell, J. J. (1994), « Six Different Kinds of Composition », Journal of Object-Oriented Programming, 5(8).

Poncin, P. (1997), « Audiovisuel : vers le tout numérique », Les dossiers de l'audiovisuel, 74.

Prié, Y., Mille, A., Pinon, J.-M. (1998), « Une approche de modélisation de documents audiovisuels en strates interconnectées par les annotations », in proceedings of the Ingéniérie des Connaissances IC'98, Pont-à-Mousson, France, pp. 143-152.

Ramaux, N. (1998), Supervision de systèmes dynamiques par reconnaissance de scénarios, Thèse de doctorat, Université de Technologie de Compiègnes.

Rector, A. L., Nowlan, W. A., consortium, t. G. (1999), « The GALEN Project », Computer Methods and Programs in Biomedicine, 45, pp. 75-78.

Roisin, C., Tran Thuong, T., Villard, L. (1999), « Integration of structured video in a multimedia authoring system », in proceedings of the Eurographics Multimedia 1999 Workshop, Milan, Italy, September 99, Springer Computer Science, pp. 133-142.

Ronfard, R. (1997), « Shot-level description and matching of video content », in proceedings of the SPIE 97IS&T/SPIE Storage and Retrieval for Image and Video Databases V, San Jose, CA, USA, pp. 70-78.

Ronfard, R., Garcia, C., Carrive, J. (2000), « Conceptual Indexing of Television Images Based on Face and Caption Sizes and Locations », in proceedings of the Fourth International Conference on Visual Information Systems, Lyon, France, 2-4 November.

Rosenthal, D. F., Okuno, H. G. (1998), Computational Auditory Scene Analysis, Lawrence Erlbaum Associates.

Rowe, L. A., Boreczky, J. S., Eads, C. A. (1994), « Indexes for user access to large video databases », in proceedings of the S&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology, San Jose, CA, USA, pp. 150-161.

Roy, P. (1999), Satisfaction de contraintes et programmation par objets, Laboratoire d'Informatique de Paris 6 (Lip6), Paris, Université Pierre & Marie Curie (Paris 6).

Roy, P., Liret, A., Pachet, F. (1999), « A Framework for Objected-Oriented Constraint Satisfaction Problem », in Implementing Application Frameworks: Object-Oriented Frameworks at work, Fayad, M., Schmidt, D., Johnson, R. (éd.), Wiley & Sons, pp. 359-401.

Rui, Y., Huang, T. S., Mehrota, S. (1999), « Constructing Table-of-Content for Videos », ACM Multimedia Systems Journal, Special Issue on Video Libraries, 7(5), pp. 359-368.

Ruiloba, R., Joly, P., Marchand-Maillet, S., Quénot, G. (1999), « Towards a Standard Protocol for the Evaluation of Video-to-Shots Segmentation Algorithms », in proceedings of the First European Workshop on Content-Based Multimedia Indexing (CBMI'99), Toulouse, France, pp. 41-48.

Sato, T., Kanade, T., Hughes, E., Smith, M., Satoh, S.-i. (1999), « Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption », ACM Multimedia Systems Special Issue on Video Librarie, 7(5), pp. 385-395.

Satoh, S. i., Nakamura, Y., Kanade, T. (1997), « Name-it: Naming and Detecting Faces in Video by the Integration of Image and Natural Language Processing », in proceedings of the Proceeding of 1997 International Joint Conference on Artificial Intelligence (IJCAI'97), Nagoya, Japan, August, pp. 1488-1493.

Schmid, C., Mohr, R. (1997), « Local Greyvalue Invariants for Image Retrieval », Transactions on Pattern Analysis and Machine Intelligence.

Schmiedel, A. (1990), « A Temporal Terminological Logic », in proceedings of the 1990 National Conference on Artificial Intelligence (AAAI'90), Boston, Massachusetts, USA, pp. 640-645.

Shoham, Y. (1987), « Nonmonotonic logics: Meaning and utility », in proceedings of the 1987 International Joint Conference on Artificial Intelligence (IJCAI'87), Milan, Italy, pp. 388-393.

Swanberg, D., Shu, C.-F., Jain, R. (1993), « Knowledge guided parsing in video databases », in proceedings of the Storage and Retrieval for Image and Video Databases (SPIE'93), San Jose, CA, USA, 31 January - 5 February, pp. 13-24.

Van Beek, P. (1992), « Reasoning about qualitative temporal information », Artificial Intelligence, 58, pp. 297-326.

Veneau, E., Ronfard, R., Bouthémy, P. (2000), « From Video Shot Clustering to Sequence Segmentation », in proceedings of the 15th International Conference on Pattern Recognition (ICPR'2000), Barcelona, Spain, September 2000.

Vila, L. (1994), « A Survey on Temporal Reasoning in Artificial Intelligence », AI Communications, 7, pp. 4-28.

Vilain, M., Kautz, H., Van Beek, P. (1989), « Constraint propagation algorithms for temporal reasoning: A revised report », Readings in Qualitative Reasoning about Physical Systems, pp. 373-381.

Watclar, H. D., Kanade, T., Smith, M. A., Stevens, S. M. (1996), « Intelligent Access to Digital Videos: Informedia Project », IEEE Computer, 29(5), pp. 46-52.

Weida, R. (1992a), Terminological Constraint Network Reasoning and its Applications to Plan Recognition, Columbia University, Report Technical Report CUCS-027-93.

Weida, R., Litman, D. (1992b), « Terminological Reasoning with Constraint Networks and an Application to Plan Recognition », in proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, Massachussetts, USA, pp. 282-293.

Weida, R. (1995), « Knowledge Representation for Plan Recognition », in proceedings of the IJCAI'95 Workshop on the Next Generation of Plan Recognition Systems, Montréal, Québec, Canada.

Weida, R., Mays, E., Dionne, R., Laker, M., White, B., Liang, C., Oles, F. J. (1996), « The K-Rep System Architecture », in proceedings of the 1996 International Workshop on Description Logics (DL'96), Cambridge, MA, USA, Padgham, L., Franconi, E., Gehrke, M., McGuinness, D., Patel-Schneider, P. (éd.), AAAI Press.

Winston, M. E., Chaffin, R., Herrmann, D. (1987), « A Taxonomy of Part-Whole Relations », Cognitive Science, 11, pp. 417-444.

Yen, J., Neches, R., McGregor, R. (1991), « CLASP: Integrating Term Subsumption Systems and Production Systems », IEEE Transactions on Knowledge and Data Ingineering, 3(1), pp. 25-32.

Yeo, B.-L. (1996), Efficient Processing of compressed images and video, PhD Thesis, Princeton University.

Yeo, B.-L., Liu, B. (1995), « Rapid Scene Analysis on Compressed Video », IEEE Transactions on Circuits and Systems for Video Technology, 5(6), pp. 533-544.

Yeung, M., Liu, B. (1995), « Efficient matching and clustering of video shots », in proceedings of the 1995 IEEE International Conference on Image Processing, pp. 338-341.

Yeung, M., Yeo, B.-L. (1997), « Video Content Characterization and Compaction for Digital Library Applications », in proceedings of the IS&T/SPIE Storage and Retrieval for Image and Video Databases V, San Jose, CA, USA, pp. 45-48.

Yeung, M., Yeo, B.-L., Liu, B. (1996), « Extracting story units from long programs for video browsing and navigation », in proceedings of the 1996 International Conference on Multimedia Computing and Systems (ICMCS'96), Hiroshima, Japan, pp. 296-305.

Zabih, R., Miller, J., Mai, K. (1995), « A Feature-Based Algorithm for Detecting and Classifying Scene Breaks », in proceedings of the ACM Conference on Multimedia, San Francisco, California, USA, pp. 189-200.

Zhang, H., Kankanhalli, A., Smoliar, S. W. (1993), « Automatic Partitioning of full-motion video », ACM Multimedia Systems, 1(1).

Zhang, H. J., Low, C. Y., Smoliar, S. W., Wu, J. H. (1997), « Video parsing, retrieval and browsing - An integrated and content-based solution », in Intelligent multimedia information retrieval, Mayburry, M. T. (éd.), AAAI Press - MIT Press.

Zhong, D., Zhang, H. J., Chang, S.-F. (1996), « Clustering methods for video browsing and annotation », in proceedings of the Storage and Retrieval for Image and Video Databases IV (SPIE'96), pp. 239-246.

ANNEXES

Annexe A : exemple de fiche collection

Titre propre	F2 le journal 20H00
Chaîne de diffusion	France 2
Canal	Réseau 2
Date de diffusion	07.12.1992
Jour	lundi
Statut de diffusion	Première diffusion
Heure de diffusion	20:00:00
Heure de fin de diffusion	20:40:00
Durée	00:40:00
Genre	Journal télévisé
Type de description	Collection d'émissions composites
Médiamétrie	Information, journal national
Générique	PRE,Masure Bruno; PRE,Bilalian Daniel
Chapeau	Deuxième édition quotidienne de l'actualité nationale et internationale de la journée, le journal de 20 heures est diffusé en direct et dure environ 40 minutes. Il est réalisé en studio et s'organise en une succession de plateaux - reportages. Le plateau est animé en alternance du vendredi au vendredi par Bruno Masure et Daniel Bilalian.
Résumé	Le présentateur annonce lui même le sommaire du journal. Au cours de ce sommaire, il peut y avoir un plan de l'invité présent sur le plateau ainsi qu'une séquence sur un fait brûlant de l'actualité. Bruno Masure, s'adresse aux téléspectateurs par un "Mesdames, messieurs Bonsoir" et lance son premier sujet. Les sujets vont de l'actualité nationale à l'internationale, pour reprendre sur le national et terminer en principe sur les nouvelles du sport et des arts et spectacles. Le présentateur nomme les journalistes ayant préparé le sujet annoncé au moment du plateau lancement. En fonction de l'actualité, il peut y avoir une chronique animée par un journaliste spécialisé de la rédaction. En fin d'édition le journaliste annonce la météo, le programme de la soirée, l'invité des Quatre vérités du lendemain. Bruno Masure souhaite généralement les anniversaires et fêtes d'une ou plusieurs personnalités. Particularités : - La durée générale de l'édition varie en fonction de l'actualité. - L'édition du jeudi dure environ 20 minutes, car elle introduit l'émission d'actualité "Invité spécial". La transition entre les deux se fait sur le même principe que le dispositif utilisé pour le duplex. - L'édition du dimanche comporte une chronique sport tenue par un journaliste de Stade 2"

Dispositifs	GENERIQUE

.- Début

Le générique est organisé en deux temps. Images de synthèse de la planète en mouvement : ton bleu foncé avec incrustation de rouge en alternance avec le titre "journal" qui s'affiche deux fois en flash sur un fond beu ciel. Arrivée plein écan du présentateur cadré serré en plan poitrine, zoom avant, plan rapproché et lancement du journal.

La musique court tout au long du générique jusqu'à la fin de l'annonce des titres avec une baisse d'intensité du volume sur les titres. Absence de musique lorsqu'il y a une séquence d'images relative à un fait marquant de l'actualité.

.- Fin

Fondu sur la dernière image arrêtée et départ du générique dans un mouvement ascendant par panneaux sur la musique de début. Le générique de fin reprend le principe du générique de début en deux temps avec l'image de la planète, puis le fond bleu ciel et le logo de la chaîne.

DISPOSITIFS PLATEAU

.- Dispositifs techniques

Trois caméras au sol : deux faisant face au journaliste, une autre derrière lui située à sa droite ou à sa gauche. Une quatrième caméra prend les plans d'ensemble en plongées plus ou moins verticales.

.- Plateau journaliste

Deux valeurs de plan pour le présentateur : un plan poitrine et un plan rapproché

Vue d'ensemble en plongée pour le dispositif plateau

.- Plateau journaliste - invité

Mise en images caractérisée par l'alternance de plans d'ensemble, poitrine, rapproché entre le journaliste et l'invité. Les plans d'ensemble sont pris de biais avec en amorce le journaliste de dos.

.- Duplex

Mise en images caractériséepar l'usage du plan d'ensemble de biais avec en amorce le journaliste de dos faisant face à un grand écran. Transition rapide sur le plein cadre du journaliste qui assure le direct

.- Incrustations

* Affichage du nom du présentateur : il n'apparait qu'une fois dès l'apparition du journaliste plein cadre.

* Affichage des inscustations plateau sous forme d'affiche : quasi constantes, elles servent d'introduction au sujet en donnant le type et la tonalité du sujet. Disposées en haut à droite, elles sont rectangulaire en position verticale. Dispositif général : fond blanc avec une barre rouge en haut et un encadré noir en bas en général. Présence de texte en bas et en haut avec au centre une image d'illustration.

.- Décor et mobilier

* Le mobilier n'apparaît que sur les vues d'ensemble du plateau. La table est ronde avec en son centre des moniteurs. Les micros sont répartis selon trois points cardinaux de part et d'autre du journaliste.
Sur les plans poitrine on peut apercevoir les micros et les notes du journaliste.
* Le décor est construit en deux demi cercle positionnés l'un derrière l'autre. Deux colonnes de moniteurs encadrent une grande vitre aux reflets bleutés sur la gauche suivi par un panneau clair en allant sur la droite. Le décor du plateau est surexposé.
Identification de deux toiles de fond en fonction de l'axe du cadrage :
Un panneau sur fond bleu ciel : le journaliste est alors pris en plan rapproché
Un panneau mixte fond bleu, moniteurs à droite : le journaliste est pris en plan poitrine
A partir de février 1997, les moniteurs du décors sont disposés de manière irrégulière.
.- Présentateur
L'habillement du présentateur est de facture classique. Costume, cravate aux couleurs foncées, souvent dans des tons de bleus pour Bruno Masure.
Bruno Masure fait parfois quelques jeux de mots ou notes d'humour lorsque l'actualité le permet.
DISPOSITIFS ELEMENTS
..- Reportage
* Incrustations :
Affichage du logo du journal uniquement sur les reportages : symbole de la planète en mouvement avec le 2 en rouge situé en bas et à droite de l'écran.
* Mention du nom du/des journalistes, commentateurs, sur un bandeau bleu transparent en bas de l'écran, vers la fin du sujet.
* Mention des lieux : en bas à doite juxtaposé au logo sans bandeau
* Mention des noms et le statut des personnes interrogées : en bas de l'écran sur un bandeau bleu transparent.
Les reportages peuvent être illustrés par des écrans synthés (cartes, graphiques, maquettes...)
Les sujets peuvent être illustrés par des interviews.
Le journaliste peut être présent à l'image en fin de reportage.
CONSTRUCTION GENERALE
.- Les changements de valeurs de plan et de cadrage sont multiples durant toute l'édition. De façon générale le rythme de l'édition est assez rapide. Les changements d'axes sont visibles à l'écran.
.- L'enchaînement sujet - reportage se fait sur un rythme rapide par un fondu, suivi parfois par une vue d'ensemble du

plateau avant de reprendre le journaliste en plan poitrine puis zoom avant plan rapproché ou bien en plan poitrine, ou directement en plan rapproché.
.- A partir de février 1997, les enchaînements se font par un volet.

Annexe B : A.E.D.I, un exemple

L'exemple de fichier A.E.D.I contient un extrait du résultat d'une segmentation en plans et en transitions progressives.

```
!!
!! Header for experiments with clavis
!!

!aedi,1.0,1.0,UTF8
:aedi: clavis_experiments
profile: [CLAVIS];
:timespace: timespaceID
document: [aim1mb05]
docname: [Arte - Huit et demi]
timebase: internal;

!!
!! Definition of types, properties and models
!!

:typedef: (kf /ina/keyframe)
basetype: string;

:propdef: classifiedAs
type: string;

:propdef: keyframe
type: kf;

:modeldef: clavis_event
keyframe: zeroormore
classifiedAs: zeroormore;

!!
!! Root layer
!!

:layer: root
overlapping: true
sorted: false
;

!!
!! Shots and transitions
!!

!! [...]

:event: shot128
time: 0:06:39:64800
duration: 0:00:02:50400
model: clavis_event
```

```
context: #root
classifiedAs: [SHOT-EVENT]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010024.jpg]
;

:event: shot129
time: 0:06:42:25200
duration: 0:00:01:10800
model: clavis_event
context: #root
classifiedAs: [SHOT-EVENT]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010070.jpg]
;

:event: tr35
time: 0:06:43:32400
duration: 0:00:00:07200
model: clavis_event
context: #root
classifiedAs: [DISSOLVE]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010084.jpg]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010085.jpg]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010086.jpg]
;

:event: shot130
time: 0:06:43:36000
duration: 0:00:01:72000
model: clavis_event
context: #root
classifiedAs: [SHOT-EVENT]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010107.jpg]
;

:event: tr36
time: 0:06:45:14400
duration: 0:00:00:07200
model: clavis_event
context: #root
classifiedAs: [DISSOLVE]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010129.jpg]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010130.jpg]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010131.jpg]
;

:event: shot131
time: 0:06:45:18000
duration: 0:00:00:75600
model: clavis_event
context: #root
classifiedAs: [SHOT-EVENT]
keyframe: [file:/aim/manual/aim1mb04/KEYFRAMES/aim1mb04-0010140.jpg]
;
```

Annexe C : A.E.D.I, la grammaire

Nous donnont ici grammaire du langage A.E.D.I sous forme BNF. Les termes entre « < » et « > » sont des terminaux. Les terminaux finissant de la forme <NNN_L> sont des littéraux ; ainsi, <STRING_L> indique une chaîne de caractères.

aediDocument	::=	documentDescription (typeModelPropertyDef ";")* (entityDef ";")*
documentDescription	::=	documentParameters (timeSpaceDef)*
documentParameters	::=	":aedi:" identifier ("profile:" <STRING_L>)? ("timescale:" number_l)? ";"
timeSpaceDef	::=	"timespace:" <ID_L> SegmentationTimeBase ";"
SegmentationTimeBase	::=	"document:" <STRING_L> ("docname:" <STRING_L>)? "timebase:" calendar_k ("projection:" projectionValues)? ("timezone:" <STRING_L>)?
projectionValues	::=	projectionTimeValue
	|	"(" projectionValuesAfterLPar
projectionValuesAfterLPar	::=	getProjectionTime ")"
	|	("(" getProjectionTime ")")+ ")"
getProjectionTime	::=	timeref_l projectionTimeValue timeref_l timeref_l
projectionTimeValue	::=	timeref_l
	|	parseADate
calendar_k	::=	"calendar"
	|	"internal"
typeModelPropertyDef	::=	":typedef:" typedefDcl
	|	":propdef:" propertydefDcl
	|	":modeldef:" modeldefDcl
typedefDcl	::=	identifier typeDefinition (infoField)?
propertydefDcl	::=	identifier ":type:" typeForProperty (infoField)?
typeDefinition	::=	"basetype:" typeRestriction
	|	"listof:" typeElement ("size:" <INTEGER_L>)?
	|	"struct:" "(" (typeElement)+ ")"
typeForProperty	::=	"integer"
	|	"float"
	|	"string"
	|	"boolean"
	|	"reference"

```
                               |   "timeref"
                               |   "timecode"
                               |   "any"
                               |   <ID_L>
typeRestriction               ::=  integerRestriction
                               |   floatRestriction
                               |   booleanRestriction
                               |   stringRestriction
                               |   referenceRestriction
                               |   timerefRestriction
                               |   timecodeRestriction
                               |   anyRestriction
                               |   userType ("scheme:" <STRING_L> )?
modeldefDcl                   ::=  identifier (propertyForModel )* (infoField )?
propertyForModel              ::=  <ID_L> "," propertyForModel2
propertyForModel2             ::=  propertyForModelWithDefault
                               |   propertyForModelWithoutDefault
propertyForModelWithDefault   ::=  "(" "required"litteralValue ")"
                               |   "(" "oneormore" litteralValue ")"
propertyForModelWithoutDefault ::= "optional"
                               |   "required"
                               |   "zeroormore"
                               |   "oneormore"
integerRestriction            ::=  "integer" ("min:" <INTEGER_L> | "max:"
                                   <INTEGER_L> | "step:" <INTEGER_L> |
                                   "scheme:" <STRING_L> | "values:" "("
                                   (<INTEGER_L> )+ ")" )*
floatRestriction              ::=  "float" ("min:" number_l | "max:" number_l |
                                   "step:" number_l | "scheme:" <STRING_L> |
                                   "values:" "(" (number_l )+ ")" )*
stringRestriction             ::=  "string" ("size:" <INTEGER_L> | "max:"
                                   <INTEGER_L> | "scheme:" <STRING_L> |
                                   "values:" "(" (<STRING_L> )+ ")" )*
timerefRestriction            ::=  "timeref:" ("min:" timeref_l | "max:" timeref_l |
                                   "scheme:" <STRING_L> )*
timecodeRestriction           ::=  "timecode:" ("scheme:" <STRING_L> )?
booleanRestriction            ::=  "boolean:" ("scheme:" <STRING_L> )?
anyRestriction                ::=  "any:" ("scheme:" <STRING_L> )?
referenceRestriction          ::=  "reference:"   (<REFERSTO_K>   <ID_L>   )?
```

```
                                ("scheme:" <STRING_L> )?
typeElement                 ::= basetype
                            |   userType
userType                    ::= <ID_L>
entityDef                   ::= entityDefParser (<MODEL_K> <ID_L> )?
                                (<CONTEXT_K> <REF_L> )? (property )* ";"
entityDefParser             ::= temporalEntityDefParser
                            |   unitDefParser
unitDefParser               ::= <UNIT_K> <ID_L>
temporalEntityDefParser     ::= temporalEntityType
temporalEntityType          ::= eventDef
                            |   layerDef
eventDef                    ::= "event:" <ID_L> "time:" unknownOrTimeref_l
                                "duration:" unknownOrTimeref_l ("overlapping:"
                                boolean_c )? ("sorted:" boolean_c )?
layerDef                    ::= "layer:" <ID_L> ("mintime:"
                                unknownOrTimeref_l )? ("maxtime:"
                                unknownOrTimeref_l )? ("overlapping:" boolean_c
                                )? ("sorted:" boolean_c )?
property                    ::= <ID_L> <COLON> litteralValue
boolean_c                   ::= "true"
                            |   "false"
number_l                    ::= <INTEGER_L>
                            |   <FLOAT_L>
basetype                    ::= "integer"
                            |   "float"
                            |   "boolean:"
                            |   "string"
                            |   "reference:"
                            |   "timeref:"
                            |   "any:"
infoField                   ::= "info:" <STRING_L>
identifier                  ::= <ID_L>
                            |   "(" localIdOrUndef <GLOBAL_ID_L> ")"
localIdOrUndef              ::= <ID_L>
                            |   "unknown"
litteralValue               ::= litteralBaseValue
                            |   "(" (litteralValue )+ ")"
```

```
litteralBaseValue        ::= number_l
                           | boolean_c
                           | <STRING_L>
                           | <REF_L>
                           | timeref_value
                           | "unknown"
parseADate               ::= <DATE_L> "T" (afterDate )?
                           | <ISO_TIME_L> (<COMMA_DIG3>)? afterTime
afterDate                ::= <ISO_TIME_L> afterTime
afterTime                ::= "Z"
                           | <GMTOFFSET>
timeref_value            ::= timeref_l
unknownOrTimeref_l       ::= timeref_l
                           | "unknown"
timeref_l                ::= <TIMEREF_L>
```

Annexe D : syntaxe de définition des canevas

Nous donnons ici la grammaire BNF définissant le langage de définition des canevas. Les termes entre « < » et « > » indiquent les symboles terminaux fournis par l'analyseur lexical.

template	::=	id "nodes" (nodeDef)+ "constraints" (constraintDef)* ("reduce")?
nodeDef	::=	<BASEID_L> id
constraintDef	::=	temporalConstraintDef
	|	noInstanceBetweenConstraintDef
noInstanceBetweenConstraintDef	::=	"no" id "between" <BASEID> <BASEID>
temporalConstraintDef	::=	rolePath allenRelation rolePath
rolePath	::=	<BASEID>
	|	"(" (<BASEID>)+ ")"
id	::=	<BASEID>
	|	<ID>
allenRelation	::=	basicAllenRelation
	|	"{" (basicAllenRelation)+ "}"
basicAllenRelation	::=	<EQUALS>
	|	<BEFORE>
	|	<AFTER>
	|	<MEETS>
	|	<ISMET>
	|	<DURING>
	|	<CONTAINS>
	|	<OVERLAPS>
	|	<ISOVERLAPPED>
	|	<STARTS>
	|	<ISSTARTED>
	|	<FINISHES>
	|	<ISFINISHED>