UNIVERSITE DE NICE-SOPHIA ANTIPOLIS – UFR SCIENCES

Ecole doctorale Sciences et Technologies
de l'Information et de la Communication

THESIS

presented for obtaining the grade of

DOCTOR OF SCIENCE
in COMPUTER SCIENCE

by

VAN-THINH VU

Temporal Scenario for Automatic Video Interpretation

Defense on the 14th October 2004, thesis committee:

Michel BARLAUD	President	Prof., Université de Nice-Sophia Antipolis, France
Bernd NEUMANN	Reviewer	Prof., Universität Hamburg, Germany
Catherine TESSIER	Reviewer	HDR, Onera-Cert, France
Jean-Phillippe BLANCHARD	Member	Dr., Crédit Agricole SA, France
Christophe DOUSSON	Member	Dr., FranceTélécom R&D, France
Monique THONNAT	Supervisor	HDR, DR, INRIA, France

This thesis is prepared in the ORION research team of INRIA Sophia-Antipolis

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS – UFR SCIENCES

Ecole doctorale Sciences et Technologies
de l'Information et de la Communication

THESE

présentée pour l'obtention du grade de

DOCTEUR EN SCIENCE

Discipline: INFORMATIQUE

par

VAN-THINH VU

Scénarios temporels pour l'interprétation automatique de séquences vidéos

Soutenance le 14 octobre 2004 devant le jury composé de :

Michel BARLAUD	Président	Prof., Université de Nice-Sophia Antipolis, France
Bernd NEUMANN	Rapporteur	Prof., Universität Hamburg, Germany
Catherine TESSIER	Rapporteur	HDR, Onera-Cert, France
Jean-Phillippe BLANCHARD	Examinateur	Dr., Crédit Agricole SA, France
Christophe DOUSSON	Examinateur	Dr., FranceTélécom R&D, France
Monique THONNAT	Directeur de thèse	HDR, DR, INRIA, France

Cette thèse est préparée au sein du projet ORION de l'INRIA Sophia-Antipolis

Abstract. *This thesis research focuses on the **recognition of temporal scenarios** for **Automatic Video Interpretation**: the goal of this work is to recognize in **real-time** the behaviors of individuals evolving in a scene depicted by video sequences which were captured by cameras. The recognition process takes the following as input: (1) human behavior (i.e., temporal scenario) models predefined by experts; (2) 3D geometric and semantic information of the observed environment; and (3) a stream of individuals tracked by a vision module.*

To deal with this issue, we have proposed a generic model of temporal scenarios and a description language to represent the knowledge of human behaviors. The representation of this knowledge needs to be clear, rich, intuitive and flexible. The proposed model of a temporal scenario M is composed of five components: (1) a set of physical object variables corresponding to the physical objects involved in M; (2) a set of temporal variables corresponding to the sub-scenarios composing M; (3) a set of forbidden variables corresponding to the scenarios that are not allowed to occur during the recognition of M; (4) a set of constraints (symbolic, logical, spatial and temporal constraints including Allen's interval algebra operators) involving these variables; and (5) a set of decisions corresponding to the tasks predefined by experts that are needed to be executed when M has been recognized.

*We have also proposed a temporal constraint resolution technique to recognize in **real-time** the temporal scenario models predefined by experts. The proposed algorithm is most of the time efficient for processing temporal constraints as well as for combining several actors defined within a given scenario M. By efficient we mean that the recognition process is linear with the number of sub-scenarios and with the number of physical object variables defined within M in most cases.*

To validate the proposed algorithm in terms of correctness, robustness and processing time with respect to scenario and scene properties (e.g., number of sub-scenarios, number of persons in the scene), we have tested the algorithm on several videos of different applications, in both on-line and off-line modes and also on simulated data.

*By the experiments conducted in metro surveillance and bank monitoring applications, the proposed scenario description language shows the capability to represent easily temporal scenarios corresponding to the human behaviors of interest in these applications. Moreover, the proposed temporal scenario recognition algorithm shows the capability to recognize in **real-time** (at least 10 frames/second) complex scenario models (up to 10 physical object variables and 10 sub-scenario variables per scenario) with complex video sequences (up to 240 persons/frame in the scene).*

Keywords: video surveillance, video interpretation, temporal constraint resolution, temporal scenario recognition, temporal scenario representation, human behavior recognition, human behavior representation, human behavior visualization.

Résumé. *Cette thèse traite de la **reconnaissance de scénarios temporels** pour l'**interprétation automatique de séquences vidéos** : l'objectif est de reconnaître à **cadence vidéo** les comportements d'individus évoluant dans des scènes décrites par des séquences vidéos (acquises par des caméras). Le processus de reconnaissance prend en entrée (1) les modèles de comportements humains (i.e. scénarios temporels) pré-définis par des experts, (2) les informations sémantiques et géometriques-3D de l'environnement observé et (3) les individus suivis par un module de vision.*

Pour résoudre ce problème, premièrement, nous avons proposé un modèle générique de scénarios temporels et un langage de description pour la représentation de connaissances décrivant des comportements humains. La représentation de ces connaissances doit être claire, riche, intuitive et flexible pour être compris par les experts du domaine d'application. Le modèle proposé d'un scénario temporel M se compose de cinq parties : (1) un ensemble de variables correspondant aux acteurs impliqués dans M, (2) un ensemble de variables temporelles correspondant aux sous-scénarios qui composent M, (3) un ensemble de variables interdites correspondant aux scénarios qui ne doivent pas être reconnus pendant la reconnaissance de M, (4) un ensemble de contraintes (symboliques, logiques, spatiales et contraintes temporelles comprenant les opérateurs de l'algèbre d'intervalles d'Allen) portant sur ces variables et (5) un ensemble de décisions correspondant aux tâches pré-définies par les experts pour être exécutées quand M est reconnu.

*Deuxièmement, nous avons proposé une technique originale de résolution de contraintes temporelles pour la reconnaissance à **cadence vidéo** de modèles de scénarios temporels pré-définis par des experts. En général, l'algorithme proposé est efficace car il propage les contraintes temporelles et combine seulement les objets physiques définis dans le scénario donné M. Par efficace, nous voulons dire que le processus de reconnaissance est linéaire en fonction du nombre de sous-scénarios et, dans quasiment tous les cas, en fonction du nombre d'objets physiques définis dans M.*

Pour valider l'algorithme proposé en termes d'exactitude, de robustesse et du temps de traitement en fonction de la complexité des scénarios et de la scène (e.g. nombre de sous-scénarios, nombre de personnes dans la scène), nous avons testé l'algorithme en appuyant sur un grand nombre de vidéos provenant de différentes applications sur des données simulées et également réelles en modes hors-ligne/en-ligne.

*Les expérimentations réalisées dans différentes applications montrent la capacité du langage de description de scénarios à représenter facilement les scénarios temporels correspondant aux comportements humains d'intérêt. De plus, ces expérimentations montrent également la capacité de l'algorithme proposé à reconnaître à **cadence vidéo** des modèles de scénarios sophistiqués (jusqu'à 10 acteurs et 10 sous-scénarios par scénario) dans des séquences vidéos complexes (jusqu'à 240 personnes/frame dans la scène).*

Mots clés : vidéo surveillance, interprétation de séquences vidéos, résolution de contraintes temporelles, reconnaissance de scénarios temporels, représentation de scénarios temporels, reconnaissance de comportements humains, représentation de comportements humains, visualisation de comportements humains.

Acknowledgements

It was my good fortune to have Doctor Monique Thonnat as my supervisor while working in France at the Institut National de Recherche en Informatique et en Automatique (INRIA). She taught me how to be a fruitful researcher, write good papers and, above all, have a good attitude. Her thorough scientific approach and unending quest for excellence have been inspirational during the years of my research. I only wish I had listened to her advice more often.

I would like to express my sincere thanks, appreciation, and gratitude to Doctor François Brémond of INRIA for his guidance, support, and encouragement, which made possible for me to successfully complete this research program. Without his understanding and inspiration, this thesis would never have existed.

I sincerely thank Professor Bernd Neumann of the Universität Hamburg, Germany and Doctor Catherine Tessier of Onera-Cert, France. They have accepted to be the two reviewers of my thesis and they gave me many good comments for my dissertation.

I sincerely thank Professor Michel Barlaud, Doctors Jean-Phillippe Blanchard and Christophe Dousson for their acceptance to be members of my thesis committee.

I am grateful to my dear colleague, Doctor Luong Chi Mai of Vietnam Institute of Information Technology (IOIT), for her advice and encouragement during my starting years of research. She is not only my colleague but also my teacher, my sister and my good friend.

I am grateful to my former supervisors and my dear professors, Professor Nguyen Ba Kim, Doctor Le Khac Thanh, Professor Nguyen Thi Tam Bac and Doctor Ho Cam Ha of Hanoi University of Pedagogy, Professor Lorne H. Bouchard of the Institut de la Francophonie pour l'Informatique (IFI) for their advice and encouragement throughout my studies.

I sincerely thank all my friends and colleagues who always supported me in times of need. I greatly appreciate my lab-mates (INRIA) and especially the ORION Research Group for their contributions in making a wonderful and supportive academic environment. The life would be nothing without friends. It is impossible to mention all of them here, but some had nevertheless a direct influence on this thesis work. Thank you, Chien, Le-Diep, Hong, Ludovic and the others, for sharing joys and sor-

rows with me. To my many Vietnamese friends from Nice, thanks for the good times over the past three years.

But life would be also difficult without financial support. I am deeply indebted to INRIA and the French project CASSIOPEE for granting me a scholarship, which made possible my study in France.

INRIA offered me the greatest research environment; it provided me a perfect computing environment, a well-known reputation among the research world, the spirit of hard-working students, and the chance to meet famous researchers all over the world. Among the friendly administrators, I owe a great deal to the International Student Section for the kind and constant assistance they provided. Without them, I would certainly have run into numerous troubles.

Finally, I have saved the best for the last. I wish to express my endless love and gratitude to my family, Mom, Dad, my sisters and my brothers for always being there when I needed and supporting me through all my school years. I am especially grateful to my parents for everything they taught me and for all the sacrifices they made in my upbringing. Most of all, I thank my wife –Ngoc Anh– and my son –Nguyen Anh–. They patiently looked after me, even though we were thousands of miles away. They never had a word of complaint when I was negligent. They comforted me when I was discouraged. I may never be able to repay them, but fortunately, I have a lifetime to try. I have written this thesis with their images in my heart.

September 2004

Sophia-Antipolis, France

Table of Contents

Table of Figures

Part I

Introduction

Chapter 1. Introduction

This chapter describes the research subject of this thesis and the plan of the manuscript.

1.1 Introduction

This thesis focuses on the **representation and recognition of temporal scenarios** for Automatic Video Interpretation. More precisely, we have two main objectives that consist in (1) conceiving a new method helping experts of different application domains to represent easily knowledge about interesting human behaviors and (2) proposing a novel method for recognizing in real-time (at video cadence) the interesting behaviors modeled by experts. In our research, we call an Automatic Video Interpretation System a system that is able to understand what is happening or what happened in the scene observed by different types of video cameras [Figure 1.1].

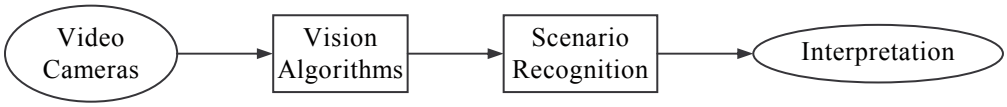


Figure 1.1. Automatic Video Interpretation: input videos are first analyzed by vision algorithms and then interpreted by a scenario recognition algorithm.

Working in the context of temporal scenario representation and recognition for Automatic Video Interpretation, the research of the thesis is related to the two following subjects:

(1) **scenario representation**: building an ontology structure to describe video events; based on the ontology, we propose a generic model of temporal scenarios to model interesting human behaviors; then based on the video event ontology and the generic temporal scenario model, we propose a new formalism to help experts of different application domains to represent easily their interesting behaviors (e.g. “Bank attack”, “Vandalism against a ticket machine”).

(2) **scenario recognition**: studying existing techniques for plan recognition, chronicle recognition and human activity recognition to build a new methodology (by taking advantages of the existing techniques) to recognize in real-time (at video cadence) temporal scenarios used for Automatic Video Interpretation.

For these purposes, we first present in this chapter our research motivations, then ORION's Automatic Video Interpretation Framework that we are working with. Finally, we present the principle of Temporal Scenario Recognition and the detailed objectives of the thesis.

1.2 Motivations

Our research motivations are multiple and consist mainly in conceiving a generic solution for **real-time recognition** (video cadence) of human activities for different Automatic Video Interpretation applications (e.g. bank monitoring, metro station surveillance) and building a **knowledge representation** method enabling experts describing clearly and easily knowledge of different application domains (e.g. different environments, widely interesting human activities). Up to our knowledge, there is no algorithm which can recognize sophisticated human activities in real-time.

In the following section, we explain our motivations through real-world applications. Through different application domains, we present various objectives and multiple types/levels of complexity of the Automatic Video Interpretation (e.g. different environments, different interesting human activities).

1.2.1 Application Domains

The objective of Automatic Video Interpretation is to understand behaviors of mobile objects evolving in the observed scene. In other words, our objective is to build a system that is capable to analyze automatically a scene depicted by sensors. The types of the sensors (e.g. cameras, infrared sensors) to be used depend on each application. The applications are multiple and concern different domains; for example, bank monitoring applications, metro monitoring applications, medical applications, road traffic monitoring,… or for general surveillance objectives, e.g. fire detection, natural environment pollution detection and security.

Human activity surveillance/monitoring: interesting behaviors of this domain concern principally persons evolving in the observed scene. The interesting behaviors can be related to activities/gestures of a few persons in the observed scene and globally of a group of persons.

- **Bank monitoring**: we are interested in the recognition of activities of persons in a bank agency. The most interesting mobile objects are employees, clients, robbers. The objective is to detect immediately abnormal situations in the observed agency (e.g. a bank attack situation) to generate alerts to enable security agents to take decisions. The interesting behaviors are not only related to individuals but also related to groups of individuals. There is currently the French project CASSIOPEE dealing with this application. The objective of this project is to build a system capable to detect behaviors pre-defined as risky (abnormal situations) from live camera networks and the a-priori knowledge of the observed agency. The a-priori knowledge (of abnormal situations and the observed agency) is supplied by security agents. Now, such a system is being built and tested in several bank agencies.
- **Hospital monitoring**: the objective of this application is to understand automatically behaviors of patients, nurses and doctors in a hospital. More precisely, the application is to detect normal and/or abnormal situations.

For example: to help a patient or to detect whether a patient has fallen, or to detect whether the behaviors of a nurse (or a group of nurses) are correct,… This application and the bank monitoring application are both indoor surveillance applications. The main difference between these two applications is the a-priori knowledge of behaviors.

Sports scene analysis: scenes observed in this application are largely different from scenes observed by the two applications shown above. Sports scenes are generally outdoor and take place in a large environment (e.g. football stadiums, racing ways). The focused sports are principally football [Choi *et al*., 1997], tennis, basketball and soccer [Intille & Bobick, 1995]. This application type aims at helping trainers to analyze the games to have adequate strategies of training/playing. The VITRA (VIsual TRAnslator) project focused on this application domain and built a system to generate verbal reports from sports videos [Herzog, 1995].

Remote conference: this application focuses on the recognition of gestures of person(s) evolving in the observed scene. The objective is to facilitate interaction between people (remotely or locally) using visual cues which are similar to those used in our everyday communications. The goal is to recognize (based on body movements) behaviors of person(s). For example, a behavior of waving the right hand above the head means an urgent wave [Howell & Buxton, 2002]. This application is also related to smart-room [Wolf & Ozer, 2001].

Traffic surveillance: interesting behaviors of this domain are related to two principal types of mobile objects, the vehicles and the persons evolving in the observed scene. Behavior detection is principally based on the speed and the position of mobile objects in the scene and the distance between them.

Metro monitoring: the European project ADVISOR aimed at designing an intelligent system capable to recognize human behaviors (e.g. vandalism against a ticket machine) in metro stations through video sequences acquired by cameras. The interesting behaviors are principally based on the position of people in the observed scene (e.g. 3D position, inside or outside an interesting zone) and the person-person and person-equipment distances. These behaviors can involve a person, a group of persons or a crowd. In this application, the observed scene is normally large and observed by several synchronized cameras.

Road traffic monitoring: interesting mobile objects are principally vehicles that are easier to be detected than the persons, because the shapes of vehicles are rigid. Moreover, interesting behaviors of this application domain are mainly related to vehicle movements that are limited by vehicle degrees of freedom. Thus they are less various than those of the previously presented application domains. The main applications are (1) accident detection, (2) intelligent intersections and (3) specific zone surveillance. Firstly, accident detection consists in predicting and detecting the situations on the road network that are considered as accidents to generate alerts corresponding to dangerous situations. For example, the Naos system is capable to describe verbally in natural language the observed road scene [Mohnhaupt & Neumann, 1991]. Although Naos has the ambitious objective to describe activities verbally, this system was only tested on simulated video sequences. Another system

called Epex is capable to interpret road scenes [Nagel, 1988]. Moreover, it was tested on real road videos. Secondly, intelligent intersection application: the objective is to optimize traffic by controlling automatically the traffic lights at a number of intersections. The system presented in [Sellam & Boulmakoul, 1994] has this objective and was used for real road traffic. Finally, specific zone surveillance: the objective is to detect abnormal behaviors in zones where there are frequently a number of vehicles, e.g. toll agents, gas stations [Nagel, 1991] or parking-lots [Tessier, 1997].

Apron monitoring: the principal objective is to survey aprons. Thus, interested mobile objects are aircraft, vehicles and persons. This application focuses on helping security agents of an airport to monitor the services and control the security around the aircraft. In other words, the application is dedicated to the security of aircraft at airports. The AVITRACK European project is now working on this objective. Figure 1.2 shows an example of the "loading" scenario used in this project. The shown images are taken at the Toulouse airport.

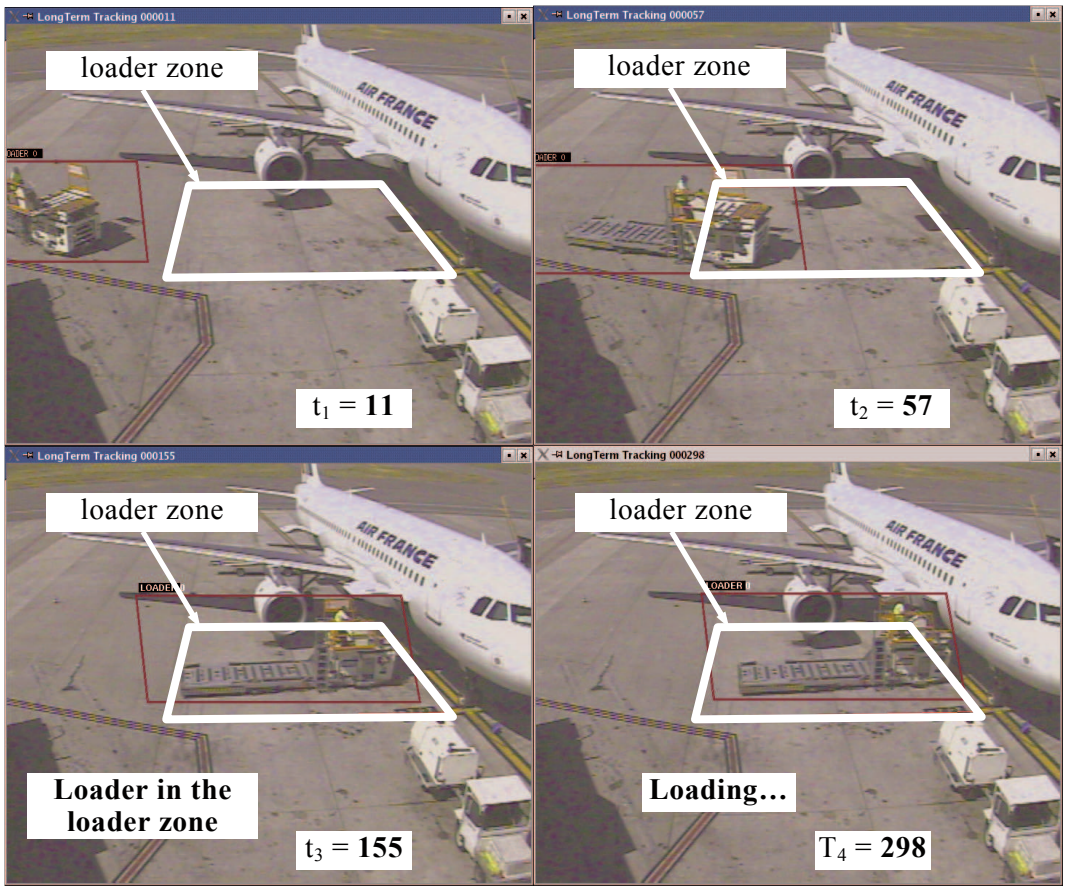


Figure 1.2. An example of loading an aircraft in the apron monitoring application. The loader vehicle is first outside the loader zone and then enters this zone. After entering the loader zone, the loader stays (longtime) inside this zone to start loading.

1.2.2 Focused Applications

There are currently four applications in our focus: bank monitoring, train surveillance, apron monitoring and metro station surveillance applications. Three of them are being studied through collaborations between the ORION research team of INRIA (France) and three projects: French project CASSIOPEE (for bank monitoring), European project SAMSIT (for train surveillance) and European project AVITRACK (for apron monitoring). The European project ADVISOR (for metro station surveillance) has just finished.

1.3 Automatic Video Interpretation

This section defines what Automatic Video Interpretation is and presents the main characteristics of an Automatic Video Interpretation System.

Definition 1.1: Automatic Video Interpretation is the process of understanding automatically what happens in scenes depicted by video sequences, and giving interpretations of these scenes.

An Automatic Video Interpretation System is an **autonomous** system capable to perform Automatic Video Interpretation. An Automatic Video Interpretation System takes video sequences as input and the output is the interpretation of these video sequences. Such a system is composed of two main modules: the first one for vision tasks and the second one for temporal scenario recognition. The vision module takes raw video sequences (acquired directly by video cameras) as input and executes a number of vision algorithms to detect and track all mobile objects evolving in the observed scene. Then, the scenario recognition module takes as input the mobile objects tracked by the vision module instead of raw videos and attempts to recognize all interesting human behaviors.

Several frameworks for Automatic Video Interpretation are proposed by both vision and AI researchers. Chapter 2 presents a selection of such frameworks. In our research context, we wish to use the framework which is described in next section.

1.4 Current ORION's Framework for Automatic Video Interpretation

The research group ORION of INRIA Sophia-Antipolis, France, has proposed a framework for Automatic Video Interpretation [Brémond, 1997; Thonnat & Rota, 1999] used in, and evaluated by, the European projects PASSWORDS & ADVISOR (for metro monitoring application), the French project CASSIOPEE (for bank monitoring application), the European project SAMSIT (for train surveillance application) and the European project AVITRACK (for apron monitoring application). The Automatic Video Interpretation System of this framework is a knowledge based system composed of (1) a knowledge base containing 3D geometric and semantic information of the observed environment and the a-priori knowledge of scenarios to be recognized and (2) three principal modules for: (i) detecting mobile regions by low level image processing methods, (ii) tracking the detected mobile regions and (iii) recognizing scenarios related to activities of mobile objects evolving in the observed scene and interpreting the recognition results. Figure 1.3 shows ORION's three-module framework for Automatic Video Interpretation.

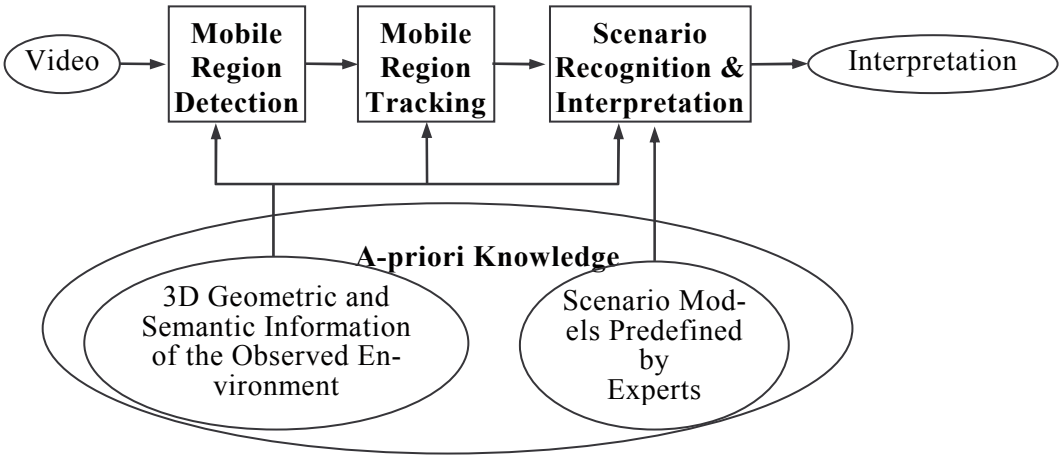


Figure 1.3. The architecture of the Automatic Video Interpretation System proposed by the ORION group and used in/evaluated by the projects ADVISOR, CASSIOPEE, SAMSIT and AVITRACK.

In this framework, the two modules dedicated to the detection and tracking of mobile regions are considered as two sub-modules of a vision module. Several cognitive vision techniques are integrated in these two modules. Specially, a *long-term tracking* method is integrated in the mobile region tracking module. All the detected mobile objects are tracked during a long time interval (e.g. in 20 frames) to be well identified and to calculate their properties (e.g. the direction, the trajectory) [Cupillard *et al*., 2004].

1.5 Temporal Scenario Recognition Principle

The process of recognizing pre-defined scenarios at each instant (i.e. video frame) can be considered as the reasoning performed on a set of entities (e.g. tracked individuals, previously recognized scenario instances) constrained by a set of conditions (both temporal and non-temporal constraints) in order to obtain the interpretation of the given scene. More precisely, the recognition process takes as input at each instant (1) the a-priori knowledge of scenarios to be recognized and of the observed environment, (2) the scenario instances previously recognized up to the current instant and (3) the individuals detected and tracked by the vision module at the current instant. Figure 1.4 shows the process of recognizing pre-defined scenarios at each instant. Figure 1.5 shows an example of steps of the recognition of a "Bank attack" scenario.

The recognition of pre-defined scenarios is specially related to Temporal Constraint Satisfaction Problem (TCSP) and several theoretical domains (e.g. plan recognition, event calculus). A number of techniques can be used to solve the problem (e.g. classification techniques, Hidden Markov Models (HMMs), neural networks, temporal constraint propagation and classical temporal resolution).

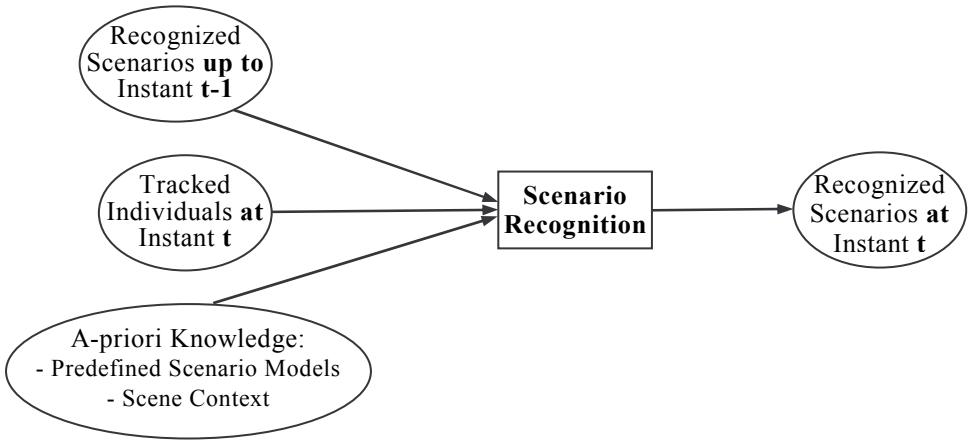


Figure 1.4. Scenario recognition process at each instant t.

Time	Recognized Scenarios
t_1	the *commercial* is at his position *behind* the counter
t_2	the *robber* enters
t_3	the *robber* moves to *the front* of the counter
t_4	the *commercial* arrives at the *safe door* and the *robber* arrives at the *safe door* **Bank attack scenario is recognized**

Figure 1.5. Example of the recognition steps of "Bank attack" scenario.

1.6 Objectives

We focus on the recognition of temporal scenarios for the Automatic Video Interpretation framework proposed by the research group ORION. We show in chapter 2 that there are several techniques for recognizing temporal scenarios or similar problems. However, several problems remain to be solved. For example, the problem of processing time in scenario recognition and its accuracy.

Through the Automatic Video Interpretation frameworks presented in section 1.3, we have found that the recognition of temporal scenarios is **important** for the Automatic Video Interpretation. Moreover, this task is really an interesting **challenge** to be studied.

Given this context, this thesis aims at studying (1) the **representation of temporal scenarios** for Automatic Video Interpretation to propose an **ontology structure** of video events, a **generic model of temporal scenarios** and a **description language** to represent knowledge about video events for different Automatic Video Interpretation applications and (2) the recognition of temporal scenarios to propose an approach **recognizing correctly** and **robustly** in **real-time** the pre-defined scenarios (e.g. human activities).

1.7 Plan of the Thesis

The thesis is composed seven chapters organized in three parts related to the temporal scenario recognition tasks. The main contributions are presented in the second part: chapter 3, chapter 4, chapter 5 and chapter 6.

Chapter 2: first presents the research works in **Automatic Video Interpretation** and then **techniques for Human Activity and Temporal Scenario Recognition**. These techniques are classified into three categories: (1) Probabilistic and Stochastic, (2) Symbolic and (3) Symbolic Temporal Techniques. Then, a synthesis of these techniques is shown. Finally, the conclusion shows the objectives of this thesis compared to the state of the art.

Chapter 3: presents our video event ontology for representing knowledge about activities in observed scenes. Then, based on the proposed video event ontology, we propose a hierarchical scenario model for modeling activities of interests for experts from different domains. Finally, we propose a description language to represent temporal scenarios (e.g. video events, human activities,…), based on the hierarchical scenario model. The language is currently used by experts for modeling scenarios in our application domains.

Chapter 4: presents the overview of the scenario recognition.

Chapter 5: presents our approach for the recognition of elementary scenario models.

Chapter 6: presents our temporal composed scenario recognition approach. This is a novel approach to recognize in real-time (video cadence) pre-defined temporal composed scenarios. The novel approach is based on techniques for the propagation and the resolution of temporal constraints.

Chapter 7: first presents our test platform for an Automatic Video Interpretation System, then the experiments realized by end-users and the obtained results.

We conclude the thesis by a brief synthesis of our contributions and the future works that can be realized.

Chapter 2. State of The Art

This chapter first presents previous research works in Automatic Video Interpretation and in Temporal Scenario Recognition. Then, the challenge of the thesis will be presented in the conclusion of the chapter.

2.1 Introduction

There is a large number of research works related to the problem of temporal scenario recognition for Automatic Video Interpretation. We first present the research works in **Automatic Video Interpretation** and then **techniques for Human Activity and Temporal Scenario Recognition**. These techniques are classified into three categories: (1) Probabilistic and Stochastic, (2) Symbolic and (3) Symbolic Temporal Techniques. Then, a synthesis of these techniques is shown in section 2.6. Finally, we conclude the chapter by describing the objectives of this thesis compared to the state of the art.

2.2 Automatic Video Interpretation

This section presents a selection of **Automatic Video Interpretation frameworks** that are used in different applications. These frameworks are proposed by both computer vision and AI researchers. The techniques used in these frameworks are classified into different categories and addressed in sections 2.3, 2.4 and 2.5.

The **Vitra** project developed an Automatic Video Interpretation System called *Soccer* [Herzog *et al.*, 1989] in collaboration with the University of Saarlandes and the University of Karlsruhe, Germany. *Soccer* had for objective: the automatic understanding and the creation of an oral description of traffic scenes or short sequences from soccer matches. The system is composed of a knowledge base containing the scene context and of three principal modules. These three modules realize all tasks necessary for an Automatic Video Interpretation System at different levels; i.e. at the lowest level - image processing, at the medium level - activity recognition and at the highest level - oral scene description. The first module contains image processing methods for detecting mobile objects with their positions and their speeds in the observed scene. The second module consists of methods for action analysis and takes the detected mobile objects (with their properties) as inputs to recognize activities in the scene. The third module creates automatically oral descriptions of the observed scene from the recognized activities.

The research group **IITB** Karlsruhe, Germany, developed a system for vehicle behavior recognition at gas stations [Nagel, 1991]. This system is composed of

two principal modules; i.e. a vision and an interpretation module. The first module contains efficient methods for detecting and tracking vehicles from a video stream. The second module takes the tracked vehicles (the output of the first module) as inputs and analyzes their behaviors (e.g. stops to get gas). Several techniques were used to build this system as a combination of vision techniques and intelligent interpretation techniques.

The European project **VIEWS** was a collaboration of 8 partners, the IITB Fraunhofer (Atlas Elektronik GmbH), GEC Hirst Research Centre, GEC Marconi Reseach Centre, Marconi Command and Control Systems, the university of Reading, Queen Mary and Westfield College and FTC (Framentec Cognitech). The Automatic Video Interpretation System of this project is composed of two main modules; i.e. a perceptual module (vision module) and a conceptual module (interpretation module) [Corrall, 1992]. The perceptual module first detects the mobile regions, then tracks the detected mobile regions and finally, identifies the tracked mobile regions by using classification methods. The conceptual module analyzes activities at three levels: event, behavior and dynamic levels. An event is an interesting change in the observed environment, a behavior is a sequence of events and an activity of the dynamic level is a set of behaviors involving a number of mobile objects in a time period. There are three tasks to be done at each level of the conceptual module concerning the classification of entities (e.g. an event, a behavior), the verification of its coherence with previously obtained results and the predicting of next entities (e.g. next event, next behavior). Figure 2.1 shows the framework proposed during this project.

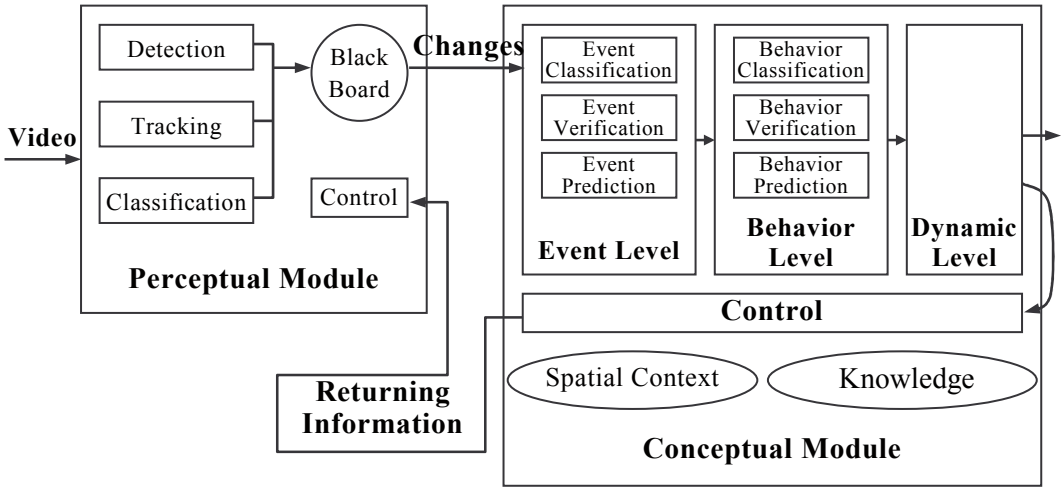


Figure 2.1. The Automatic Video Interpretation of the **VIEWS** project. The system is composed of a knowledge base and three modules. (1) The first module –Perceptual Module– contains vision algorithms for mobile object detection, tracking and classification. (2) The second module –Conceptual Module– analyzes activities at three levels: Event, Behavior and Dynamic. (3) The third module –Control– is integrated in the system for establishing collaboration between the two main modules.

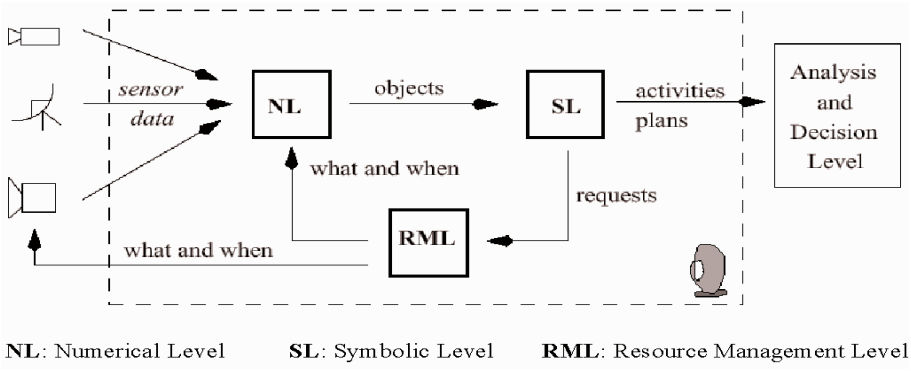


Figure 2.2. The Automatic Video Interpretation framework proposed by the **Perception** project [Castel *et al.,* 1996b] contains three modules. (1) The first module –NL– processes data at the numerical level and contains vision algorithms for mobile object detecting, recognizing and tracking. (2) The second module –SL– processes data at the symbolical level and recognizes activities evolving in the observed scene. (3) Finally, RML is a module for establishing collaboration between the two modules NL and SL.

The European project **Esprit HPCN PASSWORDS** was a collaboration of 5 partners, the University of Genova (DIBE), research center CRIF, Vigitec Belgium, ORION (INRIA Sophia-Antipolis) and Sepa (Fiat Research Center). In 1996, this project proposed and developed an Automatic Video Interpretation System of three modules for: detecting mobile regions, tracking detected mobile regions and identifying mobile objects and analyzing their behaviors [Bogaert *et al*., 1996].

The project **Perception** was a collaboration of two partners in France, ONERA and ETCA-CREA. This project developed a system composed of a context base and three principal modules [Castel *et al*., 1996b]. The context base contains 3D geometric and semantic information of the observed environment as the a-priori knowledge of the system. The first module –NL– (Numerical Processing Level) is a vision module with the objective of detecting, recognizing and tracking mobile objects. The second module –SL– (Symbolical Processing Level) deals with the interpretation of the observed scene; it means to recognize scenarios relative to activities of mobile objects evolving in the observed scene. The third module –RML– (Resource Management Level) aims at controlling the cooperation between the two modules NL and SL. Figure 2.2 shows the Automatic Video Interpretation System framework proposed by this project.

2.3 Probabilistic and Stochastic Techniques

This section presents Probabilistic and Stochastic techniques for human activity recognition. The main characteristic of these techniques is to model explicitly uncertainty using numbers. The section starts by describing **Bayesian Classifier** techniques, then **Neural Networks** techniques. Both techniques are well adapted to model the uncertainty in the recognition of events depending of visual features at a given time. With Bayesian classifiers, the combination is inferred from the frequency of the observations of events in function of visual features. With Neural Networks, the combination is stochastically adjusted by improving the recognition

over a learning set of samples. Finally, we describe **Hidden Markov Model** techniques applied to human activity recognition. These techniques are usually used to recognize sequences of events.

2.3.1 Bayesian Classifier

Dynamic scenes are an uncertain environment, thus Bayesian classifiers are well adapted to cope with this problem [Hongeng *et al.*, 2000]; if the variables (i.e. characteristics) are conditionally independent from one another, a naive classifier can be used in the structure of states previously presented and the Bayesian rule is used to infer the object class. Figure 2.3 shows an example of Bayesian classifier. This classifier needs to learn the parameters, e.g. P(ratio | car) and P(ratio | non-car).

The main advantage of Bayesian classifiers is that they are capable to model the uncertainty of the recognition by using probabilities. They have been often used to recognize elementary actions at the numerical level with only one physical object and several reference objects. However, they have two main drawbacks. First, the a priori probability needs to be learned and this learning stage is often tiresome: due to the construction of the learning sets. Second, they are not adapted to model temporal relations, because the time when the visual features have to be computed needs to be explicitly indicated.

Bayesian classifiers were also used in a multi-agent algorithm [Hongeng & Nevatia, 2001] to recognize complex events composed of action threads, each thread being executed by a single actor. A single thread of action is recognized from the characteristics of its actor using Bayesian methods. A multi-agent event is represented by a number of action threads that are linked by temporal constraints. Multi-agent events are recognized by propagating the constraints and likelihoods of event threads in a temporal logic network.

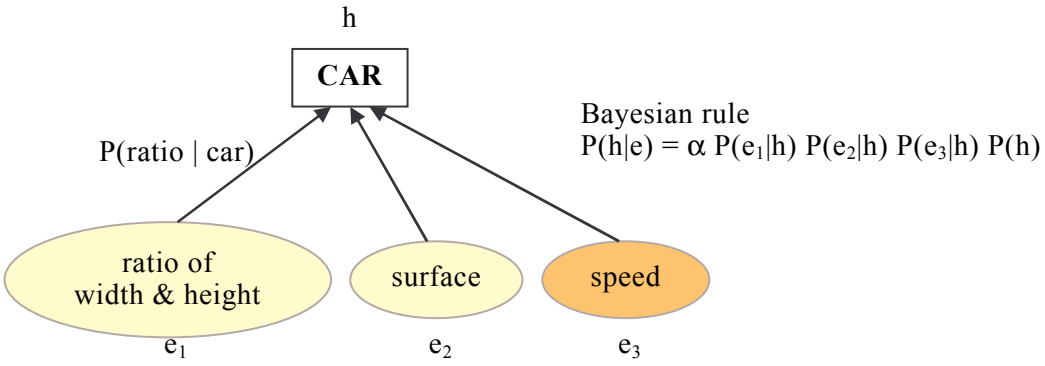


Figure 2.3. Example of a Bayesian classifier used to detect a car knowing three visual features: its ratio (e_1), its surface (e_2) and its speed (e_3).

2.3.2 Neural Networks

Howell & Buxton (1998, 2001, 2002) and Howarth & Buxton (2000) have used neural networks techniques for human behavior recognition applied in Visually Mediated Interaction. Visually Mediated Interaction can be considered as any process that facilitates the interaction between people (remotely or locally) using visual cues,

which are similar to those used in our everyday communications. A human behavior can be considered as a temporal sequence of body movements or configurations, e.g. a change of head pose, walking and sitting down. Several examples of human behaviors are shown in Figure 2.4 concerning gestures in human communication.

Gesture	Body Movement	Behavior
pntrl	point right hand to left	pointing left
pntrr	point right hand to right	pointing right
wavea	wave right hand above head	urgent wave
waveb	wave right hand below head	non-urgent wave

Figure 2.4. Example of body movement and behavior definitions for the gesture database [Howell & Buxton, 2002].

Human behaviors evolve normally in an uncertain environment thus neural networks techniques have been used to cope with this problem. More precisely, Howell and Buxton (2002) used a time-delay variant of the Radial Basis Function (RBF) network to recognize simple pointing and waving hand gestures in image sequences. In this approach, characteristic visual evidence can be automatically selected during the adaptive learning phase depending on the task demands.

This neural network approach tries to recognize human behaviors taking advantage of learning techniques which can adapt the recognition algorithm to uncertain environments. However, it is not efficient to cope with complex behaviors involving a large number of physical objects and complex temporal constraints (e.g. synchronized constraint) because it leads to a combinatorial explosion of possible behaviors corresponding to all combinations of physical objects detected in the scene.

2.3.3 Hidden Markov Model (HMM)

As Bayesian classifiers, HMMs are also used to model uncertainty of the observed environment and in particular, the uncertainty of temporal relations of events. Hongeng *et al.* (2000b) presented an approach using the HMM to recognize multi-state activities in dynamic scenes. A multi-state activity is a temporal sequence of mono-state activities and is inferred from the probability of mono-state activities observed in a time period. A mono-state activity is a primitive activity corresponding to the status of a mobile object. For example, "a car slows down toward an object" is a mono-state activity and "a car turns back at a roundabout" is a multi-state activity [Figure 2.5]. The principle of this approach is to use the Makovian hypothesis: the probability of being in a given state only depends on the probability of being in the direct previous state. The possibility of the recognition of an activity when an observation O is made is calculated through the following formula:

$$\text{P(activity/O)} = \underset{\forall(t_1,\ldots,t_n)}{Max} \prod_{1\le i\le N} \frac{A_{i,i-1} P(Si_{(t_i,t_{i+1}-1)} \mid O_{(t_i,t_{i+1}-1)})}{P(Si_{(t_i,t_{i+1}-1)})}$$

Where: - t_i refers to the time that the transition to state *i* from state *i-1* occurs,

- $t_{N+1} = t$ is the current time,

- $S_{i(t_i, t_{i+1}-1)}$ means that scenario S_i occurs between t_i and t_{i+1} - *1*,
- $A_{i,i-1}$ is the a priori probability of the transition from state *i*-1 to state *i* which is assumed to be constant for all activities,
- $\prod_{1 \le i \le N} P(S_{i(t_i, t_{i+1}-1)})$ is a characteristic of the activity and can be learned from image sequences or provided as context.

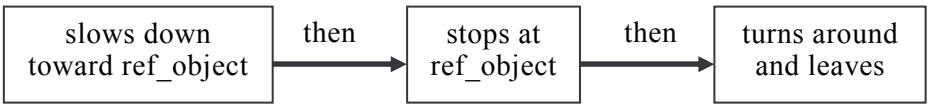


Figure 2.5. The "A car turns back at a roundabout" multi-state activity is defined as a sequence of three mono-state activities corresponding to three steps of the activity: "slows down toward the roundabout", "stops at the round about" and "turns around and leaves". This activity can be recognized by HMMs.

All these numerical approaches have the advantage of modeling the uncertainty of the observed environment. The advantage of HMMs compared to Bayesian classifier and Neural Networks is the ability to recognize sequences of events. However, they are limited in the way they recognize sequences of events where several mobile objects are involved. The probability of being in a state for a mobile object has to be combined with the probability of being in another state for all other mobile objects. These combinations lead the recognition process to a combinatorial explosion. Different usages of HMM Techniques can also be found in [Ivanov *et al.*, 1999; Bui *et al.*, 2001, 2002; Bui, 2003].

2.4 Symbolic Techniques

This section presents Symbolic techniques for human activity recognitions. These techniques aim at transforming numerical observations into symbolic scenarios. The section starts by describing an Action Classification technique, then an Automata-based technique. Finally, this section presents a generic Constraint Satisfaction Problem technique that can be easily used for video event recognition.

2.4.1 Action Classification

An action corresponds to a state characterizing a situation related to several mobile objects of the observed environment [Thonnat & Rota, 1999], for example: "walking", "be close to". Thonnat and Rota (1999) proposed to represent a state by a n-ary tree of four types of nodes: **object**, **descriptor**, **operator** and **classifier** as shown in Figure 2.6. The **object nodes** correspond to the physical objects of the observed environment at an instant *t*, e.g. a *person*, a *zone* and a *piece of equipment*. The **descriptors** are functions defined from O-the set of physical objects- into $\mathbb{R}^p$ for getting a measure of a given object, e.g. the *height*, the *position*, the *shape*, the *trajectory*, the *orientation* or the *volume* of a given physical object. The **operators** are functions defined from ($\mathbb{R}^{p_1}$ x ... x $\mathbb{R}^{p_n}$) into $\mathbb{R}^q$ to combine the measures, e.g. the *distance* between two physical objects, the *arithmetical* or *classical logic operators*. The **classifiers** are functions defined from $\mathbb{R}^p$ to S -a set of symbols-, e.g. *large*,

small, *rapid*, *slow*, *close*, *far*. These operators translate numbers into symbols by associating to each symbolical value a definition domain.

The recognition process of a state represented by a tree has four steps. (1) The recognition process first assigns the leaf nodes (i.e. object nodes) of the tree with different detected physical objects. (2) Then, it calculates all properties of these objects by performing the functions defined by descriptors. (3) Different measures between the given physical objects are calculated by evaluating the functions defined within operators on the calculated properties of physical objects. (4) Finally, based on the measures calculated by the third step, the classifier transforms different combinations of these measures into symbolic values corresponding to recognized states.

This approach shows two main advantages: (1) the representation is clear and simple to be used and (2) the recognition is simple and can be used directly at the numerical level processing, especially it can be adapted to dedicated routines by using specific functions.

The main drawback of this approach is that this action classifier algorithm can only recognize states. This approach is not adapted to model temporal relations.

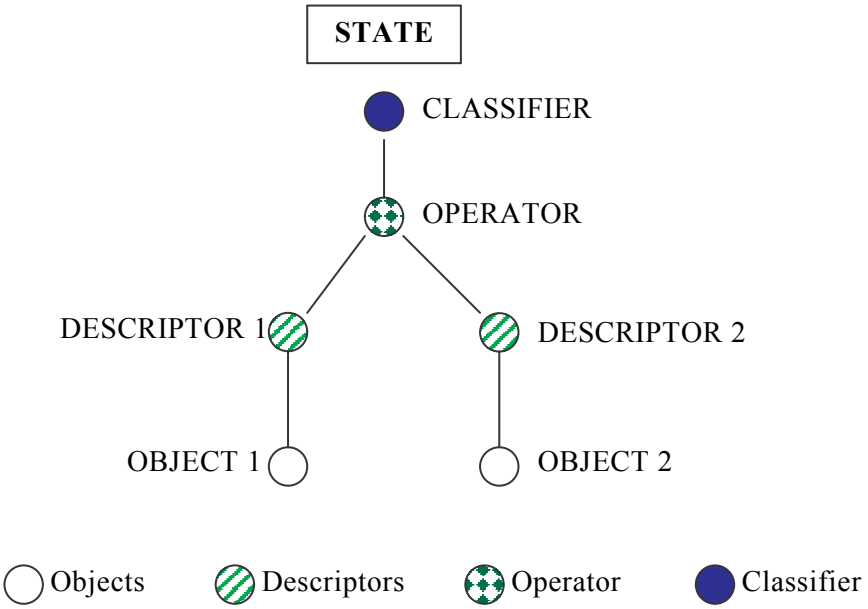


Figure 2.6. A state is represented by a tree [Thonnat & Rota, 1999]. The leaf nodes of this tree correspond to physical objects of the observed environment. The descriptors and the operator correspond to functions for numerical calculations. The classifier is a function transforming numerical values into symbolic values.

2.4.2 Automata

Recently, automata have been used to recognize human behaviors in video sequences [Cupillard *et al.*, 2004]. Cupillard *et al.* distinguish three levels of video events: *state*, *event* and *scenario*. A *state* describes a situation characterizing one or several physical objects defined at time *t* (e.g. a group is agitated) or a stable situation defined over a time interval. For the state: "an individual stays close to the ticket vend-

ing machine", two physical objects are involved: an individual and a piece of equipment. An *event* is a change of states at two successive instants (i.e. image frames) (e.g. a group enters a zone of interest). A *scenario* is a combination of states, events or sub-scenarios. They have also used constraints (both non-temporal and temporal constraints) to represent the relations between states/events/sub-scenarios composing a scenario. *Behaviors* are specific scenarios (dependent on the application) defined by the users. For example, to monitor metro stations, end-users have defined 5 targeted behaviors: “Fraud", "Fighting" "Blocking", "Vandalism" and "Overcrowding".

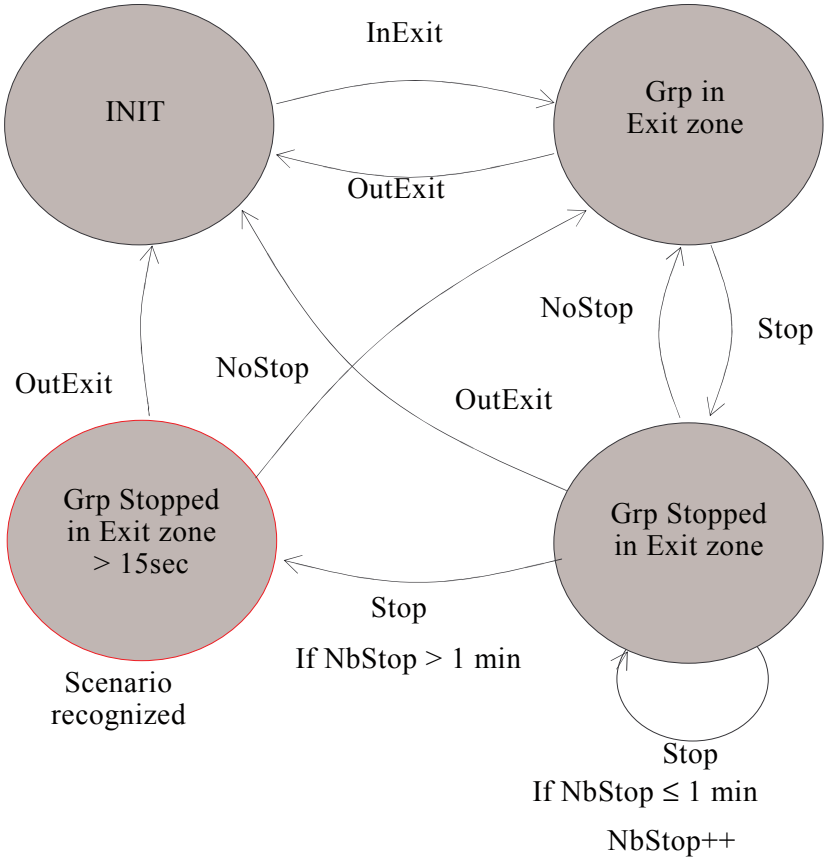


Figure 2.7. A human behavior represented by an automaton composed of four states [Cupillard *et al*., 2004].

Several numerical techniques are used to recognize video events up to the event level (e.g. numerical calculations of basic properties of physical objects, comparison of states at two consecutive instants to recognize events). At the scenario level, they use an automaton approach for recognizing pre-defined scenarios. To recognize a scenario M, the scenario recognition process creates an automaton representing the scenario M. The states of this automaton correspond to the states/events/sub-scenarios composing M. The transitions of this automaton correspond to the constraints defined between two states. Figure 2.7 represents the automaton correspond-

ing to the behavior “A Group of people blocks an Exit”. Each time the scenario recognition process receives a state/an event, it (1) initiates new structures corresponding to automata representing scenarios to recognize and (2) calculates new states of automata previously created to complete the recognition of scenarios partially recognized. A scenario is recognized if the corresponding automaton arrives to the final state.

This approach has the advantage of reusing the scenarios partially recognized at previous instants instead of recalculating them at each instant. Moreover, it also shows the capacity of predicting which scenarios will happen in the observed scenes. However, it has several drawbacks. For example: (1) if a scenario M is defined with several physical objects, the scenario recognition process has to create all the automata corresponding to all combinations of physical objects defined within M for the recognition of M. Moreover, the number of states of a scenario increases in function of the number of physical objects involved in the scenario, because these physical objects can evolve in many different situations. This can lead the algorithm to a combinatorial explosion problem. (2) It is difficult to represent the notion of synchronization and duration using these automatons. This can limit the classes of scenarios that can be recognized by this approach.

2.4.3 Constraint Satisfaction Problem Technique

Recently, temporal scenario recognition has been used to recognize human activities for automatic video interpretation [Rota & Thonnat, 2000; Rota, 2001]. Rota & Thonnat represented an activity (also called scenario) S by a set of positive (+) / negative (-) variables corresponding (at each instant t) to the detection of individuals, pieces of equipment, instantaneous recognized scenarios. A positive variable corresponds to an expected object/event, whereas a negative variable corresponds to an object/event that is not allowed to occur during the recognition of the given scenario. These variables are linked by a set of conditions $\kappa(S)$ corresponding to temporal constraints and also to non-temporal constraints. Each constraint is a Boolean predicate involving these variables. A constraint is called a negative constraint if it involves at least one negative variable; otherwise it is called a positive constraint. Figure 2.8 shows an example of "a person disappears" scenario represented by the formalism proposed in this work.

```
"a person disappears"
    (v1: +, v2: -)
    category(v1)  = person
    category(v2)  = person
    name(v1)    = name(v2)
    time(v2) - time(v1) = 1
```

Figure 2.8. An example of "a person disappears" scenario represented by Rota & Thonnat (2000). A person disappears at an instant t, if he/she was present in the scene at the instant t-1 but not at the instant t.

The recognition of a scenario is based on the evaluation of all constraints defined within the given scenario with all combinations of variable domain values. A scenario is recognized if all its positive constraints are satisfied and all its negative con-

straints are not satisfied. Rota and Thonnat proposed a three-step algorithm to recognize pre-defined scenarios based on this recognition principle.

Suppose that a scenario S = $\{v_1 : +,\ldots, v_k : +, v_{k+1} : -,\ldots, v_n : -\}$ & $\kappa(S)$. To recognize S, the algorithm will:

(1) decompose the problem (combining n positive/negative variables) into n-k+1 problems combining only positive variables (each problem is a CSP -Constraint Satisfaction Problem-):

$P_0 : \{v_1 : +,\ldots, v_k : +\}$ & $\kappa(S)/_{\{v_1,\ldots,v_k\}}$,

$P_1 : \{v_1 : +,\ldots, v_k : +, v_{k+1} : +\}$ & $\kappa(S)/_{\{v_1,\ldots,v_k,v_{k+1}\}}$,

…

$P_{n-k+1} : \{v_1 : +,\ldots, v_k : +, v_n : +\}$ & $\kappa(S)/_{\{v_1,\ldots,v_k,v_n\}}$,

(2) use AC4 algorithm [Mohr & Henderson, 1986] to check the consistency of the set of constraints defined within the scenario model. This allows to know whether the scenario model can be recognized,

(3) enumerate the set of solutions: the algorithm first finds the set S_0 of solutions of P_0 and then eliminates all solutions of S_0 satisfying $P_1,\ldots, P_{n-k+1}$.

The proposed scenario representation shows an advantage of using a unique formalism to represent spatial, temporal, symbolical and logical concepts. However, there are still some drawbacks: (1) the representation is not intuitive, (2) it is difficult to represent arithmetical concepts and (3) there is not the notion of interval; all scenarios are instantaneous.

The proposed algorithm is generally efficient for short scenarios: the satisfaction of constraints is in order $O(cf^2)$ (c: number of constraints defined within a given model, f: number of entities -total number of persons, of objects of the observed environment and of recognized scenarios-). However, it has many drawbacks: (1) in the worst cases, the algorithm has to enumerate all solutions; each solution corresponds to a combination of variable values. (2) If scenario models are not well defined, there will be a great number of recognized scenarios to be generated and to be used to recognize the other scenarios. Moreover, the scenarios need to be bounded. (3) The algorithm does not use the results obtained at the previous instants and recalculates them at each instant.

2.5 Symbolic Temporal Techniques

In this section, we present previous works in Temporal Scenario Recognition. We present only an overview (without details) of the works in the three related research domains: Temporal Constraint Satisfaction Problem, Plan Recognition and Event Calculus. Then, we describe Petri Net that is used to track event evolution. We continue by focusing on the research work in Chronicle Recognition, because Chronicle Recognition is very close to our research domain. Moreover, chronicle recognition techniques can also be used to recognize human activities for Automatic Video Interpretation. Finally, a temporal constraint propagation technique is presented.

2.5.1 Temporal Constraint Satisfaction Problem

We start this section by showing an example [Figure 2.9] of real life problem taken presented in [Dechter *et al.* , 1991].

John goes to work either by car (30–40 minutes), or by bus (at least 60 minutes). Fred goes to work either by car (20–30 minutes), or in a carpool (40–50 minutes). Today John left home between 7:10 and 7:20, and Fred arrived at work between 8:00 and 8:10. We also know that John arrived at work about 10–20 minutes after Fred left home. We wish to answer queries such as: "Is the information in the story consistent?", "Is it possible that John took the bus, and Fred used the carpool?", "What are the possible times at which Fred left home?", and so on.

Figure 2.9. An example of the problem of satisfaction of temporal constraints [Dechter *et al.* , 1991].

Dechter *et al.* (1991) presented a Temporal Constraint Satisfaction Problem (TCSP) model as an extension of the well-know Constraint Satisfaction Problem (CSP) [Wirth, 1986]. A TCSP involves a set of variables $X = \{X_1,\ldots, X_n\}$ and a set of constraints K. Each variable presents a time point and has a continuous domain. Each constraint is presented as a set of intervals:

$$\{I_1,\ldots, I_n\} = \{[a_1, b_1],\ldots, [a_n, b_n]\}$$

A unary constraint on a variable X_i restricts the value of X_i in a set of intervals related by disjunctions:

$$(a_1 \le X_i \le b_1) \vee \ldots \vee (a_n \le X_i \le b_n)$$

A binary constraint on two variables X_i and X_j represents the permissible values of the distance $X_j - X_i$ by a set of intervals related by disjunctions:

$$(a_1 \le X_j - X_i \le b_1) \vee \ldots \vee (a_n \le X_j - X_i \le b_n)$$

Dechter *et al.* also transformed a TCSP into a graph problem. They represented a TCSP by a graph called *temporal constraint graph*. The nodes of this graph correspond to the variables X. The edges of the graph correspond to the constraints K. Each edge starting and ending at the same node corresponds to a unary constraint on a variable. Each edge between two nodes corresponds to a binary constraint between the two nodes. The edges are labeled by sets of intervals expressing the corresponding constraints.

A tuple $o = \{o_i,\ldots, o_n\}$ is called a solution of the given TCSP if the assignment $\{X_i = o_i,\ldots, X_n = o_n\}$ satisfies all the constraints. A value v is called a *feasible value* for variable X_i if there exists a solution in which $X_i = v$. The set of all feasible values of a variable X_i is called the *minimal domain* of X_i. The graph is *consistent* if there exists at least one solution. Figure 2.10 shows the model of the TCSP presented in Figure 2.9 and the temporal constraint graph representing this TCSP.

After representing a given TCSP by a graph, there remain three main questions to be asked: (1) is the graph consistent? (2) what is the minimal domain for each variable? and (3) what are the solutions of the graph? To answer these questions, they have developed different algorithms to verify the consistency of a temporal constraint graph and to find out all the solutions of the given graph.

Let P_1 be the proposition "John was going to work", and P_2 the proposition "Fred was going to work". P_1 and P_2 are associated with intervals $[X_1, X_2]$ and $[X_3, X_4]$, respectively, where X_1 represents the time John left home while X_4 represents the time Fred arrived at work. Several temporal constraints are given in the story. From the fact that it takes John either 30–40 minutes or more than 60 minutes to get to work, the temporal distance between X_1 and X_2 is constrained by

$$30 \leq X_2 - X_1 \leq 40 \quad \text{or} \quad X_2 - X_1 \geq 60 .$$

Similar constraints apply to $X_4 - X_3$ and $X_2 - X_3$. Choosing $X_0 = 7{:}00$ a.m., the fact that John left home between 7:10 and 7:20 imposes the constraint

$$10 \leq X_1 - X_0 \leq 20 .$$

The constraint on $X_4 - X_0$ assumes a similar form.

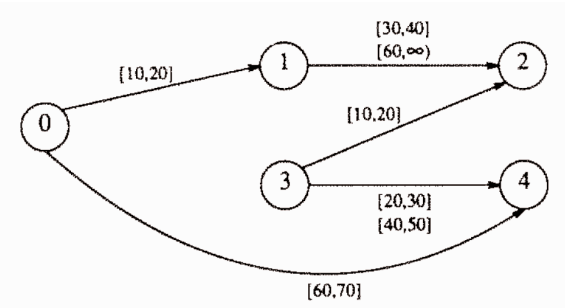


Figure 2.10. A TCSP model and the corresponding temporal constraint graph [Dechter *et al.* , 1991] representing the TCSP shown in Figure 2.9. This TCSP involves five variables: X_0 –the starting time of the problem, the chosen value is 7:00am–, X_1, X_2 are respectively the times when John left home and arrived at work and X_3, X_4 are respectively the times when Fred left home and arrived at work. There are five constraints involving the five variables corresponding the time durations that each person has to take for going to work.

To answer these three questions, there are other works in TCSP [Vila, 1994; Bistarelli *et al.*, 1995; Gennari, 1998; Mouhoub *et al.*, 1998; Schwalb, 1998; Jonsson & Frank, 2000; Rives *et al.*; Benzmüller, 2001; Mouhoub, 1997, 2001; Renz & Nebel, 2001; Khatib *et al.*, 2001]. The authors of these works proposed different algorithms to solve the problems. Especially, different algorithms are developed for finding out the solutions of a TCSP. This is the hardest problem, because the complexity of algorithms is high. To reduce the complexity of the inference algorithms, the authors focused on reducing the number of combinations of time points to be tested.

TCSP has been modeled for static temporal problem. It means that the domains of variables are fixed. However, the problem of activity recognition that we show in part II is a TCSP with non-fixed domains of variables (i.e. the domains can evolve during the process).

2.5.2 Plan Recognition

Schmidt *et al*. (1978) defined succinctly the plan recognition problem as "*to take as input a sequence of actions performed by an actor and to infer the goal pursued by the actor and also to organize the action sequence in terms of a plan structure*". The primary goal of plan recognition is to understand what is happening to predict what will happen in the observed environment. Thus, this understanding allows a user (e.g. a robot) to act adequately with the current situation of the observed environment.

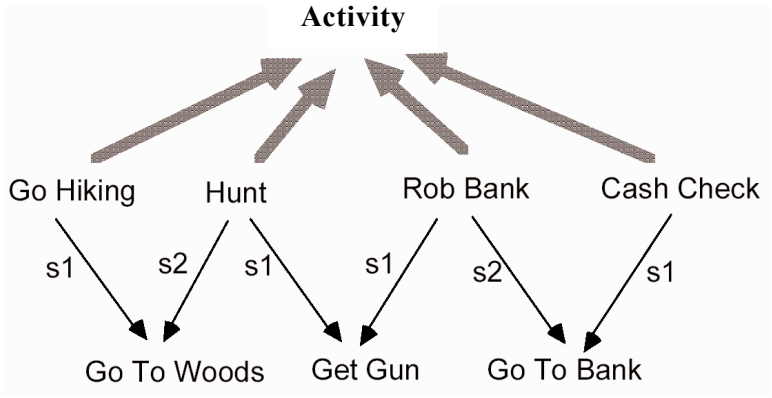


Figure 2.11. This figure shows an example of plan hierarchy taken from [Kautz, 1987]. Shaded arrows represent the abstraction relation of concepts. Thin arrows represent the decomposition relation of concepts.

There has been a large number of research works that have attempted to formalize plan recognition in the AI literature. For example, Kautz & Allen (1986) and Kautz (1987, 1990) have represented a plan as a hierarchy of sub-plans. Figure 2.11 shows an example of a hierarchy of plans taken from [Kautz, 1987]. Shaded arrows represent the abstraction relation of concepts (i.e. going from special concepts to a more general concept). Thin black arrows represent the decomposition relation of concepts. The decomposition relations for a single action type is labeled with a distinct function symbol that distinguishes the different sub-steps in the plan; for example the names *s1* and *s2* carry no formal significance but are meant to be suggestive of the first sub-step and the second sub-step. A step is associated with a time interval. Temporal relations between sub-plans of a plan are represented by temporal operators (e.g. before, during) on time intervals of sub-plans. Moreover, Kautz has used a first-order-logic of events in his logical language to describe interesting plans (Figure 2.12 presents a "PastaDish" plan -taken from the cooking domain- in the formalism proposed by Kautz) and also a first-order predicate calculus to recognize pre-defined plans. Thus, the plan hierarchy represents only reliable knowledge

(without uncertainty) about the possible plans that can be recognized. So, the plan hierarchy needs to be completed to enable the recognition of all plans.

Kautz and Allen (1986) noted that the resulting plan recognition theory is monotonic, thus, a plan recognition solver can treat all possible plans with an observation as plausible. If an action does not occur, several plans can match (with a missing step) the sequence of actions. So, it is necessary to model the probability of the plan recognition to choose which plan better fits the sequence of actions. To cope with this problem, heuristic models were used to recognize pre-defined plans as shown in [Carberry, 1990]. Heuristic models allow representing preferences between several possible plans. Moreover, probabilities are used to model the prior plausibilities of plans enabling plan recognition solvers to choose the best matched plan in the cases that there are more than one possible plan [Charniak & Goldman, 1991; Huber *et al.*, 1994; Albrecht *et al.*, 1998].

```
∀x(PastaDish(x) →
(1)    Noodles(s1(x)) ∧
(2)    Sauce(s2(x)) ∧
(3)    Boil(s3(x)) ∧
(4)    agent(s1(x)) = agent(x) ∧
(5)    agent(s2(x)) = agent(x) ∧
(6)    agent(s3(x)) = agent(x) ∧
(7)    during(time(s1(x)), time(x)) ∧
(8)    before(time(s1(x)), time(s3(x))))
```

Figure 2.12. This example shows a plan represented in Kautz's formalism (1987). The presented plan models a process in three steps (s1, s2 and s3) to prepare a pasta dish. The first three lines declare the three actions. The following three lines tell that these three actions are performed by the same agent. The last two lines describe the temporal relations between the three steps. These relations are expressed by two temporal operators (during and before) on the time intervals of these steps (accessed by the "time" predicate).

Plan recognition can be used for video interpretation. Several approaches have been derived from plan recognition to recognize activities such as chronicle recognition which is described in section 2.5.5.

2.5.3 Event Calculus

Event calculus is the process of reasoning from a set of states to deduce properties on the observed environment. The goal of event calculus is to understand the "snapshot" of the observed environment at a given moment. Event calculus is based on changes of states at different instants. Events are atomic and there is no other event triggered by the recognition of an event.

Kowalski & Sergot (1986) introduced event calculus as formalism to reason about events. The representation of events is based on first-order logic. An event can initiate or terminate a process. Figure 2.13 shows several events represented in [Shanahan, 2000].

$$\mathrm{HoldsAt}(f,t) \leftarrow \mathrm{Initiallyp}(f) \wedge \neg\, \mathrm{Clipped}(0,f,t) \quad \text{(EC1)}$$

$$\mathrm{HoldsAt}(f,t_3) \leftarrow \mathrm{Happens}(a,t_1,t_2) \wedge \mathrm{Initiates}(a,f,t_1) \wedge t_2 < t_3 \wedge \neg\, \mathrm{Clipped}(t_1,f,t_3) \quad \text{(EC2)}$$

$$\mathrm{Clipped}(t_1,f,t_4) \leftrightarrow \exists\, a,t_2,t_3\ [\mathrm{Happens}(a,t_2,t_3) \wedge t_1 < t_3 \wedge t_2 < t_4 \wedge [\mathrm{Terminates}(a,f,t_2) \vee \mathrm{Releases}(a,f,t_2)]] \quad \text{(EC3)}$$

$$\neg\, \mathrm{HoldsAt}(t_1,f,t_4) \leftarrow \mathrm{Initially}_N(f) \wedge \neg\, \mathrm{Declipped}(0,f,t) \quad \text{(EC4)}$$

$$\neg\, \mathrm{HoldsAt}(f,t_3) \leftarrow \mathrm{Happens}(a,t_1,t_2) \wedge \mathrm{Terminates}(a,f,t_1) \wedge t_2 < t_3 \wedge \neg\, \mathrm{Declipped}(t_1,f,t_3) \quad \text{(EC5)}$$

$$\mathrm{Declipped}(t_1,f,t_4) \leftrightarrow \exists\, a,t_2,t_3\ [\mathrm{Happens}(a,t_2.t_3) \wedge t_1 < t_3 \wedge t_2 < t_4 \wedge [\mathrm{Initiates}(a,f,t_2) \vee \mathrm{Releases}(a,f,t_2)]] \quad \text{(EC6)}$$

$$\mathrm{Happens}(a,t_1,t_2) \rightarrow t_1 \leq t_2 \quad \text{(EC7)}$$

Formula	Meaning
Initiates(α,β,τ)	Fluent β holds after action α at time τ
Terminates(α,β,τ)	Fluent β does not hold after action α at time τ
Releases(α,β,τ)	Fluent β is not subject to the common sense law of inertia after action α at time τ
$\mathrm{Initially}_P(\beta)$	Fluent β holds from time 0
$\mathrm{Initially}_N(\beta)$	Fluent β does not hold from time 0
Happens(α,τ1,τ2)	Action α starts at time τ1 and ends at time τ2
HoldsAt(β,τ)	Fluent β holds at time τ
Clipped(τ1,β,τ2)	Fluent β is terminated between times τ1 and τ2
Declipped(τ1,β,τ2)	Fluent β is initiated between times τ1 and τ2

Figure 2.13. Events represented for an abductive event calculus planner [Shanahan, 2000]. On the top of the figure, we can see the descriptions of seven formulas. An event is represented as a conjunction and disjunction of different predicates. At the bottom of the figure, a table describes the semantics of formulas.

There have been several other approaches in event calculus [Sadri, 1987; Borillo & Gaume, 1990; Shanahan, 1990; Sripada, 1991; Kowalski & Sadri, 1994; Missiaen *et al*., 1995]. Through these works, event calculus has become very advantageous because of two reasons. (1) First, events do not need to be totally ordered. (2) Second, the frame problem(*) has been solved for event calculus [McCarthy & Hayes, 1969].

(*) The frame problem can be formulated as the problem of representing change as performing an inference, i.e. which beliefs are true and false before and after, what remains unchanged and what changes [Vila, 1994].

Moreover, event calculus can express and calculate the truth value of derived properties [Missiaen *et al.*, 1995]. The main drawback of event calculus is the search in a large database that is expensive, because the events are represented in first-order logic.

As far as we know, event calculus has not been applied for video interpretation, but it can be adapted for this utilization. Several approaches (e.g. chronicle recognition) dealing with video interpretation are inspired by event calculus.

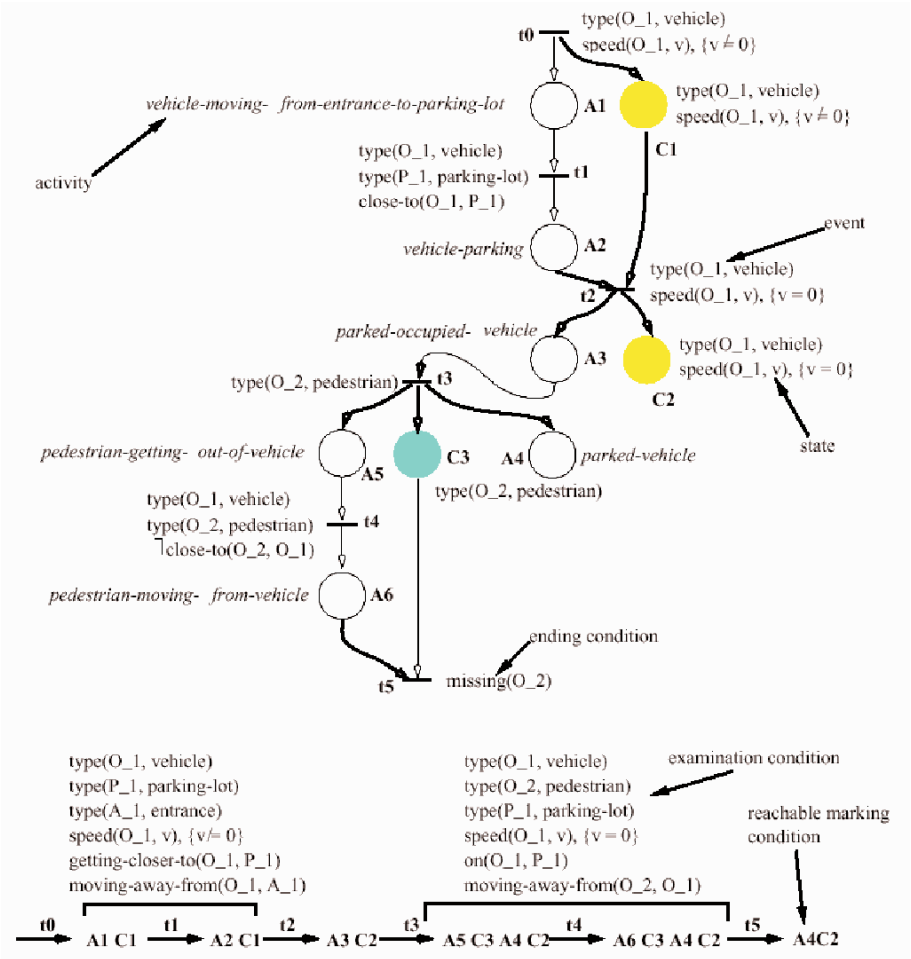


Figure 2.14. On the top of figure, Petri net is used to represent a "vehicle-arrival" scenario corresponding to the arrival of a vehicle and followed by a pedestrian leaving the parking-lot. The scenario is described using le language SL_2. The bottom of the figure illustrates the recognition of the scenario. (1) At time t_0, the vehicle appears at the entrance (speed ≠ 0). (2) Then the vehicle moves close to a parking-lot at time t_1 and (3) stops at the parking lot at time t_2. After the vehicle is stopped, (4) the driver gets out of the vehicle at time t_3 and (5) goes away from the vehicle at time t_4. (6) Finally, at time t_5 the conductor is far from the observed zone.

2.5.4 Petri Nets

[Castel *et al*., 1996] presented SL_2 - a logical language designed to capture the logical and algebraic conditions that are handled in symbolical models, in terms of properties and constraints. A scenario is represented by a set of terms (e.g. constant, variable) that are related to the other terms by constraints and predicates (e.g. type(O, vehicle) is a predicate). In this language, both numerical and Boolean constraints are taken into account and are expressed through binary relations.

To recognize predefined scenarios, Castel *et al*. have used a Petri nets approach. A Petri net is built with the predefined scenario models [Figure 2.14]. The places of this network correspond to the states of the recognition process. The transitions of the Petri net correspond to transitions between states. Once the recognition process receives an observation from a vision routine, it tries to match (*a pattern matching is possible between an observation obs(C) and an activity act(c) if and only if the discriminative properties of c are present in C*) the observation with the places of the Petri net. One (or several) places are marked. The marked places are scenarios that may be recognized.

Continuing this approach, Cossart-Jaupitre (1999) extended this approach to track/recognize situations and to apply it to parking-lot surveillance application. The extension is made for the recognition process to be able to cope with the uncertainty of the observed environment. The main point of this approach is to build a symbolical estimator inspired from Kalman numerical estimator. This approach is still continued to be studied [Dehais *et al*., 2004; Tessier, 2003] and especially it makes the research subject of Lesire's PhD thesis.

The advantages of Petri nets are expressed through: (1) the capacity of sequencing, parallelism and synchronization, (2) Petri nets allow monitoring and prediction. However, this approach can lead the recognition process to a combinatorial problem when coping with temporal scenarios defined with several physical objects and with scenes composed of a large number of mobile objects. Moreover, some temporal constraints (e.g. "person B arrives 1 minute after person A left") are difficult to express using this formalism.

2.5.5 Chronicle Recognition

A chronicle is a set of temporally sequential/parallel states/events occurring in the observed environment. Chronicle recognition is the process of reasoning incrementally from observations to chronicles. Thus, the time or temporal relations are the most important aspect to focus on.

Chronicle recognition was presented in [Kumar & Mukerjee, 1987]. A chronicle model was presented as a combination of a state based approach and an extended interval algebra. The interval algebra was extended to contain incomplete intervals (only the start of the interval is known). In this chronicle representation method, there was not the notion of quantitative temporal constraints (e.g. duration, delay). The recognition process is incremental and compares directly the conjunctions of temporal constraints with the sequence of observed events.

A chronicle was also called a dynamic situation [Nokel, 1989] and represented as a sequence of instantaneous events. In [Kumar & Mukerjee, 1987], quantitative temporal constraints were not represented. Partial order and simultaneousness of events were not allowed either. Moreover, the quality of the recognition also depends generally on the frequency of the observations.

```
chronicle IncreasingLoad
{
  timepoint t1,t2,t3,t4,t5,t6;
  //- forthcoming events and context assertion
  event (LOAD:(Steady, Increasing),t1);
  event (SHAFT:(Steady, Increasing),t2);
  event (VALVE:(Steady, Increasing),t3);
  event (EXHAUST:(Steady, Increasing),t4);
  event (NOZZLE:(Steady, Increasing),t5);
  event (NOZZLE:(Increasing, Decreasing),t6);
  hold (ALARM: None, (t1, t4));
  //- temporal constraints
  (t2-t1) in [1.00, 3.00];
  (t3-t2) in [0.00, 1.00];
  (t4-t3) in [0.00, 1.00];
  (t5-t2) in [0.00, 1.00];
  (t6-t5) in [0.00, 2.00];
  when recognized
     report "Successful increasing load";
}
```

Figure 2.15. An example of chronicle presented in [Ghallab, 1996] corresponds to the recognition of whether the load has correctly increased in a gas turbine. This chronicle is composed of six events occurring at six time instants t_1,…, t_6. The temporal relations between these events are expressed through temporal relations between the six time instants. The predicate “hold” indicates that no alarm should be triggered during a successful load.

Chronicles and their recognition were precisely presented in [Dousson, 1994; Dousson & Ghallab, 1994; Ghallab, 1996; Dousson *et al.*, 1993]. A chronicle is represented as a set of events and sub-chronicles linked by temporal constraints. The time is represented as a set of time elements: time points, complete and incomplete time intervals. The temporal aspects are the starting/ending time points of a chronicle and also the delay between two chronicles (an interval is implicitly represented by its starting/ending time points). Particularly, the recognition algorithm is well adapted to quantitative temporal constraints. Figure 2.15 shows an example of the representation of a chronicle in the formalism proposed by Dousson and Ghallab (1994). The represented chronicle is composed of six events successively occurring at six different time points t_1,…, t_6. The temporal relations between these events are expressed through five temporal constraints on the time points. A report "Successful increasing load" is displayed if the chronicle is recognized. Before being recognized, a chronicle pre-defined by experts is compiled to check the consistency, to be simplified and to build a temporal constraint graph corresponding to the given chronicle. A graph propagation technique is used to compile pre-defined chronicles. For a given chronicle, the compiler first builds a temporal constraint graph which nodes correspond to the time points defined within the chronicle and which edges correspond to the temporal relations between the time points. Figure 2.16(a) shows the graph corresponding to the chronicle represented in Figure 2.15. The proposed algorithm recognizes incrementally predefined chronicles by using this graph and by storing all partially recognized chronicles. Each time an event is detected (by sensors) or a chronicle is recognized, new temporary graph is created for the new partially recognized chroni-

cle. In each partially recognized chronicle, there are time windows corresponding to the time delay when another event/sub-chronicle is expected or forbidden. If there is any violation of these time windows, the partially recognized chronicle is deleted. If not, the chronicle is recognized when all events have been detected in the authorized time windows. Figure 2.16(b) shows an example of the recognition of the chronicle represented in Figure 2.15. Recently, Dousson (2002) extended this approach to limit the number of chronicle occurrences and to select the longest possible chronicle if there are many. Despouys (2000) has extended this approach to cope with uncertainty. A usage of this technique can also be found in [Quiniou *et al.*, 2001]. All these approaches recognize correctly predefined chronicles and enable a real-time recognition of chronicles. The advantages and drawbacks of these approaches are detailed in section 2.6 together with the ones of Temporal Constraint Propagation approaches, because of the similarity of both approaches.

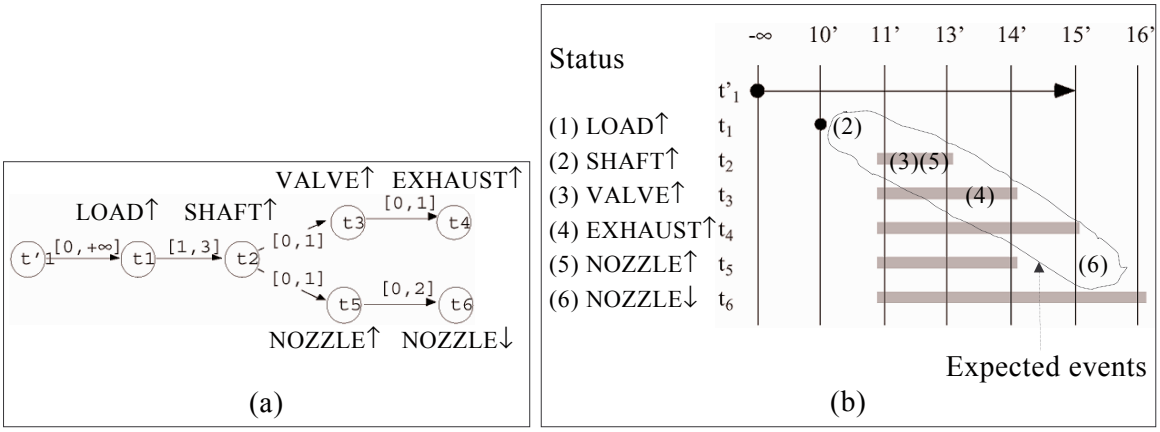


Figure 2.16. (a) The compilation result graph for the recognition of the chronicle represented in Figure 2.15 [Ghallab, 1996]. (b) An example of the recognition of this chronicle.

2.5.6 Temporal Constraint Propagation

Human activity was also called scenario [Chleq & Thonnat, 1996]. A scenario is represented as a set of independent positive/negative instantaneous events. Positive events are expected for the recognition of the given scenario, whereas negative events are not allowed to occur during the recognition of the given scenario. In the formalism proposed by the authors, a positive event is represented by a predicate "occur" and a negative event is represented by a predicate "notoccur". Each event (positive and also negative) is associated with a time point. The events composing a scenario are related by temporal constraints. A temporal constraint is a linear equation/inequation on time points associated with these events. A set of non-temporal constraints (called conditions) is also used to verify several attributes of the physical objects involved by the given scenario. Figure 2.17 shows an example of a "two persons meet at the coffee machine" scenario, this scenario involves two persons, an equipment (the coffee machine) and an interesting area (the machine area). The scenario is composed of four events occurring at four time points t_1,…, t_4.

```
Scenario(Name("two persons meet at the coffee machine"),
  Events(occur(t1, moves close to (p1: Person, e1: Equipment)),
         occur(t2, stops(p1: Person)),
         occur(t3, enters(p2: Person, a1 : Area)),
         occur(t4, moves close to (p1: Person, p2: Person))),
  Constraints(t1 ≤ t2, t3 ≤ t4),
  Conditions(name(e1, "coffee machine"),
             name(a1, "coffee area") )  )
```

Figure 2.17. A "two persons meet at the coffee machine" scenario is composed of four events and is represented by Chleq & Thonnat (1996). The scenario is constituted by four steps: (1) at time t_1, the first person moves close to the machine e_1, (2) at time t_2, the first person stops, (3) at time t_3, the second person enters the area of the coffee machine a_1 and (4) these two persons move close to each other.

The recognition process takes as input at each instant *t*: (1) the partially recognized scenarios at instant *t*-1, (2) a set of new scenario models to be recognized at instant *t* and (3) the events detected at instant *t*. The output of this process is a set of partially/totally recognized scenarios at instant *t*.

The algorithm proposed by Chleq and Thonnat is similar to the chronicle recognition algorithm proposed by Dousson & Ghallab (1994). The algorithm recognizes incrementally pre-defined scenarios representing human behaviors in the observed scene. The authors have used a temporal constraint propagation technique to recognize predefined scenarios. For each scenario model M, a pre-processing process builds a graph G_M representing the constraints defined within M. The vertices of G_M correspond to the time point variables defined within M. The edges of G_M correspond to temporal relations between time point variables. When an event *e* is made (i.e. an event is given by the vision module or is recognized at the same instant), the recognition process propagates *e* in G_M. If all the vertices of G_M are instantiated with a time instant, M is recognized.

This approach shows two main advantages: (1) the recognition is incremental, thus, at each instant, the recognition process uses the scenarios partially recognized at previous instants instead of re-computing them; this can lead to a rapid processing and (2) the recognition process keeps at each instant the partial recognition status of scenarios. However, it shows some drawbacks, for example: (1) the spatial complexity is high, therefore the scenarios need to be bounded in time and (2) if a scenario involves a large number of physical objects, many partially recognized scenarios will be generated corresponding to all combinations of physical objects at the recognition level of this scenario, this can make a combinatorial explosion.

Pinhanez and Bobick (1997) presented an algorithm reducing the complexity of the propagation of temporal constraints of Allen's interval algebra. They proposed a new brief representation of Allen's interval algebra operators by using three notions P-past, N-now and F-future. The recognition algorithm is based on the propagation of temporal constraints on PNF networks, thus the algorithm allows a fast performance compared to the original equivalent evaluations of Allen's interval algebra. The proposed recognition algorithm makes the propagation of temporal constraints less complex, because there is a smaller number of cases to propagate (compared to the original problem). However, it also leads to a combinatorial explosion of partially recognized scenarios while attempting to recognize scenarios involving several

physical objects as the one proposed by Chleq & Thonnat (1996). Moreover, the approach proposed by Pinhanez and Bobick (1997) does not cope with quantitative temporal constraints.

2.6 Scenario Recognition Technique Synthesis

Working in the context of Automatic Video Interpretation, we particularly focus on the recognition of temporal scenarios for Automatic Video Interpretation (i.e. human activity recognition for Automatic Video Interpretation). Three main categories of approaches are used to recognize human activities: (1) Probabilistic and Stochastic Techniques, (2) Symbolic Techniques and (3) Symbolic Temporal Techniques.

The symbolic (temporal) techniques were proposed by AI researchers. We classify these approaches into two categories: Store Totally Recognized Scenarios (S**T**RS) and Store Partially Recognized Scenarios (S**P**RS). The STRS algorithms recognize scenarios that are completely over and store all recognized scenarios to recognize other scenarios [Rota, 2001]; whereas, the SPRS algorithms predict what is likely to happen and store all these predictions to recognize scenarios in the future. A common SPRS algorithm is the chronicle recognition technique proposed by Dousson and Ghallab (1994).

The proposed techniques for activity recognition are generally efficient and have several common characteristics:

1) the S**P**RS algorithms are able to predict which events will occur at the next instants and efficient to process forbidden scenarios.
2) the S**P**RS algorithms get into a combinatorial explosion while coping with multi-physical-object problem, because they have to store and maintain all partially recognized scenarios.
3) the S**T**RS algorithms perform a complete search among all possible totally recognized scenarios, thus they also get into a combinatorial explosion.

This analysis shows that a central problem in scenario recognition techniques is to reduce the complexity of the algorithms to enable a real-time recognition process.

2.7 Conclusion

Plan, chronicle and human activity recognition have been in the focus of numerous research works. The objective of these works is to understand what happened or what is happening in the observed scene to enable users to take decisions adequately knowing the real situation of the environment. The application domains of these works are large, e.g. industrial process control, robotics and video interpretation.

In activity recognition, there are two main issues concerning the representation of knowledge about plans/events/activities/scenarios and the reasoning problem (i.e. recognition method). There are several formalisms for knowledge representation and several classes of recognition methods. Table 2.1 shows the main characteristics of existing techniques for scenario recognition. The short discussion in this section is based on the analysis of these characteristics.

Concerning the knowledge representation issue, there are three main problems: (1) how to represent time, (2) how to decompose a scenario into simpler sub-scenarios and (3) how to model uncertainty. The time (or temporal relation) has been the most important subject of these works. The earliest representations were based on a hier-

archy of plans [Kautz & Allen, 1986; Kautz, 1987, 1990] with qualitative relations. Symbolical temporal constraints have also been used to represent time. To decompose a scenario into simpler sub-scenarios, there have been other representations based on a set/temporal sequence of independent events/sub-scenarios with both qualitative and quantitative temporal constraints. Uncertainty has been also an important point in activity recognition and in particular in video interpretation. To represent the uncertainty of the observed environment, Bayesian classifier, HMMs and neural networks have been used. In video interpretation, the uncertainty is usually tackled at the numerical level taking advantages of vision routines. Once symbolic information has been inferred, this information is considered as true to be processed efficiently by the symbolic level.

Concerning the reasoning issue, there are also three main problems: (1) how to manage uncertainty, (2) how to process temporal relations and (3) how to reduce algorithm complexity to obtain a real time process.

The first approaches to recognize human activities in video sequences were proposed by computer vision researchers used to deal with numerical data and to managing uncertainty. These approaches include mainly Bayesian classifiers, HMMs and neural networks. These approaches have the advantage of coping with the uncertainty of the environment and are efficient for the recognition of activities defined with simple temporal constraints at the numerical level. The main drawback of these approaches is that they are not efficient to tackle complex activities involving sophisticated temporal relations.

Another type of approaches (including symbolic and temporal approaches) is derived from works on plan recognition and event calculus to model temporal relations at the symbolical level. For example, a chronicle recognition algorithm is proposed to recognize “activities” of industrial processes [Dousson & Ghallab, 1994]. These approaches are efficient for the recognition of scenarios, because they can process quickly both quantitative and qualitative complex temporal relations between events and also forbidden events. However, these approaches were designed to recognize mono-physical-object activities, so, some limitations appear to recognize multi-physical-object activities. For example, the recognition algorithm has often to create all predictions corresponding to all the combinations of physical objects while attempting to recognize an activity.

The last issue, how to reduce the algorithm complexity is still open. To analyze this issue, we have classified these approaches (symbolic and temporal approaches) into two categories: Store Totally Recognized Scenarios and Store Partially Recognized Scenarios. Generally, these algorithms can recognize efficiently pre-defined scenarios. However, they show several drawbacks. For example: (1) the Store **Partially** Recognized Scenario algorithms store and maintain all occurrences of partially recognized scenarios as a potential recognition in the future so it leads to a combinatorial explosion and (2) the Store **Totally** Recognized Scenario algorithms perform at each instant a complete search among all possible scenarios and sub-scenarios recognized in the past and all possible combinations of physical objects so it can also lead to a combinatorial explosion.

In this thesis, we focus on (1) proposing a new **real-time scenario recognition algorithm** by taking advantages of both Store **Partially** Recognized Scenario and Store **Totally** Recognized Scenario approaches and also (2) proposing a **more intuitive representation of temporal scenarios** for automatic video interpretation.

	Techniques	Advantages (+) and Drawbacks (-)	References
Probabilistic and Stochastic Techniques for Human Activity Recognition			
1	Bayesian Classification	+ uncertainty - difficulties to process temporal relations	[Hongeng *et al.*, 2000] [Hongeng & Navetia, 2001]
2	Neural Networks	+ adaptable to the environment - difficulties to cope with temporal relations	[Howell & Buxton, 1998, 2001, 2002] [Howarth & Buxton, 2000]
3	HMMs	+ uncertainty + can process sequences of events - tedious learning phase - cannot cope with complex temporal relations	[Hongeng *et al.*, 2000b] [Ivanov *et al.*, 1999] [Bui *et al.*, 2001, 2002] [Bui, 2003]
Symbolic Techniques for Human Activity Recognition			
4	Action Classification	+ simple and easy to use - combinatorial explosion for complex scenes	[Thonnat & Rota, 1999]
5	Automata	+ can process efficiently sequences of events - combinatorial explosion in function of the number of physical objects - cannot represent synchronized events	[Cupillard *et al.*, 2004]
6	CSP technique	+ simple to use - recognition algorithm complexity is prohibitive.	[Rota, 2001] [Rota & Thonnat, 2000, 2000b]
Symbolic Temporal Techniques			
7	TCSP technique	[Dechter *et al.*, 1991] [Vila, 1994] [Bistarelli *et al.*, 1995] [Gennari, 1998] [Mouhoub *et al.*, 1998] [Schwalb, 1998] [Jonsson & Frank, 2000] [Rives *et al.*] [Benzmüller, 2001] [Mouhoub, 1997, 2001] [Renz & Nebel, 2001] [Khatib *et al.*, 2001]	
8	Plan Recognition	[Schmidt *et al.*, 1978] [Kautz & Allen, 1986] [Kautz, 1987, 1990] [Carberry, 1990] [Charniak & Goldman, 1991] [Huber *et al.*, 1994] [Albrecht *et al.*, 1998]	
9	Event Calculus	[McCarthy & Hayes, 1969] [Kowalski & Sergot, 1986] [Sadri, 1987] [Borillo & Gaume, 1990] [Sripada, 1991] [Kowalski & Sadri, 1994] [Missiaen *et al.*, 1995] [Shanahan, 1990, 2000]	
Symbolic Temporal Techniques for Human Activity Recognition			
10	Petri Nets	+ allow prediction - cannot process forbidden events - combinatorial explosion in case of multi-physical-object problem	[Castel *et al.*, 1996] [Cossart-Jaupitre, 1999] [Dehais *et al.*, 2004] [Tessier, 2003]
11	Chronicle Recognition	+ very efficient for mono-physical-object problem + efficient for negative event processing - combinatorial explosion in case of multi-physical-object problem	[Dousson *et al.*, 1993] [Dousson, 1994, 2002] [Dousson & Ghallab, 1994] [Ghallab, 1996] [Despouys, 2000] [Quiniou *et al.*, 2001]
12	Temporal Constraint Propagation	+ can use efficiently the results computed in the past + allows prediction - combinatorial explosion on the number of physical objects	[Chleq & Thonnat, 1996] [Pinhanez & Bobick, 1997]

Table 2.1. This table describes the scenario recognition techniques of the state of the art. For the symbolic temporal techniques (7, 8 and 9) we have not described the advantages and drawbacks, because they have not been applied to human activity recognition. These techniques are mainly a reference for the symbolic temporal techniques (10, 11, 12) used for human activity recognition.

Part II

Temporal Scenario Representation and Recognition

Chapter 3. Temporal Scenario Representation

This chapter presents our temporal scenario representation method for Automatic Video Interpretation.

3.1 Introduction

There are two main issues in Temporal Scenario Representation. The first one consists for the experts to be able to describe easily in an intuitive way their own scenarios. The second issue is for the representation to be sophisticated enough to be able to represent any type of scenarios/relations occurring in real-world applications. In this chapter, we first present a video event ontology for Automatic Video Interpretation. Then, we present a generic hierarchical scenario model for representing video events. Third, we propose a scenario description language to describe temporal scenarios. Finally, we show the utilization of the video event ontology.

3.2 Video Event Ontology

This section presents a video event ontology built in collaboration with the INRIA research team ORION in the framework of ARDA workshop series on video events. The use of Automatic Video Interpretation System has been generalized all over the world leading to the need of building an ontology on the application domains. An ontology is the set of all the concepts and relations between concepts shared by the community in a given domain. The ontology is first useful for experts of a given application domain to use video understanding systems in an autonomous way. The ontology makes the video understanding systems user-centered and enables the experts to fully understand the terms used to describe activity models. Moreover, the ontology is useful to evaluate the video understanding systems and to understand exactly what types of events a particular video understanding system can recognize. This ontology is also useful for developers of video understanding applications to share and reuse activity models dedicated to the recognition of specific events.

Building an ontology used as a reference for video understanding applications is particularly difficult because many developers and experts of application domains all over the world have their own ideas about how to describe human activities. The terms chosen to name the ontology concepts are taken from every day life but they have been redefined to avoid ambiguities.

We first describe the structure of ontology to be used for video understanding applications by defining the meta-concepts necessary to the modeling of physical objects

and their activities. Together with these meta-concepts, we discuss the issues arisen from the ontology structure. Then, we briefly describe relations between the meta-concepts.

In this document, we call meta-concepts (e.g. event) the terms of the ontology structure and we call concepts their instances (e.g. the "stand up" event) for a particular ontology.

3.2.1 Meta-Concepts for Describing Physical Objects

The **physical objects** are all the objects of the real world in the scene observed by the cameras. The **attributes** of a physical object are pertinent for the recognition. These attributes characterize the physical object.

The class of a physical object corresponds to its nature and usually can be determined by its shape. For example, a person, a table and a car are physical objects.

A **contextual object** is a physical object which is usually static and whenever in motion, its movement can be predicted using contextual information. For example, a contextual object can be always fixed, static under conditions, movable at the same position, remotely-movable at the same position, displaceable and automatically-movable at the same position. Typical contextual objects are walls, entrance zones, doors, chairs, suitcases, escalators, trees, unoccupied scene,…

A **mobile object** is a physical object that can be perceived as moving in the scene and as initiating its motions, without the possibility to predict its movement. For example, a mobile object can be under conditions automatically-movable, remotely-displaceable, remotely-movable, programmable, partially-autonomous, fully-autonomous. Typical mobile objects are individuals, body parts, groups of people, animals, robots,…

To distinguish mobile objects and contextual objects, the main point is their ability to initiate their own motion. For example, a car without any driver is a contextual object, whereas a car with a driver is a mobile object. The cars with/without driver belong to different sub-classes of physical objects (mobile objects/contextual objects) even if there have the same appearance. So, a physical object cannot change its class during its presence in the observed scene. Two physical objects can merge to induce the creation of a new physical object. However, the perception of an object can change. A toy-car remotely controlled is a mobile object because the autonomy of the person controlling its motion is attributed to the car. So, when the toy-car is not used, it is considered as a contextual object, whereas, when someone plays with it, it is considered as a mobile object.

Another issue consists in choosing the granularity level to consider physical objects. At a coarse granularity, a group of people can be considered as one mobile object. For example, a group of people when close to each other and all having the same motion will be rather seen as one mobile object. At a finer granularity, a person can be considered as one mobile object. At an even finer granularity, we can think of a person as a complex entity capable of performing simultaneous actions with different body parts. In this case, each body part can be seen as a mobile object. Thus, a physical object can induce the creation of several new objects or merging with other objects to form a unique object.

The class of physical objects can be divided into a hierarchy of sub-classes. For example, individual, group of people, vehicle with a driver,… are usually defined as

the sub-class mobile object. For example, the sub-class contextual object contains portable object, equipment and zone.

The role is an attribute of a physical object. It defines the way it behaves and can be deduced by recognizing its past behaviors or sometimes by detecting specific properties (e.g. wearing a uniform). For example, a person in a bank who behaves like an employee serving customers behind the counter is said to be an employee.

The other properties (e.g. location, speed, size, color) characterizing the physical objects are called visual attributes. There are three types of visual attributes: position-based, global appearance and local appearance. Position-based attributes include properties on the position, speed, direction, trajectory,… Global appearance attributes describe the height, width, ratio and global color. Local appearance attributes include properties on the silhouette, posture, face, sub-part color,…

3.2.2 Meta-Concepts for Describing Activities

There are different ways for characterizing mobile object evolutions and interactions in a scene: state, event (primitive, composite and single/multi-agent composite) and activity.

A **state** is a spatio-temporal property valid at a given instant or stable on a time interval. A state characterizes only one mobile object or a mobile object with respect to other physical objects.

A **primitive state** is a spatio-temporal property valid at a given instant or stable on a time interval which is directly inferred from visual attributes of physical objects computed by perceptual components. Usually, visual attributes have a numerical value and are generic for most of video understanding applications.

A **composite state** is a combination of states. This is the coarsest granularity of states. We call **components** all the sub-states composing the state and we call **constraints** all the relations involving its components and its physical objects. Only the spatial, arithmetical and logical relations can be part of the constraints of a composite state.

An **event** is one or several change(s) of state at two successive time instants or on a time interval. Based on the composition of an event, we distinguish four types of event: primitive, composite, single-agent and multi-agent events.

A **primitive event** is a change of state. Primitive events are more abstract than states but they represent the finest granularity of events.

A **composite event** is a combination of states and events. This is the coarsest granularity of events. Usually, the most abstract composite events have a symbolical/Boolean value and are directly linked to the goals of the given application. We call **components** all the sub-states/events composing the event and we call **constraints** all the relations involving its components and its physical objects. Every type of relations can be part of the constraints of a composite event.

A **single-agent event** is an event involving a single mobile object. Here, a mobile object can be a group of people with the same type of motion.

For example, the most common single-agent composite event is an event made of a sequence of primitive events concerning the same mobile object.

A **multi-agent event** is a composite event involving several (at least two) mobile objects with different motions.

The same event can be viewed at different spatial and temporal granularities. For example, “a man running in a Marathon” can be seen as a state (“he is running”), as a composite event (sequence of acceleration, constant speed, turning,…), as a multi-agent event (motion of the right leg compared to the left one). The chosen granularity indicates the properties of interest for the user.

An event is also characterized by two attributes: its spatial location (locations of the mobile objects involved in the event) and its temporal location (time instant or interval).

3.2.3 Relations between Concepts

All these concepts describing mobile object evolutions and interactions in a scene can involve one or several (at least one) mobile objects and zero or several contextual objects.

The relations between states/events and **physical objects and/or their attributes** indicate how the states/events are inferred from the physical objects and/or their attributes. There are three types of relations: vision-based, spatial and spatio-temporal. The vision-based relations include spatial and temporal filters, arithmetical and statistical operators,… The spatial relations include distance, geometrical, topological relations,... The spatio-temporal relations characterize the evolution of spatial relations in time. The temporal relations between physical objects are treated through the events associated to them.

There are two types of relations between **events**: logical and temporal. The logical relations include “and”, “or”, conditional (“if…then…”). The temporal relations include Allen’s interval algebra operators and quantitative relations between the duration, beginning and ending of events. The most common relation is the sequence of events. Other relations between events can be of interest such as *iteration* (i.e. a event repeats several times), *interruption* (i.e. an event is stopped by another event), *resumption* (i.e. an event continues after being interrupted by another event), *fork* (i.e. an event starts another event) and *join* (i.e. two events merge into only one event). The spatial relations are assimilated to the relations between the physical objects involved in the given events.

While describing a composite event, it is often useful to define an optional sub-event. The notion of optional can be quantified using a coefficient ranging from 0 to 1 (0 indicates that the sub-event is necessary and 1 indicates is completely optional). The gradation of an optional event can be estimated by analyzing real world observations (e.g. learning from every day event occurrences). The optional notion can be used to describe several composite events (with/without optional sub-events).

3.3 Temporal Scenario Representation

We focus on video events, thus time is the most interesting point to study. The next section shows how we represent time for video events.

3.3.1 Time Representation

Time Elements

Time was represented by primitives of intervals by Allen (1981). Allen also proved that the satisfaction of constraints based on this representation is NP-complete [Allen, 1984]. This is due to a large number of combinations of intervals to be analyzed.

To reduce the complexity of the constraint satisfaction problem, Ghallab and Mounir-Alaoui (1989) presented a time representation based on time points. An interval is implicitly represented by two time points corresponding to its bounds. This representation can avoid an NP-complete algorithm for solving the corresponding CSP (Constraint Satisfaction Problem), however, the representation capacity is also reduced (e.g. the representation is less intuitive).

For Automatic Video Interpretation, we propose a representation of time based on these previously presented approaches. In other words, a combination of time point representation and time interval representation is chosen for this purpose.

Let $\mathscr{T}$ the set of time elements:

$\mathscr{T}_p \overset{def}{\cong}$ the set of time points represented by an integer,

$\mathscr{T}_i \overset{def}{\cong}$ the set of time intervals,

$\mathscr{T} \overset{def}{\cong} \mathscr{T}_p \cup \mathscr{T}_i$

For a time point $t \in \mathscr{T}_p$:

$t = 0$: is the starting instant of interest,

$t < 0$: is instant of $|t|$ time units (e.g. frames) before the starting instant,

$t > 0$: is instant of t time units (e.g. frames) after the starting instant.

For an interval $I = [I^-, I^+] \in \mathscr{T}_i$:

where $I^-, I^+ \in \mathscr{T}_p$ are two time points corresponding to the bounds of I and $I^- \leq I^+$,

$|I| \overset{def}{\cong} I^+ - I^-$, is the *duration* of I,

we note $\varnothing$ the empty interval,

the non-bounded intervals (e.g. $[10, \infty[$ and $]-\infty, 100]$) are also used in the interval set,

an interval I is a *positive* interval if $I^- \geq 0$.

For an object O, let $\tau(O)$ be the time interval during which O is present in the scene. If O is a static object of the scene (e.g. a table, a chair), $\tau(O)$ is an interval [0, *the current instant*]. We also denote $[O^-, O^+]$ the time interval $\tau(O)$.

Time Operators

The arithmetic operators {+, -, min, max } are used on time point set $\mathscr{T}_p$ with the same semantics as the original operators.

The arithmetic operators {=, <, >} are also used to compare two time points t_1 and t_2 with following temporal semantics:

$t_1 = t_2$: these two time points are identical

$t_1 < t_2$: the instant t_1 is before the instant t_2

$t_1 > t_2$: the instant t_1 is after the instant t_2

We can demonstrate that $\mathscr{T}_p$ is totally ordered by "<", because the set of integers is totally ordered by "<".

We apply all the operators of Allen's interval algebra [Allen, 1981] on the interval set $\mathscr{T}_i$. Specially, the operators "equal" and "after" are respectively denoted by "=" and ">". Moreover, we modify the following Ghallab & Mounir-Alaoui's operators (1989) on $\mathscr{T}_i$; for all $I, I_1, I_2 \in \mathscr{T}_i$, we define:

conjunction ($\cap$) of two intervals:
$$I \cap \varnothing \overset{def}{\cong} \varnothing$$
$$\text{let } t_1 = \max(I_1^{[}, I_2^{[}),\ t_2 = \min(I_1^{]}, I_2^{]})$$
$$I_1 \cap I_2 \overset{def}{\cong} \begin{cases} [t_1, t_2], & if \quad t_1 \le t_2 \\ \varnothing, & if \quad t_1 > t_2 \end{cases}$$

composition ($\oplus$) of two intervals:
$$I \oplus \varnothing \overset{def}{\cong} \varnothing$$
$$I_1 \oplus I_2 \overset{def}{\cong} [I_1^{[} + I_2^{[},\ I_1^{]} + I_2^{]}]$$

inclusion ($\subset$):
$$I \subset \varnothing \overset{def}{\cong} \text{FALSE}$$
$$\varnothing \subset I \overset{def}{\cong} \text{FALSE} \quad (*)$$
$$I_1 \subset I_2 \overset{def}{\cong} (I_2^{[} \le I_1^{[}) \wedge (I_1^{]} \le I_2^{]})$$

Dousson (1994) showed that $\oplus$ and $\cap$ are commutative, $\oplus$ is distributive on $\cap$ and $\subset$ is preserved by $\oplus$; for all $I_1, I_2, I_3 \in \mathscr{T}_i$ we have:

$$I_1 \oplus (I_2 \cap I_3) = (I_1 \oplus I_2) \cap (I_1 \oplus I_3)$$
$$I_1 \subset I_2 \Rightarrow (I_1 \oplus I_3) \subset (I_2 \oplus I_3)$$

We propose the following operators for relations between all time points $t, t_1, t_2 \in \mathscr{T}_p$ and all time intervals $I, I_1, I_2 \in \mathscr{T}_i$:

X-before, denoted <:

$$(\mathscr{T}_p \times \mathscr{T}_i) \rightarrow \text{Boolean}: \quad t < I \overset{def}{\cong} t < I^{]}$$

(*) The empty interval is not used for video event representation.

$(\mathcal{T}_i \times \mathcal{T}_p) \rightarrow \text{Boolean}: \quad I < t \quad \overset{def}{\cong} \quad I^{]} < t$

$(\mathcal{T}_i \times \mathcal{T}_i) \rightarrow \text{Boolean}: \quad I_1 < I_2 \quad \overset{def}{\cong} \quad I_1^{]} < I_2^{]}$

X-finish, denoted —|:

$(\mathcal{T}_p \times \mathcal{T}_i) \rightarrow \text{Boolean}: \quad t —| I \quad \overset{def}{\cong} \quad t = I^{]}$

$(\mathcal{T}_i \times \mathcal{T}_i) \rightarrow \text{Boolean}: \quad I_1 —| I_2 \quad \overset{def}{\cong} \quad I_1^{]} = I_2^{]}$

X-before-finish, denoted ≤:

$(\mathcal{T}_p \times \mathcal{T}_p) \rightarrow \text{Boolean}: \quad t_1 \leq t_2 \quad \overset{def}{\cong} \quad t_1$ is before or identical to t_2

$(\mathcal{T}_p \times \mathcal{T}_i) \rightarrow \text{Boolean}: \quad t \leq I \quad \overset{def}{\cong} \quad t \leq I^{]}$

$(\mathcal{T}_i \times \mathcal{T}_i) \rightarrow \text{Boolean}: \quad I_1 \leq I_2 \quad \overset{def}{\cong} \quad I_1^{]} \leq I_2^{]}$

The time is represented by a concrete set $\mathcal{T}$. We can demonstrate that $\mathcal{T}$ is totally ordered by the operator "≤", because $\mathcal{T}_p$ is totally ordered by "≤".

Section 3.2.2 has presented four concepts representing video events which are primitive state, composite state, primitive event and composite event. Those are the terms that can be used by experts to describe video events. In the following sections, we focus on the composition of video events and a description language to represent video events. Based on the composition of video events, we first distinguish two types of video events as shown in the next section.

3.3.2 Two Scenario Types

For the Automatic Video Interpretation, we use the term **scenario** as the technical term expressing all video events. Base on the composition of scenarios, we distinguish two types of scenarios: ***elementary*** and ***composed*** scenarios. An *elementary* scenario does not contain any sub-scenario (e.g. "person *p* is close to a machine *m*"). Whereas, a *composed* scenario is composed of at least one sub-scenario (e.g. "bank attack" scenario).

An elementary scenario is a primitive state. It can be calculated directly from geometrical/physical attributes of physical objects (e.g. the 3D position in the scene of an individual, the speed of an individual, the status of a door). Elementary scenarios have an important role in a scenario knowledge base; they are the kernel of scenario knowledge bases. In other words, they are primitive words and sufficiently generic to be used in knowledge bases for different automatic video interpretation applications.

To model a given scenario M, we use variables to represent physical objects involved in M and its composition. The next section details variable types used in our scenario representation.

3.3.3 Variables

In our scenario knowledge representation, we use two classes of variables: ***temporal*** and ***non-temporal*** variables. A variable is a temporal variable if its value is a scenario instance recognized during a time interval (e.g. an instance of scenario "*p* is inside zone *z*" during a time interval [10, 40]). A variable is a non-temporal variable if its value is a physical object (e.g. a table *t*, a person *p*). The type of a variable *v*, denoted *type*(*v*), is the type of objects that can be attached to *v* (e.g. Person, Zone).

Let $\mathcal{V}_N$ = the set of non-temporal variables,

$\mathcal{V}_T$ = the set of temporal variables,

$\mathcal{V} = \mathcal{V}_N \cup \mathcal{V}_T$ = the set of all variables,

$\mathcal{C}(\mathcal{V})$ = the set of all sub sets of $\mathcal{V}$.

To access the value held by a variable:

$v \in \mathcal{V}$, let *value*(*v*) be the value of *v* and we also define:

$\tau(v)$ is the time interval of *value*(*v*),

dom(*v*) is the domain of *v* (a set of objects that can be attached to *v*).

For a set of variables $V \in \mathcal{C}(\mathcal{V})$, we define:

$$value(V) \overset{def}{\cong} \{value(v), v \in V\},$$

$$dom(V) \overset{def}{\cong} \{dom(v), v \in V\},$$

$$type(V) \overset{def}{\cong} \{type(v), v \in V\}.$$

For all $v \in \mathcal{V}$, we denote $v^{[}$ and $v^{]}$ respectively the lower and the higher bounds of $\tau(v)$. We also extend all operators on $\mathcal{T}$ to $\mathcal{V}$ by applying these operators on $\tau(v)$ for all $v \in \mathcal{V}$.

Section 3.2.3 has shown the relation types between concepts for video events. The next section presents the modeling of these relations.

3.3.4 Constraints

Definition 3.1 (*constraint*): A constraint *k* is defined as a Boolean function on $\mathcal{C}(\mathcal{V})$

$$k : \mathcal{C}(\mathcal{V}) \rightarrow \text{Boolean}$$

If a constraint *k* is defined with a temporal variable, then *k* is called *temporal constraint*, otherwise, *k* is called *non-temporal constraint*.

Let $\mathcal{K}_N$ = the set of all non-temporal constraints,

$\mathcal{K}_T$ = the set of all temporal constraints,

$\mathcal{K} = \mathcal{K}_N \cup \mathcal{K}_T$ = the set of all constraints,

$\mathcal{C}(\mathcal{K})$ = the set of all sub sets of $\mathcal{K}$.

For a variable *v* and a constraint *k*, we consider that *v* is included in *k* and denote $v \in k$ if *k* is checked by *v*.

Definition 3.2 (*constraint degree*):

$V = \{v_1,\ldots, v_n\}$, is an ordered (e.g. temporally) set of variables

$k : V \rightarrow$ Boolean, is a constraint on V

We define the *degree* of *k*, denoted *deg*(*k*), as the following:

i) $deg(k) = n$, if $v_n \in k$

ii) $deg(k) = i \in [1, n\text{-}1]$, if $v_i \in k \wedge v_j \notin k, \forall j > i$.

Example:

$V = \{v_1,\ldots, v_{10}\}$, is an ordered set of ten variables,

k_1 and k_2 are two constraints defined on V:

$k_1 \overset{def}{\cong} (v_2$ **X-finish** $v_4)$

$k_2 \overset{def}{\cong} (duration$ **of** $v_3 \geq 10)$

We have:

$deg(k_1) = 4$ and $deg(k_2) = 3$.

By using the video event ontology presented in section [3.2], time representation [3.3.1], scenario types [3.3.2], variables [3.3.3] and constraint modeling [3.3.4], we proposed a hierarchical model of scenarios that is shown in the next section.

3.3.5 Hierarchical Model of Scenario

Our objective is to propose a generic scenario model that is capable to represent all types of scenarios used for Automatic Video Interpretation.

To model a scenario S, we distinguish the set of **physical objects** (e.g. persons, tables) involved in S, a set of **sub-scenarios** composing S (i.e. *components* of S) and a set of **constraints** on these physical objects and these components.

A *scenario instance* is a scenario occurrence (with its physical objects and its components) during a time interval.

Let $\mathscr{D}$ be the set of all tasks pre-defined by experts that can be executed in the observed environment (e.g. to take a decision, to generate in natural language a text describing the observed environment) and let $\mathscr{C}(\mathscr{D})$ be the set of all sub sets of $\mathscr{D}$.

We denote $\xi(M)$, the scenario modeled by a scenario model M. We also denote $\mu(S)$, the scenario model of a given scenario S.

We define the model M of a scenario S as an element of:

$(\mathscr{C}(\mathscr{V}_N) \times \mathscr{C}(\mathscr{V}_T) \times \mathscr{C}(\mathscr{V}_T) \times \mathscr{C}(\mathscr{K}) \times \mathscr{C}(\mathscr{D}))$

M =

(*physical-objects, components, forbidden-scenarios, constraints, decisions*)

Where:

physical-objects, denoted $\varphi(M)$, is a non-empty set of non-temporal variables called *physical object variables*. The values of these variables will correspond to the physical objects involved in S.

For all temporal variable *v*, we also define:

$\varphi(v)$ is a set of physical object variables and their values will correspond to the set of physical objects involved by *value*(*v*).

components, denoted $\sigma(M)$, is a set of temporal variables which values will correspond to the sub-scenarios composing S. If M_e is an *elementary* scenario model, $\sigma(M_e) = \varnothing$, because an elementary scenario does not contain any sub-scenario [section 3.3.2].

We define:

$$\varphi_{sub}(M) \overset{def}{\cong} \bigcup_{v \in \sigma(M)} \varphi(v)$$

We have $\varphi(M) \subset \varphi_{sub}(M)$, because all physical objects involved in a scenario S have to be involved in at least one sub-scenario composing S.

We also define:

$$\mu_{sub}(M) \overset{def}{\cong} \{\mu(value(v)) \mid v \in \sigma(M)\}$$

is the set of all scenario models used to define M.

forbidden-scenarios, denoted $\sigma_F(M)$, is a set of temporal variables corresponding to all scenarios that are not allowed to be recognized during the recognition of S. A variable $v \in \sigma_F(M)$ is called *forbidden variable*.

We also define:

$$\varphi_F(M) \overset{def}{\cong} \bigcup_{s \in \sigma_F(M)} \varphi(s) - \varphi_{sub}(M)$$

each $v \in \varphi_F(M)$ is called *forbidden physical object variable* and is also a *forbidden variable*.

constraints, denoted $\kappa(M)$ is a non-empty set of constraints to be verified for the recognition of S. We also classify these constraints into three classes of constraints. $\kappa_T(M)$ is the set of all ***temporal constraints*** of $\kappa(M)$; a temporal constraint involves at least one temporal variable of $\sigma(M)$ and does not involve any forbidden variable. The temporal constraints of a scenario are combined by the logical "*and*" operator. $\kappa_F(M)$ is the set of ***forbidden constraints*** of $\kappa(M)$; a forbidden constraint involves at least one forbidden variable. These forbidden constraints will be used (by the recognition process) to decide whether the occurrence of a scenario does match the scenario model M. $\kappa_N(M)$ is the set of ***non-temporal constraints***; a non-temporal constraint involves only $\varphi(M)$ and the contextual objects of the observed environment.

We have:

$\kappa_T(M) \cap \kappa_F(M) = \varnothing,$

$\kappa_T(M) \cap \kappa_N(M) = \varnothing,$

$\kappa_N(M) \cap \kappa_F(M) = \varnothing,$

$\kappa(M) = \kappa_T(M) \cup \kappa_N(M) \cup \kappa_F(M).$

decisions, denoted $\delta(M)$, is a optional part of a scenario and corresponds to a set of decisions to be executed if S is recognized. A *decision* is a task predefined by experts that can be executed in the observed environment.

Figure 3.1 shows the proposed generic hierarchical model of temporal scenarios composed of five parts corresponding to three set of variables (i.e. non-temporal, temporal and forbidden variables), a set of constraints and a set of decisions.

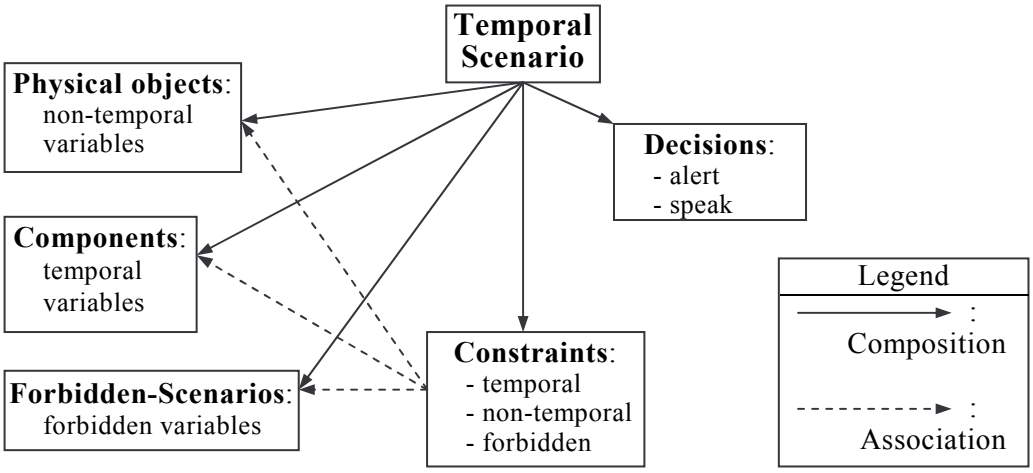


Figure 3.1. A temporal scenario M is composed of five parts: a set of physical object variables corresponding to physical objects involved in M, a set of temporal variables (components) corresponding to sub-scenarios composing M, a et of forbidden variables corresponding to scenarios that are not allowed to occur during the recognition of M, a set of constraints on these three sets of variables and a set of tasks (decisions) to be executed when M has been recognized. This model is **hierarchical** because M is modeled by its sub-scenarios expressed through temporal variables (i.e. Components).

Figure 3.2 shows an example of a “Bank Attack” scenario (the shown images are taken in a bank agency) and Figure 3.3 shows an example of a hierarchical model of this scenario. The scenario involves two physical objects with their roles "employee" and "robber". First, the employee is at his/her position behind the counter. Second, the robber enters the bank agency while the employee is still at his/her position. Third, the robber moves to the front of the counter while the employee is still at his/her position. Finally, both of them arrive to the safe door. If the scenario is recognized, an alert meaning "Bank Attack" is triggered.

Figure 3.2. Four steps of a "Bank Attack" scenario: (1) at time t_1, the employee is at his/her position behind the counter, (2) at time t_2, the robber enters by the entrance and the employee is still at his/her position, (3) at time t_3, the robber moves to the front of the counter and the employee is still at his/her position and (4) at time t_4, both of them arrive to the safe door.

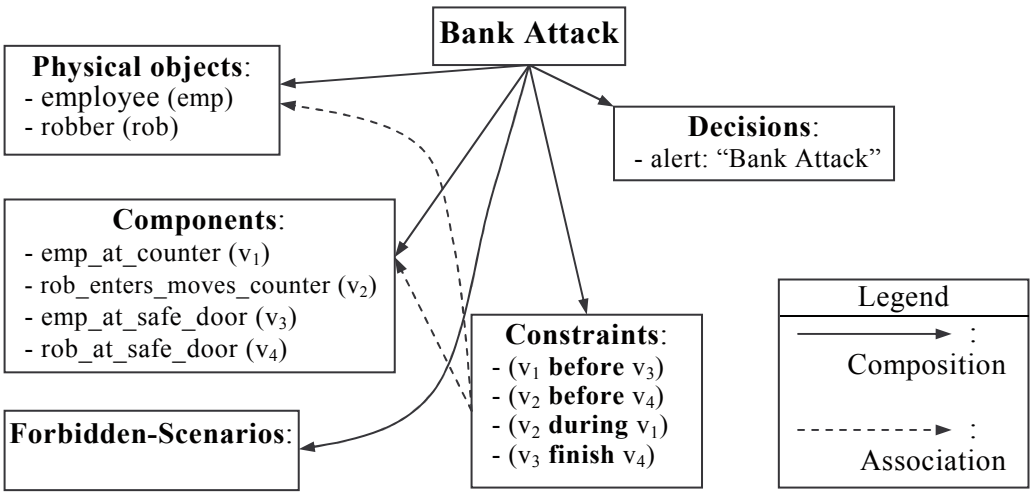


Figure 3.3. Hierarchical model of the "Bank Attack" scenario [Figure 3.2]. This scenario is composed of five parts: a set of physical object two variables corresponding to an employee and a robber, a set of temporal variables (components) corresponding to four sub-scenarios composing the scenario, an empty set of forbidden variables, a set of constraints ("before", "during" and "finish" [Allen, 1981]) on these three sets of variables and a task (decision) triggering an alert meaning "Bank Attack" to be executed when the scenario has been recognized.

We note $\mathcal{M}_e$ the set of all **elementary** scenario models, $\mathcal{M}_c$ the set of all **composed** scenario models and $\mathcal{M} = \mathcal{M}_c \cup \mathcal{M}_e$ the set of all **scenario models**.

Definition 3.3 (*inclusive relation*):

$M_1, M_2 \in \mathcal{M}$, M_1 is included in M_2, denoted $M_1 \subseteq M_2$:

$$M_1 \subseteq M_2 \overset{def}{\cong} (M_1 = M_2) \vee (\exists M \in \mu_{sub}(M_2) \mid M_1 \subseteq M), \quad (*)$$

$$M_1 \subset M_2 \overset{def}{\cong} (M_1 \subseteq M_2) \wedge (M_1 \neq M_2).$$

By a recurrent method, we can demonstrate that $\subseteq$ is transitive:

$$(M_1 \subseteq M_2) \wedge (M_2 \subseteq M_3) \Rightarrow (M_1 \subseteq M_3).$$

We can also demonstrate that the set of scenario models $\mathcal{M}$ is partially ordered by the *inclusive relation* of scenarios. Because, for two scenario models $M_1, M_2 \in \mathcal{M}$, if M_1 is not used to define M_2 and M_2 is not used to define M_1, thus, M_1 and M_2 are not related by $\subseteq$.

Definition 3.4: A scenario model M is ***coherent*** if and only if:

1) $\kappa(M)$ is a coherent set of constraints,
2) $(M' \subset M) \Rightarrow (M \not\subset M')$, called ***inclusive coherence***,
3) $\forall M' \in \sigma(M)$, M' is coherent.

We denote $\mathcal{M}_{co}$ the set of all Coherent Scenario Models and $\mathcal{C}(\mathcal{M}_{co})$ the set of all sub sets of $\mathcal{M}_{co}$. We address in chapter 4 and chapter 6 how to verify whether a scenario model is coherent.

For a scenario instance *s*, we also denote:

$\varphi(s)$ = the set of **physical objects** involved in *s*,
$\sigma(s)$ = the set of **sub-scenario instances** composing *s*,
$\delta(s)$ = the set of **decisions** that are taken when *s* has been recognized.

Definition 3.5: Two scenario instances s_1 and s_2 are of the same type, denoted $s_1 \approx s_2$, if they (a) are defined by the same scenario model and (b) involve the same physical-objects:

$$s_1 \approx s_2 \overset{def}{\cong} (\mu(s_1) = \mu(s_2)) \wedge (\varphi(s_1) = \varphi(s_2)). \quad (*)$$

Definition 3.6: Two scenario instances s_1 and s_2 are equal:

$$s_1 = s_2 \overset{def}{\cong} (s_1 \approx s_2) \wedge (\sigma(s_1) = \sigma(s_2)) \wedge (\tau(s_1) = \tau(s_2)). \quad (**)$$

(*) $\vee$: logical operator "or".

(*) μ, φ: page 45

(**) σ: page 46 τ: page 41

Definition 3.7:

A scenario instance s_1 is included in a scenario instance s_2, denoted $s_1 \subseteq s_2$:

$$s_1 \subseteq s_2 \overset{def}{\cong} (s_1 = s_2) \vee (\exists s \in \sigma(s_2) \mid s_1 \subseteq s),$$

$$s_1 \subset s_1 \overset{def}{\cong} (s_1 \subseteq s_2) \wedge (s_1 \neq s_2).$$

A composed scenario is principally viewed as a set of sub-scenario instances recognized during a time interval. The temporal relations between these sub-scenarios are expressed through temporal constraints. As shown in section 3.3.1, the set of time elements $\mathscr{T}$ is totally ordered by the operator "≤". Thus, a composed scenario can be viewed as a set of scenario instances ordered by their ending time.

The proposed generic scenario model allows experts (e.g. the end-users of video interpretation systems) to express in an intuitive way their scenarios of interest. Several experts of respectively two European projects (ADVISOR metro station surveillance and AVITRACK for apron monitoring) and two French projects (SAMSIT for inside train surveillance and CASSIOPEE for bank monitoring) have given positive feedbacks while using this generic scenario model. The scenario models defined by experts are stored in a *scenario knowledge base* which is presented in the next section.

3.3.6 Scenario Knowledge Base

A Scenario Knowledge Base is a set of scenario models pre-defined by experts to be used as a-priori knowledge of an Automatic Video Interpretation System.

The Scenario Knowledge Base of our Automatic Video Interpretation framework is represented by a graph. The nodes of this graph correspond to pre-defined scenario models. The edges of this graph correspond to inclusive relations between scenario models. For two scenario models M_1 and M_2, if $M_2 \subset M_1$, then it exists an arc from node M_1 to node M_2. Thus, in this graph, all nodes that have not any exiting arc correspond to elementary scenario models, and the other nodes correspond to composed scenario models.

Figure 3.4 shows a part of a Scenario Knowledge Base containing 21 scenario models. In this Scenario Knowledge Base, the scenario model "bank attack one commercial & one robber" is defined using six other scenario models: "inside zone", "changes zone", "tailgating", "moves close to", "opens door" and "holding gun".

Definition 3.8: A Scenario Knowledge Base $\mathscr{S}_{KB}$ is **coherent** if it does not contain any incoherent scenario model:

$\mathscr{S}_{KB}$ is coherent $\overset{def}{\cong}$ $\mathscr{S}_{KB} \in \mathscr{C}(\mathscr{M}_{co})$.

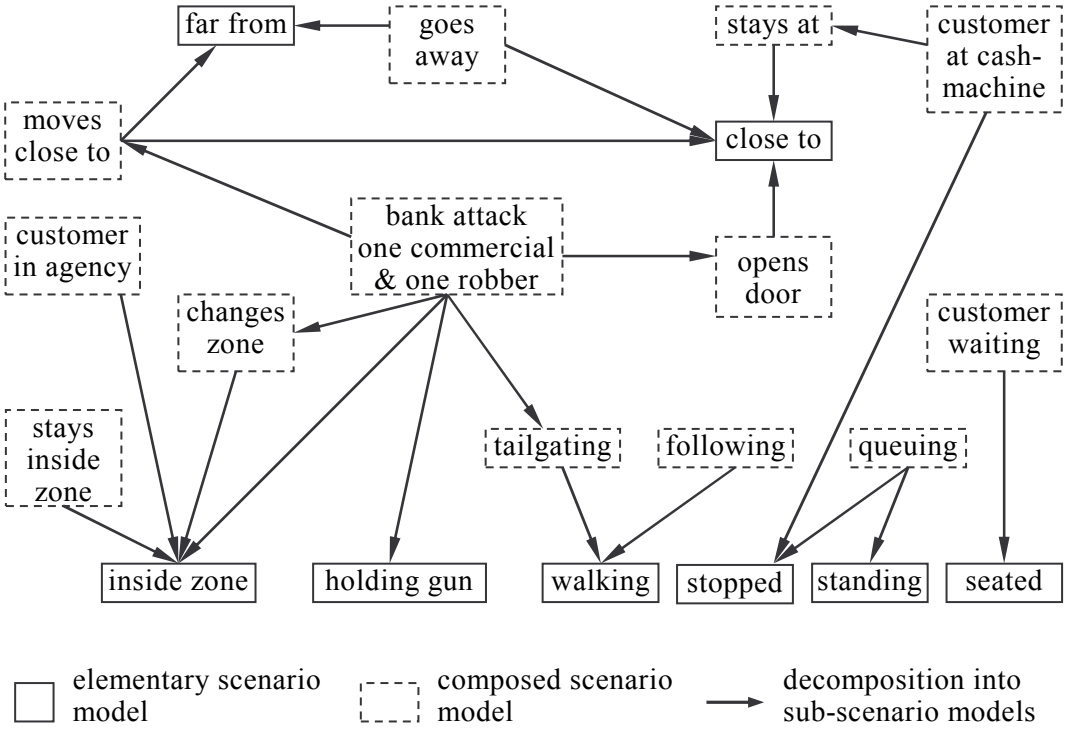


Figure 3.4. This figure shows a part of the scenario knowledge based used in the bank monitoring application. There are six elementary scenario models and fifteen composed scenario models. The "bank attack one commercial & one robber" is the most complex scenario model in this knowledge base. This scenario model is defined using six other scenario models: "inside zone", "changes zone", "tailgating", "moves close to", "opens door" and "holding gun".

The scenario knowledge base of an Automatic Video Interpretation System is an important part. This knowledge base has to contain only coherent scenario models. Thus, the detection of incoherent scenario models of a Scenario Knowledge Base is an important task for an Automatic Video Interpretation System. This issue is addressed in chapter 6.

Base on the proposed video event ontology [3.2] and scenario model [3.3.5], we proposed in the next section a **Scenario Description Language** to help experts to describe easily (i.e. in a declarative, intuitive, clear, sufficient, flexible way) their private scenarios.

3.4 Scenario Description Language

Our goal is to make explicit all the knowledge necessary for the system to be able to recognize scenarios occurring in the scene. The description of this knowledge has to be declarative and intuitive (in natural terms), so that the experts of the application domain can easily define and modify it. A scenario is represented based on the proposed hierarchical scenario model presented in section 3.3.5.

3.4.1 Keywords

A set of keywords is used to represent scenario knowledge for Automatic Video Interpretation. The keywords have for objective to start a description or declare the type of a variable.

We use the following keywords: **primitive-state**, **composite-state**, **primitive-event**, **composite-event**, **single-agent-event**, **multi-agent-event**, **State**, **Event**, **Scenario** to describe the type of scenarios. For a scenario, a *name* is given as the first attribute.

We also use other keywords **physical-objects**, **components**, **forbidden-scenarios**, **constraints** and **decisions** to declare the corresponding sets of variables, constraints and decisions.

Figure 3.5 shows the model of an elementary scenario (corresponding to a primitive state) called "inside_zone" expressing the status of a person being inside a zone. This scenario involves two physical objects, a person *p* and a zone *z*. There is only one non-temporal constraint to verify whether *p* is geometrically inside *z*. The operator "***in***" is a pre-defined spatial constraint involving two physical objects o_1 and o_2 to verify whether o_1 is geometrically inside o_2. The evaluation of this constraint is based on geometrical attribute calculations.

```
primitive-state(inside_zone,
  physical-objects(p : Person, z : Zone)
  constraints((p in z)) )
```

Figure 3.5. A representation of the "inside_zone" elementary scenario (corresponding to a state) to model the status of a person *p* being geometrically inside a zone *z*.

Similarly as the scenario model "inside_zone", another scenario (also corresponding to a primitive state) "close_to" is defined based on calculations of geometrical attributes of scene objects. "close_to" is also an elementary scenario involving two actors, a person *p* and a piece of equipment *e*. *p* is considered as close to *e* if the Euclidean distance between them is smaller than a pre-defined threshold *close_distance*.

```
primitive-state(close_to,
  physical-object(p : Person, e : Equipment)
  constraints((distance(p, e) ≤ close_distance)) )
```

Figure 3.6. A representation of the "close_to" scenario to represent the state of a person *p* that is close to an equipment *e*.

Other keywords are used to represent *variable types* (e.g. **Person**, **Zone**, **Equipment**), to define a user-defined constraint,… All keywords of the proposed language are defined and used as primitives with their semantics and cannot be redefined.

3.4.2 Temporal Constraints

Constraints are an essential component to model a scenario. Specially, we focus on temporal scenarios composed of sequential/parallel sub-scenarios, thus, temporal constraints are the most interesting relations to be studied.

Qualitative Symbolical Temporal Constraints

We are using Allen's interval operators (e.g. before, during) and also quantitative temporal constraints (e.g. s_1 has to finish 10 instants before the start of s_2) to represent temporal relations between the sub-scenarios of a composed scenario.

Figure 3.7 shows an example of using symbolical temporal constraints to model a temporal scenario. The modeled scenario (corresponding to an event) "**changes_zone**" expresses an event where a person p moves from a zone z_1 to another zone z_2. This scenario is composed of two sub-scenarios s_1 and s_2 corresponding to two steps of the represented scenario: p is first inside the zone z_1 and then z_2. The scenario model of s_1 and s_2 (i.e. "inside_zone") is also pre-defined by the expert. There is also a symbolical temporal constraint to express the sequence of s_1 and s_2: (s_1 ***before*** s_2).

Allen's interval algebra operators are well known as an efficient (i.e. clear) way to represent qualitative temporal relations between time intervals. The main advantage of using these operators is to make the representation clear and intuitive. However, they cannot represent precise temporal relations between scenarios, e.g. “person p_1 enters the room r ***10 minutes before*** person p_2 does”. We explain this temporal constraint type in the next section.

```
primitive-event(changes_zone,
  physical-objects(p : Person, z1 : Zone, z2 : Zone)
  components(
     (s1 : primitive-state inside_zone(p, z1))
     (s2 : primitive-state inside_zone(p, z2)) )
  constraints((s1 before s2) )  )
```

Figure 3.7. A description of the "changes_zone" event.

Quantitative Numerical Temporal Constraints

The scenario model represented in Figure 3.7 can be recognized even through the delay between s_1 and s_2 is long (e.g. 10000 instants) and it will be also recognized during a time interval as long as the duration of s_2. We propose to use quantitative temporal constraints to avoid this problem as shown in Figure 3.8.

```
primitive-event(changes_zone,
  physical-objects(p : Person, z1 : Zone, z2 : Zone)
  components(
     (s1 : primitive-state inside_zone(p, z1))
     (s2 : primitive-state inside_zone(p, z2)) )
  constraints(    (s1 before s2)
                  (s2[ - s1] ≤ max_delay)
                  (duration of s2 ≤ 2) )  )
```

Figure 3.8. A representation of "changes_zone" event using *quantitative* temporal constraints.

The constraint "($s_2^{[} - s_1^{]} \leq max\_delay$)" limits the delay between the ending time of s_1 and the starting time of s_2 by a pre-defined threshold *max_delay* (e.g. 20 instants). Moreover, the next constraint "(*duration* ***of*** $s_2 \leq 2$)" limits the recognition of this event during 2 instants instead of all the duration of s_2.

Quantitative temporal constraints show an efficient way to express numerical temporal relations between sub-scenarios of a composed scenario. They can be expressed by mathematical formulas, thus they can be easily used. However, they limit the intuitiveness of the representation, because qualitative temporal relations are replaced by constraints on the bounds of time intervals.

3.4.3 Number of Occurrences

To model temporal scenarios, we do not only focus on the occurrence of scenarios but also their number of occurrences during a time interval. For example, to model a "vandalism against a ticket vending machine *m*" scenario, we focus on a sequence of activities where there is a person *p* who moves close to *m* and then moves away from *m*. Such a sequence of activities can be repeated several times. To solve this issue, we need to count the number of occurrences of the activity sequence "*p* moves close to *m* and then *p* moves away from *m*".

To model the occurrence number, we propose in our language a constraint *CountOccurrences* that counts and verifies whether a scenario is being successively recognized a given number of times. This feature enables the language to express more precise real world knowledge. Figure 3.9 shows an example of using this constraint to model a situation where a person moves three times close to a machine.

```
primitive-event(approaches_machine_3_times,
  physical-objects(p : Person, m : Equipment)
  components((s : primitive-state close_to(p, m)) )
  constraints( (CountOccurrences(s, 3)) )
```

Figure 3.9. This example shows the utilization of “CountOccurrences” constraint. A person *p* approaches a machine *m* three times if he/she is detected close to *m* in three successive time intervals.

To enable the flexibility of the proposed language, we integrate in the proposed language features enabling experts to define their private constraints, object classes or add their private attributes to a pre-defined class. These features are presented in the next section.

3.4.4 User-Defined Features

The proposed language allows experts to define their own constraints in the form of Boolean functions. A function is defined as a set of constraints combined by the Boolean operator "*and*".

```
Function IsOpenDoor(d : Door)
(    ((State of d) = Open)    )
```

Figure 3.10. Example of a user-defined constraint. The evaluation of this constraint is based on physical attributes (of the door d) that are given by a vision routine.

Figure 3.10 shows an example of the definition of a user-defined constraint. The defined constraint "*IsOpenDoor*" involves an object *d* of type "**Door**" and verifies whether the door *d* is open. The evaluation of this function is based on physical attributes (of the door d) that are given by a vision routine.

Figure 3.11 shows a definition of a complex temporal constraint involving a list of scenario instances to verify whether this list is a temporal sequence of scenario instances. A temporal sequence of scenario instances *l* is recursively defined as: (1) *l* contains only one scenario instance, or (2) the first scenario instance must be before the second scenario instance of *l* and the tail of *l* (i.e. *l* without the first element) must be a temporal sequence of scenario instances.

The capacity of the language that allows experts to define their private constraints is an efficient way to improve the flexibility of the language (e.g. experts can implement a new time ontology for representing their scenarios to adapt the language to different applications) and also to decrease the time necessary to model temporal scenarios. We also propose in this language the possibility to define a new object class or redefine a pre-defined object class to help expert modeling their scenarios more easily. The verification of correctness of a user-defined function *f* can be realized by the verification of coherency of constraints defined within *f*. To realize this, a graph-based method is used. This verification is similar to the scenario model coherency verification that we explain in chapter 6.

```
Function Sequence(l : List)
( ((NumberElements of l ≤ 1) ? TRUE :
   ((First of l before Second of l) ? Sequence(Tail of l) : FALSE)))
```

Figure 3.11. An example of the definition of a complex temporal constraint to verify whether a given list of scenario instances is a temporal sequence of scenario instances.

```
Scenario(bank_attack,
  physical-objects(emp : Person, rob : Person)
  components(
     (emp_at_pos : primitive-state inside_zone(emp, "back_counter"))
     (rob_enters : primitive-event changes_zone(rob,"entrance","counter"))
     (emp_at_safe: primitive-state inside_zone(emp, "safe"))
     (rob_at_safe: primitive-state inside_zone(rob, "safe")) )
  forbidden-scenarios(
     (any_in_bank: primitive-state inside_zone(anyP, "bank")) )
  constraints(
     (emp_at_pos  before emp_at_safe)      // (1): temporal
     (rob_enters  before rob_at_safe)      // (2): temporal
     (rob_enters  during emp_at_pos)       // (3): temporal
     (rob_at_safe finish emp_at_safe)      // (4): temporal
     (anyP ≠ emp)                          // (5): forbidden
     (anyP ≠ rob)                          // (6): forbidden
     (any_in_bank[ ≤ rob_enters[)              // (7): forbidden
     (rob_enters[ ≤ any_in_bank]) )            // (8): forbidden
  decisions(Alert("Bank Attack!"))  )
```

Figure 3.12. A description of "Bank Attack" scenario composed of four sub-scenarios.

To enable Automatic Video Interpretation systems processing scenario models represented using the proposed scenario description language, we have also developed a **parser** for this language. The parser aims at: (1) verifying whether the definitions given by the experts are syntactically correct and (2) analyzing these definitions to transform them into data structures that can be directly used by the recognition process. This parser is integrated in our Automatic Video Interpretation system.

The proposed scenario description language is currently used by respectively three European projects and one French project ADVISOR, SAMSIT, AVITRACK and CASSIOPEE for metro station surveillance, train surveillance, apron monitoring and bank monitoring. The feedbacks given by several experts of these projects show that the language is clear, flexible enough and easy to use.

3.5 Examples

This section shows a complete example of the "Bank Attack" scenario. This scenario is composed of four sub-scenarios and an alert meaning "Bank Attack!". The description shown in Figure 3.12 contains a forbidden scenario with forbidden constraints to express that there is nobody in the bank during the attack. The constraints (1)-(4) express a sequence of sub-scenarios composing the "Bank Attack" scenario. Constraints (5), (6) express that *anyP* is a person different from the two persons *emp* and *rob*. The last constraints express that the third person *anyP* is in the bank during the attack.

```
1. Physical Objects in a Metro
  Mobile Objects
     Person (p).
     Group of persons (g).
     Crowd (c).
     Metro Train (m).
     Portable Objects (o).
     Other (fire)
  Contextual Objects
     Zone (z) with different roles: Entrance_Zone, Validation_Zone,
        Exit_Zone,…
     Equipment (eq) with different sub-classes:
        Ticket_Vending_Machine, Escalator, Wall,…

2. States
  primitive-state(LyingPerson,
     physical-objects(p : Person)
     constraints( (Lying(p) is true) )
  primitive-state(Groupwidthvariation,
     physical-objects(g : Group)
     constraints( (Width(g) > significantwidthvariation) )

3. Composite Events
  composite-event(Jumping,
     physical-objects( (p: Person) )
     components(  (c1 : primitive-state Speed_increase(p))
                  (c2 : primitive-state Legs_up(p)) )
     constraints( (c1, c2)) //Sequence
  composite-event(Stays_inside_zone,
     physical-objects( (e : Person), (z : Zone) )
     components((c1 : primitive-state Inside_zone(e,z)) )
     forbidden_events( (c2 : primitive-event Exit(e,z)) )
     constraints( (c2 during c1) )
```

Figure 3.13. Several concepts extracted form the ontology for visual metro monitoring application.

3.6 Ontology Utilization

The proposed structure of ontology can be used in two ways: to describe concepts or to annotate videos with concept occurrences. Based on this ontology structure, the INRIA research team ORION have described in annex I a reduced set of concepts used in video surveillance applications. Figure 3.13 shows several concepts extracted from the ontology for visual metro monitoring application represented in annex I. We claim that it is difficult to enumerate exhaustively every situation necessary to describe events from any video even in a specific domain. For instance, many things can happen in a bank: “drinking a glass of water”, “running after a kid”, “washing the windows”. Defining such scenarios leads also to the issue of the granularity (related to shape and its evolution) of the description. Defining the scenario “washing the windows” needs an accurate vocabulary for the posture and body movement description. Indeed, washing a window implies specific arm movements.

States and primitive events listed in annex I are generic and are given at a low level of granularity involving coarse attributes of physical objects (they are not intended to describe shape properties in detail). We try to enumerate all concepts (states and primitive events) relative to position-based attributes and most concepts relative to

global appearance attributes. At the implementation level, the extraction of these states and events from videos can be done simply by using bounding box and position-based attributes. For illustration, we also try to give few concepts (states and primitive events) relative to the local appearance attributes.

This basic corpus can be refined depending on the needs. Refined concepts are more difficult to extract from videos. For example, they may need posture analysis algorithms. For example, the concept "holding an object" is perceived differently depending on the posture but also in the properties of the held object. Holding a gun is perceptually different from holding a luggage. The proposed corpus should be seen as an extendable basis. The issue is now to define tools and protocols to allow a collaborative extension of the corpus.

We have found useful to define three metrics to characterize ontology: the wealth, depth and width. The wealth indicates the number of concepts and relations in the ontology. The depth indicates the maximal level of hierarchy describing activities. The width indicates the maximal number of variations of a given activity (e.g. the variations of temporal relations). These metrics are three characteristics of a scenario knowledge base which can describe its richness and give indicators on the complexity (processing time) of the recognition process.

3.7 Conclusion on Temporal Scenario Representation

We have presented in this chapter new video event ontology and a hierarchical model of scenarios for automatic video interpretation. The new model is composed of five parts: a *physical object variable* set, a *sub-scenario* variable set, a *forbidden scenario* variable set, a *constraint* set and a *decision* set. The proposed scenario model enables experts to represent clearly and sufficiently information for their scenarios.

The time is represented (in our scenario models) through both time points and time intervals based on time representations of (Allen, 1981; Ghallab & Mounir, 1989). Several new operators were also defined to represent relations between time elements.

Based on the new scenario model and time representation, we have proposed also a language to describe scenarios. This language is currently used by respectively three European projects and one French project ADVISOR, SAMSIT, AVITRACK and CASSIOPEE for metro station surveillance, train surveillance, apron monitoring and bank monitoring. The language enables experts to represent easily and intuitively their scenarios. We have also integrated in this language advanced features that enable experts to define their own constraints, object classes and new class attributes. These features enhance the flexibility of the language by allowing experts to personalize the language to better adapt it to their applications.

We have developed a parser for the proposed language and also integrated it in our Automatic Video Interpretation system. This parser enables our Automatic Video Interpretation system processing scenario models defined using the proposed scenario description language.

Chapter 4. Overview of Temporal Scenario Recognition

This chapter presents the overview of the recognition of scenario models according to the formalism proposed in chapter 3.

4.1 Introduction

In this chapter, we focus on the performance of the scenario recognition algorithm. Thus, our goal is to propose an efficient algorithm for processing temporal constraints and preventing the combinatorial explosion of physical objects defined within scenario models. For this objective, we first present an overview of the recognition. Then, we present the control of the recognition process including a necessary and sufficient condition for a scenario model to be recognized.

4.2 Recognition Process

The scenario recognition process has to detect in **real-time** (in video cadence) which scenario is happening from a stream of observed persons tracked by a vision module at each instant. It takes also as input the a priori knowledge of the scene and the scenario models pre-defined by experts. Moreover, the recognition of scenarios is based on the two following hypotheses:

Hypothesis 1 (*good detector*): All persons are correctly detected; i.e. the attributes of all persons (e.g. their position in the scene, their height) are correctly detected.

Hypothesis 2 (*good tracker*): All persons are correctly tracked; i.e. at two successive instants, two persons having the same name (i.e. identification) correspond to the same real person.

We distinguish two types of scenarios: **elementary** and **composed** scenarios as shown in chapter 3. The recognition of each scenario type is different (we will present this point later). Figure 4.1 shows the overview of the recognition process at each instant, and Figure 4.2 shows this process in a detailed view. It first attempts to recognize all **elementary scenario models** and then the **composed scenario models** for which the last sub-scenario has been recognized at the same instant. We take the temporal order of sub-scenario variables defined within a composed scenario model. If a scenario instance S has been recognized (for both types of scenarios), S is stored to be used to recognize other composed scenarios. The storing process first attempts to merge S with an already stored scenario instance S' if it is possible. If not, S is

added (as a new element) to the set of totally recognized scenarios as described in section 4.5.

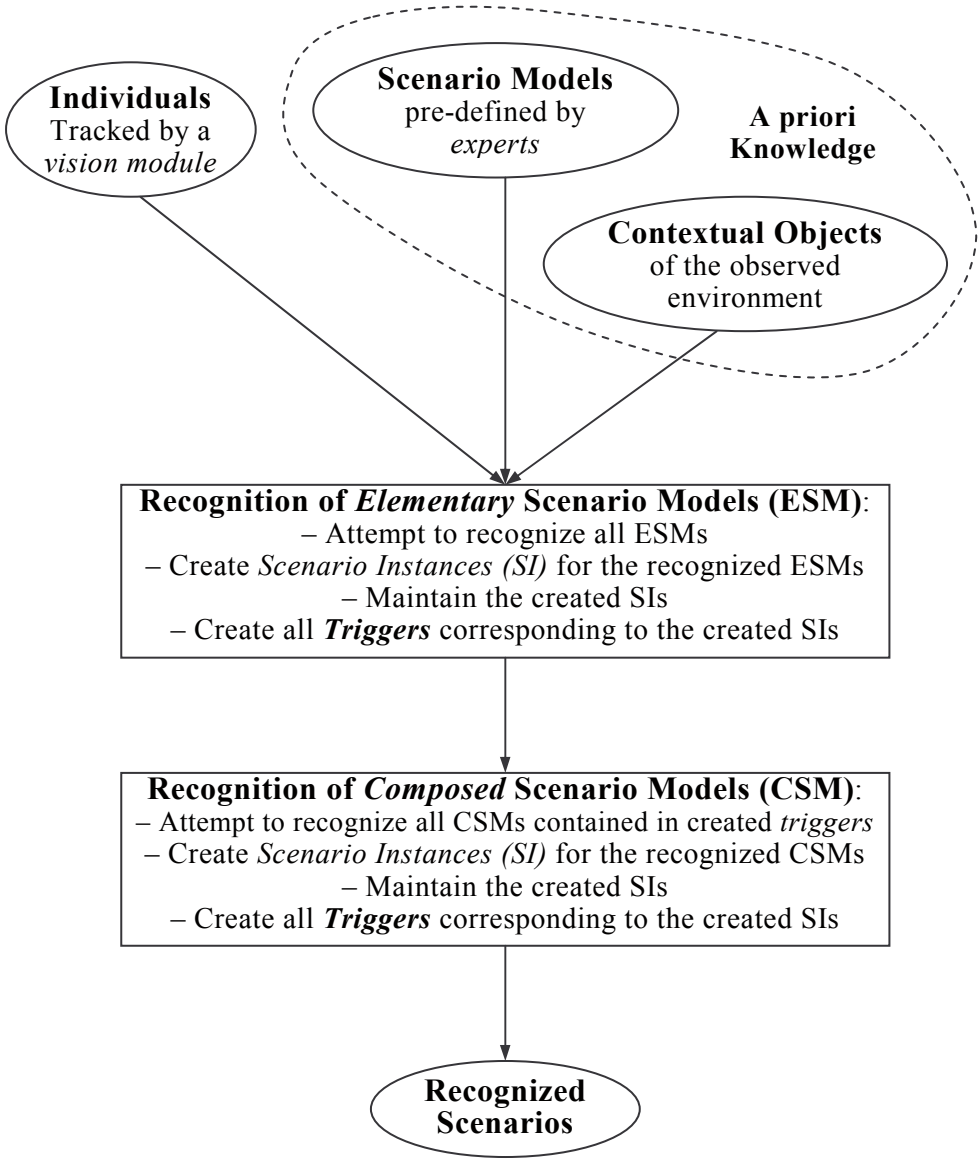


Figure 4.1. The scenario recognition process at each instant: (1) recognizes all elementary scenario models and then (2) recognizes composed scenario models triggered by the scenario instances already recognized at each instant. The recognition of composed scenario models is controlled by triggers that are explained in section 4.4.

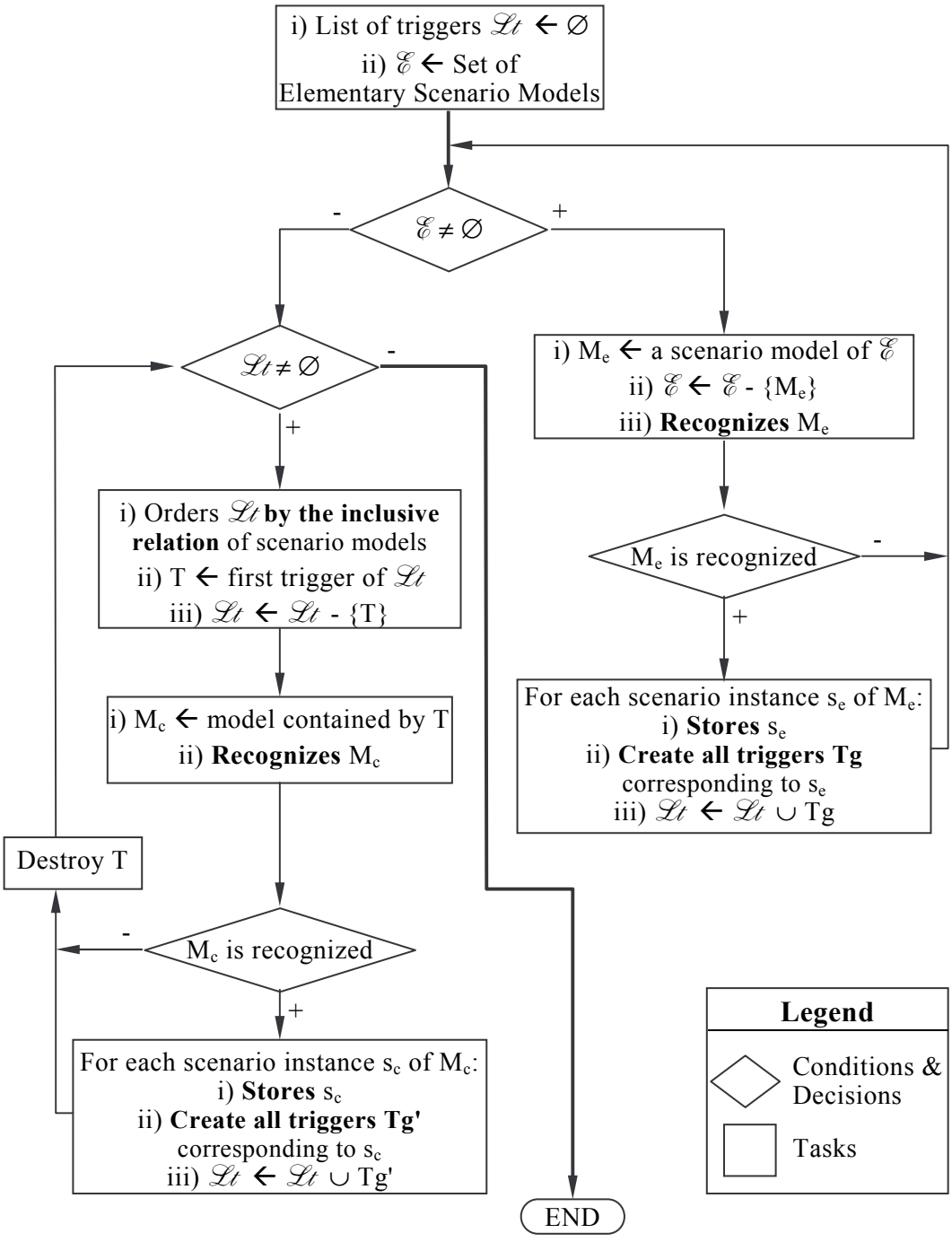


Figure 4.2. The scenario recognition process at each instant (in a **detailed** view): (1) recognizes all elementary scenario models then (2) recognizes composed scenario model contained in created triggers. Once a scenario is recognized, the recognition process stores the new recognized scenario to be used to recognize other composed scenarios, and also, creates triggers to start the recognition of several composed scenario models.

The recognition of a composed scenario is triggered when a scenario instance of its last sub-scenario model has been recognized (the sub-scenarios of a composed scenario are ordered by their ending time - as shown in chapter 3). Thanks to the triggers (section 4.4), the recognition of composed scenarios is started and controlled to be correct.

The recognition process can also be viewed as the propagation of observations (i.e. recognized elementary scenario instances) on a graph (i.e. the graph of pre-defined scenario models combined by the inclusive relation of scenario models and the temporal constraints defined within these scenario models) as shown in Figure 4.3. We take a part of the scenario knowledge base shown in chapter 3 and study the effect on the graph when a scenario instance s_i of "inside zone" (e.g. the robber is inside zone "counter") has been recognized. In the description of the four scenarios "stays_inside_zone", "customer_in_agency", "changes_zone" and "bank_attack_one_commercial_&_one_robber", the last sub-scenario of these scenarios corresponds to a scenario instance of "inside_zone". Thus, when s_i has been recognized, it triggers the recognition of these scenarios as shown by discontinuous arrows. Two of these scenarios ("changes_zone" and "customer_in_agency") are recognized. The other composed scenario models ("stays_inside_zone" and "bank_attack_one_commercial_&_one_robber") are triggered but they are not recognized because their constraints are not all satisfied.

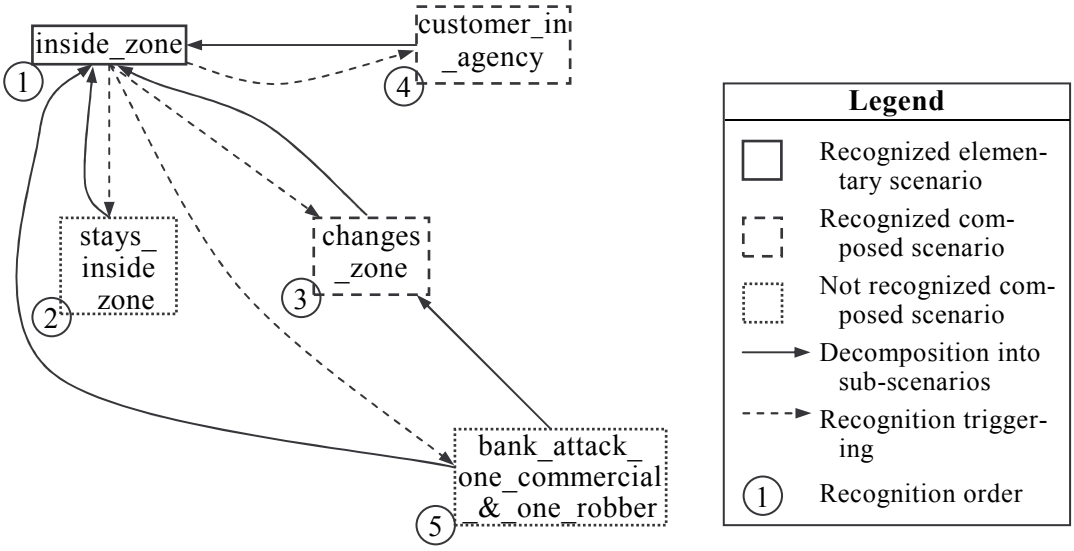


Figure 4.3. An example of the propagation of a scenario instance of the "inside_zone" scenario model on a sub-graph of the scenario model graph. The propagation process triggers the recognition of four composed scenario models ending by a scenario instance of "inside_zone". First, the elementary scenario model “inside_zone” is recognized. Then, two of four composed scenario models are recognized (3, 4). Finally, the two other composed scenario models (2, 5) are not recognized, because, their constraints are not all satisfied.

4.3 Scenario Recognition Problem

The problem of recognizing a scenario model M can be viewed as a CSP (Constraint Satisfaction Problem):

$$\mathcal{P}(M) = \{V, C, Dom\}$$

Where:

- V, denoted $vars(\mathcal{P}(M))$, is an ordered set of variables corresponding to the variables defined within M, $V = \varphi(M) \cup \sigma(M) \cup \sigma_F(M)$,
- C, denoted $\kappa(\mathcal{P}(M))$, is the set of constraints defined within M, $C = \kappa(M)$, thus C can contain temporal, non-temporal and also forbidden constraints,
- $Dom = dom(V)$, denoted $dom(\mathcal{P}(M))$, is the set of variable domains corresponding to all the possible values for physical objects or scenario instances [chapter 3].

A variable $v \in vars(\mathcal{P}(M))$ can be a non-temporal variable (i.e. physical-object variable), a temporal variable (i.e. sub-scenario variable) or a forbidden variable defined within M.

$dom(v)$ depends on the type of v and also depends temporally on the situation of the observed environment. For example, $dom(v)$ is the set of tracked persons if v is a physical-object variable of type Person, or the set of scenario instances of "inside_zone" if v is a sub-scenario variable.

The Scenario Recognition Problem is now viewed as a CSP, thus we call a scenario instance recognized during a time interval a *solution* of the CSP $\mathcal{P}(M)$. When $\mathcal{P}(M)$ has a solution (because the set of all solutions of M is dependent on the time), the scenario model M is recognized. Otherwise, M is not recognized. We denote $solutions(\mathcal{P}(M))$ the set of all solutions of $\mathcal{P}(M)$.

An elementary scenario M_e does not contain any sub-scenario (see chapter 3). Thus, a solution of $\mathcal{P}(M_e)$ is a set of physical objects corresponding to the values of non-temporal variables of $vars(\mathcal{P}(M_e))$. With M_c a composed scenario model, a solution of $\mathcal{P}(M_c)$ is a combination of an ordered set of physical objects corresponding to the values of non-temporal variables of $vars(\mathcal{P}(M_c))$ and an ordered set of scenario instances corresponding to the values of temporal variables of $vars(\mathcal{P}(M_c))$.

Let $\mathcal{P}_{sol}$ be the set of all solvable scenario recognition problems. In other words, a scenario model $M \in \mathcal{M}$ can be recognized if and only if $\mathcal{P}(M) \in \mathcal{P}_{sol}$. Thus, we have:

$$M \in \mathcal{M},\ \mathcal{P}(M) \notin \mathcal{P}_{sol} \quad \Rightarrow \quad solutions(\mathcal{P}(M)) = \varnothing.$$

We note, ***needed*(P₁, P₂)**, a predicate expressing that for solving problem P_2 we need a solution of problem P_1.

For $M_1, M_2 \in \mathcal{M}$, if $\mathbf{M_1 \subset M_2}$, we note $\mathcal{P}(\mathbf{M_1}) \subset \mathcal{P}(\mathbf{M_2})$. In this case, $\mathcal{P}(M_1)$ is called a *sub-problem* of problem $\mathcal{P}(M_2)$.

Definition 4.1: two scenario models M and M' are equivalent, denoted $M \approx M'$, if and only if they have the same recognition problem.

$$M \approx M' \quad \overset{def}{\cong} \quad \mathcal{P}(M) = \mathcal{P}(M')$$

CSP is a well-known problem of NP-complete class. Thus, for the recognition of scenarios to be real-time, we focus on reducing the complexity of the recognition algorithm by proposing a new algorithm to solve the specific CSP for temporal scenario recognition.

4.4 Triggers

A trigger is a special structure used to control the recognition of composed scenarios and exists temporarily during only one time instant –video frame– (all triggers are deleted after the recognition of all scenarios at each instant). A trigger *Tr* contains (1) a *scenario model*, denoted $\mu(Tr)$, to be recognized and (2) a *scenario instance*, denoted $\xi(Tr)$, that has been recognized. $\xi(Tr)$ will be the last scenario instance of all scenario instances of $\mu(Tr)$ that will be recognized using *Tr*.

Chapter 3 has shown a model M_c of the "changes_zone" scenario composed of two sub-scenarios and its last sub-scenario (ordered by the temporal constraints defined within M_c) has for scenario model "inside_zone". Thus, if a scenario instance s_i of "inside_zone" is recognized, then a trigger T_c containing s_i and the scenario model M_c will be created at the same instant for the recognition of "changes_zone" scenario, as shown in Figure 4.4. If a scenario instance s_c of "changes_zone" scenario is recognized by using T_c, then, s_i is the last sub-scenario instance composing s_c.

Chapter 3 has also shown a model of the "Bank_Attack" scenario composed of four sub-scenarios and its last sub-scenario is a scenario instance of the "inside_zone" scenario model. Thus, when s_i is recognized, a trigger will be also created for the recognition of the "Bank_Attack" scenario model.

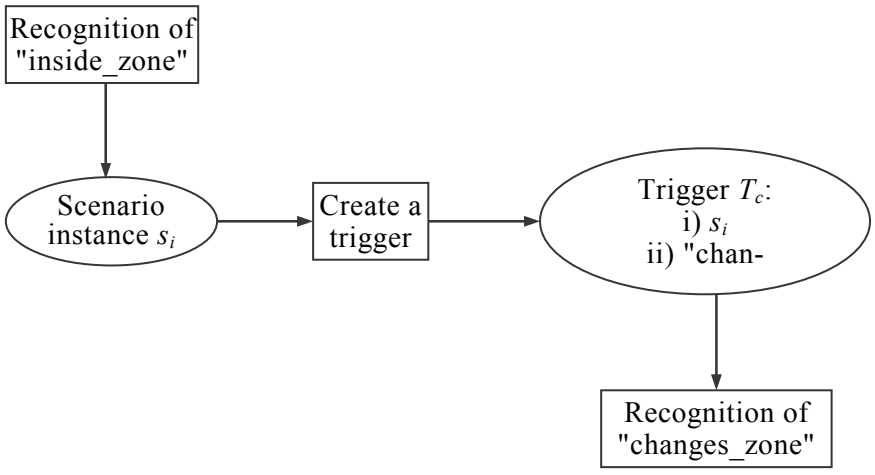


Figure 4.4. A trigger is created for the recognition of "changes_zone" scenario when an "inside_zone" scenario has been recognized.

Two questions to be asked here are "how can we create a trigger from a scenario instance that has been recognized?" and "how many triggers need to be created for a scenario instance that has been recognized?". To answer these questions, we propose a task to compile pre-defined scenario models that gathers (for each scenario model M) the information about the scenarios ending with a scenario instance of M. The

gathered information of M constructs a *trigger model* to be used to create triggers for all scenario instances of M.

Triggers are used as temporary objects that are created at each instant to control the recognition of composed scenarios and are deleted just after the termination of the recognition of its contained scenario model. Thus, they are not permanently saved in the memory as the recognized scenario instances. The recognition process creates at each instant all triggers depending on scenario instances recognized at that instant. The management of the created triggers is detailed in the next section and is an important point.

4.5 Storing Recognized Scenarios and Controlling the Recognition Process

Section 4.2 has shown an overview of the recognition process at each instant. We also explained that our algorithm is of Storing all Totally Recognized Scenarios class. Thus, the recognition process stores all already recognized scenario instances to be used to recognize the other composed scenarios. We present in this section, how to store the recognized scenario instances and how to control the recognition of composed scenarios.

4.5.1 Storing Recognized Scenario

The goal of the recognition process is to recognize correctly all scenario occurrences during a time interval as long as possible. Thus, to store a scenario instance s, the process first attempts to **merge** s with an already stored scenario instance s' if it is possible, if not the process will **store** s as a new element of a **Forest of Scenario Instances**.

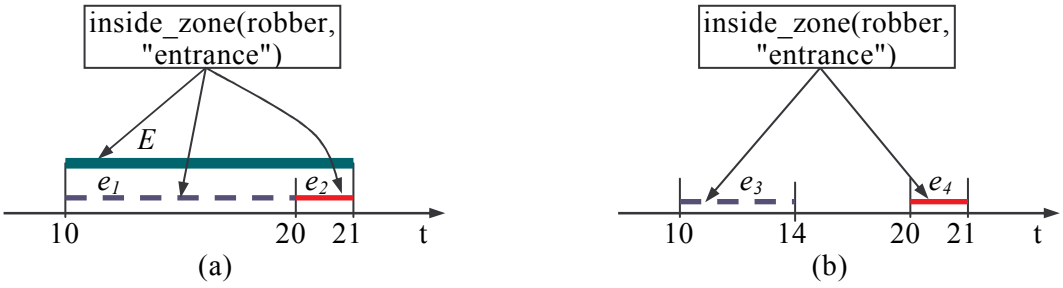


Figure 4.5. (a) An example of merging two scenario instances e_1 and e_2 of the same type into one scenario instance E of the same type as the two original scenario instances and (b) two scenario instances e_3 and e_4 that cannot be merged.

We first discuss about the conditions to merge two scenario instances s and s' (if $s'^{]} < s^{]}$) constructing a longer continuous scenario instance. The task of merging s and s' can be considered as extending $\tau(s')$ up to $s^{]}$. Thus, we can merge them if (1) s and s' are scenario instances of the same type ($s \approx s'$) and (2) there is no temporal hole between $\tau(s)$ and $\tau(s')$ (i.e. the ending time instant of $\tau(s)$ is equal to the starting time instant of $\tau(s')$). There is only one difference between s and s' concerning their time intervals.

Figure 4.5(a) shows an example of merging two scenario instances inside_zone(robber, "entrance") recognized during two time intervals [10, 20] and [21, 21] into one scenario instance during a longer time interval [10, 21]. In the (b) case, the two scenario instances cannot be merged because there is a time hole of 5 instants between these two scenario instances.

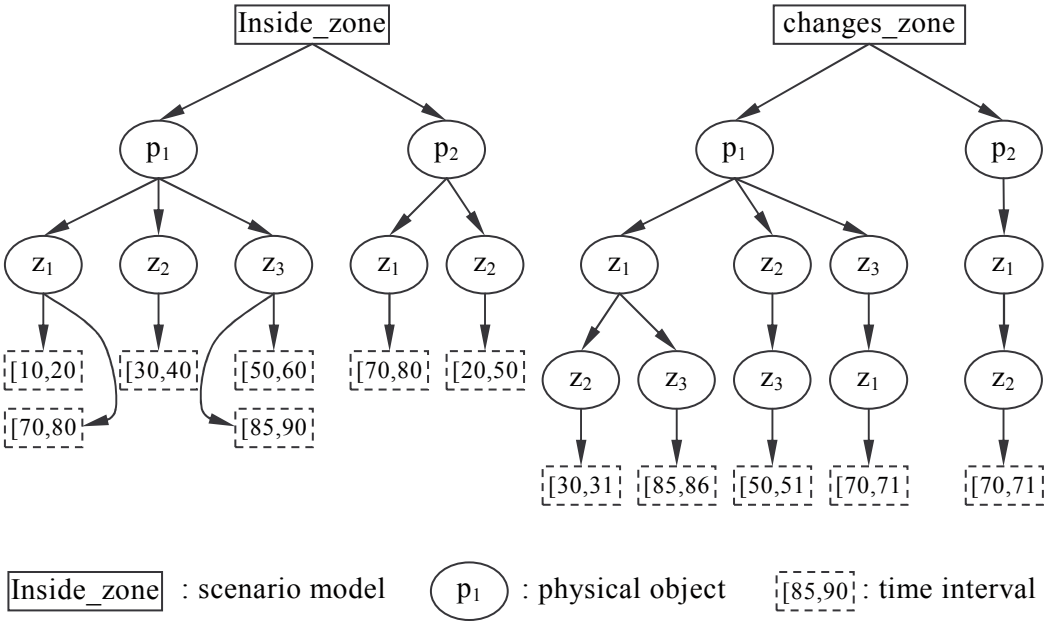


Figure 4.6. A part of a Forest of Scenario Instances to store scenario instances of "inside_zone" and "changes_zone" scenario models.

In case the storing process cannot merge s with any scenario instance s' previously recognized, s is added to a **Forest of Scenario Instances**. A Forest of Scenario Instances is used to speed up the process accessing to stored scenario instances and is a set of **Trees of Scenario Instances**. A Trees of Scenario Instances *Tree* is defined as follow:

Tree = (root, physical-object nodes, interval nodes, arcs)

Where:

root, denoted $\mathcal{R}(Tree)$, is the node corresponding to a scenario model,

physical-object nodes, denoted $\mathcal{PA}(Tree)$, is the set of intermediate nodes corresponding to the physical objects involved in the scenario instances stored in *Tree*,

interval nodes, denoted $\mathcal{IA}(Tree)$, is the set of leaf nodes corresponding to time intervals of the scenario instances stored by *Tree*. To speed up the search on this tree, we also order in time the interval nodes having the same parent node.

arcs, denoted $\mathcal{A}(Tree)$, is the set of arcs of *Tree*. For two physical-object nodes o_1, $o_2 \in \mathcal{PA}(Tree)$, there is an arc from o_1 to o_2 if and only if there is a scenario instance s involving these two physical ob-

jects and the physical-object variable corresponding to o_1 is defined before the physical-object variable corresponding to o_2 in the definition of $\mu(s)$.

A path in *Tree* starting from $\mathcal{R}(Tree)$ and stopping at a leaf node corresponds to a scenario instance stored in *Tree*. For example, in Figure 4.6, the path ("inside_zone", p_1, z_1, [10, 20]) corresponds to a scenario instance of "inside_zone" involving two physical objects p_1, z_1 and recognized during the time interval [10, 20]. For a given scenario instance s, the process to find s in a Forest of Scenario Instances is to (1) find the Tree of Scenario Instances corresponding to s using $\mu(s)$ and (2) follow the path composed of $\varphi(s)$ and $\tau(s)$.

By using a Forest of Scenario Instances and the proposed indexing method to store and index scenario instances, we can access to a scenario instance s by following a path composed of $|\varphi(s)| + 2$ nodes (one scenario model node + $|\varphi(s)|$ physical object nodes + one time interval node) on the given Forest of Scenario Instances. Thus, we can rapidly store and search any already recognized scenario instance, because $|\varphi(s)|$ is small as shown in bank monitoring and metro surveillance applications (see chapter 7).

4.5.2 Controlling the Recognition Process

In this section, we will discuss how to control the recognition of composed scenario models at each instant. We start by an example shown in Figure 4.7. This example shows two different cases of recognition of composed scenarios. In the first case (a) all scenario occurrences are correctly recognized, whereas in the second case (b) the scenario model M_c^2 is not recognized: to recognize M_c^2, we need a scenario instance of M_c^1, however, M_c^1 is not yet recognized. Thus, the order of recognizing composed scenarios at each instant is important.

We first study the necessary and sufficient condition enabling the recognition of predefined scenario models.

Lemma 4.1: $M_1, M_2 \in \mathcal{M}$, $M_1 \subset M_2 \Rightarrow$ *needed*($\mathcal{P}(M_1)$, $\mathcal{P}(M_2)$).

A short proof of this lemma is: $M_1 \subset M_2$, thus while attempting to recognize M_2, the recognition process needs a scenario instance of M_1 (i.e. *needed*($\mathcal{P}(M_1)$, $\mathcal{P}(M_2)$)).

Lemma 4.2: $M \in \mathcal{M}_c$,
$(\forall M' \in \sigma(M), \mathcal{P}(M') \in \mathcal{P}_{sol}) \wedge (\kappa(M)$ is coherent$) \Rightarrow \mathcal{P}(M) \in \mathcal{P}_{sol}$.

Proof:

$\kappa(M)$ is coherent $\Rightarrow \kappa(M)$ can be satisfied. Moreover, $(\forall M' \in \sigma(M), \mathcal{P}(M') \in \mathcal{P}_{sol})$, thus, it will exist scenario instances of all scenario models $M' \in \sigma(M)$ for the recognition of M. Thus, M can be recognized (i.e. $\mathcal{P}(M) \in \mathcal{P}_{sol}$).

From Lemma 4.2, we have:
$\forall M_e \in \mathcal{M}_e, M_e \in \mathcal{M}_{co} \Rightarrow \mathcal{P}(M_e) \in \mathcal{P}_{sol}$. (*)
because, $\sigma(M_e) = \varnothing$ and $\kappa(M_e)$ is coherent.

(*) $\mathcal{M}_{co}$: the set of all coherent scenario models [chapter 3]

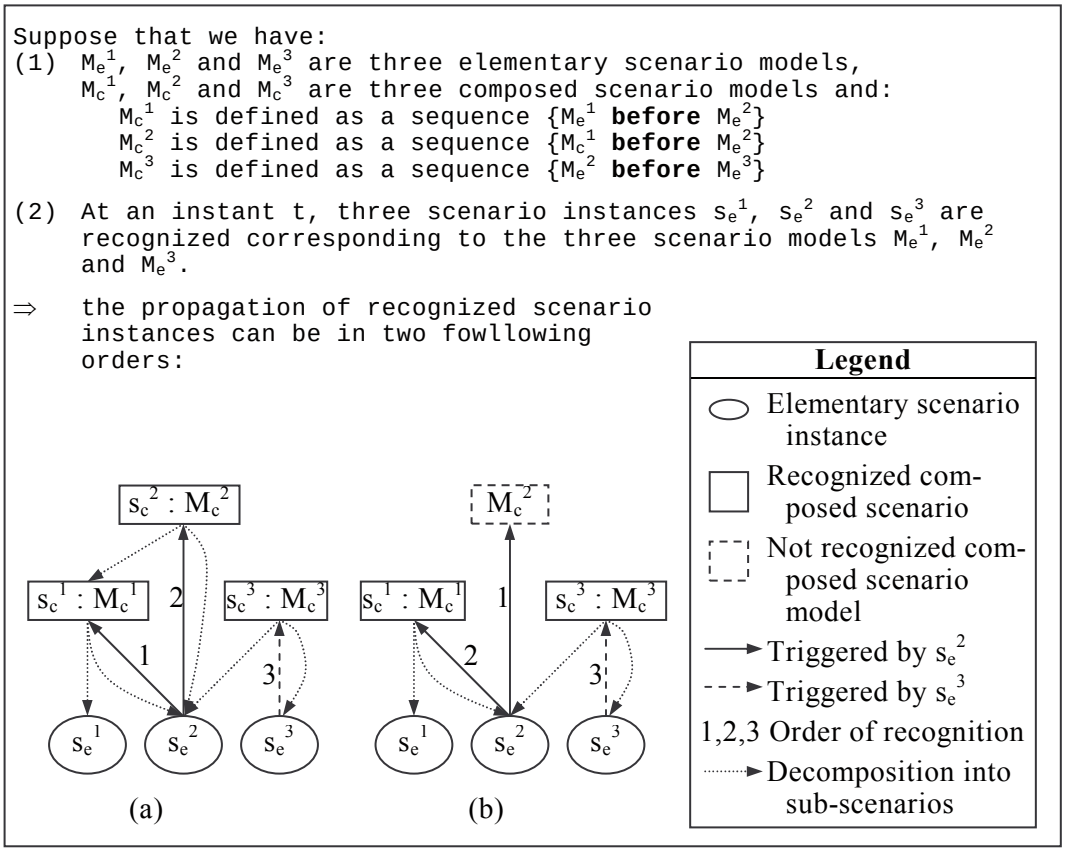


Figure 4.7. Two different orders of recognition: (a) all scenario occurrences are recognized and (b) M_c^2 is not recognized. When M_e^2 has been recognized, the recognition process generates a new scenario instance s_e^2 and triggers the recognition of M_c^1 and M_c^2. In the (a) case, the recognition of M_c^1 is triggered before the recognition of M_c^2 and both of them are recognized. In the (b) case, the recognition of M_c^2 is triggered before the recognition of M_c^1 and M_c^2 is not recognized, because the recognition process needs a scenario instance of M_c^1 to recognize M_c^2, but the recognition of M_c^1 is not yet triggered.

Theorem 4.1: $M_1, M_2 \in \mathcal{M}$, $M_1 \subset M_2$

$$\mathcal{P}(M_1) \notin \mathcal{P}_{sol} \quad \Rightarrow \quad \mathcal{P}(M_2) \notin \mathcal{P}_{sol}.$$

Proof:

$M_1 \subset M_2 \quad \Rightarrow \quad needed(\mathcal{P}(M_1), \mathcal{P}(M_2))$ (Lemma 4.1) (1)

$\mathcal{P}(M_1) \notin \mathcal{P}_{sol} \quad \Rightarrow \quad solutions(\mathcal{P}(M_1)) = \varnothing$ (2)

(1) & (2) $\quad \Rightarrow \quad \mathcal{P}(M_2) \notin \mathcal{P}_{sol}.$

Theorem 4.2: $M \in \wp\mathcal{M}$, $\mathcal{P}(M) \in \mathcal{P}_{sol} \Leftrightarrow M \in \wp\mathcal{M}_{co}$.

Proof:

a) $M \notin \wp\mathcal{M}_{co} \Rightarrow \mathcal{P}(M) \notin \mathcal{P}_{sol}$

1) $\kappa(M)$ is not coherent
$\Rightarrow \kappa(M)$ can never be satisfied
$\Rightarrow \mathcal{P}(M) \notin \mathcal{P}_{sol}$.

2) $\exists M' \in \wp\mathcal{M}, (M \subset M') \wedge (M' \subset M)$ (*) (1)
$(M \subset M') \Rightarrow$ *needed*$(\mathcal{P}(M), \mathcal{P}(M'))$ (Lemma 4.1)
$\Rightarrow \mathcal{P}(M)$ must be solved before $\mathcal{P}(M')$ (2)
$(M' \subset M) \Rightarrow \mathcal{P}(M')$ must be solved before $\mathcal{P}(M)$ (similarly to (2)) (3)
$(1) \wedge (2) \wedge (3) \Rightarrow \mathcal{P}(M) \notin \mathcal{P}_{sol}$.

3) $\exists M' \subset M, M' \notin \wp\mathcal{M}_{co}$

(i) $M' \in \wp\mathcal{M}_{e}$

$\Rightarrow \kappa(M')$ is not coherent
$\Rightarrow \mathcal{P}(M') \notin \mathcal{P}_{sol}$
$\Rightarrow \mathcal{P}(M) \notin \mathcal{P}_{sol}$ (Theorem 4.1)

(ii) $M' \in \wp\mathcal{M}_{c}$

By a recurrent demonstration, we have:
$\exists M'' \in \wp\mathcal{M}_{e}, M'' \subset M', M'' \notin \wp\mathcal{M}_{co}$
$\Rightarrow \mathcal{P}(M') \notin \mathcal{P}_{sol}$ (Theorem 4.1)
$\Rightarrow \mathcal{P}(M) \notin \mathcal{P}_{sol}$ (Theorem 4.1)

b) $M \in \wp\mathcal{M}_{co} \Rightarrow \mathcal{P}(M) \in \mathcal{P}_{sol}$

1) $M \in \wp\mathcal{M}_{e}$
$\Rightarrow \mathcal{P}(M) \in \mathcal{P}_{sol}$ (Lemma 4.2)

2) $M \in \wp\mathcal{M}_{c}$

(i) $\forall M' \subset M, M' \in \wp\mathcal{M}_{e}$
$\Rightarrow \forall M' \subset M, M' \in \wp\mathcal{M}_{co}$ (definition of coherent scenarios)
$\Rightarrow \forall M' \subset M, \mathcal{P}(M') \in \mathcal{P}_{sol}$ (Lemma 4.2)
$\Rightarrow \mathcal{P}(M) \in \mathcal{P}_{sol}$ (Lemma 4.2)

(ii) $\exists M' \subset M, M' \in \wp\mathcal{M}_{c}$
By a recurrent demonstration, we have:
$\mathcal{P}(M') \in \mathcal{P}_{sol}$
$\Rightarrow \forall M'' \in \sigma(M), \mathcal{P}(M'') \in \mathcal{P}_{sol}$
$\Rightarrow \mathcal{P}(M) \in \mathcal{P}_{sol}$ (because $\kappa(M)$ is coherent & Lemma 4.2)

(*) $\wedge$: logical "and".

Secondly, we study the order of scenario model recognition at each instant. Lemma 4.3 gives a fundamental criterion to order the list of triggers $\mathscr{Lt}$ at each instant.

Lemma 4.3: At each instant, M, M' $\in \mathscr{M}_{co}$:

$(M' \subset M) \Rightarrow \mathscr{P}(M')$ has to be solved before solving $\mathscr{P}(M)$ to avoid losing solution(s) of $\mathscr{P}(M)$.

A short proof of this lemma is: while trying to recognize M, we need a scenario instance s' of M'. But s' can only be generated when M' has been recognized. Therefore, if the recognition of M is done before the recognition of M', we can lose solution(s) because s' is not yet recognized.

To obtain a correct order of recognition, we propose to order the list $\mathscr{Lt}$ of triggers at each instant by the *inclusive relation* of scenario models (contained by the triggers of $\mathscr{Lt}$). However, chapter 3 has shown that the set of scenario models is partially and not totally ordered by the inclusive relation. Thus, we propose a relation called Δ -an extension of the inclusive relation- to order a list of scenario models.

Definition 4.2: $M_1, M_2 \in \mathscr{M}$:

$$M_1 \Delta M_2 \overset{def}{\cong} M_2 \not\subset M_1$$

We can demonstrate that all coherent sets of scenarios are inter-related by Δ.

We find that Δ is a correct relation that induces to order the trigger list $\mathscr{Lt}$ such as we will not lose scenario instances occurring at each instant, because:

Suppose, $\mathscr{Lt}$ is ordered by Δ and $Tr_1, Tr_2 \in \mathscr{Lt}$ are two triggers created at the same time instant (video frame). If $Tr_1 \Delta Tr_2$ then Tr_1 is before Tr_2 in $\mathscr{Lt}$, thus the recognition of $\mu(Tr_1)$ is performed before the recognition of $\mu(Tr_2)$. There are two cases:

a) $\mu(Tr_1) \subset \mu(Tr_2) \Rightarrow$ the order of $\mathscr{Lt}$ is correct (Lemma 4.3)

b) $(\mu(Tr_1) \not\subset \mu(Tr_2)) \wedge (\mu(Tr_2) \not\subset \mu(Tr_1)) \Rightarrow$ there is no relation between the recognition of $\mu(Tr_1)$ and the recognition of $\mu(Tr_2)$, thus in this case the order of recognition is not important.

The next chapter shows our algorithm for elementary scenario recognition.

Chapter 5. Elementary Scenario Recognition

This chapter presents the recognition of elementary scenario models according to the formalism proposed in chapter 3.

5.1 Elementary Scenario Recognition Overview

The recognition of an elementary scenario model M_e at each instant is considered as solving the corresponding CSP $\mathscr{P}(M_e)$, where $vars(\mathscr{P}(M_e))$ is an ordered set of non-temporal variables corresponding to $\varphi(M_e)$. Thus, $dom(\mathscr{P}(M_e))$ contains several sets of physical objects (e.g. set of tracked persons, set of interesting zones) of the corresponding types $type(vars(\mathscr{P}(M_e)))$. We notice that the constraints of $\kappa(M_e)$ are only non-temporal constraints. Thus, a naive constraint resolution technique can be used to solve this problem [Rota, 2001]. However, the complexity of the algorithm to recognize an elementary scenario model M_e is $O(rm^n)$ in function of constraint verifications, where: $r = |\kappa(M_e)|$, $n = |\varphi(M_e)|$ and m = median value of $|dom(v)|$, $v \in \varphi(M_e)$.

We will present in this section an approach reducing the number of combinations of physical objects to be tested for recognizing an elementary scenario. We start by the example shown in Figure 5.1. This figure shows a scene composed of three interesting zones (z_1, z_2, z_3) and one equipment (machine m). There are four persons in the scene at the given instant. In this situation, the CSP solver of Rota (2001) has to do 36 constraint verifications. In other words, it has to attempt all combinations of physical objects and verify all constraints. The next section presents a novel algorithm for elementary scenario recognition. The novel algorithm performs several combinations of physical objects instead of performing all combinations of physical objects as in the state of the art algorithm.

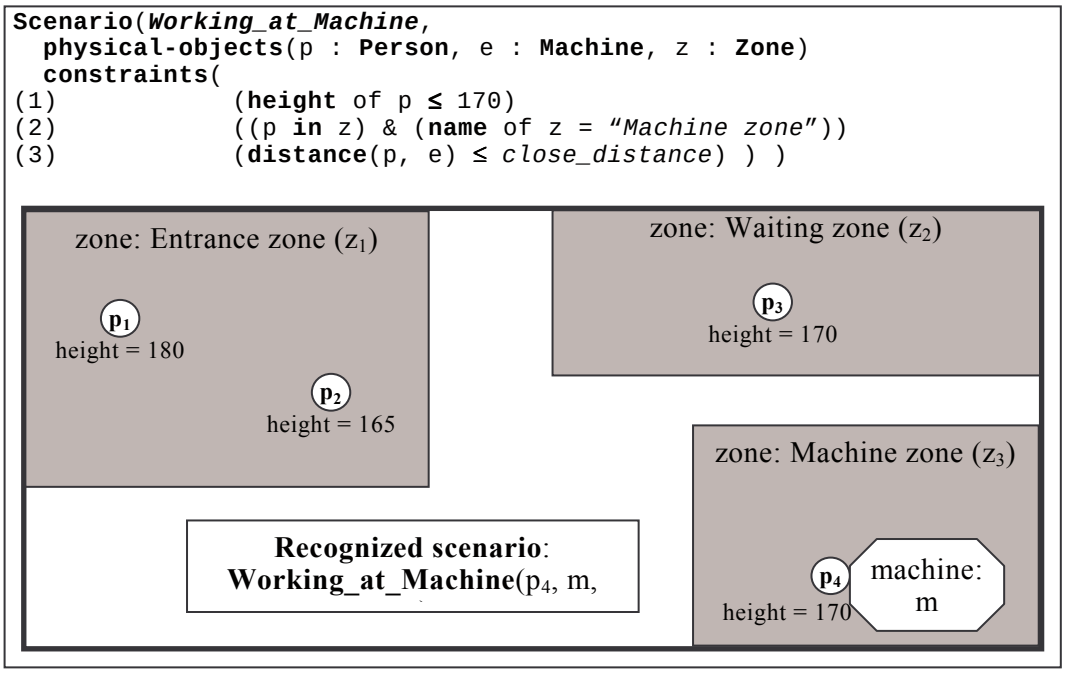


Figure 5.1. An example of the recognition of an elementary scenario. The CSP solver of Rota (2001) has to perform 36 operations to verify the constraints defined within the scenario model Working_at_Machine.

5.2 Novel approach

To reduce the complexity of the elementary scenario recognition algorithm, we propose to link the constraints defined within an elementary scenario model M_e to the physical object variables defined within M_e in a **compilation** step, such as: all constraints linked to a variable v (i) can be evaluated when v has been attached a value, and (ii) cannot be evaluated before.

For example: the "Working_at_Machine" scenario model [Figure 5.1] is defined with an ordered set of three physical object variables {p, e, z} (we take the order defined by experts). To recognize this scenario model, the recognition process attempts assigning values respectively for p, e then z. The constraint (1) can only be evaluated when p is assigned a value, thus this constraint can be linked to the variable p. In the same way, the constraint (2) and (3) are respectively linked to z and e.

Suppose that $\varphi(M_e) = \{v_1,\ldots, v_n\}$ is a set of ordered variables (e.g. the order given by the number of constraints involving each variable). We denote $\kappa(v)$, the set of the constraints linked to a physical object variable $v \in \varphi(M_e)$. We propose the following linking method of constraints for the compilation of an elementary scenario model M_e:

$$\kappa(v_i) = \{k \mid k \in \kappa(M_e), \mathit{deg}(k) = i\},\ i = 1,\ldots,n$$

We call the elementary scenario compiler a process reorganizing knowledge defined within elementary scenario models to speed up the recognition process of these models. An elementary scenario model is **compiled** if all its constraints are linked to

the corresponding physical object variables. For example, the "Working_at_Machine" scenario model [Figure 5.1] is compiled after that the constraints (1), (2) and (3) are respectively linked to the variables p, z and e.

The recognition of a compiled elementary scenario model M_e ($\varphi(M_e) = \{v_1,\ldots, v_n\}$) corresponds to a process attempting to (a) assign to each physical object variable v_i a value from its domain $dom(v_i) \in ld$ –a list of physical objects–, (b) verify all constraints $\kappa(v_i)$ and (c) eliminate immediately all values that do not correspond to v_i (i.e. that do not satisfy all the constraints $\kappa(v_i)$). If the last variable v_n has been assigned with a value and all the constraints $\kappa(v_n)$ are satisfied, M_e is recognized. In this case, the set of physical objects satisfying all the constraints corresponds to a solution of $\mathscr{P}(M_e)$.

Once M_e has been recognized, the recognition process has to create a new scenario instance s_e using M_e and $value(\varphi(M_e))$. To finish the recognition of the scenario instance s_e, the recognition process has also to execute the decisions defined within M_e using s_e, then, to store s_e in a Forest of Scenario Instances to be used to recognize composed scenario models. Finally, the recognition process has to create all triggers corresponding to M_e using s_e to trigger the recognition of composed scenario models that are ended with a temporal variable which value corresponds to a scenario instance of M_e.

```
ES_Recognition(Me : Model; lv : List; ld : List; i : Int; n : Int)
  for each o ∈ ld[i]
     lv[i] ← o
     if Satisfied(κ(lv[i]))
        if (i < n) ES_Recognition(Me, lv, ld, i+1, n) // next var.
        else se ← CreateInstance(Me, value(lv))  // recognized
             Execute(δ(Me), se)
             Store(se)
             CreateTrigers(Me, se)
```

Figure 5.2. Compiled elementary scenario models are recognized recursively by eliminating immediately values corresponding to physical objects that cannot be assigned to any physical object variable.

Figure 5.2 shows the proposed recognition algorithm of a compiled elementary scenario model M_e. The algorithm is called ES_Recognition and its parameters are: M_e, $\varphi(M_e)$, $dom(\varphi(M_e))$, 1 and $|\varphi(M_e)|$. We can notice that $dom(\varphi(M_e))$ is a list of sets of physical objects corresponding to $type(\varphi(M_e))$. This procedure attempts to assign a value and verifies constraints linked to all physical object variables of M_e from the first variable (corresponding to the value "1" in the parameter list) to the last one. The presented recognition procedure eliminates immediately values that cannot be assigned to the physical object variables $\varphi(M_e)$. Thus it avoids attempting all combinations of physical objects of the scene.

Figure 5.3a shows the distribution of constraints to the physical object variables defined within the scenario model "Working_at_Machine" shown in Figure 5.1. In this example, one constraint is distributed to each variable. Figure 5.3b shows the recognition of the "Working_at_Machine" scenario model by the situation given in Figure 5.1. For each tracked person $\{p_1, p_2, p_3\}$, the recognition process attaches each person with the variable p and verifies the constraints of $\kappa(p)$. If all constraints distributed to p are satisfied, the process continues the same tasks for the two other

physical object variables *e* and *z*. This example shows the same result (recognized scenarios) obtained in Figure 5.1. Moreover, the numbers of constraint verifications are different. The proposed approach takes 11 constraint verifications compared to 36 verifications obtained by the classical approach.

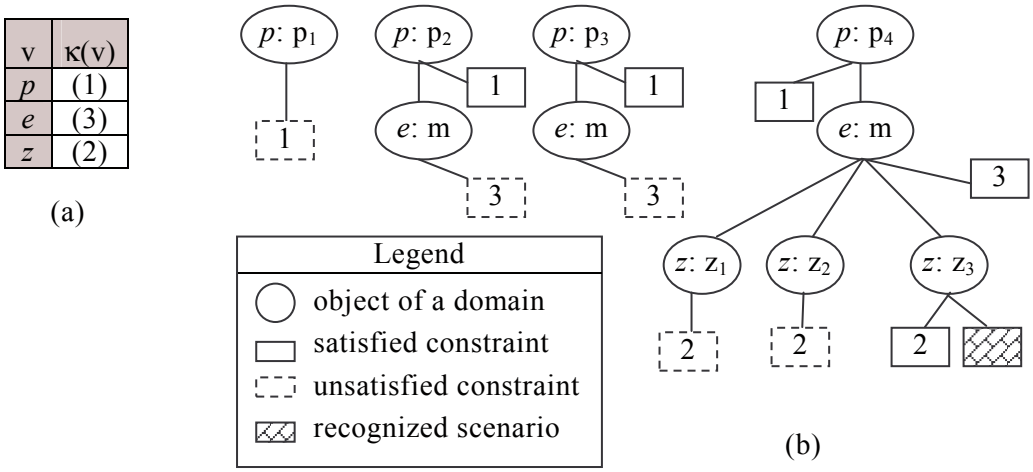


Figure 5.3. (a) The distribution of constraints to the physical object variables of the scenario model shown in Figure 5.1 and (b) the recognition of this elementary scenario model in the situation given by the same figure.

5.3 Proof and the Complexity of the Proposed Algorithm

5.3.1 Proof of the Proposed Algorithm

Suppose that M_e' is the compiled model of an elementary scenario model M_e.

To prove the correctness of the algorithm, it is necessary to prove: (1) M_e and M_e' are equivalent and (2) the algorithm recognizes correctly and exhaustively all scenario instances of M_e'.

(1) We need to prove:

(a) $\varphi(M_e) = \varphi(M_e')$,

(b) $\kappa(M_e)$ and $\kappa(M_e')$ have the same constraints.

Proof:

(a) is true because the elementary scenario model compiler does not change (add/remove) any physical object variable defined within M_e.

(b) We have:

$$\kappa(M_e') = \bigcup_{i=1}^{n} \kappa(v_i) = \kappa(M_e).$$

There is only one difference between $\kappa(M_e')$ and $\kappa(M_e)$ concerning the order of constraints of these two constraint sets. Thus, it is necessary to prove that all constraints $\kappa(v_i)$ can be evaluated only when v_i has been assigned with a value.

$\forall k \in \kappa(v_i) \Rightarrow deg(k) = i$ (compilation algorithm)

$\Rightarrow (\forall v \in k \Rightarrow v \in \{v_1,\ldots, v_i\})$ (constraint degree definition)

$\Rightarrow \forall v \in k$, v has been assigned with a value when v_i has been assigned with a value (recognition algorithm)

$\Rightarrow$ k can be evaluated when v_i has been assigned with a value.

Moreover, k cannot be evaluated when v_i has not been assigned with a value yet, because $v_i \in k$.

$\Rightarrow$ k can be evaluated only when v_i has been assigned with a value.

(a) $\wedge$ (b) $\Rightarrow$ M_e and M_e' are equivalent.

(2) We need to prove that the algorithm can find all solutions of $\mathscr{P}(M_e')$ and all found solutions are correct.

The recognition algorithm attempts all combinations of physical objects, thus, it can find all solutions for M_e'. Moreover, it verifies all constraints of M_e', thus, all found solutions are correct.

(1) $\wedge$ (2) $\Rightarrow$ the novel algorithm recognizes correctly (finds all solutions and the found solutions are correct) pre-defined elementary scenarios.

5.3.2 Complexity of the Proposed Algorithm

To estimate the complexity of the recognition algorithm, we consider that a verification of a constraint is the computational unit. Thus, the estimation of the complexity of the proposed algorithm consists of calculating how many times the algorithm has to verify the defined constraints to recognize a compiled elementary scenario model.

Suppose:

$\varphi(M_e) = \{v_1,\ldots, v_n\}$, $n \geq 1$,

$r = |\kappa(M_e)|$, is the number of constraints defined within M_e,

$r_i = |\kappa(v_i)|$, $i = 1\ldots n$, is the number of constraints included in $\kappa(v_i)$,

$M_i = |dom(v_i)|$, $i = 1\ldots n$, is the domain size of v_i,

$V_i = \{o^i_1,\ldots, o^i_{m_i}\}$, $i = 1\ldots n$, is a set of physical objects such that if we attach $o^i_j \in V_i$ to v_i then $\kappa(v_i)$ is satisfied.

$\Rightarrow$ the total number of constraint verifications involved in the recognition of M_e' is:

$$C = \sum_{i=1}^{n}(\prod_{j=0}^{i-1} m_j)M_i r_i \text{ , where } m_0 = 1.$$

The best case: no solution is found.

$\forall i \in [1..n], m_i = 0 \Rightarrow C = M_1 r_1$

$\Rightarrow$ the complexity of the algorithm is $O(M_1 r_1)$.

The median case:

Let ρ be the median value of $\frac{M_i}{m_i}$, $i = 1 \ldots n$,

Let m be the median value of M_i, $i = 1 \ldots n$, thus

$$C \approx \sum_{i=1}^{n} r_i \left(\frac{m}{\rho}\right)^i, \quad 1 \le \rho \le m$$

$\Rightarrow$ the complexity of the algorithm is $O(r\left(\frac{m}{\rho}\right)^n)$.

The worst case: all combinations of physical objects are solutions of the given scenario model.

$\forall i \in [1..n], m_i = M_i \Rightarrow \rho = 1$

$\Rightarrow$ the complexity of the algorithm is $O(rm^n)$ as the classical algorithm.

The recognition of elementary scenario models is a primary task that must be done before the recognition of composed scenario models. In some other automatic video interpretation frameworks, the recognition of elementary scenarios is done by the vision module (Hongeng & Nevatia, 2001; Pinhanez & Bobick, 1997; Tessier, 1997) or elementary scenario instances are given by the hardware as events (Dousson & Ghallab, 1994; Dousson, 2002). Thus, the recognition module has not to recognize elementary scenarios and takes as input the events detected by other modules.

In practice (as shown in chapter 7, we have realized experiments of metro surveillance and bank monitoring applications): (1) there are few physical objects that satisfy $\kappa(v_i)$ for all $i = 1...n$. In other words, m_i is usually inferior to m so ρ is large. Moreover, (2) elementary scenarios involve a small number of actors ($n \le 3$). Thus, the recognition can be real-time (video cadence).

The next chapter presents the recognition of composed scenario models.

Chapter 6. Composed Scenario Recognition and Scenario Knowledge Base Optimization

This chapter first presents the recognition of composed scenario models. Then it presents a short synthesis of the recognition process. Finally, it presents scenario knowledge base coherence verification and simplification methods.

6.1 Composed Scenario Recognition Overview

A composed scenario is viewed as a sequence of temporally ordered sub-scenarios. Thus, the recognition of a composed scenario model M_c consists principally in finding the scenario instances previously recognized corresponding to the temporal variables $\sigma(M_c)$ and verifying all constraints defined within M_c. A generic constraint resolution technique can be used to solve this problem [Rota, 2001]. This approach leads to a high complexity algorithm. Rota also showed that his technique cannot be real-time (in video cadence) with real-world problem, thus, it can only be used for off-line problem, e.g. video indexing. As shown in chapter 2, there are a number of algorithms to recognize temporal scenarios. An efficient technique of the state of the art is the Chronicle Recognition Technique proposed by Dousson & Ghallab (1994). We have also underlined in chapter 2 that the proposed chronicle recognition technique is not efficient for **multi-physical-object** scenarios because it has to duplicate all partially recognized scenarios for all combination of physical objects. Moreover, in this approach, the temporal bound of pre-defined scenarios needs to be ensured.

In our approach, the goal is to minimize the combination number of scenario instances and physical objects to be tested. As described in chapter 3, a composed scenario can be composed of several sub-scenarios and forbidden sub-scenarios. To simplify the composed scenario recognition problem, we first present the algorithm recognizing composed scenarios that do not contain any forbidden sub-scenarios. The processing of scenario models containing forbidden sub-scenarios is addressed later in section 6.5 as a filter on recognized scenario instances without forbidden sub-scenario.

```
CS_RecognitionWithoutForbiddenConstraint(T : trigger)
  Mc   ← μ(T) // Scenario model to be recognized
  ltv  ← σ(Mc)   // List of temporal variables
  n    ← |ltv|   // Number of temporal variables
  ltv[n] ← ξ(T)  // Last sub-scenario instance
  CS_Recognition_1(Mc, ltv, dom(ltv), 1, n-1)
```

```
CS_Recognition_1(Mc : Model; ltv : List; ld : List; i: Int; n : Int)
  for each o ∈ ld[i]
     ltv[i] ← o
     if (i < n) CS_Recognition_1(Mc, ltv, ld, i+1, n) // Next var.
     else
        if Satisfied(κT(Mc)) // verify the temporal constraints
           φ(Mc) ← value(φsub(Mc)) // instanciate physical obj. var.
           if Satisfied(κN(Mc)) //verify the nontemporal constraints
              // the given scenario model is recognized
              sc ← CreateInstance(Mc, value(φ(Mc)), value(ltv))
              Execute(δ(Mc), sc)
              Store(sc)
              CreateTrigers(Mc, sc)
```

Figure 6.1: Recognition algorithm of composed scenarios without forbidden scenarios.

Figure 6.1 shows the algorithm recognizing a composed scenario model M_c (contained in a given trigger) that does not contain any forbidden scenario. To recognize M_c, the algorithm first tries to instanciate the temporal variables $\sigma(M_c)$ corresponding to the sub-scenarios of M_c with scenario instances previously recognized contained in $dom(\sigma(M_c))$. We can notice that the last temporal variable of M_c is already instanciated with $\xi(T)$. A combination s of scenario instances corresponding to $\sigma(M_c)$ is acceptable if s satisfies all temporal constraints of $\kappa_T(M_c)$. Second, the algorithm instanciates the non-temporal variables of $\varphi(M_c)$ with the physical objects involved in the scenario instances composing s. Then, the algorithm verifies all non-temporal constraints of $\kappa_N(M_c)$ with the values assigned for $\varphi(M_c)$. If all these non-temporal constraints are satisfied, M_c is recognized. Once M_c is recognized, the recognition process creates a new scenario instance s_c using s and $value(\varphi(M_c))$. To finish the recognition of scenario instance s_c, the algorithm also has to execute the decisions defined within M_c using s_c, then, to store s_c in the Forest of Scenario Instances to be used to recognize the composed scenarios that contain a scenario instance of M_c. Finally, the algorithm has to create all triggers (if necessary) corresponding to M_c using s_c to trigger the recognition of other composed scenario models that end with a scenario instance of M_c.

6.2 Temporal Constraint Resolution Problem

We start by the example shown in Figure 6.2. In this example, we have a scenario model M_c composed of a temporal sequence of three temporal variables v_1, v_2, v_3. Suppose there are p_1 scenario instances of scenario model $\mu(v_1)$ and p_2 scenario instances of scenario model $\mu(v_2)$ that were already recognized. If a scenario instance o_3 of scenario model $\mu(v_3)$ has been recognized, it makes sense to try to recognize M_c. Therefore, the algorithm has to try $p_1.p_2$ combinations of scenario instances of o_3 with all scenario instances of $dom(v_1)$ and $dom(v_2)$. Thus, the algorithm can lead to a **combinatorial explosion** problem. Generally, for a composed scenario model M_c, the number of combinations of scenario instances to be verified is:

$$\prod_{i=1}^{n-1} |dom(v_i)|, \qquad \text{where: } \sigma(M_c) = \{v_1, \ldots, v_n\}$$

Let m be the median value of $|dom(v_i)|$, thus, the number of combinations of scenario instances to be verified is equivalent to m^{n-1}.

$\Rightarrow$ If $n \leq 2$, the number of combinations of scenario instances to be tested in the recognition of M_c is linear in function of m.

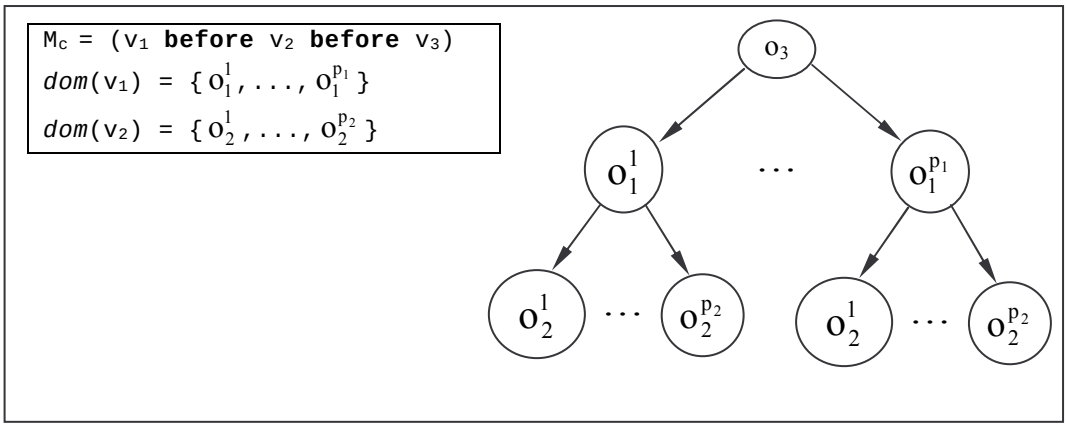


Figure 6.2. The recognition of a composed scenario model can lead to a combinatorial explosion. If o_3 of $\mu(v_3)$ has been recognized then $p_1.p_2$ combinations of scenario instances to be verified for the recognition of M_c.

To solve the combinatorial explosion problem caused by the recognition of composed scenarios, *we propose to decompose the pre-defined composed scenario models in an initial stage such that each composed scenario model is at the most composed of two sub-scenarios*. This algorithm is shown in the next section.

6.3 Scenario Compilation

All scenario models used by an Automatic Video Interpretation System are compiled in an initial stage to reorganize the knowledge represented within the scenario models. This compilation of scenario models allows reducing the processing time of the recognition algorithm. The compilation of scenario models is done as a pre-processing stage, thus this compilation does not need to be real-time. The process compiling a composed scenario model M_c consists principally in: (1) building an initial temporal constraint graph representing $\kappa_T(M_c)$, (2) verifying the temporal consistency of the scenario model, (3) ordering in time its temporal variables by their ending time, (4) simplifying the temporal constraint set of the model and (5) decomposing the scenario into intermediate scenario models that contain at the most two sub-scenarios.

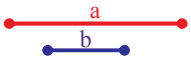
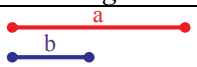
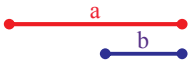
Constraint	Edge to be added (vertex : interval : vertex)	Constraint	Edge to be added (vertex : interval : vertex)
Relations between two **time intervals** from Allen's interval algebra			
 a *equal* b	$a^{[} : [0, 0] : b^{[}$ $a^{]} : [0, 0] : b^{]}$		
 a *before* b	$a^{]} : [1, \infty[: b^{[}$	 a *ibefore* b	$b^{]} : [1, \infty[: a^{[}$
 a *meet* b	$a^{]} : [0, 0] : b^{[}$	 a *imeet* b	$b^{]} : [0, 0] : a^{[}$
 a *overlap* b	$a^{[} : [1, \infty[: b^{[}$ $b^{[} : [1, \infty[: a^{]}$ $a^{]} : [1, \infty[: b^{]}$	 a *ioverlap* b	$b^{[} : [1, \infty[: a^{[}$ $a^{[} : [1, \infty[: b^{]}$ $b^{]} : [1, \infty[: a^{]}$
 a *during* b	$b^{[} : [1, \infty[: a^{[}$ $a^{]} : [1, \infty[: b^{]}$	 a *iduring* b	$a^{[} : [1, \infty[: b^{[}$ $b^{]} : [1, \infty[: a^{]}$
 a *start* b	$a^{[} : [0, 0] : b^{[}$ $a^{]} : [1, \infty[: b^{]}$	 a *istart* b	$b^{[} : [0, 0] : a^{[}$ $b^{]} : [1, \infty[: a^{]}$
 a *finish* b	$a^{]} : [0, 0] : b^{]}$ $b^{[} : [0, \infty[: a^{[}$	 a *ifinish* b	$b^{]} : [0, 0] : a^{]}$ $a^{[} : [0, \infty[: b^{[}$
New temporal operators [chapter 3]: $I, I_1, I_2 \in \mathcal{T}_i$, $p, p_1, p_2 \in \mathcal{T}_p$			
$p \text{ —| } I$	$p : [0, 0] : I^{]}$	$I \text{ —| } p$	$I^{]} : [0, 0] : p$
$p < I$	$p : [1, \infty[: I^{]}$	$p_1 \leq p_2$	$p_1 : [0, \infty[: p_2$
$I < p$	$I^{]} : [1, \infty[: p$	$p \leq I$	$p : [0, \infty[: I^{]}$
$I_1 < I_2$	$I_1^{]} : [1, \infty[: I_2^{]}$	$I_1 \leq I_2$	$I_1^{]} : [0, \infty[: I_2^{]}$
Relations between two **time points**			
$i = k$	$i : [0, 0] : k$		
$i < k$	$i : [1, \infty[: k$	$i \leq k$	$i : [0, \infty[: k$

Figure 6.3. This table shows how to transform a temporal constraint into positive time intervals. The first set of temporal constraints is the relations between two time intervals. These relations come from Allen's interval algebra. The second set describes the new operators [chapter 3] between one time point and an interval and between two intervals. The third set of constraints describes relations between two time points.

6.3.1 Initial Graph

We use a graphical method to compile pre-defined composed scenario models. For each composed scenario model **M**, we first build a directed graph **G** called **Temporal Constraint Graph**, as the following:

G = (V(G), E(G))

V(G) = a set of vertices corresponding to the temporal variables σ(M),

E(G) = a set of edges corresponding to the temporal constraints $\kappa_T(M)$.

For a temporal variable $v \in \sigma(M)$, the scenario model compiler adds to the set of vertices V(G) two vertices $v^[$ and $v^]$ corresponding to two bounds of v. Thus, V(G) contains $2|\sigma(M)|$ vertices and each vertex corresponds to a time point.

An edge $e \in E(G)$, $e = (v_1, v_2)$ from a vertex v_1 to another vertex v_2, is represented by a **positive interval** corresponding to the time delay between the two time points. For two vertices v_1 and v_2, there is an edge $[t_1, t_2]$ from v_1 to v_2 if there is a constraint $(v_1 + t_1 \leq v_2 \leq v_1 + t_2)$ defined within M. Thus, $(v_1, v_2) \in E(G) \Rightarrow v_1 \leq v_2$. Moreover, for two vertices $v^[$ and $v^]$ originated from the same temporal variable $v \in \sigma(M)$, we insert an edge "[0, ∞[" from $v^[$ to $v^]$.

To represent all temporal constraints of $\sigma(M)$ in G, the scenario model compiler transforms all its symbolical temporal constraints to positive intervals. The quantitative constraints between temporal variables can be directly transformed to positive intervals. To transform these constraints, we propose a correspondence between temporal constraints and positive intervals as shown in Figure 6.3. For example, (a) a constraint (v_1 *before* v_2) is represented by an edge "[1, ∞[" from $v_1^]$ to $v_2^[$, (b) a constraint (v_1 *meet* v_2) is represented by an edge "[0, 0]" from $v_1^]$ to $v_2^[$ and (c) a constraint (*duration* of $v \leq l$) is represented by an edge "[0, l]" from $v^[$ to $v^]$.

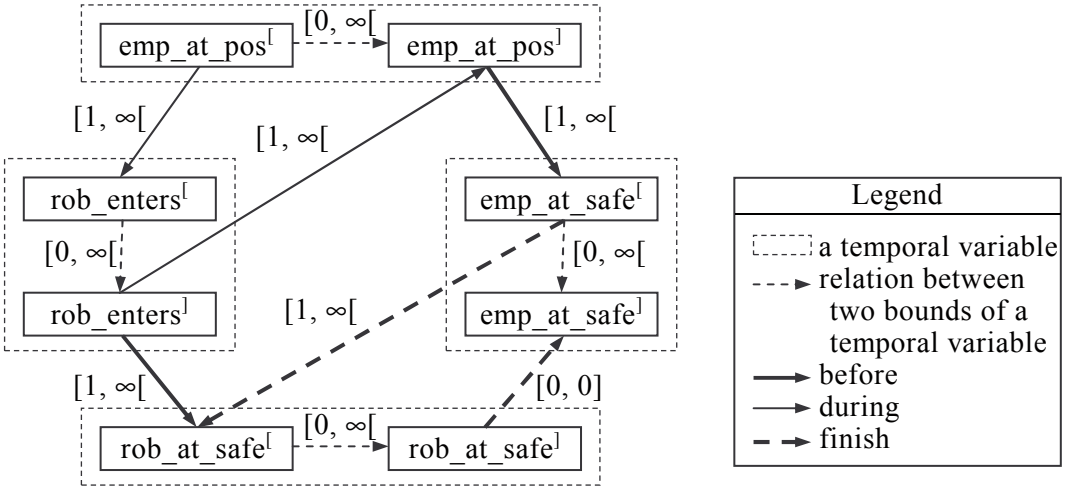


Figure 6.4. The initial temporal constraint graph generated for the compilation of the "bank_attack" scenario model defined in chapter 3.

Figure 6.4 shows the initial graph generated for the compilation of the "bank_attack" scenario represented in chapter 3. This graph is composed of eight vertices corresponding to the bounds of the four temporal variables of $\sigma(M)$ and ten edges corresponding to the temporal constraints between these vertices. An edge "[0, ∞[" (discontinuous thin arrow) is added for each couple of vertices originated from the same temporal variable. The temporal constraint *before* is represented by an edge "[1, ∞["

(continuous thick arrow). A temporal constraint *during* is represented by a couple of edges "[1, ∞[" (continuous thin arrow). Moreover, a temporal constraint *finish* is represented by a couple of edges "[1, ∞[" and "[0, 0]" (discontinuous thick arrow).

Definition 6.1:

1) A combination of time points (corresponding to the vertices of V(G)) that satisfies all constraints expressed by the edges of E(G) is called a solution of G. We denote, *solutions*(G), the set of all solutions of G.
2) Two graphs G and G' are *equivalent*, denoted $G \cong G'$, if and only if *solutions*(G) = *solutions*(G').

We can prove that $s \in solutions(\mathcal{P}(M)) \Rightarrow \sigma(s) \in solutions(G)$: because E(G) is the set of edges corresponding to all temporal constraints of $\kappa_T(M)$ and $\sigma(s)$ satisfies all constraints of $\kappa_T(M)$.

The following definitions are based on the Temporal Constraint Graph G corresponding to a composed scenario model M. Several definitions exist in graph theory [Wirth, 1986] but we adapt them to the particular problem of temporal scenario recognition.

Definition 6.2: $v, v' \in V(G)$,

for an edge $e = (v, v') \in E(G)$,

v is called the *starting vertex* of e and denoted e^+,

v' is called the *ending vertex* of e and denoted e^-,

$v^+ \overset{def}{\cong} \{e \in E(G), v = e^-\}$ = set of *entering edges* of v,

$v^- \overset{def}{\cong} \{e \in E(G), v = e^+\}$ = set of *exiting edges* of v,

$succ(v) \overset{def}{\cong} \{v' \in V(G), \exists e \in v^-, v' = e^-\}$ = set of *successive vertices* of v,

$pre(v) \overset{def}{\cong} \{v' \in V(G), \exists e \in v^+, v' = e^+\}$ = set of *precedent vertices* of v,

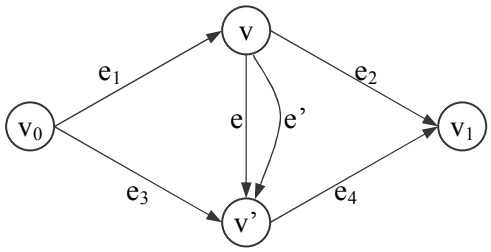
$E(v, v') \overset{def}{\cong}$ the set of all edges of E(G) from v to v',

v is called an *entering vertex* of $G \overset{def}{\cong} v^+ = \varnothing$,

$V^+(G) \overset{def}{\cong}$ the set of all entering vertices of G,

v is called an *exiting vertex* of $G \overset{def}{\cong} v^- = \varnothing$,

$V^-(G) \overset{def}{\cong}$ the set of all exiting vertices of G [Figure 6.5].



$v = e^+ = e'^+$, $v' = e^- = e'^-$
$v_0^+ = \varnothing$, $v^+ = \{e_1\}$, $v'^+ = \{e_3, e, e'\}$
$v^- = \{e, e', e_2\}$, $v'^- = \{e_4\}$, $v_1^- = \varnothing$
succ$(v) = \{v', v_1\}$
pre$(v') = \{v_0, v\}$
$E(v, v') = \{e, e'\}$
$V^+(G) = \{v_0\}$
$V^-(G) = \{v_1\}$

Figure 6.5. Example illustrating the notions given by Definition 6.2.

Definition 6.3: $v_1, v_2 \in V(G)$, (*)

v_1 and v_2 are *semi-connected*, denoted $v_1 \leftrightarrow v_2$:

$$v_1 \leftrightarrow v_2 \overset{def}{\cong} ((v_1, v_2) \in E(G)) \vee ((v_2, v_1) \in E(G)) \vee \exists v^* \in V(G), (v_1 \leftrightarrow v^*) \wedge (v^* \leftrightarrow v_2)$$

$$G \text{ is } \textit{semi-connected} \overset{def}{\cong} \forall v^1, v^2 \in V(G), (v^1 \leftrightarrow v^2)$$

Figure 6.6 illustrates couples of vertices that are semi-connected. Figure 6.5 illustrates a semi-connected graph.

Definition 6.4: $v_1, v_2 \in V(G)$,

v_1 and v_2 are *path-connected*, denoted $v_1 \rightarrow v_2$:

$$v_1 \rightarrow v_2 \overset{def}{\cong} ((v_1, v_2) \in E(G)) \vee (\exists v^* \in V(G), (v_1 \rightarrow v^*) \wedge (v^* \rightarrow v_2)).$$

if $v_1 \rightarrow v_2$, by a recurrent deduction, we have:

$\exists v^1, \ldots, v^k \in G$, such as $(v_1, v^1), (v^1, v^2), \ldots, (v^{k-1}, v^k), (v^k, v_2) \in E(G)$.

a path p is a couple of V_p –a set of vertices- and E_p –a set of edges-,

where:

$V(p) = \{v_1, v^1, \ldots, v^k, v_2\}$,

$E(p) = \{(v_1, v^1), (v^1, v^2), \ldots, (v^{k-1}, v^k), (v^k, v_2)\}$,

is a *path* in G starting at v_1 and ending at v_2.

$|p| = (v_1, v^1) \oplus (v^1, v^2) \oplus \ldots \oplus (v^{k-1}, v^k) \oplus (v^k, v_2)$ is the time interval cumulating all the delays between the vertices of the path.

We note two set of paths:

$P(v_1, v_2)$, the set of all paths started from v_1 and ended with v_2

$P(G)$, the set of all paths in G.

For example: in Figure 6.6, $p = (\{v_1, v_2, v_3, v_4\}, \{e_1, e_3, e_5\})$ is a path from v_1 to v_4 and $|p| = e_1 \oplus e_3 \oplus e_5$. $P(v_1, v_4) = \{(\{v_1, v_2, v_4\}, \{e_1, e_4\}), (\{v_1, v_3, v_4\}, \{e_2, e_5\}), p\}$.

(*) $\vee$: logical "or"

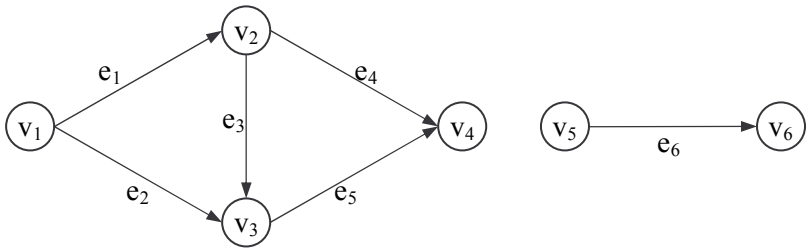


Figure 6.6. The following couples of vertices are semi-connected: (v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_4, v_3) and (v_5, v_6). The graph containing these 6 vertices is not semi-connected.

Definition 6.5:

$p \in P(G)$, is called a *cycle* on G if p starts and ends with the same vertex. We denote, C(G) the set of all cycles on G.

Definition 6.6: $v_1, v_2 \in V(G)$, $E_1 = \bigcap_{e_{1,2} \in E(v_1, v_2)} e_{1,2}$, $E_2 = \bigcap_{e_{2,1} \in E(v_2, v_1)} e_{2,1}$,

v_1 and v_2 are *arc-consistent* if and only if:

1) $E(v_1, v_2) \neq \varnothing \quad \Rightarrow \quad E_1 \neq \varnothing$
2) $E(v_2, v_1) \neq \varnothing \quad \Rightarrow \quad E_2 \neq \varnothing$
3) $(E_1 \neq \varnothing) \wedge (E_2 \neq \varnothing) \quad \Rightarrow \quad (E_1 = [0, 0]) \wedge (E_2 = [0, 0])$

Definition 6.7: $v_1, v_2 \in V(G)$, $P_1 = \bigcap_{p_{1,2} \in P(v_1, v_2)} | p_{1,2} |$, $P_2 = \bigcap_{p_{2,1} \in P(v_2, v_1)} | p_{2,1} |$,

v_1 and v_2 are *path-consistent*, denoted $v_1 \otimes v_2$, if and only if:

1) $P(v_1, v_2) \neq \varnothing \quad \Rightarrow \quad P_1 \neq \varnothing$
2) $P(v_2, v_1) \neq \varnothing \quad \Rightarrow \quad P_2 \neq \varnothing$
3) $(P_1 \neq \varnothing) \wedge (P_2 \neq \varnothing) \quad \Rightarrow \quad (P_1 = [0, 0]) \wedge (P_2 = [0, 0])$

G is *path-consistent* $\overset{\text{def}}{\cong} \forall v^1, v^2 \in V(G), v^1 \otimes v^2$.

We can demonstrate that $\otimes$ is symmetrical. Moreover, for two vertices $v_1, v_2 \in V(G)$, if v_1 and v_2 are not arc-consistent then they are not path-consistent.

The following theorem shows an important characteristic of the vertices of a Temporal Constraint Graph. The delay between two vertices v_1 and v_2 is directly linked to the time interval $|p|$ (with p -any path linking v_1 to v_2-) cumulating the delays between all the vertices of p.

Theorem 6.1: $v_1, v_2 \in V(G)$,

$p \in P(v_1, v_2) \quad \Rightarrow \quad v_1 + |p|^{[} \leq v_2 \leq v_1 + |p|^{]}$.

Proof:

Suppose: $V(p) = \{v_1, v^1, \ldots, v^k, v_2\}$
$E(p) = \{e_1, \ldots, e_{k+1}\}$
$e_1 = (v_1, v^1), \ldots, e_k = (v^{k-1}, v^k)$ and $e_{k+1} = (v^k, v_2)$

We have:

$$\begin{array}{lllll} & e_1^{[} & \leq & v^1 - v_1 & \leq & e_1^{]} \\ & e_2^{[} & \leq & v^2 - v^1 & \leq & e_2^{]} \\ + & \ldots\ldots\ldots & & & & \\ & e_k^{[} & \leq & v^k - v^{k-1} & \leq & e_k^{]} \\ & e_{k+1}^{[} & \leq & v_2 - v^k & \leq & e_{k+1}^{]} \end{array}$$

$$\Rightarrow \sum_{i=1}^{k+1} e_{k+1}^{[} \leq v_2 - v_1 \leq \sum_{i=1}^{k+1} e_{k+1}^{]}$$

$$\Leftrightarrow |p|^{[} \leq v_2 - v_1 \leq |p|^{]}$$

$$\Leftrightarrow v_1 + |p|^{[} \leq v_2 \leq v_1 + |p|^{]}.$$

Based on these definitions, we proposed in the next section a criterion to verify if a composed scenario model is well defined.

6.3.2 Scenario Model Consistency Verification

This section presents two criteria to verify whether a scenario model M is coherent. These criteria are based on the consistency of temporal constraints defined within M. Suppose that G is the temporal constraint graph corresponding to M.

Theorem 6.2: G is not path-consistent $\Rightarrow M \notin \mathcal{M}_{cc}$.

Proof:

G is not path-consistent $\Rightarrow \exists v_1, v_2 \in V(G), \neg(v_1 \otimes v_2)$

Let $P_1 = \bigcap_{p_{1,2} \in P(v_1, v_2)} | p_{1,2} |$, $P_2 = \bigcap_{p_{2,1} \in P(v_2, v_1)} | p_{2,1} |$, there are three cases:

(1) $(P(v_1, v_2) \neq \varnothing) \wedge (P_1 = \varnothing)$

$\Rightarrow$ $\exists p_1, p_2 \in P(v_1, v_2)$, such as: $|p_1|^{]} < |p_2|^{[}$ (Definition) (a)
$(\exists p_1 \in P(v_1, v_2)) \Rightarrow (|p_1|^{[} \leq v_2 - v_1 \leq |p_1|^{]})$ (Theorem 6.1) (b)
$(\exists p_2 \in P(v_1, v_2)) \Rightarrow (|p_2|^{[} \leq v_2 - v_1 \leq |p_2|^{]})$ (Theorem 6.1) (c)
(b), (c) $\Rightarrow |p_2|^{[} \leq v_2 - v_1 \leq |p_1|^{]}$ (d)
(a),(d) $\Rightarrow$ FALSE$\Rightarrow \mathcal{P}$(M) is insolvable
$\Rightarrow M \notin \mathcal{M}_{cc}$.

(2) $(P(v_2, v_1) \neq \varnothing) \wedge (P_2 = \varnothing)$: the same proof as (1)

(3) $(P_1 \neq \varnothing) \wedge (P_2 \neq \varnothing) \wedge ((P_1 \neq [0, 0]) \vee (P_2 \neq [0, 0]))$

$(P_1 \neq \varnothing) \Rightarrow v_1 \leq v_2$ (Theorem 6.1) (a)
$(P_2 \neq \varnothing) \Rightarrow v_2 \leq v_1$ (Theorem 6.1) (b)
(a), (b) $\Rightarrow v_2 = v_1$

$\Rightarrow$ $(P_1 = [0, 0]) \wedge (P_2 = [0, 0])$
$\Rightarrow$ FALSE $\Rightarrow \mathscr{P}(M)$ is insolvable
$\Rightarrow$ $M \notin \mathscr{M}_{co}$.

Theorem 6.2 gives a necessary characteristic that a coherent composed scenario model M must have. This characteristic is based on the path-consistency of all couples of vertices of G. We propose in the following a criterion to verify whether a composed scenario model is coherent:

Suppose that G is not semi-connected $\Rightarrow$

$\Rightarrow \exists v_1, v_2 \in V(G), \neg(v_1 \leftrightarrow v_2)$

$\Rightarrow \exists V_1, V_2 \subset V(G), v^1 \in V_1, v^2 \in V_2,$ such as

$\neg(\exists e = (v^1, v^2) \in E(G))$

$\Rightarrow$ M is defined with two sub-scenarios and there is not any temporal relation between these two sub-scenarios.

To compile this type of scenario models, the composed scenario compilation process creates intermediate scenario models corresponding to **all combinations of temporal orders of temporal variables σ(M)** defined within the given scenario model, such as, the temporal constraint graphs representing all these intermediate scenario models are **semi-connected**. An example of such a scenario model is shown in Figure 6.7.

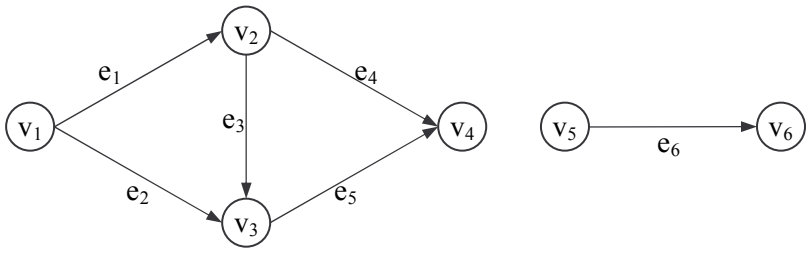


Figure 6.7. This graph containing these 6 vertices is not semi-connected. It is composed of two separated sub-graphs A and B. There are three cases for the compilation of the corresponding scenario model that correspond to three cases: A before B, A during B and B before A.

The verification of the connectivity (i.e. semi-connected) of G can be done by a "*width first search*" algorithm as shown in Figure 6.8. The verification process starts from a vertex $v \in V(G)$, it marks v as a visited vertex. Then, it propagates from v to all vertices that are semi-connected with v. After this process, all vertices semi-connected with v are marked as visited. Thus, the last task consists in verifying if all vertices of V(G) are marked to know if G is semi-connected.

```
Semi_Connected_Verification(G : graph) : Boolean
  v ← a vertex of V(G)
  mark(v)
  L ← {v}
  while L ≠ ∅
     v ← the first element of L
     for all v' ∈ v+ ∪ v-
        if v'is not marked
           mark(v')
           L ← L ∪ {v'}
  if all vertices of V(G) are marked
     exit TRUE
  else exit FALSE
```

Figure 6.8. Verification of the connectivity of a temporal constraint graph representing a composed scenario model.

From now, we work on semi-connected temporal constraint graphs. The verification of graph path-consistency consists of verifying if all couples of vertices of V(G) are path-consistent. To do this, we have to compute the length of all paths on G. Thus, this verification can be realized by the propagation of temporal constraints on G as shown by the next section.

6.3.3 Propagation in Temporal Constraint Graph

This section presents how the scenario model compiler propagates temporal constraints in the graph G corresponding to a composed scenario M. The propagation of temporal constraints in G enables the compilation process to detect the incoherence (e.g. non path-consistent) of G. Moreover, it also enables the compilation process to simplify G to obtain an equivalent graph with a smaller number of constraints. Thus, the recognition of M can be quicker.

Based on Theorem 6.1, we explain the principle of the propagation in a temporal constraint graph (representing a composed scenario model) as follows:

> For all couples of vertices v_1, $v_2 \in V(G)$, if there is a path $p \in P(v_1, v_2)$, thus, it is possible to add a new edge $|p|$ from v_1 to v_2 for E(G) to obtain an equivalent graph.

Figure 6.9. Principle of the propagation on a temporal constraint graph representing a composed scenario model.

We can demonstrate that two graphs G and G' = (V(G), E(G) ∪ {|p|}) are equivalent. Because, the added edge for G' is deduced from E(G) by Theorem 6.1.

To propagate temporal constraints in G, we propose to use the rules (called *triangle rules*) defined in any triangle composed of three vertices as shown in Figure 6.10. The basis of this rule is the transitivity of the operators "<" and "≤" on $\mathcal{I}_p$.

For $v_1, v_2, v_3 \in V(G)$, $v_i \neq v_j$, i,j =1,..,3, $E(v_1, v_2) \neq \emptyset$, $E(v_2, v_3) \neq \emptyset$,

if $E(v_2, v_1) \neq \emptyset$, $e_1 \leftarrow \bigcap_{e_{1,2} \in E(v_1, v_2)} e_{1,2} \cap \bigcap_{e_{2,1} \in E(v_2, v_1)} e_{2,1}$; if not $e_1 \leftarrow \bigcap_{e_{1,2} \in E(v_1, v_2)} e_{1,2}$,

if $E(v_3, v_2) \neq \emptyset$, $e_2 \leftarrow \bigcap_{e_{2,3} \in E(v_2, v_3)} e_{2,3} \cap \bigcap_{e_{3,2} \in E(v_3, v_2)} e_{3,2}$; if not $e_2 \leftarrow \bigcap_{e_{2,3} \in E(v_2, v_3)} e_{2,3}$,

$e \leftarrow (e_1 \oplus e_2)$

$e' \leftarrow e \cap \bigcap_{e_{1,3} \in E(v_1, v_3)} e_{1,3} \cap \bigcap_{e_{3,1} \in E(v_3, v_1)} e_{3,1}$

(1) **($e_1 = \emptyset$)** $\Rightarrow$ v_1 and v_2 are not arc-consistent

(2) **($e_2 = \emptyset$)** $\Rightarrow$ v_2 and v_3 are not arc-consistent

(3) **($e_1 \neq \emptyset$)** $\Rightarrow$ $E(v_1, v_2) \leftarrow \{e_1\}$, $E(v_2, v_1) \leftarrow \emptyset$

(4) **($e_2 \neq \emptyset$)** $\Rightarrow$ $E(v_2, v_3) \leftarrow \{e_2\}$, $E(v_3, v_2) \leftarrow \emptyset$

(i) **$E(v_1, v_3) = E(v_3, v_1) = \emptyset$** $\Rightarrow$ $E(v_1, v_3) \leftarrow \{e\}$

(ii) **($e' \neq \emptyset$)** $\Rightarrow$ $E(v_1, v_3) \leftarrow \{e'\}$, $E(v_3, v_1) \leftarrow \emptyset$

(iii) **($e' = \emptyset$)** $\Rightarrow$ v_1 and v_3 are not path-consistent.

(1, 2, iii) are called *verification rules*.

(3, 4, i, ii) are called *replacement rules*.

Figure 6.10. Triangle rules to verify the path-consistency of a Temporal Constraint Graph representing a composed scenario model and to propagate temporal constraints in this graph. Three examples of the utilization of the rules (i), (ii) and (iii) are shown in Figure 6.11.

Figure 6.11 shows three examples of using the proposed triangle rule to propagate temporal constraints in a Temporal Constraint Graph. In the first case (a), a new edge [6, 19] (i.e. [1, 10] $\oplus$ [5, 9]) is added for the graph (i). In the second case (b), the set of edges from v_1 to v_3 is replaced by only one edge [6, 15] = [1, 10] $\oplus$ [5, 9] $\cap$ [1, 15] $\cap$ [5, 17] (ii). In the last case (c), there is no-edge to be added in the graph, because [1, 10] $\oplus$ [5, 9] $\cap$ [20, 29] = $\emptyset$, thus v_1 and v_3 are not path-consistent (iii).

Figure 6.12 shows the algorithm propagating temporal constraints in G. The propagation process starts at each entering vertex v_1 of $V^+(G)$. Then, it propagates temporal constraints to all successive vertices *succ*(v_1) of v_1. For each $v_2 \in$ *succ*(v_1), if v_1 and v_2 are arc-consistent, all edges between v_1 and v_2 are replaced by only one edge e_1 from v_1 to v_2 corresponding to the conjunction of the old edges between these two vertices. The process continues to propagate temporal constrains to the successive vertices of v_2 by following all the edges starting at v_2. For each $v_3 \in$ *succ*(v_2), if v_2 and v_3 are arc-consistent, all edges between v_2 and v_3 are replaced by only one edge e_2 from v_2 to v_3 corresponding to the conjunction of the old edges between these two vertices. Then, the temporal constraint between v_1 and v_3 can be deduced by using the triangle rule. If there is no-edge between v_1 and v_3, a new edge $e_1 \oplus e_2$ is added for G from v_1 to v_3. If there is/are edge(s) between these two vertices, the process calculates the conjunction $((e_1 \oplus e_2) \cap E(v_1, v_3) \cap E(v_3, v_1))$ and adds an edge corresponding to this conjunction in G from v_1 to v_3 in the case that there is no inconsistency. The process continues until there is no more edge that can be added in G.

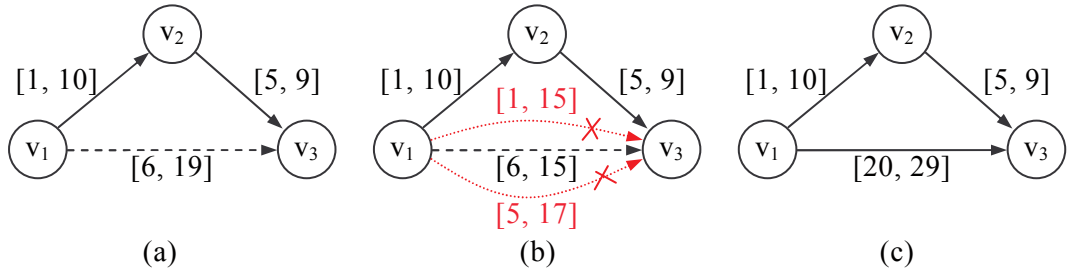


Figure 6.11. Three examples of using the proposed triangle rule: (a) an edge (discontinuous arrow) [6, 19] from v_1 to v_3 is added using the rule (i), (b) an edge (discontinuous arrow) [6, 15] from v_1 to v_3 is added and two edges (dotted arrow) are eliminated using the rule (ii) and (c) no edge is added, v_1 and v_3 are not path-consistent using the rule (iii).

Figure 6.13 shows a "snapshot" of the propagation of temporal constraints on the Temporal Constraint Graph representing the "bank_attack" scenario model. There are a number of edges that are added and there is one initial edge that is eliminated by the triangle rule.

The triangle rule does not allow only to propagate temporal constraints on G, but also to check the arc-consistency between all couples of vertices (triangle rules 1, 2) of V(G) and the path-consistency (triangle rule iii) between them. It also allows replacing two sets of edges between each couple of vertices by an equivalent set containing only one edge. Thus, the graph becomes less complex than the initial one. Moreover, it also allows eliminating any cycle (if it exists) on G (replacement rules).

The propagation of temporal constraints in G makes a graph G' equivalent to G: V(G') = V(G) and all the added edges are deduced by the transitivity of the arithmetical operators "<" and "≤", and the eliminated edges are replaced by deduced edges. Moreover, all temporal constraints (both direct and indirect) between two vertices v and v' of V(G) are included in E(G') and it does not exist more than one edge between two different vertices of V(G'). Thus, G' is called the *complete equivalent* graph of G and denoted G_c.

From now we work on the temporal constraint graph G as the complete equivalent graph G_c of the initial temporal constraint graph representing M.

6.3.4 Temporal Variable Ordering

Our scenario model decomposition method is based on the temporal order of temporal variables of pre-defined scenario models. Thus, the task ordering the vertices of G is important for the compilation of a scenario model M.

The problem of ordering temporal variables is linked to the well-known mathematical problem of graphical topological sort. As shown by the previous section, after propagating all temporal constraints, G_c is a directed acyclic graph.

The simplest topological sort algorithm consists in removing repeatedly all vertices v with $v^+ = \varnothing$ from the graph. The edges belonging to v are also removed, reducing the number of entering edges of adjacent vertices. The process is done until the graph is empty.

After numbering the vertices, $V(G_c)$ is a set of ordered time points representing temporal variables of M. Thus, the temporal variable set $\sigma(M)$ of M is also ordered in time by the ending times of these variables.

```
Propagate(G : graph)
  V ← V⁺(G)
  for all v₁ ∈ V
    for all v' ∈ succ(v₁)
      e ← ⋂_{e_{1,2}∈E(v₁,v₂)} e_{1,2}
      if (E(v₂, v₁) ≠ ∅)   e ← e ∩ ⋂_{e_{2,1}∈E(v₂,v₁)} e_{2,1}
      if (e = ∅)  exit G is not path-consistent
      else
        E(v₁, v') ← {e}    // replace the old edges between two
        E(v', v₁) ← ∅ // vertices by their conjunction
        L ← {e}            // the first edge to follow
        for all e₁ ∈ L
          v₂ ← e₁⁺         // ending vertex of e₁
          for all v₃ ∈ succ(v₂)
            e₂ ← ⋂_{e_{2,3}∈E(v₂,v₃)} e_{2,3}
            if (E(v₃, v₂) ≠ ∅) e₂ ← e₂ ∩ ⋂_{e_{3,2}∈E(v₃,v₂)} e_{3,2}
            if (e₂ = ∅)   exit G is not path-consistent
            else
              E(v₂, v₃) ← {e₂} //replace the old edges between two
              E(v₃, v₂) ← ∅   // vertices by their conjunction
              e' ← (e₁ ⊕ e₂)  // calculate a new edge
              if E(v₁, v₃) = E(v₃, v₁) = ∅   // triangle rule (i)
                E(v₁, v₃) ← {e'}
                L ← L + {e'} // propagate following this edge
              else
                e' ← e' ∩ ⋂_{e_{1,3}∈E(v₁,v₃)} e_{1,3} ∩ ⋂_{e_{3,1}∈E(v₃,v₁)} e_{3,1}
                if (e' = ∅) exit G is not path-consistent
                  if E(v₁, v₃) ≠ {e'}       // new edge
                    E(v₁, v₃) ← {e'}       // triangle rule (ii)
                    E(v₃, v₁) ← ∅
                    L ← L + {e'}//propagate following this edge
    V ← V ∪ succ(v₁) // propagate to the successive vertices of v₁
```

Figure 6.12. Propagation of temporal constraints in a graph using the proposed triangle rules [Figure 6.10]. The propagation process follows all paths starting from all entering vertices of the graph. It applies the triangle rules on the vertices of these paths to recalculate the set of edges between all couples of vertices. This process stops when there is no more path to follow.

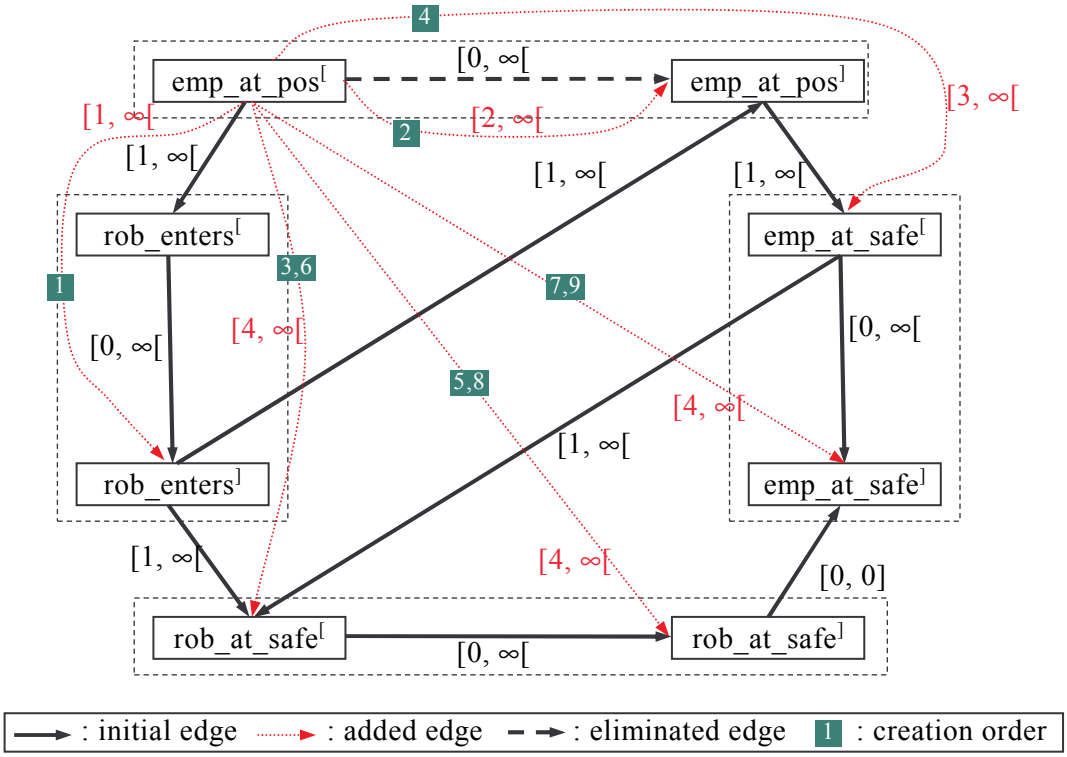


Figure 6.13. A "snapshot" of propagation in the temporal constraint graph representing the "bank_attack" scenario model represented in chapter 3.

6.3.5 Graph Simplification

The complete equivalent graph G_c can contain a number of redundant edges. An edge *e* is considered as a redundancy if *e* can be deduced from the other edges. This type of redundancy is common because the possible edge between each couple of vertices was added in the graph by the propagation of temporal constraints using the triangle rules. Thus, the simplification of G_c is important.

The principle to simplify a temporal constraint graph is to eliminate all edges that can be deduced from other edges. We propose a rule, called *eliminating triangle rule*, to simplify G_c as shown in Figure 6.14.

For $v_1, v_2, v_3 \in V(G)$,
$(E(v_1, v_2) \neq \varnothing \wedge E(v_2, v_3) \neq \varnothing) \Rightarrow$ eliminate $E(v_1, v_3)$.

Figure 6.14. Eliminating triangle rule to simplify $G_c(M)$.

For all $v_1, v_2, v_3 \in V(G)$, if there are an edge e_1 from v_1 to v_2 and another one e_2 from v_2 to v_3, the simplification process eliminates the edge e_3 from v_1 to v_3 (if e_3 exists).

We obtain an equivalent graph, because, $e_3 = e_1 \oplus e_2$ is the result of the propagation of temporal constraints in G_c. In other words, e_3 is deduced from e_1 and e_2 by the triangle rule as shown in the previous section. The process to simplify G_c is shown in Figure 6.15 with a notice that G_c is a directed acyclic graph and its vertices are ordered by temporal constraints as shown in the previous section.

```
GraphSimplification(G : graph)
  for i = 1..(|V(G)| - 2)
     v1 ← vertex number i
     for all v2 ∈ succ(v1)
        for all v3 ∈ succ(v2)
           E(v1, v3) ← ∅          // eliminate the edge from v1 to v3
```

Figure 6.15. This algorithm simplifies a Temporal Constraint Graph by eliminating its redundant edges. An edge is considered as a redundant edge if it can be deduced from other edges. G is a DAG and its vertices are numbered. The simplification process starts at the first vertex. For each vertex v_1, the process applies the eliminating triangle rules [Figure 6.14] to all triangles (v_1, v_2, v_3) by eliminating the edge between v_1 and v_3, where v_2 is a successive vertex of v_1 and v_3 is a successive vertex of v_2.

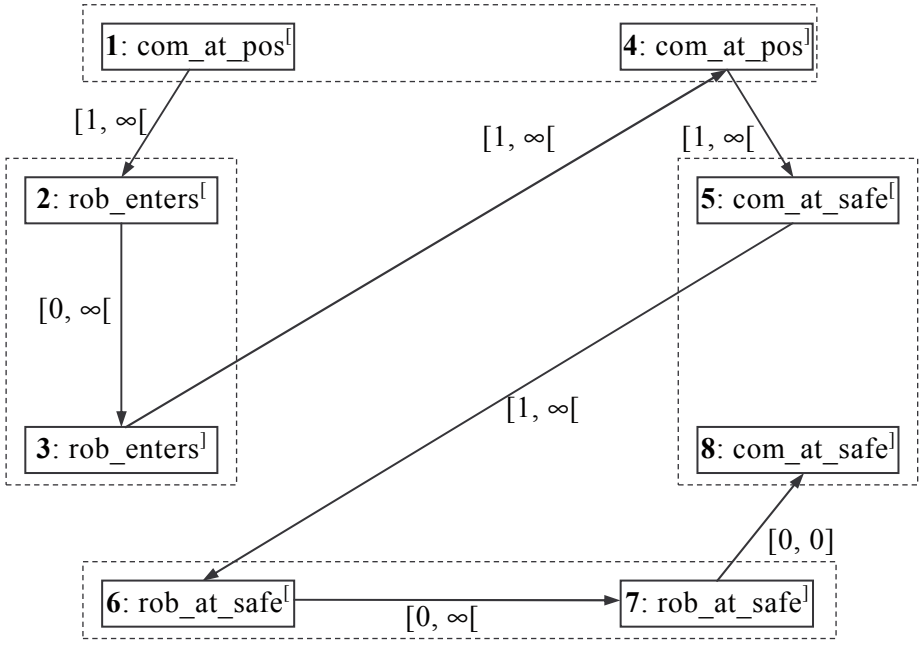


Figure 6.16. The simplified temporal constraint graph of the "Bank_Attack" scenario model. The number preceding the name of a vertex indicates the order of the vertex.

Figure 6.16 shows the simplified temporal constraint graph of the "Bank_Attack" scenario model. A number of edges are eliminated by the simplification process. Seven edges are maintained showing a simpler graph compared to the initial one.

The obtained graph is equivalent to the initial and complete graphs because the eliminated edges are deduced from other edges by using the triangle rule. Moreover, it is normally simpler than the two previous graphs (except in the case where the initial graph is already optimized) because all deducible edges are eliminated. Thus, this graph is called *simplified graph* of G and denoted G_s.

The decomposition of a composed scenario model M_c based on the temporal order of temporal variables $\sigma(M_c)$ using G_s is presented in the next section. The goal is to reduce the processing time by making scenario models simpler.

6.3.6 Scenario Model Decomposition

The goal of the decomposition of scenario models is to obtain composed scenario models containing at the most two sub-scenarios. Previous sections have shown a graph-based method ordering temporally the temporal variables of a composed scenario model M_c. We can underline that G_s is a directed acyclic graph and its vertices are ordered in time (i.e. the order given by the operator "≤" on $\mathcal{T}$). By using this method, $\sigma(M_c)$ is ordered by the ending time of all temporal variables. In other words, $\sigma(M_c)$ is ordered by the operator "≤" on $\mathcal{T}$. This order is total because time elements $\mathcal{T}$ are totally ordered by the operator "≤" as shown in chapter 3.

Definition 6.8: $M_c \in \mathcal{M}_c$, $\sigma(M_c) = \{v_1, v_2\}$, $v_1 \leq v_2$,
v_1, v_2 are respectively called the *start*, the *termination* of M_c and respectively denoted *start*(M_c), *term*(M_c).

Definition 6.9: $M_c \in \mathcal{M}_c$, $v \in \sigma(M_c)$,
$\iota(v) \overset{def}{\cong} \{v' \in \sigma(M_c), \tau(v') \subset \tau(v)\}$,
$\iota(v)$ is the set of all sub-scenarios of M_c occurring during the occurrence of *value*(v).

We have: $\forall v \in \sigma(M_c)$, $v \in \iota(v)$, thus, $|\iota(v)| \geq 1$.

For a temporal variable $v \in \sigma(M_c)$, the compilation process finds $\iota(v)$ by following all paths from the node $v^{[}$ to $v^{]}$ in G_s. A temporal variable v' is included in $\iota(v)$ if $\exists p \in P(v^{[}, v^{]})$, such that, $v'^{[}, v'^{]} \in V(p)$.

We decompose any composed scenario model M_c such as $\sigma(M_c) = \{v_1,\ldots, v_n\}$, $n > 2$, $v_1 \leq v_2 \leq \ldots \leq v_n$ in two different cases:

a) **$\forall v \in \sigma(M_c)$, card($\iota(v)$) = 1**: the decomposition is shown in Figure 6.17. In this case, M_c is decomposed into card($\sigma(M_c)$)-1 intermediate composed scenario models that contain only two sub-scenarios. Figure 6.18 shows an example of this case.

Suppose:
$\sigma(M_c) = \{v_1,\ldots, v_n\}$, $n > 2$, $v_1 \le v_2 \le \ldots \le v_n$ and
$|\iota(v_i)| = 1$, $\forall i = 1,\ldots, n$

We propose to **decompose** M_c into n-1 *intermediate scenario models* $M_1,\ldots, M_{n-1}$ as follow:

a) $\sigma(M_1)$ = the ordered set of two temporal variables $\{v_1, v_2\}$ and
$\sigma(M_i)$ = the ordered set of two temporal variables $\{v^i, v_{i+1}\}$ for $i > 1$, where v^i is a new temporal variable corresponding to a scenario instance of model M_{i-1},

b) $\varphi(M_i) = \varphi(start(M_i)) \cup \varphi(term(M_i)) = \varphi(v_1) \cup\ldots\cup \varphi(v_{i+1})$,

c) $\sigma_F(M_{n-1}) = \sigma_F(M)$ and
$\sigma_F(M_i) = \varnothing$, for $i < n-1$,

d) $\kappa_N(M_{n-1}) = \kappa_N(M)$ and
$\kappa_N(M_i) = \varnothing$, for $i < n-1$,
$\kappa_F(M_{n-1}) = \kappa_F(M)$ and
$\kappa_F(M_i) = \varnothing$, for $i < n-1$,
$\kappa_T(M_1)$ = the set of all constraints corresponding to the edges entering the three nodes $v_1^]$, $v_2^[$, $v_2^]$ of G_s,
$\kappa_T(M_i)$ = the set of all constraints corresponding to the edges entering the two nodes $v_i^[$, $v_i^]$ of G_s, for $i > 1$,

e) $\delta(M_{n-1}) = \delta(M)$ and
$\delta(M_i) = \varnothing$, for $i < n-1$.

Figure 6.17. Decomposition of a composed scenario model that does not contain any temporal variable v such as $|\iota(v)| > 1$. (*)

b) **$\exists v_i \in \sigma(M_c)$, $\iota(v_i) = \{v_j,\ldots, v_i\}$, $j < i$**: the decomposition of M_c is realized in four steps (an example of this case is shown in Figure 6.19):

1) Generate a composed scenario model M_i corresponding to $\iota(v_i)$ as following:

i) $\sigma(M_i) = \iota(v_i)$

ii) $\varphi(M_i) = \bigcup_{v\in\iota(v_i)} \varphi(v)$

(*) $\sigma_F(M)$, $\kappa_N(M)$, $\kappa_T(M)$ and $\kappa_F(M)$ are respectively the sets of forbidden physical-object variables and of non-temporal, temporal and forbidden constraints defined with in M [chapter 3].

$v^[$ and $v^]$ are respectively the lower and upper bounds of the time interval corresponding to the value of a temporal variable v [chapter 3].

iii) $\sigma_F(M_i) = \varnothing$

iv) $\kappa_T(M_i) = \{k \in \kappa_T(M_c), v \in k \Rightarrow v \in \iota(v_i)\}$

v) $\delta(M_i) = \varnothing$

2) Generate a composed scenario model M' corresponding to $\sigma(M_c)$-$\iota(v_i)$ as following:

i) $\sigma(M') = \{v_1,\ldots, v_{j-1}, v_i', v_{i+1},\ldots, v_n\}$, where the value of v_i' will (in the recognition process) correspond to a scenario instance of M_i.

ii) $\varphi(M') = \bigcup_{v \in \sigma(M')} \varphi(v)$

iii) $\sigma_F(M') = \sigma_F(M_c)$

iv) $\kappa_T(M') = \kappa_T(M_c) - \kappa_T(M_i)$

$\kappa_N(M') = \kappa_N(M_c)$

$\kappa_F(M') = \kappa_F(M_c)$

v) $\delta(M') = \delta(M_c)$

We demonstrate in section 6.6 that $M_c \approx M'$.

3) Decompose M_i in both cases a) and b).

4) Decompose M' in both cases a) and b).

This recurrent process also generates n-1 intermediate scenario models. We also call M_{n-1} the most complex generated scenario model.

Figure 6.19 shows the decomposition of "Bank_Attack" scenario model (chapter 3). The initial scenario model is decomposed into three intermediate scenario models. Each of the two first intermediate scenario models contains two temporal variables of the initial model. The third one contains two new temporal variables whose values correspond to scenario instances of the two first intermediate scenario models.

Any edge *e* of $G_s(M_c)$ from v_1 to v_2 is transformed into numerical temporal constraints as shown in Figure 6.20. An edge of the form [a, ∞[corresponds to only one constraint, whereas an edge of the form [a, b] corresponds to two constraints. Moreover, all edges [0, ∞[between two nodes originating from the same temporal variable (of the original scenario model) can be eliminated.

After generating n-1 intermediate scenario models $M_1,\ldots, M_{n-1}$, we also have to modify the constraints distributed in the generated scenario models to link these models. We will show in section 6.6 that the initial model M and the last generated intermediate scenario model M_{n-1} have similar solutions.

```
Scenario(Withdraws_Money,
  PhysicalObjects((client : Person), (e : CashMachine))
  Components(
     (cl_enters  : State inside_zone(client, "Entrance"))
     (cl_waiting : State inside_zone(client, "WaitingZone"))
     (cl_getting_money : Event stays_at(client, "CashMachine1"))
     (cl_exits   : State inside_zone(client, "ExitingZone")) )
  Constraints(
     (sequence(cl_enters, cl_waiting,cl_getting_money, cl_exits))))
```

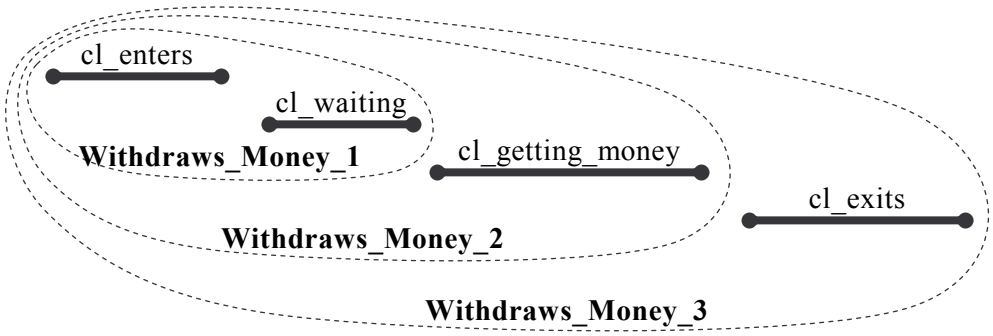


Figure 6.18. A "Withdraws_Money" scenario model is decomposed (following the a) case) into three intermediate scenario models ("Withdraws_Money_1", "Withdraws_Money_2" and "Withdraws_Money_3"). Each intermediate scenario model is composed of two temporal variables.

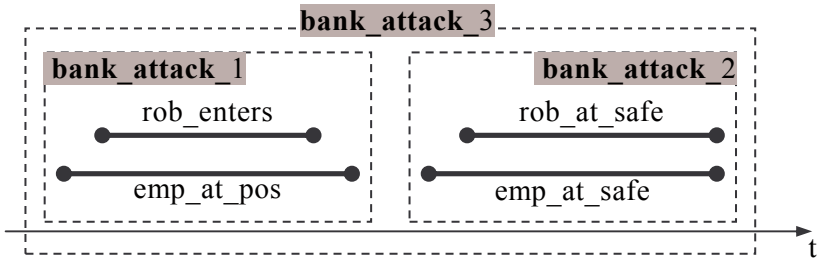


Figure 6.19. A "bank_attack" scenario model is decomposed (following the b) case) into three (bank_attack_1, bank_attack_2, bank_attack_3) intermediate scenario models by the composed scenario compiler. Each intermediate scenario model is composed of two sub-scenarios.

Figure 6.21 shows three intermediate scenario models generated by the compilation of the "bank_attack" scenario model defined in chapter 3. The initial scenario model "bank_attack" is equivalent to the last generated intermediate scenario model "bank_attack_3". The value of the first temporal variable (i.e. the *start*) of the second intermediate scenario model "bank_attack_2" corresponds to a scenario instance of the first intermediate scenario model "bank_attack_1". The temporal constraint distributed in "bank_attack_1" is obtained from the constraint "(rob_enters ***during*** com_at_pos)" of the initial scenario model. The third intermediate scenario model

contains constraints, the operands of which are modified to adapt to the new scenario models.

$v_1 = v^{[}$ and $v_2 = v^{]}$		v_1 and v_2 correspond to two bounds of two different temporal variables	
Edge	Constraints to be added	Edge	Constraints to be added
$[0, \infty[$	no constraint added	$[0, \infty[$	$(v_1 \leq v_2)$
$[a, \infty[$	(duration of $v \geq a$)	$[a, \infty[$	$(v_1 + a \leq v_2)$
[a, b]	(duration of $v \geq a$) (duration of $v \leq b$)	[a, b]	$(v_1 + a \leq v_2)$ $(v_2 \leq v_1 + b)$

Figure 6.20. Transformation of direct edges into numerical temporal constraints. There are two cases: (1) v_1 and v_2 correspond to the two bounds (starting and ending time points) of the same temporal variable and (2) v_1 and v_2 correspond to the two bounds of the different temporal variables.

```
Scenario(bank_attack_1,
  PhysicalObjects((com : Person), (rob : Person))
  Components(
     (rob_enters : Event changes_zone(rob, "Entrance", "Counter"))
     (com_at_pos : State inside_zone(com, "Back_Counter")) )
  Constraints(
     (com_at_pos[ < rob_enters[)
     (rob_enters] < com_at_pos]))   )

Scenario(bank_attack_2,
  PhysicalObjects((com : Person), (rob : Person))
  Components(
     (rob_at_safe : State inside_zone(rob, "Safe"))
     (com_at_safe : State inside_zone(com, "Safe")) )
  Constraints(
     (com_at_safe[ < rob_at_safe[)
     (com_at_safe] = rob_at_safe])) ) )

Scenario(bank_attack_3,
  PhysicalObjects((com : Person), (rob : Person))
  Components(
     (att_1 : Scenario bank_attack_1(com, rob))
     (att_2 : Scenario bank_attack_2(com, rob)) )
  ForbiddenScenarios(
     (any_in_bank : State inside_zone(any_p, "Bank")))
  Constraints(
     (att_1] < att_2[)
     (any_p ≠ com)
     (any_p ≠ rob)
     (any_in_bank[ ≤ (start of att_1)[)
     ((start of att_1)[ ≤ any_in_bank]) )
  Decisions(Alert("Bank Attack!!!"))  )
```

Figure 6.21. Three intermediate scenario models are generated for the compilation of the scenario model "bank_attack". The initial model and the "bank_attack_3" scenario model have similar solutions.

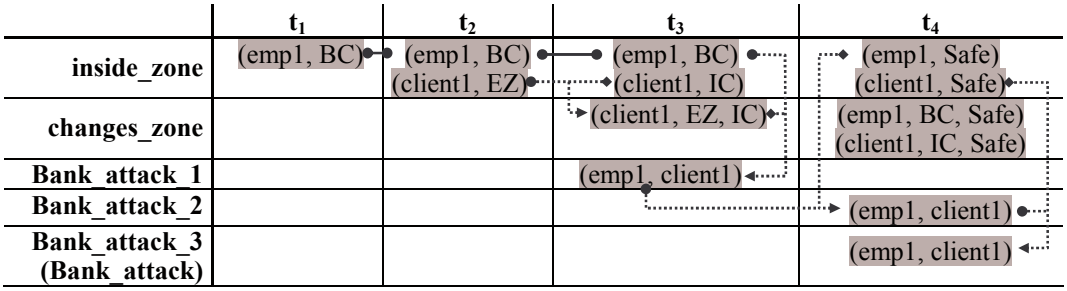
The recognition of compiled scenario models is described in the next sections. The gain in processing time is due to the search algorithm: we just try several times to link two scenario instances instead of trying to link together a whole set of combinations of scenario instances.

```
CS_Recognition_CompiledScenario(T : trigger)
  Mc    ← μ(T) // Scenario model to be recognized
  (termination of Mc) ← ξ(T)   // Last sub-scenario instance
  for each o ∈ domain(start of Mc)
     (start of Mc) ← o   // First sub-scenario instance
     if Satisfied(κT(Mc)) // verify the temporal constraints
        φ(Mc) ← value(φsub(Mc)) // instanciate physical obj. var.
        if Satisfied(κN(Mc)) //verify the nontemporal constraints
           // the given scenario model is recognized
           sc ← CreateInstance(Mc, value(φ(Mc)), value(ltv))
           Execute(δ(Mc), sc)
           Store(sc)
           CreateTrigers(Mc, sc)
```

Figure 6.22. Compiled composed scenario recognition algorithm. The algorithm takes as input a trigger and attempts recognizing the scenario model M_c contained in the trigger. The recognition process first instantiates the termination of the M_c. Then it attempts assigning the start of M_c with a scenario instance such that all temporal constraints of M_c are satisfied. If all temporal constraints are satisfied, the recognition process instantiates all physical object variables of M_c with the physical objects involved in its start and termination. Finally, the recognition process has to verify whether all non-temporal constraints of M_c are satisfied.

6.4 Compiled Scenario Recognition

The compiled scenario model recognition algorithm is similar to the non-compiled scenario model recognition algorithm. The recognition of a compiled scenario model M is triggered by a trigger *Tr*, which has been generated when a scenario instance s_t of the scenario model corresponding to *term*(M) has been recognized. The trigger *Tr* contains the scenario model M and the scenario instance s_t. The recognition of M consists of (1) instanciating *start*(M) by a scenario instance previously recognized, (2) verifying whether all temporal constraints κ_T(M) are satisfied, (3) instanciating the physical-object variables φ(M) by φ(*value*(*start*(M))) ∪ φ(*value*(*term*(M))) and then (4) verifying whether all non-temporal constraints κ_N(M) are satisfied. If all non-temporal constraints κ_N(M) are satisfied, the scenario model M is recognized (we underline that M does not contain any forbidden sub-scenario and the processing of forbidden sub-scenario will be presented in section 6.5). To finish the recognition of M, the recognition process performs another process to store the new recognized scenario to be used later to recognize other composed scenario models. Figure 6.22 shows the recognition process of compiled composed scenario models.



BC : Back_Counter, **IC**: Infront_Counter, **EZ**: Entrance_Zone.

♦ : the scenario instance that triggers the recognition of a composed scenario ended by it.

• : the start of a composed scenario.

•—• : two scenario instances are merged into one.

····· : two scenario instances construct a more composed scenario instance.

Figure 6.23. Four steps of the recognition of a "Bank_attack" scenario instance.

An example of the recognition of the Bank_attack scenario is shown in Figure 6.23. The recognition of this scenario is composed of four steps corresponding to four instants $t_1,\ldots, t_4$. In the presented situation, there are 12 scenario instances recognized during the recognition of the main scenario Bank_attack. There are 10 of these recognized scenarios used to recognize the main scenario. The other two scenario instances (i.e. two scenario instances of changes_zone scenario recognized at the instant t_4) are not used to recognize the main scenario because they are not matched any component of the "Bank_attack" scenario. At time t_1, the employee (cashier) is at his/her position behind the counter. At time t_2, a client enters and the employee is still at his/her position. Then at time t_3, the client moves to the front of the counter and the employee is still at his/her position. Finally, both of them arrive at the safe door. This sequence of actions corresponds to a bank attack situation.

6.5 Forbidden Sub-Scenarios

This section shows how the recognition algorithm processes the forbidden sub-scenarios defined within a composed scenario M. A forbidden sub-scenario instance s_f is defined within a composed scenario model M by a forbidden temporal variable v_f that is not allowed to occur during the recognition of M. Thus, the recognition process of M has to verify whether there is such a scenario instance s_f that occurs and satisfies the forbidden constraints involving v_f. If it exists such a scenario instance s_f, M is not recognized.

For each compiled composed scenario model M that contains at least one forbidden sub-scenario, $\sigma_F(M) \neq \varnothing$:

(1) The recognition process creates a scenario model M^+ that does not contain any forbidden sub-scenario and corresponds to M without forbidden sub-scenario as follow:

$$\varphi(M^+) = \varphi(M)$$

$$\sigma(M^+) = \sigma(M)$$

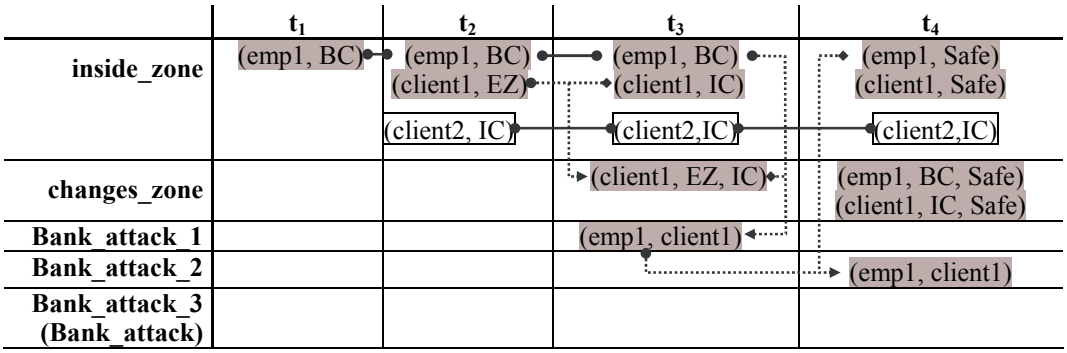
$$\sigma_F(M^+) = \varnothing$$

$$\kappa(M^+) = \kappa_N(M) \cup \kappa_T(M)$$

(2) To recognize M, the recognition process (i) first attempts to recognize M^+ by the algorithm shown in section 6.4 and then (ii) eliminates all the solutions of *solutions*(M^+) that satisfy the forbidden constraints $\kappa_F(M)$ defined within M.

We are using a classical constraint resolution method to eliminate a solution $s^+ \in \mathscr{R}(M^+)$ satisfying the forbidden constraints $\kappa_F(M)$. The recognition process attempts to find a combination s^- of scenario instances corresponding to the forbidden temporal variables $\sigma_F(M)$ such that s^+ and s^- satisfy the forbidden constraints $\kappa_F(M)$. If such a combination s^- is found, s^+ is not a solution of $\mathscr{R}(M)$ and is eliminated.

Figure 6.24 shows a situation similar to the situation shown in Figure 6.23. However, in this case, no "Bank_attack" scenario instance is recognized, because the presence of the third person (i.e. "client") in the bank violates the forbidden constraints defined within this scenario model.



BC : Back_Counter, **IC**: Infront_Counter, **EZ**: Entrance_Zone.

♦ : the scenario instance that triggers the recognition of a composed scenario ended by it.

• : the start of a composed scenario.

•—• : two scenario instances are merged into one.

····· : two scenario instances construct a more composed scenario instance.

Figure 6.24. The "Bank_attack" scenario is not recognized.

To validate the proposed composed scenario recognition algorithm we first demonstrate the algorithm by a formal way in section 6.6 and then realize experiments [chapter 7] to verify whether the proposed algorithm is correct and to estimate its limitations. The demonstration presented in the next section focuses on proving that the algorithm can recognize correctly all pre-defined scenario models.

6.6 Proof of the Proposed Composed Scenario Recognition Algorithm

This section has for objective to prove that the proposed composed scenario recognition algorithm recognizes correctly all pre-defined composed scenario models. In

other words, it can find all solutions for each composed scenario model $M \in \mathcal{M}_c$ and the found solutions are correct. Suppose that M_{n-1} is the last intermediate scenario model generated from the compilation of M. To demonstrate this, we (1) first prove that $\mathcal{P}(M)$ and $\mathcal{P}(M_{n-1})$ have corresponding solutions, then (2) we prove that the algorithm recognizes all scenario instances of M_{n-1}.

To demonstrate (2), we show that the proposed scenario recognition algorithm can recognize correctly simple cases of composed scenarios (including compiled composed scenario) that contain at the most two sub-scenarios. For such a scenario model M_c, $\sigma(M_c) = (v_1, v_2)$. v_2 is already assigned a value because it takes as value the scenario instance contained in the trigger that triggers the recognition of M_c. This is a recursive process which is initialized by the recognition of elementary scenarios. While recognizing M_c, the algorithm attempts all possible values of v_1, thus, the algorithm can recognize all solutions of M_c. Moreover, the algorithm verifies all constraints of M_c, thus, all found solutions are correct. Now, (2) is true, thus, we only need to prove (1).

To prove (1), we distinguish two cases: (1) $\sigma(M)$ does not contain any temporal variable v such as $|t(v)| > 1$, and (2) $\sigma(M)$ contains such a temporal variable.

Suppose:

$M \in \mathcal{M}_c$

$\sigma(M) = (v_1, \ldots, v_n)$, $n > 2$, (if $n \leq 2$, M does need to be decomposed)

$v_1 \leq \ldots \leq v_n$.

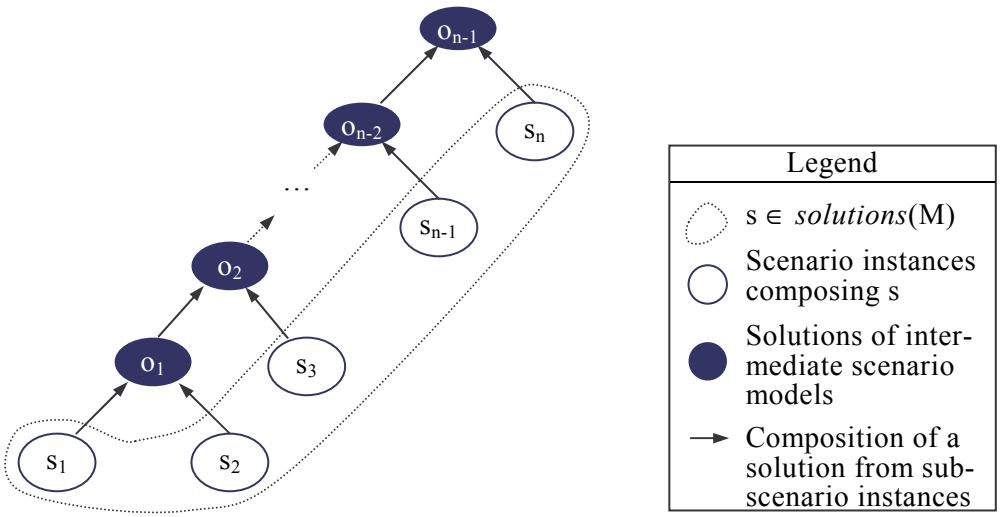


Figure 6.25. Construction of solutions of intermediate scenario models generated from a solution of the initial scenario model M. The root node corresponds to a solution of the biggest intermediate scenario model M_{n-1} generated from the compilation of M. The leaf nodes correspond to the scenario instances composing a solution of M.

By using the proposed scenario decomposition method, M_{n-1} and M have the same set of physical-object variables, non-temporal constraints, forbidden-scenarios, forbidden constraints and decisions. Thus, to prove $\mathscr{P}(M)$ and $\mathscr{P}(M_{n-1})$ have corresponding solutions, we need to focus only on the set of temporal constraints of these two scenario models.

a) The first case: $\forall v \in \sigma(M), |\iota(v)| = 1$

Suppose: $M_1,\ldots, M_{n-1}$ are n-1 intermediate scenario models generated from the compilation of M.

(i) $\mathbf{s = (s_1,\ldots, s_n) \in solutions(\mathscr{P}(M)) \Rightarrow s}$ **corresponds to a solution of** $\mathbf{\mathscr{P}(M_{n-1})}$

$(s_1,\ldots, s_n) \in solutions(\mathscr{P}(M))$

$\Rightarrow o_1 = (s_1, s_2)$ satisfies all temporal constraints between v_1 and v_2

$\Rightarrow o_1$ satisfies all temporal constraints of $\kappa_T(M_1)$

$\Rightarrow o_1 \in solutions(\mathscr{P}(M_1))$ (I)

$\Rightarrow o_2 = (o_1, s_3) \in solutions(\mathscr{P}(M_2))$ (the same as (I))

……..

$\Rightarrow \mathbf{o_{n-1} \in solutions(\mathscr{P}(M_{n-1}))}$ (the same as (I))

Thus, all intermediate scenario models have a solution corresponding to s. In particular, o_{n-1} is a solution of the intermediate scenario model M_{n-1}.

The construction of solutions $o_1,\ldots, o_{n-1}$ of intermediate scenario models generated from the compilation of M from a solution *s* of M is shown in Figure 6.25. A solution of the last intermediate scenario model M_{n-1} (i.e. the root node) corresponds to a binary tree of scenario instances, where the leaf nodes of this tree are the sub-scenario instances composing *s*. The other nodes of this tree correspond to solutions of other generated intermediate scenario models.

(ii) Now, we need to prove that if $\mathbf{o_{n-1} = (o_{n-2}, s_n) \in solutions(\mathscr{P}(M_{n-1}))}$ is a solution of M_{n-1} then $\mathbf{o_{n-1}}$ **corresponds to a solution of** $\mathbf{\mathscr{P}(M)}$

$o_{n-1} = (o_{n-2}, s_n) \in solutions(\mathscr{P}(M_{n-1}))$

$\Rightarrow o_{n-2} = (o_{n-3}, s_{n-1}) \in solutions(\mathscr{P}(M_{n-2}))$,

……

$o_1 = (s_1, s_2) \in solutions(\mathscr{P}(M_1))$

$\Rightarrow s = (s_1,\ldots, s_n)$ satisfies all temporal constraints of $\kappa_T(M_1) \cup \ldots \cup \kappa_T(M_{n-1}) = \kappa_T(M)$

$\Rightarrow s \in solutions(\mathscr{P}(M))$.

b) The second case: $\exists v_i \in \sigma(M), \iota(v_i) = \{v_j,\ldots, v_i\}, j < i$

Suppose: M_i and M' are two intermediate scenario models generated by steps 1) and 2) in the b) case of the decomposition of the composed scenario model M. We need to prove that $\mathscr{P}(M')$ and $\mathscr{P}(M)$ have corresponding solutions.

(i) If $s = (s_1,\ldots, s_n)$ is a solution of $\mathscr{P}(M)$ then s corresponds to a solution of $\mathscr{P}(M')$

$s = (s_1,\ldots, s_n) \in solutions(\mathscr{P}(M))$

$\Rightarrow o_i = (s_j,\ldots, s_i)$ satisfies all temporal constraints between the temporal variables of $\iota(v_i)$,

$\Rightarrow o_i \in$ *solutions*($\mathscr{P}(M_i)$)

$\Rightarrow o' = (s_1,\ldots, s_{j-1}, o_i, s_{i+1},\ldots, s_n)$ satisfies all temporal constraints of $\kappa_T(M')$

$\Rightarrow o' \in$ *solutions*($\mathscr{P}(M')$).

The construction of solutions $o_i,\ldots, o'$ of intermediate scenario models generated from the compilation of M from a solution *s* of M is shown in Figure 6.26. A solution of the scenario model M' (i.e. the root node) corresponds to a tree of scenario instances, where the leaf nodes of this tree are the sub-scenario instances composing *s*. The other node of this tree corresponds to a solution of the scenario model M_i.

(ii) If $o' = (s_1,\ldots, s_{j-1}, o_i, s_{i+1},\ldots, s_n)$ is a solution of $\mathscr{P}(M')$ then o' corresponds to a solution of $\mathscr{P}(M)$

$o' = (s_1,\ldots, s_{j-1}, o_i, s_{i+1},\ldots, s_n) \in$ *solutions*($\mathscr{P}(M')$)

$\Rightarrow o_i = (s_j,\ldots, s_i) \in$ *solutions*($\mathscr{P}(M_i)$)

$\Rightarrow (s_j,\ldots, s_i)$ satisfies all temporal constraints $\kappa_T(M_i)$

$\Rightarrow s = (s_1,\ldots, s_{j-1}, s_j,\ldots, s_i, s_{i+1},\ldots, s_n)$ satisfies all temporal constraints $\kappa_T(M_i) \cup \kappa_T(M')$

$\Rightarrow$ s satisfies all temporal constraints $\kappa_T(M)$

$\Rightarrow s \in$ *solutions*($\mathscr{P}(M)$).

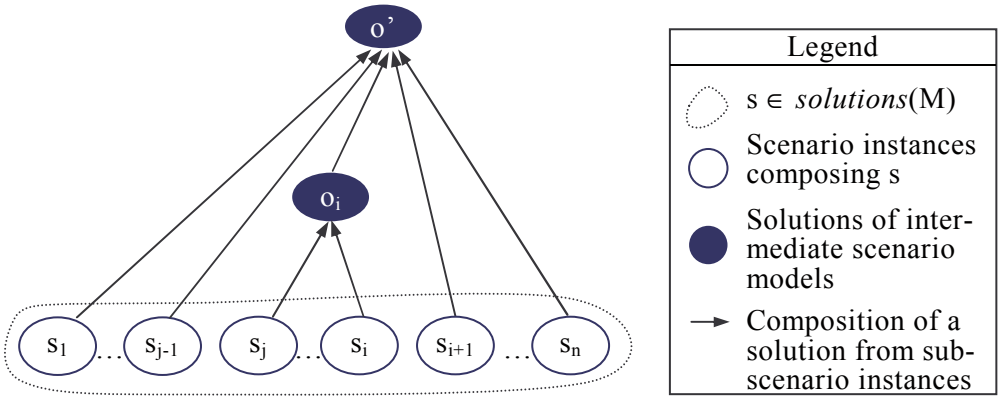


Figure 6.26. Construction of solutions of two intermediate scenario models generated from a solution of the initial scenario model M. The root node corresponds to a solution of the scenario model M'. The leaf nodes correspond to the scenario instances composing a solution of M. The node o_i corresponds to a solution of the scenario model M_i

By this demonstration, we can conclude that the proposed composed scenario recognition algorithm can recognize correctly scenario models pre-defined by experts. We now present in the next section an estimation of the complexity of this algorithm to know if it can recognize the pre-defined scenario models in real-time.

6.7 Complexity of the Proposed Composed Scenario Recognition Algorithm

To estimate the complexity of the composed scenario recognition algorithm, we consider that the verification of a constraint is the computational unit. Thus, the estimation of the complexity of the proposed algorithm consists of calculating how many times the algorithm has to verify the defined constraints to recognize a composed scenario model.

For a composed scenario model M_c containing two temporal variables $\sigma(M) = \{v_1, v_2\}$, the number of constraint verifications is linear with the value domain of v_1, because v_2 has been assigned a value at previous step.

As shown in section 6.3, a composed scenario model $M \in \wp\mathscr{M}_c$ is decomposed into intermediate scenario models that are less complex than M to be recognized faster. Suppose:

(i) $M \in \wp\mathscr{M}_c$
(ii) $\sigma(M) = \{v_1,\ldots, v_n\}$, $n > 2$
(iii) $v_1 \leq v_2 \leq \ldots \leq v_n$
(iv) $M_1, M_2,\ldots, M_{n-1}$ are the intermediate scenario models generated for the compilation of M.

To recognize M, the recognition process has to recognize all the n-1 scenario models $M_1,\ldots, M_{n-1}$. Thus, to calculate the number of constraint verifications implied by the recognition of M, we first calculate the number of constraint verifications implied by the recognition of each intermediate scenario model. Then, the number of constraint verifications implied by the recognition of M is calculated based on these constraint verification numbers.

Suppose:

- $r = |\kappa(M)|$, is the number of constraints defined within M,
- $r_i = |\kappa_T(M_i)|$, is the number of temporal constraints distributed into the intermediate scenario model M_i,
- m is the maximal number of scenario instances of a scenario model,
- $n_f = |\sigma_F(M)|$, is the number of forbidden variables defined within M,
- $r_f = |\kappa_F(M)|$, is the number of forbidden constraints defined within M.

We first calculate the number of constraint verifications implied by the recognition of M without the verification of forbidden constraints.

We have:

The number of constraint verifications implied by the recognition of an intermediate scenario model M_i is equal to mr_i.

Consequently, the maximal number of constraint verifications implied by the recognition of M is:

$$C_0 \approx m\left(\sum_{i=1}^{n-1} r_i + \text{card}(\kappa_N(M))\right)$$

$$C_0 \approx m(\text{card}(\kappa_T(M)) + \text{card}(\kappa_N(M)))$$

$$C_0 \approx m(r - r_f)$$

Second, we calculate the number of constraint verifications implied by the verification of forbidden constraints. For each combination of forbidden sub-scenario instances, the recognition implies at the most r_f constraint verifications. The maximal number of combinations of forbidden sub-scenario instances is m^{n_f}. Thus, the maximal number of constraint verifications implied by the verification of forbidden constraints is:

$$C_1 = m^{n_f} r_f$$

We have the maximal number of constraint verifications implied by the recognition of M is:

$$C \approx C_0 + C_1$$

$$C \approx m(r - r_f) + m^{n_f} r_f$$

The complexity of the composed scenario recognition algorithm is $O(mr + m^{n_f} r_f)$.

In other words, the proposed algorithm is linear with the number of constraints defined within a composed scenario model and also linear with the maximal number of scenario instances of a scenario model. However, it leads to a combinatorial problem with the number of forbidden sub-scenarios defined within a given composed scenario model. By experiments shown in chapter 7, there are few scenario models that are defined with forbidden sub-scenarios. Moreover, the number of forbidden sub-scenarios defined within a scenario model is small (one or two). Thus, the proposed algorithm can still recognize in real-time (at video cadence) pre-defined scenario models. We have also estimated the processing time of the proposed algorithm in practical terms by realizing experiments [chapter 7].

6.8 Optimization of the Algorithm

This section shows the evolution of the proposed scenario recognition algorithm over different enhancements. We started the initial version of the algorithm based on the algorithm proposed by Rota (2001) in his PhD thesis [Figure 6.27]. This initial version integrates a naive constraint resolution technique and has three main drawbacks: (1) exponential explosion with the number of physical object variables defined within scenario models, (2) exponential explosion with the number of sub-scenario variables defined within scenario models and also (3) combinatorial explosion with the total number of entities (i.e. physical objects, recognized scenarios) existing in the system.

Working in the context of scenario recognition for video interpretation with a hard challenge, we have started to enhance the recognition algorithm by proposing novel techniques to recognize interesting human behaviors. The first proposed technique had for speed-up the access to already recognized scenarios. We proposed to stored already recognized scenarios in a *tree of scenario instances* [chapter 4]. This proposition could answer the question posed by (3). Thus, we obtained an algorithm that is in general linear with the number of entities of each type (e.g. detected persons, scenario instances of “inside_zone” involving a person *p* and a zone *z*). This proposition leaded to a publication in the proceedings of the KES’2002 conference.

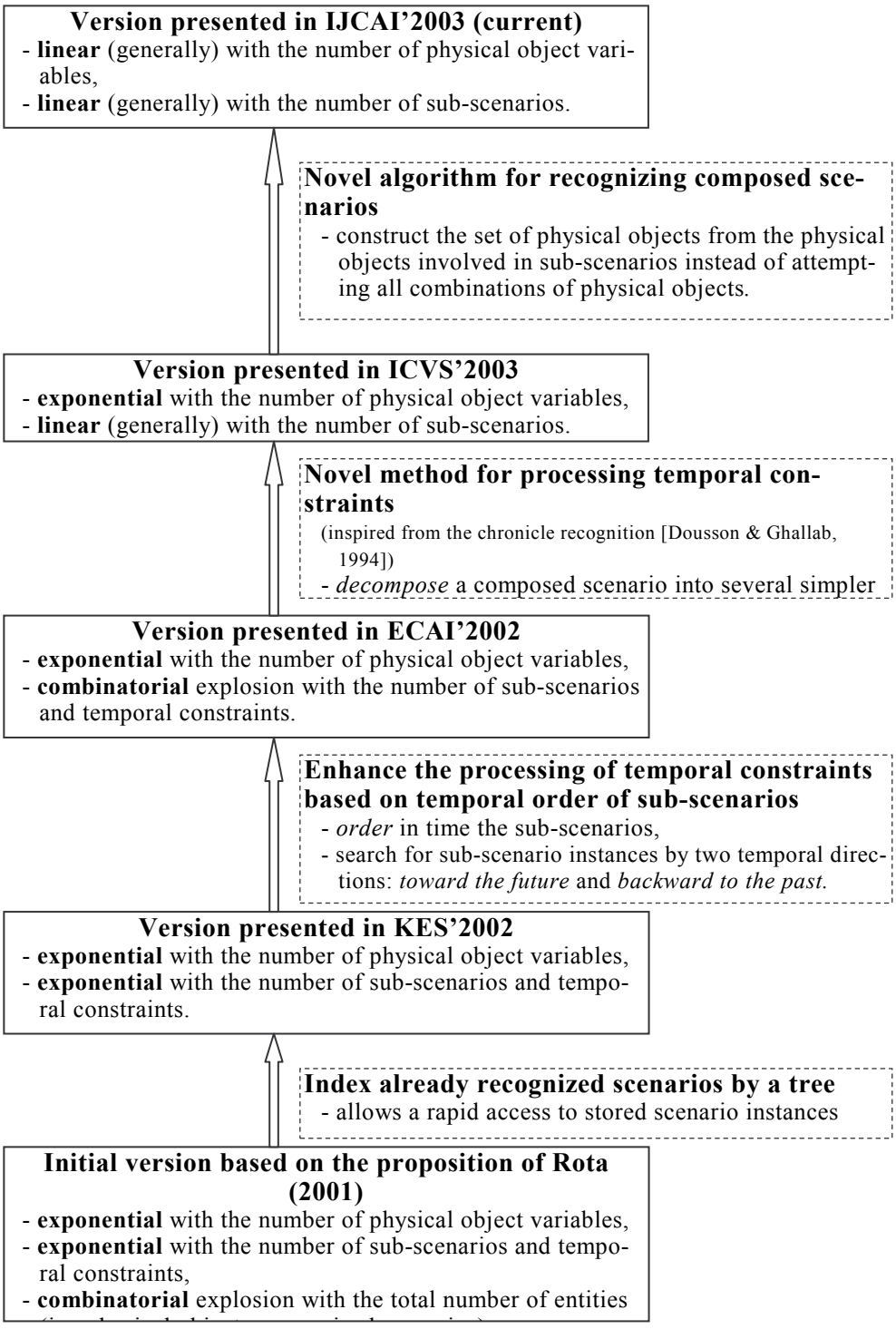


Figure 6.27. Evolution of the proposed scenario recognition algorithm.

Continuing to study the problem of temporal scenario recognition, we found that the principal drawback of the recognition algorithm was in the resolution of temporal

constraints. Thus, to cope with this problem, we proposed an enhancement for the processing of temporal constraints by ordering in time the sub-scenarios of the scenario models to be recognized. This temporal order can conduct the search for scenario instances following two directions: from the past toward the present and also backward to the past. This enhancement can prevent the recognition algorithm to be exponential with the number of sub-scenarios but still leads the algorithm to a combinatorial explosion problem (this proposition is published in the proceedings of the workshop "Modelling and Solving Problems with Constraints" belonging to the ECAI'2002 conference). Thus, this proposition did not satisfy our research objective. Then, we attempted to propose another technique to process temporal constraints. The current version of the algorithm including the compilation of composed scenarios search scenario instances in only one direction (from the past to the present) instead of two directions as the algorithm presented in ECAI'2002.

By studying the Store Partially Recognized Scenarios algorithms and specially the chronicle recognition algorithm proposed by Dousson & Ghallab (1994), we proposed to process temporal constraints by decomposing a composed scenario into intermediate scenario models such that each intermediate scenario model is composed of one or two sub-scenarios. This decomposition can conduct the recognition algorithm to be generally linear with the number of sub-scenario variables defined within a composed scenario to be recognized because, while attempting to recognize the given scenario, the recognition process has to search only one scenario instance for a temporal variable of generated scenario models. This proposition also conducted our research to a publication in the proceeding of the ICVS'2003 conference.

The version of the algorithm published in the ICVS'2003 conference was much better than the initial algorithm (in terms of processing time). Nevertheless, we continued to studied the scenario recognition problem, because the recognition algorithm was still exponential with the number of physical object variables defined within scenario models. To solve this problem, we proposed to recognize a solution of a composed scenario model by building a solution from its sub-scenario instances instead of attempting all combination of physical object variables defined within the given scenario model as in the classical algorithm. This proposition makes the algorithm to be generally linear with the number of physical object variables and has been published in the proceedings of the IJCAI'2003 conference.

The current version of our scenario recognition algorithm is efficient (in term of processing time) to recognize scenarios defined by experts as shown by the complexity estimation [6.7] and by experimental results as that are shown in chapter 7. Nevertheless, our research motivation is still not stopped. We will present our future research subjects (to optimize the recognition algorithm) in the global conclusion of the thesis.

6.9 Scenario Knowledge Base Coherence Verification and Simplification

6.9.1 Scenario Knowledge Base Coherence Verification

Chapter 4 has shown that a scenario model M can be recognized if and only if M is coherent. Thus, the pre-defined scenario models used in an Automatic Video Interpretation System can be recognized if and only if its Scenario Knowledge Base is coherent. The next section has for objective to verify whether a Scenario Knowledge Base of an Automatic Video Interpretation System is coherent.

The Scenario Knowledge Base Coherence Verification consists in verifying whether all scenario models contained in a given Scenario Knowledge Base are coherent. To verify whether a scenario model M is coherent, the scenario compiler has to check:

(1) $\kappa(M)$ is a coherent set of constraints,

(2) $\forall M' \in \wp\mathscr{M}, (M' \subset M) \Rightarrow (M \not\subset M')$, inclusive coherence,

(3) $\forall M' \subset M, M' \in \wp\mathscr{M}_c$.

The Scenario Knowledge Base Coherence Verification process realizes a Scenario Model Coherence Verification on each scenario model of the given Scenario Knowledge Base. Thus, for the verification of each scenario model M, the process does not need to verify the condition (3).

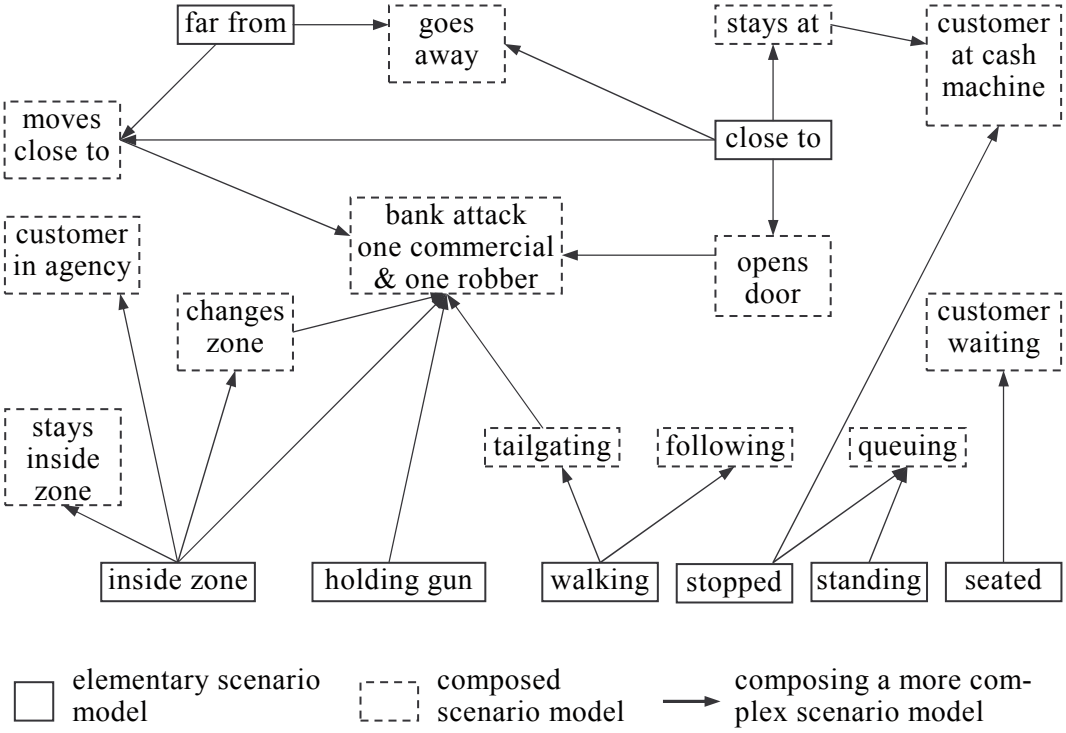


Figure 6.28. The graph of scenario models created for the coherency verification of the scenario knowledge base given in chapter 3. This graph contains no cycle, thus the corresponding scenario knowledge base is inclusive coherent.

For each $M \in \wp\mathscr{M}$, if $\kappa_F(M)$ is not coherent, then, the recognition of M is independent of $\kappa_F(M)$. Thus, the process does not need to verify whether the set of forbidden constraints $\kappa_F(M)$ is coherent. Thus, to verify (1), the Scenario Model Coherence Verification process has to verify only whether two sets of constraints $\kappa_N(M)$ and $\kappa_T(M)$ are coherent. The coherence verification of $\kappa_T(M)$ is realized by the propagation of temporal constraints in G(M) as shown in 6.3. To verify whether $\kappa_N(M)$ is coherent, we must first define what is the meaning of coherency for non-temporal relations. This can be possible by using an ontology where the meaning of the rela-

tions explicates this coherency. If it is the case, we can use existing techniques verifying the consistency of constraint sets for Constraint Satisfaction Problem Solvers, for example AC4 algorithm [Mohr & Henderson, 1986]. To verify the coherency of $\kappa_N(M)$, the scenario coherence verification process builds a graph. The nodes of this graph correspond to the non-temporal variables $\varphi(M)$ defined within M and the edges of this graph correspond to the non-temporal relations between those variables. The AC4 algorithm is used to verify whether this graph is consistent.

A graph based technique can be used to verify (2) –the inclusive coherence–. The Scenario Knowledge Base Coherence Verification process creates a graph. The nodes of this graph correspond to the scenario models of the given Scenario Knowledge Base and the edges of this graph correspond to the inclusive relation between scenario models. For two nodes corresponding to two scenario models M_1 and M_2, if $M_1 \subset M_2$ then there is an edge from M_2 to M_1. For example in Figure 6.28, inside_zone $\subset$ changes_zone, thus there is an edge from inside_zone to changes_zone. The verification of the inclusive coherence consists in verifying whether it exists a cycle in the created graph. If it exists a cycle in this graph, the given Scenario Knowledge Base is not coherent. Figure 6.28 shows an example of the graph of scenario models created for the coherency verification of the scenario knowledge base presented in chapter 3.

Figure 6.29 shows the process verifying the inclusive coherence for the coherence verification of a given scenario model. The presented algorithm is called in the form *Inclusive_Coherence*(<scenario name>, ∅). The process verifies the inclusive coherence of a given scenario model M by a recurrent **depth first search** algorithm. The process also attempts to verify the inclusive coherence of all sub-scenario models of M by the same method. If all scenario models of $\mu_{sub}(M)$ are inclusive coherent, M is inclusive coherent.

```
Inclusive_Coherence(M : Scenario_Model, l : List)
{
     if (M ∈ l)  Exit FALSE
     l ← l + {M}
     for all M' ∈ μsub(M)
        if (not Inclusive_Coherence(M', l)) Exit FALSE
     l ← l - {M}
     Exit TRUE
}
```

Figure 6.29. Scenario Knowledge Base Coherence Verification process verifies whether a scenario model is inclusive coherent.

For an Automatic Video Interpretation System be able to use efficiently a Scenario Knowledge Base, we do not only focus on verifying its coherence but also focus on eliminating redundancies existing within the given Scenario Knowledge Base. The interest of the Scenario Knowledge Base simplification that is presented in the next section is to reduce the processing time of the scenario recognition algorithm. An example of the redundancies in a scenario knowledge base is shown in Figure 6.30.

6.9.2 Scenario Knowledge Base Simplification

The scenarios of a Scenario Knowledge Base used in an Automatic Video Interpretation Application can be defined by experts and also automatically generated by the

scenario compilation process. Thus, the redundancy of scenario models is possible. There are three types of redundancy: (1) several scenario models are equivalent, (2) a scenario model is included in another scenario model and (3) two scenario models have a common part. The redundancy of scenario models in a Scenario Knowledge Base can cause the problem of processing time for the recognition process, because there are more scenario models to be recognized. Thus, the elimination of redundancies in the Scenario Knowledge Base is necessary. This task is called *Scenario Knowledge Base Simplification*.

Example: a given Scenario Knowledge Base $\mathscr{S}$ contains five elementary scenario models $e_1,\ldots, e_5$ and three composed scenario models S_1, S_2 and S_3. S_1 is defined as a sequence of four temporal variables corresponding to four components respectively of scenario models e_1, e_2, e_3 and e_4. S_2 is defined as a sequence of three temporal variables corresponding to three components respectively of scenario models e_1, e_2 and e_5. S_3 is defined as a sequence of three temporal variables corresponding to three components respectively of scenario models e_1, e_2 and e_3. These three composed scenario models have a common part ($e_1 \le e_2$). Moreover, S_3 is defined as a part of S_1. This shows an example of redundancy of the given Scenario Knowledge Base $\mathscr{S}$.

$$S_1 = (e_1 \le e_2 \le e_3 \le e_4)$$

$$S_2 = (e_1 \le e_2 \le e_5)$$

$$S_3 = (e_1 \le e_2 \le e_3)$$

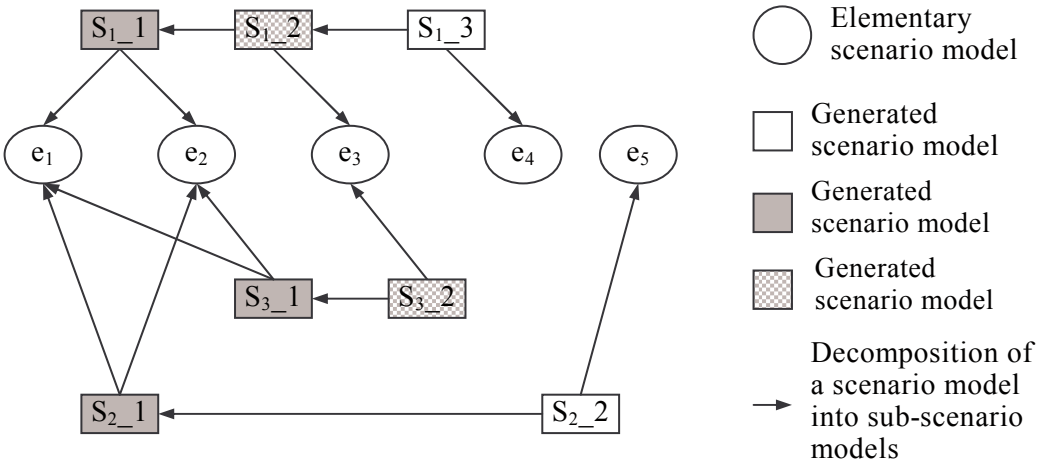


Figure 6.30. Redundant scenario models in a scenario knowledge base. The generated models (S_1_1, S_1_2, S_1_3 for the compilation of S_1; S_2_1, S_2_2 for the compilation of S_2 and S_3_1, S_3_2 for the compilation of S_3) of the dark color and gray color are equivalent: S_1_1 ≈ S_2_1 ≈ S_3_1 and S_1_2 ≈ S_3_2.

The scenario model compiler generates three intermediate scenario models (S_1_1, S_1_2 and S_1_3) for the compilation of S_1, two intermediate scenario models (S_2_1 and S_2_2) for the compilation of S_2 and also two intermediate scenario models (S_3_1 and S_3_2) for the compilation of S_3. Figure 6.30 shows the composition of intermediate scenario models generated for the compilation of the scenario models con-

tained in the given Scenario Knowledge Base $\mathscr{S}$. This figure shows the situation of Scenario Knowledge Base $\mathscr{S}$ after the scenario compilation process. In this situation, $\mathscr{S}$ contains several scenario models that are equivalent: S_1_1, S_2_1 and S_3_1 are equivalent; S_1_2 and S_3_2 are equivalent.

```
SKB_Simplification(𝒮 : Set of Scenario Models)
{
  for all S ∈ 𝒮
    for all S' ∈ 𝒮 - {S}      // find all scenario models
      if (S ≈ S')             // equivalent to S
        for all S" ∈ 𝒮 - {S} - {S'} // modify all scenario
          if S' ∈ μsub(S")          // models defined with S'
            Replace S'by S in S"
        𝒮 ← 𝒮 - {S'}  // eliminate S' from the SKB
}
```

Figure 6.31. Scenario Knowledge Base (SKB) Simplification process.

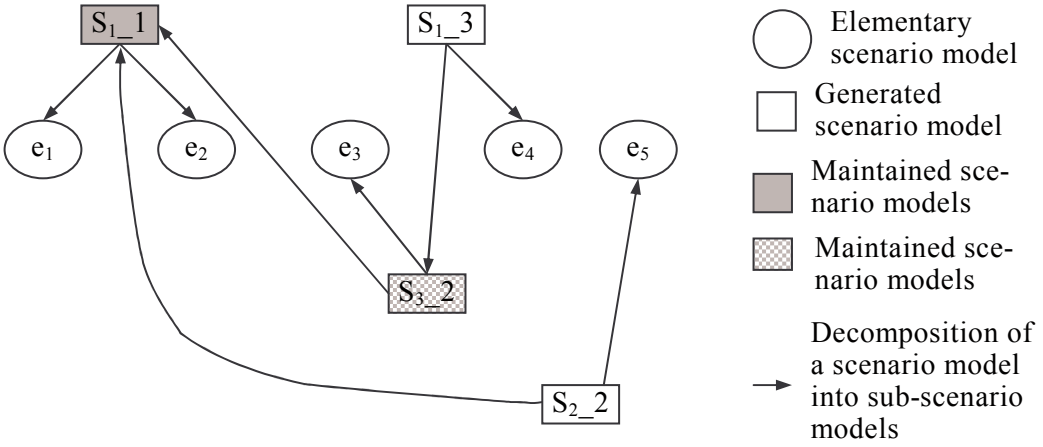


Figure 6.32. The simplified Scenario Knowledge Base corresponding to the Scenario Knowledge Base shown in Figure 6.30.

Section 6.3 has showed that a compiled composed scenario model contains at the most two sub-scenarios. Thus, after all scenario models of a given Scenario Knowledge Base $\mathscr{S}$ are compiled, it exists (if it exists) only the first type of redundancy – several scenario models are equivalent-. Thus, the simplification of a Scenario Knowledge Base consists in (1) finding all sets of equivalent scenario models and (2) for each set of equivalent scenario models, keeping only one scenario model and eliminating the others. To eliminate a scenario model, the simplification process has also to modify all the scenario models defined with a temporal variable corresponding to a scenario instance of the eliminated scenario model to adapt these scenario models to the new situation of the Scenario Knowledge Base. Figure 6.31 shows the process of simplifying a Scenario Knowledge Base.

Figure 6.32 shows the simplified Scenario Knowledge Base corresponding to the Scenario Knowledge Base shown in Figure 6.30. This figure shows the maintained scenario models after the Scenario Knowledge Base simplification process. Several scenario models are modified because their original sub-scenario models are eliminated. The Scenario Knowledge Base simplification process has eliminated three scenario models: S_2_1, S_3_1 and S_1_2. The scenario models S_1_1, S_2_1 and S_3_1 are equivalent, then, the simplification process has maintained only S_1_1 and eliminated the two other scenario models because S_1_1 is the first found (in the given Scenario Knowledge Base) of these three models. This process has eliminated S_2_2 instead of S_3_2, although S_2_2 is the first found of these two scenario models, because, S_3_2 is defined (by experts) with a decision set and S_2_2 is not.

6.10 Temporal Scenario Recognition Synthesis

Figure 6.33 presents a global view of scenario model processing and utilization. The scenario models are (1) first defined by experts (user-interaction level) of application domains (end-users) using the proposed scenario description language. Then, (2) a parser analyzes (automatic off-line level) these scenario models to create a scenario knowledge base containing all pre-defined scenario models. (3) Third, a scenario model compiler checks the consistency inside the models and reorganizes the knowledge defined within these models (automatic off-line level). (4) Fourth, the scenario knowledge base containing all these models is processed (automatic off-line level) by another module to verify the global coherency of the knowledge base and eliminate the redundancy existing in the knowledge base. (5) Finally, the simplified coherent scenario knowledge base is used (automatic on-line level) for the scenario recognition process.

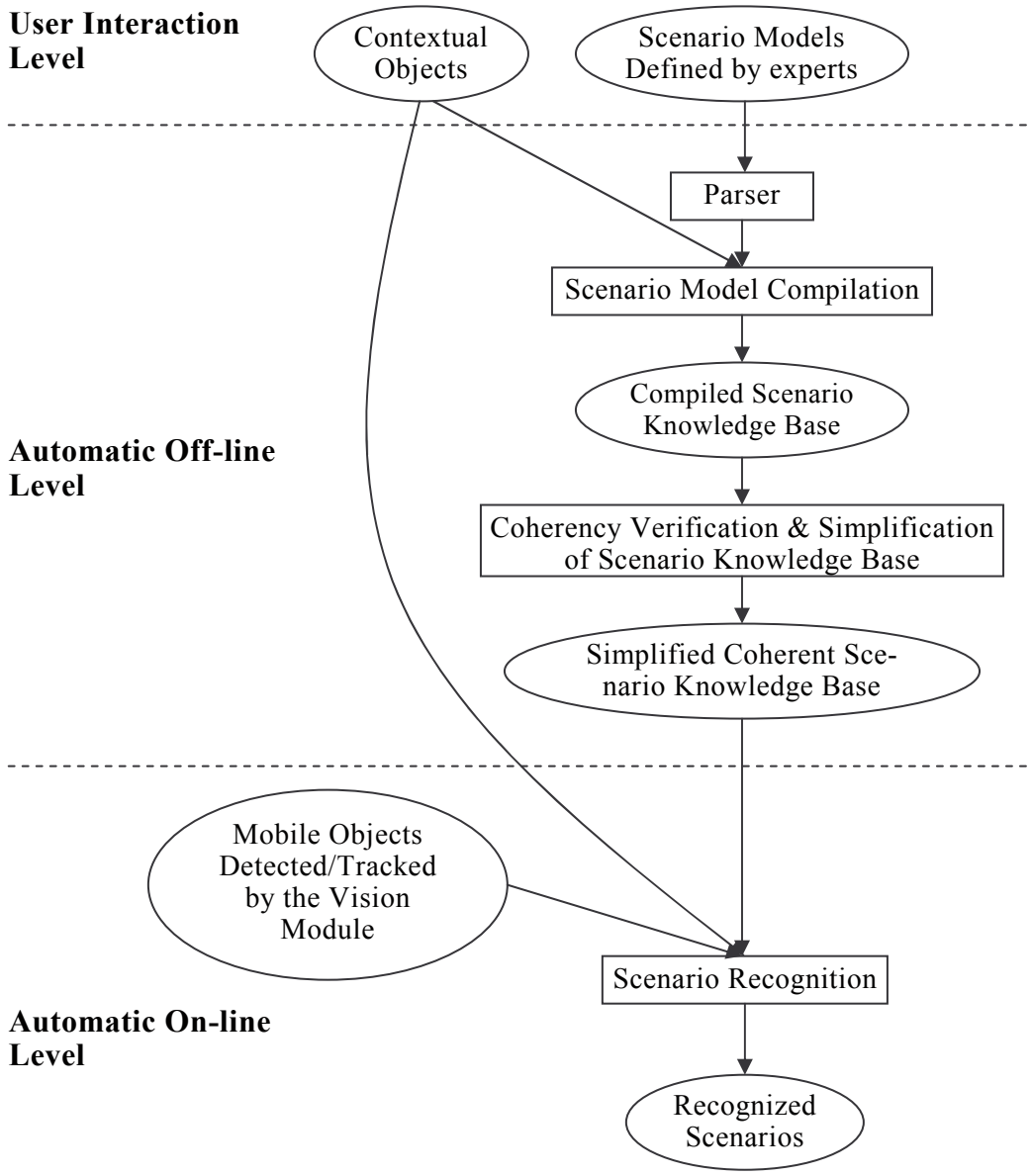


Figure 6.33. The scenario recognition process: (1) the scenario models defined by experts are analyzed by o parser (off-line) then (2) processed (off-line) by a scenario model compiler to check the consistency inside the models and to reorganize the knowledge defined within these models. (3) Third, the scenario knowledge base containing all these models is processed (off-line) by another module to verify the global coherency of the knowledge base and eliminate the redundancy existing in the knowledge base. (4) Finally, the simplified coherent scenario knowledge base is used (on-line) for the scenario recognition process.

6.11 Conclusion on Temporal Scenario Recognition

This chapter has shown an original temporal scenario recognition algorithm for the Automatic Video Interpretation. To deal with this issue, we have first proposed a

new schema to recognize at each instant scenario models pre-defined by experts. At each instant, the scenario recognition process first tries to recognize all elementary scenario models. Then, it also tries to recognize all composed scenario models that are triggered by the recognition of another scenario model. A composed scenario model is triggered if a scenario instance corresponding to its last temporal variable is recognized at the same instant. Each time a scenario is recognized, the recognition process performs another process to maintain the set of recognized scenario instances by adding a new element to this set or extending the time interval of an already recognized scenario if it is possible. Moreover, to speed up the process accessing already recognized scenarios, we have proposed to store them in a tree such that a path from the root node to a leaf node corresponds to the physical objects involved in the scenario instant stored in that leaf node.

Secondly, we have studied a necessary and sufficient condition for a scenario model to be recognized. We have demonstrated that a scenario model can be recognized if and only if it is coherent. Thus, to ensure that all scenario models (defined in the scenario knowledge base for an automatic video interpretation system) can be recognized, we have proposed to verify the coherence of those scenario models in an initial step.

Thirdly, to speed-up the recognition process of an elementary scenario model, we have proposed to distribute the constraints defined within an elementary scenario model M_e into its physical objects (in an initial step) such that a constraint f can be evaluated when and only when the corresponding physical object variable is assigned a value. Thus, the elementary scenario recognition algorithm eliminates immediately all combinations of physical objects that cannot be a solution of the given elementary scenario model M_e. Thus, it does not lead the recognition process to a combinatorial explosion as the state of the art algorithm [Rota & Thonnat, 2001].

Fourthly, we have also proposed a temporal constraint resolution technique to recognize in real-time (video cadence) temporal scenario models predefined by experts. The proposed algorithm is efficient for processing temporal constraints as well as for combining several actors defined within a given scenario M. By efficient we mean that the recognition process is linear with the number of sub-scenarios and with the number of physical object variables defined within M in most cases.

Finally, we have presented a method to optimize the recognition algorithm. Our algorithm is proposed as a Stores all Totally Recognized Scenarios algorithm. We have optimized the algorithm to obtain an efficient algorithm (in terms of processing time) by taking advantages of Stores all Partially Recognized Scenarios to prevent a combinatorial explosion algorithm.

Part III

Experiments and Results

Chapter 7. Experiments and Results

This chapter first presents our method for validating the scenario recognition algorithm. Then, it describes the experiments that we have realized in different applications. Finally, it presents an evaluation of the proposed scenario recognition algorithm through an analysis of experimental results.

7.1 Introduction

To validate the proposed scenario recognition algorithm, we have first integrated the algorithm with a vision module to obtain an operational interpretation system and then we have built a test framework. This test framework is dedicated for Automatic Video Interpretation Systems and more especially for scenario recognition algorithm. We have used it to perform four types of tests: (1) on recorded videos taken in a bank agency and in two metro stations (one in Belgium and one in Spain) to verify whether the algorithm can correctly recognize the predefined scenario models, (2) on live videos acquired on-line from cameras installed in an office, in a metro station and in a bank agency to verify whether the algorithm can work robustly on a long-time period in continuous mode, (3) on recorded videos taken in a bank agency and also on simulated data to study how the complexity of the algorithm depends on the complexity of scenario models (i.e. number of sub-scenarios and of physical-object variables defined within scenario models) and (4) on simulated data to study how the complexity of the algorithm depends on the complexity of the scene (i.e. the number of people in the scene).

The next section presents the proposed test framework for conducting the experiments and evaluating the scenario recognition algorithm.

7.2 Test Framework for Automatic Video Interpretation Systems

This section presents a *simulation test framework* for Automatic Video Interpretation. We have built the test framework composed of two systems: interpretation system and test system. As shown in chapter 1, the Automatic Video Interpretation consists in recognizing pre-defined scenarios describing human behaviors from video sequences. Thus, to test the interpretation system (specially the scenario recognition algorithm), the test framework has been conceived to realize the following tasks [Figure 7.1]:

(1) **visualize scenarios described by experts**: it is also important for the experts of the application domain (e.g. agent of security in a metro) to visualize the scenarios that they describe. The **test system** takes as input scenarios defined by experts and visualizes them through 3D animations or videos.

(2) **visualize scenarios recognized by an interpretation system**: it is important for the developer (e.g. expert in vision and scenario recognition) to visualize each step of the scenario recognition process. The **interpretation system** takes as input a video sequence and attempts to recognize scenarios evolving in the given video. Then, the **test system** takes as input the recognized scenarios and visualizes them through 3D animations.

(3) **evaluate the couple interpretation-test system**: it is important to verify the coherence between the interpretation system and the test system. The test process starts by (2) and generates videos corresponding to the output 3D animations of (2). Then, it makes a loop by taking as input the generated videos. Inside each loop, it attempts to verify whether the scenarios recognized in this loop are equivalent to the input scenarios.

(4) **validate interpretation systems**: establish the limits and robustness of interpretation systems by simulating test videos. The test system takes as input a scenario model (defined by experts) and visualizes it with different variations (e.g. different light conditions, variations of temporal relations). Then, it generates video sequences corresponding to those variations. After that, the interpretation system attempts to recognize scenarios evolving in generated videos. Finally, the framework has to verify whether the recognized scenarios are equivalent to the input scenario.

(5) **validate the temporal scenario recognition algorithm**: establish the limits and robustness of the recognition algorithm independently of vision results. The test system takes as input a scenario model defined by experts and generates a flow of mobile objects corresponding to different steps of the given scenario model. Then, the scenario recognition module takes as input the generated flow of mobile objects. It attempts to recognize scenarios corresponding to the behaviors of these mobile objects. Finally, the test framework has to verify whether the recognized scenarios are equivalent to the input scenario.

To describe the proposed test framework, we first focus in the next section on the simulation for testing the processing time of the scenario recognition algorithm.

7.3 Simulation for Testing Recognition Algorithm Processing Time

The fifth functionality of the test framework has for objective the testing the scenario recognition algorithm independently of the vision module. The goal is to avoid that errors caused by the vision module [section 7.4] prevent the scenario recognition module. Thus, we can estimate the limitations of the recognition algorithm on input data free of vision error.

The simulation of input data for the scenario recognition module can be realized as it has been for the simulation of videos for the whole video interpretation system as described in the annex 2. The only difference between these two simulations is: in this simulation, the test system generates a set of parameters of individuals instead of a set of videos as in the simulation of videos for the whole video interpretation system [Figure 7.1].

Thanks to this simulation, we can test (independently of the vision module) the proposed scenario recognition algorithm. Moreover, we can estimate the limitations of the algorithm as shown in section 7.5.

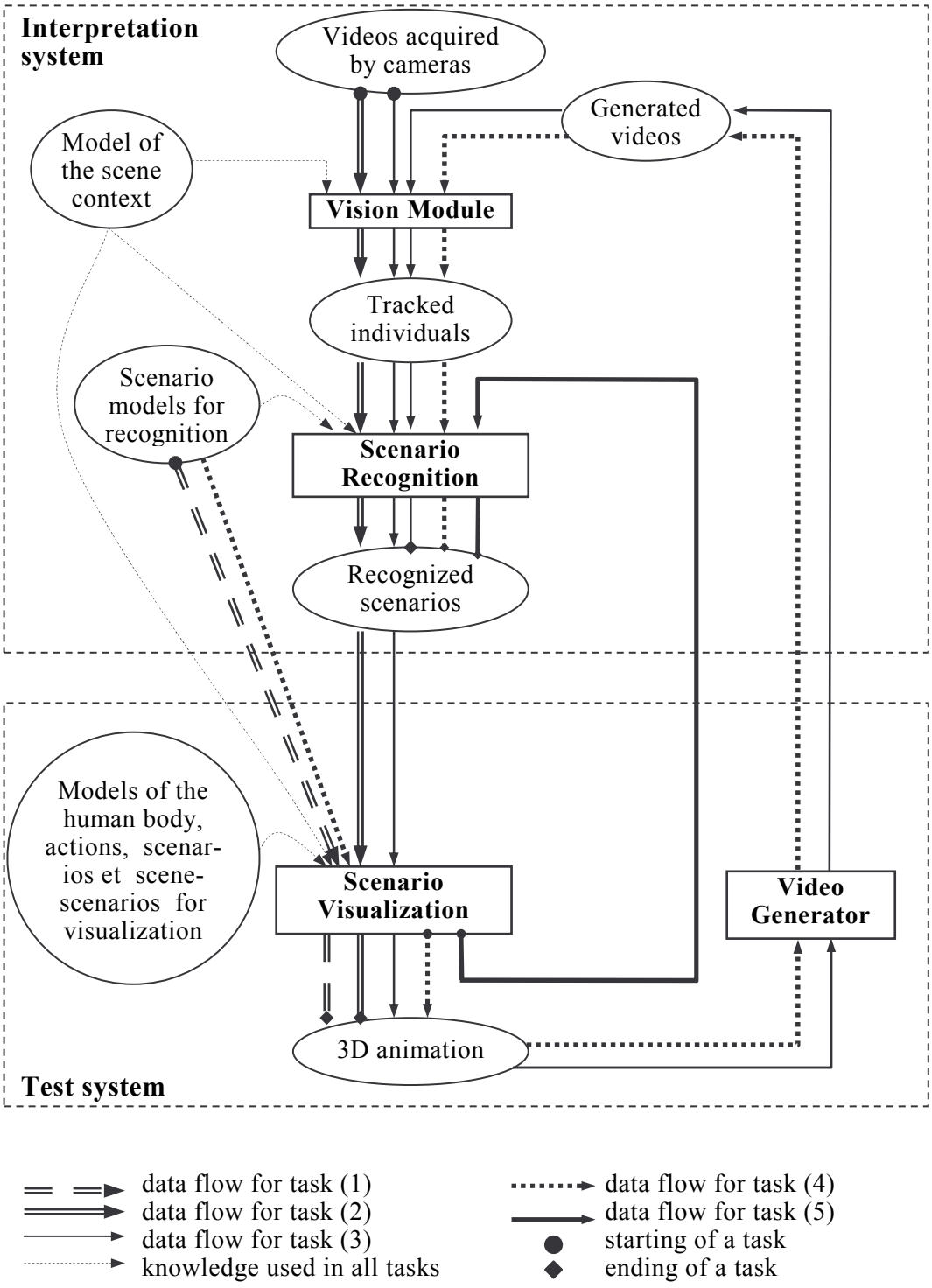


Figure 7.1. The test framework for Automatic Video Interpretation systems has to realize five tasks [section 7.2]: (1) visualize scenarios described by experts, (2) visualize scenarios recognized, (3) evaluate the couple interpretation-test system, (4) validate interpretation systems and (5) validate temporal scenario recognition algorithm.

7.4 Correctness

This section shows the experiments realized on recorded videos to verify whether the proposed scenario recognition algorithm recognizes correctly several types of "Bank attack" scenarios previously described in chapter 3 in a bank agency and several types of "Vandalism against a ticket machine" scenarios in a metro station.

For testing/evaluating the recognition algorithm with bank video sequences, experts of the French CASSIOPEE project first modeled several variations of the original "Bank attack" scenario presented in chapter 3. They slightly modified (e.g. modifications of temporal constraints to obtain different temporal order of variables) the original scenario model to obtain a set of scenario models to better model the "bank attack" scenario. Then they took 11 video sequences in a bank agency to test the whole video interpretation system. The context of the observed environment (interesting objects with their semantics) in these tests is shown in Figure 7.2. There are several objects (e.g. zones, tables, cash machines) in the bank agency, but in this experiment, the experts are interested in only four zones concerning "Bank attack" scenarios: "GATE" associated with a logical name "entrance" where clients enter/exit, "BACK BRANCH" associated with a logical name "back_counter" is the position of the employee behind the counter, "INFRONT BRANCH" associated with a logical name "counter" where clients make bank transactions and "SAFE" associated with a logical "safe" which is the most important/security zone.

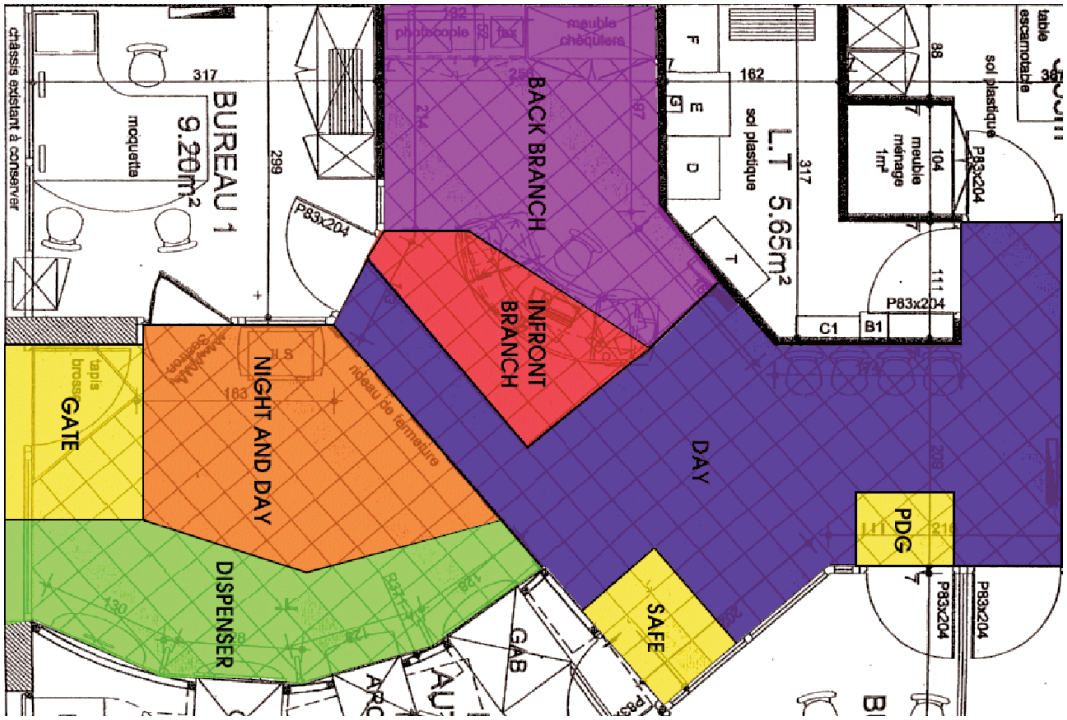


Figure 7.2. Main contextual objects of a bank agency (image given by the French CASSIOPEE project). "GATE" is the zone where clients enter/exit. "BACK BRANCH" is the position of the employee behind the counter. "INFRONT BRANCH" is the zone in front of the counter for clients to make bank transactions. "SAFE" is the most important/security zone of the bank agency.

	Number of tested sequences	Average number of persons/frame	Recognition rate (% of sequences)	Number of false alarms
Bank cam. 1	10	4	80	0
Bank cam. 2	1	2	100	0
Metro cam. 2	4	2	100	0

Table 7.1. The recognition of temporal scenarios in videos of a bank agency and of a metro station. Pre-defined scenario models are recognized in most of the cases. The scenario recognition algorithm fails to recognize some scenarios when the vision module does not detect people in the scene. These data are taken from the French project CASSIOPEE and the European project CASSIOPEE.

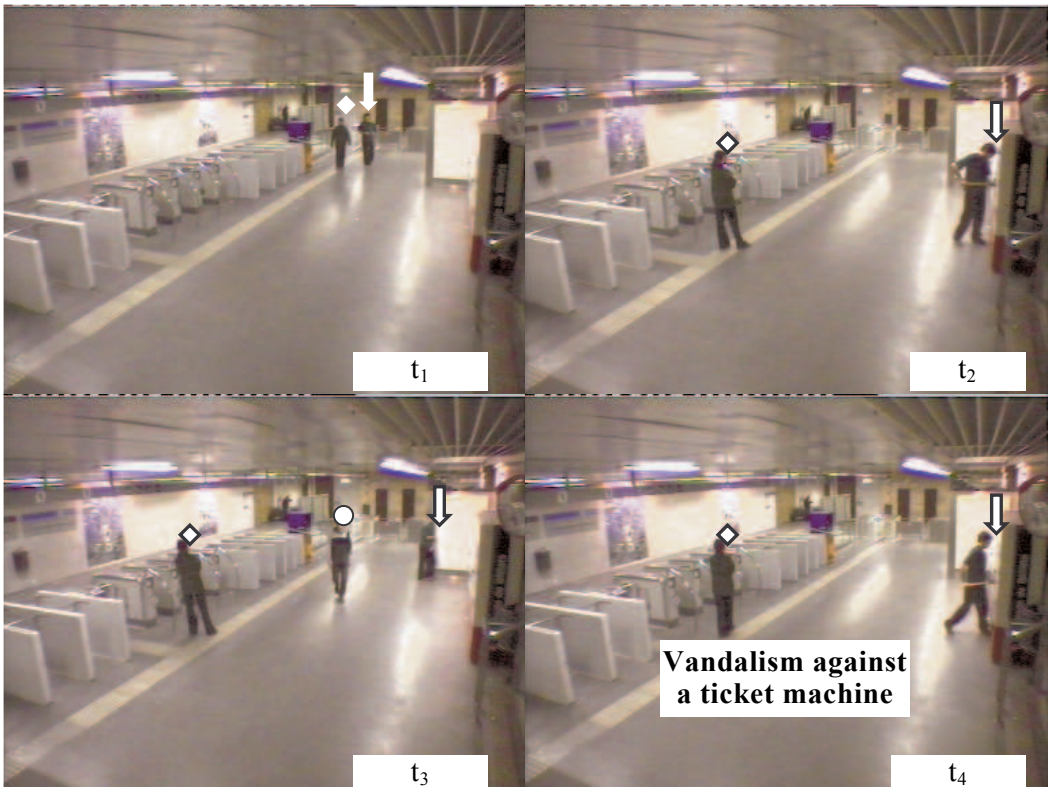


Figure 7.3. Four steps of a "Vandalism against a ticket machine" scenario used in metro surveillance application [ADVISOR project]: (1) at time t_1, two persons enter the interesting zone of a metro station, then, (2) at time t_2, the first person moves close to the ticket machine and tries to "break" the machine while the second person stays near the machine for looking around. (3) At time t_3, there is a third person who arrives in the interesting zone, thus the first person (who is breaking the machine) goes away. (4) Finally, at time t_4, the first person returns to the machine to break it.

After modeling 12 variations of the original "Bank attack" scenario and 10 intermediate scenario models, the experts of the CASSIOPEE project ran the CASSIOPEE video interpretation system (which includes the proposed scenario recognition algo-

rithm) on the 11 recorded bank video sequences. At the initial phase (off-line), the scenario compiler of the system generates 40 intermediate scenario models. Thus the recognition algorithm takes 62 scenario models as input (in online mode, need to be real time). During the execution of the system, we stored all recognized scenarios and compared them with all scenario occurrences in the input video sequences. As shown in Table 7.1, the predefined scenarios were correctly recognized in most of the cases. The interpretation system fails to recognize some scenarios only when the vision module misses to detect the people in the scene (i.e. vision errors). These data are taken from the French CASSIOPEE project.

For evaluating/testing the scenario recognition algorithm on metro video sequences, the experts of the European ADVISOR project have realized tests similar to those realized in the bank agency. They first modeled 15 variations of the original scenarios: "Vandalism against a ticket machine" [Figure 7.3], "Fighting", "blocking", "overcrowding" and "jumping over the barrier". The original vandalism scenario involves two persons in general, one looking around to check whether somebody is coming and the other one attempting to break the ticket machine. The scenario is defined with four steps. (1) At time t_1, two persons enter the interesting zone of a metro station. (2) Then at time t_2, the first person moves close to the ticket machine and tries to "break" the machine while the second person stays near the machine for looking around. (3) At time t_3, there is another person who arrives in the interesting zone, thus the first person (who is breaking the machine) goes away. (4) Finally, at the time t_4, the first person returns to the machine to break it. The contextual objects of the observed environment in these metro test series are mainly an equipment and two zones: a ticket vending machine, the zone around the machine and the zone near the validation machines where the second person stands for looking around.

The experts of the European ADVISOR project ran the ADVISOR video interpretation system (that includes the proposed scenario recognition algorithm) on four recorded video sequences taken in two metro stations of Barcelona and Nuremberg. As in the bank test series, they stored all recognized scenarios and compared them with scenario occurrences in the input video sequences. They found that all scenario occurrences are correctly recognized as shown in Table 7.1. In these cases, the scenario recognition algorithm did not fail to recognize pre-defined scenarios because the vision module was able to track correctly every person.

Table 7.1 shows that the predefined scenarios were correctly recognized in most of the cases due to the results of the vision module (i.e. vision errors). Thus, we also need to test the algorithm on simulated data as proposed in section 7.3 to have reliable input data. Moreover, we have not detected any false alarm during all the experiments. The non-detection of false alarms can be explained by the fact that the scenarios are very constrained and they are unlikely to be recognized by error.

7.5 Robustness

This section shows the experiments that we have realized to verify whether the proposed scenario recognition algorithm can work reliably and robustly in a long time period and in a continuous mode. We connected the automatic video interpretation system to one or two cameras to realize different tests on live videos in a bank agency, in an office and in a metro station. These tests were realized in collaboration with two projects: the French CASSIOPEE project for bank monitoring application and the European ADVISOR project for metro surveillance application. Thus, end-users realized themselves these tests and analyzed themselves the obtained results.

To test the scenario recognition algorithm on live bank video sequences, they installed the CASSIOPEE automatic video interpretation system in the bank agency where they took video sequences for the previously presented tests [section 7.4]. They used the scenario models of the previous tests and also added 10 scenario models corresponding to different real situations of the bank agency. Then, they connected the system to two synchronized video cameras to acquire different video sequences in the bank agency. They ran the system several times. Each execution was during a time interval of about 4 hours. They stored all recognized scenarios and compared them to real situations of the bank agency. As in the first test series [section 7.4], the pre-defined scenarios were recognized in most of the cases (95%) and no false alarm was detected.

To continue testing with bank scenarios, we also modeled our office (at INRIA) in the same way as the bank agency. We installed and connected our automatic video interpretation system to two live video cameras and have used the same scenario models as the ones used in the real bank agency. In these test series, we have obtained the same results as the ones obtained in the real bank agency. Today, we still use this test method in our projects at INRIA for evaluating different algorithms of the video interpretation system.

The experts of the European ADVISOR project used the same method as the one used in the bank agency to test the scenario recognition in a metro station in Barcelona. They installed the system and connected it to one live video camera. They also used the scenario models defined in the previous tests [section 7.4] and added several variations of scenario models. They ran the system several times, the longest time lasted one week. After analyzing the stored results, they also found that the scenarios were recognized in most of the cases (more than 90%).

With these different live test series, we can conclude that the recognition algorithm can work reliably and robustly in continuous mode.

7.6 Real-Time and Limitations

This section describes the experiments to verify whether the proposed scenario recognition algorithm can recognize in real-time (video cadence) pre-defined scenarios and to estimate its limitations. We realized several tests on a PC linux: CPU 700MHz, 320MB RAM as following.

We first study how the processing time of the recognition algorithm depends on the complexity of pre-defined scenario models. The complexity of a scenario model depends on the complexity of (i) temporal constraints that can be expressed by the number of its sub-scenario variables and (ii) physical-object variables.

To study how the processing time of the algorithm depends on the resolution of temporal constraints, we tested the algorithm on 30 scenario models of eight different configurations: the scenarios were composed of 3 to 10 sub-scenarios. We notice that there are two separated algorithms in the scenario recognition module. (1) The first algorithm is the scenario compilation algorithm that is ran at the starting of the system (in an off-line mode) and takes 30 scenario models as input. (2) The second algorithm is the scenario recognition algorithm that is ran in online mode and takes the scenario models generated by the first algorithm as input. In this test, the first algorithm generates 130 scenario models to input the second algorithm. We ran the automatic video interpretation system on the bank videos containing about 300 frames, we found that the processing time of the classical Store Totally Recognized

Scenario algorithm is quickly exponential with the number of sub-scenarios (because it has to attempt all combinations of scenario instances), whereas the processing time of the proposed algorithm is close to a linear function of the number of sub-scenarios [Figure 7.4].

To study the processing time of the algorithm focusing on the number of physical-object variables of the scenario models, we tested the algorithm on 30 scenario models of nine configurations: the scenario models are defined with 2 to 10 physical-object variables. We simulated bank videos containing 35 persons. On these videos, we found that the processing time of the state of the art STRS algorithm is quickly exponential in function of the number of physical-object variables, whereas the processing time of the proposed algorithm is close to a linear function of the number of physical-object variables [Figure 7.5].

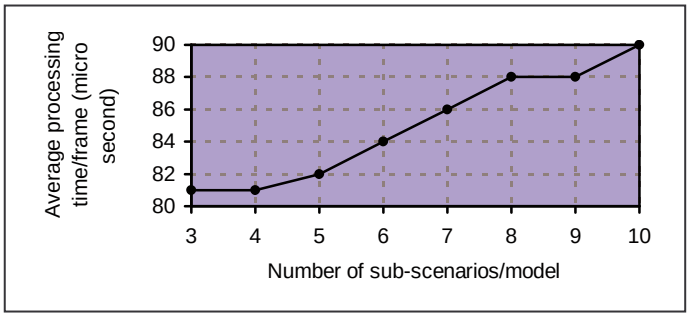


Figure 7.4. The processing time of the proposed scenario recognition algorithm is close to a linear function of the number of sub-scenarios. For 30 scenario models composed of 3 sub-scenarios, the algorithm takes 81 micro seconds to recognize them. The processing time rises dependently on the number of sub-scenarios defined with in each scenario models. Finally, the recognition algorithm takes 90 micro seconds (video cadence) to recognize 30 scenario models defined with 10 sub-scenarios.

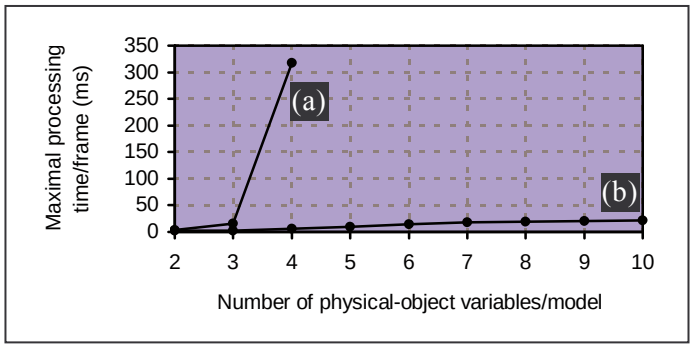


Figure 7.5. The processing time (a) of the state of the art algorithm and (b) of the proposed algorithm depends on the number of physical-object variables of predefined scenario models. The recognition algorithm takes 10 micro seconds to recognize 30 scenario models defined with 2 physical-object variables. The processing time rises up to 30 micro seconds to recognize scenario models defined with 10 physical-objects.

Second, we have studied how the processing time of the recognition algorithm depends on the scene complexity. To have a continuous variation of the scene, we simulated a number of scenes. We built a scene environment with eight zones of interest and ten 3D objects. We simulated the individuals evolving in the scene at each instant. In these simulated videos, the number of individuals changed from 30 up to 240. To verify whether the proposed algorithm can recognize in real-time (video cadence) the predefined scenarios, we measured the maximal processing time per frame. We found that the maximal processing time for each frame is 100ms for a scene composed of 240 persons. We also found that the average processing time for each frame is close to a linear function of the number of persons. Figure 7.6 shows several tests of this experiment to illustrate how the processing time depends on the complexity of the scene.

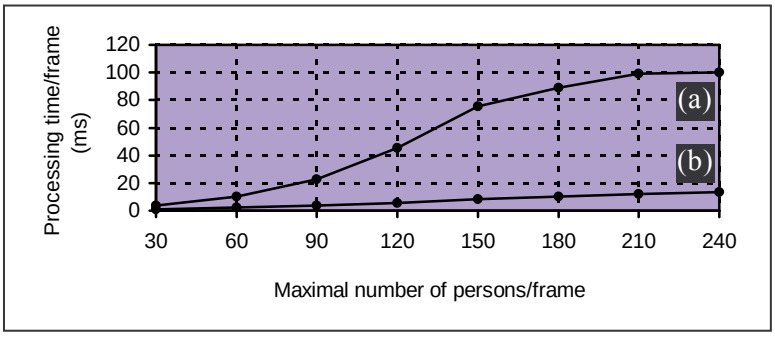


Figure 7.6. The (a) maximal and (b) average processing time/frame of the new algorithm depend on the number of detected persons. The recognition algorithm takes 5 micro seconds to recognize 30 scenario models (defined with different number of physical-object variables and different number of sub-scenarios) in video sequences containing 30 persons. The maximal processing time rises up to 100 micro seconds to recognize the same 30 scenario models in video sequences containing 240 persons.

Thanks to the last experiment, we can conclude that the proposed scenario recognition algorithm can recognized in real-time (video cadence) the predefined scenarios if the number of persons/frame is less than 240.

7.7 Conclusion

To test and evaluate the proposed scenario recognition algorithm, we have first integrated the algorithm with a vision module to obtain an operational Automatic Video Interpretation System. Then, we have also proposed a test framework for Automatic Video Interpretation systems (especially for testing the recognition algorithm). Finally, we have conducted three types of experiments (1) to verify whether the proposed scenario recognition recognizes correctly scenario models pre-defined by experts of different application domains, (2) to verify whether the proposed algorithm can work robustly in a long time period and (3) to verify whether the proposed algorithm can recognize in real-time (video cadence) pre-defined scenario models and also to estimate the limitations of the proposed recognition algorithm.

To evaluate the proposed recognition algorithm, we have proposed a test framework that is composed of methods for evaluating output data at different configurations and a test system generating simulated data. The integrated test system of this

framework was able to generate simulated videos and individuals from scenario models pre-defined by experts of different application domains. After developing such a framework, we have conducted three following experiments:

We tested the proposed algorithm by series of tests on 15 video sequences (11 sequences taken in a bank agency and 4 sequences taken in two metro stations) as presented in section 7.4. The obtained results are promising. The predefined scenario models are most of the time (95%) correctly recognized. The algorithm did not correctly recognize pre-defined scenarios in two (of fourteen) video sequences, because, the vision module of our Automatic Video Interpretation system failed to detect several persons evolving in the observed scene. There is no false alarm detected during the tests, because the scenario models are constrained enough.

Continuing to test the recognition algorithm, we have conducted a second test series on live-videos acquired directly in a bank agency, an office and a metro station. As in the first test series, in most of the time, the algorithm recognizes correctly all predefined scenario models. Moreover, our Automatic Video Interpretation system is executed many times for a longtime period (e.g. one week in a metro station in Barcelona).

Finally, thanks to the last experiment, we have tested the algorithm on simulated data to estimate how the recognition algorithm depends on the complexity of scenario models and of scenes. This experiment shows that the algorithm can correctly recognize in real-time (video cadence) complex scenario models (defined with 10 physical object variables and also 10 sub-scenario variables) in complex scenes containing up to 240 persons.

Conclusion

1 Contributions

We have presented in this thesis our research on temporal scenario representation and recognition for the Automatic Video Interpretation. The first objective is to build a method helping experts to represent clearly and intuitively interesting behaviors in a sufficient/flexible way for individuals evolving in scenes depicted by cameras. The second objective is to conceive a method recognizing in real-time (video cadence) the human behaviors modeled by experts of different application domains.

1.1 Contributions in Temporal Scenario Representation

In the first part of our research, we have built (in collaboration with the INRIA research team ORION in the framework of ARDA workshop series) a novel **video event ontology** composed of concepts and terms to represent video events (i.e. human behaviors or temporal scenarios). Then, we have proposed a **generic hierarchical model** of temporal scenarios based on the ontology to model human behaviors described by experts of different application domains. Finally, we have proposed a new **description language** based on the video event ontology and the generic temporal scenario model helping experts of different application domains to build scenario knowledge bases for their automatic video interpretation systems.

Concerning the work done on video event ontology, we have defined different meta-concepts and terms for describing physical objects, defining different types of temporal scenarios, defining different parts of temporal scenarios and describing different types of relations between physical objects and sub-scenarios composing a temporal scenario. In the proposed ontology, a physical object is characterized by its attributes and its ownership to a class. Then, based on the capacity of initiating movements of physical objects, we have distinguished two categories of physical objects: contextual objects and mobile objects. To represent human behaviors, we have defined six basic meta-concepts. Then, we have also defined meta-concepts as a combination of these basic meta-concepts to represent more complex human behaviors involving one or several individuals. A human behavior is defined using the proposed meta-concepts with (usually) several constraints on the physical objects and sub-behaviors composing the given behavior. To represent relations between concepts, we do not only use non-temporal constraints but also both numerical and symbolical temporal constraints including Allen's interval algebra operators.

Continuing to study the recognition of temporal scenarios, we have proposed a generic hierarchical model of temporal scenarios (that is based on the video event ontology) to represent human behaviors. A temporal scenario model represented by this generic model is composed of five parts: a set of physical object variables corresponding to physical objects involved in the modeled scenario, a set of temporal variables corresponding to the sub-scenarios composing the given scenario, a set of

forbidden variables corresponding to scenarios that are not allowed to be recognized during the recognition of the given scenario, a set of constraints on these three variable sets and finally a set of tasks to be executed if the given scenario model is recognized. Although the fifth part is not used to recognize the modeled scenario, it is important to trigger the post-processing of the recognition of a scenario.

Finally, based on the video event ontology and the generic model of temporal scenarios, we have built a new description language helping experts from different application domains to model easily interesting human behaviors. The proposed language is composed of a syntax and a vocabulary that construct two main parts of the proposed language: standard and advanced features. The standard features consist in representing all temporal scenario models using the generic model and pre-defined time representation. The standard features are sufficient, but the advanced features are convenient to adapt the language to specific applications. The advanced features allow experts to redefine/define existing/new object classes and operators (both temporal/non-temporal operators). Specially, experts can define new time operators to implement a new time ontology to represent specific behaviors.

To validate our method to represent temporal scenarios, we have integrated the proposed video event ontology, the generic temporal scenario model and the description language in the Automatic Video Interpretation platform and we have also applied it to different applications. The proposed description language is currently used in, and evaluated through the European projects ADVISOR (for metro station surveillance), AVITRACK (for apron monitoring), the French project CASSIOPEE (for bank monitoring) and the European project SAMSIT (for train surveillance). The utilization of the proposed scenario description language by experts of these different application domains has shown its capability to represent easily temporal scenarios corresponding to the human behaviors of interest for these applications.

1.2 Contributions in Temporal Scenario Recognition

The second part of our research focuses on the recognition of temporal scenario models. To solve this issue, we have proposed a novel method recognizing in real-time (video cadence) temporal scenario models pre-defined by experts. The recognition method has been proposed and enhanced by different steps. We have first proposed a **new schema for temporal scenario recognition** for Automatic Video Interpretation. At each instant, the recognition process first attempts to recognize **all elementary scenario models** then **several composed scenario models**. The recognition of a composed scenario is triggered if a scenario instance corresponding to its last temporal variable is recognized at the same instant. Each time, a scenario is recognized, the recognition process stores it in a **Forest of Scenario Instances** to be used to recognize other composed scenario models. This storing method can **speed up** the process accessing already recognized scenario instances.

Secondly, we have studied a **necessary and sufficient condition** for a scenario model to be recognized. We have demonstrated that a scenario model **can be recognized** if and only if it is **coherent**.

Thirdly, to **speed up** the recognition process of an **elementary scenario model**, we have proposed to reorganize the knowledge represented within an elementary scenario model M_e by distributing its constraints to its physical object variables such that a constraint f can be evaluated when and only when the corresponding physical object variable is assigned a value. Thus, the elementary scenario recognition algorithm eliminates immediately all combinations of physical objects that cannot be a

solution of the given elementary scenario model M_e. Thus, it prevents the recognition process from a combinatorial explosion.

Fourthly, continuing to **reduce the processing time** of the recognition algorithm, we have also proposed an **efficient temporal constraint resolution technique** to recognize in real-time (video cadence) **complex** temporal scenario models. The proposed algorithm is also efficient for processing temporal constraints as well as for combining several actors defined within a given scenario M.

To validate the proposed recognition method, we have estimated its complexity as shown in chapters 5 and 6. The estimation of the algorithm complexity shows that the proposed recognition algorithm is generally **linear** with the **number of physical** object variables and also with the **number of temporal variables** defined within temporal scenario models. This complexity estimation shows an efficient algorithm in terms of processing time compared to exponential algorithms of the state of the art.

We have not only estimated the complexity of the algorithm but also conducted different **experiments** (as shown in chapter 7) to measure the real processing time of the algorithm on different input data and also to validate the proposed recognition algorithm. To realize these experiments, we have first proposed a **test framework** for the temporal scenario recognition module that is able to generate different video sets corresponding to various situations of the observed environment. Then, we have tested the proposed temporal scenario recognition algorithm on different videos (on-line/off-line modes and also on simulated data). The obtained test results show the capacity of the proposed algorithm to recognize in real-time (video cadence) complex scenario models (up to 10 physical object variables and 10 sub-scenario variables per scenario) with complex video sequences (up to 240 persons/frame in the scene).

Finally, the proposed temporal scenario recognition algorithm is currently used in, and evaluated in different applications of the European projects ADVISOR (for metro station surveillance), AVITRACK (for apron monitoring), the French project CASSIOPEE (for bank monitoring) and the European project SAMSIT (for train surveillance). The obtained experimental results are promising. The proposed algorithm can work correctly and robustly in real-time (video cadence).

2 Future Works

The proposed temporal scenario representation method shows the ability to help experts to represent easily interesting scenarios and the proposed scenario recognition algorithm shows the capacity of recognizing efficiently pre-defined scenario models. Nevertheless, our research motivations on temporal scenario representation and recognition are not yet stopped. There will be different subjects to study. For example: handling errors of the vision module, learning normal situations (temporal scenarios) from different videos, cooperation between the scenario recognition module and the vision module to obtain better results, using existing techniques to build an efficient visual user interface helping experts to define their scenarios and conducting more experiments for validating and estimating the limitations of the recognition algorithm. Such a visual user interface is actually studied in the research team ORION at the INRIA Sophia-Antipolis research unit, France. Moreover, we continue in collaboration with experts of different application domains to test and analyze the

test results to evaluate more precisely the recognition algorithm. We address in the following the details of three promising research works.

2.1 Uncertainty

Chapter 4 has presented the proposed algorithm for the recognition of temporal scenarios. The presented algorithm is efficient in terms of processing time (as shown by its complexity estimation and the realized experiments) and based on two assumptions on the reliability of the input data. However, real world is a complex and often uncertain environment. Thus, the vision results are often not good enough for the recognition algorithm to recognize correctly pre-defined scenario models. For example: in the first experiment [chapter 7] that the end-users realized at a bank agency, the algorithm could not recognize every scenario (80%) in several video sequences, because there were errors of the vision module. To cope with this problem, we propose to study the existing techniques to handle uncertainty (e.g. using HMMs, Neural Networks,…) and to combine them with the proposed algorithm to propose a new temporal scenario recognition algorithm that is not only efficient in term of processing time but also capable to cope with the vision uncertainty.

2.2 Learning Temporal Scenarios

We are not only interested in recognizing abnormal behaviors occurring in the observed scene (e.g. Vandalism) as in video surveillance applications (e.g. metro station surveillance), but also normal behaviors (e.g. correct gas fill-up for an aircraft) as in video monitoring applications (e.g. apron monitoring). Both normal and abnormal behaviors (e.g. Bank attack) can be modeled by experts of application domains using the proposed temporal scenario representation method through the proposed description language or a visual user interface. Nevertheless, it will be interesting to learn normal behaviors of every day data, because normal behaviors are common and can be extracted form every day activities.

To study this, we propose to conduct a research work that consists in discovering the video events registered every day at different sites by different automatic video interpretation systems to build different temporal scenario models corresponding to the situations to be learned.

2.3 Cooperation between the Scenario Recognition Module and the Vision Module

In our current video interpretation framework, the scenario recognition algorithm takes passively as input the results of the vision algorithms. However, the vision results are often not sufficient. Thus, it will be interesting for the scenario recognition module to answer the question “are the vision results sufficient?”. In certain cases, the recognition algorithm can “guess” which persons are present in the observed scene using the probability of the recognition of temporal scenarios. In these cases, it will be useful for the scenario recognition algorithm to send a feedback to the vision module to perform certain vision tasks to track again the people in a time interval. Thus, the third future research work consists in studying how to establish a cooperation between the scenario recognition module and the vision module.

References

[Albrecht *et al.*] D. W. Albrecht, I. Zukerman & A. E. Nicholson. **Bayesian Models for keyhole plan recognition in an Adventure Game**. User Modeling and User-Adapted Interaction, vol. 8, PP 5-47, 1998.

[Allen, 1981] J. F. Allen. **An Interval-Based Representation of Temporal Knowledge**. IJCAI, 1981.

[Allen, 1984] J. F. Allen. **Towards a general theory of action and time**. Artificial Intelligence, 23:123-154, 1984.

[Benzmüller, 2001] C. Benzmüller. **An Agent Based Approach to Reasoning**. Extended abstract for invited plenary talk at AISB'01 Convention Agents and Cognition, University of York, England, 2001/03.

[Bezault *et al.*, 1992] L. Bezault, R. Boulic, N. Magnenat-Thalmann & D. Thalmann. **An Interactive Tool for the Design of Human Free-Walking Trajectories**. Proceedings of Computer Animation '92, pp. 87-10, 1992.

[Bistarelli *et al.*, 1995] S. Bistarelli, U. Montanari & F. Rossi. **Semiring-based Constraint Solving and Optimization**. Over-Constrained Systems (Selected papers from the Workshop on Over-Constrained Systems at CP'95), 1995.

[Bobick, 1997] A. F. Bobick. **Movement, Activity and Action: The Role of Knowledge in the Perception of Motion**. Phil. Trans. Royal Society London B, 352, pp. 1257-1265, 1997.

[Borillo & Gaume, 1990] M. Borillo & B. Gaume. **An Extension of Kowalski and Sergot's Event Calculus**. ECAI, pp. 99-104, 1990.

[Boulic *et al.*, 1990] R. Boulic, N. Magnenat-Thalmann & D. Thalmann. **A global human walking model with real-time kinematic personification**. The Visual Computer, 6, pp.344-358, 1990.

[Brémond, 1997] F. Brémond. **Environnement de résolution de problèmes pour l'interprétation de séquences d'images**. Thèse, INRIA-Université de Nice Sophia Antipolis, 1997/10.

[Bui, 2003] H. H. Bui. **A general model for online probabilistic plan recognition**. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003), Acapulco, Mexico, 2003/08.

[Bui *et al.*, 2002] H. H. Bui, Svetha Venkatesh & Geoff West. **Policy Recognition in the Abstract Hidden Markov Model**. Journal of Artificial Intelligence Research (JAIR), 17: 451-499, 2002.

[Bui *et al.*, 2001] H. H. Bui, Svetha Venkatesh & Geoff West. **Tracking and surveillance in wide-area spatial environments using the Abstract Hidden Markov Model**. International Journal of Pattern Recognition and Artificial Intelligence - IJPRAI (Special issue on Hidden Markov Models in Vision), 15(1): 177-195, 2001.

[Carberry, 1990] S. Carberry. **Incorporating Default Inference in Plan Recognition**. AAAI, pp. 471-478, 1990.

[Castel *et al*., 1996] C. Castel, L. Chaudron & C. Tessier. **1st Order C-Cubes for the Interpretation of Petri Nets: an Application to Dynamic Scene Understanding**. TAI'96 - 8th International conference on tools with artificial intelligence, pp 366-373, Toulouse, France, 1996.

[Charniak & Goldman, 1991] E. Charniak & R. Goldman. **A Probabilistic Model of Plan Recognition**. AAAI, 1991.

[Chleq & Thonnat, 1996] N. Chleq & M. Thonnat. **Realtime image sequence interpretation for video-surveillance applications**. International conference on Image Processing (ICIP'96). Proceeding IEEE ICIP'96. Vol 2. pp 801-804. Lausanne, Switzerland, 1996/09.

[Choi *et al*., 1997] S. Choi, Y. Seo, H. Kim & K. S. Hong. **Where are the ball and players? : Soccer Game Analysis with Color-based Tracking and Image Mosaick**. ICIAP, 1997.

[Collins *et al*., 2000] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto & O. Hasegawa. **A System for Video Surveillance and Monitoring**. Tech. report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, 2000/05.

[Corrall, 1992] D. Corrall. **Deliverable 3: Visual Monitoring and Surveillance of Wide-Area Outdoor Scenes**. Technical Report Esprit Project 2152: VIEWS, 1992/06.

[Cossart-Jaupitre, 1999] C. Cossart-Jaupitre. Un Estimateur Symbolique pour le Suivi de Situation. Thèse, l'Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, France, 1999.

[Cupillard *et al*., 2004] F. Cupillard, A. Avanzi, F. Bremond & M. Thonnat. **Video Understanding for Metro Surveillance**. The IEEE ICNSC 2004 in the special session on Intelligent Transportation Systems, Taiwan, 03/2004.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri and J. Pearl. **Temporal constraint networks**. Artificial Intelligence, 49, pp61-95, Elsevier Science Publishers B.V, 1991.

[Dehais *et al*., 2004] Frédéric Dehais, Charles Lesire, Catherine Tessier, Laurent Chaudron. **Conflits et contre-mesures dans l'activité de pilotage**. RFIA 2004, Toulouse, France, 01/2004.

[Delamarre & Faugeras, 1999] Q. Delamarre & O. Faugeras. **3D Articulated Models and Multi-View Tracking with Silhouettes**. International Conference on Computer Vision, 1999.

[Despouys, 2000] O. Despouys. **Une architecture intégrée pour la planification et le contrôle d'exécution en environnement dynamique**. LAAS-CNRS, 2000/12.

[Donikian *et al.*, 1999] S. Donikian, F. Devillers, G. Moreau. **The kernel of a scenario language for animation and simulation**. Eurographics workshop on animation and simulation, Springer Verlag, Milano, Italia, September 1999.

[Dousson, 2002] C. Dousson. **Extending and unifying chronicle representation with event counters**. ECAI, Lyon, France, 2002/07.

[Dousson, 1994] C. Dousson. **Suivi d'évolutions et reconnaissance de chroniques**. Thèse, Université Paul Sabatier de Toulouse, 1994/09.

[Dousson *et al*., 1993] C. Dousson, Paul Gaborit & Malik Ghallab. **Situation Recognition: Representation and Algorithms**. In proc. of the 13th IJCAI, pp. 166-172, 1993/08.

[Dousson & Ghallab, 1994] C. Dousson & Malik Ghallab. **Suivi et reconnaissance de chroniques**. Revue d'intelligence artificielle, Vol.8, N°1, pp.29-61, 1994.

[Gennari, 1998] R. Gennari. **Temporal Reasoning and Constraint Programming: a Survey**. CWI Quarterly, 11, Amsterdam, The Netherlands, 1998/09.

[Ghallab, 1996] M. Ghallab. **On Chronicles: Representation, On-line Recognition and Learning**. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge (USA), pp.597-606, 1996/11.

[Ghallab & Mounir-Alaoui, 1989] M. Ghallab & A. Mounir-Alaoui. **Managing Efficiently Temporal Relations through Indexed Spanning Trees**. IJCAI, 1989.

[Herzog, 1995] G. Herzog. **From Visual Input to Verbal Output in the Visual Translator**. Universität des Saarlandes, Germany, 1995/07.

[Herzog *et al*., 1989] G. Herzog, C. K. Sung, E. André, W. Enkelmann, H. H. Nagel, T. Rist, W. Wahlster & G. Zimmermann. **Incremental Natural Language Description of Dynamic Imagery**. In: C. Freksa and W. Brauer (eds.), Wissensbasierte Systeme. 3. Int. GI-Kongreß, pp. 153-162. Berlin, Heidelberg: Springer, 1989.

[Hongeng *et al*., 2000] S. Hongeng, F. Brémond & R. Nevatia. **Bayesian Framework for Video Surveillance Application**. Proc. of the 15th International Conference on Pattern Recognition (ICPR00), Barcelona, 2000/09.

[Hongeng *et al*., 2000b] S. Hongeng, F. Brémond & R. Nevatia. **Representation and Optimal Recognition of Human Activities**. In IEEE Proceedings of Computer Vision and Pattern Recognition, South Carolina, USA, 2000.

[Hongeng & Nevatia, 2001] S. Hongeng & R. Nevatia. **Multi-Agent Event Recognition**. International Conference on Computer Vision (ICCV2001), Vancouver, B.C., Canada, 2001/07/12.

[Howarth & Buxton, 2000] R. J. Howarth & H. Buxton. **Conceptual Descriptions from Monitoring and Watching Image Sequences**. Image and Vision Computing, 18, pp105-135, 2000.

[Howell & Buxton, 1998] A. J. Howell & H. Buxton. **Learning Identity with Radial Basis Function Networks**. Neurocomputing, Vol. 20, pp15-34, 1998.

[Howell & Buxton, 2001] A. J. Howell & H. Buxton. Time-delay RBF networks for attentional frames in Visually Mediated Interaction. Neural Processing Letters, 2001.

[Howell & Buxton, 2002] A. J. Howell & H. Buxton. **Active vision techniques for visually mediated interaction**. Image and Vision Computing, 2002.

[Huber *et al.*, 1994] M. J. Huber, E. H. Durfee & M. P. Wellman. **The Automated Mapping of Plans for Plan recognition**. The 10th Conference on Uncertainty in Artificial Intelligence, pp 344-351, 1994.

[Intille & Bobick, 1998] S. Intille & A. Bobick. **Representation and Visual Recognition of Complex, Multi-agent Actions using Belief Networks**. MIT Technical Report No. 454, 1998/01.

[Ivanov *et al*., 1999] Y. Ivanov, C. Stauffer & A. Bobick. **Video Surveillance of Interactions**. In 2nd International Workshop on Visual Surveillance, pp82-89, Fort Collins, Colorado, 1999/06.

[Jonsson & Frank, 2000] A. Jonsson & J. Frank. **A Framework for Dynamic Constraint Reasoning Using Procedural Constraints**. European Artificial Intelligence Conference, ECAI2000, 2000.

[Kautz, 1987] H. A. Kautz. **A Formal Theory of Plan Recognition**. Thesis-TR215, 1987.

[Kautz, 1990] H. A. Kautz. **A circumscriptive theory of plan recognition**. Intentions in Communication, 1990.

[Kautz & Allen, 1986] H. A. Kautz & J. F. Allen. **Generalized Plan Recognition**. Proceeding of the 5th AAAI, pp. 32-37, Philadelphia, Pennsylvania, 1986.

[Khatib *et al*., 2001] L. Khatib, P. Morris, R. A. Morris & F. Rossi. **Temporal Constraint Reasoning With Preferences**. IJCAI 2001, pp322-327, 2001.

[Kowalski & Sadri, 1994] R. Kowalski & F. Sadri. **The situation calculus and event calculus compared**. International Logic Programming Symposium, Ithaca, New York, Bruynooghe M. (Ed), The MIT Press, 1994.

[Kowalski & Sergot, 1986] R. Kowalski & M. Sergot. **A Logic-Based Calculus of Events**. New Generation Computing, vol 4, pp. 67-95, 1986.

[Kumar & Mukerjee, 1987] K. Kumar & A. Mukerjee. **Temporal Event Conceptualization**. The 10th IJCAI, pp. 472-475, 1987.

[Ladkin & Reinefeld] P. Ladkin and A. Reinefeld. A Symbolic Approach to Interval Constraint Problem.

[McCarthy & Hayes, 1969] J. McCarthy & P. J. Hayes. **Some Philosophical Problems from the Standpoint of Artificial Intelligence**. Machine Intelligence, 4:463-502, 1969.

[Missiaen *et al*., 1995] L. Missiaen, M. Bruynooghe & M. Denecker. **CHICA, an Abductive Planning System Based on Event Calculus**. Journal of Logic and Computation 5(5): 579-602, 1995.

[Mohnhaupt & Neumann, 1991] M. Mohnhaupt & B. Neumann. **Understanding object motion: Recognition, learning and spatiotemporal reasoning**. Journal of Robotics and Autonomous Systems, 8:65--91, 1991.

[Mohr & Henderson, 1986] R. Mohr & T. C. Henderson. **Arc and Path Consistency Revisited**. Research Note, Artificial Intelligence, pp225-233, vol28, 1986.

[Mouhoub, 1997] M. Mouhoub. **Contribution au Raisonnement Temporel: Etude des Techniques de Satisfaction de Contraintes Numeriques et Symboliques**. 3rd International Symposium on Systems and Information, pages 151-171, Algiers, 1997.

[Mouhoub, 2001] M. Mouhoub. **Analysis of Approximation Algorithms for Maximal Temporal Constraint Satisfaction Problems**. The 2001 International Conference on Artificial Intelligence (IC-AI'2001), pages 165-171, Las Vegas, 2001.

[Mouhoub *et al*., 1998] M. Mouhoub, F. Charpillet & J. P. Haton. **Experimental Analysis of Numeric and Symbolic Constraint Satisfaction Techniques for Temporal Reasoning**. Constraints: An International Journal 3, 2-3, juin 1998, p. 151-164, 1998/06.

[Nagel, 1988] H. H. Nagel. **From image sequences towards conceptual descriptions**. Image and Vision Computing, 6:59-74, 1988.

[Nagel, 1991] H. H. Nagel. **The Representation of Situations and their Recognition from Image Sequences**. RFIA, pp. 1221-1229, Lyon, France, 1991.

[Nokel, 1989] K. Nokel. **Temporal Matching: Recognizing Dynamic Situations from Discrete Measurements**. The 11th IJCAI, pp. 1255-1260, Detroit, Michigan, USA, 1989.

[Pinhanez & Bobick, 1997] C. Pinhanez & A. Bobick. **Human Action Detection Using PNF Propagation of Temporal Constraints**. M.T.T Media Laboratory Perceptual Section Technical Report No. 423, 1997/04.

[Quiniou *et al*., 2001] R. Quiniou, M. O. Cordier, G. Carrault, F. Wang, C. Rouveirol & M. Sebag. **Application of ILP to cardiac arrhythmia characterization for chronicle recognition**. Proceedings of the 11th International Conference on Inductive Logic Programming, volume 2157 of Lecture Notes in Artificial Intelligence, pages 220--227. Springer-Verlag, 2001/09.

[Renz & Nebel, 2001] J. Renz & B. Nebel. **Efficient Methods for Qualitative Spatial Reasoning**. Journal of Artificial Intelligence Research (JAIR), 15, 289-318, 2001.

[Rives *et al*.,] J. Rives, J. L. Sánchez & R. Pereira. **A Temporal Constraint Satisfaction Problem-Solver**.

[Rota, 2001] N. Rota. **Contribution à la reconnaissance de comportements humains à partir de séquences vidéos**. Thèse, INRIA-Université de Nice Sophia Antipolis, 2001/10.

[Rota & Thonnat, 2000] N. Rota & M. Thonnat. **Activity Recognition from Video Sequences using Declarative Models**. 14th European Conference on Artificial Intelligence (ECAI 2000), Berlin, Proceeding ECAI'00 - W. Horn (ed.) IOS Press, Amsterdam, 2000/08.

[Rota & Thonnat, 2000b] N. Rota & M. Thonnat. **Video Sequence Interpretation for Visual Surveillance**. 3rd IEEE International Workshop on Visual Surveillance, VS'00, pp 59-67, Dublin, Ireland Proceeding IEEE, 2000/07.

[Sadri, 1987] F. Sadri. **Three Recent Approaches to Temporal Reasoning**. Antony Galton, editor, Temopral Logics and their Applications, pp. 121-168, Academic Press, 1987.

[Schmidt *et al*., 1978] C.F. Schmidt, N. S. Sridharan & J. L. Goodson. **The Plan Recognition Problem: an Intersection of Psychology and Artificial Intelligence**. Artificial Intelligence, vol. 11, pp 45-83, 1978.

[Schwalb, 1998] E. Schwalb. **Temporal Reasoning with Constraints**. Technical report, Ph.d. thesis, Information and Computer Science, Universiy of California, Irvine, 1998.

[Sellam & Boulmakoul, 1994] S. Sellam & A. Boulmakoul. **Intelligent Intersection : Artificial Intelligence and Computer Vision Techniques for Automatic Incident Detection**. Artificial Intelligence Application to Traffic Engineering, 1994.

[Shanahan, 1990] M. P. Shanahan. **Representing Continuous Change in the Event Calculus**. ECAI, pp. 598-603, 1990.

[Shanahan, 2000] M. P. Shanahan. **An Abductive Event Calculus Planner**. Journal of Logic Programming, vol. 44, pp. 207-239, 2000.

[Sripada, 1991] S. M. Sripada. **Temporal Reasoning in Deductive Databases**. PhD thesis, Imperial College, London, England, 1991.

[Tessier, 1997] C. Tessier. **Reconnaissance de scènes dynamiques à partir de données issues de capteurs: le projet PERCEPTION**. Rapport technique, Onera-Cert, 2 avenue Edouard-Belin, BP4025 31055 Toulouse Cedex France, 1997/08/20.

[Tessier, 2003] C. Tessier. **Towards a commonsense estimator for activity tracking**. AAAI Spring symposium, Stanford University, Palo Alto, California USA, 03/2003.

[Terzopoulos, 1999] D. Terzopoulos. **Artificial life for computer graphics**. Communications of the ACM, 42(8), pp32-42, August, 1999.

[Thonnat & Rota, 1999] M. Thonnat & N. Rota. **Image understanding for visual surveillance application**. Third international workshop on cooperative distributed vision CDV-WS'99, pp51-82, Kyoto, Japan, 1999/11.

[Vila, 1994] L. Vila. **A Survey on Temporal Reasoning in Artificial Intelligence**. AI Communications 7 (1), 4-28, 1994.

[Vu *et al*., 2003] V. T. Vu, François Brémond & Monique Thonnat. **Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition**. IJCAI 2003, Acapulco, Mexico, 2003/08.

[Vu *et al*., 2003b] V. T. Vu, François Brémond & Monique Thonnat. **Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Precompiled Scenario Models**. ICVS2003, 2003.

[Vu *et al*., 2002] V. T. Vu, François Brémond & Monique Thonnat. **Human behaviour visualisation and simulation for automatic video understanding**. WSCG 2002, Czech Republic, 2002/02.

[Vu *et al*., 2002b] V. T. Vu, François Brémond & Monique Thonnat. **Temporal Constraints for Video Interpretation**. ECAI 2002, Lyon, France, 2002/07.

[Vu *et al*., 2002c] V. T. Vu, François Brémond & Monique Thonnat. **Video surveillance: human behaviour representation and on-line recognition**. KES 2002, Italy, 2002.

Publications

Published Papers

2003 [1] Van-Thinh VU, François BRÉMOND and Monique THONNAT. **Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition**. In Proceeding of *the Eighteenth International Joint Conference on Artificial Intelligence (**IJCAI'03**)*, pp. 1295-1300, Acapulco, Mexico, August 2003 (lecture presentation).

This paper presents a new scenario recognition algorithm for Video Interpretation. We represent a scenario model by specifying the characters involved in the scenario, the sub-scenarios composing the scenario and the constraints combining the sub-scenarios. Various types of constraints can be used including spatio-temporal and logical constraints. In this paper, we focus on the performance of the recognition algorithm. Our goal is to propose an efficient algorithm for processing temporal constraints and combining several actors defined within the scenario. By efficient we mean that the recognition process is linear in function of the number of sub-scenarios and in most of the cases in function of the number of characters. To validate this algorithm in term of correctness, robustness and processing time in function of scenario and scene properties (e.g. number of persons in the scene), we have tested the algorithm on several videos of a bank branch and of an office, in on-line and off-line mode and on simulated data. We conclude by comparing our algorithm with the state of the art and showing how the definition of scenario models can influence the results of the real-time scenario recognition.

[2] Van-Thinh VU, François BRÉMOND and Monique THONNAT. **Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Pre-compiled Scenario Models**. In Proceeding of *the 3rd International Conference on Vision System (**ICVS'03**)*, pp. 523-533, Graz, Austria, April 2003 (lecture presentation).

This paper presents a new scenario recognition algorithm for Video Interpretation. We represent a scenario model with the characters involved in the scenario, with its sub-scenarios and with the constraints combining the sub-scenarios. By pre-compiling the scenario models, the recognition algorithm processes temporal constraints by decomposing complex scenarios into intermediate sub-scenarios to reduce the algorithm complexity. We have tested the recognition algorithm on several videos of a bank agency to try to recognise a scenario of "Attack". We conclude by showing experimental results of the efficiency of this algorithm for real time temporal scenario recognition.

2002 [3] Van-Thinh VU, François BRÉMOND and Monique THONNAT. ***Temporal Constraints for Video Interpretation***. In Proceeding of *the 15-th European Conference on Artificial Intelligence (**ECAI'2002**), W9: Modelling and Solving Problems with Constraints*, pp. 99-114, Lyon, France, July 2002.

This paper presents an original approach for temporal scenario recognition for video interpretation. We propose a declarative model to represent scenarios and we use a logic-based approach to recognise pre-defined scenario models. To speed up the recognition process, we propose a new method to process the temporal operators of interval algebra and a method to extend the time interval of recognised scenarios. We have tested our representation formalism and the inference engine on two real video sequences.

[4] Van-Thinh VU, François BRÉMOND and Monique THONNAT. ***Video surveillance: human behaviour representation and on-line recognition***. In Proceedings of *the Sixth International Conference on Knowledge-Based Intelligent Information & Engineering Systems (**KES'2002**)*, pp. 807-811, Podere d'Ombriano, Crema, Italy, September 2002 (lecture presentation).

This paper presents a recent work in human behaviour representation and on-line recognition for video interpretation. We propose a declarative model to represent human behaviours and we use the logic-based approach to recognise pre-defined behaviour models. We demonstrate our representation formalism and the inference engine on two video sequences. We also propose a limit for the number of behaviour actors based on experimental results. The processing time is still a challenge with complex behaviour models and cluttered scenes.

[5] Van-Thinh VU, François BRÉMOND and Monique THONNAT. ***Human behaviour visualisation and simulation for automatic video understanding.*** *In Proceeding of the 10-th International Conference in Central Europe on Computer Graphics, Visualisation and Computer Vision'2002 (**WSCG'2002**)*, pp485-492 (ISSN1213-6972, Vol.10, No.2, 2002), Czech Republic, February 4 - 8, 2002 (lecture presentation).

The objective of this work is the visualisation and simulation for automatic video interpretation. We have conceived a test framework that generates 3D animations corresponding to behaviours recognised by an automatic interpretation system or corresponding to behaviours described by an expert. Conceiving this test framework is essential in order to be able to develop and validate the interpretation process. The objective of our test framework is (1) to visualise the computation of the interpretation, (2) to be flexible (configurable) enough for testing the different configurations of the interpretation and (3) to be realist enough to understand what is interpreted. To solve this problem we have defined six types of model to represent all the information that is necessary for the interpretation. First, we propose a model of the scene context (containing the 3D geometry) and a model for the virtual camera. Second, we propose an articulated and hierarchical model for representing the human body given its sub parts. We propose two other hierarchical models for modelling human actions and scenarios, and also a model of scene-scenarios that gathers all previous models. We have defined a description language

for representing these models. The obtained results are promising: we have developed a test system for a given interpretation system and started evaluating it by generating test animations.

2001 [6] Van-Thinh VU. ***Rapport du stage de fin d'études du DEPA:Visualisation de comportements humains pour l'interprétation automatique de séquences vidéos***. INRIA Sophia-Antipolis, France, August 2001.

The objective of this work is for studying the problem of the simulation for automatic video interpretation. We have conceived a visualisation system that generates the 3D animations from the recognition of behaviours (by an automatic interpretation system) or from the description of behaviours (by an expert). In our work, we used the software for automatic video interpretation VSIS (Video Surveillance Intelligent System). The visualisation of behaviours has to permit to test and to validate the interpretation process. The objective of our visualisation system is (1) to visualise the computation of the interpretation, (2) to be flexible enough and parameterable for testing the different configurations of the interpretation system et (3) to be realist enough to understand what is going on in the scene. We solve this problem by proposing an articulated and hierarchical model for the generic model of the human body. We also propose two other hierarchical models for generic model of behaviours and of scenarios, and a generic model of scenes that gathers all previous models. We also propose a description language for representing these models. The obtained results are promising: we could visualise the output of VSIS, visualise the described scenarios (by an expert) and verify that the generated animations are coherent with VSIS.

1997 [7] Khang BACH HUNG., Mai LUONG CHI, Van-Thinh VU, et al.. ***MapScan for Windows – Software Package for Automatic Map Data Entry.*** *1997 Asia-Pacific Symposium on Information and Telecommunication Technologies (**APSITT'97**)*, Hanoi, Vietnam, 13-14 March, 1997 (lecture presentation).

1996 [8] Van-Thinh VU. ***A modification of unsupervised classification algorithms***. *International Conference on Information Technology on the 25th Anniversary of the Institute of Information Technology of Vietnam*, Hanoi, Vietnam, 25-27/12/1996 (lecture presentation).

[9] Khang BACH HUNG., Mai LUONG CHI, Van-Thinh VU, et al.. ***An examination of techniques for raster-to-vector process and its implementation – MAPSCAN software package.*** *International Symposium "Advanced Manufacturing Processes, Systems and Technologies"*, Bradford, UK, 26-27/3/1996 (lecture presentation).

Master Thesis Supervision

04/2003-09/2003 **Supervision of** a Master student (Mrs. Thi-Thanh-Tu Bui) coming from the ***I**nstitut de la **F**rancophonie pour l'**I**nformatique*, *INRIA Sophia-Antipolis*, France.

Theme: ***Optimisation of Temporal Scenario Knowledge Bases for Automatic Video Interpretation***.

The objective of this work is to propose optimization methods for the library of pre-defined scenario models in order to recognize real-time scenarios. In this work, we have to (1) realize a temporal scheduling method for the sub scenarios of a composed scenario in the phase of scenario compilation; (2) propose and develop an optimization method for the library of scenario models aim at verify its consistency and eliminate the information redundancy.

In this work, we used the software VSIS (Video Surveillance Intelligent System) for automatic video interpretation and a base of scenarios predefined by the human experts in the security domains such as metro, bank or of scenarios automatically generated by the compiler.

To order the sub scenarios in time, we propose to use a graph-based approach and the interval calculus. Each scenario model is represented by a graph in which we can check the graph consistency. Then, the compiler decomposes the ordered sub scenarios into the simple scenarios which have at most two sub scenarios. This permits an efficient recognition of the composed scenario in a scene.

To optimize the library of scenario models, we propose a method based on « pattern matching and substitution» applying on the compiled scenario base. The obtained results for a base of scenario models for bank are promising, the method permits to eliminate efficiently the scenario redundancy: more than 6% for a manual optimized scenario base and more than 10% for the non-optimized scenario base.

Annex 1. Utilization of Video Event Ontology

ORION Research Group, INRIA, France

This annex presents two video event ontologies for Visual Bank Monitoring and Visual Metro Monitoring. These ontologies are built in the framework of ARDA workshop series on Video Events using the description language proposed in chapter 3.

I. Ontology for Visual Bank Monitoring

In normal font: already implemented and used
In Italic: useful but not yet implemented
a: B means a belongs to type B (i.e. p: Person means p is a Person)
// *Text* means that Text is a comment

1. Physical Objects in a Bank Agency

Mobile Objects
Person (**p**) with different roles (Robber, Employee, Customer, *Agency director, Maintenance employee*, *Security employee*, *Cleaning employee*, *Kid*).
Group of persons (**g**).
Portable Objects (**o**) with different sub-classes(*Suitcase, Stroller, Gun*).

Contextual Objects
Zone (**z**) with different roles (Entrance, *Exit*, Back_Counter, Inront_Counter, Safe, Safe_Entrance, *Agency*)
Equipment (**eq**) with different sub-classes (Counter, *Chair*, *Desk*, *ATM*, Gate, *Poster*, *Closet*)

2. States

States Involving one Mobile Object (p: Person or g: Group) and one Contextual Object (eq: Equipment or z: Zone)

primitive-state(Inside_zone,
  physical-objects(
    (p : Person), (z : Zone))
  constraints(
    (p *in* z)))

3. Simple Events

primitive-event(Changes_zone,
physical-objects(
(p : Person), (z1 : Zone), (z2 : Zone))
components(
(c1 : primitive-state Inside_zone(p, z1))
(c2 : primitive-state Inside_zone(p, z2)))
constraints(
//Sequence
(c1; c2)))

state (gate is opened)

4. Composite Events

4.1. Single-Thread Composite Events involving one Mobile Object (p: Person or g: Group) and possibly one Contextual Object (eq: Equipment)

composite-event(Safe_attack_1person_back_counter,
physical-objects(
(p : Person), (z1: Back_Counter**)**, (z2: Safe))
components(
(c1: primitive-event Changes_zone (p, z1, z2)))

composite-event(Safe_attack_1person_back_counter_and_door_opened,
physical-objects(
(p : Person), (z1: Back_Counter), (z2: Safe)**,** (g: Gate))
components(
(c1: composite-event Safe_attack_1person_back_counter(p, z1,z2)))
constraints(
(g is open)))

composite-event(Safe_attack_1person_infront_counter,
physical-objects(
(p : Person), (z1: Infront_Counter), (z2: Safe))
components(
(c1: primitive-state Inside_zone(p, z1))
(c2: primitive-state Inside_zone(p, z2)))
constraints(
(c2 *before* c1)))

composite-event(Safe_attack_1person_infront_counter_and_door_opened,
physical-objects(
(p : Person), (z1: Infront_Counter), (z2: Safe), (g: Gate))
components(
(c1: composite-event Safe_attack_1person_infront_counter(p, z1, z2)))
constraints(

(g is open)))

customer_at_ATM(p, eq: ATM)
customer_waiting(p, eq: Counter)
customer_toward_counter_and_goes_away(p, eq: Counter) =
moves_close_to(p, eq: Counter), stays_at(p, eq: Counter), goes_away_from(p, eq: Counter)
customers_queuing_at_counter(g, eq: Counter)

4.2. Multi-Thread Composite Events involving several Mobile Objects

// Scenarios with two persons

composite-event(Safe_attack_2persons_inside_safe_entrance,
physical-objects(
(p1 : Person), (p2 : Person), (z1: Safe_Entrance))
components(
(c1: primitive-state Inside_zone(p1, z1))
(c2: primitive-state Inside_zone(p2, z1)))
constraints(
(c2 *during* c1)))

composite-event(Safe_attack_2persons_inside_safe_entrance_and_door_opened,
physical-objects(
(p1 : Person), (p2 : Person), (z1: Safe_Entrance))
components(
(c1: composite-event Safe_attack_2persons_inside_safe_entrance(p1, p2, z1)))
constraints(
(g is opened)))

composite-event(Safe_attack_2persons_infront_counter,
physical-objects(
(employee : Person), (robber : Person), (z1: Infront_Counter), (z2: Safe)))
components(
(c1: composite-event Safe_attack_1person_infront_counter(employee, z1, z2))
(c2: composite-event Safe_attack_1person_infront_counter(robber, z1, z2)))
constraints(
(c2 *during* c1)))

composite-event(Safe_attack_2persons_infront_counter_and_door_opened,
physical-objects(
(employee : Person), (robber : Person),
(z1: Infront_Counter), (z2: Safe), (g : Gate))
components(
(c1: composite-event Safe_attack_2persons_infront_counter(employee,
robber, z1))
constraints(
(g is opened)))

composite-event(Safe_attack_2persons_back_counter,
physical-objects(
(employee : Person), (robber : Person), (z1: Back_Counter), (z2: Safe))
components(
(c1: composite-event Safe_attack_1person_back_counter(employee, z1, z2))
(c2: composite-event Safe_attack_1person_back_counter(robber, z1, z2)))
constraints(
(c2 *during* c1)))

composite-event(Safe_attack_2persons_back_counter_and_door_opened,
physical-objects (
(employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Safe), (g : Gate))
components(
(c1: composite-event Safe_attack_2persons_back_counter(employee,
robber, z1)))
constraints(
(g is opened)))

composite-event(Safe_attack_2persons_back/infront_counter_1,
physical-objects(
(employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe))
components(
(c1: composite-event Safe_attack_1person_infront_counter(employee, z2, z3))
(c2: composite-event Safe_attack_1person_back_counter(robber, z1,z3)))
constraints(
(c2 *during* c1)))

composite-event(Safe_attack_2persons_back/infront_counter_and_door_opened_1,
physical-objects(
(employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe), (g : Gate))
components(
(c1: composite-event Safe_attack_2persons_back/infront_counter(employee,
robber, z1,z2)))
constraints(
(g is opened)))

composite-event(Safe_attack_2persons_back/infront_counter_2,
physical-objects(
((employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe)))
components(
(c1: composite-event Safe_attack_1person_infront_counter(robber, z2, z3))
(c2: composite-event Safe_attack_1person_back_counter(employee, z1, z3)))
constraints(
(c2 *during* c1)))

composite-event(Safe_attack_2persons_back/infront_counter_and_door_opened_2,
physical-objects(
(employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe), (g : Gate)))
components(
(c1: composite-event Safe_attack_2persons_back/infront_counter(employee, robber, z1,z2)))
constraints(
(g is opened)))

composite-event(Safe_attack_2persons,
physical-objects(
(employee : Person), (robber : Person),
(z1: Entrance), (z2: Back_Counter), (z3: Infront_Counter), (z4: Safe)))
components(
(c1 : primitive-state Inside_zone(employee, z2))
(c2 : primitive_sevent Changes_zone(robber, z1,z3))
(c3: composite-event Safe_attack_1person_back_counter(employee, z2, z4))
(c4: composite-event Safe_attack_1person_infront_counter(robber, z3, z4)))
constraints(
(c2 *during* c1)
(c2 *before* c3)
(c1 *before* c3)
(c2 *before* c4)
(c4 *during* c3)))

composite-event(Safe_attack_2persons_and_door_opened,
physical-objects(
(employee : Person), (robber : Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe), (g : Gate)))
components(
(c1: composite-event Safe_attack_2persons(employee, robber, z1, z2, z3, z4)))
constraints(
(g is opened)))

// Scenarios with three persons

composite-event(Safe_attack_3persons_back/infront_counter,
physical-objects(
(employee : Person), (robber : Person), (customer : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe)))
components(
(c1: primitive-state Inside_zone(customer, z2))
(c2: composite-event Safe_attack_1person_infront_counter(robber, z2, z3))
(c3: composite-event Safe_attack_1person_back_counter(employee, z1,z3)))
constraints(
(c1 *before* c2)
(c3 *during* c2)
(duration of employee >=10)
(duration of robber >= 10)
(duration of customer >= 10)))

composite-event(Safe_attack_3persons_back/infront_counter_and_door_opened,
physical-objects(
(employee : Person), (robber : Person), (customer : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe), (g : Gate)))
components(
(c1: composite-event Safe_attack_3persons_back/infront_counter(employee,
robber, customer, z1, z2, z3)))
constraints(
(g is opened)))
composite-event(Safe_attack_3persons_back/infront_counter_entrance,
physical-objects(
(employee : Person), (robber : Person), (customer : Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe)))
components(
(c1: composite-event Safe_attack_3person_back/infront_counter(employee,
robber, customer, z2, z3, z4))
(c2 : primitive-event Changes_zone(customer, z3,z1)))
constraints(
(c2 *during* c1)))

composite-event(
Safe_attack_3persons_back/infront_counter_entrance_and_door_opened,
physical-objects(
(employee : Person), (robber : Person), (customer : Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe), (g : Gate))
components(
(c1: composite-event
Safe_attack_3persons_back/infront_counter_entrance(employee,
robber, customer, z1, z2, z3, z4)))
constraints(
(g is opened)))

composite-event(Safe_attack_3persons_back/infront_counter_safe,
physical-objects(
(employee : Person), (robber : Person), (customer : Person),
(z1: Entrance), (z2: Back_Counter), (z3: Infront_Counter), (z4: Safe))
components(
(c1: composite-event(Safe_attack_3person_back/infront_counter(employee,
robber, customer, z2, z3, z4))
(c2 : primitive-event Changes_zone(customer, z3,z1)))
constraints((c2 *during* c1)))

composite-event(
Safe_attack_3persons_back/infront_counter_safe_and_door_opened,
physical-objects(
(employee : Person), (robber : Person), (customer : Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe), (g : Gate))
components(
(c1: composite-event(
Safe_attack_3persons_back/infront_counter_entrance(employee,
robber, customer, z1,z2, z3,z4)))
constraints((g is opened)))

II. Ontology for Visual Metro Monitoring

In normal font: already implemented and used
In Italic: useful but not yet implemented
a: B means a belongs to type B (i.e. p: Person means p is a Person)
// Text means that Text is a comment

1. Physical Objects in a Metro

Mobile Objects
Person (**p**).
Group of persons (**g**).
Crowd (**c**).
Metro Train (**m**).
*Portable Objects (**o**).*
Other (fire)

Contextual Objects
Zone (**z**) with different roles (Entrance_Zone, Validation_Zone, Exit_Zone, Tracks, Platform, Ticket_Vending_Machine_Zone, Surveillance_Zone, Corridor, Hall).
Equipment (**eq**) with different sub-classes (Ticket_Vending_Machine , Escalator, Wall, Seat, Trashcan, Validation machine, Poster*, Door, Booth, Map*)

2. States

primitive-state(Inside_zone,
  physical-objects (
    (e : Ent), (z : Zone))
  constraints(
    (e ***in*** z)))

primitive-state(Stopped,
  physical-objects(
    (e : Ent))
  constraints(
    (speed of e < minspeed)))

primitive-state(LyingPerson,
  physical-objects(
    (p : Person))
  constraints(
    (Lying(p) is true)))

primitive-state(Groupwidthvariation,
  physical-objects (
    (g : Group))
  constraints(
    (Width(g) > significantwidthvariation)))

primitive-state(Quicksplit,
physical-objects (
(g : Group))
constraints(
(Split(g) > quicksplit)))

primitive-state(Trajectoryvariation,
physical-objects (
(g : Group))
constraints(
(Trajectory (g) > significanttrajectoryvariation)))

primitive-state(Speed_increase,
physical-objects (
(p : Person))
constraints(
(IncreaseSpeed(p) is true)))

primitive-state(Legs_up,
physical-objects (
(p : Person))
constraints(
(LegsUp(p) is true)))

3. Simple Events

primitive-event(Changes_zone,
physical-objects(
(p : Person), (z1 : Zone), (z2 : Zone))
components(
(c1 : primitive-state Inside_zone(p, z1))
(c2 : primitive-state Inside_zone(p, z2)))
constraints(
//Sequence
(c1;c2)))

4. Composite Events and States

4.1. Single-Thread Composite Events involving one Mobile Object (p: Person or g: Group) and possibly one Contextual Object (eq: Equipment or z: Zone)

composite _state(Fighting,
physical-objects(
(g: Group))
components(
(c1: primitive-state LyingPerson(g))
(c2: primitive-state Groupwidthvariation(g))
(c3: primitive-state Quicksplit(g))
(c4: primitive-state Trajectoryvariation(g)))
constraints(
//Alternatives

```
        (c1 or c2 or c3 or c4)  )   )
composite _event(Jumping,
    physical-objects(
        (p: Person) )
    components(
        (c1: primitive-state Speed_increase (p))
        (c2: primitive-state Legs_up (p)) )
    constraints(
        //Sequence
        (c1; c2) )  )

composite-event(Stays_inside_zone,
    physical-objects(
        (e : Ent), (z : Zone) )
    components(
        (c1: primitive-state Inside_zone(e,z)) )
    forbidden-events(
        (c2: primitive-event  Exit(e,z)) )
    constraints(
        (c2 during c1))   )

composite _event(Validating_ticket,
    physical-objects(
        (p: Person), (z1: Entrance_Zone), (z2: Validation_Zone), (z3: Platform) )
    components(
        (c1: primitive-event Changes_zone (p, z1, z2))
        (c2: primitive-event Changes_zone (p, z2, z3)) )
    forbidden-events(
        (c3: composite-event  Jumping (p))
    constraints(
        //Sequence
        (c1; c3; c2) )   )

composite _event(Jumping_over_barrier,
    physical-objects(
        (p: Person), (z1: Entrance_Zone), (z2: Validation_Zone), (z3: Platform) )
    components(
        (c1: primitive-event Changes_zone (p, z1, z2))
        (c2: composite-event Jumping(p))
        (c3: primitive-event Changes_zone (p, z2, z3)) )
    constraints(
        //Sequence
        (c1; c2; c3) )   )

composite-event(Group_staying_in_zone,    // Blocking the access to a Zone Of  Interest
    physical-objects(
        (g: group), (z: Zone) )
    components(
        (c1: primitive-event enter (g, z))
        (c2: primitive-state Stays_inside_zone(g, z)) )
    constraints(
        (c1 before c2)
```

(duration of c2 > 120 seconds))

composite-event(Group_stopped_in_zone,
physical-objects(
(g: Group),(z: Zone))
components(
(c1: primitive-event Enter (g, z))
(c2: primitive-state Stopped(g)))
constraints(
(c1 before c2)
(duration of c2 > 30 seconds)))

composite-event(Blocking_ZOI, // Blocking the access to a Zone Of Interest
physical-objects(
(g: Group), (z: Zone))
components(
(c1: composite-event Group_stopped_in_zone(g, z))
(c2: composite-event Group_staying_in_zone(g,z)))
constraints(
(c1 or c2)))

composite-event(Vandalism_against_ticket_machine_one_man,
physical-objects(
(p: Person), (eq1:Ticket_Vending_Machine),
(z1: Ticket_Vending_Machine_Zone))
components(
(c1 : primitive-event Enters_zone(p, z1))
(c2 : primitive-event Move_close_to(p, eq1))
(c3 : composite-event Stays_at(p, eq1))
(c4 : primitive-event Goes_away_from(p, eq1))
(c5 : primitive-event Move_close_to(p, eq1))
(c6 : composite-event Stays_at(p, eq1)))
constraints(
// sequence
(c1; c2; c3 ; c4 ; c5 ; c6)))

overcrowding(c)
access_to_forbidden_area(p or g, z)
waiting(p or g)
backward_escalator(p, eq: Escalator)
rapid_increase_of_crowding_level(c)
unbalanced_floor_occupation(c)
jumping_on_the_seat(p, eq)
buying_ticket(p, eq)
graffiti(p, eq)

4.2. Multi-Thread Composite Events involving several Mobile Objects

composite-event(Vandalism_against_ticket_machine_two_men,
physical-objects(
(p1 : Person), (p2 : Person), (eq1: Ticket_Vending_Machine),
(z1: Ticket_Vending_Machine_Zone), (z2: Surveillance_Zone))
components(
(c1 : primitive-event Enters_zone(p1, z1))
(c2 : primitive-event Move_close_to(p1, eq1))
(c3 : composite-event Stays_at(p1, eq1)))
(c4 : primitive-event Goes_away_from(p1, eq1))
(c5 : primitive-event Move_close_to(p1, eq1))
(c6 : composite-event Stays_at(p1, eq1))
(c7 : primitive-event Enters_zone(p2, z2)))
forbidden_components(
(c8 : primitive-event Exits_zone(p2, z2)))
constraints(
(c7 *before* c2)
// sequence
(c1; c2; c3 ; c4 ; c5 ; c6)
(c8 before c6)))

attacking(p1 or g, p2) % one person p1 or one group g attacks one single person p2
pickpocketing_one_man(p1, p2) % p2 is victim of one person p1
pickpocketing_several_men(p1,p2, g) % p2 is victim of one person p1 and one group g
following_someone(p1, p2)
sells(p1, p2)
dealing_drug(p1, p2)

Annex 2. Simulation for Automatic Video Interpretation

This annex presents our simulation method for testing the Automatic Video Interpretation systems.

1 Introduction

This section presents a *simulation test framework* for Automatic Video Interpretation [chapter 7]. We have built the test framework composed of two systems: interpretation system and test system. As shown in chapter 1, the Automatic Video Interpretation consists in recognizing pre-defined scenarios describing human behaviors from video sequences. Thus, to test the interpretation system (specially the scenario recognition algorithm), the test framework has been conceived to realize the following tasks [Figure 1]:

(1) **visualize scenarios described by experts**: it is also important for the experts of the application domain (e.g. agent of security in a metro) to visualize the scenarios that they describe. The **test system** takes as input scenarios defined by experts and visualizes them through 3D animations or videos.

(2) **visualize scenarios recognized by an interpretation system**: it is important for the developer (e.g. expert in vision and scenario recognition) to visualize each step of the scenario recognition process. The **interpretation system** takes as input a video sequence and attempts to recognize scenarios evolving in the given video. Then, the **test system** takes as input the recognized scenarios and visualizes them through 3D animations.

(3) **evaluate the couple interpretation-test system**: it is important to verify the coherence between the interpretation system and the test system. The test process starts by (2) and generates videos corresponding to the output 3D animations of (2). Then, it makes a loop by taking as input the generated videos. Inside each loop, it attempts to verify whether the scenarios recognized in this loop are equivalent to the input scenarios.

(4) **validate interpretation systems**: establish the limits and robustness of interpretation systems by simulating test videos. The test system takes as input a scenario model (defined by experts) and visualizes it with different variations (e.g. different light conditions, variations of temporal relations). Then, it generates video sequences corresponding to those variations. After that, the interpretation system attempts to recognize scenarios evolving in generated videos.

Finally, the framework has to verify whether the recognized scenarios are equivalent to the input scenario.

(5) **validate the temporal scenario recognition algorithm**: establish the limits and robustness of the recognition algorithm independently of vision results. The test system takes as input a scenario model defined by experts and generates a flow of mobile objects corresponding to different steps of the given scenario model. Then, the scenario recognition module takes as input the generated flow of mobile objects. It attempts to recognize scenarios corresponding to the behaviors of these mobile objects. Finally, the test framework has to verify whether the recognized scenarios are equivalent to the input scenario.

To describe the proposed test framework, we first focus in the next section on the simulation for testing the processing time of the scenario recognition algorithm.

2 Simulation for Evaluation

For a long time, the problem of 3D scene visualization has been approached. There are several laboratories [Terzopoulos, 1999; Bezault *et al*, 1992; Boulic *et al*, 1990; Donikian *et al*, 1999] who study the visualization of a 3D scene from its description. For example, at the Faculty of Computer Science of Toronto University [Terzopoulos, 1999], researchers generate 3D animations where several fishes and swimmers evolve in the bottom of the sea. To visualize these animations, they have modeled the behaviors of individuals, fishes and also their interactions in groups. In particular, they have modeled all the physical and biological rules for a fish to swim, eat, reproduce and perceive other fishes. At the Computer Graphics Lab of the Swiss Technology Institute of Lausanne [Bezault *et al*, 1992; Boulic *et al*, 1990], researchers have modeled individuals evolving in a museum, in a street and in a supermarket. They have also modeled crowd behaviors like the reaction of people in fire situations.

These laboratories have obtained interesting results in the domain of 3D animations from a scene description. However, there are few laboratories who study the *visualization of scenarios recognized by an automatic video interpretation system*. For example, the Robotvis group at the research unit INRIA Sophia-Antipolis [Delamarre and Faugeras, 1999] visualizes the tracking of the members (legs, arms,…) of an individual who is running. The Robotics Institute at Carnegie Mellon University [Collins *et al*, 2001], computes 3D animations where a group of individuals enters/leaves the university site by taking as input cameras surrounding the university. The goal of these animations is mainly to demonstrate the tracking of the groups all around the university.

To our knowledge, we did not find any system that visualizes the recognition of human behaviors from a video by an automatic interpretation system.

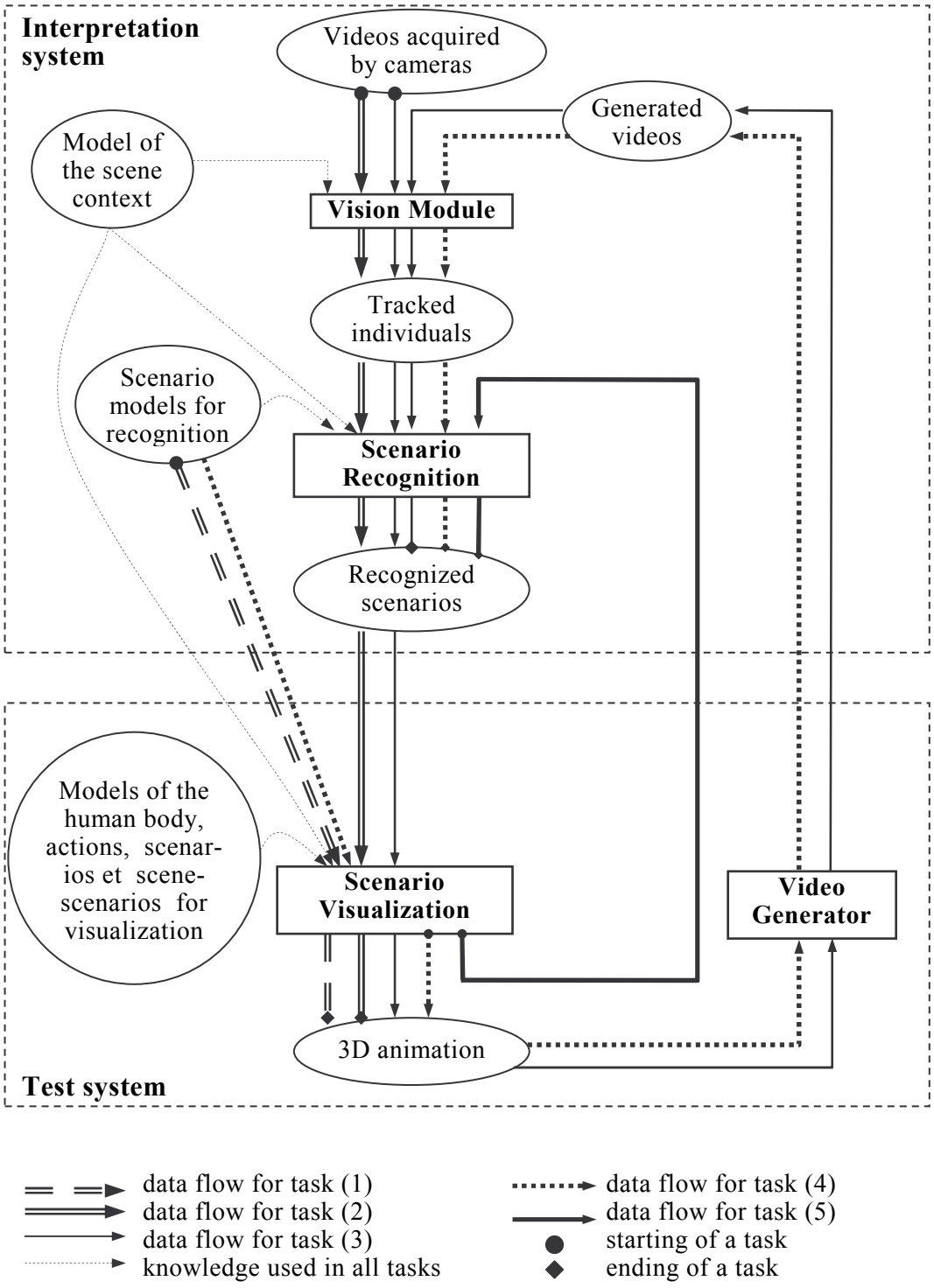


Figure 1. The test framework for Automatic Video Interpretation systems has to realize five tasks [section 1]: (1) visualize scenarios recognized, (2) visualize scenarios described by experts, (3) evaluate the couple interpretation-test system, (4) validate interpretation systems and (5) validate temporal scenario recognition algorithm.

2.1 3D Scene Representation and Visualization

The basic task of the visualization of a 3D scene is the visualization of human behaviors. To cope with this issue, we first propose an approach describing human behaviors that consists in defining four generic models (i.e. meta-class): *human body*, *human action*, *scenario* and *scene-scenario*. Using these generic models, we can construct specific models (e.g. the scenario class "two persons meet at a coffee machine") described in model libraries. Then these specific models are used to generate instances (e.g. scenario "individuals A and B meet at the coffee machine M") to visualize what is occurring in a given real scene (corresponding to videos or scene descriptions). We also propose a description language to represent all these models.

a) Human Body

We use a hierarchical and articulated model as the *generic model* of human body parts and also of the whole human body. A human body part is composed by subparts or geometric primitives. These primitives are the same as those suggested by Delamarre and Faugeras (1999): *spheres*, *truncated cones* and *parallelepipeds*. Figure 5.2 shows the 26 geometric primitives composing a human body.

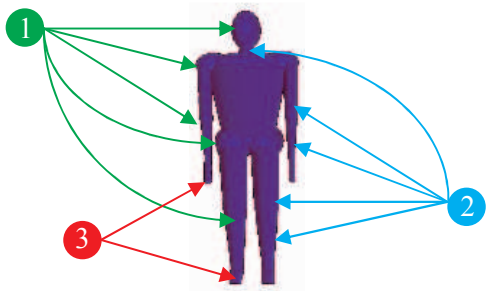


Figure 5.2. Hierarchical and articulated model of the human body using three types of primitives (1) spheres, (2) truncated cones and (3) parallelepipeds.

In the proposed description language we have defined 14 classes for modeling human body parts: the whole human body, the head, the arms, the legs, the neck, the shoulders, the hips, the trunk, the foot and the hand. Figure 5.3 shows the defined human body from different view points.

There are two ways for visualizing the human body. First, we can visualize an individual from its description by an expert. Second, we can visualize an individual detected by an interpretation system from a video sequence. In both cases, we visualize a body part by displaying the geometric primitives composing it through GEOMVIEW (a free software visualizing 3D objects defined by its vertex and its facets [http://www.geomview.org/]).

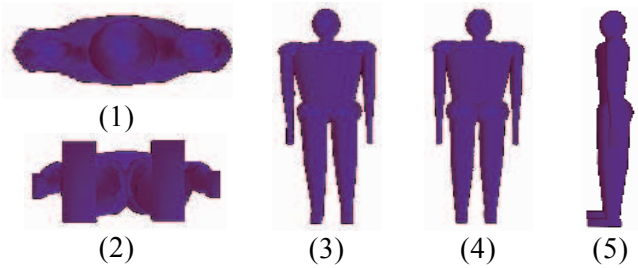


Figure 5.3. Visualisation of a 3D model of the human body: (1) top view, (2) bottom view, (3) front view, (4) back view and (5) view from the left.

b) Human Behavior

In the proposed test framework, the notions of posture, action, scenario and scene-scenario are defined to visualize behaviors recognized by an interpretation system or described by an expert. A posture corresponds to all body 3D geometrical parameters of an individual to be visualized at one instant. An action characterizes an individual motion when one (or several) of its body parameters change(s). Behaviors are represented by scenarios. A scenario combines the individuals of the scene and the context objects with sub-scenarios which are related to the same activity. An elementary scenario is an action. A scene-scenario combines and instantiates all previously defined scenarios.

In our formalism, an action (or scenario) can be visualized at different speeds which indicate how many frames per second are displayed. An action (or scenario) can have a departure/arrival position which locates an individual at the beginning and the end of the action (or scenario). The temporal constraints are expressed by intervals (named *periods*) that correspond to the duration of an action (or scenario). The interval of a sub-action (or sub-scenario) is defined relatively to the period of the containing action (or scenario).

For the purpose of conceiving a test framework for automatic video interpretation systems, we do not consider more precise actions such as “swing the arms” and “move a finger” which are difficult to detect by interpretation systems.

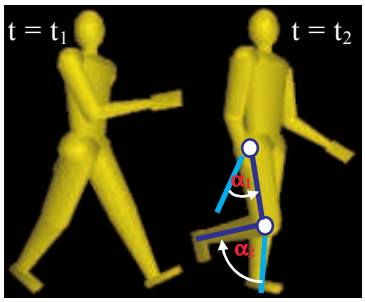


Figure 5.4: In the action “walking” during interval $[t_1, t_2]$, the right leg rotates with angle α_1 around the hip; and in its sub action “the right leg up”, the lower part of the leg rotates with angle α_2 around the knee.

Action

Generic model of actions

An action is relative to the motion of one body part (or the whole human body) which is characterized by changes of the body part parameters. These changes concern mainly rotations around the body part axis. An action is described by a *hierarchical model*: an action can be decomposed into *sub-action(s)* describing the motion of sub-part(s) (see Figure 5.4). In our formalism, to ease the description of actions by experts, it is possible to indicate the departure/arrival position in the case where the body part is the whole individual. There are two types of actions: periodic (e.g. "walking") and non-periodic (e.g. "move close to"). For non-periodic actions, the period corresponds to the duration of the action. For periodic actions, the number of periods is defined in the containing action and the duration is obtained by multiplying the number of periods by the action period.

To represent human actions, we have proposed a generic hierarchical model of actions composed of the following attributes:

- the *concerned part* of human body: the human body part that moves during the action. For example, the concerned part of the action "right leg moves up" is the whole right leg of the person.
- the *fixed part of human body on the ground*: the human body part that is fixed on the ground during the action. For example, in the action "a person walks", the left big toe is fixed on the ground while the right leg is moving.
- the *global period* of the action: the global time duration of an action including all time periods of its sub-actions.
- the *variation of angles* of rotation around the part referential: the angle that the concerned part have to rotate around its referential. For example, in Figure 5.4, the right leg rotates angle α_1 around the hip for the "walking" action.
- the *speed* of the action: a scale to change the visualization speed of the given action.
- the *departure/arrival position* (optional, used only when the part is the whole individual): 3D positions in the observed environment. A position can be defined by a 3D point and also by the position of another object.
- the list of ***sub-actions*** with:
  + their relative period,
  + the concerned sub part of human body,
  + the variation of angles of rotation around the sub part referential.

Visualization of actions

An action is visualized by displaying the individual performing the action at regular instants. In the case where the test framework visualizes the actions recognized by an interpretation system, the individual posture to be visualized is obtained by the posture detected by the interpretation system. Therefore the test framework just needs to display the individual where it has been detected. If the visualization frequency is greater than the frequency of the input video, then it is necessary to inter-

polate linearly the intermediary positions of the individual. Knowing the global position of the individual, we calculate the vertices of the geometric primitives of the individual body in the scene referential and display the primitives by GEOMVIEW in the same way with the visualization of 3D context objects.

In the case where the test framework takes as input the actions modeled by an expert, the visualization process displays an action in three steps:

1) calculation of the current posture from the previous instant. By using the posture of the previous instant and the angular variations of the action, the visualization process calculates the new angular co-ordinates of each sub part of the human body at the current instant. From the new angular co-ordinates, the visualization process can calculate the new vertices of the primitives of each body part by multiplying their co-ordinates by the referential transformation matrix. This transformation matrix is defined for each body part and enables to compute co-ordinates in body part referential to co-ordinates in its containing body part referential. By this way the visualization process obtains the vertices of the body part defined relatively to the global position of the individual. These new co-ordinates define the new posture of the individual in the individual referential.

2) calculation of the global position of the individual. To calculate all positions of the individual, we make the following assumption: at each moment, there is a fixed point of a body part on the ground (see Figure 5.5). Currently, the actions that we are interested in are actions where the individual has a fixed part on the ground (e.g. “walking”, “running”). In the near future, we are planning to extend our formalism to handle actions such as “jumping above a barrier”. To calculate the global position, we first compute the distance between two successive fixed points on the ground (if the fixed point of the action has changed since last instant). Second, we compute the motion of the referential point of the individual relatively to the current fixed point. These two points (referential/fixed points) are defined by experts. By applying the transformation corresponding to this motion to the vertices of primitives defining the individual, we obtain the new co-ordinates of these vertices that correspond to the current posture of the individual. There are other approaches to calculate the position of an individual from its motion description. In [Delamarre and Faugeras, 1999], the authors have proposed a method to calculate the trajectory of individuals based on the combination of human body contour points. In [Bezault *et al*, 1992; Boulic *et al*, 1990], the authors describe the motion by mathematical equations (based on experimental data) and calculate the position of individuals by solving the equation system.

3) visualization: after computing the geometric primitives of the human body relatively to the new global position of the individual, we display all the primitives with GEOMVIEW.

Scenario

Generic Model of Scenarios

A scenario combines the individuals of the scene and the context objects which are relevant to the same activity with more elementary sub-scenarios. An elementary scenario is an action that corresponds to the motion of the whole human bodies of

the involved individuals. The model of scenarios is defined as the model of actions. It is a hierarchy of sub scenarios. Each sub scenario is ordered in time thanks to intervals (called *periods*) that correspond to the duration of the sub scenarios defined relatively to the global period of the main scenario. Unlike actions, a scenario has an attribute corresponding to the list of actors and context objects involved in the scenario. At the level of scenarios, an actor (or a context object) is represented by a variable that corresponds to the role of the actor/context object in the scenario.

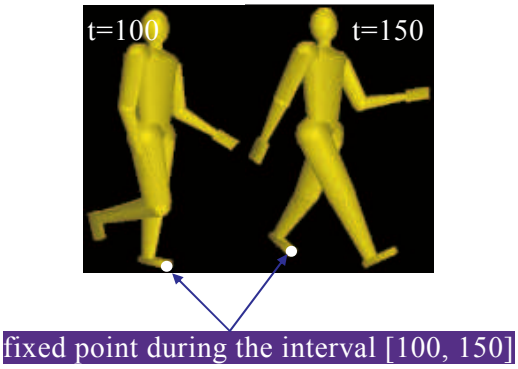


Figure 5.5: one of the fixed points while the individual is walking.

Visualization of Scenarios

We visualize a scenario in three steps. First, we link all actors and context objects of the scene involved in the scenario to the variables defined in the actions composing the scenario. Second, we order these actions in time: for each action, we calculate its duration (start and end point) relatively to the scenario period, defining when the action is active (is displayed). Third, at each instant, we display all actors involved in active actions using GEOMVIEW. Figure 5.6 presents the visualization of the scenario "two persons meet at a coffee machine" between the instants 80 and 240.

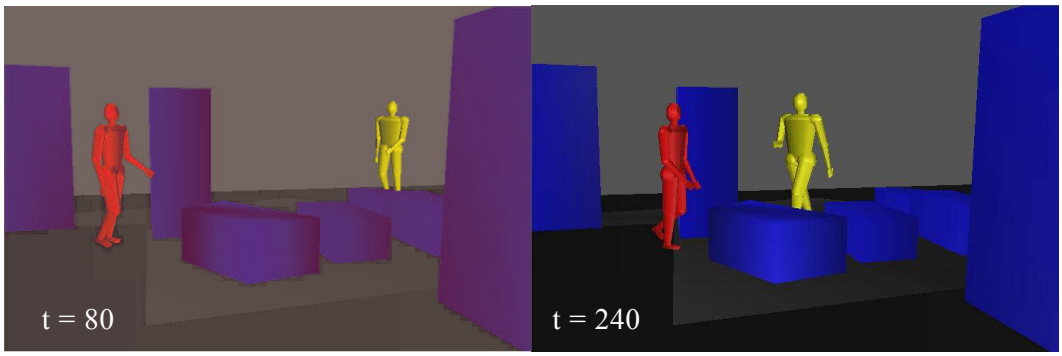


Figure 5.6: Visualization of the scenario "two persons meet at a coffee machine" at the instants 80 and 240.

Scene-Scenario

Generic Model of Scene-Scenarios

A scene-scenario combines and instantiates all previously defined scenarios. To represent a scene-scenario we use a generic model that has five attributes:

- the scene context includes the list of context objects involved in the scene. The expert describing the scene can change the default attributes of the context objects (e.g. their color).
- the virtual camera information that corresponds to the viewpoint from where the 3D animation is visualized. This information includes the 3D position, the direction and the field of view (FOV) of the camera.
- the list of actors involved in the scene with their initial positions, sizes, postures and colors. If this information is not provided, default values are used.
- a set of scenarios occurring in the scene. For each scenario, we first specify which actor corresponds to which role defined in the scenario and we also specify the scenario period relatively to the global period of the scene-scenario.
- the visualization speed of the scene-scenario.

Visualization of Scene-Scenarios

We display a scene-scenario in three steps. First, we initialize and connect the actors and the context objects to the scenarios defined in the scene. Second, we calculate the parameters of the virtual camera of GEOMVIEW. Third, we display all active scenarios composing the scene at each instant. The visualization frame rate can be specified either at the level of the scene-scenario or at the level of the scenarios or actions.

3 Visualization of Temporal Scenario Models

Section 1 has showed the proposed test framework for video interpretation systems. One of the proposed functionalities of the test system integrated in this framework is to generate automatically a set of videos corresponding to the variants of a given temporal scenario model M for the recognition [chapter 3]. The objective is to have different videos corresponding to a modeled situation to test the video interpretation system. There are a number of interesting variants to be studied, for example: variant of (1) temporal constraints, (2) individuals present in the scene, (3) trajectories of individuals and (4) visual phenomena (e.g. color, number/positions of light sources).

For a given scenario model M (for the recognition), the simulation process has to build and solve a system of inequations corresponding to the temporal constraints defined within M to generate different scenario models for the visualization [2.1]. Moreover, it also has to add random phenomena (e.g. trajectories, visual phenomena) to obtain a rich enough set of testing videos. After computing all scenario models for the visualization, the test system can generate all corresponding videos as shown in [2.1].

The tasks of generating different scenario models for the visualization corresponding to a given temporal scenario model concern different research/application domains and contribute a part of the master thesis of Jihene Bannour, Orion research team, INRIA Sophia-Antipolis research unit, France.

Index

This thesis research focuses on the **recognition of temporal scenarios** for **Automatic Video Interpretation**: the goal of this work is to recognize in **real-time** the behaviors of individuals evolving in a scene depicted by video sequences which were captured by cameras. The recognition process takes the following as input: (1) human behavior (i.e., temporal scenario) models predefined by experts; (2) 3D geometric and semantic information of the observed environment; and (3) a stream of individuals tracked by a vision module.

To deal with this issue, we have proposed a generic model of temporal scenarios and a description language to represent the knowledge of human behaviors. The representation of this knowledge needs to be clear, rich, intuitive and flexible. The proposed model of a temporal scenario M is composed of five components: (1) a set of physical object variables corresponding to the physical objects involved in M; (2) a set of temporal variables corresponding to the sub-scenarios composing M; (3) a set of forbidden variables corresponding to the scenarios that are not allowed to occur during the recognition of M; (4) a set of constraints (symbolic, logical, spatial and temporal constraints including Allen's interval algebra operators) involving these variables; and (5) a set of decisions corresponding to the tasks predefined by experts that are needed to be executed when M has been recognized.

We have also proposed a temporal constraint resolution technique to recognize in **real-time** the temporal scenario models predefined by experts. The proposed algorithm is most of the time efficient for processing temporal constraints as well as for combining several actors defined within a given scenario M. By efficient we mean that the recognition process is linear with the number of sub-scenarios and with the number of physical object variables defined within M in most cases.

To validate the proposed algorithm in terms of correctness, robustness and processing time with respect to scenario and scene properties (e.g., number of sub-scenarios, number of persons in the scene), we have tested the algorithm on several videos of different applications, in both on-line and off-line modes and also on simulated data.

By the experiments conducted in metro surveillance and bank monitoring applications, the proposed scenario description language shows the capability to represent easily temporal scenarios corresponding to the human behaviors of interest in these applications. Moreover, the proposed temporal scenario recognition algorithm shows the capability to recognize in **real-time** (at least 10 frames/second) complex scenario models (up to 10 physical object variables and 10 sub-scenario variables per scenario) with complex video sequences (up to 240 persons/frame in the scene).

Cette thèse traite de la **reconnaissance de scénarios temporels** pour l'**interprétation automatique de séquences vidéos** : l'objectif est de reconnaître à **cadence vidéo** les comportements d'individus évoluant dans des scènes décrites par des séquences vidéos (acquises par des caméras). Le processus de reconnaissance prend en entrée (1) les modèles de comportements humains (i.e. scénarios temporels) pré-définis par des experts, (2) les informations sémantiques et géometriques-3D de l'environnement observé et (3) les individus suivis par un module de vision.

Pour résoudre ce problème, premièrement, nous avons proposé un modèle générique de scénarios temporels et un langage de description pour la représentation de connaissances décrivant des comportements humains. La représentation de ces connaissances doit être claire, riche, intuitive et flexible pour être compris par les experts du domaine d'application. Le modèle proposé d'un scénario temporel M se compose de cinq parties : (1) un ensemble de variables correspondant aux acteurs impliqués dans M, (2) un ensemble de variables temporelles correspondant aux sous-scénarios qui composent M, (3) un ensemble de variables interdites correspondant aux scénarios qui ne doivent pas être reconnus pendant la reconnaissance de M, (4) un ensemble de contraintes (symboliques, logiques, spatiales et contraintes temporelles comprenant les opérateurs de l'algèbre d'intervalles d'Allen) portant sur ces variables et (5) un ensemble de décisions correspondant aux tâches pré-définies par les experts pour être exécutées quand M est reconnu.

Deuxièmement, nous avons proposé une technique originale de résolution de contraintes temporelles pour la reconnaissance à **cadence vidéo** de modèles de scénarios temporels pré-définis par des experts. En général, l'algorithme proposé est efficace car il propage les contraintes temporelles et combine seulement les objets physiques définis dans le scénario donné M. Par efficace, nous voulons dire que le processus de reconnaissance est linéaire en fonction du nombre de sous-scénarios et, dans quasiment tous les cas, en fonction du nombre d'objets physiques définis dans M.

Pour valider l'algorithme proposé en termes d'exactitude, de robustesse et du temps de traitement en fonction de la complexité des scénarios et de la scène (e.g. nombre de sous-scénarios, nombre de personnes dans la scène), nous avons testé l'algorithme en appuyant sur un grand nombre de vidéos provenant de différentes applications sur des données simulées et également réelles en modes hors-ligne/en-ligne.

Les expérimentations réalisées dans différentes applications montrent la capacité du langage de description de scénarios à représenter facilement les scénarios temporels correspondant aux comportements humains d'intérêt. De plus, ces expérimentations montrent également la capacité de l'algorithme proposé à reconnaître à **cadence vidéo** des modèles de scénarios sophistiqués (jusqu'à 10 acteurs et 10 sous-scénarios par scénario) dans des séquences vidéos complexes (jusqu'à 240 personnes/frame dans la scène).