Ontologies For Video Events

François Bremond, Nicolas Maillot, Monique Thonnat, Van-Thinh Vu

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Ontologies For Video Events

Francois Bremond, Nicolas Maillot, Monique Thonnat and Van-Thinh Vu

N° 5189

05/2004

THÈME Cog

ISSN 0249-6399

Ontologies For Video Events

Francois Bremond, Nicolas Maillot, Monique Thonnat and Van-Thinh Vu[1]

Thème Cog C - Données multi-media : interprétation et interaction home-machine
Projet Orion

Rapport de recherche n° 5189 – Mai 2004 - 21 pages

Abstract: This report shows how we represent video event knowledge for Automatic Video Interpretation. To solve this issue, we first build an ontology structure to design concepts relative to video events. There are two main types of concepts to be represented: physical objects of the observed scene and video events occurring in the scene. A physical object can be a static object (e.g. a desk, a machine) or a mobile object detected by a vision routine (e.g. a person, a car). A video event can be a primitive state, composite state, primitive event or composite event. Primitive states are atoms to build other concepts of the knowledge base of an Automatic Video Interpretation System. A composed concept (i.e. composite state and event) is represented by a combination of its sub-concepts and possibly a set of events that are not allowed occurring during the recognition of this concept. We use non-temporal constraints (logical, spatial constraint) to specify the physical objects involved in a concept and also temporal constraints including Allen's interval algebra operators to describe relations (e.g. temporal order, duration) between the sub-concepts defined within a composed concept. Secondly, we validate the proposed video event ontology structure by building two ontologies (for Visual Bank and Metro Monitoring) using ORION's Scenario Description Language.

Keywords: video event ontology, video event representation, video event language, video interpretation.

[1] Orion - {Francois.Bremond, Nicolas.Maillot, Monique.Thonnat, Thinh.Vu}@sophia.inria.fr

Unité de recherche INRIA Sophia-Antipolis
2004, route des Lucioles, 06902 Sophia-Antipolis (France)
Téléphone : +33 (0)4 92 38 77 77 – Télécopie : +33 (0)4 92 38 77 65

ontologies pour
la représentation d'événements de vidéo

Résumé: ce rapport décrit comment représenter les connaissances sur les événements vidéos pour l'Interprétation Automatique de Séquences Vidéos. Notre approche consiste d'abord à construire une structure d'ontologie pour représenter les concepts de base relatifs aux événements vidéos. Il y a deux types de concepts de base : les objets physiques de la scène observée et les événements vidéos apparaissants dans la scène. Un objet physique peut être un objet statique (e.g. un bureau, une machine) ou un objet mobile détecté par un programme de vision (e.g. une personne, une voiture). Un événement vidéo est un état ou un événement qui peut être primitif ou composite. Les états primitifs sont des atomes permettant de construire d'autres concepts de la base de connaissances dédiés à l'Interprétation Automatique de Séquences Vidéos. Un concept composé (i.e. état composite ou événement) est représenté par une combinaison de ses sous-concepts et éventuellement d'un ensemble d'événements ne devant pas se produire pendant la reconnaissance de ce concept. Nous utilisons des contraintes non-temporelles (contraintes logiques et spatiales) pour spécifier les objets physiques impliqués dans un concept donné et également des contraintes temporelles incluant l'algèbre d'intervalles d'Allen pour décrire des relations (e.g. ordre temporelle ou durée) entre les sous-concepts définis à l'intérieur du concept composé. Deuxièmement, nous validons cette structure de l'ontologie relative aux événements vidéos par la construction de deux ontologies (pour la Surveillance d'agences bancaires et pour les stations de métro) utilisant le Langage de Description de Scénarios développé dans l'équipe ORION.

Mots clés: ontologies pour événements vidéos, représentation d'événements vidéos, interprétation de vidéos.

1 Introduction

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 domain. 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 the 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 systems 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 domain 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.

In the two next sections, 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. In the fourth section, we briefly describe relations between the meta-concepts. In the fifth section, we propose a syntax for a representation language describing activities based on this ontology structure. We hope that this syntax can be used to describe the already built ontologies for video surveillance applications.

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.

2 Meta-Concepts for Describing Physical Objects

The **physical objects** are all the objects of the real world in the scene observed by the camera.
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.
The **attributes** of a physical object are all the properties characterizing the physical object.
The **liveliness** of a physical object is an attribute characterizing its mobility and autonomy in general. The value of this attribute can be: fixed (cannot be moved, e.g. a wall), static (can be moved under conditions, e.g. a closet), movable at the same position (e.g. a door), remotely-movable at the same position (e.g. a remotely-controlled barrier), automatically-movable at the same position (e.g. an automatic barrier, an escalator), displaceable (can be moved at different positions, include portable, e.g. a chair, a suitcase), remotely-displaceable (e.g. a ball, a puppet), remotely-movable (e.g. a remotely-controlled toy-car), programmable (e.g. a robot), autonomous (can initiate its motions with constraints, e.g. a train, a hand), fully-autonomous (can initiate motions with several degrees of freedom).
The **current liveliness** of a physical object is an attribute characterizing the perception by the observer of its liveliness at the current time in the scene. There are two values for current liveliness: *mobile* and *contextual*. In the following, we call **mobile object** the physical object with current liveliness equal to mobile and **contextual object** the physical object with current liveliness equal to contextual.
A **mobile object** can be perceived as moving in the scene and as initiating its motions. Its liveliness can be under conditions automatically-movable at the same position and always remotely-displaceable, remotely-movable, programmable, autonomous, fully-autonomous. Typical mobile objects are individuals, body parts, groups of people, animals, robots,…

An **agent** is a mobile object with a remotely-movable, programmable, autonomous or fully-autonomous liveliness. As an agent is a notion very close to mobile object, these two terms are considered as identical.
A **contextual object** cannot change its location in the scene or can be displaced by mobile objects. Its liveliness can be under conditions automatically-movable at the same position and always fixed, static, movable at the same position, remotely-movable at the same position, displaceable. Typical contextual objects are walls, entrance zones, doors, chairs, suitcases, escalators, trees, unoccupied scene,…
To distinguish mobile objects and contextual objects, the main point is their ability to initiate their own motion. For example, a car without driver is a contextual object, whereas, a car with a driver is a mobile object. The cars with/without driver are different physical objects with different liveliness even if there are the same appearance. So, a physical object cannot change its class/liveliness during an activity. Two physical objects can merge to induce the creation of a new physical object with the liveliness of the most autonomous object. However, the current liveliness 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 a contextual object, whereas, when someone plays with it, it is 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 an 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 defined as sub-classes of physical objects with usually a mobile current liveliness. Examples of sub-class for physical objects with usually a contextual current liveliness are portable object, equipment, zone.
The **role** is an attribute of a physical object. It defines the way it behaves and can be deduced by recognizing its past behavior 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 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.
There are four types of **relations**: (1) specialization between a class of physical objects and its sub-classes, (2) the composition of a physical object or an event made of sub-parts, (3) vision-based and spatio-temporal relations between physical objects and their attributes and (4) logical and temporal relations between states and events. These relations are more described in the next section.
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 are 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 relations of type (3) and logical relations between states of type (4) 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.
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. Only the relations of types (3) and (4) 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.
An **activity** is a set of loosely coupled events. The combinations of events are vaguely defined. Thus, an activity is a composite event with only constraints on its physical objects.
These meta-concepts are the sufficient and necessary meta-concepts enabling the description of any people evolution and any people interaction in a scene. However, there are other meta-concepts linked to the previous ones but used by other video understanding applications: action, situation, process, chronicle, scenario and plan. These meta-concepts can be defined based on previously defined meta-concepts.
An **action** is an event where a mobile object is achieving a task.
A **situation** is a state involving several physical objects.
A **process** is a composite event combining tightly coupled events. There are generally causal connections between the sub-events in a process.
A **chronicle**, **scenario** and **plan** are composite events with a temporal combination.
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).

4 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 in the time of spatial relations. 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*, *interruption*, *resumption*, *fork* and *join*. 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 optional notion can be used to describe several composite events (with/without optional sub-events).

5 Proposed Syntax to Describe Concepts

We have proposed a syntax to describe states, events and activities. These meta-concepts are described by a name and four parts:
PhysicalOobjects: declares the physical objects involved in the concept, their class and their role if needed. Only the physical objects that can change between concept occurrences need to be declared. The role of a physical object is indicated between brackets after its class.
Components: declares all the components constituting the concept. This is an optional part of the concept.
ForbiddenComponents: declares the events which should not happen during the concept occurrence. This is an optional part of the concept.
Constraints: declares all the relations between the sub-concepts (physical objects, components and forbidden events) of the concept.
Figures 1, 2 and 3 give a description of a model and an associated instance respectively of a primitive state, a primitive event and a composite multi-agent event.

```
Model:
PrimitiveState Inside_zone
  PhysicalObjects:
  ( p : Person, z : Zone)
  Constraints:
  ( p in z)

Instance:
s1: PrimitiveState Inside_zone(Paul, Entrance_zone)
```

Figure 1. Description of a primitive state model and an associated instance.

```
Model:
PrimitiveEvent Changes_zone
  PhysicalObjects:
  ( p : Person, z1 : Zone, z2 : Zone)
  Components:
  ( (c1: PrimitiveState Inside_zone(p, z1))
    (c2: PrimitiveState Inside_zone(p, z2)) )
  constraints:
  ( (distance(z1, z2) ≤ 1 m)
    (c1 before c2)

Instance:
e1: PrimitiveEvent Changes_zone(Paul, Entrance_zone,Front_counter)
```

Figure 2. Description of a primitive event model and an associated instance. In the constraints part, *distance* is a geometrical function between two physical objects.

```
Model:
CompositeEvent Bank_attack_one_robber_one_employee
  PhysicalObjects:
  ( e : Person[employee], r : Person[robber])
  Components:
  ( (c1: PrimitiveState Inside_zone(e, "Back_Counter"))
    (c2: PrimitiveEvent Changes_zone(r,"Entrance_Zone",
                                     "Front_Counter"))
    (c3: PrimitiveState Inside_zone(e, "Safe"))
    (c4: PrimitiveState Inside_zone(r, "Safe")) )
  Constraints:
  ( (duration-of(c3) ≥ 1 second)
    (c2 during c1)
    (c2 before c3)
    (c1 before c3)
    (c2 before c4)
    (c4 during c3) )

Instance:
e2:CompositeEvent Bank_attack_one_robber_one_employee(Paul,Robber1)
```

Figure 3. Description of a composite multi-agent event and an associated instance. In the constraints part, *duration-of* is an attribute of a state.

We can represent the relations between states/events using symbols:
In the components part:

1. logical "and" between two states/events: s1 s2
2. logical "or" between two states/events: s1 | s2.
3. an event/state is optional: [e]
4. iteration of N state/event occurrences: (e) * N

In the constraints part:

5. logical "and" between two constraints: ct1 ct2
6. logical "or" between two constraints: ct1 | ct2
7. sequence of two states/events: e1; e2
8. two states/events are in parallel: e1 || e2
9. condition between states/events: if e1 then e2 else e3

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, we have described in a separate document a reduced set of concepts used in video surveillance applications. We claim

that it is difficult to enumerate exhaustively every situations 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 movements description. Indeed, washing a window implies specific arm movements.

States and primitive events listed in the annex 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.

Annex 1: 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)

PrimitiveState Inside_zone
physical_objects:
((p : Person), (z : Zone))
constraints:
(p *in* z)

3. Simple Events

PrimitiveEvent Changes_zone
physical_objects:
((p : Person), (z1 : Zone), (z2 : Zone))
componants:
((c1 : PrimitiveState Inside_zone(p, z1))
(c2 : PrimitiveState 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)

CompositeEvent Safe_attack_1person_back_counter
physical_objects:
((p : Person), (z1: Back_Counter**)**, (z2: Safe))
componants:
(c1: PrimitiveEvent Changes_zone (p, z1, z2))
constraints:
(c2 *before* c1)

CompositeEvent Safe_attack_1person_back_counter_and_door_opened
physical_objects:
((p : Person), (z1: Back_Counter), (z2: Safe)**,** (g: Gate))
componants:
(c1: CompositeEvent Safe_attack_1person_back_counter(p, z1,z2))
constraints:
(g is opened)

CompositeEvent Safe_attack_1person_infront_counter
physical_objects:
((p : Person), (z1: Infront_Counter), (z2: Safe))
componants:
((c1: PrimitiveState Inside_zone(p, z1))
(c2: PrimitiveState Inside_zone(p, z2)))
constraints:
(c2 *before* c1)

CompositeEvent Safe_attack_1person_infront_counter_and_door_opened
physical_objects:
((p : Person), (z1: Infront_Counter), (z2: Safe), (g: Gate))
componants:
(c1: CompositeEvent Safe_attack_1person_infront_counter(p, z1, z2))
constraints:
(g is opened)

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

CompositeEvent Safe_attack_2persons_inside_safe_entrance
physical_objects:
((p1 : Person), (p2 : Person), (z1: Safe_Entrance))
componants:
((c1: PrimitiveState Inside_zone(p1, z1))
(c2: PrimitiveState Inside_zone(p2, z1)))
constraints:
(c2 *during* c1)

CompositeEvent Safe_attack_2persons_inside_safe_entrance_and_door_opened
physical_objects:
((p1 : Person), (p2 : Person), (z1: Safe_Entrance))
componants:
(c1: CompositeEvent Safe_attack_2persons_inside_safe_entrance(p1, p2, z1))
constraints:
(g is opened)

CompositeEvent Safe_attack_2persons_infront_counter
physical_objects:
((employee : Person), (robber : Person), (z1: Infront_Counter), (z2: Safe))
componants:
((c1: CompositeEvent Safe_attack_1person_infront_counter(employee, z1, z2))
(c2: CompositeEvent Safe_attack_1person_infront_counter(robber, z1, z2)))
constraints:
(c2 *during* c1)

CompositeEvent Safe_attack_2persons_infront_counter_and_door_opened
physical_objects:
((employee : Person), (robber : Person),
(z1: Infront_Counter), (z2: Safe), (g : Gate))
componants:
(c1: CompositeEvent Safe_attack_2persons_infront_counter(employee,
robber, z1))
constraints:
(g is opened)

CompositeEvent Safe_attack_2persons_back_counter
physical_objects:
((employee : Person), (robber : Person), (z1: Back_Counter), (z2: Safe))
componants:
((c1: CompositeEvent Safe_attack_1person_back_counter(employee, z1, z2))
(c2: CompositeEvent Safe_attack_1person_back_counter(robber, z1, z2)))
constraints:
(c2 *during* c1)

CompositeEvent Safe_attack_2persons_back_counter_and_door_opened
physical_objects:
((employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Safe), (g : Gate))
componants:
(c1: CompositeEvent Safe_attack_2persons_back_counter(employee, robber, z1))
constraints:
(g is opened)

CompositeEvent Safe_attack_2persons_back/infront_counter_1
physical_objects:
((employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe))
componants:
((c1: CompositeEvent Safe_attack_1person_infront_counter(employee, z2, z3))
(c2: CompositeEvent Safe_attack_1person_back_counter(robber, z1,z3)))
constraints:
(c2 *during* c1)

CompositeEvent 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))
componants:
(c1: CompositeEvent Safe_attack_2persons_back/infront_counter(employee, robber, z1,z2))
constraints:
(g is opened)

CompositeEvent Safe_attack_2persons_back/infront_counter_2
physical_objects:
((employee : Person), (robber : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe))
componants:
((c1: CompositeEvent Safe_attack_1person_infront_counter(robber, z2, z3))
(c2: CompositeEvent Safe_attack_1person_back_counter(employee, z1, z3)))
constraints:
(c2 *during* c1)

CompositeEvent 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))
componants:
(c1: CompositeEvent Safe_attack_2persons_back/infront_counter(employee, robber, z1,z2))
constraints:
(g is opened)

CompositeEvent Safe_attack_2persons
physical_objects:
((employee : Person), (robber : Person),
(z1: Entrance), (z2: Back_Counter), (z3: Infront_Counter), (z4: Safe))
componants:
((c1 : PrimitiveState Inside_zone(employee, z2))
(c2 : primitive_sevent Changes_zone(robber, z1,z3))
(c3: CompositeEvent Safe_attack_1person_back_counter(employee, z2, z4))
(c4: CompositeEvent Safe_attack_1person_infront_counter(robber, z3, z4)))
constraints:
((c2 *during* c1)
(c2 *before* c3)
(c1 *before* c3)
(c2 *before* c4)
(c4 *during* c3))

CompositeEvent Safe_attack_2persons_and_door_opened
physical_objects:
((employee : Person), (robber : Person), (z1: Entrance),
(z2: Back_Counter), (z3: Infront_Counter), (z4: Safe), (g : Gate))
componants:
(c1: CompositeEvent Safe_attack_2persons(employee, robber,
z1, z2, z3, z4))
constraints:
(g is opened)

// Scenarios with three persons

CompositeEvent Safe_attack_3persons_back/infront_counter
physical_objects:
((employee : Person), (robber : Person), (customer : Person),
(z1: Back_Counter), (z2: Infront_Counter), (z3: Safe))
componants:
((c1: PrimitiveState Inside_zone(customer, z2))
(c2: CompositeEvent Safe_attack_1person_infront_counter(robber, z2, z3))
(c3: CompositeEvent 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))

CompositeEvent 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))
componants:
(c1: CompositeEvent Safe_attack_3persons_back/infront_counter(employee,
robber, customer, z1, z2, z3))
constraints:
(g is opened)

CompositeEvent 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))
componants:
((c1: CompositeEvent Safe_attack_3person_back/infront_counter(employee, robber, customer, z2, z3, z4))
(c2 : PrimitiveEvent Changes_zone(customer, z3,z1)))
constraints:
(c2 *during* c1)

CompositeEvent
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))
componants:
(c1: CompositeEvent
Safe_attack_3persons_back/infront_counter_entrance(employee, robber, customer, z1, z2, z3, z4))
constraints:
(g is opened)

CompositeEvent 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))
componants:
((c1: CompositeEvent Safe_attack_3person_back/infront_counter(employee, robber, customer, z2, z3, z4))
(c2 : PrimitiveEvent Changes_zone(customer, z3,z1)))
constraints:
(c2 *during* c1)

CompositeEvent afe_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))
componants:
(c1: CompositeEvent
Safe_attack_3persons_back/infront_counter_entrance(employee, robber, customer, z1,z2, z3,z4))
constraints:
(g is opened)

Annex 2: 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

PrimitiveState Inside_zone
physical_objects:
((e : Ent), (z : Zone))
constraints:
(e ***in*** z)

PrimitiveState Stopped
physical_objects:
(e : Ent)
constraints:
(speed of e < minspeed)

PrimitiveState LyingPerson
physical_objects:
(p : Person)
constraints:
(Lying(p) is true)

PrimitiveState Groupwidthvariation
physical_objects:
(g : Group)
constraints:
(Width(g) > significantwidthvariation)

PrimitiveState Quicksplit
physical_objects:
(g : Group)
constraints:
(Split(g) > quicksplit)

PrimitiveState Trajectoryvariation
physical_objects:
(g : Group)
constraints:
(Trajectory (g) > significanttrajectoryvariation)

PrimitiveState Speed_increase
physical_objects:
(p : Person)
constraints:
(IncreaseSpeed(p) is true)

PrimitiveState Legs_up
physical_objects:
(p : Person)
constraints:
(LegsUp(p) is true)

3. Simple Events

PrimitiveEvent Changes_zone
physical_objects:
((p : Person), (z1 : Zone), (z2 : Zone))
componants:
((c1 : PrimitiveState Inside_zone(p, z1))
(c2 : PrimitiveState 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: PrimitiveState LyingPerson(g))
(c2: PrimitiveState Groupwidthvariation(g))
(c3: PrimitiveState Quicksplit(g))
(c4: PrimitiveState Trajectoryvariation(g)))
constraints:
//Alternatives
(c1 or c2 or c3 or c4)

composite _event Jumping
physical_objects:
(p: Person)
components:
((c1: PrimitiveState Speed_increase (p))
(c2: PrimitiveState Legs_up (p)))
constraints:
//Sequence
(c1; c2)

CompositeEvent Stays_inside_zone
physical_objects:
((e : Ent), (z : Zone))
components:
(c1: PrimitiveState Inside_zone(e,z))
forbidden_events:
(c2: PrimitiveEvent 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: PrimitiveEvent Changes_zone (p, z1, z2))
(c2: PrimitiveEvent Changes_zone (p, z2, z3)))
forbidden_events:
(c3: CompositeEvent 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: PrimitiveEvent Changes_zone (p, z1, z2))
(c2: CompositeEvent Jumping(p))
(c3: PrimitiveEvent Changes_zone (p, z2, z3)))
constraints:
//Sequence
(c1; c2; c3)

CompositeEvent Group_staying_in_zone // Blocking the access to a Zone Of Interest
physical_objects:
((g: group), (z: Zone))
components:
((c1: PrimitiveEvent enter (g, z))
(c2: PrimitiveState Stays_inside_zone(g, z)))
constraints:
(c1 before c2)
(duration of c2 > 120 seconds)

CompositeEvent Group_stopped_in_zone
physical_objects:
((g: Group),(z: Zone))
components:
((c1: PrimitiveEvent Enter (g, z))
(c2: PrimitiveState Stopped(g)))
constraints:
(c1 before c2)
(duration of c2 > 30 seconds)

CompositeEvent Blocking_ZOI // Blocking the access to a Zone Of Interest
physical_objects:
((g: Group), (z: Zone))
components:
((c1: CompositeEvent Group_stopped_in_zone(g, z))
(c2: CompositeEvent Group_staying_in_zone(g,z)))
constraints:
(c1 or c2)

CompositeEvent Vandalism_against_ticket_machine_one_man
physical_objects:
((p: Person), (eq1:Ticket_Vending_Machine),
(z1: Ticket_Vending_Machine_Zone))
componants:
((c1 : PrimitiveEvent Enters_zone(p, z1))
(c2 : PrimitiveEvent Move_close_to(p, eq1))
(c3 : CompositeEvent Stays_at(p, eq1))
(c4 : PrimitiveEvent Goes_away_from(p, eq1))
(c5 : PrimitiveEvent Move_close_to(p, eq1))
(c6 : CompositeEvent 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

CompositeEvent Vandalism_against_ticket_machine_two_men
physical_objects:
((p1 : Person), (p2 : Person), (eq1: Ticket_Vending_Machine),
(z1: Ticket_Vending_Machine_Zone), (z2: Surveillance_Zone))
componants:
((c1 : PrimitiveEvent Enters_zone(p1, z1))
(c2 : PrimitiveEvent Move_close_to(p1, eq1))
(c3 : CompositeEvent Stays_at(p1, eq1)))
(c4 : PrimitiveEvent Goes_away_from(p1, eq1))
(c5 : PrimitiveEvent Move_close_to(p1, eq1))
(c6 : CompositeEvent Stays_at(p1, eq1))
(c7 : PrimitiveEvent Enters_zone(p2, z2)))
forbidden_componants:
(c8 : PrimitiveEvent 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)