International Journal on Artificial Intelligence Tools
Vol. 19, No. 2 (2010) 193–209

DOI: 10.1142/S021821301000011X

World Scientific
www.worldscientific.com

BEHAVIOUR RECOGNITION FROM VIDEO CONTENT: A LOGIC PROGRAMMING APPROACH

ALEXANDER ARTIKIS*, ANASTASIOS SKARLATIDIS*,† and GEORGIOS PALIOURAS*

*Institute of Informatics and Telecommunications
National Centre for Scientific Research "Demokritos", Athens, 15310, Greece
†Department of Information and Communication Systems Engineering
University of the Aegean, Karlovassi, Samos 83200, Greece
{a.artikis, anskarl, paliourg}@iit.demokritos.gr

We present a system for recognising human behaviour given a symbolic representation of surveillance videos. The input of our system is a set of time-stamped short-term behaviours, that is, behaviours taking place in a short period of time — walking, running, standing still, etc — detected on video frames. The output of our system is a set of recognised long-term behaviours — fighting, meeting, leaving an object, collapsing, walking, etc — which are pre-defined temporal combinations of short-term behaviours. The definition of a long-term behaviour, including the temporal constraints on the short-term behaviours that, if satisfied, lead to the recognition of the long-term behaviour, is expressed in the Event Calculus. We present experimental results concerning videos with several humans and objects, temporally overlapping and repetitive behaviours. Moreover, we present how machine learning techniques may be employed in order to automatically develop long-term behaviour definitions.

Keywords: Event recognition; event calculus; machine learning.

1. Introduction

We address the problem of human behaviour recognition by separating low-level recognition, detecting activities that take place in a short period of time — "short-term behaviours" — from high-level recognition, recognising "long-term behaviours", that is, pre-defined temporal combinations of short-term behaviours. In this paper we present our work on high-level recognition. We evaluate our approach on a benchmark dataset of short-term behaviours identified on surveillance videos.

To perform high-level recognition we define a set of long-term behaviours of interest — for example, "leaving an object", "fighting" and "meeting" — as temporal combinations of short-term behaviours — for instance, "walking", "running" and "inactive". We employ a logic programming implementation of the Event Calculus,[8] an expressive, declarative temporal reasoning formalism, in order to define long-term behaviours. More precisely, we employ the Event Calculus to express the temporal constraints on a set of short-term behaviours that, if satisfied, lead to the recognition of a long-term behaviour.

A logic programming approach to behaviour recognition has, among others, the advantage that machine learning techniques can be directly employed in order to automatically develop a knowledge base of behaviour definitions. We present initial experiments with a combination of abductive and inductive logic programming techniques to learn behaviour definitions for the aforementioned dataset of surveillance videos.

The remainder of the paper is organised as follows. First, we present the Event Calculus. Second, we describe the dataset of short-term behaviours on which we perform long-term behaviour recognition. Third, we present our knowledge base of long-term behaviour definitions. Fourth, we present our experimental results. Fifth, we describe the use of machine learning techniques for automatically developing long-term behaviour definitions given the dataset of short-term behaviours. Finally, we discuss related work and outline directions for further research.

2. The Event Calculus

Our system for Long-Term Behaviour Recognition (LTBR) is a logic programming implementation of an Event Calculus formalisation, expressing long-term behaviour definitions. The Event Calculus (EC), introduced by Kowalski and Sergot,[8] is a formalism for representing and reasoning about actions or events and their effects. We present here the version of the EC that we employ (more details about this version may be found in Ref. 2).

EC is based on a many-sorted first-order predicate calculus. For the version used here, the underlying time model is linear and it may include real numbers or integers. Where F is a *fluent* — a property that is allowed to have different values at different points in time — the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. Informally, $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an action at some earlier time-point, and not *terminated* by another action in the meantime.

An *action description* in EC includes rules that define, among other things, the action occurrences (with the use of the $\mathsf{happens}$ predicate), the effects of actions (with the use of the $\mathsf{initiates}$ and $\mathsf{terminates}$ predicates), and the values of the fluents (with the use of the $\mathsf{initially}$, $\mathsf{holdsAt}$ and $\mathsf{holdsFor}$ predicates). Table 1 summarises the main EC predicates. Variables, starting with an upper-case letter, are assumed to be universally quantified unless otherwise indicated. Predicates, function symbols and constants start with a lower-case letter.

The $\mathsf{holdsAt}$ predicate is defined as follows:

$$\begin{aligned} &\mathsf{holdsAt}(F = V, T) \leftarrow \\ &\quad \mathsf{initially}(F = V), \\ &\quad \mathsf{not\ broken}(F = V, 0, T) \end{aligned} \tag{1}$$

Table 1. Main predicates of the event calculus.

Predicate	Meaning
$\mathsf{happens}(Act, T)$	Action Act occurs at time T
$\mathsf{initially}(F = V)$	The value of fluent F is V at time 0
$\mathsf{holdsAt}(F = V, T)$	The value of fluent F is V at time T
$\mathsf{holdsFor}(F = V, I)$	I are the maximal intervals for which $F = V$ holds continuously
$\mathsf{initiates}(Act, F = V, T)$	The occurrence of action Act at time T initiates a period of time for which $F = V$
$\mathsf{terminates}(Act, F = V, T)$	The occurrence of action Act at time T terminates a period of time for which $F = V$

$$\begin{aligned} &\mathsf{holdsAt}(F = V, T) \leftarrow \\ &\quad \mathsf{happens}(Act, T'), T' < T, \\ &\quad \mathsf{initiates}(Act, F = V, T'), \\ &\quad \mathsf{not\ broken}(F = V, T', T) \end{aligned} \tag{2}$$

According to rule (1) a fluent holds at time T if it held initially (time 0) and has not been "broken" in the meantime, that is, terminated between times 0 and T. Rule (2) specifies that a fluent holds at a time T if it was initiated at some earlier time T' and has not been terminated between T' and T. "not" represents "negation by failure".[3] The auxiliary predicate broken is defined as follows:

$$\begin{aligned} &\mathsf{broken}(F = V, T_1, T_3) \leftarrow \\ &\quad \mathsf{happens}(Act, T_2), \\ &\quad T_1 \leq T_2, T_2 < T_3, \\ &\quad \mathsf{terminates}(Act, F = V, T_2) \end{aligned} \tag{3}$$

$F = V$ is "broken" between T_1 and T_3 if an event takes place in that interval that terminates $F = V$. Note that, according to the above rules, a fluent does not hold at the time that was initiated but holds at the time it was terminated.

A fluent cannot have more than one value at any time. The following rule captures this feature:

$$\begin{aligned} &\mathsf{terminates}(Act, F = V, T) \leftarrow \\ &\quad \mathsf{initiates}(Act, F = V', T), \\ &\quad V \neq V' \end{aligned} \tag{4}$$

Rule (4) states that if an action Act initiates $F = V'$ then Act also terminates $F = V$, for all other possible values V of the fluent F. We do not insist that a fluent must have a value at every time-point. In this version of EC, therefore, there is a difference between initiating a Boolean fluent $F = \mathsf{false}$ and terminating $F = \mathsf{true}$: the first implies, but is not implied by, the second.

The intervals for which a fluent has a particular value are computed with the use of the holdsFor predicate. Below is a skeleton of this predicate:

$$\begin{aligned} &\mathsf{holdsFor}(F = V, I) \leftarrow \\ &\quad \mathsf{start}(F = V, StartPts), \\ &\quad \mathsf{end}(F = V, EndPts), \\ &\quad \mathsf{compute\_intervals}(StartPts, EndPts, I) \end{aligned} \tag{5}$$

The start predicate computes a list of time-points in which $F = V$ is initiated. If $F = V$ held initially then the output of start includes 0. The end predicate computes a list of time-points in which $F = V$ is terminated. Given the output of these predicates, compute_intervals computes the maximal intervals of time-points for which $F = V$ holds continuously. The computed intervals are of the form $(T_1, T_2]$ or $\mathsf{since}(T)$. To save space we do not present here the complete formalisation of holdsFor; the interested reader is referred to the source code of LTBR, which is available upon request.

3. Short-Term Behaviours: The CAVIAR Dataset

LTBR includes an EC action description expressing long-term behaviour definitions. The input to LTBR is a symbolic representation of short-term behaviours. The output of LTBR is a set of recognised long-term behaviours. In this paper we present experimental results given the short-term behaviours of the first dataset of the CAVIAR project.[a] This dataset includes 28 surveillance videos of a public space. The videos are staged — actors walk around, browse information displays, sit down, meet one another, leave objects behind, fight, and so on. Each video has been manually annotated in order to provide the ground truth for both short-term and long-term behaviours.

For the presented set of experiments the input to LTBR is: (i) the short-term behaviours walking, running, active (body movement in the same position) and inactive (standing still) — these behaviours are mutually exclusive — along with their time-stamps, that is, the frame in which a short-term behaviour took place, (ii) the coordinates of the tracked people and objects as pixel positions at each time-point, and (iii) the first and the last time a person or object is tracked "appears"/"disappears"). Given this input, LTBR recognises the following long-term behaviours: a person leaving an object, a person being immobile, people meeting, moving together, or fighting.

[a] `http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/`

Long-term behaviours are represented as EC fluents in order to use the built-in (domain-independent) holdsFor predicate for computing the intervals of these behaviours. Short-term behaviours are represented as EC actions in order to use the initiates and terminates predicates for expressing the conditions in which these behaviours initiate and terminate a long-term behaviour. In the following section we present example fragments of all long-term behaviour definitions.

4. Long-Term Behaviour Definitions

The long-term behaviour "leaving an object" is defined as follows:

$$\begin{aligned}&\mathsf{initiates}(\mathit{inactive}(\mathit{Object}), \mathit{leaving\_object}(\mathit{Person}, \mathit{Object}) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(\mathit{appearance}(\mathit{Object}) = \mathit{appear}, T), \\ &\quad \mathsf{holdsAt}(\mathit{close}(\mathit{Person}, \mathit{Object}, 30) = \mathsf{true}, T), \\ &\quad \mathsf{holdsAt}(\mathit{appearance}(\mathit{Person}) = \mathit{appear}, T_0), \\ &\quad T_0 < T\end{aligned} \tag{6}$$

$$\mathsf{initiates}(\mathit{exit}(\mathit{Object}), \mathit{leaving\_object}(\mathit{Person}, \mathit{Object}) = \mathsf{false}, T) \tag{7}$$

Rule (6) expresses the conditions in which a "leaving an object" behaviour is recognised. The fluent recording this behaviour, *leaving_object*(*Person*, *Object*), becomes true at time T if *Object* is inactive at T, *Object* "appears" at T, there is a *Person* close to *Object* at T (in a sense to be specified below), and *Person* has appeared at some time earlier than T. The *appearance* fluent records the times in which an object/person "appears" and "disappears". The $\mathit{close}(A, B, D)$ fluent is true when the distance between A and B is at most D. The distance between two tracked objects/people is computed given their coordinates. The value of 30 pixel positions was based on an empirical analysis of the dataset.

An object exhibits only inactive short-term behaviour. Any other type of short-term behaviour would imply that what is tracked is not an object. Therefore, the short-term behaviours active, walking and running do not initiate the *leaving_object* fluent. In the CAVIAR videos an object carried by a person is not tracked — only the person that carries it is tracked. The object will be tracked, that is, "appear", if and only if the person leaves it somewhere. Consequently, given rule (6), the *leaving_object* behaviour will be recognised only when a person leaves an object (see the second line of rule (6)), not when a person carries an object.

Rule (7) expresses the conditions in which a *leaving_object* behaviour ceases to be recognised. In brief, *leaving_object* is terminated — the value of this fluent becomes false — when the object in question is picked up. $\mathit{exit}(A)$ is an event that takes place when $\mathit{appearance}(A) = \mathit{disappear}$. An object that is picked up by someone is no longer tracked — it "disappears" — triggering an *exit* event which in turn terminates *leaving_object*.

In the present formalisation of the CAVIAR long-term behaviours, we chose to adopt the "strong termination" of fluents, and thus write rules of the form

initiates($ST, LT =$ false, T) instead of terminates($ST, LT =$ true, T), to express that short-term behaviour ST terminates long-term behaviour LT at time T. Consequently, in the present formalisation the heads of the rules expressing long-term behaviour definitions are only initiates predicates.

The long-term behaviour *immobile* is defined in order to signify that a person is resting in a chair or on the floor, or has fallen on the floor (fainted, for example). Note that there is no short-term behaviour in the annotation of the CAVIAR dataset for the motion of leaning towards the floor or a chair. The absence of such a short-term behaviour substantially complicates the definition of *immobile* (and reduces the accuracy of recognising *immobile* — see Section 5). Below is (a simplified version of) one of the rules of the *immobile* definition:

$$\begin{aligned}&\mathsf{initiates}(inactive(Person), immobile(Person) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{happens}(active(Person), T_0), \\ &\quad T_0 < T, \\ &\quad duration(inactive(Person), Intervals), \\ &\quad (T, T_1) \in Intervals, \\ &\quad T_1 > T{+}54\end{aligned} \tag{8}$$

According to rule (8), the behaviour *immobile*(*Person*) is recognised if *Person*: (i) has been active some time in the past (see lines 2–3 of rule (8)), and (ii) stays inactive for more than 54 frames (see lines 4–6 of rule (8)) — we chose this number of frames, like all other numerical constraints of the definitions, based on our empirical analysis of the CAVIAR dataset. *duration* is a predicate computing the maximal duration of inactive behaviour, that is, the number of consecutive instantaneous inactive events. The output of *duration* is a set of tuples of the form (s, e) where s is the time in which *inactive*(*Person*) started and e is the time in which *inactive*(*Person*) ended. (holdsFor computes the duration of fluents and thus cannot be used for computing the duration of inactive behaviour.) Note that this is not the only way to represent durative events in EC. See Ref. 25 for alternative representations.

Rule (8) has an additional constraint, requiring that *Person* is not close to an information display or a shop. If *Person* were close to a shop, then he would have to stay inactive much longer than 54 frames before *immobile* could be recognised. In this way we avoid classifying the behaviour of browsing a shop as *immobile*. To simplify the presentation we do not present here the extra constraint of rule (8).

The definition of *immobile* includes rules according to which *immobile*(*Person*) is recognised if *Person*: (i) has been walking some time in the past, and (ii) stays inactive for more than 54 frames. We insist that *Person* in *immobile*(*Person*) has been active or walking before being inactive, in order to distinguish between a left object, which is inactive from the first time it is tracked, from an immobile person.

immobile(*Person*) is terminated when *Person* starts walking, running or "disappears" — see rules (9)–(11) below:

$$\mathsf{initiates}(walking(Person), immobile(Person) = \mathsf{false}, T) \tag{9}$$

$$\mathsf{initiates}(running(Person), immobile(Person) = \mathsf{false}, T) \tag{10}$$

$$\mathsf{initiates}(exit(Person), immobile(Person) = \mathsf{false}, T) \tag{11}$$

The following rules represent a fragment of the *moving* behaviour definition:

$$\begin{aligned}&\mathsf{initiates}(walking(Person), moving(Person, Person_2) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(close(Person, Person_2, 34) = \mathsf{true}, T), \\ &\quad \mathsf{happens}(walking(Person_2), T)\end{aligned} \tag{12}$$

$$\begin{aligned}&\mathsf{initiates}(walking(Person), moving(Person, Person_2) = \mathsf{false}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(close(Person, Person_2, 34) = \mathsf{false}, T)\end{aligned} \tag{13}$$

$$\begin{aligned}&\mathsf{initiates}(active(Person), moving(Person, Person_2) = \mathsf{false}, T) \leftarrow \\ &\quad \mathsf{happens}(active(Person_2), T)\end{aligned} \tag{14}$$

$$\mathsf{initiates}(running(Person), moving(Person, Person_2) = \mathsf{false}, T) \tag{15}$$

$$\mathsf{initiates}(exit(Person), moving(Person, Person_2) = \mathsf{false}, T) \tag{16}$$

According to rule (12) *moving* is initiated when two people are walking and are close to each other (their distance is at most 34). *moving* is terminated when the people walk away from each other, that is, their distance becomes greater than 34 (see rule (13)), when they stop moving, that is, become active (see rule (14)) or inactive, when one of them starts running (see rule (15)), or when one of them "disappears" (see rule (16)).

The following rules express the conditions in which *meeting* is recognised:

$$\begin{aligned}&\mathsf{initiates}(active(Person), meeting(Person, Person_2) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(close(Person, Person_2, 25) = \mathsf{true}, T), \\ &\quad \mathsf{not\ happens}(running(Person_2), T)\end{aligned} \tag{17}$$

$$\begin{aligned}&\mathsf{initiates}(inactive(Person), meeting(Person, Person_2) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(close(Person, Person_2, 25) = \mathsf{true}, T), \\ &\quad \mathsf{not\ happens}(running(Person_2), T)\end{aligned} \tag{18}$$

meeting is initiated when two people "interact': at least one of them is active or inactive, the other is not running, and the distance between them is at most 25. This interaction phase can be seen as some form of greeting (for example, a handshake). *meeting* is terminated when the two people walk away from each other, or one

of them starts running or "disappears". The rules representing the termination of *meeting* are similar to rules (13), (15) and (16). Note that *meeting* may overlap with *moving*: two people interact and then start *moving*, that is, walk while being close to each other. In general, however, there is no fixed relationship between *meeting* and *moving*.

The rules below present the conditions in which *fighting* is initiated:

$$\begin{aligned}&\mathsf{initiates}(\mathit{active}(\mathit{Person}), \mathit{fighting}(\mathit{Person}, \mathit{Person}_2) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(\mathit{close}(\mathit{Person}, \mathit{Person}_2, 24) = \mathsf{true}, T), \\ &\quad \mathsf{not\ happens}(\mathit{inactive}(\mathit{Person}_2), T)\end{aligned} \tag{19}$$

$$\begin{aligned}&\mathsf{initiates}(\mathit{running}(\mathit{Person}), \mathit{fighting}(\mathit{Person}, \mathit{Person}_2) = \mathsf{true}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(\mathit{close}(\mathit{Person}, \mathit{Person}_2, 24) = \mathsf{true}, T), \\ &\quad \mathsf{not\ happens}(\mathit{inactive}(\mathit{Person}_2), T)\end{aligned} \tag{20}$$

Two people are assumed to be *fighting* if at least one of them is active or running, the other is not inactive, and the distance between them is at most 24. We have specified that running initiates *fighting* because, in the CAVIAR dataset, moving abruptly, which is what happens during a fight, is often classified as running. *fighting* is terminated when one of the people walks or runs away from the other, or "disappears" — see rules (21)–(23) below:

$$\begin{aligned}&\mathsf{initiates}(\mathit{walking}(\mathit{Person}), \mathit{fighting}(\mathit{Person}, \mathit{Person}_2) = \mathsf{false}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(\mathit{close}(\mathit{Person}, \mathit{Person}_2, 24) = \mathsf{false}, T)\end{aligned} \tag{21}$$

$$\begin{aligned}&\mathsf{initiates}(\mathit{running}(\mathit{Person}), \mathit{fighting}(\mathit{Person}, \mathit{Person}_2) = \mathsf{false}, T) \leftarrow \\ &\quad \mathsf{holdsAt}(\mathit{close}(\mathit{Person}, \mathit{Person}_2, 24) = \mathsf{false}, T)\end{aligned} \tag{22}$$

$$\mathsf{initiates}(\mathit{exit}(\mathit{Person}), \mathit{fighting}(\mathit{Person}, \mathit{Person}_2) = \mathsf{false}, T) \tag{23}$$

Under certain circumstances LTBR recognises both *fighting* and *meeting*. This happens when two people are active and the distance between them is at most 24. This problem would be resolved if the CAVIAR dataset included a short-term behaviour for abrupt motion, which would be used (instead of the active short-term behaviour) to initiate *fighting*, but would not be used to initiate *meeting*.

5. Experimental Results

We present our experimental results on 28 surveillance videos of the CAVIAR project. These videos contain 26419 frames that have been manually annotated in order to provide the ground truth for short-term and long-term behaviours. Table 2 shows the performance of LTBR — it shows, for each long-term behaviour, the number of True Positives (TP), False Positives (FP) and False Negatives (FN), as well as Recall and Precision. Long-term behaviours are recognised with the use of the holdsFor EC predicate.

Table 2. Experimental results.

Behaviour	True positive	False positive	False negative	Recall	Precision
leaving object	4	0	1	0.8	1
immobile	9	8	0	1	0.52
moving	16	12	1	0.94	0.57
meeting	6	3	3	0.66	0.66
fighting	4	8	2	0.66	0.33

LTBR correctly recognised 4 *leaving_object* behaviours. Moreover, there were no FP. On the other hand, there was 1 FN. This, however, cannot be attributed to LTBR because in the video in question the object was left behind a chair and was not tracked. In other words, the left object never "appeared", it never exhibited a short-term behaviour.

Regarding *immobile* we had 9 TP, 8 FP and no FN. The recognition of *immobile* would be much more accurate if there was a short-term behaviour for the motion of leaning towards the floor or a chair. In the absence of such a short-term behaviour, the recognition of *immobile* is primarily based on how long a person is inactive. In the CAVIAR videos, a person falling on the floor or resting in a chair stays inactive for at least 54 frames. Consequently LTBR recognises *immobile* if, among other things, a person stays inactive for at least 54 frames. There are situations, however, in which a person stays inactive for more than 54 frames and has not fallen on the floor or sat in a chair: people watching a fight, or just staying inactive waiting for someone. It is in those situations that we have the FP concerning *immobile*.

LTBR recognised correctly 16 *moving* behaviours. However, it also recognised incorrectly 12 such behaviours. Half of the FP concern people that do move together: walk towards the same direction while being close to each other. According to the manual annotation of the videos, however, these people do not exhibit the *moving* long-term behaviour. The remaining FP fall into two categories. First, people walk close to each other as they move to different directions — in these cases the duration of FP is very short. These FP could have been eliminated if there was (reliable) information about the orientation of the tracked people. Second, the short-term behaviours of people *fighting* are sometimes classified as walking. Consequently, the behaviour of these people is incorrectly recognised by LTBR as *moving* since, according to the manual annotation of the CAVIAR dataset, they are walking while being close to each other (moreover, their coordinates change). Introducing a short-term behaviour for abrupt motion would resolve this issue, as abrupt motion would not initiate *moving*.

LTBR did not recognise 1 *moving* behaviour. This FN was due to the fact that the distance between the people walking together was greater than the threshold we have specified. Increasing this threshold would result in substantially increasing the number of FP. Therefore we chose not to increase it.

LTBR recognised 9 *meeting* behaviours, 6 of which took place and 3 did not take place. 2 FP concerned *fighting* behaviours realised by people being active and close to each other. As mentioned in the previous section, in these cases LTBR recognises both *meeting* and *fighting*. The third FP was due to the fact that two people were active and close to each other, but were not interacting. LTBR did not recognise 3 *meeting* behaviours. 2 FN were due to the fact that the distance between the people in the meeting was greater than the threshold we have specified. If we increased this threshold LTBR would correctly recognise these 2 *meeting* behaviours. However, the number of FP for *meeting* would substantially increase. Therefore we chose not to increase this threshold. The third FN was due to the fact that the short-term behaviours of the people interacting — handshaking — were classified as walking, although one of them was actually active. We chose to specify that walking does not initiate a *meeting* in order to avoid incorrectly recognising meetings when people simply walk close to each other.

Regarding *fighting* we had 4 TP, 8 FP and 2 FN. The FP were mainly due to the fact that when a meeting takes place LTBR often recognises the long-term behaviour *fighting* (as well as *meeting*). LTBR did not recognise 2 *fighting* behaviours because in these two cases the short-term behaviours of the people *fighting* were classified as walking (recall the discussion on the recognition of *moving*). We chose to specify that walking does not initiate *fighting*. Allowing walking to initiate *fighting* (provided, of course, that two people are close to each other) would substantially increase the number of FP for *fighting*, because *fighting* would be recognised every time a person walked close to another person.

6. Machine Learning Event Definitions

Given the analysis presented so far, it is clear that the manual development of long-term behaviour definitions is a tedious and error-prone process. Consequently, a method for automatically generating such definitions from temporal data is required.

As LTBR is a logic program, Inductive Logic Programming (ILP) methods are an obvious candidate for constructing domain-dependent rules expressing long-term behaviour definitions. ILP can be used to generalise observations, producing hypotheses about yet unseen instances. A typical type of ILP learning, Observational Predicate Learning (OPL),[13] requires that both observations and their generalisations are described by the same predicate. In our EC-based behaviour recognition approach, however, OPL cannot be directly applied. This is due to the fact that the sequence of observations, expressing the ground truth for long-term behaviours, is represented by a set of holdsAt predicates, whereas the domain-dependent rules that need to be learnt, expressing long-term behaviour definitions, are represented by initiates predicates.

To overcome this problem, we employ a combination of abduction and induction techniques, as proposed in the literature.[10,18–20,22] The learning procedure

we developed has the following input: (i) a background theory, including the EC domain-independent rules — rules (1)–(5) — and domain-specific knowledge, such as the rules computing the distance between two people/objects, (ii) a narrative, expressing the short-behaviours taking place in each video, represented by happens Prolog facts, and the coordinates of each person/object in the video, and (iii) a set of observations, represented by holdsAt facts, expressing the long-term behaviours taking place in each video. (ii) and (iii) correspond, respectively, to the ground truth for short-term and long-term behaviours. The output of our learning procedure consists of domain-dependent rules, where the head of each rule is an initiates predicate expressing a long-term behaviour definition.

Our learning procedure follows three steps. First, given the background theory, narrative and observations, an Abductive Logic Programming (ALP) reasoner is employed to generate a set of abduced explanations in terms of ground initiates facts. Second, a *Kernel Set*[18,21] is formed by adding a set of body atoms, entailed by the background theory, to each ground initiates fact produced in the previous step. Third, an ILP reasoner is employed to induce a set of hypotheses, in terms of non-ground initiates rules, given the Kernel Set, background theory, narrative and observations. A detailed description of the operation of our learning procedure is given below.

For the first step of our procedure we employ the ProLogICA[23] ALP reasoner to construct abduced explanations given the background theory, narrative and observations. An additional input to the ALP reasoner is a set of integrity constraints, such as, for example, that an event may not initiate and terminate the same fluent at the same time. Due to substantial delays in ProLogICA's reasoning time, instead of parsing the complete set of observations and short-term behaviours of the narrative, we employ a sliding window approach. Each sliding window contains a subset of observations and short-term behaviours of the narrative. For example, a window of temporal range 12600 to 12760 (each video frame takes 40ms) includes the following narrative (short-term behaviours in terms of happens facts, and the coordinates of people/objects in terms of holdsAt facts) and observations (long-term behaviours in terms of holdsAt facts):

$\mathsf{happens}(walking(id6), 12600), \mathsf{happens}(walking(id7), 12600),$
$\mathsf{happens}(walking(id6), 12640), \mathsf{happens}(walking(id7), 12640),$
$\mathsf{happens}(walking(id6), 12680), \mathsf{happens}(walking(id7), 12680),$
$\mathsf{happens}(active(id6), 12720), \mathsf{happens}(active(id7), 12720).$

$\mathsf{holdsAt}(coord(id6) = (165, 79), 12600), \mathsf{holdsAt}(coord(id7) = (40, 64), 12600),$
$\mathsf{holdsAt}(coord(id6) = (168, 69), 12640), \mathsf{holdsAt}(coord(id7) = (\ 42, 65), 12640),$
$\mathsf{holdsAt}(coord(id6) = (165, 79), 12680), \mathsf{holdsAt}(coord(id7) = (160, 74), 12680),$
$\mathsf{holdsAt}(coord(id6) = (168, 69), 12720), \mathsf{holdsAt}(coord(id7) = (165, 65), 12720).$

$\mathsf{holdsAt}(\mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{false}, \mathit{12640}),$
$\mathsf{holdsAt}(\mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{false}, \mathit{12680}),$
$\mathsf{holdsAt}(\mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{true}, \mathit{12720}),$
$\mathsf{holdsAt}(\mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{true}, \mathit{12760}).$

Given that, in the employed EC version (see Section 2), the effects of an action *Act* (short-term behaviour in this example) taking place at time T come into play at the time-point following T, the observations of each window start from the second frame of the window. According to the sliding window approach, ProLogICA produces, for each window i, explanations Δ_i, given the observations that exist within the range of i, O_i, and the narrative within the range of i, N_i. Each O_i is entailed from the background theory, integrity constraints, N_i and Δ_i. At the end of this step, the complete set of explanations Δ is computed as the union of each Δ_i. Note that, in this example, the combination of background knowledge and integrity constraints eliminates the possibility of alternative inconsistent explanations. A fragment of an example output of our procedure's first step is presented below:

$$\begin{gathered}\mathsf{initiates}(\mathit{walking}(\mathit{id6}), \mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{false}, \mathit{12600})\\ \mathsf{initiates}(\mathit{walking}(\mathit{id6}), \mathit{fighting}(\mathit{id7}, \mathit{id6}) = \mathsf{false}, \mathit{12640})\\ \mathsf{initiates}(\mathit{walking}(\mathit{id6}), \mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{true}, \mathit{12680})\\ \mathsf{initiates}(\mathit{active}(\mathit{id6}), \mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{true}, \mathit{12720})\end{gathered}$$

In the second step of our learning procedure, the Kernel Set is constructed by a deductive process which adds to each ground initiates fact produced in the previous step, a body atom expressing the distance between the people/objects exhibiting a particular long-term behaviour. Consider the following fragment of an example output of this step of the learning procedure:

$$\begin{gathered}\mathsf{initiates}(\mathit{walking}(\mathit{id6}), \mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{false}, \mathit{12640}) \leftarrow\\ \mathsf{holdsAt}(\mathit{close}(\mathit{id6}, \mathit{id7}, \mathit{24}) = \mathsf{false}, \mathit{12640})\end{gathered}$$

$$\begin{gathered}\mathsf{initiates}(\mathit{walking}(\mathit{id6}), \mathit{fighting}(\mathit{id6}, \mathit{id7}) = \mathsf{true}, \mathit{12680}) \leftarrow\\ \mathsf{holdsAt}(\mathit{close}(\mathit{id6}, \mathit{id7}, \mathit{24}) = \mathsf{true}, \mathit{12680})\end{gathered}$$

Recall that the $\mathit{close}(A, B, D)$ fluent is true when the distance between A and B is at most D.

Effectively in this step we compute, for each long-term behaviour, the threshold distance associated with the initiation (respectively termination) of this behaviour; when the distance between two people/objects is less (respectively greater) than the threshold, then the long-term behaviour under consideration may be initiated (respectively terminated). The computation of the threshold distance for a long-term behaviour is based on the cumulative distribution of distances between the people/objects initiating this behaviour. The threshold distance is set to the minimum distance that maximizes the corresponding distribution.

For the third step of our learning procedure we employ the Aleph[26] ILP reasoner to induce non-ground initiates rules, given the Kernel Set produced in the previous step, background theory, narrative and observations. The produced Kernel Set forms the positive example set of the induction phase. The negative example set is constructed as follows. For every rule r of the Kernel Set, if the head of r is $\mathsf{initiates}(st, lt = \mathsf{true}, t)$, meaning that short-term behaviour st initiates long-term behaviour lt at time t, we add in the negative example set $\mathsf{initiates}(st', lt = \mathsf{false}, t)$, for every short-term behaviour st'. Similarly, for every rule r of the Kernel Set, where the head of r is $\mathsf{initiates}(st, lt = \mathsf{false}, t)$, we add in the negative example set $\mathsf{initiates}(st', lt = \mathsf{true}, t)$, for every short-term behaviour st'.

In order to deal with the noise in the video dataset — as mentioned in Section 5, according to the annotation of the dataset, a pair of short-term behaviours may initiate a long-term behaviour at some time-points and terminate the same behaviour at other time-points — we have configured the ILP reasoner to allow for (restricted) coverage of negative examples.

We performed initial experiments with the learning procedure on selected videos of the CAVIAR dataset. We observed that the "quality" of the induced rules (expressing long-term behaviour definitions), that is, the accuracy of behaviour recognition achieved by the induced rules, heavily depends on the choice of tolerance threshold for noise handling (the allowed coverage of negative examples and the required coverage of positive examples). Qualitatively, our preliminary results are encouraging in the sense that, when the noise tolerance threshold is set appropriately, the induced rules are a good approximation of the manually developed rules. Further experimentation is of course necessary in order to validate our approach. Additionally, we need to reduce the sensitivity of our method on the choice of noise tolerance threshold.

7. Related Work

A well-known system for behaviour recognition is the Chronicle Recognition System (CRS).[b] A "chronicle" can be seen as a long-term behaviour — it is expressed in terms of a set of events (short-term behaviours in our example), linked together by time constraints, and, possibly, a set of context constraints. The input language of the CRS relies on a reified temporal logic, where propositional terms are related to time-points or other propositional terms. Time is considered as a linearly ordered discrete set of instants. The language includes predicates for persistence and event absence. Details about the input language of the CRS, and CRS in general, may be found on the web page of the system and in Refs. 4–6.

The CRS language does not allow mathematical operators in the constraints of atemporal variables. Consequently, the computation of the distance between two people/objects cannot be computed. CRS, therefore, cannot be directly used

[b] `http://crs.elibel.tm.fr/`

for behaviour recognition in video surveillance applications. More generally, CRS cannot be directly used for behaviour recognition in applications requiring any form of spatial reasoning, or any other type of atemporal reasoning. These limitations could be overcome by developing a separate tool for atemporal reasoning that would be used by CRS whenever this form of reasoning was required. To the best of our knowledge, such extensions of CRS are not available. Clearly, the computational efficiency of CRS, which is one of the main advantages of using this system for behaviour recognition, would be compromised by the integration of an atemporal reasoner.

In our approach to behaviour recognition, the availability of the full power of logic programming, which is one of the main attractions of employing the Event Calculus (EC) as the temporal formalism, allows for the development of behaviour definitions including complex temporal (EC is at least as expressive as the CRS language with respect to temporal representation) and atemporal constraints. When necessary, more expressive EC versions may be employed (see Refs. 9 and 25 for presentations of the EC expressiveness).

Paschke and colleagues[15–17] have proposed a logic programming implementation of an EC version for behaviour recognition. Unlike our EC version, there is no support for multi-valued fluents — only Boolean fluents are considered. The use of multi-valued fluents makes the representation considerably more succinct. Moreover, the EC version of Paschke and colleagues does not include rules for recognising a behaviour that has been initiated at some earlier time-point and has not (yet) terminated. For example, there is no built-in support for recognising an on-going *fighting* behaviour. Our treatment of behaviours as EC fluents in combination with the holdsFor predicate for computing the intervals in which a fluent holds, allows us to overcome the above limitation. In the case of an on-going *fighting* behaviour, for instance, an answer to a query regarding *fighting* would be of the form since(T), indicating that the recognition of *fighting* started at T and has not (yet) ended.

We described our first steps towards automatically constructing behaviour definitions expressed as logic programming implementations of EC. There are several approaches in the literature for learning EC-based logic programs. Moyle and Muggleton,[11] for instance, combined a generalisation of mode-directed inverse entailment[12] with Stickel's theorem prover,[27] in order to perform non-observational predicate learning and produce EC domain-dependent rules from observations. The ALECTO system[10] combines abduction with induction to address the same problem. Our approach was motivated by the more recently developed HAIL and XHAIL systems[1,18–20,22] that also use a hybrid abductive-inductive method to perform non-observational predicate learning.[c] We opted for developing a new system based on the philosophy of HAIL and XHAIL, in order to have the flexibility of adding new features to it. One such feature is the use of a sliding window for abduction, which was necessary in order to handle the size of the CAVIAR dataset with

[c]There are currently no publicly available versions of HAIL and XHAIL.

ProLogICA. Since the same abductive reasoner forms the basis of HAIL and XHAIL, we expect that a similar extension of these systems will be necessary. Clearly HAIL and XHAIL have features that our preliminary approach is lacking, including the computation of a consistent rule-set, as opposed to single rules. We are currently working towards incorporating such features in our approach.

8. Summary and Future Work

We presented a behaviour recognition system based on an EC logic programming implementation, and outlined the advantages of our system with respect to state-of-the-art recognition systems. We showed experimental results on a benchmark video surveillance dataset. We are currently investigating the extent to which recognition accuracy may improve. For instance, we are investigating the extent to which a finer classification of short-term behaviours, offered by existing short-term behaviour recognition systems (such as, for example, the system presented in Ref. 7), and respective modification of long-term behaviour definitions, reduce False Positives and False Negatives.

A logic programming approach to behaviour recognition has the additional advantage that machine learning techniques can be directly employed in order to automatically develop a knowledge base of behaviour definitions. We presented initial experiments with a combination of abductive and inductive logic programming techniques for constructing such definitions. There are several directions for further work concerning this task. We are investigating the effects of enhancing the Kernel Set (adding more body atoms in the rules of this set) on the efficiency of the induction phase as well as on the "quality" of the induced rules. Moreover, we are considering techniques for theory learning. To address the problems that arise from large search spaces, such as that of our example, we are considering techniques for theory revision,[14,24,28] assuming that some rules, however incomplete, may be manually produced. Further experimentation with noisy datasets is also necessary. In the context of the EU-project PRONTO[d] we will automatically develop long-term behaviours for emergency rescue operations and public transport services, using data from video cameras, microphones, accelerometers and other sensor types that are available in these fields.

Acknowledgements

This work has been partially funded by the EC, in the context of the PRONTO Project (FP7-ICT 231738).

We would like to thank Christophe Dousson for his suggestions regarding the Chronicle Recognition System. The authors themselves, however, are solely responsible for any misunderstanding about the use of this technology.

[d]`http://www.ict-pronto.org/`

References

1. D. Alrajeh, O. Ray, A. Russo, and S. Uchitel. Extracting requirements from scenarios with ILP. In *Inductive Logic Programming*, volume LNAI 4455. Springer, 2007.
2. A. Artikis, M. Sergot, and J. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1), 2009.
3. K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, 1978.
4. C. Dousson. Extending and unifying chronicle representation with event counters. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 257–261. IOS Press, 2002.
5. C. Dousson and P. Le Maigat. Chronicle recognition improvement using temporal focusing and hierarchisation. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 324–329, 2007.
6. M. Ghallab. On chronicles: Representation, on-line recognition and learning. In *Proceedings of Conference on Principles of Knowledge Representation and Reasoning*, pages 597–606, 1996.
7. D. Kosmopoulos, P. Antonakaki, K. Valasoulis, A. Kesidis, and S. Perantonis. Human behaviour classification using multiple views. In *Proceedings of Hellenic Conference on Artificial Intelligence*, volume 5138. Springer, 2008.
8. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.
9. R. Miller and M. Shanahan. The event calculus in a classical logic — alternative axiomatizations. *Journal of Electronic Transactions on Artificial Intelligence*, 4(16), 2000.
10. S. Moyle. Using theory completion to learn a robot navigation control program. In *ILP*, volume 2583 of *Lecture Notes in Computer Science*, pages 182–197, 2002.
11. S. Moyle and S. Muggleton. Learning programs in the event calculus. In *ILP*, volume 1297 of *Lecture Notes in Computer Science*, pages 205–212. Springer, 1997.
12. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3):245–286, 1995.
13. S. Muggleton and C.H. Bryant. Theory completion using inverse entailment. In *ILP*, volume 1866 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2000.
14. A. Paes, G. Zaverucha, and V. Santos Costa. Revising first-order logic theories from examples through stochastic local search. In *Inductive Logic Programming*, pages 200–210, 2007.
15. A. Paschke. ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. Technical Report 11, Technische Universität München, 2005.
16. A. Paschke and M. Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1):187–205, 2008.
17. A. Paschke, A. Kozlenkov, and H. Boley. A homogeneous reaction rule language for complex event processing. In *Proceedings of International Workshop on Event-driven Architecture, Processing and Systems*, 2007.
18. O. Ray. *Hybrid abductive inductive learning*. PhD thesis, Department of Computing, Imperial College London, UK, 2005.
19. O. Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009.
20. O. Ray, K. Broda, and A. Russo. Hybrid abductive inductive learning: A generalisation of Progol. In T. Horvath and A. Yamamoto, editors, *International Conference on Inductive Logic Programming*, pages 311–328. Springer, October 2003.

21. O. Ray, K. Broda, and A. Russo. Generalised kernel sets for inverse entailment. In *ICLP*, volume 3132 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 2004.
22. O. Ray, K. Broda, and A. Russo. A hybrid abductive inductive proof procedure. *Logic Journal of the IGPL*, 12(5):371–397, 2004.
23. O. Ray and A. Kakas. Prologica: a practical system for abductive logic programming. In J. Dix and A. Hunter, editors, *International Workshop on Non-monotonic Reasoning*, pages 304–312, May 2006.
24. B. Richards and R. Mooney. Automated refinement of first-order horn-clause domain theories. *Machine Learning*, 19(2):95–131, 1995.
25. M. Shanahan. The event calculus explained. In M. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today*, LNAI 1600, pages 409–430. Springer, 1999.
26. A. Srinivasan. The aleph manual. Available at: `http://web.comlab.ox.ac.uk/activities/machinelearning/Aleph/`.
27. M. Stickel. A prolog technology theorem prover: A new exposition and implementation in prolog. *Theoretical Computer Science*, 104(1):109–128, 1992.
28. S. Wrobel. First-order theory refinement. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, chapter First order theory refinement, pages 14–33. IOS Press, 1996.