Event Detection from Video Using Answer Set Programming

Abdullah Khan[1,2,3], Luciano Serafini[1], Loris Bozzato[1], and Beatrice Lazzerini[3]

[1] Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
[2] University of Florence, Via di Santa Marta, 3, 50139 Firenze, Italy
[3] University of Pisa, Largo L. Lazzerino 1, 56122 Pisa, Italy

{akhan,serafini,bozzato}@fbk.eu, b.lazzerini@iet.unipi.it

Abstract. Understanding of the visual world is not limited to recognizing individual object instances, but also extends to how those objects interact in the scene, which implies recognizing events happening in the scene. In this paper we present an approach for identifying complex events in videos, starting from object detection using a state-of-the-art object detector (YOLO), providing a set of candidate objects. We provide a logic based representation of events by using a realization of the Event Calculus that allows us to define complex events in terms of logical rules. Axioms of the calculus are encoded in a logic program under Answer Set semantics in order to reason and query over the extracted events. The applicability of the framework is demonstrated over the scenario of recognizing car parking on a handicap slot.

Keywords: Event detection in video, Event Calculus, Answer Set Programming

1 Introduction

The increase in availability of data in both structured (e.g. sensors) and unstructured (e.g. images, video, and audio) formats is a common trend nowadays, but information extraction for a meaningful use from this ocean of data is still a challenging task. The interpretation of these data need to be automated in order to be transformed into operational knowledge [3, 10]. Events are mostly important pieces of knowledge, as they represent activities of unique significance. Therefore, the recognition of events is of fundamental importance.

The goal of *event detection* from unstructured data formats (i.e. videos, images) is to identify and localize specified spatio-temporal patterns in videos, where each pattern represents a significant event. Understanding of events taking place in videos is a challenging problem for the scientific community due to factors such as, e.g. background clutter, pose, illumination and camera point of view variations. Complex video sequences [6] contain many activities and involve multiple interactions between objects. Determining which objects are relevant to a particular event type is the basic building block in understanding the dynamics

of the event. Hence, it is of fundamental importance not only to detect, but also keep a track of such candidate objects over the period of time.

Event recognition is considered to be the paragon of all computer vision tasks [2], because of its wide applications and involvement that they have in our daily life. Advances in deep convolutional neural networks in recent times have mostly focused on developing end-to-end black box architectures that achieve a high accuracy in recognizing events. However, the major drawback of such approaches is the interpretability of the model [17]. For complex events, humans can analyze the properties of complex actions and inject some semantic knowledge to extract semantically meaningful events. Whereas, *CNN (convolutional neural network)* based black box architectures often rely on high accuracy given the event is happening or not.

In this paper we take on the event recognition problem by aiming at bridging the gap between the effectiveness of deep learning and logical reasoning. We believe that building a hybrid solution exploiting end-to-end learning and logical reasoning is a good trade-off between accuracy and semantic richness of the event. To achieve this objective, we make use of the state-of-the-art object detector *YOLO (You only look once)* [19] for extracting basic information (appearance and movement) about objects from video streams. Events are then represented inside the logical framework of the Event Calculus [11], which allows for the definition of complex events: the calculus and events representation are implemented as a logic program interpreted under Answer Set semantics in order to reason and ask queries about the represented scenario.

2 Problem Description

Problem Statement. The core focus of our work is to extract complex events from the scene starting from the simple facts that are detectable from the visual information in the frames. Complex events require a level of understanding that pushes beyond the number of objects present in the scene to detailed comprehension of interactions between actors and objects across different video frames.

Use-Case: Handicap parking occupancy. The deployment of sensors in parking lots to address the issue of automatic parking-lot detection is performed with great success, but results in high deployment cost. Recently, smart cameras have been used to detect the occupancy of the slots in real-time relying on CNN-based architectures to be executed on these cameras [4, 14]. But, most of these camera-based solutions cannot be generalized for different parking lots. Visual occupancy detection in parking lots essentially involves the detection of vehicles (car, bike, bus, etc.) and parking spaces. However, to the best of our knowledge, the detection of vacant parking space for people with special needs by considering visual information is still an open problem.
The experimental data at our disposal consists of an approximately 4 min. long video, composed of multiple sequences, where each sequence is approximately 12 to 15 seconds, depicting the event of interest under different viewing conditions

including camera movement, ego-motion, change in illumination, clutter, motion artifacts.

3 Related Work

Object detection in videos aims to detect objects belonging to a pre-defined class and localize them with bounding boxes in a given video stream [19]. Object detectors based on bounding boxes have seen a steady improvement over the years. One of the pioneer CNN-based object detector was R-CNN [9], which involved two-stage pipeline, one part of the system provides region proposals, then for each proposal CNN is used for classification. To reduce the computational cost Region of Interest Pooling is used in FAST R-CNN [8] leading to efficient results. Furthermore, the most recent object detectors [13, 19] combine the two tasks of region proposal and classification in one system. Single shot object detectors, YOLO, SSD (single shot multi-box detector) significantly improved the detection efficiency compared to prior object detection systems.

Advances in deep convolutional neural networks in recent times have mostly focused on developing end-to-end black box architectures that achieve a high accuracy in recognizing events. Most of the work [7, 16, 18] in this area makes use of the recurrent neural networks which process the images or video frames one at a time, and incrementally combine information building up a dynamic internal representation of the scene. Such models try to capture the pose, movements that are the part of actions known as atomic actions and interactions (*e.g., walking, running, surfing, riding etc.*), but such methods are not very successful in capturing semantically meaningful representation of actions. Injecting semantic definition and structural knowledge in these approaches is rather difficult. Hence, it is of great importance for the model to be interpretable, and this is a part where neural networks fall short. To make up for this, we exploit a classic logic-based event recognition framework, like *Event Calculus* for complex event representation based on logical rules and answer set programs to reason and query over the events. The role of Answer set programming in conjunction with computer vision to formalize rules for visual scene understanding and answer queries about the event occurring in the scene is very recent [1, 20, 22].

4 Proposed Architecture

Figure 1 describes the workflow for our proposed architecture. Our method follows two phases: (1) objects are detected and tracked from every single frame using YOLO, providing simple events such as appearance of the object, disappearance of the object; (2) based on those candidate objects, events are represented in the logical framework of the *Event Calculus*. Reasoning on complex events is obtained by an encoding in logic programs under *Answer set* semantics (in particular, programs are run using DLV [12]). In the following sections, we detail the realization of the two steps and their results in the application to our experimental scenario.

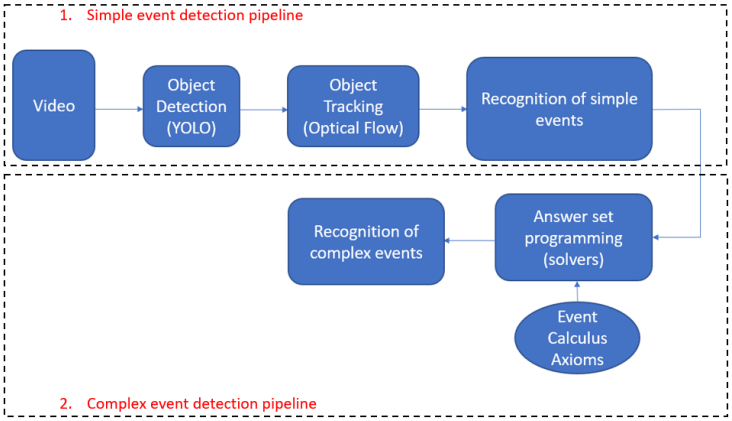


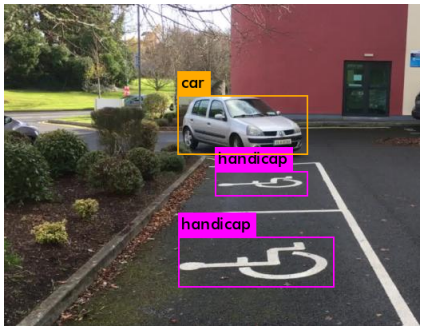
Fig. 1. Block diagram of the proposed architecture

5 Objects and simple events extraction from video

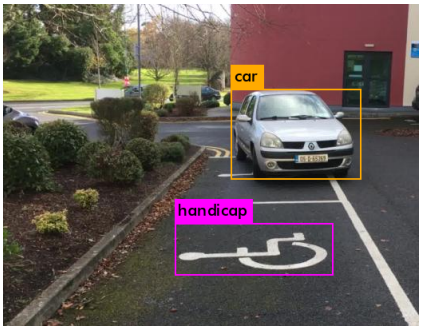
Given a video as input, the task of object detector is: (1) determine whether an object exists or not in the image/video, (2) determine the location of the object by putting a bounding box around it. Most of the methods previously used for object detection have one thing in common: they have one part of their system dedicated to providing region proposals which includes re-sampling of pixels and features for each bounding box, followed by a classifier to classify those proposals. These methods are useful but are computationally expensive resulting in a low frame rate. Another simpler way of doing object detection is by using the YOLO system, which combines the two tasks of region proposal and classification in one system. The key idea behind YOLO is the use of small convolutional filters applied to feature maps of bounding boxes to predict the category scores, using separate predictors for different aspect ratios to perform detection on multiple scales. YOLO uses *optical flow* method from OpenCV[4] to track objects by determining the pattern of motion of objects for two consecutive frames, which occurs due to the movement of the objects, helping in image segmentation and tracking. It works on the following assumptions. (1) Pixels grouped with similar motion, result in blob of pixels for all objects having different motion. (2) Intensities of pixels do not change between consecutive frames. (3) Neighbouring pixels have similar motion.

We applied this method on our sample video data of multiple short video clips, where each clip is 10 to 12 seconds: Figure 2 and Figure 3 show detection and tracking results of a car moving towards the handicap slot and a car parked at the handicap slot. As in common practice, we evaluate the performance of our model for object detection task by assessing the *Average Precision (AP)* of each class. Average precision (AP) is equal to the area under Precision-Recall curve, where Precision and Recall are calculated for every possible confidence threshold

[4] see [5], `https://opencv.org/` and `https://github.com/AlexeyAB/darknet`

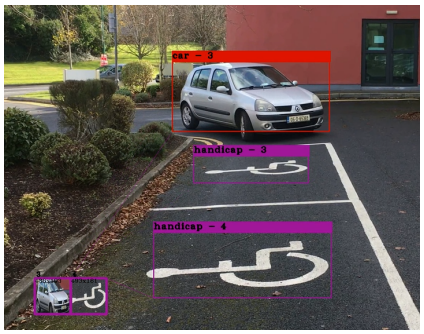


Car moving to the handicap parking slot

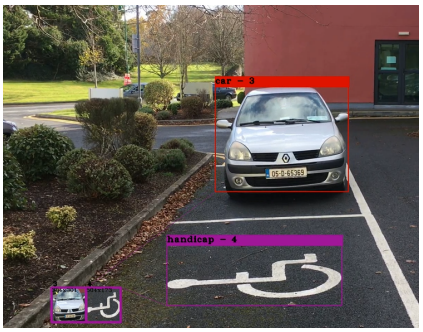


Car parked at the handicap slot

Fig. 2. Object detection using YOLO



Car3 moving to the handicap parking slot3



Car3 parked at handicap parking slot3

Fig. 3. Object tracking using YOLO

(for confidence of each detection), and for IoU_thresholds. Table 1 shows the average precision of the objects at different *Intersection over union (IoU)* threshold values. IoU measures how much overlap exists between the ground truth and actual prediction: this measures how good is our prediction in the object detector with the ground truth (the real object boundary). Results, depicted in Table 1, show the object detector detects objects with high accuracy.

Table 1. Average precision at different IoU thresholds. AP(0.75) means the AP with IoU=0.75.

Object	**AP(0.25)**	**AP(0.50)**	**AP(0.75)**
Handicap slot	99.01	90.55	77.04
Car	90.76	90.76	88.22

Table 2. Event Calculus predicates

Basic Predicates	Description
$holdsAt(f,t)$	fluent f is true at time-point t
$happens(e,t)$	event e occurs at time-point t
$initiates(e,f,t)$	if event e occurs at time-point t, then fluent f will be true after t.
$terminates(e,f,t)$	if event e occurs at time-point t, then fluent f will be false after t

6 Logical reasoning on complex events

In this section we review the definition of Event Calculus as presented in [21]; then, we present the encoding of its axioms as a logic program and we show how we can use it to reason over the events extracted from video in our scenario.

Event Calculus. Event Calculus (EC) was first introduced by Kowalski and Sergot in [11] as a logic framework for representing and reasoning about events and their effects. EC has been frequently used for event recognition as it provides a set of rich axioms for capturing the behavior of events and their effects. The EC language consists of (ordered) time-points, events and fluents. A *fluent* is a property whose truth value may change over time, such as the location of a physical object. The expressions referring to temporal entities that occur over some time interval are called *events*. After an event occurs, it may change the truth value of a fluent. It is assumed that the value of a fluent is preserved in successive time points, if no event changes its state. In particular, an event can *initiate* a fluent, meaning that the fluent is true after the happening of the event, or *terminate* a fluent, meaning that the occurrence of the event makes the fluent false.

The calculus makes use of the predicates listed in Table 2. The language provides predicates expressing the various states of an event occurrence: *happens* defines the occurrence of an event at a given time point, while *holdsAt* states that a fluent holds in a point in time. The predicates *initiates* and *terminates* specify under which circumstances a fluent is initiated or terminated by an event at a specific time point.

Event Calculus in ASP. An implementation of the Event Calculus into ASP is provided in [15]. The EC axioms determining the relation across fluents and

events are defined by the rules that follow.[5]

$$initiated(F,T) \leftarrow happens(E,T), initiates(E,F,T). \quad (1)$$

$$terminated(F,T) \leftarrow happens(E,T), terminates(E,F,T). \quad (2)$$

$$holdsAt(F,T_1) \leftarrow holdsAt(F,T), \texttt{not}\ terminated(F,T), time(T), T_1 = T+1. \quad (3)$$

$$\leftarrow holdsAt(F,T_1), \texttt{not}\ holdsAt(F,T), \quad (4)$$
$$\texttt{not}\ initiated(F,T), time(T), T_1 = T+1.$$

$$holdsAt(F,T_1) \leftarrow happens(E,T), initiates(E,F,T), time(T), T_1 = T+1. \quad (5)$$

$$\leftarrow holdsAt(F,T_1), happens(E,T), terminates(E,F,T), \quad (6)$$
$$time(T), T_1 = T+1.$$

Axiom (1) and (2) state that a fluent is *initiated* with the occurrence of an event that *initiates* it, and that fluent will be *terminated* when another event occurs and *terminates* it. Axiom (3) states that if a fluent holds at time-point T and is not terminated in T, then the fluent is true at the next time-point T_1. Axiom (5) states that if a fluent is initiated by some event that occurs at time-point T, then the fluent is true at T_1. Constraint in Axioms (4) state that it can not be that a fluent F that is not initiated nor true at time T becomes true at time $T+1$. Similarly, constraint in axiom (6) state that it can not be that fluent F holds at time $T+1$ if an event happened at time T that terminated F.

Event reasoning on example scenario. We are now ready to express our example scenario in terms of the presented ASP encoding of the Event Calculus. For explaining perceived dynamics of objects in the scene, we define the simple

Table 3. Description of simple and complex events

Simple event	**Description**
$appearsCar(A,T)$	The object corresponding to car A enters the scene at time T
$disappearsCar(A,T)$	The object corresponding to car A leaves the scene at time T
$appearsSlot(L,T)$	The object corresponding to parking slot L appears in the scene at time T
$disappearsSlot(L,T)$	The object corresponding to parking slot L disappears from the scene at time T
Complex event	**Description**
$covers(A,L,T)$	The object car A covers the slot L at time T
$uncovers(A,L,T)$	The object car A uncovers the slot L at time T

and complex events listed in Table 3. The focus is on explaining what is visible in the frames by identifying appearance and disappearance of objects in the

[5] We use the DLV syntax[12] of rules (in particular, for the use of functors and for expressing number operations in rules).

scene: thus, the fluents of our scenario are *visibleCar* and *visibleSlot*, that are true respectively if a car or a slot are currently visible in the scene.[6] Table 3 provides the description of simple events. The occurrences of these events are directly extracted from the output of the tracker: in other words, they will be compiled as facts in the final program. Given this information, complex events are then defined by combining simple events and conditions on fluents: in our example, we can detect when a car *covers* and *uncovers* a parking slot using the information about what is visible at a given time-point. Assuming the previous rules defining the EC axioms, we encode our scenario in a logic program with the rules that follow. We first declare objects, events and fluents of the scenario:

$$
\begin{aligned}
event(appearsCar(A)) &\leftarrow agent(A).\\
event(disappearsCar(A)) &\leftarrow agent(A).\\
event(appearsSlot(L)) &\leftarrow location(L).\\
event(disappearsSlot(L)) &\leftarrow location(L).\\
fluent(visibleCar(A)) &\leftarrow agent(A).\\
fluent(visibleSlot(L)) &\leftarrow location(L).
\end{aligned}
$$

We can then specify the effects of events on fluents:

$$
\begin{aligned}
initiates(appearsCar(A), visibleCar(A), T) &\leftarrow agent(A), time(T).\\
terminates(disappearsCar(A), visibleCar(A), T) &\leftarrow agent(A), time(T).\\
initiates(appearsSlot(L), visibleSlot(L), T) &\leftarrow location(L), time(T).\\
terminates(disappearsSlot(L), visibleSlot(L), T) &\leftarrow location(L), time(T).
\end{aligned}
$$

Basically, the rules define that the appearance of an object (car or slot) initiates its visibility, while its disappearance from the scene terminates the validity of the visibility fluent. Occurrences of complex events are derived from event calculus reasoning:

$$
\begin{aligned}
happens(covers(A, L), T) \leftarrow\ & agent(A), location(L), time(T),\\
& happens(disappearsSlot(L), T),\\
& holdsAt(visibleCar(A), T).\\
happens(uncovers(A, L), T) \leftarrow\ & agent(A), location(L), time(T),\\
& happens(appearsSlot(L), T),\\
& holdsAt(visibleCar(A), T).
\end{aligned}
$$

By these rules, we recognize that a car *covers* a slot if the car is visible at the time that the slot disappears. Similarly, the *uncovers* event occurs when a slot appears and the car is still visible. By combining the information on complex events, we can define that a *parking* from time T_1 to time T_2 is detected whenever a car *covers* a slot at time T_1, *uncovers* the slot at time T_2 and it stands on the slot for at least a number of frames defined by *parkingframes*:

$$
\begin{aligned}
parking(A, L, T1, T2) \leftarrow\ & happens(covers(A, L), T1), happens(uncovers(A, L), T2),\\
& parkingframes(N), T3 = T1 + N, T2 >= T3.
\end{aligned}
$$

[6] We are currently assuming a simple scenario with one car and one slot in the scene.

In our scenario, a query on *parking* can be used to obtain the parking events detected in the scenes and their information.

The final program, enconding the scenario, is obtained by combining these rules (together with the EC axiom rules) with the facts obtained from the tracker output. Let us consider an example instantiation:

$$holdsAt(visibleSlot(hp\_slot), 0).$$
$$happens(appearsCar(car), 1).$$
$$happens(disappearsSlot(hp\_slot), 2).$$
$$happens(appearsSlot(hp\_slot), 4).$$
$$happens(disappearsCar(car), 5).$$

According to the input evidence, initially only one slot *hp_slot* is visible. Then, object *car* appears and object *hp_slot* disappears from the scene at time-points 1 and 2, respectively. Whereas, at time-points 4 and 5, appearance and disappearance of the *hp_slot* and *car* occur. Using the rules, we can thus derive the occurrence of complex events $happens(covers(car, hp\_slot), 2)$ and $happens(uncovers(car, hp\_slot), 4)$. Say that we define $parkingframes(1)$, then we can detect the parking $parking(car, hp\_slot, 2, 4)$, meaning that *car* parks on *hp_slot* at time-point 2 and leaves the slot at time-point 4.

Over our sample video data, we run the program on DLV using the output of the tracker from previous step. We were able to detect complex events for some of the video sequences (e.g. *car 3* covers the *handicap slot 3* at time-point 87 and uncovers the slot at time-point 107). Unfortunately, we could not apply the method to the whole video: the reason stands in the ambiguities of tracker output (e.g. multiple labelling of the same object, incorrect disappearence of objects) which produce unclean data. A solution to this problem would be to include a pre-processing step for data cleaning (possibly encoded as logical constraints) which is able to resolve such ambiguities.

7 Conclusion and Future Work

We proposed a hybrid architecture for visual explanations encompassing logical reasoning with video data for *Handicap parking occupancy* use-case. The overall goal of this work is the integration of knowledge representation and computer vision: (1) Visual processing pipeline for detection based object tracking, leading to the extraction of simple events; (2) Answer set programming based reasoning to derive complex events.

The limitations of the object tracker result in noisy data which restricts us to reason on events for the whole video. As a future work, we aim to manage these inaccuracies by a (possibly logical based) data cleaning step. We also want to apply and evaluate the presented method in different scenarios (e.g. sports events [10]).

References

1. Aditya, S., Yang, Y., Baral, C., Fermuller, C., Aloimonos, Y.: Visual commonsense for scene understanding using perception, semantic parsing and reasoning. In: 2015 AAAI Spring Symposium Series (2015)
2. Ahmad, K., Mekhalfi, M.L., Conci, N., Melgani, F., Natale, F.D.: Ensemble of deep models for event recognition. ACM TOMM **14**(2), 51 (2018)
3. Akbar, A., Khan, A., Carrez, F., Moessner, K.: Predictive analytics for complex iot data streams. IEEE Internet of Things (2017)
4. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Vairo, C.: Car parking occupancy detection using smart camera networks and deep learning. In: IEEE ISCC 2016. pp. 1212–1217. IEEE (2016)
5. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
6. Budvytis, I., Badrinarayanan, V., Cipolla, R.: Label propagation in complex video sequences using semi-supervised learning. In: BMVC 2010. pp. 27.1–12 (2010)
7. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: IEEE CVPR 2015. pp. 2625–2634 (2015)
8. Girshick, R.: Fast R-CNN. In: IEEE ICCV 2015. pp. 1440–1448 (2015)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE CVPR 2014. pp. 580–587 (2014)
10. Khan, A., Lazzerini, B., Calabrese, G., Serafini, L.: Soccer event detection. In: IPPR 2018. pp. 119–129 (2018)
11. Kowalski, R.A., Sergot, M.J.: A logic-based calculus of events. New Generation Comput. **4**(1), 67–95 (1986). https://doi.org/10.1007/BF03037383
12. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. CoRR **cs.AI/0211004** (2002)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: ECCV 2016. pp. 21–37. Springer (2016)
14. Mauro, D.D., Furnari, A., Patanè, G., Battiato, S., Farinella, G.M.: A comparison of techniques based on image classification and object detection to count cars and non-empty stalls in parking spaces. In: ICETE (1). pp. 494–502. SciTePress (2018)
15. Mueller, E.T.: Commonsense reasoning: an event calculus based approach. Morgan Kaufmann (2014)
16. Nilsson, D., Sminchisescu, C.: Semantic video segmentation by gated recurrent flow propagation. In: IEEE CVPR 2018. pp. 6819–6828 (2018)
17. Prapas, I., Paliouras, G., Artikis, A., Baskiotis, N.: Towards human activity reasoning with computational logic and deep learning. In: SETN 2018. p. 27. ACM (2018)
18. Ramanathan, V., Huang, J., Abu-El-Haija, S., Gorban, A., Murphy, K., Fei-Fei, L.: Detecting events and key actors in multi-person videos. In: IEEE CVPR 2016. pp. 3043–3053 (2016)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE CVPR 2016. pp. 779–788 (2016)
20. Sampat, S., Lee, J.: A model-based approach to visual reasoning on cnlvr dataset. In: Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (2018)

21. Skarlatidis, A., Paliouras, G., Artikis, A., Vouros, G.A.: Probabilistic event calculus for event recognition. ACM TOCL **16**(2), 11 (2015)
22. Suchan, J., Bhatt, M., Wałega, P., Schultz, C.: Visual explanation by high-level abduction: on answer-set programming driven reasoning about moving objects. In: AAAI 2018 (2018)