Journal of Artificial Intelligence Research 53 (2015) 41-90 Submitted 03/14; published 05/15

Learning Relational Event Models from Video

Krishna S. R. Dubba KRISHNA.DUBBA@GMAIL.COM
Anthony G. Cohn A.G.COHN@LEEDS.AC.UK
David C. Hogg D.C.HOGG@LEEDS.AC.UK
School of Computing, University of Leeds,
Leeds, UK. LS2 9JT

Mehul Bhatt BHATT@INFORMATIK.UNI-BREMEN.DE
Frank Dylla DYLLA@INFORMATIK.UNI-BREMEN.DE
Cognitive Systems, SFB/TR 8 Spatial Cognition
University of Bremen, Bremen 28334, Germany

Abstract

Event models obtained automatically from video can be used in applications ranging from abnormal event detection to content based video retrieval. When multiple agents are involved in the events, characterizing events naturally suggests encoding interactions as relations. Learning event models from this kind of relational spatio-temporal data using relational learning techniques such as Inductive Logic Programming (ILP) hold promise, but have not been successfully applied to very large datasets which result from video data. In this paper, we present a novel framework REMIND (**R**elational **E**vent **M**odel **IND**uction) for supervised relational learning of event models from large video datasets using ILP. Efficiency is achieved through the *learning from interpretations* setting and using a typing system that exploits the type hierarchy of objects in a domain. The use of types also helps prevent over generalization. Furthermore, we also present a type-refining operator and prove that it is optimal. The learned models can be used for recognizing events from previously unseen videos. We also present an extension to the framework by integrating an abduction step that improves the learning performance when there is noise in the input data. The experimental results on several hours of video data from two challenging real world domains (an *airport* domain and a physical action *verbs* domain) suggest that the techniques are suitable to real world scenarios.

1. Introduction

With the advent of digital technology and wide availability of cameras and video recorders, the quantity of video data has increased enormously over recent years, e.g., YouTube users upload about 100 hours of video to the site every minute ("YouTube", 2015). This data is semantically rich but there is a lack of algorithms to process and utilize this data effectively. There are a number of applications that demand video processing, especially event modelling and recognition, such as content based video search, robotics, automatic description of activities, video surveillance etc. The main objective of our work is *to provide a supervised relational learning framework to learn high level human understandable event models and use them to recognize events in video.* Supervised learning is the machine learning task of inferring a model from labelled training data.

Video is considered as a sequence of images and the area of video analysis poses several challenges. The most interesting aspect of video when compared to images is that objects (or parts of objects) in video can be perceived to move in space over time. These changes of *state* in the space dimension are interesting and we call them *events* if they satisfy certain properties such as being sufficiently frequent and having sufficiently well defined boundaries etc. An *event* can be a change of *state* of a single object, such as moving some parts of its body (for example people waving their hands) or it can be an interaction between multiple objects. By *interactions*, in this context, we mean the movement of all the objects relative to their surroundings as well as relative to each other. For example, an interaction between two objects might be both objects moving towards each other or one at rest and the other moving away from it. Before events can be recognized, we assume that the objects involved have been detected and tracked from the source video. This is not a requirement in general since some approaches (Laptev, 2005) do not require detection of objects prior to event detection.

In events involving multiple objects, interactions between the objects become a distinguishing factor in recognizing the event instance. Capturing these interactions is the crux of event modelling and recognition as it is our hypothesis that each event can be distinguished by the interactions between the objects involved. Some events might have more than one interaction pattern that identifies the event. One way to capture these interactions is to abstract the interactions into relations between the objects. In order to represent the interactions of objects in an abstract form, we can use relations between the objects that depend on the spatial configuration or motion pattern of objects over a period of time. We call these *spatio-temporal* relations and in this paper we focus purely on the use of *qualitative* spatial relations since these abstract away from the metric details of particular object trajectories and thus facilitate the recognition of interactions as being instances of some event class (Cohn et al., 2006). There is no unique way to represent interactions using qualitative spatio-temporal relations, and the best set of relations to use depends on the domain, kind of data available (speed, orientation, size of moving objects, etc.) and the objectives of the task.

Though each event class has distinguishing interaction patterns, there are two particular challenges in event learning from examples expressed as qualitative spatio-temporal relations. Firstly, automatic object detection and tracking from video is not perfect and will introduce errors into the relations. Secondly, the same event may be performed in different ways.

1.1 Overview of the Framework

We follow the relational learning approach to the cognitive vision task of learning event models from videos and using them for recognition (Cohn et al., 2006; Dubba, Cohn, & Hogg, 2010). The video data (sequence of images with pixel data) is converted to relational facts involving qualitative spatio-temporal relations using the tracking data of the objects involved in the scenes. We use several qualitative spatial calculi to represent the video data in relational form. Event instances are annotated temporally and spatially though the objects involved in each event are not delineated separately and these annotations are used for obtaining the positive and negative examples of events. The learning procedure as well as the extension of this procedure using abduction (explained in later sections) is applied on this relational data to obtain the event models. These event models are in the form of Prolog rules that can be used as queries in the relational data from an unseen video. From the answer substitutions we extract the spatial and temporal extensions of recognized event instances.

The main contributions[1] of the paper are:

- a novel supervised relational learning framework REMIND for learning event models from video and recognizing event instances using these models.
- an optimal *Type Refinement* operator for upward refinement of hypotheses that exploits a type hierarchy in a domain for finding better event models.
- an extended framework to integrate induction and abduction in an interleaved fashion with an embedded spatial theory for improving the learning of event models.
- an evaluation of the framework on two real world video data sets (aircraft turn-arounds where the events include aircraft arrival, luggage loading and human interactions where the events are common action verbs such as exchange, follow, dig etc).

Though we concentrate on relational data obtained from tracking objects from video, the principles and techniques in this work equally apply to spatio-temporal relational data acquired from non-visual sources (e.g. laser mapping, GPS tracks, textual descriptions etc).

2. Related Work

Much of the work in event analysis (Ivanov & Bobick, 2000; Medioni, Cohen, Brémond, Hongeng, & Nevatia, 2001; Vu, Bremond, & Thonnat, 2003; Albanese, Moscato, Picariello, Subrahmanian, & Udrea, 2007; Ryoo & Aggarwal, 2009, 2011; Morariu & Davis, 2011), does not involve learning of the models used. Instead high level event models are hand-coded using different representations (Nevatia, Hobbs, & Bolles, 2004; Hakeem, Sheikh, & Shah, 2004).

Techniques that are based on a similarity based metric in a space of low level pixel based features such as local space-time features (Laptev, 2005) are frequently used for modelling and recognizing events. These are generally more suitable for single agent events like human activities based on motion. These kind of activities generally include a particular motion signature with which an event can be recognized such as *running, jumping, waving hands* etc. In some event recognition systems, hand-coded high level event models are used on top of the learned low level human activity models (Ivanov & Bobick, 2000; Ryoo & Aggarwal, 2009, 2011).

One of the best performances to date in event recognition using low level pixel-based features is obtained by the Stack convolutional Independent Subspace Analysis (ScISA) (Le, Zou, Yeung, & Ng, 2011) algorithm. ScISA is based on pixel level flow based features which are then used to model events using a hierarchical representation using deep learning techniques (Bengio, 2009). The authors present an extension of Independent Subspace

1. This paper is an extended version of the work by Dubba et al. (2010, 2012).

Analysis to learn invariant spatio-temporal features in an unsupervised fashion instead of using predefined features.

If events are considered as a sequence of primitive states or events, state-space models are useful in representing the event models. It is also easy to hand-code the structure of the state space models, though the parameters are better if learned than encoded by hand. They provide a more robust statistical event model than hand-coded models and event recognition is done using inference on these models. Bayesian Networks are not very popular in event modelling as they lack the temporal aspect though other state space models such as Hidden Markov Models (HMM) (Rabiner, 1989) and Dynamic Bayesian Networks (DBN) (Ghahramani, 1998) are extensively used in event modelling and recognition. A simple HMM is not very effective in modelling complex events. Several extensions of HMM are used to suit the context and the type of event models. Hoogs and Perera (2008) proposed a DBN for jointly solving event recognition and broken tracks linking problems. The event model is a set of discrete states which expresses how the actors in the event interact over time. They assume the states are strictly ordered and this may limit learning some events that involve complex temporal relations such as *during*, *overlaps* etc.

The main problem with state space models is that it is difficult to encode high-level temporal relations such as *during, overlaps* etc. The states or sub-events in an event are assumed to be in a sequential order which is not the case in many domains. Also the states are propositional in nature and hence are semantically less complex than a relational representation.

Veeraraghavan et al. (2007) learn Stochastic Context Free Grammar based models from traffic videos using predefined regions in the image. Each event model is a spatio-temporal pattern of primitive actions expressed as a string, $S = a_1, a_2, \dots, a_n$. The event learning algorithm aims to find a grammar that can generate the corresponding pattern for an event. The primitive actions are sequentially arranged, hence Allen's temporal relations are not used to connect the primitive actions. Gupta et al. (2009) claim that the fixed structure of the DBNs poses serious limitations for modelling events if there are many variations in the way an event can happen. Instead they use AND-OR graphs for modelling event models. The order of the nodes imposes the causal relationship among the nodes. Because of this, some Allen relationships such as *during*, *overlaps* etc. cannot be modelled which limits its application since modelling these relations is very important in many domains.

Though low-level features and state space models are popular for simple motion patterns, it is possible to build high-level event recognition systems through several layers of reasoning. These systems use simple pattern recognition techniques to detect primitive events and then use a temporal structure to reason about complex events. The main motivation for using a high level temporal structure is that the low level features (like bag-of-features) discard most of the information regarding the relations between different entities in the data and thus makes it hard to recognize events involving complex interactions between multiple objects.

Moyle and Muggleton demonstrated using a simple blocks world that *domain specific axioms* can be learned from temporal observations using an ILP framework (Moyle & Muggleton, 1997). In work by Needham et al. (2005), the Progol system (Muggleton, 1995) was used to learn the protocols of table top games from real sensory data from a video camera and microphone. A key aspect of this work is a method for spatio-temporal attention applied to the sensor data from audio and video devices. This identifies subsets of the sensor data relating to discrete concepts. Symbolic description of the continuous data is obtained by clustering within continuous feature spaces from processed sensor data. The Progol ILP system is subsequently used to learn symbolic models of the temporal protocols present in the presence of noise and over-representation in the symbolic input data. The framework is based on time points and used only the *successor* temporal relation.

Könik and Laird (2006) proposed a *learning by observation* framework to learn an agent program that mimics a human expert's behaviour in domains such as games. The learned concepts are used to generate behaviour rather than classification. They applied ILP techniques on artificially created examples from expert behaviour traces and goal annotations. The relational data used is simple as each predicate is valid in a situation (an abstract time point) and hence concepts with sophisticated temporal relations such as Allen's interval algebra (Allen, 1983) that use intervals cannot be learned. This limits the real world applicability of the framework where there are different events occurring in parallel and hence requires using Allen's interval algebra to model them. The framework uses only positive examples and the negative examples are generated randomly in a controlled fashion.

Fern, Givan and Siskind (2002) introduced a system, leonard, that learns event definitions from videos by following a standard specific-to-general learning approach from only positive data. There are seven simple event types that are learned in this system namely *pick up, put down, stack, unstack, move, assemble* and *disassemble.* The relational data is obtained by tracking objects in indoor scenarios. No negative examples are supplied and the event models are found by computing the least-general covering formula (LGCF) of each positive example and then computing the least-general generalisation (LGG) of all these resulting formulae. When computing the LGCF of each example, the resulting LGCF will not have any interval information. Hence the model can only support *before* and *equal* temporal relations between states.

The important aspect to note for the above review is that most of the work in this area has been done on either artificial or simulated data (Moyle & Muggleton, 1997; Könik & Laird, 2006) or very simple real world data (Fern et al., 2002; Needham et al., 2005) that involves few objects, the events are of short duration and all the objects in the scene are involved in the events. In our case, the tracked data from videos is very large and at the same time more complex and noisy and contains more objects.

Several attempts were made in the literature for integrating induction and abduction for learning better theories. It was pointed out by Tammaddoni-Nezhad et al. (2006) that abduction and induction are integrated in general when two conditions hold: the background knowledge is incomplete and the hypothesis language is disjoint from the observation language. The setting in which the latter condition holds is called *non Observation Predicate Learning* (non-OPL) setting (i.e. in the OPL setting, the examples and hypotheses define the same predicate). They assume the existence of a theory that connects the hypothesis language and the observation language to start with. Since this theory is not learned, it can be considered as the background theory. The general strategy in this case is to abduce (Kakas & Riguzzi, 2000) the missing observations using background theory and use this abduced data for inducing new theories. Muggleton and Bryant (2000) proposed *Theory Completion using Inverse Entailment* (TCIE) in the non-OPL setting. TCIE abduces and adds facts called the *Start Set* that connect the target predicate with observable predicates to the observation data and generalizes this data. In our case, the missing facts are because of noise in the observed data and the set of target predicates is the same as the set of observables whereas in TCIE, it is because the target predicate is not observable and the set of target predicates and the set of observables are disjoint.

Moyle (2003) introduces an ILP system (alecto) that combines abduction and induction to learn theories for robot navigation. One limitation of this system is that it is restricted to positive observations only learning. The integration is not interleaved in nature as abduction is first used to generate explanations for each example and induction is applied on this set of explanations. This means the abduction phase does not take into consideration the concepts learned in the induction phase and dealing with noise in data was left as future work.

3. Relational Representation of Scenes in Video

To represent interactions of objects by relational data, we use spatial and temporal relations. Since the input in this work is from video, the spatial relations are defined either in the image plane or in the ground plane (if a homography is used to map the image plane to the ground plane). The spatial relations are necessary to encode the state a particular pair of objects is in. These states between two objects change as time progresses, hence we need temporal relations to connect these states. In this section, we explain how objects' interactions are converted to relational data.

Notation: We use a first-order typed language ($\mathcal{L}$) with the following alphabet: $\{\neg, \wedge, \vee, \forall, \exists, \supset, \equiv, \mathcal{R}\}$. Let $\mathcal{R} = \{r_1,\ r_2, \ldots, r_m\}$ denote a set of m qualitative spatial relationships in an arbitrary qualitative spatial calculus. There are sorts (and corresponding variables) as given below (upper case letter denotes a set and lower case letter denotes a set element):

time points	$\mathcal{T} — t$
time intervals	$\Delta — \delta$
spatial objects	$\mathcal{O} — o$
events	$\mathcal{E} — \varepsilon$
temporal relations	$\mathcal{A} — \alpha$
object types	$\chi — \tau$

The special event-predicate $tran(r_i,\ o_k,\ o_l,\ t_m) \in \mathcal{E}$ denotes a transition from a spatial relation r_i between objects o_k and o_l at time point t_m. Note that in this work, α can only take values from the set of 13 Allen's base relations (Allen, 1983) i.e. $\mathcal{A} = \{$*before, after, meets, met by, overlaps, overlapped by, during, contains, equals, finishes, finished by, starts, started by*$\}$. We say two intervals are *disjoint* if the Allen relation between them is from the set $\{$*before, after, meets, met by*$\}$.

3.1 Spatial Relations

In order to get a high level description of the interactions of objects in videos, we need relations that can encode the interactions of objects without loss of essential information (Cohn et al., 2006). There are several possibilities on what kind of relations we have to choose. Since the interactions of objects in video take place in spatial dimensions, it is natural to

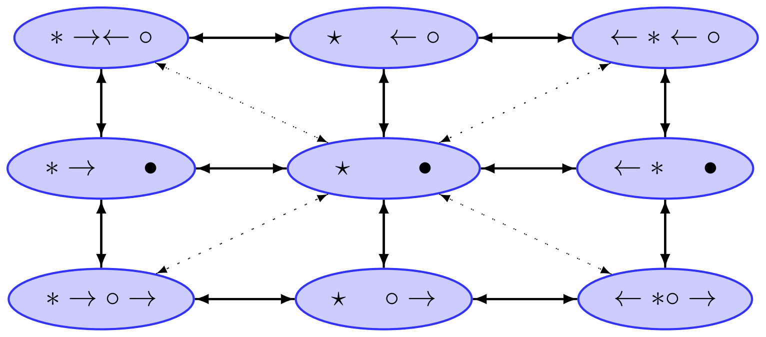


Figure 1: Qualitative Trajectory Calculus (QTC_{L1}) (Van de Weghe et al., 2006): Each blob is a possible QTC_{L1} spatial relation. In a blob, an asterix (left object) or circle (right object) represent objects in motion while a star and black dot represent objects at rest and the direction of the arrow shows the direction of motion of the object. For example, the top-left ellipse is interpreted as two objects moving towards each other and the bottom-left ellipse is interpreted as the right object is moving away from the left object while the left object is moving towards the right object (i.e the left object is chasing the right object). Though nine relations are possible in QTC_{L1} as shown in the figure, in practice we can reduce them to six exploiting symmetry in some relations. When only one object changes its motion state (note that an object cannot change direction without going through the *rest* state), the QTC relation changes along a thick line connecting two relations. When both objects change motion state instantaneously, the relation changes along a dotted line.

use qualitative spatial calculi to model the interactions. These interactions also have a temporal dimension as they occur over a period of time, so we extend the spatial relations with arguments modelling the temporal dimension. When we say interactions of objects we mean the interactions of the bounding boxes[2] (aligned to the axes) of these objects that we get from tracking the objects using computer vision algorithms (Yilmaz, Javed, & Shah, 2006). There are different kinds of spatial calculi that target different aspects of object interactions like topology, orientation, direction, trajectories etc. and which calculi to use is a domain dependent choice (Chen, Cohn, Liu, Wang, Ouyang, & Yu, 2015). We primarily use three spatial relations that encode the object interactions at the topological level: *dc* (*Discrete*) when the intersection of pixels in the bounding boxes of two objects is empty, *in* (*Inside*) when the intersection of pixels is the same as the pixels in the bounding box of one of the objects and *touch* in every other case. This set of simple topological relations is an abstracted version[3] of RCC-8 (Randell, Cui, & Cohn, 1992) spatial calculus, reduced for practical purposes without loss of essential information for event analysis. We also use QTC_{L1} (Van de Weghe et al., 2006) (Fig.1) and domain specific relations as primitives to represent the interactions of objects in the videos.

2. In principle, other shape abstractions could be used as well, e.g. convex hulls, silhouettes, bounding ovals etc.
3. The other two relations in this version of RCC called RCC-5 are *equal* and *contains* (inverse of *in*). The relation *equal* rarely occurs in our experiments and we do not use *contains* as we can convert it to *in* by reversing the arguments.

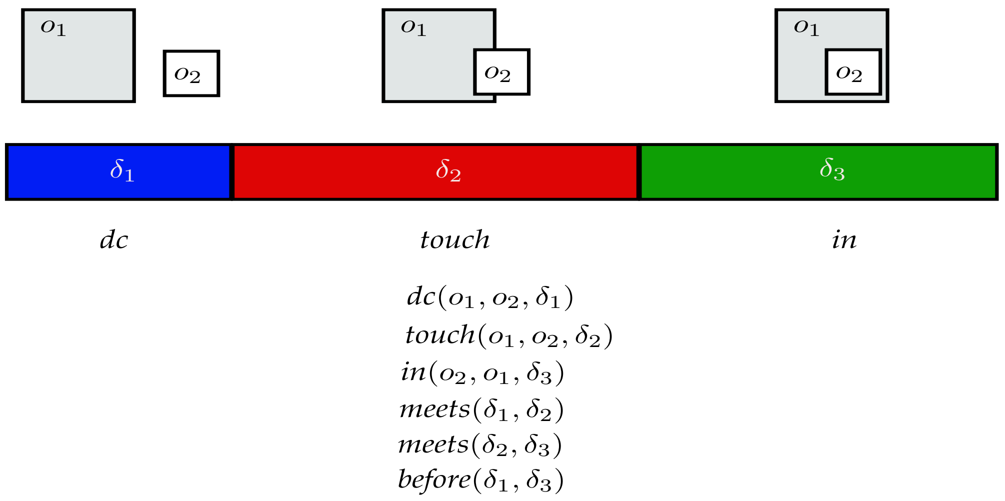

o_1
o_2
δ_1
δ_2
δ_3
dc
touch
in
$dc(o_1, o_2, \delta_1)$
$touch(o_1, o_2, \delta_2)$
$in(o_2, o_1, \delta_3)$
$meets(\delta_1, \delta_2)$
$meets(\delta_2, \delta_3)$
$before(\delta_1, \delta_3)$

Figure 2: Converting interactions of objects to relational data.

3.2 Temporal Interval Relations

We can define temporal relations between time intervals based on Allen's interval algebra. We use *start* and *end* frames of an interval to represent the intervals. An advantage of this approach is, we do not have to precalculate temporal relations and store them beforehand in the database for inference. Instead, Prolog rules that calculate temporal relations given *start* and *end* time points of two intervals are used. In order to incorporate temporal information in describing a scenario, we extend the spatial relations with a temporal interval as an extra argument.

3.2.1 Temporally Extending a Spatial Relation

The state where a spatial relation r between objects o_1 and o_2 holds throughout an interval δ is represented as $r(o_1, o_2, \delta)$ where $r \in \mathcal{R}$, $o_1, o_2 \in \mathcal{O}$ and $\delta \in \Delta$. Grounding this expression with objects and intervals from a database will provide us with *spatio-temporal facts*.

A temporal relation between two spatio-temporal facts is the Allen relation between the intervals in the spatio-temporal facts.

3.3 Representing an Event Class

An event class is represented by a set of Horn clauses where the head predicate is the same as the event name under consideration and the body is a non empty conjunction of atoms consisting of spatial and temporal predicates.

The structure of each clause in an event model for an event class ε is as follows:

$\varepsilon() \ :- \ \beta_1, \ldots, \beta_i, \ldots, \beta_n$

where each β_i is either of the form $r(o_1, o_2, \delta)$ where $r \in \mathcal{R}$, $o_1, o_2 \in \mathcal{O}$ and $\delta \in \Delta$, or is of the form $\alpha(\delta_1, \delta_2)$ where $\alpha \in \mathcal{A}$ and $\delta_1, \delta_2 \in \Delta$.

4. Deictic Supervision

For supervised learning, we need positive and preferably negative examples too of event instances. One major problem in supervised learning is collecting the labelled training data. Because of the general ambiguity in defining the spatial and in particular the temporal extent of an event (i.e. where do events precisely start and finish), it is difficult to annotate videos with event labels. A possible approach is to annotate the objects involved in each event and give the event's temporal extent. But annotating objects is tedious and prone to human error and for some events there may be uncertainty in the objects involved. We can avoid this by using *Deictic Supervision* (Dubba et al., 2010). Instead of annotating the exact objects involved in the training event instances, we only give a bounding spatial and temporal extent of each event instance which may contain other objects. The spatial extent is a region indicating where the event is happening in the video. The temporal extent is an interval which includes the actual temporal extent of the event, but may be deliberately longer in order to avoid accidentally truncating state changes relevant for the event. This makes preparation of training data easier and the learning process more robust and less biased to the labelling and the learning algorithm should be able to induce reasonable models even with this data.

Delineating spatio-temporal volumes in videos from which to learn feature-based representations of actions such as hand gestures is not without precedent in the computer vision literature (Laptev & Pérez, 2007), but our use here extends it to multiple simultaneous actors and relational descriptions and resilience to perturbations in the placement of cuboids provided events are fully enclosed.

4.1 Deictic Interval and Region

In this work, a deictic spatial region is a rectangle on the image plane indicating where the event happened and the deictic interval is a time interval indicating when the event happened. A deictic spatial region is obtained by hand-delimiting the event in the image plane with a rectangle[4], hence can be represented using a coordinate point (top-left corner vertex), height and width of the rectangle (x, y, h, w). A deictic temporal interval is provided by specifying the start and end time points of the interval. Together they define a space-time cuboid which delimits the spatial and temporal extension of an event.

A deictic cuboid defines a set of spatial facts and temporal relations between them; an event instance is a subset of these facts and corresponds to a *positive* example in the learning from interpretations setting. Obtaining positive and negative examples for learning using event annotations in the form of deictic spatial regions and deictic temporal intervals is explained in the following sections. Note that the deictic interval and region are regarded as any other interval and object in Δ and $\mathcal{O}$ respectively and the spatial relations are computed accordingly. After the positive and negative examples are computed, the spatial relations involving the deictic regions as one of the objects are discarded from the database as they are of no further use.

4. If the tracking data is on the ground plane then we can back-project this rectangle automatically into a minimum enclosing rectangle on the ground plane using homography (Hartley & Zisserman, 2004).

4.2 Herbrand Interpretation for an Event

Let s_i and δ_i be the deictic spatial region and deictic temporal interval for an instance ε_i of an event class ε in video v. Let Γ_v be the set of spatio-temporal facts present in v, $\mathcal{O}_v$ be the set of objects in v and Δ_v be the set of all time intervals in v. The set of facts $E_{\varepsilon_i} \subseteq \Gamma_v$ is the Herbrand Interpretation for the event ε_i over Γ_v iff all the facts contained in it are entailed by Γ_v, whose temporal intervals are not disjoint with the deictic interval and whose objects have relation *touch* or *in* within the deictic region.

$$\begin{aligned} E_{\varepsilon_i} = \{ & r(o_1, o_2, \delta) : \Gamma_v \vDash r(o_1, o_2, \delta) \wedge \\ & \Gamma_v \vDash \alpha(\delta_i, \delta) \text{ where } \alpha \notin \{\text{before,after,meets,metby}\} \wedge \\ & \exists \delta_1 [\Gamma_v \vDash r_1(s_i, o_1, \delta_1) \text{ where } r_1 \in \{\text{touch,in}\} \wedge \\ & \Gamma_v \vDash \alpha_1(\delta_i, \delta_1) \text{ where } \alpha_1 \notin \{\text{before,after,meets,metby}\}] \wedge \\ & \exists \delta_2 [\Gamma_v \vDash r_2(s_i, o_2, \delta_2) \text{ where } r_2 \in \{\text{touch,in}\} \wedge \\ & \Gamma_v \vDash \alpha_2(\delta_i, \delta_2) \text{ where } \alpha_2 \notin \{\text{before,after,meets,metby}\}] \wedge \\ & o_1, o_2, s_i \in \mathcal{O}_v \ \wedge r \in \mathcal{R} \ \wedge \ \delta_1, \delta_2 \in \Delta_v \\ \} & \end{aligned}$$

An example interpretation for an event instance of *AFT_Bulk_LoadUnload* in the Airport domain is illustrated in Fig.3. The interpretation includes all those spatial facts involving objects that have the relation *touch* or *in* with the deictic region and which lie within the two vertical dashed lines (the deictic interval). The set of Herbrand Interpretations corresponding to the set of deictic regions and intervals for an event form the positive examples for the learning phase. The rest of the relational facts in each video form the negative example where if an event model fires an instance in the database, it is considered as a false positive. When a Herbrand Interpretation is extracted from a set of spatio-temporal facts of a video, this interpretation is independent of all the other facts in the spatio-temporal database[5] for the video and hence other facts can be assumed false from this interpretation's point of view.

Note that by definition the spatio-temporal facts that do not spatio-temporally overlap the deictic region and interval of an event instance are not relevant to the event. Considering facts outside the indicated event occurrence not only increases the size of the training data but also makes the example instances for different event classes less distinct.

One limitation of using a cuboid shaped deictic region for delineating an event instance is that it is not possible to differentiate among multiple co-occurring instances of the same event type involving different objects in a region. One way to overcome this limitation is to use more than one cuboid to enclose an event instance allowing the elimination of unwanted facts.

5. This spatio-temporal database is itself a subset of the Herbrand Base of the video obtained using the predicates (spatio-temporal relations) and the constants (objects and time intervals) from the video.

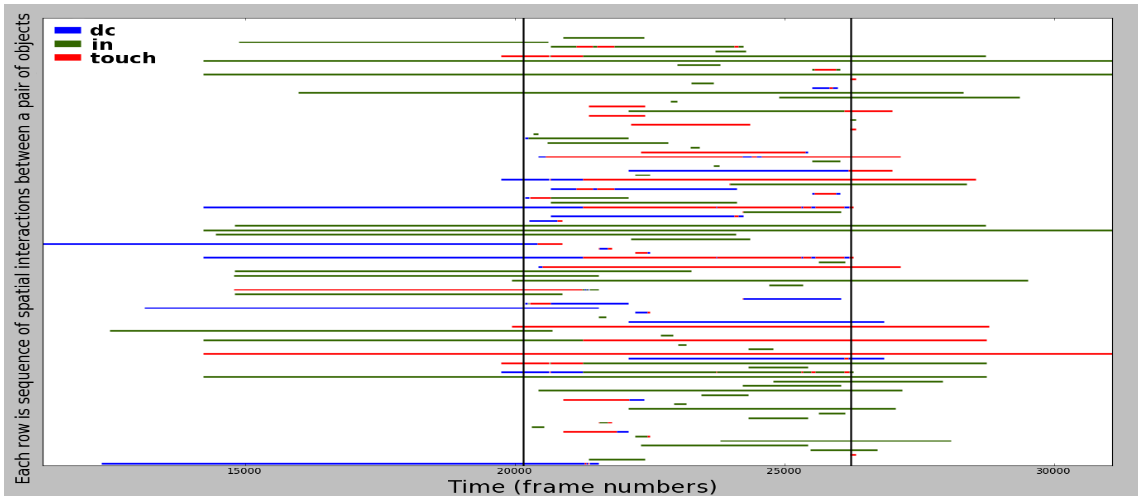


Figure 3: An example interpretation for the event *AFT_Bulk_LoadUnload* in the Airport domain. The vertical black lines are the start and end of the deictic interval. Each row represents the interactions of two of the objects present in the deictic region during the deictic interval in the video. The colours of the lines represent the spatial relations between the pairs of objects at that point of time. This figure does not show the effect of a deictic spatial region, but this would correspond to the elimination of certain rows (where the objects do not have a spatial relation of *touch* or *in* whilst in the deictic spatial region during the deictic temporal interval.

4.3 Herbrand Interpretation for a Non-event Interval (Negative Example)

In our framework, the negative examples are not explicitly labelled. The negative example for a given event in a video is the set of spatio-temporal facts from the database of the video that are not present in the positive examples of the event in that video. Note that the negative example will in general contain data that might be in positive examples of other event classes in that video. Another alternative is to use labelled positive examples of other events as negative examples for the event we are learning. This is convenient for classification purposes but not in recognition tasks as this will miss the background data that might be useful to minimize detections in the background regions.

Let Γ_{ε_v} be the union of all spatio-temporal facts from all Herbrand Interpretations of event ε in video v. The set of facts $\mathrm{NI}_{\varepsilon_v} \subseteq \Gamma_v$ is the Herbrand Interpretation for a negative example for event ε in video v iff it contains all the facts in Γ_v which are not in Γ_{ε_v}, i.e., $\mathrm{NI}_{\varepsilon_v} = \Gamma_v - \Gamma_{\varepsilon_v}$.

5. Typed ILP

In any event learning and recognition system, low level image processing and computer vision techniques may introduce noise into the system. One kind of noise, in particular when video quality is bad as in videos from some CCTVs, is that the wrong type may be assigned by the tracker to an object history. An object detector is typically trained with

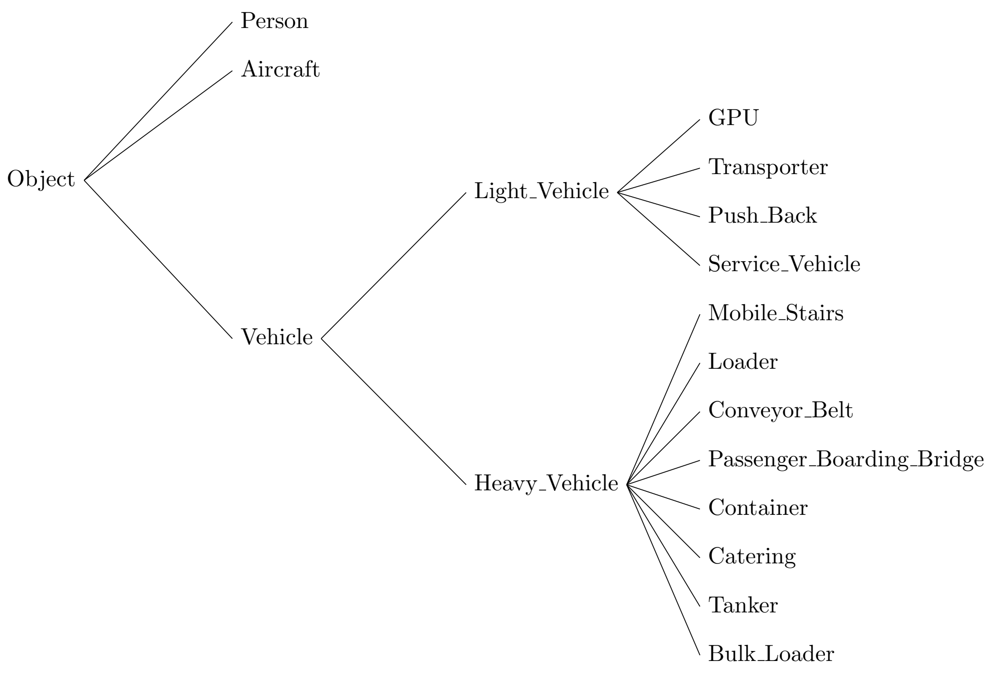


Figure 4: Tree-structured object type hierarchy in the Airport domain.

many example images of objects to be detected. Even though many example images are given for training, it is not possible to capture all the possible ways an object can appear because of lighting, viewing direction, size, shape, etc. (Lowe, 2004). This may result in correctly localized objects but with the wrong categories of the objects, especially those that look visually similar in low contrast images.

When the input data is huge and noisy, there are several problems an ILP system can face. One of these is that hypothesis evaluation can take a lot of time because of the size of the data. Also the noise will tend to make the hypothesis over specific as the system learns more rules to cover the inconsistent examples. Using a typed ILP system can speed up evaluation because of typed arguments in the hypothesis (Walther, 1985; Cohn, 1989) and also reduce the number of false positives through avoiding certain cases where the types of the arguments do not match. Any event model that has objects with a specialized type will fail to recognize some event instances where the object appears with a different type. In contrast, if the event model does not have any type system and uses a very generic type for all the objects such as *object*, *thing* etc., then this approach will have many false positives as it cannot differentiate between events with same structure but involving different types of objects. One possible approach is to find an appropriate type generalization instead of using one of the two extremes: most generic type and most specialized type.

In most ILP systems, the type hierarchy of objects is not integrated into the learning process. For example, in Progol, types of the objects are used only in mode declarations and since it assumes a flat type hierarchy of the domain, the search procedure cannot take any type hierarchy into consideration. For example, if the tracking system sometimes confuses two types of objects (τ_1, τ_2) such that some objects of type τ_1 are misclassified as

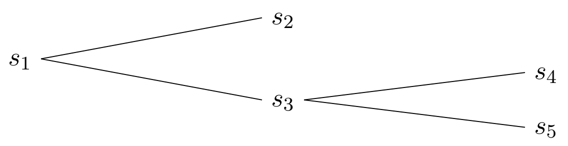


Figure 5: Tree-structured example object type hierarchy. s_1 is the most general type and s_2, s_4, s_5 are the most specific types.

type τ_2, then Progol generates two rules, one with τ_1 and another with τ_2. Even if we are not dealing with a vision system that introduces noise into the high level learning and reasoning system, in some cases an event might involve objects from a particular sub-group of objects. In this case, instead of using a very generic type like *object* or very particular types like the type of the object itself, it is more efficient to use an intermediate generic type that represents this sub-group. A variable without type restrictions will be satisfied by any type of object when instantiating the Horn clause. However, an appropriate generalization can be enforced by the learning system with a variable of type $\tau_1 \sqcup \tau_2$ from the type hierarchy, which is satisfied[6] only by objects of type τ_1 or τ_2, thereby reducing false positives.

5.1 Representing a Typed Hierarchy

If we wish to use an existing Prolog engine for hypothesis evaluation then some way of encoding type using terms must be found. There are several ways of doing this depending on whether the structure of the object type hierarchy is a tree or a lattice. We use the type representation proposed by Bundy, Byrd and Mellish (1985) that can deal with tree structured type hierarchies; we then develop a refinement operator by incorporating this representation in the hypothesis search procedure. An advantage of using this representation is that ordinary unification can be used to determine whether two types are compatible.

We will write $\tau_i \sqsubset \tau_j$, if τ_i is a subtype of τ_j and $\tau_i \neq \tau_j$. Every object o of type τ_n in a hypothesis can be represented by the term $\tau_1(\tau_2(\ldots\tau_n(o)\ldots))$ where $\tau_1, \ldots, \tau_n$ is the maximal sequence of types such that $\tau_n \sqsubset \ldots \sqsubset \tau_2 \sqsubset \tau_1$. We denote this representation function by Υ. Note that we need a constraint, i.e. a tree structured type hierarchy, in order to guarantee the uniqueness of the sequence $\tau_1, \ldots, \tau_n$.

For example, let s_1, s_2, s_3, s_4, s_5 be types such that $s_4 \sqsubset s_3 \sqsubset s_1$, $s_5 \sqsubset s_3 \sqsubset s_1$ and $s_2 \sqsubset s_1$ as shown in Fig.5. Then any object o of type s_4 can be represented as follows:

$$\Upsilon(o) = s_1(s_3(s_4(o)))$$

An object o_i is not compatible with an object o_j in a hypothesis if $\Upsilon(o_i)$ is not unifiable with $\Upsilon(o_j)$. For example: $s_1(s_3(o_1))$ will not unify with $s_1(s_2(o_2))$ but will unify with $s_1(s_3(s_4(o_3)))$ and $s_1(s_3(s_5(o_4)))$, hence they are compatible.

5.1.1 Example of Representing a Type Hierarchy

An object type hierarchy that occurs in one of the two domains used in the evaluation section of this work is shown in Fig.4. The hierarchy from Fig.4 is hand defined based

6. A variable of type $\tau_1 \sqcup \tau_2$ can unify with a term of type τ_1 or τ_2.

on observed errors in object classification of the tracking data from the Airport domain. In the airport domain, the ground power unit (GPU), transporter and push back vehicle are small vehicles that look similar as the videos from CCTV cameras in the airport have low resolution contrast without much colour or sharp edges. This makes it challenging to train an object detector and use it for detecting objects in these videos. The objects from the Verbs domain present no particular challenges from an automatic classification point of view but some events involve objects from a particular subset of objects, for example, the *throw* event involves balls of different types like *small ball, basket ball,* etc. Hence using a type hierarchy based on utility is expected to help find event models that have better performance in detecting events in unseen videos.

A vehicle V of type GPU will be represented as `obj(veh(light_veh(gpu(`V`))))`[7] while V of type `light_veh` is represented as `obj(veh(light_veh(`V`)))`. Note that `obj(veh(light_veh(`V`)))` unifies with vehicles of type GPU and vehicles of type Transporter. So using `obj(veh(light_veh(`V`)))` in a model can cover examples that either involve a GPU or a Transporter and hence can handle the noise from an object detector if it confuses these vehicles by outputting GPU in place of Transporter or vice versa.

5.2 Type Refinement Operator

A refinement operator is used to traverse through the hypothesis lattice. There are two types of refinement operators: *upward* and *downward* (Nienhuys-Cheng & De Wolf, 1997). We write $H_g \succeq H_s$ if H_g is a more generic[8] hypothesis than H_s. If we assume that the top most element of the hypothesis lattice is the most generic hypothesis and the bottom most hypothesis is the most specific hypothesis, then the *upward refinement operator* can be defined as follows (the *downward refinement operator* can be defined in a similar fashion):

Let Ł be the set of all possible hypotheses. An (upward) refinement operator ρ is defined such that for a hypothesis H, ρ produces only generalizations of $H, \rho(H) = \{H_g \mid H_g \succeq H, H_g \in \text{Ł}\}$.

We define the (upward) Type Refinement operator ρ_t as an operator that generalizes the object types of H. Apart from object types, the structure of H and members of $\rho_t(H)$ is identical.

We define a type generalizing operator as follows:

$$\text{generalize\_type}(\tau_1(\tau_2(\ldots\tau_{n-1}(\tau_n(o))\ldots))) = \tau_1(\tau_2(\ldots\tau_{n-1}(o)\ldots))$$

The *Type Refinement* operator, ρ_t, applies the *generalize_type* operator to a selected object type present in a hypothesis, resulting in a more generic hypothesis by moving up exactly one level in the type hierarchy.

Though the specific current representation of type hierarchy using functors requires a tree structured hierarchy, having a tree structured hierarchy is beneficial from a computational viewpoint in limiting the type generalizations, i.e., there are no multiple ancestors. A tree-like type hierarchy is very natural in many domains though some domains might not

7. Note the short forms used for Object, Vehicle and Light_Vehicle.
8. There are several possible generality orders, most important are subsumption and logical implication (Nienhuys-Cheng & De Wolf, 1997).

have a well defined tree-like object type hierarchy. In such cases, a lattice structured type hierarchy is more suitable though this will increase the size of the search space since the number of possible refinements is increased, in particular in a tree structure type generalization is deterministic whilst this is not the case in a lattice structure.

5.2.1 Optimality of the Type Refinement Operator

Refinement operators can be *ideal* or *optimal* (Nienhuys-Cheng & De Wolf, 1997)[9]. An *optimal* refinement operator generates any hypothesis in the hypothesis lattice only once and there is a unique way to produce each hypothesis. This kind of refinement operator is desirable in complete search algorithms as duplicate generation of hypotheses will increase the cost of the search procedure. The *optimality* of the *Type Refinement* operator is proved in Appendix A.

6. Learning from Interpretations Setting for Learning Event Models

The result of deictic supervision gives us examples that are sets of spatio-temporal facts. Though these examples (sets of facts) come from the same or different videos, they are independent of each other, i.e., the mapping of each example to a class is independent of other examples. For this kind of learning setting where each example is independent of each other and each example is a set of facts, the *learning from interpretations* setting is an apt choice (Blockeel, De Raedt, Jacobs, & Demoen, 1999). The setting can be specified formally thus:

Given:

- a set of classes C (each class label c is a nullary predicate).
- a set of classified examples E (each element of E is of the form (E_i, c) with E_i a set of facts and c a class label)
- and a background theory B,

Find: a hypothesis H (a set of Horn clauses), such that for all $(E_i, c) \in E$:

- $H \wedge E_i \wedge B \models c$, and
- $\forall c' \in C - \{c\} : H \wedge E_i \wedge B \not\models c'$

In the current event learning problem, the above setting is applied for each event class where in each case the set of classes has only two elements, the event class and the background class. Background class represents the negative examples and each class label c is a nullary predicate.

9. An *ideal* refinement operator is *proper* and *complete* whereas an *optimal* refinement operator is *weakly-complete* and *non-redundant*. See Appendix A for formal definitions.

6.1 Traversing the Search Space

The search process for a hypothesis starts with an initial hypothesis which has a nullary predicate as head and an empty body. The hypothesis lattice is traversed using the Progol and Type Refinement operators in an interleaved fashion. The Progol refinement operator is a specialization operator that adds atoms from the *bottom clause* to a hypothesis. A specialization operator moves from the top (empty clause) to bottom in the hypothesis space which is a lattice bounded by the *bottom clause* from the bottom. Adding atoms from the *bottom clause* makes the hypothesis more specialized because the body of the hypothesis is a conjunction of atoms and each atom can be considered as a constraint. Adding more atoms to the body increases the constraints it has to satisfy to become true.

The Progol refinement operator that we use here is based on the *bottom clause* also called the *most-specific clause* and is non-redundant though it is not weakly-complete with respect to the general subsumption order (Tamaddoni-Nezhad & Muggleton, 2009). The *most-specific clause* that the Progol refinement operator uses can be computed from training examples, mode declarations and the background knowledge (Muggleton, 1995). Mode declarations are user defined syntactic biases in the form of predicates that specify what predicates from the background knowledge are expected in the target hypothesis and also the nature of the variables (input, output, or constant). The selection of atoms to be added to the hypothesis from *bottom clause* is done in a controlled manner. The atoms are only considered starting from left and moving to the right and each atom can be added only once (Tamaddoni-Nezhad & Muggleton, 2009). These constraints on the selection of atoms makes the refinement process non-redundant, i.e., a hypothesis is not generated twice. There is an additional refinement operator that refines by unifying two variables arbitrarily selected from the hypothesis or by substituting a variable with a constant. We do not use this operator as unifying two variables needs checking the hypothesis for consistency with respect to the underlying spatial theory and there are no fixed constants (apart from frame numbers) in the domain as constants in any example are independent from constants from other examples. For example, consider the three relations from Section 4.1 for the spatial theory and Allen's relations for the temporal relations: we cannot unify two arguments of a predicate (spatial or temporal) as this violates the semantics of these relations.

The Type Refinement operator generalizes a hypothesis by generalizing the type of an object in the hypothesis (Fig.6). There are two possible approaches to apply the Type Refinement operator: *type first* approach is to select a type from the set of types for a hypothesis and generalize the type of the variables that belong to the selected type and *variable first* approach is to select a variable from the hypothesis and generalize the type of all occurrences of this variable in the hypothesis. The *type first* approach generalizes the selected type throughout the hypothesis and this may involve several variables while the *variable first* approach only generalizes the type of one variable. In our work, we use the *type first* approach as this has a fewer number of refined hypotheses and a smaller search space while the *variable first* approach has a larger number of choices and hence a larger search space. One more reason to use the *type first* approach is that the computer vision algorithm might confuse the type of a whole group of objects that belong to a particular type rather than a single object in a video because of an inaccurate object detector.

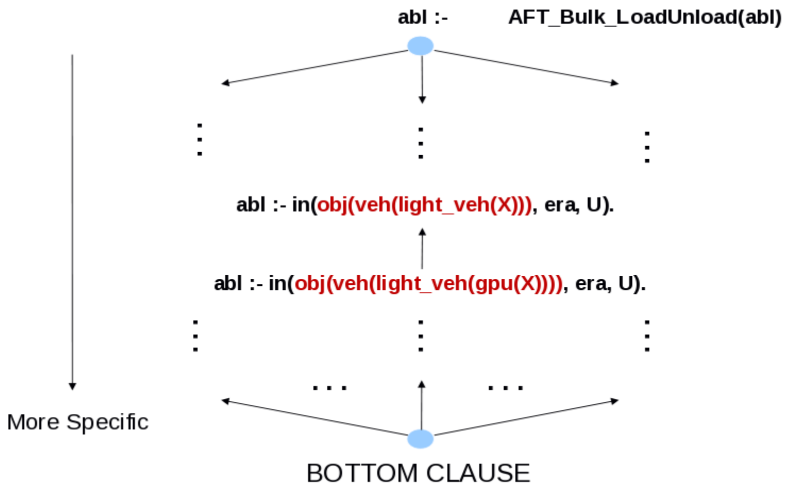

abl :-
AFT_Bulk_LoadUnload(abl)
abl :- in(obj(veh(light_veh(X))), era, U).
abl :- in(obj(veh(light_veh(gpu(X)))), era, U).
More Specific
BOTTOM CLAUSE

Figure 6: Type refinement operator (generalization).

6.2 Searching the Event Model

Once the *most-specific clause* is computed, the sub-lattice bounded from below by the *most-specific clause* is searched using a *best-first* search for the hypothesis that has a *maximum* score calculated based on a combination of (1) the number of positive examples covered, (2) the number of *answer substitutions* in the negative examples, (3) the length of hypothesis and (4) the number of distinct variables in the hypothesis subject to the given constraints (discussed in the next subsection).

$$\text{score}(H) = \gamma * p - (\varrho * n + l + v)$$

where

γ = weight to positive examples

p = number of positive examples covered

ϱ = weight to answer substitutions in negative examples

n = number of answer substitutions in negative examples

l = length of the hypothesis

v = number of distinct variables in the hypothesis

An *answer substitution* θ for an example e is a substitution that grounds the hypothesis, $h \leftarrow b_1, \ldots, b_n$, and the query $\leftarrow b_1\theta, \ldots, b_n\theta$ succeeds in the database of e. Note that in the *learning from interpretations* setting each positive example is a separate database and when a hypothesis is used as a Prolog query in each database, it might result in multiple answer substitutions. An example is considered *covered* by the hypothesis if there are one or more answer substitutions. While testing the hypothesis in a test database, each answer substitution is considered as one recognized event instance. If the recognized event interval falls outside the event ground truth in a test video, it is considered as false positive. In the event recognition domain, the hypothesis is used for recognizing events in unseen videos instead of classifying videos, a hypothesis with fewer false positives is desirable. Hence hypotheses are penalized using the number of answer substitutions[10] in negative examples. If the number of positive and negative examples are disproportionate in numbers, giving more weight to the positive examples and negative examples using γ and ϱ will result in an hypothesis that has better performance in test data.

Since the starting hypothesis is empty and completely generic, it will cover all the negative examples. As the hypothesis is specialized by Progol's refinement operator, the number of false positives decreases. When the score of the hypothesis no longer increases, the Type Refinement operator is used to generalize the types thereby increasing the generality of the hypothesis with a possible increase of positive examples covered (as well as false positives). This process of interleaved application of both operators is continued until the hypothesis score no longer increases.

Once a satisfactory hypothesis is found, an argument representing temporal information in the form of a list of time intervals formed using the time intervals in the body of the event model can be introduced into the head in order to explicitly represent *when* the event occurs – this is useful when using the hypothesis for event monitoring – it allows the interval during which the event occurs to be explicitly flagged when viewing the video.

The learning algorithm uses a *set covering* method (Quinlan, 1990) to learn an event model that is a set of clauses interpreted as a disjunction. The covering method starts with an empty model and learns a clause using the provided positive and negative examples and adds this clause to the model. It repeats this procedure but now with positive examples that are not covered by the earlier clause. This process is repeated until all the positive examples are covered.

6.3 Constraining the Search Space

The size of the search space depends on the size of the *bottom clause* (Muggleton, 1995). Thus, in the event learning domain, it depends on the number of spatial relations being used and the number of objects in the event instances used as positive examples. The size of the *bottom clause* increases with the number of Allen's temporal relations as each interval in the atoms with spatial predicates is temporally connected to every other interval in other atoms with spatial predicates. This creates many atoms with temporal predicates in the *bottom clause.*

In order to decrease the size of the search space, the algorithm makes use of domain-dependent and domain-independent constraints on the structure of the hypothesis. The constraints the algorithm uses such as restrictions on the hypothesis length and the number of variables in the hypothesis etc. are domain-independent structural constraints as they do not depend on the predicates used or any domain knowledge. The following are the two domain-dependent constraints that reduce the search space and time thereby making the learning process more efficient.

- Upper bounds on the number of atoms in the body of a rule.

10. Counting the number of answer substitutions instead of number of examples covered is a heuristic used in the FOIL system (Quinlan & Cameron-Jones, 1993).

- Any interval in an atom with spatial (temporal) predicate should appear in an atom with a temporal (spatial) predicate. Any hypothesis with atoms that do not satisfy this criteria is not semantically meaningful since it might be satisfied by facts not at all related to the event in question.

However the constraints listed above are domain dependent constraints rather than application specific constraints, i.e., these constraints that involve the spatial and temporal predicates should be applicable in most, if not all event learning scenarios. Note that some of the constraints are hard (i.e. inviolate). If a hypothesis violates any such constraint, then it is discarded without scoring or refining. For example, the domain-independent constraints and the first domain-dependent constraint mentioned above are hard. In contrast, if a hypothesis violates a constraint that is not hard, for example, the second domain-dependent constraint listed above, it is not scored and discarded only after generating refined hypotheses from it. This is because discarding such hypotheses without refining might obstruct the traversal of the lattice. For example, in the current work, the algorithm starts the search process with an empty hypothesis and the initial hypotheses obtained by refining the empty hypothesis do violate the second domain-dependent constraint listed above (since they contain exactly one predicate and therefore cannot contain both a spatial and a temporal predicate).

6.4 Event Recognition

The learned event models are used for event recognition in unseen videos. For this purpose, the test video is converted to relational data and used as a database and the event models are used as Prolog queries. The querying is done in the whole database and the intervals extracted from the answer substitutions from these queries give the temporal extent of the recognized instances of the events. In order to record when the event takes place, we change the arity of the event predicate (the rule head) to be monadic such that the argument is a list of all interval variables occurring in the body. Note that it would also be possible to introduce a second argument to record which objects are involved in the event (i.e. a list of all variables of type object - or equivalently that occur as the first two arguments of any spatial predicate in the body).

An issue that arises is exactly when an event occurs given that it consists of multiple overlapping temporal intervals from the instantiated predicates in any given answer substitution. Given the list of all intervals λ occurring in the instantiated body of the hypothesis, various possibilities present themselves. One could take the maximal interval which exactly spans all intervals in λ. Or one could take the interval which exactly spans the interval from the first transition (i.e. pair of meeting intervals involving the same pair of objects) to the last such transition. Clearly there are other possibilities too. Ultimately this is probably a domain dependent decision. For our experiments, we take the list of all intervals in λ and the temporal extension of the event is obtained by taking the minimum and maximum of the time points in λ.

Note that there may be several rules for an event class, each rule capturing a variation in which an event can happen. These rules do not have weights specifying how important or reliable a rule is for recognizing events. When recognizing events, all the rules of an event class are used and this may result in multiple answer substitutions.

7. Interleaved Induction and Abduction ($\mathcal{IIA}$)

In the previous section we showed how ILP can be applied to learn rule-based relational event/activity models, given an observation dataset, positive and negative examples of events whose models have to be learnt. However, data from visual and other sensors tend to be noisy with high variability in the sample space. This leads to *over-fitted* models (i.e., more rules), as the model has to cover some of the examples that are corrupt because of the sensor noise. A model with more rules can result in many false positives when used for event-recognition in test data.

In this section, we show how well-fitted, semantically meaningful event models can be learned from noisy data by interleaving induction and abduction. This acquires significance in cases where training data is scarce and noisy. We apply the Typed ILP system presented in the previous section to learn event-based models and using these models as the domain theory, we explain the examples/observations not covered by the induced theory using abduction. The uncovered examples are either noisy, or are examples for the same event that in reality happened in a different way. Using the explanation we rectify the errors in the noisy examples corrupted by tracking errors and thus reduce the requirement for additional rules. In our framework, the examples themselves are noisy (i.e. incorrect) thereby requiring *observation data revision* in a manner that is consistent with the initially learned theory, and general common-sense knowledge about *space*, *spatial change*, and the dynamics of the domain. Note that many ILP approaches discard examples considering them as noisy by using a heuristic stopping criteria. This is not acceptable in cases where there is scarcity of training data, where learning from every example is potentially important.

7.1 Domain-Independent Spatial Theory

In order to pursue our goal, an *Axiomatic Characterisation of the Spatial Theory* is necessary. Many spatial calculi exist, each corresponding to a different aspect of space. Here, it suffices to focus on one spatial domain, e.g., topology, with a corresponding mereotopological axiomatization by way of the binary relationships of the RCC-8 fragment $\mathcal{R}_{rcc8}$. From an axiomatic viewpoint, a spatial calculus defined on $\mathcal{R}$ has some general properties (P1–P5), which can be assumed to be known apriori. To realize a domain-independent spatial theory that can be used for reasoning (e.g., spatio-temporal abduction) across dynamic domains, it is necessary to formalize a domain-independent spatial theory (Σ_{space}) which preserves the high-level axiomatic semantics of these generic properties. For reasons of space we only sketch the properties P1–P5 and neglect the formal axiomatization.

(P1–P2) The **Basic Calculus Properties (Σ_{cp})** describe the *jointly exhaustive & pairwise disjoint* (JEPD) property, i.e., for any two entities in $\mathcal{O}$, one and only one spatial relationship from $\mathcal{R}$ holds in a given situation. The *jointly exhaustive* property of $n = |\mathcal{R}|$ base relations can be axiomatized by n ordinary state constraints and, similarly, the *pairwise disjoint* property can be axiomatized by $[n(n-1)/2)]$ constraints. Other miscellaneous properties such as symmetry and asymmetry can be expressed in the same manner.

(P3) The primitive relationships in $\mathcal{R}$ have a *continuity structure*, referred to as its **Conceptual Neighbourhood (Σ_{cn})** (CND) (Freksa, 1991), which determines the direct, continuous changes in the quality space (e.g., by deformation and/or translational motion).

(P4) From an axiomatic viewpoint, a spatial calculus defined on $\mathcal{R}$ is (primarily) based

on the derivation of a set of **Composition Theorems (Σ_{ct})** between the JEPD set $\mathcal{R}$. In general, for a calculus consisting of n JEPD relationships, $[n \times n]$ compositions are precomputed. Each of these composition theorems is equivalent to an ordinary state constraint, which every n-clique spatial situation description should satisfy.
(P5) Additionally, **Axioms of Interaction (Σ_{ai})** are necessary when more than one spatial calculus is modelled in a non-integrated manner (i.e., with independent composition theorems). These axioms explicitly characterize the relative entailments between inter-dependent aspects of space, e.g., topology and size.
Now, let $\Sigma_{space} \equiv_{def} [\Sigma_{cp} \cup \Sigma_{cn} \cup \Sigma_{ct} \cup \Sigma_{ai}]$ denote a domain-independent spatial theory that is based on the axiomatizations encompassing (P1–P5).

7.2 Physically Plausible Scenarios

Corresponding to each spatial situation (e.g., within a hypothetical situation space), there exists a situation description that characterizes the spatial state of the system. It is necessary that the spatial component of such a state be a 'complete specification', possibly with disjunctive information. For k spatial calculi being modelled, the initial situation description involving m domain objects requires a complete n-clique specification with $[m(m-1)/2]$ spatial relationships for each calculus. Therefore, we need to define a scene description to be **$\mathcal{C}$-Consistent**, i.e., compositionally consistent, if the n-clique state or spatial situation description corresponding to the situation satisfies all the composition constraints of every spatial domain (e.g., topology, orientation, size) being modelled. If more than one calculus is modelled the inter-dependent constraints (P5) must hold as well.

From the viewpoint of model elimination of narrative descriptions during an (abductive) explanation process, *$\mathcal{C}$-Consistency* of scenario descriptions is a key (contributing) factor determining the commonsensical notion of the *physically realizability* of the (abduced) scenario completions. Bhatt and Loke (2008) show that a standard completion semantics with *causal minimization* in the presence of frame assumptions and ramification constraints preserves this notion of $\mathcal{C}$-Consistency for Σ_{space} within a logic programming framework, as well as with arbitrary basic action theories.

7.3 The Inductive-Abductive Framework

We interleave inductive and abductive commonsense reasoning about space, events and change within a logic programming framework. Induction is used as a means to learn event models by generalizing from sensory data, whereas abductive reasoning is used for noisy data correction by *scenario and narrative completion*, thereby improving the learning.

7.3.1 Explanation by Abduction

Diametrically opposite to projection and planning is the task of post-dictum or explanation (Poole, Goebel, & Aleliunas, 1987), where given a set of time-stamped observations or snap-shots, the objective is to explain which events and/or actions may have caused the observed state-of-affairs. Explanation problems demand the inclusion of a narrative description, which is essentially a distinguished course of actual events about which we may have incomplete information (Miller & Shanahan, 1994). Narrative descriptions are typically available as sensory *observations* from the real execution of a system or process. Given narratives, the objective is often to assimilate/explain them with respect to an underlying process model.

The *abductive explanation* problem can be stated as follows (Kakas, Kowalski, & Toni, 1992):
Given: Theory T and observations G, find an explanation $\triangle$ such that:

- $T \bigcup \triangle \models G$
- $T \bigcup \triangle$ is consistent

i.e., the observation follows logically and non trivially from the theory extended given the explanation. Abductive explanations are usually restricted to ground literals with predicates that are undefined in theory, namely the *abducibles*. Abductive explanations are derived by trying to prove the observation from the initial theory alone: whenever a literal is encountered for which there is no clause to resolve with, the literal is added to the explanation.

The abduction procedure results in many valid explanations. In order to reduce the number of explanations, several restrictions as listed below can be used (Kakas et al., 1992):

Explanations should be basic This means one explanation should not explain another explanation. This is enforced by not allowing *abducibles* in the head of any rule.

Explanations should be minimal This means one explanation should not subsume another explanation.

Explanations should satisfy all integrity constraints With this restriction, we obtain explanations that are valid in the domain under consideration. In our work, explanations should satisfy all the spatial constraints of the underlying spatial theory.

7.3.2 Scenario and Narrative Completion

It is easy to intuitively infer the general structure of narrative completion by abductive explanation. Consider the illustration in Fig.7 for a hypothetical situation space that characterizes the complete evolution of a system. In Fig.7 the situation-based history given by the solid arrows represents one path, corresponding to an actual time-line discretized into intervals $\langle \delta_0,\ \delta_1, \ldots,\ \delta_m \rangle$, within the overall branching-tree structured situation space. Given incomplete narrative descriptions, e.g., corresponding to only some ordered intervals in terms of high-level spatial (e.g., topological, orientation) and occurrence information, the objective of explanation is to derive one or more paths from the branching situation space, that could best-fit the available narrative information. Formally:

$$\left\{\begin{array}{l} \Phi_1 \ \equiv\ touch(a,\ c,\ \delta_1) \\ \Phi_2 \ \equiv\ dc(a,\ c,\ \delta_4)\ \wedge\ in(b,\ a,\ \delta_4)\ \wedge\ dc(b, c,\ \delta_4) \\ [\Sigma_{space}\ \wedge\ \Phi_1\ \wedge\ \Omega]\ \models\ \Phi_2, where \\ \quad \Omega\ \equiv\ (\exists\ \delta_i,\ \delta_j).[\ meets(\delta_1, \delta_i)\ \wedge\ before(\delta_i, \delta_4)\ \wedge\ dc(b,\ a,\ \delta_i) \\ \qquad\qquad\qquad\qquad \wedge\ touch(a,\ c,\ \delta_i)\ \wedge\ dc(b, c,\ \delta_i)] \\ \qquad\qquad\qquad \wedge\ [\ meets(\delta_i, \delta_j)\ \wedge\ \ meets(\delta_j, \delta_4)\ \wedge\ touch(b,\ a,\ \delta_j) \\ \qquad\qquad\qquad\qquad \wedge\ touch(a,\ c,\ \delta_j)\ \wedge\ dc(b, c,\ \delta_j)] \end{array}\right\} \tag{7.1}$$

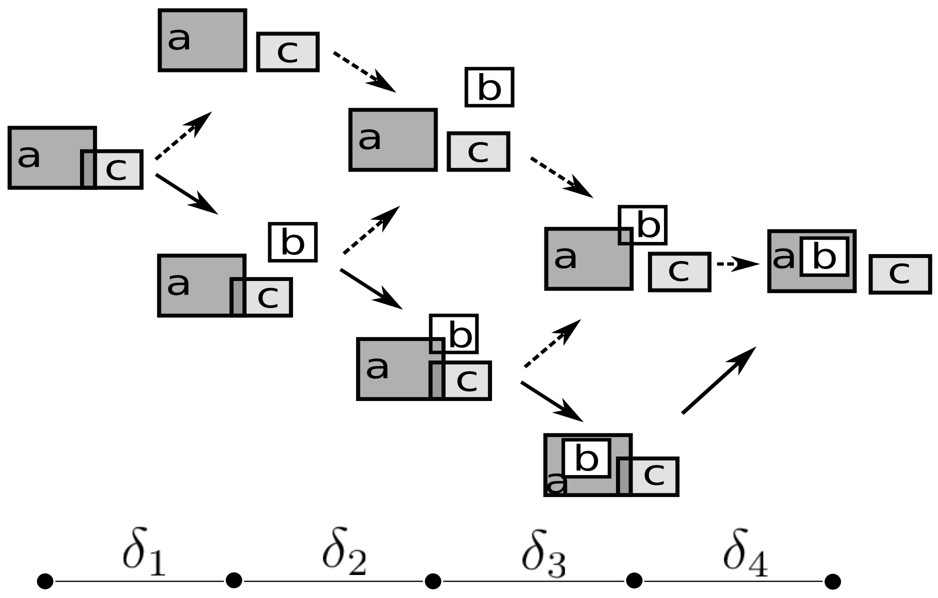


Figure 7: Branching/Hypothetical Situation Space. Only a few possibilities are shown. There are clearly more paths from the initial scenario to the target scenario. There are also more possible states.

In (7.1), Φ_1 denotes the initial situation and Φ_2 denotes the final situation represented in terms of spatial relations (RCC-5) among the objects present in the scene. The abductive derivation of Ω, that explains how the scene changed from situation Φ_1 to situation Φ_2, primarily involves non-monotonic reasoning in the form of minimizing change, in addition to making the default assumptions about inertia, and an appropriate treatment of ramification constraints (Bhatt & Loke, 2008).

7.4 $\mathcal{IIA}$ Algorithm

Most ILP systems use a *covering* algorithm to learn models from examples. The search ranges over a hypothesis lattice and each hypothesis is evaluated based on the number of positive and negative examples it covers. If it is selected as a suitable hypothesis based on some scoring function, this hypothesis (rule) is added to the model, the covered examples are removed and this process is repeated until all the positive examples are covered. Examples can be corrupted by noise resulting in missing or incorrect facts. In such cases, more rules are learned than necessary in order to cover all the examples. As the number of rules for a concept increases, this may result in many false positives when the rules are used for classification/recognition in test examples. In order to avoid learning from corrupt examples, the framework identifies examples as being corrupted by explaining them through abduction using the already induced model and the background theory (Dubba et al., 2012).

The main assumption we make here is that the noise in the examples is not consistent. If the noise is consistent (i.e., present in most of the examples in a similar fashion) then it becomes part of the pattern that defines the concept and might be learned by the learning algorithm.

The pseudo algorithm is given in Algorithm 1. The induction algorithm induces an initial hypothesis based on the score function explained in previous sections. The positive examples covered (E^+_{Rule}) by this hypothesis are removed from the list of positive examples yet to be covered. The induced theory along with background knowledge is used to explain the uncovered examples treating each example as a narrative. Abduction gives several possible explanations each with different cost (based on the nature and the number of facts in the explanation). The explanations are rejected if they have a cost more than a specified threshold. Furthermore, given the formulation of the spatial theory Σ_{space}, $\mathcal{C}$-Consistency of abduced explanations is ensured. The examples ($E^+_{\triangle}$) that have an explanation whose cost is less than the specified threshold are removed from the positive examples list that are yet to be covered, as they are now considered to be covered by the already induced model. This process of induction and abduction is repeated until all the positive examples are covered.

Apart from the constraints enforced by the spatial theory to filter abduced explanations, several heuristics can be used to give a score to each explanation so that a low cost consistent explanation can be selected by the system. One of the several possible heuristics is to prefer explanations where the number of transitions in spatial relations is minimal (Hazarika & Cohn, 2002). This heuristic is a direct consequence of McCarthy's *Common Sense Law of Inertia* (McCarthy, 1986) which states that change is abnormal and persistence is to be preferred in the absence of data. In a spatio-temporal domain, each explanation abduced in the absence of data is a set of spatio-temporal facts and there are three ways to add them to the explanation: (i) Extend the current relation between two objects (can be done in both directions of the timeline if the situation permits) (ii) change the current relation between two objects to its neighbouring relation in a CND (iii) introduce a new object (hypothetical) into the scene and its spatial relations with other objects in the scene as well. The cost of each explanation is based on the type of spatio-temporal fact chosen and is calculated as explained below.

7.4.1 Cost of an Abduced Explanation

Let $\triangle$ be an explanation from the abduction procedure where $\triangle$ is a set of grounded spatio-temporal facts of the form $r(o_i,\ o_j,\ \delta_k)$ denoted as f_{ijrk}, E^+_p is the current positive example (an interpretation, i.e., a set of facts) and let r be from the set of spatial relations $\mathcal{R}$ in a spatial calculus. Let $\mathcal{O}$ be the set of objects in E^+_p. Let $c_{f_{ijrk}}$ be the cost of abducing f_{ijrk}. The total cost of $\triangle$ denoted as $C_\triangle$ is $\sum\limits_{f_{ijrk} \in \triangle} c_{f_{ijrk}}$.

The cost of abducing f_{ijrk} is calculated as follows:

$$c_{f_l} = \begin{cases} \iota, & \text{if there exists a } f_{ijrm} \text{ in } E^+_p \text{ such that } \delta_k \text{ and } \delta_m \text{ are } \mathit{disjoint} \\ \kappa, & \text{if there exists a } f_{ijsm} \text{ in } E^+_p \text{ such that } \delta_k \text{ and } \delta_m \text{ are } \mathit{disjoint} \text{ and } r \neq s \\ \mu * n, & \text{where } n \text{ is the number of hypothetical objects (objects not in } \mathcal{O}) \text{ in } f_{ijrk} \end{cases}$$

where $\iota < \kappa < \mu$.

The first case in the cost function occurs when the system abduces a fact that extends a relation between two objects in the temporal dimension. This does not count as a spatial transition and hence has a very low cost. In contrast, the second case occurs when the system abduces a fact that extends the existence of two objects in temporal dimension with a different relation (the new spatial relation must be a neighbour to the existing relation in the CND) than the one that already exists between them. This counts as a spatial transition and has a cost more than the first case. The third case occurs if it is necessary to have a hypothetical object to satisfy the hypothesis in E_p^+. This case is used when an object involved in an event is completely missed by the object tracker while first two cases are used in scenarios where an object is not detected in some temporal slice in its life time. Note that the first case is clearly the most preferred if the abduction procedure has to find a low cost explanation and the third case which is the most expensive applies when an object is completely missed by the object tracker. Though it is possible to avoid transitions to reduce the score, sometimes it is mandatory to consider transitions. For example, consider a scenario where two objects have a *dc* relation and in the final state they have an *in* relation. In this case, the algorithm has to abduce facts where there are two transitions (one when the *dc* relation changes to *touch* and another when *touch* changes to *in* relation). Note that it is not necessary to abduce temporal relational facts as the Prolog definitions for temporal relations in the background theory can be used to compute them when needed. This can be achieved by not including temporal predicates in the list of *abducibles*.

The abduction procedure uses the existing constants in the database and one issue with this is that though the number of relations and objects is small, the number of possible intervals is very large if not constrained. In order to constrain the possible explanations, we introduce intervals with predefined duration into the database so that abduction uses only these intervals for abducing explanations. Note that abduction as we have defined it only adds missing spatio-temporal information and cannot be used to retract corrupted data resulting from noise.

Algorithm 1 Interleaved Induction and Abduction algorithm ($\mathcal{IIA}$)

```
procedure IIA($E^+, E^-, B$) ▷ training sets and background knowledge (includes spatial theory)
    $H \leftarrow \emptyset$
    $\triangle \leftarrow \emptyset$
    while $E^+ \neq \emptyset$ do
        $Rule \leftarrow Induce(B, E^+, E^-)$
        $H \leftarrow H \bigcup \{Rule\}$
        $E^+ \leftarrow E^+ - E^+_{Rule}$
        $\triangle \leftarrow Abduce(B, H, E^+)$
        $E^+ \leftarrow E^+ - E^+_{\triangle}$
    end while
    return $H$                                    ▷ Learned theory
end procedure
```

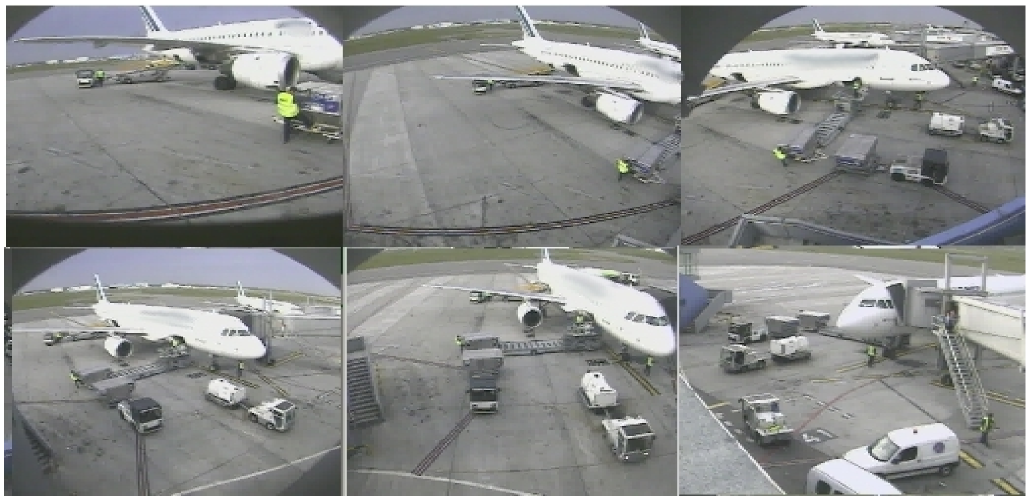


Figure 8: Airport domain: The videos are recorded using 6 static cameras looking at the same scene from different angles.

8. Experimental Results

In this section, we present an evaluation of REMIND, as well as the extension presented in Section 7. For the experiments, we used two real world video datasets that are different from each other in many aspects. The videos from these datasets are shot in outdoor settings and in different weather and light conditions (rainy, cloudy, sunny, night). These variations in the videos present various challenges to the vision system and subsequently to the learning system both in the training phase and the event recognition phase.

The two datasets used in this work for evaluation are from airport logistics and verb videos. The datasets from these domains differ in many aspects such as number of objects in the video, length of video, duration of events, background structures, the number of cameras used to capture the events and also the plane (image plane or ground plane) in which the tracking data is made available. We view the differences in the datasets as a positive aspect - the framework shown to work in two very different kinds of scenarios.

REMIND[11] is implemented in Python and for speed, some modules are implemented in Cython; SWI-Prolog is used as the underlying Prolog engine for storing and querying relational facts and background knowledge.

8.1 Airport Logistics

For experiments in the airport logistics domain, 15 turn-arounds[12] were used where each turn-around was shot using 6 cameras from different angles (Fig.8) and each video is on average one hour long (15 frames per sec).

The following are informal descriptions of the International Air Transport Association (IATA) events we aim to learn models for:

11. Available on request from the first author and will be made public in the near future.
12. A turn-around is the duration of a plane entering and leaving the apron area.

Aircraft_Arrival	Aircraft comes into the apron
Aircraft_Departure	Aircraft moves away from its position on the apron
GPU_Positioning	Ground power unit comes and positions in its zone
Left_Refuelling	Fuel truck arrives on the left side of aircraft for refuelling
PB_Positioning	Push-back vehicle positioning in front of the aircraft
PBB_Positioning	Passenger Boarding Bridge attaches itself to the aircraft
PBB_Removing	Passenger Boarding Bridge detaches itself from the aircraft
FWD_CN_LoadUnload	Container Loading/Unloading at the front end of the aircraft
AFT_CN_LoadUnload	Container Loading/Unloading at the rear end of the aircraft
AFT_Bulk_LoadUnload	Baggage Loading/Unloading at the rear end of the aircraft
FWD_Bulk_LoadUnload	Catering Loading/Unloading at the front end of the aircraft

Within each event, there is high variability because of noise in tracking and also because of objects extraneous to the event entering the event scene. Note that some events might not be present or may occur multiple times in some turn-arounds. The scenes involve interactions of vehicles and people with *zones* on the apron. These zones are specified on the ground plane according to the IATA specifications and the position of the zones depends on the type of aircraft. These zones are used for parking and steering vehicles for different operations carried out in a turn-around. Note that these zones are static throughout the video and do not change size or position, unlike the bounding boxes of the vehicles obtained through tracking. Hence zones are not included in the type hierarchy used for the domain (Fig.4) since they do not suffer from visual noise. The main reason to use zones is that the RCC-5 spatial relations between bounding boxes of vehicles and people in the ground plane rarely touch, hence most of the interactions are encoded as *dc* if zones are not used. It is important to use the zones as most of these interactions happen in the zones. According to the IATA specifications, a vehicle transition through the zones and the position of the vehicle in particular zones is important to determine the events.

We use the object tracks provided by a partner in the Co-Friend project (Ferryman, Borg, Thirde, Fusier, Valentin, Bremond, Thonnat, Aguilera, & Kampel, 2005); certain details in some events are not detectable with this tracking system such as the direction of baggage on the rail of the loader vehicle and whether the trolleys are empty when they arrive at the scene. The *Load/Unload* events obtained from IATA events differ in such details (if the trolleys are loaded when they arrive at the scene and the baggage is moving towards the aircraft, then the event is *loading* and if the trolleys are empty when they arrive at the scene and the baggage is moving away from the aircraft, the event is *unloading*). Apart from these details, they are semantically similar and hence are regarded as the same events (for example, FWD_CN_Load and FWD_CN_Unload are regarded as the same events and named FWD_CN_LoadUnload, the same strategy is followed with other Load/Unload events).

8.1.1 Tracking and Obtaining Relational Data

The apron scene area is too large to be covered by a single static camera. The events on the apron occur on both sides of the aircraft which is very difficult to cover with a single camera and because of the size of the aircraft it is possible to have many occluded objects in the scene. In order to solve these problems, six cameras are used to shoot the scene from different angles so that the entire area is covered and the number of occluded objects is minimized. Working on the ground plane data results in learned models that are independent of the camera view and the airport as these models can be readily applied at different airports with different camera configurations.

The tracking data is obtained for each of the videos from six cameras of a turn-around and fused together to get 3D data on the ground plane (Ferryman et al., 2005). The tracking data is noisy because of low quality, bad light and weather conditions and low contrast of CCTV videos. The noise can be the presence of phantom objects, missing objects, wrong types of vehicles, inaccurate bounding boxes, broken trajectories, object identity, inconsistencies etc. which are typical problems in any computer vision tracking system. Each turn-around is separately processed to get relational data that consists of a set of spatial relations among the vehicles and zones on the apron. Prolog rules that decide the temporal relationships among intervals are considered as background information in the ILP system. The data for each video has between 250 and 500 spatial relational facts (excluding temporal relational facts) depending on the number of objects and the interactions between these objects.

Note that an *event* requires at least one change in the *state* (here, spatio-temporal relations between pairs of objects) of the objects. If the relation between any two objects is *dc* and does not change during the life span of the objects, it signifies that the objects are not interacting and the relational fact is discarded as these spatio-temporal facts do not contain relevant information in defining event models. The tracking data also consists of bounding boxes for people in the scenes, but these are discarded as people are not germane in the semantics of the events and also these increase the size of the relational data.

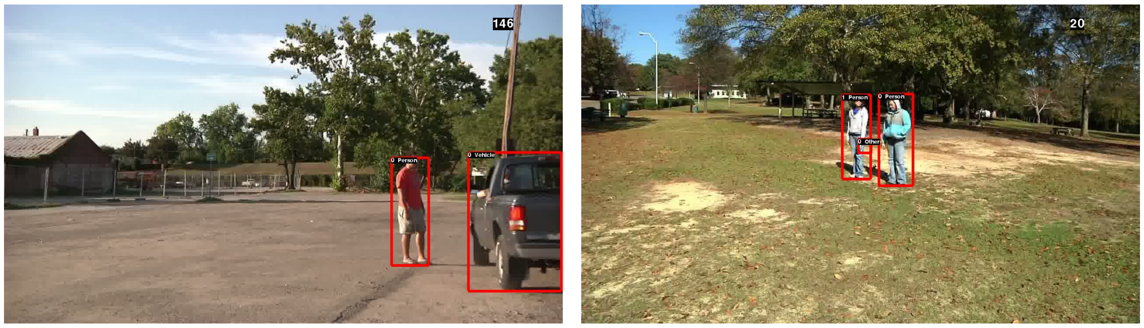
8.1.2 Annotation of Events

For supervised learning we need positive and preferably negative example instances of events. In the airport domain, the temporal extent of the events is provided by individuals who have expertise in the IATA protocols and apron activities, by specifying the start and end frame numbers of the event instance in that video. The spatial extent was obtained by using a tool with which a polygon can be drawn on one of the image planes and the corresponding ground plane region is obtained using an *homography* (it was easier for a human annotator to watch an actual video to provide the spatial annotation rather than view a 3D visualization in the ground plane, which being the fusion of the imperfectly tracked data that does not always show all relevant objects). This region gives a spatial extent for the event instance.

8.2 Physical Action Verbs Dataset

The *Action Verbs* dataset[13] is a corpus of video vignettes (Fig.9) that portray motion verbs such as *approach, exchange, jump, collide,* etc. enacted in natural environments like parks,

13. This dataset (Mind's Eye Year 1 recognition task videos) is provided by DARPA and publicly available from `http://www.visint.org/datasets`



(a) *Approach* event with tracked objects

(b) *Snatch* event with tracked objects

Figure 9: Example event instances for *Approach* and *Snatch* in *Action Verbs* dataset

urban places, etc. These vignettes are short in duration when compared to videos from the airport domain, a few tens of seconds. The full list of verbs is given in Table 3.

Though each vignette was shot to portray a single verb action, other verbs are inevitably present as well, sometimes overlapping in time. This is primarily unavoidable, for example, a vignette that portrays the verb *carry*, will automatically include *walk* if a person is carrying some object in their hand. This aspect is taken into consideration while annotating the vignettes. We were able to obtain tracked data from an external source (Morariu, Harwood, & Davis, 2013) including object type information (Fig.10).

The new challenge in using this dataset is the different ways a verb can be enacted. There are 48 verbs in the dataset and a total of 1615 vignettes are used for training and 2348 vignettes are used for testing.

8.2.1 Tracking and Obtaining Relational Data

The tracking data available to us often suffers from errors, e.g., a bouncing ball is often only tracked once it is being held and fast moving objects such as a running person are missed. We used Qualitative Trajectory Calculus relations (QTC_{L1}) (Van de Weghe et al., 2006) as the primitive spatio-temporal relations. We did not choose RCC for this dataset as it seemed unlikely that a purely topological representation would be sufficient. In contrast, the QTC_{L1} relations capture the typical movements in the verbs dataset like *moving away, approaching, follow* etc. For example, in a *chase* event where one object is following another object the relation is *dc*, using RCC, which is also the same for two objects standing still with some distance in between.

It is difficult to model motion patterns of objects like *run, walk, raise, bend* etc. using only relational data without referring to parts of a person. The verbs dataset contains some events that involve such motion patterns and to recognize these, pixel based models are more appropriate. The primitive events were recognized in all the videos using the method proposed by Jiang, Lin and Davis (2010), where an action is represented as a sequence of joint HOG-flow descriptors (Dalal & Triggs, 2005) extracted independently from each frame. Instead of applying this approach to the entire frame and video as proposed by Jiang et al. (2010), the input was restricted to sliding temporal windows along the spatio-

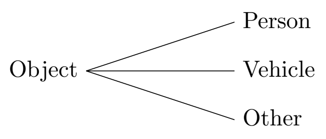


Figure 10: Tree-structured object type hierarchy in the Action Verbs domain.

temporal volume defined by a person's bounding box. These primitive events[14] in addition to QTC_{L1} relations provide the relational data in the verbs domain.

8.2.2 Annotation of Events

The ground truth for events in the verbs dataset vignettes is of a different nature to the ground truth in the airport domain. The development and the test set were annotated by 10 people using Amazon Mechanical Turk (AMT). Each vignette was presented to the annotator and 48 questions were presented in the form: *Is verb X present in this vignette?* Only verbs annotated by more than 50% of the annotators are considered as events present in the vignette. For the development set, the annotations were extended by providing for each event instance its temporal extent.

8.3 Experimental Results and Evaluation for the Typed ILP Framework

Sample rules[15] learned for Aircraft Arrival and AFT_Bulk_LoadUnload events are given below. For example, the Aircraft Arrival rule can be interpreted as: an aircraft arrives when the aircraft bounding box has the *in* relation with the right_AFT_Bulk_TS_Zone and then moves forward thereby changing the relation to *touch*. This happens when the aircraft arrives and is moving to its position. The rule also correctly identifies that this bounding box should belong to an object of type *aircraft*. The goals in the rule are ordered such that spatial predicates come before (to the left of) the temporal predicates since there are more temporal facts[16] compared to spatial facts and this ordering speeds up the query execution.

```
aircraft_arrival([intv(T1,T2), intv(T3,T4)]) :-
    in(obj(aircraft(V)), right_AFT_Bulk_TS_Zone, intv(T1,T2)),
    touch(right_AFT_Bulk_TS_Zone, obj(aircraft(V)), intv(T3,T4)),
    meets(intv(T1,T2), intv(T3,T4)).
```

```
aft_bulk_loadunload([intv(T1,T2), intv(T3,T4)]) :-
    touch(left_TK_Zone, obj(veh(heavy_veh(V1))), intv(T1,T2)),
    touch(obj(veh(V2)), left_TK_Zone, intv(T3,T4)),
    meets(intv(T3,T4), intv(T1,T2)).
```

We followed the standard *leave-one-out* methodology for testing performance in the airport domain. All turn-arounds except one are used for training and the remaining one is used as a test case. This process is iterated until each turn-around is used as the test case

14. This data was provided by Vlad Morariu from University of Maryland.
15. A temporal interval is represented as $intv(T_1, T_2)$ for programming convenience where T_1 and T_2 are the starting and ending frame numbers of the interval.
16. As already noted, temporal facts are not explicitly stored but are computed via background knowledge rules.

Event	#Examples	Without Type Generalization			With Type Generalization		
FWD_CN_LoadUnload	7	0.86	0.06	0.11	0.86	0.08	**0.15**
GPU_Positioning	16	0.4	0.03	**0.05**	0.27	0.02	0.04
Aircraft_Arrival	15	0.43	0.01	**0.02**	0.36	0.01	**0.02**
AFT_Bulk_LoadUnload	29	0.72	0.20	**0.31**	0.72	0.20	**0.31**
PBB_Removing	15	0.43	0.06	0.10	0.36	0.12	**0.18**
Left_Refuelling	8	0.25	0.03	0.05	0.12	0.10	**0.11**
PB_Positioning	14	0.28	0.04	0.07	0.14	0.06	**0.08**
Aircraft_Departure	12	0.41	0.11	0.17	0.33	0.19	**0.24**
AFT_CN_LoadUnload	15	0.80	0.05	0.09	0.67	0.07	**0.13**
PBB_Positioning	15	0.73	0.16	0.26	0.67	0.34	**0.45**
FWD_Bulk_LoadUnload	3	1.00	0.24	0.39	1.00	1.00	**1.00**
Weighted Average				0.15			0.20

Table 1: Performance comparison of models obtained without and with using types using RCC-5 primitives in the airport domain. The first, second and third columns for each category are *recall*, *precision* and *f1* respectively. The best *f1* value in each case is presented in bold. It is clear from the table that using types improves the overall performance. By *without type generalization* we mean, type information from tracker is ignored (all objects have the same type) and type generalization is not performed during learning.

exactly once. The results of our experiments are summarised in Table 1. The third and fourth columns show the recall and precision without using types in the 15 turn-arounds (i.e., type information from the tracker is ignored, hence all objects have the same type and type generalization is not performed during learning). The fifth and sixth columns show the recall and precision using the type hierarchy. From the tables it is clear that using type information can increase the accuracy of event recognition. Also the combined execution time for all the experiments using type generalization was reduced by roughly 30% when compared to the execution time of experiments without type generalization.

The detailed recognition results (temporal localization) for events in a turn-around is shown in Fig. 11 (best seen in colour). The plot shows a turn-around with one subplot showing ground truth of event instances and another subplot showing the recognized instances by the Typed ILP system in that turn-around. Each event is colour coded for comparing ground truth with the recognized instance intervals. Note that a recognized event instance is considered a true positive if it overlaps at least 20% with the corresponding event ground truth interval (Oh et al., 2011). In some cases the temporal extent of recognized event in-

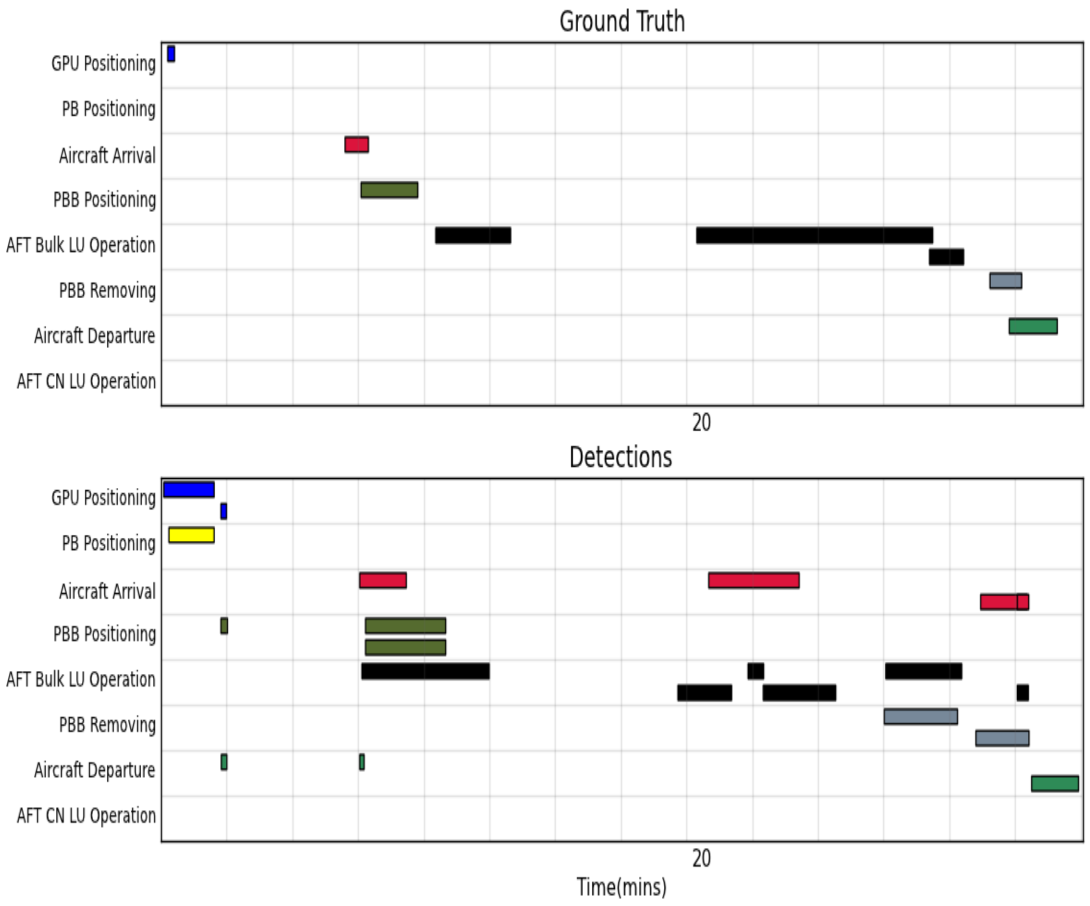


Figure 11: Recognition of events in turn-around 1 in the airport domain (best viewed in colour).

stances is long because some spatial relations that are important in the event extend beyond the deictic interval of the event.

8.3.1 Evaluating Learned Event Models Against Hand-Coded Event Models

The learned models are also evaluated by comparing them with hand-coded models. The hand-coded models were provided by domain experts using a set of domain-dependent spatial relations (Ferryman et al., 2005). In order to directly compare performance without any change in underlying representation, rather than using the RCC-5, we recomputed relational data for REMIND using the same domain-dependent primitives. The comparisons are given in Table 2. It is clear from the table that learned models have a better performance in all the event categories when compared to the performance of hand-coded models. The hand-coded models were single primitives rather than a set of primitives connected by temporal relations. These kind of models with only one single predicate have far more false positives when compared to models that have a set of spatial relations connected by

Event	#Examples	Learned (RCC-5)			Learned (d-d)			Hand-coded (d-d)		
FWD_CN_LoadUnload	7	0.86	0.08	0.15	0.14	0.50	**0.22**	0.71	0.04	0.07
GPU_Positioning	16	0.27	0.02	0.04	0.44	0.03	**0.05**	0.00	1.00	0.00
Aircraft_Arrival	15	0.36	0.01	0.02	0.07	0.01	0.01	0.07	0.05	**0.06**
AFT_Bulk_LoadUnload	29	0.72	0.20	0.31	0.59	0.27	**0.37**	0.03	0.05	0.04
PBB_Removing	15	0.36	0.12	0.18	0.26	0.14	**0.18**	0.00	1.00	0.00
Left_Refuelling	8	0.12	0.10	0.11	0.38	0.23	**0.28**	0.00	1.00	0.00
PB_Positioning	14	0.14	0.06	0.08	0.07	0.08	0.07	0.21	0.09	**0.12**
Aircraft_Departure	12	0.33	0.19	**0.24**	0.00	1.00	0.00	0.00	1.00	0.00
AFT_CN_LoadUnload	15	0.67	0.07	**0.13**	0.33	0.05	0.08	0.47	0.07	0.12
PBB_Positioning	15	0.67	0.34	**0.45**	0.26	0.20	0.22	0.40	0.07	0.12
FWD_Bulk_LoadUnload	3	1.00	1.00	**1.00**	0.00	1.00	0.00	1.00	0.02	0.04
Weighted Average				0.20			0.16			0.05

Table 2: Table comparing learned (RCC-5), learned (domain-dependent) and hand-coded models performance (domain-dependent). The first, second and third columns for each category are *recall*, *precision* and *f1* respectively. The best *f1* value in each case is presented in bold.

temporal relations. Also the hand-coded models use a very specific vehicle type in the event models which affects the performance by reducing the true positives as there is noise in the object type detection, whereas the learned models use an appropriate generalized object type to *cover* these instances.

8.3.2 Evaluating Learned Event Models with Different Spatial Relations

We also performed an evaluation to investigate the effects of different spatial relations. For comparison we used RCC-5 and domain specific relations in the airport domain. We did not use QTC relations for this domain as there were very few examples to learn from and because of the many spatial relations in the QTC spatial calculus, the patterns for events do not emerge. The results are given in Table 2. From the table it is clear that models learned using RCC-5 have a better recognition performance (mean **f1**: 0.25) when compared to the models learned using domain-dependent relations (mean **f1**: 0.13). One reason might be that RCC-5 has a better representation granularity when compared to the domain-dependent primitives. Also RCC-5 has the JEPD (*jointly exhaustive and pair-wise disjoint*) property while the domain-dependent primitives in the airport domain does not (it lacks the *pair-wise disjoint* property).

8.3.3 Evaluating on the Verbs Dataset

The framework that uses the type generalization has also been applied to the verbs dataset for the 48 verbs. Table 3 shows the *precision*, *recall* and **f1** scores for the *classification* task. For each video in the test set, all the event models are used as queries and if an event model succeeds, that particular verb is considered to be present in the video (and the variable bindings give the time of the occurrence and what objects are involved). This is compared with the ground truth to obtain the *precision* and *recall* values.

Below we provide sample rules learned for the events *Approach* and *Snatch* which cover the instances shown in Fig.9. The QTC_{L1} relations *moto* (short form for *moving towards a stationary object*), *static* and *depart* (short form for *moving away from a stationary object*) corresponds to the relations in blobs in row 2 and column 1, row 2 and column 2, row 2 and column 3 respectively in Fig.1. Also note that unlike the models learned in the Airport Dataset, there is no list of temporal intervals as an argument in the head of the rules here. This is because we just want to recognize the events in the videos in the Action Verbs dataset and the videos are too short to find the temporal extent of an event.

```
approach() :-
    moto(obj(vehicle(J)), obj(person(K)), intv(V_32,V_33)),
    static(obj(vehicle(J)), obj(person(K)), intv(V_34,V_35)),
    meets(intv(V_32,V_33), int(V_34,V_35)).
```

```
snatch() :-
    static(obj(person(J)), obj(person(K)), intv(V_24,V_25)),
    moto(obj(other(L)), obj(person(J)), intv(V_40,V_41)),
    depart(obj(other(L)), obj(person(K)), intv(V_18,V_51)),
    overlaps(intv(V_40,V_41), int(V_18,V_51)),
    during(intv(V_40,V_41), intv(V_24,V_25)),
    during(intv(V_18,V_51), intv(V_24,V_25)).
```

The proposed framework, was compared with other existing systems and the results are presented in Tables 4-7. One of the systems that we compared with, the *RedVine* system, is a supervised learning version of the framework proposed by Sridhar, Cohn and Hogg (2010). It is based on a graphical representation of relational facts, where an event is represented by a histogram of graphemes (small graphs that represent spatio-temporal interactions of the objects involved in the event) mapped into a vector space to facilitate classification. The *Stack convolutional Independent Subspace Analysis* (ScISA) (Le et al., 2011)[17] is based on pixel level flow based features which are then used to model events using a neural network. The spatio-temporal features used in this algorithm are learned in unsupervised fashion instead of using predefined features such as SIFT (Lowe, 2004), HoG (Dalal & Triggs, 2005), etc.

The evaluation dataset as provided by DARPA has a total of 2348 vignettes. It was found that some vignettes (1294) in the training set also appeared in the evaluation set. We call the dataset with 2348 vignettes the Verb Evaluation Dataset-1 (VED1) and the remaining vignettes after discarding the 1294 vignettes that appeared in training dataset VED2. Evaluation on VED1 gives interesting insights into *overfitting* and *underfitting* in the different learning frameworks that are compared. We chose two different average mechanisms (*macro* and *micro*)[18] to get an overall **f1** and *Matthews correlation coefficient* (MCC) scores over all the verbs and vignettes. True Negatives do not play a role in **f1**

17. Results using this system have been provided by Tuyen Huynh of SRI.
18. Macro-average is calculated by first calculating precision and recall for each category and then taking the average of these values, while micro-average is calculated by constructing a global contingency table and then calculating precision and recall using these sums.

Verb	precision	recall	f1
approach	0.36	0.78	0.49
arrive	0.28	0.73	0.40
attach	0.08	0.49	0.14
bounce	0.11	0.71	0.19
bury	0.05	0.57	0.10
carry	0.14	0.53	0.22
catch	0.05	0.58	0.10
chase	0.04	0.44	0.07
close	0.07	0.29	0.11
collide	0.14	0.83	0.24
dig	0.02	0.36	0.04
drop	0.08	0.47	0.14
enter	0.17	0.74	0.28
exchange	0.06	0.56	0.11
exit	0.15	0.71	0.25
fall	0.10	0.62	0.17

Verb	precision	recall	f1
flee	0.07	0.82	0.14
fly	0.06	0.42	0.11
follow	0.09	0.62	0.16
get	0.15	0.50	0.23
give	0.11	0.66	0.18
go	0.52	0.79	0.63
hand	0.10	0.66	0.17
haul	0.09	0.46	0.15
have	0.46	0.60	0.52
hit	0.14	0.64	0.23
hold	0.47	0.69	0.56
jump	0.06	0.25	0.09
kick	0.07	0.38	0.11
leave	0.29	0.76	0.42
lift	1.00	0.00	0.00
move	0.76	0.74	0.75

Verb	precision	recall	f1
open	0.10	0.77	0.17
pass	0.22	0.39	0.28
pickup	1.00	0.00	0.00
push	0.16	0.82	0.27
putdown	1.00	0.00	0.00
raise	0.33	0.63	0.44
receive	0.15	0.70	0.25
replace	0.07	0.77	0.13
run	0.10	0.82	0.18
snatch	0.11	0.51	0.19
stop	0.39	0.78	0.52
take	0.24	0.62	0.35
throw	0.06	0.39	0.10
touch	0.64	0.53	0.58
turn	0.47	0.53	0.50
walk	0.32	0.80	0.45

Table 3: Classification results per verb in the physical action verbs domain.

scores but have a considerable effect on MCC scores as MCC does not differentiate between positive and negative classes. MCC will give the same scores even if the class labels are interchanged while **f1** scores change. Note that much of the work in the literature on activity recognition use **f1** scores.

From the Tables 4-7, it is clear that ScISA has better MCC scores in all cases while REMIND has a better **f1** score on the VED2, though it has lower MCC scores than the MCC scores of the other two algorithms. Also note the drop of performance of ScISA in VED2 set when compared to VED1, whereas REMIND and RedVine have almost the same performance indicating ScISA is *overfitting* the data while REMIND and RedVine are *underfitting* the data. The reason for the very high **f1** but very low MCC score in REMIND is because of few True Negatives.

ScISA performs quite well (w.r.t. MCC score) but does not have the modelling capability our framework has since it underutilizes the temporal domain. While we do not outperform the state-of-the-art for all evaluation measures, the proposed scheme is still general, i.e., (i) it gives good interpretations of activities in video scenes; (ii) does take the temporal domain into account unlike the ScISA and therefore provides better modelling capabilities; (iii) gives high recall while precision can be improved with further post-processing and (iv) provides (elegant) logical rules which can be easily interpreted by a human observer. One major drawback of ScISA is the lack of spatio-temporal localization of the recognized event. It is suitable only for the event classification tasks (verbs dataset) but not for the event recognition tasks (airport domain). Although we do not report on localization here (owing to the short videos in the data set), deriving localization (or position) information from REMIND is trivial once the event is recognized since the intervals and objects involved are explicitly identified in the rule body.

Method	avg-prec	avg-rec	f1	MCC
REMIND	0.24	0.56	0.34	0.05
RedVine	0.32	0.24	0.28	0.16
ScISA	0.91	0.54	0.68	0.67

Table 4: Performance in the verbs domain: **VED1**, *macro-average* per verb.

Method	total prec	total rec	f1	MCC
REMIND	0.21	0.59	0.31	0.0
RedVine	0.37	0.24	0.30	0.19
ScISA	0.92	0.60	0.72	0.70

Table 5: Performance in the verbs domain: **VED1**, *micro-average* (total detection classification)

Method	avg-prec	avg-rec	f1	MCC
REMIND	0.25	0.59	0.35	0.04
RedVine	0.35	0.25	0.29	0.17
ScISA	0.49	0.20	0.29	0.21

Table 6: Performance in the verbs domain: **VED2**, *macro-average* per verb

Method	total prec	total rec	f1	MCC
REMIND	0.21	0.63	0.32	0.08
RedVine	0.40	0.26	0.31	0.20
ScISA	0.59	0.29	0.39	0.33

Table 7: Performance in the verbs domain: **VED2**, *micro-average* (total detection classification)

8.4 Experimental Results and Evaluation for $\mathcal{IIA}$

The $\mathcal{IIA}$ framework has been evaluated on both the airport and verb datasets. We use *Hyprolog*, a logic programming framework capable of abductive inference (Christiansen & Dahl, 2005).

8.4.1 Embedding Spatial Theory for the Airport Domain

For the airport domain, we have encoded the RCC-5 spatial theory Σ_{space} into the framework that contains the conceptual neighbourhood graph, the JEPD relationships and the composition theorems of the spatial relations used as follows:

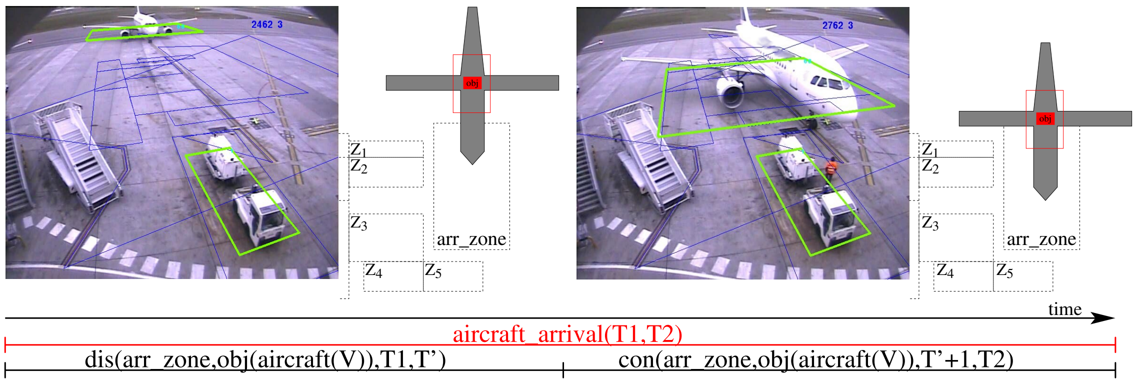
```
% Sample JEPD constraints (P1 - P2) for RCC-5
dc(X, Y, T) , touch(X, Y, T)                      <=> fail.
dc(X, Y, T1), touch(X, Y, T2), during(T1, T2)    <=> fail.
dc(X, Y, T1), touch(X, Y, T2), during(T2, T1)    <=> fail.
dc(X, Y, T1), touch(X, Y, T2), overlaps(T1, T2) <=> fail.
dc(X, Y, T1), touch(X, Y, T2), overlaps(T2, T1) <=> fail.

% Conceptual Neighbourhood constraints (P3) for RCC-5
dc(X, Y, T1), in(X, Y, T2), meets(T1, T2) <=> fail.
in(X, Y, T1), dc(X, Y, T2), meets(T1, T2) <=> fail.

% Sample Composition Theorem (P4) for RCC-5
in(X, Y, T1), dc(Y, Z, T2), touch(X, Z, T3), during(T2, T1),
                                    during(T3, T2) <=> fail.
```

The JEPD and the CND property constraints forbid the abduction of facts which contradict the spatial theory thus avoiding physically impossible scenarios and this also helps abduction to complete in reasonable time.
To explain our approach, consider the following fragments of actually occurring datasets (`Ex:1` - `Ex:4`) for the event *Aircraft Arrival*:

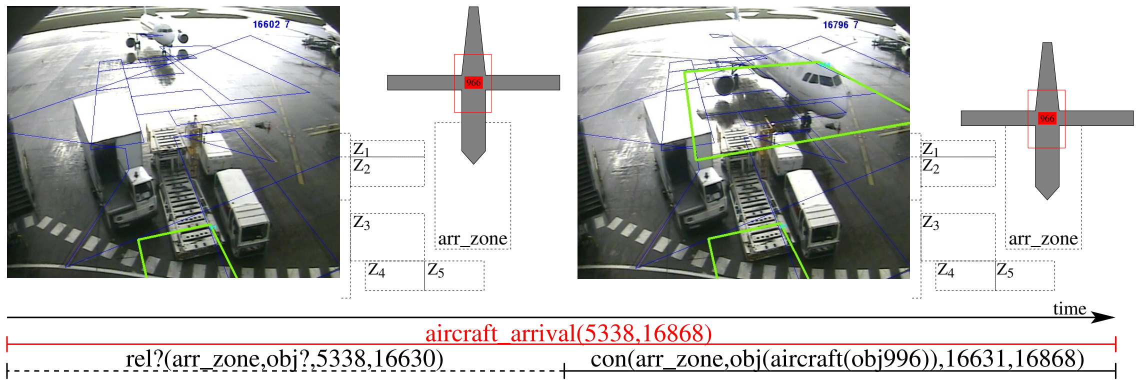
```
Ex:1  dc(arr_zone,obj(aircraft(obj45)),intv(6661,7137))
      touch(arr_zone,obj(aircraft(obj45)),intv(7138,29114))
      touch(arr_zone,obj(veh(light_veh(gpu(obj54)))),intv(7154,8161))
      dc(arr_zone,obj(veh(heavy_veh(loader(obj2)))),intv(749,30380))

Ex:2  dc(arr_zone,obj(aircraft(obj68)),intv(2342,2663))
      touch(arr_zone,obj(aircraft(obj68)),intv(2664,29524))

Ex:3  dc(arr_zone,obj(veh(light_veh(trolley(obj0)))),intv(285,21494))
      touch(arr_zone,obj(aircraft(obj41)),intv(4458,32404))
      touch(arr_zone,obj(veh(light_veh(trolley(obj2)))),intv(1712,32405))

Ex:4  dc(arr_zone,obj(aircraft(obj33)),intv(2435,6987))
      touch(arr_zone,obj(veh(heavy_veh(loader(obj27)))),intv(2197,2310))
      dc(arr_zone,obj(veh(heavy_veh(loader(obj27)))),intv(2311,2645))
```

We obtain the following model for *Aircraft Arrival* event learned by the ILP approach from the first two examples of the given examples with *arr_zone* denoting a specific zone on the apron and any T_i denotes a time point. Each fact has two time points indicating the start and end of an interval in which the spatio-temporal fact holds.

```
 aircraft_arrival([intv(T1,T2), intv(T3,T4)]) :-
     dc(arr_zone, obj(aircraft(V)), intv(T1,T2)),
     touch(arr_zone, obj(aircraft(V)), intv(T3,T4)),
     meets(intv(T1,T2), intv(T3,T4)).
```


2462 3
obj
Z1
Z2
Z3
Z4
Z5
arr_zone
2762 3
obj
time
aircraft_arrival(T1,T2)
dis(arr_zone,obj(aircraft(V)),T1,T')
con(arr_zone,obj(aircraft(V)),T'+1,T2)

(a) Spatial primitive based event modelling


16602 7
996
Z1
Z2
Z3
Z4
Z5
arr_zone
16796 7
time
aircraft_arrival(5338,16868)
rel?(arr_zone,obj?,5338,16630)
con(arr_zone,obj(aircraft(obj996)),16631,16868)

(b) Narrative completion (of data video) and previously learned model

Figure 12: $\mathcal{IIA}$ Scenario and Narrative Completion; E.g., *aircraft_arrival*

This rule states that an *aircraft arrival* takes place if there is some interval in which an aircraft is disconnected from *arr_zone* directly followed by an interval, i.e., *meets*, where the same aircraft is connected to *arr_zone*. This model does not cover any other examples apart from Ex:1 and Ex:2. Ex:3 has a missing *dc* relation related to the aircraft whereas Ex:4 has a missing *touch* relation (Fig. 12b). These represent the typical data corruption at a higher level because of tracking error at lower level video processing at different stages in the video.

8.4.2 Narrative Completion in the Airport Domain

Multiple explanations are interesting as they give several possible scenarios that are all consistent with the narrative. For example, consider Ex:4 where a *touch* fact related to aircraft arrival event is missing in the narrative. This happens when the vision algorithm fails to detect the aircraft when it is coming towards the parking zone as such a big object changes the light conditions in the scene. The abduction system comes up with two explanations (as shown in the following sample interactive run of the system), one filling the missed fact that is consistent with the narrative and the background knowledge and constraints. Another explanation is using a hypothetical object (_G41673) that is not present in the database.

This explanation is more expensive than the first explanation, so the system chose the first explanation.

```
%A small narrative with three observations (The 'touch' fact
%is missing, this happens, when the vision algorithm fails
%to detect the aicraft when it is close: Approximate
%interval can be specified for aircraft-arrival query
%dc(arr_zone,obj(aircraft(obj33)),intv(2435,6987))
%touch(arr_zone,obj(veh(heavy_veh(loader(obj7)))),intv(2197,2310))
%dc(arr_zone,obj(veh(heavy_veh(loader(obj7)))),intv(2311,2645))
?- aircraft_arrival(intv(2000,12000)).
touch(arr_zone,obj(aircraft(obj33)),intv(6988,7988))
true ;
dc(arr_zone,obj(aircraft(_G41673)),intv(2435,6987))
touch(arr_zone,obj(aircraft(_G41673)),intv(6988,7988))
true ; false.
```

With narrative completion, it is possible to cover all the examples given above, with one single Aircraft Arrival model learned. This avoids learning spurious rules to cover these corrupted examples thus giving us compact and semantically meaningful models.
To evaluate our approach, we compare the rules learned using only induction and rules learned using the $\mathcal{IIA}$ algorithm. The first column in Table 8 shows the events that we considered for the experiments, the second column shows the number of instances of that particular event in the 15 turnarounds. The third column shows the number of rules learned by using only ILP while the fourth column shows the results using the $\mathcal{IIA}$ algorithm while fifth column shows the number of examples that were not covered by the induced rules but were explained using abduction and hence no rules learned from them. By interleaving induction and abduction, we were able to avoid learning spurious rules as shown by the results. For most classes, the number of rules are reduced by about 50% and the overall performance is also increased. We also observed that the rules which had been previously learned for the examples now covered by abduction did not semantically correspond to the events.

8.4.3 Evaluating $\mathcal{IIA}$ on the Verbs Dataset

For the verbs domain, we encoded the spatial theory for QTC_{L1} spatial calculi. Though we also used domain-dependent primitive events for this domain besides QTC_{L1}, we did not encode any spatial theory for these relations as it is not well defined. For example, the domain-dependent primitives for this domain are neither *jointly exhaustive* nor *pair-wise disjoint*. We also avoided abducing explanations with these relations by not including these relations in the list of *abducibles*. 10-fold cross-validation was used for evaluation for verbs dataset. Since each video is short in duration with around 200 frames, we used classification instead of recognition. From Table 9 it is clear that using abduction reduces the number of rules in event model thereby giving a more compact model. In the verbs dataset results, there is no considerable change in performance as this is a classification task rather than a recognition task. The main performance increase with $\mathcal{IIA}$ during inference comes when

Airport Events	#pos	●	□	◇	RoI	PoI	RIA	PIA
FWD_CN_LoadUnload	5	2	1	2	0.8	0.3	0.8	0.4
GPU Positioning	15	5	3	4	1	0.2	1	0.4
Aircraft Arrival	15	5	2	5	0.38	0.26	0.33	0.32
Aircraft Departure	15	5	2	7	0.8	0.15	0.71	0.26
AFT_Bulk_LoadUnload	12	5	2	4	0.63	0.43	0.63	0.65
Left Refuelling	6	2	1	2	0.66	0.5	0.66	0.55
PB Positioning	15	4	3	2	0.33	0.34	0.33	0.42
AFT_CN_LoadUnload	7	3	1	3	0.57	0.4	0.57	0.51
PBB Positioning	15	4	3	2	1	0.57	1	0.62
PBB_Removing	15	5	2	5	0.54	0.23	0.54	0.31
FWD_Bulk_LoadUnload	3	2	1	1	1	1	1	1

● Num of rules with only Induction □ Num of rules with $\mathcal{IIA}$ ◇ avg num of examples covered by abd

Table 8: Airport domain $\mathcal{IIA}$ results averaged from all iterations in leave-one-out testing. RoI, PoI - recall and precision with only induction: RIA, PIA - recall and precision using $\mathcal{IIA}$.

Verb Events	#pos	●	□	◇	RoI	PoI	RIA	PIA
Approach	584	12	5	45	0.73	0.12	0.74	0.12
Arrive	8	2	1	2	0.50	0.05	0.50	0.05
Attach	48	6	3	12	1.00	0.14	1.00	0.17
Bounce	22	2	2	0	0.95	0.06	0.95	0.08
Catch	201	7	4	31	0.59	0.11	0.56	0.11
Chase	108	11	7	19	0.59	0.08	0.57	0.08
Collide	101	6	4	14	0.98	0.16	0.98	0.18
Dig	140	10	7	21	0.96	0.38	0.96	0.39
Drop	44	2	2	0	1.00	0.16	1.00	0.16
Exchange	18	6	3	4	0.40	0.03	0.40	0.03
Fall	134	8	5	18	0.92	0.35	0.90	0.35
Give	552	27	20	54	0.94	0.56	0.94	0.60
Jump	150	6	4	14	0.98	0.13	0.98	0.13
Kick	48	4	3	6	1.00	0.15	1.00	0.15
Leave	116	10	4	34	0.67	0.20	0.67	0.22
Lift	78	8	5	17	0.67	0.24	0.67	0.24
Pass	76	8	4	13	0.87	0.10	0.87	0.12
Pickup	40	6	4	8	0.81	0.13	0.81	0.16
Run	76	7	5	7	0.57	0.12	0.57	0.12
Throw	26	3	2	5	0.67	0.11	0.67	0.11

● Num of rules with only Induction □ Num of rules with $\mathcal{IIA}$ ◇ avg num of examples covered by abd

Table 9: *Verbs* dataset $\mathcal{IIA}$ results averaged from all iterations in 10-fold cross-validation testing. RoI, PoI - recall and precision with only induction: RIA, PIA - recall and precision using $\mathcal{IIA}$.

there is reduction of false positives because of fewer rules and in recognition there is a high possibility of multiple rules firing in the test data thereby giving many false positives. In classification, this is not the case, as once a vignette is classified as being a member of a particular event class by a rule, the classification by other rules of the event class does not affect the overall outcome for that vignette.

9. Limitations and Future Work

The models used by REMIND are *local*, i.e., without the context of a wider activity model that could be used to filter out recognized instances thereby increasing performance. For example, in some turn-arounds in the airport domain, the Aircraft Departure event is sometimes recognized even before Aircraft Arrival is recognized resulting in false positives for Aircraft Departure. Another limitation of the learned models is the lack of representation of duration for events. Many recognized event instances are rejected by the system because the temporal extent of the recognized instances is long and fails our criteria of 20% overlap with the ground truth. These can be reduced by learning a global model (Greenall, Cohn, & Hogg, 2011) that constrains the ordering of events such that Aircraft Departure detections can happen only after Aircraft Arrival. The activity models can also represent expected duration of events, the temporal separation of events and the number of occurrences.

The framework is sensitive to the initial example selected to start the learning procedure. The induction system used is based on the algorithm that uses a *bottom clause* (Muggleton, 1995) constructed from the selected example to guide the refinement of the hypothesis while searching in the lattice. Hence it is possible it might select a corrupted example initially and this might affect the whole induction process. This is a typical problem in machine learning and there are several ways to avoid this. One promising approach that we followed in this work is to repeat the learning with different examples chosen randomly as the starting point then selecting the iteration that gives the minimum number of rules.

Another limitation of the framework is its dependency on the tracking of objects as it uses the interactions of objects to model events. Challenging scenarios for object tracking pose limitations for the current framework. The current framework is not probabilistic, i.e., neither the input data nor the learned models are probabilistic. One direction of future work is to extend the framework using statistical relational learning so that it can use soft evidence and learn more robust probabilistic relational models. Since the current framework does not handle hierarchies of events, the framework could be extended to handle hierarchical composition of events. One possible approach for this is to learn models of events at a particular layer using the events at lower layers as primitives.

10. Conclusion

In this paper, we have proposed a supervised relational learning framework, and an extension using abduction, to learn event models from complex videos. Such event models can be used to recognize event instances from unseen videos. We presented a Type Refinement operator that exploits the object type hierarchy in the domain to search for better hypotheses and also proved that it is an *optimal* refinement operator. We presented an empirical evaluation of the proposed framework on two real world video data sets and the results are encouraging, showing that the framework can be effectively used in real world systems for event recognition in various domains. We also showed that the proposed framework has better generalization capabilities and performance when compared to the state-of-the-art systems in event modelling. Finally, we note that although we have focused on learning from video data here, in fact the approach would also be suitable for learning from other data sources which provide tracks of interacting moving objects (e.g. from GPS streams).

Acknowledgements

We thank colleagues in the CO-FRIEND, RACE, STRANDS and VIGIL projects consortia for their valuable inputs to this research, and the respective funding from EU Framework 7 (FP7-ICT-214975, FP7-ICT-27752, FP7-ICT-600623) and DARPA (W911NF-10-C-0083). Also financial support by the Deutsche Forschungsgemeinschaft in the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition project R3-[Q-Shape] is gratefully acknowledged.

Appendix A. Proof of Optimality of Type Refinement Operator ρ_τ

Let T be the type hierarchy tree with set of nodes T_V, and set of leaf nodes T_L, i.e. most specific types ($T_L \subset T_V$) and τ_r be the root of the tree (most generic type). A type at a parent node τ in T is more generic than the types τ_i at its children nodes and we write $\tau \sqsupset \tau_i$.

Let g be a function, $g : T_V \rightarrow T_V$, that maps a child node to its immediate parent. The function g can be considered as a generalizing operator that generalizes a type to its nearest generic type. g can be applied as long as $\tau_i \neq \tau_r$

Let S_{τ_i} be an ordered (from most-specific to most general) set of all possible generalizations of τ_i including τ_i.

$$S_{\tau_i} = \{\tau_i, g(\tau_i), g(g(\tau_i)), \dots, \tau_r\}$$

For any set of types $\{\tau_1, \tau_2, \dots, \tau_n\}$, we can define corresponding sets $S_{\tau_1}, S_{\tau_2}, \dots, S_{\tau_n}$. Let $\{h_1, h_2, \dots, h_m\}$ be the set of types[19] in the clause C where $\{h_1, h_2, \dots, h_m\} \subseteq T_V$. For $\{h_1, h_2, \dots, h_m\}$, we can define $S_{h_1}, S_{h_2}, \dots, S_{h_m}$. We can make the set $\{h_1, h_2, \dots, h_m\}$ more generic by applying g (one or more times) on any arbitrarily selected subset of types one type at a time.

The Cartesian product $S_{h_1} \times S_{h_2} \times \dots \times S_{h_m}$ is the set of tuples where each tuple is a possible generalization of $\{h_1, h_2, \dots, h_m\}$.

Let l be a function mapping a non-leaf type node to an integer that specifies how many times g was applied to the original leaf node to obtain the non-leaf node[20] in T, $l : T_V \rightarrow N$.

Using l, we can generate a new set N_{h_i} from S_{h_i} by replacing τ_i by $l(\tau_i)$. Now we can generate a new Cartesian product $N_{h_1} \times N_{h_2} \times \dots \times N_{h_m}$.

Example .1. Let (τ_1, τ_2, τ_3) be the set of types from a clause C and let the type hierarchy be as given in Fig.13. Then we can define $S_{\tau_1}, S_{\tau_2}, S_{\tau_3}$ and $N_{\tau_1}, N_{\tau_2}, N_{\tau_3}$ as follows and the tree representation of the Cartesian products $S_{\tau_1} \times S_{\tau_2} \times S_{\tau_3}$ and $N_{\tau_1} \times N_{\tau_2} \times N_{\tau_3}$ are given in Fig.14 and Fig.15 respectively.

19. If we consider the list of types from a clause C where some types may be repeated because of some arguments have the same type, the results in this appendix are still valid.
20. Note that in general, a non-leaf node can be obtained from any of the leaf nodes that are its descendants but a unique leaf node can be obtained if we store the original leaf node that is generalized using g to get the non-leaf type node.

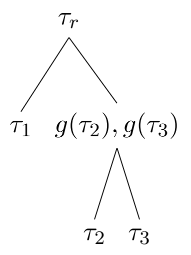


Figure 13: An example type hierarchy.

$$\begin{aligned}
S_{\tau_1} &= \{\tau_1, g(\tau_1)\}\\
S_{\tau_2} &= \{\tau_2, g(\tau_2), g(g(\tau_2))\}\\
S_{\tau_3} &= \{\tau_3, g(\tau_3), g(g(\tau_3))\}\\
N_{\tau_1} &= \{0, 1\}\\
N_{\tau_2} &= \{0, 1, 2\}\\
N_{\tau_3} &= \{0, 1, 2\}
\end{aligned}$$

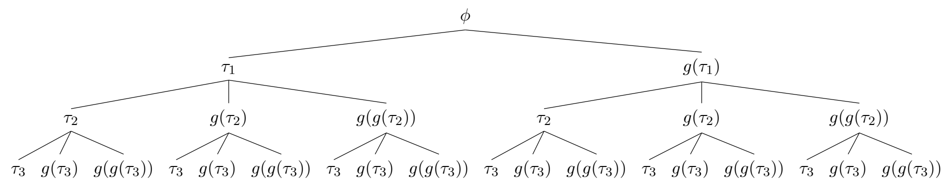


Figure 14: Representing the Cartesian product $S_{\tau_1} \times S_{\tau_2} \times S_{\tau_3}$ as a tree. The root is empty and the next layer corresponds to S_{τ_1} and so on. Note that $g(\tau_1) = g(g(\tau_2)) = g(g(\tau_3)) = \tau_r$. Each path in the tree to a leaf is a possible generalization of (τ_1, τ_2, τ_3) – the leftmost path being the null generalization, i.e. (τ_1, τ_2, τ_3).

Definition .2. (Type Substitution, θ_τ) A type substitution θ_τ is a set $\{h_1/\gamma_1, h_2/\gamma_2, \ldots, h_n/\gamma_n\}$ where each h_i is the type of a subset of variables in the clause C and γ_i is the immediate generic type of h_i (parent node of h_i in the tree T). We say γ_i is substituted for h_i in the clause. The set $\{h_1, h_2, \ldots, h_n\}$ is called the domain of θ_τ, denoted $dom(\theta_\tau)$ and the set $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ is called range of θ_τ, denoted $rng(\theta_\tau)$.

A type substitution is used to generalize the type of a subset of variables in the clause.

Definition .3. (Most Generic Type Substitution, θ_{τ_r}) A most generic type substitution is a type substitution whose range is the set $\{\tau_r\}$ where τ_r the root of type hierarchy tree T.

A most generic type substitution is used to check if two clauses are *structurally equivalent* (Def:4) by substituting the types of all variable by τ_r. Note that every clause C has a unique most generic type substitution, $\theta^C_{\tau_r}$, whose domain is the set of all types in C and range is the set $\{\tau_r\}$.

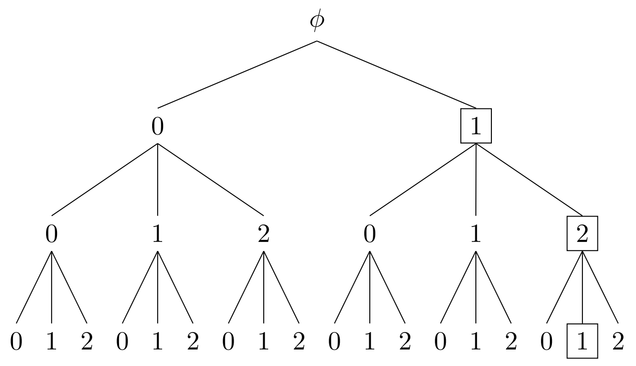


Figure 15: Representing the Cartesian product $N_{\tau_1} \times N_{\tau_2} \times N_{\tau_3}$ as a tree. The root is empty and the next layer corresponds to N_{τ_1} and so on. Each path to a leaf represents a possible generalization of (τ_1, τ_2, τ_3). Each generalization is obtained by following the path from root to leaf and generalizing the type at each layer the number of times indicated by the node value. For example, the highlighted sequence (1,2,1) corresponds to the generalization $(g(\tau_1), g(g(\tau_2)), g(\tau_3))$. This can be obtained by generalizing τ_1 once and generalizing τ_2 twice and generalizing τ_3 once. If a top to bottom order (left to right in case of tuples) is followed, this is the only unique way to achieve the generalization $(g(\tau_1), g(g(\tau_2)), g(\tau_3))$ from (τ_1, τ_2, τ_3).

Definition .4. (Structurally Equivalent, $\asymp$) Two clauses, C and C' are structurally equivalent, denoted $C \asymp C'$, if $C\theta^{C}_{\tau_r} \equiv C'\theta^{C'}_{\tau_r}$.

A clause C is structurally equivalent to all the clauses obtained by replacing a subset of types of variables in C by any of their generalizations.

Definition .5. (Generic Order w.r.t. types, $\preceq_{\mathcal{T}}$) A clause C is said to be more general w.r.t. type than another clause C', denoted $C \preceq_{\mathcal{T}} C'$, iff $C \asymp C'$ and the set of types of C are correspondingly more generic than the set of types of C'.

Definition .6. (Type-Refinement Operator) Let ς be a clausal language, T a type hierarchy and C a clause in ς. ς^{T}_{C} is a subset of ς defined over T and C where a clause $C' \in \varsigma^{T}_{C}$ is structurally equivalent to C, i.e., $C \asymp C'$. Let $\preceq_{\mathcal{T}}$ be the subsumption order as defined above. The Type-Refinement operator $\rho_{\mathcal{T}}$ for $\langle \varsigma^{T}_{C}, \preceq_{\mathcal{T}} \rangle$ is a function such that $\rho_{\mathcal{T}}(C) \subseteq \{D | D \preceq_{\mathcal{T}} C\}$.

- A *One-step type refinement* of C is defined when $\rho_{\mathcal{T}}$ is applied only once, i.e. $\rho^{1}_{\mathcal{T}} = \rho_{\mathcal{T}}(C)$. An *n-step type refinement* can be defined similarly, i.e. $\rho^{n}_{\mathcal{T}} = \{D \mid \exists E, E \in \rho^{n-1}_{\mathcal{T}}(C)$ such that $D \in \rho_{\mathcal{T}}(E)\}$. The set of all type *refinements* on C is given by $\rho^{*}_{\mathcal{T}}(C) = \rho^{1}_{\mathcal{T}}(C) \cup \rho^{2}_{\mathcal{T}}(C) \cup \ldots$.
- $\rho_{\mathcal{T}}$ is *locally finite* if for every $C \in \varsigma$, $\rho_{\mathcal{T}}(C)$ is finite and computable.
- $\rho_{\mathcal{T}}$ is *proper* if for every $C \in \varsigma, \rho_{\mathcal{T}}(C) \subseteq \{D | D \prec_{\mathcal{T}} C\}$.
- $\rho_{\mathcal{T}}$ is *complete* if for every $C, D \in \varsigma$ such that $D \prec_{\mathcal{T}} C$, there is an $E \in \rho^{*}_{\mathcal{T}}(C)$ such that $D \sim E$ (i.e. D and E are equivalent in the $\preceq_{\mathcal{T}}$ order).

- ρ_τ is *weakly complete* for $\langle \varsigma_C^T, \preceq_\tau \rangle$ if $\rho_\tau^*(C) = \varsigma_C^T$.
- ρ_τ is *non-redundant* if for every $C, D, E \in \varsigma, E \in \rho_\tau^*(C)$ and $E \in \rho_\tau^*(D)$ implies $C \in \rho_\tau^*(D)$ or $D \in \rho_\tau^*(C)$
- ρ_τ is *ideal* if it is locally finite, proper and complete.
- ρ_τ is *optimal* if it is locally finite, non-redundant and weakly complete.

The type refinement operator selects a type h_i from the set $\{h_1, h_2, \ldots, h_m\}$ for C and applies the type generalizing operator to the type and the resulting set $\{h_1, h_2, \ldots, g(h_i), \ldots, h_m\}$ is used for substitution in C to obtain a more generic clause C' with respect to type, i.e., $C' \preceq_\tau C$. The type refinement operator follows a left to right order in generalizing types to avoid generating redundant clauses, i.e., if a type at position i is generalized then in the next step of refinement of h' no type at position j, $j < i$ is selected for generalizing.

Theorem .7. *ρ_τ is locally finite*

Proof. Let T be the type hierarchy tree and T_V be the set of all nodes in T and $T_C = \{h_1, h_2, \ldots, h_m\}$ be the set of types in the clause C where $T_C \subset T_V$. Let g be the type generalizing operator and ρ_τ be the type refinement operator. ρ_τ operates on C by selecting a type from the set $\{h_1, h_2, \ldots, h_m\}$ and generalizing it by applying g. There are only $|T_C|$ possibilities to select from and for each possible type selected there is only one possible generalization, as a type only has one parent in the tree T. Also each type can only be generalized finite number of times (i.e., until it becomes τ_r). Hence the number of possible refinements, i.e., $|\rho_\tau(C)|$ is finite making ρ_τ *locally finite.* □

Theorem .8. *ρ_τ is weakly complete*

Proof. For any given clause C with set of types $\{h_1, h_2, \ldots, h_m\}$ and T as defined above. Let X_1 be the set of substitutions $\{\theta^1_{\tau,1}, \theta^1_{\tau,2}, \ldots\}$ obtained from one-step type refinement such that $C'_i = C\theta^1_{\tau,i}$ and $C'_i \in \rho^1_\tau(C)$. Let $X = X_1 \cup X_2 \cup \ldots$ where X_2 is the set of substitutions obtained from two-step type refinement and so on and $\{\gamma^1_{i1}, \gamma^1_{i2}, \ldots, \gamma^1_{im}\}$ be the $rng(\theta^1_{\tau,i})$.

Let Γ^1 be the set of tuples $\{(\gamma^1_{11}, \gamma^1_{12}, \ldots, \gamma^1_{1m}), \ldots, (\gamma^1_{i1}, \gamma^1_{i2}, \ldots, \gamma^1_{im}), \ldots\}$, i.e. Γ^1 is the set of tuples where each tuple represents a possible type refinement of $\{h_1, h_2, \ldots, h_m\}$ through one-step type refinement and let $\Gamma = \Gamma^1 \cup \Gamma^2 \cup \ldots$.

It is easy to observe that any tuple $P \in \Gamma$ is also a tuple in the Cartesian product $S_{h_1} \times S_{h_2} \times \ldots \times S_{h_m}$. In fact there is an exact one to one matching for the members of Γ and members of $S_{h_1} \times S_{h_2} \times \ldots \times S_{h_m}$. It is easy to obtain each member tuple, say P where $P = (\gamma_1, \gamma_2, \ldots, \gamma_m)$ of the Cartesian product by generalizing $\{h_1, h_2, \ldots, h_m\}$. A type h_i is generalized until it is equal to γ_i before moving to the next type on the immediate right of h_i. In this way all possible type generalizations of $\{h_1, h_2, \ldots, h_m\}$ are reachable from $\{h_1, h_2, \ldots, h_m\}$, i.e., all possible type generalized clauses are reachable from C, hence ρ_τ is *weakly complete.* □

Theorem .9. *ρ_τ is non-redundant*

Proof. Let $N_{h_1} \times N_{h_2} \times \ldots \times N_{h_m}$ be as defined previously for a set of types $\{h_1, h_2, \ldots, h_m\}$ from a clause C. This Cartesian product can be represented as a tree where the root is empty and the next level is the elements from N_{h_1} and so on. Each path to a leaf represents a possible generalization of $\{h_1, h_2, \ldots, h_m\}$. Each generalization is obtained by following the path from root to leaf and generalizing the type at each layer the number of times indicated by the node value. If a top to bottom order (left to right in case of tuples) is followed, there is a unique way to obtain the generalization that path generates. Hence $\rho_\mathcal{T}$ is non-redundant. $\square$

Example .10. For example in Fig.15, the sequence (1,2,1) in bold corresponds to the generalization $(g(\tau_1), g(g(\tau_2)), g(\tau_3))$. This can obtained by generalizing τ_1 once and generalizing τ_2 twice and generalizing τ_3 once. If a top to bottom order is followed, this is the only unique way to achieve the generalization $(g(\tau_1), g(g(\tau_2)), g(\tau_3))$ from (τ_1, τ_2, τ_3).

Theorem .11. $\rho_\mathcal{T}$ *is optimal*

Proof. Since the type refinement operator is *locally finite*, *weakly complete* and *non-redundant*, it is *optimal.* $\square$

References

Albanese, M., Moscato, V., Picariello, A., Subrahmanian, V., & Udrea, O. (2007). Detecting stochastically scheduled activities in video. In *Proceedings of the International Joint Conference on Aritificial Intelligence (IJCAI)*, pp. 1802–1807.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, *26*, 832–843.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bhatt, M., & Loke, S. (2008). Modelling dynamic spatial systems in the situation calculus. *Spatial Cognition & Computation*, *8*(1-2), 86–130.

Blockeel, H., De Raedt, L., Jacobs, N., & Demoen, B. (1999). Scaling up Inductive Logic Programming by learning from interpretations. *Data Mining and Knowledge Discovery*, *3*(1), 59–93.

Bundy, A., Byrd, L., & Mellish, C. (1985). Special-purpose, but domain-independent, inference mechanisms. In *Progress in Artificial Intelligence*, pp. 93–111. London: Ellis Horwood.

Chen, J., Cohn, A. G., Liu, D., Wang, S., Ouyang, J., & Yu, Q. (2015). A survey of qualitative spatial representations. *The Knowledge Engineering Review*, *30*, 106–136.

Christiansen, H., & Dahl, V. (2005). HYPROLOG: A new logic programming language with assumptions and abduction. *Logic Programming*, 159–173.

Cohn, A. G. (1989). Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, *3*(2), 89–128.

Cohn, A. G., Hogg, D. C., Bennett, B., Devin, V., Galata, A., Magee, D. R., Needham, C., & Santos, P. (2006). Cognitive vision: integrating symbolic qualitative representations with computer vision.. Vol. 3948 of *LNCS*, chap. 14, pp. 221–246. Springer.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 886–893.

Dubba, K. S., Bhatt, M., Dylla, F., Hogg, D. C., & Cohn, A. G. (2012). Interleaved inductive-abductive reasoning for learning complex event models. In *Inductive Logic Programming*, pp. 113–129. Springer.

Dubba, K. S., Cohn, A. G., & Hogg, D. C. (2010). Event model learning from complex videos using ILP. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, Vol. 215, pp. 93–98.

Fern, A., Givan, R., & Siskind, J. (2002). Specific-to-general learning for temporal events with application to learning event definitions from video. *Journal of Artificial Intelligence Research*, *17*, 379–449.

Ferryman, J., Borg, M., Thirde, D., Fusier, F., Valentin, V., Bremond, F., Thonnat, M., Aguilera, J., & Kampel, M. (2005). Automated scene understanding for airport aprons. *LNCS-3809, Springer Verlag*, *3809*, 593.

Freksa, C. (1991). Conceptual neighborhood and its role in temporal and spatial reasoning. In Singh, M., & Travé-Massuyès, L. (Eds.), *Decision Support Systems and Qualitative Reasoning*, pp. 181–187. North-Holland, Amsterdam.

Ghahramani, Z. (1998). Learning Dynamic Bayesian networks. *Adaptive Processing of Sequences and Data Structures*, 168–197.

Greenall, J., Cohn, A. G., & Hogg, D. C. (2011). Temporal structure models for event recognition. *British Machine Vision Conference (BMVC)*.

Gupta, A., Srinivasan, P., Shi, J., & Davis, L. (2009). Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2004–2011.

Hakeem, A., Sheikh, Y., & Shah, M. (2004). CASE^{E}: A hierarchical event representation for the analysis of videos. In *Proceeding of the National Conference on Artificial Intelligence (AAAI)*, pp. 263–268.

Hartley, R., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision* (Second edition). Cambridge University Press.

Hazarika, S. M., & Cohn, A. G. (2002). Abducing qualitative spatio-temporal histories from partial observations. In *International Conference on Principles Of Knowledge Representation And Reasoning*, pp. 14–25.

Hoogs, A., & Perera, A. G. A. (2008). Video activity recognition in the real world. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1551–1554.

Ivanov, Y., & Bobick, A. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *22*(8).

Jiang, Z., Lin, Z., & Davis, L. S. (2010). A tree-based approach to integrated action localization, recognition and segmentation. *Third Workshop on Human Motion (in conjuntion with ECCV)*.

Kakas, A., Kowalski, R., & Toni, F. (1992). Abductive logic programming. *Journal of Logic and Computation*, *2*(6), 719.

Kakas, A., & Riguzzi, F. (2000). Abductive concept learning. *New Generation Computing*, *18*(3), 243–294.

Könik, T., & Laird, J. (2006). Learning goal hierarchies from structured observations and expert annotations. *Machine Learning*, *64*(1), 263–287.

Laptev, I. (2005). On space-time interest points. *International Journal of Computer Vision*, *64*(2), 107–123.

Laptev, I., & Pérez, P. (2007). Retrieving actions in movies. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8.

Le, Q., Zou, W., Yeung, S., & Ng, A. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3361–3368. IEEE.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, *28*(1), 89–116.

Medioni, G., Cohen, I., Brémond, F., Hongeng, S., & Nevatia, R. (2001). Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *23*(8), 873–889.

Miller, R., & Shanahan, M. (1994). Narratives in the Situation Calculus. *Journal of Logic and Computation*, *4*(5), 513–530.

Morariu, V. I., & Davis, L. S. (2011). Multi-agent event recognition in structured scenarios.. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3289–3296.

Morariu, V. I., Harwood, D., & Davis, L. S. (2013). Tracking people's hands and feet using mixed network and/or search.. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Moyle, S. (2003). Using theory completion to learn a robot navigation control program. *Proceedings of the International Conference on ILP*, 182–197.

Moyle, S., & Muggleton, S. (1997). Learning programs in the Event Calculus. *LNAI-1297, Springer-Verlag*, 205–212.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*(3&4), 245–286.

Muggleton, S., & Bryant, C. H. (2000). Theory completion using inverse entailment. In *Proceedings of the International Conference on ILP*, pp. 130–146, UK. Springer-Verlag.

Needham, C., Santos, P., Magee, D., Devin, V., Hogg, D., & Cohn, A. (2005). Protocols from perceptual observations. *Artificial Intelligence*, *167*(1-2), 103–136.

Nevatia, R., Hobbs, J., & Bolles, B. (2004). An ontology for video event representation. In *Computer Vision and Pattern Recognition Workshop (CVPRW-04)*, pp. 119–119. IEEE.

Nienhuys-Cheng, S., & De Wolf, R. (1997). *Foundations of Inductive Logic Programming*, Vol. 1228. Springer Verlag.

Oh, S., Hoogs, A., Perera, et al. (2011). A large-scale benchmark dataset for event recognition in surveillance video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3153–3160.

Poole, D., Goebel, R., & Aleliunas, R. (1987). Theorist: A logical reasoning system for defaults and diagnosis. In *The Knowledge Frontier*, pp. 331–352.

Quinlan, J., & Cameron-Jones, R. (1993). FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning (ECML)*, pp. 1–20.

Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.

Rabiner, L. (1989). A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Randell, D. A., Cui, Z., & Cohn, A. (1992). A spatial logic based on regions and connection. In *Proceedings of the International Conference on Knowledge Representation and Reasoning*, pp. 165–176. Morgan Kaufmann.

Ryoo, M. S., & Aggarwal, J. K. (2009). Semantic representation and recognition of continued and recursive human activities. *International Journal of Computer Vision*, *82*(1), 1–24.

Ryoo, M., & Aggarwal, J. (2011). Stochastic representation and recognition of high-level group activities. *International Journal of Computer Vision*, *93*(2), 183–200.

Sridhar, M., Cohn, A. G., & Hogg, D. C. (2010). Unsupervised learning of event classes from video. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1631–1638.

Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., & Muggleton, S. (2006). Application of abductive ILP to learning metabolic network inhibition from temporal data. *Machine Learning*, *64*(1), 209–230.

Tamaddoni-Nezhad, A., & Muggleton, S. (2009). The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause. *Machine learning*, *76*(1), 37–72.

Van de Weghe, N., Cohn, A., De Tre, G., & De Maeyer, P. (2006). A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. *Control and Cybernetics*, *35*(1), 97.

Veeraraghavan, H., Papanikolopoulos, N., & Schrater, P. (2007). Learning dynamic event descriptions in image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–6.

Vu, V.-T., Bremond, F., & Thonnat, M. (2003). Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, Vol. 3, pp. 1295–1300.

Walther, C. (1985). A mechanical solution of Schubert's Steamroller by many-sorted resolution. *Artificial Intelligence*, *26*(2), 217–224.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys (CSUR)*, *38*(4), 13.

"YouTube" (2015) `http://www.youtube.com/yt/press/statistics.html`. Accessed January 25, 2015.