Towards Ontology-based Realtime Behaviour Interpretation

Wilfried Bohlken, Patrick Koopmann, Lothar Hotz and Bernd Neumann
Cognitive Systems Laboratory, University of Hamburg

Abstract

We describe a generic framework for model-based behaviour interpretation and its application to monitoring aircraft service activities. Behaviour models are represented in a standardised conceptual knowledge base using OWL-DL for concept definitions and the extension SWRL for constraints. The conceptual knowledge base is automatically converted into an operational scene interpretation system implemented in Java and JESS which accepts tracked objects as input and delivers high-level activity descriptions as output. The interpretation process employs Beam Search for exploring the interpretation space, guided by a probabilistic rating system. The probabilistic model cannot be efficiently represented in the ontology, but it has been designed to closely correspond to the compositional hierarchy of behaviour concepts. Experiments are described which demonstrate the system performance with real airport data.

1 Introduction

This chapter is about realtime monitoring of object behaviour in aircraft servicing scenes, such as arrival preparation, unloading, tanking and others, based on video streams from several cameras[1]. The focus is on high-level interpretation of object tracks extracted from the video data. The term "high level interpretation" denotes meaning assignment above the level of individually recognised objects, typically involving temporal and spatial relations between several objects and qualitative behaviour descriptions corresponding to concepts used by humans. We prefer to use the term "scene interpretation" in order to avoid reference to a particular level structure. Scene interpretation is understood to include the recognition of multi-object structures (e.g. the facade of a building) as well as the recognition of activities and occurrences (e.g. criminal acts). Regarding its scope, scene recognition can be compared to silent-movie understanding.

For aircraft servicing, scene interpretation has the goal to recognise the various servicing activities at the apron position of an aircraft, beginning with arrival preparation, passenger disembarking via a passenger bridge, unloading and loading operations involving several kinds of vehicles, refuelling, catering, and other activities. Real-time monitoring may serve several purposes. For one, delays in performing scheduled activities can be noticed and counteracted early. Secondly, predictions about the completion of a turnaround can be provided, alleviating planning. Thirdly, monitoring of service activities can be extended to include unrelated object behaviour, e.g. of vehicles not allowed in the proximity of the aircraft.

[1] This work was partially supported by EC Grant 214975, Project Co-Friend.

Our approach aims at developing a largely domain-independent scene interpretation framework, designed to be adaptable to changes and to be reusable for other applications. In fact, our basic approach of structuring the conceptual knowledge base in terms of compositional hierarchies and guiding the interpretation process accordingly has been used in other domains (Hotz & Neumann, 2006; Hotz, Neumann & Terzic, 2008) and by other authors (Rimey, 1993; Fusier, Valentin, Brémond, Thonnat, Borg, D. Thirde & Ferryman 2007; Mumford & Zhu 2007). In this introductory section we shortly discuss major contributions of past research which have influenced our current understanding of scene interpretation and our design decisions for a framework. We also compare with recent work on ontology-based scene interpretation.

Although scene interpretation has enjoyed much less attention in Computer Vision research than object recognition, there exists a considerable body of related work dating as far back as into the seventies. Badler (1975) was one of the first to derive high-level descriptions of simple traffic scenes represented by hand-drawn sketches for lack of computer-generated low-level data about real scenes. He used spatial relations between pairs of objects, corresponding to spatial adverbials, to describe a snapshot of a scene, and changes of these relational structure to describe the temporal development. A temporal concept such as "across-motion" would be recognised by rules defined in terms of preconditions and postconditions. His work showed that spatial predicates form the bridge from quantitative low-level data to qualitative high-level descriptions.

A first systematic approach to motion analysis is due to Tsotsos, Mylopoulos, Covvey, and Zucker 1980) who introduced a taxonomy of motion types. Specific high-level motion events (in this case pathological human-heart motions) were described as a composition of elementary motions, thus also establishing a compositional hierarchy.

Structuring motion by taxonomical and compositional hierarchies, reflecting the logical structure of natural language terms, has played a significant part in most approaches to model scene interpretation, also in the early work of Neumann (1989) on natural-language description of traffic scenes. One of his additional achievements was the separation of behaviour models from control structures for behaviour recognition. Occurrences in street traffic were modelled as declarative aggregates enabling both bottom-up and top-down recognition. Temporal relations between occurrences were modelled by constraints and processed by a constraint system. As in earlier work, low-level image analysis had to be bypassed by manually providing input data in terms of a "Geometric Scene Description" (GSD) consisting of typed objects and quantitative object trajectories.

A first attempt of modelling scene interpretation in a logical framework is due to Reiter and Mackworth (1987). They showed that, for a finite domain, scene interpretation could be formulated as a logical model construction task, i.e. as a search for instantiations of the conceptual knowledge covering the domain, consistent with the factual evidence. This search could be implemented as constraint satisfaction.

Later, the model construction paradigm was extended to Description Logics (DLs) (Schroeder & Neumann, 1996; Neumann & Moeller, 2006). This was motivated by the attractive possibilities for object-centred concept representations offered by a DL and its guarantee for decidable reasoning processes. Unfortunately, it remains difficult to realise scene interpretation with a DL system. One of the reasons is the need to model the quantitative-qualitative mapping from sensory to symbolic data outside the logics.

Also, a central reasoning service for scene interpretation, the incremental construction of a model, is not provided by existing systems.

A slightly different logical paraphrase of scene interpretation was given by Cohn, Magee, Galata, Hogg, and Hazarika (2003), and Shanahan (2005) in terms of abduction, i.e. as a search for high-level concepts whose instantiation would entail the evidence. Similar to logical model construction, experiments are difficult because few logical systems offer abduction services which can be used for scene interpretation. The pioneering work on multimedia interpretation in the group of Moeller (Gries, Moeller, Nafissi, Rosenfeld, Sokolski & Wessel, 2010) is a recent example for using a DL system for abduction. Here, the RacerPro reasoner has been extended by abduction facilities.

Several researchers propose ontology-based activity recognition in a customised logical framework which does not necessarily realise model construction or abduction. Chen and Nugent (2009) present an activity recognition procedure where conceptual descriptions constructed from evidence are tested for equivalence with ontological concepts using a DL reasoner.

An interesting approach to exploiting an OWL ontology using a DL reasoner has been reported in (Riboni & Bettini, 2011a). They use RacerPro to check the consistency of sensor data in the elderly-care domain with asserted interpretations. This procedure realises logical model construction for a fixed inventory of activities. It does not offer a solution for an incremental and multi-level interpretation process.

Irrespective of difficulties in using existing reasoners , the work on a logical formulation of scene interpretation has provided important clarifications, in particular that, from a logical perspective, scene interpretation is inherently ambiguous. The ambiguity is quite striking in temporal scenes where interpretations typically imply predictions about the future. For example, a driver assistance system has to interpret an observed scene with regard to its future development. Will the pedestrian enter the street or wait? As humans, we seem to exploit our experiences for such decisions and prefer the most likely or utile choice given all we know about the domain and the current scenario. Hence, it appears natural to provide a probabilistic model for the uncertainty of logically ambiguous choices.

The need for a preference measure is all the more evident when one considers the stepwise process which must be used when interpreting scenes with several objects and extending over time. As evidence is incorporated piece by piece, early and intermediate interpretation decisions may have to be made with poor context, i.e. in highly ambiguous interpretation situations.

Independently of the logically motivated need for a preference measure, probabilistic approaches to scene interpretation have been developed, motivated by the success of probabilistic methods in other areas of Computer Vision and Robotics, and the desire to develop a seamless integration of low-level image analysis with high-level scene interpretation. In early work (Binford, Levitt & Mann, 1989; Rimey, 1993), Bayesian Networks (BNs) were conceived isomorphic to the compositional structure of high-level concepts, i.e. with aggregates "causing" their parts. While this basic structure is intuitively plausible, it implies that parts are statistically independent given aggregate data. This is not the case in many applications, hence more general probabilistic structures have been developed. In (Koller & Pfeffer, 1997; Getoor & Taskar, 2007) BNs were extended in an object-oriented manner, introducing Probabilistic Relational

Models. This way, crisp relational structures can be augmented with an arbitrary probabilistic dependency structure. To describe multiple levels of granularity, hierarchical BNs were proposed by Gyftodimos and Flach (2007). More recently, Markov Logic Networks have been employed for modelling events and activities (Murariu & Davis, 2011). While these contributions improve the expressive power of BNs or propose an alternative probabilistic framework, they do not adequately support compositional hierarchies of aggregates as required for scene interpretation. For this purpose, Bayesian Compositional Hierarchies (BCHs) have been developed (Neumann, 2008) which are also employed for the work in this chapter and will be described in detail in Section 4.

An interesting alternative approach to combining compositional hierarchies with BNs has been presented by Mumford and Zhu (2007). Here, a grammatical formalism takes the place of hierarchical knowledge representation, and (probabilistic) parsing algorithms are applied for scene interpretation, leading to efficient processing, but complicating the integration with large-scale knowledge representation.

Our approach, presented in the following sections, tries to build on the insights gained from past research, while simultaneously employing standardised or generally applicable formalisms as far as possible. We have chosen OWL DL[2], the standardised web ontology language, for the representation of the conceptual models for scene interpretation in general and for the specific application domain, this is presented in Section 2. The choice of OWL emphasises our interest in connecting to logic-based knowledge representation and reasoning services. But, as will be shown, we must make use of the SWRL extension of OWL to express crisp dependencies between concepts, and we must connect with components outside OWL for probabilistic modelling.

An important contribution of our work is the automatic translation of OWL concepts into an interpretation procedure in JESS, described in Section 3. The procedure realises a bottom-up, evidence-driven parallel Beam Search for all probable interpretations, guided by a Bayesian Compositional Hierarchy (BCH). One of the achievements is a decomposition of this process into independent subgoals which may provide evidence for several higher-level concepts.

The probabilistic model is described in detail in Section 4. It provides a rating for alternative interpretations given the evidence available so far, and also for predictions of future events. We use a BCH to model the temporal relationships of servicing activities and derive ratings using a very efficient probabilistic inference process.

In Section 5, we present results for interpreting aircraft turnarounds at the Toulouse Blagnac Airport. We demonstrate the predictive power of our probabilistic framework and show its effect when rating alternative interpretations.

We conclude with a summary of insights gained and an outlook on future work.

[2] http://www.w3.org/TR/owl-semantics/

2 Behaviour Modelling

In this section, we describe the representation of behaviour models in a formal ontology. Our main concern is the specification of aggregate models which can be used to describe an aircraft turnaround and its constituting activities.

2.1 Aggregate Representation

In a nutshell, an aggregate is a representational structure consisting of

- a specification of aggregate properties,
- a specification of parts, and
- a specification of constraints between parts.

To illustrate the requirements for aggregate specifications, consider the aggregate `Arrival-Preparation` as an example. It has the parts `GPU-Enters-GPU-Zone`, `GPU-Stopped-Inside-GPU-Zone`, and `Drop-Chocks` (Fig. 1).

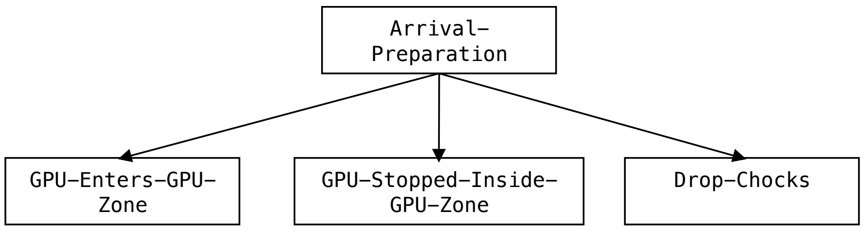


Fig. 1: Compositional structure of the aggregate `Arrival-Preparation` within an aircraft turnaround

The first part, `GPU-Enters-GPU-Zone`, marks the event when the GPU enters its designated standing area (GPU denotes Ground Power Unit). `GPU-Stopped-Inside-GPU-Zone` begins at the moment, the GPU has come to a stop, and ends when the GPU begins leaving the area much later in the turnaround. We use predefined zones for locations following the work of Fusier et al. (2007). The event `Drop-Chocks` marks the moment an operator has deposited the chocks where the airplane is expected to stop.

The example shows that behaviour concepts of different kinds play a role: some extend over a time interval, some mark a time point, some can be derived from tracking data, some require special image analysis procedures. Furthermore, all components of the aggregate `Arrival-Preparation` must obey certain temporal constraints for the model to be realistic. `GPU-Enters-GPU-Zone` typically precedes `GPU-Stopped-Inside-GPU-Zone` by less than a minute. After the GPU has been positioned, it may take up to 10 minutes until a person steps out and deposits the chocks. The duration of `GPU-Stopped-Inside-GPU-Zone` roughly corresponds to a complete turnaround, but does not matter for the definition of `Arrival-Preparation`.

As mentioned in the introduction, we have chosen the web ontology language OWL DL for defining aggregates and related concepts. OWL DL is a standardised formalism with clear logical foundations and provides the chance for a smooth integration with large-scale knowledge representation. Furthermore, the object-centred style of concept definitions in OWL and its support by mature editors such as Protégé[3] promise

[3] http://protege.stanford.edu/

transparency and scalability. Simple constraints can be represented with SWRL, the Semantic Web Rule Language, albeit not very elegantly, as will become apparent shortly.

The availability of DL reasoning systems such as Pellet or Racer may be considered an additional bonus, but - as pointed out in the introduction - incremental scene interpretation in terms of model-construction or abduction steps is not supported by current DL systems. Hence, in our application, the use of a commercial DL reasoner is limited to consistency checks of the conceptual knowledge base. The interpretation process is realised with our own framework called SCENIOR (SCENe Interpretation with Ontology-based Rules) which translates the OWL concepts into a search for instantiations, see Section 3.

In a DL setting, an aggregate has the following generic structure:

$$\begin{aligned}&\texttt{Aggregate\_Concept} \sqsubseteq \texttt{Parent\_Concept} \sqcap \\ &\quad \exists_{=1}\ \texttt{hasPartRole}_1.\texttt{Part\_Concept}_1 \sqcap \ldots \sqcap \\ &\quad \exists_{=1}\ \texttt{hasPartRole}_k.\texttt{Part\_Concept}_k \sqcap \\ &\quad \texttt{conceptual constraints}\end{aligned}$$

Hence an aggregate with the class name Aggregate_Concept is defined by a taxonomical parent, by the parts to which it is related via hasPartRoles, and by conceptual constraints relating aggregate and parts to each other. The left-hand side implies the right-hand side, corresponding to an abductive framework. In our definition, the aggregate may name only a single taxonomical parent because of the intended mapping to single-inheritance Java templates. Furthermore, the aggregate must have exactly one part for each hasPartRole. While the DL syntax would allow number restrictions for optional or multiple parts, we found it useful to have different aggregate names for different part configurations and a distinct hasPartRole for each part to simplify the definition of conceptual constraints.

Within the OWL ontology, conceptual constraints can be expressed using the Semantic Web Rule Language SWRL[4] which combines OWL with RuleML[5] (Rule Markup Language). It is well-known that SWRL rules may cause undecidability of the conceptual knowledge base unless they are restricted to be DL-safe, i.e. only applied to facts (to ABox content in DL terminology). This is the case during scene interpretation where rules are only applied to instances grounded in scene data. However, there exists no reasoner which evaluates SWRL rules for a consistency check of the conceptual knowledge base.

SWRL is supported by the Protégé editor, which guarantees that a certain amount of consistency is maintained. For example, it is not possible to define a rule with classes which are not defined in the knowledge base, or to use variables in the consequent which are not introduced in the antecedent.

For the aggregate `Arrival-Preparation`, the taxonomical and partonomical sections of the concept definition in Protégé read as follows:

$\texttt{Arrival-Preparation} \sqsubseteq \texttt{Composite-Event} \sqcap$

[4] http://www.w3.org/Submission/SWRL/

[5] http://ruleml.org/

```
has-part1 exactly 1 GPU-Enters-GPU-Zone ⊓
has-part2 exactly 1 GPU-Stopped-Inside-GPU-Zone ⊓
has-part3 exactly 1 Drop-Chocks
```

The parent concept `Composite-Event` refers to a concept type defined in the Upper Model which will be discussed further down. We now discuss the SWRL section of the definition:

```
Arrival-Preparation(?arr-prep)                         1
∧  has-part1(?arr-prep, ?GPU-enters)                   2
∧  has-part2(?arr-prep, ?GPU-stopped)                  3
∧  has-part3(?arr-prep, ?drop)                         4
∧  has-start-time(?arr-prep, ?arr-prep-st)             5
∧  has-finish-time(?arr-prep, ?arr-prep-ft)            6
∧  has-time-point(?GPU-enters, ?GPU-enters-tp)         7
∧  has-agent(?GPU-enters, ?GPU-enters-ag)              8
∧  has-start-time(?GPU-stopped, ?GPU-stopped-st)       9
∧  has-agent(?GPU-stopped, ?GPU-stopped-ag)            10
∧  has-time-point(?drop, ?drop-tp)                     11
→
   equal(?arr-prep-st, ?GPU-enters-tp)                 12
∧  equal(?arr-prep-ft, ?drop-tp)                       13
∧  equal(?GPU-enters-ag, ?GPU-stopped-ag)              14
∧  min-before(?GPU-stopped-st, ?arr-prep-st, 100)      15
∧  min-before(?drop-tp, ?GPU-stopped-st, 1000)         16
```

The antecedent part of this rule has the single purpose to establish local variables (prefixed by a '?') for the constraints in the consequent part. The first two constraints (12, 13) relate the starting and finish time of the aggregate to specific time points of the parts. The next constraint (14) requires that the vehicles referred to in `GPU-Enters-GPU-Zone` and `GPU-Stopped-Inside-GPU-Zone` are the same. Finally, `min-before` checks whether the difference between the first and the second argument is less than the third argument. In this case, the gaps between consecutive events are constrained by 100 ms (14) and 1000 ms (15), respectively. When the ontology is translated into an interpretation procedure, all constraints of the consequent part are transformed into a temporal constraint system (TCN) which generates a conflict when a potential instantiation of the aggregate and its parts violates the constraints.

Note that a concept such as `GPU-Stopped-Inside-GPU-Zone` could have also been expressed by the more general concept `Vehicle-Stopped-Inside-Zone`, constrained by the predicates `GPU(?veh-stopped-ag)` and `GPU-Zone(?veh-stopped-zn)` in the SWRL section. In this case, `?veh-stopped-zn` must have been introduced as a variable for the filler of the `has-zone` property. We have taken the design decision to provide specific concept names for all activities in a turnaround in order to establish a transparent correspondence to the probabilistic aggregate models which will be introduced in Section 4.

The complete ontology comprises a domain-independent Upper Model, shown in Fig. 2, with concept types which are generally useful for scene interpretation, roughly corresponding to the Fluent Calculus (Russell & Norvig, 2010). The Domain Model with the specific concepts for aircraft servicing is shown in Table 1. We first describe the concepts of the Upper Model.

2.2 Upper Model

The root concept `Thing` (analogous to the OWL identifier owl:thing) represents every entity which is needed for scene interpretation (not only physical objects). A

Conceptual-Object is used to represent a behaviour concept and may be either a State or an Event. A State is a predicate true over an interval and all its subintervals, a property which is often called "durative". A State may be a Composite-State with several states as parts, all of which must be true over the interval of the parent state. An Event is a non-durative Conceptual-Object, for example marking a transition from one state to another. A Composite Event may have states or events as parts and is used, for example, to describe a complete aircraft turnaround or the aggregate Arrival-Preparation presented above. Primitive states and events have no parts and thus constitute the leaves of the compositional hierarchy. Instances of primitives are typically determined by lower-level processes which provide the input for scene interpretation.

The ontology also includes some unusual concepts which play a specific part in our work but are not further addressed in this contribution. With Intention we refer to the mental state of a person. An Intention is not observable, of course, but intentions can be made part of event models and may be inferred from observations.

While most primitives can be generated from object trajectories obtained from stationary wide-angle cameras, there are important events which require a dynamically controlled pan-tilt-zoom (PTZ) camera and a special image analysis module, for example to recognise whether a pipe is attached to the airplane for refuelling. An event obtained this way is addressed as a Special-Vision-Task.

Another unusual concept is Zone which describes a qualitative location in the horizontal plane. Predefined zones on the apron play a significant part in event models for service activities, exemplified by the primitive event GPU-Enters-GPU-Zone and its property has-zone in the example above.

It may be surprising that the ontology does not include views or similar concepts related to the appearance of objects. In our work on structure recognition in the facade domain (Hotz et al., 2008), instances of 2D object views have been provided as input for high-level interpretation. Hence interpretation included object recognition based on appearances. In the work described here, however, the primitives entering high-level interpretation are recognised 3D entities without appearance properties.

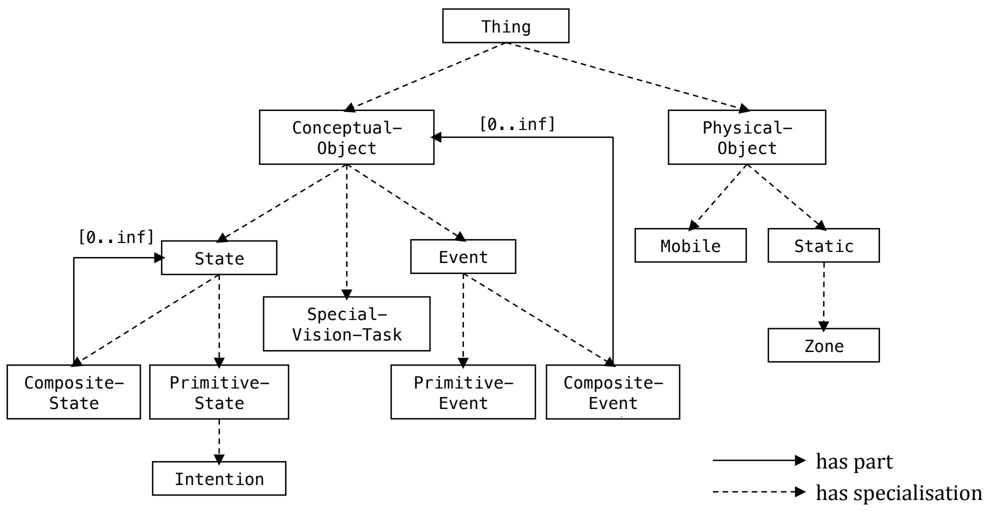


Fig. 2: Upper Model of the behaviour ontology for the aircraft servicing domain

2.3 Domain Model

The Domain Model comprises more than 50 behaviour concepts which are part of a compositional hierarchy with `Turnaround` as the roof concept (see Table 1). Activities difficult to recognise have been omitted here, such as `Catering`, `Air-Conditioning`, `Waste-Removal`, `Replace-Drinking-Water` etc. Note that some concepts occur as a part in more than one aggregate. For example, `Loader-Enters-Right-AFT-LD-Zone` is part of `Unload-Right-AFT` and `Load-Right-AFT`. This illustrates the ambiguities which may arise in bottom-up interpretation and the need for constraints provided by the established context.

In setting up a domain model, it may be useful to introduce abstract temporal behaviour patterns as a common parent for more specific models, so that inheritance can be exploited. For example, `Passenger-Activity` and `Refuelling` each consist of two consecutive parts, the first a `Vehicle-Enters-Zone`, and the second a `Vehicle-Stopped-Inside-Zone`. This behaviour pattern could be defined as a `Visit`:

```
Visit ⊑ Composite-Event ⊓
      has-part1 exactly 1 Vehicle-Enters-Zone ⊓
      has-part2 exactly 1 Vehicle-Stopped-Inside-Zone
```

The SWRL constraints define the start and finish time of a visit and ensure that (i) the agents of the two parts are the same, (ii) the zones are the same, and (iii) the first event happens before the second begins.

```
Visit(?vis)
∧  has-part1(?vis, ?veh-enters)
∧  has-part2(?vis, ?veh-stopped)
∧  has-start-time(?vis, ?vis-st)
∧  has-finish-time(?vis, ?vis-ft)
∧  has-time-point(?veh-enters, ?veh-enters-tp)
∧  has-agent(?veh-enters, ?veh-enters-ag)
∧  has-zone(?veh-enters, ?veh-enters-zn)
∧  has-start-time(?veh-stopped, ?veh-stopped-st)
∧  has-agent(?veh-stopped, ?veh-stopped-ag)
∧  has-zone(?veh-stopped, ?veh-stopped-zn)
→
   equal(?vis-st, ?veh-enters-tp)
∧  equal(?vis-ft, ?veh-stopped-ft)
∧  equal(?veh-enters-ag, ?veh-stopped-ag)
∧  equal(?veh-enters-zn, ?veh-stopped-zn)
∧  min-before(?veh-enters-tp, ?veh-stopped-st, 10000)
```

The definitions of `Passenger-Activity` and `Refuelling` could then simply read

```
Passenger-Activity ⊑ Visit ⊓
      has-part1 exactly 1 Passenger-Stairs-Enters-PS-Zone ⊓
      has-part2 exactly 1 Passenger-Stairs-Stopped-Inside-PS-Zone

Refuelling ⊑ Visit ⊓
      has-part1 exactly 1 Tanker-Enters-Tanking-Zone ⊓
      has-part2 exactly 1 Tanker-Stopped-Inside-Tanking-Zone
```

with no additional SWRL constraints.

Table 1: Domain model with aggregates of the `Turnaround` hierarchy

```
Turnaround
    Arrival
        Arrival-Preparation
            GPU-Enters-GPU-Zone
            GPU-Stopped-Inside-GPU-Zone
                GPU-Stopped
                GPU-Inside-GPU-Zone
            Drop-Chocks
        Airplane-Enters-ERA
        Airplane-Stopped-Inside-ERA
            Airplane-Stopped
            Airplane-Inside-ERA
    Services
        Passenger-Activity
            Passenger-Stairs-Enters-PS-Zone
            Passenger-Stairs-Stopped-Inside-PS-Zone
                Passenger-Stairs-Stopped
                Passenger-Stairs-Inside-PS-Zone
        Unload-Right
            Unload-Right-AFT
                Loader-Enters-Right-AFT-LD-Zone
                Transporter-Enters-Right-AFT-TS-Zone
                Unload-Motion-Right-AFT-Belt
                Transporter-Leaves-Right-AFT-TS-Zone
                Loader-Leaves-Right-AFT-LD-Zone
            Unload-Right-FWD
                Loader-Enters-Right-FWD-LD-Zone
                Transporter-Enters-Right-FWD-TS-Zone
                Unload-Motion-Right-FWD-Belt
                Transporter-Leaves-Right-FWD-TS-Zone
                Loader-Leaves-Right-FWD-LD-Zone
        Refuelling
            Tanker-Enters-Tanking-Zone
            Tanker-Stopped-Inside-Tanking-Zone
                Tanker-Stopped
                Tanker-Inside-Tanking-Zone
            Pumping-Operation
        Load-Right
            Load-Right-AFT
                Loader-Enters-Right-AFT-LD-Zone
                Transporter-Enters-Right-AFT-TS-Zone
                Load-Motion-Right-AFT-Belt
                Transporter-Leaves-Right-AFT-TS-Zone
                Loader-Leaves-Right-AFT-LD-Zone
            Load-Right-FWD
                Loader-Enters-Right-FWD-LD-Zone
                Transporter-Enters-Right-FWD-TS-Zone
                Load-Motion-Right-FWD-Belt
                Transporter-Leaves-Right-FWD-TS-Zone
                Loader-Leaves-Right-FWD-LD-Zone
    Departure
        Start-Beacon
        Pushback
```

In addition to behaviour concepts, the domain ontology contains definitions of specific zones, specific mobile objects (`Person` or `Vehicle`) and refinements of `Vehicle` in terms of `GPU`, `Loader`, `Tanker` etc.

The temporal constraints specified for each aggregate provide only a crude model of temporal relationships in a turnaround. As the data of real turnarounds show, there are sometimes exceptions and unusual delays, hence tight crisp constraints may exclude valid interpretations, while overly loose constraints may spawn too many false positives. As pointed out in the introduction, we therefore developed a probabilistic model for temporal relations. Conceptually, it should be part of the ontology, but unfortunately there is currently no way to efficiently connect probability distributions with OWL. Our probabilistic framework, described in Section 4, is therefore represented outside of OWL and connects to the interpretation system at runtime.

Key terms: Behaviour recognition, monitoring, event ontology, Beam Search, probabilistic guidance, Bayesian Compositional Hierarchy, prediction, aircraft servicing

Generating a Scene Interpretation System from the Ontology

In this section, we describe the scene interpretation system SCENIOR which is automatically generated from the conceptual knowledge base represented in OWL and SWRL. We first give an overview and provide some motivation for design decisions. We then describe the tasks of the conversion process in detail, in particular generating rules for rule-based scene interpretation and generating a temporal constraint system (TCN) for the evaluation of SWRL constraints.

2.4 Motivation and System Overview

The design decisions which have led to SCENIOR are based on insights gained from past scene interpretation projects (Neumann & Weiss, 2003; Hotz & Neumann, 2005) and shared by many researchers in the field:

(i) The interpretation system must be based on conceptual knowledge represented in a standardised way. This is a prerequisit for reusability of knowledge and for an economical development of application systems.

(ii) Interpretation can be viewed as a mixed bottom-up and top-down search for the best explanation of evidence in compositional and taxonomical hierarchies. Hence interpretation steps can be modelled with predefined generic patterns.

(iii) For stepwise scene interpretation, it is necessary to entertain several possible interpretations in parallel. A greedy interpretation strategy would be likely to fail, in general, and backtracking would cause high system complexity and inefficient performance.

(iv) In addition to a logic-based framework and crisp constraints, a probabilistic model is required to provide preferences among interpretation alternatives.

We have therefore chosen an approach where the conceptual knowledge base is converted into a rule-based system with rules realising both bottom-up and top-down processing, and parallel threads for processing alternative interpretations. Probabilistic guidance is provided by a probabilistic model homomorphic with compositional hierarchies of the conceptual knowledge base.

Concretely, SCENIOR is implemented as a Java application with JESS[6] (Java Expert System Shell) for rule-based processing. JESS is one of the fastest rule engines available, it can directly manipulate and reason about Java objects. Fig. 3 shows the main components of the system.

In the initialisation phase of the system, a converter (which is also part of SCENIOR) loads the knowledge base and translates it into rules and templates in JESS format (see Subsection 3.2). The result is called *JESS conceptual knowledge base* (JCKB). The temporal constraints defined with SWRL rules are translated into temporal constraint nets (see Subsection 3.3). Then a JESS thread is created for each submodel of the compositional hierarchy and provided with a submodel hypothesis composed of working memory elements (WMEs) for each expected instantiation for this submodel,

[6] http://www.jessrules.com/

and with a corresponding TCN. Now the system is ready to start the interpretation process (see Section 2.8).

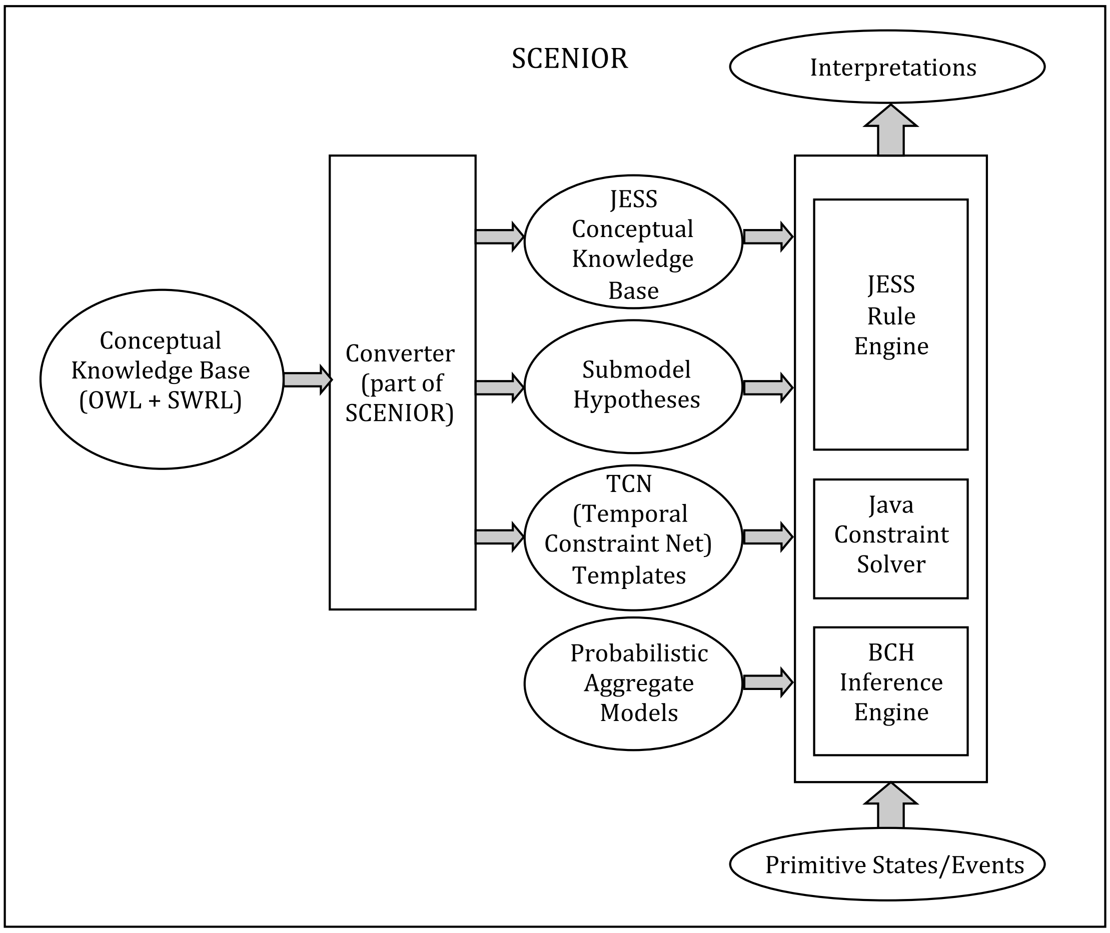

SCENIOR
Interpretations
Conceptual Knowledge Base (OWL + SWRL)
Converter (part of SCENIOR)
JESS Conceptual Knowledge Base
Submodel Hypotheses
TCN (Temporal Constraint Net) Templates
Probabilistic Aggregate Models
JESS Rule Engine
Java Constraint Solver
BCH Inference Engine
Primitive States/Events

Fig. 3: Main components of the scene interpretation system SCENIOR

SCENIOR expects real-time evidence about an evolving scene in the form of primitive symbolic activity tokens with attached quantitative temporal information and other useful properties as input. For aircraft servicing scenes, this evidence is provided in terms of primitive states or events generated by a tracking component and a simple-event detector developed by project partners.

The interpretation process, described in detail in Subsection 3.5, employs Beam Search to generate possible interpretations in parallel threads. In each thread, a JESS rule engine effects data-driven rule applications, and a Java constraint solver evaluates crisp temporal constraints and terminates a thread in the case of a conflict. In addition, a probabilistic inference engine for Bayesian Compositional Hierarchies (BCHs) provides a rating of each partial interpretation to control the Beam Search.

2.5 Rule Generation from the Ontology

It has been shown by Neumann and Moeller (2006) that scene interpretation can be viewed as a search in the space of possible interpretations defined by taxonomical and compositional relations and controlled by constraints. Four kinds of interpretation steps are required to navigate in interpretation space and to construct interpretations:

- Aggregate instantiation (moving up a compositional hierarchy)
- Aggregate expansion (moving down a compositional hierarchy)

- Instance specialisation (moving down a taxonomical hierarchy)
- Instance merging (unifying instances obtained separately)

In our framework, we create rules for the first three steps, with some additional supporting rules. It will be shown that the step "instance merging“ becomes expendable with the use of submodel hypotheses and parallel search.

Creating templates and slots. Before creating JESS rules, every concept of the OWL knowledge base is converted into a JESS *template* with the same name as the concept. Templates are the main structuring feature of the JESS rule language, they are analogous to Java classes. A template is defined by a name and a number of *slots*, which are comparable to the member variables of a Java class. Here, slots of a template are defined corresponding to the properties of the concept and complemented with an additional slot *name*, which will hold the name of a particular instance (e.g. `vehicle_17`). To preserve the taxonomical hierarchy of the OWL knowledge base, we use the inheritance mechanism of JESS, where templates can be defined as sub-templates which inherit the slots of the parent template. Our approach differs from the transformation of OWL and SWRL to JESS described in (Erikson, 2003), where the properties are modelled as *ordered facts* which are simply JESS lists, and the template structure is flat. This would have the effect that properties are decoupled from the concept template, and the taxonomy must be emulated by duplicating facts along the taxonomical hierarchy, destroying the object-centered structure of an aggregate and possibly leading to scalability problems.

Submodel hypotheses. In the next step, a hypothesis structure is generated for each submodel identified in the conceptual knowledge base. A submodel hypothesis represents the compositional structure of the submodel, respecting the equality constraints described with SWRL rules, and provides placeholders for all expected instantiations. Each submodel hypothesis defines an independent interpretation goal. This facilitates evidence assignment and splitting of the search tree into several alternatives which are tracked in parallel. A submodel hypothesis can also be viewed as a structure for coherent expectations. During interpretation it can be used to "hallucinate" missing evidence and thus to continue a promising interpretation thread. Fig. 4 shows a simplified submodel hypothesis for the submodel `Arrival`.

.

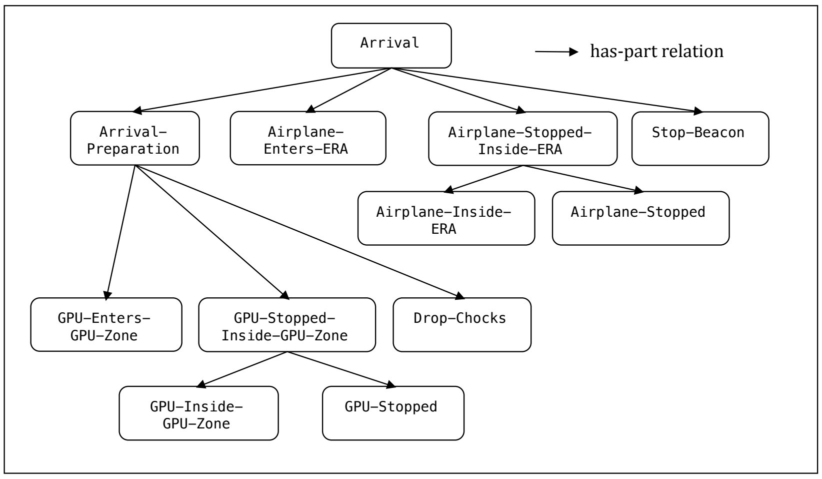


Fig. 4: Submodel hypothesis for the submodel `Arrival`

Rules. In the next step of the transformation process, the following rules are created fully automatically from the ontology:

- Evidence-assignment rules
- Aggregate-instantiation and hallucination rules
- Aggregate-expansion rules
- Specialisation rules
- Time-update rules.

An *evidence-assignment rule* assigns evidence provided by lower-level processing to a hypothesis element which is the leaf of a submodel hypothesis. In the premise of the rule, the template corresponding to the aggregate is addressed (`GPU-Enters-GPU-Zone` in the example below), and the temporal constraints are checked with a test function (*test conditional element*). The processing of the temporal constraints is described in detail in the next subsection. In the action part of the rule, the status of the evidence is changed from `evidence` to `assigned`, and the status of the hypothesis element is changed from `hypothesised` to `instantiated`.

The following simplified example illustrates the generic pattern of an evidence-assignment rule:

```
(defrule GPU-Enters-GPU-Zone_ea_rule
       ?e-id <- (GPU-Enters-GPU-Zone
                     (name ?gegz_17)
                     (status evidence))
       ?h-id <- (GPU-Enters-GPU-Zone
                     (name ?gegz_h)
                     (status ?status_1))
       (test (or (eq ?status_1 hypothesised)
                 (eq ?status_1 hallucinated)))
       ;; check temporal constraints
 =>
       (modify ?e-id (status assigned))
       (modify ?h-id (status instantiated))
       ;; update temporal constraint net
)
```

Fig. 5: Example illustrating the generic pattern of an evidence-assignment rule.

An a*ggregate-instantiation rule* instantiates an aggregate with status `hypothesised` if all its parts are `instantiated` or `hallucinated`. This is a bottom-up step in the compositional hierarchy and the backbone for the scene interpretation process. It is not necessary to check the temporal constraints here, because this has already been done in the evidence-assignment rules of the parts. A simplified example illustrating the generic pattern of an aggregate-instantiation rule is given below:

```
(defrule Arrival-preparation_ai_rule
       ?h-id <- (Arrival-Preparation
                     (name ?ap_h)
                     (status hypothesised)
                     (has-part-1 p1)
                     (has-part-2 p2)
                     (has-part-3 p3))
       (GPU-Enters-GPU-Zone
                     (name ?p1)
                     (status ?status_1))
       (test (or (eq ?status_1 instantiated)
                 (eq ?status_1 hallucinated)))
       (GPU-Stopped-Inside-GPU-Zone
                     (name ?p2)
```

```
                  (status ?status_2))
      (test (or (eq ?status_2 instantiated)
                (eq ?status_2 hallucinated)))
      (Drop-Chocks
                  (name ?p3)
                  (status ?status_3))
      (test (or (eq ?status_3 instantiated)
                (eq ?status_3 hallucinated)))
 =>
      (modify ?h-id (status instantiated)))
```

Fig. 6: Example illustrating the generic pattern of an aggregate-instantiation rule.

A *specialisation rule* refines an instance to a more specialised instance. Unfortunately, the inbuilt JESS template structure only recognises if an instance of a concept satisfies a more general concept restriction, for example, an instance of `GPU-Enters-GPU-Zone` is recognised as a an instance of `Vehicle-Enters-Zone`. However, it is not recognised that an instance of `Vehicle-Enters-Zone` with `has-agent GPU` and `has-location GPU-Zone` is an instance of `GPU-Enters-GPU-Zone`. This classification could be easily performed by any DL reasoner, but we preferred a solution within the JESS system, hence the rules for specialisation. A simplified rule for this example is given below.

```
      (defrule GPU-Enters-GPU-Zone_s_rule
      ?e-id <- (Vehicle-Enters-Zone
                   (name ?vez_14)
                   (status evidence)
                   (has-agent ?a1)
                   (has-location ?l1))
      (GPU (name ?a1))
      (GPU-Zone (name ?l1))
      (not (GPU-Enters-GPU-Zone (name ?vez_14)))
 =>
      (retract ?e-id))
      (assert (GPU-Enters-GPU-Zone
                 (name ?vez_14)
                 (status evidence)
                 (has-agent ?a1)
                 (has-location ?l1)))
```

Fig. 7: Example illustrating the generic pattern of a specialisation rule.

An *aggregate-expansion rule* instantiates part of an aggregate if the aggregate itself is instantiated or hallucinated. This is a top-down step in the compositional hierarchy. A separate rule is created for every part of the aggregate. It will be invoked if a fact has not been asserted bottom-up but by other means, e.g. common-sense reasoning. Such facts receive the status `hallucinated`, alluding to a sentence attributed to Max Clowes (1971): "Vision is controlled hallucination". If an aggregate has the status `hallucinated`, aggregate-expansion rules will also hallucinate all its parts. If some of the parts have already been instantiated by an evidence-assignment rule or an aggregate-instantiation rule, then the corresponding rules will not fire; a mechanism for instance merging is not necessary.

In the current state of our interpretation system, an event (or state) will be hallucinated if (i) it is defined as *hallucinatable* in the ontology, and (ii) it can be derived from the temporal constraint net that the event should have happened by now. This check is done in the Java background. For example, the event `Drop-Chocks` in an `Arrival-Preparation` is difficult to detect by an image processing module, so it could be defined as *hallucinatable*.

The following simplified example illustrates the generic pattern of an aggregate-expansion rule:

```
(defrule Arrival-preparation_ae1_rule
      (Arrival-Preparation
           (name ?ap_h)
           (status hypothesised)
           (has-part ?p))
      ?p-id <- (GPU-Enters-GPU-Zone
                    (name ?p)
                    (status hypothesised))
 =>
      (modify ?p-id (status hallucinated)))
```

Fig. 8: Simplified generic pattern for an aggregate-expansion rule.

The *time-update rules* set the `has-finished` property of a primitive state to `true` and update the temporal constraint net. These rules are necessary, because a primitive state has a duration and is added as evidence to the working memory at the moment of its first occurrence in the input data stream. This is desirable so that other rules, like evidence-assignment rules and aggregate-instantiation rules, can fire before the primitive state has finished. For example, a `Vehicle-Enters-Zone` event can be inferred even though the primitive state `Vehicle-Inside-Zone` has not yet finished. If the update caused by a time-update rule leads to an inconsistency of the TCN, the thread representing this interpretation alternative will die.

Summarising rule generation, we note that format and flavour of the rules are noticeably influenced by the expressivity of a classical rule language like JESS, and may raise doubts regarding the verifiability of the intended behaviour. Generated from an OWL ontology, however, most of the drawbacks of large rule systems can be avoided, as will be explained in the following.

First, the ontology provides an unerring inventory of aggregate models organised below the Upper-Model node `Conceptual-Object`. Hence a rule generation process can be set up which is complete and non-redundant regarding all aggregate-related rules. Second, the taxonomical structure of the ontology specifies exactly, which specialisation inferences have to be provided by rules, supplementing the inbuilt inferences.

A commonly experienced drawback of rule systems is the intransparent flow of control, influenced by conflict resolution between multiple rules applicable to a processing state. Rule conflicts are not avoided in our framework, but they are resolved by giving every rule an equal chance in a separate interpretation thread, see Section 3.5. Furthermore, control decisions regarding termination of a thread are subject to declarative constraints and an independent rating scheme.

2.6 Temporal Constraints

The temporal constraints in the SWRL section of the ontology use expressions of the *convex time point algebra* (Vila, 1994). The *Allen temporal operators* used in the SWRLTemporalOntology[7] are not expressive enough for our purposes, because they only allow to model qualitative relations, whereas the complexity of our domain requires quantitative temporal models.

[7] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTemporalOntology

As the basic format of a temporal relation in the convex time point algebra, we use

$$t_2 - t_1 \leq c_{12} \qquad (1)$$

where t_1 and t_2 are range-valued variables for time points and c_{12} is a constant. A range is a unary constraint specifying the minimal value t_{min} and maximal value t_{max} which a time variable t may take. An event is described with a single time point, a state with a start and a finish time point. We use SWRL rules to express arbitrary linear-inequality relations between time points of different aggregates. This way, it is possible to model important features of the temporal structure of a scene.

Temporal constraint nets are generated for each aggregate of the ontology and then merged into a constraint net for each submodel. Fig. 9 shows the temporal constraint net for the submodel `Arrival` introduced in Fig. 4. The suffix "st" indicates a start time point and the suffix "ft" a finish time point of a state. The suffix "tp" indicates the time point of an event. A directed arc represents an inequality according to Eq. 1, with the arrow pointing from t_1 to t_2 and the number at the arrow representing c_{12}. A double-ended arrow with the offset 0 indicates equality of the connected time points.

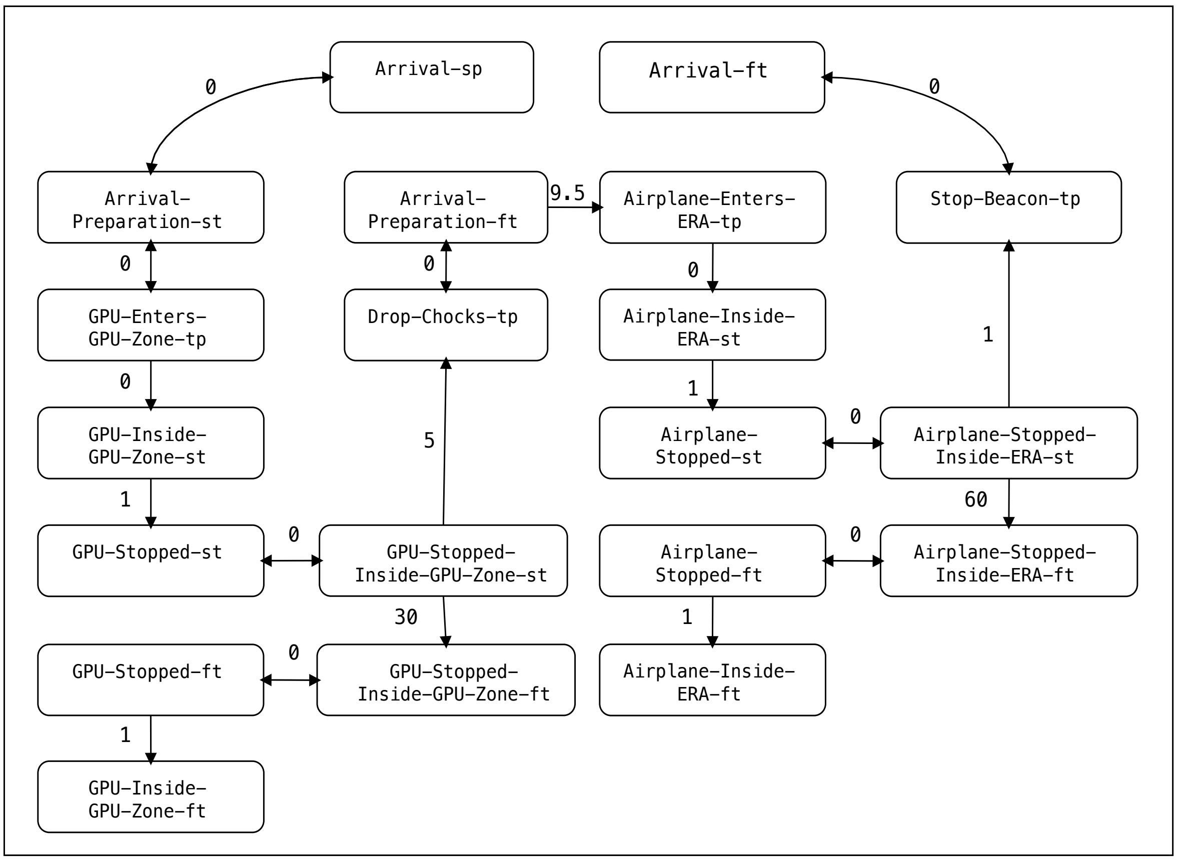


Fig. 9: Temporal constraint net for the submodel `Arrival`.

Initially the range of each time variable is open-ended, i.e. $[-\infty\ +\infty]$. When an evidence-assignment rule or a time-update rule fires, the corresponding variables will receive concrete values which are then propagated through the constraint net as follows (Neumann, 1989):

- Minima are propagated in edge direction: $t_{2min}' = \max(t_{2min}, t_{1min} + c_{12}) \qquad (2)$

- Maxima are propagated against edge direction: $t_{1max}' = \min(t_{1max}, t_{2max} - c_{12})$ (3)

A TCN is consistent, if $t_{max} \geq t_{min}$ holds for all time variables. The TCN maintains the consistency of the crisp temporal constraints, while the scene is evolving. It is implemented procedurally in Java using *shadow facts* (Friedmann-Hill, 2003). This way, changes of the TCN are immediately visible to JESS and may effect rule activations.

2.7 Using Submodels

Usually, several models have to be considered in a scene interpretation task. One reason is the need to cope with several alternative variants of a scene model. In our domain of airport activity monitoring, for example, there are turnarounds with or without refuelling, with or without de-icing etc. One can easily imagine that there are similar situations in other domains, where alternative models exist that vary in some parts but are identical otherwise.

One way of coping with alternatives is to create separate complete models. This approach is illustrated with the abstract models M_1 and M_2 on the left-hand side of Fig. 10. It has the obvious disadvantage of leading to redundancies in the interpretation process, since identical alternatives have to be maintained in parallel. A better approach is to decompose the alternative models into several submodels, as depicted on the right-hand side of Fig. 10. This has the added advantage of allowing subgoals to be defined, each of which may be interesting for a monitoring task in its own right.

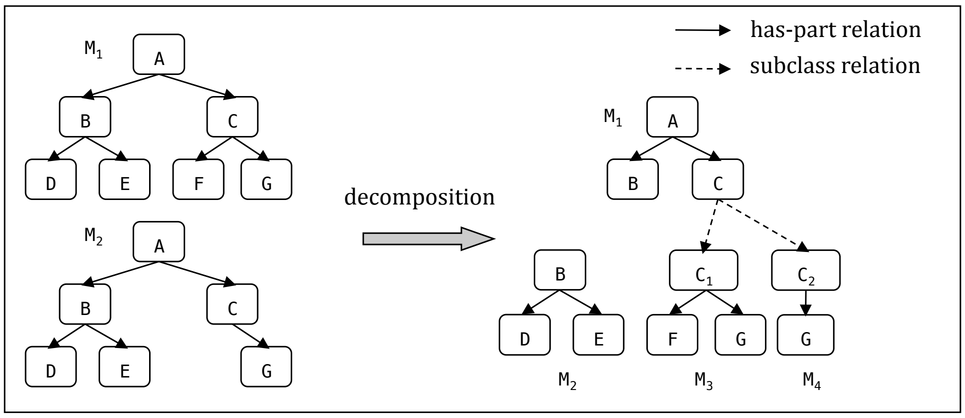


Fig. 10: Decomposition of separate models into submodels

Subgoals lead to intermediate results which will not be discarded and can be used as "higher-level evidence" for other aggregates. In the OWL ontology, concepts corresponding to subgoals are marked as `context-free`, indicating that the conceptual definition is designed to be valid in several contexts. This is a signal for the transformation process to generate submodels with these concepts as roots. Alternative submodels can be defined as variants of a common parent. If a root R of a submodel M_k is instantiated, this instance is offered as *higher-level evidence* to every interpretation thread with R or a superclass of R as a leaf.

On the right-hand side of Fig. 10, the concepts B and C are marked as `context-free`. Furthermore, $C_1 \sqsubseteq C$ and $C_2 \sqsubseteq C$. For example, C_1 could stand for a `Service` with `De-icing` (F), and C_2 for a `Service` without `De-icing`. If C_1 or C_2 is recognised (instantiated), a corresponding higher-level evidence is provided for every interpretation thread containing a concept of type C as a leaf. In our example this could be M_1 modelling a `Turnaround`.

For the transformation process from OWL and SWRL to JESS rules, submodels cause separate threads to be created, and a separate temporal constraint net is generated for every submodel. Furthermore, evidence-assignment rules are generated not only for primitive but also for higher-level evidence.

2.8 Interpretation Process

We can now give an overview of the interpretation process with SCENIOR. In the initialisation phase the following steps are performed:

- The OWL conceptual knowledge base (including SWRL rules) is automatically transformed into a JESS conceptual knowledge base. All necessary templates and rules are generated.
- A separate interpretation thread is created for every submodel, equipped with a submodel hypothesis structure and a temporal constraint net. Each thread has its own independent JESS engine.

Now the system is ready to start the interpretation process. It receives primitive states and events as input, and feeds these as WMEs to every interpretation thread. Then the rules are applied and may eventually lead to instantiated aggregates, which again may cause higher-level aggregates to be instantiated. If there is more than one activation for an evidence-assignment rule within one thread (i.e. if multiple evidence assignments are possible), then this thread is cloned into several threads, one for each possible alternative assignment. A newly created thread is an exact copy of the original thread. Then each of the possible assignments is performed on a separate clone by forcing the corresponding rule to fire. This way, a search tree is established which examines all interpretation possibilities in parallel.

So far, we have not yet discussed how to deal with noise, which can either occur in terms of activities not modelled in the ontology, or as errors of low-level processing. Both kinds of noise are abundantly present in our aircraft servicing domain. Various kinds of vehicles not taking part in a service or performing some unknown task enter and leave the servicing area throughout a turnaround. Also, low-level processing in our application is difficult and not at all perfect, hence strange events not corresponding to real activities are delivered as input to SCENIOR.

We have therefore extended our scene model to include a noise model which allows anything not covered by the turnaround model to happen at any time. Naturally, this causes an exponential explosion of interpretation threads: For each evidence, the number of threads increases, reduced only by inconsistent threads.

As it turned out, SCENIOR can process up to ca. 100 threads in parallel and in real-time on an ordinary PC. Nevertheless, the number of interpretation threads required for all possibilities will exceed the capacity after a few steps. Clearly, a preference measure is required which allows to discard less promising threads even if they satisfy the scene model. To this end, we have developed a probabilistic scene model which will be presented in the next section.

3 A Probabilistic Preference Model

As explained in the preceding section, scene interpretation can be controlled by a crisp constraint net, in our case the temporal constraint net TCN, such that only consistent interpretations survive in a parallel search. This approach has the advantage that the scene model can be completely represented in OWL and SWRL, and that the constraint solver can be realised efficiently. However, as pointed out before, constraints may not be tight, else correct interpretations may be missed. Hence the interpretation system may generate many false positives, depending on the discriminative qualities of the evidence.

We have therefore developed a probabilistic rating system to obtain a preference measure for consistent interpretations. By computing this rating for intermediate partial interpretations, the most promising ones can be kept, unlikely ones can be discarded. Applied to multiple consistent final interpretations, a single most likely interpretation can be selected.

We first formulate a general probabilistic model for scene interpretation and derive a rating for partial and final interpretations. We then present Bayesian Compositional Hierarchies (BCHs) which are special probabilistic models structured in congruence with the compositional hierarchy of the OWL ontology.

3.1 A General Probabilistic Model for Scene Interpretation

In a general form, probabilistic scene interpretation can be modelled as evidence-based reasoning with large joint probability distributions (JPDs). In the context of a parallel search, the task is to determine which of M alternative models applies to a scene. A generative probabilistic model for a scene can be written as

$$P_{scene} = q_m \; P^{(m)}(\underline{X}_1^{(m)} \ldots \underline{X}_{N^{(m)}}^{(m)} \underline{Y}_1^{(m)} \ldots \underline{Y}_{K^{(m)}}^{(m)}) \; P_{clutter} \tag{4}$$

We assume that there are M competing models with priors q_m, m = 1 .. M. Each model is described by a JPD consisting of hidden variables X = [X_1 .. X_N] and observable variables Y = [Y_1 .. Y_K]. The indices suggest distinct conceptual entities (for example aggregates), each described by a vector of random variables, indicated by the underline.

Values for observable variables are provided by evidence from low-level processing, values of hidden variables are determined by probabilistic inference. In our temporal model for the aircraft servicing domain, the observables correspond to time points marking a primitive event such as `Vehicle–Stopped–Inside–Zone`, whereas hidden variables could describe beginning and duration of higher-level activities such as `Arrival–Preparation`.

$P_{clutter}$ is a catch-all distribution for evidence not fitting a model. This could simply be a JPD modelling the occurrence of "unexplainable" evidence objects during a turnaround as independent events.

To guide the interpretation process, we are interested in ranking alternative partial interpretations given evidence $\underline{e}$. Alternatives do not only arise from the models 1 .. M but also from alternative assignments of evidence within a model. For example, a `Vehicle–Enters–ERA` event (ERA is the large Entrance Restricted Zone where all turnaround activities occur) can be part of many service activities of a turnaround, in particular, if the type of vehicle is uncertain or tracking errors have occurred.

Alternative assignments cause additional competing interpretations. Further alternatives arise from assigning some of the evidence - say $\underline{e}_n^+$ - to the model and the rest - say $\underline{e}_n^-$ - to clutter, possibly different for each model. To simplify the notation, we enumerate all alternatives - due competing models and alternative assignments - using the index n.

The ranking R_n of a scene model n is given by the probability that the model has generated $\underline{e}_n^+$ as part of the service model and $\underline{e}_n^-$ as clutter. This is captured by the following equation:

$$R_n = q_n \; P^{(n)}(\underline{e}_n^+) \; P_{clutter}(\underline{e}_n^-) \tag{5}$$

Eq. (2) shows that alternative rankings can be determined from Eq. (1) by marginalising the observables of each model n which have been chosen for evidence assignment, and computing the resulting probabilities.

To determine the final interpretation, one has to perform two maximisations. First, determine the highest-ranking model n*, and then determine the values $\underline{x}_n$* for hidden variables of this model which maximise its posterior probability. These steps are described by the following equations:

$$n^* = \arg\max_n (q_n P^{(n)}(\underline{e}_n^+) P_{clutter}(\underline{e}_n^-)) \tag{6}$$

$$[\underline{X} = \underline{x}^*, \underline{Y} = \underline{e}^+] = \arg\max_{\underline{x}} (q_{n^*} P^{(n^*)}(\underline{x},\; \underline{e}_{n^*}^+)) \tag{7}$$

Note that the probabilistic model given by Eq. (4) does not explicitly account for missing evidence, for example due to occlusion or tracking limitations. To deal with this, the range of observables could be extended to include "missing evidence" as a possible "value", but an assignment and probabilistic appreciation will necessarily depend on the context. The issue of missing information will not be treated in the sequel.

If interpretation is performed in real-time, probabilistic models may be adapted to the progressing time. In our application, a scene model as given by Eq. (4) will involve temporal random variables representing observable events on a quantitative time scale relative to some common reference event, for example relative to an initial observation. Real-time processing using such a model implies that we have a current time t_c which progresses as we observe a concrete scene, and that modelled events not observed so far are bound to happen at times $t > t_c$, if at all. This should influence our ranking of alternatives to the effect that reduced chances for an event cause a reduced ranking.

Let $\underline{e}$ be evidence assigned up to time t_c, and $\underline{T}_n \subseteq \underline{Y}$ be unassigned temporal observables of a model. Then the rank of model n at time t_c is given by

$$R_n(t_c) = q_n P^{(n)}(\underline{e}_n^+, \underline{T}_n > t_c) P_{clutter}(\underline{e}_n^-) \tag{8}$$

Eq. (5) shows that the ranking of an alternative model changes according to its share in the probability space for the remaining temporal variables. This refines Eq. (5) which implied that the complete probability space was left for unassigned variables. Note that real-time updating does not apply to hidden temporal variables for which values $t < t_c$ may be inferred.

3.2 Bayesian Compositional Hierarchies

In order to preserve the advantages of ontology-based scene models, it is useful to achieve a tight integration of the probabilistic model with a logic-based compositional hierarchy. As an important step towards integration we present an approach for formulating probabilistic scene models in terms of probabilistic aggregate models complementing the aggregate definitions in OWL. Rimey (1993) modelled compositional hierarchies using tree-shaped Bayesian Networks (BNs). To ensure efficient processing, he had to assume that parts of an aggregate are statistically independent given the parent aggregate. In (Neumann, 2008) a more powerful hierarchical probabilistic model has been presented, called Bayesian Compositional Hierarchy (BCH). In the following, we briefly summarise the definition of a BCH for arbitrary probability distributions. Thereafter, we describe the structure of a Gaussian BCH which is the kind used for modelling the temporal structure of aircraft services in our work.

A BCH is a probabilistic model of a compositional hierarchy. It consists of aggregates, each modelled individually by an unrestricted JPD in an object-centered manner. The hierarchy is formed by using the aggregate headers as part descriptions in aggregates of the next hierarchical level, abstracting from details of parts at the lower level.

Figure 11 illustrates the schematic structure of a BCH. Each aggregate is described by a JPD $P(\underline{A}\ \underline{B}_1..\underline{B}_K\ \underline{C})$ where $\underline{A}$ is the aggregate header providing an external description to the next higher level, $\underline{B}_1..\underline{B}_K$ are descriptions of the parts, and $\underline{C}$ expresses conditions on the parts. The hierarchy is constructed by taking the aggregate headers at a lower level as part descriptions at the next higher level, hence $\underline{B}_1^{(1)} = \underline{A}^{(2)}$ etc.

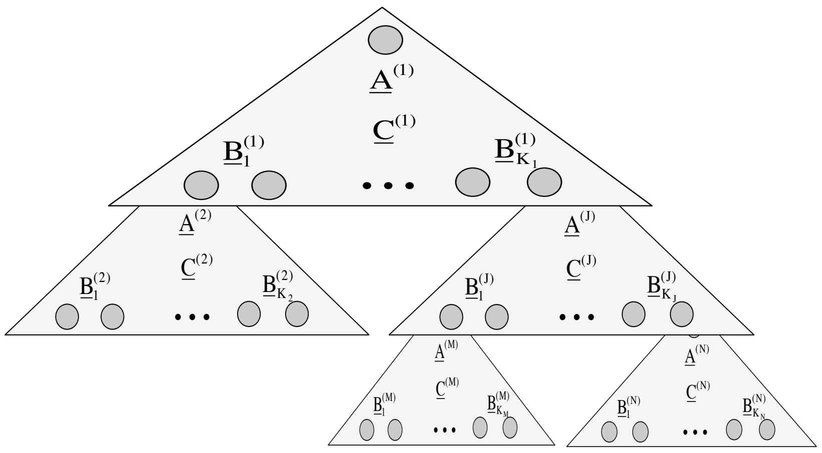


Fig. 11: Schematic structure of a BCH. Triangles represent aggregates, circles represent parts. Aggregate models overlap, their headers represent parts of aggregates at the next higher level.

In our aircraft servicing domain, for example, a `Turnaround` aggregate consists of a header which provides an external description of a turnaround in terms of its duration (abstracting from details about the parts), and an internal description of the temporal structure of the three parts `Arrival`, `Services` and `Departure`. The parts are also described as aggregates themselves, for example `Arrival` is an aggregate with parts `Arrival–Preparation`, `Airplane–Enters–ERA` and `Stop–Beacon`, compare with the hierarchy shown in Table 1.

In general, the JPD of a complete hierarchy is given by

$$P(\underline{A}^{(1)} .. \underline{A}^{(N)}) = P(\underline{A}^{(1)}) \prod_{i=1..N} P(\underline{B}_1^{(i)} .. \underline{B}_{K_i}^{(i)} \underline{C}^{(i)} \mid \underline{A}^{(i)}) \tag{9}$$

This remarkable formula shows that the JPD of a BCH can be easily constructed from individual aggregate representations, and belief updates can be performed by propagation along the tree structure. Let $P'(B_i)$ be an update of $P(B_i)$, by evidence or propagation from its parts below. Then the updated aggregate JPD is

$$P'(A\ B_1 .. B_K\ C) = P(A\ B_1 .. B_K\ C)\ P'(B_i)/P(B_i) \tag{10}$$

A similar equation holds when P(A) is updated by propagation from its parent above.

Storage and updating operations for large hierarchies can be computationally very expensive. We have therefore developed an implementation for aggregates with multivariate Gaussian distributions. Roughly symmetric, unimodal distributions can often be approximated by a Gaussian in a range corresponding to $-2\sigma .. +2\sigma$, where σ is the standard deviation. Multivariate Gaussian aggregate models can be compactly represented by means and covariance matrices, and propagation in a BCH can be performed very efficiently by closed-form solutions, as shown in the following.

Let G = [$\underline{E}$ $\underline{F}$] be a vector of Gaussian random variables representing an aggregate. Let $\underline{F}$ be the subset whose distribution is changed by evidence or incoming propagation. $\underline{F}$ can be the aggregate header in the case of downward propagation or a part header in the case of upward propagation. We want to compute the effect of the changed distribution of $\underline{F}$ on $\underline{G}$. Before propagation, the distribution of $\underline{G}$ is $P(\underline{G}) = N(\underline{\mu}_G, \Sigma_G)$ where $\underline{\mu}_G$ is the mean vector and Σ_G the covariance matrix. The partitions corresponding to $\underline{E}$ and $\underline{F}$, respectively, are denoted as shown:

$$\Sigma_G = \begin{bmatrix} \Sigma_E & \Sigma_{EF} \\ \Sigma_{EF}^T & \Sigma_F \end{bmatrix} \quad \underline{\mu}_G = \begin{bmatrix} \underline{\mu}_E \\ \underline{\mu}_F \end{bmatrix}$$

For a probability update, we assume that the distribution of $\underline{F}$ is changed to $P'(\underline{F}) = N(\underline{\mu}_F', \Sigma_F')$. Then the new distribution of $\underline{G}$ is $P'(\underline{G}) = N(\underline{\mu}_G', \Sigma_G')$ with

$$\Sigma_G' = \begin{bmatrix} \Sigma_E' & \Sigma_{EF}' \\ \Sigma_{EF}'^T & \Sigma_F' \end{bmatrix} \qquad \underline{\mu}'_G = \begin{bmatrix} \underline{\mu}'_E \\ \underline{\mu}'_F \end{bmatrix} \qquad \text{where}$$

$$\Sigma_E' = \Sigma_E - \Sigma_{EF}\,\Sigma_F^{-1}\,\Sigma_{EF}^T + \Sigma_{EF}\,\Sigma_F^{-1}\,\Sigma_F'\,\Sigma_F^{-1}\,\Sigma_{EF}^T \tag{11}$$

$$\Sigma_{EF}' = \Sigma_{EF}\,\Sigma_F^{-1}\,\Sigma_F' \tag{12}$$

$$\underline{\mu}_E' = \underline{\mu}_E + \Sigma_{EF}\,\Sigma_F^{-1}(\underline{\mu}_F' - \underline{\mu}_F) \tag{13}$$

It is evident that both upward and downward propagation for an aggregate with random variables $\underline{A}$ $\underline{B}_1$ ... $\underline{B}_K$ $\underline{C}$ can be performed by fairly simple matrix computations.

Multivariate Gaussians are also very convenient for computing the ranking as described in Eq. (5). The marginalisations required for ranking alternative interpretations are directly available from the aggregate covariances, and the final maximizing interpretation according to Eqs. (6) and (7) can be given in terms of the mean values of hidden variables.

There are, however, clear limitations of the applicability of multivariate Gaussian BCHs, for example in connection with discrete random variables, range-limited flat distributions or the truncated distributions arising in real-time updates according to Eq. (8). In some cases it may be possible though to use Gaussians as approximations. This will be shown in the following for the temporal structure of aircraft services.

3.3 A probabilistic model for the temporal structure of aircraft services

To perform real-time interpretation of aircraft servicing operations, a BCH for all aggregates shown in Table 1 has been determined from the statistics of 52 turnaround records. For each aggregate, the temporal structure of its parts is specified in terms of correlated random variables for durations and delays. Fig. 12 illustrates the structure of the aggregate `Arrival-Preparation` as an example. It is defined by the two random variables `Delay1` and `Delay2` which denote the delays between the point event `GPU-Enters-GPU-Zone`, the beginning of `GPU-Stopped-Inside-GPU-Zone` and the point event `Drop-Chocks`, respectively. The aggregate header is a random variable for the duration of `Arrival-Preparation`, its value is defined as the sum of `Delay1` and `Delay2`.

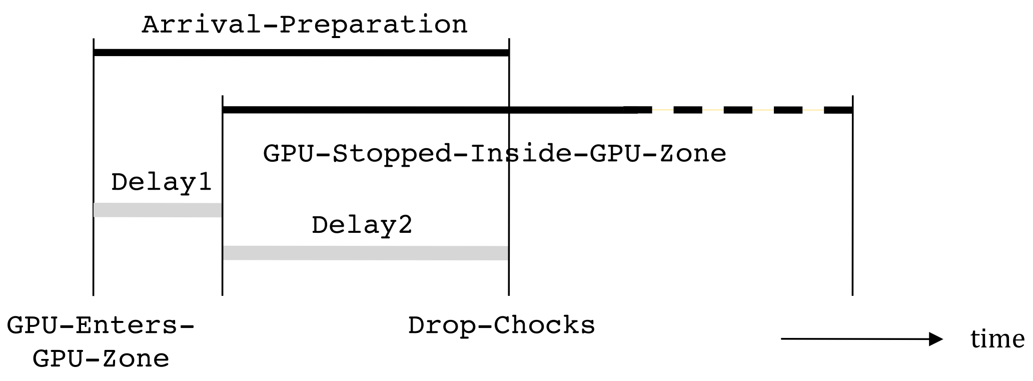


Fig. 12: Temporal structure of the aggregate `Arrival-Preparation`

Table 2 shows the means and the covariance matrix of the Gaussian JPD. Since the duration of `Arrival-Preparation` is determined by the sum of the random variables describing the parts, the covariance is singular, but this does not jeopardise the updating procedure. The positive correlation between the delays reflects the observation that activities in some turnarounds are generally faster than in others.

Table 2: Multivariate Gaussian JPD for `Arrival-Preparation` (units in minutes)

	mean	covariance		
Arrival-Preparation	6	29,3	2,25	27
Delay1	1	2,25	0,25	25
Delay2	5	27	2	25

All aggregate models have a similar structure, with activities described by their durations and related to each other by delays. Gaussians are used with the understanding that only the range $-2\sigma \ldots +2\sigma$ is valid in the model. To ensure that durations of activities take only positive values, their models are constrained by $\mu > 2\sigma$.

The reader may wonder how the probabilistic model relates to the crisp temporal constraints of the ontology. As a matter of fact, the crisp constraints have initially been the only means to express temporal relationships (Bohlken & Neumann, 2009), and experiments showed their strengths and weaknesses. A carefully tailored probabilistic model with zero probabilities outside a specific range would make the crisp model dispensable, of course. But since we decided to employ a multivariate Gaussian model with tails which are not quite realistic, a combination of the two models, where the crisp constraints cut off the distributions outside their $-2\sigma \dots +2\sigma$ ranges, seems to be a good solution. .

5. Interpreting Aircraft Turnarounds

In this section, we present results obtained in experiments with real turnaround data. First, we demonstrate the predictive power of the BCH and changes caused by partial evidence. To this end, the estimated timeline for turnaround events and the remaining uncertainty (measured in standard deviations) has been determined for two cases, (i) after observing the very first event, `GPU-Enters-GPU-Zone`, and (ii) after observing all events up to a late `Airplane-Enters-ERA` , see Table 3.

Table 3: Estimated timeline of a turnaround after an initial observation (Case 1) and after observations up to `Airplane-Enters-ERA` (Case 2). Columns show times T and uncertainties of estimates D (standard deviations) in minutes.

	Case 1		Case 2	
	T	D	T	D
Turnaround-Beg	**0**	0	**0**	0
Arrival-Beg	0	0	**0**	0
Arrival-Preparation-Beg	0	0	**0**	0
GPU-Enters-GPU-Zone-Eve	0	0	**0**	0
GPU-Stopped-Inside-GPU-Zone-Beg	1	0,5	**1**	0
Drop-Chocks-Eve	6	3	**6**	0
Arrival-Preparation-End	6	3	**6**	0
Airplane-Enters-ERA-Eve	9	6	**15**	0
Airplane-Stopped-Inside-ERA-Beg	9	6	15	0
Stop-Beacon-Eve	17	6	23	1
Arrival-End	17	6	23	2
Services-Beg	19	8	26	3
Passenger-Activity-Beg	19	8	26	3
Passenger-Stairs-stopped-Inside-PS-Zone-Beg	19	8	26	3
Passenger-Stairs-stopped-Inside-PS-Zone-End	55	16	62	15
Passenger-Activity-End	58	17	65	15
Unload-Right-Beg	23	9	30	5
Unload-Right-AFT-Beg	23	9	30	5
Loader-Stopped-Inside-Right-AFT-LD-Zone-Beg	23	9	30	5
Transporter-Stopped-Inside-Right-AFT-TS-Zone-Eve	25	9	32	5
Unload-Motion-Right-AFT-Belt-Beg	29	9	36	6
Unload-Motion-Right-AFT-Belt-End	39	10	46	7
Transporter-Stopped-Inside-Right-AFT-TS-Zone-End	41	10	48	7
Loader-Stopped-Inside-Right-AFT-LD-Zone-End	43	10	50	7
Unload-Right-AFT-End	43	10	50	7
Unload-Right-FWD-Beg	24	12	31	10
Loader-Stopped-Inside-Right-FWD-LD-Zone-Beg	24	12	31	10
Transporter-Stopped-Inside-Right-FWD-TS-Zone-Beg	25	12	33	10
Unload-Motion-Right-FWD-Belt-Beg	29	13	36	11
Unload-Motion-Right-FWD-Belt-End	39	13	46	11
Transporter-Stopped-Inside-Right-FWD-TS-Zone-End	41	13	48	11
Loader-Stopped-Inside-Right-FWD-LD-Zone-End	43	14	50	12
Unload-Right-End	43	13	50	11
Refuelling-Beg	33	31	40	30
Tanker-Stopped-Inside-Tanking-Zone-Beg	33	31	40	30
Pumping-Operation-Beg	36	31	43	30
Pumping-Operation-End	43	31	50	30
Tanker-Stopped-Inside-Tanking-Zone-End	47	31	54	30
Services-End	55	23	62	22
Departure-Beg	57	24	69	21
Start-Beacon	57	24	69	21
Pushback-Beg	58	25	70	22
Pushback-End	60	25	72	22
Departure-End	60	25	72	22
Turnaround-End	60	25	72	22

Note that extended activities are marked with the suffix -Beg and -End indicating begin and end, respectively, while point events are marked with the suffix -Eve. It can be seen that observations in Case 2 significantly change the expectations of future events due to the correlations within aggregate models. Also, as to be expected, the uncertainty of estimates decreases with additional evidence.

Fig. 13 illustrates the change of expectation for the `Stop-Beacon` event in terms of its probability density before and after the `Airplane-Enters-ERA` evidence. Note that the density values scale the rating, hence the evidence will have a significant influence on controlling the Beam Search.

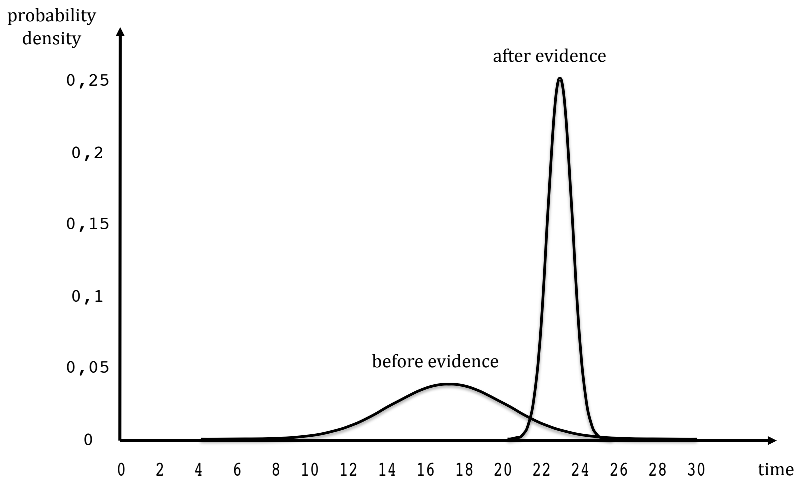
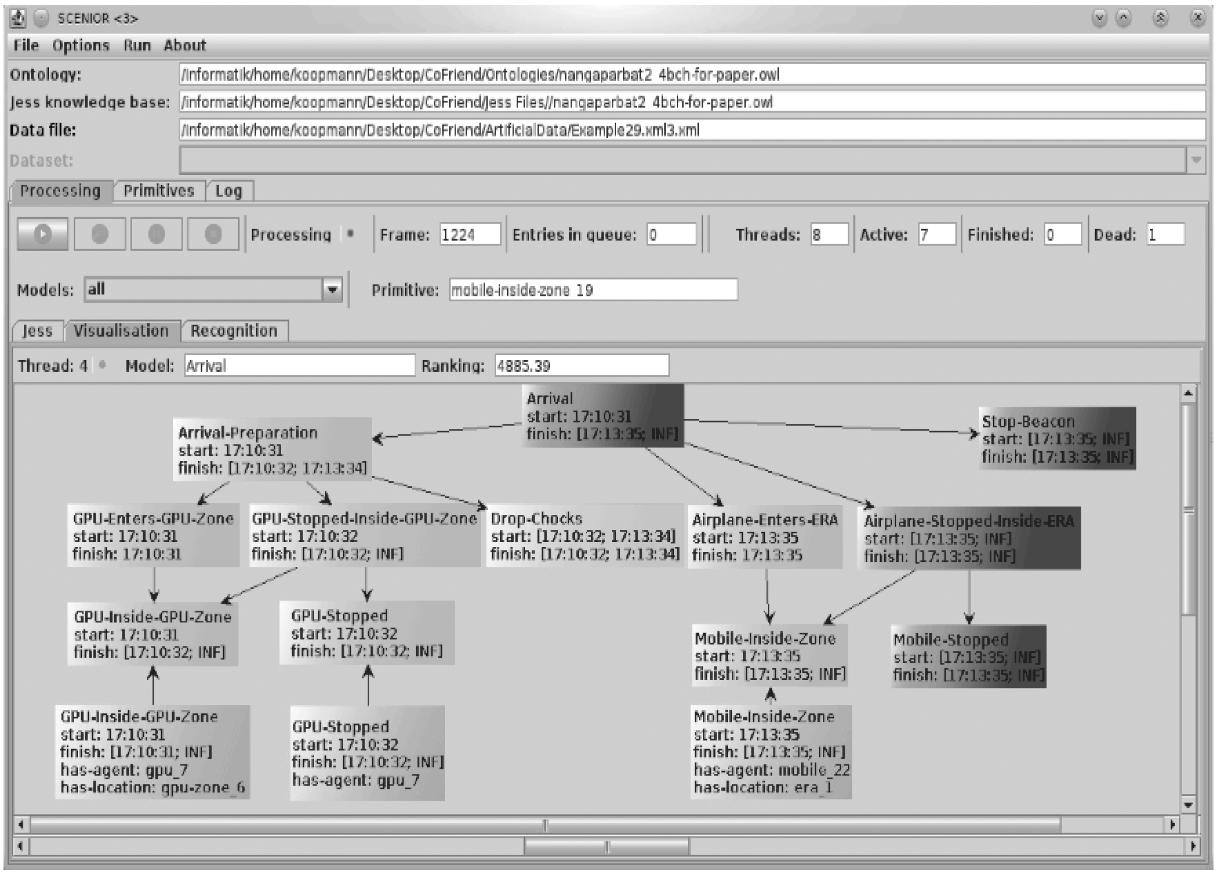


Fig. 13: Change of the probability density of `Stop-Beacon` caused by evidence for a late arrival.

We now describe the initial phase of a concrete scene interpretation task to demonstrate the selective effect of the ranking provided by the BCH in a Beam Search. The input data have been obtained from one of the 80 turnarounds recorded at the Blagnac Airport in Toulouse by low-level processing of project partners in France and England.

To rate interpretations in this experiment, the probability density of clutter has been set to 0,01 which is less than the typical probability of a regular piece of evidence for a turnaround. Note that the probability density is taken to measure the "probability" of an event. A small constant factor Δt for a time span, over which a density must be integrated, is omitted for clarity. Since the ratings are naturally decreasing with each step and may reach very small numbers, the natural logarithm of a probability is taken, resulting in negative ratings.

Based on evidence up to the event `Airplane-Enters-ERA`, SCENIOR generates 8 alternative interpretations (one of which has already been disqualified), performing a complete search where every event could also be a clutter event. In Figures 14 and 15, we show two competing interpretations, with boxes at the bottom for the evidence received so far, dark boxes for expected further events according to hypotheses trees, and other boxes representing instantiations. The `Drop-Chocks` event could not be observed and was inferred from the context as "hallucinated". The figures do not show any of the several clutter events which did not fit the partially instantiated models.


SCENIOR <3>

Ontology: /informatik/home/koopmann/Desktop/CoFriend/Ontologies/nangaparbat2 4bch-for-paper.owl

Jess knowledge base: /informatik/home/koopmann/Desktop/CoFriend/Jess Files//nangaparbat2 4bch-for-paper.owl

Data file: /informatik/home/koopmann/Desktop/CoFriend/ArtificialData/Example29.xml3.xml

Dataset:

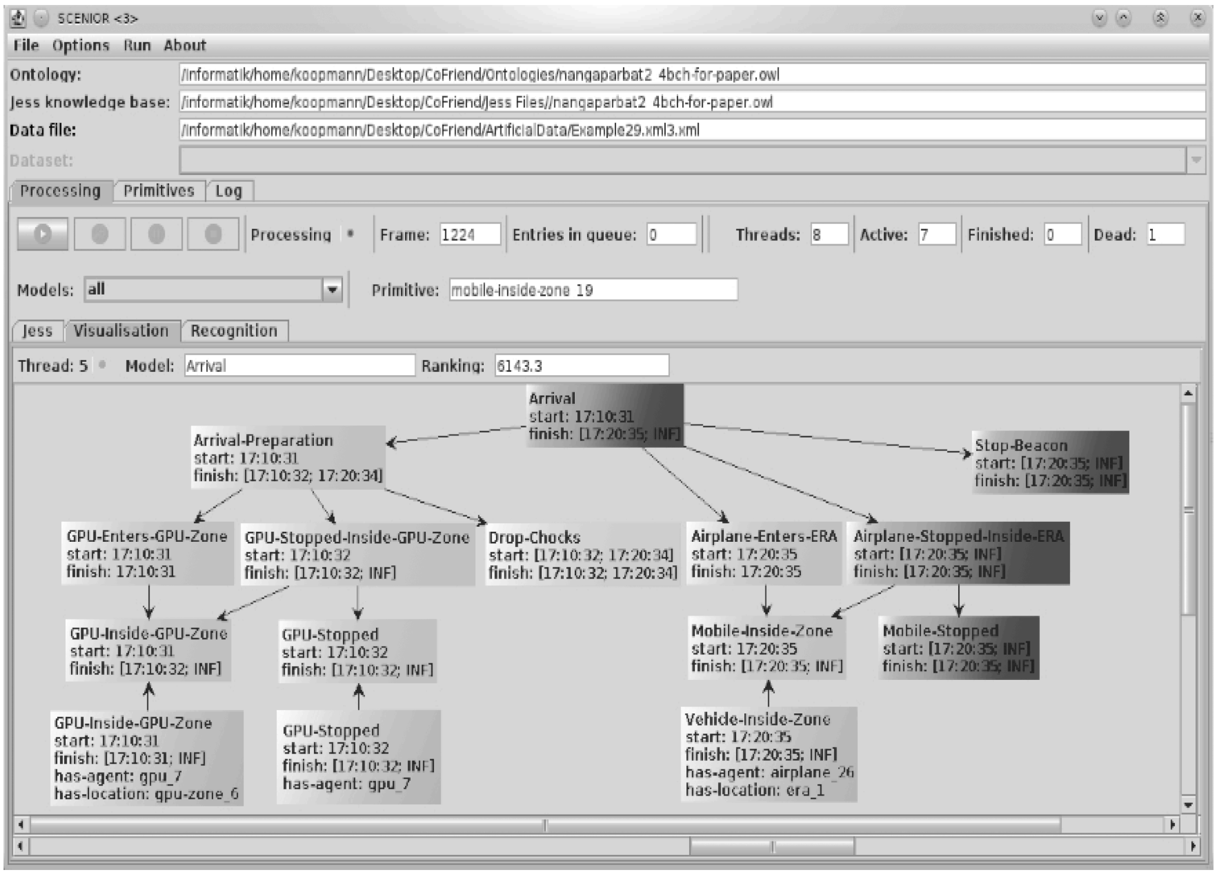
Processing | Frame: 1224 | Entries in queue: 0 | Threads: 8 | Active: 7 | Finished: 0 | Dead: 1

Models: all | Primitive: mobile-inside-zone 19

Thread: 4 | Model: Arrival | Ranking: 4885.39

File
Options
Run
About
Processing
Primitives
Log
Jess
Visualisation
Recognition
Arrival start: 17:10:31 finish: [17:13:35; INF]
Arrival-Preparation start: 17:10:31 finish: [17:10:32; 17:13:34]
Stop-Beacon start: [17:13:35; INF] finish: [17:13:35; INF]
GPU-Enters-GPU-Zone start: 17:10:31 finish: 17:10:31
GPU-Stopped-Inside-GPU-Zone start: 17:10:32 finish: [17:10:32; INF]
Drop-Chocks start: [17:10:32; 17:13:34] finish: [17:10:32; 17:13:34]
Airplane-Enters-ERA start: 17:13:35 finish: 17:13:35
Airplane-Stopped-Inside-ERA start: [17:13:35; INF] finish: [17:13:35; INF]
GPU-Inside-GPU-Zone start: 17:10:31 finish: [17:10:32; INF]
GPU-Stopped start: 17:10:32 finish: [17:10:32; INF]
Mobile-Inside-Zone start: 17:13:35 finish: [17:13:35; INF]
Mobile-Stopped start: [17:13:35; INF] finish: [17:13:35; INF]
GPU-Inside-GPU-Zone start: 17:10:31 finish: [17:10:31; INF] has-agent: gpu_7 has-location: gpu-zone_6
GPU-Stopped start: 17:10:32 finish: [17:10:32; INF] has-agent: gpu_7
Mobile-Inside-Zone start: 17:13:35 finish: [17:13:35; INF] has-agent: mobile_22 has-location: era_1

Fig. 14: Interpretation alternative No. 4 generated by SCENIOR after 3 minutes real-time


SCENIOR <3>

Ontology: /informatik/home/koopmann/Desktop/CoFriend/Ontologies/nangaparbat2 4bch-for-paper.owl

Jess knowledge base: /informatik/home/koopmann/Desktop/CoFriend/Jess Files//nangaparbat2 4bch-for-paper.owl

Data file: /informatik/home/koopmann/Desktop/CoFriend/ArtificialData/Example29.xml3.xml

Dataset:

Processing | Frame: 1224 | Entries in queue: 0 | Threads: 8 | Active: 7 | Finished: 0 | Dead: 1

Models: all | Primitive: mobile-inside-zone 19

Thread: 5 | Model: Arrival | Ranking: 6143.3

File
Options
Run
About
Processing
Primitives
Log
Jess
Visualisation
Recognition
Arrival start: 17:10:31 finish: [17:20:35; INF]
Arrival-Preparation start: 17:10:31 finish: [17:10:32; 17:20:34]
Stop-Beacon start: [17:20:35; INF] finish: [17:20:35; INF]
GPU-Enters-GPU-Zone start: 17:10:31 finish: 17:10:31
GPU-Stopped-Inside-GPU-Zone start: 17:10:32 finish: [17:10:32; INF]
Drop-Chocks start: [17:10:32; 17:20:34] finish: [17:10:32; 17:20:34]
Airplane-Enters-ERA start: 17:20:35 finish: 17:20:35
Airplane-Stopped-Inside-ERA start: [17:20:35; INF] finish: [17:20:35; INF]
GPU-Inside-GPU-Zone start: 17:10:31 finish: [17:10:32; INF]
GPU-Stopped start: 17:10:32 finish: [17:10:32; INF]
Mobile-Inside-Zone start: 17:20:35 finish: [17:20:35; INF]
Mobile-Stopped start: [17:20:35; INF] finish: [17:20:35; INF]
GPU-Inside-GPU-Zone start: 17:10:31 finish: [17:10:31; INF] has-agent: gpu_7 has-location: gpu-zone_6
GPU-Stopped start: 17:10:32 finish: [17:10:32; INF] has-agent: gpu_7
Vehicle-Inside-Zone start: 17:20:35 finish: [17:20:35; INF] has-agent: airplane_26 has-location: era_1

Fig. 15: Interpretation alternative No. 5 generated by SCENIOR after 10 minutes

The main difference between the interpretations is an erroneous `Airplane-Enters-ERA` event generated by low-level processing for a tanker crossing the ERA shortly before the arrival of the airplane. Figure 16 shows the corresponding video frames taken by one of the eight cameras. The crossing tanker is visible in the far background of the image on the left.

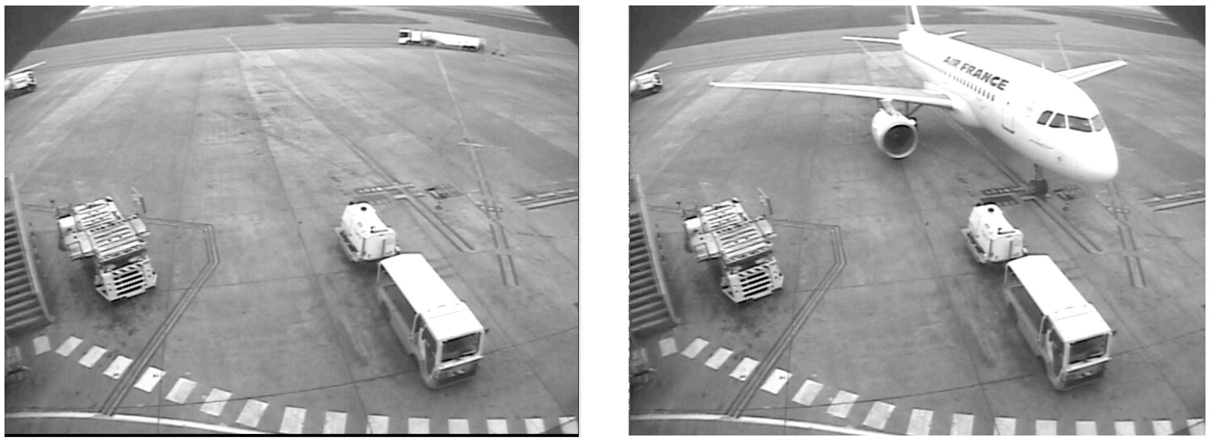


Fig. 16: Snapshots of the ERA (Entrance Restricted Area) after completing `Arrival-Preparation`. The GPU (Ground Power Unit) and chocks are in place. The tanker crossing the ERA in the background (left) causes an erroneous interpretation thread (see text).

The ratings for the partial interpretations of both alternatives are shown in Table 4. Interpretation 4 is the erroneous and Interpretation 5 is the correct one. Initially, the arrival of the GPU sets a context where a vehicle is expected to enter the ERA, hence the crossing tanker is a candidate. But as soon as the true airplane enters, an alternative arises and is favoured because the probabilistic model expects an `Airplane-Enters-ERA` event 8 minutes after `GPU-Enters-GPU-Zone`, and the airplane's arrival is closer to that estimate than the tanker's. Note that clutter events not assigned to either of the two interpretations are not shown in the table.

Table 4. Initial ratings of the two alternative interpretations shown in Figures 14 and 15

```
e1 =  mobile-inside-zone-86
e2 =  mobile-stopped-90
e3 =  mobile-inside-zone-131
e4 =  mobile-inside-zone-155
est = estimated event
```

Evidence	Time	Interpretation 4	Ranking 4	Interpretation 5	Ranking 6
e1	17:10:31	GPU-Enters..	0	GPU-Enters..	0
e2	17:10:32	GPU-Stopped..	-2,16	GPU-Stopped..	-2,16
e3	17:13:31	Airplane-Enters-ERA	-5,32	Clutter	-2,16
e4	17:20:35	Clutter	-5,32	Airplane-Enters-ERA	-5,09
est	≥17:13:35	Airplane-Stopped..	-6,24		
est	≥17:13:35	Stop-Beacon	-7,71		
est	≥17:20:35			Airplane-Stopped..	-6,01
est	≥17:28:35			Stop-Beacon	-7,48

The table also includes the estimated times of the next events `Airplane-Stopped-Inside-ERA` and `Stop-Beacon` together with the expected ratings of the competing interpretations. Note that estimated time windows may begin earlier than the actual time, allowing for hallucinated events in the past. Considering that `Stop-Beacon` will occur after the true aircraft arrival and not at the time expected in Interpretation 4, the rating of this interpretation will surely be much lower than the estimated value, further increasing the distance between the right and the wrong interpretation.

The performance of SCENIOR was evaluated for 20 annotated turnarounds, with primitive events provided by low-level image analysis of the project partners. The ontology and the probabilistic model were derived from 32 other turnarounds. Because of the noisy input data, it was necessary to interpret each evidence both as belonging to a turnaround (given that the constraints were satisfied) and as clutter. 17 of the 20 turnarounds resulted in complete interpretations. This was facilitated by Special Vision Tasks with controlled cameras for three crucial events and by "hallucinations" for missing evidence in certain contexts. The three problematic sequences were highly irregular and did not match the conceptual model (e.g. GPU arrival after aircraft arrival). SCENIOR showed a reliable system performance with up to 100 parallel threads (limited by a preset beam width) for partial alternative interpretations, as shown in Fig. 17. It can be seen that altogether more than 1000 partial interpretations have been initialised, many caused by the context-free submodels which posed interpretation goals throughout the sequence.

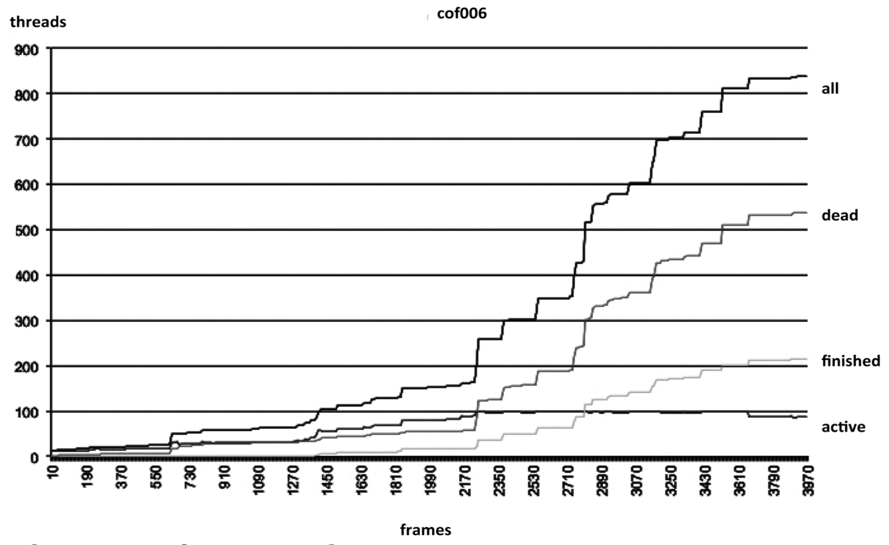


Fig. 17. Thread statistics for a typical interpretation process

The recognition rate of subactivities is shown in the table below. It was limited to 75% because of the noisy low-level input data with missing crucial evidence.

Table 5. Correctly recognised subactivities in 20 test sequences.

SEQUENCE	1	2	3	4	5	6	8	9	18	25	29	58	59	62	63	66
Arrival	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Passenger-Boarding-Preparation	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Unloading-Loading-AFT	1	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0
Unloading-Loading-FWD				1		1	0	0				1	1		1	1
Refuelling			0	0	0			0		0		0		0		
Pushback-Arrival	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
Passenger-Bridge-Leaves-PBB-Zone	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Departure	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

6. Conclusions and Outlook

We have presented an approach to high-level scene interpretation with several novel features. First, the interpretation system is automatically generated from an ontology of behaviour concepts represented in the standardised language OWL-DL and its extension SWRL. Second, the interpretation process is organised as a Beam Search allowing several parallel interpretation threads. Third, a probabilistic model homomorphic to the compositional concept hierarchy provides a preference measure which rates competing interpretations and controls the Beam Search.

The approach is a step ahead on the way to a generic framework for scene interpretation, but it also shows current limitations regarding a standardised representation of behaviour models. Constraints, which are indispensable for object-centered aggregate representations, have to be expressed with SWRL rules and cannot be included in a consistency check of the conceptual knowledge base. Unfortunately, this drawback prevails in OWL 2 (Riboni & Bettini, 2011b). Probabilistic relationships between properties, in our application time points marking the beginning and ending of activities, cannot be efficiently expressed in OWL or SWRL and must be represented separately. Also, we could not yet demonstrate the benefits of OWL regarding inference services by a DL reasoning system. As pointed out in the introduction, one reason is the need for incremental model construction which cannot be answered by the available DL systems. Another reason is the absence of large-scale knowledge bases to which scene interpretation might eventually be connected. We still believe that OWL (or its successors) may provide an attractive basis for such larger knowledge bases, hence justifying our OWL-based scene interpretation framework.

In the near future, our approach will be extended into several directions. One goal is to connect the rule system with common sense rules which can evaluate the interpretation context and determine dynamically whether missing evidence is "hallucinatable". Another goal is to extend quantitative constraints to also cover spatial information, refining the current representations based on "zones". To this end, a more powerful constraint system and an extended implementation of a BCH must be developed.

References

Badler, N.I. (1975). Temporal scene analysis: conceptual descriptions of object movements. Report TR 80, Dep. of Computer Science, University of Toronto, 1975

Binford, T.O., Levitt, T.S., & Mann,W.B. (1989). Bayesian inference in model-based machine vision. In Levitt et al. (Eds.), *Uncertainty in AI(3)*, 73-96

Bohlken, W., & Neumann, B. (2009). Generation of rules from ontologies for high-level scene interpretation. In Governatori et al. (Eds.): *Rule Interchange and Applications*, Proc. Int. Symp. RuleML 2009, Springer, 93-107

Chen, L., & Nugent, C.D. (2009). Ontology-based activity recognition in intelligent pervasive environments. Int. J. Web Information Systems 5 (4), 410-430

Cohn, A.G., Magee, D., Galata, A., Hogg, D., & Hazarika, S. (2003). Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In Freksa et al. (Eds.), *Spatial Cognition III*, 232–248

Eriksson, H. (2003). Using JessTab to integrate Protégé and Jess. *IEEE Intelligent Systems 18(2)*, 43-50

Friedman-Hill, E. (2003). *Jess in Action: Java rule-Based systems.* Manning, Greenwich, 2003.

Fusier, F., Valentin, V., Brémond, F., Thonnat, M., Borg, M., Thirde, D., & Ferryman, J. (2007). Video understanding for complex activity recognition. *Machine Vision and Applications,* 18(3), Springer, 167-188

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to Statistical Relational Learning.* The MIT Press, 129–174

Gries, O., Moeller, R., Nafissi, A., Rosenfeld, M., Sokolski, K., & Wessel, M. (2010). A probabilistic abduction engine for media interpretation. Proc. Fourth Int. Conf. on Web reasoning and rule systems, 182-194

Gyftodimos, E., & Flach, P.A. (2002). Hierarchical Bayesian Networks: A probabilistic reasoning model for structured domains. In de Jong, E., Oates, T. (Eds.), Proc. Workshop on Development of Representations, ICML, 23–30

Hotz, L., & Neumann, B. (2005). Scene interpretation as a configuration task. *Kuenstliche Intelligenz,* 3/2005, BöttcherIT, Bremen, Germany, 59-65

Hotz, L., Neumann, B., & Terzic, K. (2008). High-level expectations for low-level image processing. Proc. KI-2008, Springer, 87-94

Koller, D., & Pfeffer, A. (1997). Object-oriented Bayesian Networks. In *The Thirteenth Annual Conf. on Uncertainty in Artificial Intelligence*, 302–313

Mumford, D., & Zhu, S.-C. (2007). A stochastic grammar of images. Now Publishers

Morariu, V.I. , & Davis, L.S. (2011). Multi-agent event recognition in structured scenarios. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2011

Neumann, B. (1989). Natural language description of time-varying scenes. In Waltz, D. (Ed.), *Semantic structures,* Lawrence Erlbaum, 167-206

Neumann, B., & Weiss, T. (2003). Navigating through logic-based scene models for high-level scene interpretations. In Proc. 3rd Int. Conf. on Computer Vision Systems (ICVS 2003), Springer, 212-222

Neumann. B. (2008). Bayesian Compositional Hierarchies - A probabilistic structure for scene interpretation. TR FBI-HH-B-282/08, Dep. of Informatics, Univ. of Hamburg, Germany

Neumann, B., & Moeller, R. (2006). On scene interpretation with Description Logics. In Nagel, H.-H. & Christensen, H. (Eds.), *Cognitive Vision Systems*, , Springer, 247-275

Reiter, R., & Mackworth, A. (1987): The Logic of Depiction. TR 87-23, Dept. Computer Science, Univ. of British Columbia, Vancouver, Canada

Riboni, D., & Bettini, C. (2011a). COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing 15*, 271-289

Riboni, D., & Bettini, C. (2011b). OWL 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing*, doi: 10.1016/j.pmcj.2011.02.001

Rimey, R.D. (1993). Control of selective perception using Bayes Nets and Decision Theory. Dissertation, TR 468, 1993, Univ. of Rochester, USA

Russell, S., & Norvig, P. (3rd edition) (2010). Artificial Intelligence: A modern approach. Pearson, 266

Schroeder, C., & Neumann, B. (1996). On the logics of image interpretation: Model-construction in a formal knowledge-representation framework. In Proc. Int. Conf. Image Processing (ICIP-96), IEEE Computer Society, Vol. 2/785-788

Shanahan, M. (2005). Perception as abduction: Turning sensor data into meaningful representation. Cognitive Science, 29, 103–134.

Shearer, R., Motik, B., & Horrocks, I. (2008). HermiT: A highly-efficient OWL reasoner. Proc. 5th OWLED Workshop on OWL: Experiences and Directions, 7, 2008

Tsotsos, J.K. , Mylopoulos, J., Covvey, H.D., & Zucker, S.W. (1980). A framework for visual motion understanding. IEEE PAMI-2, 563-573

Vila, L. (1994). A survey on Temporal Reasoning in Artificial Intelligence. AI Communications 7 (1), 4-28