Generation of Rules from Ontologies for High-Level Scene Interpretation

Wilfried Bohlken and Bernd Neumann

Cognitive Systems Laboratory, Department Informatik, University of Hamburg
22527 Hamburg, Germany
{bohlken|neumann}@informatik.uni-hamburg.de

Abstract. In this paper, a novel architecture for high-level scene interpretation is introduced, which is based on the generation of rules from an OWL-DL ontology. It is shown that the object-centered structure of the ontology can be transformed into a rule-based system in a native and systematic way. Furthermore the integration of constraints - which are essential for scene interpretation - is demonstrated with a temporal constraint net, and it is shown how parallel computing of alternatives can be realised. First results are given using examples of airport activities.

1 Introduction

High-level scene interpretation can be roughly defined as understanding images or video streams at abstraction levels above single objects. Typical tasks are traffic scene interpretation in driver assistance systems, criminal acts recognition, and other monitoring tasks such as airport activity recognition, which is used as an example domain in this paper. Scene interpretation systems are typically conceived as knowledge-based systems where extensive high-level knowledge is modelled using declarative knowledge representation techniques. So far, no standard architecture has emerged. In his long-standing work on traffic scene interpretation [7, 1, 2], Nagel developed situation-graph trees, where hierarchically organized frame-based state descriptions of traffic situations are embedded in a state-transition structure. A similar structure was also realised by [3, 4, 8] in terms of scenarios for recognizing bank robberies or airport activities. Compositional and taxonomical hierarchies of structure-based configuration systems as a framework for flexible scene interpretation strategies realising both bottom-up and top-down interpretation steps are proposed in [10]. Using a hierarchical framework ranging from the pixel level to high-level semantic structures, a grammar-based scene interpretation system was developed [17].

The usefulness of high-level symbolic scene interpretation on top of low-level image processing for activity recognition in video streams was demonstrated by [6, 16]. In this approach the activity models are represented in a self-made, non-standardized formalism. Scene interpretation for more complex activities, considered in this paper, calls for for well-founded knowledge representation and standardized inference procedures.

This was the motivation to investigate the use of Description Logics (DL) for scene interpretation [13] and multimedia interpretation and retrieval [11]. It was shown that a DL system can transparently represent compositional and taxonomical hierarchies (which provide the backbone for scene interpretation) in an object-centered manner, and that several DL inference services (such as inheritance and classification) can be exploited for the interpretation process. On the other hand, it is difficult in a DL system to represent constraints between objects, which are often decisive for defining and recognizing high-level entities. Furthermore, a DL system does not provide a framework for flexible, stepwise scene interpretation as required for complex applications. In this paper, we propose a novel architecture for high-level scene interpretation which exploits well-founded object-centered knowledge representation using OWL-DL, but avoids the limitations of DL systems by transforming knowledge structures into rules in the rule-based system Jess. This approach promises several advantages:

- High-level knowledge can be represented in OWL in an object-centered transparent manner, scaling well to large knowledge bases.
- The consistency of the knowledge base can be checked automatically using a DL reasoner connected to OWL (such as Pellet or Racer).
- Logic-based inferences implied by the OWL representation can be automatically translated into corresponding rules and inheritance mechanisms of Jess, realising the skeleton of a scene interpretation system.
- Constraint processing and other procedural components not representable in OWL can be realised in the Java background of Jess.
- Data-driven rule-based processing facilitates flexible interpretation as required for highly variable scenes and realistic image analysis results.

Note that in this architecture, as in most other approaches, high-level scene interpretation is conceived as a process, which takes low-level image analysis results in terms of primitive objects as input and delivers assertions about the scene as output, for example "Aircraft refueling has begun at 13:02:23" for airport activity monitoring. As in many applications objects cannot be recognized reliably, it remains the task of high-level interpretation to disambiguate or even correct low-level classifications. This must be kept in mind when devising high-level inference rules.

A basic interpretation step is recognizing an aggregate from its parts in accordance with the compositional hierarchy defined in the OWL knowledge base. In Section 2, we describe how OWL aggregates are transformed into Jess rules providing such interpretation steps. Our implementation differs from an automatic transformation of OWL to Jess described in [5] in several respects and promises advantages with respect to scalability and generality. We also show how the constraint checks required for aggregate instantiations are embedded into the rules to be executed in the Java background system.

In Section 3, we describe the basic bottom-up interpretation process based on the rules generated from the compositional hierarchy. Here, one of the problems is conflict resolution when several rules can fire. This must be expected throughout

the interpretation process because of several possible lines of interpretation in the face of partial or unspecific evidence. It is shown how parallel interpretation threads can be generated automatically which terminate either with a valid interpretation or incomplete as dead ends.

In Section 4, we present an extended example from the airport activity domain. Section 5, finally, concludes with a discussion of the results and information about future work.

2 Rule Generation from Ontology

In the first part of this section, the usage of OWL-DL for an ontology for scene interpretation is motivated. Then aggregates are introduced as the main representational units of the ontology and finally the transformation of aggregates to Jess rules is presented. In the second part the integration of constraints into the rule-based system is described using a temporal constraint net.

2.1 Object-Centered Definition of Aggregates

Ontology and OWL DL. To describe scenes at a high conceptual level requires the expressiveness to define concepts like objects and events (e.g. for interpretation of video sequences), together with their properties and relations (e.g. temporal and spatial relations between parts of the scene) [13]. A description language for a conceptual knowledge base, which is used for scene interpretation, has to provide the expressiveness to satisfy these requirements. On the other hand, an ontology, which is developed for the purpose of scene interpretation tasks, is typically large and difficult to maintain manually. Therefore it is necessary that a reasoner (like Pellet or Racer) is available to perform automatic checks, e.g. with respect to class consistency, class equivalence, sub-class relation, disjunctiveness and global consistency. The description language OWL DL provides the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will theoretically finish in finite time). Nevertheless the expressiveness is not sufficient for our purposes, particularly to specify n-ary constraints, but this gap can be closed with the OWL extension SWRL (*Semantic Web Rule Language*).[1] All in all, OWL DL has many desirable properties for our purposes, and in the following we assume that OWL DL is used for our ontology representation. It is also worth noting that OWL ontologies have become increasingly popular over the last years, primarily because of the idea of the *Semantic Web*, which increases the availability of tools for creating, publicising and distributing ontologies.

Aggregates. The main concepts in our OWL ontology are *physical* objects, for example `Vehicle`, `Person` or `Equipment`, and *conceptual* objects. These are more abstract objects, for example *events* as `Vehicle-Enters-Zone` or `Refueling`.

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

Concepts are related to each other by super-class and sub-class relations, thus, forming a *taxonomy*. Other essential relations for a knowledge base for scene interpretation are the compositional relations, which express that a concept may have other concepts as parts inducing a compositional hierarchy. For example, the conceptual object `Vehicle-Enters-Zone` is composed of the conceptual objects `Vehicle-Outside-Zone` and `Vehicle-Inside-Zone`. Instances (or individuals in OWL terms) of these parts have to be in a specific temporal relation: an instance of `Vehicle-Outside-Zone` has to occur before the instance of `Vehicle-Inside-Zone`. Another constraint is that the respective instances of the physical objects `Vehicle` and `Zone` of both events are the same. These are the *conceptual constraints*. A concept and its parts together with the conceptual constraints form an *aggregate*, given by the following generic structure in a description logic setting [13]:

$$\begin{aligned} Aggregate\_Concept \equiv\ & Parent\_Concept_1 \sqcap ... \sqcap Parent\_Concept_n \sqcap \\ & \exists_{\geq m_1} hasPartRole.Part\_Concept_1 \sqcap \\ & ... \\ & \exists_{\geq m_k} hasPartRole.Part\_Concept_k \sqcap \\ & conceptual\ constraints \end{aligned}$$

In Figure 1, a simplified extract of the OWL ontology, used for modelling airport activities, is shown as a screenshot of Protégé, as pure OWL notation is not convenient to read. In the left frame the taxonomy is shown and in the right frame the properties of the selected conceptual object `Vehicle-Enters-Zone` are displayed.

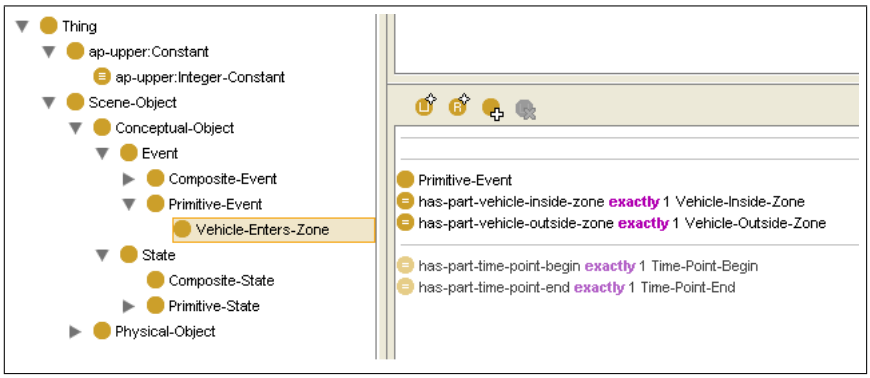

Fig. 1. OWL ontology in Protégé.

For modelling the *conceptual constraints* we use SWRL. An example of a SWRL rule, which expresses the conceptual constraint that instances of the physical objects `Vehicle` and `Zone` have to be the same in both events, is given

below (for simplification the temporal constraints are omitted here.

```
Vehicle-Enters-Zone(?vez) ^
has-part-vehicle-inside-zone(?vez, ?viz) ^
has-part-vehicle(?viz, ?v1) ^
has-part-zone(?viz, ?z1) ^
has-part-vehicle-outside-zone(?vez, ?voz) ^
has-part-vehicle(?voz, ?v2) ^
has-part-zone(?voz, ?z2) ^
->
swrlb:equal(?v1, ?v2) ^
swrlb:equal(?z1, ?z2)
```

In this way an aggregate hierarchy is modelled with primitive aggregates - like `Vehicle-Inside-Zone` - as leaves, which can be directly instantiated based on visual attributes of physical objects computed by the perceptual components of the scene interpretation system (see Section 3.1), and more complex aggregates, defined with aggregates as parts.

Transforming Aggregates to Jess Rules. In this paragraph it will be described, how aggregates can be transformed into rules for the rule-engine Jess in a systematic, automatable way, realising possible interpretation steps and sustaining the object-centered structure of aggregates.

Scene interpretation cannot be solely modelled as deduction. It has been shown in [14] that constructing a scene interpretation is essentially a search problem in the space of possible interpretations defined by the taxonomical and compositional relations by incrementally instantiating concepts while maintaining consistency. Four kinds of interpretations steps are necessary:

- 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 this paper, we will focus on the first step - aggregate instantiation - which is a bottom-up step and the backbone for scene interpretation.

A main structuring feature of the Jess rule language is a *template*, which can be seen as analogon to a Java class. A template is defined by a name and a number of *slots*, which are comparable to member variables of a Java class. In the first step of the transformation of aggregates to Jess, every concept is defined by a template with the name of the concept. The slots of the template are defined corresponding to the properties of the concept with an additional slot *name*, which holds the name of the instance (e.g. `vehicle_17`). Here our approach differs from the transformation of OWL and SWRL to Jess described in [5], where the properties are modelled as *ordered facts*. Ordered facts are simply Jess lists, which perform an implied template creation. This would mean to lose the object-centered structure of an aggregate, as the properties are decoupled

from the concept template. The other significant difference is that we keep the OWL taxonomy by defining the templates with `extends`, which is used to express inheritance in Jess. In this way the OWL subclass relation of class C and D

$$C \sqsubseteq D \tag{1}$$

is directly transformed into the template inheritance structure of Jess. In the realisation described in [5] the template structure is flat and the taxonomy is emulated by duplicating the facts along the taxonomical hierarchy, which could lead to problems with scalability.

In the second step of the transformation, a rule is defined for every aggregate of the OWL ontology. In the predicate part (LHS) of the rule, the parts of the aggregate are listed together with the slots which are needed to express the conceptual constraints, as far as possible. With `part-of` relations it can be checked that a part is not already integrated into another aggregate instance. Constraints that go beyond the scope of a single aggregate - for example temporal constraints - are processed procedurally in the Java part of the system and appear in the predicate part of the rule as a test function (*test conditional element*). This will be described in detail in Section 2.2. In the action part (RHS) of the rule, the aggregate is instantiated, properties are modified accordingly, and the temporal constraint net is updated.

An example for a Jess rule for the (simplified) aggregate `Refueling` is given below[2] (the temporal constraint processing is only sketched here).

```
(defrule Refueling
   ?tez-id <-
   (Tanker-Enters-Zone (name ?tez)
      (has-part-tanker ?v1)
      (has-part-zone ?z1)
      (part-of-refueling nil))
   ?dr-id <-
   (Do-Refuel (name ?dr)
      (has-part-tanker ?v1)
      (has-part-zone ?z1)
      (part-of-refueling nil)
   ?tlz-id <-
   (Tanker-Leaves-Zone (name ?tlz)
      (has-part-tanker ?v1)
      (has-part-zone ?z1)
      (part-of-refueling nil))
   ;; check temporal constraints in a test function
   =>
   ;; create new instance of Refueling
   (assert
      (Refueling (name ?rf-new)
```

[2] `?x-id<-` is an identifier needed for modification of facts in the RHS part of a rule.

```
        (has-part-tanker-enters-zone ?tez)
        (has-part-do-refuel ?dr)
        (has-part-tanker-leaves-zone ?tlz)))
   ;; modify properties of parts
   (modify ?tez-id (part-of-refueling ?rf-new))
   (modify ?dr-id (part-of-refueling ?rf-new))
   (modify ?tlz-id (part-of-refueling ?rf-new))
   ;; update temporal constraint net
)
```

2.2 Constraints

Spatial and temporal context play a special part in scene interpretation. But as already mentioned and shown in [13], it is difficult in a DL system to represent constraints between conceptual objects, and a DL system does not provide a framework for flexible, stepwise scene interpretation. In this section the integration of a global *temporal constraint net* is introduced which controls the activation of rules and stepwise aggregate instantiations, maintaining consistency of the temporal constraints.

Temporal Constraint Net. Temporal constraints are essential in a domain like airport activity monitoring. For the modelling of temporal relations, we use the convex time point algebra [15]. The *Allen temporal operators* used in the SWRLTemporalOntology[3] are not expressive enough for our purposes, because they only allow the modelling of qualitative relations, whereas the complexity of our domain requires quantitative models.

The basic format of a temporal relation in the convex time point algebra is

$$t_1 \geq t_2 + c_{12} \tag{2}$$

where t1 and t2 are interval-valued time points and c12 is an integer-valued constant. Using such inequalities, it is possible to model important features of the temporal structure of a scene model.

Figure 2 illustrates a more detailed aggregate of `Refueling`. In the OWL ontology every concept has two temporal data type properties: `has-start-time` for the beginning time point (`x-tb`) and `has-finish-time` for the ending time point (`x-te`). Other properties are not listed here. The temporal constraints are as follows. A `Tanker-Enters-Fuel-Access-Area` event has to occur before a `Tanker-Stopped-In-Fuel-Access-Area` event, which has to happen before a `Tanker-Leaves-Fuel-Access-Area` event. A `Handler-Plugged-Fuel` event has to occur before a `Handler-Unplugged-Fuel` event. Both events occur during the `Tanker-Stopped-In-Fuel-Access-Area` event. Every event has to fulfill a certain duration. Analog to the conceptual constraints in Section 2.1 also these

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

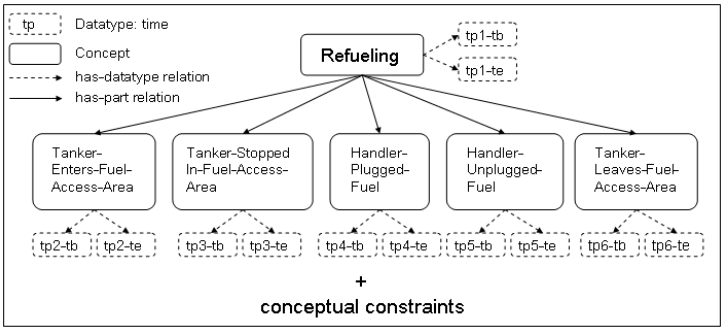

Fig. 2. Aggregate `Refueling` with time properties.

temporal constraints can be expressed with SWRL rules in the form of inequalities, mentioned in (2). Part of the SWRL rule for the `Refueling` aggregate which concerns the temporal constraints of `Tanker-Stopped-In-Fuel-Access-Area` is given below.

```
Refueling(?rf) ^
has-part-tanker-stopped-in-fuel-access-area(?rf, ?tsifaa) ^
duration-of-tanker-stopped-in-fuel-access-area(?rf, ?tsifaa-dur) ^
has-Start-Time(?tsifaa, tsifaa-tb) ^
has-Finish-Time(?tsifaa, tsifaa-te) ^
swrlb:add(?sum-tb-dur, ?tsifaa-tb, ?tsifaa-dur) ^
swrlb:greaterThanOrEqual(?tsifaa-te, ?sum-tb-dur) ^
...
```

Beside the transformation of aggregates to Jess rules, the transformation process must also generate a *temporal constraint net* (TCN) out of the SWRL rules. The outcome is a single global TCN which includes all aggregates modelled in the OWL ontology. An extract of the TCN concerning the `Refueling` aggregate, is shown in Figure 3.

The nodes are time points corresponding to the time marks given in the SWRL rules. The directed arcs represent inequalities, marked with an offset which represents the ideal value of the duration. Each node is interval-valued, where the interval denotes the range of the time points which is consistent with the constraints. Initially the intervals are open-ended, i.e. $[-\infty\ +\infty]$. When an aggregate is instantiated - that is when a rule fires - the corresponding nodes will receive concrete values. For example, if a `Tanker-Stopped-In-Fuel-Access-Area` event starts at 27, then the time point `tp3-tb` will receive the value [27 27]. New values are propagated through the constraint net as follows [12]:

- minima in edge direction: $t_{2min}' = \max(t_{2min}, t_{1min} + c_{12})$.
- maxima against edge direction: $t_{1max}' = \min(t_{1max}, t_{2max} - c_{12})$.

A TCN is inconsistent, if for any node $t_{min} > t_{max}$ holds.

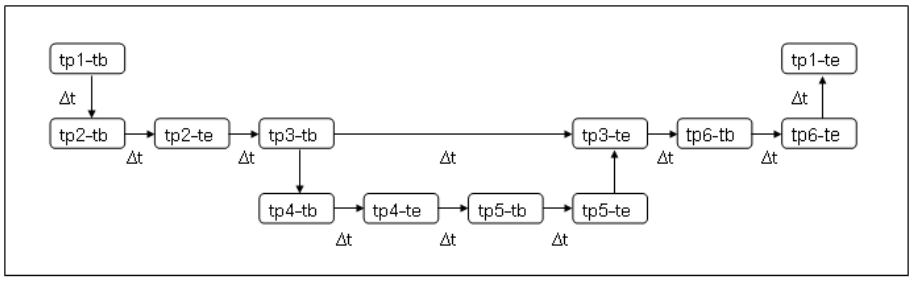

Fig. 3. Temporal constraint net for `Refueling`.

Implementation of Temporal Constraint Net with Shadow Facts. As the propagation of values through the constraint net has a procedural character and can effect all time points - i.e. not only the time points in one single aggregate - it is reasonable and efficient to implement this in Java, whereas a pure Jess implementation would be unnecessary complex and intransparent.

To establish a connection between the Java implementation and Jess objects, all time points and the TCN itself are implemented as *shadow facts* [9], i.e. every time point in Jess has a corresponding time point object in Java.[4] The usage of shadow facts enables Jess to perform reasoning about Java objects. Together with the integration of the TCN into the rules, a general structure of an aggregate bottom-up rule can be given schematically as follows (t_{xy} denote time points):

```
(defrule Aggregate-X-Rule
   - part1 with t11, t12
   - part2 with t21, t22
   ...
   (TCN (tMins $?tMins) (tMaxs $?tMaxs) (OBJECT ?tcn-obj))
   (test (call ?tcn-obj propagateAndCheckConsistency
      $?tMins
      $?tMaxs
      t11, t12, t21, t22,...))
   =>
   - instantiate aggregate X with t31, t32
   - modify part-of relations of parts
   (call ?tcn-obj update
      $?tMins
      $?tMaxs
      t11, t12, t21, t22,..., t31, t32))
```

[4] Every shadow fact has a slot OBJECT which holds a reference to the Java object itself.

When the TCN is initialised, all time points are "normal" objects (not connected to Jess facts). At the moment a rule is matched against the working memory, the `propagateAndCheckConsistency` function in the LHS part of the rule propagates the time point values of the parts of the aggregate through a copy of the original TCN (because the original TCN must not be changed). If the TCN becomes inconsistent the function returns `false`, that means it cannot be satisfied with the currently checked instances, thus the rule must not be activated. If the function returns `true` (and all other constraints of the LHS are fulfilled), then the rule will be activated. If the rule fires, the `update` function in the RHS part of the rule integrates the time points into the original TCN (now these nodes are shadow facts) and propagates the values. In this way the instantiation of aggregates is achieved, while maintaining consistency of the temporal constraints.

Here it must be mentioned, that a pure Jess implementation of the temporal constraint net would be possible in principle with Jess functions. But the realisation of the TCN introduced here is only a preliminary stage for a more sophisticated component where probability distributions replace crisp time intervals so that a context-dependent certainty value can be generated for activated rules. This calls for more complex computations not easily realisable in Jess.

3 Interpretation Process

In this section, we describe the basic bottom-up interpretation process based on the rules generated from the OWL ontology. In the first part, a system overview of a general scene interpretation system is given and the interpretation process is described. In the second part, it is demonstrated, how parallel processing of interpretations can be realised.

3.1 Interpretation Process and System Overview

A basic framework for high-level scene interpretation can be subdivided into three main layers:

- The segmentation and tracking unit (low-level processing layer).
- The metric-symbolic interface (middle layer).
- The high-level interpretation layer.

In the segmentation and tracking unit, static or moving objects are detected by low-level image processing components. The objects are classified into *view* types. A view is a representation of the visual evidence of a physical object. Objects are tracked throughout image sequences, and object trajectories are computed for moving objects. In the middle layer, primitive aggregates are computed. In our domain of airport activities these are primitive states (which describe properties of physical objects that are true for a given time interval), like `Vehicle-Stopped-In-Zone` and primitive events (which describe one or several change(s) of properties of physical objects in a time interval), like

`Vehicle-Enters-Zone`. These primitive aggregates serve as input for our rule-based high-level interpretation layer. With several interpretation steps, mentioned in Section 2.1, and the usage of conceptual knowledge, the interpretation layer performs the inference of high-level aggregates which represent assertions about complex activities in the scene. The usefulness of this architecture for scene interpretation was already demonstrated by [6].

In the initialisation (or offline) phase of the system, the concepts of the conceptual knowledge base are transformed to templates and aggregates are transformed to rules, all written to data files. An initialisation file for the temporal constraint net is generated out of the SWRL rules. These files together form the *Jess conceptual knowledge base*. In the working (or online) phase of the system, these data files are read by the Java application with the embedded Jess engine. The templates and rules are added to the engine, a temporal constraint net is initialised and also added to the Jess engine as a shadow fact. Now the system is ready to process the primitive aggregates provided by the middle layer. In the present stage of the project, the primitive aggregates are read from XML files, in the future they will be provided by a CORBA interface. Corresponding to the time marks given in the XML files, the primitive aggregates are added successively as facts to the working memory of the Jess engine, simulating an evolving scene. Then the *agenda*, i.e. the list of *activations* (rules that can fire when the engine is started) of the Jess engine is analysed. If the agenda is not empty, the command is given to run the engine. The rules fire and add new facts, representing instances of higher level aggregates, to the working memory. Continuing this in a loop, more and more aggregates - defined higher up in the hierarchy and representing more complex activities - are instantiated, consistent with the corresponding conceptual constraints (see Figure 4).

This way a framework for stepwise scene interpretation is realised. The consistency of the rules is guaranteed as far as possible, as they are generated from an OWL ontology which provides automatic consistency checks (except for SWRL rules).

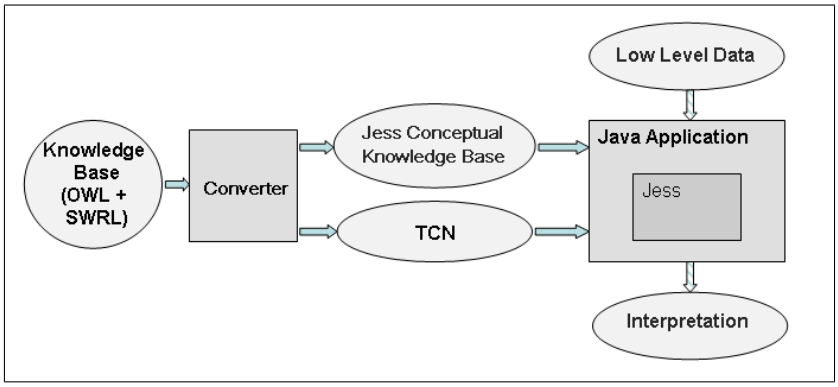

Fig. 4. Architecture of interpretation system.

3.2 Parallelisation

In this section the necessity of parallel computing in the scene interpretation process is motivated and the technical realisation is demonstrated.

As mentioned before, constructing a scene interpretation is essentially a search problem in the space of possible interpretations. In a real-time scene interpretation system, e.g. for airport activity monitoring, it cannot be avoided that evidence is processed incrementally. That means, early interpretation steps may be ambiguous because of lack of supporting context. This problem can be solved either by allowing backtracking to undo faulty decisions, or by parallel computing to follow several alternatives. We will show that parallel computing can be implemented in a transparent and efficient way, using Jess.

In our domain of airport activities it is not unusual that an instance of an aggregate, for example an instance of `Vehicle-Enters-Zone`, could be a part of one of several different instantiations (see Figure 5).

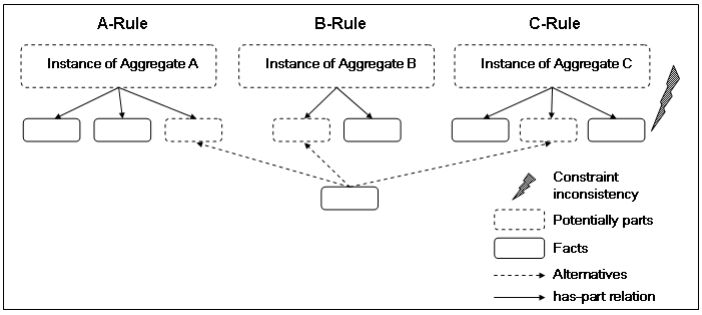

Fig. 5. Possible alternatives in an interpretation step.

Assuming that the conceptual constraints are only fulfilled in rule A and rule B, both will be activated and put onto the agenda. Rule C will not be activated. Because the fact is exclusively part of either instance A or instance B, the order in which the rules fire is decisive for the result: if rule A fires, then rule B will be deactivated and vice versa (this is controlled by the `part-of` relation, mentioned above). To follow both alternative interpretation paths, the actual Jess engine is cloned in this situation. By using the Jess mechanism of serialisation and deserialisation, it is ensured that correct (deep) copies of Java objects, implemented as shadow facts, are created. After cloning the Jess engine, rule A and rule B are activated in both engines. Now, we want to fire rule A in clone_1 and rule B in clone_2. This can be achieved by using a special conflict strategy which can be easily set in the Jess engine to manipulate the execution engine accordingly. Concretely, to explicitly fire a certain rule, a strategy is set, which gives the priority to the activation with a certain activation name (this name is unique). Then both engines are executed in different threads. Directly after the first rule has fired in a clone, the strategy is reset to the original strategy

(this can be done by an event handler). When the `setStrategy` function of the Jess engine returns, all remaining activations are re-ordered, according to the new (original) strategy.

In the next loop, new facts, provided by the middle layer, will be added to the working memory of every clone. If the TCN is not satisfiable anymore, then the thread dies. It can be assumed that in the beginning of the scene, the initial thread branches into several parallel threads very quickly, as there is less context information. But with preceding evolution of the scene the number of threads will decrease. For example a primitive event like `Person-Enters-Zone` can be a part of various events, whereas higher aggregates like `Refueling` are only part of one or two higher events. Future experience will show which maximal number of threads is useful.

4 Results

In this section a simple example of a scene interpretation process is demonstrated in the domain of airport activities which is the application domain in in the EU project *Co-Friend*.[5]

For our experiment we assume a simplified aggregate `Refueling` with the parts `Tanker-Enters-Zone`, `Do-Refuel` and `Tanker-Leaves-Zone`. Another aggregate `Tanker-Enters-And-Leaves-Zone` consists only of the parts `Tanker-Enters-Zone` and `Tanker-Leaves-Zone`, both with their respective conceptual constraints. The second aggregate is a model for the activity that a tanker enters and leaves the specific zone without refueling the aircraft for any reason. Normally an assured evidence for `Do-Refuel` should inhibit the activation of the rule `Tanker-Enters-And-Leaves-Zone-Rule`, but in future work other interpretation steps - beside bottom-up - will be realised which also include hypothesising facts, for example, in cases where evidence is missing because of occlusion. Hence, in general it could make sense to follow both alternatives.

An OWL-DL ontology, including the aggregates and physical objects described above, has been created with Protégé 3.4. The global consistency of the ontology has been checked with the OWL reasoner RacerPro 2.0. SWRL rules have been defined to express the conceptual constraints of the aggregates with the integrated SWRL functionality of Protégé 3.4.

The data files for the rules and the templates have been generated manually. The instances of the three aggregates are read from an XML data file (for simplification we assume `Do-Refuel` to be a primitive aggregate here) and added to the Jess engine one after another.

In Figure 6, an extract of the output of the experiment is shown. It can be seen that as soon as the instance of `Tanker-Leaves-Zone` is added to the working memory, the `Refueling-Rule` and the `Tanker-Enters-And-Leaves-Zone-Rule` are activated and put onto the agenda in engine_1. Then the engine is cloned,

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

thus, a new engine_2 is created with the same status. Then both engines are executed. As desired, the `Refueling-Rule` fires in engine_1, and the `Tanker-Enters-And-Leaves-Zone-Rule` fires in engine_2. Furthermore, the instantiation of `Refueling` results in adding a `Refueling` fact to the working memory in engine_1, and a `Tanker-Enters-And-Leaves-Zone` fact is added in engine_2.

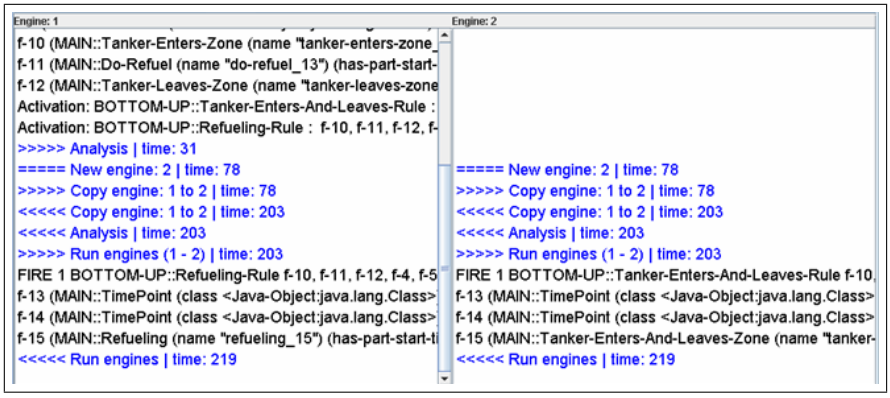

Fig. 6. Output for example of interpretation process.

5 Conclusion and Future Work

In this paper we have presented a novel architecture for high-level scene interpretation, which is based on the generation of rules from an OWL-DL ontology. It has been shown how aggregates can be transformed into a rule base of Jess in a systematic way and how a global temporal constraint net can be integrated. A general rule pattern has been given for the transformation of aggregates into bottom-up interpretation rules which provide the backbone of scene interpretation. Furthermore, the usage of these rules in the scene interpretation process was explained with examples of airport activities. The technical functionality of parallel computing, with the intention to follow alternative interpretation steps, has been shown and a first simple experiment was demonstrated.

In ongoing work, rule generations for the remaining interpretation steps, i.e. aggregate expansion, instance specialisation and instance merging, are elaborated. All transformations from the OWL ontology into rules, templates, and the temporal constraint net will be fully automated.

As an advanced use of rule-based processing, the inclusion of *common sense* inferences will be investigated. The goal here is to conclude missing facts not provided by low-level image analysis by rules which reflect every-day human experiences, for example about natural motion of physical objects.

The original conflict strategy of Jess will be replaced by a probabilistic strategy to provide a preference measure for interpretations steps. In this way the

most promising alternatives of interpretations will be traced in parallel as a beam search. Finally more complex experiments will be performed with real input data obtained from aircraft activities captured at Toulouse Airport in project Co-Friend.

References

1. Arens, M., H.-H. Nagel, H.-H.: Behavioural Knowledge Representation for the Understanding and Creation of Video Sequences. In *Proc. 26th German Conf. on Artificial Intelligence* (KI-2003), LNCS 2821, Springer, 149-163, 2003.
2. Arens M., Ottlik A., Nagel H.-H.: Using Behavioral Knowledge for Situated Prediction of Movements. In *Proc. 27th German Conference on Artificial Intelligence* (KI-2004), Springer LNAI 3238, 141-155, 2004.
3. Borg, M., Thirde D., Ferryman J., Fusier F., Valentin V., Brémond F., Thonnat, M.: A Real-Time Scene Understanding System for Airport Apron Monitoring. In *Proc. of IEEE International Conference on Computer Vision Systems* (ICVS-06), 2006.
4. Bremond, F., Thonnat, M., Zuniga, M.: Video Understanding Framework for Automatic Behavior Recognition. *Behaviour Research Methods* 3, 38, 416-426, 2006.
5. Eriksson, H.: Using JessTab to Integrate Protégé and Jess. *IEEE Intelligent Systems* 18(2), 43-50, 2003.
6. Fusier, F., Valentin, V., Brémond, F., Thonnat, M., Borg, M., Thirde, D., Ferryman, J.: Video understanding for complex activity recognition. *Machine Vision and Applications* Volume 18, Numbers 3-4, 167-188, 2007.
7. Gerber, R., Nagel, H.-H.: Occurrence Extraction from Image Sequences of Road Traffic Scenes. In L. van Gool and B. Schiele (eds.), *Proc. Workshop on Cognitive Vision*, Switzerland, 1-8, 2002.
8. Georis B., Mazire M., Brémond F., Thonnat M.: Evaluation and Knowledge Representation Formalisms to Improve Video Understanding. *Proc. ICVS-06*, 2006.
9. Friedman-Hill, E.: *Jess in Action: Java Rule-Based Systems*. Manning, Greenwich, 2003.
10. Hotz, L., Neumann B.: Scene Interpretation as a Configuration Task. *Kuenstliche Intelligenz* 3, BoettcherIT Verlag, 59-65, 2005.
11. Moeller, R., Neumann B.: Ontology-based reasoning techniques for multimedia interpretation and retrieval. In: Y. Kompatsiaris, P. Hobson (Eds.): *Semantic Multimedia and Ontologies: Theory and Applications*, Springer, 55-98, 2008.
12. Neumann, B.: Description of Time-Varying Scenes. *Semantic Structures*, D. Waltz, Ed., Lawrence Erlbaum, 1989.
13. Neumann B., Moeller, R.: On Scene Interpretation with Description Logics. In: *Cognitive Vision Systems*, Springer, LNCS 3948, 247-275, 2006.
14. Neumann, B., Weiss, T.: Navigation through logic-based scene models for high-level scene interpretations. *Proc. 3rd Int. Conf. on Computer Vision Systems* (ICVS-2003), 212-222, 2003.
15. Vila, L.: A survey on Temporal Reasoning in Artifical Intelligence. *AI Communications* 7 (1), 4-28, 1994.
16. Van-Thinh V., Brémond, F., Thonnat, M.: Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Pre-compiled Scenario Models. *Computer Vision Systems*, Springer, LNCS 2626, 523-533, 2003.
17. Zhu S.-C., Mumford, D.: A Stochastic Grammar of Images. *Now Publishers*, 2007.