

Building Petri Nets from Video Event Ontologies

Gal Lavee, Artyom Borzin, Ehud Rivlin, and Michael Rudzsky

Technion - Israel Institute of Technology, Haifa, Israel 32000

Abstract. Video event understanding requires a formalism that can model complex logical temporal and spatial relations between composing sub-events. In this paper we argue that the Petri-Net is such a formalism. We go on to define a methodology for constructing Petri-Net event models from semantic descriptions of events in two well known video event ontology standards, VERL and CASE^E.

1 Introduction

Understanding events in video data is a research area that has received much attention in recent years. Video events are compositional entities made up of smaller sub-events which combine in logical, spatial and temporal relationships. These sub-events can occur in sequence, be partially ordered, or occur altogether asymmetrically. An interesting problem is selecting a formalism that can be used to represent this kind of structure. Given such a formalism events of interest within a scene can be defined in the set up phase of a surveillance system and recognized as they occur.

Other formalisms such as Bayesian networks, Hidden Markov Models and Stochastic Grammars have been suggested as such formalism in previous work. However, these methods do not define semantically meaningful sub-events. Thus it is not straightforward to describe the composition of an event on a semantic level. Furthermore these models are not well equipped to describe the sometimes complex (usually temporal) relationships between the sub-events.

Another problem in the discipline of event understanding is translating user's/expert's knowledge of events of interest in a particular domain into an event model. For the specification of domain knowledge, several ontology languages have been proposed to allow robust definition of events. It has not been established how these event descriptions can be reconciled with the machine formalisms for describing and recognizing events described above.

The main contribution of this paper is a solution towards bridging these two fields of study. Given a description of an event in an ontology language we offer a way of translating this information into a formalism understood by a machine event recognition system. We have chosen the Petri-Net formalism to this end.

This work extends previous work in modeling video events using Petri-Nets by describing a structured approach to constructing the event model. Previous work has illustrated how Petri-Nets are useful and effective for the modeling of

events, however, the principles of construction of the Petri-Nets have generally been ad hoc and expert oriented.

The remainder of this paper is organized as follows: In Section 2 we will briefly define Petri-Nets and discuss their useful properties for event modeling. In Section 3 we will discuss related work and approaches to constructing Petri-Nets for video event understanding. In Section 4 we will discuss our approach for integrating the Petri-Net formalism with standard video event ontology languages. In Section 5 we will show how this formalism can be applied to video input to provide an event summary as output. In Section 6 we will offer conclusions and discussion derived from our experiments.

2 Petri-Nets and Their Properties

A Petri-Net (PN) model is graphically represented by a directed bipartite graph in which the two types of nodes (places and transitions) are drawn as circles, and either bars or boxes, respectively. The arcs of the graph are classified (with respect to transitions) as: input arcs - arrow-headed arcs from places to transitions, output arcs - arrow-headed arcs from transitions to places, inhibitor arcs - circle-headed arcs from places to transitions. Multiple arcs between places and transitions are permitted and annotated with a number specifying their multiplicities. Places can contain tokens that are drawn as black dots within places. The state of a Petri-Net is called marking, and is defined by the number of tokens in each place. The initial Petri-Net state is called the initial marking. For further details interested readers are referred to [1] and [2].

The temporal dimension is of great importance in the domain of video understanding. The possible relationships between temporal intervals, defined by Allen [3], are used to describe the relationships between sub-events within the same event. Petri-Net fragments (sub-networks) modeling each of the seven of Allen's temporal relations are well understood.

Petri-Nets are also a powerful tool for modeling other aspects of video events. It is straightforward to model logical (AND, OR and NOT) relations between sub-events as Petri-Nets fragments. Spatial distance relations can be enforced as transition enabling rules. Petri-Nets may also be used to model concurrency and partial ordering among sub-events. The fragments modeling the relationships between the sub-events can be combined hierarchically to form an event model Petri-Net.

3 Related Work

While it has been noted in the literature that Petri-Nets are a useful tool for modeling the structure of events, it has not been generally agreed upon how to build a particular event model using the Petri-Net formalism. For this reason the design of networks has been rather ad hoc and dependant on decisions of the implementor. However, two classes of approaches have emerged, the Plan Petri-Net and the Object Petri-Net. In this section we will discuss and compare these approaches.

The Object Petri-Net is a term we have defined for the class of event model that appears in [4] and [5]. It is so called because of design choices in constructing the model. In particular, a token in this type of Petri-Net corresponds to a detected object in the scene. Places in Object Petri-Nets represent particular states of the object. Enabling rules in transitions in this model class are conditions on the properties of the object tokens. The events of interest are the transitions themselves and a complex event of interest may lie at the end of a chain of a number of such events. An advantage of this style of network construction is that multiple events of interest can be considered within the same network. Because tokens take on the properties of the objects they represent, this type of network may be considered a type of colored Petri-Net.

Plan Petri-Nets are another class of event model which appear in [6]. In this type of model each place represents a sub-event and the occurrence of this sub-event is represented by a token in this place (only a one token capacity is afforded to each place). The enabling rules of the transitions in this type of network are based on scene states rather than on particular object properties (the tokens in each place represent the existence of scene states rather than particular objects). Plan Petri-Nets model each event of interest as a separate network which has a "sink" transition that indicates whether the event has occurred or not. Each internal transition is the beginning/end of a sub-event.

Seemingly, Object Petri-Net model allow a more robust model due to the fact that they can accommodate multiple tokens for multiple objects and model multiple events within the same event network. However, the complexity necessary to facilitate this is not conducive to a semi-automatic construction of Petri-Nets (Section 4) and hence the simplicity and separation of the Plan Petri-Nets gives them an advantage in this regard.

4 Ontology Languages

The domain of video events is inherently ambiguous and the definition of events of interest is usually left up to the human designers of a particular system. However, these events do have some common aspects that can be isolated. These are usually the compositional structure of the event which includes logical, spatial and temporal relations as well as hierarchical structure and causality. This observation has led to the development of video event description ontologies such as VERL [7] and CASE^E [8]. These ontologies allow a straightforward description of the event which can then be used to construct the event model. In this section we argue that these descriptions can easily be transformed into a Petri-Net model which can then be used to analyze video events.

4.1 Video Event Representation Language

The Video Event Representation Language (VERL) was introduced in [7] as representation of video events based on the Event Representation Language. Nevatia et al. explain the representation scheme and illustrate its use through

examples such as a tailgating scenario in a secure facility entryway. The emphasis of this ontology language is to capture relations such as temporal and logical relationships and distinguish between single-thread (linearly ordered sub-events) and multi-thread (simultaneously occurring sub-events). The ontology language assumes that "primitive" events have been defined and are taken care of by a lower level. As we have mentioned in previous sections these aspects are well modeled by the Petri-Net formalism and thus it is reasonable to conclude that an event structure detailed in the VERL representation can be modeled into a Petri-Net. Of course, just as there are many ways to construct a Petri-Net there are equally many ways to transform the ontology representation into a suitable Petri-Net model. We propose one such transform. The advantage of having this transform is that similar events will have similar Petri-Net representations rather than ad hoc representations as we have seen in the past. We describe this technique in more detail with an example in section 4.3.

4.2 CASE^E Ontology

CASE^E is another ontology scheme proposed in [8]. It is based on a natural language scheme proposed by Fillmore in the 1960s [9] which utilizes the basic unit of a case frame. Case frames are structural entities which define the skeleton of a particular sentence. The extension proposed in the CASE^E framework is to allow a hierarchy of case frames to define a video event, allow for multiple agents to be included in the event descriptions, and include support for causal relationships. [8] gives an example of defining events in the railroad crossing domain using CASE^E. An event specified within the framework of the CASE^E ontology may also be converted into a Petri-Net representation. An example of this conversion is given in section 4.3.

4.3 Translation of Ontology Description to Petri-Net Representation

In this section we discuss our approaches for translating the ontology language event description into an equivalent Petri-Net model. It is important to point out that what we propose here is not an automated process, but rather a methodology for constructing Petri-Net event models that seeks to minimize arbitrary design differences between event models representing the same or similar events.

We offer here two different approaches to constructing the event model. The difference between these approaches is the class of Petri-Net event model they produce. One of these approaches corresponds to the Object Petri-Net and the other to the Plan Petri-Net.

We will begin the description of the transform technique by describing a special Petri-Net fragment we call the "sub-event fragment". As the name implies we utilize this fragment to represent sub-events composing our event of interest. This fragment has two versions. If we are not going to apply temporal relations to the fragment then it is sufficient to represent it as a linear sequence of a place, transition and another place. The first place represents the pre-conditions of the

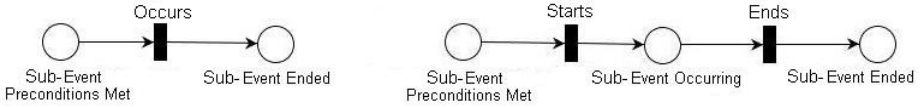


Fig. 1. Sub-Event Fragment

sub-event being met, the transition represents the occurrence of the event, and the final place represents the state of the sub-event having been concluded. We call this fragment the "simple" sub-event fragment. Having such a representation for each sub-event allows us to connect them using logical and spatial relations.

For sub-events to have temporal relations applied to them we extend the fragment into a chain of five nodes. A place node linked to a transition, place, transition and finally a place. The beginning and ending places, as in the smaller fragment, represent the preconditions for a sub-event occurring and the sub-event having concluded, respectively. The additional middle place represents the state of the sub-event currently in progress. The first of the two transitions indicates the start of the sub-event while the second represents the end of the sub-event. This structure allows us to relate fragments using temporal relation constructions (see below). For this reason we have named this type of fragment the "temporal" sub-event fragment. An example of the two possible versions of the sub-event fragment is shown in figure 1.

The next step is to connect the fragment representing the various sub-events in a manner that corresponds to their event description in the ontology language. Simple logical relations are straightforward to attach to the sub-event fragments. Temporal relations must be connected in a way that enforces that the occurrence of the sub-events complies with the temporal ordering defined by the relation. The OVERLAPS relation, for example, requires that sub-event A (the top fragment in figure Figure 2) starts before sub-event B (the bottom fragment) and ends after sub-event B has started but before sub-event B has ended. To enforce this we connect the transition "starts" in fragment A to the "Precondition" place in fragment B (i.e. A having started is a pre-condition for B to start). Similarly, we connect the "Ended" place in fragment A to the "Ends" transition in fragment B (i.e. sub-event B can only end if sub-event A has already ended). Any two sub-events which conform to the temporal relation OVERLAPS will be able to both reach their ending state. We preform a similar construction for each of the remaining of Allen's temporal relations. It may be possible to construct a temporal relation in more than one way. However, choosing a consistent representation for each of the temporal relations will allow us to avoid decisions on these matters when building the event model. Figure 2 illustrates how we have chosen to relate sub-events for each of the temporal relations.

When connecting the sub-event fragments we can fuse nodes that are shared between fragment. For example, if the ending state of a particular sub-event is the precondition of another we can merge the two places into one. As a result of this, some of the places and transitions in the resulting Petri-Net model may

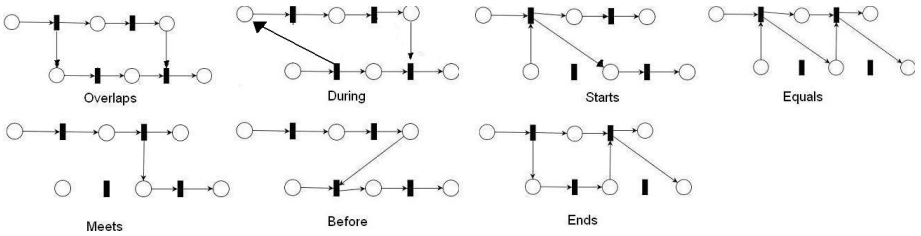


Fig. 2. Petri-Net Fragments Corresponding to Temporal Relations

be redundant. We eliminate such unconnected places and transitions to make a more compact model. The unconnected nodes are shown in the figure to illustrate this idea.

In an Object Petri-Net approach each token represents an object in the system and each place represents a possible state for this type of token. To generalize this intuitive concept into a semi-automatic construction we first define all possible objects and object states of interest to us in a particular event. We must then construct Petri-Nets that model each of these states as a place and the transition between them as a Petri-Net transition. The next step is to add the special fragments representing the structure of the event. The traversal of these structure fragments will be dependant on the objects and object states and thus we make the appropriate connections from the pertinent nodes in the object state transition fragment to their appropriate place in the event structure fragment.

In a Plan Petri-Net each place represents a system state and a token in a place indicates that the system state holds. Transitions can be defined to directly represent atomic sub-events (ontology language primitives). The resulting structure is straightforward to project upon our special event structure fragment. We will illustrate the construction processes of the Plan and Object Petri-Nets using examples from the ontology languages discussed in sections 4.1 and 4.2.

In the first example we consider the scenario of train tracks intersecting a road. Our event of interest is a "safe crossing event", that is when a train approaching has caused the signal to change, the gate to lower and the oncoming car to stop. The train can then proceed to cross the intersection zone safely. The first step in constructing an event model is building the corresponding temporal fragment for each of the sub-events (train approaching, signal change, gate lower and car stop in our example). We then relate these sub-event fragments using the appropriate temporal relations. In our scenario, each of the sub-event are related through the AFTER relation (a variant on Allen's BEFORE) and the CAUSE relation (which we also model using BEFORE) so we connect each of the sub-event fragments in accordance with the BEFORE relation in figure 2. We place a transition node at the conclusion of the fragment string to indicate our safe crossing event has occurred. The resulting construction is illustrated in figure 3. This network is pictured before possible simplification to illustrate the connection of the sub-event fragments to enforce temporal relations.

Once we have this network we can simplify it. Place nodes with with no incoming arcs are removed from the network. The middle transition, place, transition

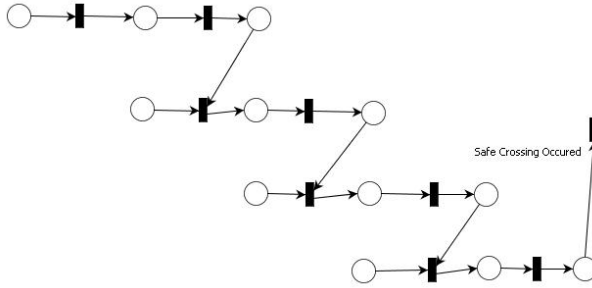


Fig. 3. Safe Crossing Event Plan Petri-Net Before Simplification

sequence of each sub-event fragment can be consolidated into a single transition if it does not affect the event composition structure (i.e. there are no arcs connected to the "event occurring" place, no external outgoing arcs from the "event started" transition and no external incoming arcs to the "event ended" transition.) The resulting simplification of the network in our example is a linear chain pictured in figure 4b (the label SE indicates each of the above mentioned sub-events).

We will now illustrate the construction of an Object Petri-Net representing this event. Our objects of interest in this event are the train, car, signal and gate. This information is given to us by the CASE^E ontology event specification (domain entities). Not provided are the possible states each of these objects can assume. This information may be implied by the domain predicates and domain event specification, however, it can also be added explicitly with a small extension to the ontology event specification. For our purposes we will assume that this object state information is available. The first step in constructing the Object Petri-Net is to construct fragments representing the state transitions of each of our objects of interest. For example a car may take on the states inscene, stopped, inzone2 and stoppedinzone2. Each of these would be represented by a place and the transitions between them would be represented by a transition node with an enabling rule possibly defined on one of the domain predicates (given by the ontology language). Figure 4a shows this construction for the car object. We construct a similar state transition fragment for Train, Gate, and Signal.

The next step is to examine the event construction, construct a Petri-Net fragment for each sub-event and combine them according to the specified relationship. The process of building this fragment is discussed above. Figure 4b shows the simplified Petri-Net fragment representing this structure. Finally we connect the two pieces of our network by connecting the appropriate object in their corresponding place in the event structure chain using arcs. An example of this is attaching the place corresponding to "train_approaching" to the beginning of our event structure fragment to indicate that this state must exist to begin the evaluation of this chain. After these connection we can further simplify the Petri-Net by removing any meaningless places and transitions. The resulting Petri-Net is picture in figure 5.

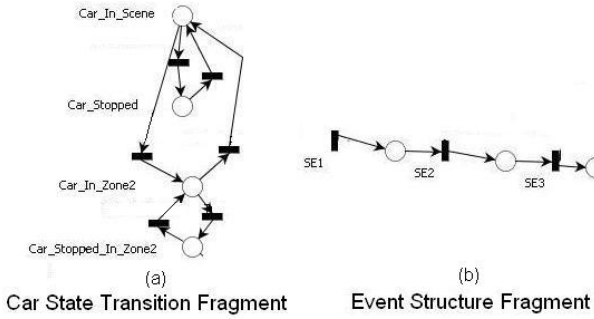


Fig. 4. Object Petri-Net Fragments

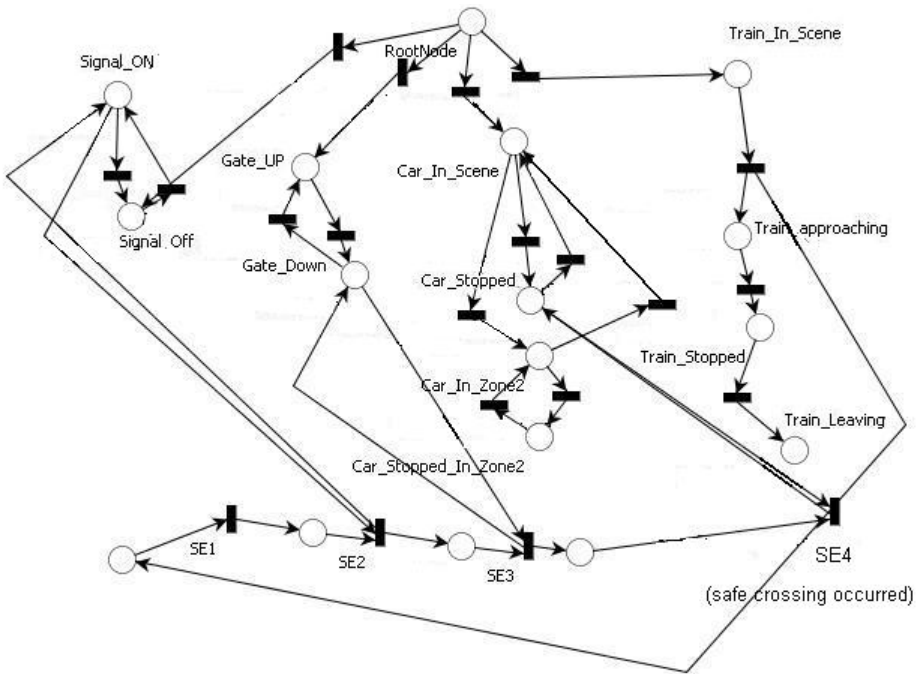


Fig. 5. Safe Crossing Event Object Petri-Net

We will now discuss implementing the same example within the framework of a Plan Petri-Net. This construction results in a more compact network since we model domain predicates as transitions within the event structure fragment. To construct this representation we build the fragment for each of the sub-events and connect them appropriately as we have shown above. In this construction approach we allow a condition on the state of the network to be in the enabling rule of each transition. Thus we can condition the start and end of each of the sub-events on system states and do not need to explicitly link them to object

System Variables: v, frame

```

Process (Turn-Left-At-Intersection(vehicle x, intersection y))
{
  AND (Sequence(
    Enter(x,y),
    AND(exit(x,y), orientation(x)=v+90)
  ),
  duration=currframe-frame <threshold
)
}
    
```

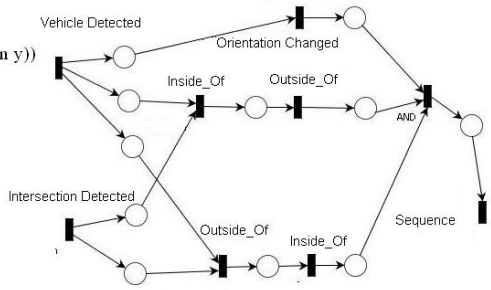


Fig. 6. Left Turn Event Plan Petri-Net

states or model all possible object states. This is an advantage over the Object Petri-Net for events that consider many objects and their joint states as it allows maintaining a much simpler model. The resulting Petri-Net is thus just the event structure fragment with added transition enabling rules indicating the start and end condition of each event. The visualization of the network is the same as that pictured in figure 3.

In the next example we consider an event specified in the VERL ontology framework. Along with our fragments for temporal and logical relations we also define fragments for control structures in VERL such as "sequence" and "change". These structures are illustrated in this example. The scenario we will consider is the road intersection scenario. We wish to determine when a left-turn event has taken place. This event is related to one modeled by Higgins in [10], however we have altered the VERL expression representing the event. This event is unique because it considers object position and orientation at different times. We have decided to approach this as a plan network augmented with system variables. Each of the transitions will have an enabling rule defined by a system state as in the previous example as well as a firing rule which may set system variables. Thus, we construct sub-event fragments for each of the sub-events in our example (we add the implicit `vehicle_detected` and `intersection_detected` along with `enter_intersection`, `exit_intersection` and `orientation_changed`). To this end we use the "simple" sub-event fragments because we do not intend to relate the sub-events using temporal relations. We then connect the sub-event fragments using the defined VERL relations (Sequence, AND, change). Figure 6 shows the VERL notation for the event along with the resulting Petri-Net representation. Note that the "enter" and "exit" processes have been defined using lower-level primitives "inside_of" and "outside_of". Also noteworthy is that the events `vehicle_detected` and `intersection_detected` have multiple place node outputs because they serve as preconditions for multiple sub events.

5 Conclusion

Petri-Nets are a powerful formalism for describing video events for event monitoring systems. In this paper we have discussed how the Petri-Net formalism can

model well the properties inherent to video events such as logical, temporal and spatial composition. We have further discussed the different classes of Petri-Net based event models which appear in the literature. We went on to provide a methodology and examples on how formal ontology language definitions for an event can be transformed into a Petri-Net formalism. The models generated in this paper can be used in an event recognition system based on lower-level vision systems such as object detectors and trackers. The results of such analysis would be a video summary indicating when events of interest occurred and also which semantically meaningful sub-events composed those events. An article describing such results is currently in preparation. The methods described in this paper can in the future be extended to automatically translate ontology specified events to Petri-Net models. Similar methodologies can also be used to translate ontology events into other known formalisms to allow comparison between event models.

References

1. Balbo, G., Conte, G., Donatelli, S., Franceschinis, G., Marsan, M.A.: *Modelling with Generalized Stochastic Petri Nets* (Hardcover). John Wiley & Sons, Inc., New York, NY, USA (1995)
2. <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>
3. Allen, J.F., Ferguson, G.: *Actions and events in interval temporal logic*. Technical Report TR521 (1994)
4. Borzin, A., Rivlin, E., Rudzsky, M.: *Surveillance interpretation using generalized stochastic petri nets*. In: *WIAMIS 2007* (2007)
5. Ghanem, N., DeMenthon, D., Doermann, D., Davis, L.: *Representation and recognition of events in surveillance video using petri nets*, 112 (2004)
6. Castel, C., Chaudron, L., Tessier, C.: *What is going on? a high level interpretation of sequences of images*. In: *ECCV* (1996)
7. Nevatia, R., Hobbs, J., Bolles, B.: *An ontology for video event representation*. In: *CVPRW 2004*, Washington, DC, USA, p. 119. IEEE Computer Society Press, Los Alamitos (2004)
8. Hakeem, A., Sheikh, Y., Shah, M.: *A hierarchical event representation for the analysis of videos*. In: *AAAI* (2004)
9. Fillmore, C.J.: *The case for case*. *Universals in Linguistic Theory*, 1–88 (1968)
10. Higgins, R.P.: *Automatic event recognition for enhanced situational awareness in uav video*. In: *Military Communications Conference* (2005)