Attempting to increase the Performance of Petri net based Situation Recognition

Anders Dahlbom Lars Niklasson Göran Falkman

Informatics Research Centre, University of Skövde,
P.O. Box 408 SE-541 28 Skövde, Sweden

Abstract

Situation recognition is an important problem to solve for introducing new capabilities in surveillance applications. It is concerned with recognizing a priori defined situations of interest, which are characterized as being of temporal and concurrent nature. The purpose is to aid decision makers with focusing on information that is known to likely be important for them, given their goals. Besides the two most important problems: knowing what to recognize and being able to recognize it, there are three main problems coupled to real time recognition of situations. *Computational complexity* — we need to process data and information within bounded time. *Tractability* — human operators must be able to easily understand what is being modelled. *Expressability* — we must be able to express situations at suitable levels of abstraction. In this paper we attempt to lower the computational complexity of a Petri net based approach for situation.

1 Introduction

Situation recognition is an important problem to solve within the surveillance domain, which targets the problem of recognizing a priori defined situations of interest in a continuous flow of data and information. The purpose is to aid decision makers in focusing on data and information that is known to likely be interesting, given their goals. Situation recognition thus targets the information gap [11], in other words, finding the information that is needed, when it is needed. Already, techniques for data and information fusion are seen as key enablers for providing decision support in surveillance applications [1]. These are typically discussed using the Joint Directors of Laboratories (JDL) model for information fusion [16], which distinguishes between signals, objects, situations, and impacts. Situations are essentially collections of related facts consisting of relations between objects [14]. Situation recognition is thus concerned with recognizing patterns in collections of facts. Although situation recognition is acknowledged within the information fusion community [21], the problem is not defined at a level of abstraction suitable for investigating promising solutions. We have previously viewed situation recognition as the task of assembling a list of all situations in a state space consisting of all relations inferred between objects over time, constrained by a situation template $T = (X, C)$, where X is a set of variables for objects, and where C is a set of constraints on the domain (spatiotemporal relations between objects or temporal relations between non-temporal constraints) [7, 5]. A template thus constrains the domain of situations, to result in all situations unifiable over T.

In previous work, we have investigated a Petri net based approach for modelling and recognizing situations [5, 6]. This approach extends previous work by [13, 3, 15], to manage the complete space of partial matches, and for automatically managing role assignment. Petri nets serve as a good foundation for situation recognition since they allow for sequencing, parallelism and synchronization to be easily represented and visualized. These are important aspects when building systems that should support human decision making, since it is of importance that decision makers are able to easily understand the contents of the support [12]. An analysis of the Petri net based approach (and the problem itself) however reveal that it in cases of complex patterns, or in dense and high paced scenarios, consumes much of the available resources. This may be important to look into, since surveillance systems often are intended to operate in real time. We need to be able to process data and information at least on the average rate at which it is made available. In this paper we analyse this problem and investigate a potential solutions for increasing the performance of Petri net based situation recognition, with respect to computational complexity, and at the expense of memory.

1.1 Related work

In 1993, Dousson et al. [9] presented an approach based on propositional reified logic and temporal constraint propagation, for addressing the situation recognition problem within environment surveillance. In later work [8, 10] the focus has shifted to recognizing chronicles in network surveillance applications, and the terminology has been changed accordingly, to chronicle recognition. The main difference between our work and theirs, is that we use relations as primary symbolic concept, and that they explicitly reason about time. Meyer-Delius et al. [17] use hidden Markov models (HMMs) for probabilistic recognition of situations, in their work on intelligent driving assistants. HMMs describing situations consisting of sequences of relations are used on top of dynamic Bayesian networks (for estimating relations). The main difference compared to HMMs is that Petri nets more easily allow situations to consist of multiple simultaneously ongoing activities. Furthermore, it can also be problematic to construct probabilistic transition matrices in a sound way, since labelled data of interesting situations usually does not exist.

Also related is the work on systems diagnosis of discrete event systems, using timed automata [22, 2]. The use of such techniques for situation recognition however requires one to address the problem of using predicates as symbols, which also involves the problem of unification. The work of Walzer [23], who use a rule based approach for complex event recognition, is also related. In that work, the rete algorithm is extended to allow for temporal constraints to be modelled using rules. A potential problem can however be that it is hard to easily understand and define nested rules describing complex situations.

The perhaps most closely related work is that presented in [13, 3, 15, 19]. Ghanem et al. [13] use Petri nets for complex event recognition in video surveillance. Catel et al. [3] use Petri nets for modelling plan and activity prototypes, in their work on automated scene recognition. Lavee et al. [15] use Petri nets for video surveillance. Finally, Perše et al. [19] extend the work of [13, 15], and use Petri nets for multi agent activity recognition, more specifically, to recognize play tactics in basketball games. The main difference with respect to the approach used in this paper, is that tokens do not represent any single aspect in the universe of interest, but rather, it represent unifications of sub parts of an event stream and a modelled situation. The main focus is thus on partial match space modelling and management.

2 Background

2.1 Petri nets

Petri nets are accroding to Murata [18] directed, weighted, bipartite graphs, in which two types of nodes are used: places and transitions. Places correspond to states in finite state automata and transitions correspond to events in flow charts [20]. As in any bipartite graph, there are two disjoint sets of edges, and in Petri nets these consist of edges from places to transitions (input arcs) and edges from transitions to places (output arcs). In a finite state automaton, a single token is used for denoting the current state of the process being modelled. In Petri nets however, places can contain multiple tokens and many places can simultaneously contain tokens, representing sub states of multiple processes. The "global" state of a Petri net, called its marking, consist of all tokens in all places. Transitions are used to change the marking of a Petri net, by consuming tokens from input places and by producing tokens at output placs. In graphical notation, places are drawn as circles, transitions as vertical bars, and tokens as dots inside places. The main strength of Petri nets is according to Sowa [20] their ability to represent parallelism and concurrency.

2.2 Petri nets for recognition

Petri nets are not usually thought of as mechanisms for recognition. Castel et al. [3] however argue that they are quite suited for this since they allow for sequencing, parallelism, and synchronization to be easily represented and vizualised. These aspects can be of major importance when a decision maker is to understand the underlying components of a recognized situation, as well as when defining what we are interested in recognizing through the use of expert knowledge.

In our previous work [5], we have proposed an approach based on Petri nets, for recognizing situations and for managing the space of partial matches coupled to situation recognition. The approach extends previous work by [13, 15, 3] in order to implicitly manage role assignment (which observed object represents which object in a modelled situation) and in order to model the complete space of partial matches between modelled situations and the flow of information.

2.2.1 Representation

To more easily understand the approach, we start with an example of an interesting situation. Picture a shopping zone where a number of pedestrians move around. In this flow of people we are interested in recognizing pick-pocket situations. Imagine a thief and an accomplice. The thief selects a suitable target, approaches this victim, picks its pocket, after which the thief and the accomplice meet each other to hand the stolen goods over. This situation can be modeled with a Petri net as illustrated in figure 1.

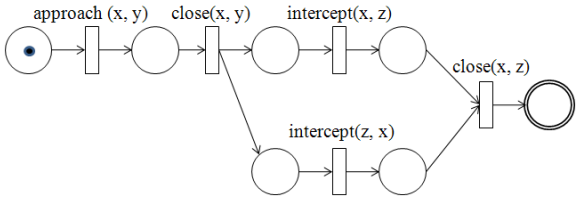

Figure 1: Illustration of a Petri net describing a fictive pick-pocket situation.

Tokens represent partial matches between a situation template and a stream of events. This follows the ideas of coloured Petri nets, in which tokens are used as carriers of information. Recall, a template $T = (X, C)$ consists of a set of variables X and a set of constraints C. A token represents a subset of a template, namely the variables X, and a subset $C^{'}$ of C consisting of all non-temporal constraints in C. Temporal constraints do not need to be modeled since these implicitly are modeled by the structure of a Petri net. Constraints in a token can be bound to predicates of processed events, and variables can be bound to real objects denoted by predicates. Furthermore, tokens can be combined with each other in case there are no variable or constraint bindings that stand in conflict with each other. Three or more tokens can also be combined since token combination is commutative; tokens can thus be combined recursively in any order.

Two types of transitions are used, where the first is activated as soon as a new set of combinable tokens exist in its input places. This type of transition will be referred to as an unconditional transition and will when activated combine all new sets of combinable tokens over their input places. The second type of transition is a conditional transition, introduced by [13], to which constraints can be assigned. These are activated when the condition is fulfilled, and in case a combinable set of tokens exist on in its input places. In contrast to conventional Petri nets, tokens are not consumed from input places when transitions are activated. The reasoning behind this is that we wish to keep the complete matching space. However, the space of partial matches grows rather rapidly when tokens are not consumed, and to counter for this, we have suggested setting a global time bound on a situation template.

Places represent partial stages of the matching procedure between a modeled situation and the stream of events. A token at a specific place thus mean that there is a partial match at that stage in the matching procedure. Input places (no arcs leading to them) are assigned empty partial matches and do not have any arcs leading to them. These serve as initiators for matching new occurrences of a situation, since any transition that only have input places as input always can produce new tokens for every event type that is matched. This solves role assignment implicitly in the Petri net based approach. Match places denotes the existence of complete matches, and as soon as a token is inserted to a match place, a situation has been recognized and we can alert an operator to analyze this information further.

2.2.2 Algorithm

The algorithm for updating the marking of a Petri net for situation recognition follows three distinct steps that are taken for each new event that is processed: (1) partial match space pruning, (2) event-token derivation, and (3) token propagation. In the first step, too old information, with respect to time, is removed. Each place is therefore inspected to remove partial matches that break the global time constraint.

In the second step, the event is processed by each conditional transition to derive a set of new tokens to be inserted to their output places. In case the condition of the transition is matched, new tokens are constructed by combining the new information with all valid combinations of tokens in the input places of the transition. Tokens are however not inserted immediately into their respective output places (new information must not be used by two sequentially ordered transitions), but are instead kept in a list until all conditional transitions have processed the event. The processing in the second step is illustrated in algorithm 1.

Algorithm 1 Derivation of output in a conditional transition t, given a list of partial matches $input$ and an optional source place p_{source}.

DERIVEOUTPUT($t, input, p_{source}$)

```
1: for all input places p_input in t do
2:   if not p_input = p_source then
3:     output ← ∅
4:     for all partial matches pm_1 in input do
5:       for all partial matches pm_2 in p_input do
6:         if pm_1 and pm_2 does not stand in conflict then
7:           output ← COMBINE(pm_1,pm_2)
8:     input ← output
9: return output
```

Finally, in the third step of the update algorithm new partial matches are propagated to their respective output places. Furthermore, tokens are also propagated further on in the Petri net, and for each conditional transition that a token passes, a missed event is bound instead of a real event. This allows us to recognize partial matches with missed events, and this can be an important aspect when working in domains where information can be incomplete. Lastly, unconditional transitions are also invoked when tokens are made available in their input places. The procedure for insertion is illustrated in algorithm 2.

Algorithm 2 Insertion of a partial match pm into a place p, including derivation of output for consecutive non-conditional transitions.

INSERT(p, pm)

```
1: if pm does not exist in p then
2:   if number of missed constraints in pm < max_misses then
3:     add pm to storage in p
4:     for all output transitions t_output from p do
5:       if t_output is a non-conditional transition then
6:         output ← DERIVEOUTPUT(t_output, pm, p)
7:         for all partial matches pm' in output do
8:           for all output places p' in t_output do
9:             INSERT(p', pm')
10:        PROPAGATE(t_output, pm)
```

3 Suggestions for increased performance

An inspection of algorithm 1 reveals that for each event that is matched, we need to try and combine all combinations of input tokens from each respective input place, with a newly produced partial match. In fact, this turns out to be on the order of magnitude of the Cartesian product over the content in the input places, iteratively restricted by the set of matching sub combinations. This is done for each event that matches the relational condition in a transition. In case each transition only has one input place, then the Cartesian product is relaxed to a linear comparison, however, in cases of two or more input places, the complexity of the problem increase quickly. It is often possible to trade computational complexity at the expense of memory usage, and one approach for doing so in the case of Petri nets, is to have all valid combinations for each conditional transition precomputed. In other words, when the content of an input place changes, this can be propagated to all affected transitions to construct valid combinations.

There are two types of transitions: those with a condition, and those without. The conditional transitions are only activated upon receiving external input. The nonconditional transitions are however activated when their input change. The nonconditional transitions can thus be used as a mechanism for precombining valid combinations for consecutive conditional transitions. Another approach would be to extend the conditional transitions to have distinct memory and preprocessing units, which keeps and maintains valid combinations. It can be aruged that the second approach is favorable as we avoid polluting the target concept with details for efficiency. Furthermore, it is not left for a template designer to think about issues coupled to efficiency.

For both methods, we still do need to compute the restricted Cartesian product however, instead of determining all valid combinations that each event can be combined with, this is instead done for each new token that is produced. It may be the case that neither of the two approaches are more efficient than the original specification, this mainly depends on the ratio of new events and new tokens. In this paper we investigate the second approach in which we explicitly use precomputing units and storage coupled to conditional transitions.

4 Experimental setup

4.1 Pick-pocket scenario

In order to investigate the suggested performance enhancements to the Petri net based approach for situation recognition, we have implemented an imaginary pick-pocket scenario in a simulator previously presented in [4]. The scenario plays out in a shopping zone (mall or similar), where numerous pedestrians move around on their daily business. Some pedestrians move straight through the environemnt, others move with a purpose and head for specific shops, whilst still others are more dynamic in their behavior. Two spawn zones are modelled, in which pedestrians are randomly created according to a uniform distribution. Also modelled, are four different shops. These attract pedestrians to move towards them by distributing fictional advertisment. The result is a rahter noisy enviroment with many objects that move around in various patterns.

On top of this model of normal behaviour, we have implemented an imaginary pick-pocket situation that proceeds as follows. First, two thieves enter the area. After a random amount of time, one of the thieves starts scouting for a suitable victim, which the thief tries to intercept. A successful interception is interpreted as the pocket of the victim having been picked clean. After this, the thieves approach each other to hand the stolen goods over, to finally leave the scenario. This situation is repeated at random points throughout each simulation. Figure 2 gives an illustration of the scenario. For more details, c.f. [7, 4, 5].

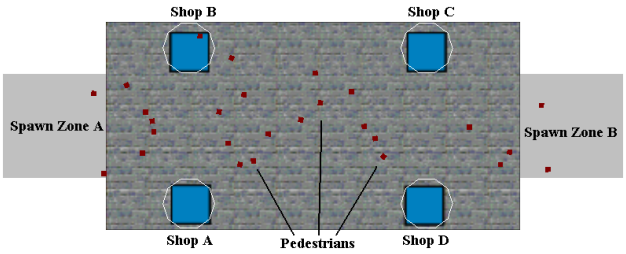

Figure 2: Illustration of the setup of the pick-pocket scenario.

Two scenarios have been used, which mainly differ in the number of pedestrians that are created. In the first scenario, there is a 10 % chance of creating one pedestrian and a 10 % chance of creating two pedestrians, in each of the spawn zones every five seconds. In the second scenario, there is a 35 % chance of creating one pedestrian, and a 10 % chance of creating two. Thirty simulations has been conducted using each of the two scenarios, resulting in a total of 60 data files containing tracks for all objects (perfect tracks at this point), sampled at 4 Hz. The scenarios are 15 minutes long, and each contains approximately 10 intentionally instantiated pick-pocket situations (the pattern may also emerge unintentionally).

4.2 Extraction of events

Three different track analyzers have been used to extract relational information from the data. The first analyzer produces and maintains $Close(x_1, x_2)$ relations in case objects x_1 and x_2 are close to each other. This is a symmetrical relation, i.e. $Close(x_2, x_1)$ is also true. The second analyzer produces and maintains $Approach(x_1, x_2)$ relations in case object x_1 is approaching object x_2, where approach is defined as x_1 heading towards x_2. The third analyzer produces and maintains $Intercept(x_1, x_2)$ relations when object x_1 is on an intercept path towards object x_2, where intercept is implemented through the use of the closest point of approach (CPA) metric. For each of the analyzers, in case a relation is found to hold, which previously did not, then an event is created and distributed to the recognition system. Similarly, in case a relation is found to not hold, which previously did, then an event is also produced and distributed to the recognition system. For more information about the extraction procedures, we refer the reader to [7, 4, 5].

5 Experimental results

In all results presented in this section, we have used the Petri net presented in section 2. In the first experiment we have run each of the 30 data files in each of the two scenarios, 30 times. In each run, we have measured the total time that has been spent in the Petri net. These results have been used to form an average of the total time, for each data file. This procedure has been carried out for both the original specification of the Petri nets, and when using explicit precombiners . The results are presented in figure 3.

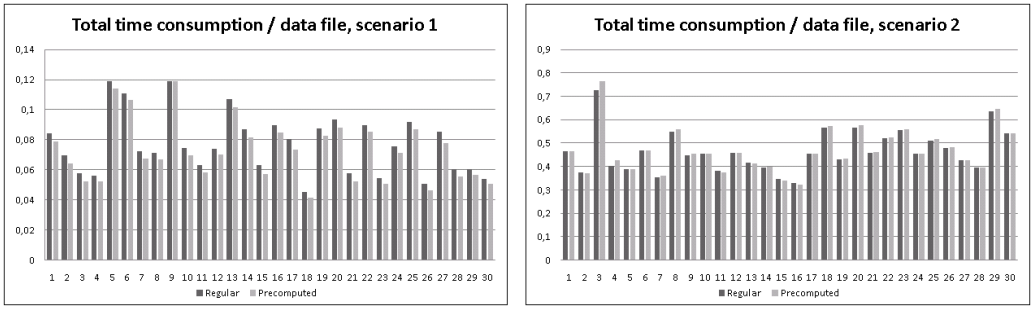

Figure 3: Average total time spent on recognition in the two scenarios.

As can be seen in figure 3, there seems to be some benefit of using precombination in the first scenario, however, these benefits disappear as the number of events increase (scenario 2). Precomputing possible combinations does therefore not seem to give any significant benefits. It should however be remembered that the Petri net we have used only combines at most two inputs. It may very well be benefits in case the number of inputs to combine increase. Benefits in time however often comes at a cost in memory. In figure 4 we show the memory usage, relative to the size of partial matches. The results are based on memory usage for each algorithmic setting, for both scenarios, for a single input file. As can be seen, memory consumption is more than doubled in the modified version using precombination. This makes it event more questionable if the potentially small benefits in computational complexity really is worth the expense in memory.

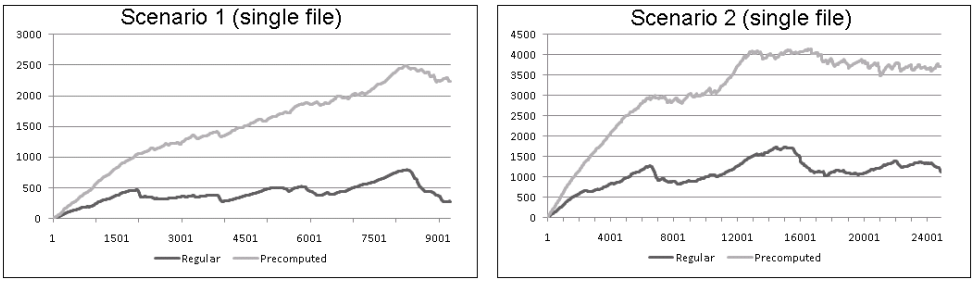

Figure 4: Memory consumption for a single input file, in each of the scenarios..

Finally, the main focus is to recognize situations. It is thus of interest to inspect the performance in terms of what we are able to recognize. This is normally measured using recall and precision, defined as follows:

$$Recall = TP/(TP + FN) \text{ and } Precision = TP/(TP + FP), \quad (1)$$

where TP, FP, and FN represents true positives, false positives, and false negatives. Table 1 shows precision and recall results with and without precombinations. Unsurprisingly, the modified version behaves exactly the same as the original version. This merely tells us that the logic of the recognition process has not been altered. The numbers are however on the lower end of the scale and calls for future research.

Table 1: Table illustrating precision and recall for each of the algorithmic settings and scenarios.

	Precision (S1)	Recall (S1)	Precision (S2)	Recall (S2)
Regular	0.358	0.916	0.108	0.823
Precombinations	0.358	0.916	0.108	0.823

6 Discussion

Having valid combinations precomputed does not seem to give any great performance boost. There is however another approach which possibly could be used to gain speed at the cost of memory. Wherever we put the combination logic, all partial matches in all input places are inspected for usage. A second approach could thus be to structure the content of places so as to allow for selective retrieval, i.e. instead of iterating over all partial matches in an input place, conditions for retrieval can be used to only yield interesting content. This can for example be done by ordering partial matches according to variable bindings (what has been unified), and then retrieving all partial matches that matches the new token constructed from event information. At first glance this may seem as a trivial problem to solve, since we simply could use some form of hashing function that depend on variable bindings. However, in a specific transition we do not know which variables that have been bound in preceeding places. This could be reasoned around analytically, but in the end, we would need to inspect all keys for all possible sub unifications since we allow for missed events. It is however possible to use some form of tree structure to more quickly find matches. In the worst case, this would still result in all tokens being inspected, however, in most cases we would likely restrict the set of interesting tokens. A tree structure does however not give any boost in cases where only single variables have been bound, without having nodes indexed according to variable bindings. Therefore, we suggest using a tree structure over unifiable variables, and then to index tree nodes of each consecutive level using a hashing function of the variable binding at the next consecutive level. It could however also be interesting to look closer at the work on timed automata for diagnosis of discrete event systems [22, 2], and related work, where computational complexity issues are addressed.

7 Conclusion

Situation recognition is an important problem to look into for introducing new capabilities in the surveillance domain. In this paper we have investigated a technique for enhancing the performance of a Petri net based approach to situation recognition. Although the computational load can be lowered to some degree, the benefit is not significant, and it will come at the expense of higher memory consumption. Instead, we need to focus on enhancing the performance in terms of what we recognize. At present, we are not very successful in separating intentionally instantiated patterns from unintentionally instantiated patterns. The reason behind this is likely that the underlying symbolic alphabet is not expressive enough, or does not capture the essence of the relations we are interested in. Hence, this should preferably be the focus of future work.

Acknowledgements

This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with Saab AB and the Swedish Knowledge Foundation under grant 2003/0104.

References

[1] E. Bossé, J. Roy, and S. Wark. *Concepts, Models, and Tools for Information Fusion*. Artech House, Norwood, MA, 2007.

[2] P. Bouyer, F. Chevalier, and D. DSouza. Fault diagnosis using timed automata. In V. Sassone, editor, *8th International Conference, FOSSACS 2005*, volume 3441 of *LNCS*. Springer, 2005.

[3] C. Castel, L. Chaudron, and C. Tessier. What is going on? a high level interpretation of sequences of images. In *Proceedings of Workshop on Conceptual Descriptions from Images, in the 4th European Conference on Computer Vision*, Cambridge, UK, 1996.

[4] A. Dahlbom, L. Niklasson, and G. Falkman. A component-based simulator for supporting research on situation recognition. In S. Mott, J. F. Buford, G. Jakobson, and M. J. Mendenhall, editors, *Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing*, volume 7352. SPIE, 2009.

[5] A. Dahlbom, L. Niklasson, and G. Falkman. Situation recognition and hypothesis management using petri nets. In V. Torra, Y. Narukawa, and M. Inuiguchi, editors, *Proceedings of Modeling Decisions for Artificial Intelligence (MDAI2009)*, volume 5861 of *LNAI*. Springer-Verlag, 2009.

[6] A. Dahlbom, L. Niklasson, and G. Falkman. Evolving petri nets for situation recognition. In *Proceedings of the International Conference on Genetic and Evolutionary Methods*, Las Vegas, USA, 2010.

[7] A. Dahlbom, L. Niklasson, G. Falkman, and A. Loutfi. Towards template-based situation recognition. In S Mott, J F Buford, G Jakobson, and M J Mendenhall, editors, *Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing*, volume 7352. SPIE, 2009.

[8] C. Dousson. Extending and unifying chronicle representation with event counters. In *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, 2002.

[9] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: Representation and algorithms. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence*, 1993.

[10] C. Dousson and P. Le Maigat. Chronicle recognition improvement using temporal focusing and hierarchization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

[11] M. R. Endsley. Theoretical underpinnings of situation awareness: a critical review. In M R Endsley and D J Garland, editors, *Situation Awareness Analysis and Measurement*, pages 3–32. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 2000.

[12] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, New York, 2002.

[13] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis. Representation and recognition of events in surveillance video using petri nets. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop*, volume 7, Washington, DC, 2004. IEEE Computer Society.

[14] D. A. Lambert. An exegesis of data fusion. In L Reznik and V Kreinovich, editors, *Soft computing in measurement and information aquisition*, pages 68–75. Springer, Berlin, 2003.

[15] G. Lavee, A. Borzin, E. Rivlin, and M. Rudzsky. Building petri nets from video event ontologies. In *Proceedings of the Third International Symposium on Advances in Visual Computing*, volume 4841 of *Lecture Notes in Computer Science*, pages 442–451, Berlin Heidelberg, 2007. Springer-Verlag.

[16] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisiting the jdl data fusion model ii. In *Proceedings of the 7th International Conference on Information Fusion*, June 2004.

[17] D Meyer-Delius, C Plagemann, G von Wichert, W Freiten, G Lawitzky, and W Gurgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *Proceedings of the 31st Annual Conference of the German Classification Society and Analysis, Machine Learning, and Applications*, 2007.

[18] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[19] M. Perše, M. Kristain, J. Perš, and S. Kovačič. Recognition of multi-agent activities with petri nets. In *Proceedings of the 17th International Electrotechnical and Computer Science Conference*, Portorož, Slovenia, 2008.

[20] J. F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, 2000.

[21] A. N. Steinberg. Foundations of situation and threat assessment. In M E Liggins, D L Hall, and J Llinas, editors, *Handbook of multisensor data fusion: theory and practice*, chapter 18, pages 437–501. CRC Press, 2 edition, 2009.

[22] P. Supavatanakul, J. Lunze, V. Puig, and J. Quevedo. Diagnosis of timed automata: Theory and application to the damadics actuator benchmark problem. *Control Engineering Practice*, 14(6):609 – 619, 2006.

[23] K. Walzer. *Temporal Complex Event Processing with Rule-based Systems using the Rete Algorithm*. PhD thesis, Technical University Dresden, 2009.