Willem L. van Norden
Defence Material Organisation,
CAMS – Force Vision,
P.O. Box 10000,
1780 CA Den Helder,
The Netherlands.
w.l.van.norden@forcevision.nl

Catholijn M. Jonker
Delft University of Technology,
Faculty of EE, Math & CS,
Man-Machine Interaction Group,
2628 CD Delft,
The Netherlands.
c.m.jonker@tudelft.nl

Utilizing classifier conflict for sensor management and user interaction

Published in:
Florentin Smarandache & Jean Dezert (Editors)
Advances and Applications of DSmT for Information Fusion
(Collected works), Vol. III
American Research Press (ARP), Rehoboth, 2009
ISBN-10: 1-59973-073-1
ISBN-13: 978-1-59973-073-8
Chapter XII, pp. 371 - 386

Abstract: *This chapter describes how the conflict encountered by the PCR6 rule can be utilized in sensor management. We therefore discuss the classification model that is used in the fusion problem and two different types of conflict. To enable operators to exert constraints on singletons we propose a (slightly) altered PCR6 rule, dubbed* $PCR6^a$. *We show how the algorithm works and we illustrate how the amount of conflict can be used for sensor management and/or for operator feedback by using an example.*

12.1 Introduction

In recent decennia, a need has occured to develop support functionalities for obtaining and maintaining situation awareness within the combat management system aboard the frigates of the Royal Netherlands Navy. This is due to three factors. Firstly, because the missions have become more complex in several ways. The mission goals are more diverse and the political climate in which these goals need to be met, are more complex compared to the Cold War period. The geographical location where the mission is executed, has shifted to the littoral, which means the meteorological conditions can change rapidly and there is much presence of civilian traffic. The latter makes missions more complex because the threat has shifted from military forces to asymetrical threats.

Secondly, much more complex and modern sensor systems, like multifunction radar and optical sensor with staring *360* degrees capabilities, are being placed aboard the Dutch frigates. This means that deploying these sensors and combining their information is a difficult and highly knowledge intensive task. Especially in the littoral environment, choosing the right sensor for the right task at the right time, given the meteorological conditions, is quite difficult.

Finally, budget cuts have led to reduced training and education time as well as a tendancy for a reduction in ships' complement. This means that the readily available knowledge aboard our frigates is decreasing. This discrepancy between required and available knowledge requires more support from the CMS for gathering and combining sensor information and for sensor management. Work has already been done in the field of automatic classification and how different classifier opinions can be combined, [3, 6]. This chapter describes how the results from the PCR6 rule of combination within the Dezert-Smarandache theory (DSmT) can be used as a feedback mechanism for automated sensor management.

Section 12.2 revisits the general rule of combination from DSmT, [8], and the PCR6 rule described by Martin and Oswald in [4]. Section 12.3 describes how classification and sensor management are related within Command and Control. Furthermore, it discusses the classification space within the military domain and shortly discusses the required interaction with the operator. Section 12.4 introduces how the conflict can be utilized within the PCR6 algorithm. The way this conflict can be used as a feedback to sensor management is discussed in section 12.5. Finally, section 12.6 closes with conclusions and future work.

12.2 Combination rules

Within the DSmT framework, the generalized basic belief that is assigned[1] by k different and independent sources or experts — $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$ — can be combined using equation (12.1). This equation holds $\forall X \in D^\Theta$ and $X \notin \emptyset$, where D^Θ denotes the hyper power set of Θ, the belief of each expert $\mathcal{E}_i$ with $i = [1, 2, \ldots, k]$ is denoted

[1] This is called a generalized belief assignment, or just a *gbba* for short.

$m_i(X)$ and $\emptyset$ denotes the classical empty set. Since this classic rule of combination only assumes exhaustiveness within the frame of discernment, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$, other rules of combination have been proposed to redistribute the conflict that might occur for applications in real fusion problems [9]. One of those rules is the PCR6 rule proposed by Martin and Oswald in [4] and is given in equation (12.2) for $\forall X \in D^\Theta$ and $X \notin \emptyset_\mathcal{M}$, where $\emptyset_\mathcal{M}$ denotes both the classical empty set and the set containing all elements from D^Θ that are constrained by fusion model $\mathcal{M}$. In equation (12.2), F_i is defined by equation (12.3). In equation (12.3) properties for the summation are given by equation (12.4) and equation (12.5).

In equation (12.3–12.5), $\varphi(i)$ denotes a function that ensures that i is skipped in a summation and is given by equation (12.6). In [4] this function is denoted σ_i. We use a different notation to prevent notational confusion for the classifiers that assume Gaussian distributions where σ_i denotes the standard deviation in variable i given some measurements. In [4], algorithm *3* gives the implementation of the PCR6 rule.

$$m_c^f(X) = \sum_{\substack{Y_1, Y_2, \ldots, Y_k \in D^\Theta \\ Y_1 \cap Y_2 \cap \ldots \cap Y_k = X}} \prod_{i=1}^{k} m_i(Y_i) \tag{12.1}$$

$$m_c^{PCR6}(X) = m_c^f(X) + \sum_{i=1}^{k} F_i \cdot m_i(X)^2 \tag{12.2}$$

$$F_i = \sum_{\substack{P_1 \\ P_2}} \frac{\prod_{l=1}^{k-1} m_{\varphi_i(l)}(Y_{\varphi_i(l)})}{m_i(X) + \sum_{l=1}^{k-1} m_{\varphi_i(l)}(Y_{\varphi_i(l)})} \tag{12.3}$$

$$P_1 : \quad \bigcup_{j=1}^{k-1} Y_{\varphi_i(j)} \cap X \in \emptyset_\mathcal{M} \tag{12.4}$$

$$P_2 : \quad (Y_{\varphi_i(1)}, Y_{\varphi_i(2)}, \ldots, Y_{\varphi_i(k-1)}) \in (D^\Theta)^{k-1} \tag{12.5}$$

$$\varphi_i(l) \rightarrow \begin{cases} \varphi_i(l) = l & \text{if } l < i \\ \varphi_i(l) = l + 1 & \text{if } l \geq i \end{cases} \tag{12.6}$$

12.3 Classification and sensor management

This chapter discusses how the conflict in combining classification solutions can be utilized in sensor management. Before modeling the classification model itself and how solutions are combined, this section will first briefly discuss how this may improve automated sensor management performance.

12.3.1 Sensor management

Optimally deploying a total sensor suite requires knowledge about:

- the meteorological and oceanographical conditions;
- the geographical location;
- the available sensor systems and their specifications; and
- the (expected) target characteristics.

The use of the target characteristics is e.g. discussed by Bar-Shalom in [1]. Furthermore, we know that prioritizing sensor functions can be done using risk, as proposed in [2]. This notion of risk requires characteristics of possible objects in the environment. Obtaining a good classification solution is therefore important to execute the process of sensor management.

On the other hand, the classification process itself has a certain need for information provided by the available sensor systems. In order to achieve good classification solutions, the sensor(s) need to be deployed as optimal as possible. This research therefore focuses on the information requirements of the classification process. In order to do this, we need to describe the classification model.

12.3.2 Modeling the classification space

In general, the possible solutions for classification are given by a so-called classification tree, [5, 7]. The drawback of using such trees is that the branching order is fixed. Describing the different classes as sets at different levels of specificity provides more flexibility in reducing the search space [6]. In the case of classification in the maritime military environment, we define three different levels of specificity. At the lowest specificity level we define the set of super classe, $\mathcal{S} = \{\vartheta_1, \vartheta_2, \ldots, \vartheta_s\}$ to contain s exhaustive elements. In this set the different domains are represented: air, surface, subsurface, land and sea respectively, therefore $s = 5$ holds.

At the medium specificity level we define generic classes, $\mathcal{G} = \{\gamma_1, \gamma_2, \ldots, \gamma_g\}$ with g mutually exclusive and exhaustive elements. At the final level we define the specific classes, $\mathcal{C} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_c\}$, with c mutually exclusive and exhaustive elements. Joined, these three sets define the frame of discernment for classification, $\Theta = \mathcal{S} \bowtie \mathcal{G} \bowtie \mathcal{C}$. We define the operator $\bowtie$ in a way that when $A = \{\alpha_1, \ldots, \alpha_a\}$ and $B = \{\beta_1, \ldots, \beta_b\}$ are joined then $A \bowtie B = \{\alpha_1, \ldots, \alpha_a, \beta_1, \ldots, \beta_b\}$.

Throughout this work we assume an example frame of discernment and three classifiers that assign generalized belief, given by tables 12.1–12.3. In these tables, $\mathcal{H}_h$ with $h = [1, 2, \ldots, (s + g + c)]$ is used to denote elements from the frame of discernment Θ. Also, when set $A = \{\alpha_1, \ldots, \alpha_a\}$ holds, we define $\widehat{A}$ as:

$$\widehat{A} = \bigcup_{i=1}^{a} \alpha_i.$$

$\mathcal{H}$	ϑ_1	ϑ_2	ϑ_3	ϑ_4	ϑ_5
Name	Air	Surface	Subsurface	Land	Sea
$m_1(\mathcal{H})$	*0.15*	*0.1*	*0*	*0.01*	*0.04*
$m_2(\mathcal{H})$	*0.13*	*0.12*	*0.005*	*0.02*	*0.025*
$m_3(\mathcal{H})$	*0.4*	*0.4*	*0.2*	*0*	*0*

Table 12.1: Super classes in the database with their gbba's.

$\mathcal{H}$	**Name**	$m_1(\mathcal{H})$	$m_2(\mathcal{H})$	$\mathcal{H}$	**Name**	$m_1(\mathcal{H})$	$m_2(\mathcal{H})$
γ_1	*Helo*	*0.25*	*0.175*	γ_4	*Frigate*	*0.041*	*0.1*
γ_2	*Fighter*	*0.002*	*0.002*	γ_5	*Tank*	*0.005*	*0.01*
γ_3	*Submarine*	*0*	*0.01*	γ_6	*Airliner*	*0.002*	*0.003*

Table 12.2: Generic classes in the database with their gbba's.

$\mathcal{H}$	**Name**	$m_1(\mathcal{H})$	$m_2(\mathcal{H})$	$\mathcal{H}$	**Name**	$m_1(\mathcal{H})$	$m_2(\mathcal{H})$
ς_1	*Seahawk*	*0.15*	*0.075*	ς_6	*Apache*	*0.15*	*0.075*
ς_2	*F-16*	*0.005*	*0.01*	ς_7	*M-frigate*	*0.04*	*0.0075*
ς_3	*Walrus*	*0*	*0.02*	ς_8	*Kilo sub*	*0*	*0.002*
ς_4	*7 Provinciën*	*0.036*	*0.075*	ς_9	*F-14 Tomcat*	*0.005*	*0.02*
ς_5	*Leopard II*	*0.009*	*0.01*	ς_{10}	*Boeing 747*	*0.005*	*0.02*

Table 12.3: Specific classes in the database with their gbba's.

12.3.3 Intersection between elements

The set-up of the classification model with three specificity levels immediately imposes that not all elements in the frame of discernment are mutually exclusive. This, of course, fits well within the DSmT framework. Each element at the most specific level has a *parent* at a higher level. E.g., the Seahawk and the Apache in table 12.3 are *children* of the generic class Helicopter. In turn, the helicopter belongs to the air domain, $\varsigma_1 \cup \varsigma_6 \subseteq \gamma_1 \subseteq \vartheta_1$, where $a \subseteq b$ is used to denote that a is a subproposition of b which holds if and only if $a \cap b = a$. Due to the helicopter's low-flight capabilities, it can also belong to the surface domain, $\gamma_1 \subseteq (\vartheta_1 \cap \vartheta_2)$. Similar reasoning can be done for all elements at the three specificity levels. From this example, we can already say that elements in $\mathcal{S}$ are not all mutually exclusive. For set $\mathcal{S}$ we know that $\vartheta_1 \cap \vartheta_2 \notin \emptyset_{\mathcal{M}}$, $\vartheta_1 \cap \vartheta_4 \notin \emptyset_{\mathcal{M}}$, $\vartheta_1 \cap \vartheta_5 \notin \emptyset_{\mathcal{M}}$ and that $\vartheta_3 \cap \vartheta_5 \notin \emptyset_{\mathcal{M}}$ holds. Furthermore, we know that $(\vartheta_4 \cup \vartheta_5) \cap \vartheta_2 = (\vartheta_4 \cup \vartheta_5)$ and $\widehat{\mathcal{C}} \subset \widehat{\mathcal{G}} \subset \widehat{\mathcal{S}}$ hold in the classification solution space.

For the intersections between elements of $\mathcal{S}$ and set $\mathcal{G}$ we can say that the following equalities hold: $\vartheta_1 \cap \widehat{\mathcal{G}} = \{\gamma_1, \gamma_2, \gamma_6\}$, $\vartheta_2 \cap \widehat{\mathcal{G}} = \{\gamma_1, \gamma_3, \gamma_4, \gamma_5\}$, $\vartheta_3 \cap \widehat{\mathcal{G}} = \{\gamma_3\}$, $\vartheta_4 \cap \widehat{\mathcal{G}} = \{\gamma_5\}$ and $\vartheta_5 \cap \widehat{\mathcal{G}} = \{\gamma_3, \gamma_4\}$. Furthermore, we can say that the following equalities also hold for the intersections of elements from $\mathcal{G}$ intersected with elements from $\mathcal{C}$: $\gamma_1 \cap \widehat{\mathcal{C}} = \{\varsigma_1, \varsigma_6\}$, $\gamma_2 \cap \widehat{\mathcal{C}} = \{\varsigma_2, \varsigma_9\}$, $\gamma_3 \cap \widehat{\mathcal{C}} = \{\varsigma_3, \varsigma_8\}$, $\gamma_4 \cap \widehat{\mathcal{C}} = \{\varsigma_4, \varsigma_7\}$, $\gamma_5 \cap \widehat{\mathcal{C}} = \{\varsigma_5\}$ and $\gamma_6 \cap \widehat{\mathcal{C}} = \{\varsigma_{10}\}$.

12.3.4 Interaction with the user

In [3] it was already stated how classifier belief can be combined using the PCR6 rule. Here, we expand the usage of PCR6 by having the user — or operator — as an additional information source. This user-influence can be exerted in two ways:

1. the operator is an information source and
2. the operator can place additional constraints.

Figure 12.1 depicts the resulting system architecture to achieve the required user interaction. The main difference between the user-imposed constraints (denoted $\emptyset_{\mathcal{U}}$) and the model constraints is that in $\emptyset_{\mathcal{U}}$ singletons can occur as opposed to combinations of elements from D^Θ that occur in $\emptyset_{\mathcal{M}}$: $\emptyset_{\mathcal{M}} \cap \widehat{\Theta} \in \emptyset$ whereas $\emptyset_{\mathcal{U}} \cap \widehat{\Theta} \notin \emptyset$. This means that the PCR6 rule needs to be adapted slightly to cope with this, section 12.4 describes how this is done. In [10] the Belief Conditioning Rules (BCR) were introduced to perform simular operations. Here however, we use the known structure of the frame of discernment to transfer belief. This has the advantage that we do not need to compute the subsets D_1, D_2 and D_3, where $\Theta \backslash \emptyset = D_1 \cup D_2 \cup D_3$, where $b \backslash a$ denotes all elements in b that are not in a. The approach mentioned in this chapter can therefore be seen as a specific BCR rule (somewhat similar to BCR17) where the construction of the subsets of Θ is not required since they are already given in the structure of the classification solution space.

Another difference is that the belief conditioning rules are used to indicate where belief should be held and that the methodology presented here indicates where belief should *not* be held. In other words: the operator indicates that what is absolutely not possible given the circumstances. Belief on what can be true is added into the fusion algorithm as just another source. This is done to keep options open as much as possible, following the operational credo: *expect the unexpected*!

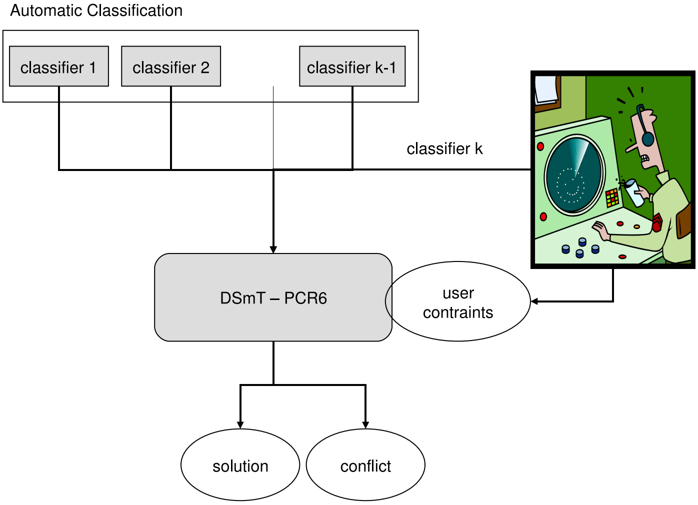

Figure 12.1: System Architecture where the user interacts with the system.

12.4 Conflict

Where belief from different sources is combined, chances are that conflict occurs. This conflict can be utilized in various ways. Firstly, we can look at which of the sources is responsible for most of this conflict. This could indicate that a particular source is malfunctioning. Also, in the case of automated classifiers it could indicate that an object is behaving unexpectedly, an important discovery when dealing with asymmetrical threats. By allowing the user to constrain elements from the frame of discernment, more conflict is introduced. This section describes how the conflict can be tracked within the PCR6 combination rule.

12.4.1 Tracking conflict in PCR6

In order to take the user-imposed constraints into account we say that equation (12.2) holds $\forall X \in D^{\Theta} \backslash (\emptyset_{\mathcal{U}} \cup \emptyset_{\mathcal{M}})$. Furthermore, the property P_1 of the summation in equation (12.4) is changed to equation (12.7). This adaptation ensures that all constraints are taken into account during combination. Now, suppose an operator indicates that the object under consideration does not belong to the subsurface domain $\emptyset_{\mathcal{U}} = \{\vartheta_3, \gamma_3, \varsigma_3, \varsigma_8\}$. Combining the three sources while taking the user-imposed constraints into account produces the combined gbba's in figure 12.2. However, there is a drawback to this approach. Not all conflict is redistributed due to the fact that singletons are being constrained, a situation that is usually not taken into account in applications of PCR6. This is illustrated by the fact that the assignments from figure 12.2 sum up to *0.756*.

$$P_1 \to \bigcup_{j=1}^{k-1} Y_{\varphi_i(j)} \cap X \in (\emptyset_M \cup \emptyset_U) \tag{12.7}$$

This does however give us a measure of the conflict, namely *0.244*, that is produced by the user constraints. Within the PCR6 algorithm we can track the total conflict from both the model constraints and the user-imposed constraints. Tracking the total conflict — that is conflict from both $\emptyset_{\mathcal{U}}$ and $\emptyset_{\mathcal{M}}$ — to the responsible sources for this conflict, $C_{\mathcal{E}_i}$, produces table 12.4.

Source, i	1	2	3	total
$C_{\mathcal{E}_i}(.)$	*0.0777*	*0.0564*	*0.2514*	*0.3855*

Table 12.4: Conflict produced by each source.

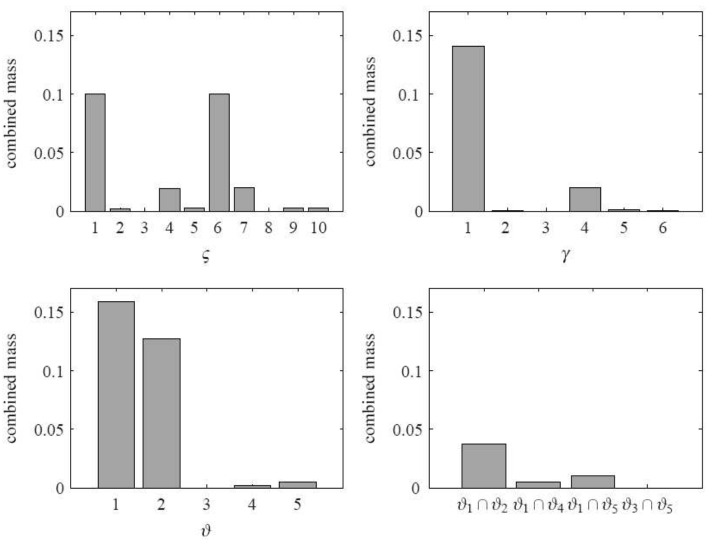

Figure 12.2: Combined generalised belief assignments.

In table 12.4 we see that source three is responsible for a great deal of conflict: an expected result looking at tables 12.1–12.3. However, the total output does not sum up to 1, which is an undesired result. We therefore use a modified PCR6 rule, denoted $m_c^{PCR6^a}$ that is given by equation (12.8), $\forall X \in D^\Theta \backslash (\emptyset_M \cup \emptyset_U)$. In equation (12.8), equations (12.9) and (12.10) hold. In order to keep track of the conflict on each individual element in $\emptyset_U$, we define equation (12.11) which holds $\forall X \in (\emptyset_U \cap D^\Theta)$. Within equation 12.11, equations (12.12) and (12.13) are defined. The adaptations on PCR6 proposed here, lead to the algorithm in appendix. In this algorithm the function call `Intersect` is used. This function is based on section 12.3.3.

$$m_c^{PCR6a}(X) = m_c^f(X) + \sum_{i=1}^{k} Q_i \cdot m_i(X)^2 \tag{12.8}$$

$$Q_i = \sum_{\substack{P_3 \\ P_2}} \frac{\prod_{l=1}^{k-1} m_{\sigma_i(l)}(Y_{\sigma_i(l)})}{m_i(X) + \sum_{l=1}^{k-1} m_{\sigma_i(l)}(Y_{\sigma_i(l)})} \tag{12.9}$$

$$P_3 \rightarrow \bigcup_{j=1}^{k-1} Y_{\varphi_i(j)} \cap X \in (\emptyset_{\mathcal{M}} \backslash \emptyset_{\mathcal{U}}) \tag{12.10}$$

$$C_{\mathcal{H}}(X) = m_c^f(X) + \sum_{i=1}^{k} T_i \cdot m_i(X)^2 \tag{12.11}$$

$$T_i = \sum_{\substack{P_4 \\ P_2}} \left(\frac{\prod_{l=1}^{k-1} m_{\varphi_i(l)}(Y_{\varphi_i(l)})}{m_i(X) + \sum_{l=1}^{k-1} m_{\varphi_i(l)}(Y_{\varphi_i(l)})} \right) \tag{12.12}$$

$$P_4 \rightarrow \bigcup_{j=1}^{k-1} Y_{\varphi_i(j)} \cap X \in \emptyset_{\mathcal{U}} \tag{12.13}$$

12.4.2 Redistribution of remaining conflict

For the redistribution of conflict introduced by the assumed model $\mathcal{M}$, we use the adapted $PCR6^a$ rule. Since,

$$\sum_{\forall X \in D^\Theta \backslash (\emptyset_{\mathcal{U}} \cup \emptyset_{\mathcal{M}})} m_c^{PCR6^a}(X) \neq 1$$

holds, we want to distribute the masses in $C_{\mathcal{H}}$ to the masses on $m_c^{PCR6^a}$ to obtain $m_c^{PCR6distr}$,

$$m_c^{PCR6distr} = \texttt{ReDistribute}(m_c^{PCR6^a}, C_{\mathcal{H}}, \emptyset_{\mathcal{U}}).$$

Due to the fact that

$$\sum_{\forall X \in D^\Theta \backslash \emptyset_{\mathcal{M}}} (m_c^{PCR6^a}(X) + C_{\mathcal{H}}(X)) = 1$$

holds, the quantity will sum up to 1 after this operation while maintaining the insights in the conflict produced by $\varnothing_{\mathcal{U}}$ and $\varnothing_{\mathcal{M}}$. The distribution of masses from $C_{\mathcal{H}}$ is done based on the same priniciples as the general PCR rules. That means that masses are distributed to related elements as much as possible. Therefore, when a element with high specificity is constrained, its gbba is distributed to its parent at the next level since that element was involved in calculating $m_c^f(X)$, equation (12.1). A problem occurs when elements at the lowest specificity level are constrained since they have no parent to distribute the mass to. This is solved by looking at the possible intersections of elements in set $\mathcal{S}$.

from	**to**
ϑ_1	ϑ_2, ϑ_4, ϑ_5
ϑ_2	ϑ_1, ϑ_3
ϑ_3	ϑ_2, ϑ_5
ϑ_4	ϑ_2
ϑ_5	ϑ_2, ϑ_3

Table 12.5: Distributing masses at the highest hierarchical level.

Table 12.5 shows how these transfers should be handled when using this approach. Only when these distributions are no longer possible, are masses distributed to the other non-constrained elements. We already mentioned that masses are distributed based on the principles of PCR, all transfers are therefore done proportionally. Let us look at the example given in tables 12.1–12.3 and place a user constraint on all elements of the air domain, note that this also means all underlying children in sets $\mathcal{G}$ and $\mathcal{C}$. When combining the sources using equation (12.8) and distributing $C_{\mathcal{H}}$ using the aforementioned method figure 12.3 is produced.

These results are not very intuitive and a change in transferral methodology is required. We expand the distribution scheme in order to transfer masses to other elements on the same specificity levels. To do this, a distribution tree is built based on $\varnothing_{\mathcal{U}}$ to transfer masses from elements in $\mathcal{C}$ to other elements in $\mathcal{C}$ according to its parents and table 12.5, this produces figure 12.4, which corresponds to a more intuitive result.

Since the elements to which the mass is transferred to is not directly involved in the original conflict, one could argue that within this redistribution scheme the transfers do not need to be proportional.

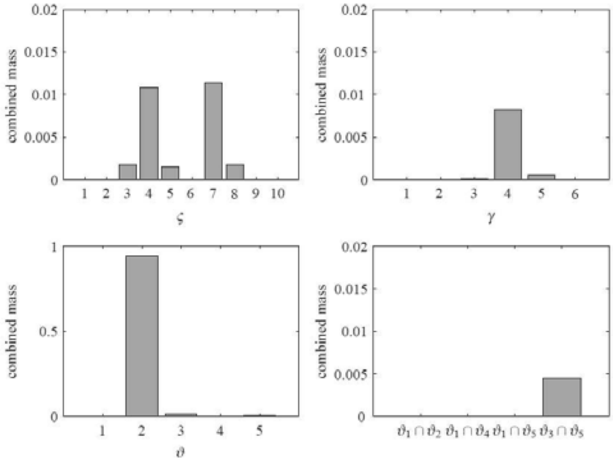

Figure 12.3: Results for $PCR6^a$ after a redistribution of conflict from $\emptyset_U$ when conflict goes to parent elements.

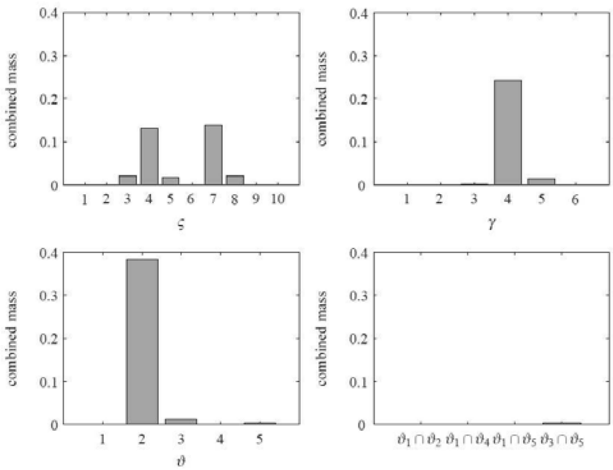

Figure 12.4: Results for $PCR6^a$ after a proportional redistribution of conflict from $\emptyset_U$.

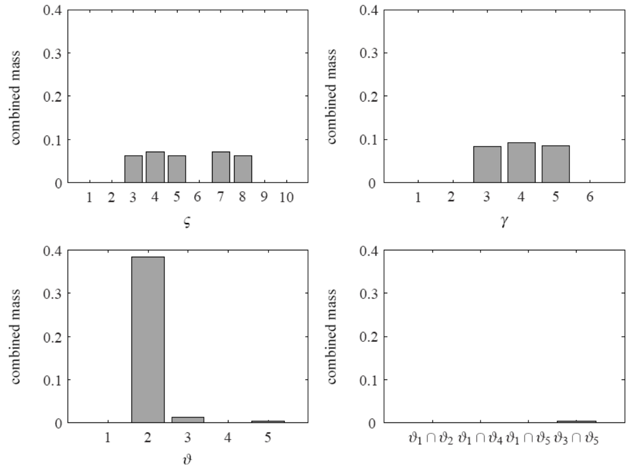

Figure 12.5: Results for $PCR6^a$ after unproportional redistribution of conflict from $\emptyset_U$.

Figure 12.5 shows the results when masses are not transferred proportionally. The difference with figure 12.4 is the relative difference between the masses assigned to elements is maintained better when transferring proportionally. In figure 12.6 we see the results from the different steps combined, first we see the results when equation (12.1) is used, then the results from $PCR6^a$ and finally the results of the $PCR6^a$ after proportionally redistributing masses from $C_{\mathcal{H}}$.

12.5 Utilizing the conflict in sensor management

In previous sections we have seen how belief on classification solutions from different sources can be combined and how user-imposed constraints on singletons can be taken into account with a slightly altered PCR6 algorithm. The question of course is: *why do we want to track the conflict?*

In essence the answer is simple, once we know where conflict is introduced we can try to reduce it. In this section we will first discuss tracking the conflict per source or expert and we will follow up with the conflict traced back to elements in $\varnothing_{\mathcal{U}}$.

12.5.1 Conflict per source

Where belief from different sources is combined, conflict occurs. Combined belief is obtained by proportionally redistributing these conflict using by the $PCR6^a$ rule. By tracking the conflicting masses that need to be redistributed, we can say which of the

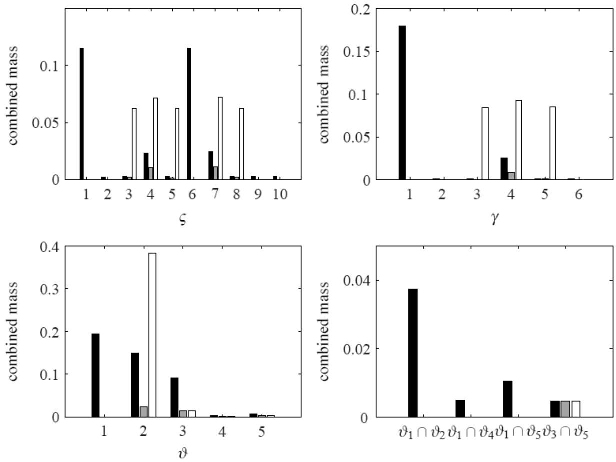

Figure 12.6: Results for unconstrained PCR6 (*black*), constrained $PCR6^a$ (*gray*) and after redistribution of conflict from $\emptyset_U$ (*white*).

sources is responsible for an amount of conflict. When one specific source produces the most conflict this could indicate that:

1. a sensor system that provides information to that source is degraded;

2. the classifier is malfunctioning or ill-trained;

3. the object under consideration is behaving unexpectedly.

Tracking the conflict does not answer the question as to which one of these three is the case, but it gives a trigger to take actions to find out. Especially the case when the classification solution is visually confirmed and all sensors are performing correctly is operational important. Section 12.1 already mentioned the amount of civil traffic in the current mission environments. When the conflict based on a subset of attributes indicates that one of those objects is behaving strangely, this is valuable in situations where asymetrical threats are expected.

Another option for a large conflicts between different sources occurs when a lot of uncertainty resides in the sensor measurements. By looking at the source that produces most conflicting information and combining that with the knowledge about the source, namely the attributes it uses to find solutions, we know which types of sensor measurements are required to reduce the conflict between sources, which in turn improves the combined classification solution.

12.5.2 Conflict per hypotheses

When the operator imposes constraints, $\emptyset_{\mathcal{U}} \notin \emptyset$, the conflict that each of these constraints introduces can be tracked. When combined with a machine learning algorithm, this conflict can be used to do some online training of automated classification algorithms to have them adapt to the current situation. On the other hand, it can be used to train personnel aboard during transit to the mission area. In order to find out whether the system was mistaken or if the user was mistaken, sensor functions can be requested to reduce the conflict on each of the elements in $\emptyset_{\mathcal{U}}$. If the newly obtained sensor measurements confirm the combined belief of the sources (the conflict increases) the operator can be alerted to further investigate this conflict and then remove the constraint for instance. When the operator is certain about the constraint, the conflict on the specific hypothesis can indicate a malfunctioning sensor or ill-trained classifiers although this is not very probable if the sources do not have much conflict amongst themselves. The most likely option then is an object that is behaving very unexpectedly.

12.6 Conclusions

This papes shows that it is possible to combine the information of different automated classifiers using the PCR6 rule of combination. By introducing an addition to the PCR6 rule we show that constraints on singletons can be taken into account. By tracking the conflict during the execution of the PCR6, the sources of the conflict can be identified. Furthermore, the quantity of the conflict can be utilised in automated sensor management and provides valuable feedback to the operator.

Future work is to implement more accurate sensor models and objects in order to validate this methodology in more realistic scenarios. After this validation it will be implemented in an actual combat management system in order to further test the system with real operators. In this stage a comparison is planned to validate the improved performance of our system compared to the systems currently in use.

12.7 References

[1] Y. Bar-Shalom, T. Kirubarajan, C. Gokberk, *Tracking with classification-aided multiframe data association*, IEEE Trans. on AES, Vol. 41, No.3, pp. 868–878, July 2005.

[2] F. Bolderheij, *Mission Driven Sensor Management: Analysis, design, implementation and simulation*, PhD. Thesis with Delft Univ. of Tech., Nov. 2007.

[3] A. Martin, C. Oswald, *Generalized proportional conflict redistribution rule applied to Sonar imagery and Radar target classification*, Chap. 11 in [11].

[4] A. Martin, C. Oswald, *The generalised PCR6 rule stable in terms of solution*, Chap. 2 in [11].

[5] B. Mertens, *Reasoning with uncertainty in the situational awareness of air targets*, Masters Thesis with the Man-Machine Interaction Group of the Delft University of Technology, Delft, 2004.

[6] W. van Norden, F. Bolderheij, C. Jonker, *Combining system and user belief on classification using the DSmT combination rule*, Proc. of the 11th Int. Conf. on Inf. Fusion (Fusion 2008), Cologne, Germany, June, 2008.

[7] K. Scholte, *Dynamic Bayesian networks for reasoning about noisy target data*, Den Helder: Royal Netherlands Naval College, Research Report for the Combat Systems Department, December 2005.

[8] F. Smarandache, J. Dezert, *An Introduction to the DSm Theory for the Combination of Paradoxical, Uncertain, and Imprecise Sources of Information*, Proc. 13th Int. Cong. of Cybernetics and Systems, Maribor, July 6-10, 2005 (see also the Chapter 1 of this volume).

[9] F. Smarandache and J. Dezert, *Proportional Conflict Redistribution Rules for Information Fusion*, Chap. 1 in [11].

[10] F. Smarandache, J. Dezert, *Belief Conditioning Rules*, Chap. 9 in [11].

[11] F. Smarandache, J. Dezert (Editors), *Advances and Applications of DSmT for Information Fusion, Collected works, Vol. 2*, Rehoboth, USA, 2006.

Appendix

The PCR6 algorithm with embedded conflict tracking and that enables constraints on singletons.

```
Data    : k experts ex: ex[i], ... , ex[n]
        : User-imposed constraints UC
Results : Fusion of ex by PCR6, ep
        : Conflict on each hypothesis, CH
        : Conflict per expert, CE
```

for *i = 1 to k* **do**
  foreach *c in ex[i]* **do** APPEND *c to cl[i];*
foreach *ind in [1,size(cl[1])] x ... x [size(cl[k])]* **do**
  [c,u] ← INTERSECT *(s, ind);*
  if $s \equiv \emptyset$ **then**

```
  lconf ← 1; sum ← 0;
  for i=1 to k do
    lconf ← lconf * ex[i](cl[i][ind[i]]);
    sum ← sum + ex[i](cl[i][ind[i]]);
  for i=1 to k do
    if u ∩UC ≡ ∅ then
      ep(ex[i][ind[i]]) ← ep(ex[i][ind[i]]) +
        ex[i](cl[i][ind[i]]) * lconf/sum;
      if u ≠ ∅ then
        CE(i) ← CE(i) + ex[i](cl[i][ind[i]]) * lconf/sum;
      endif
    else
      CH(u) ← CH(u) + (ex[i](cl[i][ind[i]] * lconf/sum)/size(u);
    endif
else
  lconf ← 1;
  for i = 1 to k do
   lconf ← lconf * ex[i](cl[i][ind[i]]);
    ep(s) ← ep(s) + lconf;
endif
```