

Correcting Missing Data Anomalies with Clausal Defeasible Logic

Peter Darcy, Bela Stantic, and Abdul Sattar

Institute for Integrated and Intelligent Information Systems
Griffith University
{P.Darcy, B.Stantic, A.Sattar}@griffith.edu.au

Abstract. Databases are used globally to store essential information required for various business applications such as automated data capturing. Unfortunately, due to missing record anomalies present within the repository, the overall integrity of stored information is compromised. Currently, filtration and rule-based techniques have been proposed to correct the problem, but due to a lack of high-level reasoning, ambiguous scenarios lead to anomalies persisting within the database. In this paper, we propose an enhanced *Non-Monotonic Reasoning* cleaning architecture that utilises intelligent analysis coupled with *Clausal Defeasible Logic* to rectify the missing data by generating and restoring imputed data. From our experimental evaluation, we have found that our proposed technique surpasses other leading intelligence classifiers such as Bayesian and Neural Networks.

1 Introduction

False negative anomalies are missing gaps of information that are meant to be present within databases. This problem is significant due to its potential to lower the accuracy and quality of the entire data set resulting in further data processes being hindered. Missing data anomalies are most persistent within automatic data capturing technology that utilises hardware to record information. Without high level intelligence designed to correct false negative anomalies, data sets that store information acquired from technologies such as automated recording devices will continue to process incorrect information hindering the applications it was designed for.

Radio Frequency Identification (RFID) is data capturing technology which automatically records data from tags that respond to readers. Of the problems found within passive RFID systems, ambiguous false negative observations remain the hardest to correct [1]. To correct the problem of missing observations, past approaches have utilised filtering algorithms and rule-based correction techniques to prevent the false negatives where possible. Unfortunately, within ambiguous situations in which the correction method is not easily determined, these methodologies fail to clean the data set adequately. The problems hindering these approaches include the fact that there is a lack of analytical information and intelligence when cleaning the data set with filters and rules respectively.

In this paper, we propose a methodology that cleans the stored data set using a high level intelligence reasoning engine. The system utilises analytical information generated from the missing observation coupled with Non-Monotonic Reasoning engines to correctly establish the likeliest set of data to insert back into the database. We specifically employ Clausal Defeasible Logic (CDL) as our Non-Monotonic Reasoning approach to arrive at a conclusion. We believe that by cleaning the data at a deferred stage allows the cleaning algorithm to have access to enough information needed to correctly impute the correct results. Additionally, by incorporating a high level intelligence algorithm such as Non-Monotonic Reasoning, we ensure the maximum likelihood to decipher highly ambiguous situations.

To evaluate the performance of our proposed system, we have conducted two experiments. The first was designed to assess which CDL formulae provided the highest cleaning rate, while the second evaluated the performance of our approach against Bayesian and Neural Network approaches. From our experiments, we have found the highest performing Clausal Defeasible Logic formula. With regards to our significance investigation, we have demonstrated that our proposed concept achieves the highest average cleaning rate when compared to Bayesian and Neural Networks under the same conditions.

The remainder of this paper is structured as follows: Section 2 will deliver background information relating to RFID and Non-Monotonic Reasoning crucial to understanding our proposed concept. A concise description of previous and related work will be provided in Section 3 shortly before we explore our own methodology in Section 4. An evaluation of our experimentation will be provided in Section 5 followed by our results and analysis in Section 6. Finally, Section 7 will discuss conclusions we have drawn from the our proposed methodology and future work we intend to investigate following the research we have conducted.

2 Background

RFID systems refer to a system that automatically identifies multiple amounts of tagged objects within a certain proximity to a reader. Although the potential benefits of cheap and durable passive tagging architectures are great, ambiguous anomalies such as missed readings hinder the world-wide adoption of this technology. High level intelligence engines, such as Non-Monotonic reasoning, may be harnessed to provide the resolution to highly ambiguous scenarios such as missing data to increase the integrity of the data set. Non-Monotonic reasoning approaches such as defeasible logic and clausal defeasible logic are employed to determine the optimal conclusion when given ambiguous information as input.

2.1 Radio Frequency Identification

Radio Frequency Identification (RFID) utilises radio transmissions between a reader and identifying tags to wirelessly locate objects automatically. Items are fitted with tags which are interrogated by readers resulting in the return of the

unique Electronic Product Code (EPC) [2]. Unfortunately, there are several issues associated with the RFID architecture, specifically the passive tag system. There are two persistent anomalies found within recorded data sets that are continually introduced. This may be attributed to various factors such as collision, detuning or water/metallic interference among tags [3]. These anomalies are false positive readings where the data captured did not exist in reality, and false negative readings where data is not present in the data set but is required to be. Of these two anomalies, false negative errors may be considered the hardest to correct as the data is not recorded into the database minimising the contextual information needed to correctly impute what readings may have originally been present. It has been estimated that only 60%-70% of recordings have been estimated to be captured within any RFID architecture [4].

2.2 Non-Monotonic Reasoning

Non-Monotonic Reasoning utilises logic to eliminate potential answers to a given problem based on available information until a single conclusion has been determined. When given the option to arrive at a conclusion, it will utilise present information to either prove or disprove an ambiguous situation with high level intelligence. An example in which Non-Monotonic Reasoning would decipher the correct solution would be when determining if Siberian Huskies bark. It is generally accepted that all dogs inherently bark, however, the Siberian Huskies are one of a few rare breeds of dogs which do not [5].

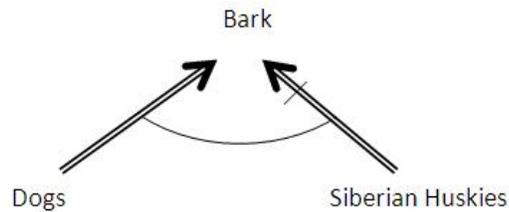


Fig. 1. The Logic Map that houses the clausal defeasible rules used when deciding if the Siberian Huskies will bark due to it being a dog or not.

Clausal Defeasible Logic (CDL) is software designed to incorporate the logic found in Non-Monotonic Reasoning within a computational environment and to be integrated into various applications [6]. The benefit of embedding Clausal Defeasible Logic is that the engine will deterministically arrive at various intelligent conclusions due to the use of different levels of ambiguous strengths. The scenario in which Siberian Huskies do not bark has been represented within the logical map utilised in CDL as pictured in Figure 1. As seen in the logic map, the specific rule that Siberian Huskies do not bark outweighs the general rule in

which dogs bark is represented with an arch that favours rules which are further anti-clockwise. In most cases, the logic arrives at the general conclusion, however this may be beaten by the specific rules that defy the previously established deduction. To properly find the correct conclusion, there are five core ambiguity strengths [7] which include the following:

- μ : The strongest of the formulae which will only prove the conclusion with factual information.
- α : A formula in which any conjunction of the π and β formulae are used to reach its conclusion.
- π : The formula in which ambiguity is propagated to reach its conclusion.
- β : The formula in where ambiguity will not be allowed to be used to draw its conclusions.
- δ : The disjunction of π and β are used to draw conclusions.

3 Related Work

Passive RFID anomalies has been studied extensively within related work with two main methods used to correct the missed readings. The first methodology uses filtering algorithms to correct incoming records when data is first captured. The filtration processes rely on anti-collision protocols to correct tagged observations as they are being physically read pre-data storage [8]. The second approach applies correcting algorithms to the stored data, which enhances the integrity of the observations within the data storage. This approach employs user-specified rules to modify the recorded observations in an attempt to enhance the integrity post-data storage [9].

While the proposed approaches can correct simple false-negative anomalies, highly ambiguous missing data may not be recovered due to the complex nature of the observations. We believe that due to flaws such as low level of analytical information and lack of high level intelligence, the filtration and rule-based methodologies will not be able to handle scenarios in which missing data is not easily replaced. With regards to the filtration approach, the cleaning process is performed at the edge which will only review the present and past information passed to the middleware. This results in the correction not taking future observations into account and has been compared with a Bayesian Network in previous literature [10]. In contrast, the rule-based approach utilises user-specified rules that are applied to the data set after the readings have been stored. However, logical anomalies may be introduced unknowingly in the event where false negative anomalies have a huge level of ambiguity. Previous research has found that by adding a higher level intelligence such as Plausible Logic improves upon the framework [11].

4 Methodology

In this paper, we propose the use of an advanced data analysis methodology coupled with high level intelligence to correctly decipher the likeliest candidates of observations to be returned into the data set. In the following section, we will outline the motivation and scenario considered in this work followed by a description of the system architecture of our approach. These discussions will be followed by the database structure that houses the RFID observations and all assumptions made towards our methodology.

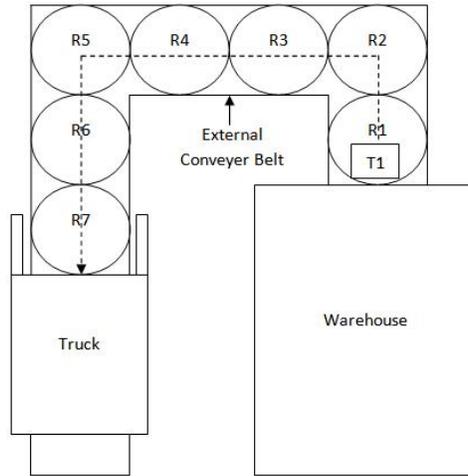


Fig. 2. A graphical representation of our ideal scenario in which a RFID-enabled conveyor belt tracks items from a warehouse to a truck.

4.1 Motivation and Scenario

False Negative anomalies are hazardous to all applications that utilise RFID as it prevents the recording of data lowering the overall integrity of the whole data set. Within previous work, we have established that there is large potential for the fusion of thorough analysis coupled with high level intelligence to adequately replace missed readings. Additionally, we have also put forth a preliminary analysis of a simplistic CDL logic engine using high level analysis which resulted in high cleaning rates. We have since decided to enhance the cleaning rules using different Non-Monotonic Reasoning software and comparing these against an already state-of-the-art approach.

The scenario upon which we have tailored our approach would include an enclosed static environment where tracking items is essential. Missed readings, however, may occur in this approach which need to be corrected before the

Table 1. A Table housing our ideal sample scenario RFID captured data.

EPC	Reader ID	Timestamp
T1	R1	2009-11-24 10:30:00.000
T1	R2	2009-11-24 10:31:00.000
T1	R3	2009-11-24 10:32:00.000
T1	R7	2009-11-24 10:26:00.000

data can be utilised in meaningful contexts. The most beneficial scenario would include false negative readings to occur randomly and rarely in which case it would be easier to correct. Our concept has been focused to provide higher intelligence for case studies in which missed readings occur consecutively.

For example, within a stocking warehouse that transfers stocked items via conveyer belt to various locations of the environment, such as the mock environment depicted in Figure 2. The circles within the diagram represent the readers and their respective ranges. The box with T1 inside represents the stocked item being tracked as Tag 1 and the dotted line represents the path of which T1 would travel in this scenario. If the items come too close within proximity or the tag is facing the opposite direction to the reader, the stock’s tags may not be observed at certain locations.

Within this scenario, there may be a situation in which the insertion of the observation may not be straight forward. Within our sample scenario previously mentioned, and the sample data generated within Table 1, we can see that *T1* is read at readers *R1*, *R2*, *R3* *R7*. From this information, conventional data correction algorithms would either replace all readings between *R3* and *R7* with the reader location *R3* or discard *R7* as a false positive anomaly. However, our approach would instead derive the shortest path from the available map data and insert it within the data set, thereby increasing the integrity of the overall information of *T1*.

4.2 System Architecture

As seen in Figure 3, we have divided our system’s architecture into three core components. The first is designed to analyse the data where the missed reading occurred which we have named the *Analysis Phase*. The data discovered in this Analysis Phase is then passed onto the *Intelligence Phase* where the correct permutation is selected. After the resulting data set has been chosen, the *Loading Phase* will complete the program’s cycle by inputting the information back into the data warehouse.

Analysis Phase The analysis phase consists of our tool locating missed readings and then discovering essential data about the anomaly. The first process is to divide the tags into “Tag Streams” as seen in Figure 4. These tag streams include chronological information relating only to one individual tag. From these tag

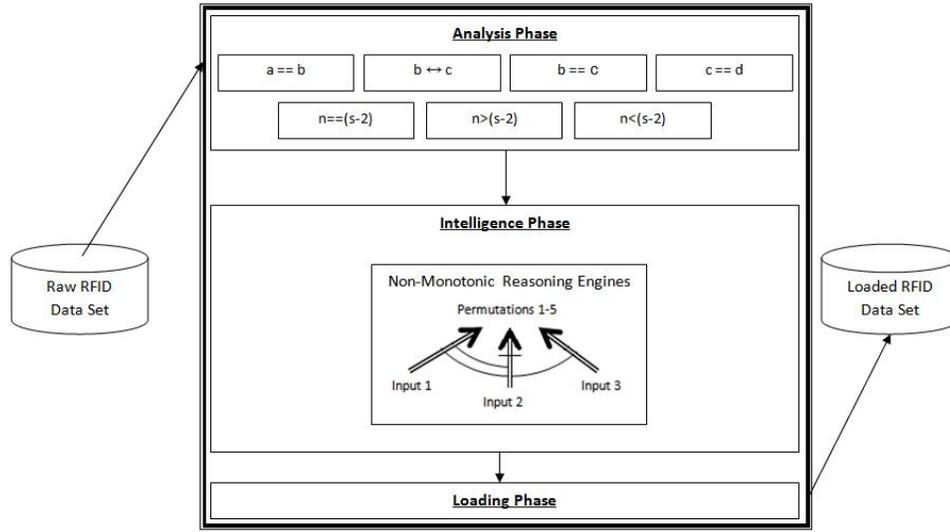


Fig. 3. A high level visualisation of inner processes used within our approach.

streams, certain information is ascertained relating to the nature of the false negative anomaly. This includes finding the reader locations of the observations two readings before and directly before the anomaly (a and b respectively), directly after and two readings after the reader (c and d respectively). Additionally, the shortest path between readings b and c using the map data is found. The total missing readings calculated via the number of missing timestamps (n), and the amount of observations within the shortest path (s), is then calculated. From this information, we ascertain the following analytical data:

- $a == b$
- $b \leftrightarrow c$
- $b == c$
- $d == c$
- $n == (s - 2)$
- $n > (s - 2)$
- $n < (s - 2)$

We utilise four main arithmetic operations to obtain these binary analytical information. These include the equivalent symbol $==$, the less $<$ and greater than $>$ symbols, and the \leftrightarrow symbol we have elected to represent geographical proximity. The reason as to why s is always having two taken away from it is that the shortest path always includes the boundary readers b and c which are not included within the n calculation.

Intelligence Phase The intelligence phase is when the various permutations of the missing data are generated as candidates to be restored in the data set.



Fig. 4. A visual representation of how we analyse one tag at a given moment within a Tag Stream.

The five different permutations that are been generated depicted in Figure 5 are described below:

- Permutation 1: All missing values are replaced with the reader location of observation *b*.
- Permutation 2: All missing values are replaced with the reader location of observation *c*.
- Permutation 3: The shortest path is slotted into the middle of the missing data gap. Any additional missing gaps on either end of the shortest path are substituted with values *b* for the left side, and *c* for the right.
- Permutation 4: The shortest path is slotted into the latter half of the missing data gap. Any additional missing gaps on the former end are substituted with value *b*.
- Permutation 5: As the anti-thesis of Permutation 4, the shortest path is slotted into the former half of the missing data gap. Additional missing gaps found at the latter end of the missing data gap are substituted with value *c*.

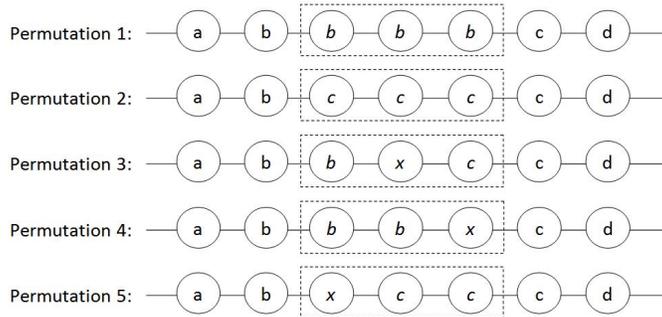


Fig. 5. An illustration of what reader values are placed into each false negative anomaly for each of the five different permutations. Please note that the shortest path is represented as *x*.

After the data analysis and permutation formations are completed, all relevant information is passed into the *Non-Monotonic Reasoning Engine*. The engine will then return an answer deterministically as to which permutation

Table 2. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 1 logic engine.

Rule No.	Rule	Conclusion
1	$b==c$	$\sim\text{perm1}$
2	$b==c \wedge n < (s-2)$	$\sim\text{perm1}$
3	$b==c \wedge n == (s-2)$	$\sim\text{perm1}$
4	$a==b$	perm1
5	$c==d \wedge n == (s-2)$	$\sim\text{perm1}$
6	$c==d \wedge n > (s-2)$	$\sim\text{perm1}$
7	$c==d \wedge b \leftrightarrow c \wedge n == (s-2)$	$\sim\text{perm1}$
8	$c==d \wedge b \leftrightarrow c \wedge n > (s-2)$	$\sim\text{perm1}$
9	$a==b \wedge b \leftrightarrow c$	perm1
10	$a==b \wedge b \leftrightarrow c \wedge n == (s-2)$	perm1
11	$a==b \wedge b \leftrightarrow c \wedge b==c \wedge \sim c==d \wedge n == (s-2)$	perm1
12	$c==d$	$\sim\text{perm1}$
13	$c==d \wedge b \leftrightarrow c$	$\sim\text{perm1}$
14	$a==b \wedge b \leftrightarrow c \wedge n > (s-2)$	perm1
15	$a==b \wedge b==c \wedge b \leftrightarrow c \wedge \sim c==d \wedge n > (s-2)$	perm1
16	$\sim b \leftrightarrow c$	$\sim\text{perm1}$
17	$n < (s-2)$	$\sim\text{perm1}$

best suits the missing gap of data, based on the rules we have created shown in Tables 2 - 6. Each of the rules present within the tables are combinations found from the analytical data joined by “and” statements (\wedge) which have been gathered within the Analysis Phase. Within the logic engine build, the precedence of the rules correspond to the larger number of the rule (for example, rule 17 will beat rule 4 in Table 3. In the event that more than one permutation has been found to be ideal in a given situation, we use the following hierarchical weighting: Permutation 3 > Permutation 1 > Permutation 2 > Permutation 4 > Permutation 5. In the unlikely case where no conclusions have been drawn from the *Non-Monotonic Reasoning Engine*, Permutation 3 will be elected as the default candidate due to it having perfect symmetry within the imputed data. This ordering has been configured to be the most accurate conclusion assuming that the amount of consecutive missed readings are low due to the randomness of the anomalies.

Loading Phase The loading phase consists of the selected permutation being uploaded back into the data storage at the completion of the *Intelligence Phase*. The user will have the opportunity to either elect to load the missing data into the current data repository, or copy the entire data set and only modify the copied data warehouse. This option would effectively allow the user to revisit the original data set in the event that the restored data is not completely accurate.

Table 3. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 2 logic engine.

Rule No.	Rule	Conclusion
1	$b==c$	$\sim\text{perm2}$
2	$b==c \wedge n < (s-2)$	$\sim\text{perm2}$
3	$b==c \wedge n == (s-2)$	$\sim\text{perm2}$
4	$c==d$	perm2
5	$a==b \wedge n == (s-2)$	$\sim\text{perm2}$
6	$a==b \wedge n > (s-2)$	$\sim\text{perm2}$
7	$a==b \wedge b \leftrightarrow c \wedge n == (s-2)$	$\sim\text{perm2}$
8	$a==b \wedge b \leftrightarrow c \wedge n > (s-2)$	$\sim\text{perm2}$
9	$b \leftrightarrow c \wedge c == d$	perm2
10	$b \leftrightarrow c \wedge c == d \wedge n == (s-2)$	perm2
11	$\sim a == b \wedge b \leftrightarrow c \wedge b == c \wedge c == d \wedge n == (s-2)$	perm2
12	$a == b$	$\sim\text{perm2}$
13	$a == b \wedge b \leftrightarrow c$	$\sim\text{perm2}$
14	$b \leftrightarrow c \wedge c == d \wedge n > (s-2)$	perm2
15	$\sim a == b \wedge b == c \wedge b \leftrightarrow c \wedge c == d \wedge n > (s-2)$	perm2
16	$\sim b \leftrightarrow c$	$\sim\text{perm2}$
17	$n < (s-2)$	$\sim\text{perm2}$

Table 4. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 3 logic engine.

Rule No.	Rule	Conclusion
1	$a == b \wedge c == d$	perm3
2	$\sim a == b \wedge \sim c == d$	$\sim\text{perm3}$
3	$a == b \wedge c == d \wedge n > (s-2)$	perm3
4	$\sim a == b \wedge \sim c == d \wedge \sim n > (s-2)$	$\sim\text{perm3}$
5	$n == (s-2)$	perm3
6	$b == c$	$\sim\text{perm3}$
7	$a == b \wedge c == d \wedge n == (s-2)$	perm3
8	$n < (s-2)$	$\sim\text{perm2}$

Table 5. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 4 logic engine.

Rule No.	Rule	Conclusion
1	$c == d$	$\sim\text{perm4}$
2	$b == c$	$\sim\text{perm4}$
3	$a == b$	perm4
4	$\sim a == b$	$\sim\text{perm4}$
5	$n > (s-2) \wedge a == b$	perm4
6	$\sim n > (s-2)$	$\sim\text{perm4}$
7	$\sim a == b \wedge \sim n > (s-2)$	$\sim\text{perm4}$

Table 6. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 5 logic engine.

Rule No.	Rule	Conclusion
1	$a==b$	$\sim\text{perm5}$
2	$b==c$	$\sim\text{perm5}$
3	$c==d$	perm5
4	$\sim c==d$	$\sim\text{perm5}$
5	$n>(s-2)\wedge c==d$	perm5
6	$\sim n>(s-2)$	$\sim\text{perm5}$
7	$\sim c==d\wedge\sim n>(s-2)$	$\sim\text{perm5}$

4.3 Database Structure

To store the information recorded from the RFID reader, we utilise portions of the ‘‘Data Model for RFID Applications’’ DMRA database structure found in Siemens Middleware software [12]. Additionally, we have introduced a new table called MapData designed to store the map data crucially needed within our application. Within the MapData table, two Reader IDs are stored in each row to dictate if the two readers are geographically within proximity.

4.4 Assumptions

We have made three assumptions that are required for the entire process to be completed. The first assumption is that the data recorded will be gathered periodically. The second assumption we presume within our scenario is that the amount of time elected for the periodic readings is less than the amount of physical time needed to move from one reader to another. This is important as we base our methodology around the central thought that the different readings will not skip over readers that are geographically connected according to the mapdata. The final assumption we make is that all readers and items required to be tracked will be enclosed in a static environment that has readers which cover the tracking area.

5 Experimental Evaluation

Within the following section, we have included a thorough description of the setup of the experimentation used in our methodology. First, we discuss the environment used to house the programs. This is followed by a detailed discussion of our experimentation including the two experiments we performed and their respective data sets used.

5.1 Environment

Our methodology has been coded in the C++ language and compiled with Microsoft Visual Studio C++. The code written to derive the lookup table needed

for the Non-Monotonic Reasoning data has been written in Haskell and compiled using Cygwin Bash Shell. All programs were written and executed on Dell machine with Windows XP Service Pack 3 operating system installed.

5.2 Experiments

We have conducted two experiments to adequately measure the performance of our methodology. The first experiment conducted was to determine which of the clausal defeasible logic formulae performs most successfully when attempting to correct large amounts of scenarios. The training cases used in the first experiment consisted of three sets of data consisting of 100, 500 and 1,000 ambiguous false negative anomaly cases. These three sets of data is used to represent the 60% - 70% anomalies of a small, medium and large database respectively.

The second experiment was designed to test the performance of our selected highest performing Non-Monotonic Reasoning logic against both a bayesian and neural Network approach. The reason as to why these techniques were selected as opposed to other related work is that only other state-of-the-art classifying techniques can be compared in respect to seeking the select solution from a highly ambiguous situation.

The second experiment testing sets included four data repositories consisting of 500, 1,000, 5,000 and 10,000 ambiguous false negative anomaly cases. We defined our scoring system as if the respective methodologies were able to return the correct permutation of data that had been previously defined. All data within the training and testing set have been simulated to emulate real RFID observational data.

6 Results and Analysis

To thoroughly test our application, we devised two different examinations which we have labelled the *Non-Monotonic Reasoning* and *Significance Experiments*. The Non-Monotonic reasoning experiment compared the cleaning rate of each of the *Clausal Defeasible Logic* formulae. The highest performing Non-Monotonic reasoning setup was then compared to Bayesian and Neural network approaches to demonstrate the significance of our cleaning algorithm.

6.1 Non-Monotonic Reasoning Experiment

We created the first experiment with the goal of determining which of the five CDL formulae would be able to clean the highest rate of highly ambiguous missing RFID observations. We did this by comparing the cleaning results of the μ , α , π , β and δ formulae on various training cases. There were three training sets in all with 100, 500 and 1,000 ambiguous false negative anomalies. Additionally, at the completion of these experiments, the average was determined for all three test cases and was used to ascertain which of the five formulae would be used within the Significance Experiment.

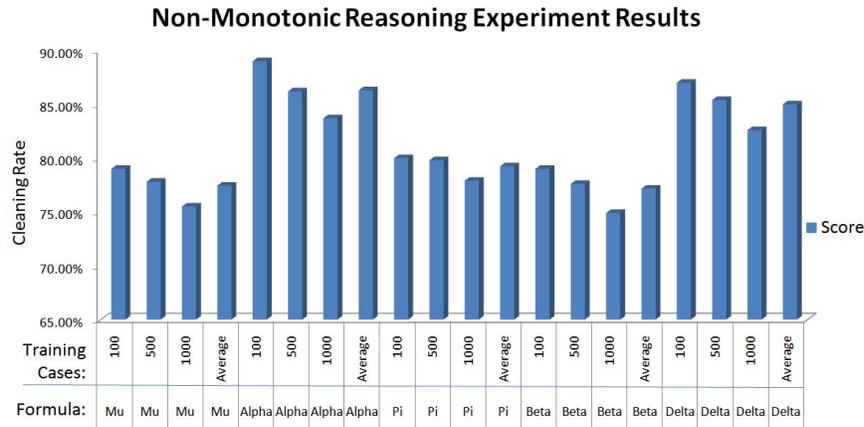


Fig. 6. The results of the Non-Monotonic Reasoning Experiment where the cleaning rate has been found for the five CDL formulae μ (Mu), α (Alpha), π (Pi), β (Beta) and δ (Delta).

As seen in Figure 6, the highest average achieving formula has been found to be α (Alpha). This is probably due to the fact that it discovers cases in which both the β and π formulae agree upon, increasing the intelligence of the decision. Also of note is that the disjunction of β and π formulae shown within δ achieves a relatively high average cleaning rate also. The lowest performing average cleaning rate has been found to be β , which is probably due to its non-acceptance of ambiguity when drawing its conclusion. We believe it is crucial for the cleaner to have a low level of ambiguity when drawing its conclusions as the problem of missed readings needs a level probability to infer what readings need to be replaced. As stated above, we have chosen the α formula as the highest performing cleaner to be used within the Significance Experiment.

6.2 Significance Experiment

The goal of our second experimental evaluation was designed to put three classifiers through a series of test cases with large amounts of ambiguous missing observations. The three different classifications techniques included our Non-Monotonic Reasoning engine with CDL using the α compared against both Bayesian and Neural Networks.

Both of the networks were trained using a Genetic Algorithm using 10 chromosomes with a single-point crossover function bred for 50 generations. We designed the experiment to have an abnormally high amount of ambiguous false negative anomalies consisting of 500 and 1,000 test cases to thoroughly evaluate each approach. After these experiments had concluded, we derived the average of each technique to find the highest performing classifier.

The results of this experiment, depicted in Figure 7, has shown that on average, our Non-Monotonic Reasoning approach achieved the highest cleaning rate.

It is important to note that it achieved the highest results when cleaning the data set with 1,000 test cases. We believe that the high results may be due to the deterministic nature of our methodology preventing erroneous permutations being selected. In contrast, the probabilistic methodologies have a higher chance of selecting incorrect imputed values which in turn produces artificial false positive anomalies and not recovering the original missing data.

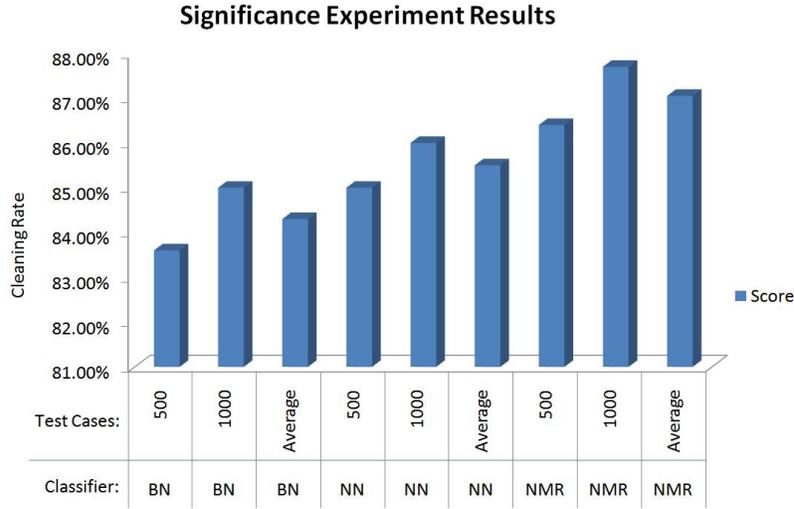


Fig. 7. The results of the Significance Experiment where the cleaning rate has been found for the Bayesian Network, Neural Network and Non-Monotonic Reasoning using the α formula.

7 Conclusion

In this paper, we have proposed the utilisation of intelligent Clausal Defeasible Logic engines to clean highly ambiguous false negative RFID anomaly cases. Through experimental studies we have identified that our concepts performs most effectively when used to restore consecutive missed observations. We have also conducted experiments to demonstrate the significance of our approach compared to other state-of-the-art technologies. More specifically this work makes the following contributions to the field:

- We put forth an enhanced logic engine and showed that it can deal with ambiguous false negative RFID observation anomalies.
- We found that the highest performing Clausal Defeasible Logic formula is α .
- We compared our proposed concept to other leading state-of-the-art classification techniques and found that our approach obtains a higher cleaning rate.

We have specifically tailored our approach to be compatible with RFID technology. However, this concept may be applied to other applications that have missing spatio-temporal observational data. With regards to future work, it would be interesting to investigate “Would the increase of complexity in the analysis phase result in a higher cleaning rate?” Specifically, increasing the amount of observational data collected in the analysis phase. We would also like to apply our methodology to a practical RFID supply chain and other applications that would benefit from our concept.

Acknowledgment

This research is partly sponsored by ARC (Australian Research Council) grant no. DP0557303.

References

1. Derakhshan, R., Orłowska, M.E., Li, X.: RFID Data Management: Challenges and Opportunities. In: RFID 2007. (2007) 175 – 182
2. Chawathe, S.S., Krishnamurthy, V., Ramachandran, S., Sarma, S.E.: Managing RFID Data. In: VLDB. (2004) 1189–1195
3. Floerkemeier, C., Lampe, M.: Issues with RFID usage in ubiquitous computing applications. In Ferscha, A., Mattern, F., eds.: Pervasive Computing: Second International Conference, PERVASIVE 2004. Number 3001, Linz/Vienna, Austria, Springer-Verlag (apr 2004) 188–193
4. Jeffery, S.R., Garofalakis, M.N., Franklin, M.J.: Adaptive Cleaning for RFID Data Streams. In: VLDB. (2006) 163–174
5. Lam, B., NICTA: SPINdle [online]. NICTA (2009) Available from: <<http://spin.nicta.org.au/spindleOnline/demo.html>> [Accessed: 25th October 2009].
6. Billington, D.: Propositional Clausal Defeasible Logic. In: European Conference on Logics in Artificial Intelligence (JELIA). (2008) 34–47
7. Billington, D.: An Introduction to Clausal Defeasible Logic [online]. David Billington’s Home Page (Aug 2007) Available from: <<http://www.cit.gu.edu.au/~db/research.pdf>> [Accessed: 3rd July 2008].
8. Shih, D.H., Sun, P.L., Yen, D.C., Huang, S.M.: Taxonomy and Survey of RFID Anti-Collision Protocols. Computer Communications **29**(11) (2006) 2150–2166
9. Rao, J., Doraiswamy, S., Thakkar, H., Colby, L.S.: A Deferred Cleansing Method for RFID Data Analytics. In: VLDB. (2006) 175–186
10. Darcy, P., Stantic, B., Sattar, A.: Improving the Quality of RFID Data by Utilising a Bayesian Network Cleaning Method. In: Proceedings of the IASTED International Conference Artificial Intelligence and Applications (AIA 2009). (2009) 94–99
11. Darcy, P., Stantic, B., Derakhshan, R.: Correcting Stored RFID Data with Non-Monotonic Reasoning. Principles and Applications in Information Systems and Technology (PAIST) **1**(1) (2007) 65–77
12. Liu, S., Wang, F., Liu, P.: A Temporal RFID Data Model for Querying Physical Objects. Technical Report TR-88, TimeCenter (2007)