Diagnostic Alarm Sequence Maturation in Timed Failure Propagation Graphs

Shane Strasser and John Sheppard
Department of Computer Science
347 EPS Building
Montana State University, Bozeman, Montana 59717
Email: {shane.strasser, john.sheppard}@cs.montana.edu

***Abstract*—Diagnostic model development presents a significant engineering challenge to ensure subsequent diagnostic processes using such models will yield accurate results. One approach to developing system-level diagnostic models that has been receiving attention is the Timed Failure Propagation Graph (TFPG), developed at Vanderbilt University. Unfortunately, developing TFPG models is also difficult and error-prone. In this paper, we extend previous work in using historical maintenance and diagnostic information to identify potential errors in the TFPG-based diagnostic models and recommend ways of maturing these models. This is done by extending the maturation process to incorporate historical alarm sequences and to model these sequences using a probabilistic transition matrix (similar to a Markov chain). The resulting sequence model is compared to the causal relationships identified in the original TFPG to discover discrepancies between the two. Potential sequence modeling errors with recommendations are given back to an engineer or analyst. We report on the maturation process and algorithms and also provide preliminary experimental results.**

I. Introduction

Developing diagnostic models that yield accurate results when used by subsequent diagnostic processes presents a difficult engineering process [1]. The Timed Failure Propagation Graph (TFPG) developed at Vanderbilt University is one form of system-level diagnostic model that has received attention [2], [3]. The TFPG model specifies causal relationships between faults and discrepancies, which are irregular conditions that are the effects of the faults (whether monitored or unmonitored). TFPG diagnostic algorithms are then able to diagnosis faults solely from knowing which alarms (monitored discrepancies) were or were not triggered. The diagnosis process, however, can be improved by incorporating the temporal information of when the alarms occurred [4], [5].

Fig. 1 shows an example TFPG model. Nodes labeled with F1, F2, F3, and F4 represent faults in the TFPG model. The labels D1 through D11 denote nodes that represent discrepancies. Monitored discrepancies are represented by nodes labeled M2, M3, M9, M10, and M11. If Fault F1 were to occur, then the the signal would propagate to D1 and then split and propagate to D3 and D3, which would be detected by the the alarms M2 and M3. The signal would the continue to propagate to the rest of the system and would continue to trigger alarms M9, M10, and M11. In the diagnostic setting, the algorithm would only see alarms M2, M3, M9, M10, and M11 fire. Because all of those alarms indicate fault F1, F1 would be reported as the most likely occurring fault.

The diagnostic performance of TFPG models for fault diagnostics is dependent on how accurate the TFPG model is. TFPG models are constructed by a domain expert of the system, which is difficult because the expert must accurately represent causal relationships given an often incomplete understanding of actual system behavior, therefore introducing error into the model. A bad TFPG model will result in poor diagnosis from the reasoner, causing an increase in time, money, parts, and labor in the maintenance of the modeled system since the corrective action will not be known [6]. For example, suppose the TFPG model in Fig. 1 is modeling some subsystem. Suppose that in designing the model, the engineer overlooked the relationship between the discrepancies D7 and D9 and did not add a causal relationship between the two discrepancies. In this faulty model, the reasoner will not be able to diagnosis fault F4 as the model does not have any relationship between F4 and any alarm. If fault F4 does occur in the physical system, alarms M9, M10, and M11 will all occur and the reasoner will likely diagnosis fault F2 because it is the simplest explanation as to what fault would trigger M9. The result is an increase in maintenance time since the true fault has to be located by alternative means [7].

Creating error-free TFPG models becomes even more difficult when hierarchical TFPG models are used. In [8], the authors proposed a hierarchical diagnosis approach for complex causal systems. In their approach, the system is partitioned into a set of local subsystems, each of which represents a sub-graph of the entire system. All of the local subsystems are contained within a global system that obtains a globally consistent diagnosis of the entire TFPG system. The difficulty in developing these hierarchical TFPG models is determining what relations should exist between different subsystems or how the dynamical system will behave in different environmental conditions [6]. For example, the hydraulic subsystem of an aircraft may have a relationship with the radar subsystem because of how the power subsystem connects the hydraulic and radar subsystems.

Because creating error-free TFPG models is difficult, a process is needed to mature the TFPG models over time. If the system user knows when the TFPG model is misdiagnosing a fault, he or she should be able to use that information to mature

978-1-4244-9363-0/11/$26.00 ©2011 IEEE

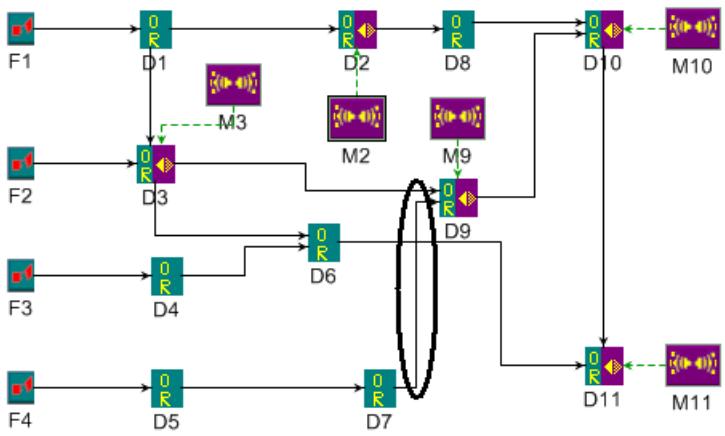

Fig. 1. A faulty TFPG model. Nodes labeled with F1, F2, F3, and F4 represent faults in the TFPG model. The labels D1 through D11 denote nodes that represent discrepancies. Monitored discrepancies are represented by nodes labeled M2, M3, M9, M10, and M11. The circled link from discrepancy D7 to discrepancy D9 should be included in the TFPG model but was overlooked when the model was created, resulting in a poor diagnosis of faults.

the TFPG model, resulting in a more accurate diagnosis. To determine whether the reasoner diagnosed the correct fault, one must compare the reasoner's diagnosis with the actual fault found by alternative means. By storing past maintenance history, detailed engineering analysis can be performed to determine the actual fault that occurred. The maintenance information can then be compared to the reasoner history and searched for any discrepancies between the two data sources. If there is a discrepancy between the two histories, then the user knows that the reasoner misdiagnosed a fault. For example, if the reasoner is misdiagnosing a particular fault numerous times, then there could an error in the TFPG model. These discrepancies between the reasoner and maintenance histories can then be used to modify the TFPG model such that the reasoner will diagnosis the correct fault [6].

TFPG maturation is a difficult problem [9]. To perform maturation, the reasoner diagnosis history and maintenance history are needed to be able to locate where the reasoner is misdiagnosing a fault. However, these data sources are often stored in heterogeneous systems, making retrieval and analysis of the data difficult [9]. In previous work, Wilmering and Sheppard suggested an approach to utilizing domain ontologies as a means to focus and filter data analysis in knowledge discovery [10]. The specific focus of that work was utilizing the ontologies to guide the process by which diagnostic models could be matured over time. That paper proposed using a method such as the *Apriori* algorithm to discover new relationships within historical maintenance data that could be used to determine diagnostic relationships, improved probability estimates, or better specification of test processes [10].

In our previous work, we extended the work of Wilmering and Sheppard to mapping diagnostic models and historical diagnostic session data to two ontologies derived from IEEE Standards [11]. In that work we mapped the reasoner information and D-matrix model to the IEEE Std 1232 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) and the maintenance history to IEEE Std 1632.2 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI) [12] [13]. The IEEE models, which were defined by using the EXPRESS language, were redefined using the Web Ontology Language (OWL) [14], [15]. This allowed us to use the graph-theoretic representations of the models and sessions to determine statistical discrepancies between what was expected by the models and what had been encountered in practice. From this, we were able to recommend changes to the TFPG model.

Our previous work focused on the ontology aspect of the maturation process and the maturation algorithm of the relationships between the faults and alarms in the TFPG models [11]. In this paper, we discuss the extension of our prior work by looking at how to mature the relationships between the different alarms in the TFPG model. To do so, we work directly with the TFPG model, allowing us to know the causal relationships between all the alarms in the TFPG model. Given a set of maintenance events, we model all of the alarm sequences using a probabilistic transition matrix similar to a Markov chain. This probabilistic transition matrix can then be compared to the TFPG model to find discrepancies between the two. These recommendations are then reported back to an engineer or analyst for further review. In this paper, we present our algorithm to perform the maturation of the relationships between the alarms in the TFPG model and test our algorithm on several scenarios.

II. Timed Failure Propagation Graphs

A Time Failure Propagation Graph (TFPG) model is a directed graph where each vertex represents a failure node or a discrepancy [2], [3]. Failure nodes represent faults in the target system and discrepancies are causal nodes that are affected by failure nodes. Discrepancies can be monitored or unmonitored. Monitored discrepancies are often referred to as alarms. The edges between the nodes represent the effect

of failure propagation over time in the underlying system that is being modeled. Formally, this is represented as TFPG $= (F, D, E, M, ET, EM, DC, DS)$ where:

- F is a set of failure nodes
- D is a set of discrepancy nodes
- $E = V \times V$ is a set of edges, where $V = F \bigcup D$
- $ET : E \rightarrow (Int, Int)$ is a mapping that associates each edge in E with a finite time interval
- $DC : D \rightarrow \{\text{AND, OR}\}$ is a mapping that defines each discrepancy as either an AND or an OR discrepancy
- $DS : D \rightarrow \{A, I\}$ is a mapping that defines whether a discrepancy is monitored or not monitored

The set of discrepancies that are monitored are defined by the map DS. The map ET associates with each edge e in E a minimum and maximum time for the failure to propagate along the edge. DC defines if each discrepancy is an AND or an OR node. The goal of a diagnostic algorithm is to find a hypothetical state that tries to explain the physical system based on observed alarms [4]. Fig. 1 is an example TFPG model with four faults and eleven discrepancies, five of which are monitored (alarms).

Since TFPG models have been introduced, several diagnostic algorithms have been developed to utilize these models by determining the most likely fault occurrence given a set of alarms that have been triggered [4], [5]. TFPG models have also been extended by Abdelwahed in 2004 to include model dependency constraints on the propagation links [16]. These extended models, referred to as a Hybrid Failure Propagation Graphs (HFPG), allow the model to operate in various operational modes. These different operational modes allow alarms to be either enabled or disabled.

One way a TFPG model can be represented is with the D-matrix, which is a matrix representation that relates the faults and the discrepancies that monitor or observe those faults. We can also formally represent it as the following: Let F represent a set of faults. Let D represent the set of discrepancies. Assume each $f_i \in F$ is a Boolean variable such that $eval(f_i) \in \{0, 1\}$ and each $d_j \in D$ is also a Boolean variable such that $eval(a_j) \in \{0, 1\}$. Then a diagnostic signature is defined to be the vector

$$f = \left[eval(d_1), \ldots, eval(d_{|T|}\right]$$

where

$$eval(d_j) = \begin{cases} 1 & \text{if } d_j \text{ detects } f_i \\ 0 & \text{otherwise} \end{cases}$$

and $fi[j]$ is the jth element in vector f_i. A D-matrix is then defined to be the set of diagnostic signatures d_i for all $d_i \in D$ [17]. Rows represent faults and columns represent discrepancies. The ith column corresponds to discrepancy i in the TFPG model. In most situations, the only discrepancies that are included or shown in the D-matrix are the monitored ones.

```
Unclose (Dmatrix D)
for all alarms a_i ∈ D do
  for all alarms a_j ≠ a_i ∈ D do
    if Obsv(a_i) ⊆ Obsv(a_j) then
      a_i → a_j
    end if
  end for
end for
```

Fig. 2. Algorithm to find the logical closure of the D-matrix.

D-matrices do not fully represent TFPG models because they do not capture the temporal relationships. However, one can find the logical relationship between the alarms by computing the logical closure of the matrix [18]. This is done by determining which attributes have a parent set that is a subset of another attribute's parent set. Let a_i be an alarm that monitors faults f_i and let a_j be an alarm that monitors faults f_j. We can represent this as $f_i \rightarrow a_i$ and $f_j \rightarrow a_j$. If f_i is a subset of f_j, then f_j contains f_i and $f_j \rightarrow a_i$. If a_j is true, f_j must also be true. This means a_i must be true and therefore $a_i \rightarrow a_j$ [1], [18]. The pseudocode to find the logical closure can be found in Fig. 2.

Using the original D-matrix one can find the logical closure matrix of the discrepancies. This matrix relates discrepancies to other discrepancies. Again, similar to the first D-matrix, a 1 in the ith row and jth column means that the jth discrepancy observes the ith discrepancy.

In addition, after taking the logical closure of the D-matrix, one can take the transitive reduction of a graph, which is also known as the logical unclosed matrix [18], [19]. In taking the transitive reduction of the graph, all of the transitive links between discrepancies can be removed by using logical relationships. After finding the logical closure of the matrix, the logical NOT is taken over the subset and AND is performed between the subset of parents for an alarm and the set of alarms. In doing so, the transitive edges in the TFPG model and the correspond D-matrices are removed. This induced D-matrix is then able to show the first order dependencies between the discrepancies.

In our previous work, we focused on maturing the D-matrices of the TFPG model because that is how the models are represented in the IEEE Std. 1232 AI-ESTATE. Therefore we were able to use the ontologies to relate the maintenance history and reasoner history to each other [11] [12]. One problem with using the induced D-matrix to represent the TFPG model, however, is that the resulting unclosed D-matrix will not always represent the true causal relationships between alarms. For example, in Fig. 3, after the logical closure and unclosure of the two models is found, the D-matrices will be the same.

Additionally, because the closure step depends on which faults alarms monitor, the logical closure D-matrix will not be able to determine the correct causal relationships in a sequence of alarms which all monitor the same set of faults. Therefore, for accurate maturation of the relationship between

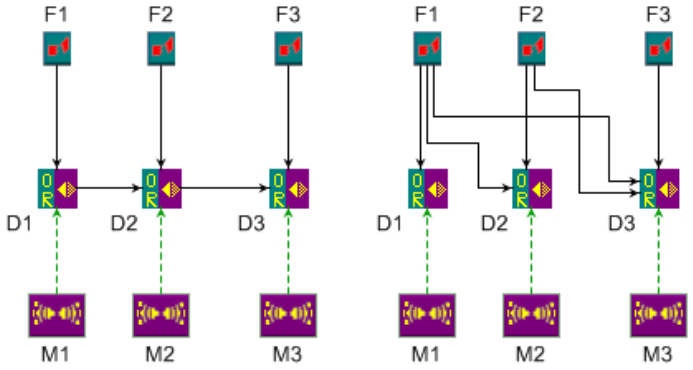

Fig. 3. Two different TFPG models. If the logical closure and unclosure of these two TFPG models is taken, the unclosed D-matrix of the two models will be the same.

the different alarms, the full TFPG model is needed. However, because the D-matrix is derived from the TFPG model, we are indirectly maturing the D-matrix.

III. Related Work

The idea behind diagnostic maturation has been discussed in several papers, but only one paper has published a formal algorithm for large amounts of corrective actions in which non-detects or false alarms could be occurring [11]. In [6], Wilmering points out there are unexpected and unplanned system interactions that can degrade the performance of the diagnostic design. In order to increase performance of the diagnostic model, historical maintenance actions will be used to help mature the model. Wilmering however, points out that the process requires ready access to the model, maintenance events, and any other information that could aid in the maturation process [6]. In order to utilize all of these resources, the author proposes using an ontology to gather all the required data together in a meaningful way.

In [1], the authors discuss using explanation-based learning for the diagnostic model. If misdiagnoses occurs, then additional testing is done until a correct diagnosis has been made. This information can then be used to modify the structure of the model so that the correct diagnosis is consistent with testing. The authors also give a proof that, given a single misdiagnoses, the model can be modified so that the misdiagnoses never occurs again. This was only valid for a single training example, however, and did not include how to deal with faulty or false alarms.

In our previous work, we presented an ontology-guided approach for D-matrix model maturation [11]. In that work we were limited to only maturing the relationship between alarms and faults. Once the discrepancies between the maintenance and reasoner history were found, the algorithm would find the probability of an alarm i firing given a single ground truth (maintenance event) fault j was diagnosed as the true fault. These probabilities were then compared against the D-matrix. If an alarm occurred with a high probability and the D-matrix showed that the alarm was not observing the fault or if the alarm occurred with a low probability and the D-matrix showed that the alarm is observing the fault, then the relationships were flagged as erroneous and a recommendation is made to the engineer.

The problem with this approach is that it does not give any information on how the erroneous relationship between the fault and alarm should be fixed, such as which link should be added or removed in the TFPG model. We extend that previous work by using the sequences of alarms to inform which links should be removed or added. Additionally, we present a more developed version of the sequence maturation along with several scenarios to test our algorithm.

IV. Maturation of Alarm Sequences

Suppose a TFPG model has a sequence of alarms that diagnoses a particular fault; however, maintenance events are finding that when the fault occurs, the sequence of alarms that should diagnosis the fault are not occurring. The sequence of alarms need to be adjusted so that the alarm sequences indicated by the model correctly identify the fault. In addition, there will also be alarms that do not fire when they should (non-detects) and alarms that fire when they should not (false alarms). These alarms need to be analyzed to gain an accurate picture of the alarms that should be occurring based on the maintenance events. Such analysis can also assist incorporating uncertainty measures into the diagnostic process.

Whenever a particular fault was found during post-maintenance analysis to be the true cause of a maintenance event, we analyze all of the reasoner logs that caused the unit to be pulled for maintenance. We then examine all of the reasoner logs for the corresponding maintenance event and pull all of the alarm sequences out of the reasoner logs. Using these alarm sequences, we calculate a post-occurrence probability matrix that gives the probability of an alarm occurring directly after another alarm. For alarms i and j, $[i, j]$ represents the probability that alarm j occurred sometime after alarm i with respect to the total number of alarm sequences that have occurred given that fault F was diagnosed as the ground truth in the maintenance event. Note that alarm j does not have to

```
BuildPostOccurProbMatrix (SetOfAlarms A)
for all Unique alarms a_i ∈ A do
  x = Number of times a_i occurs
  for all Unique alarms a_j ∈ A do
    if a_i ≠ a_j then
      y = Number of times a_j occurs after a_i
    end if
    M[i, j] = y/x
  end for
end for
return M
```

Fig. 4. Building the post-occurrence probability matrix M. The algorithm takes in a sequence of alarms A that where triggered when fault F occurred

```
FindDescrp (ProbMatrix M, TFPG T, SetOfAlarms A)
for all Unique alarms a_i ∈ A do
  for all Unique alarms a_j ≠ a_i ∈ A do
    if a_j directly observes a_i then
      if M[i, j] < ε_1 then
        F+ = Flag i and j
      end if
    else
      if M[i, j] > ε_2 then
        F+ = Flag i and j
      end if
    end if
  end for
end for
return AllFlags F
```

Fig. 5. Algorithm to find discrepancies between the TFPG model T and the post-occurrence probability matrix M. The algorithm takes in a post-probability matrix M, a TFPG model T, and the set of unique alarms A.

occur directly after alarm i. This matrix represents the temporal occurrences of the alarms that are being observed. The pseudocode to build the post-occurrence probability matrix is given in Fig. 4 (BuildPostOccurProbMatrix).

Next, we need to determine the expected post-occurrence probability matrix. As noted in our previous work, this is a combinatorial problem and therefore has an exponential run time [11]. Therefore, we use the TFPG model to find the causal relationships between the various alarms. If there is a causal relationship between two alarms i and j, we would expect a high probability in $[i, j]$ from the post-occurrence probability matrix. Similarly, if there is no causal relationship between two alarms, the corresponding probability in the post-occurrence probability matrix should be low. If there is a discrepancy occurring between the TFPG model and the post-occurrence probability matrix, we can then flag those two alarms to be observed in greater detail. The pseudocode for the alarm-sequence maturation algorithm is given in Fig. 5 (FindDescrp).

In the algorithm in Fig. 5, the user must set the threshold values ϵ_1 and ϵ_2 which are compared against the probabilities in the post-occurrence probability matrix. When the post-occurrence probability matrix indicates that there should not be a relationship between two alarms, but there is a causal relationship between the two alarms, we call it a Type I error. This is the first *if* statement in Algorithm 5 ($M[i, j] < \epsilon_1$). We call a Type II error when the post-occurrence probability matrix says that is a relationship but there is no causal relationship when there should be ($M[i, j] > \epsilon_2$).

After all of the discrepancies are flagged, they are then further analyzed to see if any of the discrepancies were falsely flagged. For example, suppose we have the TFPG model in Fig. 6. In this example the TFPG model is error-free. After we calculate the post-occurrence probability matrix, suppose we find that the probability of alarm M18 occurring after alarm M15 is very high. Similarly, the probability that alarm M18 precedes before alarm M17 is very high. Our algorithm will flag the relationships between M15 and M18 and M17 and M18 because there is no causal relationship between those alarms. In this scenario, however, that is acceptable because of how the signal is split after discrepancy D13 and propagates through the two links. To account for this scenario, we added a step that checks to see if the alarms have a related "ancestor," i.e. alarms that share a common causal discrepancy. If the alarms do have a common alarm or fault for which a causal relationship exists, then the alarms are not flagged as an erroneous relationship.

The process to find common ancestors is a straightforward process. Using an adjacency matrix to represent the TFPG model (using only faults and monitored faults), we find if there is a path from alarm (or fault) i to alarm j by using Warshall's algorithm [20]. We then take all of the Type II flagged relationships and check to see if there exists another alarm or fault for which there is a path to the two alarms that have been flagged as having a false causal relationship. If there is a common ancestor, we then remove that flagged relationship. All of the remaining Type II flags, along with all of the Type I flags are then recommended back to an engineer as changes to be made to the TFPG model.

V. Demo

To test our algorithm for TFPG alarm sequence maturation, we developed several different scenarios and ran our algorithm on these scenarios to demonstrate how the process works and to also show that our maturation algorithm is able to correctly find the erroneous links in the models. In all of our scenarios, the TFPG models were developed using the Generic Modeling Environment (GME) and were run using the Fault Adaptive Control Technology (FACT) reasoner [21], [22].

A. Missing Alarm

In the first scenario, we used the alarm sequence maturation algorithm developed in this paper to aid in the fault-alarm maturation process reported in our earlier work [11]. In this scenario, a particular alarm is not monitoring a fault that it should be. In our previous algorithm, we would be able to detect that there should be a relationship between the fault and the alarm but not exactly where to link the alarm to the

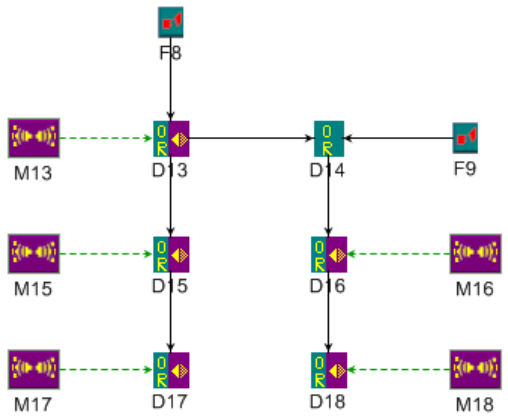

Fig. 6. A correct TFPG model. Note that alarm-sequence maturation algorithm may flag certain relationships as errors unless we add a step to check if for relationships that were falsely flagged.

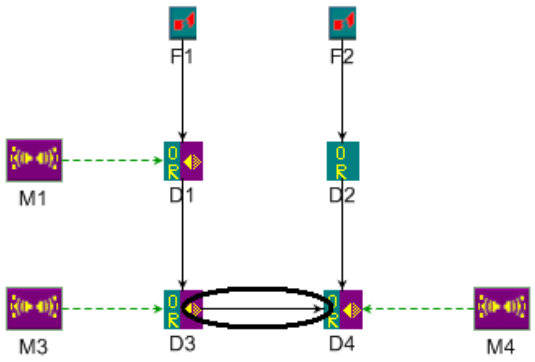

Fig. 7. A TFPG model with a missing link. The circled link should be in the TFPG model but was not included when the model was created. Therefore, the reasoner will not diagnosis the fault F1 with high confidence levels.

fault. Using the alarm sequence maturation algorithm, we are able to narrow the scope of possible links to add.

In our scenario, we created a TFPG model in Fig. 7. The link circled from D3 to D4 in the figure denotes the missing link in the TFPG model. Through several maintenance events, we observe that the alarm M4 occurs many times when fault F1 is diagnosed. Without knowing anything about the alarm sequences, we could guess several different ways to add the causal relationship between alarm M4 and fault F1, such as adding a link from F1 to D4 or D1 to D4. Suppose we also observe that alarm M4 almost always occurs after alarm M3. Using this information, we observe that the correct way to add the causal relationship is by adding a link between fault F1 and alarm M4. Using our algorithm we are able to correctly identify that a relationship should exist between D3 and D4.

B. Extra Alarm

Similar to the missing alarm scenario, the extra alarm scenario used the alarm sequence maturation algorithm to aid the fault-alarm maturation algorithm presented in our previous work [11]. In this scenario, the model indicates there is an alarm monitoring a fault that it should not be observing. By using the sequence maturation algorithm, we can identify the alarm and also how to remove the causal relationship between the fault and the alarm.

The TFPG model in Fig. 8 represents the incorrect TFPG model with the extra alarm M6 monitoring fault F3. After several maintenance events, we start to observe that alarm M6 never fires when fault F3 is diagnosed as the correct fault. Based on our fault-alarm maturation algorithm, we would need to remove the causal relationship between fault F3 and alarm M6. We can not, however, simply remove the link from fault F3 to alarm M6 because alarm M8 correctly observes fault F3. Based on the alarm sequences, we observe that alarm M8 almost always follows alarm M7. Our alarm-sequence maturation algorithm would tell us that a relationship should not exist between alarms M5 and M6, but that a relationship between alarms M7 and M8 should exist. Again, our alarm-sequence maturation algorithm is able to correctly diagnosis

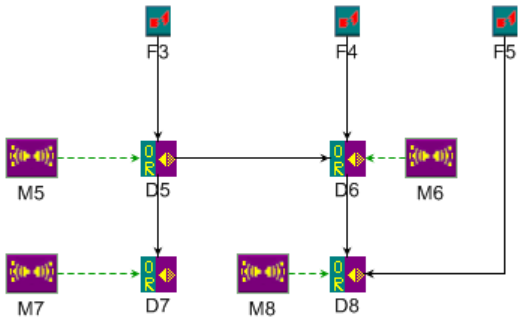

Fig. 8. A TFPG model in which there is extra alarm M6 observing fault F3. In the correct TFPG, there should be no link from D5 to D6. However, a link should exist between D7 and D8.

not only which links to delete in order to remove the fault, but also what links need to be added to keep all of the other alarms that may be affected by removing the link.

C. Reversed Order Of Alarms

The third scenario involves the order of alarms in a sequence. For example, suppose there is a TFPG model in which one alarm is causally dependent on another alarm, but the order of those two alarms should be reversed. In this scenario, the TFPG model in Fig. 9 was used to demonstrate that the alarm-sequence maturation algorithm finds the erroneous relationships. In all of the events that diagnose fault F6, we observe that alarm M11 almost always occurs before alarm M10, which is the opposite of what we would expect based on the TFPG model. We also observe that alarm M9 is almost always followed by alarm M11. Similarly, alarm M12 is almost always preceded by alarm M10. Based on these 3 erroneous relationships, we can observe that alarms M10 and M11 need to be flipped in the TFPG model. The final model would have a link from alarm M9 to alarm M11, a link from alarm M11 to M10, and a link from M10 to M12. Our alarm-sequence maturation algorithm is able to report these three erroneous relationships.

VI. Conclusion

Our alarm-sequence maturation algorithm is able to correctly identify erroneous relationships between sequences of alarms. Not only can this be used in the maturation of relationships between alarms, but also in the maturation of relationships between alarms and faults, as demonstrated in our first two scenarios.

For future work we want to also include maturing the time intervals used on the TFPG models. Since the diagnostic algorithms use the time intervals to diagnose a fault, a wrong time value could greatly change how the reasoner diagnoses a fault. Again, using an alarm sequence from a maintenance event, one could find those faulty time values and adjust them.

Finally, we would like to include maturation of probabilistic values in the TFPG models that utilize probabilistic values. If those probabilities are faulty, then the diagnostic reasoner could end up diagnosing the wrong faults. Again, if we have maintenance events that inform us which alarms were triggered and what fault was actually found during maintenance, we could find those faulty values in the model and recommend changes to them.

Acknowledgment

The authors would like to thank NASA for funding this project through a grant in the VLRS program in conjunction with the Boeing Research and Technology. Additionally, we would like to thank the Institute of Software Integrated Systems (ISIS) at Vanderbilt University for providing us with their Fault-Adaptive Control Technology (FACT) TFPG Reasoner and Generic Modeling Environment (GME).

References

[1] W. Simpson and J. Sheppard, *System test and diagnosis*. Kluwer Academic, 1994.

[2] A. Misra, "Sensor-based diagnosis of dynamical systems," Ph.D. dissertation, Nashville, TN, 1994, electrical Engineering.

[3] A. Misra, J. Sztipanovitz, and J. R. Carnes, "Robust diagnostic system: structural redundancy approach," in *Knowledge Based Artificial Intelligence Systems in Aerospace and Industry, SPIE's Symposium on Intelligent Systems*, vol. 2244, no. 1, apr. 1994, pp. 249–260.

[4] S. Abdelwahed, G. Karsai, and G. Biswas, "A consistency-based robust diagnosis approach for temporal causal systems," in *The 16th International Workshop on Principles of Diagnosis*, 2005, pp. 73–79.

[5] S. Abdelwahed, G. Karsai, N. Mahadevan, and S. C. Ofsthun, "Practical implementation of diagnosis systems using timed failure propagation graph models," *Instrumentation and Measurement, IEEE Transactions on*, vol. 58, no. 2, pp. 240–247, feb. 2009.

[6] T. Wilmering, "Semantic requirements on information integration for diagnostic maturation," in *Proc. IEEE AUTOTESTCON 2001*, Valley Forge, PA, aug. 2001, pp. 793–807.

[7] C. Byington, P. Kalgren, and B. Donovan, "Portable diagnostic reasoning for improved avionics maintenance and information capture continuity," in *Proc. IEEE AUTOTESTCON 2004*, sep. 2004, pp. 518–524.

[8] N. Mahadevan, S. Abdelwahed, A. Dubey, and G. Karsai, "Distributed diagnosis of complex systems using timed failure propagation graph models," in *Proc. IEEE AUTOTESTCON 2010*, Orlando, FL, sept. 2010, pp. 1–6.

[9] T. Wilmering, "When good diagnostics go bad - why maturation is still hard," in *Proc. IEEE Aerospace Conference 2003*, vol. 7, mar. 2003, pp. 3137–3147.

[10] T. J. Wilmering and J. W. Sheppard, "Ontologies for data mining and knowledge discovery to support diagnostic maturation," in *Proc. of the 18th International Workshop on Principles of Diagnosis*, Nashville, TN, may, 2007.

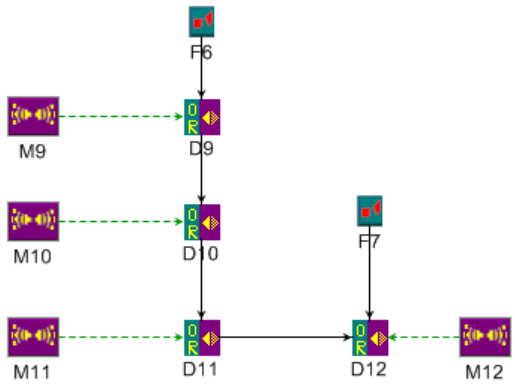

Fig. 9. A TFPG model which has alarms D10 and D11 that should be switched. In the correct model, D11 directly observers D9, D10 directly observes D11, and D12 directly observes D10.

[11] S. Strasser, J. Sheppard, M. Schuh, R. Angryk, and C.Izurieta, "Graph-based ontology-guided data mining for d-matrix model maturation," in *Proc. IEEE Aerospace Conference 2011*, mar. 2011.

[12] *Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*, IEEE Std. 1232-2002, 2002.

[13] *Draft Trial-Use Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Maintenance Action Information (MAI) via the Extensible Markup Language (XML)*, IEEE Std. P1636.2/D3, 2008.

[14] *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*, International Organization for Standardization Std. 10 303-11, 2004.

[15] *OWL 2 Web Ontology Language Document Overview*, http://www.w3.org/TR/owl2-overview/, W3C, 2009.

[16] S. Abdelwahed, G. Karsai, and G. Biswas, "System diagnosis using hybrid failure propagation graphs," in *The 15th International Workshop on Principles of Diagnosis*, Carcassonne, France, jun. 2004.

[17] J. W. Sheppard and S. G. Butcher, "A formal analysis of fault diagnosis with d-matrices," *J. Electron. Test.*, vol. 23, pp. 309–322, aug. 2007.

[18] S. Wahl, "Logical closure for diagnostic network simplification," may 2009, unpublished.

[19] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *Siam Journal on Computing*, vol. 1, pp. 131–137, 1972.

[20] S. B. Maurer and A. Ralston, *Discrete Algorithmic Mathematics*. A K Peters, 2004.

[21] *GME User's Manual*, http://www.isis.vanderbilt.edu/Projects/gme, Institute for Software Integrated Systems, 2005.

[22] G. Karsai, G. Biswas, S. Abdelwahed, N. Mahadevan, and E. J. Manders, "Model-based software tools for integrated vehicle health management," in *The Second IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, Pasadina, CA, 2006.