Generalized Diagnostics with the Non-Axiomatic Reasoning System (NARS)

Bill Power(✉), Xiang Li(✉), and Pei Wang(✉)

Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{tug00038,xiangliAGI,pei.wang}@temple.edu

Abstract. Symbolic reasoning systems have leveraged propositional logic frameworks to build diagnostics tools capable of describing complex artifacts, while also allowing for a controlled and efficacious search over failure modes. These diagnostic systems represent a complex and varied context in which to explore general intelligence. This paper explores the application of a different reasoning system to such frameworks, specifically, the Non-Axiomatic Reasoning System. It shows how statements can be built describing an artifact, and that NARS is capable of diagnosing abnormal states within examples of said artifact.

Keywords: Diagnostics · Model Based Diagnostics · Abductive inference

1 Introduction

The task of diagnosis highlights many of the core issues in general intelligence. It is a process that works from insufficient knowledge, building hypotheses and testing them to explain an observed state. These steps, each of which encompasses a wide body of work on their own, are combined into a process that feels uniquely human. As such, the field of Artificial Intelligence (AI) has made many attempts to understand these elements, and to compose their understanding into methods of automated diagnosis [1,3,13].

The general approach is to somehow encode a description of the artifact under diagnosis into a logic or model. Then, compare observations about the true artifacts behaviour, with observations of the predicted output of the model. The differences between the output are then the starting point to understand, or diagnose the physical artifact. This forms the basis of the Model Based Diagnostic Framework [12].

The goal of this paper is to show that the Non-Axiomatic Reasoning Logic, and associated reasoning system, can be used to build descriptions of systems, and similarly encode examples of the artifact for diagnosis. It will also show that the resulting descriptions can be fed into an "off the shelf" implementation of NARS, which yields expected results.

P. Hammer et al. (Eds.): AGI 2019, LNAI 11654, pp. 159–167, 2019.
https://doi.org/10.1007/978-3-030-27005-6_16

2 Prior Work

Prior to the work of Reiter, these attempts were largely separate, context dependant endeavours. Reiters work took these varied symbolic approaches, and create a generalized abstraction he referred to as Model Based Diagnostics. MBD could be considered the first formal 'consistency based' diagnostic method. The general approach is to encode a description of the artifact under diagnosis into a logical model. Then, compare observations about the true artifacts behaviour, with observations of the predicted output of the model. The differences between the output are then the starting point to understand, or diagnose the physical artifact [8]. Model Based Diagnosis describes a way to encode a model of an artifact in a predicate logic. This system description consists of predicate statements that connect components of the system. These components are encoded with their proper behaviour using the language of the predicate logic. Then, these component descriptions are conjuncted with the negation of a reserved predicate; "AB(x)", intuitively meaning "if the situation is abnormal".

This system description can then be combined with observations of the workings of a physical realization of the model. Again, such observations are also written in the language of the predicate logic used to describe the system. If these observations include statements that produce contradictions in the system, then it can be assumed that some subset of the components are behaving abnormally. In other words, there is a set of components, where if the predicate "AB(X)" is no longer negated, then the system description, along with the observations, becomes consistent [10].

Various methods have been developed to search for the resolution of these contradictory statements [2,5,9].

Extensions of the consistency based methods have been proposed. These incorporate more information into the system description. Instead of limiting the state of components to either normal or abnormal, abductive systems allow for a richer set of "failure states" that could be applied to a component [7]. These states relate failure to specific behaviours of the components. For instance, in the canonical example of the adder system, a gate component could be in a failure mode corresponding to "always outputs a value of true". The resolution of a diagnosis is then the minimal set of failure modes that must be applied in order for the system to be brought into a consistent state.

In addition to these new failure modes, abductive based systems also make new distinctions between the types and kinds of observations the system can have made on it. [11] introduces two main classes of observation. First, the set of observations that describe the context of the problem, yet do not require an 'explanation' via the diagnosis. Second, the set of observations that require an explanation. These two sets are referred to as CONTEXT and SYMPTOMS, respectively. This distinction is required to ensure that the resulting diagnosis is truly parsimonious, and does not include unnecessary explanations of contextual observations.

A unique approach to diagnosis comes from [8] extending the work of Reiter to handle multiple-fault diagnosis. In this, de Kleer develops an extension of truth maintenance systems that allows for tracking and tagging of assumptions.

Truth maintenance systems are a two part system. First, a problem solver, working in the language of some logic. The statements it derives and operates on are passed to the second component, a truth-maintenance system. This component views the logical statements on their own, and attempts to identify, correct, and isolate the inconsistencies that may develop in the running of the problem solver [4].

De Kleers system applies these notions to the arena of diagnosis. The problem solver sees the logical description of the artifact, its components, their relations, and then the observations of physical instances of the artifact. If the statements regarding an instance of the artifact bring the problem solver into an inconsistency, then the working of the truth maintenance system will isolate the minimal sets of components that need be labeled abnormal to bring the system back to 'working' order.

This combination of truth maintenance system with a system description most closely matches the approach presented in this work. While a truth maintenance system solely works by inducing consistency amongst observed statements, NARS would also include its additional methods of inference with prior knowledge to find conclusions.

3 NARS Overview

NARS (Non-Axiomatic Reasoning System) is a general purpose AI built in the framework of a reasoning system. It operates under a definition of intelligence that includes a notion of insufficient resources, specifically: "Intelligence" in NARS is defined as the ability for a system to adapt to its environment and to work with insufficient knowledge and resources. Detailed discussions can be found in many publications, including two main books [14,15], describing the full logic. The following will highlight the elements of the logic that are used by this paper to encode the system description of an artifact under diagnosis.

Narsese is the formal language which is used by NARS for its knowledge representation, defined using a formal grammar in [14]. The logic is developed from the traditional "term logic", where a "term" is the smallest element of each statements. Statements in this logic have the form *subject-copula-predicate*.

Inheritance statements are the most basic statement in Narsese, with the form "S $\rightarrow$ P", where S is the subject, and P is the predicate term. The "$\rightarrow$" is the *inheritance* copula, defined in ideal situations as a reflexive and transitive relation from one term to another term. The intuitive meaning of $S \rightarrow P$ is "S is a special case of P" and "P is a general case of S". For example, the statement "$CRV \rightarrow Honda$" intuitively means "CRV is a type of Honda vehicle".

A compound term (con, $C_1, C_2 ... C_n$) is formed by a term connector, "con" and one or more components terms $(C_1, C_2, ... C_n)$. The term connector is a logical constant with predefined meaning in the system. Major types of compound terms in Narsese include the following:

- **Sets:** Term $\{Honda, Toyota\}$ is an *extensional set* specified by enumerating its instances; term $[coupe, SUV]$ is an *intensional set* specified by enumerating its properties.
- **Products and images:** The relation "Mike is the patient of Dylan" is represented as "$(\{Mike\} \times \{Dylan\}) \rightarrow patient\text{-}of$", "$\{Mike\} \rightarrow (uncle\text{-}of\ /\ \diamond\ \{Dylan\})$", and "$\{Dylan\} \rightarrow (patient\text{-}of\ /\ \{Mike\}\ \diamond)$", equivalently.
- **Statement:** "Bill notices check engine light is on" can be represented as a *higher-order statement* "$\{Bill\} \rightarrow (notice\ /\ \diamond\ \{\texttt{check\_engine\_light} \rightarrow [on, off]\})$", where the statement "$\texttt{check\_engine\_light} \rightarrow$ [on]" is used as a term.
- **Compound statements:** Statements can be combined using term connectors for disjunction('$\vee$'), conjunction('$\wedge$'), and negation('$\neg$'), which are intuitively similar to those in propositional logic, but not defined using truth-tables [14].

4 System Descriptions in NAL

Artifacts can typically be described as a hierarchical set of components, along with some type of indication, or feedback about the state of an artifact. That is, we can break apart an artifact into components, which themselves can be composed of other components. The relations between these entities must account for relationships that would cause the 'parent' to behave abnormally, if any of their children were to behave abnormally.

In addition, it is assumed that there are classes of components that provide feedback to an observer. These are essentially the set of "senseable" components. They are the manner in which an example artifact can be described, because it is assumed that an example is simply a list of values, corresponding to a set of 'sensor readings'.

Encoding a system description then requires a representation for the components, a representation of the possible states of the senseable nodes, and a representation of the possible relations between the sensable values, and relevant components.

The relationship between parent components and their constituent parts can be expressed with the following inheritance statement.

$$Component_P \rightarrow (*, C_1, C_2, \ldots, C_N)$$

This roughly translates to "Component$_P$ inherits from the set of components: C_1, C_2,..., C_N."

The relationship between components and their sensors is slightly more complicated. We need to encode the possible states sensors can be in, then describe the relationship between these states and the working state of components. The sensor states are encoded as possible properties for each sensor.

$$\text{SensorX} \rightarrow (\|, [\text{on, off}])$$
$$\text{SensorY} \rightarrow (\|, [\text{state}_1, \text{state}_2, \ldots, \text{state}_n])$$

An implication is used to describe the relation between these sensor properties and the possible 'not working' property of components.

$$(\text{SensorX} \rightarrow [\text{on}]) \Longrightarrow (\text{ComponentY} \rightarrow [\text{Not Working}])$$
$$(\text{Check\_Engine\_Light} \rightarrow [\text{on}]) \Longrightarrow (\text{Engine} \rightarrow [\text{Not Working}])$$

This simplifies the method in which an instance of an object is presented to the system. We want to find the components that have a high frequency of having the property "Not Working", after listing the set of sensor properties for a particular object. For the example of a car, we would provide a conjunction of statements, where each statement is describing the properties of the car's components.

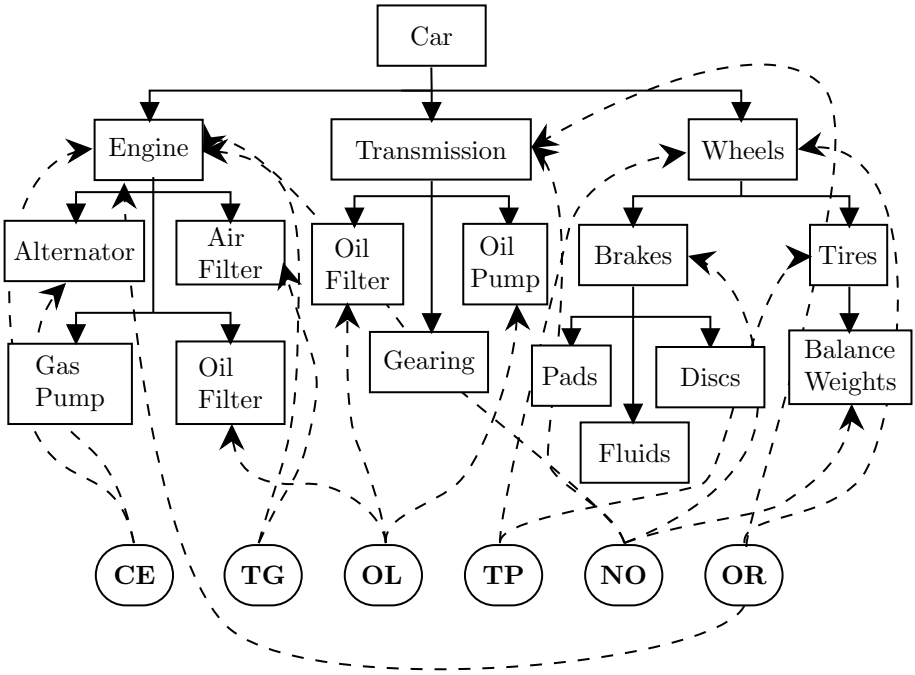
$$(\&, \text{SX} \rightarrow [\text{PA}], \text{SY} \rightarrow [\text{PB}], \ldots, \text{SN} \rightarrow [\text{PQ}]) \Longrightarrow (? \rightarrow [\text{Not Working}])$$

The above roughly translates to "Given a conjunction of properties, what Components have the property 'Not Working"'.

Two types of inference are typically involved in a complete diagnostic process; forward and backward inference. The goal of forward inference is to apply the current knowledge to the problem directly. An example of this is the simple chain of logic that starts with a cars 'check engine light'. The fact that the engine light is on, can be carried forward with the fact that 'engine light implies the engine is not working' to arrive at the fact that the engine is indeed not functioning.

In contrast, backward inference is used when the system does not have complete information, or a complete set of rules to use to arrive at a 'straight forward' inference. This typically occurs when different components of the system have an overlap in symptoms, or when the rules describing the system do not explicitly contain a mention of a particular combination of symptoms. In these situations, the system must 'work backwards' from what it knows to determine where it lacks appropriate information. In these situations, the system may need to prompt for additional rules.

Consider the car example. In our formulation, the engine itself is comprised of multiple sub components. The indicator for these components have some overlap; the check engine light could indicate a 'general' engine problem, or a 'specific' problem with the alternator. If the system assumes there are some cases, say the presence of smoke, that could be applied to only one of these sub components, then asking the clarifying question "is smoke observed?" is an example of backward inference.



Symbol	Interpretation	Values
CE	Check Engine Light	On, Off
TG	Temperature Gauge	Low, High
OL	Oil Level	Low, Middle, High
TP	Tire Pressure Indicator	On, Off
NO	Noise	None, High, Low
OR	Odometer Reading	Low, Middle, High

Fig. 1. Senseables

5 Automotive Example

The detailed car use case breaks down the object to be diagnosed, the car, according to the following diagram. In this diagram, two types of nodes exist, one denoting a component, and one denoting a facet of the system that can be sensed. These nodes represent items that would be 'sensed' by a user of the vehicle. This list was created from the point of view of a lay user describing issues to a mechanic, and try to reflect the high level indicators that exist on common dashboards.

The goal with this graph is to capture enough complexity for the system to have fodder for interesting and useful derived inferences. A key part of this complexity is ensuring that there are many-to-one relationships between components and the indicators. In addition, there is a deeper hierarchy of components, where the relationships with indicator nodes can span multiple levels.

The links between components (solid lines) indicate the relevant "required by" or "composed of" relations. These are implemented with the inheritance operator. A higher level component inherits its lower level constituent components.

There is a second type of edge/relationship in the graph; the relation between a component and a sensible thing (dashed lines). These represent the possible relations between an indicator and the components it might 'indicate' a problem for. Figure 1 shows the set of senseables exposed to the system, and the possible values they can take.

The actual NAL statements that encode the relations above are as follows.

```
//Engine, Transmission and Wheels are main components of the car
<{Engine, Transmission, Wheels} --> Main_Component>.

//Alternator, Air_Filter and so on are sub-components of the car
<{Alternator, Air_Filter} --> Sub_Component>.
<{Gas_Pump, Oil_Filter, Oil_Pump} --> Sub_Component>.
<{Gearing, Brakes, Tires} --> Sub_Component>.
<{Pads, Discs, Fluids, Balances_Weights} --> Sub_component>.

//Alternator, Air_Filter, Gas_Pump, Oil_Filter are parts of Engine
<{Alternator, Air_Filter, Gas_Pump, Oil_Filter} --> Engine>.
//Oil_Filter, Oil_Pump, Gearing are parts of Transmission
<{Oil_Filter, Oil_Pump, Gearing} --> Transmission>.
//Brakes and tires are components of wheel
<{Brakes, Tires} --> Wheels>.
//Pads, Discs and fluids are components of brakes
<{Pads, Discs, Fluids} --> Brakes>.
//Balance weights is component of tire
<{Balances_Weights} --> Tires>.

//Check engine light has no stage, ON or Off
<{CE} --> (||, [ON, OFF])>.
<{TG} --> (||, [LOW, HIGH])>.
<{OL} --> (||, [LOW, MIDDLE, HIGH])>.
<{TP} --> (||, [ON,OFF])>.
<{NO} --> (||, [NONE, HIGH, LOW])>.
<{OR} --> (||, [LOW, MIDDLE, HIGH])>.

// If check engine light is on, then alternator might not working
<<{CE} --> [ON]> ==> <Alternator --> [Not_Working]>>.
<<{CE} --> [ON]> ==> <Engine --> [Not_Working]>>.
<<{TG} --> [High]> ==> <Air_Filter --> [Not_Working]>>.
<<{TG} --> [High]> ==> <Engine --> [Not_Working]>>.
```

```
<<{OL} --> [High]> ==> <Oil_Filter --> [Not_Working]>>.
<<{OL} --> [low]> ==> <Oil_Pump --> [Not_Working]>>.
<<{TP} --> [ON]> ==> <Tires --> [Not_Working]>>.
<<{TP} --> [ON]> ==> <Wheels --> [Not_Working]>>.
<<{NO} --> [High]> ==> <Engine --> [Not_Working]>>.
<<{NO} --> [High]> ==> <Wheels --> [Not_Working]>>.
<<{NO} --> [High]> ==> <Brakes --> [Not_Working]>>.
<<{NO} --> [High]> ==> <Balances_Weights --> [Not_Working]>>.
<<{OR} --> [High]> ==> <Transmission --> [Not_Working]>>.
<<{OR} --> [High]> ==> <Engine --> [Not_Working]>>.
```

6 Results

The encoded system description was provided to an off-the-shelf version of the OpenNARS implementation of the NARS [6]. The system was run with its stock control mechanisms and parameters, and allowed to settle on answers to simple diagnostic queries. The resulting output is as follows.

```
// Car1's check engine light is on
<{Car1} ==> <CE --> [ON]>>.g
<{Car1} ==> <TG --> [High]>>.
<{Car1} ==> <NO --> [High]>>.
<{Car1} ==> <OR --> [High]>>.

// Car1 has a diagnosis on main components
<{Car1} --> [Diagnosis_On_Main_Component]>.
<{Car1} --> [Diagnosis_On_Sub_Component]>.
<{Diagnosis_On_Main_Component} --> Main_Component>.
<{Diagnosis_On_Sub_Component} --> Sub_component>.
//Which main component is not working?
<Diagnosis_On_Main_Component <-> ?x>?
<Diagnosis_On_Sub_Component <-> ?y>?

// Transmission is not working
Answer: <Diagnosis_On_Main_Component <-> Transmission>. %1.00;0.42%
// Alternator is not working
Answer: <Alternator <-> Diagnosis_On_Sub_Component>. %1.00;0.42%
// Engine is not working
Answer: <Engine <-> Diagnosis_On_Main_Component>. %1.00;0.59%
Answer: <Alternator <-> Diagnosis_On_Sub_Component>. %1.00;0.56%
```

7 Conclusion

This work shows that a separate logical framework can indeed solve the same problems covered by Model Based Diagnosis. However, this simplified representation does not leverage the full set of language levels, and mental operations

available in the NARS. Future work will include a closer inspection of the different types of operations that could be beneficial to automated diagnosis, while also developing a more robust control mechanism for generalized diagnostics. In addition, it would be useful to test the framework on more complex artifacts, such as models of human biological systems, or complex circuit designs.

References

1. Brusoni, V., Console, L., Terenziani, P., Dupré, D.T.: A spectrum of definitions for temporal model-based diagnosis. Artif. Intell. **102**(1), 39–79 (1998)
2. Ceriani, L., Zanella, M.: Model-based diagnosis and generation of hypothesis space via AI planning. In: 25th International Workshop on the Principles of Diagnosis (DX-14) (2014)
3. Console, L., Torasso, P.: A spectrum of logical definitions of model-based diagnosis1. Comput. Intell. **7**(3), 133–141 (1991)
4. De Kleer, J.: An assumption-based TMS. Artif. Intell. **28**(2), 127–162 (1986)
5. Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artif. Intell. **152**(2), 213–234 (2004)
6. Hammer, P., Lofthouse, T., Wang, P.: The OpenNARS implementation of the non-axiomatic reasoning system. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) AGI -2016. LNCS (LNAI), vol. 9782, pp. 160–170. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41649-6_16
7. ohan de Kleer, J., Williams, B.C.: Diagnosis with behavioral modes. Proc. IJ CAI **1**(989), 1324–1330 (1989)
8. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artif. Intell. **32**(1), 97–130 (1987)
9. Koitz, R., Wotawa, F.: SAT-based abductive diagnosis. In: International Workshop on Principles of Diagnosis, vol. 26, p. 10 (2015)
10. McCarthy, J.: Applications of circumscription to formalizing common-sense knowledge. Artif. Intell. **28**(1), 89–116 (1986)
11. Peng, Y., Reggia, J.A.: Abductive Inference Models for Diagnostic Problem-Solving. Springer, New York (2012). https://doi.org/10.1007/978-1-4419-8682-5
12. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
13. ten Teije, A., van Harmelen, F.: An extended spectrum of logical definitions for diagnostic systems. In: Proceedings of DX-94 Fifth International Workshop on Principles of Diagnosis, pp. 334–342 (1994)
14. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006). https://doi.org/10.1007/1-4020-5045-3
15. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)