PRUNING PROCEDURE FOR INCOMPLETE-OBSERVATION DIAGNOSTIC MODEL SIMPLIFICATION

Ivan Havel
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
Email: havel@labe.felk.cvut.cz

Abstract—**Model-based diagnostics deals with diagnosing systems, it means determining health of system components based on a system description and an observation of system variables. This paper focuses on systems that can be described in propositional logic, particularly on simplification of their diagnostic models for some given conditions of observation. It is known that performing diagnostics on the entire model of a system when only few variables are expected to be observed is not efficient. If we knew the limited set of variables which may appear in the observation then they would be used to simplify the diagnostic model before the diagnosis inference takes place. A pruning procedure which systematically removes segments of a model that do not contribute to the overall system diagnosis is proposed. The procedure employs an algorithm deciding component diagnosability which is based on directional resolution. Diagnoses retrieved using the simplified model are equivalent to those retrieved using the original model.**

Keywords— **diagnosability, incomplete-observation, model-based diagnostics, propositional logic, pruning.**

I. Introduction

The framework for this work is *model-based diagnostics*, whose goal is to determine health of system components based on a system description and an observation of system variables. The focus of this work is on *structured systems* with components that can be described in the framework of *propositional logic*. A typical representative of such system is a digital circuit. This work *does not* apply to dynamical (e.g. discrete event system) nor continuous systems.

The aim of this paper is to *simplify the diagnostic model* of a system for some given conditions of observation. It is obvious that performing diagnostics on the entire model of a system when only few variable observations are expected is not efficient. Consider a single component in the model – when we observe only some of its ports, then it might be the case that all obtainable diagnoses claim the behavior of the component is normal. In such a case, there is no point in computing diagnoses for this component. These observation conditions should be detected and used to simplify the diagnostic model of the system before the diagnosis inference takes place.

A procedure for model simplification (or model pruning) which systematically removes segments of a model that do not contribute to the overall system diagnosis is proposed. It uses an algorithm for deciding *component diagnosability* based on directional resolution, which was described in [Havel, 2006], to find a new simplified model of a system while preserving the ability to obtain all possible diagnoses of the system for a given set of observed system variables. That is to say, the diagnosis retrieved using the simplified model is equivalent to the diagnosis retrieved using the original model of the system under investigation.

This paper is structured as follows. An overview of model-based diagnostics and a description of algorithm for deciding component diagnosability is given in section II. Model pruning procedure is proposed in section III. Finally, an experiment is conducted in section IV and some concluding remarks are given in section V.

II. Component Diagnosability in Model-based Diagnosis

In [Reiter, 1987], Reiter laid the foundations for model-based diagnostics. His *diagnosis from first principles* acts on a logic based description of a system and an observation of its behavior. It attempts to find a set of components in the system which, when assumed to be faulty, explains the abnormal behavior of the system. Reiter suggests a procedure (without any implementation) that computes *minimal diagnoses*, i.e. diagnoses containing minimal subsets of faulty components. Together with assumption-based truth maintenance system (ATMS) by De Kleer and Williams [de Kleer and Williams, 1987] it belongs to *consistency-based approaches* which compute minimal diagnoses from conflicts.

A. Fundamentals and Notations

The formal definitions described below are variations of those from [Darwiche, 1998], that were adapted for diagnosis of components in [Havel, 2006].

Definition 1: Let $\mathbf{S}$ be a set of atomic propositions (atoms).

- An $\mathbf{S}$-*literal* is a literal whose atom is in $\mathbf{S}$.
- An $\mathbf{S}$-*instantiation* is a conjunction of $\mathbf{S}$-literals, one for each atom in $\mathbf{S}$.

For example, for $\mathbf{S} = \{\mathbf{P}, \mathbf{Q}\}$ there are four instantiations: $P \wedge Q$, $P \wedge \neg Q$, $\neg P \wedge Q$, and $\neg P \wedge \neg Q$.

Definition 2 (Component description) Description of a component X is a triple $(\mathbf{P}, \mathbf{A}, \mathbf{\Delta_X})$, where $\mathbf{P}$ and $\mathbf{A}$ are sets of atomic propositions such that $\mathbf{P} \cap \mathbf{A} = \emptyset$, and Δ_X is a set of propositional sentences constructed from atoms in $\mathbf{P}$ and $\mathbf{A}$. Here, $\mathbf{P}$ is called the set of non-assumables; $\mathbf{A}$ is called the set of assumables; Δ_X is called a database. It is required that Δ_X be consistent with every $\mathbf{A}$-instantiation.

The propositional sentences in Δ_X describe normal behavior of the component. The set of assumables

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

$\mathbf{A} = \{\mathbf{ok1}, \mathbf{ok2}, \ldots\}$ represents the health of the component. Variables $ok1, ok2$ are called assumables since they are initially assumed to be true, i.e. the component is initially assumed to be healthy. Typically, there is only one assumable – ok – for a each component. $\mathbf{P}$ is a set of the component ports (non-assumables). Ports are the inputs and outputs of a component.

Definition 3 (Observation) Given a component X description $(\mathbf{P}, \mathbf{A}, \mathbf{\Delta_X})$, a component observation is a consistent conjunction of $\mathbf{P}$-literals.

An observation of component behavior is a sentence containing a conjunction of a subset (not all variables has to be present in the observation) of non-assumable literals, e.g. $A \wedge \neg C \wedge D$.

A component is considered faulty when true valuation of assumables (health variables) can no longer be justified or, strictly speaking, when the observation ϕ is inconsistent with $\Delta_X \cup \mathbf{A}$. In that case, it is necessary to relax some assumables (i.e. replace some instances of ok with $\neg ok$) in order to restore consistency. Such relaxation is called *diagnosis* providing it is consistent with the component description and observation.

Definition 4 (Diagnosis) Given a component description $(\mathbf{P}, \mathbf{A}, \mathbf{\Delta_X})$ and a component observation ϕ, a diagnosis is an $\mathbf{A}$-instantiation that is consistent with $\Delta_X \cup \{\phi\}$.

If we take a closer look at the condition of consistency given in Definition 2, we may realize the following fact – the condition guarantees that when we have no observation of the component behavior, i.e. $\phi = true$, then there are no diagnoses and thus we cannot conclude anything about its health.

The strongest conclusion one can come to about the health of a component for a given state of its ports is called *component consequence*.

Definition 5 (Projection) The projection of an instantiation (clause) α on a set of atoms $\mathbf{S}$, written $\alpha_{\mathbf{S}}$, is the conjunction (disjunction) of all $\mathbf{S}$-literals in α. If Δ is a database, then $\alpha_{\Delta}$is the projection of α on the atoms appearing in Δ.

For example, the projection of $\alpha = A \wedge B \wedge \neg C$ on atoms $\{A, C\}$ is $A \wedge \neg C$.

Theorem 1 (Component consequence) Let X be a component with a set of ports $\mathbf{P}$ and description Δ_X (in clausal form). If ϕ is an instantiation of atoms $\mathbf{P}$, then

$$Cons_{\mathbf{A}}^{\Delta_X}(\phi) = \bigwedge_{\alpha \in \Delta_X, \phi \models \neg\alpha_{\mathbf{P}}} \alpha_{\mathbf{A}}.$$

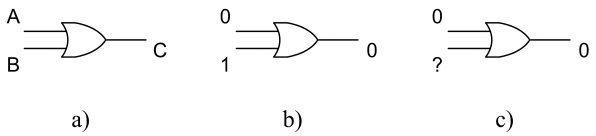

Fig. 1: Diagnosing an OR gate.

■ *Example 1:* Consider an OR gate (X) from Figure 1a. It has the inputs A, B and an output C; the assumable ok denotes the health state of the component. The component description in clausal form is as follows.

$$\Delta_X = \left\{ \begin{array}{c} \neg ok \vee A \vee B \vee \neg C \\ \neg ok \vee \neg A \vee C \\ \neg ok \vee \neg B \vee C \end{array} \right\}$$

The projections $\alpha_{\mathbf{P}}$ and their negations for each clause α in Δ_X are as follows.

$$\begin{array}{lll} \alpha_{\mathbf{P}}: & A \vee B \vee \neg C & \neg\alpha_{\mathbf{P}}: \quad \neg A \wedge \neg B \wedge C \\ & \neg A \vee C & \quad\quad A \wedge \neg C \\ & \neg B \vee C & \quad\quad B \wedge \neg C \end{array}$$

Let us assume we observe $\phi_1 = \neg A \wedge B \wedge \neg C$ (see Figure 1b). Then the third clause is the only one that fulfils condition $\phi_1 \models \neg\alpha_{\mathbf{P}}$. Therefore, the component consequence for observation ϕ_1 is $\neg ok$ which is the projection $\alpha_{\mathbf{A}}$ of the third clause on the assumables. The component is considered as faulty.

We may as well observe different state of the ports $\phi_2 = \neg A \wedge \neg C$ (see Figure 1b). In this case, there is no clause fulfilling condition $\phi_2 \models \neg\alpha_{\mathbf{P}}$. Therefore, the consequence for observation ϕ_2 is $true$. The component is considered as exhibiting normal behavior.

$$\begin{array}{rl} Cons_{\mathbf{A}}^{\Delta_X}(\neg A \wedge B \wedge \neg C) & \equiv \neg okX, \\ Cons_{\mathbf{A}}^{\Delta_X}(\neg A \wedge \neg C) & \equiv true. \end{array}$$

B. Component Diagnosability Definition

We say that a component is diagnosable with a given set of observed ports if there is at least one valuation of these ports that implies abnormal behavior of the component. Using the notation of consequence, described in section II, the component diagnosability is defined formally as follows:

Definition 6 (Component diagnosability) Given a component description Δ, set of component health variables $\mathbf{H}$, set of observed component ports $\mathbf{P_o}$, and set of unobserved component ports $\mathbf{P_u}$, $\mathbf{P_o} \cap \mathbf{P_u} = \emptyset$, one can state that component is diagnosable iff

$$\exists \phi_o \forall \phi_u \quad Cons_{\mathbf{H}}^{\Delta}(\phi_o \wedge \phi_u) \neq true.$$

Given the ports we cannot observe, the definition claims that if there is at least one combination of values for the remaining ports (that are observed) which may entail the fact that the component is faulty, no matter what the values of the unobserved ports may be, the component *is diagnosable under given conditions*.

■ *Example 2:* In Fig. 2 a), there is a three-port AND gate with inputs A and B, and output C. Clearly, when we observe all its ports, the component *is diagnosable*. On the other hand, when we observe only one of its ports, the component is not diagnosable. The question is, are two ports still enough for diagnosability?

When observing two out of the three ports, we either happen to observe the output and one of the inputs (Fig. 2b) or only the inputs (Fig. 2c).

The first possibility is more effective since there exists a combination (input A at zero, output C at one) that would prove the component is faulty no matter what the state of unobserved input B is. Therefore, an AND

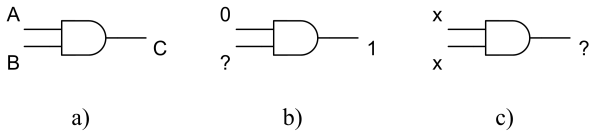

Fig. 2: AND gate example.

gate is diagnosable when observing its output and one of the inputs.

In the other possibility, there is no combination of A and B values that may on its own evince components malfunction. Therefore, an AND gate *is not diagnosable* when observing only its inputs.

C. Directional Resolution

In 1960, Davis and Putnam came up with a uniform proof procedure for quantification theory [Davis and Putnam, 1960]. Their so-called *'refutation algorithm'*, at present referred to as *DP-resolution* or *directional resolution* [Rish and Dechter, 2000], is a resolution algorithm deciding propositional satisfiability. It performs resolution along some ordering of propositional variables. The idea is that each clause is assigned the index of its highest literal in the ordering. Resolution is applied to the clauses with the same index and only on their highest literal. The algorithm systematically eliminates the literals from the set of clauses that are candidates for future resolution, starting with the highest literals. If, during the resolution, an empty resolvent is found, i.e. there is a contradiction in the theory φ, the algorithm stops and claims the theory is unsatisfiable. Otherwise, *directional extension* $E_o(\varphi)$ of φ along o is returned. It contains all the new resolved clauses together with the original clauses from theory φ. Detailed description of the algorithm can be found in [Rish and Dechter, 2000] or [Havel, 2006].

D. Diagnosability via Resolution

The idea behind deciding component diagnosability is based on *proof by contradiction*. Instead of looking for a specific combination of port values proving component's malfunction, *unsatisfiability* of a theory claiming the opposite, i.e. that the component is healthy, is examined.

We already know that a component has a set of health variables (assumables), typically one, that are true when the component exhibits normal behavior. The theory is therefore extended with one positive literal for each such health variable. On this extended theory, we resolve first over these assumables and then over the unobserved port variables (note that we are not interested in distinction between input and output ports of the component, the relation among them is the only thing that matters). The remaining buckets, which have not been resolved yet, must belong to the observed port variables. Now, there are only two possibilities – these buckets are either empty or they contain some clauses.

In the first case, if there are no clauses to resolve, we may never get an empty resolvent (contradiction) as a result of resolution and thus the theory *cannot be unsatisfiable*. The component is *not diagnosable*.

In the second case, there is at least one clause in these buckets. Since this clause (let us denote it γ) contains only literals with observed variables, one can find a set of clauses $\mathbf{C}$ which, if added to the theory, would lead to contradiction. Those would be negations of the literals from the mentioned clause.

$$\gamma = \bigvee_i \alpha_i \quad ; \qquad \mathbf{C} = \bigcup_{\mathbf{i}} \beta_{\mathbf{i}}, \quad \beta_{\mathbf{i}} = \neg\alpha_{\mathbf{i}}$$

The theory is then *unsatisfiable*. If there is a combination of observed variables that implies component's malfunction independently of the state of the other (unobserved) variables, we can claim that this component is *diagnosable*.

Formal description of the component diagnosability algorithm follows.

Component diagnosability algorithm

Input: a component description Δ in CNF, a set of health variables $\mathbf{H}$, a set of observed ports $\mathbf{P_o}$, and a set of unobserved ports $\mathbf{P_u}$.
Output: decision whether component is diagnosable or not.

1. **For** each health variable H from $\mathbf{H}$ **do**: $\Delta = \Delta \cup \{H\}$.
2. Create ordering $o = (O_1, \ldots, O_k, U_1, \ldots, U_l, H_1, \ldots, H_m)$, where $O_1, \ldots, O_l$ are members of $\mathbf{P_o}$, $U_1, \ldots, U_l$ are members of $\mathbf{P_u}$ and $H_1, \ldots, H_m$ members of $\mathbf{H}$ in arbitrary order.
3. Perform **directional resolution** on theory Δ, ordering o, and only over variables in $\mathbf{H} \cup \mathbf{P_u}$.
4. **If** the result is 'Δ is unsatisfiable',
   return 'component is faulty by nature'
   else
   store the *buckets*.
5. **If** $\forall i, i \in (1 \ldots Card(\mathbf{P_o}))$ $\quad \mathbf{bucket_i} = \emptyset$,
   return 'component is not diagnosable'
   else
   return 'component is diagnosable'.

■ *Example 3:* To illustrate how the procedure works, let us take a look at a 4-bit multiplexer from Fig. 3. It has four data inputs *X0, . . . , X3*, two selection inputs *A0, A1*, and a data output *Y*. We will take a look at two similar cases, in each observing the output *Y*, the selection input *A0*, but different data inputs of the multiplexer.

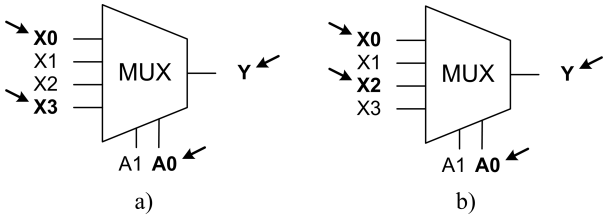

Fig. 3: A 4-bit multiplexer with two different observation conditions.

In the first case (Fig. 3a), the algorithm is told that only ports *A0*, *Y*, *X0*, and *X3* may be observed. After

resolution over the component health variable and unobserved variables, buckets belonging to the observed variables remain empty. Under such observation conditions, the multiplexer *is not diagnosable*.

In the second case (Fig. 3b), the observed ports are *A0*, *Y*, *X0*, and *X2* (instead of *X3*). After resolution, one of the observed variable buckets contain clauses. The multiplexer *is diagnosable*. These clauses tell us which combinations of the observed variable values would prove the component malfunction.

More details on component diagnosability procedure and a proof of the method can be found in [Havel, 2006].

III. Model Pruning

Suppose we examine a system that is only partially observed. We know the system structure, formal descriptions of the components, and the set of nodes that can be possibly observed. Having a tool for examining diagnosability of the components, we can try to systematically remove those components from the system that are not diagnosable under the given observation conditions and therefore do not contribute to the overall system diagnosis. This way, we may eventually get a simplified model of the system without unnecessary (not diagnosable) elements.

However, it still remains to be sorted out which components and in what order should be examined. Even though there are descriptions which are inherently non-diagnosable, we may assume that if we observe all component's ports, then the component is typically diagnosable. Therefore, we want to start from those components which have some of their ports unobserved.

Note that even if a node is not observed, it cannot be automatically considered as "not having a specific value". Assume that a port of the examined component connected via this node to another component. This other component may have values on its remaining ports which may in turn force some value on the node common to both components. Therefore, the examined component cannot consider its port adjoining to this node as unobserved even though the node itself is technically unobserved.

Then we may come to a single conclusion – the examination should start from the components that have at least one of its ports *not connected to any other component* and, at the same time, *unobserved*. In the following description, those ports (or in fact the adjacent nodes) are referred to as *"free dead-ends"*.

A. Algorithm Description

The pruning algorithm works on a set of system components and their ports, a set of system connections, and an incidence relation (edges) over the ports and the connections. The connections, here called *nodes*, connect together ports of the components. In an extreme case, a node can be incident to only one component port – then this node is called a *"dead-end"*. At any stage of the processing, the nodes are in one of three states: *unknown*, *bound*, or *free*. Initially, each node is in either *unknown* (providing it is not observed) or *bound* state (providing it is observed). The states of the nodes may change in time. Typically, when a node in *unknown* state has been examined, its state is changed to either *bound* or *free* state.

Examination of a node comprises of deciding diagnosability of the adjoining component. State of the nodes connected to the component ports are interpreted so that a *free* node marks the incident port as unobserved and a *bound* or *unknown* node marks the incident port as observed. The procedure from Section II-D is then applied to decide the component diagnosability.

Formal description of the proposed *model pruning algorithm* follows.

Model pruning algorithm

Input: a set of component descriptions and health variables, a set of nodes $\mathbf{N}$, a set of observed nodes $\mathbf{N_o} \subseteq \mathbf{N}$, and a projection $\varepsilon : \mathbf{P} \to \mathbf{N}$ of the ports onto nodes.
Output: simplified model of the system.

1. **Initialize:**
   Assign a state to every node, *'bound'* if the node is observed, *'unknown'* otherwise.
   For each node n in the model **do**:
   if $state(n) \neq$ *'bound'* and $card(n) = 1$ **then** insert n into **DEL**.
2. **If DEL** is not empty **then**
   remove the first node n from **DEL** and go to step 3
   else
   stop and **return** the current model.
3. Find the component c_n connected to the node n.
   For each port p of the component c_n **do**:
   if $state(\varepsilon(p)) =$ *'free'* **then**
   insert p into the set of unobserved ports $\mathbf{P_u^{c_n}}$
   else
   insert p into the set of observed ports $\mathbf{P_o^{c_n}}$.
4. Decide **component diagnosability** of the component c_n, using component description Δ^{c_n}, set of health variables $\mathbf{H^{c_n}}$, and sets of ports $\mathbf{P_o^{c_n}}$ and $\mathbf{P_u^{c_n}}$.
5. **If** the result is 'component is diagnosable' **then**
   set $state(n) =$ *'bound'* and go back to step 2.
6. Delete the component c_n and the node n from the model.
   For each node n' previously connected to the component c_n **do**:
   if n' is in **DEL then**
   remove n' from **DEL** and delete it from the model
   else
   if $card(n') = 0$ **then** delete n' from the model
   if $card(n') = 1$ and $state(n') \neq$ *'bound'* **then**
   set $state(n') =$ *'free'* and insert n' into **DEL**.
   Go back to step 2.

The algorithm maintains a list of nodes, called "dead-end list" or **DEL**, which contains nodes that are not *bound* and are "dead-ends" in the actual model. The initial order of nodes in **DEL** is arbitrary. Also, the newly created "dead-ends" can be inserted either at the beginning or the end of the list without any impact on the result of the procedure. The list is used for finding components which are candidates for diagnosability testing and therefore for potential removal from the model – the components belonging to the nodes from the list are one by one examined. When a component is removed, then the incident nodes which became "dead-ends" are inserted to the **DEL** and the incident nodes which became "orphans" (because they were al-

ready in the **DEL**) are removed from the model. This way, the non-diagnosable components are being bit by bit "torn off" from the model. The strategy is based on an assumption that only the components having some of its ports unobserved are likely to be non-diagnosable.

■ *Example 4:* Let us illustrate the model pruning procedure on an fictional example from Fig. 4. The

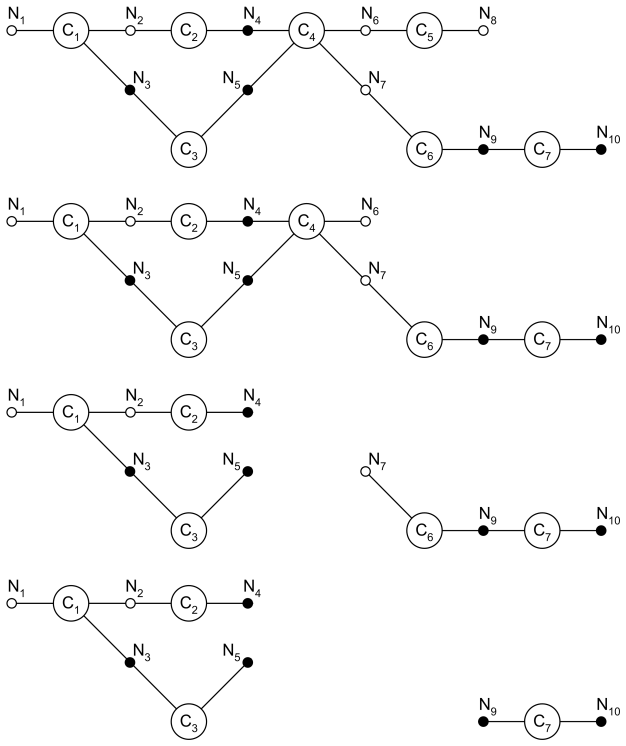

Fig. 4: An example of model pruning sequence.

model of a system consists of components $C_1, \ldots, C_7$ and nodes $N_1, \ldots, N_{10}$. The nodes N_1, N_2, N_6, N_7, N_8 are not to be observed. In the initial step, the **DEL** is filled with nodes N_1 and N_8.

In the next step, N_1 is removed from the list. The port of the adjoining component C_1 connected to the *free* node N_1 is considered as unobserved, the other two ports are considered as observed. Suppose the result of examining C_1 is that the component is diagnosable in this setting. C_1 is kept in the model.

After this, the other "dead-end", N_8, is removed from the list. The port of the adjoining component C_5 connected to the *free* node N_8 is considered as unobserved, the other port connected to the *unknown* node N_6 is considered as observed. Suppose the result of examining C_5 is the component is not diagnosable. The component is therefore removed from the model and the node N_6 becomes a "dead-end" and is inserted into the **DEL**.

Because N_6 is the only node left in the **DEL**, the component C_4 is examined. Suppose C_4 is not diagnosable when not observing its port connected to N_6. This causes the component removal and N_7 becoming a "dead-end".

Similarly the scenario repeats for the component C_6 except that this time no node is added to the **DEL** since node N_9 is *bound*. As the **DEL** is empty, the procedure stops. The result of model pruning is shown at the bottom of Fig. 4. Three components – C_4, C_5, C_6 – were evaluated as irrelevant for diagnosis under the previously mentioned observation conditions.

IV. Experiments

We will examine diagnosability of a digital circuit under various observation conditions revealing different aspects of the simplification procedure. To verify their functionality, both the component diagnosability and model pruning algorithms were implemented in *Python* programming language.

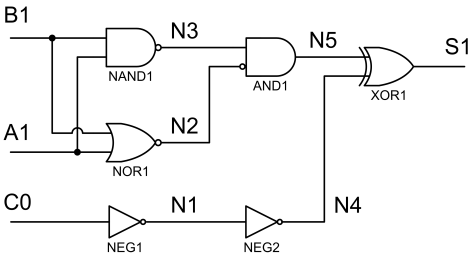

Fig. 5: A fragment of 4-bit binary adder with fast carry.

The circuit in Fig. 5 is a fragment of 4-bit binary adder with fast carry, namely the part belonging to the first sum output-bit. There are inputs *C0, A1, A2*, an output *S1*, five internal nodes *N1* $\cdots$ *N5*, and six logic gates denoted *NEG1*, *NEG2*, *NOR1*, *NAND1*, *AND1*, and *XOR1*.

We are going to analyse three distinct cases, in each observing different set of nodes. The first two cases will demonstrate the model simplification process with results as they are expected, whereas the last case will reveal a deficiency in the model pruning algorithm that is to be eliminated yet.

Case 1

Let us suppose we are in a situation where we observe the inputs *A1, A2*, the output *S1*, and the internal nodes *N2, N4*. The state of the input *C0* and of all the other internal nodes is not known.

Initially, we set the state of nodes *A1, A2, S1, N4* to *bound*, and the state of the remaining nodes to *unknown*. The node *C0*, which is the only one "dead-end" in the model, enters the **DEL** and change its state to *free*. Subsequently, the adjoining component *NEG1* is tested for diagnosability with negative result. Therefore, *NEG1* is removed from the model together with the node *C0*. The node *N1* now becomes a *free* "dead-end" and is put into **DEL**. The next step is similar except that the node *N4* does not appear in the **DEL** because its state is *bound*. Thus the **DEL** is empty and the procedure stops.

Two components were eliminated from the model based on the given observation conditions. The simplified model is depicted in Fig. 6 (a).

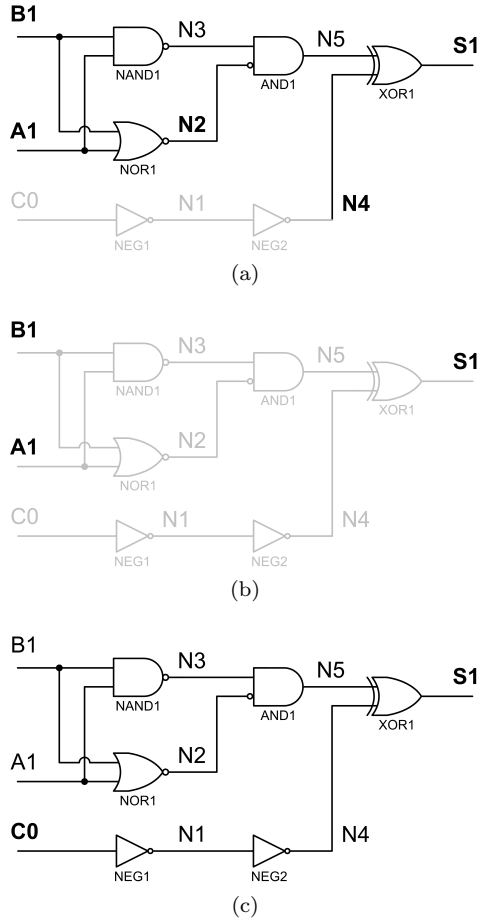

Fig. 6: Simplified models for different observation conditions (names of the observed nodes are printed in bold).

Case 2

Consider a situation similar to the one from case 1 except that we do not observe the internal nodes *N2* and *N4*. Therefore, we initially set the state of nodes *A1, A2, S1* to *bound*, and the state of the remaining nodes to *unknown*.

The algorithm first removes both negators *NEG1* and *NEG2*, just as in the previous case. Since *N4* is not *bound* this time, it gets to the **DEL**, its state is changed from *unknown* to *free*, and the *XOR1* is subsequently examined for diagnosability. Because the XOR gate is not diagnosable even if not observing only one of its inputs, it is taken away from the model as well. Along with this change, the nodes *N4* and *S1* disappear, and the node *N5* is put into the **DEL** with its state set to *free*. At this moment, the adjoining component *AND1* happens to be non-diagnosable, because its output is connected to *free N5*. *AND1* is removed and the nodes *N2* and *N3* become the new *free* nodes in **DEL**. Since the outputs of the two remaining gates *NAND1* and *NOR1* are now connected to *free* nodes, both of them are in turns removed too.

As you can see from Fig. 6 (b), the simplified model is empty. Interestingly, the impossibility to observe just one of the inputs (*C0*) caused the whole circuit to be non-diagnosable.

Case 3

The last case exposes a weak point of the current version of the pruning algorithm. Consider a case, where only the input *C0* and the output *S1* are observed, all other variables are hidden.

Intuitively, we would anticipate an attempt to remove the components incident to the unobserved inputs *A1* and *B1*. In fact, nothing is done since the algorithm does not consider the inputs as "dead-ends". The other two "dead-end" candidates, *C0* and *S1*, are *bound* and so the **DEL** stays empty.

The result is depicted in Fig. 6 (c). Nothing has been removed even though, in this particular case, the gates *NAND1* and *NOR1* could have been removed. To correct this, one would need to consider internal *unbound* nodes to search for removal candidates as well. This is an issue that needs to be explored yet.

V. Conclusion

The concept of component diagnosability was used to construct a pruning procedure which simplifies diagnostic model of a system for some given observation conditions. This is achieved by systematical removing of segments of the model which are not diagnosable. An experiment demonstrated functionality of the procedure as well as it revealed its weak point – inability to cut out the model from its inner nodes – which leaves an open question for the future work.

VI. Author biography

IVAN HAVEL was born in Prague, Czech Republic and went to the Czech Technical University in Prague, where he studied technical cybernetics and obtained his master's degree (Ing.) in 2003. In the meantime, he also studied for one year at the Union College in New York, USA, where he earned Master of Science in Electrical Engineering in 2002. Currently, he is working at the Rockwell Automation Research Center in Prague and he is pursuing a PhD degree at the Department of Cybernetics of the Czech Technical University. His e-mail address is: havel@labe.felk.cvut.cz.

References

[Darwiche, 1998] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[Davis and Putnam, 1960] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3):201–215, 1960.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[Havel, 2006] Ivan Havel. Component diagnosability via directional resolution. In *Cybernetics and Systems 2006*. Vienna: Austrian Society for Cybernetic Studies, 2006. ISBN 3-85206-172-5.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Rish and Dechter, 2000] Irina Rish and Rina Dechter. Resolution versus search: Two strategies for SAT. *Journal of Automated Reasoning*, 24(1/2):225–275, 2000.