Automatische constructie van hiërarchische modellen voor model-gebaseerde diagnose[1]

D.-J. Out
R.P. van Rikxoort
Universiteit Twente

Samenvatting
Een van de belangrijkste problemen bij model-gebaseerde diagnose is het vinden van het meest bruikbare model of verzameling van modellen van het systeem onder diagnose. Vaak is er alleen een model met de fysieke componenten en de verbindingen daartussen beschikbaar. Bij grotere systemen kunnen de huidige diagnostische algoritmen niet goed omgaan met dergelijke gedetailleerde modellen. Een oplossing hiervoor is het gebruik van een hiërarchisch model. Hiermee kunnen we eerst een oplossing bepalen met behulp van een abstract model en deze oplossing vervolgens gebruiken om het diagnostisch proces in het meer gedetailleerde model te sturen. Het grote probleem bij deze aanpak is echter het verkrijgen van het hiërarchische model. In dit artikel laten we een generiek diagnostisch algoritme zien en tonen aan de hand daarvan aan dat het gebruik van bepaalde klassen van hiërarchische modellen computationele voordelen biedt. We presenteren lineaire tijd algoritmen waarmee we deze hiërarchische modellen automatisch kunnen genereren. Hiervoor maken we gebruik van informatie over kosten van meetpunten en inverteerbaarheid van componenten.

Trefwoorden
Model-gebaseerde diagnose, automatisch modelleren, hiërarchisch modelleren.

1 Inleiding

Model-gebaseerde diagnose van technische systemen (de Kleer en Williams, 1987; Davis, 1984) is een aanpak die een model van een systeem gebruikt voor simuleren en redeneren. Dit model wordt gebruikt om discrepanties te vinden tussen geobserveerd gedrag en gedrag dat door het model wordt voorspeld. Deze discrepanties worden vervolgens gebruikt om de diagnoses van het systeem te berekenen.

Een model van een technisch systeem bestaat uit (1) componenten en verbindingen tussen deze componenten, en (2) gedragsbeschrijvingen van deze componenten. Als een component correct werkt, dan is zijn gedrag gelijk aan de gedragsbeschrijving, anders kan hij ieder mogelijk gedrag vertonen. In model-gebaseerde diagnose proberen we verzamelingen componenten te vinden, zodanig dat als van ieder van deze componenten wordt aangenomen dat hij niet werkt, er geen discrepanties meer zijn tussen geobserveerd gedrag en gedrag dat door het model wordt voorspeld. Deze verzamelingen componenten worden diagnoses genoemd. We kunnen deze diagnoses gebruiken bij verdere diagnostische acties (zoals het bepalen van het beste meetpunt).

---

[1] Dit onderzoek wordt mede gefinancierd door de Stichting Knowledge-Based Systems (SKBS). De SKBS stimuleert de samenwerking tussen universiteiten en het bedrijfsleven op het gebied van de kennistechnologie.

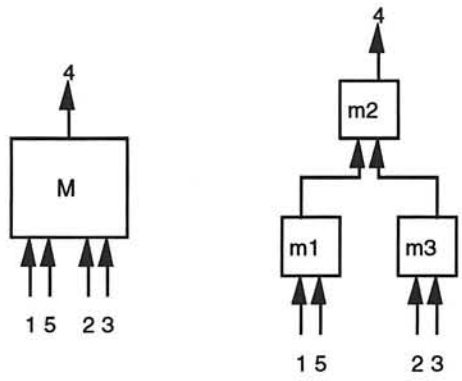
2 Hiërarchie en model-gebaseerde diagnose

Traditionele diagnostische algoritmes hebben een tijdscomplexiteit van $O(2^n)$ als n het aantal componenten in het model is. Met hiërarchische modellen kunnen we dit reduceren tot O(n) (Mozetic, 1990). Meer recente best-first algoritmes (de Kleer, 1991) hebben een tijdscomplexiteit van $O(n^{i+1})$ waarin i het aantal defecte componenten is. Hammink (1991) laat zien dat we dit kunnen reduceren tot $O(^d\log(n)*d^{i+1})$ als iedere component van een model in de hiërarchie uit precies d componenten bestaat en we een unieke diagnose op ieder niveau van de hiërarchie kunnen bepalen[2]. Het algemene algoritme voor hiërarchische model-gebaseerde diagnose is algoritme 1.

Algoritme 1

1. Genereer een verzameling diagnoses $D_{abstract}(a)$ voor het meest abstracte model (niveau a).
2. Transformeer $D_{abstract}(a)$ naar een verzameling mogelijke diagnoses $D_{detailed}(a-1)$ voor het gedetailleerde model.
3. Test iedere mogelijke diagnose in $D_{detailed}(a-1)$, wat een verzameling diagnoses $D_{tested}(a-1)$ oplevert.
4. Discrimineer tussen de diagnoses in $D_{tested}(a-1)$ door het uitvoeren van metingen[3] totdat de verzameling voldoet. We noemen het resultaat $D_{probed}(a-1)$.
5. Als a=2 dan zijn we klaar,
   anders $D_{abstract}(a-1):=D_{probed}(a-1)$; a:=a-1 en ga naar Stap 2.

We illustreren dit algoritme met een voorbeeld:



Figuur 1. Een 4-input maximizer geïmplementeerd door drie 2-input maximizers

Voorbeeld 1
Beschouw het hiërarchische model afgebeeld in Figuur 1. Op het hoogste niveau hebben we slechts een diagnose, dus Dabstract(2) = {[M]}. Als we dit vertalen naar een meer gedetailleerd niveau, krijgen we de verzameling mogelijke diagnoses Ddetailed(1) = {[m1], [m2], [m3], [m1, m2], [m1, m3], [m2, m3], [m1, m2, m3]}. Als we deze mogelijke diagnoses nu

---

2 Deze complexiteit is onder de aanname dat het testen van kandidaten op ieder niveau van de hiërarchie even duur is. Voor een discussie hiervan, zie (Genesereth, 1984).

3 In modelgebaseerde diagnose vindt discriminatie traditioneel plaats door metingen. Er zijn ons geen efficiënte algoritmes bekend voor generatie van tests die discrimineren tussen diagnoses.

testen, zien we dat [m3] geen diagnose is, want als m1 and m2 correct werken, dan moet de uitvoer van m2 tenminste 5 zijn. Dus Dtested(1) wordt {[m1], [m2], [m1, m2], [m1,m3], [m2, m3]; [m1, m2, m3]}. We kunnen hier vervolgens tussen discrimineren.

2.1 Gewenste eigenschappen van hiërarchische modellen

Uiteraard zijn sommige hiërarchische modellen beter dan andere. We noemen hier een drietal gewenste eigenschappen van hiërarchische modellen.

Downward failure eigenschap (Weld en Addanki, 1990) : Als een hiërarchisch model deze eigenschap heeft, betekent dit dat kandidaten die niet consistent zijn op een abstract niveau, deze dat ook niet zijn op een gedetailleerd niveau in de hiërarchie. We kunnen dan ook de resultaten die we op een abstract niveau berekend hebben, gebruiken om de zoekruimte in te perken op het gedetailleerde niveau. Deze inperking van de zoekruimte is de opbrengst van het gebruik van een hiërarchisch model. In ander werk aan abstractie bestaan concepten die gelijksoortig zijn aan de downward failure eigenschap, bijvoorbeeld TI-abstracties (Giunchiglia en Walsh, 1990), upward solution eigenschap (Knoblock, 1990), de consistentie conditie C2 (Mozetic, 1990), en C-herformuleringen (Out en Bakker, 1992).

Makkelijk testbaar : Het testen van de verzameling van mogelijke diagnoses (in stap 3 van algoritme 1) kan een kostbaar proces zijn (Out en Bakker, 1992). Het is daarom gewenst om hiërarchische modellen zo te ontwerpen dat dit wordt geminimaliseerd. We zullen een klasse van hiërarchische modellen definiëren, ID-hiërarchieën genaamd, die stap 3 van algoritme 1 totaal overbodig maken.

Goede discrimineerbaarheid : Als we een hoog niveau diagnose transformeren (in stap 2 van algoritme 1) kan de resulterende verzameling van laag niveau diagnoses erg groot zijn. We willen dan ook zo goed (en goedkoop) mogelijk discrimineren (in stap 4) als mogelijk is, zodat er zo min mogelijk diagnoses overblijven. We zullen een klasse van hiërarchische modellen definiëren, genaamd SD-hiërarchieën, waarbij er voor gezorgd wordt dat we altijd tegen redelijke kosten kunnen discrimineren totdat er een unieke diagnose overblijft.

2.2 Creëren van hiërarchische modellen

Een van de grootste problemen bij het gebruik van hiërarchische modellen is het creëren van deze modellen. In dit artikel genereren we een hiërarchisch model vanuit een niet-hiërarchisch model. Voor ieder model in de hiërarchie moeten we twee dingen specificeren:

1. Een structuurmodel, dat laat zien hoe componenten zijn verbonden.
2. Een verzameling gedragsbeschrijvingen van de componenten.

In dit artikel construeren we alleen abstracte structuurmodellen voor onze hiërarchie. De gedragsbeschrijvingen van de componenten in abstracte modellen kunnen direct worden afgeleid uit de gedragsbeschrijvingen van de componenten waar ze uit zijn samengesteld en worden niet geabstraheerd[4].

4 Automatisch abstraheren van gedragsbeschrijvingen is een extreem moeilijk probleem (Hamscher, 1988; Lbath, 1987) dat, hoewel er wel werk is gedaan in deze richting (Kumar en Upadhyaya, 1990), nog steeds grotendeels onopgelost is.

3 ID[5]-hiërarchieën

In stap 2 van algoritme 1, wordt een verzameling van diagnoses $D_{abstract}$ voor een abstract model getransformeerd in een verzameling van mogelijke diagnoses $D_{detailed}$ voor een gedetailleerd model. Vervolgens (stap 3), worden deze mogelijke diagnoses getest op het diagnose zijn voor het gedetailleerde model, resulterend in de verzameling D_{tested}. Dit testen kan een zeer tijdrovend proces zijn (Out en Bakker, 1992).

In deze sectie zullen we een korte[6] bespreking geven van ID-hiërarchieën. Deze kunnen informeel worden gedefinieerd als hiërarchieën met de eigenschap dat als we een diagnose van een abstract model vertalen naar een verzameling mogelijke diagnoses van een gedetailleerd model, dat iedere mogelijke diagnose in die verzameling ook een diagnose is, zodat $D_{tested}=D_{detailed}$. We illustreren dit met een voorbeeld:

Voorbeeld 2
Beschouw nogmaals het hiërarchische model in figuur 1. In voorbeeld 1 hebben we gezien dat $D_{tested}=D_{detailed}$, dus deze hiërarchie is geen ID-hiërarchie. Als we nu echter aannemen dat M een 4-input opteller is en m1, m2 en m3 zijn 2-input optellers, dan is de mogelijke diagnose [m3] nu wel een diagnose, net als alle andere mogelijke diagnoses, dus $D_{tested} = D_{detailed}$ waaruit volgt dat deze hiërarchie nu wel een ID-hiërarchie is.

Het voordeel van ID-hiërarchieën is dat stap 3 van algoritme 1 kan worden overgeslagen, aangezien $D_{tested}=D_{detailed}$.

3.1 Genereren van ID-hiërarchieën met twee niveaus

In deze sectie presenteren we een algoritme wat een abstract model genereert vanuit een gedetailleerd model, zodanig dat de resulterende hiërarchie van twee niveaus een ID-hiërarchie is. Het bewijs dat dit algoritme inderdaad een ID-hiërarchie vormt is vrij ingewikkeld en is opgenomen in (van Rikxoort, 1991). Het algoritme neemt als invoer een (gedetailleerd) model en informatie over de inverteerbaarheid[7] van de componenten van dat model en genereert als uitvoer een abstract model.

Laat $C_{detailed}$ de verzameling componenten van het gedetailleerde model zijn. Het algoritme genereert vervolgens een verzameling $C_{abstract}$ van componenten van het abstracte model zodanig dat ieder component in $C_{abstract}$ correspondeert met een deelverzameling van $C_{detailed}$. Voor ieder component $c \in C_{detailed}$ definiëren we de volgende twee verzamelingen: **Pre(c)**, de directe voorgangers van c (componenten wiens uitvoer direct verbonden is met een van de invoeren van c). **Post(c)**, de opvolgers van c (componenten wiens invoer beïnvloed wordt door de uitvoer van c, danwel direct, danwel via andere componenten.

Algoritme 2
1. $C_{abstract} := \{\{c\} \mid c \in C_{detailed} \wedge$ een van de uitvoeren van c is een systeem uitvoer$\}$
2. Neem een component c van een verzameling $S \in C_{abstract}$ zodat er een component $v \in$ Pre(c) bestaat die nog geen element is van enige verzameling in $C_{abstract}$.
3. Als Post(c) $\cup$ {c} = Post(v) en c is inverteerbaar dan S := S $\cup$ {v} anders $C_{abstract} := C_{abstract} \cup \{v\}$

---

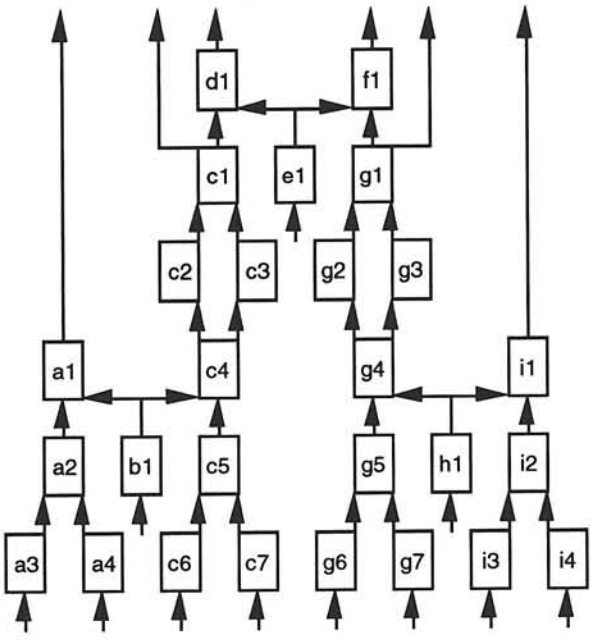
5 ID staat voor indistinguishability (Ononderscheidbaarheid). Dit betekent dat fouteffecten van alle componenten binnen een abstracte component ononderscheidbaar zijn van elkaar. Deze eigenschap is gerelateerd aan het concept van 'fault-collapsing' (McCluskey en Clegg, 1971).

6 Voor een uitgebreide bespreking, zie (van Rikxoort, 1991).

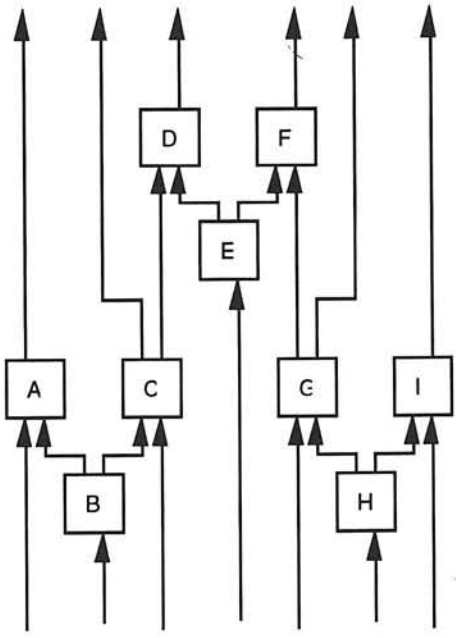
7 Een component is inverteerbaar als we van iedere invoer van die component de waarde kunnen berekenen gegeven de waardes van de uitvoer en alle andere invoeren van die component.

4. Als nog niet alle componenten toegekend zijn aan een verzameling in $C_{abstract}$ ga naar Stap 2.

We illustreren dit algoritme met het volgende voorbeeld:



Figuur 2. Gedetailleerd model



Figuur 3. Abstract model

Voorbeeld 3
In dit voorbeeld illustreren we algoritme 2 door te laten zien hoe we het abstracte model in figuur 3 kunnen construeren uit het gedetailleerde model in figuur 2. We nemen aan dat in het gedetailleerde model alle componenten inverteerbaar zijn.

Stap 1: In deze stap verzamelen we alle componenten wiens uitvoer een systeemuitvoer is in $C_{abstract}$, dus $C_{abstract}$ wordt {{a1}, {c1}, {d1}, {f1}, {g1}, {i1}}.

Stap 2: In deze stap nemen we een component uit $C_{abstract}$ zodat het tenminste een directe voorganger heeft die niet in $C_{abstract}$ zit. We kiezen hier d1 met voorganger e1.

Stap 3: Aangezien Post(d1) $\cup$ d1 = {d1} en Post(e1) = {d1, f1}, wordt component e1 een deel van een nieuwe component, dus $C_{abstract}$:= {{a1}, {c1}, {d1}, {e1}, {f1}, {g1}, {i1}}.

Stap 4: Ga naar stap 2, aangezien niet alle componenten toegekend zijn.

Als we in stap 2, c1 met voorganger c2 hadden gekozen, dan zouden we in stap 3 hebben gevonden dat aangezien c1 inverteerbaar is en Post(c1) $\cup$ c1 = Post(c2), c1 en c2 tot dezelfde abstracte component behoren, dus $C_{abstract}$:= {{a1}, {c1, c2}, {d1}, {f1}, {g1}, {i1}}.

Het is makkelijk te zien dat het algoritme in O(n) tijd verloopt, waarin n het aantal componenten is in het gedetailleerde model. Het grootste minpunt ligt in de grote afhankelijkheid van inverteerbaarheid van de componenten; veel niet-inverteerbare

componenten zorgen voor veel (en dus kleine) abstracte componenten, zodat het abstracte model grotendeels hetzelfde is als het gedetailleerde model.

3.2 Genereren van ID-hiërarchieën met meerdere niveaus

Algoritme 2 neemt een gedetailleerd model als invoer en genereert een abstract model als uitvoer, zodat een hiërarchisch model van twee niveaus ontstaat. We kunnen dit uitbreiden naar ID-hiërarchieën van meerdere niveaus gebruikmakend van klassen van meetkosten (Hammink en Out, 1990; Hammink, 1991).

3.2.1 Klassen van meetkosten

Een fysiek systeem bevat meetpunten, alwaar we metingen kunnen uitvoer en om tussen diagnoses te discrimineren. Vaak zullen deze kosten niet identiek zijn voor alle meetpunten, sommige zijn fysiek moeilijk te bereiken, voor sommige is dure meetapparatuur nodig, op sommige staan signalen die moeilijk interpreteerbaar zijn, etc. We nemen aan dat er een schatting van de meetkosten aanwezig is en gebruiken deze om meetpunten met ongeveer gelijke kosten in klassen van meetkosten te verzamelen (CPCs). De afmeting en aantal van deze klassen hangen af van de applicatie die gediagnostiseerd wordt. De goedkoopste klasse bevat de systeem uitvoeren.

3.2.2 Genereren van multi-level ID-hierarchieën

Met behulp van CPCs, kunnen we een ID-hiërarchie van meerdere niveaus opbouwen. We nemen aan dat we n CPCs hebben, gelabeld $CPC_1 ... CPC_n$ zodat CPC_n de goedkoopste klasse is. We hebben al het meest gedetailleerde niveau, en we kunnen algoritme 2 gebruiken om level n te genereren (het meest abstracte model). Om niveau n-1 van de hiërarchie te creëren, gebruiken we algoritme 2 weer, maar nu nemen we de verzameling $CPC_n \cup CPC_{n-1}$ als systeem uitvoer. Zo ook voor het niveau n-2, waar we $CPC_n \cup CPC_{n-1} \cup CPC_{n-2}$ gebruiken etc.

3.3 Hoe goed zijn ID-hiërarchieën?

Welke van de eerder genoemde gewenste eigenschappen hebben ID-hiërarchieën?

Downward failure eigenschap: Bij het creëren van ID-hiërarchieën gebruiken we twee operaties: samennemen van componenten en verwijderen van verbindingen. Mozetic (1990) bewijst dat als we alleen deze operaties gebruiken, we voldoen aan de consistentie conditie, die sterker is dan de downward failure eigenschap.

Makkelijk testbaar: Testen is niet nodig.

Goede discrimineerbaarheid: Het is niet altijd mogelijk om een unieke diagnose te bepalen tegen redelijke kosten. De meetpunten die mogelijk nodig zijn om een unieke diagnose te bepalen kunnen arbitrair duur zijn, zoals het volgende voorbeeld toont:

> **Voorbeeld 4**
> Stel dat we in het model in figuur 2, de uitvoer van component i2 hebben gemeten en het blijkt dat deze afwijkt. Dit betekent dat een van de componenten i2, i3 of i4 stuk moet zijn. Als de uitvoer van zowel i3 als i4 erg moeilijk meetbaar is, kunnen we de unieke diagnose niet goedkoop bepalen.

Zoals we zullen zien in de volgende sectie, vormen SD-hiërarchieën een mogelijke oplossing voor dit probleem.

4 SD[8]-hiërarchieën

In stap 4 van algoritme 1 discrimineren we tussen een verzameling diagnoses D_{tested} en we kunnen in voorbeeld 4 zien dat de kosten hiervan arbitrair hoog kunnen zijn. De complexiteit van hiërarchische model-gebaseerde diagnose is $O(^{d}\log(n)*(dp)^{i+1})$, waarin p het aantal diagnoses is wat overblijft op ieder niveau van de hiërarchie. Hierin kan n vrij groot zijn, maar d en i zijn meestal klein, dus we willen graag dat p zo klein mogelijk is. In deze sectie zullen we een korte[9] bespreking geven van SD-hiërarchieën. Deze kunnen informeel worden gedefinieerd als zodanig opgebouwd dat we altijd een unieke diagnose kunnen vinden met gebruikmaking van alleen meetpunten die goedkoper zijn dan een of andere van te voren vastgestelde kostenfactor.

4.1 Genereren van SD-hiërarchieën met twee niveaus

Om de unieke diagnose te vinden, moeten we soms zeer dure meetpunten meten. SD-hiërarchieën zijn zo geconstrueerd dat dit niet op kan treden. Dit geschiedt door het construeren van modellen die alleen relatief goedkope meetpunten bevatten; alle andere meetpunten worden verwijderd.

Algoritme 3
1. Als twee componenten alleen verbonden worden door 'dure' meetpunten, neem ze dan samen tot een component.
2. Verwijder alle 'dure' verbindingen.

Dit algoritme heeft een lineaire tijdscomplexiteit. Aangezien alle meetpunten in het abstracte model 'goedkoop' zijn, kunnen we voor dit abstracte model een unieke diagnose bepalen waarbij we alleen gebruik maken van "goedkope" meetpunten (om de unieke diagnose op het gedetailleerde niveau te bepalen hebben we wel de "dure" meetpunten nodig).

4.2 Genereren van SD-hiërarchieën met meerdere niveaus

We kunnen een SD-hiërarchie van meerdere niveaus genereren met gebruikmaking van klassen van meetkosten op een zelfde methode als beschreven in §3.2. Het meest abstracte model bevat alleen de goedkoopste klasse van verbindingen, etc. We geven hiervan weer een voorbeeld:

Voorbeeld 5
In figuur 5 is een gedetailleerd model afgebeeld. Dit model heeft vier CPCs:
Zeer duur : de verbindingen die met vier letters gelabeld zijn, zoals oaca.
Duur : de verzameling {oaa,oac,oca,occ,oba,obc}.
Goedkoop : de verzameling {oa2,oc1}.
Zeer goedkoop : de systeem invoeren en uitvoeren: {in0, ... ,in27}; {out0, ... ,out3}.

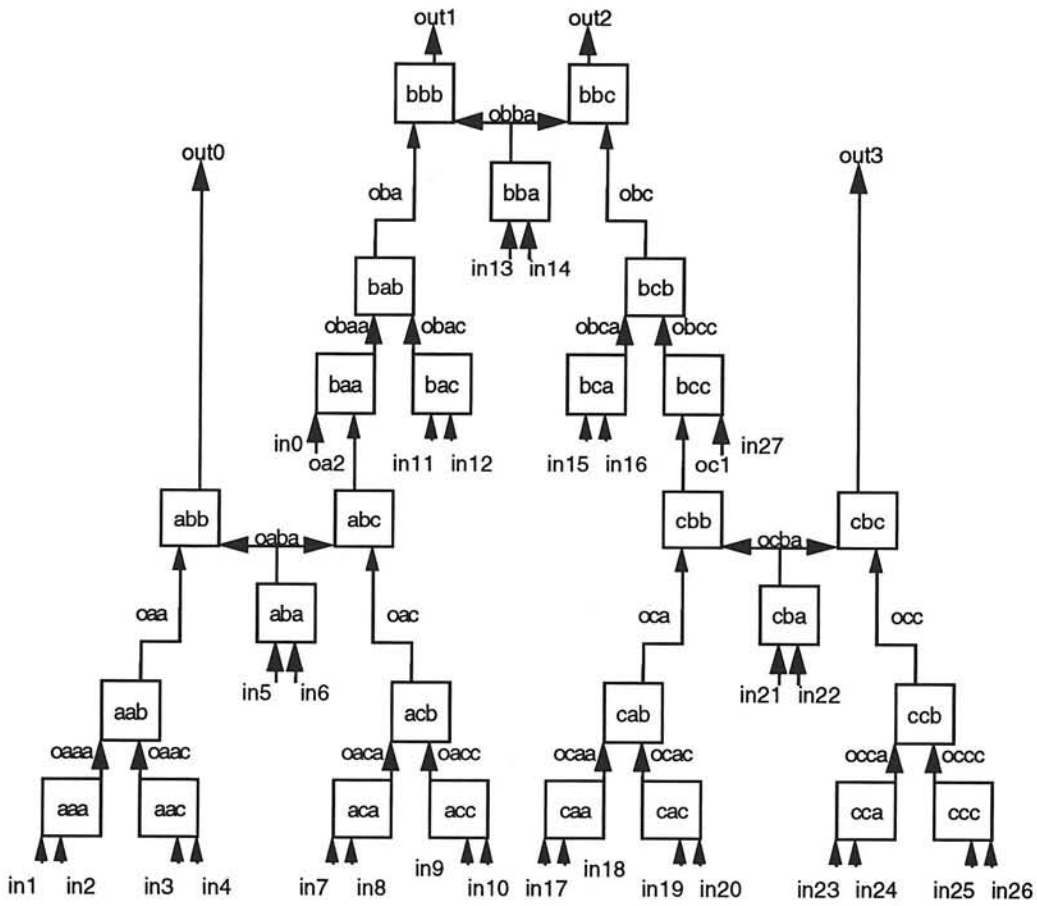
Als we de Zeer Dure verbindingen verwijderen uit het model in figuur 4 krijgen we het model in figuur 5. Als we vervolgens de Dure verbindingen verwijderen, resulteert dat in het model in figuur 6.

We kunnen ook de Goedkope verbindingen verwijderen uit het model in figuur 6 waarna een model ontstaat wat uit nog slechts een component bestaat.

---

8 SD staat voor Single Diagnosis (enige diagnose).

9 Voor een uitgebreide bespreking, zie (Hammink, 1991).



Figuur 4. Meest gedetailleerd model

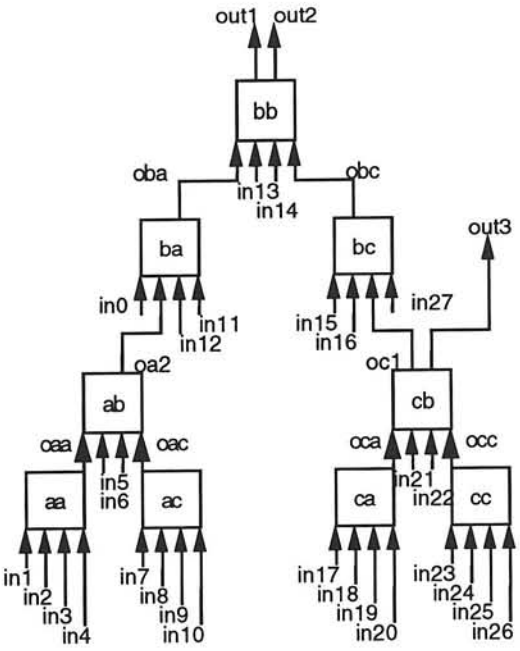
4.3 Hoe goed zijn SD-hiërarchieën?

Welke van de eerder genoemde gewenste eigenschappen hebben SD-hiërarchieën?

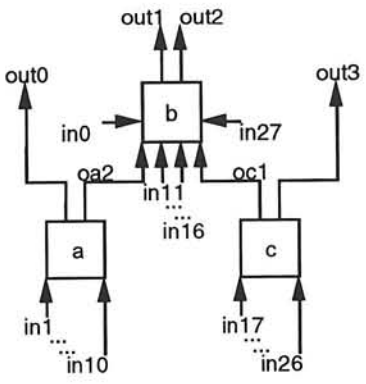
Downward failure eigenschap: Net als bij ID-hiërarchieën groeperen we componenten en verwijderen verbindingen, dus ID-hiërarchieën hebben de downward failure eigenschap.

Makkelijk testbaar: SD-hiërarchieën zijn niet noodzakelijk ID-hiërarchieën. Daarom moeten we alle mogelijke diagnoses testen.

Goede discrimineerbaarheid: In SD-hiërarchieën kunnen we altijd de unieke diagnose bepalen op een gegeven niveau met gebruikmaking van alleen meetpunten uit de corresponderende (of goedkopere) CPCs. Op hogere niveaus zullen dit goedkope meetpunten zijn, maar hoe dieper men afdaalt, hoe duurder de meetpunten kunnen zijn.



Figuur 5. Zeer Duur verwijderd



Figuur 6. Duur verwijderd

5 Conclusies

We hebben een generiek algoritme voor hiërarchische diagnose gegeven. Dit algoritme bevat twee potentieel kostbare operaties: testen van mogelijke diagnoses en discrimineren tussen diagnoses. We hebben twee klassen van hiërarchische modellen gedefinieerd om hiermee om te gaan:

- ID-hiërarchieën die testen overbodig maken.
- SD-hiërarchieën die er voor zorgen dat alleen relatief goedkope meetpunten worden gebruikt om een unieke diagnose te bepalen op een gegeven niveau.

We hebben algoritmes gegeven om deze hiërarchieën te creëren. Beide algoritmes hebben een lineaire tijdscomplexiteit. Helaas zijn ID-hiërarchieën niet noodzakelijk SD-hiërarchieën (of andersom), dus we kunnen de voordelen niet combineren. Voor een gegeven toepassing moeten we dus beslissen of een SD-hiërarchie danwel een ID-hiërarchie beter is. In sommige gevallen is dit makkelijk, zoals:

- De toepassing heeft veel niet-inverteerbare componenten. In deze gevallen leveren ID-hiërarchieën maar weinig op.
- Alle meetpunten in de toepassing hebben ongeveer gelijke kosten. In dit geval zijn SD-hiërarchieën minder goed bruikbaar.

In andere gevallen zal dit experimenteel moeten worden vastgesteld.

Referenties

R. Davis, 'Diagnostic reasoning based on structure and behaviour', *Artificial Intelligence*, Vol. 24, 1984, pp. 347-410.

M.R. Genesereth, 'The use of design descriptions in automatic diagnosis', *Artificial Intelligence*, Vol. 24, 1984, pp. 411-436.

F. Giunchiglia en T. Walsh, *A theory of abstraction*, Istituto per la Ricera Scientifica e Tecnologica, IRST - Technical Report # 9001-14, Trento, 1990.

E. Hammink en D.-J. Out, 'Hiërarchie bij model-gebaseerde diagnose', in *Proceedings NAIC*, 1990, pp. 189-198.

E. Hammink, D.-J. Out en R.R. Bakker, 'Controlled reasoning and hierarchy in model-based diagnosis', in *Proceedings of the 11th Conference on Expert Systems and their Applications*, Avignon, 1991, pp. 217-230.

W. Hamscher, *Model-based troubleshooting of digital systems*, MIT Press, Technical Report 1074, Cambridge MA, 1988.

J. de Kleer, 'Focusing on probable diagnoses', in *Proceedings of the 8th AAAI*, The MIT Press, Menlo Park, 1991, pp. 842-848.

J. de Kleer en B.C. Williams, 'Diagnosing multiple faults', *Artificial Intelligence*, Vol. 32, 1987, pp. 97-130.

C.A. Knoblock, J.D. Tenenberg en Q. Yang, 'A spectrum of abstraction hierarchies for planning', in *Working notes of the AGAA Workshop*, Boston, 1990, pp. 24-35.

A.N. Kumar en S.J. Upadhyaya, 'Automating representation', in *Working papers of the Second AAAI Workshop on Model Based Reasoning*, Boston, 1990, pp. 124-130.

R. Lbath, N. Giambiasi en C. Delorme, 'FRaHM: Framework for reasoning about hierarchical / multi-vues models', in *System fault diagnostics, reliability and related knowledge-based approaches*, Eds.: S. Tzafestas, M. Singh en G. Schmidt, Vol. 2, Reidel Publishing Company, 1987, pp. 29-41.

E.J. McCluskey en F.W. Clegg, 'Fault equivalence in combinational logic networks', in *IEEE Transactions on Computers C-20*, 1971, pp. 1286-1293.

I. Mozetic, *Hierarchical model-based diagnosis*, Austrian Research Institute for AI, TR-90-3, Wenen, 1990.

D.-J. Out en R.R. Bakker, *An analysis of model-based diagnosis using problem reformulation*, Universiteit Twente, Memoranda Informatica 92-03, 1992.

R.P. van Rikxoort, *Criteria voor clustering in model-gebaseerde diagnose*, Afstudeerverslag, Universiteit Twente, 1991.

D. Weld en S. Addanki, *Task-driven model abstraction*, University of Washington, 1990.