OM KONSTRUKSJON AV FEILTRÆR VED HJELP AV DATAMASKIN

AV

stud. techn. MORTEN ØSTBY

Hovedoppgave i statistikk

Linjen for teknisk fysikk

Almenavdelingen

Norges Tekniske Høgskole

Universitetet i Trondheim

TRONDHEIM 1979

ALMENAVDELINGEN
UNIVERSITETET I TRONDHEIM - NORGES TEKNISKE HØGSKOLE
7034 TRONDHEIM - NTH

HOVEDOPPGAVE

for

Stud.techn. Morten Østby,

Avd. VIIA, NTH.

Innleveringsfrist: 8. juni 1979.

PÅLITELIGHETSANALYSE: "Om konstruksjon av feiltrær ved hjelp av datamaskin."

Oppgaven utføres under veiledning av forsker Stein Bjørnar Jensen, SINTEF.

Trondheim, den 1. februar 1979.

Eivind Hiis Hauge
Avdelingsformann

Arnljot Høyland
Professor i Matematisk statistikk

FORORD

Undertegnede ønsket etter en prosjektoppgave ved Institutt for Industriell Økonomi og Organisasjon å ta hovedoppgave innen fagområdet risikoanalyse, pålitelighetsanalyse.

Det viste seg at de på SINTEF avdeling 18 hadde liggende endel programmer for automatisk konstruksjon av feiltrær som de var interessert i å få noen til å se nærmere på. Etter samtale med Torkell Gjerstad ved SINTEF avdeling 18 ble dette emnet valgt som tema for hovedoppgaven som er tatt ved Institutt for Matematisk Statistikk ved Almenavdelingen, Norges tekniske Høgskole.

Jeg vil i den forbindelse takke min ansvarlige faglærer, professor Arnljot Høyland og veileder Stein B. Jensen. Jeg vil spesielt takke Torkell Gjerstad for at han satte meg på oppgaven og for verdifull veiledning og hjelp i arbeidet med besvarelsen.

Trondheim, 6. juni 1979

Morten Østby

INNHOLD

SAMMENDRAG

Vi har i denne rapporten beskrevet og vurdert fire metoder for konstruksjon av feiltrær ved hjelp av datamaskin. De fire metodene er Synthetic Tree Model (STM), CAT-code, Lapp og Powers metode og RIKKE fra Risø.

Vi har ikke gått inn på metodenes programmer, men sett på algoritmene som ligger til grunn for programmene og vurdert deres evne til å konstruere korrekte feiltrær.

Rapporten innledes med en generell beskrivelse av feiltreanalyse, og deretter følger et kapittel om hver av metodene. Ved hjelp av algoritmene til CAT-code og Lapp og Powers metode har vi manuelt konstruert to feiltrær for et prosess-system.

CAT-code bygger i vesentlig grad på STM. Desisjonstabellene i CAT-code er ikke annet enn en mer hensiktsmessig måte å oppgi feiltransferfunksjonene i STM.

I STM er behandlingen av løkker mangelfull. CAT-code skal kunne takle både forover- og tilbakekoplinger, men som vi har vist, er ikke dette alltid tilfelle for foroverkoplinger.

STM er konstruert for elektriske systemer, mens CAT-code er utvidet til også å beskrive mekaniske komponenter. Felles for begge metodene er at de ikke tar hensyn til tidssekvenser og tidsforsinkelser i systemer. Programmene for STM og CAT-code er tilgjengelige.

De to andre metodene vi har undersøkt er ikke ferdig utviklet.

Lapp og Powers metode er konstruert for kjemiske prosesser og har nok sin styrke på dette området, selv om den skal kunne løse feiltrær også for andre typer systemer. Metoden har det særtrekk at den først konstruerer en modell ut fra systemets blokkdiagram og så konstruerer feiltreet ut fra modellen.

RIKKE er vel det mest ambisiøse systemet i undersøkelsen. Kanskje er algoritmene og modellene blitt for innviklede og omfattende, for metoden har ennå ikke vist seg anvendelig på virkelige systemer. RIKKE er basert på en interaktiv bruk som skal gjøre metoden så rask og effektiv at en kan følge opp forandringer i en konstruksjonsfase med stadig nye feiltrær.

Både Lapp og Powers metode og RIKKE skal kunne løse feiltrær for sekvensielle systemer og for systemer med tidsforsinkelse. Lapp og Powers metode er ikke tilgjengelig, mens RIKKE kan kjøpes av forskningsinstitusjoner.

Automatisk feiltrekonstruksjon vil aldri kunne oppheve de svakheter feiltreanalysen har som metode. Hendelsene i feiltreet blir fortsatt sett på som uavhengige. Komponentenes mulige tilstander er begrenset til normal funksjon og feil, og det er like vanskelig å oppdrive nok data om komponentene slik at en kan foreta en fornuftig kvantitativ evaluering av feiltreet. De fleste feiltrær konstruert ved de metodene vi har sett på, blir inkoherente.

Algoritmene som brukes i programmene er av vesentlig betydning for det endelige feiltreet. Men det er like viktig at komponentmodellene som brukes i konstruksjonen er av beste kvalitet.

Det bør legges mye arbeid i å konstruere korrekte og komplette komponentmodeller.

Automatisk konstruksjon av feiltrær er kommet for å bli, men vil nok aldri gjøre manuell konstruksjon overflødig. Det vil bli konstruert nye system med uforutsatte vanskeligheter som datamaskinene ikke kan takle. Da er det godt å kunne ty til den menneskelige hjerne.

Vi tror at feiltrekonstruksjon ved hjelp av datamaskin kan bli et nyttig supplement til konvensjonell feiltrekonstruksjon. Datamaskinene er hurtige, og ved hjelp av dem kan konstruktøren lage feiltrær for ulike topphendelser og bruke feiltrærne til å få bedre innsikt i systemet. Eller han kan sette datamaskinen til å gjøre den rutinemessige delen av konstruksjonen og få frigjort tid til å konsentrere seg om de vanskeligere sidene ved systemet han analyserer.

1. INNLEDNING

Vår tilværelse blir stadig mer avhengig av tekniske hjelpemidler. Samtidig vokser de teknologiske systemene i størrelse og kompleksitet slik at det blir vanskeligere å forhindre uhell og ulykker. Når ulykkene også får konsekvenser for flere og flere mennesker, er det naturlig at søkelyset i den senere tid i større grad enn tidligere er blitt rettet mot sikkerheten og påliteligheten til systemer som for eksempel kjernekraftverk. Det har ført til at mer arbeid og penger legges ned i analyser av risiko og sikkerhet.

For å kunne vurdere påliteligheten til et system, må en kjenne de mulige feilemåtene til systemet og deres konsekvenser. Videre må sannsynligheten for at de forskjellige feilene skal inntreffe være kjent.

Feiltreanalysen er en metode som på en logisk måte beskriver de mulige årsakene til en feil. Det er et tidkrevende arbeid å produsere et feiltre for hånd. Derfor er det gjort flere forsøk på å automatisere prosessen.

Vi har i denne rapporten sett på fire metoder for automatisk konstruksjon av feiltrær. Hensikten har vært å beskrive metodene og deres evne til å produsere feiltrær. Vi har ikke testet selve programmene, men sett på algoritmene og undersøkt hvilke problemer metodene takler og hvilke de ikke takler.

Det har vært naturlig å dele rapporten i fem deler: et innledende kapittel om feiltreanalyse og et kapittel for hver av metodene. De fem kapitlene kan gjerne leses uavhengig av hverandre.

2. FEILTREANALYSE

2.1 Generell beskrivelse

Feiltreanalyse (FTA) er en systematisk metode som anvendes i analyser av tekniske systemers pålitelighet.

Tidlig på 60-tallet sto flyindustrien overfor så store og kompliserte systemer at de eksisterende analysemetoder ikke var gode nok. En ny metodikk var påkrevet. Begrepet FTA ble lansert ved Bell Telephone Laboratory /1/ som et hjelpemiddel i studiene av sikkerheten ved et kontrollsystem for rakett-oppskyting. Prinsippet viste seg å være fruktbart. Senere har FTA blitt forbedret og utvidet og er idag i utstrakt bruk i såvel flyindustri som i kjernekraftindustri, se Haasl /2/ og Rasmussenrapporten /3/.

Systemtanken er sentral i FTA. Christiansen /4/ sier:

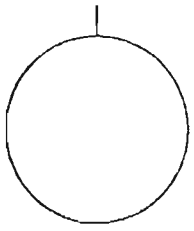
> "Et system er en samling av deler eller komponenter som er sammenkoplet, og som samvirker på en organisert måte".

Ideen bak FTA er å omforme et fysisk system til et logisk diagram eller feiltre som beskriver hvordan en spesifisert systemfeil, kalt topphendelsen, kan oppstå på grunn av feil i systemets komponenter. En topphendelse kan være en reaktornedsmelting eller en mer triviell hendelse som at vannet ikke kommer i vannkranen.

Det at en feil oppstår, er en uønsket hendelse. Feil og hendelser blir i det følgende brukt synonymt.

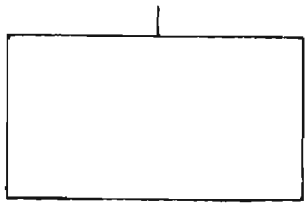
De mest grunnleggende definisjoner og symboler i FTA er vist på figur 2.1.

PRIMÆRHENDELSE/-FEIL



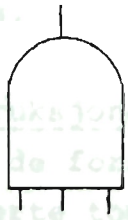
Uønsket hendelse eller feil i en komponent som representerer det ønskede detaljeringsnivå for FTA. Feilsannsynligheten er kjent eller kan bestemmes deterministisk.

SAMMENSATT HENDELSE



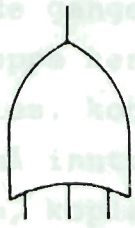
Hendelse ved utgangen av en logisk port. Hendelsen er beskrevet inne i rektangelet, kan f.eks. være topphendelsen.

LOGISK OG-PORT



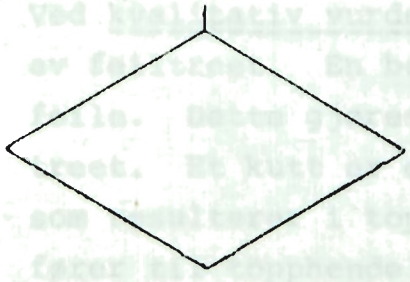
Utgangshendelsen A inntreffer bare hvis samtlige inngangshendelser inntreffer.

LOGISK ELLER-PORT



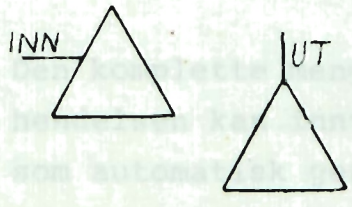
Utgangshendelsen B inntreffer hvis minst en av inngangshendelsene inntreffer.

IKKE UTVIKLET HENDELSE



Utviklingen av hendelsen ligger utenfor oppgaven eller er ikke mulig.

OVERFØRINGS SYMBOLER



Trianglene markerer overføringer fra et sted i feiltreet til et annet.

Figur 2.1 Feiltresymboler.

FTA kan sies å bestå av følgende trinn, Fussell /5/:

1) Systemdefinisjon
2) Konstruksjon av feiltre
3) Kvalitativ vurdering
4) Kvalitative beregninger

Systemdefinisjonen består i å formulere problemet. Systemets begrensninger og komponenter samt komponentenes forbindelseslinjer og begynnelsestilstander skal defineres. I tillegg må komponentenes måter å feile på bestemmes, og en topphendelse velges.

Konstruksjonen av feiltreet går ut på å kartlegge på hvilke måter de forskjellige komponentene kan medvirke til at den definerte topphendelsen inntreffer. En tar utgangspunkt i topphendelsen og gjennomfører en deduktiv analyse ved gjentagende ganger å stille spørsmålet: "Hvordan kan dette skje?" En gruppe hendelser som hver for seg fører til hendelsen som utvikles, koples sammen gjennom en ELLER-port. Hendelser som alle må inntreffe for at den interessante hendelsen skal inntreffe, koples sammen ved hjelp av en OG-port.

Ved kvalitativ vurdering får en innsikt i systemet ved hjelp av feiltreet. En beregner på hvilke ulike måter systemet kan feile. Dette gjøres ved å finne de minimale kuttene til feiltreet. Et kutt er en hendelse eller en kombinasjon av hendelser som resulterer i topphendelsen. Et minimalt kutt er et kutt som fører til topphendelsen hvis og bare hvis alle hendelsene i kuttet inntreffer.

Den komplette mengde av minimale kutt beskriver alle måter topphendelsen kan inntreffe på. Det er konstruert datamaskinprogram som automatisk genererer de minimale kuttene til et feiltre.

Kvantitative beregninger av sannsynligheten for topphendelsen gjøres på grunnlag av sannsynlighetsdata for primærfeilene. Et datamaskinprogram, som KITT /6/, regner ut sannsynligheten for topphendelsen på bakgrunn av sviktintensitet og midlere reparasjonstid for komponentene og de minimale kuttene. Den pålitelighetsmessige betydningen av de enkelte komponentene kan bestemmes, se Barlow og Proschan /7,s.26/. De kvantitative beregningene er til stor hjelp når en skal bestemme hvor det skal settes inn ressurser for å forbedre systempåliteligheten.

FTA kan være en enkel eller sofistikert analyse alt etter hvilke behov den skal tilfredsstille.

Et krav må imidlertid være at feiltreet inneholder alle hendelser som gir et signifikant bidrag til sannsynligheten for topphendelsen.

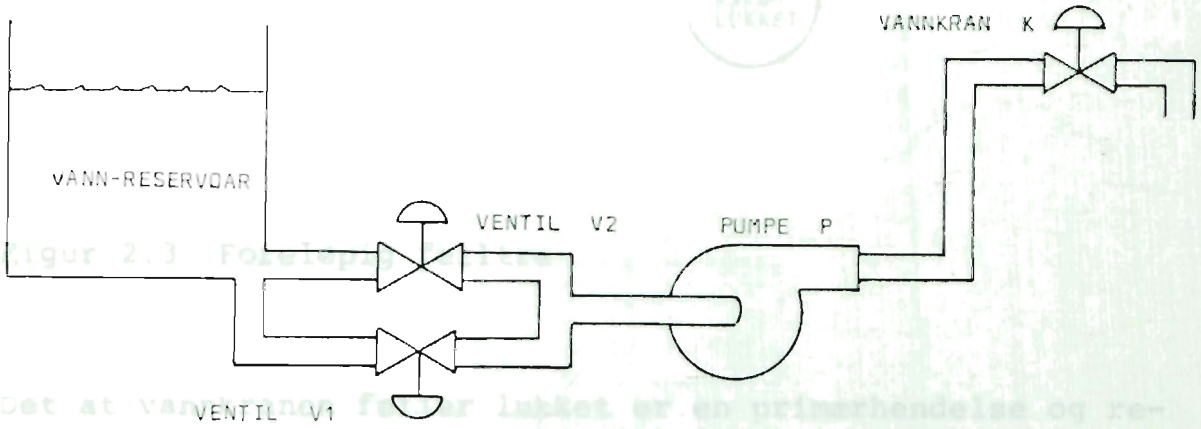
En svakhet ved FTA er at de enkelte primærhendelsene antas å være uavhengige. Avhengighet innføres når /8,s.34/

1) Flere feil har samme årsak (felles kraftforsyning).
2) Komponentene bærer en felles belastning. Feiler en komponent blir de andre komponentene utsatt for større belastning.
3) Komponentene påvirker hverandre via det interne miljø (trykk, temperatur).
4) Komponentene har gjensidig utelukkende feilmodi (feiler åpnen/feiler lukket).
5) Vedlikehold medfører at tidspunktet da en komponent feiler avhenger av de øvrige komponentenes tilstand.

Vi kjenner ikke til metoder som løser problemet med avhengige komponenter. En må være oppmerksom på problemet og arbeide bevist for å redusere avhengighet.

2.2 Feiltrekonstruksjon

Ved beskrivelse av feiltrekonstruksjon er det naturlig å ta utgangspunkt i et enkelt eksempel. Systemet på figur 2.2 skal bringe vann fra et reservoar til vår vannkran.



Figur 2.2 Vanntilførselssystem.

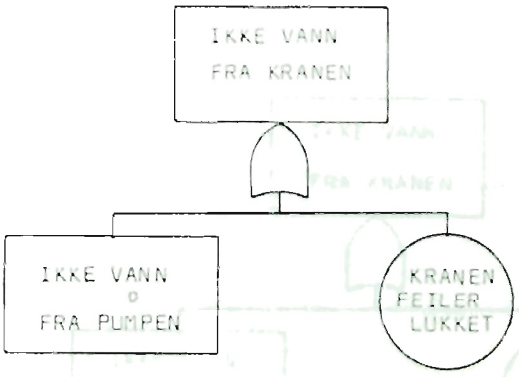
Systemet består av to pneumatiske ventiler, V1 og V2, en manuell vannkran K og en elektrisk pumpe P, bundet sammen med et rørsystem. For enkelhets skyld ser vi bort i fra at:

- det ikke er vann i reservoaret
- kommandosignal (for åpning/lukking av ventiler og start/stopp av pumpe) uteblir
- pneumatisk trykk og elektrisk kraft mangler
- rørbrudd eller lekkasje.

En interessant topphendelse er da "ikke vann ut av kranen".

Dette kan ha en av to årsaker: enten er vannkranen istykker eller så kommer det ikke vann fra pumpa.

Hver av hendelsene vil resultere i topphendelsen, og feiltreet på figur 2.3 oppstår ved bruk av en ELLER-port.



Figur 2.3 Foreløpig feiltre.

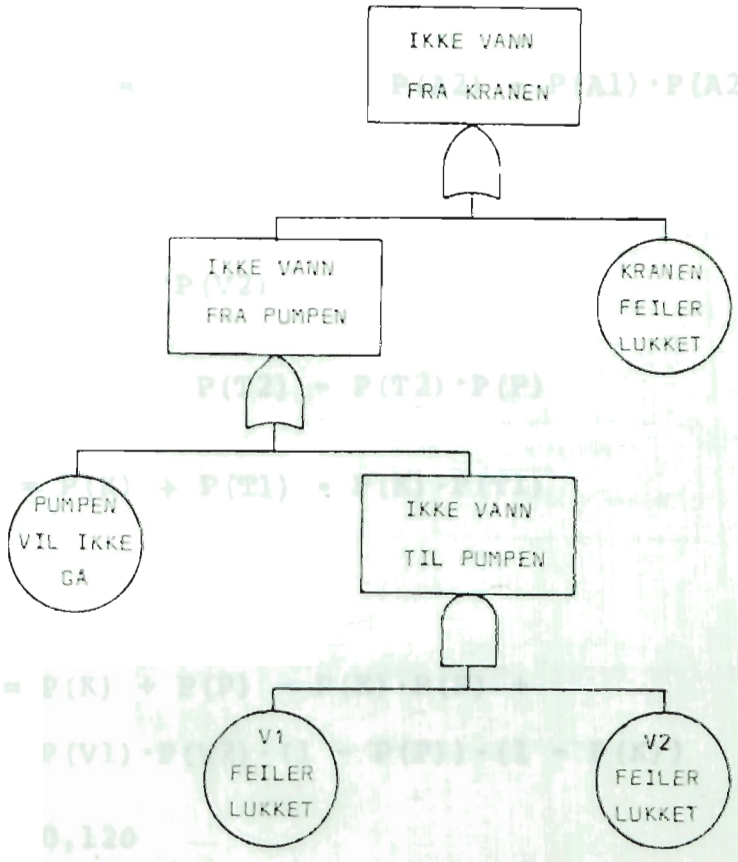
Det at vannkranen feiler lukket er en primærhendelse og representeres med en sirkel, mens hendelsen "ikke vann fra pumpa" må utvikles videre og representeres med et rektangel.

Hvis det ikke kommer vann fra pumpa , må enten pumpa ha feilet eller så kommer det ikke vann til pumpa . Pumpefeil er en ny primærhendelse mens hendelsen "ikke vann til pumpa" skyldes at begge ventilene V1 og V2 er lukket. Disse to primærfeilene settes sammen med en OG-port, og feiltreet er ferdigkonstruert, se figur 2.4.

Da feiltreet bare består av primærhendelser, som kan sies å være uavhengige, kan sannsynligheten for topphendelsen beregnes når sannsynligheten for primærhendelsene er kjent.

De minimale kuttene til feiltreet er:

1. Kranen K feiler lukket.
2. Pumpa P vil ikke gå.
3. Ventil V1 feiler lukket og ventil V2 feiler lukket.



Figur 2.4 Feiltre for systemet på figur 2.2.

Feil i pumpa eller kranen vil alene føre til topphendelsen, mens begge ventilene må feile for at topphendelsen skal inntreffe. Dette indikerer at pumpa og kranen er de komponentene vi bør konsentrere oss om å forbedre påliteligheten til.

Vi anslår sannsynligheten for feil i de enkelte komponentene til å være:

P (Kranen feiler lukket) = P(K) = 0,1
P (Pumpa vil ikke gå) = P(P) = 0,02
P (Ventil V2 feiler lukket) = P(V1) =
P (Ventil V2 feiler lukket) = P(V2) = 0,05

For to uavhengige hendelser A1 og A2 gjelder:

$$P(A1 \text{ OG } A2) = P(A1) \cdot P(A2)$$

$$P(A1 \text{ ELLER } A2) = P(A1) + P(A2) - P(A1) \cdot P(A2)$$

Vi får da

$$P(T2) = P(V1) \cdot P(V2)$$

$$P(T1) = P(P) + P(T2) - P(T2) \cdot P(P)$$

$$P(T) = P(K) + P(T1) - P(K) \cdot P(T1)$$

Og videre

$$P(T) = P(K) + P(P) - P(K) \cdot P(P) + P(V1) \cdot P(V2) \cdot (1 - P(P)) \cdot (1 - P(K))$$

$$= 0,120$$

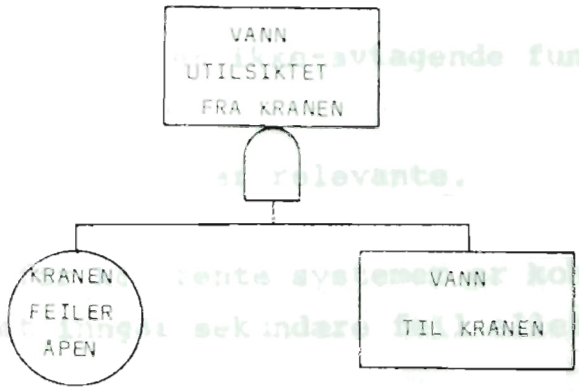
La oss si at vi ønsker å forbedre påliteligheten til en av komponenttypene med 20%. Da blir sannsynlighetene for feil henholdsvis: $P(K) = 0,08$, $P(P) = 0,016$ og $P(V1) = P(V2) = 0,04$.

Sannsynligheten for topphendelsen blir:

- med forbedret kran : $P(T) = 0,101$
- med forbedret pumpe : $P(T) = 0,117$
- med forbedret ventil: $P(T) = 0,119$

Det er ingen tvil om at det lønner seg mest å forbedre kranen.

Ved valg av en annen topphendelse som "vann utilsiktet fra kranen", fås et helt annet feiltre. Feilene i dette treet blir av en annen type, se figur 2.5. Det må derfor klart defineres hvordan komponentene feiler.



Figur 2.5 Feiltre for topphendelsen "vann utilsiktet fra kranen".

2.3 Koherente og sekvensielle feiltrær

De to begrepene er sentrale i FTA og skal her forklares nærmere.

Hver primærfeil i et feiltre kan representeres ved en lineær indikatorvariabel y_i:

$$y_i = \begin{cases} 1 \text{ hvis i'te primærhendelse har inntruffet} \\ 0 \text{ ellers} \end{cases}$$

for i = 1, 2,...,n, der n er antall primærhendelser i feiltreet.

La topphendelsen være representert ved den binære indikatorvariable ψ.

$$\psi(y_1, y_2, \ldots, y_n) = \begin{cases} 1 \text{ hvis topphendelsen har inntruffet} \\ 0 \text{ ellers.} \end{cases}$$

ψ kalles feiltreets strukturfunksjon.

Et feiltre er koherent hvis:

1. $\psi(y_1, y_2, \ldots, y_n)$ er en ikke-avtagende funksjon av y_i $\quad i = 1, \ldots, n$.
2. Alle primærhendelser er relevante.

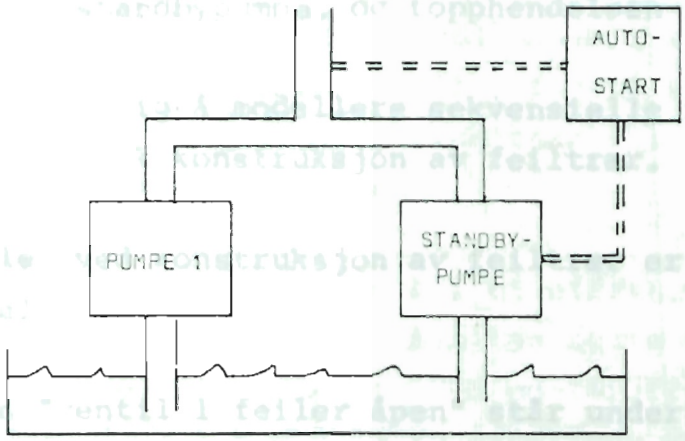
De fleste feiltrær for koherente systemer er koherente. Unntak er feiltrær hvor det inngår sekundære feil eller normale tilstander.

En sekundærfeil er en feil som skyldes ekstreme belastninger på grunn av miljø eller bruksmåte. Feiltrær med sekundære feil er inkoherente fordi den feilende komponenten ikke blir reparert når grunnen til feilen fjernes.

Den sammensatte hendelsen "vann til kranen" på figur 2.5 er et eksempel på en normal handelse. For at topphendelsen skal inntreffe, må vi altså ha både en normalhendelse og en feil. Hvis det skjer en feil i systemet slik at det ikke lenger kommer vann til kranen, vil topphendelsen ikke inntreffe. Det er i strid med punkt 1 i definisjonen av koherente systemer.

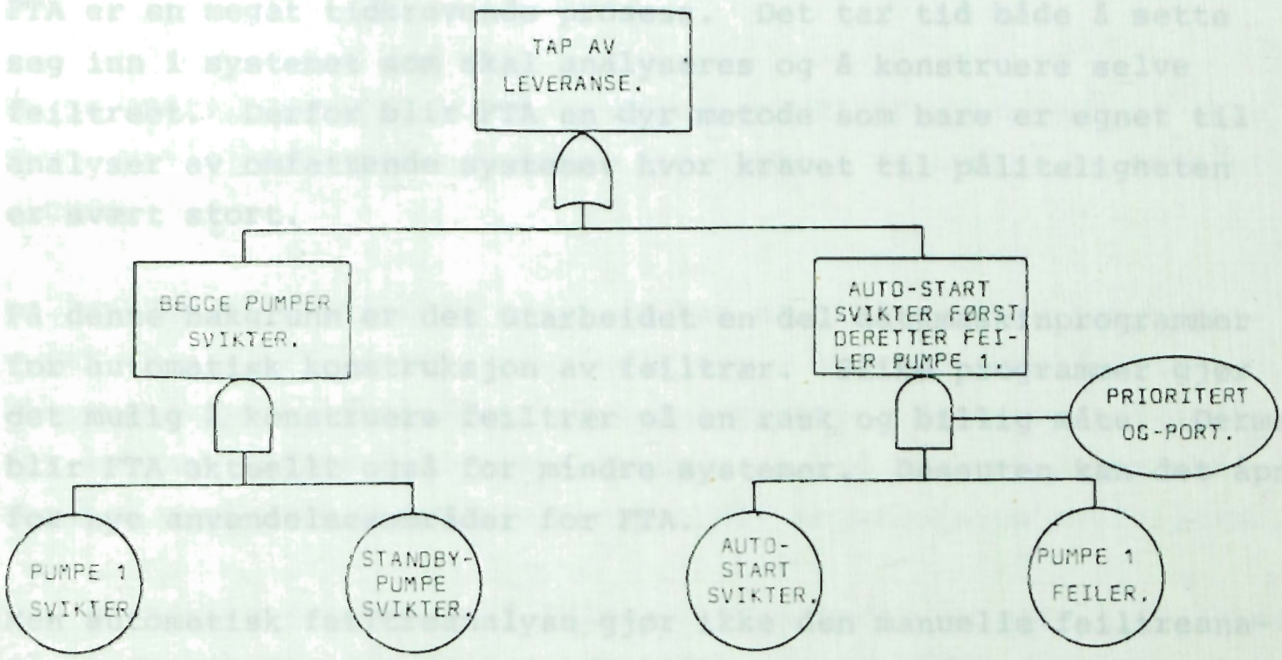
Sekvensielle feiltrær oppstår når det ikke er likegyldig hvilken rekkefølge komponenter feiler i.

På figur 2.6 er vist et system som består av to pumper koplet i parallell. Den ene pumpa er en såkalt standbypumpe som startes av en autostart hvis den andre pumpa feiler.



Figur 2.6 Pumpesystem.

Feiltreet for topphendelsen "ikke leveranse av væske" er vist på figur 2.7.



Figur 2.7 Feiltre for topphendelsen "ikke leveranse av væske".

Hvis autostarten svikter etter at pumpe 1 har feilet, har den allerede startet standbypumpa, og topphendelsen inntreffer ikke.

Det kan være vanskelig å modellere sekvensielle feilhendelser, særlig ved automatisk konstruksjon av feiltrær.

Et annet problem ved konstruksjon av feiltrær er gjensidig utelukkende hendelser.

Hvis hendelsen "ventil 1 feiler åpen" står under en OG-port, kan ikke hendelsen "ventil 1 feiler lukket" stå under den samme OG-porten. Ventilen kan jo ikke feile både åpen og lukket på en gang. Dette problemet dukker opp ved forover- og tilbakekoplinger og må tas spesielt hensyn til ved automatisk konstruksjon av feiltrær.

2.4 Automatisk konstruksjon av feiltrær

FTA er en meget tidkrevende prosess. Det tar tid både å sette seg inn i systemet som skal analyseres og å konstruere selve feiltreet. Derfor blir FTA en dyr metode som bare er egnet til analyser av omfattende systemer hvor kravet til påliteligheten er svært stort.

På denne bakgrunn er det utarbeidet en del datamaskinprogrammer for automatisk konstruksjon av feiltrær. Slike programmer gjør det mulig å konstruere feiltrær på en rask og billig måte. Dermed blir FTA aktuellt også for mindre systemer. Dessuten kan det åpne for nye anvendelsesområder for FTA.

Men automatisk feiltreanalyse gjør ikke den manuelle feiltreanalysen overflødig. De metodene som eksisterer idag, er ikke så gode at de til enhver tid vil konstruere korrekte feiltrær. Med

korrekte feiltrær menes her feiltrær hvor alle signifikante feilemåter for systemet er med på en logisk riktig måte. I tillegg er en av de største fordelene ved manuell konstruksjon av feiltrær at konstruktøren tvinges til å forstå systemet han arbeider med. Mange av systemets svakheter vil kunne utbedres mens feiltreet er under utvikling.

Men automatisk feiltreanalyse kan bli et effektivt hjelpemiddel for konstruktøren i hans arbeid. Han kan sette datamaskinen til å gjøre den rutinemessige delen av FTA og får frigitt tid til å konsentrere seg om de mer kompliserte aspektene ved systemets feilemåter.

FTA er en relativt ny form for systembeskrivelse som ikke har funnet sin endelige form. De eksisterende metodene for automatisk konstruksjon av feiltrær er ferske,og flere av dem er ikke ferdigutviklet ennå.

To av metodene vi ser på, "CAT-code" og "Synthetic Tree Model", er relativt like, og det arbeides oss bekjent ikke med å forbedre dem.

De to andre, "Lapp og Powers metode" og "RIKKE systemet", er ikke ferdigutviklet og vil nok om en tid komme i forbedret og utvidet utgave.

Felles for alle metodene er at de har et bibliotek av komponenttyper (ventiler, pumper, sikringer osv.) hvor de forskjellige komponenttypenes feilemåter er katalogisert. Dette er nødvendig for å kunne utføre feiltrekonstruksjonen ved bruk av datamaskin, men kan også være til hjelp ved manuell konstruksjon av feiltrær.

Det er vanlig i metodene å følge en gren under en port i feiltreet så langt bakover som mulig, før andre grener under den samme porten utvikles.

3 CAT-CODE

3.1 Innledning

Datamaskinprogrammet CAT-code /9/ er laget ved University of California for å forbedre beregningsmetodene i kostnad/nytte-analyser av kjernekraftanlegg. Metoden hevdes å ha anvendbarhet på alle typer teknologiske systemer hvor også mennesket kan inngå.

For å kunne konstruere et feiltre til et generelt system, trenger en informasjon om de enkelte komponentene i systemet og om hvordan de er knyttet sammen. Ethvert punkt i systemet hvor en utgang fra en komponent er forbundet med inngangene til en eller flere etterfølgende komponenter, kalles i CAT-code et *knutepunkt*. En restriksjon i denne forbindelse er at innganger kan kobles i parallell, mens utganger ikke kan, se /9, s.80-83/.

En *systemtilstand* er en spesifikk tilstand for systemet, definert enten ved et knutepunkt eller som en intern funksjonsmåte for en komponent.

3.2 Desisjonstabeller

I CAT-code er komponentene beskrevet ved hjelp av *desisjonstabeller*. En desisjonstabell viser alle mulige utgangshendelser fra en komponent som funksjon av de mulige inngangshendelsene og funksjons- og feilemåtene til komponenten. I tabell 3.1 er vist en desisjonstabell for en pumpe. Desisjonstabellen definerer en komponent*type*. Andre pumper kan virke anderledes og er da av en annen type.

Tabell 3.1

DESISJONSTABELL FOR PUMPE

Rekke	Inngang 1 Hovedstrøm	Inngang 2 Kraft	Intern funksjonsmåte	Utgang
1	0	-1	-1	0
2	-1	0	-1	0
3	-1	-1	5	0
4	-1	-1	3003	0
5	1	1	0	1

Signaler: -1 : likegyldig
0 : ikke signal
1 : signal normalt

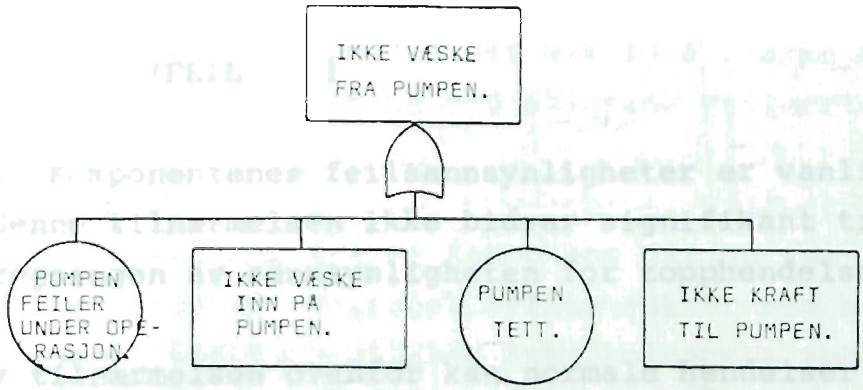
Intern funksjon: -1 : likegyldig
0 : normalt (OK)
5 : feil under operasjon
3003 : tett

Rekke en i tabell 3.1 sier at når det ikke er væske inn på pumpa, kommer det ikke væske ut, uansett om pumpa virker eller ikke og om den får kraft.

De likegyldige tilstandene medfører en betydelig forenkling i feiltrekonstruksjonen. Når en feil forfølges bakover i systemet ved hjelp av desisjonstabeller kan en se bort fra kolonnene representert med minus en nettopp fordi de er likegyldige for utgangshendelsen. For eksempel kan "ikke væske fra pumpa" være en topphendelse eller en lavere hendelse vi ønsker å utvikle.

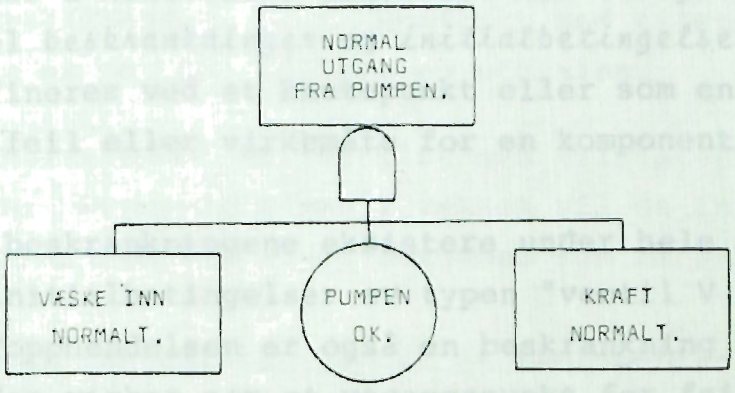
Av tabell 3.1 sees at rekke en til fire alle fører til denne utgangshendelsen. Disse rekkene koples da sammen med en ELLER-port.

Videre kan enhver rekke i en desisjonstabell representeres ved en OG-port med antall innganger lik summen av inngangs- og internfunksjonskolonnene minus likegyldige (-1) tilstander. Det vil i dette tilfelle si at vi får feiltreet på figur 3.1 da rekke en til fire bare inneholder en ikke-likegyldig tilstand hver.



Figur 3.1 Minifeiltre for hendelsen "ikke væske fra pumpa".

Utgangshendelsen "normal utgang fra pumpa" er bare tilfredsstilt av rekke fem, som ikke har noen likegyldige tilstander. Dermed oppstår feiltreet på figur 3.2.



Figur 3.2 Minifeiltre for hendelsen "normal utgang fra pumpa".

Det framgår av feiltreet på figur 3.2 at en ved bruk av CAT-code får med normale hendelser i feiltreet som da blir inkoherent. Ikke alle kvantitative FTA metoder kan løse slike inkoherente feiltrær.

Sannsynligheten for at en komponent feiler er som regel liten, slik at approksimasjonen:

$$P(OK) = 1 - P(FEIL) \simeq 1$$

er brukbar. Komponentenes feilsannsynligheter er vanligvis så usikre at denne tilnærmelsen ikke bidrar signifikant til usikkerheten i beregningen av sannsynligheten for topphendelsen.

Ved bruk av tilnærmelsen ovenfor kan normale hendelser utelates fra feiltreet, og dette blir koherent.

Desisjonstabeller er nærmere beskrevet i /9/ og i /10/, hvor det er gjengitt desisjonstabeller for endel komponenter.

3.3 Redigering

Før en begynner å konstruere feiltreet til et system, må en definere endel *beskrankninger* og *initialbetingelser* for systemet. Disse kan defineres ved et knutepunkt eller som en allerede eksisterende feil eller virkemåte for en komponent.

Generelt vil beskrankningene eksistere under hele feiltrekonstruksjonen, men initialbetingelser av typen "ventil V er lukket" kan forandres. Topphendelsen er også en beskrankning med den tilleggsfunksjon at den virker som et utgangspunkt for feiltreanalysen.

Redigeringen består av tre deler:

1. Redigering i forbindelse med konstruksjon av en port.
2. Videregående redigering etter at hver port eller gruppe porter er ferdig konstruert.
3. Sluttredigering etter at feiltreet er konstruert.

Ved utviklingen av en hendelse er det viktig å sjekke om hendelsen (systemtilstanden) er konsistent med allerede definerte systemtilstander.

Ved et gitt utgangssignal fra en komponent går en inn i desisjonstabellen for komponenten og finner hvilke rekker som har den ønskede utgangen. Deretter sjekkes hver kolonne i rekkene med spørsmålet: "Hvordan stemmer den definerte tilstanden overens med systemtilstanden definert tidligere?.

Det er tre mulige svar på spørsmålet:

1. Systemtilstanden er ikke definert før. Hvis ikke andre kolonne i rekken motsier allerede eksisterende tilstander, godtas rekken , og rekkens tilstandsdefinisjoner blir gjort gjeldende for systemet.
2. Systemtilstanden er i konflikt med tilstanden i rekken. Rekken er da ikke tillatt og utelates.
3. Systemtilstanden er identisk med tilstanden beskrevet i rekken. Denne kolonnen i rekken vil da inntreffe med sannsynlighet lik en og betegnes som en *sikker hendelse*.

Ved redigeringen tas det hensyn til at knutepunkt- og komponenttilstandene i en gren under en ELLER-port ikke har betydning for tilstandene i de andre grenene under porten. Hendelsene i en gren under en OG-port må derimot ikke ha gjensidig utelukkende hendelser i en annen gren under den samme OG-porten.

Hvis en sikker hendelse inntreffer under:

1. en OG-port med flere innganger, fjernes hendelsen fra feiltreet.

2. en ELLER-port eller en OG-port med en inngang, fjernes porten og i tillegg alle ELLER-porter og singelinngangs OG-porter over, opp til den laveste OG-porten med flere innganger.

Hvis en hendelse som ikke kan inntreffe står under:

1. en ELLER-port med mer enn en inngang, fjernes hendelsen.

2. en OG-port eller en ELLER-port med en inngang, fjernes porten og alle OG-porter og ELLER-porter med en inngang opp til den laveste ELLER-porten med flere innganger.

Disse reglene brukes til å behandle tilbakekoplingssløyfer. Utgangspunktet er en hendelse et sted i løkken. Når en så går bakover for å finne årsaken til hendelsen og kommer tilbake til utgangspunktet, har en en tilbake-koplingssløyfe. Hvis tilstandene beskrevet i desisjonstabellen, som brukes på det tidspunktet, er i overensstemmelse med systemtilstandene definert i starten, er sløyfen konsistent, og sjekkingen opphører med en sikker hendelse.

Motsier derimot tilstandene hverandre, må en sjekke om andre rekker i desisjonstabellen passer bedre. Hvis ingen rekker kan brukes, er det ikke mulig å gjøre løkken ferdig, og starthendelsen kan ikke inntreffe.

Foroverkoplinger er ikke beskrevet i CAT-code. Slike koplinger kan, som vi skal vise i et senere eksempel, skape problemer.

Den videregående redigeringen beskrives kort i tre punkter:

1. Fjerne porter med bare en inngang.

2. Sjekke om primærhendelser under ferdiggjorte OG-porter er inkonsistente eller overflødige.

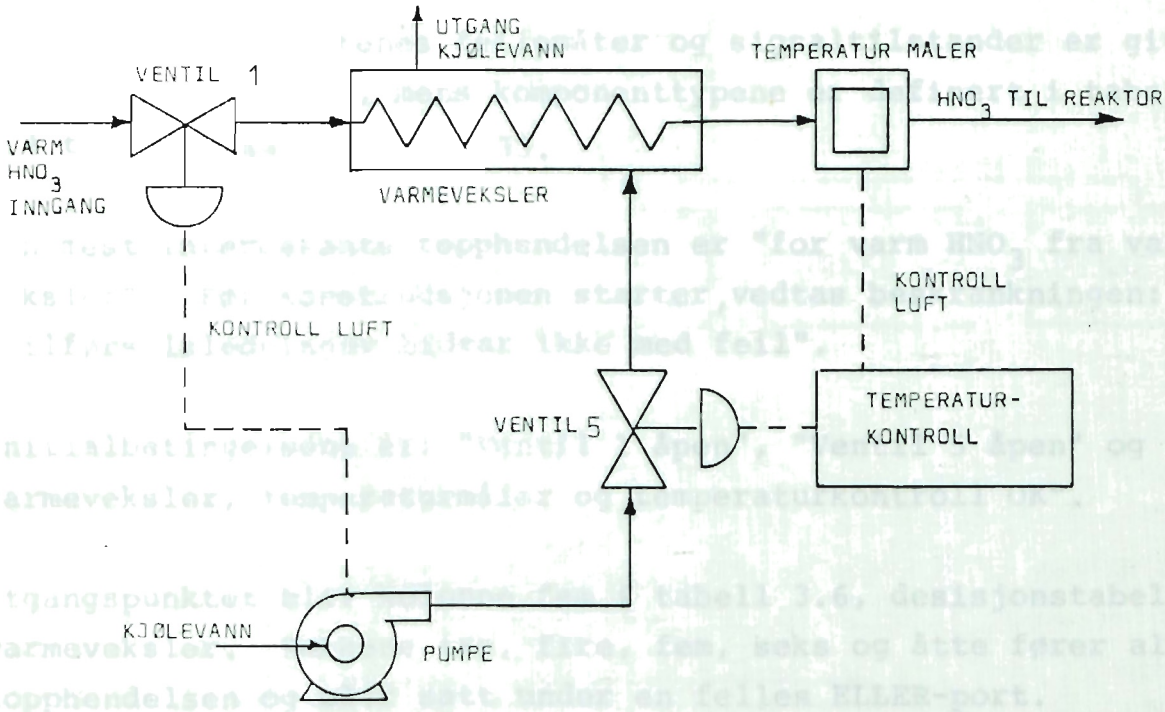
3. Sjekke om primærhendelser under ferdiggjorte ELLER-porter skaper overflødige OG-porter. Er en primærhendelse koblet direkte både til en ELLER- port og til en OG-port under ELLER-porten, er OG-porten overflødig og kan utelates. Dette følger av definisjonen av minimale kutt.

Når feiltreet er ferdig, fjernes alle sub-trær som er like og erstattes med nummererte overføringssymboler slik at hvert subtre bare vises en gang. Videre kan om ønskelig alle hendelser som beskriver komponenter som er i orden, utelates, og feiltreet er klart for videre analyse. Flytskjema for CAT-code er vist på figur 3.13, side 34.

3.4 Eksempel på anvendelse

I /9/ er feiltrekonstruksjon ved bruk av CAT-code vist for et elektrisk kontrollsystem for en trykktank og for et system som fjerner varme fra en kjernereaktor. Det kan være interessant å se hvordan CAT-code lager feiltrær for en annen type systemer, som for eksempel et prosesskontrollsystem. Lapp og Powers /11,s.3/ har beskrevet et kjølesystem for salpetersyre, HNO_3, se figur 3.3.

En varmeveksler får varm HNO_3 gjennom en ventil som har til oppgave å stoppe syra hvis kjølevannet blir borte. Den avkjølte salpetersyra blir levert til en reaktor som kan eksplodere hvis syra er for varm. Før syra går inn på reaktoren blir temperaturen målt, og en temperaturkontroll regulerer kjølevannsventilen slik at syra får riktig temperatur. Kjølevannet pumpes opp fra en ekstern kilde.



Figur 3.3 Kjølesystem for salpetersyre /11,s.3/.

Moderate forandringer i syras trykk eller temperatur inn på varmeveksleren reguleres ved hjelp av tilbakekoplingssløyfa.

Foroverkoplinen fra pumpa til ventil 1 virker slik at når pumpa stanser, blir det trykk inn på kontrollinngangen for ventil 1 som dermed lukker.

Systemet består av seks komponenttyper. De to ventilene har forskjellige desisjonstabeller da de har ulike virkemåter. Ventil 1 trenger trykk for å lukke, mens ventil 5 åpner ved økende trykk.

Desisjonstabellene for pumpe og manuelle ventiler som står i /10/ må justeres for å kunne brukes i denne analysen. Desisjonstabeller for de øvrige komponentene har vi konstruert selv. Det viste seg å være vanskeligere enn først antatt. Tabellene er ikke komplette.

Blant annet er rekker med mer enn en ikke-normal tilstand på innganger eller som interne funksjonsmåter utelatt.

Koder for komponentenes feilemåter og signaltilstander er gitt i tabell 3.2 og 3.3, mens komponenttypene er definert i tabell 3.4 til 3.9, se side 30 - 33.

Den mest interessante topphendelsen er "for varm HNO_3 fra varmeveksler". Før konstruksjonen starter, vedtas beskrankningen: "tilførselsledninger bidrar ikke med feil".

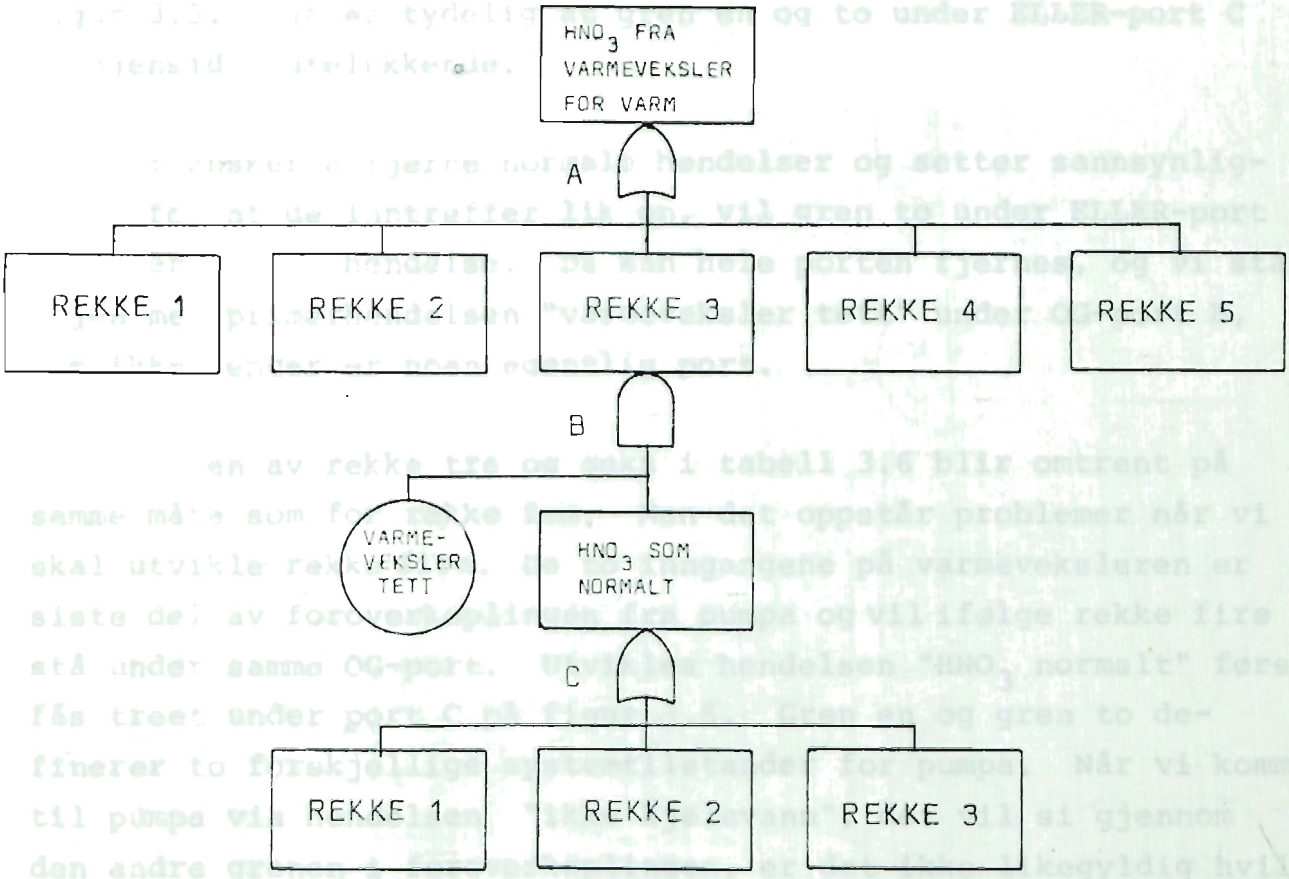
Initialbetingelsene er: "Ventil 1 åpen", "Ventil 5 åpen" og "pumpe, varmeveksler, temperaturmåler og temperaturkontroll OK".

Utgangspunktet blir kolonne fem i tabell 3.6, desisjonstabell for varmeveksler. Rekkene tre, fire, fem, seks og åtte fører alle til topphendelsen og blir satt under en felles ELLER-port.

I rekke fem blir primærhendelsen "varmeveksler tett" og den sammensatte hendelsen "HNO_3 normalt" koplet sammen ved hjelp av en OG-port. Syra kommer fra ventil 1, tabell 3.5, hvor rekke fem til sju tilfredsstiller utgangshendelsen "HNO_3 normalt". Disse rekkene blir satt under en ny ELLER-port, og vi har feiltreet på figur 3.4.

Vi utvikler ELLER-port C på figur 3.4 ferdig før vi begynner på en ny rekke under ELLER-port A.

Rekke fem består av primærfeilen "ventil 1 omvendt virkemåte", "HNO_3 normalt", som ikke utvikles videre, og hendelsen "kontrolltrykk på ventil 1". Den sistnevnte hendelsen søkes tilbake til pumpen, tabell 3.7, hvor rekkene en til fem har den rette utgangen.



Figur 3.4 Foreløpig feiltre for HNO_3 kjølesystem

Disse rekkene består bare av primærhendelser, likegyldige tilstander og hendelser som ikke utvikles videre.

Når vi nå går tilbake til ventil 1 for å utvikle rekke seks, er tilstanden for pumpa igjen udefinert fordi rekkene står under en ELLER-port.

Rekke seks i tabell 3.5 sier at vi får normal HNO_3 strøm ut når HNO_3 inn er normal, ventil 1 funksjonerer normalt og det ikke er kontrolltrykk på ventil 1. Dette er bare tilfelle når pumpa fungerer normalt, og den får vann og kraft.

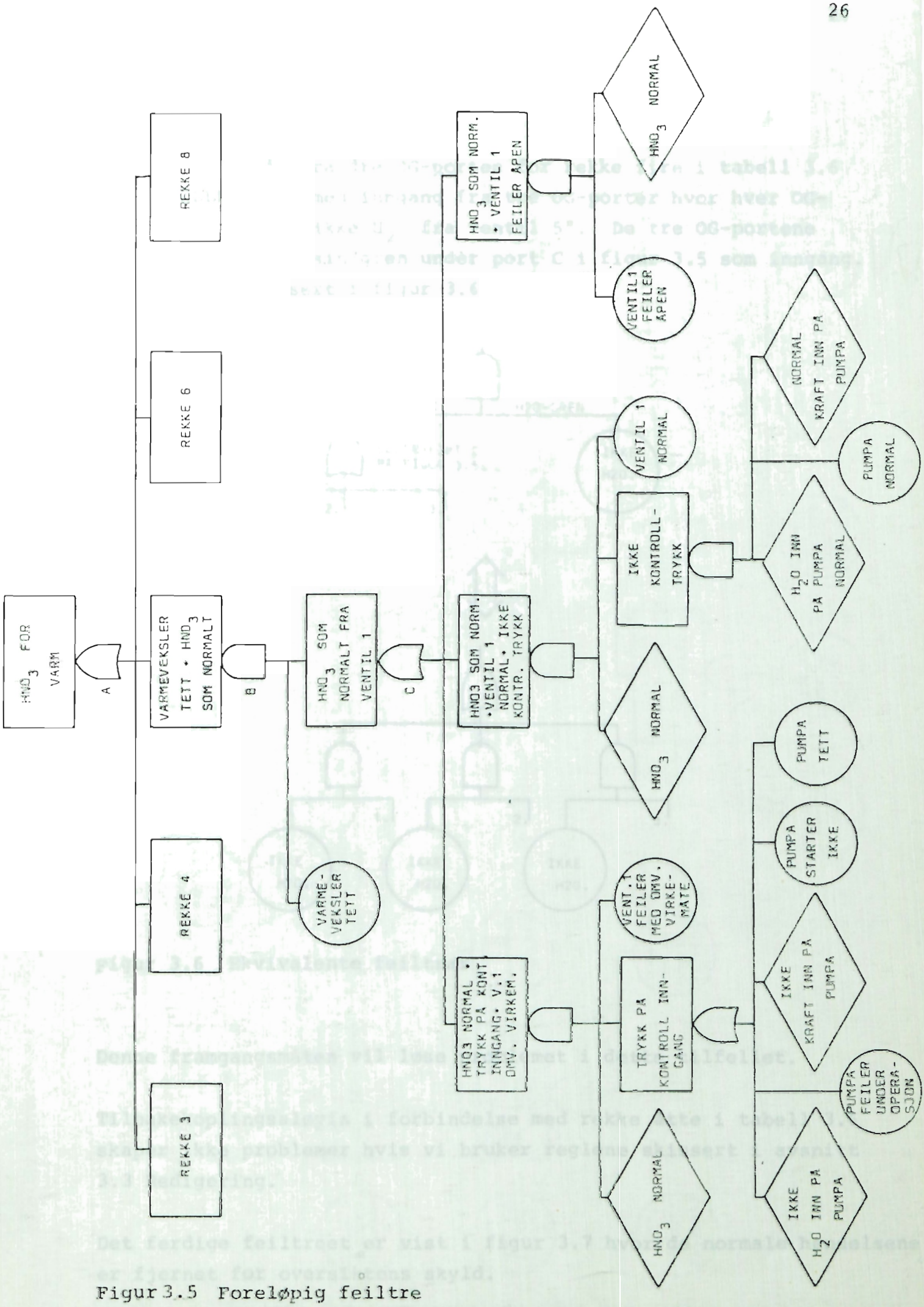
Av rekke sju sees at "HNO_3 normalt" og "ventil 1 feiler åpen" fører til normal HNO_3 strøm ut. Dermed oppstår feiltreet på figur 3.5. Det er tydelig at gren en og to under ELLER-port C er gjensidig utelukkende.

Hvis vi ønsker å fjerne normale hendelser og setter sannsynligheten for at de inntreffer lik en, vil gren to under ELLER-port C bli en sikker hendelse. Da kan hele porten fjernes, og vi står igjen med primærhendelsen "varmeveksler tett" under OG-port B, som ikke lenger er noen egentlig port.

Utviklingen av rekke tre og seks i tabell 3.6 blir omtrent på samme måte som for rekke fem. Men det oppstår problemer når vi skal utvikle rekke fire. De to inngangene på varmeveksleren er siste del av foroverkoplingen fra pumpa og vil ifølge rekke fire stå under samme OG-port. Utvikles hendelsen "HNO_3 normalt" først, fås treet under port C på figur 3.5. Gren en og gren to definerer to forskjellige systemtilstander for pumpa. Når vi kommer til pumpa via hendelsen "ikke kjølevann", det vil si gjennom den andre grenen i foroverkoplingen, er det ikke likegyldig hvilken tilstand pumpa er i. At det ikke kommer kjølevann kan nemlig skyldes at det ikke kommer vann fra pumpa, og det er i konflikt med gren to under port C i figur 3.5, mens det stemmer bra med gren en.

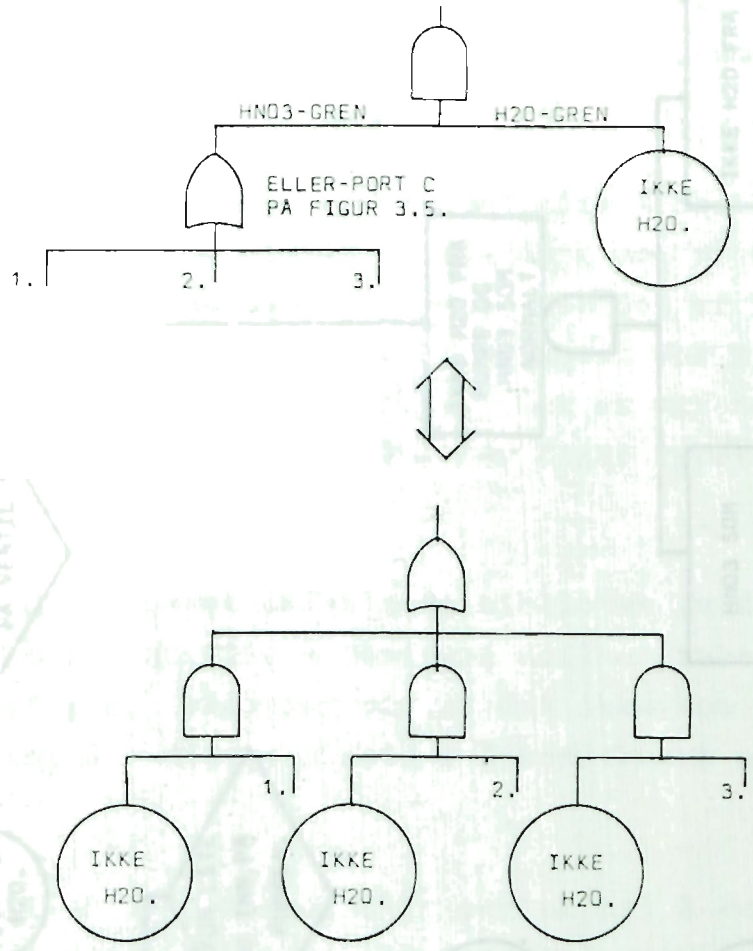
Det er ikke beskrevet hvordan dette skal løses i journalen for CAT-code /9/. Vi har ikke funnet at metoden i det hele tatt har hensyn til problemet. Men i tabell "Sammenligning av noen programmer for automatisk konstruksjon av feiltrær", hevdes det at CAT-code skal håndtere foroverkoplinger.

En løsning på problemet kan være å utvikle hendelsen "ikke H_2O fra ventil 5" først med den ene definisjonen på tilstanden til pumpa og så med den andre.



Figur 3.5 Foreløpig feiltre

Det tilsvarer å forandre OG-porten for rekke fire i tabell 3.6 til en ELLER-port med inngang fra tre OG-porter hvor hver OG-port har inngang "ikke H_2O fra ventil 5". De tre OG-portene har dessuten hver sin gren under port C i figur 3.5 som inngang. Dette er visualisert i figur 3.6.

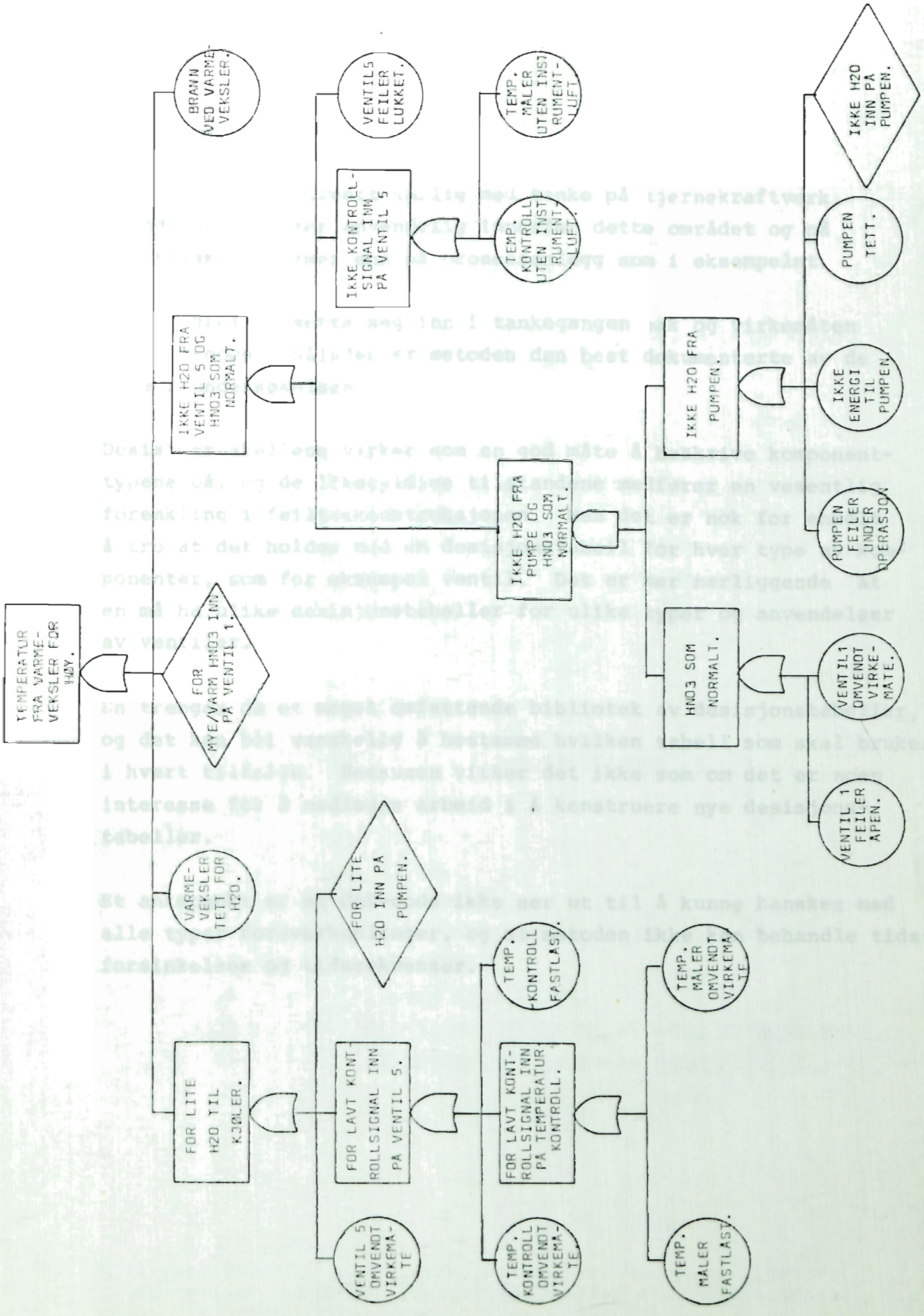


Figur 3.6 Ekvivalente feiltrær.

Denne framgangsmåten vil løse problemet i dette tilfellet.

Tilbakekoplingssløyfa i forbindelse med rekke åtte i tabell 3.6 skaper ikke problemer hvis vi bruker reglene skissert i avsnitt 3.3 Redigering.

Det ferdige feiltreet er vist i figur 3.7 hvor de normale hendelsene er fjernet for oversiktens skyld.



Figur 3.7 Feiltre for topphendelsen "for høy temperatur fra varmeveksler"

3.5 Konklusjon

CAT-code er konstruert særlig med tanke på kjernekraftverk. Metoden er nok mer anvendelig innenfor dette området og på elektriske systemer enn på prosessanlegg som i eksempelet.

Det er greit å sette seg inn i tankegangen bak og virkemåten til CAT-code. Således er metoden den best dokumenterte av de fire i undersøkelsen.

Desisjonstabellene virker som en god måte å beskrive komponenttypene på, og de likegyldige tilstandene medfører en vesentlig forenkling i feiltrekonstruksjonen. Men det er nok for enkelt å tro at det holder med en desisjonstabell for hver type av komponenter, som for eksempel ventil. Det er mer nærliggende at en må ha ulike desisjonstabeller for ulike typer og anvendelser av ventiler.

En trenger da et meget omfattende bibliotek av desisjonstabeller, og det kan bli vanskelig å bestemme hvilken tabell som skal brukes i hvert tilfelle. Dessuten virker det ikke som om det er noen interesse for å nedlegge arbeid i å konstruere nye desisjonstabeller.

Et ankepunkt er at CAT-code ikke ser ut til å kunne hanskes med alle typer foroverkoplinger, og at metoden ikke kan behandle tidsforsinkelser og tidssekvenser.

TABELL 3.2. KOMPONENTERS FEILEMÅTER /8,s16/

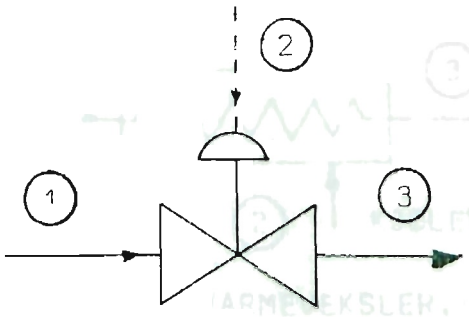
-1 :	LIKEGYLDIG.
0 :	I ORDEN.
1 :	FEILER ÅPEN.
2 :	FEILER LUKKET.
3 :	INTERN FEIL (UDEFINERT).
4 :	STARTER IKKE.
5 :	FUNGERER IKKE RIKTIG (FEIL UNDER OPERASJON).
6 :	OPERERER UTEN STARTSIGNAL.
7 :	INGEN ENERGIKILDE.
10 :	OMVENDT VIRKEMÅTE.
11 :	BRANN.
17 :	IKKE INSTRUMENTLUFT.
3002 :	BRUKKET ISTYKKER.
3003 :	TETT.
3004 :	FASTLÅST.

TABELL 3.3. SIGNALTILSTANDER /8,s17/

-1 :	LIKEGYLDIG.
0 :	INTET SIGNAL.
1 :	NORMALT SIGNAL.*
2 :	OVERBELASTNING (FOR HØYT SIGNAL).
3 :	FOR LAVT SIGNAL.
9 :	SIGNAL UTEN REGULERING.
92 :	FOR HØYT SIGNAL PÅ GRUNN AV REGULERINGSSLØYFE.
93 :	FOR LAVT SIGNAL PÅ GRUNN AV REGULERINGSSLØYFE.

* Signal innenfor visse grenser definert ved hvor store forstyrrelser systemets reguleringssløyfer skal kunne oppta.

VENTILER



FIGUR 3.8 VENTIL.

TABELL 3.4. DESISJONSTABELL FOR REGULERINGS-VENTIL (TRYKK FOR Å ÅPNE).

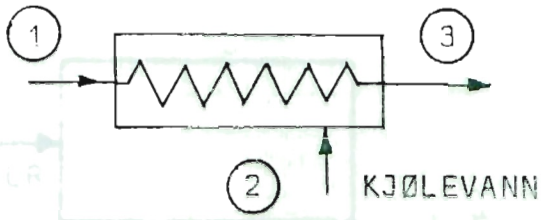
REKKE	INNGANG (1) (HOVEDSTRØM)	INNGANG (2) (KONTR. SIGNAL)	INTERN FUNKSJON	UTGANG (3)
1	0	-1	-1	0
2	-1	0	0	0
3	-1	-1	2	0
4	1	1	0	1
5	1	3	0	1 *
6	1	2	0	1 *
7	1	-1	1	2
8	1	1	10	93
9	1	9	0	93

TABELL 3.5. DESISJONSTABELL FOR NØDSTOPP-VENTIL (TRYKK FOR Å LUKKE).

REKKE	INNGANG (1) (HOVEDSTRØM)	INNGANG (2) (KONTR. SIGNAL)	INTERN FUNKSJON	UTGANG (3)
1	0	-1	-1	0
2	-1	-1	2	0
3	-1	0	10	0
4	-1	1	0	0
5	1	1	10	1
6	1	0	0	1
7	1	-1	1	1
8	2	0	0	2

* Utslagene er for store eller raske til at ventilen reagerer raskt nok.

VARMEVEKSLER

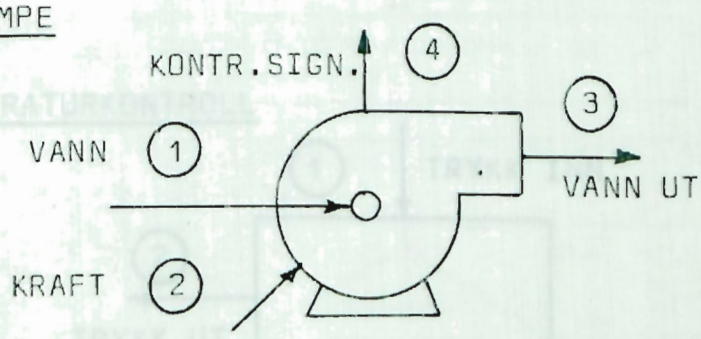


FIGUR 3.9 VARMEVEKSLER.

TABELL 3.6. DESISJONSTABELL FOR VARMEVEKSLER.

REKKE	INNGANG (1) (HOVEDSTRØM)	INNGANG (2) (KJØLEVANN)	INTERN FUNKSJON	UTGANG (3)
1	0	-1	-1	0
2	1	1	0	1
3	2	-1	-1	2
4	1	0	-1	2
5	1	-1	3003*	2
6	1	-1	11	2
7	3	1	0	3
8	1	93	0	92

PUMPE



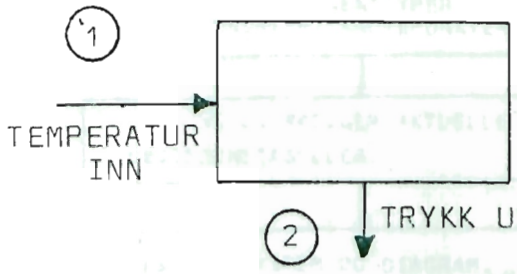
FIGUR 3.10 PUMPE.

TABELL 3.7. DESISJONSTABELL FOR PUMPE.

REKKE	INNGANG (1) (HOVEDSTRØM)	INNGANG (2) (KRAFT)	INTERN FUNKSJON	UTGANG (3) (HOVEDSTRØM)	UTGANG (4) (KONTR.SIG
1	0	-1	-1	0	1
2	-1	0	-1	0	1
3	-1	-1	4	0	1
4	-1	-1	5	0	1
5	-1	-1	3003	0	1
6	1	1	0	1	0

* Tett for kjølevann

TEMPERATURMÅLER

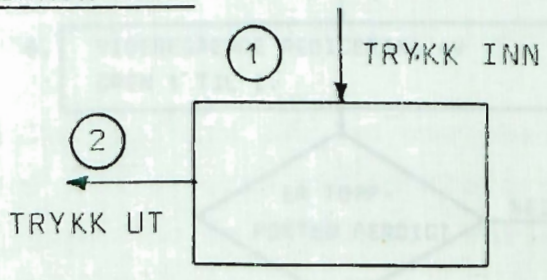


FIGUR 3.11 TEMPERATURMÅLER.

TABELL 3.8 DESISJONSTABELL FOR TEMPERATURMÅLER.

REKKE	INNGANG (1)	INTERN FUNKSJON	UTGANG (2)
1	-1	17	0
2	-1	3003	0
3	1	0	1
4	92	0	1
5	93	0	1
6	2	0	2
7	3	0	3
8	-1	10	9
9	-1	3004	9

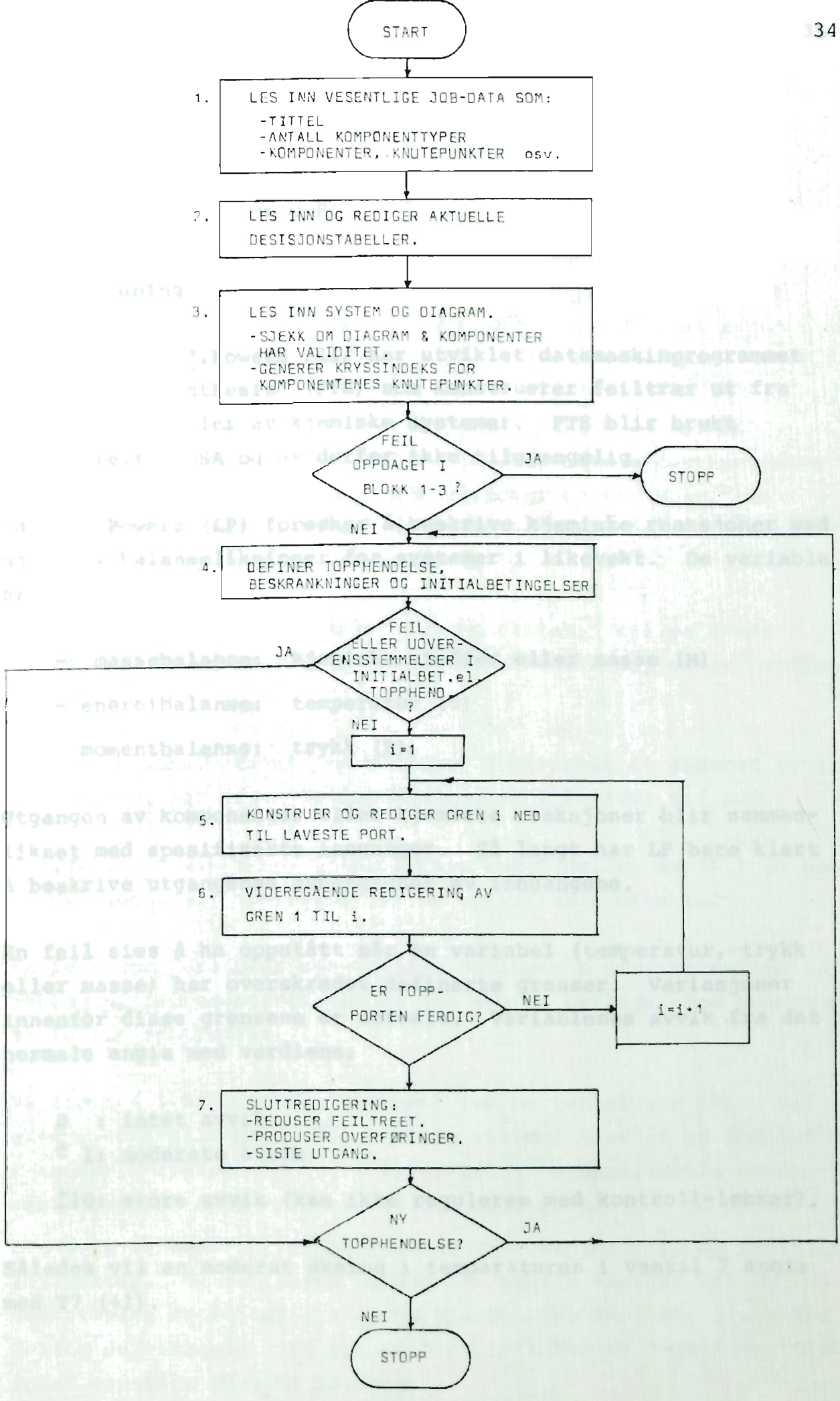
TEMPERATURKONTROLL



FIGUR 3.12 TEMPERATURKONTROLL.

TABELL 3.9 DESISJONSTABELL FOR TEMPERATURKONTROLL.

REKKE	INNGANG (1)	INTERN FUNKSJON	UTGANG (2)
1	-1	17	0
2	-1	3003	0
3	1	0	1
4	2	0	2
5	3	0	3
6	-1	10	9
7	-1	3004	9
8	9	0	9



Figur 3.13 Flytskjema for CAT-code /9, s.180/.

4. LAPP OG POWERS METODE

4.1 Innledning

S.A.Lapp og G.J.Powers /11/ har utviklet datamaskinprogrammet "Fault Tree Synthesis" (FTS) som konstruerer feiltrær ut fra "digraph" modeller av kjemiske systemer. FTS blir brukt kommersielt i USA og er derfor ikke tilgjengelig.

Lapp og Powers (LP) forsøker å beskrive kjemiske reaksjoner ved hjelp av balanselikninger for systemer i likevekt. De variable er:

- massebalanse: kjemiske stoffer eller masse (M)
- energibalanse: temperatur (T)
- momentbalanse: trykk (P).

Utgangen av komponenter eller kjemiske reaksjoner blir sammenliknet med spesifiserte innganger. Så langt har LP bare klart å beskrive utgangenes avhengighet av inngangene.

En feil sies å ha oppstått når en variabel (temperatur, trykk eller masse) har overskredet definerte grenser. Variasjoner innenfor disse grensene er normale. Variablenes avvik fra det normale angis med verdiene:

0 : intet avvik
± 1: moderate avvik
±10: store avvik (kan ikke reguleres med kontroll-løkker).

Således vil en moderat økning i temperaturen i ventil 7 angis med T7 (+1).

4.2 "Digraph" modeller

Med utgangspunkt i et fysisk system konstruerer LP først en "digraph" modell, som visualiserer årsak-virkningsforholdet mellom variable og hendelser i systemet. Deretter anvendes FTS på "digraph" modellen.

En "digraph" er en samling knutepunkter bundet sammen med rettete linjestykker, kalt piler. Knutepunktene representerer variable eller feilhendelser som påvirker de variable. En pil beskriver forholdet mellom de to variable den forbinder eller mellom en feil og en variabel.

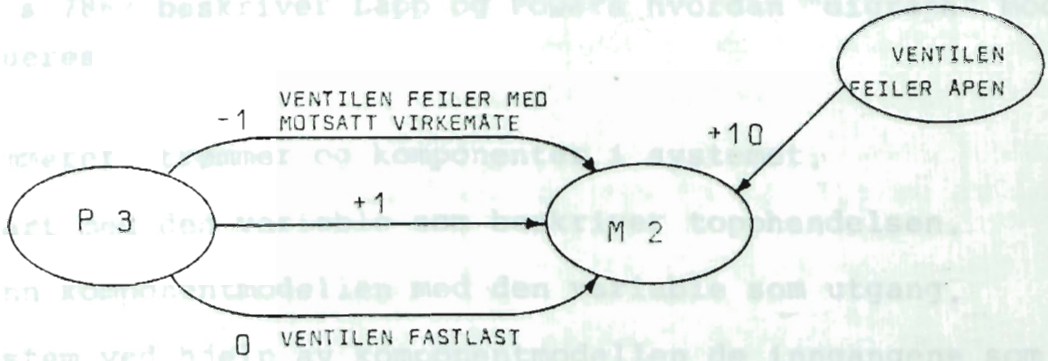
Økningen i avviket "the gain" fra en variabel til en annen finnes fra balanselikningene og diskretiseres til en av verdiene -10, -1, +1, +10.

Hvis et moderat avvik i en variabel forårsaker et moderat avvik i en annen, har pilen mellom dem tallet 1 knyttet til seg.

Forårsaker avvik i en variabel bare ubetydelige avvik i en annen, er det ingen pil som forbinder de to variable direkte.

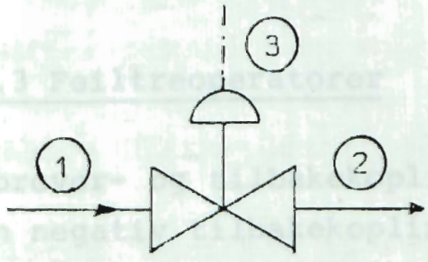
En feil som forandrer den vanlige økningen i avviket mellom to variable, representeres med en pil som er betinget med hensyn på at feilen inntreffer.

På figur 4.1 er vist en "digraph" for en ventil som åpner ved økende kontrolltrykk. Den normale virkemåte er at en moderat økning i kontrolltrykket (P3 (+1)) fører til at massestrømmen ut av ventilen (M2) øker moderat, se den midterste pilen på figur 4.1. Den øvre og nedre pilen på figuren representerer to av ventilens feilemåter. De betingede pilene har alltid knyttet til seg en beskrivelse av feilen i tillegg til den nye økningen i avviket mellom de variable. En ser også at hvis feilen "ventilen feiler åpen" oppstår, øker M2 kraftig.



Figur 4.1 "Digraph" for ventil.

"Digraph"ene som beskriver en komponent kan settes sammen til en komponentmodell som i figur 4.2.



UT / INN	M2	T2
P1	+1	0
P3	+1	0
T1	0	+1
FEIL: VENTIL FEILER ÅPEN	+10	0

VENTILEN FEILER MED OMVENDT VIRKEMÅTE	FORANDRER ØKNINGEN I AVVIKET MELLOM P3 OG M2 TIL -1.
VENTILEN FEILER FASTLÅST	FORANDRER ØKNINGEN I AVVIKET MELLOM P3 OG M2 TIL 0.

Figur 4.2 Komponentmodell for ventil.

Pilenes retning er fra rekke til kolonne. Null betyr ingen påvirkning. Slik kan en lage et bibliotek av komponentmodeller.

I /13, s 786/ beskriver Lapp og Powers hvordan "digraph" modeller konstrueres:

1. Nummerer strømmer og komponenter i systemet.
2. Start med den variable som beskriver topphendelsen.
3. Finn komponentmodellen med den variable som utgang.
4. Bestem ved hjelp av komponentmodellen de inngangene som påvirker den variable.
5. Forfølg hver inngang bakover i komponentene til du kommer til en variabel eller hendelse uten inngang.
6. Hvis det er løkker i systemet, stoppes utviklingen når du kommer til en variabel som er utviklet tidligere.
7. Feil representert ved betingede piler utvikles på samme måte som andre innganger.

Avvikets størrelse og retning når det forplanter seg gjennom systemet finnes ved multiplikasjon med det unntak at 10 x 10 = 10.

4.3 Feiltreoperatorer

Forover- og tilbakekoplinger er vesentlige i kontrollsystemer. En negativ tilbakekoplingssløyfe er karakterisert ved at produktet av de normale økningene i avviket rundt sløyfa er negativt. I en negativ foroverkopling er produktet av de normale økningene i avviket i den ene grenen positivt, mens det er negativt i den andre.

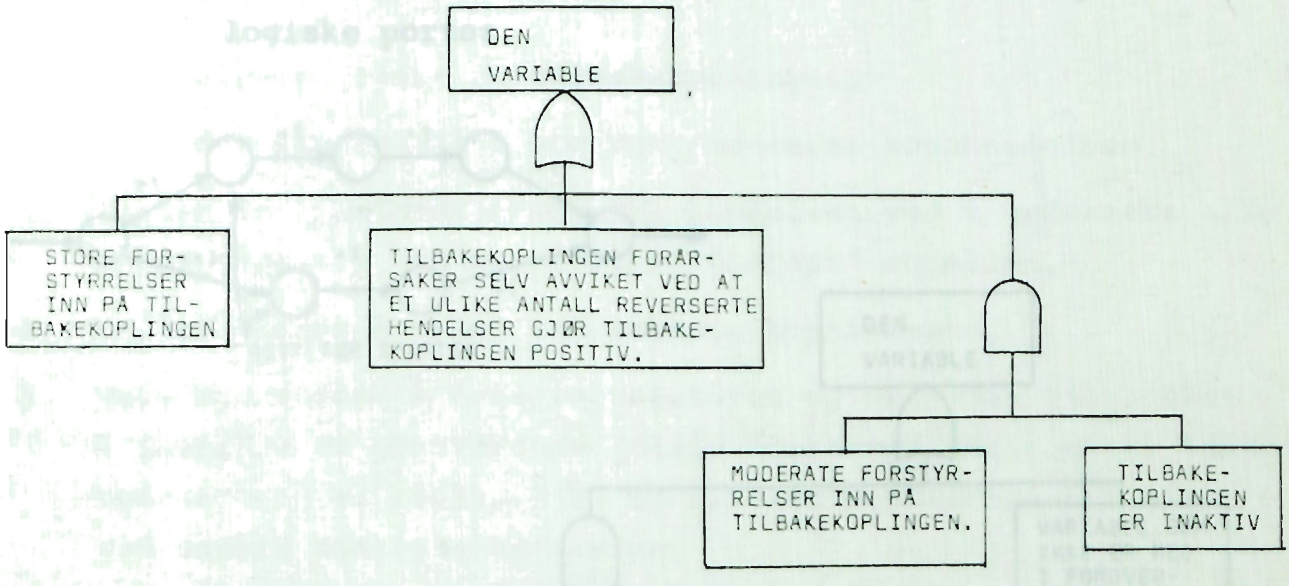
En forstyrrelse (avvik) forplanter seg gjennom en negativ tilbakekopling hvis:

1. Forstyrrelsen er svært stor i hastighet eller størrelse ($\pm$ 10). Tilbakekoplingen klarer ikke å nøytralisere slike avvik.

2. Tilbakekoplingen selv er skyld i forstyrrelsen. Det vil alltid være støy tilstede i systemet. Hvis en feil oppstår slik at tilbakekoplingen blir positiv, fører støyen til at tilbakekoplingen blir ustabil, og en forstyrrelse oppstår.

3. Tilbakekoplingen ikke klarer å nøytralisere moderate forstyrrelser ($\pm$ 1) som kommer inn på tilbakekoplingen. Feil som fører til null økning i avviket mellom to variable på tilbakekoplingen,gjør denne inaktiv. Da vil alle forstyrrelser forplante seg gjennom sløyfa.

Ved utviklingen av en variabel brukes forskjellige operatorer. Er den variable en del av en tilbakekoplingssløyfe anvendes operatoren på figur 4.3.



Figur 4.3 Operator for variabel i tilbakekoplingssløyfe /13, s 784/.

De tre grenene i treet svarer til de tre punktene nevnt ovenfor. Gren to fører til at feiltreet vil inneholde såkalte EKSELLER-porter. Hvis en tilbakekopling består av to komponenter, blir den positiv hvis en, men bare en, av komponentene får reversert virkemåte. Reverseres begge komponentene, er tilbakekoplingen fortsatt negativ.

EKSELLER-porter har utgangshendelse hvis en og bare en av inngangene inntreffer.

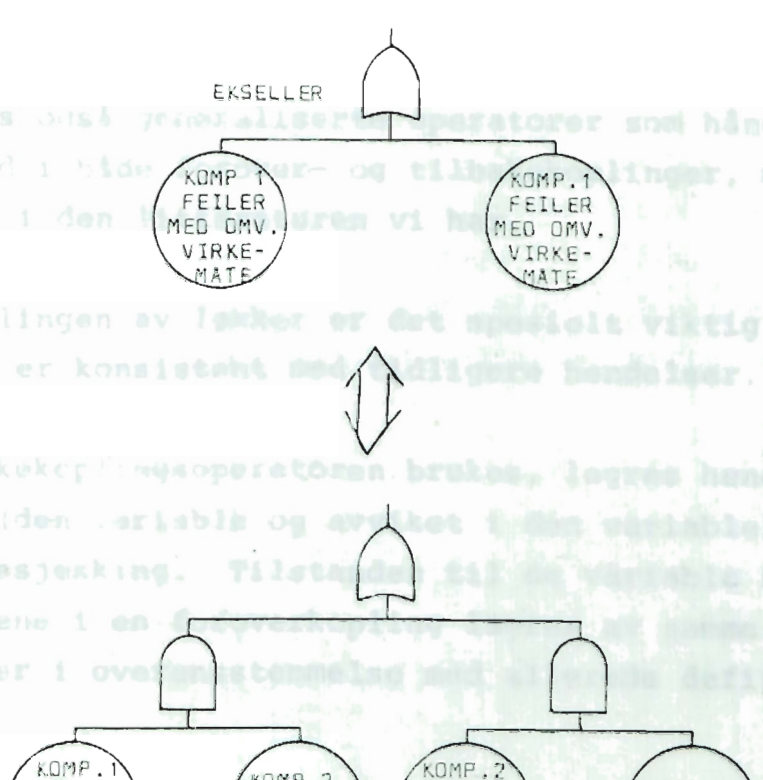
Logikken i EKSELLER-portene medfører implisitt at normale tilstander kommer med i feiltrærene.

I eksempelet over vil gren to bestå av en EKSELLER-port med to innganger: "komponent en feiler med omvendt virkemåte" og "komponent to feiler med omvendt virkemåte". Det tilsvarer en ELLER-port med to OG-porter som innganger. Den ene OG-porten har inngangene "komponent en feiler med omvendt virkemåte" og "komponent to normal", mens den andre OG-porten har inngangen "komponent en normal" og "komponent to feiler med omvendt virkemåte", se figur 4.4.

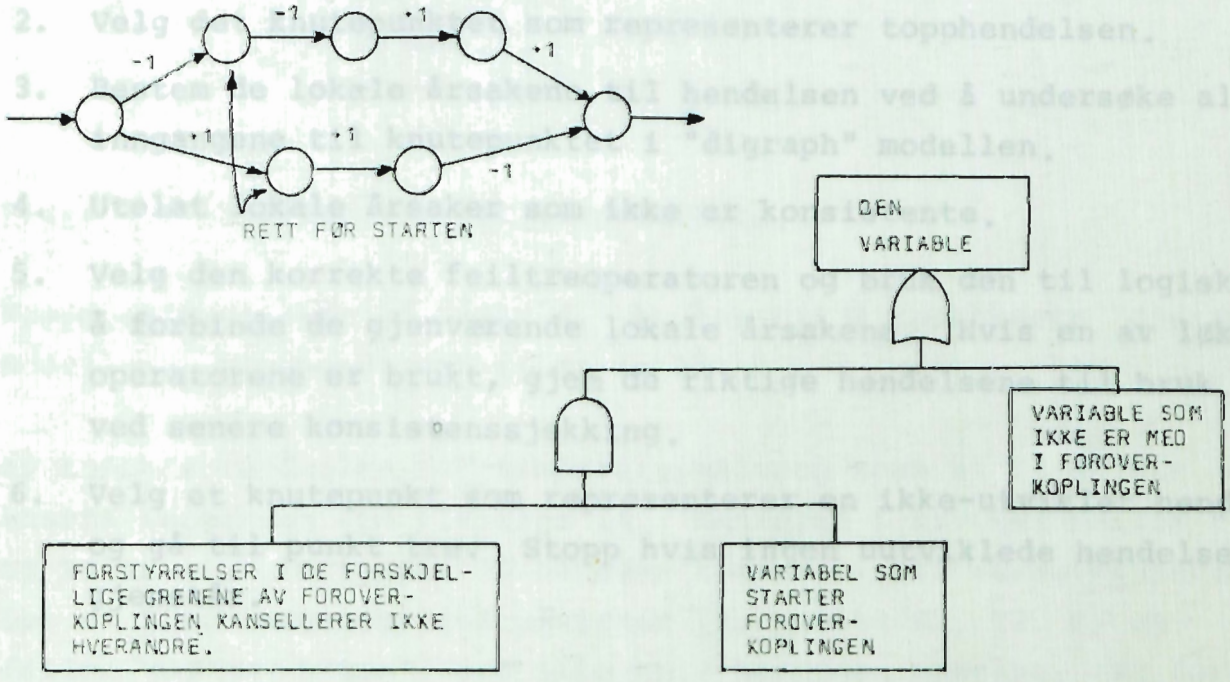
Lambert /15, s2/ skriver at EKSELLER-portene er unødvendige ut fra en ingeniørs synspunkt. Den heldige situasjon at to feil inntreffer samtidig og opphever hverandre, er så lite sannsynlig at en kan se bort fra den. Dermed kan EKSELLER-porten byttes med vanlige ELLER-porter, og feiltreet blir koherent.

Hvis den variable er rett før starten av en foroverkopling, brukes operatoren i figur 4.5 hvor det også er visualisert hva som menes med "rett før starten".

Hvis den variable ikke er rett før starten av en foroverkopling eller med i en tilbakekopling, brukes en vanlig ELLER-port med de aktuelle inngangene.



Figur 4.4 Ekvivalens mellom EKSELLER-porter og "vanlige" logiske porter.



Figur 4.5 Operator for variable rett før starten av foroverkoplinger /13, s 784/.

Det finnes også generaliserte operatorer som håndterer variable som er med i både forover- og tilbakekoplinger, men de er ikke beskrevet i den litteraturen vi har.

Ved utviklingen av løkker er det spesielt viktig å sjekke om hendelser er konsistent med tidligere hendelser.

Når tilbakekoplingsoperatoren brukes, lagres hendelsen som utvikles (den variable og avviket i den variable) for senere konsistenssjekking. Tilstanden til de variable som er felles for alle grenene i en foroverkopling lagres av samme grunn. Hendelser som ikke er i overensstemmelse med allerede definerte hendelser, utelates.

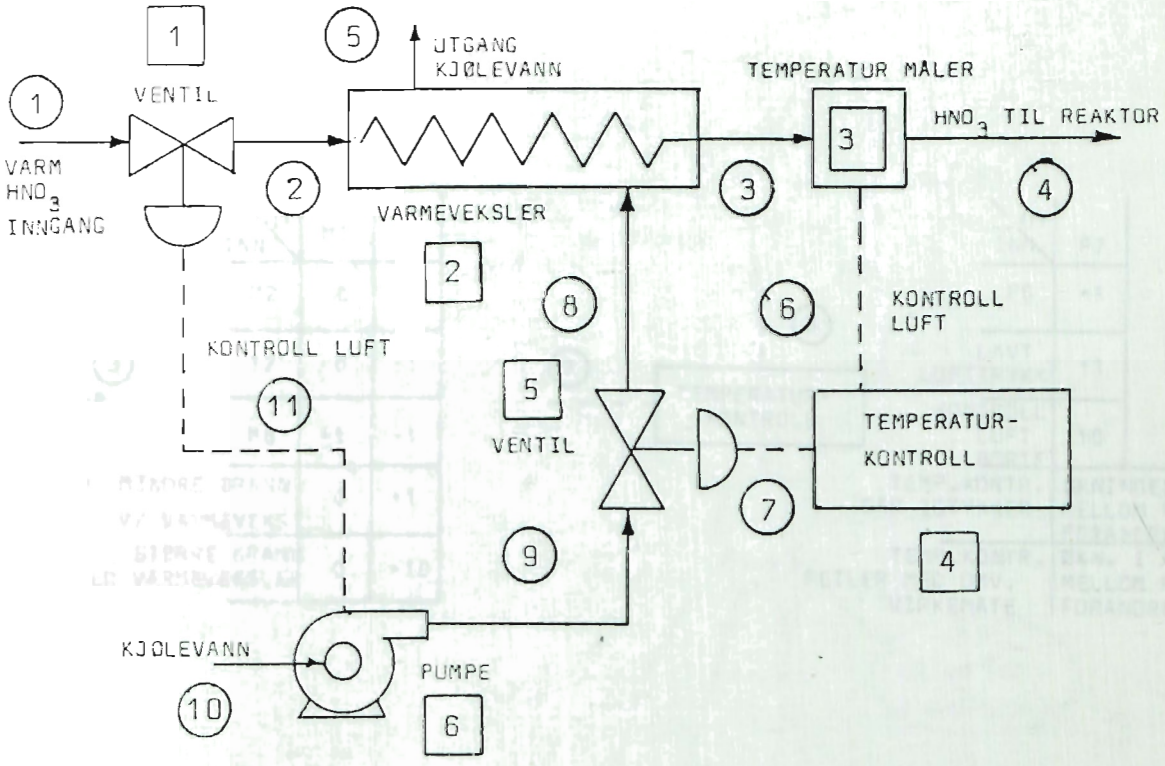
4.4 Feiltre algoritmer

Lapp og Powers gir i /11, s 9/ en algoritme som konstruerer feiltrær fra "digraph" modeller:

1. Finn alle forover- og tilbakekoplinger.
2. Velg det knutepunktet som representerer topphendelsen.
3. Bestem de lokale årsakene til hendelsen ved å undersøke alle inngangene til knutepunktet i "digraph" modellen.
4. Utelat lokale årsaker som ikke er konsistente.
5. Velg den korrekte feiltreoperatoren og bruk den til logisk å forbinde de gjenværende lokale årsakene. Hvis en av løkke-operatorene er brukt, gjem de riktige hendelsene til bruk ved senere konsistenssjekking.
6. Velg et knutepunkt som representerer en ikke-utviklet hendelse og gå til punkt tre. Stopp hvis ingen uutviklede hendelser gjenstår.

4.5 Eksempel på anvendelse

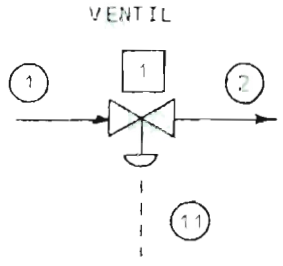
Vi vil anvende Lapp og Powers metode på det samme systemet som vi brukte for å demonstrere CAT-code. LP /11, s 11/ har allerede konstruert et feiltre for kjølesystemet. Systemet er vist i figur 4.6 hvor strømmer og komponenter er nummerert.



Figur 4.6 Kjølesystem for salpetersyre.

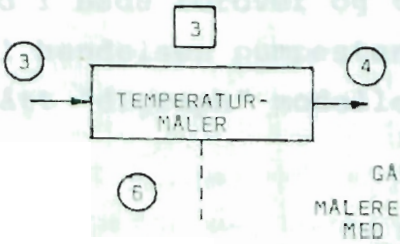
Komponentmodellene på figur 4.7 brukes til å konstruere "digraph" modellen. Topphendelsen er T4 (+ 1).

Av komponentmodellen for temperaturmåleren sees at T3 er den eneste inngangen som påvirker T4. Økningen i avviket mellom T3 og T4 er +1. Forfølges T3 bakover kommer vi til varmeveksleren. Det er fire innganger som påvirker T3, nemlig M2, T2, M8 og feilen "ekstern brann", som ikke selv har noen inngang. T2 forfølges gjennom ventil en til T1 som ikke utvikles videre. M2



UT / INN	M2	T2
P 1	+1	0
P11	-10	0
T 1	0	+1

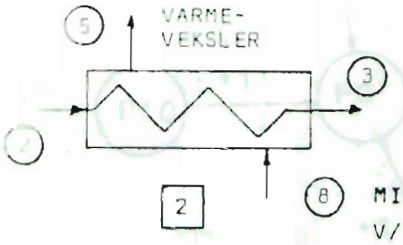
VENTILEN FEILER MED OMVENDT VIRKEMÅTE: ØKNINGEN I AVVIKET MELLOM P11 OG M2 FORANDRES TIL +10.



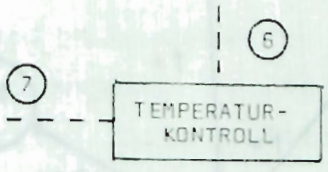
UT / INN	T4	P6
T3	+1	+1

MÅLEREN GÅR I STYKKER: ØKNINGEN I AVVIKET MELLOM T & P6 BLIR 0.

MÅLEREN FEILER MED OMVENDT VIRKEMÅTE: ØKNINGEN I AVVIKET MELLOM T P6 FORANDRES TIL -1



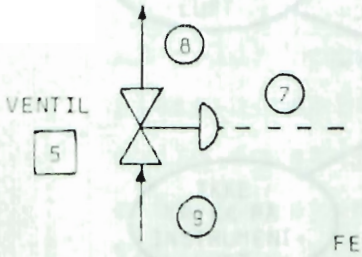
UT / INN	M5	T3
M2	0	+1
T2	0	+1
M8	+1	-1
MINDRE BRANN V/ VARMEVEKS.	0	+1
STØRRE BRANN VED VARMEVEKSLER	0	+10



UT / INN	P7
P6	+1
LAVT LUFTTRYKK	-1
KONTROLL LUFT BORTE	-10

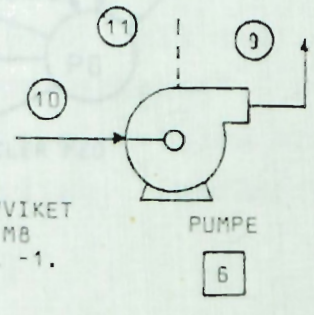
TEMP.KONTR. GÅR I STYKKER: ØKNINGEN I AVVIKE MELLOM P6 OG P7 FORANDRES TIL 0.

TEMP.KONTR. FEILER MED OMV. VIRKEMÅTE: ØKN. I AVVIKET MELLOM P6 OG P7 FORANDRES TIL -1.



UT / INN	M8
P6	+1
P7	+1

VENTILEN FEILER MED OMV. VIRKEMÅTE: ØKNINGEN I AVVIKET MELLOM P7 OG M8 FORANDRES TIL -1.

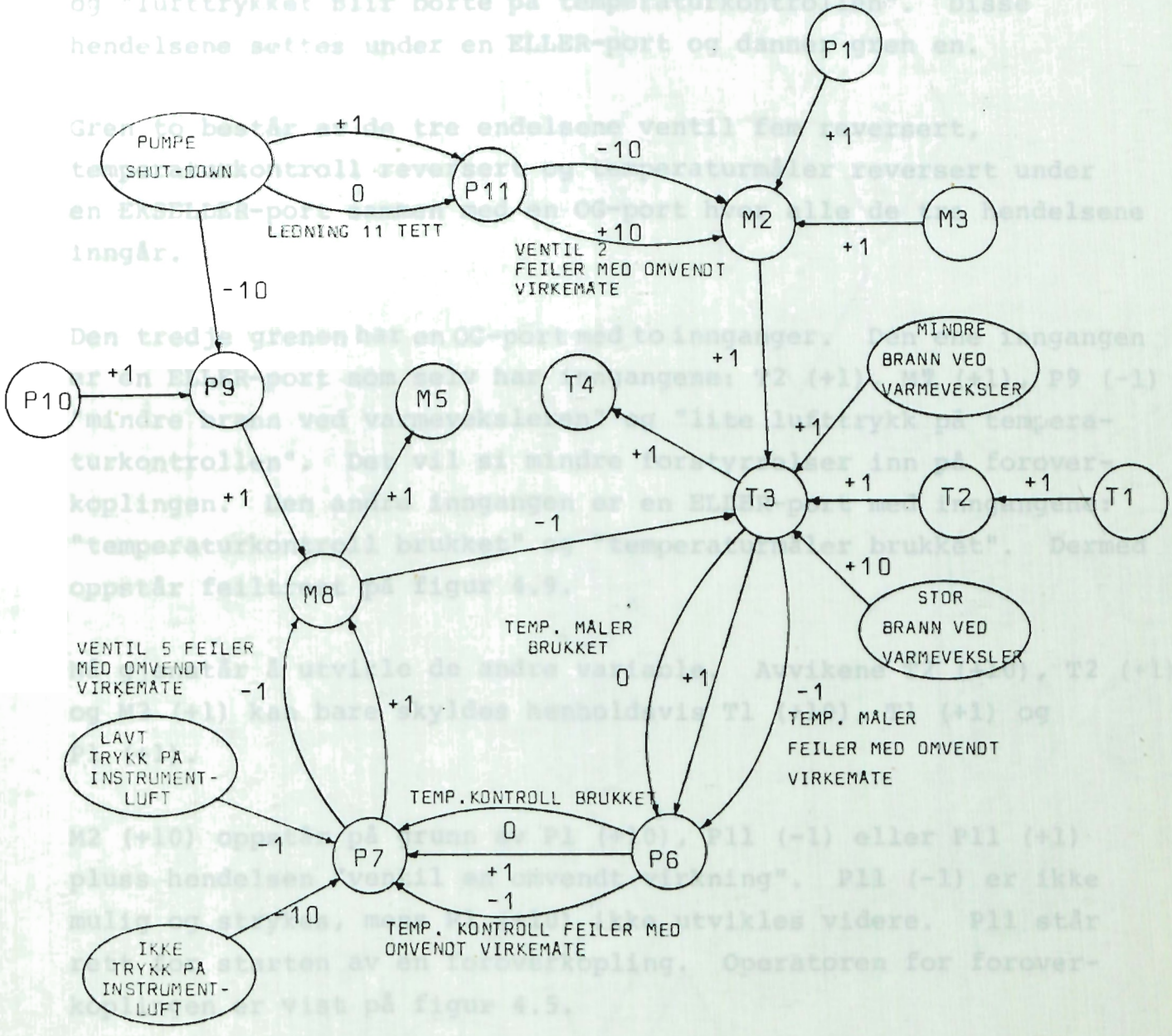


UT / INN	P9	P11
P10	+1	0
PUMPE SHUT-DOWN	-10	+1

RØR 11 TETT VED PUMPEN: ØKN. I AVVIKET MELLOM SHUT-DOWN OG P11 FORANDRES TIL 0.

Figur 4.7 Komponentmodeller.

er med i foroverkoplingen fra pumpa og utvikles bakover til hendelsen: pumpestans. M8 er med i både forover og tilbakekoplingen og utvikles bakover til hendelsen pumpestans og tilbake til T3. Dermed har vi fått "digraph" modellen på figur 4.8.



Figur 4.8 "Digraph" modell av kjølesystem for salpetersyre.

Så skal feiltreet lages av "digraph" modellen.

Topphendelsen T4 (+1) kan bare skyldes T3 (+1). Variablen T3 er med i tilbakekoplingen, og vi bruker operatoren på figur 4.3. De mulige store forstyrrelsene inn på tilbakekoplingen er: M2 (+10), T2 (+10), P9 (-10), "stor brann ved varmeveksleren" og "lufttrykket blir borte på temperaturkontrollen". Disse hendelsene settes under en ELLER-port og danner gren en.

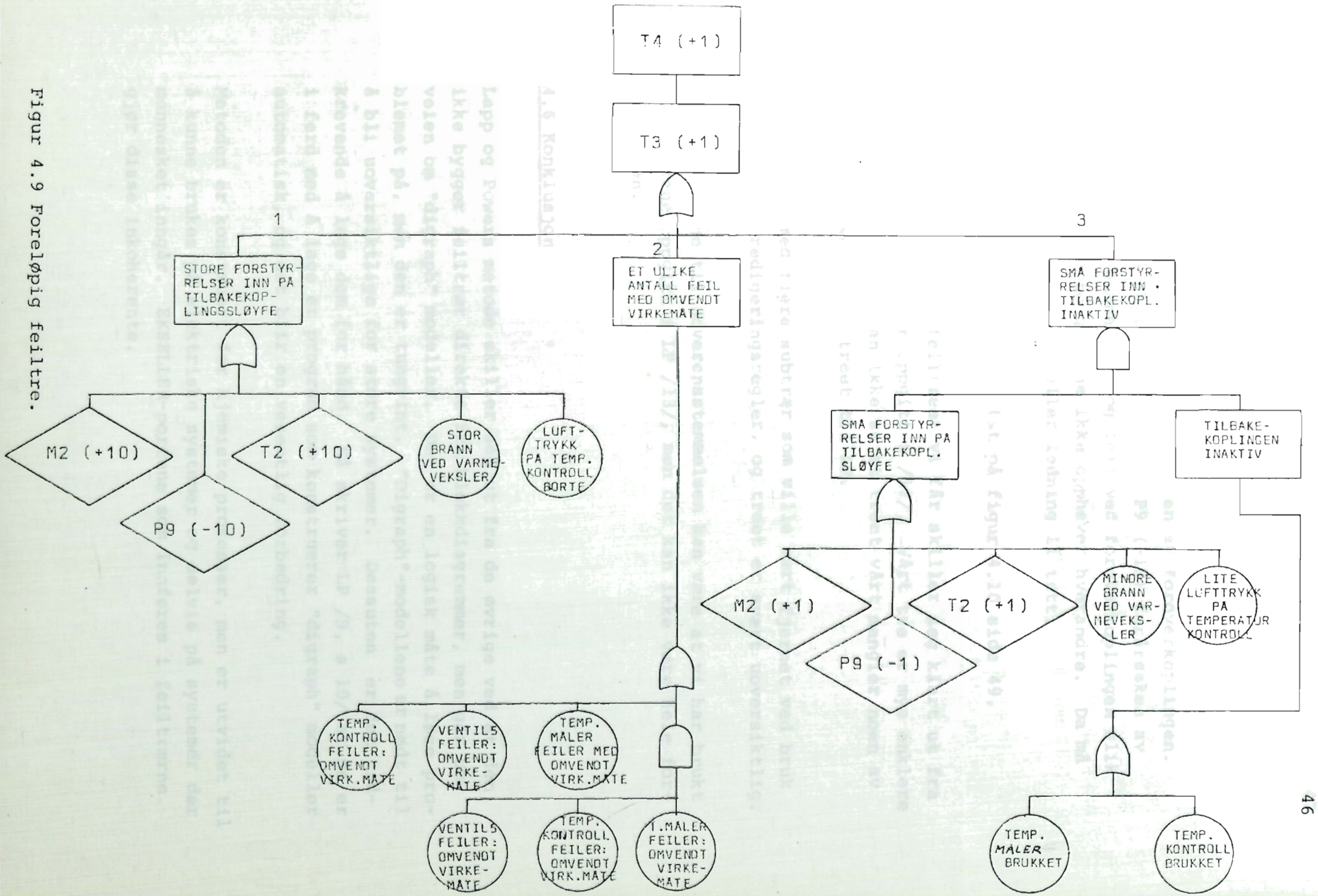
Gren to består av de tre endelsene ventil fem reversert, temperaturkontroll reversert og temperaturmåler reversert under en EKSELLER-port sammen med en OG-port hvor alle de tre hendelsene inngår.

Den tredje grenen har en OG-port med to innganger. Den ene inngangen er en ELLER-port som selv har inngangene: T2 (+1), M2 (+1), P9 (-1) "mindre brann ved varmeveksleren" og "lite lufttrykk på temperaturkontrollen". Det vil si mindre forstyrrelser inn på foroverkoplingen. Den andre inngangen er en ELLER-port med inngangene: "temperaturkontroll brukket" og "temperaturmåler brukket". Dermed oppstår feiltreet på figur 4.9.

Nå gjenstår å utvikle de andre variable. Avvikene T2 (+10), T2 (+1) og M2 (+1) kan bare skyldes henholdsvis T1 (+10), T1 (+1) og P1 (+1).

M2 (+10) oppstår på grunn av P1 (+10), P11 (-1) eller P11 (+1) pluss hendelsen "ventil en omvendt virkning". P11 (-1) er ikke mulig og strykes, mens P1 (+10) ikke utvikles videre. P11 står rett for starten av en foroverkopling. Operatoren for foroverkoplingen er vist på figur 4.5.

P11 påvirkes ikke av noen variable utenom foroverkoplingen. Vi får at en OG-port med inngangene:"pumpa stanser" og "ventil en omvendt virkemåte", sammen med P1 (+10) danner inngangene til en ELLER-port som resulterer i M2 (+10).



Figur 4.9 Foreløpig feiltre.

Variabelen P9 er også rett før starten av foroverkoplingen. P9 (-1) kan bare skyldes P10 (-1). P9 (-10) forårsakes av P10 (-10) eller pumpestans og feil ved foroverkoplingen slik at avvikene i de to grenene ikke opphever hverandre. Da må ventil en være reversert eller ledning 11 tett.

Det resulterende treet er vist på figur 4.10 side 49.

Det viser seg at det feiltreet vi får skiller seg klart ut fra det Lapp og Powers har oppgitt i /11/. Vårt tre er mye enklere en LP sitt. Men vi kan ikke se at treet vårt mangler noen av de minimale kuttene i treet til LP.

LP har fått med flere subtrær som ville vært fjernet ved bruk av CAT-codes redigeringsregler, og treet er svært uoversiktlig.

En av grunnene til uoverensstemmelsen kan være at vi har brukt operatorene oppgitt i LP /13/, men det kan ikke være hele forklaringen.

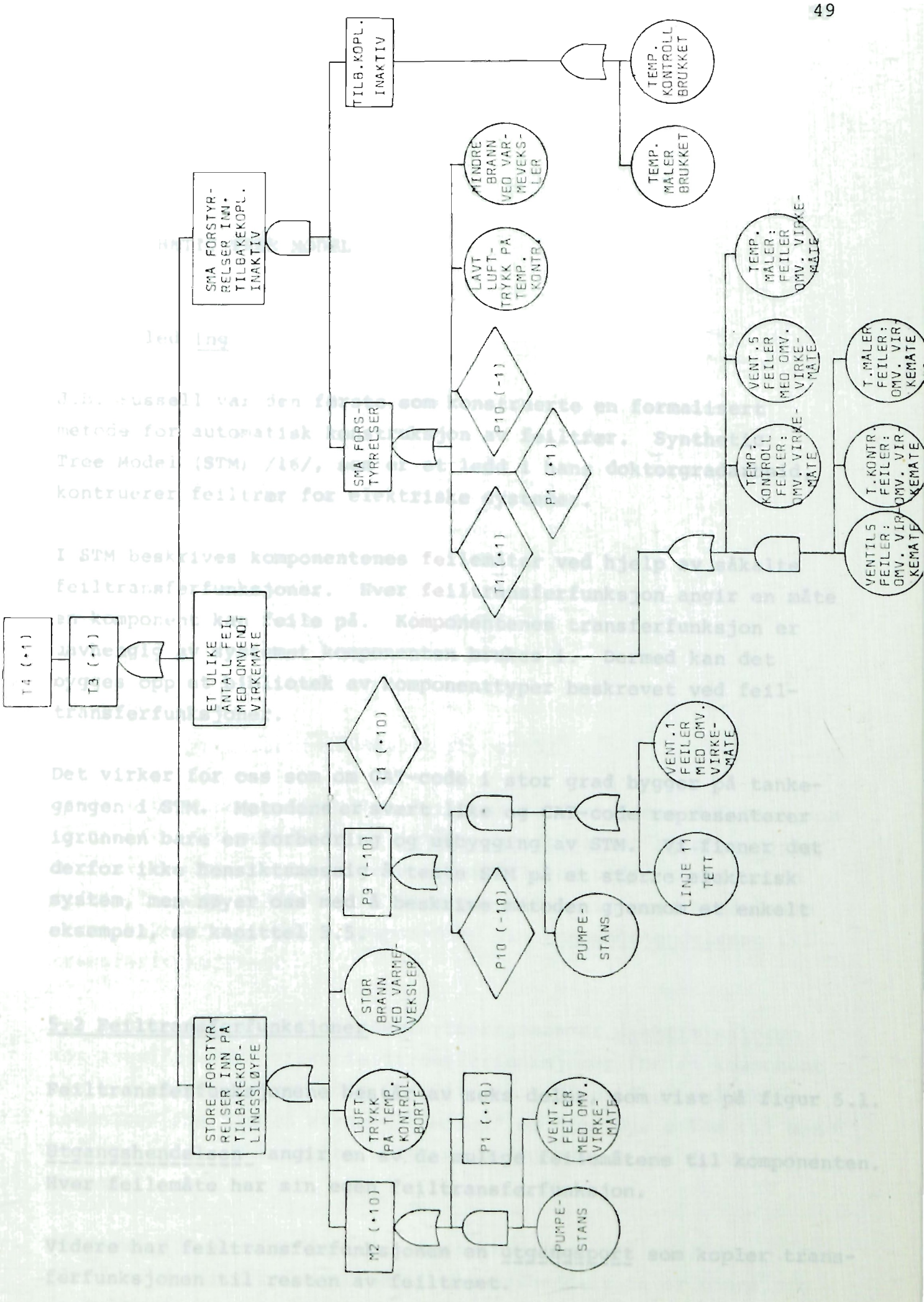
4.6 Konklusjon

Lapp og Powers metode skiller seg ut fra de øvrige ved at den ikke bygger feiltrær direkte fra blokkdiagrammer, men går omveien om "digraph" modeller. Det er en logisk måte å løse problemet på, men den er tungvint. "Digraph"-modellene er nødt til å bli uoversiktlige for store systemer. Dessuten er det tidkrevende å lage dem for hånd. Nå skriver LP /9, s 10/ at de er i ferd med å lage et program som konstruerer "digraph" modeller automatisk, og det blir en vesentlig forbedring.

Metoden er konstruert for kjemiske prosesser, men er utvidet til å kunne brukes på elektriske systemer og delvis på systemer der mennesket inngår. EKSELLER-portene som innføres i feiltrærne gjør disse inkoherente.

Lapp og Powers utvikler stadig sin metode, og den skal nå også kunne løse feiltrær for sekvensielle systemer og for systemer med tidsforsinkelse. Det har ikke vært mulig å teste om dette er riktig, da metoden som tidligere nevnt, brukes kommersielt og derfor ikke er tilgjengelig.

Akkurat dette gjør metoden mindre aktuell, selv om den er vel egnet til å løse feiltrær, særlig for kjemiske systemer.



Figur 4.10 Feiltre for topphendelsen T4 (+1)

5. SYNTHETIC TREE MODEL

5.1 Innledning

J.B. Fussell var den første som konstruerte en formalisert metode for automatisk konstruksjon av feiltrær. Synthetic Tree Model (STM) /16/, som er et ledd i hans doktorgradarbeid, kontruerer feiltrær for elektriske systemer.

I STM beskrives komponentenes feilemåter ved hjelp av såkalte feiltransferfunksjoner. Hver feiltransferfunksjon angir en måte en komponent kan feile på. Komponentenes transferfunksjon er uavhengig av systemet komponenten brukes i. Dermed kan det bygges opp et bibliotek av komponenttyper beskrevet ved feiltransferfunksjoner.

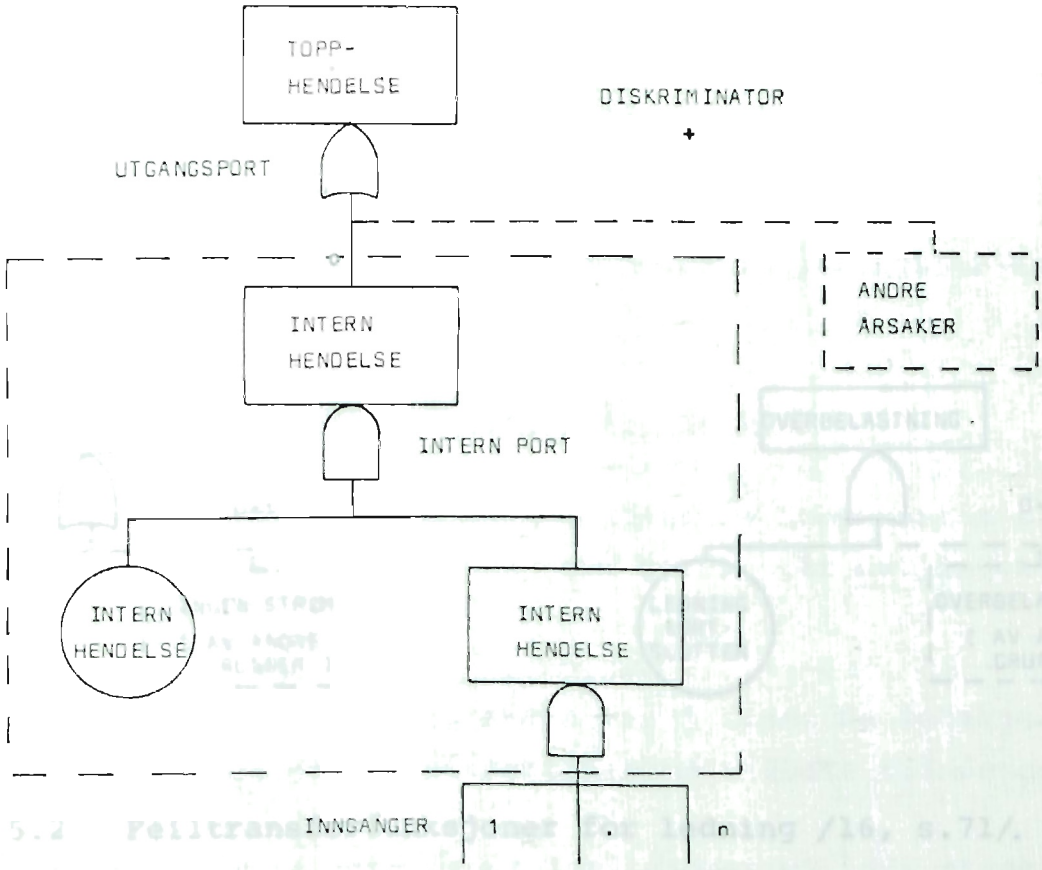
Det virker for oss som om CAT-code i stor grad bygger på tankegangen i STM. Metodene er svært like og CAT-code representerer igrunnen bare en forbedring og utbygging av STM. Vi finner det derfor ikke hensiktsmessig å teste STM på et større elektrisk system, men nøyer oss med å beskrive metoden gjennom et enkelt eksempel, se kapittel 5.5.

5.2 Feiltransferfunksjoner

Feiltransferfunksjonene består av seks deler, som vist på figur 5.1.

Utgangshendelsen angir en av de mulige feilemåtene til komponenten. Hver feilemåte har sin egen feiltransferfunksjon.

Videre har feiltransferfunksjonen en utgangsport som kopler transferfunksjonen til resten av feiltreet.

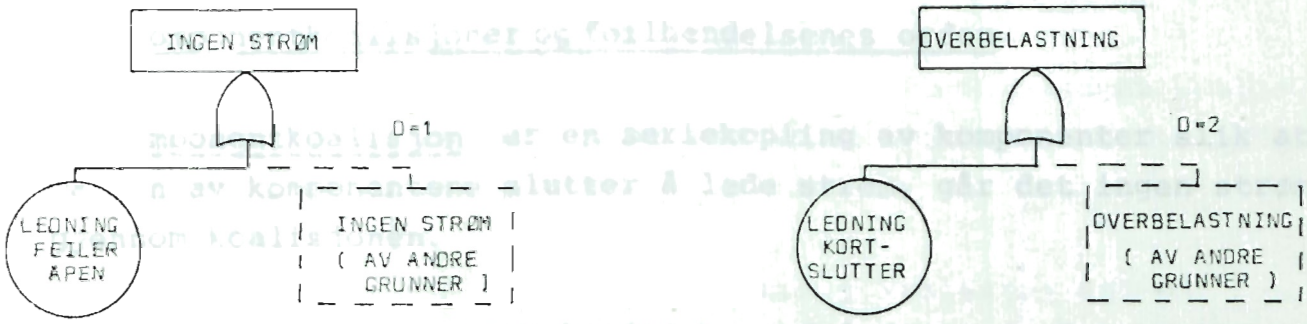


Figur 5.1 Feiltransferfunksjon /9, s.53/.

Under utgangsporten er selve feiltransferfunksjonen definert som et minifeiltre bestående av interne hendelser og interne porter. Noen av de interne hendelsene kan forklares utfra komponenten selv. I andre tilfeller må årsakene til de interne hendelsene søkes bakover til andre komponenter via inngangshendelsene til transferfunksjonen.

Den sjette delen av feiltransferfunksjonen er diskriminatoren som spesifiserer hvilke feiltransferfunksjoner for en komponent som kan eksistere samtidig i feiltreet. Gjensidig utelukkende hendelser som "ingen strøm til motor" og "for mye strøm til motor", kan som kjent ikke opptre under samme OG-port i det ferdige feiltreet.

På figur 5.2 er vist feiltransferfunksjonene til en ledning. Det går fram av diskriminatorene at utgangshendelsene er gjensidig utelukkende.



Figur 5.2 Feiltransferfunksjoner for ledning /16, s.71/.

Oppgaven til STM blir da å kombinere definerte minifeiltrær til et stort feiltre og samtidig ta hensyn til systemets beskrankninger.

Systembeskrankningene må bestemmes før feiltreskonstruksjonen starter og definerer sammen med systembeskrivelsen situasjonen feiltreet skal konstrueres for.

Topphendelsen er den viktigste beskrankningen og må velges først. De øvrige beskrankninger avhenger av den valgte topphendelsen.

Begynnelsestilstandene er beskrankninger som definerer hvordan systemet normalt funksjonerer. Hver komponent med mer enn en funksjonerende tilstand, genererer en begynnelsestilstand som sier hvilken tilstand komponenten er i.

Beskrankningene inkluderer hendelser som allerede eksisterer, eller som ikke kan oppstå under feiltrekonstruksjonen. Disse hendelsene kalles henholdsvis eksisterende- og ikke-tillatte beskrankninger.

De eksisterende beskrankningene vil inntreffe med sannsynlighet lik en, mens sannsynligheten for at de ikke-tillatte beskrankningene inntreffer er lik null.

Flere beskrankninger kan oppstå i løpet av feiltrekonstruksjonen.

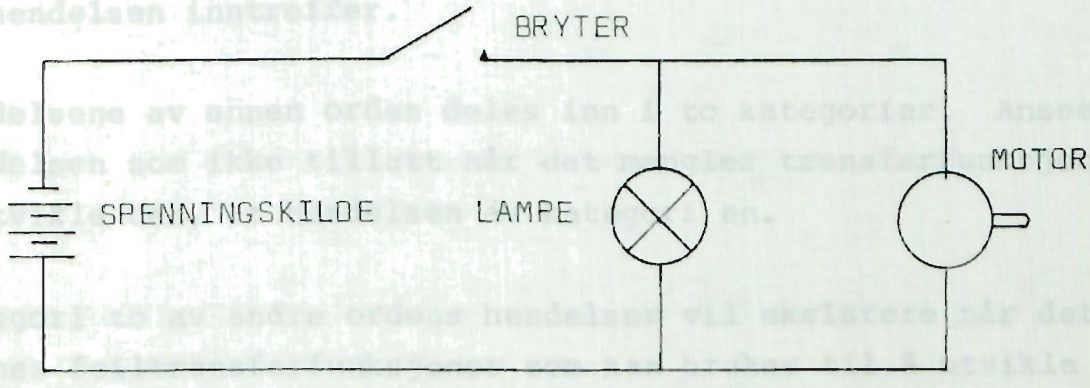
5.3 Komponentkoalisjoner og feilhendelsenes orden

En komponentkoalisjon er en seriekopling av komponenter slik at når en av komponentene slutter å lede strøm, går det ingen strøm gjennom koalisjonen.

Koalisjonene til et system bestemmes ved å finne de forskjellige veiene strømmen kan gå fra strømkilden og tilbake til denne.

Hver komponent kan tilhøre flere koalisjoner og lede strøm via en koalisjon selv om det ikke går strøm i en annen.

I kretsen på figur 5.3 er det to koalisjoner. Koalisjon en består av spenningskilde, bryter og lampe, mens koalisjon to inneholder komponentene: spenningskilde, bryter og motor.



Figur 5.3 Strømkrets med to koalisjoner.

Hvis lampen feiler åpen,går det ingen strøm i koalisjon en. Men det kan likevel gå strøm gjennom bryteren hvis alle komponentene i koalisjon to tillater det.

I STM inndeles feilhendelsene i fire nivåer. Topphendelsen er en første ordens hendelse. Før feiltrekonstruksjonen kan ta til, må topphendelsen manuelt utvikles til høyere ordens hendelser.

Andre ordens hendelser beskriver situasjonen i en bestemt koalisjon og utvikles ved hjelp av transferfunksjoner til komponentene i koalisjonen.

Skal andre ordens hendelsen "ikke strøm i koalisjon en" på figur 5.3 utvikles, benyttes feiltransferfunksjonene til spenningskilden, bryteren og lampen med utgangshendelser "ingen strøm".

Først blir feiltransferfunksjonene med OG-porter som utgangsport samlet under en felles OG-port. Deretter samles transferfunksjonene med utgangsporter av ELLER typen under en felles ELLER-port som settes under OG-porten.

Det kan imidlertid oppstå situasjoner hvor det ikke eksisterer feiltransferfunksjoner til å utvikle hendelsen. Et eksempel er at ingen av komponentene i koalisjonen kan feile på en slik måte at hendelsen inntreffer.

Hendelsene av annen orden deles inn i to kategorier. Ansees hendelsen som ikke tillatt når det mangler transferfunksjoner til å utvikle den, er hendelsen av kategori en.

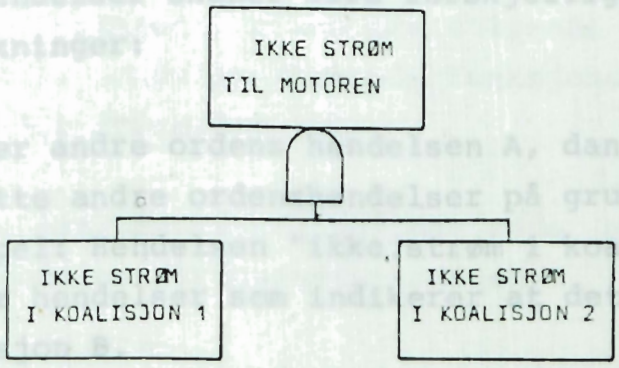
Kategori to av andre ordens hendelser vil eksistere når det ikke finnes feiltransferfunksjoner som kan brukes til å utvikle hendelsene.

Hendelser av typen "ingen strøm" er av kategori en, mens hendelser som "strøm for lenge" er av kategori to.

Tredje og fjerde ordens hendelser beskriver tilstandene i bestemte komponenter. Når en komponent oppfører seg som om den har feilet fordi det er feil et annet sted i systemet, har en tredje ordens hendelse inntruffet.

Et eksempel er "ikke strøm til motoren" på figur 5.3.

Tredje ordens hendelser utvikles ved hjelp av andre ordens hendelser. I eksemplet over må ingen av koalisjonene som motoren er med i, føre strøm. Det vil si at hverken koalisjon en eller koalisjon to i figur 5.3 kan lede strøm. Vi har en situasjon hvor andre ordens hendelser knyttes sammen med en OG-port, se figur 5.4. Tredje ordens hendelser er da av klasse en.



Figur 5.4 Utvikling av tredje ordens hendelsen "ikke strøm til motoren".

Knyttes andre ordens hendelsene sammen med en ELLER-port, er tredje ordens feilen av klasse to.

Fjerde ordens hendelser skyldes at inngangene til en komponent gjør at denne oppfører seg som om den har feilet. Et eksempel er "bryter X holdes lukket". Fjerde ordens hendelser utvikles ved bruk av de rette feiltransferfunksjonene til komponentene hvis

utgang er direkte inngang til komponenten feilen har oppstått i.

5.4 Sammenheng mellom hendelser og beskrankninger

Beskrankningenes oppgave er a redigere komponentenes feiltransferfunksjoner. Det vil si å bestemme hvilke hendelser som er tillatt og hvilke som ikke er det. Første og andre ordens hendelser produserer nye beskrankninger.

Første ordens hendelser genererer beskrankninger av typen ikke-tillatte hendelser. Feiltransferfunksjoner til den komponenten som første ordens hendelsen har oppstått i, kan nemlig aldri opptre i feiltreet. Er topphendelsen "lyspære Y lyser ikke" kan ikke lyspæren feile igjen lenger ned i feiltreet.

Andre ordens hendelser skaper fire forskjellige typer av andre ordens beskrankninger:

Type 1: Opptrer andre ordens hendelsen A, dannes det ikke-tillatte andre ordenshendelser på grunn av A.
Eksempel: Hendelsen "ikke strøm i koalisjon B" utelukker hendelser som indikerer at det går strøm i koalisjon B.

Type 2: En andre ordens hendelse kan medføre at visse komponentfeil er ikke-tillatt beskrankninger under utviklingen av hendelsen.
Eksempel: Utvikles hendelsen "strøm i koalisjon B", kan ingen komponent i koalisjon B feile slik at strømmen blir borte i noen av koalisjonene komponenten er med i.

Type 3: Når en transferfunksjon C brukes til å utvikle en andre ordens hendelse i en komponent D, er transferfunksjoner for komponent D med andre diskriminatorverdier en C's ikke-tillatte beskrankninger.

Eksempel: Brukes den transferfunksjonen som indikerer at en sikring er skyld i overbelastning under utviklingen av en hendelse, kan ikke den samme sikringen feile åpen under utviklingen av hendelsen.

Type 4: Andre ordens hendelser kan også generere eksisterende beskrankninger. Hvis en andre ordens hendelse tilsier at en komponent er i en tilstand som er identisk med begynnelsestilstanden til komponenten, er denne tilstanden en eksisterende beskrankning.
Eksempel: Opptrer en andre ordens hendelse som medfører at det blir ført strøm til en relespole slik at releet lukker, og releet i begynnelsestilstanden er lukket, er denne tilstanden en eksisterende beskrankning.

STM sjekker alltid om en hendelse eller feiltransferfunksjon som skal plasseres i feiltreet, er en eksisterende eller ikke-tillatt beskrankning. Hvis hendelsen/transferfunksjonen er en ikke-tillatt beskrankning, skjer følgende:

- Står hendelsen/transferfunksjonen under en ELLER-port, fjernes hendelsen/transferfunksjonen.

- Står hendelsen/transferfunksjonen under en OG-port, fjernes OG-porten og alle direkte foranstående OG-porter opp til den første ELLER-porten.

Er hendelsen/transferfunksjonen en eksisterende beskrankning, hender dette:

- Står hendelsen/transferfunksjonen under en OG-port, fjernes hendelsen/transferfunksjonen.

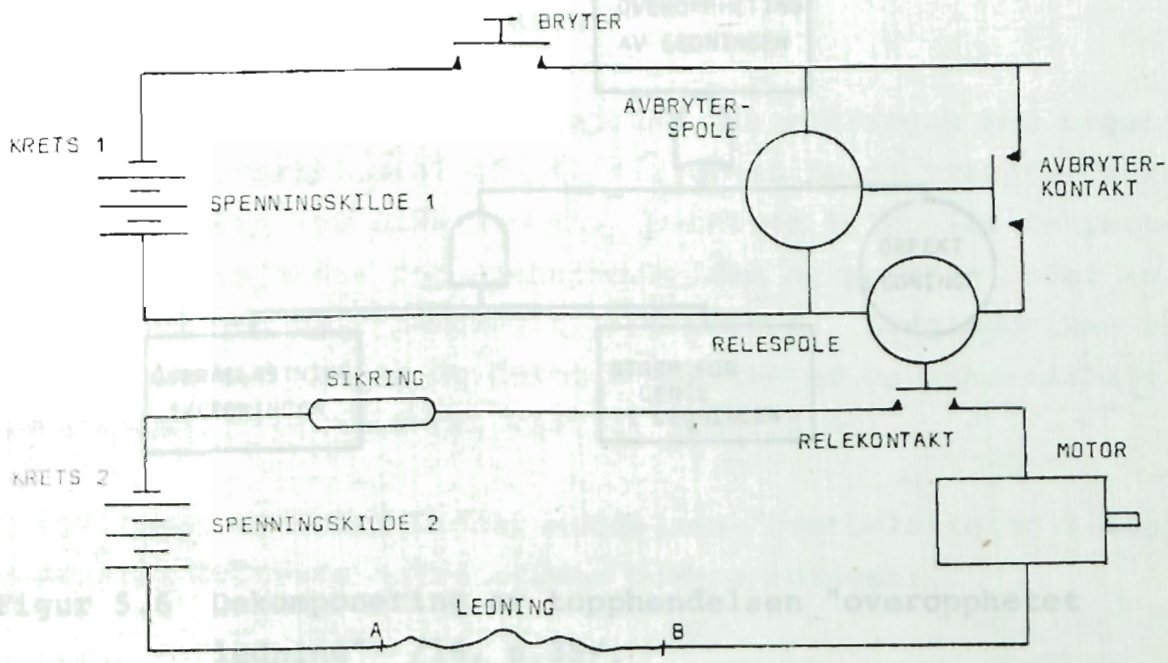
- Står hendelsen/transferfunksjonen under en ELLER-port, fjernes ELLER-porten og alle ELLER-porter direkte over ELLER-porten opp til den nærmeste OG-porten.

Disse reglene er svært lik redigeringsreglene i Cat-code.

I STM løses problemet med løkker på følgende måte: Hvis en hendelse opptrer under utvikling av den samme hendelsen, blir hendelsen betraktet som en ikke-tillatt beskrankning andre gangen den opptrer.

5.5 Eksempel på anvendelse

Systemet på figur 5.5 er beskrevet av Fussell /16, s.34/.



Figur 5.5 System som skal forhindre at ledningen mellom A og B blir overopphetet /16, s.34/.

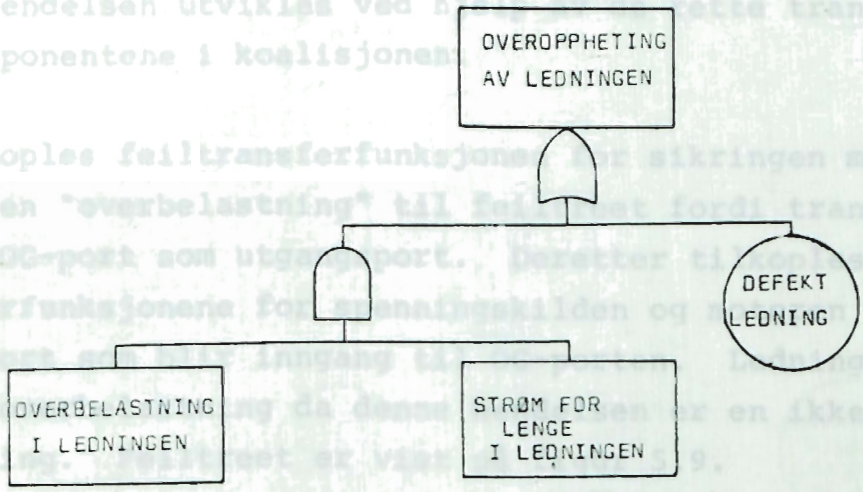
En ønsker å unngå overoppheting av ledningen fra A til B og har bygget et system som skal forhindre dette.

Når bryteren i strømkrets 1 lukkes, blir avbryterspolen tilført strøm, og avbryterkontakten lukkes. Dermed får relespolen strøm, og relekontakten slutter strømkrets 2.

Hvis motoren kortslutter mens relekontakten er lukket, går sikringen, og strømmen blir borte fra ledningen.

Skulle bryteren ikke åpnes etter en forhåndsbestemt tid, vil avbryterkontakten åpne seg og fjerne strømmen i relespolen slik at relekontakten åpnes og strømkrets 2 brytes.

Topphendelsen er overoppheting av ledningen fra A til B. Denne første ordens hendelsen må dekomponeres manuelt til hendelser av høyere orden og/eller primærhendelser. Det er vist på figur 5.6.



Figur 5.6 Dekomponering av topphendelsen "overopphetet ledning" /16, s.35/.

Samtidig genererer topphendelsen de ikke-tillatte beskrankningene "ingen strøm på grunn av ledningen" og "overbelastning på grunn av ledningen". Vi har følgende begynnelsestilstander: "relekontakt lukket", "avbryterkontakt lukket" og "bryter lukket".

Systemet består av tre koalisjoner:

Koalisjon 1: Bryter, avbryterspole og spenningskilde 1.

Koalisjon 2: Bryter, avbryterkontakt, relespole og spenningskilde 1.

Koalisjon 3: Sikring, relekontakt, motor, ledning og spenningskilde 2.

Feiltransferfunksjonene til komponentene er vist på figur 5.7 og 5.8. For kontakter (brytere), relespole og avbryterspole er ikke alle transferfunksjonene tatt med da det ville ta for stor plass.

Vi velger å utvikle tredje ordens hendelsen "overbelastning i ledningen" først. Det er bare koalisjon 3 som inneholder ledningen. Dermed er hendelsens klasse uten betydning. Hendelsen kan bare skyldes andre ordens hendelsen "overbelastning i koalisjon 3". Denne hendelsen utvikles ved hjelp av de rette transferfunksjonene til komponentene i koalisjonen.

Først koples feiltransferfunksjonen for sikringen med utgangshendelsen "overbelastning" til feiltreet fordi transferfunksjonen har en OG-port som utgangsport. Deretter tilkoples de rette feiltransferfunksjonene for spenningskilden og motoren under en ELLER-port som blir inngang til OG-porten. Ledningen kan ikke forårsake overbelastning da denne hendelsen er en ikke-tillatt beskrankning. Feiltreet er vist på figur 5.9.

Utviklingen av andre ordens hendelsen "overbelastning i koalisjon 3" genererer følgende andre ordens beskrankninger:

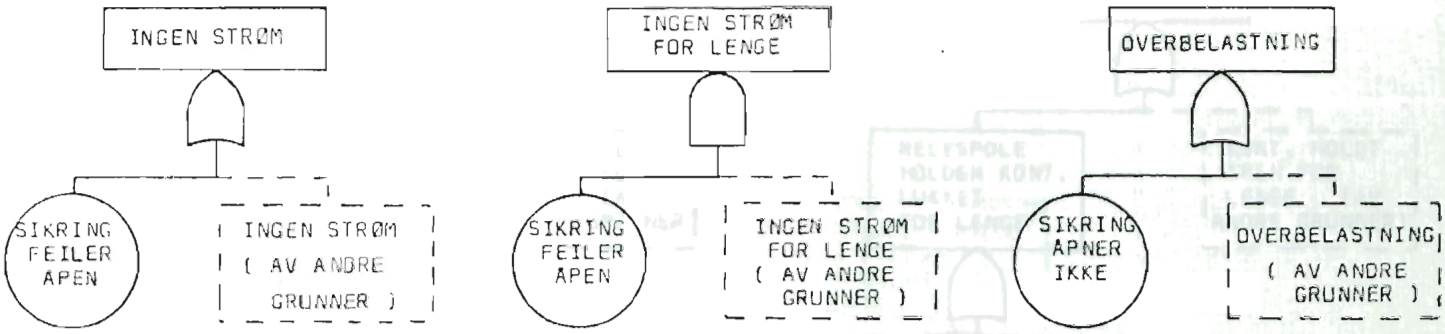
Type 1: Ikke strøm i koalisjon 3.

Type 2: Ikke strøm på grunn av relekontakten.

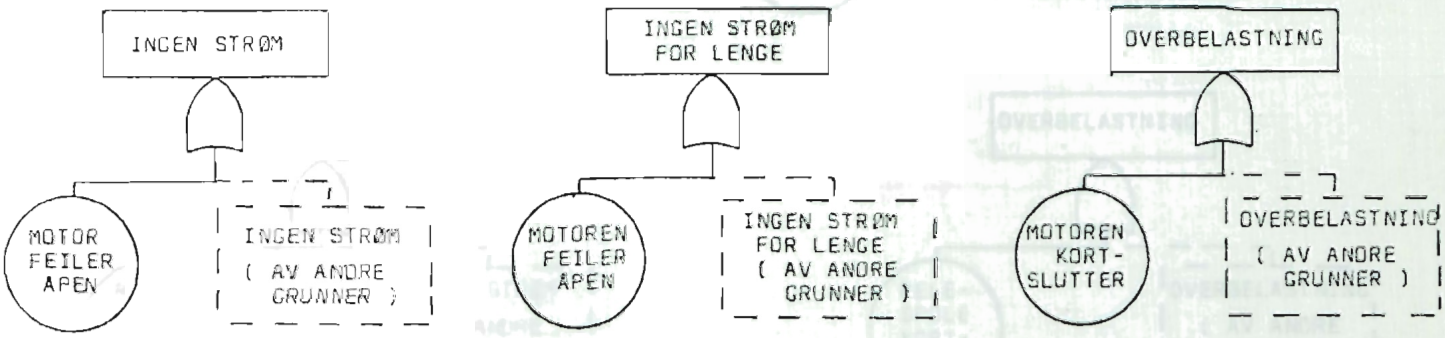
Type 2 og 3: Ikke strøm på grunn av spenningskilden.
Ikke strøm på grunn av motoren.
Ikke strøm på grunn av sikringen.

Disse beskrankningene blir ikke effektive da denne grenen av feiltreet er ferdigkonstruert.

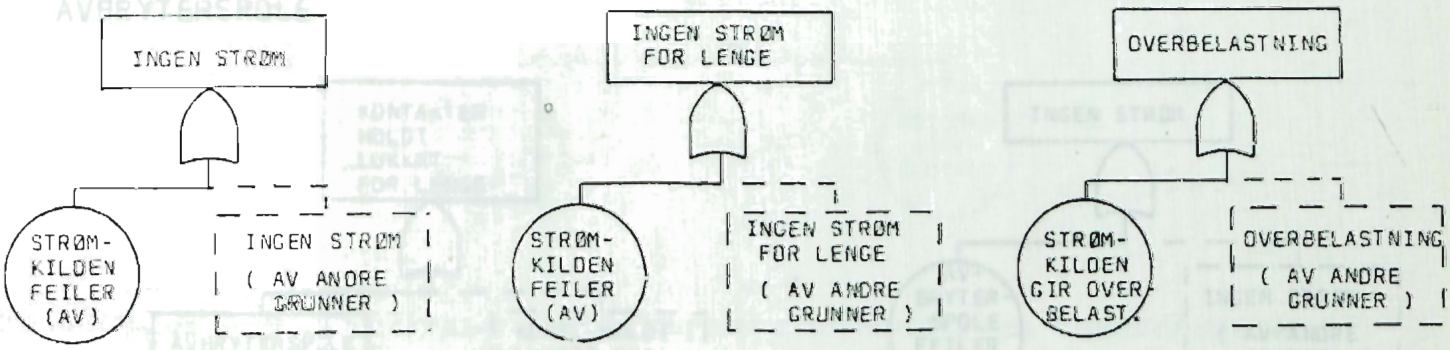
SIKRING:



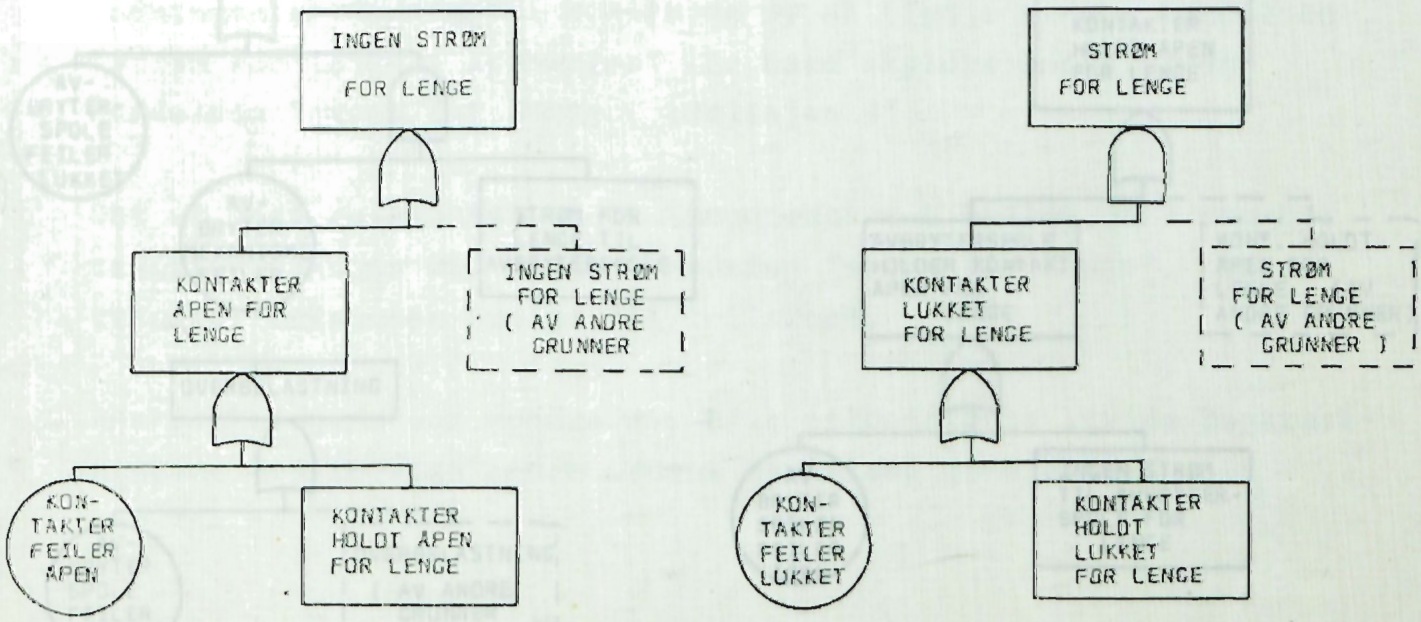
MOTOR :



STRØMKILDE :

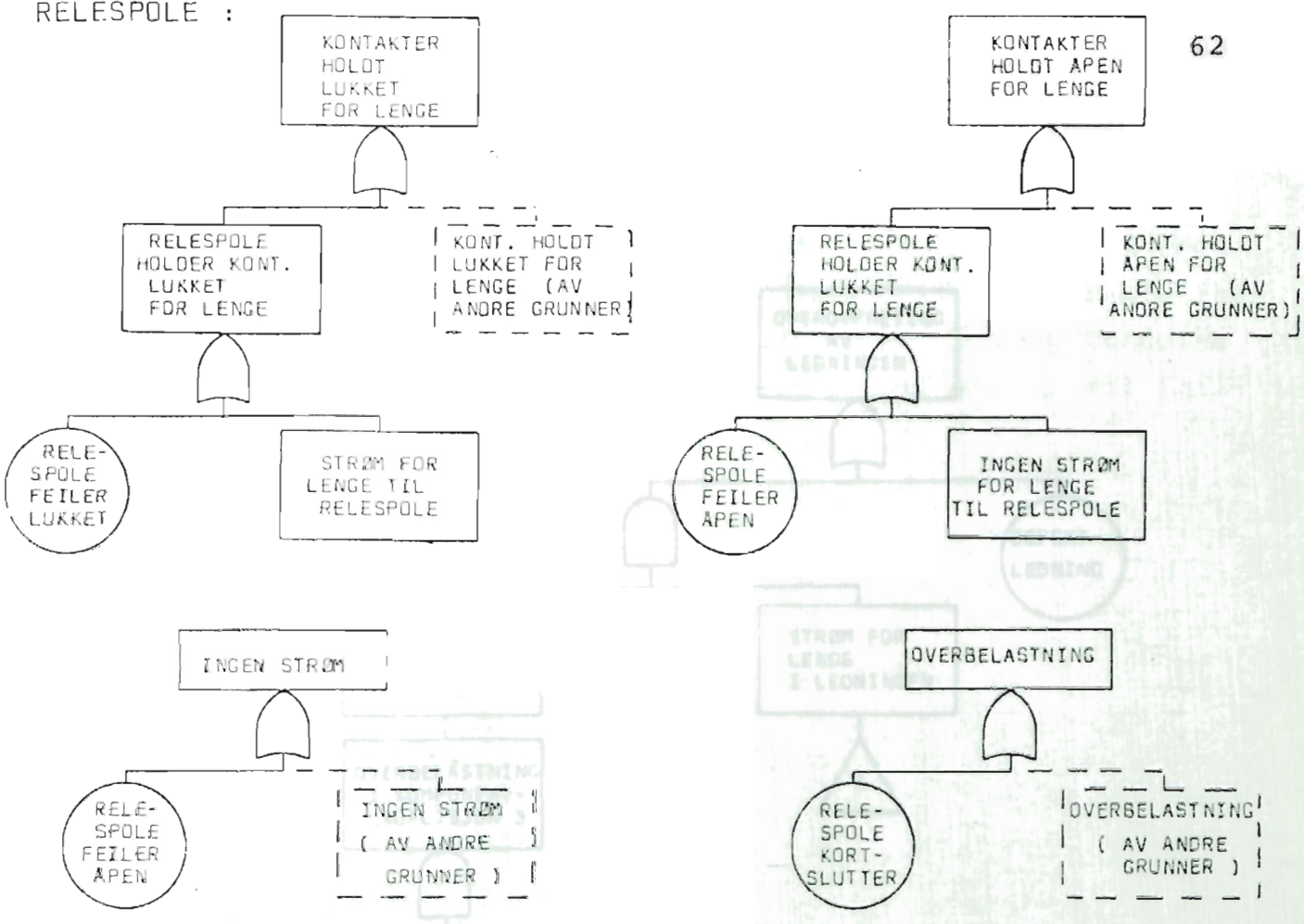


KONTAKTER (BRYTER) :

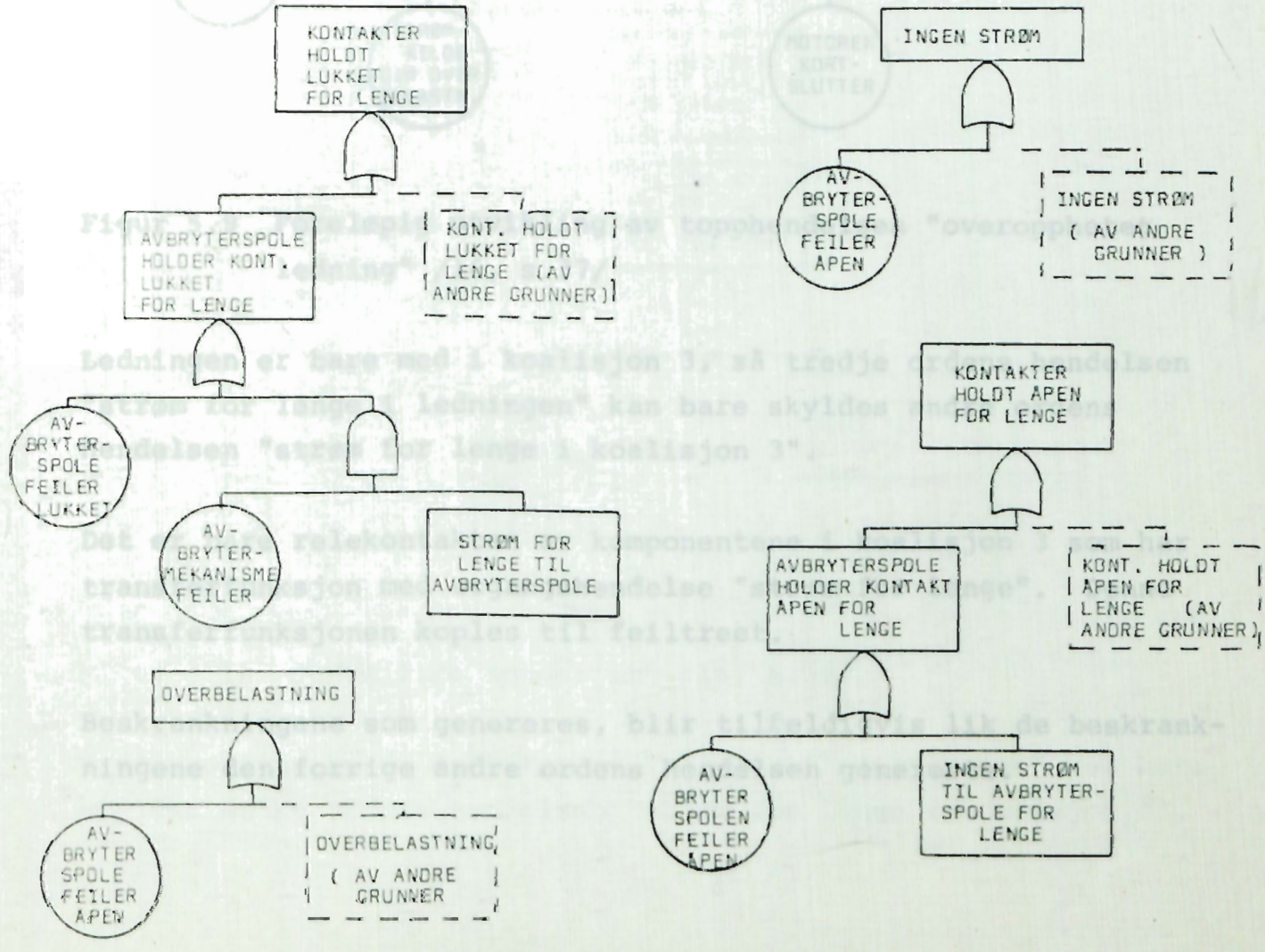


Figur 5.7 Feiltransferfunksjoner for sikring, motor, strømkilde og kontakter (bryter).

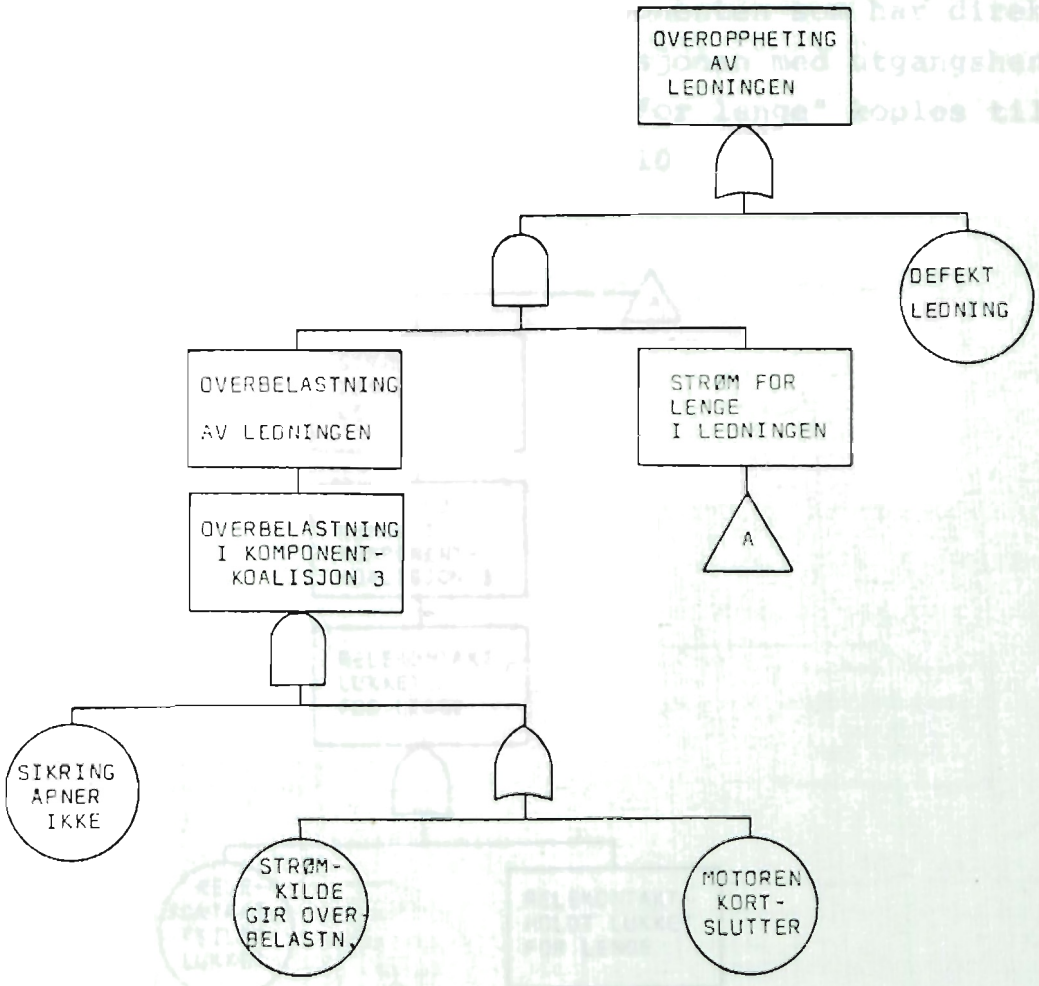
RELESPOLE :



AVBRYTERSPOLE :



Figur 5.8 Feiltransferfunksjoner for relespole og avbryterspole.



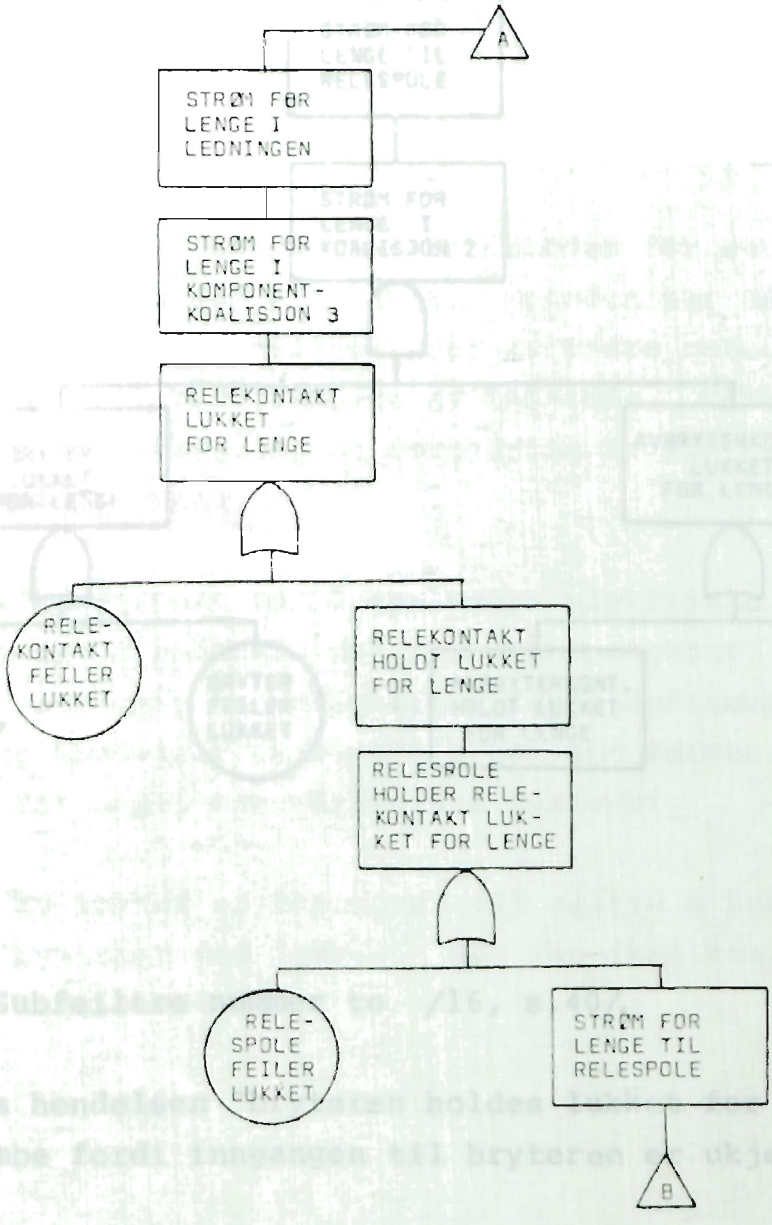
Figur 5.9 Foreløpig utvikling av topphendelsen "overopphetet ledning" /16, s.37/

Ledningen er bare med i koalisjon 3, så tredje ordens hendelsen "strøm for lenge i ledningen" kan bare skyldes andre ordens hendelsen "strøm for lenge i koalisjon 3".

Det er bare relekontakten av komponentene i koalisjon 3 som har transferfunksjon med utgangshendelse "strøm for lenge". Denne transferfunksjonen koples til feiltreet.

Beskrankningene som genereres, blir tilfeldigvis lik de beskrankningene den forrige andre ordens hendelsen genererte.

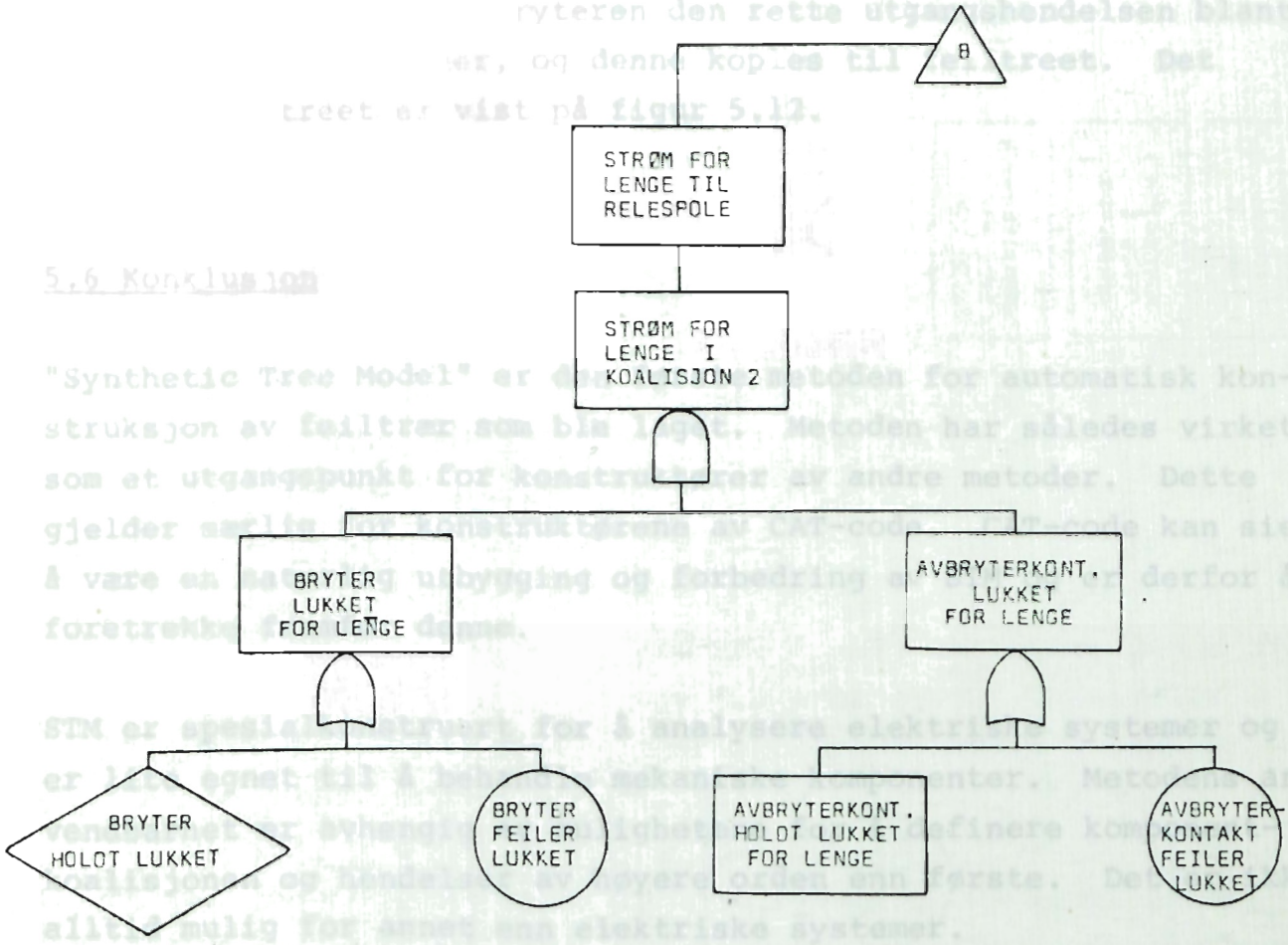
"Relekontakten holdt lukket for lenge" er en hendelse av fjerde orden. Relespolen er den eneste komponenten som har direkte inngang til relekontakten. Transferfunksjonen med utgangshendelse "relespole holder relekontakt lukket for lenge" koples til feiltreet, og vi har subtreet på figur 5.10.



Figur 5.10 Subfeiltre nummer en /16, s.39/.

Tredje ordens hendelsen"strøm for lenge til relespolen" kan bare skyldes andre ordens hendelsen "strøm for lenge i koalisjon 3".

Ved å bruke de riktige feiltransferfunksjonene til bryteren og avbryterkontakten fås subtreet på figur 5.11.



Figur 5.11 Subfeiltre nummer to /16, s.40/.

Fjerde ordens hendelsen "bryteren holdes lukket for lenge" er satt i en rombe fordi inngangen til bryteren er ukjent.

Beskrankningene som utviklingen av andre ordens hendelsen "strøm for lenge i koalisjon 2" genererer, blir ikke benyttet i den videre konstruksjon og gjengis derfor ikke her.

Fjerde ordens hendelsen "avbryterkontakt holdt lukket for lenge" søkes tilbake til avbryterspolen og dennes riktige transferfunksjon, se figur 5.8, koples til feiltreet.

Hendelsen "strøm for lenge til avbryterspole" kan bare skyldes "strøm for lenge i koalisjon 1" som er en hendelse av andre orden.

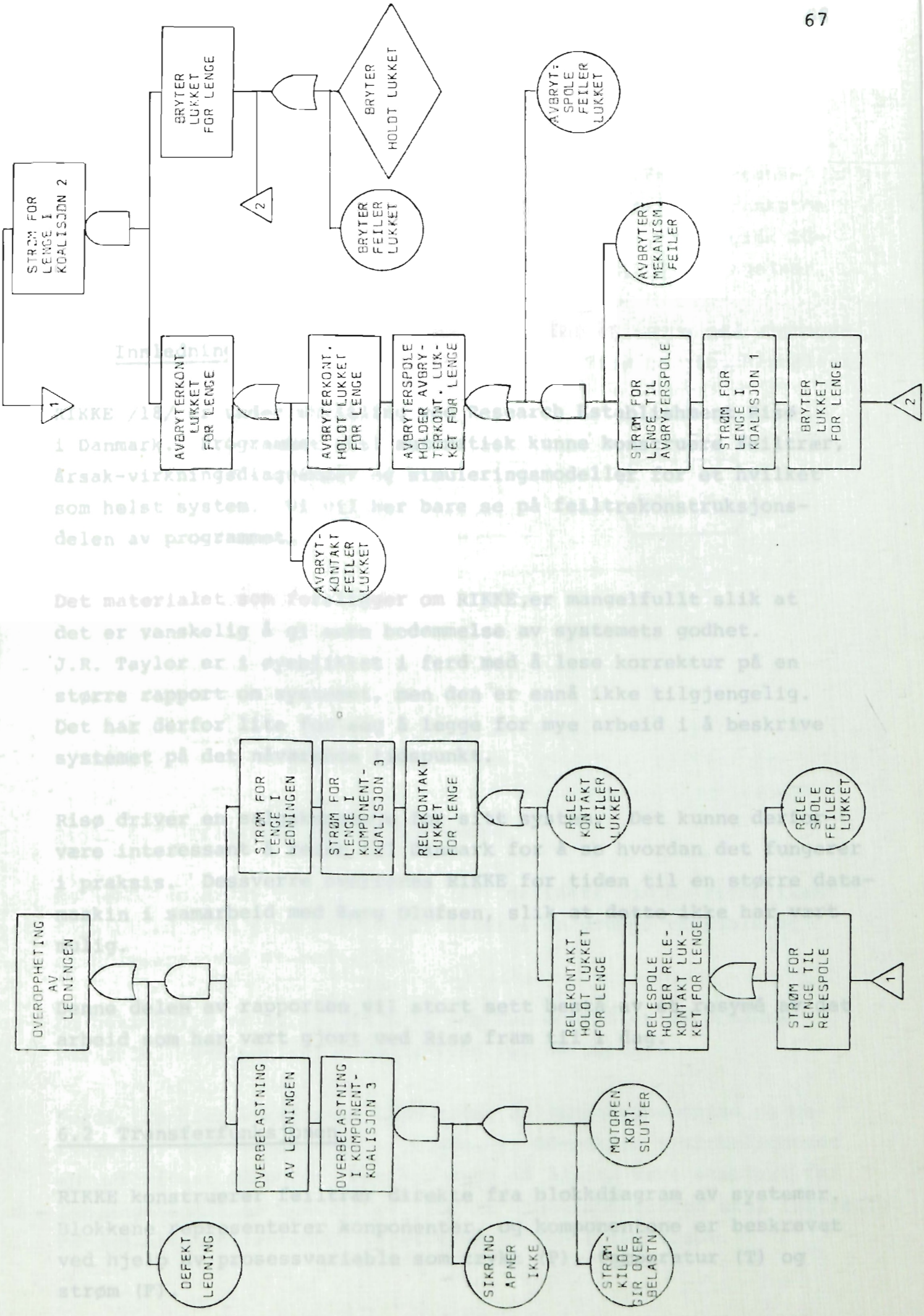
I koalisjon 1 har bare bryteren den rette utgangshendelsen blant sine transferfunksjoner, og denne koples til feiltreet. Det ferdige feiltreet er vist på figur 5.12.

5.6 Konklusjon

"Synthetic Tree Model" er den første metoden for automatisk konstruksjon av feiltrær som ble laget. Metoden har således virket som et utgangspunkt for konstruktører av andre metoder. Dette gjelder særlig for konstruktørene av CAT-code. CAT-code kan sies å være en naturlig utbygging og forbedring av STM og er derfor å foretrekke framfor denne.

STM er spesialkonstruert for å analysere elektriske systemer og er lite egnet til å behandle mekaniske komponenter. Metodens anvendbarhet er avhengig av mulighetene for å definere komponentkoalisjonen og hendelser av høyere orden enn første. Det er ikke alltid mulig for annet enn elektriske systemer.

Behandlingen av løkker er for enkel til alltid å kunne gi korrekte feiltrær for systemer med løkker. STM kan ikke takle tidsforsinkelser og tidssekvenser.



Figur 5.12 Feiltre for topphendelsen "overoppheting av ledning" /16, s.42/.

6. RIKKE

6.1 Innledning

RIKKE /18/ er under utvikling ved Research Establishment Risø i Danmark. Programmet skal automatisk kunne konstruere feiltrær, årsak-virkningsdiagrammer og simuleringsmodeller for et hvilket som helst system. Vi vil her bare se på feiltrekonstruksjons-delen av programmet.

Det materialet som foreligger om RIKKE,er mangelfullt slik at det er vanskelig å gi noen bedømmelse av systemets godhet. J.R. Taylor er i øyeblikket i ferd med å lese korrektur på en større rapport om systemet, men den er ennå ikke tilgjengelig. Det har derfor lite for seg å legge for mye arbeid i å beskrive systemet på det nåværende tidspunkt.

Risø driver en salgskampanje for sitt system. Det kunne derfor være interessant å reise til Danmark for å se hvordan det fungerer i praksis. Dessverre overføres RIKKE for tiden til en større data-maskin i samarbeid med Bang Olufsen, slik at dette ikke har vært mulig.

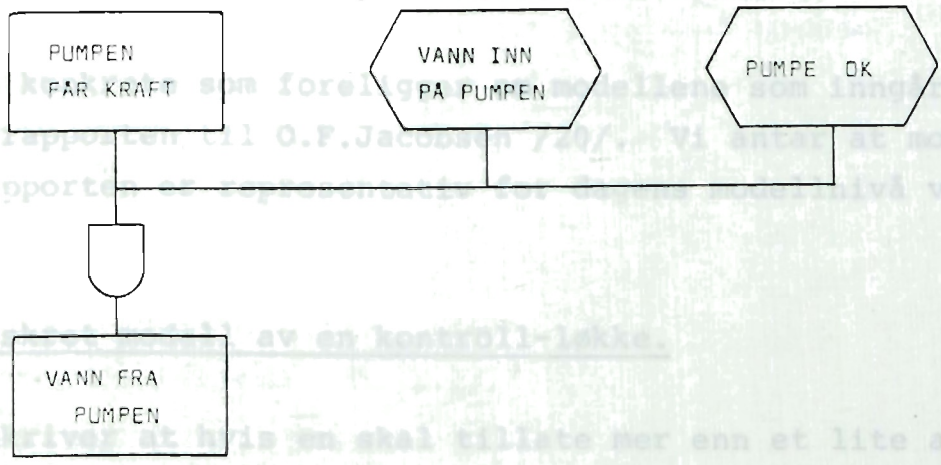
Denne delen av rapporten vil stort sett bestå av et resymé av det arbeid som har vært gjort ved Risø fram til i dag.

6.2 Transferfunksjonen

RIKKE konstruerer feiltrær direkte fra blokkdiagram av systemer. Blokkene representerer konponenter, og komponentene er beskrevet ved hjelp av prosessvariable som trykk (P), temperatur (T) og strøm (F).

Feilemåtene til de enkelte komponentene er definert ved transferfunksjoner som er samlet i et bibliotek. En transferfunksjon er et minifeiltre bestående av en utgangshendelse, en logisk OG-port, en inngangshendelse og en eller flere inngangsbetingelser.

På figur 6.1 er vist en transferfunksjon for en pumpe med utgangshendelse "vann fra pumpen". Feiltrærne fra Risø har topphendelsen nederst.



Figur 6.1 Transferfunksjon for pumpe.

En hendelse er av Taylor /19, s.760) definert som en signifikant forandring i en prosessvariabel eller i en gruppe variable og symboliseres ved et rektangel.

En betingelse er et logisk krav som kan oppfylles av prosessvariable. Betingelser symboliseres ved sekskanter.

Fordi transferfunksjonene inneholder en inngangshendelse og betingelser som allerede er tilstede, er OG-porten i virkeligheten en prioritert OG-port. Betingelsene må alltid være etablert før inngangshendelsen inntreffer, for at utgangshendelsen skal inntreffe. På den måten blir tidsbegrepet trukket inn i feiltreet.

Målet for Risø er å lage komponentmodeller der hendelsene er beskrevet i form av algebraiske funksjoner som inneholder prosessvariable og tiden. De hevder så langt å ha konstruert komponentmodeller som er beskrevet ved stykkevis lineære funksjoner. Vi har en mistanke om at dette gjelder for simuleringsdelen av RIKKE og ikke for feiltrekonstruksjonsdelen.

Skriftene utgitt fra Risø inneholder mange fine teorier. Men det sies lite om hvilke modeller og algoritmer som er i praktisk bruk, og hvilke som bare er på skrivebordsstadiet.

Det eneste konkrete som foreligger om modellene som inngår i RIKKE, er rapporten til O.F.Jacobsen /20/. Vi antar at modellen i denne rapporten er representativ for dagens modellnivå ved Risø.

6.3 En diskret modell av en kontroll-løkke.

Jacobsen skriver at hvis en skal tillate mer enn et lite antall diskrete verdier, blir modellen svært stor og opptar for mye plass i datamaskinens lager. Dessuten vil eksekveringstiden for datamaskinprogrammet bli uforholdsmessig lang.

Kontroll-løkken som kan være forover- eller tilbakekoplet alt etter hvilken vei hovedstrømmen går, er vist på figur 6.2.

Den diskrete modellen er basert på følgende kontinuerlige modell:

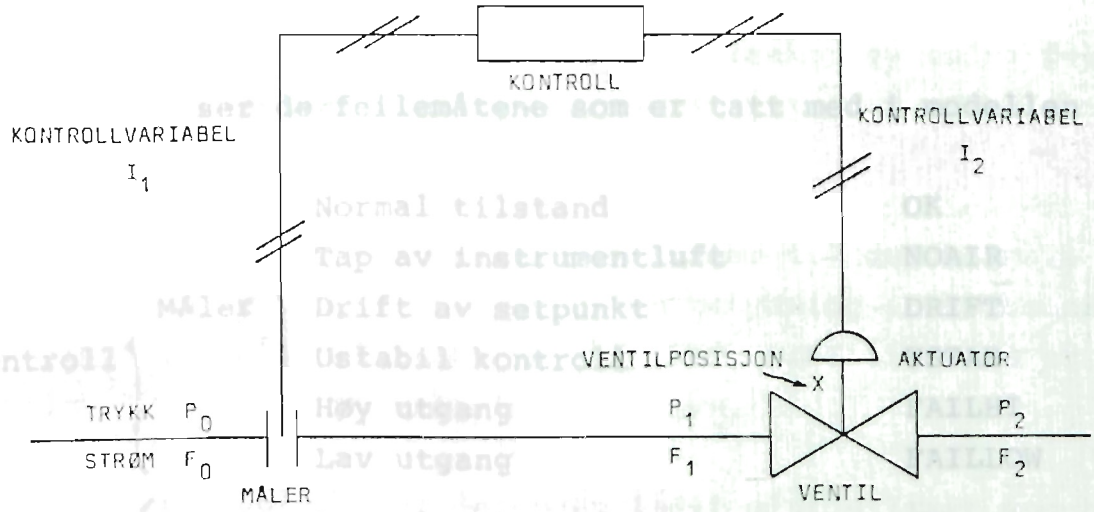
Måler: $I_1 = K_1 P_o$ eller $I_1 = K_1 F_o$

Kontroll: $I_2 = K_2 I_1$

Aktuar: $X = K_3 I_2$

Ventil: $P_1 - P_2 = K_4 F^2 X$

der K_1 til K_4 er konstanter.



Figur 6.2 Kontroll-løkke /20, s.1/.

De variable er diskretisert til nivåene vist på figur 6.3.

Kontrollbare nivåer	Normal	N
	Kompensert	COMP
	Forstyrret	DIST
Ikke-kontrollerbare nivåer	Null	Zero
	Veldig høy	VH
	Tilbakeslag	BACKW

Figur 6.3 Nivåene i den diskrete modellen.

Den normale virkemåten til løkken er beskrevet ved nivåene N, COMP og DIST. En liten forstyrrelse (avvik fra det normale) betegnes med DIST.

Kontrollvariabelen I vil kompenserer for forstyrrelsen ved å gå over til nivået COMP. Deretter vil kontrollvariabelen og prosessvariabelen sammen gå kontinuerlig over i normalnivået N.

Nivået tilbakeslag, BACKW, er oss bekjent ikke tatt med i noen av de andre feiltrekonstrukjonsmetodene vi har undersøkt. Det er bare væskestrømmen F og kontrollvariabelen I som kan ha tilbakeslag.

Figur 6.4 viser de feilemåtene som er tatt med i modellen

Kontroll	Måler	Normal tilstand	OK
		Tap av instrumentluft	NOAIR
		Drift av setpunkt	DRIFT
		Ustabil kontroll	USTAB
		Høy utgang	FAILHI
		Lav utgang	FAILLOW
Ventil		Normal tilstand	OK
		Fastlåst	SF
		Feiler åpen	SO
		Feiler lukket	SC

Figur 6.4 Feilemåter for komponentene i kontroll-løkken.

Tilstandene DRIFT og USTAB resulterer begge i en ustabil prosess. Skillet mellom dem kommer ikke fram i en diskret modell. De er tatt med for å lette overgangen til en kontinuerlig modell.

Følgende regler gjelder for diskretiseringen:

1. Bare hendelsesforløp som følger fysiske lover kan modelleres ved minifeiltrær.

2. Minifeiltrærne må være i overensstemmelse med mulige kausale forhold. (Feiltrærne kan bare gi uttrykk for de påvirkninger mellom variable som også eksisterer i virkeligheten).

3. Minifeiltrærne beskriver kontrollerbare forstyrrelser i prosessen og den påfølgende kompensering via kontroll-løkker.

4. Minifeiltrærne beskriver overgangen til en ikke-kontrollerbar tilstand og videre til shut-down eller andre større konsekvenser.

5. Minifeiltrærne beskriver feil forårsaket av andre feil, men ikke overganger fra en komponentfeil til en annen uavhengig feilemåte for komponenten.

Overgangen fra ikke-kontrollerbare nivåer til det normale er ikke tatt med i modellen, da de har liten betydning i risikoanalysen. Overgangene BACKW til ZERO, VH til ZERO, DIST til ZERO er tatt med for å beskrive sikkerhets-shut-down.

Ut fra disse reglene har Jacobsen laget minifeiltrær som beskriver de enkelte komponentene i systemet. Det er for omfattende å ta med disse her, men som et eksempel viser vi feiltrærne for måleren, se figur 6.5a og b.

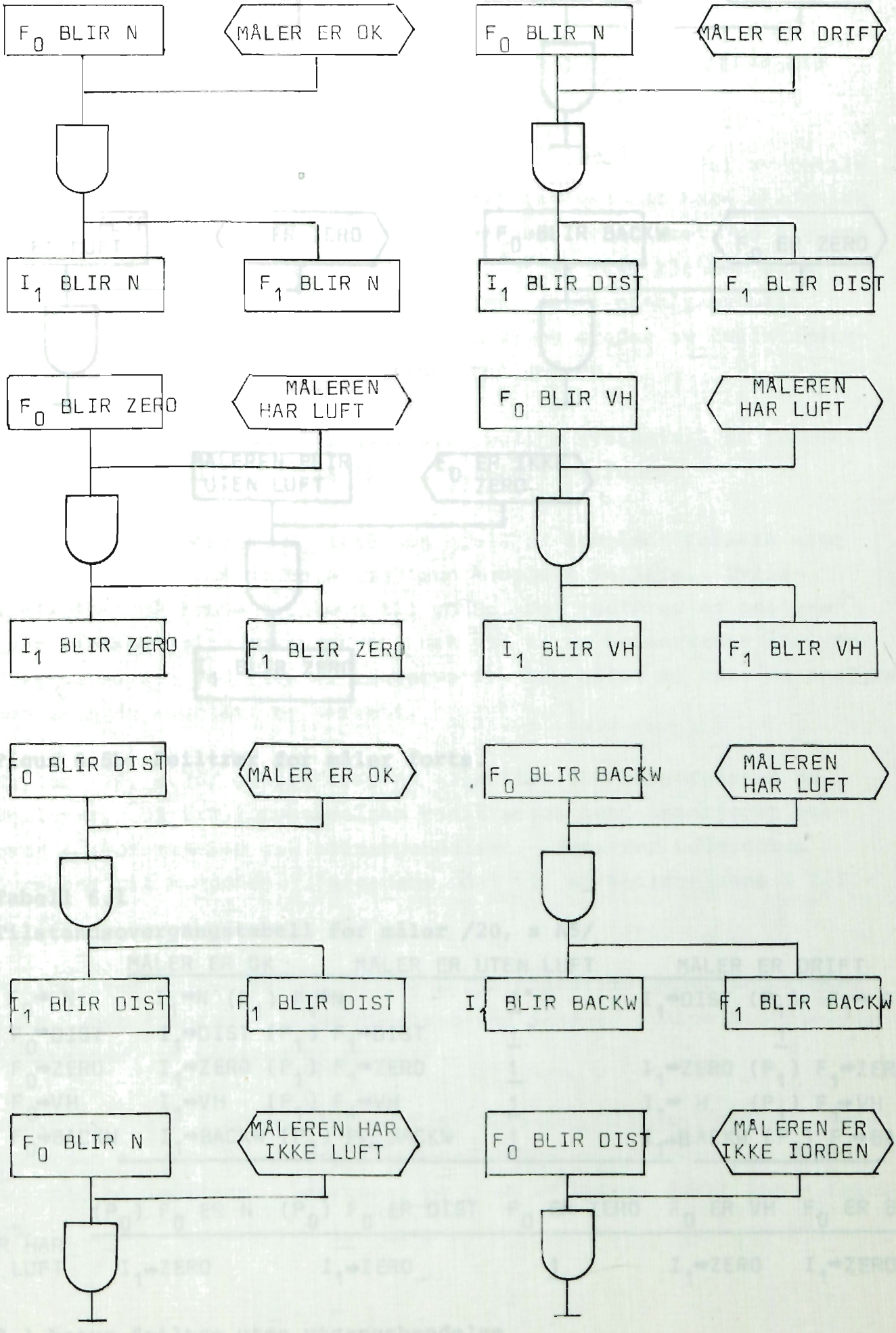
Feiltrær uten utgangshendelser viser at tilstandsforandring i den inngangsvariable ikke medfører forandring i noen utgangsvariable ved betingelsene gitt i feiltreet. Vi ser også at minifeiltrærne kan ha flere utganger. Dette vil ikke resultere i porter med flere utganger i det endelige feiltreet. Men konsekvensene av de utgangshendelsene som ikke brukes direkte i feiltreet må undersøkes nærmere når systemet inneholder forover- og tilbakekoplinger.

Minifeiltrærne kan mer kompakt og oversiktlig gjengis i en såkalt tilstandsovergangstabell, se tabell 6.1.

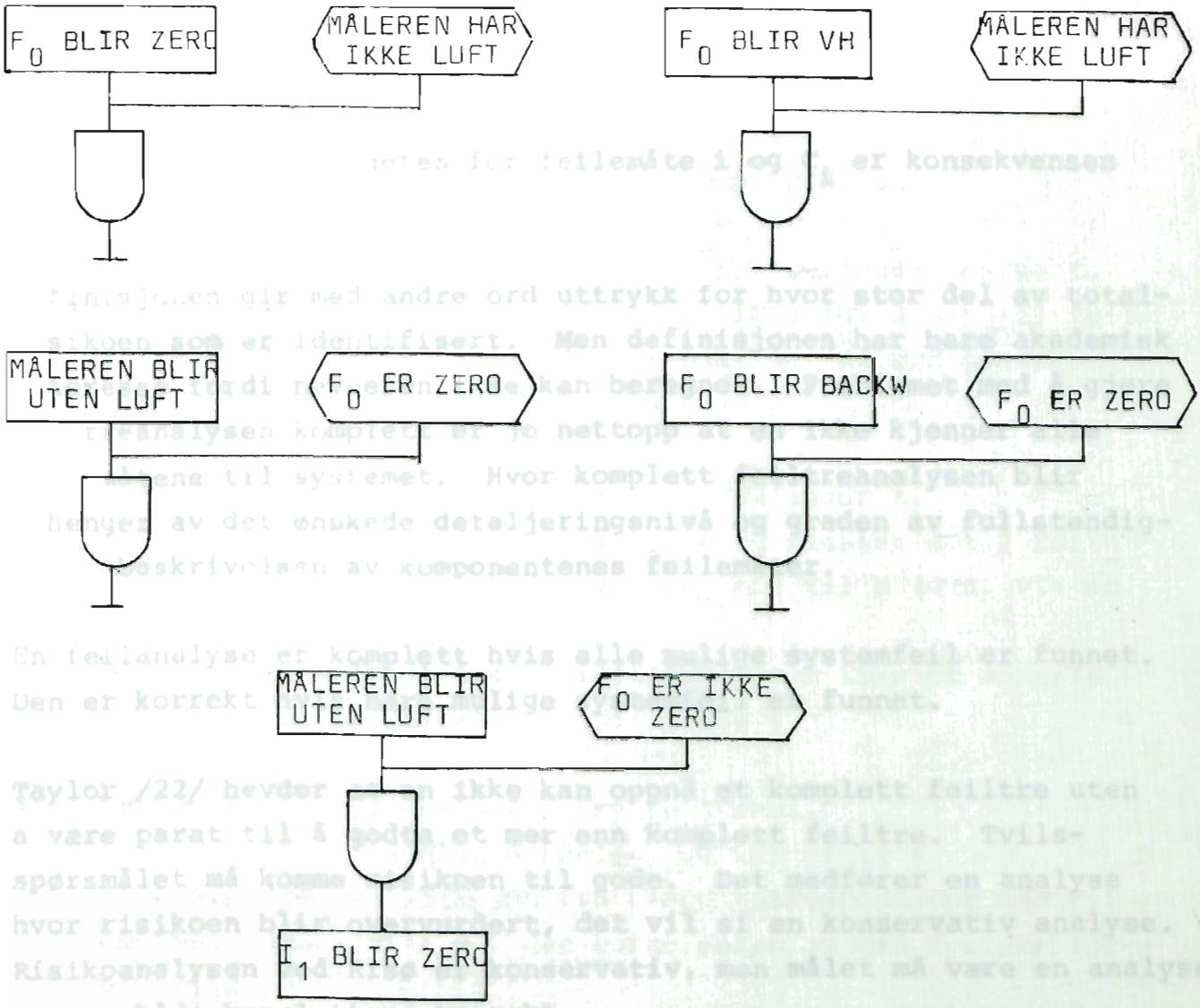
6.4 Feiltrekonstruksjonsalgoritme

J.R.Taylor /21/ er opptatt av hvor fullstendig forskjellige analysemetoder er: Han definerer:

$$\text{Grad av fullstendighet} = \frac{\sum_{i = \text{alle identifiserte feilemåter}} p_i C_i}{\sum_{i = \text{alle mulige feilemåter}} p_i C_i}$$



Figur 6.5a Feiltrær for måler. /20, s.A8 og A9/



Figur 6.5b Feiltrær for måler forts.

Tabell 6.1

Tilstandsovergangstabell for måler /20, s A5/

	MÅLER ER OK	MÅLER ER UTEN LUFT	MÅLER ER DRIFT
(P_0) F_0⇒ N	I_1⇒N (P_1) F_1⇒N	1*	I_1⇒DIST (P_1) F_1⇒ DIST
(P_0) F_0⇒DIST	I_1⇒DIST (P_1) F_1⇒DIST	1*	1
(P_0) F_0⇒ZERO	I_1⇒ZERO (P_1) F_1⇒ZERO	1	I_1⇒ZERO (P_1) F_1⇒ZERO
(P_0) F_0⇒VH	I_1⇒VH (P_1) F_1⇒VH	1	I_1⇒ H (P_1) F_1⇒VH
(P_0) F_0⇒BACKW	I_1⇒BACKW (P_1) F_1⇒BACKW	1	I_1⇒BACKW (P_1) F_1⇒BACKW

	(P_0) F_0 ER N	(P_0) F_0 ER DIST	F_0 ER ZERO	F_0 ER VH	F_0 ER BACK
MÅLER HAR IKKE LUFT	I_1⇒ZERO	I_1⇒ZERO	1	I_1⇒ZERO	I_1⇒ZERO

* 1 betyr feiltre uten utgangshendelse

der p_i er sannsynligheten for feilemåte i og C_i er konsekvensen av feilemåte i.

Definisjonen gir med andre ord uttrykk for hvor stor del av totalrisikoen som er identifisert. Men definisjonen har bare akademisk interesse fordi nevneren ikke kan beregnes. Problemet med å gjøre feiltreanalysen komplett er jo nettopp at en ikke kjenner alle feilemåtene til systemet. Hvor komplett feiltreanalysen blir avhenger av det ønskede detaljeringsnivå og graden av fullstendighet i beskrivelsen av komponentenes feilemåter.

En feilanalyse er komplett hvis alle mulige systemfeil er funnet. Den er korrekt hvis bare mulige systemfeil er funnet.

Taylor /22/ hevder at en ikke kan oppnå et komplett feiltre uten a være parat til å godta et mer enn komplett feiltre. Tvilsspørsmålet må komme risikoen til gode. Det medfører en analyse hvor risikoen blir overvurdert, det vil si en konservativ analyse. Risikoanalysen ved Risø er konservativ, men målet må være en analyse som er både komplett og korrekt.

Taylor /21, s.20/ presenterer en algoritme for konstruksjon av feiltrær. Ut fra topphendelsen konstrueres hendelseskjeder bakover i systemet ned til primærhendelser. Deretter undersøkes årsakene til komponenttilstandene, det vil si betingelsene i feiltreet.

Betingelser som tilsvarer mulige begynnelsestilstander undersøkes ikke. Men årsakene til betingelser som ikke er mulige begynnelsestilstander, må undersøkes nærmere.

Punktene 1 til 5 er en ordinær feiltrekonstruksjonsalgoritme, mens punktene 6, 7 og 8 er ledd i undersøkelsen av årsakene til ikkenormale betingelser. Lagrene A og Q er av typen: først inn - først ut.

1. Start med en topphendelse E i en komponent C.

2. Finn alle minifeiltrærne for C som har utgangshendelse E.
Hvis det ikke eksisterer slike feiltrær, føy usann til feiltreet som årsak til E og gå til punkt 6. Ellers plasseres minifeiltrærne på lager A.

3. Stopp hvis det ikke er minifeiltrær på lager A.
Ta et av minifeiltrærne fra lager A og plasser det i feiltreet som en av de alternative årsakene til E (dvs. via en ELLER-port).
Føy de betingelsene fra minifeiltreet som ikke er normale - eller begynnelsestilstander,til liste B.

4. Bestem inngangshendelsen F fra feiltreet.
Hvis det er en spontan hendelse, gå til punkt 6.
Finn den fysiske utgangen fra C som korresponderer med F, og finn komponenten D i den andre enden av den fysiske forbindelsen.
Finn deretter utgangshendelsen G i D som fører til F.

5. Gjør hendelse G om til E og komponent D til C og gå til punkt 2.

6. Hvis det ikke er noen betingelser på liste B, gå til punkt 8.
Generer alle mulige nye permutasjoner av betingelser på liste B med betingelser på liste P som opprettholder rekkefølgen til betingelsene på liste P. Gjem permutasjonene på lager Q.

7. Hvis det ikke er permutasjoner på lager Q, gå til punkt 3.
Ta en permutasjon fra lager Q og plasser den på liste P.

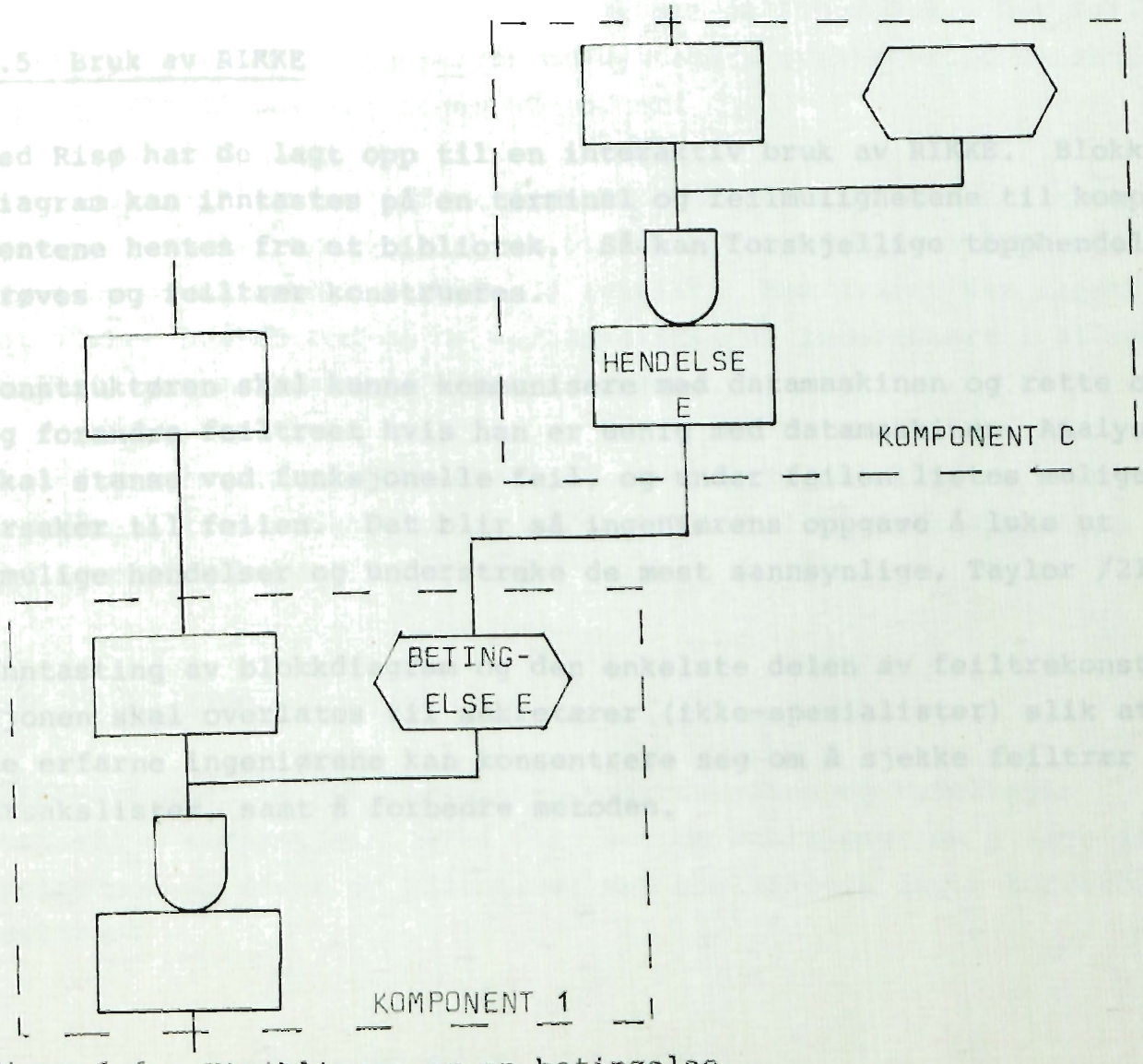
8. Hvis det ikke er noen betingelser på liste P, gå til punkt 7.
Ta den første betingelsen fra liste P og finn den korresponderende komponent C og hendelse E og gå til punkt 2.

Når en gren av feiltreet er ferdig utviklet, og en kommer tilbake til punkt 3 i algoritmen for å hente et nytt minifeiltre, er ikke utgangshendelsen til feiltreet nødvendigvis identisk med den hendelsen E som er definert i øyeblikket. Minifeiltreet kan gjerne ha en utgangshendelse som svarer til en hendelse lenger ned i feiltreet, det vil si en tidligere E.

Det står ikke eksplisitt uttrykt i algoritmen hvilken ELLER-port minifeiltreet skal koples til. Problemet er forsåvidt lite, og det kan løses for eksempel ved hjelp av en tellevariabel.

Enhver betingelse som ikke er normal eller en begynnelsesbetingelse, må være forårsaket av en tidligere hendelse.

I punkt 8 i algoritmen finnes den komponent C med utgangshendelse E som fører til den første betingelsen på liste P, se figur 6.6.



Figur 6.6 Utviklingen av en betingelse.

OG-portene i feiltreet oppstår altså ved utvikling av ikke-normale betingelser. Ved undersøkelse av årsakene til ikke-normale betingelser må algoritmen undersøke alle permutasjoner av hendelser (punkt 6). Det er nemlig ikke likegyldig hvilken rekkefølge hendelsene inntreffer i. Særlig gjelder dette for foroverkoplinger.

Skal algoritmen ta hensyn til lokale inkonsistenser som oppstår på grunn av løkker, må den modifiseres i punkt 3, se vedlegg 1.

De lagrede komponenttilstandene slettes når en kommer til punkt 6 i algoritmen.

Før treet er ferdig, redigeres Og-porter og ELLER-porter med sanne og usanne innganger på samme måte som de andre metodene.

6.5 Bruk av RIKKE

Ved Risø har de lagt opp til en interaktiv bruk av RIKKE. Blokkdiagram kan inntastes på en terminal og feilmulighetene til komponentene hentes fra et bibliotek. Så kan forskjellige topphendelser prøves og feiltrær konstrueres.

Konstruktøren skal kunne kommunisere med datamaskinen og rette opp og forandre feiltreet hvis han er uenig med datamaskinen. Analysen skal stanse ved funksjonelle feil, og under feilen listes mulige årsaker til feilen. Det blir så ingeniørens oppgave å luke ut umulige hendelser og understreke de mest sannsynlige, Taylor /21/.

Inntasting av blokkdiagram og den enkelste delen av feiltrekonstruksjonen skal overlates til sekretærer (ikke-spesialister) slik at de erfarne ingeniørene kan konsentrere seg om å sjekke feiltrær og årsakslister, samt å forbedre metoden.

På denne måten regner en med å kunne gjøre feiltrekonstruksjonen så rask og effektiv at en kan følge opp forandringer under konstruksjonen av et system med nye feiltrær.

Alt dette er vel og bra, men duger systemet i praksis? De feiltrærne som er presentert av Risø er ikke overbevisende.

Taylor /20, s. 23a og b/ viser et feiltre for mindre temperaturforstyrrelser i Lapp og Powers salpetersyrekjølesystem, se figur 6.7.
Feiltreet er svært merkelig og inneholder blant annet en løkke som er helt uforståelig for oss.

Vi har også fått tilsendt et feiltre fra Taylor som skal vise instabilitetsproblemer ved Lapp og Powers system. Når alle normale tilstander er fjernet, ser feiltreet ut som på figur 6.8. Det feiltreet gir ikke mye informasjon om systemets svakheter og inneholder bare et fåtall feil og ingen OG-porter!

Vi skrev til Risø for å få tilsendt transferfunksjoner for komponentene i kjølesystemet slik at vi kunne konstruere et feiltre ved hjelp av algoritmen i foregående avsnitt. Men svaret var negativt. Det virker nesten som om de ved Risø ikke er interessert i at noen skal gå systemet nærmere etter i sømmene.

De feiltrærne som er presentert er uoversiktlige på grunn av alle de normale tilstandene som er med. Disse tilstandene gjør også feiltreet inkoherent.

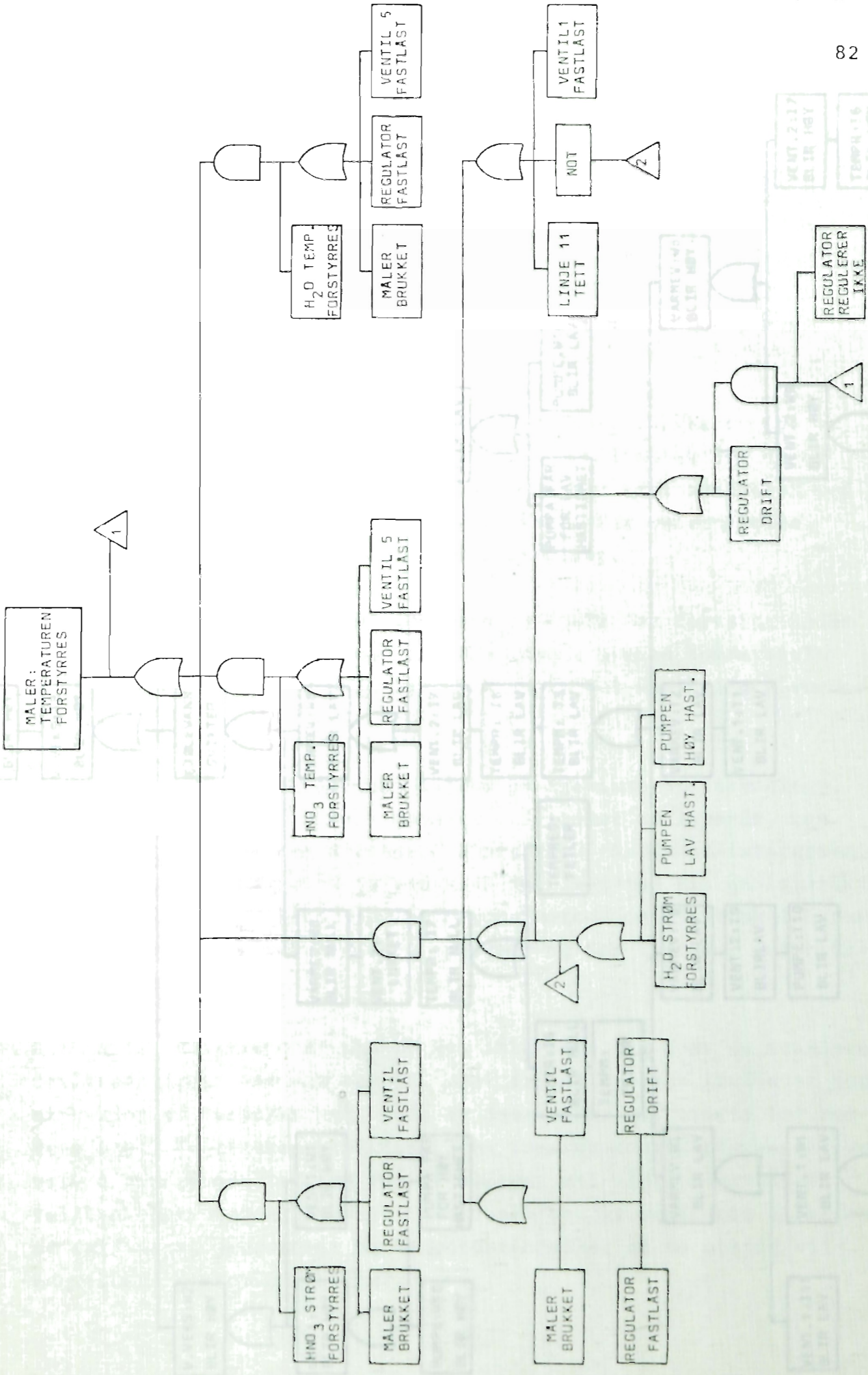
6.6 Konklusjon

RIKKE-systemet er det ferskeste i undersøkelsen og tydeligvis fortsatt i støpeskjeen. Ved Risø har de ambisjoner om å lage et system som er raskt og effektivt, men som likevel lager korrekte feiltrær.

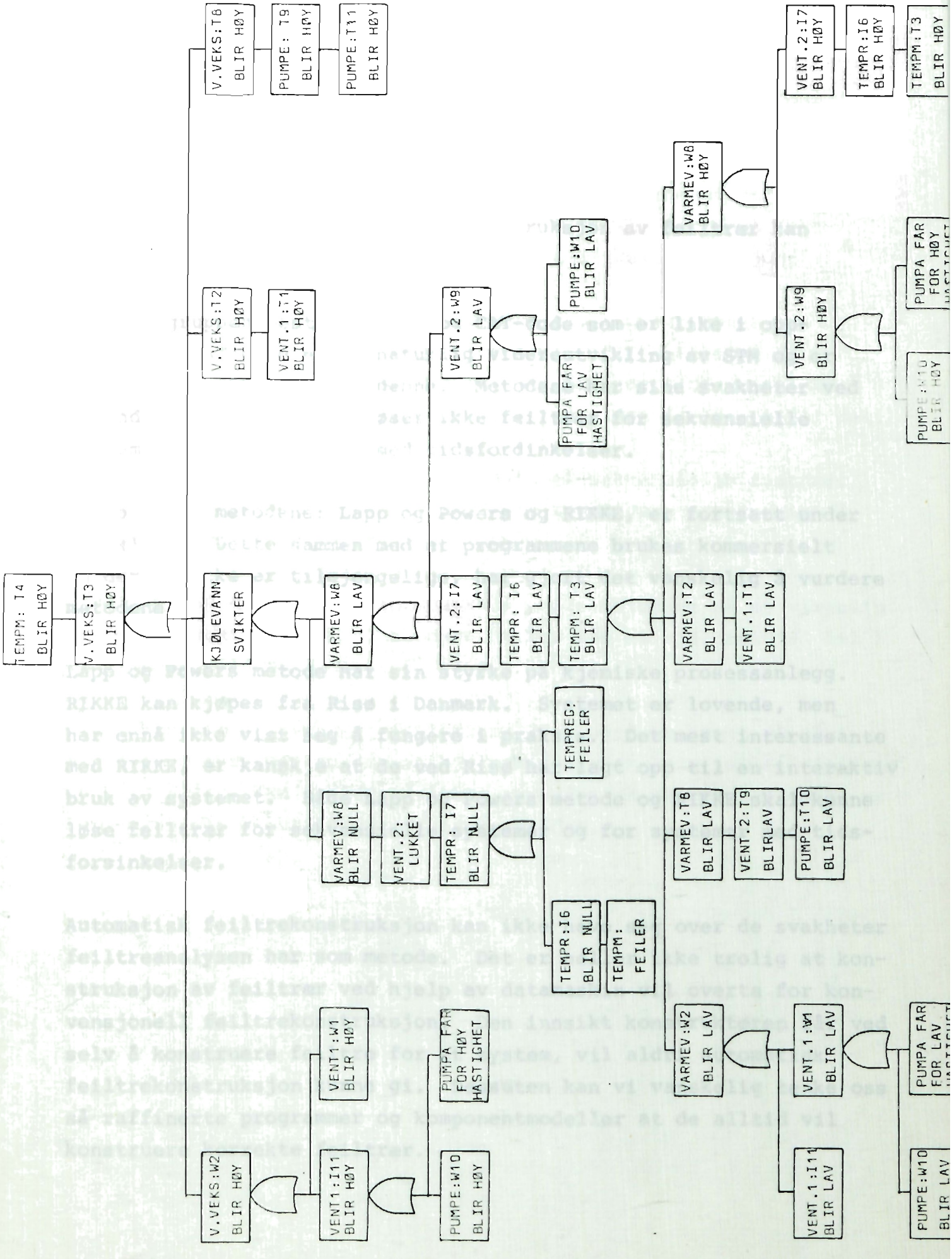
RIKKE skal kunne løse feiltrær for tidssekvenser og ta hensyn til tidsforsinkelser. Dessuten er målet at komponentmodellene skal være beskrevet ved hjelp av likninger.

Faren er at teorien bak systemet blir så komplisert at det ikke er anvendelig på virkelige systemer. Pr. i dag er vi ikke overbevist om at metoden har noen fortrinn framfor de andre.

Det mest positive er kanskje at RIKKE kan brukes interaktivt. Det muliggjør en kommunikasjon mellom menneske og maskin som kan vise seg å gi resultater. Men konklusjonen må bli at RIKKE til nå ikke har vist seg praktisk anvendelig.



Figur 6.7 Feiltre for mindre temperaturforstyrrelser i Lapp og Powers salpetersyrekjølesystem /21, s.23a og 23b/



Figur 6.8 Feiltre for Lapp og Powers kjølesystem /23/. Nummereringen følger den på figur 4.6. T = temperatur, W = væskestrøm, I = kontrollvariabel.

7. KONKLUSJON

De fire metodene for automatisk konstruksjon av feiltrær kan deles inn i to grupper.

Den ene gruppen består av STM og CAT-code som er like i oppbygging. CAT-code er en naturlig videreutvikling av STM og er nok å foretrekke framfor denne. Metodene har sine svakheter ved behandling av løkker og løser ikke feiltrær for sekvensielle systemer og for systemer med tidsfordinkelser.

De to andre metodene: Lapp og Powers og RIKKE, er fortsatt under utvikling. Dette sammen med at programmene brukes kommersielt og derfor ikke er tilgjengelige, har gjort det vanskelig å vurdere metodene.

Lapp og Powers metode har sin styrke på kjemiske prosessanlegg. RIKKE kan kjøpes fra Risø i Danmark. Systemet er lovende, men har ennå ikke vist seg å fungere i praksis. Det mest interessante med RIKKE, er kanskje at de ved Risø har lagt opp til en interaktiv bruk av systemet. Både Lapp og Powers metode og RIKKE skal kunne løse feiltrær for sekvensielle systemer og for systemer med tidsforsinkelser.

Automatisk feiltrekonstruksjon kan ikke heve seg over de svakheter feiltreanalysen har som metode. Det er heller ikke trolig at konstruksjon av feiltrær ved hjelp av datamaskin vil overta for konvensjonell feiltrekonstruksjon. Den innsikt konstruktøren får ved selv å konstruere feiltre for et system, vil aldri automatisk feiltrekonstruksjon kunne gi. Dessuten kan vi vanskelig tenke oss så raffinerte programmer og komponentmodeller at de alltid vil konstruere korrekte feiltrær.

Men automatisk feiltrekonstruksjon kan bli et viktig hjelpemiddel i konstruktørens arbeid. Han kan for eksempel benytte datamaskinens hurtighet til å konstruere feiltrær for forskjellige topphendelser og bruke feiltrærne til å lære systemet bedre å kjenne. Vi tror at en slik interaktiv bruk vil være den mest interessante anvendelsen av automatisk feiltrekonstruksjon.

Det er vesentlig at algoritmene som brukes i programmene er korrekte og så omfattende som mulig. Men like viktig er de komponentmodellene som ligger til grunn for konstruksjonen. Er ikke komponentmodellene fullstendige, kan aldri feiltreet bli det.

Vi har i denne undersøkelsen ikke sett på sekvensielle systemer og systemer med tidsforsinkelser. Det kunne være interessant å de nærmere på disse typer av systemer.

Videre ville det være av interesse å prøve metodene på et virkelig system. Eventuelt kunne metodene brukes som et hjelpemiddel til å konstruere feiltrær for hånd.

Til slutt vil vi nevne at feiltreanalysen også er brukt på administrative systemer og som et hjelpemiddel for ledelsesbeslutninger, se MORT /24/. Kan feiltreanalysen anvendes på slike problemer, er det en vesentlig utvidelse av bruksområdet.

8. REFERANSER

/1/ Mearns, A.B., "Fault Tree Analysis: The Study of Unlikely Events in Complex Systems", System Safety Sympositum, juni 1965, Seattle: The Boeing Company.

/2/ Haasle, D.F., "Advanced Concepts in Fault Tree Analysis", University of Washington og the Boeing Company, Seattle, 1965.

/3/ U.S. Atomic Energy Commission,"Reactor Safety Study" (Rasmussen-rapporten), DRAFT WASH-1400, Washington D.C., August 1974.

/4/ Christensen, H., "Systemer og systemmodeller", kompendium i maskindeler hovedfag, NTH, Trondheim, 1978.

/5/ Fussell, J.B., Powers, G.J. og Bennetts, R.G., "Fault Trees - A State of the Art Discussion", IEEE Transactions on Reliability, Vol. R-23, No. 1, April 1974, s. 51 - 55.

/6/ Vesely, W.E. og Narum, R.E., "PREP and KITT Computer Codes for the Automatic Evaluation of a Fault Tree", Idaho Nuclear Corporation, Idaho, 1970.

/7/ Barlow, R.E. og Proschan, F., "Statistical Theory of Reliability and Life Testing, Probability Models", Holt, Rinehart and Winston, Inc., USA, 1975.

/8/ Gjerstad, T. og Wiencke, P.M., "Vurdering av pålitelighet til styresystem på store skip", Hovedoppgave, NTH, Trondheim, 1978.

/9/ Salem, S.L., Apostolakis, G.E. og Okrent, D., "A Computer - Oriented Approach to Fault-Tree Construction", University of California, Los Angeles, April 1976.

/10/ Wu, I.S., Salem, S.L. og Apostolakis, G.E., "Decision-Table Development for Use with the CAT-Code for the Automated Fault-Tree Construction", University of California, Los Angeles, Januar 1977.

/11/ Lapp, S.A. og Powers, G.J., "Computer-aided Synthesis of Fault-trees", IEEE Trans. on Reliability, Vol. R-26, April 1977. s. 2 - 13.

/12/ Wu, I.S., Salem, S.L. og Apostolakis, G.E., "The Use of Decision Tables in The Systematic Construction of Fault Trees", i /25/, s. 800 - 824.

/13/ Lapp, S.A. og Powers, G.L., "The synthesis of Fault Trees", i /25/ s. 778 - 799.

/14/ Powers, G.J. og Tompkins, F.C., "Fault Tree Synthesis for Chemical Processes", AICLE Journal, Vol. 20. No. 2., Mars 1974, s. 376 - 387.

/15/ Lambert, H.E., "Comments on the Lapp-Powers: Computer-aided Synthesis of Fault Trees", Lawrence Licermore Lab., Livermore, Oktober 1978.

/16/ Fussell, J.B., "Synthetic Tree Model - A Formal Methodology for Fault Tree Construction", Aersjet Nuclear Company, Idaho, Mars 1973.

/17/ Fussell, J.B., "Fault Tree Analysis - Concepts and Techniques", i: Generic Techniques in Systems Reliability Assessment, redigert av Henley, E.J. og Lynn, J.W. Series E: Applied Science No. 5 i Nato Advanced Study Institutes Series. Noordhoff International Publishing, Leyden, 1976, s. 133 - 162.

/18/ Olsen, J.V., Lind, M. og Taylor, J.R., "RIKKE - A Computer Program for Automatic Fault Tree, Cause - Consequence Diagram, and Simulation Model Construction", Research Establishment Risø, Roskilde, May 1978.

/19/ Taylor, J.R. og Hollo, E., "Experience with Algorithms and Automatic Failure Analysis", i /25/ s. 759 - 777.

/20/ Jacobsen, O.F., "A Discrete Model for a Control Loop", Research Establishment Risø, Roskilde, November 1978.

/21/ Taylor, J.R., "Completeness of Automatic Methods for Failure Analysis I", Research Establishment Risø, Roskilde, May 1978.

/22/ Taylor, J.R.: Foredrag om RIKKE holdt ved SINTEF, NTH den 23. februar 1979.

/23/ Taylor, J.R.: Eksempel på feiltre konstruert av RIKKE, sendt til T. Gjerstad fra Taylor.

/24/ Johnson, W.G., "MORT - The Management Oversight and Risk Tree", US Atomic Energy Commission, SAN 821 - 2, Februar 1973.

/25/ Fussell, J.B. og Burdick, G.R., redaktører av "Nuclear Systems Reliability Engineering and Risk Assessment" SIAM 1977. Society for Industrial and Applied Mathematics, Philadelphia, 1977.

VEDLEGG 1

Modifikasjon av algoritmen gitt i kapittel 6.4

For at algoritmen skal ta hensyn til lokale inkonsistenser som oppstår på grunn av løkker, må den modifiseres i punkt 3, se Taylor /21, s.21/.

I det et minifeiltre tas fra lager A, beregnes tidspunktet t for inngangshendelsen. Deretter lagres minifeiltreets betingelser som "komponenttilstanden ved tidspunkt t".

Rett før komponenttilstanden lagres sjekkes det om et minifeiltres utgangshendelse ødelegger eller forårsaker en allerede eksisterende "komponenttilstand ved tidspunkt t'", med t' større eller lik hendelsestidspunktet t.

Hvis hendelsen forårsaker tilstanden, merkes årsaken til tilstanden (betingelsen i feiltreet) med sann og betingelsen fjernes fra liste P.

Hvis hendelsen umuliggjør tilstanden, merkes årsaken til betingelsen i feiltreet med usann.

Har minifeiltreet mer enn en utgangshendelse, utvikles konsekvensene av de andre utgangshendelsene (ikke E) ved hjelp av algoritmen i figur A.1. De andre utgangshendelsene vil jo forplante seg gjennom systemet og kan forårsake nye hendelser som i sin tur kan skape eller umuliggjøre betingelser i feiltreet.

Hvis en hendelse A som er funnet ved bruk av konsekvensalgoritmen, forårsaker en lagret komponenttilstand, settes en OG-port foran tilstanden (betingelsen i feiltreet) med en sann inngang. Foran OG-porten

plasseres også de betingelsene i hendelseskjeden som er nødvendig for at hendelsen A skal inntreffe. Betingelsen selv fjernes fra liste P.

Hvis en lagret tilstand blir ødelagt av en hendelse i konsekvens-kjeden, plasseres en OG-port foran betingelsen i feiltreet, og for hver konsekvenskjede som ødelegger betingelsen, settes en ELLER-port på inngangen til OG-porten.

Inngangene til ELLER-porten er komplementet til betingelsene for at hendelsen skal inntreffe i konsekvens-kjeden. Komplementet til en tilstand C i en komponent D er alle de mulige komponenttil-stander til D unntatt C. De komplementære betingelsene plasseres på liste B.

1. Start med en hendelse X i en komponent R ved tidspunktet t.

2. Hvis R har en lagret tilstand, sjekk om denne er i overensstemmelse med X. Hvis det ikke eksisterer minifeiltrær for R med utgangshendelse X, gå til punkt 5. Finn alle minifeiltrær for R med X som utgangshendelse og plasser dem på lager A.

3. Hvis det ikke er minifeiltrær på liste A, stopp konsekvenstreutviklingen. Ta et feiltre fra liste A.

4. Hvis betingelsene i minifeiltreet er i uoverensstemmelse med en lagret komponenttilstand, kan ikke utgangshendelsen inntreffe. Gå derfor til punkt 3.
   Finn utgangshendelsen og tidspunktet for denne fra minifeiltreet. Lagre dem på liste L.
   Registrer utgangshendelsen, hendelsestidspunktet og betingelsene i minifeiltreet i hendelseskjedelageret M med pekere tilbake til hendelsen X.
   Lagre komponenttilstanden forårsaket av X sammen med komponenten.

5. Hvis det ikke er komponenter på liste L, gå til punkt 3.
   Velg den tidligste hendelsen på L og finn hvilken komponent den påvirker. Kall komponenten R og hendelsen i R kalles X.
   Gå til punkt 2.

Figur A.1 Konsekvensalgoritme.