Vind de defecte poorten

opdracht 2 INFOEXP - juni 2008

Geerten Doornenbal	3019748	gjdoorne
Pieter van Ede	3019764	pjede
Timon Snetselaar	3019977	ctsnetse

Let op: Dit systeem is gemaakt met CLIPS versie 6.241

Inleiding

In deze interessante opdracht hebben we een expert systeem gemaakt dat het opsporen van defecten in schakelingen helpt opsporen. Traditioneel is dit een tijdrovende klus. Via dit expertsysteem kan dit in aanzienlijk kortere tijd.

Gebruiksaanwijzing

Allereerst heeft u een schakeling nodig in het format zoals in de opdracht aangedragen. Het inlezen van een schakeling kan door de juiste file onderaan in de run file te zetten.

We hebben geprobeerd om de gebruiker een file te laten opgeven, en die vervolgens in te lezen. Dit werkte echter niet, omdat CLIPS geen 'deffacts' meer wil inlezen na de start van het programma. Hierdoor zouden we of de in te lezen file moeten herschrijven naar allemaal asserts, of er een gestructureerde file (XML bijvoorbeeld) van moeten maken om deze met een ingewikkeldere parser in te lezen, of gewoon de gebruiker doorverwijzen naar een simpele edit in het bestand 'run'. Wij hebben vanwege de eenvoud voor deze laatste optie gekozen, omdat parsen geen hoofdonderdeel van de opdracht is.

Daarna is het starten van het batch bestand voldoende om dit programma in werking te stellen:

```
(batch run)
```

Zoals u opmerkt is al dit enigzins omslachtig. Echter het opstartproces was niet het speerpunt van deze opdracht en CLIPS is ook geenszins een programmeer taal die dit makkelijk maakt. De vragen (de interface) die daarna volgt is wel straightforward. In een concrete implementatie zou er een interface in bijvoorbeeld Java om dit expertsysteem liggen, die dit soort opstarthandelingen automatiseerd.

Aanpak

Direct bij onze basis implementatie hebben we gekozen voor een ambitieuze aanpak. Onderdeel hiervan is onder andere het gebruik van het Certainty Factor-netwerk per Gate per toestand. Hierover kunt u meer lezen bij het kopje "features".

Via deze CF-netwerken wordt steeds gezocht naar de meest waarschijnlijke fout. Hierna wordt de gebruiker gevraagd om de desbetreffende gate te testen. Aan de hand van dit antwoord worden de waarden in het CF-netwerk aangepast en het netwerk opnieuw doorgerekend. Ook kunnen netwerken voor bepaalde toestanden van gates worden verwijderd als een antwoord deze toestand uitsluit.

Functie per bestand

Waar vind ik wat? Hier vindt u een overzicht.

schakeling.clp
Dit bestand bevat de schakeling die getest gaat worden. Gates & invoer/uitvoer worden hierin expliciet opgenomen. De 'lines' die de gates verbinden worden hier impliciet in opgenomen. Dit heeft als gevolg dat de 'lines' alleen te achterhalen zijn door de connecties tussen gates na te gaan.

gate_information.clp
Alle nodige facts over de gates wordt verschaft in dit bestand. Allereerst de formats waarin dit wordt genoteerd. Verder ook alle mogelijke fouten en de kans dat deze fout voorkomt. Het is belangrijk om de gevolgen van een fout te weten. Deze informatie vindt u ook in gate_information. Verder bevat dit bestand het format hoe de schakelingen te noteren.

main.clp
Hoofdbestand om het programma in werking te stellen en definieert de cf-nodes. Hier wordt beschreven hoe het CF-netwerk moet worden opgebouwd en hoe het CF-netwerk moet worden doorgerekend.

run
Batch bestand om alles te runnen. Let op! Hier moet de schakeling als laatste worden toegevoegd.

parser.clp
Het parsen van de invoer en functies voor communicatie met de gebruiker.

domain.clp
Hier zit het hart van de berekeningen. Het te testen component wordt bepaald. De test-resultaten worden hier verwerkt en aangegeven welke CF-netwerken moeten worden aangepast.

A priori defecten

De a priori waarschijnlijkheden van de defecten zoals door ons gekozen.

Poort	**Defect**	**A priori**
AND	correct	0.98
AND	kortsluiting	0.007
AND	geenContact	0.004
AND	signaalP	0.003
AND	signaalQ	0.003
AND	inverted	0.003
OR	correct	0.97
OR	kortsluiting	0.007
OR	geenContact	0.005
OR	signaalP	0.006

OR	signaalQ	0.006
OR	inverted	0.006
XOR	correct	0.99
XOR	kortsluiting	0.003
XOR	geenContact	0.004
XOR	signaalP	0.001
XOR	signaalQ	0.001
XOR	inverted	0.001
NOT	correct	0.97
NOT	kortsluiting	0.008
NOT	geenContact	0.008
NOT	inverted	0.014
LINE	correct	0.99
LINE	kortsluiting	0.003
LINE	geenContact	0.004
LINE	inverted	0.003

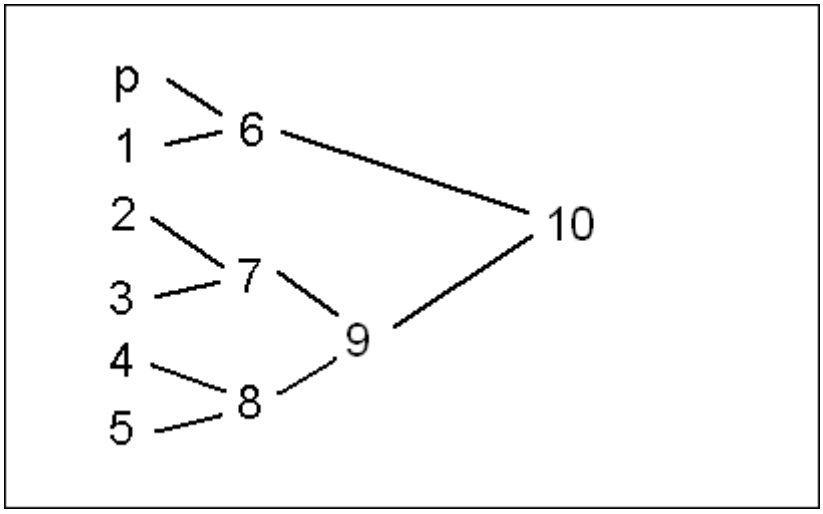
Features

De volgende features zijn opgenomen in het systeem.

CF-netwerken per gate per toestand

De implementatie voor het vinden van de meest waarschijnlijke fout bij een gate maakt gebruik van een certainty factor-netwerk. Per gate en per toestand hebben wij een certainty factor netwerk. Voor iedere test die op een gate wordt uitgevoerd, worden de kansen van de certainty factor netwerken aangepast en opnieuw doorgerekend. In onderstaand figuur ziet u de topologie voor de AND, OR en XOR poorten. Als het netwerk wordt aangemaakt, staat p op de a priori kans op dit gebrek voor deze gate en 1 – 4 op 0. Dit komt omdat 1 – 4 staan voor de testen die op een gate kunnen worden uitgevoerd en deze dus in het begin geen invloed hebben. Dit propageert dan via de hulpnodes naar node 8, die de kans bevat dat dit defect voor deze gate aan de hand is.

Als een test een gunstige uitkomst heeft voor deze toestand voor deze gate (bijvoorbeeld voor een zekere input is wijst een 1 op kortsluiting en de test geeft ook daadwerkelijk 1 aan) dan komt de node in het certainty factor netwerk voor deze test op een kans te staan. Deze kans is 1 / (aantal toestanden waarvoor de test positief was). Een rekenvoorbeeldje verduidelijkt dit wellicht: als de gebruiker 0 op beide inputs van een AND gate moest zetten en de test leverde 1 op, dan wordt de node in het certainty factor netwerk van deze AND gate voor toestand kortsluiting op ½ gezet, omdat deze testuitslag alleen gunstig was voor kortsluiting en inverted.
Hierin zijn 1, 2, 3 en 4 de mogelijke invoeren en p de a priori kans op deze toestand. 5 is hier het gevoel van de technicus die aangeeft dat er iets verdacht is aan deze gate (zoals bijvoorbeeld schroeiplekken).



Hieronder ziet u een voorbeeldrun waarin de certainty op een bepaald defect boven de threshold komt:

```
Test component number 8, set the first input to 0 and the second
input to 0. What is the output?
replace
Which gate do you want to replace?
8
Test component number 7, set the first input to 0 and the second
input to 0. What is the output?
1
Processing..
Test component number 7, set the first input to 0 and the second
input to 1. What is the output?
1
Processing..
Test component number 7, set the first input to 1 and the second
input to 0. What is the output?
1
Processing..
Gate number 7 is probably inverted. You should replace it.
Is the circuit still broken? Type (Y) to continue, (N) to stop.
n
```

Directe Defect Herkenning

Wij vonden echter dat alleen werken met het certainty factor model niet krachtig genoeg was, om de reden dat je soms na twee testen al met zekerheid weet dat een bepaalde toestand voor een gate aan de hand is, terwijl dit uit certainty factors dit nog niet valt te concluderen. Daarom hebben we de volgende wijziging toegepast in het certainty factor model: als een test negatief is voor een bepaalde toestand van een gate, dan verwijderen we het deelnetwerk voor deze toestand. Hiermee kan een toestand die door een test is uitgesloten, later nooit meer als mogelijke toestand worden gezien. Veel belangrijker nog, dit stelt ons systeem in staat om wanneer er nog maar een enkel deelnetwerk over is voor een gate direct de toestand van die gate te bepalen. Wanneer nodig wordt de gebruiker hiervan op de hoogte gesteld.

Een voorbeeldrun die deze mogelijkheid demonstreert is als volgt:

```
Test component number 8, set the first input to 0 and the second
input to 0. What is the output?
1
Processing..
Test component number 8, set the first input to 0 and the second
input to 1. What is the output?
1
Processing..
We have concluded the gate with number 8 has state: kortsluiting
Please replace it for a new one.
Is the circuit still broken? Type (Y) to continue, (N) to stop.
n
```

Handigheidjes

Salience
De salience levels zorgen voor de goede volgorde van de verschillende rules. We hebben de salience zowel ingezet om verschillende delen van het expertsysteem van elkaar te scheiden als om de verschillende componenten binnen een deel van het systeem te scheiden. We hebben de volgende grote en fijne afstemming binnen ons systeem:

Ten eerste is er het aanmaken van de gate facts voor de line gates (salience 1000+). Dit is nodig als eerste, want hierna worden de netwerken (cfNodes) van de gates aangemaakt (salience 900).

De coConclude regel heeft een salience van 800, aangezien deze moet uitgevoerd worden zo gauw er iets doorgerekend kan worden.

Hierna komen een aantal regels met salience 740-751, dit zijn onder andere het verwijderen van overbodige cfNodes, en het verwijderen van die facts die daarvoor zorgen.

Dan is er zijn er 3 regels van salience 700/701 die zorgen dat er conclusies getrokken worden als dat mogelijk is.

Daarna is er het salience niveau van 100-120, waar het te testen component geselecteerd wordt, de test gevraagd, en daarna die test verwerkt wordt.

Modify vs assert & retract
Mooie overzichtelijke code is erg belangerijk in het bedrijfsleven, maar ook voor het nakijken van de code. In CLIPS code moet je regelmatig facts updaten. Dit kan echter alleen door het facts te retracten en opnieuw (zij het iets aangepast) te asserten. Dit zorgt voor veel rommel in de code. In onze code hebben we daarom gebruik gemaakt van de functie 'modify'. Op de achtergrond doet dit hetzelfde, echter het gebruik van modify maakt de code een stuk overzichtelijker.

Extra's

De volgende extra's zijn geïmplementeerd.

e) De mogelijkheid om een gate in de schakeling te vervangen.

Indien het systeem vraagt om een component (gate) te testen kan men 'replace' invoeren. Daarna wordt gevraagd of de desbetreffende gate op te geven die vervangen moet worden. Het systeem gaat er nu van uit dat dat desbetreffend component correct werkt.

d-1) Aan te geven dat een bepaald component niet testbaar is.
Indien het systeem vraagt om een component (/gate) te testen wat niet mogelijk is voert u 'replace' in. Daarna vult u het nummer van de gate in die niet getest kan worden. Het systeem zal deze component vervolgens overslaan. Dit werkt dus hetzelfde als onderdeel e). Hieronder ziet u hiervan een demonstratie.

```
Test component number 8, set the first input to 0 and the second
input to 0. What is the output?
replace
Which gate do you want to replace?
8
Test component number 7, set the first input to 0 and the second
input to 0. What is the output?
0
Processing..
Test component number 7, set the first input to 0 and the second
input to 1. What is the output?
0
Processing..
Test component number 7, set the first input to 1 and the second
input to 0. What is the output?
0
Processing..
Test component number 7, set the first input to 1 and the second
input to 1. What is the output?
0
Processing..
We have concluded the gate with number 7 has state: geenContact
Please replace it for a new one.
Is the circuit still broken? Type (Y) to continue, (N) to stop.
n
```

d-2) Aan te geven dat een gate waarschijnlijk kapot is (verdachtmaking).
Bij de start van het programma wordt u gevraagd om eventuele vermoedes van defecte schakelingen aan te geven. Het certainty factor netwerk van deze component wordt vervolgens aangepast zodat die schakeling eerder verdacht wordt. Helaas hebben we dit niet werkend gekregen, omdat CLIPS niet matched zoals we verwachten. Het probleem zit rond regel 72 in domain.clp. Hieronder geven we een voorbeeldrun van hoe het momenteel werkt.

```
Test component number 8, set the first input to 0 and the second
input to 0. What is the output?
doom
Which gate do you want to doom?
7
Test component number 8, set the first input to 0 and the second
input to 0. What is the output?
stop
```

Als de dooming functie correct zou werken, dan wordt de certainty voor een defect van gate 7 zo hoog, dat ons systeem een test van gate 7 zou gaan vragen. Ook ziet u hier hoe de stop functie werkt.

d-3) Drempelwaarde opgeven
In schakeling.clp kan u middels een optie aangeven wat de threshold is, die gebruikt wordt om te bepalen of een bepaald defect misschien optreedt.

g-1) Directe Defect Herkenning
Zie hiervoor het onderdeel "Directe Defect Herkenning" bij het hoofdstuk Features.

g-2) Lines wel/niet parsen
Met deze optie, die aanpasbaar is in schakeling.clp, kunt u aangeven of aangenomen mag worden dat alle draden perfect zijn. Dit voorkomt dat de parser de draden expliciet gaat maken. Als de parser dat niet hoeft te doen, scheelt dit veel performance in zowel de parsing als de redeneerfase.

Further research

De volgende extra's zouden op de volgende manier kunnen worden geimplenteerd.

a): Divide and conquer met behulp van c): delen kunnen meten
Niet alle gates hebben invloed op alle outputs. Dit kan je op de volgende manier gebruiken om het aantal tests drastisch te verkleinen. Dit noemen we even horizontaal splitsen.

1. Er moet een output gekozen worden (dat kan door een gate te vinden met outputs 0)
2. Daarna kies je een output waarde (0 of 1)
3. Vervolgens zoek je in gateBehaviour op welke inputs passen bij de gekozen output van de huidige gate
4. Ga daarna verder naar de vorige gates met de inputs van de huidige gate
5. Totdat je bij een gate komt met inputs 0, en kijk welke input er in moet om de juiste output te krijgen
6. Laat de gebruiker met die input de uiteindelijke output testen, met die output kan dus gekeken worden of het hele gedeelte goed is of niet
7. Wanneer de output verkeerd is, kies een gate in het midden van het vorige doorlopen pad, en voer dezelfde procedure uit, als die goed is, dan moet er de andere kant op getest worden.

Dit is een toepassing van divide and conquer met behulp van het c) gedeelte. In principe kun je dit probleem zien als de schakeling op zich, met de inputs en outputs. Door dit op te splitsen wordt het verdeeld en kan er sneller naar de fout gezocht worden op een efficientere manier. Het divide and conquer principe wordt toegepast door de fout te elimineren met het ‘binaire’ zoeken op de schakeling.

Daarnaast zou er nog onderzoek moeten komen naar de efficiëntie van ons systeem, want die is lang niet optimaal. Bijvoorbeeld het opbouwen van de certainty factor netwerken zou bijvoorbeeld niet voor iedere toestand per gate hoeven te gebeuren. In de plaats daarvan zou er ook per gate één certainty factor netwerk kunnen worden opgebouwd die voor alle toestanden in één node de certainty kunnen bevatten. Op die manier wordt het aantal feiten sterk teruggebracht en zou ons systeem sneller moeten draaien. Aangezien dit wel een complexer netwerk oplevert, kunnen we niet met zekerheid zeggen of dit de ideale oplossing

is. We hebben gewoon helaas te weinig tijd overgehouden om deze efficientieproblemen grondig te kunnen aanpakken.

Ervaringen & Taakverdeling

Het was lastig om een goede taakverdeling te vinden, maar de globale verdeling was:

Timon: verslag maken + meekijken code

Geerten: code schrijven

Pieter: code schrijven

Slotwoord

Rest ons de nakijker veel succes te wensen met het nakijken van deze opdracht.