Inteligência Artificial
Diagnóstico em Circuitos Digitais
Relatório

Equipe:

Crislânio de Souza Macêdo

Sumário

1 - Introdução

Os circuitos digitais podem falhar. Um circuito digital combinatório pode ser especificado de duas formas: pela sua função lógica ou pela sua tabela verdade. Dizemos que um circuito lógico contém uma ou mais falhas quando a função lógica que ele executa de fato não corresponde àquela da especificação. A forma de observar que um circuito lógico combinatório está com falhas é verificando que a tabela verdade do seu funcionamento real não corresponde a tabela verdade da função lógica especificada. Para efeito deste trabalho vamos supor que uma porta lógica só possui quatro tipos de falha: a saída ou uma entrada (ou mais de uma entrada) pode estar "presaEm0" ou "presaEm1". Esse trabalho de programação consiste em criar um programa de computador que recebe como entrada um arquivo texto e gera como saída outro arquivo texto. O arquivo texto de entrada deve conter duas especificações: a especificação nominal de um circuito digital combinatório, ou seja, de como ele deveria funcionar, e a especificação do conjunto das falhas existentes no circuito real. O arquivo texto gerado como saída deve especificar (apresentar) o conjunto de todas as possíveis falhas, simples ou múltiplas, que explicam o comportamento falho observado. O objetivo é que o conjunto das falhas reais indicadas no arquivo texto de entrada esteja no conjunto das falhas possíveis indicado no arquivo texto de saída gerado pelo programa. Esta é uma tarefa inteligente de diagnóstico sobre a qual veremos mais. É também uma introdução e motivação para SEs.

Os problemas de busca podem ser de diversas formas, busca informada, não informada, por restrição. Existem diversas estratégias, heurísticas, algoritmos para cada tipo de abordagem, porém, dependendo do problema, determinados algoritmos, heurísticas podem exigir um tempo computacional, armazenamento maior.

1.1 - O que é o Problema.

O problema de diagnóstico em circuito consiste em separar soluções com falhas de um determinado conjunto. Dado um circuito com sem falhas e um circuito com falhas geradas pelo usuário, o problema consiste em gerar todas as combinações de falhas do circuito sem falha, combinações 1x1, 2x2, 3x3 que originam a tabela verdade do circuito com falha informado.

1.2 - Justificativa.

Uma vez que tivermos todas as combinações de falhas do circuito, seguimos para uma etapa de diagnóstico. Para concluirmos essa tarefa é necessário varrermos o espaço de busca

de todas as falhas geradas e fazermos uma comparação com a tabela verdade das saídas do circuito com falhas informado.

Essa é uma tarefa de diagnóstico e detectar as soluções no espaço de busca não é uma tarefa trivial, assim que não pretendemos demonstrar que nossa solução é a melhor, porém, nossa abordagem resolve o problema dada as restrições detalhadas a seguir.

2 - Formulação do problema

Em geral um problema de busca pode ser modelado em seis etapas, são elas: Estados, Estado Inicial, Ações, Modelo de transição, Teste de objetivo, Custo do caminho. Formulamos o problema de diagnóstico de circuitos lógicos da seguinte forma

• **Estados**: São as cláusulas do circuito, as portas com as entradas com falhas ou sem falha.

• **Estado inicial**: A Cláusula inicial representada pela porta inicial do circuito P0.

• **Ações**: A formulação mais simples define as ações como expandir as portas iniciais P0, P1..Pn, PSaida1, PSaida2,... pSaidaN respectivamente.

• **Modelo de transição**: Expandir P0 significa colocar a cláusula P0 em qualquer aparição de P0 no circuito, se faz isso para cada porta respectivamente. Depois resolver P0, P1,...Pn, PSaida1, PSaida2,... pSaidaN, ou seja, fazer a tabela verdade para cada porta respectivamente. Uma Observação a ser feita é que primeiro é gerado o circuito sem falha, logo em seguida é expandido o circuito com falha.

• **Teste de objetivo**: A tabela verdade gerada é igual a tabela verdade do circuito com falha informada pelo usuário.

• **Custo de caminho**: Cada passo custa 1 e, assim, o custo do caminho é o número de passos do caminho.

2.1 - Abordagem do problema

Para resolvermos o problema de diagnóstico em circuitos digitais consideramos as seguintes restrições e sintaxe.

Cada variável de entrada do circuito será indicado por letras do alfabeto.

Cada porta do circuito será indicado letra letra P e o número da porta. Exemplo p0,p1...pn, pF0, pF1, ... pFn.

Descrição do circuito lógico:

Var indica as variáveis de entrada, coloque as variáveis de entrada em uma lista separado por vírgula e sem espaço.

Para representar as Falhas de entrada coloque E0 para representar uma entrada presa em 0 e E1 para representar uma entrada presa em 1

Fout representa falhas de saída para as portas, no caso só coloque na lista as postas que tem falha e o valor, no exemplo temos p1=0 que indica que p1 está preso na saída da porta por 0.

Em seguida temos a representação das portas com sua entrada lógica. Por exemplo, p0 = (A and B). É necessário coloca colocar espaço em cada caractere de entrada.

var = [A, B, C]

Fout = [p1 = 0]

p0 = (A and E0) # E0 indica que a porta P0 está com um erro de entrada, preso em 0

p1 = (p0 and B)

pF2 = (p1)

2.2 Abordagem técnica

Nossa abordagem para resolução do problema consistem em 4 etapas.

1° Mapeamento de falhas e propagação das cláusulas:
Nessa etapa aplicamos as funções de propagação **aplicaClausula()** e **aplicaClausulaComFalhas()** no quais são responsáveis por estruturar o circuito para um formato de *Dataframe* e por aplicar as falhas no circuito com falha . Uma vez que estruturamos o circuito no formato de *Dataframe* e temos uma lista com as cláusulas de cada porta podemos acionar uma função para gerar a tabela verdade do circuito.

2° Gerar Falhas:
Nessa etapa aplicamos as as funções **simularFalhasPorta13x3(), simularFalhasPorta23x3(), simularFalhasPorta22x2(), simularFalhasPorta12x2(), simularFalhasPorta2(), simularFalhasPorta1()** que são responsáveis por gerar falhas aleatórias nas portas do circuito. Obtemos os índices das variáveis de cada cláusula e aplicados as combinaçõe necessárias para gerar todas as combinações e gerar todas as cláusulas de cada porta no circuito, assim teremos uma lista de cláusulas para serem executadas em uma função para gerar as tabelas verdades.

3° Gerar tabelas verdades:
Nessa etapa geramos as tabelas verdades do circuitos juntamente com os arquivos de csv e .txt. Aplicamos as funções **Truths()** que são funções para gerar as tabelas verdades do circuito

com falha e sem falha com 1x1 combinações de falhas, 2x2 e 3x3. Como guardamos nosso circuito em um *Dataframe* e vamos gerando iterativamente a tabela verdade do circuito demandamos de uma complexidade de espaço linear ao tamanho do circuito.

4° Diagnóstico do circuito:
Nesta etapa analisamos os arquivos gerados na etapa anterior e analisamos se as tabelas verdades das saídas do circuito são iguais às saídas do circuito com falha informado.
Guardamos em um novo arquivo csv e txt o conjunto de saídas no qual gera a tabela verdade do circuito com falha informado.

4° Tempo computacional e saída do circuito:
Nessa etapa analisamos o tempo computacional do diagnóstico e geramos um relatório com o circuito e as saídas pelo qual a tabela verdade corresponde a tabela verdade do circuito com falhas.

2.3 Instruções da execução do programa e dependências
Para resolução do problema usamos a linguagem de programação python 3.0 ou superior juntamente com o **jupyter notebook.**

Para executar o programa temos a variável *gerarCSV* no qual o valor 1 indica que testamos o programa com falhas e 0 sem falhas.

gerarCSV = 1 # 1 Roda o programa com Falha

gerarCSV = 0 # 0 Roda o programa sem Falha

É recomendado que gere primeiro o circuito sem falha e em seguida o circuito com falha pois as tabelas de falhas geradas são analisadas levando em consideração o circuito sem falha.

A organização do repositório do projeto segue da seguinte forma: Uma pasta com arquivos de saída para os 4 circuitos, uma pasta para o relatório.

3 - Resultados
Os resultados mostrados a seguir mostra que para execução de falhas 2x2, 3x3 combinação há uma tendência em que as tabelas geradas sejam iguais assim que em geral o algoritmo não consegue detectar as falhas do circuito com falhas informado
3.1 - Alguns resultados obtidos.

Como exemplo prático para o circuito 1 simulamos uma falha na porta 4 pressa em 0. Um erro na porta 0 na entrada B presa em 0.

```
var = [ A, B, C ]
Fout = [ p4 = 0 ]
p0 = ( A ^  E0 )
p1 = ( ( ( A ^  E0 ) ^ C ) )
```

p2 = (C and (A ^ B))
p3 = (B and A)
p4 = ((C and (A ^ B)) or (B and A))
pF1 = (((C and (A ^ B)) or (B and A)))
pF2 = ((((A ^ B) ^ C)))

4 formas de se chegar a tabela verdade do circuito com falha 1x1 combinações para a saída 1 do circuito.

Tempo de Execução do Diagnóstico
Tempo de Execução do programa: 1.6481096744537354 segundos

	((((0 ^ B) ^ C)))	**((((1 ^ B) ^ C)))**	**((((A ^ 0) ^ C)))**	**((((A ^ 1) ^ C)))**
0	0	1	0	1
1	1	0	1	0
2	1	0	0	1
3	0	1	1	0
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	0	1	0	1

Nossa abordagem gera as combinações de falhas para as saídas do circuito e guarda em um csv, a seguir comparamos com o circuito com falhas e guardamos em um csv e txt as tabelas verdades iguais. Com isso obtermos nosso conjunto de falhas para o diagnóstico do circuito.

Nossa abordagem cresce em tamanho de espaço e não de tempo, assim que a medida que temos um circuito com um número maior de cláusulas e variáveis teremos um tamanho de armazenamento maior.

Para os circuitos 2,3,4 não foi encontrado solução dentro o conjunto de falhas gerados por nossa abordagem. Na referência você poderá conferir os resultados com mais detalhes nos arquivos gerados para cada circuito.

5 - Referências

Repositótio do código: https://github.com/crislanio/LogicCircuitIA

Relatório web: https://docs.google.com/document/d/e/2PACX-1vQDYoy0xF4ZPWdy3I_JCgEjcxPQFQGDSmVfKV3fO7KmUy5mgPFeWVNRhJeeqPY3HL78Vypgo_PBDxUW/pub

HTML do Código: https://github.com/crislanio/LogicCircuitIA/blob/master/Trabalho_01_IA-MACC.slides.html