Diagnóstico Baseado em Modelo

Silvio do Lago Pereira

Laboratório de Inteligência Artificial
Instituto de Matemática e Estatística
— Universidade de São Paulo —
slago@ime.usp.br

1 de julho de 2003

1 Introdução

Sejam $\mathcal{SD}$ um conjunto de fórmulas descrevendo o funcionamento de um sistema, $\mathcal{VAR}$ um conjunto de variáveis representando os componentes desse sistema e $\mathcal{OBS}$ um conjunto de literais descrevendo um estado observado do sistema. A necessidade de diagnóstico surge quando o conjunto $\mathcal{SD} \cup \mathcal{VAR} \cup \mathcal{OBS}$ é inconsistente. Nesse caso, o *diagnóstico* consiste de um subconjunto de $\mathcal{VAR}$, contendo variáveis cujos componentes correspondentes podem estar funcionando de maneira anormal.

Nesse trabalho, implementamos um programa para diagnóstico baseado em modelo formal, conforme descrito em [*Reiter, 1987*]. Na seção 2, apresentamos instruções para instalação e uso desse programa, bem como alguns detalhes de sua implementação; na seção 3, apresentamos os resultados dos testes realizados e, finalmente, no apêndice, apresentamos o código-fonte do programa.

2 O programa `diag`

Nessa seção, apresentamos o programa `diag`, que realiza diagnóstico baseado em modelo formal. Todo o código-fonte desse programa está contido num único arquivo, denominado `diag.c`. Assim, para instalar esse programa, basta executar o seguinte comando de compilação:

```
% cc diag.c -o diag
```

Uma vez instalado, para executarmos o programa `diag`, precisamos ter um arquivo contendo a descrição de um problema de diagnóstico. Um arquivo de

descrição contém três seções: `SD`, `VAR` e `OBS`. A primeira delas é formada pelas cláusulas que descrevem o funcionamento do sistema; a segunda, é formada apenas pelas variáveis proposicionais que representam os componentes do sistema e a terceira contém literais que descrevem o estado observado. Por exemplo, para o circuito na figura 1, uma descrição de problema poderia ser a seguinte:

```
SD
  -A -OKX C
  -B -OKX C
  A B -OKX -C
  -A -B -OKY D
  A -OKY -D
  B -OKY -D
  -D -OKZ -E
  D -OKZ E
  -C -E -OKW F
  C -OKW -F
  E -OKW -F
VAR
  OKX
  OKY
  OKZ
  OKW
OBS
  A
  B
  F
```

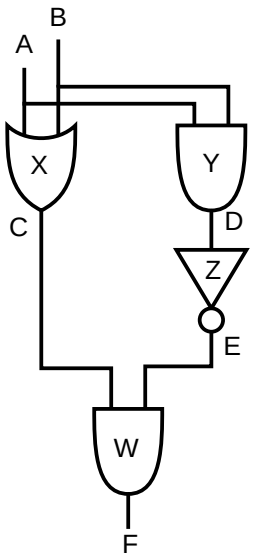

Figura 1: Circuito que implementa o operador *xor*.

Assim, supondo que essa descrição esteja no arquivo `xor.cnf`, para encontrar os diagnósticos para esse sistema, podemos executar o comando `diag xor.cnf`

e, então, os diagnósticos encontrados serão gravados num arquivo denominado `xor.dia`. Se for desejado rastrear os passos de construção do grafo acíclico, a partir do qual são extraídos os diagnósticos, podemos usar a opção `-r` do comando `diag`, conforme exemplificado a seguir:

```
% diag xor.cnf -r

Rotular raiz com {OKX,OKY,OKZ,OKW}.

=================== Vértices ====================
0: [1, {}, {OKX,OKY,OKZ,OKW}]
-------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKX.
    * criar o vértice 1 com rótulo {OKY,OKZ,OKW}
    * alterar vértice 0 de {OKX,OKY,OKZ,OKW}
      para {OKY,OKZ,OKW},
      removendo os arcos em {OKX}.
      - remover o arco OKW de 1.
      - remover o arco OKZ de 1.
      - remover o arco OKY de 1.
      - remover o vértice 1.
      - remover o arco OKX de 0.
 => A partir do arco OKY.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.
 => A partir do arco OKW.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {OKY}
2: {OKZ}
3: {OKW}
```

2.1 Detalhes de implementação

Nessa seção, descrevemos alguns detalhes da implementação do programa `diag`.

2.1.1 Os algoritmos

Para encontrar os diagnósticos, adaptamos o algoritmo de corte minimal de *Reiter* de modo que, em vez de fornecermos como entrada a família de conjuntos cujo corte minimal deverá ser computado, vamos gerando esses conjuntos sob demanda. Quando precisamos de um conjunto conflito para rotular um determinado vértice m, chamamos o algoritmo `DPL` com o conjunto de cláusulas $\mathcal{SD} \cup \mathcal{VAR} \setminus H(m) \cup \mathcal{OBS}$, onde $H(m)$ é o conjunto de rótulos do caminho que le-

va da raiz até o vértice m. Então, se esse conjunto de cláusulas for insatisfatível, `DPL` devolve um conjunto conflito $\mathcal{C} \subset \mathcal{VAR}$ tal que $\mathcal{C} \cap H(m) = \emptyset$.

2.1.2 As estruturas de dados

Na implementação do algoritmo de *Reiter*, os conjuntos de variáveis que rotulam os vértices do grafo são representados por vetores de 32 bits. Isso é possível porque garantimos que as variáveis do tipo `OK*` sejam alocadas nas primeiras 32 posições da tabela de símbolos. Por exemplo, se as variáveis são `OKV`, `OKW`, `OKX`, `OKY` e `OKZ`, o conjunto $\{OKV, OKY, OKZ\}$ é representado pelo binário `11001`. Assim, todas as operações sobre conjuntos podem ser realizadas eficientemente, em tempo constante, com o uso de operadores lógicos bit-a-bit.

Para armazenar o grafo acíclico construído pelo programa, utilizamos a estrutura de dados ilustrada na figura 2. Nessa estrutura, cada vértice é representado por um nó que armazena os seguintes campos:

- `ref`: um contador de referências, indicando o grau de entrada desse vértice;
- `path`: um conjunto que rotula os arcos que levam da raiz até esse vértice;
- `label`: um conjunto que rotula o próprio vértice, e
- `arcs`: uma lista dos arcos que partem do vértice.

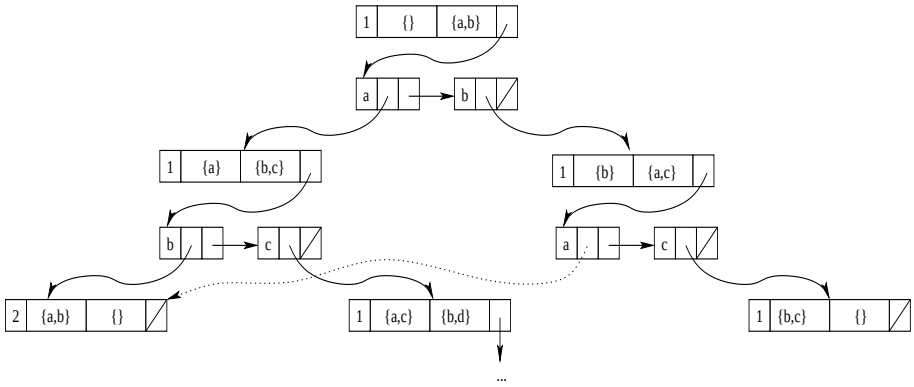

Figura 2: Estrutura de dados para o armazenamento do DAG.

O campo `ref` de um vértice é incrementado a cada vez que esse vértice é reutilizado como destino de um arco recém criado. Inversamente, cada vez que um arco é removido, o campo `ref` do vértice apontado por esse arco é decrementado. Então, se esse contador de referências é zerado, o vértice correspondente é removido do grafo. O campo `path` de vértice m, nos permite obter, em tempo constante, o conjunto $H(m)$. O campo `label` armazena o rótulo do vértice; particularmente, se esse rótulo for o conjunto vazio, então esse vértice é um nó terminal no grafo e o seu campo `label` armazena um possível diagnóstico.

3 Testes realizados

3.1 Anderson

```
Cláusulas

 0: -A -okX -B
 1: A -okX B
 2: -B -okY -C
 3: B -okY C
 4: okX
 5: okY
 6: A
 7: -C

Rotular raiz com {okX,okY}.

==================== Vértices ====================
0: [1, {}, {okX,okY}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco okX.
    * utilizar o rótulo especial.
 => A partir do arco okY.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {okX}
2: {okY}
```

3.2 André

```
Cláusulas

 0: -A -B -OKCP C
 1: A -OKCP -C
 2: B -OKCP -C
 3: -D -E -OKEV F
 4: D -OKEV -F
 5: E -OKEV -F
 6: -C -F -OKBD G
 7: C -OKBD -G
 8: F -OKBD -G
 9: OKCP
10: OKEV
11: OKBD
12: -G
13: F
14: A
15: B
```

```
Rotular raiz com {OKCP,OKEV,OKBD}.

==================== Vértices ====================
0: [1, {}, {OKCP,OKEV,OKBD}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKCP.
    * utilizar o rótulo especial.
 => A partir do arco OKEV.
    * criar o vértice 2 com rótulo {OKCP,OKBD}
    * alterar vértice 0 de {OKCP,OKEV,OKBD}
      para {OKCP,OKBD},
      removendo os arcos em {OKEV}.
      - remover o arco OKBD de 2.
      - remover o arco OKCP de 2.
      - remover o vértice 2.
      - remover o arco OKEV de 0.
 => A partir do arco OKBD.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {OKCP}
2: {OKBD}
```

3.3 Antônio

```
Cláusulas

 0: -A -okX -C
 1: A -okX C
 2: -B -okY D
 3: -C -okY D
 4: B C -oky -D
 5: -D -okZ F
 6: -E -okZ F
 7: D E -okZ -F
 8: okX
 9: okY
10: okZ
11: B
12: -F

Rotular raiz com {okX,okY,okZ}.

==================== Vértices ====================
0: [1, {}, {okX,okY,okZ}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco okX.
```

```
    * criar o vértice 1 com rótulo {okY,okZ}
    * alterar vértice 0 de {okX,okY,okZ}
      para {okY,okZ},
      removendo os arcos em {okX}.
      - remover o arco okZ de 1.
      - remover o arco okY de 1.
      - remover o vértice 1.
      - remover o arco okX de 0.
 => A partir do arco okY.
    * utilizar o rótulo especial.
 => A partir do arco okZ.
    * utilizar o rótulo especial.

================== Diagnósticos ====================

1: {okY}
2: {okZ}
```

3.4 Barcelos

```
Cláusulas

 0: -A -B -okX -D
 1: A -okX -D
 2: B -okX -B
 3: C -okY -E
 4: D -okY -E
 5: -B -okZ F
 6: -E -okZ F
 7: B E -okZ -F
 8: okX
 9: okY
10: okZ
11: -B
12: -C
13: F

Rotular raiz com {okX,okY,okZ}.

=================== Vértices ====================
0: [1, {}, {okX,okY,okZ}]
-------------------------------------------------

Expandir vértice 0.

 => A partir do arco okX.
    * criar o vértice 1 com rótulo {okY,okZ}
    * alterar vértice 0 de {okX,okY,okZ}
      para {okY,okZ},
      removendo os arcos em {okX}.
      - remover o arco okZ de 1.
      - remover o arco okY de 1.
      - remover o vértice 1.
      - remover o arco okX de 0.
 => A partir do arco okY.
```

```
    * utilizar o rótulo especial.
 => A partir do arco okZ.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {okY}
2: {okZ}
```

3.5 Christian

```
Cláusulas

 0: -A -okN1 -W
 1: A -okN1 W
 2: -C -B -okA1 Z
 3: C -okA1 -Z
 4: B -okA1 -Z
 5: -W -C -okA2 Y
 6: W -okA2 -Y
 7: C -okA2 -Y
 8: -A -C -okA3 F
 9: A -okA3 -F
10: C -okA3 -F
11: -A -okO1 D
12: -Z -okO1 D
13: A Z -okO1 -D
14: -Y -okO2 E
15: -B -okO2 E
16: Y B -okO2 -E
17: okN1
18: okA1
19: okA2
20: okA3
21: okO1
22: okO2
23: -A
24: B
25: C
26: -D
27: -E
28: F

Rotular raiz com {okN1,okA1,okA2,okA3,okO1,okO2}.

==================== Vértices ====================
0: [1, {}, {okN1,okA1,okA2,okA3,okO1,okO2}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco okN1.
    * criar o vértice 1 com rótulo {okA1,okA2,okA3,okO1,okO2}
    * alterar vértice 0 de {okN1,okA1,okA2,okA3,okO1,okO2}
      para {okA1,okA2,okA3,okO1,okO2},
```

```
      removendo os arcos em {okN1}.
      - remover o arco okO2 de 1.
      - remover o arco okO1 de 1.
      - remover o arco okA3 de 1.
      - remover o arco okA2 de 1.
      - remover o arco okA1 de 1.
      - remover o vértice 1.
      - remover o arco okN1 de 0.
 => A partir do arco okA1.
    * criar o vértice 2 com rótulo {okN1,okA2,okA3,okO1,okO2}
 => A partir do arco okA2.
    * criar o vértice 3 com rótulo {okN1,okA1,okA3,okO1,okO2}
 => A partir do arco okA3.
    * criar o vértice 4 com rótulo {okN1,okA1,okA2,okO1,okO2}
 => A partir do arco okO1.
    * criar o vértice 5 com rótulo {okN1,okA1,okA2,okA3,okO2}
 => A partir do arco okO2.
    * criar o vértice 6 com rótulo {okN1,okA1,okA2,okA3,okO1}

==================== Vértices ====================
0: [1, {}, {okA1,okA2,okA3,okO1,okO2}]
2: [1, {okA1}, {okN1,okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okN1,okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okA2,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA2,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA2,okA3,okO1}]
--------------------------------------------------

Expandir vértice 2.

 => A partir do arco okN1.
    * criar o vértice 7 com rótulo {okA2,okA3,okO1,okO2}
    * alterar vértice 0 de {okA1,okA2,okA3,okO1,okO2}
      para {okA2,okA3,okO1,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okO2 de 2.
      - remover o arco okO1 de 2.
      - remover o arco okA3 de 2.
      - remover o arco okA2 de 2.
      - remover o arco okO2 de 7.
      - remover o arco okO1 de 7.
      - remover o arco okA3 de 7.
      - remover o arco okA2 de 7.
      - remover o vértice 7.
      - remover o arco okN1 de 2.
      - remover o vértice 2.
      - remover o arco okA1 de 0.

==================== Vértices ====================
0: [1, {}, {okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okN1,okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okA2,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA2,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA2,okA3,okO1}]
--------------------------------------------------

Expandir vértice 3.
```

```
 => A partir do arco okN1.
    * criar o vértice 8 com rótulo {okA1,okA3,okO1,okO2}
    * alterar vértice 3 de {okN1,okA1,okA3,okO1,okO2}
      para {okA1,okA3,okO1,okO2},
      removendo os arcos em {okN1}.
      - remover o arco okO2 de 8.
      - remover o arco okO1 de 8.
      - remover o arco okA3 de 8.
      - remover o arco okA1 de 8.
      - remover o vértice 8.
      - remover o arco okN1 de 3.
 => A partir do arco okA1.
    * criar o vértice 9 com rótulo {okN1,okA3,okO1,okO2}
 => A partir do arco okA3.
    * criar o vértice 10 com rótulo {okN1,okA1,okO1,okO2}
    * alterar vértice 4 de {okN1,okA1,okA2,okO1,okO2}
      para {okN1,okA1,okO1,okO2},
      removendo os arcos em {okA2}.
      - remover o arco okA2 de 4.
 => A partir do arco okO1.
    * criar o vértice 11 com rótulo {okN1,okA1,okA3,okO2}
    * alterar vértice 5 de {okN1,okA1,okA2,okA3,okO2}
      para {okN1,okA1,okA3,okO2},
      removendo os arcos em {okA2}.
      - remover o arco okA2 de 5.
 => A partir do arco okO2.
    * criar o vértice 12 com rótulo {okN1,okA1,okA3,okO1}
    * alterar vértice 6 de {okN1,okA1,okA2,okA3,okO1}
      para {okN1,okA1,okA3,okO1},
      removendo os arcos em {okA2}.
      - remover o arco okA2 de 6.

==================== Vértices ====================
0: [1, {}, {okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
9: [1, {okA1,okA2}, {okN1,okA3,okO1,okO2}]
10: [1, {okA2,okA3}, {okN1,okA1,okO1,okO2}]
11: [1, {okA2,okO1}, {okN1,okA1,okA3,okO2}]
12: [1, {okA2,okO2}, {okN1,okA1,okA3,okO1}]
--------------------------------------------------

Expandir vértice 4.

 => A partir do arco okN1.
    * criar o vértice 13 com rótulo {okA1,okA2,okO1,okO2}
 => A partir do arco okA1.
    * criar o vértice 14 com rótulo {okN1,okA2,okO1,okO2}
 => A partir do arco okO1.
    * criar o vértice 15 com rótulo {okN1,okA1,okA2,okO2}
 => A partir do arco okO2.
    * criar o vértice 16 com rótulo {okN1,okA1,okA2,okO1}

==================== Vértices ====================
```

```
0: [1, {}, {okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
9: [1, {okA1,okA2}, {okN1,okA3,okO1,okO2}]
10: [1, {okA2,okA3}, {okN1,okA1,okO1,okO2}]
11: [1, {okA2,okO1}, {okN1,okA1,okA3,okO2}]
12: [1, {okA2,okO2}, {okN1,okA1,okA3,okO1}]
13: [1, {okN1,okA3}, {okA1,okA2,okO1,okO2}]
14: [1, {okA1,okA3}, {okN1,okA2,okO1,okO2}]
15: [1, {okA3,okO1}, {okN1,okA1,okA2,okO2}]
16: [1, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
-------------------------------------------------

Expandir vértice 5.

 => A partir do arco okN1.
    * criar o vértice 17 com rótulo {okA1,okA2,okA3,okO2}
 => A partir do arco okA1.
    * criar o vértice 18 com rótulo {okN1,okA2,okA3,okO2}
 => A partir do arco okA3.
    * reutilizar o vértice 15.
 => A partir do arco okO2.
    * criar o vértice 19 com rótulo {okN1,okA1,okA2,okA3}

==================== Vértices ====================
0: [1, {}, {okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
9: [1, {okA1,okA2}, {okN1,okA3,okO1,okO2}]
10: [1, {okA2,okA3}, {okN1,okA1,okO1,okO2}]
11: [1, {okA2,okO1}, {okN1,okA1,okA3,okO2}]
12: [1, {okA2,okO2}, {okN1,okA1,okA3,okO1}]
13: [1, {okN1,okA3}, {okA1,okA2,okO1,okO2}]
14: [1, {okA1,okA3}, {okN1,okA2,okO1,okO2}]
15: [2, {okA3,okO1}, {okN1,okA1,okA2,okO2}]
16: [1, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okA3,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [1, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
-------------------------------------------------

Expandir vértice 6.

 => A partir do arco okN1.
    * criar o vértice 20 com rótulo {okA1,okA2,okA3,okO1}
 => A partir do arco okA1.
    * criar o vértice 21 com rótulo {okN1,okA2,okA3,okO1}
 => A partir do arco okA3.
    * reutilizar o vértice 16.
 => A partir do arco okO1.
    * reutilizar o vértice 19.

==================== Vértices ====================
```

```
0: [1, {}, {okA2,okA3,okO1,okO2}]
3: [1, {okA2}, {okA1,okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
9: [1, {okA1,okA2}, {okN1,okA3,okO1,okO2}]
10: [1, {okA2,okA3}, {okN1,okA1,okO1,okO2}]
11: [1, {okA2,okO1}, {okN1,okA1,okA3,okO2}]
12: [1, {okA2,okO2}, {okN1,okA1,okA3,okO1}]
13: [1, {okN1,okA3}, {okA1,okA2,okO1,okO2}]
14: [1, {okA1,okA3}, {okN1,okA2,okO1,okO2}]
15: [2, {okA3,okO1}, {okN1,okA1,okA2,okO2}]
16: [2, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okA3,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
-------------------------------------------------

Expandir vértice 9.

 => A partir do arco okN1.
    * criar o vértice 22 com rótulo {okA3,okO1,okO2}
    * alterar vértice 0 de {okA2,okA3,okO1,okO2}
      para {okA3,okO1,okO2},
      removendo os arcos em {okA2}.
      - remover o arco okO1 de 12.
      - remover o arco okA3 de 12.
      - remover o arco okA1 de 12.
      - remover o arco okN1 de 12.
      - remover o vértice 12.
      - remover o arco okO2 de 3.
      - remover o arco okO2 de 11.
      - remover o arco okA3 de 11.
      - remover o arco okA1 de 11.
      - remover o arco okN1 de 11.
      - remover o vértice 11.
      - remover o arco okO1 de 3.
      - remover o arco okO2 de 10.
      - remover o arco okO1 de 10.
      - remover o arco okA1 de 10.
      - remover o arco okN1 de 10.
      - remover o vértice 10.
      - remover o arco okA3 de 3.
      - remover o arco okO2 de 9.
      - remover o arco okO1 de 9.
      - remover o arco okA3 de 9.
      - remover o arco okO2 de 22.
      - remover o arco okO1 de 22.
      - remover o arco okA3 de 22.
      - remover o vértice 22.
      - remover o arco okN1 de 9.
      - remover o vértice 9.
      - remover o arco okA1 de 3.
      - remover o vértice 3.
      - remover o arco okA2 de 0.
```

```
==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okN1,okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
13: [1, {okN1,okA3}, {okA1,okA2,okO1,okO2}]
14: [1, {okA1,okA3}, {okN1,okA2,okO1,okO2}]
15: [2, {okA3,okO1}, {okN1,okA1,okA2,okO2}]
16: [2, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okA3,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
--------------------------------------------------

Expandir vértice 13.

 => A partir do arco okA1.
    * criar o vértice 23 com rótulo {okA2,okO1,okO2}
    * alterar vértice 13 de {okA1,okA2,okO1,okO2}
      para {okA2,okO1,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okO2 de 23.
      - remover o arco okO1 de 23.
      - remover o arco okA2 de 23.
      - remover o vértice 23.
      - remover o arco okA1 de 13.
 => A partir do arco okA2.
    * criar o vértice 24 com rótulo {okA1,okO1,okO2}
    * alterar vértice 4 de {okN1,okA1,okO1,okO2}
      para {okA1,okO1,okO2},
      removendo os arcos em {okN1}.
      - remover o arco okO2 de 13.
      - remover o arco okO1 de 13.
      - remover o arco okO2 de 24.
      - remover o arco okO1 de 24.
      - remover o arco okA1 de 24.
      - remover o vértice 24.
      - remover o arco okA2 de 13.
      - remover o vértice 13.
      - remover o arco okN1 de 4.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
14: [1, {okA1,okA3}, {okN1,okA2,okO1,okO2}]
15: [2, {okA3,okO1}, {okN1,okA1,okA2,okO2}]
16: [2, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okA3,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
```

```
------------------------------------------------

Expandir vértice 14.

 => A partir do arco okN1.
    * criar o vértice 25 com rótulo {okA2,okO1,okO2}
    * alterar vértice 14 de {okN1,okA2,okO1,okO2}
      para {okA2,okO1,okO2},
      removendo os arcos em {okN1}.
      - remover o arco okO2 de 25.
      - remover o arco okO1 de 25.
      - remover o arco okA2 de 25.
      - remover o vértice 25.
      - remover o arco okN1 de 14.
 => A partir do arco okA2.
    * criar o vértice 26 com rótulo {okN1,okO1,okO2}
 => A partir do arco okO1.
    * criar o vértice 27 com rótulo {okN1,okA2,okO2}
    * alterar vértice 15 de {okN1,okA1,okA2,okO2}
      para {okN1,okA2,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okA1 de 15.
 => A partir do arco okO2.
    * utilizar o rótulo especial.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okA3,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
15: [2, {okA3,okO1}, {okN1,okA2,okO2}]
16: [2, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okA3,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [1, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [1, {okA1,okA3,okO2}, {}]
------------------------------------------------

Expandir vértice 15.

 => A partir do arco okN1.
    * criar o vértice 29 com rótulo {okA1,okA2,okO2}
    * alterar vértice 17 de {okA1,okA2,okA3,okO2}
      para {okA1,okA2,okO2},
      removendo os arcos em {okA3}.
      - remover o arco okA3 de 17.
 => A partir do arco okA2.
    * criar o vértice 30 com rótulo {okN1,okA1,okO2}
    * alterar vértice 5 de {okN1,okA1,okA3,okO2}
      para {okN1,okA1,okO2},
      removendo os arcos em {okA3}.
      - remover o arco okA3 de 5.
```

```
 => A partir do arco okO2.
    * utilizar o rótulo especial.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okA3,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
15: [1, {okA3,okO1}, {okN1,okA2,okO2}]
16: [2, {okA3,okO2}, {okN1,okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [1, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [1, {okA1,okA3,okO2}, {}]
29: [1, {okN1,okA3,okO1}, {okA1,okA2,okO2}]
30: [1, {okA2,okA3,okO1}, {okN1,okA1,okO2}]
31: [1, {okA3,okO1,okO2}, {}]
--------------------------------------------------

Expandir vértice 16.

 => A partir do arco okN1.
    * criar o vértice 32 com rótulo {okA1,okA2,okO1}
    * alterar vértice 16 de {okN1,okA1,okA2,okO1}
      para {okA1,okA2,okO1},
      removendo os arcos em {okN1}.
      - remover o vértice 31.
      - remover o arco okO2 de 15.
      - remover o arco okO2 de 30.
      - remover o arco okA1 de 30.
      - remover o arco okN1 de 30.
      - remover o vértice 30.
      - remover o arco okA2 de 15.
      - remover o arco okO2 de 29.
      - remover o arco okA2 de 29.
      - remover o arco okA1 de 29.
      - remover o vértice 29.
      - remover o arco okN1 de 15.
      - remover o vértice 15.
      - remover o arco okO1 de 32.
      - remover o arco okA2 de 32.
      - remover o arco okA1 de 32.
      - remover o vértice 32.
      - remover o arco okN1 de 16.
 => A partir do arco okA1.
    * reutilizar o vértice 28.
 => A partir do arco okA2.
    * criar o vértice 33 com rótulo {okN1,okA1,okO1}
    * alterar vértice 6 de {okN1,okA1,okA3,okO1}
      para {okN1,okA1,okO1},
      removendo os arcos em {okA3}.
      - remover o arco okA3 de 6.
```

```
 => A partir do arco okO1.
    * utilizar o rótulo especial.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okN1,okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okN1,okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [1, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [2, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [1, {okA3,okO1,okO2}, {}]
--------------------------------------------------

Expandir vértice 17.

 => A partir do arco okA1.
    * criar o vértice 35 com rótulo {okA2,okA3,okO2}
    * alterar vértice 18 de {okN1,okA2,okA3,okO2}
      para {okA2,okA3,okO2},
      removendo os arcos em {okN1}.
      - remover o arco okN1 de 18.
 => A partir do arco okA2.
    * criar o vértice 36 com rótulo {okA1,okA3,okO2}
 => A partir do arco okO2.
    * criar o vértice 37 com rótulo {okA1,okA2,okA3}
    * alterar vértice 19 de {okN1,okA1,okA2,okA3}
      para {okA1,okA2,okA3},
      removendo os arcos em {okN1}.
      - remover o arco okN1 de 19.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3,okO1}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [1, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [2, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [1, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
```

```
36: [1, {okN1,okA2,okO1}, {okA1,okA3,okO2}]
37: [1, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
-------------------------------------------------

Expandir vértice 18.

 => A partir do arco okA2.
    * criar o vértice 38 com rótulo {okN1,okA3,okO2}
 => A partir do arco okA3.
    * reutilizar o vértice 27.
 => A partir do arco okO2.
    * criar o vértice 39 com rótulo {okN1,okA2,okA3}
    * alterar vértice 21 de {okN1,okA2,okA3,okO1}
      para {okN1,okA2,okA3},
      removendo os arcos em {okO1}.
      - remover o arco okO1 de 21.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [2, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [2, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [1, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
36: [1, {okN1,okA2,okO1}, {okA1,okA3,okO2}]
37: [1, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
38: [1, {okA1,okA2,okO1}, {okN1,okA3,okO2}]
39: [1, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
-------------------------------------------------

Expandir vértice 19.

 => A partir do arco okA1.
    * reutilizar o vértice 39.
 => A partir do arco okA2.
    * criar o vértice 40 com rótulo {okN1,okA1,okA3}
 => A partir do arco okA3.
    * reutilizar o vértice 34.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
```

```
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA1,okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [2, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [2, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [2, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
36: [1, {okN1,okA2,okO1}, {okA1,okA3,okO2}]
37: [1, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
38: [1, {okA1,okA2,okO1}, {okN1,okA3,okO2}]
39: [2, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
------------------------------------------------

Expandir vértice 20.

 => A partir do arco okA1.
    * criar o vértice 41 com rótulo {okA2,okA3,okO1}
    * alterar vértice 20 de {okA1,okA2,okA3,okO1}
      para {okA2,okA3,okO1},
      removendo os arcos em {okA1}.
      - remover o arco okO1 de 41.
      - remover o arco okA3 de 41.
      - remover o arco okA2 de 41.
      - remover o vértice 41.
      - remover o arco okA1 de 20.
 => A partir do arco okA2.
    * criar o vértice 42 com rótulo {okA1,okA3,okO1}
 => A partir do arco okA3.
    * reutilizar o rótulo {okA1,okA2,okO1}
 => A partir do arco okO1.
    * reutilizar o vértice 37.

==================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [2, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [2, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [2, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
36: [1, {okN1,okA2,okO1}, {okA1,okA3,okO2}]
37: [2, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
```

```
38: [1, {okA1,okA2,okO1}, {okN1,okA3,okO2}]
39: [2, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
-------------------------------------------------

Expandir vértice 21.

 => A partir do arco okN1.
    * reutilizar o rótulo {okA2,okA3,okO1}
 => A partir do arco okA2.
    * criar o vértice 45 com rótulo {okN1,okA3,okO1}
 => A partir do arco okA3.
    * reutilizar o vértice 28.

=================== Vértices ====================
0: [1, {}, {okA3,okO1,okO2}]
4: [1, {okA3}, {okA1,okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
14: [1, {okA1,okA3}, {okA2,okO1,okO2}]
16: [1, {okA3,okO2}, {okA1,okA2,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
26: [1, {okA1,okA2,okA3}, {okN1,okO1,okO2}]
27: [2, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [3, {okA1,okA3,okO2}, {}]
33: [1, {okA2,okA3,okO2}, {okN1,okA1,okO1}]
34: [2, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
36: [1, {okN1,okA2,okO1}, {okA1,okA3,okO2}]
37: [2, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
38: [1, {okA1,okA2,okO1}, {okN1,okA3,okO2}]
39: [2, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
45: [1, {okA1,okA2,okO2}, {okN1,okA3,okO1}]
-------------------------------------------------

Expandir vértice 26.

 => A partir do arco okN1.
    * criar o vértice 46 com rótulo {okO1,okO2}
    * alterar vértice 0 de {okA3,okO1,okO2}
      para {okO1,okO2},
      removendo os arcos em {okA3}.
      - remover o arco okO1 de 16.
      - remover o arco okO1 de 33.
      - remover o arco okA1 de 33.
      - remover o arco okN1 de 33.
      - remover o vértice 33.
```

```
      - remover o arco okA2 de 16.
      - remover o arco okA1 de 16.
      - remover o vértice 16.
      - remover o arco okO2 de 4.
      - remover o arco okO2 de 36.
      - remover o arco okA3 de 36.
      - remover o arco okA1 de 36.
      - remover o vértice 36.
      - remover o arco okO1 de 4.
      - remover o arco okO2 de 14.
      - remover o arco okO1 de 14.
      - remover o arco okO2 de 26.
      - remover o arco okO1 de 26.
      - remover o arco okO2 de 46.
      - remover o arco okO1 de 46.
      - remover o vértice 46.
      - remover o arco okN1 de 26.
      - remover o vértice 26.
      - remover o arco okA2 de 14.
      - remover o vértice 14.
      - remover o arco okA1 de 4.
      - remover o vértice 4.
      - remover o arco okA3 de 0.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okA1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
17: [1, {okN1,okO1}, {okA1,okA2,okO2}]
18: [1, {okA1,okO1}, {okA2,okA3,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
27: [1, {okA1,okA3,okO1}, {okN1,okA2,okO2}]
28: [1, {okA1,okA3,okO2}, {}]
34: [1, {okA3,okO1,okO2}, {}]
35: [1, {okN1,okA1,okO1}, {okA2,okA3,okO2}]
37: [2, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
38: [1, {okA1,okA2,okO1}, {okN1,okA3,okO2}]
39: [2, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
45: [1, {okA1,okA2,okO2}, {okN1,okA3,okO1}]
--------------------------------------------------

Expandir vértice 27.

 => A partir do arco okN1.
    * criar o vértice 47 com rótulo {okA2,okO2}
    * alterar vértice 17 de {okA1,okA2,okO2}
      para {okA2,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okO2 de 47.
      - remover o vértice 28.
      - remover o arco okA2 de 47.
```

```
      - remover o vértice 47.
      - remover o arco okO2 de 35.
      - remover o arco okA3 de 35.
      - remover o arco okA2 de 35.
      - remover o vértice 35.
      - remover o arco okA1 de 17.
 => A partir do arco okA2.
    * criar o vértice 48 com rótulo {okN1,okO2}
    * alterar vértice 5 de {okN1,okA1,okO2}
      para {okN1,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okO2 de 18.
      - remover o arco okO2 de 27.
      - remover o arco okO2 de 48.
      - remover o arco okN1 de 48.
      - remover o vértice 48.
      - remover o arco okA2 de 27.
      - remover o vértice 164608.
      - remover o arco okN1 de 27.
      - remover o vértice 27.
      - remover o arco okA3 de 18.
      - remover o arco okO2 de 38.
      - remover o arco okA3 de 38.
      - remover o arco okN1 de 38.
      - remover o vértice 38.
      - remover o arco okA2 de 18.
      - remover o vértice 18.
      - remover o arco okA1 de 5.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA1,okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
37: [2, {okN1,okO1,okO2}, {okA1,okA2,okA3}]
39: [1, {okA1,okO1,okO2}, {okN1,okA2,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
45: [1, {okA1,okA2,okO2}, {okN1,okA3,okO1}]
--------------------------------------------------

Expandir vértice 37.

 => A partir do arco okA1.
    * criar o vértice 49 com rótulo {okA2,okA3}
    * alterar vértice 19 de {okA1,okA2,okA3}
      para {okA2,okA3},
      removendo os arcos em {okA1}.
      - remover o arco okA3 de 39.
      - remover o arco okA2 de 39.
      - remover o arco okN1 de 39.
```

```
      - remover o vértice 39.
      - remover o arco okA1 de 19.
 => A partir do arco okA2.
    * criar o vértice 50 com rótulo {okA1,okA3}
    * alterar vértice 37 de {okA1,okA2,okA3}
      para {okA1,okA3},
      removendo os arcos em {okA2}.
      - remover o arco okA3 de 50.
      - remover o arco okA1 de 50.
      - remover o vértice 50.
      - remover o arco okA2 de 37.
 => A partir do arco okA3.
    * fechar o arco.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA2,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
37: [2, {okN1,okO1,okO2}, {okA1,okA3}]
40: [1, {okA2,okO1,okO2}, {okN1,okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
45: [1, {okA1,okA2,okO2}, {okN1,okA3,okO1}]
49: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
--------------------------------------------------

Expandir vértice 40.

 => A partir do arco okN1.
    * criar o vértice 51 com rótulo {okA1,okA3}
    * alterar vértice 40 de {okN1,okA1,okA3}
      para {okA1,okA3},
      removendo os arcos em {okN1}.
      - remover o arco okA3 de 51.
      - remover o arco okA1 de 51.
      - remover o vértice 51.
      - remover o arco okN1 de 40.
 => A partir do arco okA1.
    * criar o vértice 52 com rótulo {okN1,okA3}
    * alterar vértice 21 de {okN1,okA2,okA3}
      para {okN1,okA3},
      removendo os arcos em {okA2}.
      - remover o arco okO1 de 45.
      - remover o arco okA3 de 45.
      - remover o arco okN1 de 45.
      - remover o vértice 45.
      - remover o arco okA2 de 21.
 => A partir do arco okA3.
    * fechar o arco.

==================== Vértices ====================
```

```
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA2,okA3}]
20: [1, {okN1,okO2}, {okA2,okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
37: [2, {okN1,okO1,okO2}, {okA1,okA3}]
40: [1, {okA2,okO1,okO2}, {okA1,okA3}]
42: [1, {okN1,okA2,okO2}, {okA1,okA3,okO1}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
49: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
52: [1, {okA1,okA2,okO1,okO2}, {okN1,okA3}]
-------------------------------------------------

Expandir vértice 42.

 => A partir do arco okA1.
    * criar o vértice 53 com rótulo {okA3,okO1}
    * alterar vértice 20 de {okA2,okA3,okO1}
      para {okA3,okO1},
      removendo os arcos em {okA2}.
      - remover o arco okO1 de 42.
      - remover o arco okA3 de 42.
      - remover o arco okO1 de 53.
      - remover o arco okA3 de 53.
      - remover o vértice 53.
      - remover o arco okA1 de 42.
      - remover o vértice 42.
      - remover o arco okA2 de 20.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okN1,okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA2,okA3}]
20: [1, {okN1,okO2}, {okA3,okO1}]
21: [1, {okA1,okO2}, {okN1,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
37: [2, {okN1,okO1,okO2}, {okA1,okA3}]
40: [1, {okA2,okO1,okO2}, {okA1,okA3}]
43: [1, {okN1,okA3,okO2}, {okA1,okA2,okO1}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
49: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
52: [1, {okA1,okA2,okO1,okO2}, {okN1,okA3}]
-------------------------------------------------

Expandir vértice 43.

 => A partir do arco okA1.
    * utilizar o rótulo especial.
 => A partir do arco okA2.
    * criar o vértice 55 com rótulo {okA1,okO1}
    * alterar vértice 6 de {okN1,okA1,okO1}
```

```
      para {okA1,okO1},
      removendo os arcos em {okN1}.
      - remover o arco okO1 de 20.
      - remover o arco okO1 de 43.
      - remover o arco okO1 de 55.
      - remover o arco okA1 de 55.
      - remover o vértice 55.
      - remover o arco okA2 de 43.
      - remover o vértice 54.
      - remover o arco okA1 de 43.
      - remover o vértice 43.
      - remover o arco okA3 de 20.
      - remover o vértice 20.
      - remover o arco okN1 de 6.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA2,okA3}]
21: [1, {okA1,okO2}, {okN1,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
37: [1, {okN1,okO1,okO2}, {okA1,okA3}]
40: [1, {okA2,okO1,okO2}, {okA1,okA3}]
44: [1, {okN1,okA1,okO2}, {okA2,okA3,okO1}]
49: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
52: [1, {okA1,okA2,okO1,okO2}, {okN1,okA3}]
--------------------------------------------------

Expandir vértice 44.

 => A partir do arco okA2.
    * criar o vértice 56 com rótulo {okA3,okO1}
    * alterar vértice 44 de {okA2,okA3,okO1}
      para {okA3,okO1},
      removendo os arcos em {okA2}.
      - remover o arco okA3 de 37.
      - remover o arco okA3 de 52.
      - remover o arco okN1 de 52.
      - remover o vértice 52.
      - remover o arco okA3 de 49.
      - remover o arco okA2 de 49.
      - remover o vértice 49.
      - remover o arco okA1 de 37.
      - remover o vértice 37.
      - remover o arco okO1 de 56.
      - remover o arco okA3 de 56.
      - remover o vértice 56.
      - remover o arco okA2 de 44.
 => A partir do arco okA3.
    * utilizar o rótulo especial.
 => A partir do arco okO1.
    * reutilizar o rótulo {okA2,okA3}

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
```

```
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA2,okA3}]
21: [1, {okA1,okO2}, {okN1,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
40: [1, {okA2,okO1,okO2}, {okA1,okA3}]
44: [1, {okN1,okA1,okO2}, {okA3,okO1}]
57: [1, {okN1,okA1,okA3,okO2}, {}]
58: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
-------------------------------------------------

Expandir vértice 58.

 => A partir do arco okA2.
    * criar o vértice 59 com rótulo {okA3}
    * alterar vértice 19 de {okA2,okA3}
      para {okA3},
      removendo os arcos em {okA2}.
      - remover o arco okA3 de 40.
      - remover o vértice 57.
      - remover o arco okA1 de 40.
      - remover o vértice 40.
      - remover o arco okA2 de 19.
 => A partir do arco okA3.
    * fechar o arco.

==================== Vértices ====================
0: [1, {}, {okO1,okO2}]
5: [1, {okO1}, {okN1,okO2}]
6: [1, {okO2}, {okA1,okO1}]
17: [1, {okN1,okO1}, {okA2,okO2}]
19: [2, {okO1,okO2}, {okA3}]
21: [1, {okA1,okO2}, {okN1,okA3}]
34: [1, {okA3,okO1,okO2}, {}]
44: [1, {okN1,okA1,okO2}, {okA3,okO1}]
58: [1, {okN1,okA1,okO1,okO2}, {okA2,okA3}]
59: [1, {okN1,okA1,okA2,okO1,okO2}, {okA3}]
-------------------------------------------------

Expandir vértice 59.

 => A partir do arco okA3.
    * fechar o arco.

================== Diagnósticos ==================

1: {okA3,okO1,okO2}
```

3.6 Fábio

```
Cláusulas

 0: -A -B -OKY F
```

```
 1: A -OKY -F
 2: B -OKY -F
 3: -A -OKU -C
 4: A -OKU C
 5: -B -OKV -D
 6: B -OKV D
 7: -C -D -OKX E
 8: C -OKX -E
 9: D -OKX -E
10: -E -OKZ G
11: -F -OKZ G
12: E F -OKZ -G
13: OKU
14: OKV
15: OKX
16: OKY
17: OKZ
18: A
19: -B
20: G

Rotular raiz com {OKU,OKV,OKX,OKY,OKZ}.

==================== Vértices ====================
0: [1, {}, {OKU,OKV,OKX,OKY,OKZ}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKU.
    * utilizar o rótulo especial.
 => A partir do arco OKV.
    * criar o vértice 2 com rótulo {OKU,OKX,OKY,OKZ}
    * alterar vértice 0 de {OKU,OKV,OKX,OKY,OKZ}
      para {OKU,OKX,OKY,OKZ},
      removendo os arcos em {OKV}.
      - remover o arco OKZ de 2.
      - remover o arco OKY de 2.
      - remover o arco OKX de 2.
      - remover o arco OKU de 2.
      - remover o vértice 2.
      - remover o arco OKV de 0.
 => A partir do arco OKX.
    * utilizar o rótulo especial.
 => A partir do arco OKY.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {OKU}
2: {OKX}
3: {OKY}
4: {OKZ}
```

3.7 Fernando

```
Cláusulas

 0: -A -B -OKX C
 1: A -OKX -C
 2: B -OKX -C
 3: -C -OKY -D
 4: C -OKY D
 5: -D -OKZ -E
 6: D -OKZ E
 7: OKX
 8: OKY
 9: OKZ
10: -A
11: -C
12: E

Rotular raiz com {OKX,OKY,OKZ}.

==================== Vértices ====================
0: [1, {}, {OKX,OKY,OKZ}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKX.
    * criar o vértice 1 com rótulo {OKY,OKZ}
    * alterar vértice 0 de {OKX,OKY,OKZ}
      para {OKY,OKZ},
      removendo os arcos em {OKX}.
      - remover o arco OKZ de 1.
      - remover o arco OKY de 1.
      - remover o vértice 1.
      - remover o arco OKX de 0.
 => A partir do arco OKY.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.

================== Diagnósticos ==================

1: {OKY}
2: {OKZ}
```

3.8 Germano

```
Cláusulas

 0: -A -OKX -D
 1: A -OKX D
 2: -B -D -OKZ F
 3: B -OKZ -F
 4: D -OKZ -F
```

```
 5: -B -OKY -E
 6: B -OKY E
 7: -E -C -OKW G
 8: E -OKW -G
 9: C -OKW -G
10: -F -OKU H
11: -G -OKU H
12: F G -OKU -H
13: OKX
14: OKY
15: OKZ
16: OKW
17: OKU
18: -A
19: C
20: -H

Rotular raiz com {OKX,OKY,OKZ,OKW,OKU}.

==================== Vértices ====================
0: [1, {}, {OKX,OKY,OKZ,OKW,OKU}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKX.
    * utilizar o rótulo especial.
 => A partir do arco OKY.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.
 => A partir do arco OKW.
    * utilizar o rótulo especial.
 => A partir do arco OKU.
    * utilizar o rótulo especial.

================= Diagnósticos ==================

1: {OKX}
2: {OKY}
3: {OKZ}
4: {OKW}
5: {OKU}
```

3.9 Juliana

```
Cláusulas

 0: -c -okN1 -e
 1: c -okN1 e
 2: -b -okO1 f
 3: -e -okO1 f
 4: b e -okO1 -f
 5: -a -b -okA1 d
 6: a -okA1 -d
```

```
 7: b -okA1 -d
 8: -c -okO2 g
 9: -d -okO2 g
10: c d -okO2 -g
11: -f -g -okA2 h
12: f -okA2 -h
13: g -okA2 -h
14: okA1
15: okN1
16: okO1
17: okA2
18: okO2
19: a
20: b
21: c
22: -h

Rotular raiz com {okA1,okN1,okO1,okA2,okO2}.

==================== Vértices ====================
0: [1, {}, {okA1,okN1,okO1,okA2,okO2}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco okA1.
    * criar o vértice 1 com rótulo {okN1,okO1,okA2,okO2}
    * alterar vértice 0 de {okA1,okN1,okO1,okA2,okO2}
      para {okN1,okO1,okA2,okO2},
      removendo os arcos em {okA1}.
      - remover o arco okO2 de 1.
      - remover o arco okA2 de 1.
      - remover o arco okO1 de 1.
      - remover o arco okN1 de 1.
      - remover o vértice 1.
      - remover o arco okA1 de 0.
 => A partir do arco okN1.
    * criar o vértice 2 com rótulo {okA1,okO1,okA2,okO2}
 => A partir do arco okO1.
    * utilizar o rótulo especial.
 => A partir do arco okA2.
    * utilizar o rótulo especial.
 => A partir do arco okO2.
    * utilizar o rótulo especial.

==================== Vértices ====================
0: [1, {}, {okN1,okO1,okA2,okO2}]
2: [1, {okN1}, {okA1,okO1,okA2,okO2}]
3: [1, {okO1}, {}]
4: [1, {okA2}, {}]
5: [1, {okO2}, {}]
--------------------------------------------------

Expandir vértice 2.

 => A partir do arco okA1.
    * criar o vértice 6 com rótulo {okO1,okA2,okO2}
```

```
    * alterar vértice 0 de {okN1,okO1,okA2,okO2}
      para {okO1,okA2,okO2},
      removendo os arcos em {okN1}.
      - remover o arco okO2 de 2.
      - remover o arco okA2 de 2.
      - remover o arco okO1 de 2.
      - remover o arco okO2 de 6.
      - remover o arco okA2 de 6.
      - remover o arco okO1 de 6.
      - remover o vértice 6.
      - remover o arco okA1 de 2.
      - remover o vértice 2.
      - remover o arco okN1 de 0.

================== Diagnósticos ==================

1: {okO1}
2: {okA2}
3: {okO2}
```

3.10 Karina

```
Cláusulas

 0: -P -C -OKX I
 1: P -OKX -I
 2: C -OKX -I
 3: -C -OKY -K
 4: C -OKY K
 5: -K -L -OKW J
 6: K -OKW -J
 7: L -OKW -J
 8: -I -OKZ A
 9: -J -OKZ A
10: I J -OKZ -A
11: -P0 -OKV P
12: -P1 -OKV P
13: -P2 -OKV P
14: -P3 -OKV P
15: P0 P1 P2 P3 -OKV -P
16: OKX
17: OKY
18: OKW
19: OKZ
20: OKV
21: P0
22: -C
23: -L
24: A

Rotular raiz com {OKX,OKY,OKW,OKZ,OKV}.

==================== Vértices ====================
0: [1, {}, {OKX,OKY,OKW,OKZ,OKV}]
--------------------------------------------------
```

```
Expandir vértice 0.

 => A partir do arco OKX.
    * utilizar o rótulo especial.
 => A partir do arco OKY.
    * criar o vértice 2 com rótulo {OKX,OKW,OKZ,OKV}
    * alterar vértice 0 de {OKX,OKY,OKW,OKZ,OKV}
      para {OKX,OKW,OKZ,OKV},
      removendo os arcos em {OKY}.
      - remover o arco OKV de 2.
      - remover o arco OKZ de 2.
      - remover o arco OKW de 2.
      - remover o arco OKX de 2.
      - remover o vértice 2.
      - remover o arco OKY de 0.
 => A partir do arco OKW.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.
 => A partir do arco OKV.
    * criar o vértice 5 com rótulo {OKX,OKY,OKW,OKZ}

==================== Vértices ====================
0: [1, {}, {OKX,OKW,OKZ,OKV}]
1: [1, {OKX}, {}]
3: [1, {OKW}, {}]
4: [1, {OKZ}, {}]
5: [1, {OKV}, {OKX,OKY,OKW,OKZ}]
--------------------------------------------------

Expandir vértice 5.

 => A partir do arco OKX.
    * fechar o arco.
 => A partir do arco OKY.
    * criar o vértice 6 com rótulo {OKX,OKW,OKZ}
    * alterar vértice 0 de {OKX,OKW,OKZ,OKV}
      para {OKX,OKW,OKZ},
      removendo os arcos em {OKV}.
      - remover o arco OKZ de 5.
      - remover o arco OKW de 5.
      - remover o arco OKZ de 6.
      - remover o arco OKW de 6.
      - remover o arco OKX de 6.
      - remover o vértice 6.
      - remover o arco OKY de 5.
      - remover o arco OKX de 5.
      - remover o vértice 5.
      - remover o arco OKV de 0.

================== Diagnósticos ==================

1: {OKX}
2: {OKW}
3: {OKZ}
```

3.11 Seiji

```
Cláusulas

 0: -A -B -OKX -E
 1: A -OKX -E
 2: B -OKX -E
 3: -C -D -OKY F
 4: C -OKY -F
 5: D -OKY -F
 6: -E -OKZ G
 7: -F -OKZ G
 8: E F -OKZ G
 9: -G -OKT -H
10: G -OKT H
11: OKX
12: OKY
13: OKZ
14: OKT
15: A
16: B
17: H

Rotular raiz com {OKX,OKY,OKZ,OKT}.

==================== Vértices ====================
0: [1, {}, {OKX,OKY,OKZ,OKT}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKX.
    * criar o vértice 1 com rótulo {OKY,OKZ,OKT}
    * alterar vértice 0 de {OKX,OKY,OKZ,OKT}
      para {OKY,OKZ,OKT},
      removendo os arcos em {OKX}.
      - remover o arco OKT de 1.
      - remover o arco OKZ de 1.
      - remover o arco OKY de 1.
      - remover o vértice 1.
      - remover o arco OKX de 0.
 => A partir do arco OKY.
    * criar o vértice 2 com rótulo {OKX,OKZ,OKT}
 => A partir do arco OKZ.
    * utilizar o rótulo especial.
 => A partir do arco OKT.
    * utilizar o rótulo especial.

==================== Vértices ====================
0: [1, {}, {OKY,OKZ,OKT}]
2: [1, {OKY}, {OKX,OKZ,OKT}]
3: [1, {OKZ}, {}]
4: [1, {OKT}, {}]
--------------------------------------------------

Expandir vértice 2.
```

```
 => A partir do arco OKX.
    * criar o vértice 5 com rótulo {OKZ,OKT}
    * alterar vértice 0 de {OKY,OKZ,OKT}
      para {OKZ,OKT},
      removendo os arcos em {OKY}.
      - remover o arco OKT de 2.
      - remover o arco OKZ de 2.
      - remover o arco OKT de 5.
      - remover o arco OKZ de 5.
      - remover o vértice 5.
      - remover o arco OKX de 2.
      - remover o vértice 2.
      - remover o arco OKY de 0.

================== Diagnósticos ==================

1: {OKZ}
2: {OKT}
```

3.12 Silvio

```
Cláusulas

 0: -A -OKX C
 1: -B -OKX C
 2: A B -OKX -C
 3: -A -B -OKY D
 4: A -OKY -D
 5: B -OKY -D
 6: -D -OKZ -E
 7: D -OKZ E
 8: -C -E -OKW F
 9: C -OKW -F
10: E -OKW -F
11: OKX
12: OKY
13: OKZ
14: OKW
15: A
16: B
17: F

Rotular raiz com {OKX,OKY,OKZ,OKW}.

=================== Vértices ====================
0: [1, {}, {OKX,OKY,OKZ,OKW}]
-------------------------------------------------

Expandir vértice 0.

 => A partir do arco OKX.
    * criar o vértice 1 com rótulo {OKY,OKZ,OKW}
    * alterar vértice 0 de {OKX,OKY,OKZ,OKW}
      para {OKY,OKZ,OKW},
      removendo os arcos em {OKX}.
```

```
      - remover o arco OKW de 1.
      - remover o arco OKZ de 1.
      - remover o arco OKY de 1.
      - remover o vértice 1.
      - remover o arco OKX de 0.
 => A partir do arco OKY.
    * utilizar o rótulo especial.
 => A partir do arco OKZ.
    * utilizar o rótulo especial.
 => A partir do arco OKW.
    * utilizar o rótulo especial.

================== Diagnósticos ====================

1: {OKY}
2: {OKZ}
3: {OKW}
```

3.13 Thiago

```
Cláusulas

 0: -A -okX D
 1: -B -okX D
 2: A B -okX -D
 3: -C -okY -E
 4: C -okY E
 5: -D -E okZ J
 6: D -okZ -J
 7: E -okZ -J
 8: -J -okW -G
 9: J -okW G
10: okX
11: okY
12: okZ
13: okW
14: C
15: -G

Rotular raiz com {okX,okY,okZ,okW}.

==================== Vértices ====================
0: [1, {}, {okX,okY,okZ,okW}]
--------------------------------------------------

Expandir vértice 0.

 => A partir do arco okX.
    * criar o vértice 1 com rótulo {okY,okZ,okW}
    * alterar vértice 0 de {okX,okY,okZ,okW}
      para {okY,okZ,okW},
      removendo os arcos em {okX}.
      - remover o arco okW de 1.
      - remover o arco okZ de 1.
      - remover o arco okY de 1.
```

```
      - remover o vértice 1.
      - remover o arco okX de 0.
 => A partir do arco okY.
    * utilizar o rótulo especial.
 => A partir do arco okZ.
    * utilizar o rótulo especial.
 => A partir do arco okW.
    * utilizar o rótulo especial.

================== Diagnósticos ====================

1: {okY}
2: {okZ}
3: {okW}
```

A Código-fonte do programa `diag`

```
/*-------------------------------------------------------------------------+
| diag.c - Diagnóstico baseado em modelo formal            (slago@ime.usp.br) |
+-------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <string.h>

/*-------------------------------------------------------------------------+
| Constantes e macros                                                      |
+-------------------------------------------------------------------------*/

#define strncmpi     strncasecmp
#define MAXATOMS     (2*8*sizeof(long)+1)

/*  Na tabela de símbolos, as variáveis do tipo "OK*" são alocadas nas primeiras
 *  32 posições. Isso permite que, no algoritmo de Reiter, os conjuntos conflito
 *  sejam representados por vetores de 32 bits. Por exemplo, se as variáveis são
 *  V, W, X, Y e Z, o conjunto {V,Y,Z} é representado pelo binário 11001. Assim,
 *  todas as operações sobre conjuntos podem ser realizadas, eficientemente, com
 *  o uso de operações bit-a-bit, conforme definidas a seguir:
 */

#define EMPTYSET     (0)
#define set(x)       (1<<(x))
#define in(x,S)      ((1<<(x))&S)
#define diff(S,T)    ((S)&~(T))
#define inter(S,T)   ((S)&(T))
#define union(S,T)   ((S)|(T))
#define subset(S,T)  (((S)&(T))==S)
#define psubset(S,T) (subset(S,T) && (S)!=(T))
#define emptyset(S)  ((S)==0)

/*-------------------------------------------------------------------------+
| Tipos de dados                                                           |
+-------------------------------------------------------------------------*/
```

```
typedef struct tree {
  int number;
  char *atom;
  struct tree *left;
  struct tree *right;
} *Tree;

typedef enum {
  false,
  true,
  undef
} Bool;

typedef struct list {
  short        item;
  struct list *next;
}*List;

typedef struct clause {
  Bool  active;
  short size;
  short totf;
  short tott;
  Bool  value;
  List  disjuncts;
} Clause;

typedef struct atom {
  Bool value;
  List pos;
  List neg;
} Atom;

typedef unsigned Set;

typedef struct arc {
  int label;
  struct vertex *vertex;
  struct arc *next;
} *Arc;

typedef struct vertex {
  int num;
  int ref;
  Set path;
  Set label;
  Arc arcs;
} *Vertex;

typedef struct qlist {
  Vertex vertex;
  struct qlist *next;
} *Qlist;

typedef struct queue {
  Qlist first;
  Qlist last;
```

```
} Queue;


/*-------------------------------------------------------------------------+
| Variáveis globais                                                        |
+-------------------------------------------------------------------------*/

static Tree    symTable  = NULL;                      /* tabela de símbolos */
static int     varAtoms  = 0;                  /* número de variáveis em VAR */
static int     obsAtoms  = 0;                  /* número de variáveis em OBS */
static int     sd_cls    = 0;                  /* número de cláusulas em SD  */
static int     maxcls    = 0;              /* total de cláusulas no problema */
static int     maxvar    = 0;              /* total de variáveis no problema */
static char   *strbool[] = {"f","t","u"};
static int     numcls    = 0;                  /* para numeração de cláusulas */
static Clause *cnf       = NULL;                /* forma normal conjuntiva */
static Atom    atomtb[MAXATOMS];             /* tabela de descrição de átomos */
static Queue   vtxList   = {NULL,NULL};          /* lista de vértices no DAG */
static Queue   queue     = {NULL,NULL};         /* fila para busca em largura */
static Bool    trace     = false;        /* para acompanhar construção do DAG */

/*-------------------------------------------------------------------------+
| Implementação da tabela de símbolos                                      |
+-------------------------------------------------------------------------*/

/*
 *  Insere um átomo na tabela de símbolos e devolve seu número.
 */

int insTree(char *atom, Tree *T) {
  static int number = 0;

  if( *T==NULL ) {
    *T = malloc(sizeof(struct tree));
    assert(*T );
    (*T)->number = ++number;
    (*T)->atom   = strdup(atom);
    (*T)->left   = NULL;
    (*T)->right  = NULL;
    return number;
  }

  if( strcmp(atom,(*T)->atom)==0 ) return (*T)->number;
  if( strcmp(atom,(*T)->atom)< 0 ) return insTree(atom,&(*T)->left);
  return insTree(atom,&(*T)->right);
}

/*
 *  Lê seção SD, VAR ou OBS, de um arquivo de problema.
 */

int getSection(char *header, FILE *in) {
  char buf[80], *lit;
  int  m=0, n;

  while( fgets(buf,511,in) && strncmpi(buf,header,strlen(header)) );
```

```
  while( fgets(buf,511,in) && strncmpi(buf,"sd",2) &&
         strncmpi(buf,"var",3) && strncmpi(buf,"obs",3) ) {
    lit = strtok(buf,"\r\n ");
    if( lit ) maxcls++;
    while( lit ) {
      n = insTree(*lit=='-' ? lit+1 : lit, &symTable);
      if( n>m ) m = n;
      lit = strtok(NULL,"\r\n ");
    }
  }

  rewind(in);
  return m;
}

/*
 *  Monta a tabela de símbolos e atualiza variáveis globais.
 */

void getSymTable(FILE *in) {
  varAtoms = getSection("VAR", in);
  obsAtoms = getSection("OBS", in) - varAtoms;
  maxvar = getSection("SD",in);
  sd_cls = maxcls - varAtoms - obsAtoms;

  assert( varAtoms <= 32 );
  assert( maxvar <= MAXATOMS );
}

/*
 *  Devolve o átomo correpondente a um dado número.
 */

char *getAtom(int num, Tree T) {
   char *atom;
   if( !T ) return NULL;
   if( num == T->number ) return T->atom;
   if( atom = getAtom(num, T->left) ) return atom;
   return getAtom(num, T->right);
}


/*-------------------------------------------------------------------------+
| Implementação da Forma Normal Conjuntiva                                 |
+-------------------------------------------------------------------------*/

/*
 *  Insere um literal numa disjunção.
 */

void ins(short item, List *tail) {
  List head = malloc(sizeof(struct list));
  assert( head );
  head->item = item;
  head->next = *tail;
  *tail = head;
}
```

```
/*
 *  Reinicia a valoração da FNC.
 */

void resetCNF(void) {
  int i;

  assert( cnf );

  for(i=0; i<maxcls; i++) {
     cnf[i].active = true;
     cnf[i].totf = 0;
     cnf[i].tott = 0;
     cnf[i].value = undef;
  }

  for(i=1; i<MAXATOMS; i++)
    atomtb[i].value = undef;
}

/*
 *  Lê a FNC de uma arquivo em disco.
 */

void getCnf(char *fname) {
  FILE *in;
  char buf[512], *literal;

  if( (in=fopen(fname,"rt")) == NULL ) {
    fprintf(stderr,"Arquivo não encontrado.\n");
    exit(1);
  }

  getSymTable(in);

  cnf = malloc(maxcls*sizeof(struct clause));
  assert( cnf );
  numcls=0;

  while( numcls<maxcls && fgets(buf,511,in) ) {
    if(strncmpi(buf,"sd",2) && strncmpi(buf,"var",3) && strncmpi(buf,"obs",3)) {
      cnf[numcls].active = true;
      cnf[numcls].size = 0;
      cnf[numcls].totf = 0;
      cnf[numcls].tott = 0;
      cnf[numcls].value = undef;
      cnf[numcls].disjuncts = NULL;
      literal = strtok(buf,"\r\n ");
      if( !literal ) continue;
      while( literal ) {
        short signal = literal[0]=='-' ? -1 : +1;
        short atom = insTree( signal==-1 ? literal+1 : literal, &symTable);
        ins(signal*atom,&cnf[numcls].disjuncts);
        cnf[numcls].size++;
        atomtb[atom].value = undef;
        ins(numcls, signal>0 ? &atomtb[atom].pos : &atomtb[atom].neg);
```

```
        literal = strtok(NULL,"\r\n ");
      }
      numcls++;
    }
  }

  fclose(in);
}

/*
 *  Mostra uma cláusula no vídeo.
 */

void show(List list) {
  if( !list ) return;
  show(list->next);
  if( list->item<0 ) printf("-");
  printf("%s ", getAtom(abs(list->item),symTable) );
}

/*
 *  Mostra a FNC no vídeo.
 */

void putCnf() {
  int i;
  List list;

  system("clear");
  printf("Cláusulas\n\n");

  for(i=0; i<maxcls; i++) {
    printf("%2d: ", i);
    show(cnf[i].disjuncts);
    printf("\n");
  }

  fprintf(stderr,"\nPressione <enter>...");
  getchar();
  system("clear");
}

/*-------------------------------------------------------------------------+
| Implementação do DPL                                                     |
+-------------------------------------------------------------------------*/

/*
 *  Escolhe um átomo para bifurcar a árvore de busca de valoração.
 */

int chooseAtom(void) {
  int i;
  List p;

  /* prioriza cláusulas unitárias */

  for(i=0; i<maxcls; i++)
```

```
    if( cnf[i].active && cnf[i].totf==cnf[i].size-1 && cnf[i].value==undef) {
      for(p=cnf[i].disjuncts; p!=NULL; p=p->next)
        if( atomtb[abs(p->item)].value==undef )
          return abs(p->item);
    }

  /* escolhe o primeiro átomo com valor indefinido */

  for(i=1; i<=maxvar; i++)
    if( atomtb[i].value == undef )
      return i;
}

/*
 *  Tenta encontrar um modelo para a FNC.
 */

Bool DPL(unsigned *C) {
  int i, ttrue=0, guess;
  List p;

  for(i=0; i<maxcls; i++) {

    if( !cnf[i].active ) {
       ttrue++;
       continue;
    }

    switch( cnf[i].value ) {
      case false: /* coleta variável para diagnóstico */
                  if( cnf[i].size == 1 ) {
                    int atom = abs(cnf[i].disjuncts->item);
                    if( atom>=1 && atom <= varAtoms)
                      *C |= 1<< (atom-1);
                  }
                  return false;
      case true : ttrue++;
    }
  }

  if( ttrue==maxcls ) return true;

  /* escolhe um átomo para bifurcar */

  i = chooseAtom();

  if( i>maxvar ) return false;

  for(guess=1; guess>=0; guess--) {

    /* atribui um valor para o átomo */

    atomtb[i].value = guess;

    /* atualiza a valoração */

    p = (guess==true) ? atomtb[i].pos : atomtb[i].neg;
```

```
    for( ; p!=NULL; p=p->next) {
      cnf[p->item].tott++;
      cnf[p->item].value = true;
    }

    p = (guess==true) ? atomtb[i].neg : atomtb[i].pos;

    for( ; p!=NULL; p=p->next) {
      cnf[p->item].totf++;
      if( cnf[p->item].totf == cnf[p->item].size )
        cnf[p->item].value = false;
    }

    /* verifica se a valoração é um modelo */

    if( DPL(C) == true ) return true;

    /* RETROCESSO: restaura a valoração */

    atomtb[i].value = undef;

    p = (guess==true) ? atomtb[i].pos : atomtb[i].neg;

    for( ; p!=NULL; p=p->next) {
      cnf[p->item].tott--;
      if( cnf[p->item].tott == 0 )
        cnf[p->item].value = undef;
    }

    p = (guess==true) ? atomtb[i].neg : atomtb[i].pos;

    for( ; p!=NULL; p=p->next) {
      cnf[p->item].totf--;
      if( cnf[p->item].tott==0 )
        cnf[p->item].value = undef;
    }
  }

  return false;
}

/*-----------------------------------------------------------------------------+
| Implementação do algoritmo de diagnóstico de REITER                          |
+-----------------------------------------------------------------------------*/

/*
 *  Verifica se fila para busca em largura está vazia.
 */

int isempty(Queue *queue) {
  return queue->first==NULL;
}

/*
 *  Insere um item no final da fila.
 */
```

```
void enqueue(Vertex vertex, Queue *queue) {
  Qlist item = malloc(sizeof(struct qlist));

  assert( item );
  item->vertex = vertex;
  item->next = NULL;

  if( queue->first==NULL ) queue->first = item;
  else  queue->last->next = item;

  queue->last = item;
}

/*
 *  Remove um item do início da fila.
 */

Vertex dequeue(Queue *queue) {
  Vertex vertex;
  Qlist item;
  assert( queue->first );
  item = queue->first;
  vertex = item->vertex;
  queue->first = item->next;
  free(item);
  return vertex;
}

/*
 *  Remove um item de qualquer posição da fila.
 */

void qkill(Vertex v, Queue *queue) {
  Qlist item = queue->first;

  if( !item ) return;

  if( item && item->vertex==v ) {
    queue->first = item->next;
    free(item);
  }
  else {
    while( item->next ) {
      if( item->next->vertex==v ) {
        Qlist p= item->next;
        item->next = p->next;
        free(p);
        break;
      }
      item = item->next;
    }
  }

  if( queue->first==NULL )
    queue->last=NULL;
  else {
```

```
    item = queue->first;
    while( item->next ) item = item->next;
    queue->last = item;
  }
}

/*
 *  Cria uma lista de arcos rotulados com elementos do conjunto S.
 */

Arc arcList(Set S) {
  Arc head, tail=NULL;
  int x;

  for(x=31; x>=0; x--) {
    if( in(x,S) ) {
      head = malloc(sizeof(struct arc));
      assert( head );
      head->label = x;
      head->next = tail;
      tail = head;
    }
  }

  return tail;
}

/*
 *  Mostra uma lista de arcos no vídeo.
 */

void showArcs(Arc arc) {
  printf("{");
  while( arc!=NULL ) {
    printf("%s", getAtom(arc->label+1,symTable));
    arc = arc->next;
    if( arc!=NULL ) printf(",");
  }
  printf("}");
}

/*
 *  Mostra os elementos de um conjunto no vídeo.
 */

void showSet(Set S) {
  int x, j=0;
  printf("{");
  for(x=0; x<=31; x++)
    if( in(x,S) ) {
      if(j && x<31) printf(",");
      printf("%s", getAtom(x+1,symTable));
      j=1;
    }
  printf("}");
}
```

```
/*
 *  Mostra um vértice no vídeo.
 */

void showVertex(Vertex v) {
  assert( v );
  printf("%d: [%d, ", v->num, v->ref);
  showSet(v->path);
  printf(", ");
  showArcs(v->arcs);
  printf("]\n");
}

/*
 *  Mostra todos os vértices existentes no DAG.
 */

void showVertices(void) {
  Qlist item = vtxList.first;
  assert( item );
  while( item ) {
    showVertex(item->vertex);
    item = item->next;
  }
}

/*
 *  Cria um novo vértice e o registra na lista de vértices.
 */

Vertex newVertex(Set H, Set S) {
  static int num = 0;
  Vertex vertex = malloc(sizeof(struct vertex));

  assert( vertex );
  vertex->num = num++;
  vertex->ref = 1;
  vertex->path = H;
  vertex->label = S;
  vertex->arcs = arcList(S);

  /* registra vértice na lista global */

  enqueue(vertex, &vtxList);

  return vertex;
}

/*
 *  Encontra um conjunto conflito C tal que inter(C,H) é vazio.
 */

Set findSet(Set H) {
  int x;
  Set C = EMPTYSET;

  resetCNF();
```

```
  /* destiva as cláusulas na FNC correspondentes aos elementos em H */

  for(x=31; x>=0; x--)
   if( in(x,H) )
     cnf[sd_cls+x].active = false;

  return DPL(&C)==false ? C : EMPTYSET;
}

/*
 *  Verifica se já há um outro nó no DAG cujo caminho é H.
 */

Vertex reuseVertex(Set H) {
  Qlist item = vtxList.first;

  while( item ) {
    if( item->vertex->path==H ) return item->vertex;
    item = item->next;
  }

  return NULL;
}

/*
 *  Verifica se já há um outro nó no DAG cujo caminho é subconjunto de H.
 */

Vertex closeNode(Set H) {
  Qlist item = vtxList.first;

  while( item ) {
    if( item->vertex->label==0 && subset(item->vertex->path,H) ) return item->vertex;
    item = item->next;
  }

  return NULL;
}

/*
 *  Verifica se já há um outro nó cujo rótulo pode ser reutilizado.
 */

Set reuseLabel(Set H) {
  Qlist item = vtxList.first;

  while( item ) {
    if( inter(item->vertex->label,H)==0 ) return item->vertex->label;
    item = item->next;
  }

  return EMPTYSET;
}

/*
 *  Verifica se há um vértice cujos arcos possam ser podados.
```

```
 */

Vertex shrink(Vertex v) {
  Qlist item = vtxList.first;

  while( item ) {
    if( v->num > item->vertex->num  &&
        psubset(v->label,item->vertex->label) )
      return item->vertex;
    item = item->next;
  }

  return NULL;
}

/*
 *  Elimina arcos e vértices do DAG.
 */

void trim(int num,Arc *arc, Set S) {
  if( *arc==NULL ) return;

  trim(num, &(*arc)->next,S);

  if( in((*arc)->label,S) ) {
    Arc a;

    if( (*arc)->vertex ) {
      if( (*arc)->vertex->ref == 1 ) {
        trim((*arc)->vertex->num,&(*arc)->vertex->arcs,~0);
        if( trace )
          printf("      - remover o vértice %d.\n", (*arc)->vertex->num);
        qkill((*arc)->vertex,&vtxList);
        qkill((*arc)->vertex,&queue);
        free((*arc)->vertex);
        (*arc)->vertex = NULL;
      }
      else  (*arc)->vertex->ref--;
    }

    if( trace )
      printf("      - remover o arco %s de %d.\n",
           getAtom((*arc)->label+1,symTable), num);

    a = (*arc);
    (*arc) = (*arc)->next;
    free(a);
  }
}

/*
 *  Constrói DAG conforme algoritmo de REITER.
 */

void DAG(char *fname) {
  Vertex v, w, a;
  Set S;
```

```
getCnf(fname);

if( trace ) putCnf();

S=findSet(EMPTYSET);

if( trace ) {
  printf("Rotular raiz com ");
  showSet(S);
  printf(".\n");
}

enqueue( v=newVertex(EMPTYSET,S), &queue );

while( !isempty(&queue) ) {
  Arc arc;

  v = dequeue(&queue);

  if( v->label==EMPTYSET ) continue;

  if( trace ) {
    puts("\n==================== Vértices ====================");
    showVertices();
    puts("--------------------------------------------------\n");
    printf("Expandir vértice %d.\n\n", v->num);
  }

  for(arc=v->arcs; arc; arc=arc->next) {
    Set H;

    if( trace )
      printf(" => A partir do arco %s.\n", getAtom(arc->label+1,symTable));

    H = union(v->path,set(arc->label));

    if( w=reuseVertex(H) ) {
      if( trace ) printf("    * reutilizar o vértice %d.\n", w->num);
      w->ref++;
      arc->vertex = w;
    }
    else
      if( closeNode(H) ) {
        if( trace ) printf("    * fechar o arco.\n");
        arc->vertex = NULL;
      }
      else
        if( S=reuseLabel(H) ) {
          if( trace ) {
            printf("    * reutilizar o rótulo ");
            showSet(S);
            printf("\n");
          }
          enqueue(arc->vertex = newVertex(H,S), &queue );
        }
        else
```

```
            if( S=findSet(H) ) {
              enqueue(arc->vertex = a = newVertex(H,S), &queue );
              if( trace ) {
                printf("    * criar o vértice %d com rótulo ", a->num);
                showSet(S);
                printf("\n");
              }
              if( w=shrink(a) ) {
                if( trace ) {
                  printf("    * alterar vértice %d de ", w->num);
                  showSet(w->label);
                  printf("\n      para ");
                  showSet(S);
                  printf(",\n      removendo os arcos em ");
                  showSet(diff(w->label,S));
                  printf(".\n");
                }
                trim(w->num,&w->arcs,diff(w->label,S));
                w->label = S;
              }
            }
            else {
              if( trace ) printf("    * utilizar o rótulo especial.\n");
              enqueue(arc->vertex = newVertex(H,EMPTYSET), &queue );
            }
        if( trace ) getchar();
      }
    }
  }

  /*
   *  Encontra diagnósticos.
   */

  void diagnostics(char *fname) {
    Qlist item;
    int i=1, x;
    char *s;
    FILE *out;

    /* constrói DAG para o problema */

    if( trace ) system("clear");

    DAG(fname);

    /* os diagnósticos são gravados num arquivo com extensão .dia */

    s = strchr(fname,'.');

    if( s ) strcpy(s,".dia");
    else    strcat(fname,".dia");

    if( (out=fopen(fname,"wt")) == NULL ) {
      fprintf(stderr,"Arquivo não pode ser criado.\n");
      exit(2);
    }
```

```
  if( trace ) puts("================== Diagnósticos ==================\n");

  item = vtxList.first;

  while( item ) {

    assert( item->vertex );

    if( emptyset(item->vertex->label) ) {
      if( trace ) {
        printf("%d: ", i++);
        showSet(item->vertex->path);
        printf("\n");
      }

      /* grava diagnóstico no arquivo de saída */

      for(x=0; x<=31; x++)
        if( in(x,item->vertex->path) )
          fprintf(out,"%s ", getAtom(x+1,symTable));

      fprintf(out,"\n");
    }

    item = item->next;
  }

  fclose(out);
  fprintf(stderr,"\nDiagnósticos gravados no arquivo %s\n\n", fname);
}


/*-------------------------------------------------------------------------+
| Programa principal                                                       |
+-------------------------------------------------------------------------*/

int main(int ac, char *av[]) {

  if( ac<2 || ac>3 ) {
    fprintf(stderr,"\nUso: diag <problema> [-r]\n\n");
    fprintf(stderr,"-r : rastreia construção do DAG\n\n");
    exit(1);
  }

  if( ac==3 && strcmp(av[2],"-r")==0 ) trace = true;

  diagnostics(av[1]);

  return 0;
}
```