SATISFATIBILIDADE NÃO-CLAUSAL RESTRITA ÀS VARIVÁVEIS DE ENTRADA

Bruno César Ribas

Laboratório de Inteligência Artifical e Métodos Formais

13 de abril de 2011

1. Introdução
   - Visão Geral
2. Satisfatibilidade
   - Heurísticas para ramificação
   - BCP rápido
   - Retrocesso não-cronológico
3. Satisfatibilidade em Fórmulas Não-Clausais
   - CNF
   - NNF
   - ISCAS
   - Satisfatibilidade em NNF
   - NOCLAUSE
4. LIAMFSAT
5. Avaliação Experimental
6. Conclusão e trabalhos futuros

Visão Geral

- Sistemas computacionais usados em aplicações críticas;
  - ex: automóveis, aviões, trens
- garantir o funcionamento é crucial;
- a complexidade de garantia de funcionamento aumenta com a complexidade do sistema;
- verificação automática necessária;
- descrições formais de circuitos podem ser convertidas para instâncias SAT e verificadas por um resolvedor SAT;

Visão Geral

Definição Satisfatibilidade

É o problema de decidir se uma fórmula é satisfatível, ou seja, verificar se há uma valoração para as variáveis da formula que a torne verdadeira.

Visão Geral - Definições

Literal

é uma variável x ou a sua negação $\neg x$

Literal Puro

é o literal que aparece em apenas uma forma em toda a fórmula

Visão Geral - Definições

Cláusula

é a disjunção de literais

- ex: $(x_1 \vee x_2)$

Cláusula Unitária

é uma cláusula representada por apenas um literal

- ex: x_1

CNF - Forma Normal Conjuntiva

É a conjunção de cláusulas

- ex: $(p \vee q) \wedge (q \vee \neg p) \wedge (\neg p \vee \neg q)$

Satisfatibilidade

- Tabela verdade
- Primeiro Algoritmo em 1960, DP
- DPLL refinamento em 1962
- Algoritmos completos baseados no DPLL
- SATO(1996), GRASP(1996), ZCHAFF(2001), MINISAT(2003)
  - Otimizações do DPLL

DPLL

Tautologia

Remove Cláusulas Tautológicas

Cláusula Unitária

Remove Cláusulas Unitárias

Literal Puro

Remove Literais Puros

Ramificação

Escolhe uma variável e assinala Verdadeiro ou Falso e abre a fórmula em duas: um lado com x verdadeiro; outro com x falso

DPLL

Decisão

É o análogo da Ramificação, onde alguma variável é escolhida e valorada

Nível

Cada decisão corresponde a um nível de decisão

BCP (Boolean Constraint Propagation)

É o algoritmo que propaga o efeito da valoração de uma variável.
Corresponde às regras: Cláusula Unitária, Literal Puro e Tautologia

Retrocesso

Sempre que o BCP encontrar alguma cláusula falsa o retrocesso é feito

DPLL

```
1  enquanto 1==1 faça
2      se Decide() == OK então
3          BCP()
4          se BCPRetornouConflito() == SIM então
5              se Diagnostico() == NaoPodeSerResolvido então
6                  retorna INSTATISFATIVEL
7              fim
8              senão
9                  RETROCESSO()
0              fim
1          fim
2      fim
3      senão
4          retorna SAT
5      fim
6  fim
```

DPLL - Pontos Críticos

- Escolha de variável
- Propagação da valoração
- Retrocesso

Heurísticas para ramificação

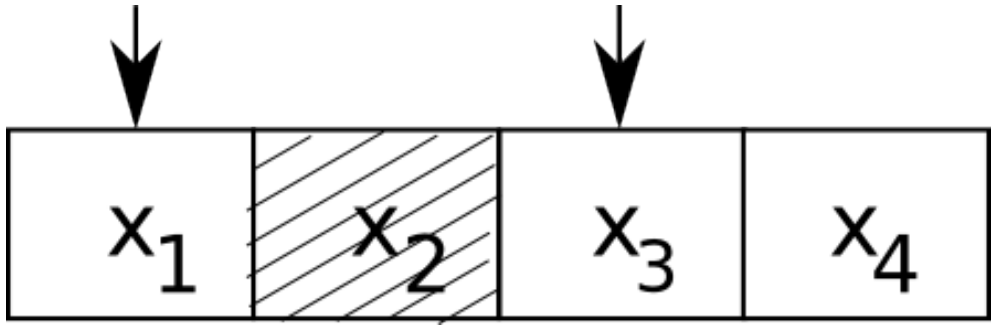
- Consome até 10% do tempo de processamento
- Escolha aleatória (RAND)
- MOM(1995), BOHM(1992)
- VSIDS(2001) (*Variable State Independent Decaying Sum*)
  - Maior eficiência com informação dinâmica
  - Melhorou desempenho em uma ordem de grandeza;

Heurística para ramificação VSIDS

1. Todo literal tem um contador iniciado como 0;
2. Quando uma cláusula nova é adicionada, o contador associado com cada literal presente na cláusula é incrementado;
3. A variável (não valorada) e seu literal com o contador mais alto é escolhida para a decisão;
4. Periodicamente todos os contadores são divididos por uma constante.

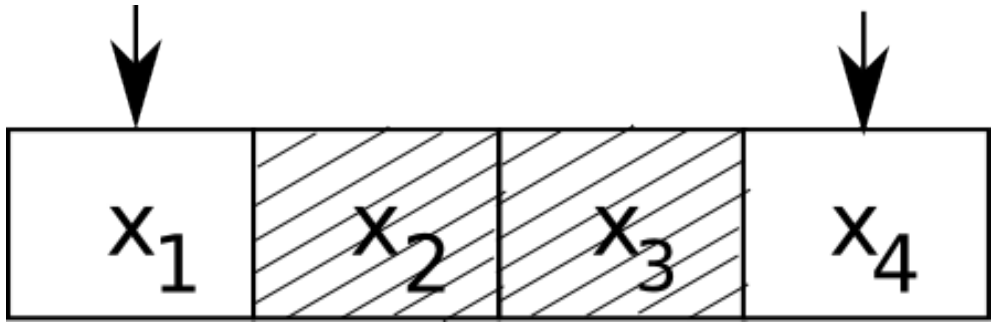
BCP rápido

- Consome 80% do tempo de processamento
- Lista de observação
- Estado crítico de uma cláusula
- Inferência



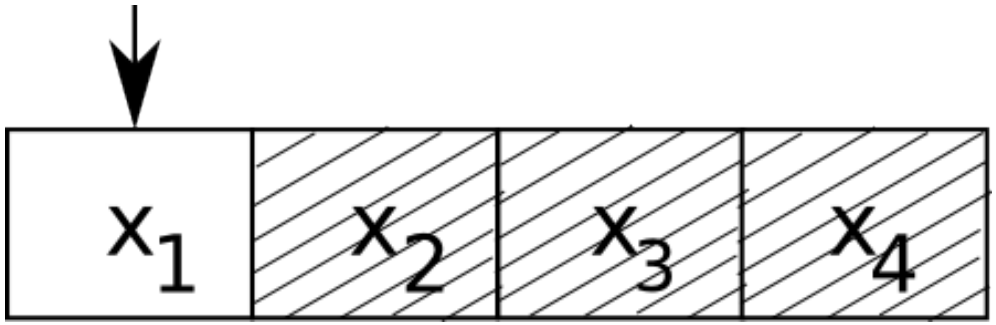
BCP rápido

- Consome 80% do tempo de processamento
- Lista de observação
- Estado crítico de uma cláusula
- Inferência



BCP rápido

- Consome 80% do tempo de processamento
- Lista de observação
- Estado crítico de uma cláusula
- Inferência



Retrocesso não-cronológico

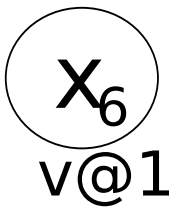
- Retrocesso para um nível de decisão mais distante
- Aprender valorações parciais conflitantes, evita repetir erros
- Um grafo é mantido com as decisões e implicações
- Atualizado sempre a cada Decisão e execução do BCP

Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$

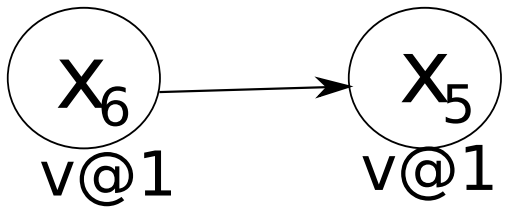
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



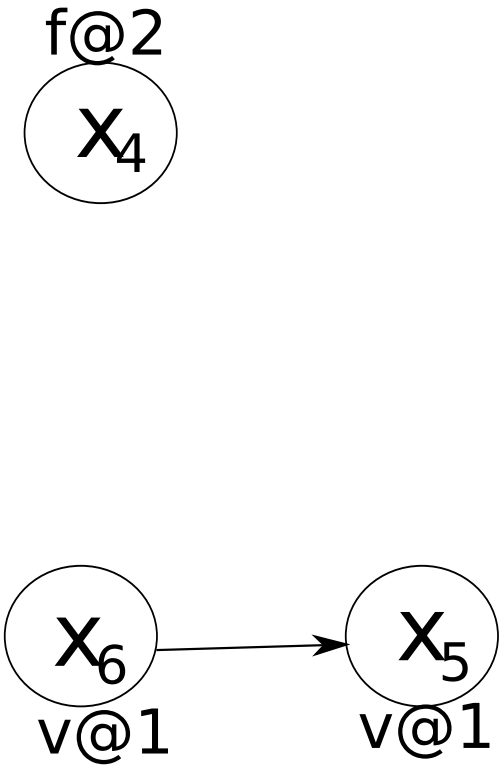
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



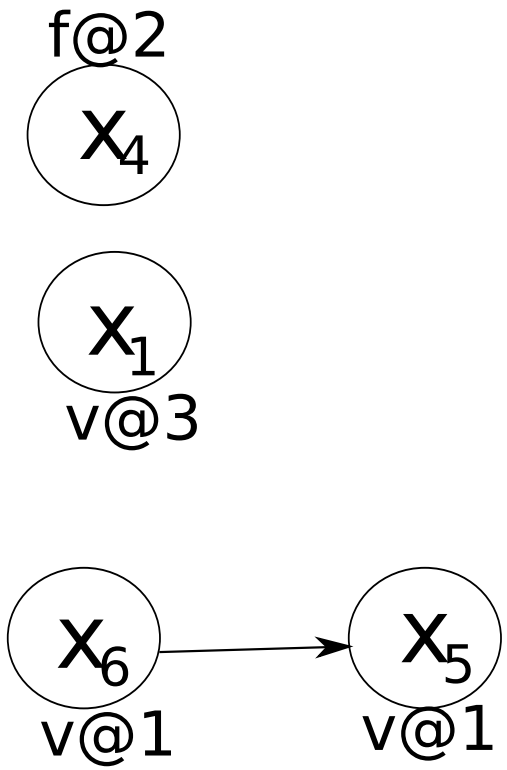
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



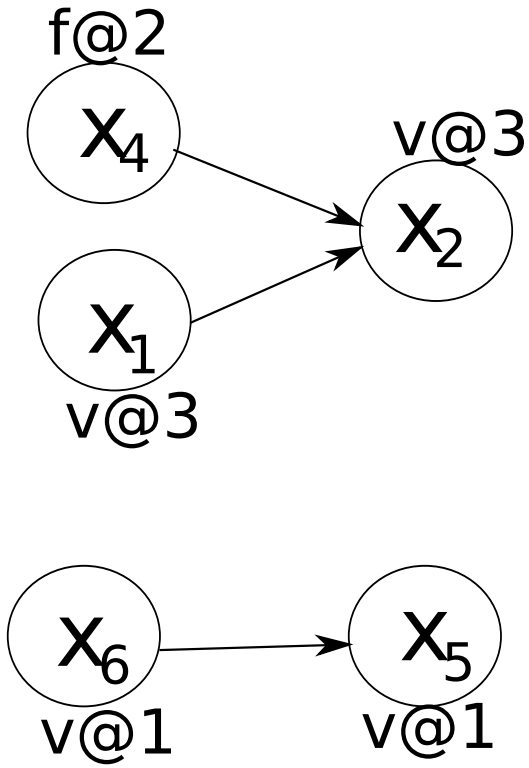
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



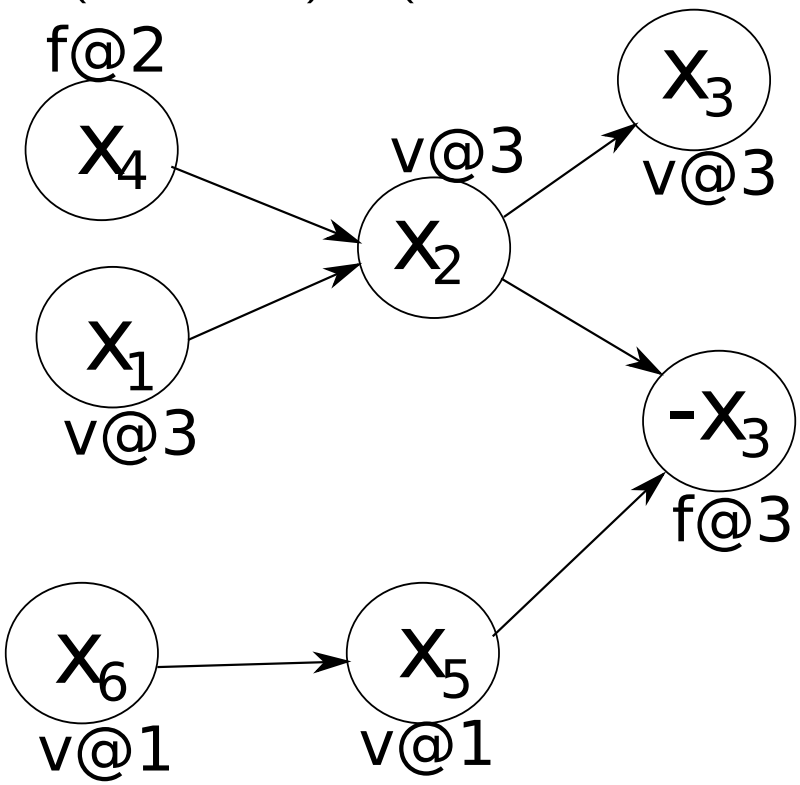
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



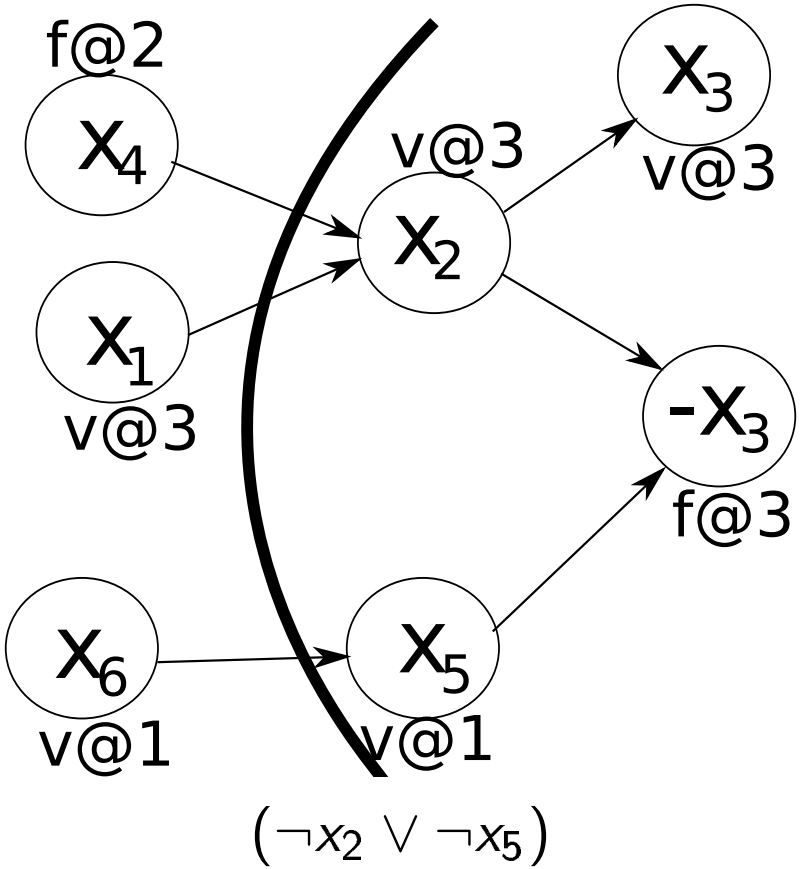
Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



Retrocesso não-cronológico

$$(\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_5 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_6 \vee x_5)$$



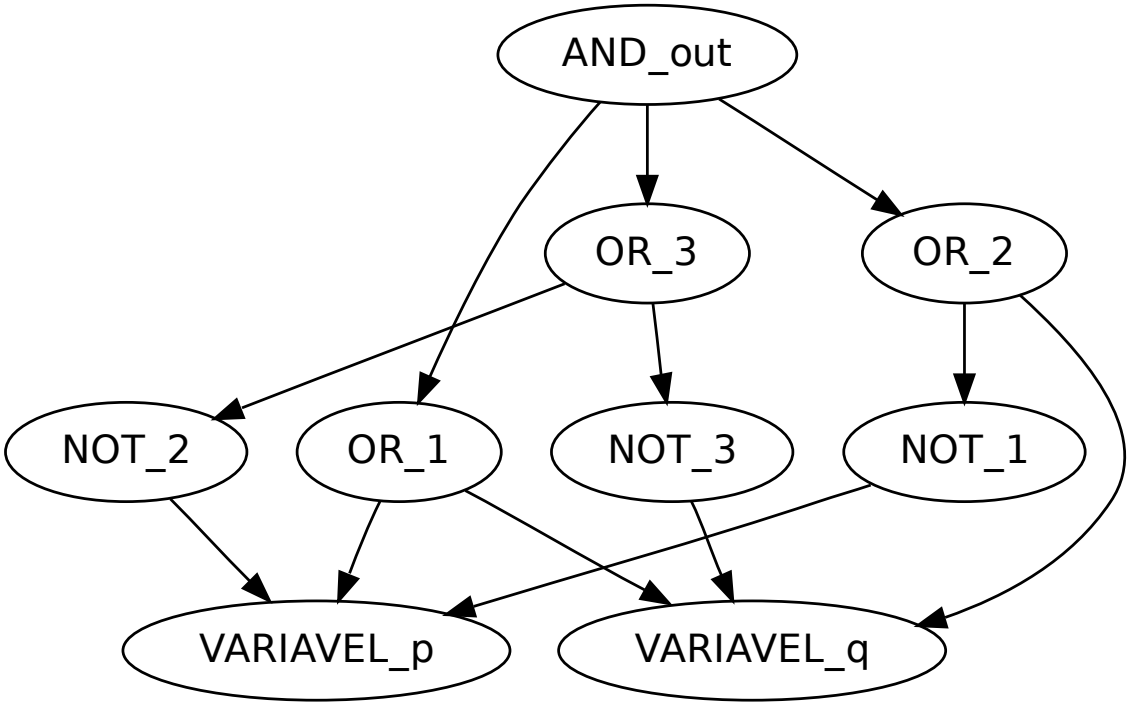
Satisfatibilidade em Fórmulas Não-Clausais

- Representação Não-Clausal mais abrangente que CNF
  - NNF - Representação mais restrita que o ISCAS
  - ISCAS - Representação abrangente
- Duas abordagens:
  - diretamente no grafo original
  - conversão para outra representação não-clausal
- NOCLAUSE(2004), SATMATE(2006), NFLSAT(2009)

CNF - Conjuctive Normal Formula

- DAG
- Nós Folha:
  - *Verdadeiro*
  - *Falso*
  - X
  - $\neg X$
- Raiz:
  - $\wedge$ (E)
- Nós internos:
  - $\vee$ (OU)

CNF - Exemplo

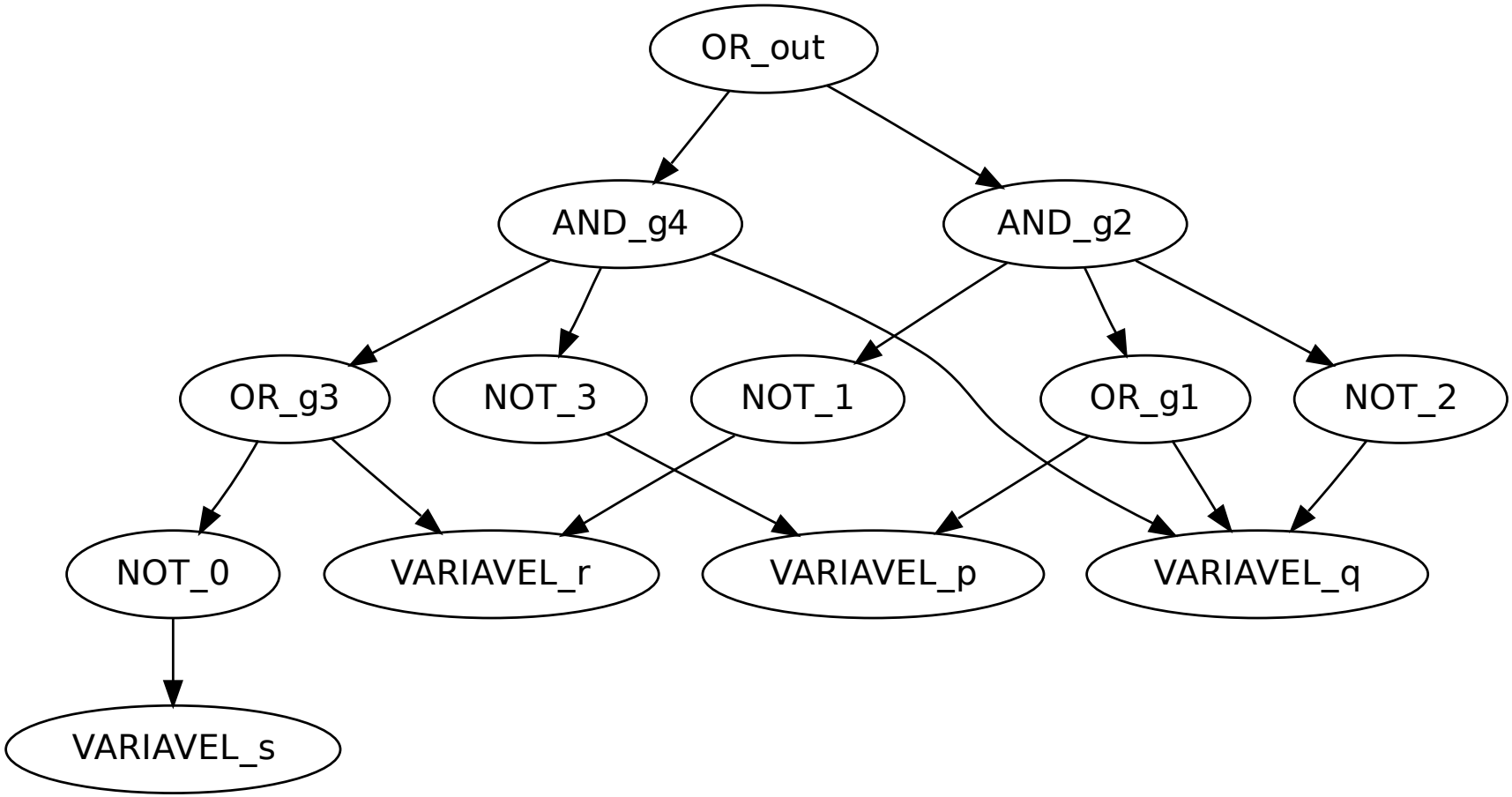


$$(p \vee q) \wedge (q \vee \neg p) \wedge (\neg p \vee \neg q)$$

NNF - Negation Normal Formula

- DAG
- Nós Folha:
  - *Verdadeiro*
  - *Falso*
  - X
  - $\neg X$
- Nós Internos:
  - $\wedge$ (E)
  - $\vee$ (OU)
- Nó interno pode possuir apenas 1 nó pai
- Nó interno chamado de nó operador

NNF - Exemplo

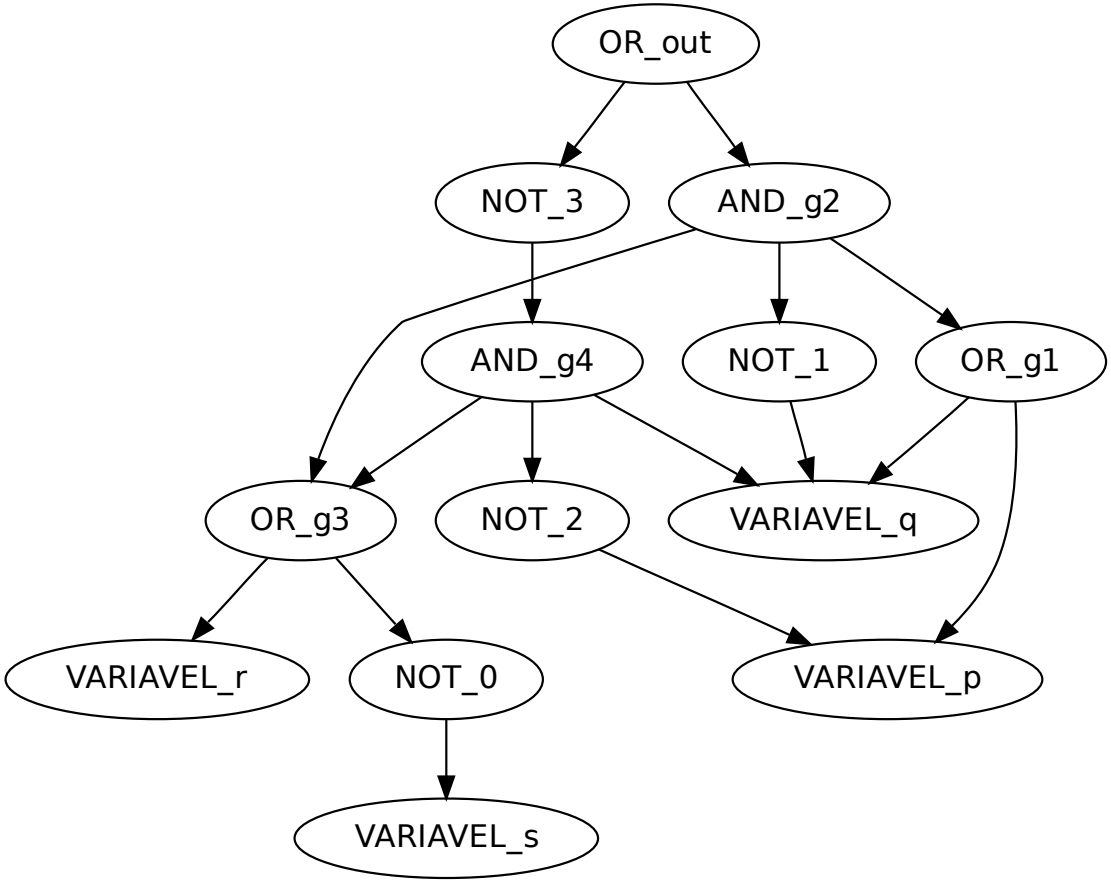


$$(((p \vee q) \wedge \neg r \wedge \neg q) \vee (\neg p \wedge (r \vee \neg s) \wedge q))$$

ISCAS

- DAG
- Nós Folha:
  - *Verdadeiro*
  - *Falso*
  - X
  - $\neg X$
- Nós internos:
  - $\wedge$ (AND)
  - $\vee$ (OR)
  - $\neg$ (NOT)
  - $\oplus$ (XOR)
  - NOR, NAND ...

ISCAS - Exemplo



$$((p \vee q) \wedge (r \vee \neg s) \wedge \neg q) \vee \neg(\neg p \wedge (r \vee \neg s) \wedge q)$$

SATMATE/NFLSAT

- Atua em fórmula NNF

$$\left[\left[\begin{array}{c} p \vee q \\ \neg r \\ \neg q \end{array}\right] \vee \left[\begin{array}{c} \neg p \\ r \vee \neg s \\ q \end{array}\right]\right]$$

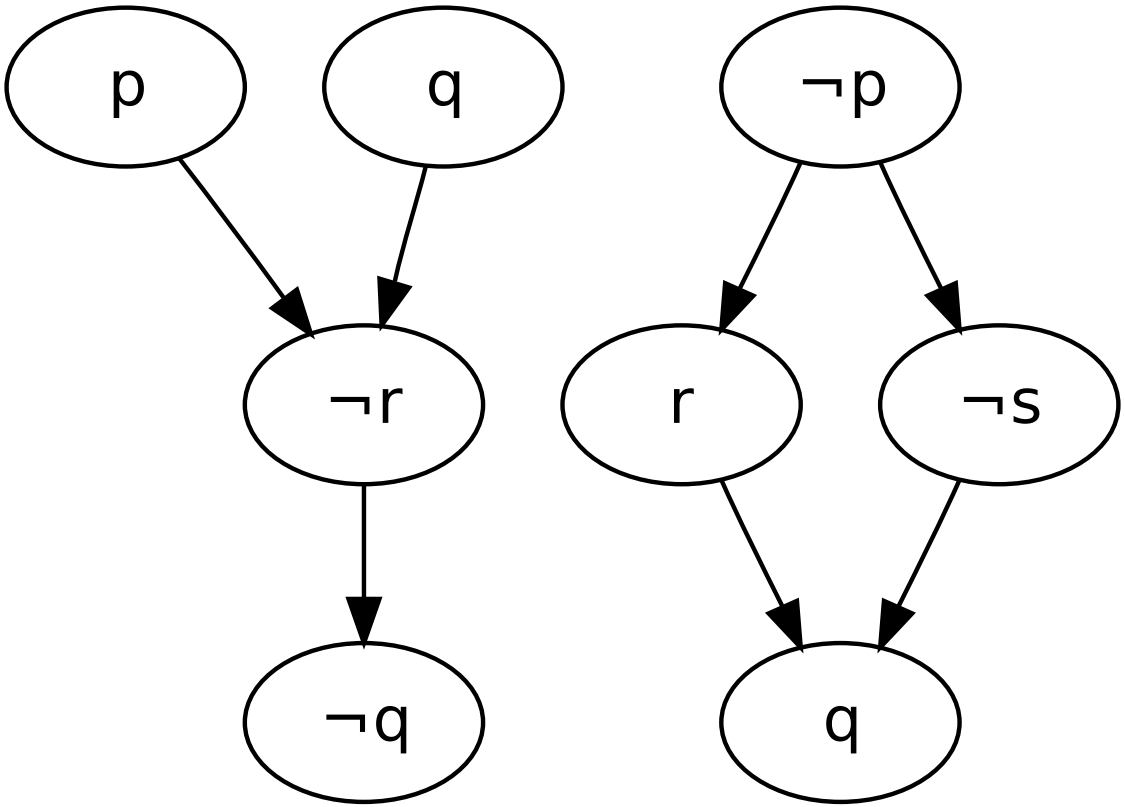
Figura: *vhpform* para a fórmula $(((p \vee q) \wedge \neg r \wedge \neg q) \vee (\neg p \wedge (r \vee \neg s) \wedge q))$

Caminho Vertical : $\{\langle p, \neg r, \neg q\rangle, \langle q, \neg r, \neg q\rangle, \langle \neg p, r, q\rangle, \langle \neg p, \neg s, q\rangle\}$;

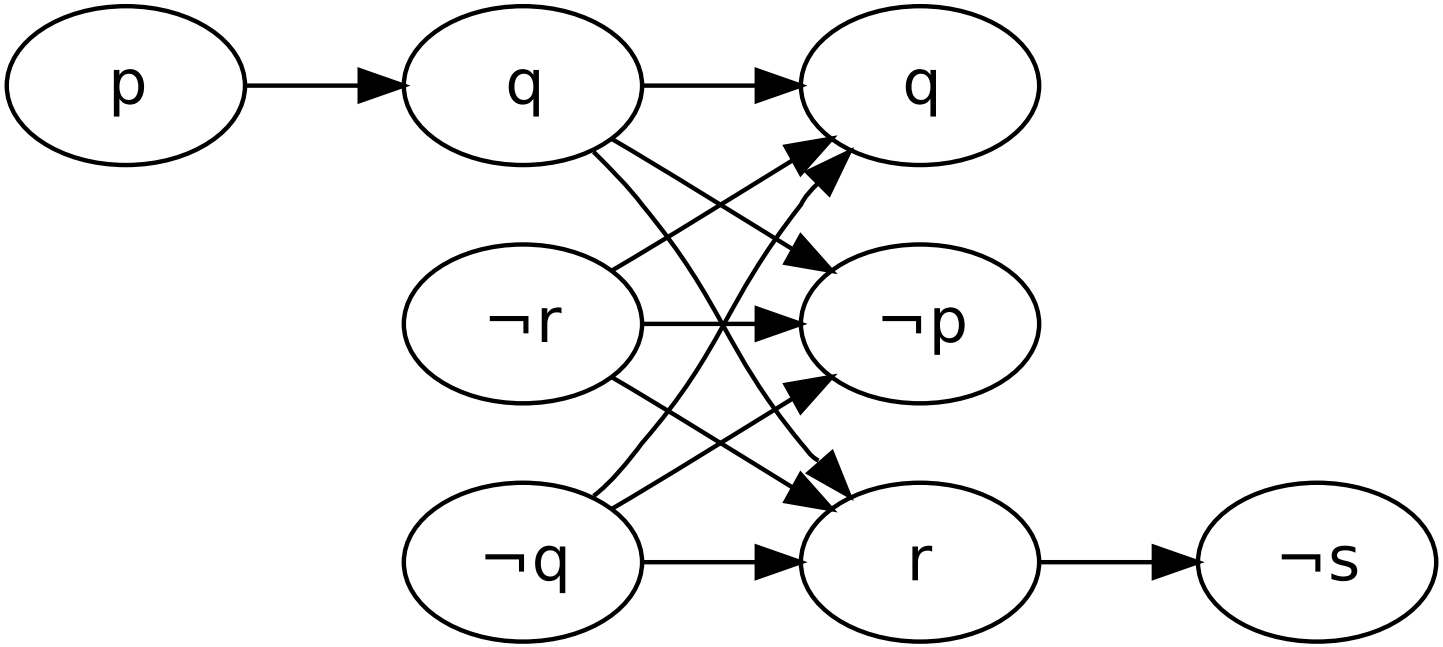
Caminho Horizontal :

$\{\langle p, q, \neg p\rangle, \langle p, q, r, \neg s\rangle, \langle p, q, q\rangle, \langle \neg r, \neg p\rangle, \langle \neg r, r, \neg s\rangle, \langle \neg r, q\rangle,$
$\langle \neg q, \neg p\rangle, \langle \neg q, r, \neg s\rangle, \langle \neg q, q\rangle\}$

SATMATE/NFLSAT - Grafo Vertical



SATMATE/NFLSAT - Grafo Horizontal



NOCLAUSE

- Atua diretamente no ISCAS
- Considera qualquer nó como váriável

1. Se uma conjunção ($\wedge$) se torna Verdadeiro (V), propaga Verdadeiro para todos os seus filhos;
2. Se um filho se torna Falso ($\neg$), propaga Falso (F) para a conjunção ($\wedge$) pai;
3. Se todos os filhos se tornam Verdadeiros (V), propaga Verdadeiro (V) para a conjunção ($\wedge$) pai;
4. Se uma conjunção ($\wedge$) é Falsa (F) e pelo menos um filho é Verdadeiro (V) propaga Falso (F) para um filho sem valoração.

NOCLAUSE

- Atua diretamente no ISCAS
- Considera qualquer nó como váriável

1. Se uma conjunção ($\wedge$) se torna Verdadeiro (V), propaga Verdadeiro para todos os seus filhos;
2. Se um filho se torna Falso ($\neg$), propaga Falso (F) para a conjunção ($\wedge$) pai;
3. Se todos os filhos se tornam Verdadeiros (V), propaga Verdadeiro (V) para a conjunção ($\wedge$) pai;
4. Se uma conjunção ($\wedge$) é Falsa (F) e pelo menos um filho é Verdadeiro (V) propaga Falso (F) para um filho sem valoração.

LIAMFSAT

- Resolvedor Não-Clausal
- Atua diretamente no ISCAS, assim como o NOCLAUSE
- Propagação ocorre EXCLUSIVAMENTE de baixo para cima
- Retrocesso não-cronológico
- Fórmula Nivela os operadores

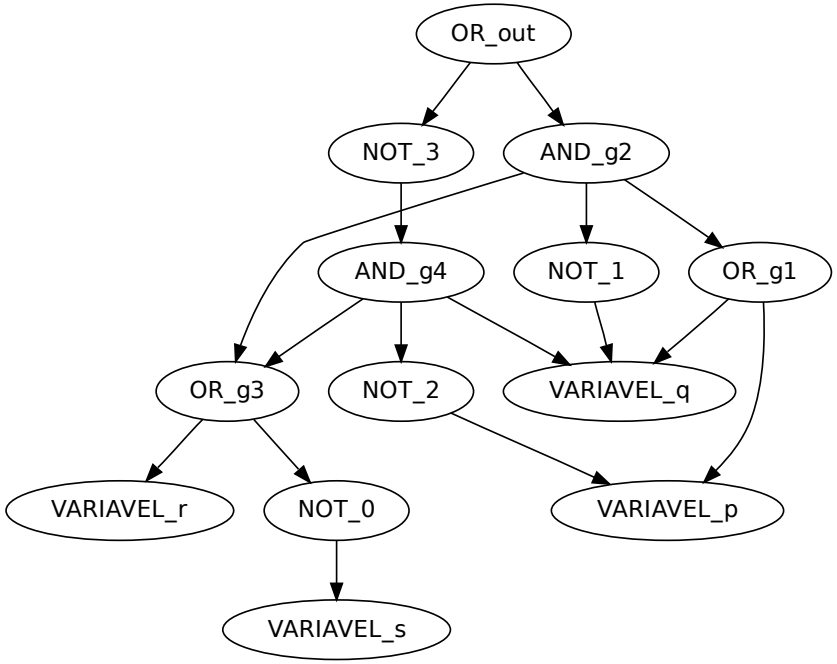
$$Nivel(G) = max_{i \in filho(G)}(Nivel(i)) + 1$$

LIAMFSAT

- Resolvedor Não-Clausal
- Atua diretamente no ISCAS, assim como o NOCLAUSE
- Propagação ocorre EXCLUSIVAMENTE de baixo para cima
- Retrocesso não-cronológico
- Fórmula Nivela os operadores

$$Nivel(G) = max_{i \in filho(G)}(Nivel(i)) + 1$$

LIAMFSAT



- se todos os filhos de um operador $\wedge$ são marcados como verdadeiro, o nó então se torna verdadeiro e envia um sinal de verdadeiro para os seus pais;
- se qualquer filho de um operador $\wedge$ for falso, o nó se torna falso e é enviado falso para todos os seus pais;

LIAMFSAT

($\wedge$, verdadeiro)

Nível de decisão do último filho valorado;

($\wedge$, falso)

Nível de decisão do filho falso de menor nível de decisão;

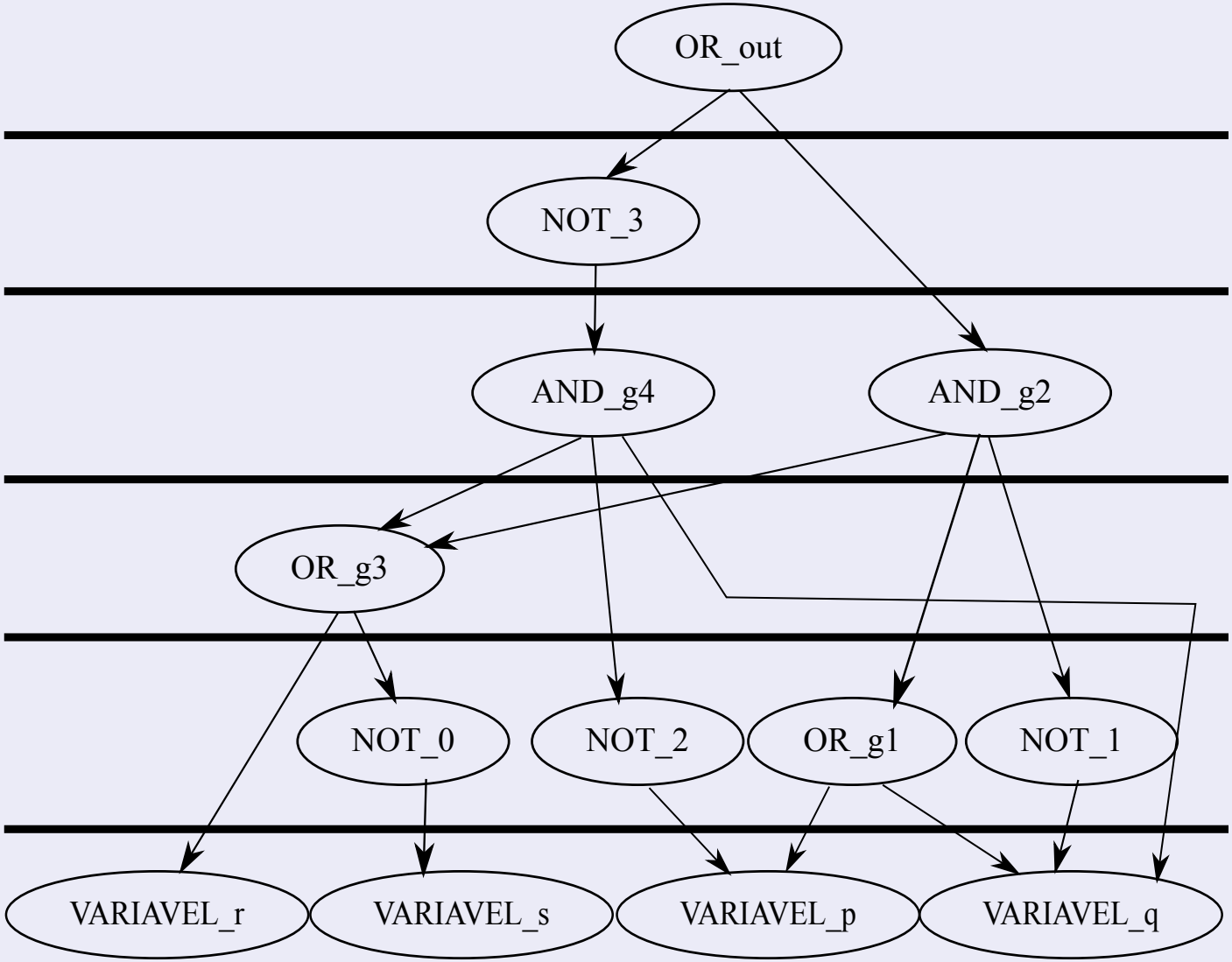
($\vee$, verdadeiro)

Nível de decisão do filho verdadeiro de menor nível de decisão;

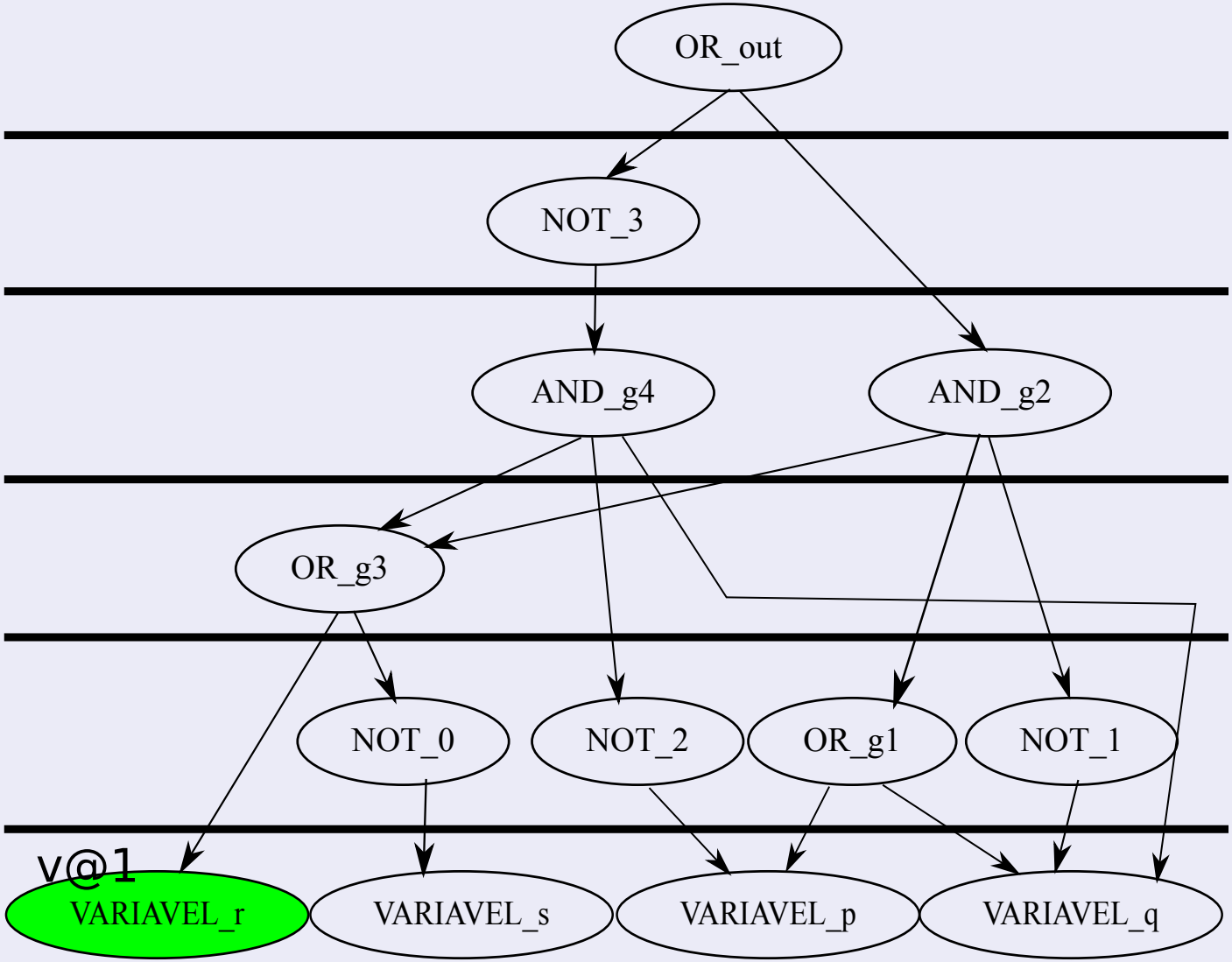
($\vee$, falso)

Nível de decisão do último filho valorado.

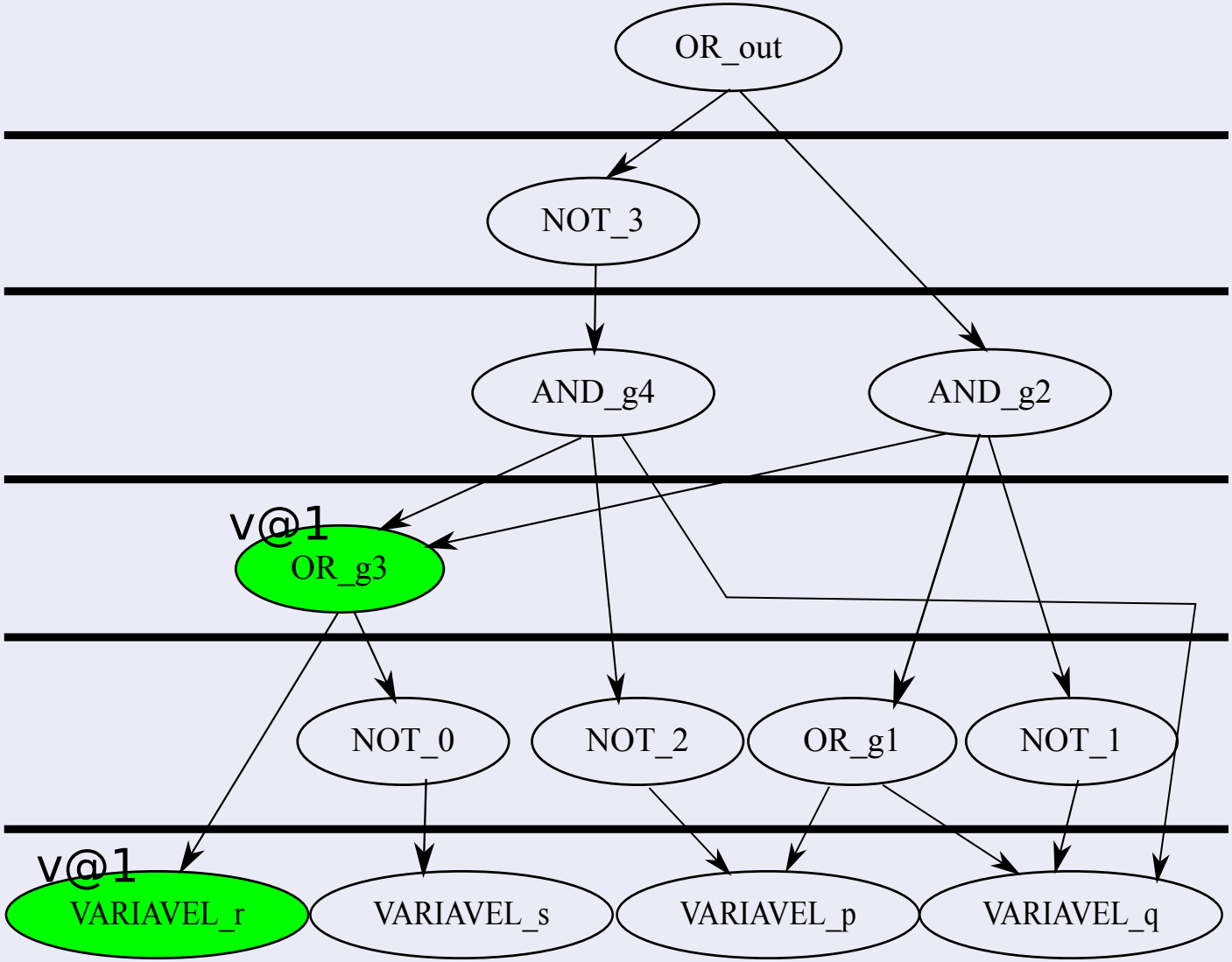
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q

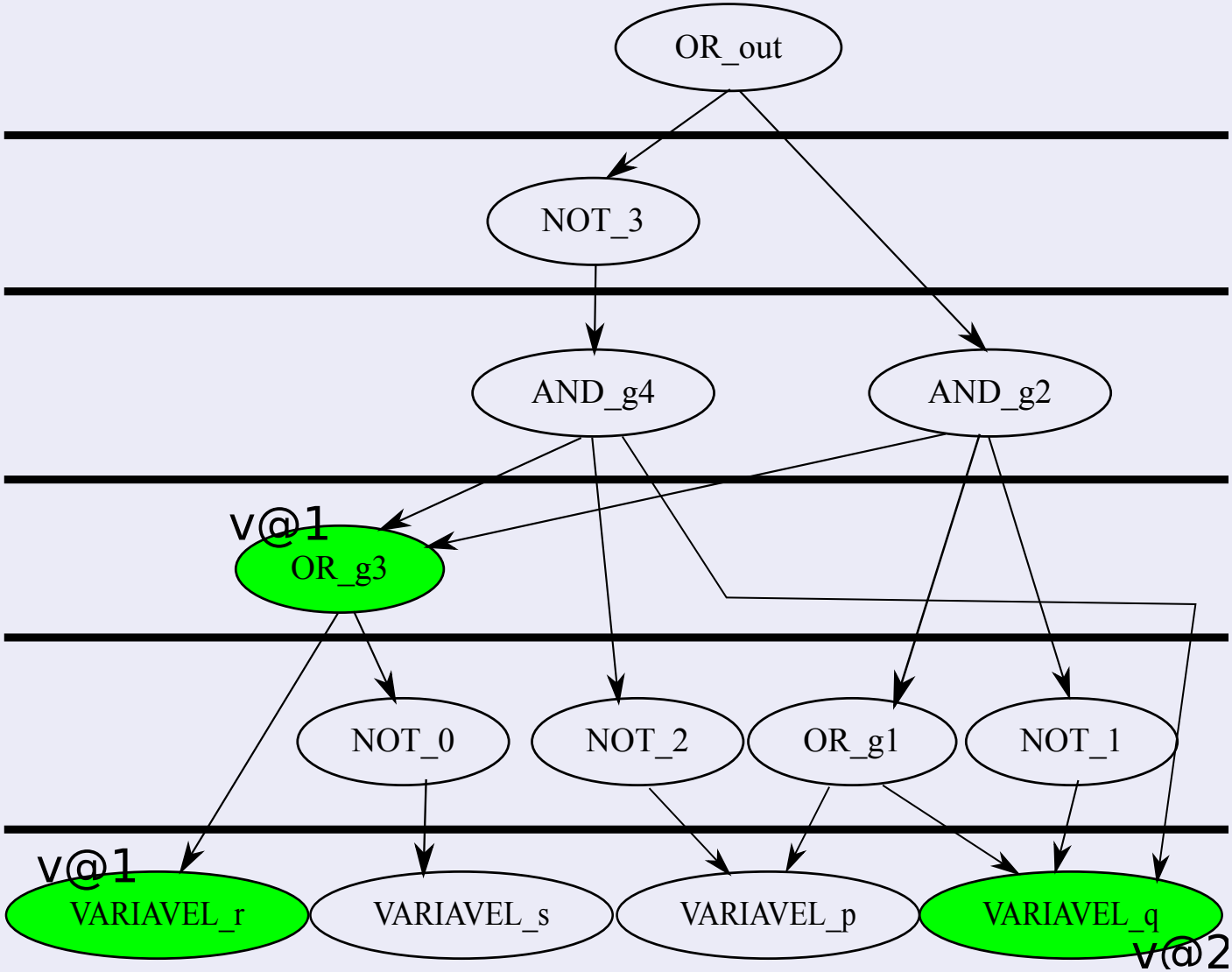
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q

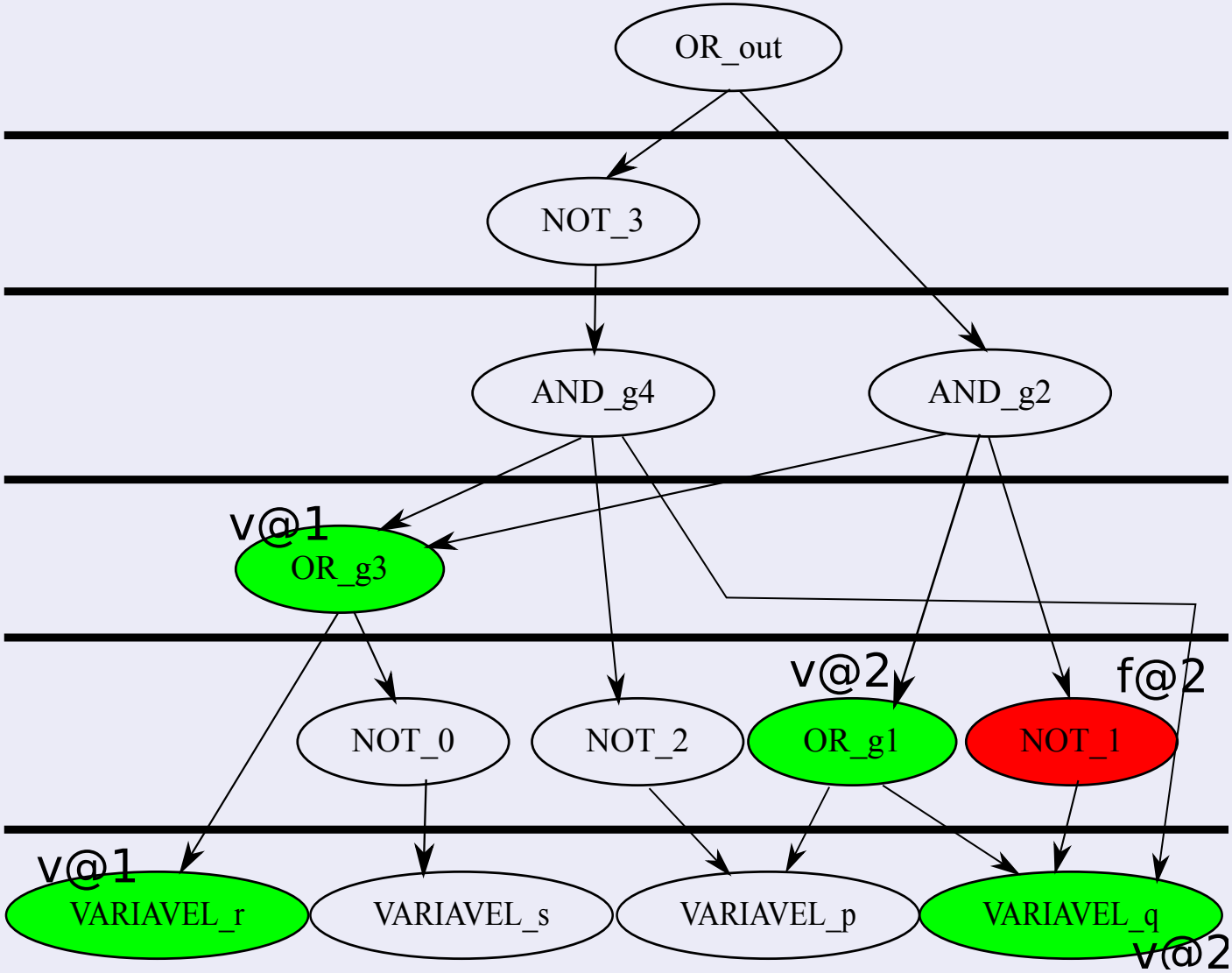
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q

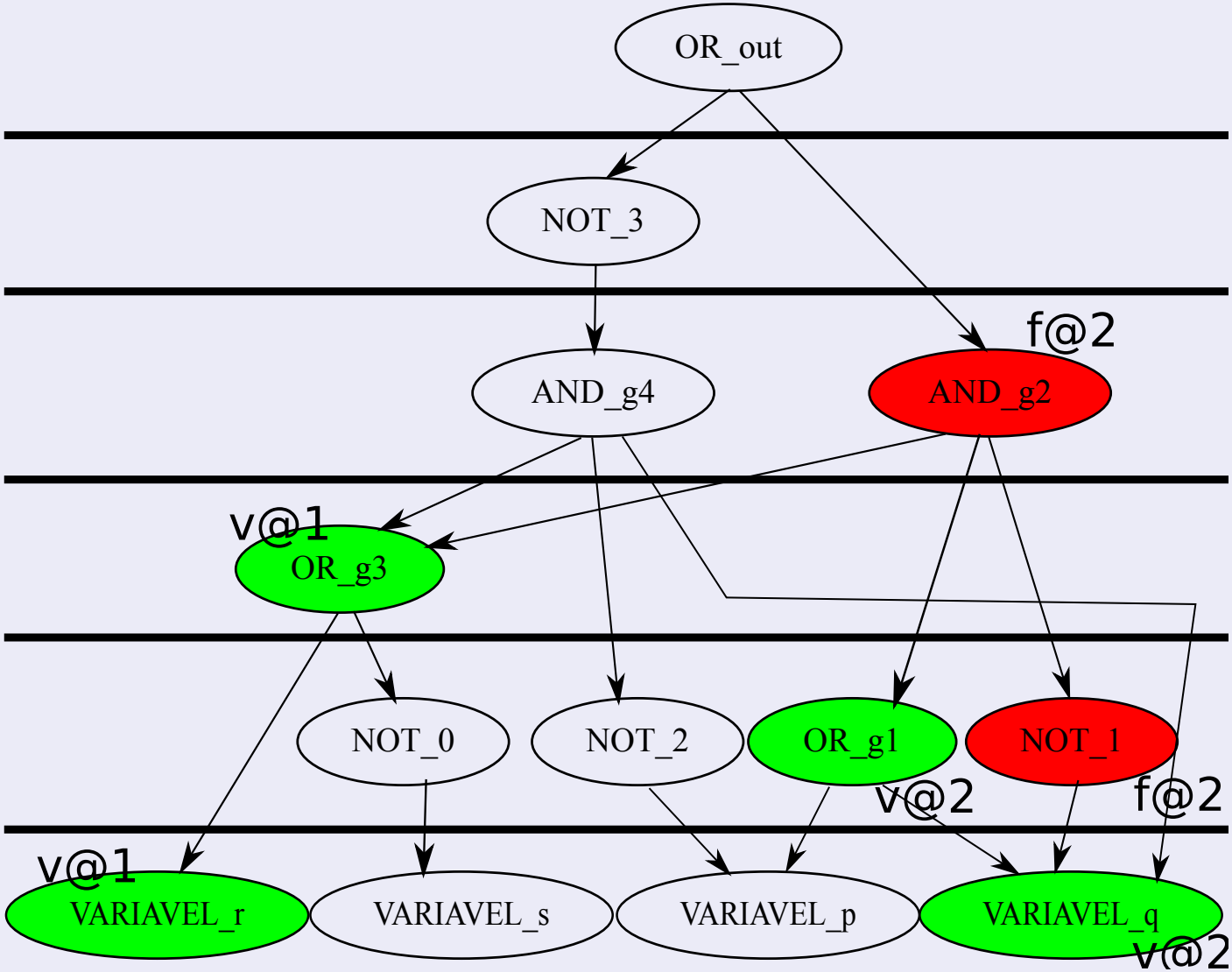
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
v@2

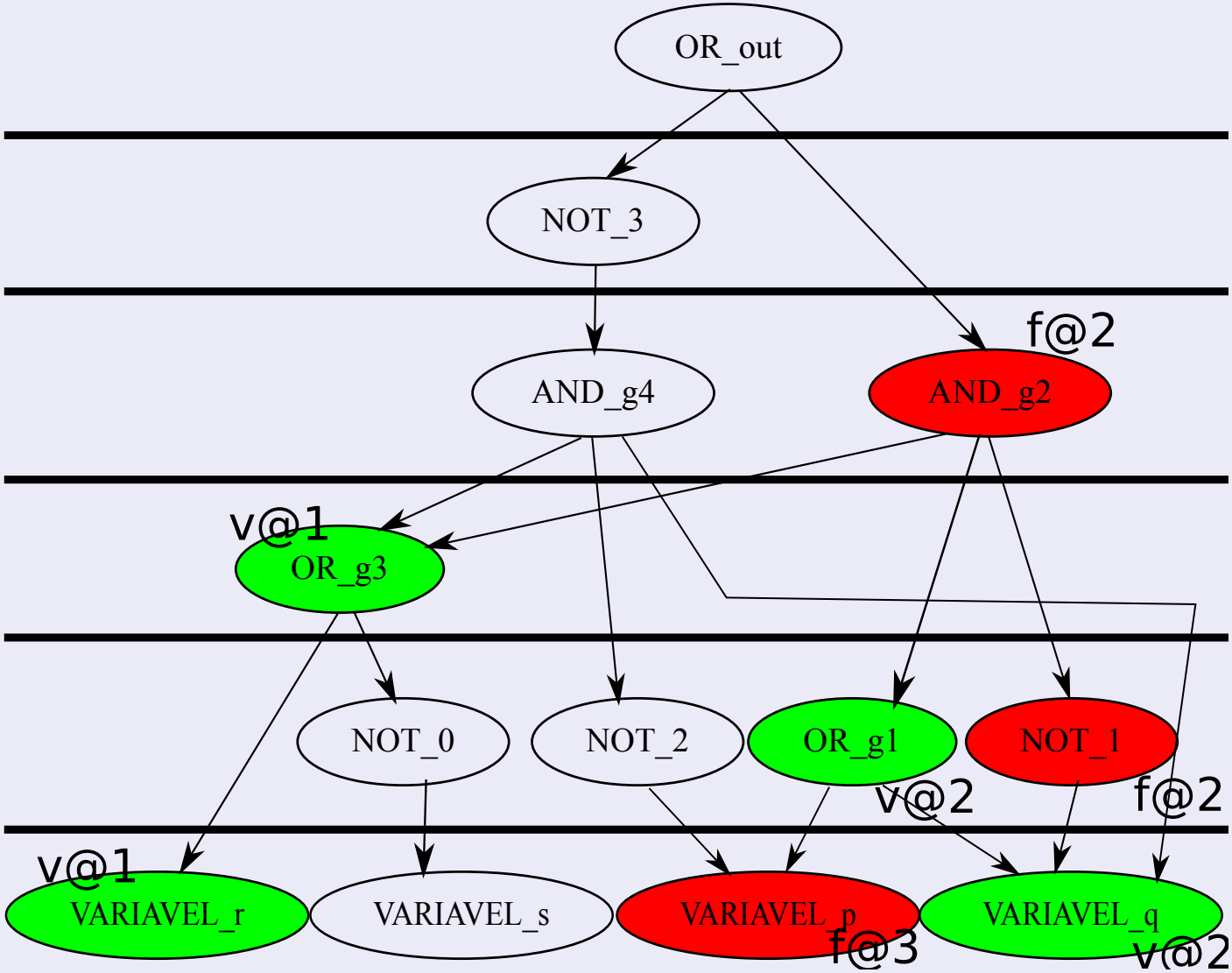
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
v@1
OR_g3
v@2
f@2
NOT_0
NOT_2
OR_g1
NOT_1
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
v@2

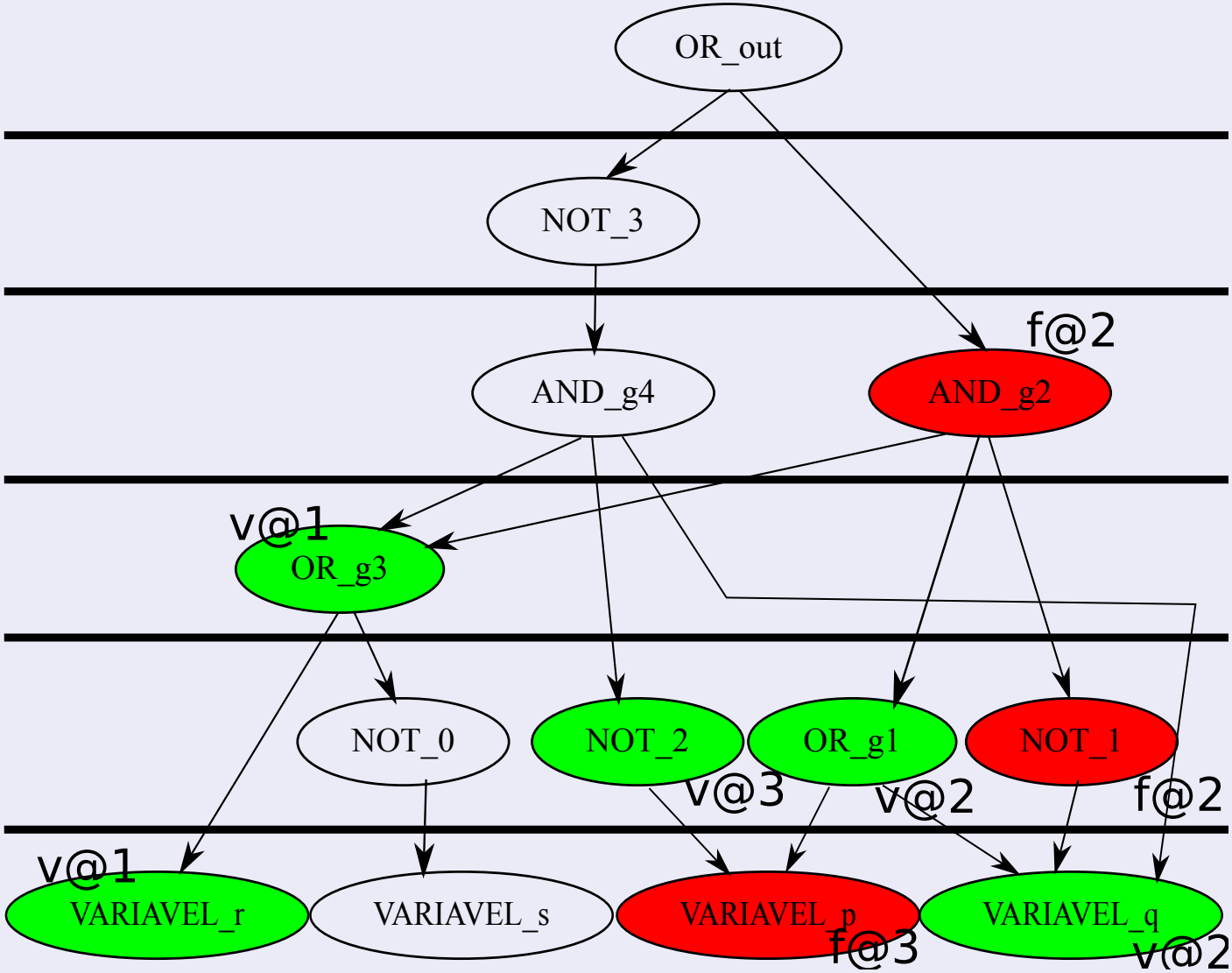
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
f@2
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@2
f@2
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
v@2

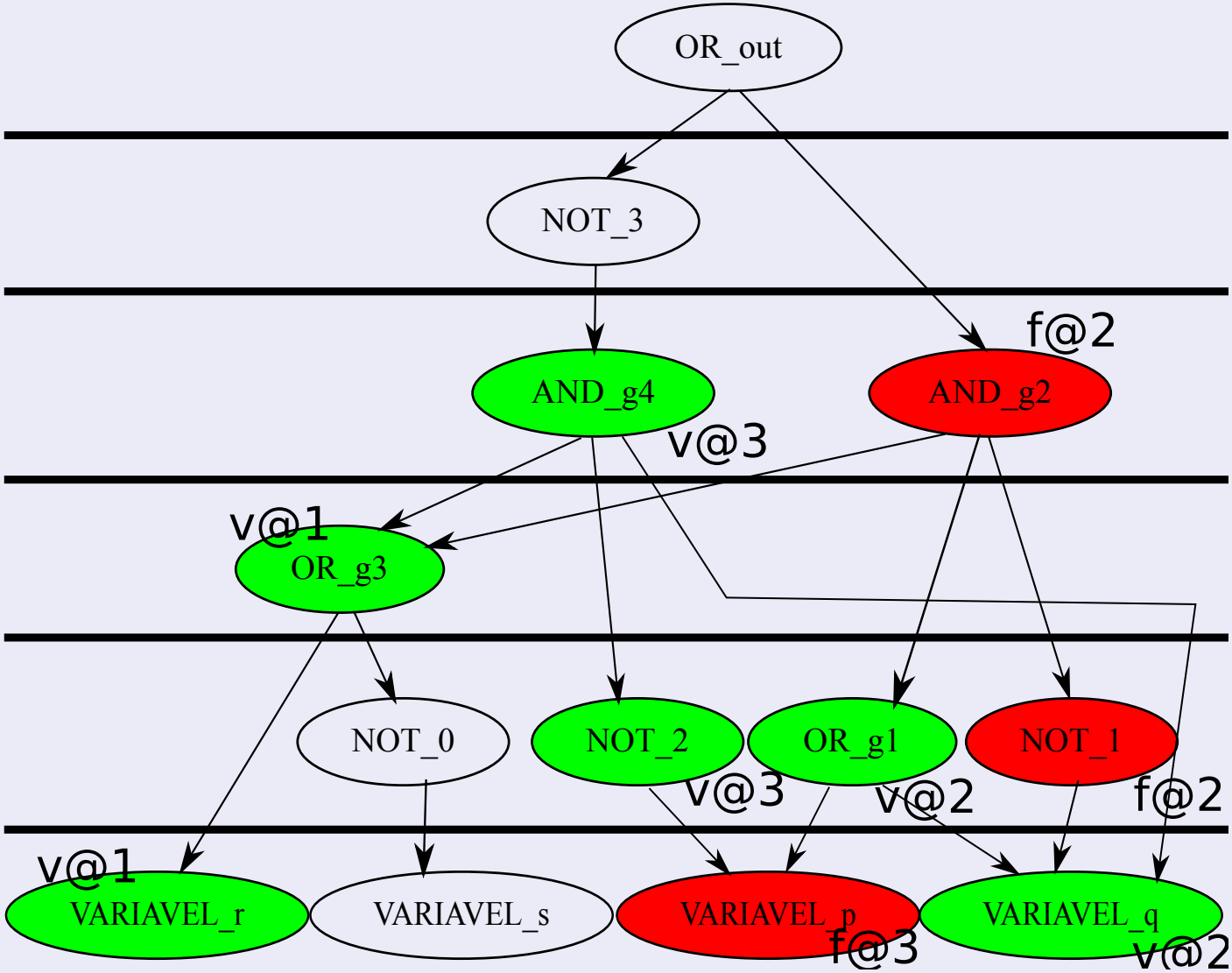
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
f@2
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@2
f@2
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
f@3
v@2

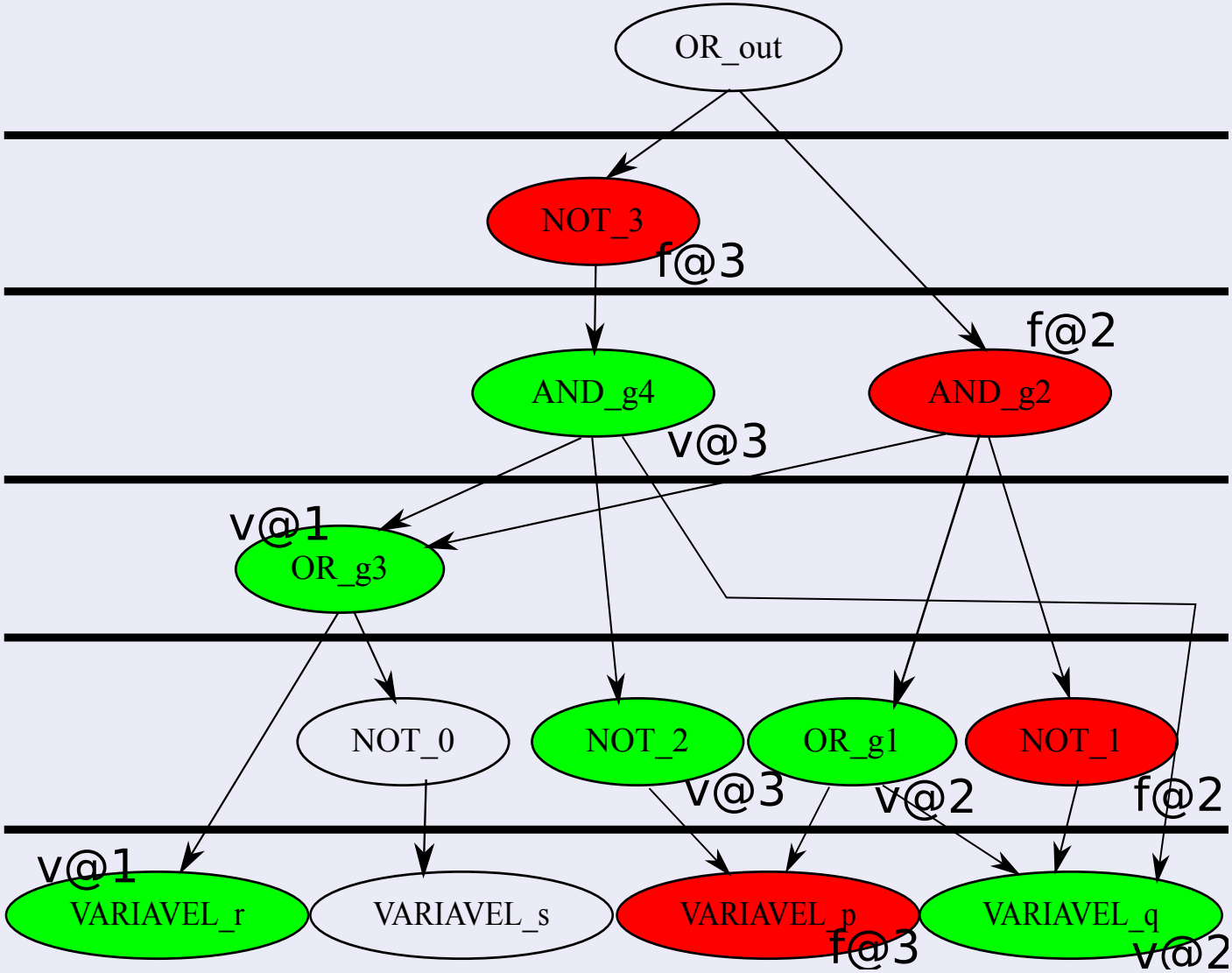
LIAMFSAT


OR_out
NOT_3
AND_g4
AND_g2
f@2
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@3
v@2
f@2
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
f@3
v@2

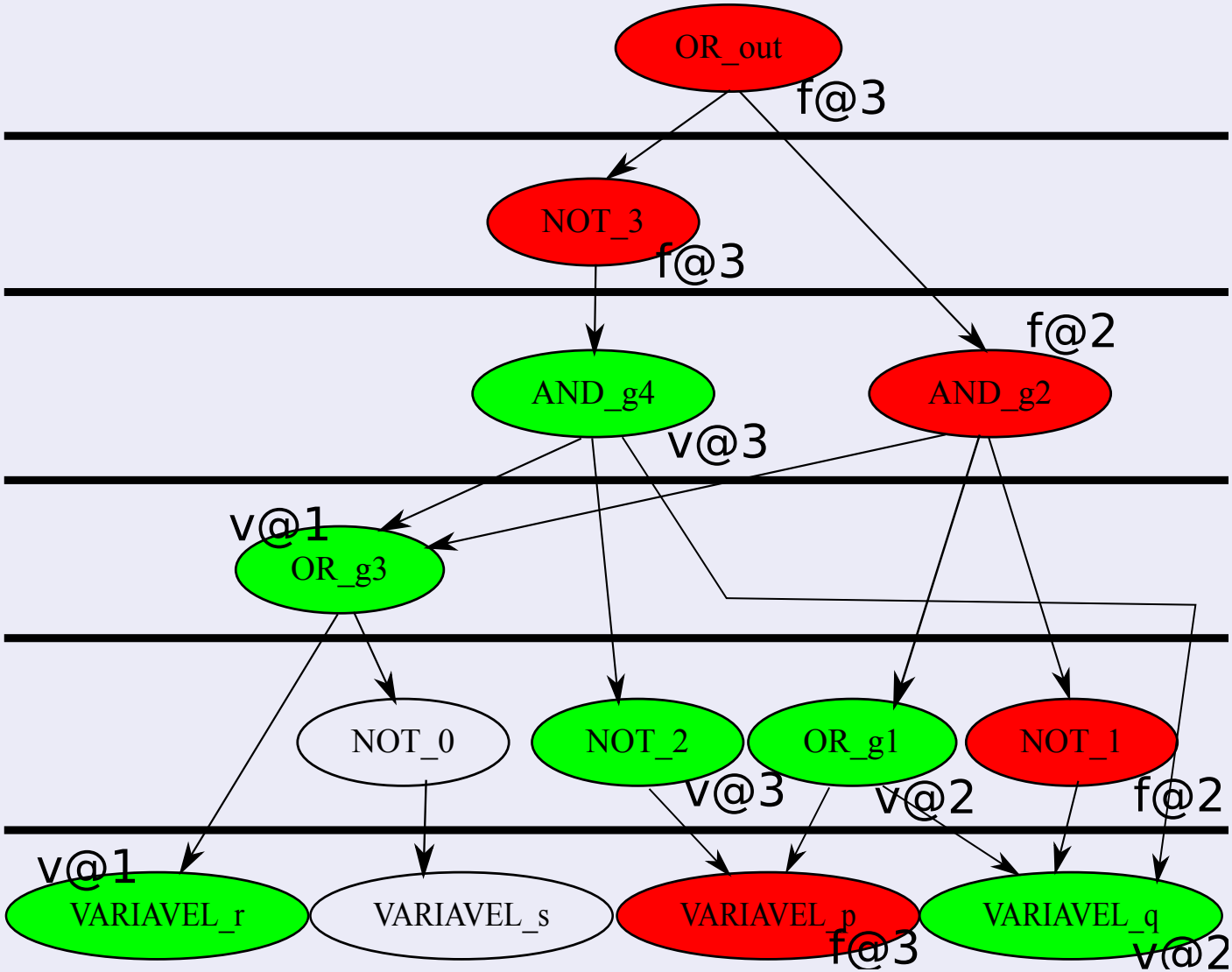
LIAMFSAT


OR_out
NOT_3
f@2
AND_g4
AND_g2
v@3
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@3
v@2
f@2
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
f@3
v@2

LIAMFSAT

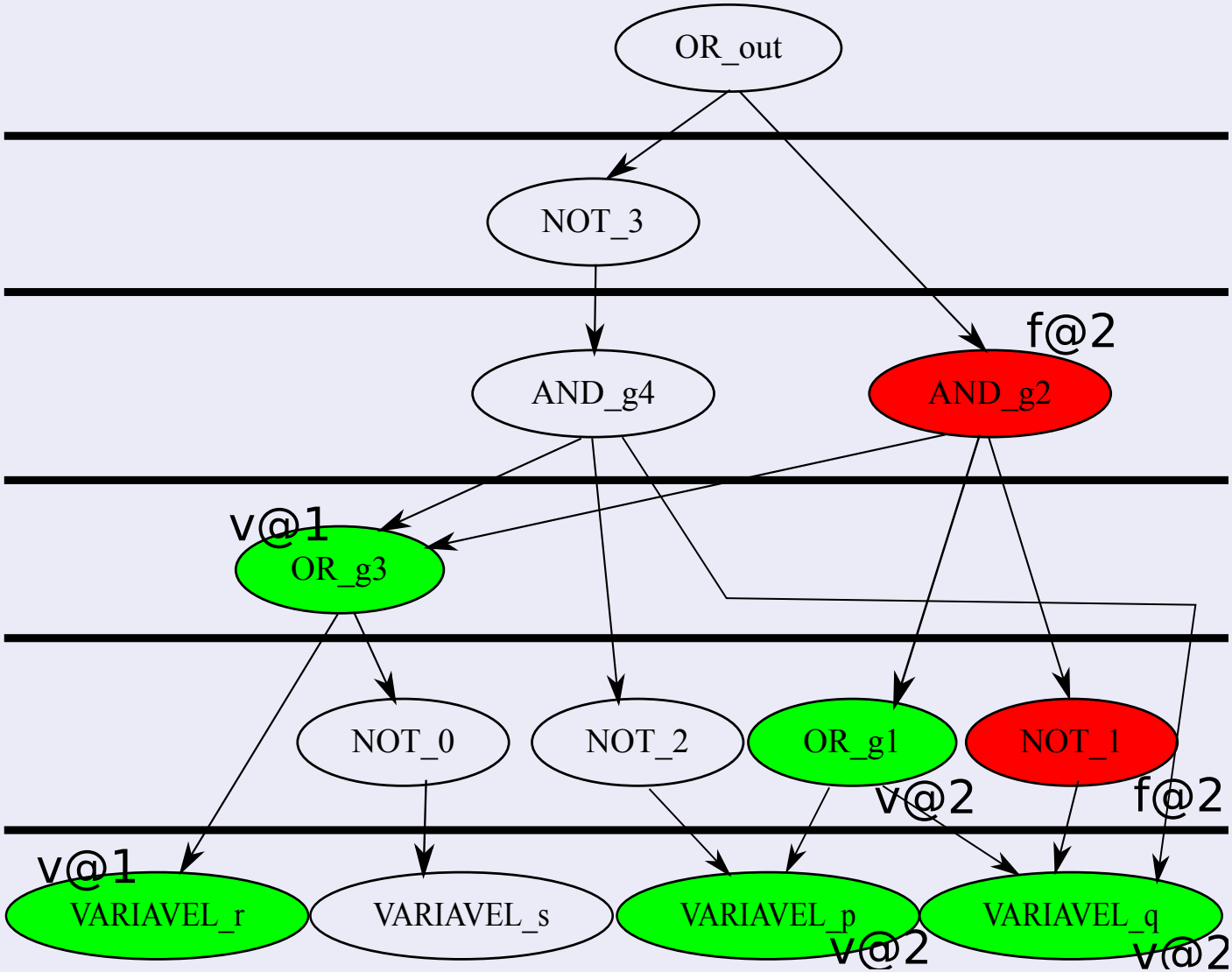

OR_out
NOT_3
f@3
f@2
AND_g4
AND_g2
v@3
v@1
OR_g3
NOT_0
NOT_2
OR_g1
NOT_1
v@3
v@2
f@2
v@1
VARIAVEL_r
VARIAVEL_s
VARIAVEL_p
VARIAVEL_q
f@3
v@2

LIAMFSAT



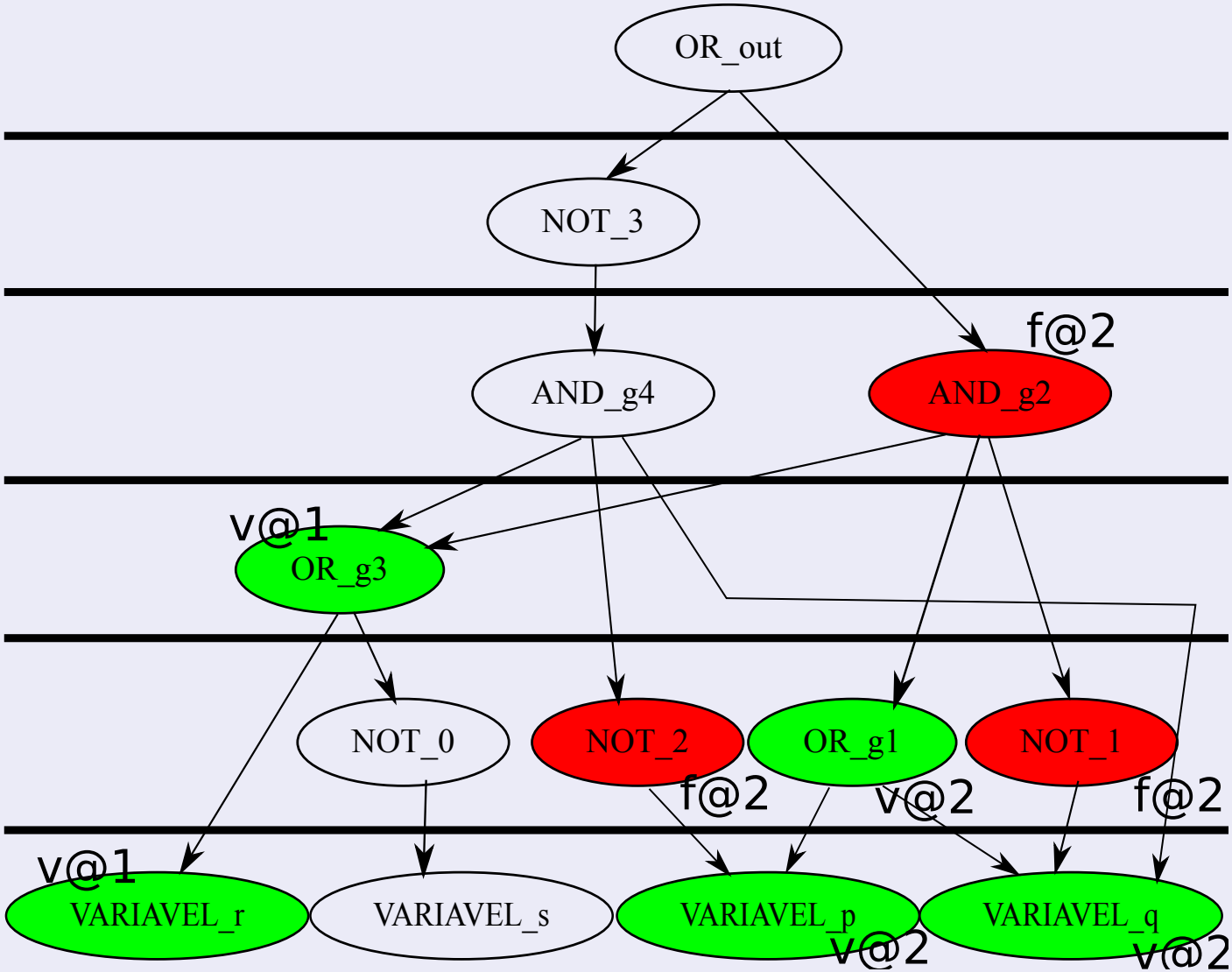
$$(\neg r \vee \neg q \vee p)$$

LIAMFSAT



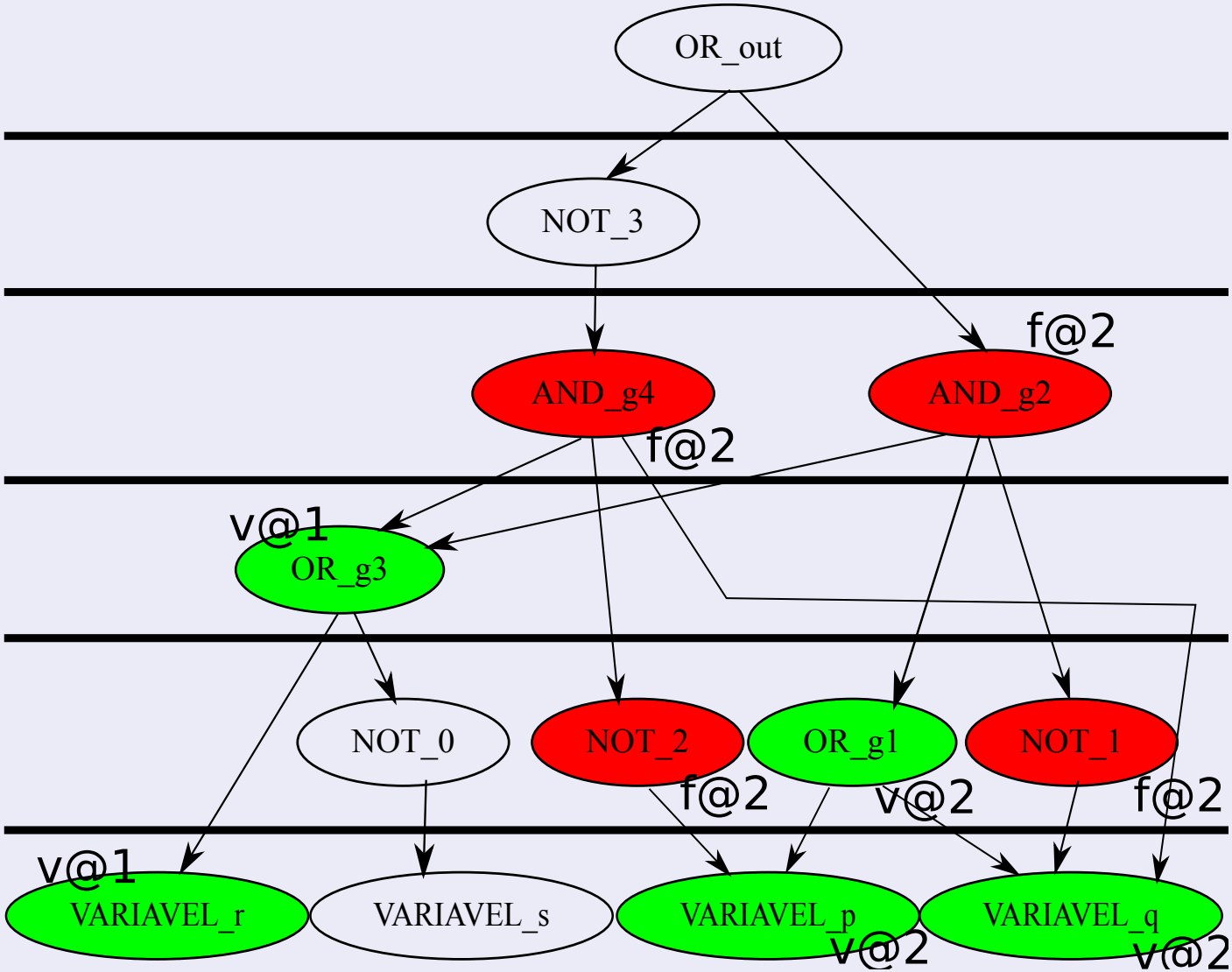
$$(\neg r \vee \neg q \vee p)$$

LIAMFSAT



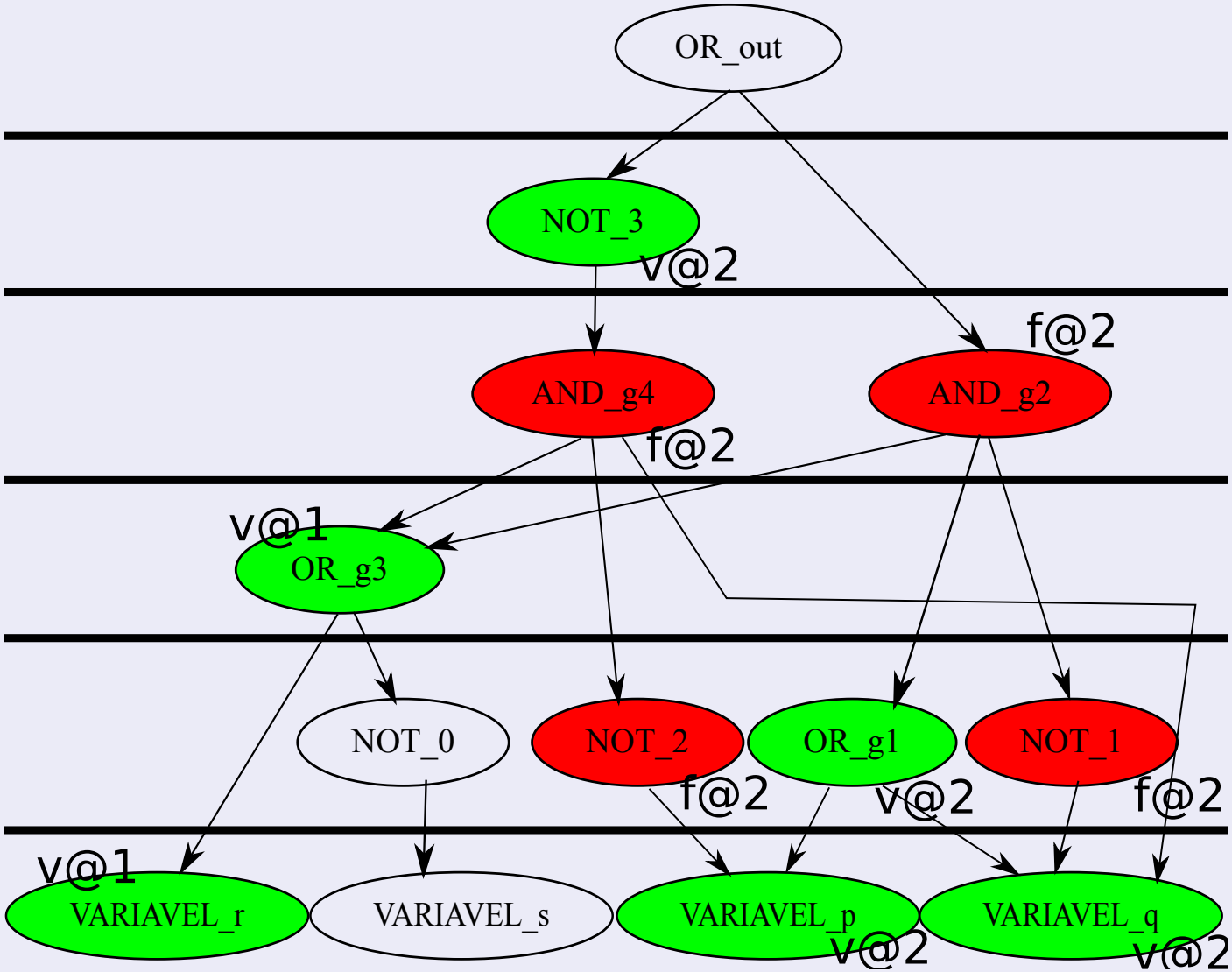
$$(\neg r \vee \neg q \vee p)$$

LIAMFSAT



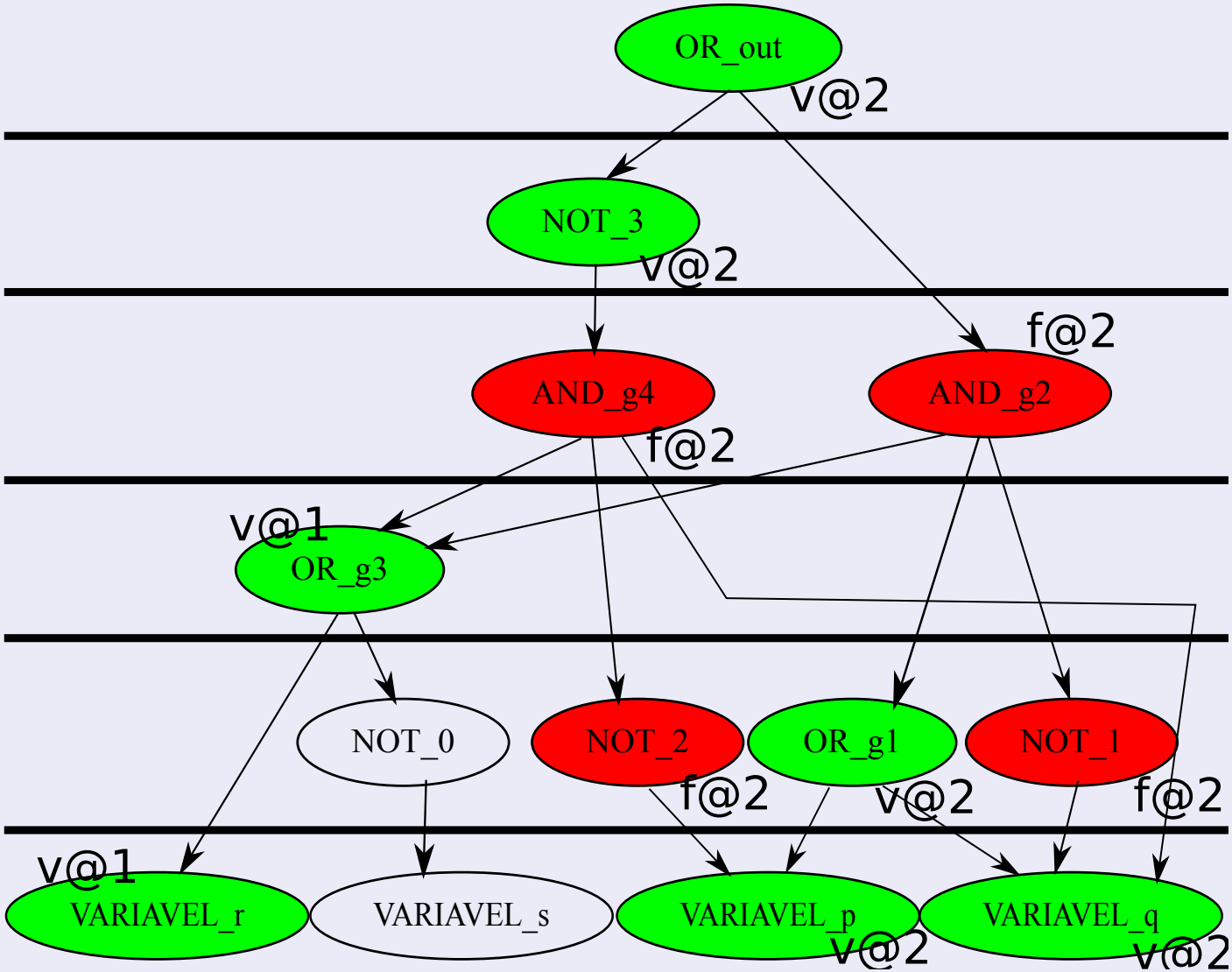
$$(\neg r \vee \neg q \vee p)$$

LIAMFSAT



$$(\neg r \vee \neg q \vee p)$$

LIAMFSAT



$$(\neg r \vee \neg q \vee p)$$

Avaliação Experimental

- Dividida em duas partes:
  1. Fórmulas mais fáceis retiradas de problemas de planejamento de jogos;
     - ★ Dificuldade em encontrar um conjunto de problemas
     - ★ Objetivo de testar retrocesso não-cronológico e aprendizado
  2. Fórmulas difíceis utilizadas em competições de SAT.
     - ★ Objetivo de comparar o LIAMFSAT com outros resolvedores

Avaliação Experimental

LIAMFSAT

possui aprendizado e retrocesso não-cronológico;

LIAMFSATr

possui apenas retrocesso não-cronológico;

LIAMFSAT0

não possui aprendizado nem retrocesso não cronológico;

Avaliação Experimental

	LIAMFSAT	LIAMFSATr	LIAMFSAT0
Tempo gasto	92,28s	100,60s	0,58s
Problemas Resolvidos	49	49	48

Avaliação Experimental

buttons.iscas_6_UNSAT

- 96 variáveis;
- 398 operadores;
- 419 literais;

Variação	Tempo	Aprendidas
LIAMFSAT	0,004s	17
LIAMFSATr	0,000s	0
LIAMFSAT0	0,35s	0

Avaliação Experimental

2pipe

- 96 variáveis;
- 812 operadores;
- 1326 literais;

Variação	Tempo	Aprendidas
LIAMFSAT	0,04s	223
LIAMFSATr	25,72s	0
LIAMFSAT0	TLE	0

Avaliação Experimental

SATMATE

resolvedor não-clausal baseado em NNF

NFLSAT

resolvedor não-clausal baseado em NNF, evolução do SATMATE

BCMINISAT

resolvedor MINISAT baseado em fórmulas em CNF

Avaliação Experimental

Problema	SATMATE	NFLSAT	LIAMFSAT	BCMINISAT
2pipe.iscas	29,47s	0,21s	0,02s	47,57s
2pipe_1_ooo.iscas	21,60s	0,17s	0,06s	4,46s
2pipe_2_ooo.iscas	311,42s	0,19s	0,07s	111,49s
3pipe.iscas	TLE	3,21s	0,30s	TLE
3pipe_1_ooo.iscas	TLE	2,79s	312,95s	TLE
3pipe_2_ooo.iscas	TLE	3,66s	114,17s	TLE
3pipe_3_ooo.iscas	TLE	3,56s	10,91s	TLE
4pipe.iscas	TLE	56,82s	10,59s	TLE
4pipe_3_ooo.iscas	TLE	13,49s	397,73s	TLE
4pipe_1_ooo.iscas	TLE	10,19s	1310,39s	TLE
5pipe.iscas	TLE	15,42s	131,26s	TLE
6pipe.iscas	TLE	TLE	1265,19s	TLE
7pipe.iscas	TLE	TLE	1260,44s	TLE
7pipe_bug.iscas	TLE	70,04s	TLE	1123,02s

Conclusão

- Resolvedor não-clausal que evita conversão para CNF
- Técnicas CNF adaptadas
  - BCP rápido
  - Retrocesso não-cronológico
  - Aprendizado de conflitos
- Ramificação apenas em variáveis (Järvisalo e Junttila 2007)
- Desempenho compatível com resolvedores atuais (NFLSAT, NOCLAUSE)
- Artigo submetido no ENIA 2011

Trabalhos futuros

- Implementação do VSIDS
- Melhorar técnicas de aprendizado
- Processamento Paralelo