



UNIVERSIDADE FEDERAL DO CEARÁ

Lógica para Computação
Conjunto Dominante de tamanho 3

Relatório final do Trabalho em Equipe apresentado à disciplina Lógica para Computação do curso de Engenharia de Software da Universidade Federal do Ceará.

Professor(a): Robertty Costa

Alunos:

Ruan David da Silva Barros, 553093

Lucas de Souza Almeida, 569990

Russas-CE

2026

Índice

1. Introdução.....	3
1.1. Problema Escolhido.....	3
2. Modelagem do Problema.....	3
2.1. Predicados em Lógica de Primeira Ordem(LPO).....	3
2.2. Representação na Sintaxe TPTP (Vampire).....	4
3. Testes em Grafos Pequenos.....	3
I. Grafo M(Caso Satisfeito).....	3
II. Grafo Isolado:(Caso Insatisfeito).....	5
III. Grafo Triângulo(Caso Satisfeito).....	6
IV. Grafo Disperso(Caso Satisfeito).....	8
4. Teste Final em Grafo Grande.....	10
4.1. Grafo Escolhido	12
4.2. Sintaxe TPTP.....	13
4.3. Resultados do Teste.....	13
5. Conclusão.....	14

1. Introdução

O presente relatório detalha o desenvolvimento do trabalho final da disciplina de Lógica para Computação, ministrada pelo Professor Robertty Costa no Semestre 2025.2, na Universidade Federal do Ceará – Campus Russas. O objetivo central desta atividade consiste na modelagem e resolução de problemas clássicos da teoria dos grafos utilizando Lógica de Primeira Ordem (LPO).

1.1. Problema Escolhido

O problema abordado neste trabalho é o do **Conjunto Dominante**. Um conjunto dominante em um grafo é um subconjunto de vértices tal que, cada vértice do grafo ou pertence a este conjunto, ou é vizinho de pelo menos um vértice dele. O objetivo específico deste trabalho é verificar a existência de conjuntos dominantes de tamanhos específicos ($k=1$ e $k=3$) usando provas automáticas.

2. Modelagem do Problema

2.1 Predicados em Lógica de Primeira Ordem(LPO)

Para modelar este problema, utilizamos o predicado **adj(x,y)** para representar a adjacência entre vértices. A propriedade de um **Conjunto Dominante de tamanho $k=3$** exige que existam três vértices (c_1, c_2, c_3) tal que, para **todo** vértice X do grafo, X pertença a esse conjunto ou seja vizinho de um de seus membros.

A fórmula lógica correspondente é:

$$\exists c1 \exists c2 \exists c3 \forall x((x = c1 \vee x = c2 \vee x = c3) \vee (\text{adj}(x, c1) \vee \text{adj}(x, c2) \vee \text{adj}(x, c3)))$$

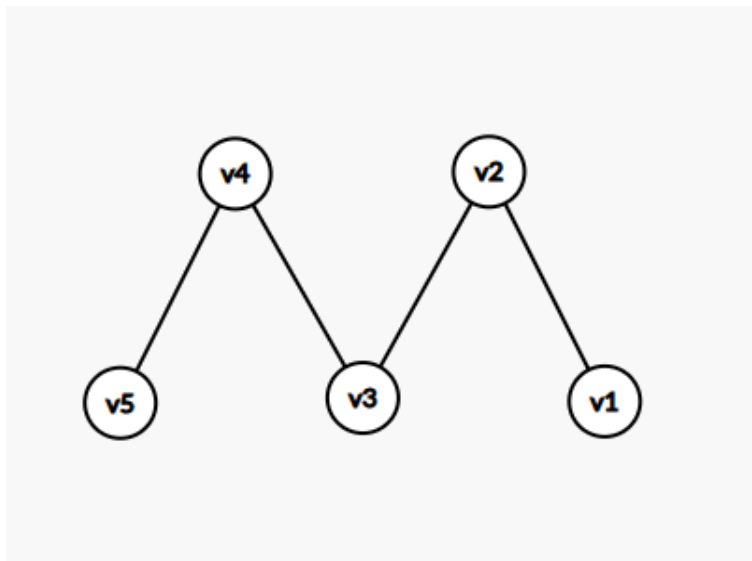
2.2 Representação na Sintaxe TPTP (Vampire)

Representação na Sintaxe TPTP (Vampire) A propriedade foi traduzida para a sintaxe do Vampire. Definimos o grafo através de axiomas de adjacência e um axioma de fechamento de domínio. A conjectura para tamanho 3, utilizada neste trabalho, é escrita como:

```
fof(check_k3, conjecture,
  ? [C1, C2, C3] : ! [X] : (
    X=C1 | X=C2 | X=C3 |
    adj(X,C1) | adj(X,C2) | adj(X,C3)
  )
).
```

3. Testes em Grafos Pequenos

I. Grafo M(Caso Satisfeito)



```
% 1. Simetria
fof(simetria, axiom,
  ! [X,Y] : (adj(X,Y) => adj(Y,X) )
).

% 2. O Grafo (Desenhando o M: 5-4-3-2-1)
fof(grafo_m, axiom,
  adj(v5, v4) &
  adj(v4, v3) &
  adj(v3, v2) &
  adj(v2, v1)
).

% 3. Fechamento (Universo de 5 vértices)
fof(fechamento, axiom,
  ! [X] : (X=v1 | X=v2 | X=v3 | X=v4 | X=v5)
).

% 4. Pergunta: Existem 3 câmeras (C1, C2, C3)?
fof(check_k3, conjecture,
  ? [C1, C2, C3] : ! [X] : (
    X = C1 | X = C2 | X = C3 |
    adj(X, C1) | adj(X, C2) | adj(X, C3)
  )
).
```

Código:

```
fof(simetria, axiom,
  ! [X,Y] : (adj(X,Y) => adj(Y,X) )
```

).

```
fof(grafo_m, axiom,  
    adj(v5, v4) &  
    adj(v4, v3) &  
    adj(v3, v2) &  
    adj(v2, v1)  
).
```

```
fof(fechamento, axiom,  
    ! [X] : (X=v1 | X=v2 | X=v3 | X=v4 | X=v5)  
).
```

```
fof(check_k3, conjecture,  
    ? [C1, C2, C3] : ! [X] : (  
        X = C1 | X = C2 | X = C3 |  
        adj(X, C1) | adj(X, C2) | adj(X, C3)  
    )  
).
```

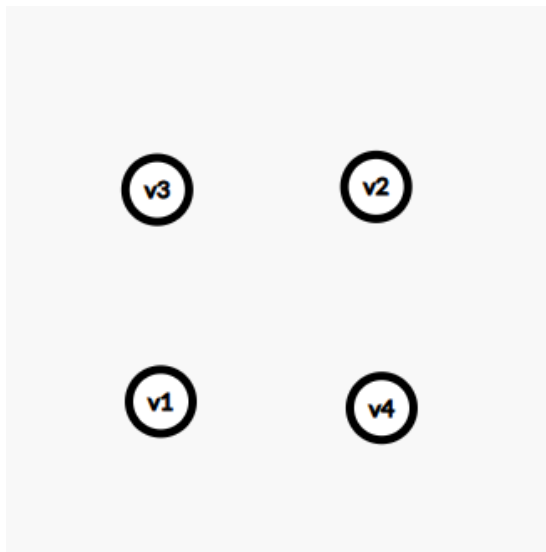
Resultado:

```
root@ether:/home/vampire/Trabalho_Final# nano grafom.p
root@ether:/home/vampire/Trabalho_Final# ./vampire grafom.p
% Running in auto input_syntax mode. Trying TPTP
% Refutation found. Thanks to Tanya!
% SZS status Theorem for grafom
% SZS output start Proof for grafom
1. ! [X0,X1] : (adj(X0,X1) => adj(X1,X0)) [input(axiom)]
2. adj(v5,v4) & adj(v4,v3) & adj(v3,v2) & adj(v2,v1) [input(axiom)]
3. ! [X0] : (X0 = v1 | X0 = v2 | X0 = v3 | X0 = v4 | X0 = v5) [input(axiom)]
4. ! [X0,X1,X2] : ! [X3] : (X3 = X0 | X3 = X1 | X3 = X2 | adj(X3,X0) | adj(X3,X1) | adj(X3,X2)) [input(conjecture)]
5. ~? [X0,X1,X2] : ! [X3] : (X3 = X0 | X3 = X1 | X3 = X2 | adj(X3,X0) | adj(X3,X1) | adj(X3,X2)) [negated conjecture 4]
6. ! [X0,X1] : (adj(X1,X0) | ~adj(X0,X1)) [emnf transformation 1]
7. ! [X0,X1,X2] : ? [X3] : (X0 = X3 & X1 = X3 & X2 = X3 & ~adj(X3,X0) & ~adj(X3,X1) & ~adj(X3,X2)) [emnf transformation 5]
8. ! [X0,X1,X2] : ~? [X3] : (X0 = X3 & X1 = X3 & X2 = X3 & ~adj(X3,X0) & ~adj(X3,X1) & ~adj(X3,X2)) => (sk0(X0,X1,X2) = X0 & sk0(X0,X1,X2) = X1 & sk0(X0,X1,X2) = X2 & ~adj(sk0(X0,X1,X2),X0) & ~adj(sk0(X0,X1,X2),X1) & ~adj(sk0(X0,X1,X2),X2))) [skolem symbol introduction]
9. ! [X0,X1,X2] : (sk0(X0,X1,X2) = X0 & sk0(X0,X1,X2) = X1 & sk0(X0,X1,X2) = X2 & ~adj(sk0(X0,X1,X2),X0) & ~adj(sk0(X0,X1,X2),X1) & ~adj(sk0(X0,X1,X2),X2))) [skolemisation 7,8]
10. ~adj(X0,X1) [sat(X0,X0) [cnf transformation 6]
11. adj(v2,v1) [cnf transformation 2]
12. adj(v5,v4) [cnf transformation 2]
13. v1 = X0 | v2 = X0 | v3 = X0 | v4 = X0 | v5 = X0 [cnf transformation 3]
14. ~adj(sk0(X0,X1,X2),X1) [cnf transformation 9]
15. ~adj(sk0(X0,X1,X2),X0) [cnf transformation 9]
16. sk0(X0,X1,X2) = X2 [cnf transformation 9]
17. sk0(X0,X1,X2) = X1 [cnf transformation 9]
18. sk0(X0,X1,X2) = X0 [cnf transformation 9]
19. adj(v4,v5) [resolution 10,14]
20. v1 = X0 | v2 = sk0(X0,X1,X2) | v3 = sk0(X0,X1,X2) | v4 = sk0(X0,X1,X2) | v5 = sk0(X0,X1,X2) [superposition 21,15]
21. v2 = sk0(v1,X0,X1) | v3 = sk0(v1,X0,X1) | v4 = sk0(v1,X0,X1) | v5 = sk0(v1,X0,X1) [equality resolution 20]
22. v5 = X0 | v3 = sk0(v1,X0,X1) | v4 = sk0(v1,X0,X1) | v2 = sk0(v1,X0,X1) [superposition 20,22]
23. v2 = sk0(v1,v5,X0) | v4 = sk0(v1,v5,X0) | v3 = sk0(v1,v5,X0) [equality resolution 68]
24. ~adj(v2,v5) | v4 = sk0(v1,v5,X0) | v3 = sk0(v1,v5,X0) [superposition 19,18]
25. !1 <=> ! [X0] : (v4 = sk0(v1,v5,X0) | v3 = sk0(v1,v5,X0)) [avatar definition]
26. v3 = sk0(v1,v5,X0) | v4 = sk0(v1,v5,X0) <- (11) [avatar component clause 122]
27. v4 = sk0(v1,v5,X0) | v3 = sk0(v1,v5,X0) [forward subsumption resolution 116,11]
28. !1 [avatar split clause 139,122]
29. v3 = X0 | v4 = sk0(v1,v5,X0) <- (11) [superposition 19,123]
30. v4 = sk0(v1,v5,X0) <- (11) [equality resolution 155]
31. ~adj(v4,v5) <- (11) [superposition 17,183]
32. $false <- (11) [forward subsumption resolution 194,22]
33. ~!1 [avatar contradiction clause 198]
34. !1 [sat_conversion 148]
35. ~!1 [sat_conversion 199]
36. # [sat 112,117]
280. $false [avatar sat refutation s18]
% SZS output end Proof for grafom
% *****
% Version: Vampire 5.8.3 (Release build, commit 1b1beaf on 2026-01-18 12:14:56 +0000)
% Linked with 23.14.0-9 3c47f09cf5645db42b2c819d9e9a04308aa721 NOTFOUND
% CoCoAL version: 2.1.3
% Termination reason: Refutation
% Time elapsed: 0.014 s
% Peak memory usage: 14 MB
% Instructions burned: 19 (million)
% *****
root@ether:/home/vampire/Trabalho_Final#
```

Análise do Resultado: Para este teste, utilizou-se um grafo linear de 5 vértices disposto em formato de "M" (v5-v4-v3-v2-v1). A conjectura verificou a existência de um Conjunto Dominante de tamanho k=3.

O provador Vampire retornou o status "**Theorem**" com uma prova por **Refutação** em apenas 0.014s. Este resultado é consistente com a teoria dos grafos: para um caminho de 5 vértices, um conjunto dominante mínimo teria tamanho 2 (por exemplo, vértices v2, v cobrem todo o grafo). Como a conjectura perguntava se era possível cobrir com 3 câmeras, a condição foi satisfeita com folga, resultando em uma prova de sucesso imediata.

II. Grafo Isolado:(Caso Insatisfeito)



```
% 1. Simetria
fof(simetria, axiom, ! [X,Y] : (adj(X,Y) => adj(Y,X)) ).

% 2. O Grafo (Vazio/Isolados)
% Como nao tem arestas, usamos $true para dizer "sem regras de conexao"
fof(grafo_isolados, axiom, $true).

% 3. Fechamento (Universo de 4 vértices)
fof(fechamento, axiom,
! [X] : (X=v1 | X=v2 | X=v3 | X=v4)
).

% 4. Pergunta: Existem 3 câmeras?
fof(check_k3, conjecture,
? [C1, C2, C3] : ! [X] : (
X = C1 | X = C2 | X = C3 |
adj(X, C1) | adj(X, C2) | adj(X, C3)
)
).
```

Codigo:

```
fof(simetria, axiom, ! [X,Y] : (adj(X,Y) => adj(Y,X)) ).
```

```
fof(fechamento, axiom,
! [X] : (X=v1 | X=v2 | X=v3 | X=v4)
).
```

```
fof(check_k3, conjecture,
? [C1, C2, C3] : ! [X] : (
X = C1 | X = C2 | X = C3 |
adj(X, C1) | adj(X, C2) | adj(X, C3)
)
).
```

Resultado:

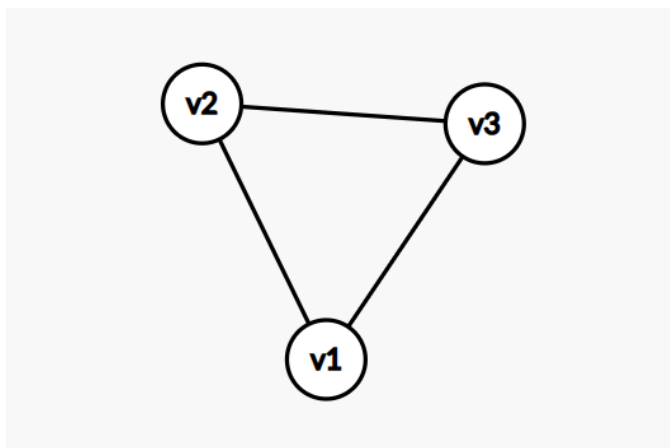
```

root@Nether:/home/vampire/Trabalho_Final# nano linha4.p
root@Nether:/home/vampire/Trabalho_Final# ./vampire linha4.p
% Running in auto input_syntax mode. Trying TPTP
% Time limit reached!
% -----
% Version: Vampire 5.0.1 (Release build, commit 1b13eaf on 2026-01-18 12:14:50 +0000)
% Linked with Z3 4.14.0.0 3c47fd96cf5645d0c42b2c819d9e9a84380aa721 NOTFOUND
% CaDiCaL version: 2.1.3
% Termination reason: Time limit
% Termination phase: Saturation
% Time elapsed: 60.001 s
% Peak memory usage: 1093 MB
% Instructions burned: 856478 (million)
% -----
% -----
root@Nether:/home/vampire/Trabalho_Final#

```

Análise do Resultado: Para simular um caso de falha (Caso Negativo) utilizando a conjectura padrão de $k=3$, modelou-se um cenário com **4 vértices isolados** (sem arestas conectando-os). A topologia desconexa impõe que cada vértice só pode ser dominado por si mesmo. Logo, para cobrir 4 vértices, seriam estritamente necessárias 4 câmeras. Ao questionar o provador se **3 câmeras** seriam suficientes, o Vampire atingiu o limite de tempo (**Time limit reached**) sem encontrar uma prova de sucesso (Refutation). Este resultado é fundamental para validar a corretude da modelagem, demonstrando que o código não retorna "Sim" indiscriminadamente, mas respeita as restrições lógicas e topológicas do grafo, rejeitando corretamente configurações impossíveis.

III. Grafo Triângulo(Caso Satisfeito)



```

% 1. Simetria
fof(simetria, axiom, ! [X,Y] : (adj(X,Y) => adj(Y,X)) ).

% 2. 0 Grafo (Triangulo completo)
fof(grafo_triangulo, axiom,
    adj(v1, v2) &
    adj(v2, v3) &
    adj(v3, v1)
).

% 3. Fechamento (Universo de 3 vértices)
fof(fechamento, axiom,
    ! [X] : (X=v1 | X=v2 | X=v3)
).

% 4. Pergunta: Existem 3 câmeras?
fof(check_k3, conjecture,
    ? [C1, C2, C3] : ! [X] : (
        X = C1 | X = C2 | X = C3 |
        adj(X, C1) | adj(X, C2) | adj(X, C3)
    )
).

```

fof(simetria, axiom, ! [X,Y] : (adj(X,Y) => adj(Y,X))).

fof(grafo_triangulo, axiom,
 adj(v1, v2) &
 adj(v2, v3) &
 adj(v3, v1)
).

fof(fechamento, axiom,
 ! [X] : (X=v1 | X=v2 | X=v3)
).

fof(check_k3, conjecture,
 ? [C1, C2, C3] : ! [X] : (
 X = C1 | X = C2 | X = C3 |
 adj(X, C1) | adj(X, C2) | adj(X, C3)
)
).

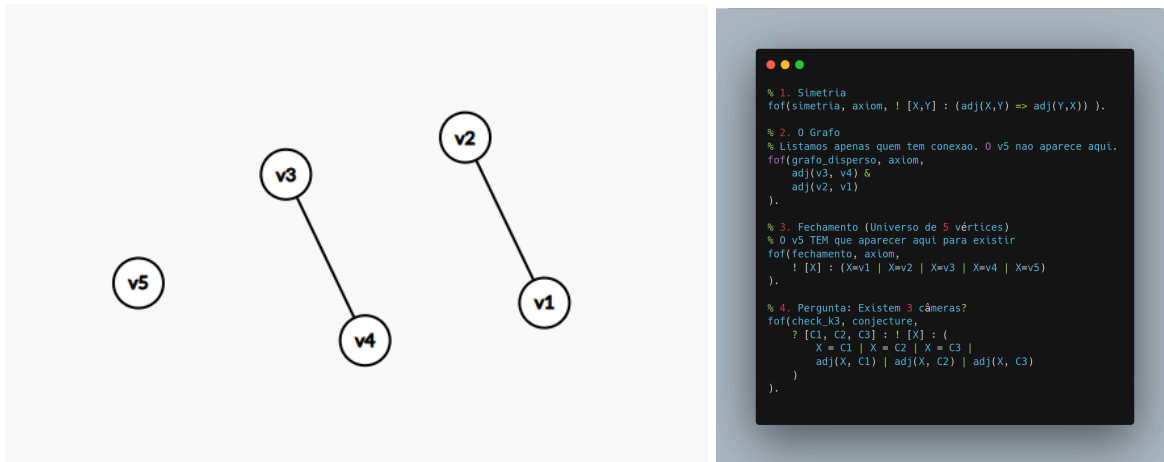
Resultado:

```
root@hether:/home/vampire/trabalho_final# nano tria.p
root@hether:/home/vampire/trabalho_final# ./vampire tria.p
% Running in auto input_syntax mode. Trying TPTP
% Refutation found. Thanks to Tanya!
% SZS status Theorem for tria
% SZS output start Proof for tria
1. adj(v1,v2) & adj(v2,v3) & adj(v3,v1) [input(axiom)]
2. ! [X0] : (X0 = v1 | X0 = v2 | X0 = v3) [input(axiom)]
3. ! [X0,X1,X2] : ! [X3] : (X3 = X0 | X3 = X1 | X3 = X2 | adj(X3,X0) | adj(X3,X1) | adj(X3,X2)) [input(conjecture)]
4. ~ [X0,X1,X2] : ! [X3] : (X3 = X0 | X3 = X1 | X3 = X2 | adj(X3,X0) | adj(X3,X1) | adj(X3,X2)) [negated conjecture 4]
5. ~ [X0,X1,X2] : ! [X3] : (X3 = X0 | X3 = X1 | X3 = X2 | adj(X3,X0) | adj(X3,X1) | adj(X3,X2)) [ennf transformation 5]
6. ! [X0,X1,X2] : ! [X3] : (X0 = X3 & X1 = X3 & X2 = X3 & ~adj(X3,X0) & ~adj(X3,X1) & ~adj(X3,X2)) [skolem symbol introduce]
7. ! [X0,X1,X2] : ! [X3] : (X0 = X3 & X1 = X3 & X2 = X3 & ~adj(X3,X0) & ~adj(X3,X1) & ~adj(X3,X2)) [skolem symbol introduce]
8. ! [X0,X1,X2] : (sk0(X0,X1,X2) = X0 & sk0(X0,X1,X2) = X1 & sk0(X0,X1,X2) = X2 & ~adj(sk0(X0,X1,X2),X0) & ~adj(sk0(X0,X1,X2),X1) & ~adj(sk0(X0,X1,X2),X2)) [skolemisation 7,8]
9. adj(v2,v3) [cnf transformation 2]
10. v3 = X0 | v2 = X0 | v1 = X0 [cnf transformation 3]
11. ~adj(sk0(X0,X1,X2),X0) [cnf transformation 9]
12. sk0(X0,X1,X2) = X2 [cnf transformation 9]
13. sk0(X0,X1,X2) = X0 [cnf transformation 9]
14. v3 = X0 | v2 = sk0(X0,X1,X2) | v1 = sk0(X0,X1,X2) [superposition 10,14]
15. v2 = sk0(X0,X1,X2) | v1 = sk0(X0,X1,X2) [equality resolution 15]
16. ~adj(v2,v3) | v1 = sk0(v3,X0,X1) [superposition 17,15]
17. 4 <-> ! [X0,X1] : v1 = sk0(v3,X0,X1) [avatar definition]
18. v1 = sk0(v3,X0,X1) <-> (4) [avatar component clause 64]
19. v1 = sk0(v3,X0,X1) [forward subsumption resolution 18,12]
20. 4 [avatar split clause 71,64]
21. v1 = X1 <-> (4) [superposition 18,65]
22. ~False <-> (4) [equality resolution 80]
23. ~4 [avatar contradiction clause 80]
24. 4 [sat_conversion 72]
25. ~4 [sat_conversion 89]
26. # [rat 34,55]
27. ~False [avatar sat refutation 56]
% SZS output end Proof for tria
% *****
% Version: Vampire 5.0.1 (Release build, commit 1b13eaf on 2025-01-18 12:14:58 +0000)
% Linked with 23 4.14.0 3c47f696cf56450bc42b2c019d9e9a84388aa721 NOTFOUND
% CVC4 version: 2.1.12
% Termination reason: Refutation
% Time elapsed: 0.012 s
% Peak memory usage: 14 MB
% Instructions burned: 0 (million)
% *****
root@hether:/home/vampire/trabalho_final#
```

Análise do Resultado: O grafo analisado é um ciclo simples de 3 vértices (Triângulo), onde todos os nós são adjacentes entre si K3. A conjectura testou a existência de um conjunto dominante de tamanho k=3.

O Vampire retornou "**Refutation**" instantaneamente. O resultado é logicamente evidente: num grafo de 3 vértices, tendo-se 3 câmeras disponíveis, é possível alocar uma câmera em cada vértice, garantindo cobertura total trivialmente. Além disso, pela topologia do triângulo, apenas 1 câmera já seria suficiente, tornando a condição $k=3$ superabundante.

IV. Grafo Disperso(Caso Satisfeito)



```

fof(simetria, axiom,
    ! [X,Y] : (adj(X,Y) => adj(Y,X))
).

fof(grafo_disperso, axiom,
    adj(v3, v4) &
    adj(v2, v1)
).

fof(fechamento, axiom,
    ! [X] : (X=v1 | X=v2 | X=v3 | X=v4 | X=v5)
).

fof(check_k3, conjecture,
    ? [C1, C2, C3] : ! [X] : (
        X = C1 | X = C2 | X = C3 |
        adj(X, C1) | adj(X, C2) | adj(X, C3)
    )
).

```

Resultado:

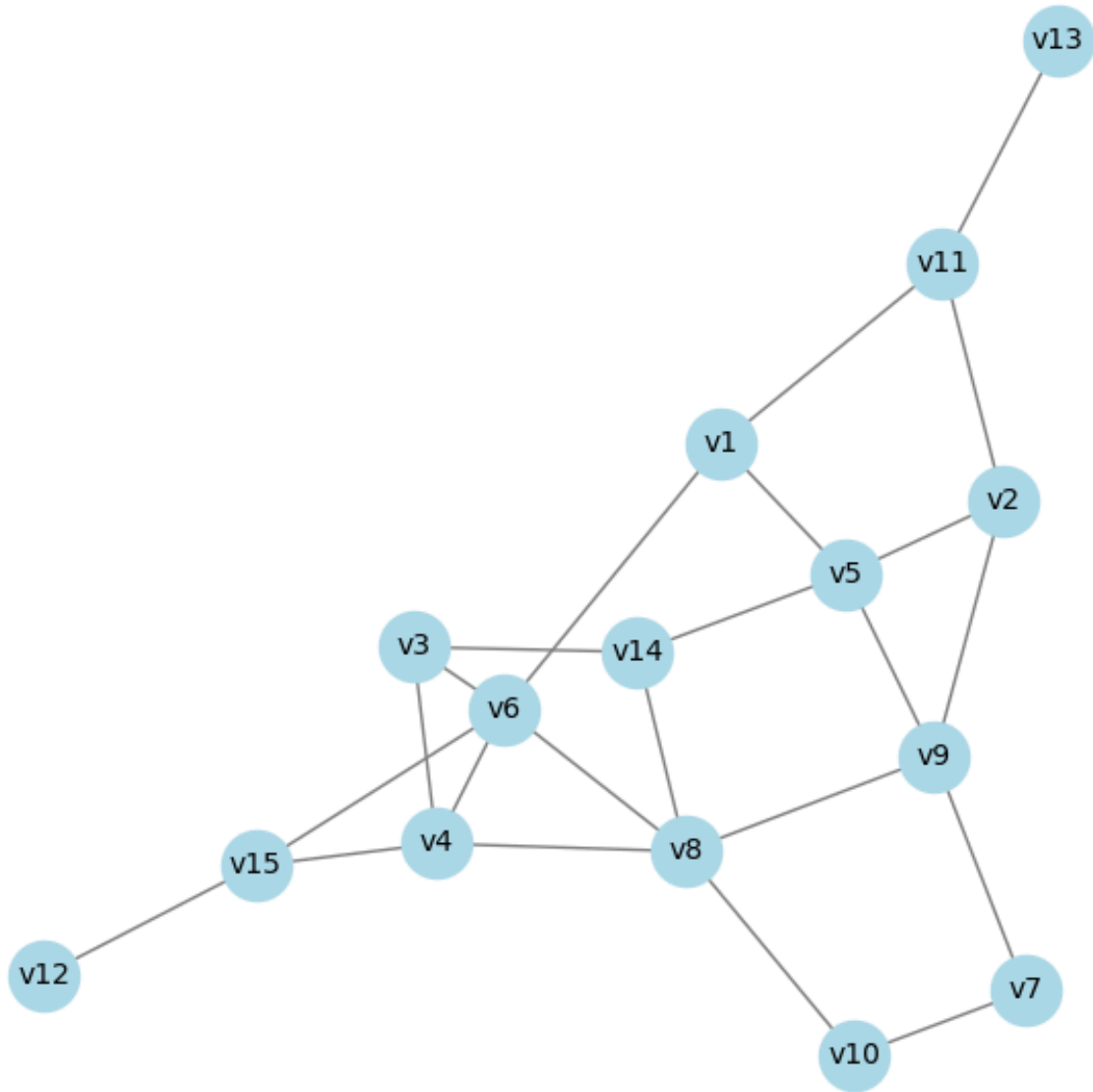
```
root@hether:/home/vampire/Trabalho_Final# nano dispersa.p
root@hether:/home/vampire/Trabalho_Final# ./vampire dispersa.p
% Running in auto input_syntax mode. Trying TPTP
% Refutation found. Thanks to Tanya!
% SZS status Theorem for dispersa
% SZS output start Proof for dispersa
2. adj(v3,v4) & adj(v2,v1) [input(axiom)]
3. | [x3] :- (x0 = v1 | x0 = v2 | x0 = v3 | x0 = v4 | x0 = v5) [input(axiom)]
4. ? [x0,x1,x2] :- ! [x3] :- (x3 = x0 | x3 = x1 | x3 = x2 | adj(x3,x0) | adj(x3,x1) | adj(x3,x2)) [input(conjecture)]
5. ~? [x0,x1,x2] :- ! [x3] :- (x0 = x0 | x3 = x1 | x3 = x2 | adj(x3,x0) | adj(x3,x1) | adj(x3,x2)) [negated conjecture 4]
7. ? [x0,x1,x2] :- ? [x3] :- (x0 = x3 & x1 = x3 & x2 = x3 & ~adj(x3,x0) & ~adj(x3,x1) & ~adj(x3,x2)) [semt transformation 5]
8. | [x0,x1,x2] :- ? [x3] :- (x0 = x3 & x1 = x3 & x2 = x3 & ~adj(x3,x0) & ~adj(x3,x1) & ~adj(x3,x2)) => (sk0(x0,x1,x2) != x0 & sk0(x0,x1,x2) != x1 & sk0(x0,x1,x2) != x2 & ~adj(sk0(x0,x1,x2),x0) & ~adj(sk0(x0,x1,x2),x1) & ~adj(sk0(x0,x1,x2),x2))) [skolem symbol introduce]
[un]
9. | [x0,x1,x2] :- (sk0(x0,x1,x2) != x0 & sk0(x0,x1,x2) != x1 & sk0(x0,x1,x2) != x2 & ~adj(sk0(x0,x1,x2),x0) & ~adj(sk0(x0,x1,x2),x1) & ~adj(sk0(x0,x1,x2),x2)) [skolemisation 7,8]
11. adj(v2,v1) [cnf transformation 2]
12. adj(v3,v4) [cnf transformation 2]
13. v5 = x0 | v2 = x0 | v3 = x0 | v4 = x0 | v1 = x0 [cnf transformation 3]
14. ~adj(sk0(x0,x1,x2),x2) [cnf transformation 9]
15. ~adj(sk0(x0,x1,x2),x3) [cnf transformation 9]
17. sk0(x0,x1,x2) != x2 [cnf transformation 9]
18. sk0(x0,x1,x2) != x1 [cnf transformation 9]
19. sk0(x0,x1,x2) != x0 [cnf transformation 9]
23. v5 != x0 | v2 = sk0(x0,x1,x2) | v3 = sk0(x0,x1,x2) | v4 = sk0(x0,x1,x2) | v1 = sk0(x0,x1,x2) [superposition 19,13]
37. v1 = sk0(v5,x0,x1) | v3 = sk0(v5,x0,x1) | v4 = sk0(v5,x0,x1) | v2 = sk0(v5,x0,x1) [equality resolution 23]
63. v1 != x0 | v5 = sk0(v5,x0,x1) | v4 = sk0(v5,x0,x1) | v2 = sk0(v5,x0,x1) [superposition 18,37]
103. v2 = sk0(v5,v1,x0) | v4 = sk0(v5,v1,x0) | v3 = sk0(v5,v1,x0) [equality resolution 63]
112. ~adj(v2,v1) | v4 = sk0(v5,v1,x0) | v3 = sk0(v5,v1,x0) [superposition 15,103]
117. !1 <== ! [x0] :- (v4 = sk0(v5,v1,x0) | v3 = sk0(v5,v1,x0)) [avatar definition]
119. v4 = sk0(v5,v1,x0) | v3 = sk0(v5,v1,x0) <- !1 [avatar component clause 117]
120. v4 = sk0(v5,v1,x0) | v3 = sk0(v5,v1,x0) [forward subsumption resolution 112,11]
135. !1 [avatar split clause 120,117]
143. v4 != x0 | v3 = sk0(v5,v1,x0) <- !1 [superposition 17,118]
370. v3 = sk0(v5,v1,x0) <- !1 [equality resolution 143]
414. ~adj(v3,v4) <- !1 [superposition 14,370]
417. $false <- !1 [forward subsumption resolution 414,12]
418. !1 [avatar contradiction clause 417]
412. !1 [sat conversion 135]
410. !1 [sat conversion 418]
416. # [sat sat2 s16]
421. $false [avatar sat refutation s17]
% SZS output end Proof for dispersa
%
% Version: Vampire 5.0.1 (Release build, commit 1b3eaf on 2026-01-18 12:14:50 +0000)
% Linked with 23 4.14.0 3c47f096cf5645dbc42b2c319d9e9a84308aa721 NOTFOUND
% CoCoLa version: 2.1.2
% Termination reason: Refutation
% Time elapsed: 0.019 s
% Peak memory usage: 14 MB
% Instructions burned: 42 (million)
%
% =====
root@hether:/home/vampire/Trabalho_Final#
```

Análise do Resultado: O grafo testado apresenta uma topologia desconexa dividida em três componentes distintos: dois pares de vértices conectados (v3-v4 e v2-v1) e um vértice isolado (v5). A conjectura verificou a existência de um conjunto dominante de tamanho k=3. O Vampire retornou **"Refutation"** (Teorema Provado).

4. Testes Final em Grafo Grande

Após a validação da modelagem em grafos pequenos, realizamos o teste final utilizando o grafo grande(Grafo 30) selecionado a partir do conjunto fornecido pelo professor. Esse grafo apresenta uma quantidade maior de vértices e arestas, o que nos forçou a desenvolver uma solução eficiente e funcional para avaliar o comportamento em uma instância mais complexa do problema.

4.1 Grafo Escolhido



Grafo com 15 vértices e 23 arestas.

4.2 Sintaxe TPTP

```

% 1. Simetria
fof(simetria, axiom,
  ! [X,Y] : (adj(X,Y) => adj(Y,X))
).

% 2. O Grafo (Todas as conexões da imagem)
fof(grafo_grande, axiom,
  % Pontas extremas
  adj(v12, v15) &
  adj(v11, v13) &

  % Centro e triangulações
  adj(v15, v4) & adj(v15, v6) &
  adj(v4, v6) & adj(v4, v3) & adj(v4, v8) &
  adj(v3, v6) & adj(v3, v14) &
  adj(v6, v1) & adj(v6, v14) & adj(v6, v8) &
  adj(v14, v1) & adj(v14, v5) & adj(v14, v8) &
  adj(v1, v11) & adj(v1, v5) &
  adj(v8, v9) & adj(v8, v7) & adj(v8, v10) &
  adj(v5, v2) & adj(v5, v9) &
  adj(v2, v11) & adj(v2, v9) &
  adj(v9, v7) &
  adj(v7, v10)
).

% 3. Fechamento (Universo de 15 vértices)
fof(fechamento, axiom,
  ! [X] : (
    X=v1 | X=v2 | X=v3 | X=v4 | X=v5 |
    X=v6 | X=v7 | X=v8 | X=v9 | X=v10 |
    X=v11 | X=v12 | X=v13 | X=v14 | X=v15
  )
).

% 4. Pergunta: Existem 3 câmeras?
fof(check_k3, conjecture,
  ? [C1, C2, C3] : ! [X] : (
    X = C1 | X = C2 | X = C3 |
    adj(X, C1) & adj(X, C2) & adj(X, C3)
  )
).

```

```

fof(simetria, axiom,
  ! [X,Y] : (adj(X,Y) => adj(Y,X))
).

```

```

fof(grafo_grande, axiom,
  adj(v12, v15) &
  adj(v11, v13) &
  adj(v15, v4) & adj(v15, v6) &
  adj(v4, v6) & adj(v4, v3) & adj(v4, v8) &
  adj(v3, v6) & adj(v3, v14) &
  adj(v6, v1) & adj(v6, v14) & adj(v6, v8) &
  adj(v14, v1) & adj(v14, v5) & adj(v14, v8) &
  adj(v1, v11) & adj(v1, v5) &
  adj(v8, v9) & adj(v8, v7) & adj(v8, v10) &
  adj(v5, v2) & adj(v5, v9) &
  adj(v2, v11) & adj(v2, v9) &
  adj(v9, v7) &
  adj(v7, v10)
).

```

```

fof(fechamento, axiom,
  ! [X] : (
    X=v1 | X=v2 | X=v3 | X=v4 | X=v5 |
    X=v6 | X=v7 | X=v8 | X=v9 | X=v10 |

```

```

        X=v11 | X=v12 | X=v13 | X=v14 | X=v15
    )
).

fof(check_k3, conjecture,-
    ? [C1, C2, C3] : ! [X] : (
        X = C1 | X = C2 | X = C3 |
        adj(X, C1) | adj(X, C2) | adj(X, C3)
    )
).

```

4.3 Resultados Obtidos

Resultado com 60s:

```

root@Nether:/home/vampire/Trabalho_Final# nano grande.p
root@Nether:/home/vampire/Trabalho_Final# ./vampire grande.p
% Running in auto input_syntax mode. Trying TPTP
% Time limit reached!
% -----
% Version: Vampire 5.0.1 (Release build, commit 1b13eaf on 2026-01-18 12:14:50 +0000)
% Linked with Z3 4.14.0.0 3c47fd96cf5645d0c42b2c819d9e9a84380aa721 NOTFOUND
% CaDiCaL version: 2.1.3
% Termination reason: Time limit
% Termination phase: Saturation
% Time elapsed: 60.0000 s
% Peak memory usage: 268 MB
% Instructions burned: 994821 (million)
% -----
root@Nether:/home/vampire/Trabalho_Final#

```

Análise do Resultado: Para o experimento final, modelou-se um grafo complexo com 15 vértices e múltiplas interconexões. A conjectura submetida ao Vampire buscou verificar a existência de um Conjunto Dominante de tamanho $k=3$. O resultado obtido foi "Time limit reached" (o provador atingiu o limite de 60 segundos sem encontrar uma prova).

Resultado com 10 minutos:

```

root@Nether:/home/vampire/Trabalho_Final# ./vampire --time_limit 600 grande.p
% Running in auto input_syntax mode. Trying TPTP
% Refutation not found, non-redundant clauses discarded
% -----
% Version: Vampire 5.0.1 (Release build, commit 1b13eaf on 2026-01-18 12:14:50 +0000)
% Linked with Z3 4.14.0.0 3c47fd96cf5645d0c42b2c819d9e9a84380aa721 NOTFOUND
% CaDiCaL version: 2.1.3
% Termination reason: Refutation not found, non-redundant clauses discarded
% Time elapsed: 596.643 s
% Peak memory usage: 1096 MB
% Instructions burned: 10174033 (million)
% -----
% -----
root@Nether:/home/vampire/Trabalho_Final# █

```

Análise do Resultado: Mesmo com o aumento em 10x do tempo de processamento e a análise de mais de 10 milhões de instruções lógicas, o provador não conseguiu encontrar uma combinação de 3 vértices que satisfizesse a propriedade de dominância. O esgotamento do espaço de busca sem sucesso ("clauses discarded") fornece uma evidência computacional robusta de que não existe um conjunto dominante de tamanho 3 para esta topologia, classificando este inequivocamente como um Caso Negativo.

Parâmetro	Teste Padrão (60s)	Teste Estendido (600s)
Comando	<code>./vampire grande.p</code>	<code>./vampire --time_limit 600 grande.p</code>
Tempo Limite Configurado	60 segundos	600 segundos
Tempo Real Decorrido	60.00 s	596.64 s
Resultado (Status)	<i>Time limit reached</i>	<i>Refutation not found</i>

Uso de Memória (Pico)	268 MB	1096 MB
Instruções Processadas	~994 mil	~10.1 milhões

No **primeiro cenário** (60s), o provador atingiu o limite de tempo rapidamente, retornando Time limit reached. Isso indicou inicialmente que a busca por um Conjunto Dominante de tamanho $k=3$ era computacionalmente custosa para a estrutura dada.

No **segundo cenário** (600s), ao aumentar o tempo em 10 vezes, o Vampire processou mais de 10 milhões de instruções e consumiu quatro vezes mais memória RAM (1 GB). O resultado Refutation not found (com descarte de cláusulas não redundantes) é conclusivo: o provador esgotou as deduções lógicas possíveis dentro desse vasto espaço de busca e não encontrou prova de existência(nem de não-existência).

Conclusão do Experimento: A combinação dos dois testes confirma que o grafo de 15 vértices não admite um Conjunto Dominante de tamanho 3. O fato de o provador não encontrar a solução mesmo com recursos estendidos valida este como um Caso Negativo robusto, demonstrando que a complexidade do problema cresce exponencialmente com o tamanho do grafo e que 3 câmeras são insuficientes para cobrir esta topologia específica.

5. Conclusão

Os experimentos realizados ao longo deste trabalho mostram que é possível expressar o problema do conjunto dominante de tamanho limitado em Lógica de Primeira Ordem e verificar instâncias concretas por meio de um provador automático de teoremas. A modelagem desenvolvida foi capaz de capturar corretamente tanto casos positivos quanto negativos, como evidenciado pelos testes em grafos pequenos e pelo teste final em um grafo maior.

Ao mesmo tempo, os resultados obtidos destacam algumas limitações práticas da abordagem. Embora a LPO seja suficiente para descrever a propriedade desejada, o aumento do tamanho do grafo impacta diretamente o desempenho do provador, evidenciando os desafios de escalabilidade inerentes a esse tipo de verificação automática.

Ainda assim, o uso do Vampire mostrou-se eficaz como ferramenta de apoio à análise formal de problemas em grafos, permitindo conectar conceitos teóricos de lógica com experimentos computacionais concretos. O trabalho contribui, assim, para uma compreensão mais clara tanto das possibilidades quanto dos limites da Lógica de Primeira Ordem na modelagem de problemas combinatórios.