Relatório de Lógica para Computação

Pedro Levi Silva Nunes, Rafael Rodrigo Araújo da Silva

21 de janeiro de 2026

1 Introdução

A teoria dos grafos é amplamente utilizada para modelar relações entre elementos em diversos contextos da ciência da computação. Um conceito fundamental nessa área é a conectividade, que indica se existe um caminho entre quaisquer dois vértices de um grafo. Verificar se um grafo é conexo é essencial para garantir propriedades como comunicação e acessibilidade.

Neste trabalho, esse problema é abordado por meio da Lógica de Primeira Ordem (LPO), na qual a conectividade do grafo é descrita formalmente usando predicados e quantificadores. A partir dessa formulação, um solver de LPO, Vampire, é utilizado para decidir automaticamente se o grafo satisfaz as condições necessárias para ser considerado conexo.

2 Objetivo

O objetivo desta pesquisa é identificar se um grafo $G = (V,E)$ é conexo ou não conexo por meio da Lógica de Primeira Ordem (LPO). Para isso, busca-se modelar formalmente os vértices e as relações de adjacência do grafo, bem como a noção de alcançabilidade entre vértices, utilizando um solver de LPO para decidir automaticamente se a propriedade de conectividade é satisfeita.

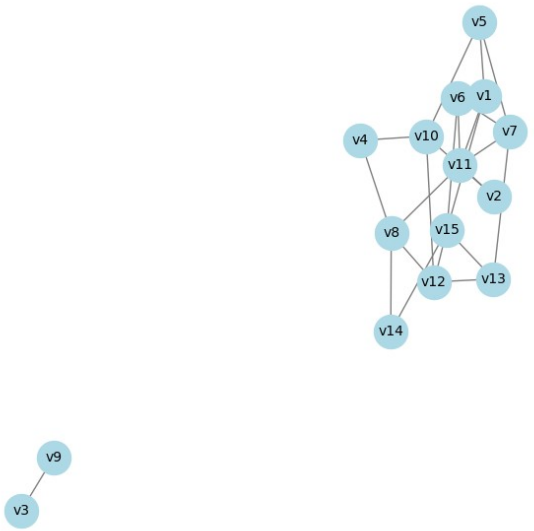

Figura 1: Grafo Principal

3 Metodologia

A metodologia adotada neste trabalho baseia-se na modelagem lógica de grafos por meio da Lógica de Primeira Ordem (LPO) e na verificação automática de conectividade utilizando um solver de LPO. Inicialmente, foram construídos quatro grafos distintos, cada um contendo cinco vértices, por meio da

ferramenta online ***Graph Editor***, disponível na **CS Academy**. Os grafos foram definidos de forma a incluir exemplos tanto de grafos conexos quanto não conexos, possibilitando a validação do modelo lógico em diferentes cenários.

A modelagem em LPO foi realizada definindo explicitamente o conjunto de vértices do grafo e restringindo o domínio às constantes *v1, v2, v3, v4, v5.* A relação de adjacência foi caracterizada como simétrica, representando grafos não direcionados. Em seguida, foi introduzido o predicado de alcançabilidade, responsável por representar a existência de caminhos entre vértices, utilizando axiomas que garantem adjacência direta e transitividade.

As ligações específicas de cada grafo foram inseridas por meio do predicado adj, variando conforme a estrutura de cada instância gerada no editor. Por fim, a conectividade do grafo foi expressa como uma conjectura, afirmando que todo par de vértices distintos é alcançável entre si. Cada grafo foi então submetido ao solver, que determinou se a conjectura era satisfeita ou não, permitindo identificar automaticamente se o grafo é conexo ou se gera um contraexemlo.

4 Modelagem LPO

Durante a contrução do modelo LPO, foram definidos predicados e, a partir desses predicados, modeladas as fórmulas compreendendo as propriedades de um grafo não direcionado, onde as arestas não têm direção; se existe $\mathbf{E} = (\mathbf{A}, \mathbf{B}) \rightarrow \mathbf{E} = (\mathbf{B}, \mathbf{A})$.

4.1 Predicados

Para execução dos testes foram definidos os seguintes predicados:

- vertice(X) : X é um vértice do grafo.
- adj(X, Y) : X é adjascente a Y.
- alc(X, Y) : X alcança Y.

4.2 Modelo

Para Modelar o problema compreendendo as propriedades de um grafo não direcionado definimos as seguintes fórmulas em LPO, usando "caminho"para definir alcançabilidade. Abaixo se encontra listadas as fórmulas utilizadas nos grafos testes, de 5 vértices. Quando aplicadas no grafo principal, apenas foram incrementados o dominio, os vértices e as arestas com o fito de compreender o grafo de uma maior proporção.

1. Define o universo/dominio:
   fof(dominio, axiom, (

   $$![X] : ((vertice(X)) \rightarrow (X = v1 \quad | \quad X = v2 \quad | \quad X = v3 \quad | \quad X = v4 \quad | \quad X = v5)))$$

).

2. Define os vertice do grafo:

   fof(vertices, axiom,

   $$(vertice(v1) \quad \& \quad vertice(v2) \quad \& \quad vertice(v3) \quad \& \quad vertice(v4) \quad \& \quad vertice(v5))$$

).

3. Define a propriedade de simetria do grafo não direcionado:

   fof(h1, axiom,

   $$![X, Y] : (adj(X, Y) \rightarrow adj(Y, X))$$

).

4. Define que se entre dois vértices há uma aresta, então há um caminho entre eles dois

   fof(h3, axiom,
   $$![X,Y] : (adj(X,Y) \rightarrow alc(X,Y))$$
).

5. Define se há um caminho entre dois vértices, X e Y, e há uma aresta entre Y e um outro vértice Z, então há um caminho entre X e Z:

   fof(h4, axiom,
   $$![X,Y,Z] : ((alc(X,Y) \quad \& \quad adj(Y,Z)) \rightarrow alc(X,Z))$$
).

6. Define as arestas entre os vértices:

   fof(ligacoes, axiom,
   $$adj(v1,v2) \quad \& \quad adj(v2,v5) \quad \& \quad adj(v5,v3) \quad \& \quad adj(v5,v4) \quad \& \quad adj(v4,v3)$$
).

7. Conclui que se X e Y são vértices distintos, então há um caminho entre X e Y:

   fof(conexo, conjecture,
   $$![X,Y] : ((vertice(X) \quad \& \quad vertice(Y) \quad \& \quad (X! = Y)) \rightarrow (alc(X,Y)))$$
).

5 Testes

Foram gerados 4 grafos com a ferramenta ***Graph Editor***, disponível na **CS Academy**, para testar o modelo LPO de conectividade, onde cada grafo foi composto por 5 vértices, mesclando entre grafos conexos e desconexos.

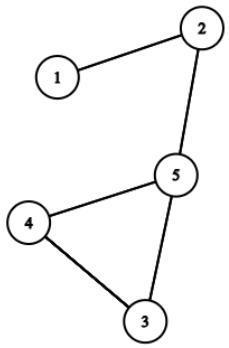

Figura 2: Grafo Teste1

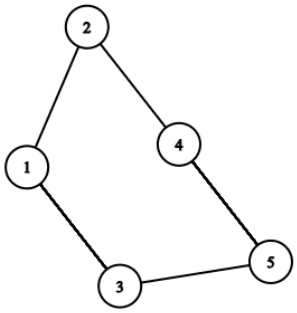

Figura 3: Grafo Teste2

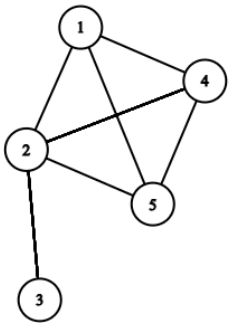

Figura 4: Grafo Teste3

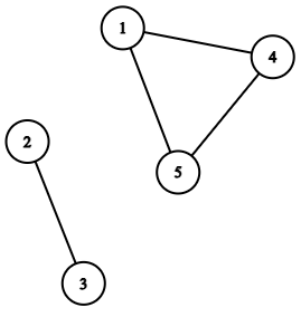

Figura 5: Grafo Teste4

6 Resultados

Os resultados foram obtidos a partir de solver de LPO, Vampire, onde o método de resolução consiste em definir o domínio e predicados do problema, modelando objetos e relações. As propriedades e regras do sistema são codificadas como fórmulas lógicas, incluindo axiomas e conjecturas. Essas fórmulas são fornecidas ao Vampire, que utliza um sistema de prova similar ao Tableaux Analítico, onde assume a negação da conjectura e usa essa valoração como base para chegar a um resultado. Caso a negação seja inconsistente, segundo os axiomas definidos, gera um estado de refutação ("Refutation"), provando que a conjectura é verdadeira. Caso contrário, gera um estado de satisfazível ("Satisfiable"), afirmando que, dado os axiomas, existe pelo menos um modelo onde a negação da conjectura é consistente.

6.1 Grafo teste 1:

De acordo com os teste realizados pelo solver, seguindo os axiomas e a conjectura para definir a conectividade de forma lógica, se foi possível identificar que o grafo teste 1 é conexo, pois ao negar da conjectura chegou-se a um estado de refutação, provando que a conjectura é válida.

```
rodrigo@rodrigo-Nitro-AN515-51:~/Desktop/vampire/build/Exercises$ ../vampire grafo1.p
s11. ~7 | ~1 [sat_conversion 199]
s12. ~9 | ~4 [sat_conversion 200]
s13. ~6 | ~4 [sat_conversion 232]
s15. ~8 | ~4 [sat_conversion 244]
s17. ~10 | ~4 [sat_conversion 257]
s18. ~10 | ~5 [sat_conversion 258]
s19. ~9 | ~5 [sat_conversion 274]
s20. ~9 | ~1 [sat_conversion 297]
s22. ~9 | ~3 [sat_conversion 304]
s25. ~8 | ~5 [sat_conversion 309]
s27. ~6 | ~5 [sat_conversion 318]
s33. ~6 | ~1 [sat_conversion 335]
s36. ~6 | ~3 [sat_conversion 346]
s38. ~10 | ~3 [sat_conversion 355]
s41. ~8 | ~3 [sat_conversion 365]
s42. ~10 | ~1 [sat_conversion 371]
s43. ~8 | ~1 [sat_conversion 376]
s44. ~5 [rat s2,s8,s18,s19,s25,s27]
s45. ~4 [rat s2,s10,s12,s13,s15,s17]
s46. ~3 [rat s2,s9,s22,s36,s38,s41]
s47. ~2 [rat s2,s3,s4,s5,s6,s7]
s48. 1 [rat s1,s44,s45,s46,s47]
s49. ~8 [rat s43,s48]
s50. ~10 [rat s42,s48]
s51. ~6 [rat s33,s48]
s52. ~9 [rat s20,s48]
s53. ~7 [rat s11,s48]
s54. # [rat s2,s50,s51,s49,s52,s53]
377. $false [avatar sat refutation s54]
% SZS output end Proof for grafo1
% ------------------------------
% Version: Vampire 5.0.1 (Release build, commit 1b13eafde on 2026-01-18 12:14:50 +0000)
% CaDiCaL version: 2.1.3
% Termination reason: Refutation
% Time elapsed: 0.014 s
% Peak memory usage: 13 MB
% ------------------------------
```

Figura 6: Resultado Teste 1

6.2 Grafo teste 2:

De acordo com os teste realizados pelo solver, seguindo os axiomas e a conjectura para definir a conectividade de forma lógica, se foi possível identificar que o grafo teste 2 é conexo, pois ao negar da conjectura chegou-se a um estado de refutação, provando que a conjectura é válida.

```
rodrigo@rodrigo-Nitro-AN515-51:~/Desktop/vampire/build/Exercises$ ../vampire grafo2.p
s12. ~7 | ~1 [sat_conversion 195]
s14. ~9 | ~4 [sat_conversion 206]
s16. ~6 | ~4 [sat_conversion 214]
s17. ~8 | ~4 [sat_conversion 219]
s18. ~10 | ~4 [sat_conversion 224]
s19. ~10 | ~5 [sat_conversion 225]
s20. ~9 | ~5 [sat_conversion 241]
s21. ~9 | ~1 [sat_conversion 245]
s22. ~9 | ~3 [sat_conversion 259]
s25. ~8 | ~5 [sat_conversion 275]
s26. ~6 | ~5 [sat_conversion 282]
s27. ~6 | ~1 [sat_conversion 286]
s29. ~6 | ~3 [sat_conversion 294]
s30. ~10 | ~3 [sat_conversion 300]
s33. ~8 | ~3 [sat_conversion 310]
s34. ~10 | ~1 [sat_conversion 321]
s36. ~8 | ~1 [sat_conversion 329]
s37. ~5 [rat s2,s9,s19,s20,s25,s26]
s38. ~4 [rat s2,s11,s14,s16,s17,s18]
s39. ~3 [rat s2,s10,s22,s29,s30,s33]
s40. ~2 [rat s2,s3,s5,s6,s7,s8]
s41. 1 [rat s1,s37,s38,s39,s40]
s42. ~8 [rat s36,s41]
s43. ~10 [rat s34,s41]
s44. ~6 [rat s27,s41]
s45. ~9 [rat s21,s41]
s46. ~7 [rat s12,s41]
s47. # [rat s2,s43,s44,s42,s45,s46]
330. $false [avatar sat refutation s47]
% SZS output end Proof for grafo2
% ------------------------------
% Version: Vampire 5.0.1 (Release build, commit 1b13eafde on 2026-01-18 12:14:50 +0000)
% CaDiCaL version: 2.1.3
% Termination reason: Refutation
% Time elapsed: 0.018 s
% Peak memory usage: 13 MB
% ------------------------------
```

Figura 7: Resultado Teste 2

6.3 Grafo teste 3:

De acordo com os teste realizados pelo solver, seguindo os axiomas e a conjectura para definir a conectividade de forma lógica, se foi possível identificar que o grafo teste 3 é conexo, pois ao negar da conjectura chegou-se a um estado de refutação, provando que a conjectura é válida.

```
rodrigo@rodrigo-Nitro-AN515-51:~/Desktop/vampire/build/Exercises$ ../vampire grafo3.p
s11. ~7 | ~1 [sat_conversion 235]
s13. ~9 | ~4 [sat_conversion 240]
s14. ~6 | ~4 [sat_conversion 253]
s15. ~8 | ~4 [sat_conversion 267]
s17. ~10 | ~4 [sat_conversion 280]
s18. ~10 | ~5 [sat_conversion 281]
s19. ~9 | ~5 [sat_conversion 284]
s21. ~9 | ~3 [sat_conversion 291]
s22. ~9 | ~1 [sat_conversion 296]
s24. ~8 | ~5 [sat_conversion 302]
s25. ~6 | ~5 [sat_conversion 308]
s26. ~6 | ~1 [sat_conversion 313]
s28. ~6 | ~3 [sat_conversion 321]
s30. ~10 | ~3 [sat_conversion 330]
s33. ~8 | ~3 [sat_conversion 340]
s34. ~10 | ~1 [sat_conversion 346]
s35. ~8 | ~1 [sat_conversion 351]
s36. ~5 [rat s2,s8,s18,s19,s24,s25]
s37. ~4 [rat s2,s10,s13,s14,s15,s17]
s38. ~3 [rat s2,s9,s21,s28,s30,s33]
s39. ~2 [rat s2,s3,s4,s5,s6,s7]
s40. 1 [rat s1,s36,s37,s38,s39]
s41. ~8 [rat s35,s40]
s42. ~10 [rat s34,s40]
s43. ~6 [rat s26,s40]
s44. ~9 [rat s22,s40]
s45. ~7 [rat s11,s40]
s46. # [rat s2,s42,s43,s41,s44,s45]
352. $false [avatar sat refutation s46]
% SZS output end Proof for grafo3
% ------------------------------
% Version: Vampire 5.0.1 (Release build, commit 1b13eafde on 2026-01-18 12:14:50 +0000)
% CaDiCaL version: 2.1.3
% Termination reason: Refutation
% Time elapsed: 0.014 s
% Peak memory usage: 13 MB
% ------------------------------
```

Figura 8: Resultado Teste 3

6.4 Grafo teste 4:

De acordo com os teste realizados pelo solver, seguindo os axiomas e a conjectura para definir a conectividade de forma lógica, se foi possível identificar que o grafo teste 4 é desconexo, pois ao negar da conjectura chegou-se a um estado de satisfazível, provando que existe pelo menos um modelo onde a negação da conjectura é consistente.

```
rodrigo@rodrigo-Nitro-AN515-51:~/Desktop/vampire/build/Exercises$ ../vampire grafo4.p
cnf(u122,axiom,
    ~adj(v4,X0) | alc(v1,X0)).

cnf(u115,negated_conjecture,
    v1 != v3).

cnf(u34,negated_conjecture,
    vertice(sK0)).

cnf(u27,axiom,
    ~alc(X0,X1) | alc(X0,X2) | ~adj(X1,X2)).

cnf(u127,axiom,
    ~adj(v1,X0) | alc(v5,X0)).

cnf(u46,axiom,
    alc(v3,v2)).

cnf(u39,axiom,
    adj(v4,v1)).

cnf(u28,axiom,
    adj(v4,v5)).

cnf(u21,axiom,
    vertice(v4)).

% SZS output end Saturation.
% SZS output start Definitions and Model Updates.
% SZS output end Definitions and Model Updates.
% ------------------------------
% Version: Vampire 5.0.1 (Release build, commit 1b13eafde on 2026-01-18 12:14:50 +0000)
% CaDiCaL version: 2.1.3
% Termination reason: Satisfiable
% Time elapsed: 0.006 s
% Peak memory usage: 12 MB
% ------------------------------
```

Figura 9: Resultado Teste 4

6.5 Grafo Principal:

De acordo com os teste realizados pelo solver, seguindo os axiomas e a conjectura para definir a conectividade de forma lógica, se foi possível identificar que o grafo principal é desconexo, pois ao

negar da conjectura chegou-se a um estado de satisfazível, provando que existe pelo menos um modelo onde a negação da conjectura é consistente.

```
cnf(u34,axiom,
    vertice(v1)).

cnf(u709,axiom,
    alc(v10,v14)).

cnf(u849,axiom,
    ~adj(v4,X0) | alc(v15,X0)).

cnf(u87,axiom,
    alc(v4,v8)).

cnf(u428,axiom,
    ~adj(v8,X0) | alc(v1,X0)).

cnf(u391,axiom,
    alc(v8,v6)).

cnf(u56,axiom,
    adj(v2,v11)).

cnf(u384,axiom,
    alc(v7,v2)).

cnf(u65,axiom,
    adj(v7,v5)).

% SZS output end Saturation.
% SZS output start Definitions and Model Updates.
% SZS output end Definitions and Model Updates.
% ------------------------------
% Version: Vampire 5.0.1 (Release build, commit 1b13eafde on 2026-01-18 12:14:50 +0000)
% CaDiCaL version: 2.1.3
% Termination reason: Satisfiable
% Time elapsed: 0.046 s
% Peak memory usage: 12 MB
% ------------------------------
```

Figura 10: Resultado Grafo Final

7 Conclusão

O modelo LPO desenvolvido foi testado inicialmente em quatro grafos pequenos, nos quais apresentou resultados corretos para a conectividade, comprovando seu funcionamento esperado. Ao ser aplicado no grafo final, o modelo indicou que ele é desconexo, confirmando a capacidade do modelo em identificar a ausência de conectividade em grafos maiores.

Assim, pode-se concluir que o modelo é eficaz na verificação da conectividade de grafos, funcionando corretamente tanto em casos simples quanto em casos mais complexos.

Referências

[Feofiloff 2026]FEOFILOFF, P. P. *Algoritmos para Grafos*. 2026. https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/graphs.html.

[Graph Editor - CS Academy 2026]GRAPH Editor - CS Academy. 2026. https://csacademy.com/app/graph_editor/.

[Vampire: Theorem Prover 2026]VAMPIRE: Theorem Prover. 2026. https://github.com/vprover/vampire.git.

[Feofiloff 2026] [Graph Editor - CS Academy 2026] [Vampire: Theorem Prover 2026]