Propositional Logic and Directed Graph Framework for Modelling Testimony Contradiction in Phoenix Wright: Ace Attorney

Sherin Felicia Danessa - 13525089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: danessasherin@gmail.com , 13525089@std.stei.itb.ac.id

***Abstract*—**This paper presents a formal framework for modeling the testimony-contradiction mechanics of the visual novel video game *Phoenix Wright: Ace Attorney* using two branches of discrete mathematics: propositional logic and directed graph theory. Each witness testimony statement is encoded as a propositional formula, and the consistency between a set of statements and a given body of evidence is evaluated through logical satisfiability analysis. Simultaneously, the relationships between statements and evidence items are represented as a directed graph G = (V, E), where logical contradictions show themselves as specific edges detectable through graph traversal. A Python implementation using the NetworkX library is provided to demonstrate automated contradiction detection on the first case of the game, *Turnabout Sisters*. The results show that the court deduction mechanics of Ace Attorney map naturally onto formal logical and graph-theoretic structures, providing a reproducible model for contradiction-based reasoning.

Keywords—propositional logic, directed graph, contradiction detection, Ace Attorney, testimony modeling, discrete mathematics

I. Introduction

Ace Attorney is a visual novel adventure game series developed by Capcom in 2001. The first entry of the series was released in 2001 with the title *Phoenix Wright: Ace Attorney* with a story telling fashioned after legal dramas around a courtroom setting. The core interactive gameplay mechanic consists of courtroom trial sequences where the prosecution presents witnesses whose testimonies require the player to identify the logical contradictions or fallacies between individual statements and pieces of evidence collected during investigation phases which is done by identifying a structural breakdown in a witness's narrative by presenting conflicting evidence during a cross examination.

While the game presents these contradictions as intuitive "OBJECTION!" momentum, the underlying architecture operates as a thorough structural reasoning system. Each testimony statement makes a factual claim about the world and each piece of evidence constrains the set of possible worlds in which those claims can be true at the same time. When the world does not exist or when a set of statements and evidence is jointly unsatisfiable, a contradiction has been found.

This paper formalizes the gameplay mechanics of *Ace Attorney* by demonstrating that its core legal disputes are completely mapping compatible with formal discrete mathematics. First, propositional logic is used to encode statements and evidence by logical formulas and contradictions are defined as the unsatisfiability of the conjunction. Second, directed graph theory is used to represent the dependency and conflict relationships between statements and evidence which enables algorithmic contradiction detection through edge traversal. By using both domains, we are able to construct a computational engine that treats contradiction detection not as a certain mathematical execution instead of an interpretive challenge.

This paper has three primary objectives: (1) to demonstrate that the courtroom deduction mechanics in Ace Attorney can be formally modeled using discrete mathematics concepts; (2) to apply this formal framework to the game's first major case as a practical case study; and (3) to develop a Python-based implementation of the framework for automated contradiction detections.

II. Theoretical Background

A. Propositional Logic

Propositional logic, also known as sentential logic or zeroth-order logic, is the branch of formal logic concerned with the study of propositions and their logical relationships. Statements that are either true (**T**) or false (**F**) are combined using logical connectives to form compound formulas. Propositions are the atomic building blocks of propositional logic and are typically denoted by lowercase letters such as **p, q, r.**

a. Logical Connectives

Compound propositions are formed from atomic propositions using logical connectives. The standard connectives are defined as follows:

a) Negation (¬)
Given a proposition **p**, its negation ¬**p** is true if and only if **p** is false
b) Conjunction (∧)
The conjunction **p**∧**q** is true if and only if both **p** and **q** are true
c) Disjunction (∨)
The disjunction **p**∨**q** is true if and only if at least one of **p** or **q** is true
d) Implication (→)
The implication **p**→**q** reads "if **p**, then **q**," is false only when **p** is true and **q** is false
e) Biconditional (↔)
The biconditional **p**↔**q** reads "**p** if and only if **q**," is true when **p** and **q** have the same truth value

b. Tautologies, Contradictions, and Satisfiability

A propositional formula φ is evaluated under a truth assignment $v : P \to \{T, F\}$, where P is the set of atomic propositions. A formula φ is satisfiable if there exists an assignment of truth values to its atomic propositions that makes φ evaluate to true. A set of formulas $\Gamma = \{\varphi_1, \varphi_2, ..., \varphi_n\}$ is jointly satisfiable if there exists a single assignment making all formulas in Γ true simultaneously. If no such assignment exists, the set is unsatisfiable, which we write as:

$$\boldsymbol{\Gamma} \vDash \bot$$

This is equivalent to saying that the formula $\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n$ is itself a contradiction. A fundamental result is that joint unsatisfiability is equivalent to deriving both a proposition and its negation from Γ:

$$\boldsymbol{\Gamma} \vDash \bot \iff \exists \mathbf{p} : (\boldsymbol{\Gamma} \vDash \mathbf{p}) \wedge (\boldsymbol{\Gamma} \vDash \neg \mathbf{p})$$

This result is the logical foundation of the contradiction detection approach: a witness statement s_i and an evidence item e_n form a contradiction if and only if their propositional encodings $\varphi(s_i)$ and $\varphi(e_n)$ together entail both some atomic proposition and its negation — that is, if $\{\varphi(s_i), \varphi(e_n)\} \vDash \bot$.

Satisfiability for a formula over n atomic propositions can be verified by constructing a truth table of 2^n rows. For this paper, each statement-evidence pair shares at most three propositions, making this approach tractable.

B. Directed Graph Theory

A directed graph (digraph) G = (V, E) consists of a vertex set V and an edge set $E \subseteq V \times V$ of ordered pairs. An edge (u, v) represents a directed relationship from u to v, distinct from (v, u) [3]. In this paper, G is extended to an edge-labeled digraph G = (V, E, Σ, λ), where Σ = {CONTRADICTS, DEPENDS ON, REVISED TO, CONSISTENT} and $\lambda : E \to \Sigma$ assigns a semantic label to each edge.

The vertex set is partitioned into three disjoint subsets:

$$\mathbf{V = S \cup E_v \cup C,}$$

$$\mathbf{with\ \ S \cap E_v = S \cap C = E_v \cap C = \varnothing}$$

where S is the set of testimony statement nodes, E_v is the set of evidence nodes, and C is the set of contradiction result nodes introduced for visualization purposes.

Contradiction detection reduces to finding all edges $(s_i, e_n) \in E$ such that $\lambda(s_i, e_n) = \text{CONTRADICTS}$. This can be performed in O(|E|) time by iterating over all edges and filtering by label — a linear-time operation that scales efficiently with the size of the testimony.

III. Proposed Modeling Framework

To bridge the gap between abstract game mechanics and discrete mathematics, we can establish a formal mapping that translates witness statements and evidence items into logical formulas, as well as their structural relationships into an edge-labeled directed graph.

A. Encoding Testimony Statements

Let a testimony $T = (S, E_v)$ consist of an ordered set of statement nodes $S = \{s_1, ..., s_n\}$ and a set of evidence nodes $E_v = \{e_1, ..., e_n\}$. Each node is assigned a propositional formula via an encoding function:

$$\boldsymbol{\varphi : V \to L}$$

where L is the set of well-formed propositional formulas over a shared set of atomic propositions P. Atomic propositions represent factual claims about the world of the case, such as the time of death, a character's location, or knowledge of an object's identity.

A contradiction between statement s_i and evidence e_n is formally defined as:

$$\boldsymbol{\varphi(s_i) \wedge \varphi(e_n) \vDash \bot}$$

B. Graph Construction Algorithm

Given testimony $T = (S, E_v)$, the testimony graph G_T is constructed in three steps. First, each element of S and E_v becomes a vertex in V. Second, for each pair (s_i, e_n) such that $\varphi(s_i) \wedge \varphi(e_n) \vDash \bot$, a CONTRADICTS edge is added. Third, for each pair (s_i, s_n) such that $\varphi(s_i) \to \varphi(s_n)$, a DEPENDS ON edge is added. Finally, for each statement s_i revised during cross-examination, a REVISED TO edge connects the original statement to its revised form.

The contradiction detection algorithm then runs in $O(|S| \times |E_v| \times 2|P|)$ time, where |P| is bounded by the number of shared propositions in any pair, typically two or three in Ace Attorney testimonies, making the approach computationally efficient though it may vary.

IV. Case Study

A. Case Overview

Turnabout Sisters is the first full case of *Phoenix Wright: Ace Attorney*. The victim is defense attorney, Mia Fey,

who was found murdered in her office on September 5th at 9:00 PM, struck by a blunt object identified as "The Thinker", which is a clock disguised as a statue, created by Larry Butz. The defendant is Mia's younger sister, Maya Fey. Two witnesses provide cross-examinable testimony: hotel guest April May, who claims to have witnessed the murder from across the street, and powerful corporate executive Redd White, who later also claims to have witnessed the scene.

B. Evidence Encoding

Table I presents the propositional encoding of the key testimony statements and evidence items used in the case analysis. Each node is assigned a label and a formula $\varphi(\cdot)$ over atomic propositions drawn from the case facts.

Node	Label	Natural Language Statement	Propositional Encoding $\varphi(\cdot)$
S1	DODGE_RIGHT	Victim dodged first attack and ran to the right	*dodged(victim)* $\wedge$ *ran_right(victim)*
S2	HIPPIE_HIT	Girl in hippie clothes attacked victim with The Thinker	*attacker(maya, victim, thinker)*
S3	WEAPON_ID	Weapon used was "The Thinker" clock-statue	*weapon(thinker)*
S4	APRIL_KNOWS	April May knows The Thinker is a clock	*knows_clock(april, thinker)*
S5	HEARD_CLOCK	April claims she heard it was a clock (1st revision)	*knows_clock(april)* $\leftarrow$ *heard*
S5b	SAW_STORE	April claims she saw The Thinker at a store (2nd revision)	*saw_at_store(april, thinker)*
S6	WHITE_SAW	Redd White saw a man attack a woman from his window	*saw(white, attack, window)*
S7	RAN_LEFT	White says victim ran to the left (his perspective)	*ran_left_whitePOV(victim)*
S8	ONE_BLOW	Victim was struck twice (implied by two directions)	*struck_twice(victim)*
S9	SAW_STAND	White saw the glass light stand fall from his window	*visible_from_window(light_stand)*
S10	STAND_THERE	Light stand was already in the office that evening	*in_office_before_sep5(light_stand)*
E1	AUTOPSY	Autopsy: death at 9PM, single blunt force trauma, instantaneous	*time_of_death(victim,9PM)* $\wedge$ *single_blow(victim)*
E2	CELLPHONE	Maya's cellphone: Mia holds The Thinker @ 9:27 AM Sep 5	*holds(mia,thinker,9:27AM_sep5)*
E3	THINKER	The Thinker: made by Larry Butz only, never sold publicly	*maker(thinker,larry)* $\wedge$ $\neg$*ever_sold(thinker)* $\wedge$ $\neg$*public_knowledge(thinker_clock)*
E4	WIRETAP	Wiretap found in April May's hotel room	*has_wiretap(april)* $\wedge$ *was_listening(april, fey_office)*
E5	BELLBOY	Bellboy: April ordered room service at exactly 9PM	*location(april, hotel_room, 9PM)* $\wedge$ *ordering_room_service(april,9PM)*
E6	GLASS	Glass shards: light stand purchased September 4th	*purchase_date(light_stand, sep4)*

Fig.1: Propositional Encoding of Turnabout Sisters Testimony Nodes

C. Contradiction Analysis April May's Testimony

April May provides the following key statement during cross-examination: she identifies the murder weapon as "The Thinker" and claims to know it is a clock. When pressed on how she knows this, she first claims she heard it from somewhere (S5), then revises to claiming she saw it at a store (S5b). Both revised accounts are contradicted by evidence.

The primary contradiction is between S4 and evidence E3. April knows The Thinker is a clock (S4), but The Thinker was made by Larry Butz exclusively and has never been sold anywhere (E3). Their conjunction yields:

$$\varphi(S4) \wedge \varphi(E3) = \text{knows\_clock(april)} \wedge \neg\text{ever\_sold(thinker)} \wedge \neg\text{public\_knowledge(thinker\_clock)}$$

$$\Rightarrow \text{how\_does\_she\_know(april, thinker\_clock) has no valid assignment} \Rightarrow \bot$$

Figure 2 presents the truth table verifying this contradiction across the relevant atomic propositions p (April knows it is a clock), q (it was never sold), and r (she claims to have seen it at a store):

P April knows it's a clock	q The clock was never sold	r April claims she saw it at the store	p∧q∧r	Contradiction ?
T	T	T	T	Yes
T	T	F	F	No
T	F	T	F	No
F	T	T	F	No
F	F	F	F	No

Fig. 2: Truth table for April May's clock knowledge contradiction.

When p ∧ q ∧ r are all assigned true (row 1), the conjunction is satisfiable, but this is precisely the scenario that constitutes the contradiction, since q and r are mutually exclusive facts and can not exist in the same world.

The second contradiction involves S5b and E3. April revises her account to claim she saw The Thinker at a store. However, E3 establishes that The Thinker was never sold anywhere, making saw_at_store(april, thinker) directly inconsistent with ¬ever_sold(thinker):

$$\varphi(S5b) \wedge \varphi(E3) = \text{saw\_at\_store(april,thinker)} \wedge \neg\text{ever\_sold(thinker)} \models \bot$$

This reveals the actual source of April's knowledge: the wiretap (E4) found in her hotel room, which allowed her to overhear conversations in Mia's office — and thereby learn that The Thinker was a clock — without ever being sold or publicly known.

D. Contradiction Analysis: Redd White's Testimony

Redd White's first contradiction arises from the directional perspective of the attack. April testifies that the victim "ran to the right" (S1). White, however, testifies that the victim ran to the left (S7). Initially, this appears to be a direct contradiction. However, the floor plan reveals that both accounts are actually correct *only if the two witnesses were facing opposite directions*. Since the victim's back was to the window (meaning April, looking through the hotel window, was facing the same direction as the victim), White could only have seen her run "left" if he was standing inside the office facing her.

To cover up this slip and maintain his lie that he was at the hotel window, White changes his testimony to claim the victim ran left, dodged, and *then* ran right (S7b), delivering a final blow (S8). This combined account implies two separate attacks, which directly contradicts the autopsy report (E1) stating the victim died from a single blow:

$$\varphi(S8\_implied) \wedge \varphi(E1) = \text{struck\_twice(victim)} \wedge \text{single\_blow(victim)} \models \bot$$

White's second contradiction involves the glass light stand. White testifies that he saw the light stand fall through the window (S9). However, the floor plan shows the fallen light stand lies on the left side of the office, which is not visible from White's (or April's) hotel window. Furthermore, E6 establishes that the glass light stand was only purchased on September 4th — the day before the murder:

$$\varphi(S10) \wedge \varphi(E6) = \text{in\_office\_before\_sep5(light\_stand)} \wedge \text{purchase\_date(light\_stand,sep4)} \models \bot$$

This means White could not have seen the light stand through the window during a prior visit to place a wiretap, since it did not exist in the office at that time. The contradiction reveals White's claim to have entered the office "the week before the murder" as fabricated, and implicates him as the true murderer.

E. Full Testimony Graph

Figure 3 presents the complete testimony of graph G_T for the case *Turnabout Sisters*. The graph incorporates all statement nodes from both April May's and Redd White's testimonies, all relevant evidence nodes, and the complete set of contradiction, dependencies, and revision edges identified through propositional analysis as the game progresses.

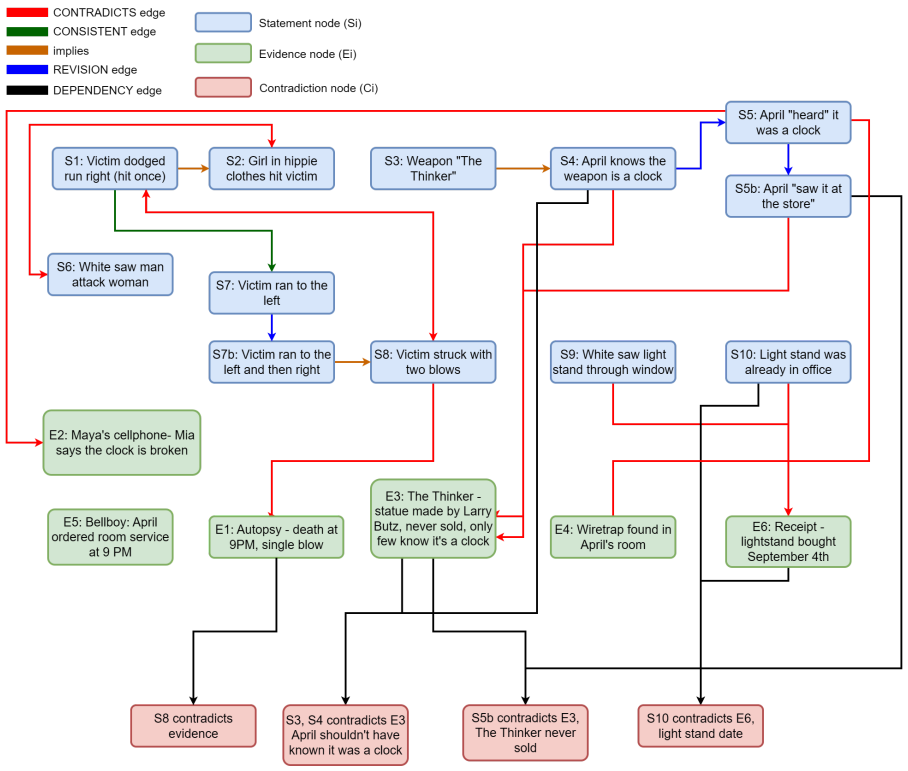

Fig. 3: Full testimony graph G_T for Turnabout Sisters. Red edges = CONTRADICTS, green = CONSISTENT, others= DEPENDS ON / REVISION. Contradiction nodes C1–C4 represent the four objection points available in the game.

V. Python Implementation

A python implementation of the framework was developed using NetworkX. Nodes are initialized with type attributes ("statement" or "evidence") and edges are labeled with their semantic types. As a whole, the program can be separated into different sections:

1. Data Structures

```
class Node:
    """represents a vertex v ∈ V (Statement or
Evidence) with valuation φ(v)"""
    def __init__(self, node_id, node_type, label,
props):
        self.node_id = node_id
 # e.g., "S1", "E1"
        self.node_type = node_type
# "statement" | "evidence"
        self.label = label
# Unique descriptor string
        self.props = props
# Dictionary: {proposition_name: bool}

class Testimony:
        """encapsulates the complete bipartite
evaluation environment"""
         def __init__(self, name, statements,
evidence, dep_edges=None):
        self.name = name
        self.statements = statements
# List of Node objects
        self.evidence = evidence
# List of Node objects
        self.dep_edges = dep_edges or []
# Manually defined directed arcs
```

To model a courtroom cross-examination, we represent statements and physical evidence as distinct nodes within a network. Each node contains a dictionary of propositions (*props*), which act as specific factual claims. These claims are assigned a boolean value (True or False) depending on what that specific statement or piece of evidence asserts. Finally, the *Testimony* class bundles these nodes together and tracks structural relationships, such as when a witness panics and revises a statement during testimony

2. Contradiction Engine

```
def is_contradictory(node_a, node_b):
    """checks if φ(node_a) ∧ φ(node_b) ⊨ ⊥ over
shared propositions"""
     # find intersection of atomic propositions:
P_a ∩ P_b
      shared_props = set(node_a.props.keys()) &
set(node_b.props.keys())
    conflicts = []

    for prop in shared_props:
        # if valuations diverge, a contradiction
is proven
                    if node_a.props[prop] !=
node_b.props[prop]:
            conflicts.append(prop)

    return len(conflicts) > 0, conflicts
```

Using the law of non-contradiction, the core engine simulates the objections by identifying when a witness's claim directly clashes with reality. Algorithmically, the function looks for the intersection of topics discussed by both a statement and a piece of evidence. If they share a common topic but assign them opposite truth values (for example, a witness claims a clock was working, but the evidence proves it was broken), the engine flags a logical contradiction (Statement $\wedge$ Evidence $\models \bot$). This disagreement forms the basis for breaking the witness's testimony.

3. Automating Contradiction Discovery through Graph Compilation

```
import networkx as nx
def build_testimony_graph(testimony):
    """Generates the directed framework graph G_T
= (V, E)"""
    G = nx.DiGraph()
    # fill graph vertices
        for node in testimony.statements +
testimony.evidence:
                        G.add_node(node.node_id,
type=node.node_type, props=node.props)

    # make automated semantic contradiction edges
    for s in testimony.statements:
        for e in testimony.evidence:
                has_contradiction, conflicts =
is_contradictory(s, e)
            if has_contradiction:
                G.add_edge(s.node_id, e.node_id,
type="CONTRADICTS", keys=conflicts)

    # append manual discourse/structural edges
        for (src, dst, relation_label) in
testimony.dep_edges:
        G.add_edge(src, dst, type=relation_label)

    return G
```

The graph builder automates the process of mapping out a trial's hidden logic. The algorithm takes full view of the case by pairing every single witness statement against every available piece of evidence. Whenever the contradiction engine spots a factual mismatch between a pair, the builder automatically draws a directed *contradicts* line between them. This systematically converts a courtroom transcript into a visual network map of lies, tracking exactly which piece of evidence can tear down which statement.

VI. Discussion

The results confirm that Ace Attorney's cross-examination mechanics are formally grounded in propositional logic and directed graph theory. Each "OBJECTION!" moment corresponds precisely to a contradiction edge in G_T — a logically unsatisfiable statement-evidence pair — and the revision structure of testimony maps naturally onto labeled directed paths.

The framework also suggests a graph-theoretic measure of cross-examination difficulty. Testimonies where multiple dependency edges connect statements to the key contradiction node require the player to reason through longer chains of implication before identifying the correct evidence to present. This corresponds to longer directed paths in G_T between the contradiction edge and its logically prerequisite statements, providing a testable, formal notion of puzzle difficulty.

A current limitation of the framework is its reliance on binary propositional logic. Some Ace Attorney contradictions involve temporal or positional reasoning that exceeds the expressiveness of propositional logic — for instance, determining which direction is "left" or "right" relative to a witness's viewpoint requires spatial reasoning more naturally captured by first-order or modal logic. Future work may extend the encoding function φ to richer logical languages.

VII. Conclusion

This paper demonstrates that the deduction mechanics of visual novels such as *Phoenix Wright: Ace Attorney* map cleanly to the propositional logic and graph-theoretic traversal algorithms. By translating interactive trial components into explicit mathematical symbols, we are able to show that complex reasoning games can be evaluated deterministically. In this game, testimony statements and evidences that are identified as logically unsatisfiable pairs are represented as edges in the testimony graph shown in *Figure 3*.

Applied to *Turnabout Sisters*, the framework correctly identifies all four exploitable contradictions across the testimonies of April May and Redd White, matching the game's designed objection points exactly. The Python implementation validates the framework computationally, and the resulting testimony graph provides a complete structural representation of the cross-examination's logical architecture.

By formalizing the courtroom deduction mechanics of *Ace Attorney* through propositional logic and directed graph theory, this work illustrates how discrete mathematics can effectively model complex reasoning processes in interactive games.

Links

http://tiny.cc/videomakalahmatdis
https://github.com/eronessa/matdis-ace-attorney.git

https://docs.google.com/document/d/1fTlP6XWeJb05GYN02HD6ekSwUervSahQrqPg-ezR7s0/edit?usp=sharing

References

[1] Capcom Co., Ltd., Phoenix Wright: Ace Attorney, Nintendo DS, 2001 (JP), 2005 (EN). [Online]. Available: https://www.capcom.com/aceattorney/. [Accessed: June 2026]

[2] StrategyWiki contributors, "Phoenix Wright: Ace Attorney/Episode 2: Turnabout Sisters/Day 1 - Investigation," *StrategyWiki*. [Online]. Available: https://strategywiki.org/wiki/Phoenix_Wright:_Ace_Attorney/Episode_2:_Turnabout_Sisters/Day_1_-_Investigation. [Accessed: Jun. 19, 2026]

[3] "Turnabout Sisters - Transcript (Parts 1–4)," Ace Attorney Wiki, Fandom. [Online]. Available:

https://aceattorney.fandom.com/wiki/Turnabout_Sisters_-_Transcript_-_Part_1. [Accessed: June 2026].

[4] K. H. Rosen, Discrete Mathematics and Its Applications, 8th ed. New York: McGraw-Hill, 2018.

[5] R. Diestel, Graph Theory, 5th ed. Berlin: Springer, 2017. [Online]. Available: https://diestel-graph-theory.com/. [Accessed: June 2026].

[6] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," in Proc. 7th Python in Science Conf. (SciPy), 2008, pp. 11–15. [Online]. Available: https://networkx.org/. [Accessed: June 2026].

[7] M. Huth and M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems, 2nd ed. Cambridge: Cambridge University Press, 2004.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Sherin Felicia Danessa 13525089