The Verification of Decision Tree Model for Coreference Resolution Using Marked Transition Systems, Petri Nets and Büchi Automata

Sergiy Pogorilyy[1], Maksym Slynko, Pavlo Biletskyi[1]

[1] *Taras Shevchenko National University of Kyiv, 60 Volodymyrska Street, Kyiv, 01033, Ukraine*

Abstract

This paper addresses the problem of coreference resolution, which involves identifying words or phrases in a text that refer to the same real or imaginary entity. The solution for this task is developed for Ukrainian-language texts using decision trees, which autonomously structure themselves based on training data. Decision trees, unlike other machine learning algorithms such as neural networks, allow for the analysis of their internal structure through graphical representation, significantly easing the formal verification of their properties. Vector representations of words (such as ELMo) and other linguistic features are used to create decision trees. These trees are employed for the binary classification of input pairs potentially referring to the same coreferent objects. Based on the binary classifier, coreferent objects are grouped into clusters, followed by an evaluation of clustering accuracy using specialized metrics.

To guarantee the reliability of large, complex software systems, formal verification methods are applied. A formal model of the coreference resolution system is constructed using marked transition systems. This model describes the system with a set of discrete states and transitions between them under certain conditions. The properties of the system are formalized and verified using network models, automata models and linear-temporal logic, ensuring error-free execution on infinite state sequences. The work explores the use of Petri nets for analyzing the correctness of the system model. The synchronous product of transition systems is verified for liveliness, boundedness, deadlocks, and traps, ensuring that the model operates correctly without redundancy. Büchi automata are created to accept words confirming the properties, with examples and counterexamples found during the analysis.

The proposed method serves as a foundation for creating automated analyzers for coreference resolution applications based on decision trees, demonstrating high efficiency and accuracy. The approach allows for the formal verification of system properties on potentially infinite state sequences, ensuring the reliability and correctness of the coreference resolution system throughout its runtime.

Keywords

artificial intelligence, natural language processing, coreference resolution, decision trees, transition systems, Büchi automata, Petri nets, formal verification

1. Introduction

Coreference Resolution is a task in Natural Language Processing (NLP) that involves finding all the linguistic objects in a text (such as nouns, pronouns, and noun phrases) that refer to the same real or imaginary entity. The result of solving this task is establishing correspondences between text objects that indicate the same entity; such correspondence can be established for a pair of objects or their cluster.

Examples of coreferent objects [1] are provided below; the referent (noun) is highlighted in bold, and pronouns are underlined.

- Simple anaphora (the noun precedes the pronoun in the text): "He crossed the **mountain**. It was high."
- Simple cataphora (the noun is mentioned after the pronoun): "She walked onto the road leading to the right. **Maria** was in a good mood today."

14th International Scientific and Practical Conference from Programming UkrPROG'2024, May 14-15, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

sdp77@i.ua (S. Pogorilyy); maxim.slinko@gmail.com (M. Slynko); 1234bpv@i.ua (P. Biletskyi)

0000-0002-6497-5056 (S. Pogorilyy); 0000-0001-9667-8729 (M. Slynko); 0000-0001-5425-3706 (P. Biletskyi)

- Compound referent: "**Ivan, Mykhailo, and Ostap** — all worked underground."

Automated methods for solving the coreference resolution task include algorithms based on strict rules formulated by qualified linguists and artificial intelligence (AI) methods. AI methods include neural networks, language models, and decision trees. Decision trees, unlike other AI methods, have a structure that allows relatively easy analysis of their internal logic and corrections to their structure to change the classification logic.

In the process of creating large complex software systems, errors often arise. Therefore, ensuring the reliability of such systems is relevant. Testing is a common method to find errors in software products during the development. However, testing allows for finding errors in the program but cannot guarantee their absence. For a more detailed analysis of the application reliability, formal verification methods are used.

To conduct formal verification, it is necessary to build a formal model of the system. The system can be formally represented using various methods. We propose using the marked transition systems [2] apparatus, which allows describing the system with a set of discrete states between which transitions occur under certain conditions, indicating the operations that system performs while transitioning from one state to another.

Once the formal model is built, the next step of the analysis is to define the properties of the system to be analyzed and to express them in a formal way - formalize them. After obtaining the system model and the properties for its verification, formal methods are used to prove the verification or falsification of the properties. For this, Petri nets and Büchi automata [3] were used.

The paper considers the problem of coreference resolution in Ukrainian-language texts using decision trees. The application of transition systems is proposed to build a high-level specification model for coreference resolution. Formalization is carried out, and network/automata models and linear-temporal logic are used to verify a set of properties of the obtained specification. Büchi automata are created to accept words confirming the properties, and examples and counterexamples of the analyzed properties are found.

2. Using Decision Trees for Coreference Resolution

- A decision tree is a hierarchical structure consisting of nodes (the root—the initial node, internal nodes, and leaf nodes). Each non-leaf node of such a tree refers to two subtrees (or child nodes). Decision tree structure can be generated automatically based on a training dataset.
- The dataset should contain elements and class labels for them. Each element consists of features that can take real or Boolean values (i.e., values that support comparison operations necessary for the tree's functioning). In the work [1], the dataset for coreference resolution consists of elements describing pairs of potentially coreferent objects and a label indicating whether these objects are coreferent. Each element contains features of the pair, such as: matching number, gender, part of speech of the first and second object, lemmatized versions of the objects, the number of words between the objects, the cosine similarity measure of the vectors of the considered objects, and others. All these features are obtained automatically using the UDpipe library [4], the ELMo model for creating word vector representations [5], and own algorithms. The dataset (2500 texts containing 2.4 million examples) was divided into training (1500) and test (1000) samples.
- The decision tree for coreference resolution in the work [1], created using the scikit-learn library [6], is formed by selecting a specific feature at each step in the process of deepening. This feature is chosen in such a way as to best separate the set considered in the specific subtree into classes. The Gini impurity coefficient [7] is used to select this feature, which allows evaluating the probability of incorrect classification of a randomly chosen object from the subgroup.

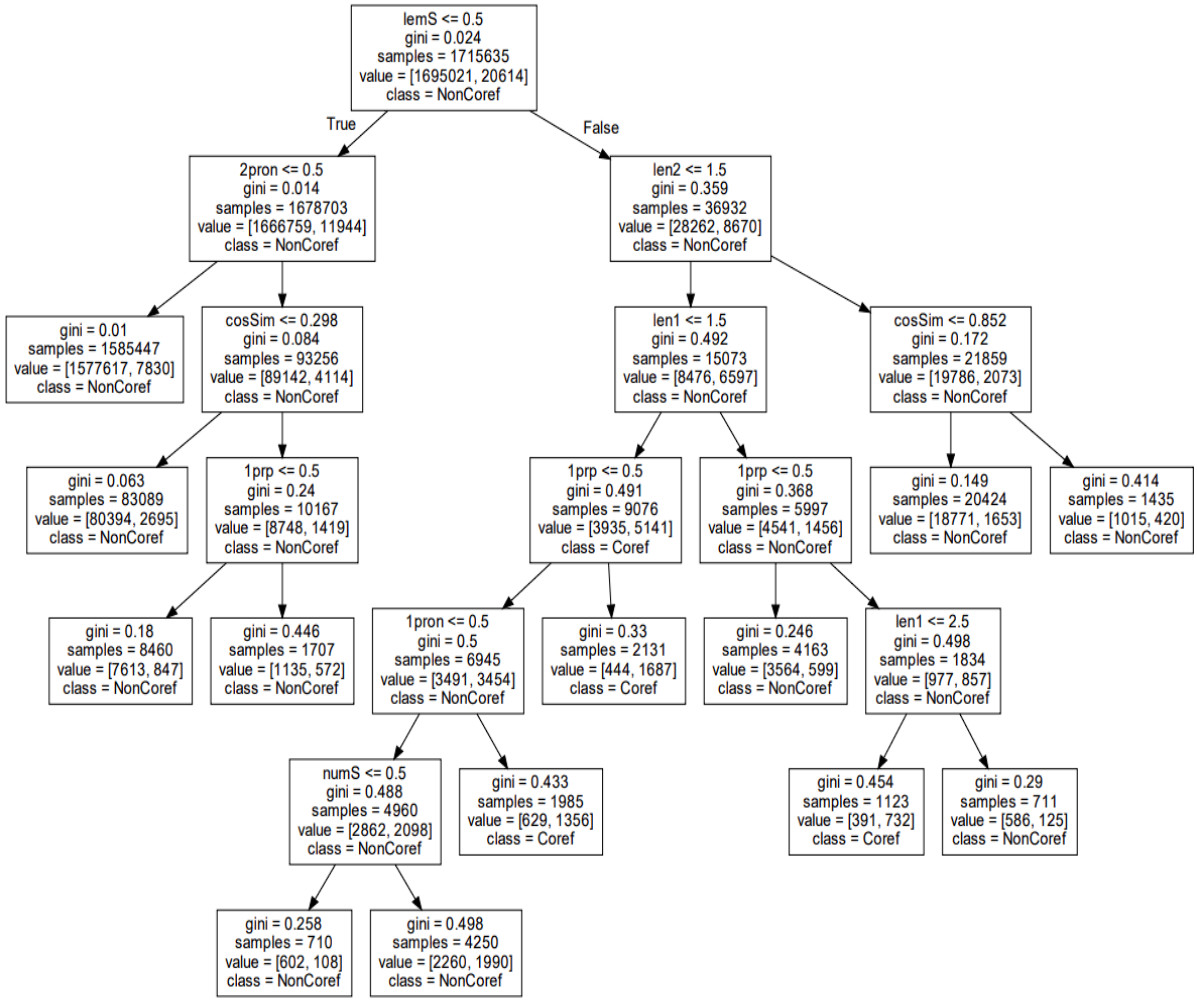

Figure 1: Decision tree (limited size) [1]

As with other AI methods, decision trees are prone to overfitting, which is the excessive adaptation of the tree structure to the dataset used for its creation. In this case, the algorithm's accuracy on data not used for training decreases. To overcome this phenomenon, the parameter `min_impurity_decrease` is used in the work [1], which allows limiting the tree's depth if further splitting into subsets decreases the Gini coefficient by less than the threshold value. The size of the decision tree from figure 1 is limited by `min_impurity_decrease` set to 0.00005 for illustration purposes (in the final decision tree it was 0.000003).

The created decision tree allows classifying input objects by transitioning into subtrees starting from the root following the rule specified in the current node. Since each input object describes a pair of potentially coreferent objects, the decision tree performs a binary classification task.

Coreferent links can exist between more than two objects in the text, for example, including three, four, or more objects. Therefore, to obtain results, coreferent objects are grouped into clusters. In the work [1], initially, all potentially coreferent objects are considered clusters. Their merging occurs if at least one pair of potentially coreferent objects from the first and second clusters is recognized as coreferent.

The quality of clustering is evaluated by comparing the obtained clusters with the original ones using special metrics. Such an assessment is performed on the test sample. The results obtained in the work [1] show high algorithm efficiency, close to the results of the BiLSTM neural network-based model [8].

The parts of the decision tree used for further algorithm analysis are shown at figures 2 and 3.

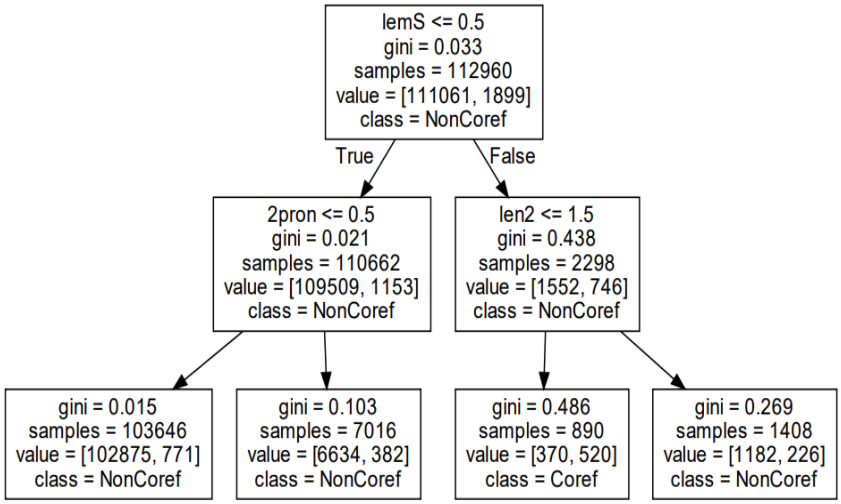

Figure 2: Decision tree subtree (with root node) [1]

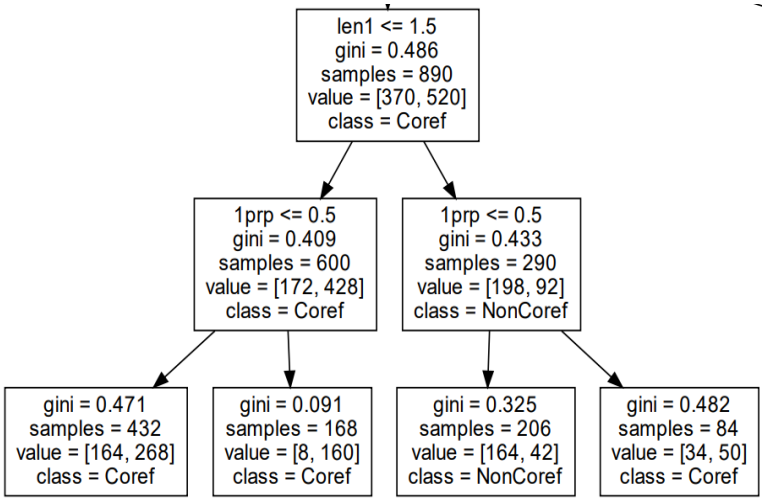

Figure 3: Decision tree subtree (continuation of tree from figure 2) [1]

3. Formal Verification of Algorithms Using Decision Trees

As shown in the review article [9], existing studies use formal verification for machine learning models using Satisfiability Modulo Theories (SMT) and Linear Programming (LP). In this paper we propose an approach to verification using automata models and linear-temporal logic, which allows exploring the temporal characteristics of the model on potentially infinite state sequences. Typically, a mathematical model of a discrete system represents a graph where vertices correspond to states (or state classes) the system can be in at different times, and edges represent transitions between states that may have labels indicating actions or events performed by the system.

The system's functioning is represented by sequences of transitions from one state to another. If an edge has a label, this label represents the system's action performed when transitioning from the state at the beginning of the edge to the state at its end. The work uses marked transition systems (marked TS or MTS) as a discrete model of general-type computations [2].

4. Creation of MTS Models at a High Level of Abstraction

From the modeling perspective, an application using decision trees for coreference resolution can be represented as the interaction of such systems:

- TS 1 or "control" system: responsible for interaction with external resources;
- TS 2 or "core": system representing the traversal of the decision tree.

The control system is modeled by TS:

$$M = (\{v_0, v_1, v_2\}, \{a_1, a_2, a_3\}, \alpha, \beta, v_0), \quad (1)$$

where v_0 is the system in the availability state; v_1 is the state where the system processes input data; v_2 is the state where the system outputs the result. Transitions are interpreted as follows: a_1 is receiving a new set of input data; a_2 is forming the classification result; a_3 is transitioning to the availability state.

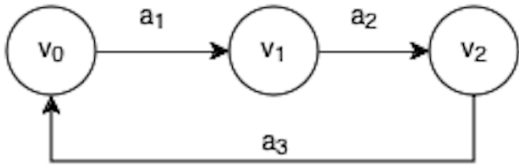

Figure 4: TS representation of the “control” system.

For clarity, let's use a subtree of the decision tree obtained in [1] as the “core”. Note that during the model creation process, two surrogate states were added to the subtree: the initial s_0 and the final s_6; as well as the transition t_{11} between them. This is necessary to represent the decision tree as a continuously functioning system, allowing the use of temporal logic for further analysis. The final model is defined as:

$$S_K = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}, \quad (2)$$

$$T_K = \{t_1, t_2, \ldots, t_{11}\}, \quad (3)$$

$$K = (S_K, T_K, \alpha, \beta, s_0), \quad (4)$$

The set of propositional formulas associated with states and the labeling function for each state are shown in the figure 5.

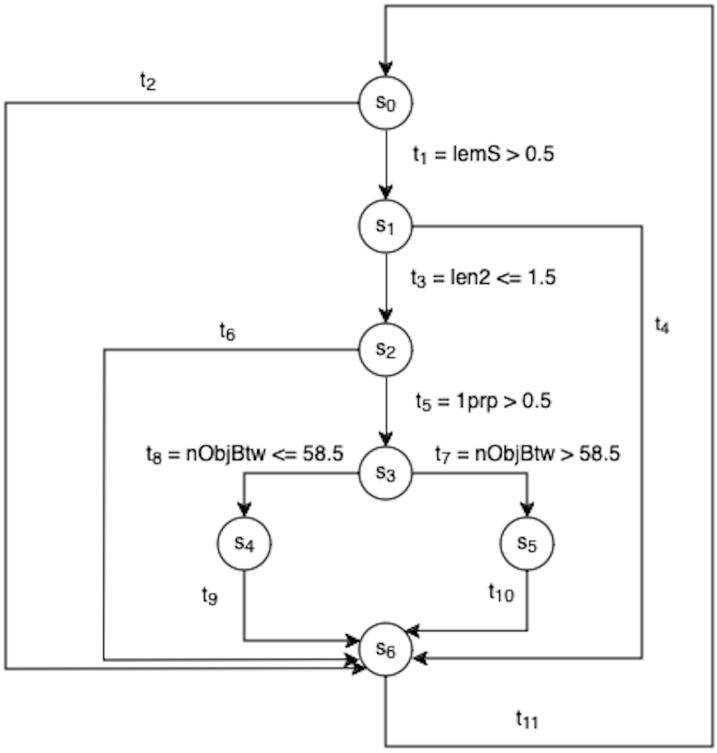

Figure 5: TS representation of the "core" system.

It is important to note that in the subtree, which was selected for modeling, each subset of states {s2, s3, s4} asserts coreference of the input data. Accordingly, each of the states {s0, s1, s5} asserts the absence of coreference, and the state s6 preserves the coreference class determined earlier.

For the above TS, we construct a synchronized parallel composition (synchronous product) with global transitions modeling the application's operation as a whole. The set of synchronization constraints includes the following elements (ε - identical action indicating no transition in TS):

$$T = \{(a_1, t_1), (a_1, t_2), (\varepsilon, t_3), (\varepsilon, t_4), (\varepsilon, t_5), (\varepsilon, t_6), (\varepsilon, t_7), (\varepsilon, t_8), (\varepsilon, t_9), (\varepsilon, t_{10}), (a_2, \varepsilon), (a_3, t_{11})\}. \quad (5)$$

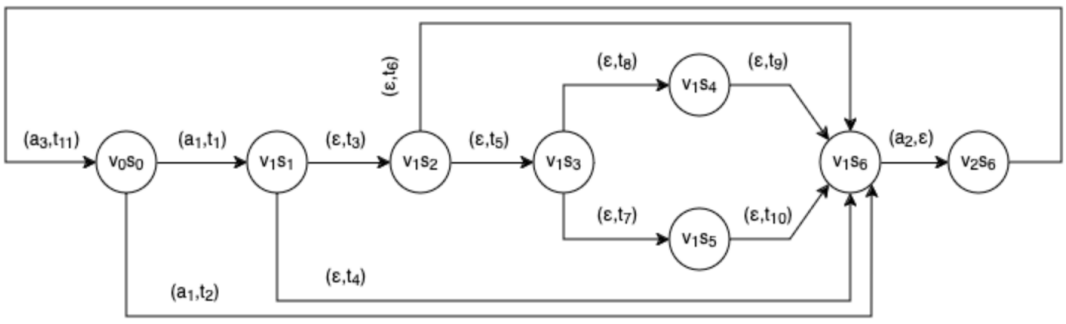

Figure 6: synchronized composition of TS 1 and 2.

5. Using Petri Nets for Verifying Model Correctness

Once the synchronous product of the above TS with global transition constraints is defined, it becomes possible to proceed with its verification. The main models of such a process are automata and network models. The work considers using Petri nets (PN) for which there is a wide range of analysis methods. In [10], it is described that the semantics of the TS product and the PN semantics modeling it are consistent in the sense that the sequence of global transitions t1,...,tk represents the

global history of the TS product if and only if it is an acceptable sequence of transitions in the PN. Accordingly, the elements of the set T become PN transitions, and the global states of TS (the set of states of each TS participating in the synchronous product before or after the global transition) become places in the obtained network. According to the set of synchronization constraints T, we build a Petri net modeling the joint operation of the core model and the control system:

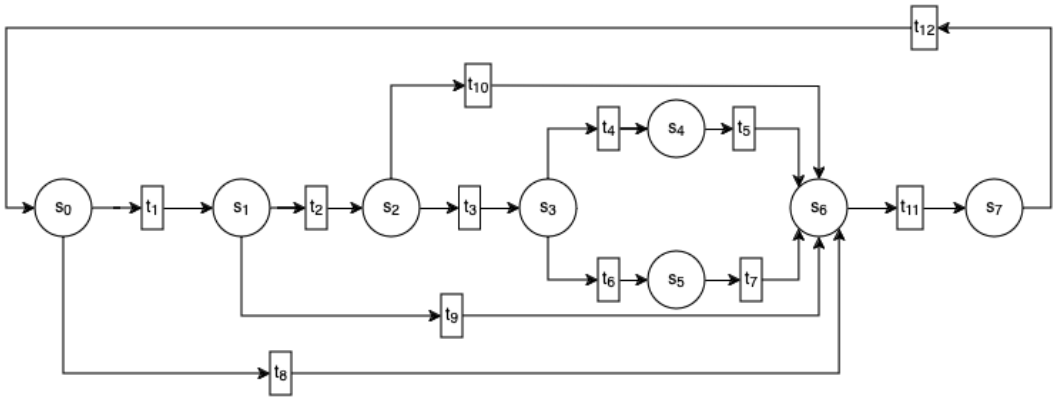

Figure 7: PN that represents synchronized composition of TS 1 and 2.

We will verify the liveliness of such a system. The "liveliness" property means that all transitions in the obtained model will participate in its operation process [2]. If certain transitions in the PN never fire, it means that the system's design is incorrect or redundant. We will examine the PN for liveliness by solving the state equation of the form $A\cdot x = 0$ where A is the incidence matrix of the PN.

Table 1

Incidence matrix *A*

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
S_0	-1	0	0	0	0	0	0	-1	0	0	0	1
S_1	1	-1	0	0	0	0	0	0	-1	0	0	0
S_2	0	1	-1	0	0	0	0	0	0	-1	0	0
S_3	0	0	1	-1	0	-1	0	0	0	0	0	0
S_4	0	0	0	1	-1	0	0	0	0	0	0	0
S_5	0	0	0	0	0	1	-1	0	0	0	0	0
S_6	0	0	0	0	1	0	1	1	1	1	-1	0
S_7	0	0	0	0	0	0	0	0	0	0	1	-1

Solutions to the state equations with the incidence matrix from Table 1 are listed in Table 2. The TSS algorithm [11] was used to solve the state equation.

Table 2

State equation solutions

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}

1	1	1	1	1	0	0	0	0	0	1	1
1	1	1	0	0	1	1	0	0	0	1	1
1	1	0	0	0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1

The incidence matrix and the state equation solutions indicate that all transitions in the PN are covered by positive invariants. Additionally, only transitions corresponding to a single decision tree branch fire at any given time. Analysis of the boundedness PN property is performed by solving the system of equations of the form $A^T \cdot x = 0$ where A^T is the transposed matrix of A. The set of solutions includes vector $x = \{1,1,1,1,1,1,1\}$, that covers all places of the PN with positive values. Therefore, the PN under analysis is bounded, and there are no unreachable places.

Let's consider the Petri net for the presence of deadlocks and traps. Semantically, a deadlock state is a reachable marking of the net from which no transition is possible [12]. A Petri net is structurally live if and only if each of its deadlocks has a trap. The logical dependency system for deadlock detection in Petri nets in Figure 7 is presented as follows:

$$s_0 \to s_7; s_1 \to s_0; s_2 \to s_1; s_4 \to s_3; s_5 \to s_3;$$
$$s_6 \to s_0 \vee s_1 \vee s_2 \vee s_4 \vee s_5;\ s_7 \to s_6. \quad (6)$$

The system of linear homogeneous Diophantine inequalities (SLHDI), corresponding to this system of logical dependencies, can be represented in the form of the Table 3:

Table 3
SLHDI for deadlock detection

-1	0	0	0	0	0	0	1	≥ 0
1	-1	0	0	0	0	0	0	≥ 0
0	1	-1	0	0	0	0	0	≥ 0
0	0	1	-1	0	0	0	0	≥ 0
0	0	0	1	-1	0	0	0	≥ 0
0	0	0	0	1	-1	0	0	≥ 0
1	1	1	0	1	1	-1	1	≥ 0
0	0	0	0	0	0	-1	1	≥ 0

The set of deadlocks in the Petri net under consideration contains the following elements, among which the first deadlock is a combination of basic others:

$$D_1 = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\},$$
$$D_2 = \{s_0, s_6, s_7\},$$
$$D_3 = \{s_1, s_6, s_7\},$$
$$D_4 = \{s_2, s_6, s_7\}, \quad (7)$$
$$D_5 = \{s_3, s_6, s_7\},$$
$$D_6 = \{s_4, s_6, s_7\},$$
$$D_7 = \{s_5, s_6, s_7\}.$$

Let's proceed to investigate the traps of the Petri net depicted in Figure 7. The logical dependency system for trap detection looks like:

$$s_0 \to s_1 \wedge s_6;\ s_1 \to s_2 \wedge s_6;\ s_2 \to s_3 \wedge s_6;$$
$$s_4 \to s_6;\ s_5 \to s_6;\ s_6 \to s_7;\ s_7 \to s_0. \quad (8)$$

Table 4

SLHDI for trap detection

-1	1	0	0	0	0	1	0	≥ 0
0	-1	1	0	0	0	1	0	≥ 0
0	0	-1	1	0	0	1	0	≥ 0
0	0	0	0	-1	0	1	0	≥ 0
0	0	0	0	0	-1	0	0	≥ 0
0	0	0	0	0	0	-1	1	≥ 0
1	0	0	0	0	0	0	-1	≥ 0

The following set of traps is obtained after solving the SLHDI in Table 4:

$$\begin{gathered} Tr_1 = \{s_0, s_1, s_2, s_6, s_7\}, \\ Tr_2 = \{s_1, s_2, s_6, s_7\}, \\ Tr_3 = \{s_2, s_3, s_6, s_7\}, \\ Tr_4 = \{s_4, s_6, s_7\}, \\ Tr_5 = \{s_5, s_6, s_7\}. \end{gathered} \quad (9)$$

Since a given Petri net is free-choice, the following statement holds for such nets: a free-choice Petri net is live if and only if every deadlock in such a net includes the trap marked by the initial marking. As evident from the given Petri net, each of its basic deadlock (7) includes at least one of the traps (9), thus the Petri net depicted on Fig 7 is live.

6. Verification of Model Properties Using Büchi Automata

We propose using the following algorithm to verify a linear-temporal formula *P* representing a property that determines the system's semantic correctness:

1. Create a Büchi automaton that accepts words confirming *P*.
2. Construct the product of the automaton and the TS modeling the original system.
3. Find the intersection of paths generated by the transition system and the paths accepted by the automaton. Further analysis of the reachable states of the intersection allows finding both examples and counterexamples of the formula *P* [3].

Let's consider the use of the algorithm with an example: suppose there is a hypothesis that, during the analysis of coreference between two objects, if the length of the second object is small, then the objects are coreferent. Such a property can be represented by a following formula in linear temporal logic:

$$P_1 = (len2 \leq 1.5) \to G(F(coref)), \quad (10)$$

In other words, if the condition holds true, the system will eventually transition to a state that establishes the coreference class and remain in that state.

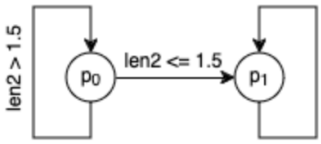

Figure 8. Representation of a Büchi automaton corresponding to the linear temporal formula P1.

The Büchi automaton that accepts words corresponding to formula P1 has two states: initial state p0 and final state p1. Once it transitions to p1, the automaton remains in that state regardless of the input words.

Let's construct the intersection of the Büchi automaton (Figure 8) and TS, which models the synchronous product of TS 1 and 2 (Figure 6). A simplified visual representation of the obtained intersection is shown in Figure 9 (for clarity, unreachable states are hidden).

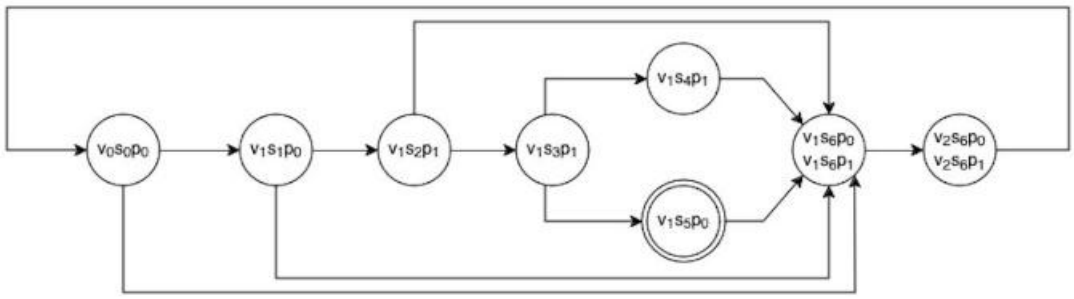

Figure 9. Intersection of the Büchi automaton and the TSю

The obtained intersection allows for further analysis of feasible paths, cycles and traces. In particular, we observe that there exists a path where a transition occurs from state p1 to state p0 (the final state of such transition is highlighted by two concentric circles on Fig 9), indicating a counterexample to property P. Indeed, having just the information about the length of one of the objects is insufficient to assert their coreference: traversal of the tree with the specified constraint can end in either state s4, which asserts coreference, or state s5, which asserts its absence.

If we consider another linear temporal logic formula, for example,

$$P_2 = (lemS \wedge 1prp \wedge len2 \leq 1.5 \wedge len1 \leq 1.5 \wedge nObjBtw \leq 58.5) \rightarrow G(F(coref)), \quad (7)$$

Then the analysis of the Büchi automaton product and the synchronous product of TS will demonstrate the absence of a path-cycle that is accessible from the initial state and includes a state from the set of unreachable states. Thus, formula P2 is true, indicating that the property it represents holds true universally.

7. Conclusions

The study presented in this paper explores the use of decision trees for coreference resolution in Ukrainian-language texts. Decision trees are demonstrated to be an effective method for this task, providing clear and interpretable structures that facilitate the analysis and formal verification of their properties. The application of decision trees allows for binary classification of potentially coreferent object pairs, leading to the formation of coreferent clusters with high accuracy, as evidenced by the experimental results.

A significant contribution of this work is the integration of formal verification methods to ensure the reliability and correctness of the coreference resolution system. By constructing a formal model using marked transition systems, we enable the detailed examination of the system's behavior across potentially infinite state sequences. This approach guarantees the absence of errors during the entire runtime of the system, a feat not achievable with other formalization tools such as SMT, which only address verification at specific static moments.

The use of Petri nets further strengthens the analysis by providing a robust framework for examining the correctness of the model. The synchronous product of the transition systems is

analyzed for liveliness, boundedness, and the presence of deadlocks and traps, ensuring the system operates correctly and efficiently without redundancy.

Additionally, the paper proposes using Büchi automata and linear-temporal logic to verify properties of the AI classifiers. This approach allows for the formal verification of semantic correctness by intersecting the paths generated by the transition system with those accepted by the automaton. The approach is illustrated with practical examples, demonstrating both the identification of counterexamples and the verification of property compliance.

Overall, the research lays a solid foundation for creating automated analyzers for coreference resolution applications based on decision trees. The methodologies developed in this work ensure high efficiency, accuracy, and reliability of the system, providing valuable insights for future advancements in the field of natural language processing and formal verification.

References

[1] S. Pogorilyy, P. Biletskyi. Coreference resolution algorithm for Ukrainian-language texts using decision trees, Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG, 2022, pp. 81-90.

[2] Y. Boyko, S. Kryvyi, S. Pogorilyy et al. Methods and innovative approaches to designing, managing, and deploying high-performant IT infrastructures. PPC “Kyiv University”, 2016, p. 447 [in Ukrainian]

[3] S. Kryvyi, S. Pogorilyy, M. Slynko, A. Kramov. Method of semantic application verification in GPGPU technology. System Research & Information Technologies № 3, 2020, pp. 7-22.

[4] UDpipe library. Accessed: 06.04.2024. https://lindat.mff.cuni.cz/services/udpipe/

[5] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer.. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, № 1, 2018, pp. 2227–2237.

[6] Scikit learn for decision trees. Accessed: 06.04.2024. https://scikit-learn.org/stable/modules/tree.html

[7] S. Tangirala. Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm, International Journal of Advanced Computer Science and Applications, 2020, pp. 612-619.

[8] S. Telenyk, S. Pogorilyy, A. Kramov. The complex method of coreferent clusters detection based on a BiLSTM neural network, Knowledge Based Systems, 2021, pp. 205-210.

[9] M. Krichen et al. Are Formal Methods Applicable To Machine Learning And Artificial Intelligence? In Proceedings of 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), 2022, pp. 48-53.

[10] A. Arnold. Finite Transition Systems: Semantics of Communicating Systems. - Paris: Prentice Hall, 1994, p. 177.

[11] S. Kryvyi. Linear Diophantine limits and their application. Chernivtsi: “Bukrek” Publishing House, 2015, ISBN 978-966-399-650-9 [in Ukrainian]

[12] S. Kryvyi et al. Design of Grid Structures on the Basis of Transition Systems with the Substantiation of the Correctness of Their Operation. Cybernetics and Systems Analysis, Volume 53, Issue 1, Springer Science+Business Media New York 2017, January 2017, pp 105–114.