Merging Two Grammar Worlds: Exploring the Relationship between Universal Dependencies and Signal Temporal Logic

Anonymous EMNLP submission

Abstract

Translating natural language requirements into Signal Temporal Logic (STL) is essential for safety-critical systems but requires mathematical expertise. We propose a translational grammar mapping Universal Dependencies (UD) structures to STL operators through twelve theoretically-motivated patterns, evaluated on 7,003 expert-annotated sentence-STL pairs. Our patterns achieve 97.8% coverage in detecting temporal expressions, revealing systematic correspondences between syntactic structures and logical operators. Analysis uncovers rich syntactic variation (e.g., 70+ variants for conditionals) and probabilistic pattern-operator relationships, demonstrating that while our patterns reliably identify temporal expressions, their mapping to STL operators exhibits statistical tendencies rather than deterministic rules. These findings provide foundational insights for developing hybrid approaches that model uncertainty explicitly, combining syntactic patterns with semantic understanding to advance interpretable natural language interfaces for temporal logic specification.

1 Introduction

Formal specifications in temporal logic are essential for verifying safety-critical systems, synthesizing correct-by-construction controllers, and defining precise requirements for autonomous agents. However, translating natural language requirements into Signal Temporal Logic (STL) remains challenging due to expertise required in mathematical logic. Consider a seemingly simple requirement: *"For each moment within the first 2 to 46 time units, the signal must consistently stay above 8.9."* Translating this requires understanding that "*for each moment*" maps to a global operator ($\mathcal{G}$), "within the first 2 to 46 time units" defines temporal interval $[2, 46]$, and "consistently stay" reinforces universal quantification, yielding $\mathcal{G}_{[2,46]}(signal \geq 8.9)$. This challenge has motivated emerging research heavily relying on deep learning and generative AI. Researchers train neural networks or fine-tune models like DeepSTL (He et al., 2022) and NL2TL (Chen et al., 2023), using human feedback and part-of-speech tagging to produce labeled NL-STL datasets for training (Fang et al., 2025; Chen and Manning, 2014). However, these approaches lack generalizability and provide insufficient interpretability in how linguistic structures map to STL syntax.

In this proposed paper, we argue that Universal Dependencies (UD) (De Marneffe et al., 2021; Nivre, 2020) provides cross-linguistically consistent syntactic annotation enabling systematic mapping between natural language and temporal logic operators. Grounded in universal grammar principles (Chomsky, 1965), UD's success across 100+ languages suggests temporal reasoning follows universal syntactic patterns. Specifically, one may ask: *Is there a fundamental logical relationship between UDR components and STL operator composition that transcends individual languages?* This motivates our research objective: **to propose a translational grammar that maps syntactic dependency relationships in Universal Dependencies to Signal Temporal Logic operator composition, leveraging universal grammatical principles to ensure broad applicability across natural language variations**.

Our investigation focuses on identifying and analyzing these patterns, not to propose yet another translation framework, but to enhance our understanding of how natural language grammar relates to STL grammar. Our paper makes two unique contributions:

First, we propose a formal logic for mapping UDR in combination with a dictionary for temporal words into UDR patterns. We then, establish a *new grammar* that defines a formal (probabilistic) relationship between UDRs and STL, also referred to as UDR patterns in this paper.

And second, we empirically examine this translational grammar based on a dataset of 7,003 pairs of natural language and STLs used in prior research. Specifically, we examine twelve core patterns that emerge from analyzing the relationship between UD syntactic structures and STL operators. For instance, we find that `advmod(stay, consistently)` systematically signals universal quantification, `nmod(within, units)` + `nummod(units, 46)` specifies temporal bounds, and `mark(if, _)` + `advmod(then, _)` indicates logical implication. These patterns are not learned from data but derived from linguistic principles, providing an interpretable bridge between natural language and formal logic. The identification of such patterns follows the tradition of linguistic analysis in temporal expression recognition (Verhagen et al., 2010) while extending it to formal logic operators.

This understanding has important implications for future research and the design of learning-based architectures, including in-context learning as well as LLM-finetuning (Chen and Manning, 2014). By understanding these fundamental correspondences, we increase the generalizability of future learning-based approaches (e.g., if trained based on datasets enriched by our logic, and also increase the interpret ability of LLM). Beyond that, it will also allow the successful creation of STL that ensure the safety of AI systems used in robotics and other applications while simultaneously offering important means to verify the safety of a system.

2 Theoretical Foundations

2.1 Signal Temporal Logic

Signal Temporal Logic extends propositional logic with temporal operators over dense-time signals. STL formulas follow the grammar:

$$\phi ::= \pi^{\mu} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \mathcal{F}_{[a,b]}\phi \mid \mathcal{G}_{[a,b]}\phi \mid \phi_1 \mathcal{U}_{[a,b]} \phi_2 \quad (1)$$

where π^{μ} represents atomic predicates ($x \sim \mu$), ϕ denotes STL formulas, and $[a, b]$ specifies time intervals. The temporal operators $\mathcal{F}_{[a,b]}$ (Eventually), $\mathcal{G}_{[a,b]}$ (Always), and $\mathcal{U}_{[a,b]}$ (Until) quantify over time intervals, with Boolean connectives preserving their classical semantics (Maler and Nickovic, 2004). Complete formal definitions are provided in Appendix A.1.

2.2 Universal Dependencies

Universal Dependencies (Nivre et al., 2016) represents syntax as directed graphs $G = (V, E, \ell)$ with word nodes V, dependency edges E, and labels $\ell : E \to R$ from 37 universal relations. Dependency patterns $P = (r, h, d, C)$ consist of relation r, head/dependent constraints h/d, and contextual constraints C, matching edges when all constraints are satisfied. Complete formal definitions are provided in Appendix A.1.

2.3 Translational Grammar: UDR-STL Patterns

We propose twelve core patterns (Table 1) providing comprehensive STL coverage for cyber-physical system specifications. These patterns, selected through theoretical analysis of STL operators (Maler and Nickovic, 2004) and empirical validation showing over 90% coverage (Chen et al., 2023), systematically map Universal Dependencies to STL: advmod relations encode temporal quantification (patterns 4, 5, 8, 9), mark+nummod capture bounded constraints (patterns 3, 6, 7), and lexical patterns identify state transitions (patterns 2, 10, 12) and logical relationships (patterns 1, 8).

Pattern No.	**Pattern Name**	**Pattern**	**Logic Op.**
1	Any Time Global	det("any")+nmod("time")	G
2	Become/Change Rise	compound("become"/ "change")	$\uparrow \phi$
3	Bounded After	mark("after")+nummod	F[k,∞]
4	Always	advmod("always")	G
5	Eventually	advmod("eventually")	F
6	Bounded For	mark("for")+nummod	G[0,k]
7	Bounded Within	mark("within")+nummod	F[0,k]
8	If-Then Implication	advmod("then")+mark("if")	$\rightarrow$
9	Negated Always	advmod("never")	¬G
10	No Longer Fail	advmod("no longer")	$\downarrow \phi$
11	Until	mark("until")+advcl	U
12	When First Rise	mark("when")+advmod("first")	$\uparrow \phi$

Table 1: Pattern definitions and their corresponding logic operators

The patterns exploit systematic correspondences between linguistic quantification and temporal logic. Determiners (*any*, *every*) parallel universal quantification (**G**), while indefinites map to existential quantification (**F**) (Barwise and Cooper, 1981). Temporal adverbs directly lexicalize quantifiers, prepositional phrases with numerals encode metric constraints, and subordinating conjunctions establish temporal ordering (Partee, 1984). The Until operator's dual requirements – eventual satisfaction and continuous maintenance – mirror adverbial clause structures, demonstrating how syntactic subordination encodes semantic scope (Emerson and Halpern, 1986).

Pattern design leverages formal semantic principles: negation duality ($\neg\mathbf{G}_{[a,b]}\phi \equiv \mathbf{F}_{[a,b]}\neg\phi$) motivates Pattern 9's *never* encoding (Horn, 2001);

material conditional correspondence justifies Pattern 8's if-then mapping (Kratzer, 1991); and edge operators ($\uparrow \phi \equiv \neg\phi \mathbf{U} \phi$, $\downarrow \phi \equiv \phi \mathbf{U} \neg\phi$) capture state transitions through inchoative/cessative predicates (Dowty, 1979). This compositional approach aligns with natural language semantics (Montague, 1973), where patterns serve as atomic operations combining per STL formation rules, preserving interpretability essential for verification tasks (Konrad and Cheng, 2005).

3 Methodology

We evaluate the proposed Universal Dependencies to Signal Temporal Logic mapping framework using the `circuit_total_refined` dataset (Chen et al., 2023), containing 7,003 natural language sentences paired with expert-annotated STL formulas. The dataset provides comprehensive coverage of temporal operators (**G**, **F**, **U** and bounded variants) with varying proposition complexity (0-4 per sentence). Implementation details including computational infrastructure, parsing pipeline configuration, and complete algorithmic descriptions are provided in Appendix A.2.

Our approach employs SpaCy 3.8.0 (Honnibal and Montani, 2017) to generate Universal Dependencies parses, from which we extract temporal and logical components using twelve theoretically-motivated patterns (Table 1). The pattern-based extraction framework maps UD structures to STL operators through four components: pattern detection, temporal component extraction, logical operator identification, and atomic proposition extraction (where atomic predicates are assumed to be pre-identified in the dataset). Extracted components are compositionally combined following STL formation rules, with operator precedence and scope determined by syntactic hierarchy.

Evaluation employs multiple metrics including exact match accuracy, similarity scores (0-1), and component-wise assessment of temporal operators, logical connectives, and proposition boundaries. We conduct three primary analyses: (i) Pattern-to-STL mapping analysis examining relationships between UD patterns and STL operators, (ii) Pattern variation analysis quantifying syntactic diversity beyond canonical forms, and (iii) Co-occurrence analysis identifying systematic associations between syntactic structures and logical operators.

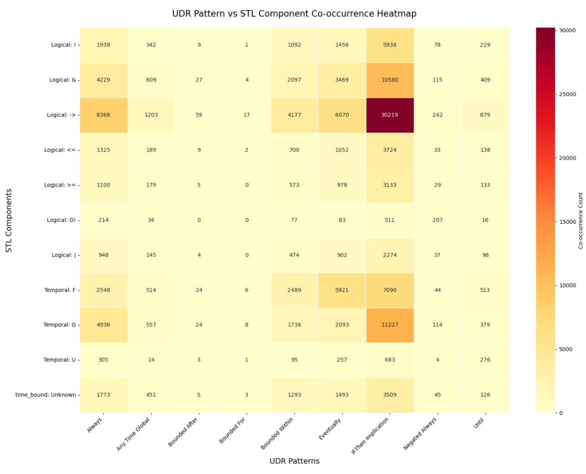

Figure 1: Heatmap of UDR and STL Co-occurences

4 Results

We analyzed 7,003 natural language sentences paired with STL formulas to evaluate our pattern-based extraction approach using multiple statistical techniques.

4.1 Pattern Descriptives

Pattern detection analysis achieved 97.8% coverage (6,850 out of 7,003 sentences), with an average of 4.76 patterns detected per sentence. However, only 9 of the 12 proposed patterns showed actual occurrences in the dataset, with Pattern 8 (If-Then Implication) dominating at 16,098 instances, followed by Pattern 4 (Always) with 6,251 instances and Pattern 5 (Eventually) with 5,353 instances. The complete absence of edge detection patterns (Patterns 2, 10, 12) suggests a fundamental misalignment between the theoretical pattern design and actual linguistic usage in the dataset.

Pattern variation analysis, detailed in Table 3 in the Appendix, revealed extensive syntactic diversity beyond canonical forms. Pattern 8 (If-Then Implication) exhibited the highest variation with 63,078 total occurrences across multiple realizations including "then" (16,098), "it is always" (5,543), "when" (5,229), and numerous other variants. Pattern 5 (Eventually) showed 10,181 occurrences with variants like "at most" (1,840) and adverbial modifications, while Pattern 7 (Bounded Within) had 6,240 instances dominated by "within" (3,000), and Pattern 1 (Any Time Global) showed modest usage (1,783 total).

Co-occurrence analysis in Figure 1 in the Appendix showed the If-Then Implication pattern with particularly high co-occurrence with logical implication operators (30,219 instances), while temporal operators demonstrated broad distribution across patterns. The extensive variation and unexpected mappings indicate that single syntactic patterns correspond to multiple logical interpretations, requiring context-dependent analysis beyond canonical pattern matching.

4.2 UDR-STL Pattern

Pattern	Expected STL	Inst.	Total Var.	Actual Map	Key Finding
1. Any Time Global	$\mathcal{G}$	1,079	1,783	$\wedge$	Unexpected logic
2. Become/Change	$\uparrow \phi$	0	0	–	Absent
3. Bounded After	$\mathcal{F}_{[k,\infty]}$	32	32	$\mathcal{F}_{[k,\infty]}$	Aligned
4. Always	$\mathcal{G}$	6,251	10,893	$\mathcal{G}$	1,139 unique STLs
5. Eventually	$\mathcal{F}$	5,353	10,181	$\mathcal{F}$	927 mappings
6. Bounded For	$\mathcal{G}_{[0,k]}$	10	10	$\mathcal{G}_{[0,k]}$	Minimal
7. Bounded Within	$\mathcal{F}_{[0,k]}$	3,568	6,240	$\mathcal{F}$	Lost bounds
8. If-Then Impl.	$\rightarrow$	16,098	63,078	$\rightarrow$	70+ variants
9. Negated Always	$\neg\mathcal{G}$	206	206	$\neg\mathcal{G}$	Limited
10. No Longer Fall	$\downarrow \phi$	0	0	–	Absent
11. Until	$\mathcal{U}$	712	782	$\mathcal{U}$	Moderate
12. When First Rise	$\uparrow \phi$	0	0	–	Absent

Table 2: UDR-STL Pattern Analysis Summary

The empirical analysis (Table 2) reveals 97.8% pattern detection yet fundamental mapping challenges. Pattern 8 (If-Then Implication) dominates with 16,098 instances across 63,078 occurrences in 70+ variants, while edge detection patterns (2, 10, 12) are completely absent. Co-occurrence analysis shows strong If-Then/logical operator alignment (30,219 instances) but broad distribution of $\mathbf{G}$ and $\mathbf{F}$ across patterns. Critically, unexpected mappings (Pattern 1$\rightarrow\wedge$ instead of $\mathbf{G}$, Pattern 7$\rightarrow\mathbf{F}$ instead of $\mathbf{F}_{[0,k]}$) demonstrate that syntactic patterns alone cannot determine semantic interpretation, necessitating probabilistic frameworks that integrate syntax with semantic and contextual analysis.

5 Discussion

Our empirical investigation reveals both the promise and fundamental challenges of syntax-based approaches to temporal logic translation. The striking contrast between high pattern detection (97.8%) and low exact match accuracy illuminates a critical insight: while syntactic patterns successfully identify temporal expressions, the mapping to STL operators is inherently probabilistic rather than deterministic, with significant variability yet to be explored.

5.1 The Syntax-Semantics Gap

Pattern 8 (If-Then Implication) exemplifies the fundamental mismatch between linguistic expression and logical formalization – despite 16,098 correct mappings to implication operators, it manifests through 70+ syntactic variants. This many-to-many relationship reveals natural language's richer representational system compared to temporal logic's rigid operators. Unexpected mappings (Pattern 1 producing $\wedge$ instead of $\mathbf{G}$, Pattern 7 losing bounds) confirm that syntax alone cannot disambiguate logical intent.

5.2 Probabilistic Nature of Patterns

Our patterns demonstrate statistical tendencies rather than deterministic rules. While reliably identifying temporal expressions, their STL operator mappings exhibit probabilistic behavior influenced by context and linguistic variation. This suggests successful NL-to-STL systems must model uncertainty explicitly, treating pattern-operator mappings as distributions. The extensive within-pattern variability indicates rich linguistic phenomena requiring empirical characterization of these probability distributions.

5.3 Theoretical and Practical Implications

The absence of edge detection patterns (2, 10, 12) suggests STL's mathematically rise ($\uparrow \phi$) and fall ($\downarrow \phi$) operators lack direct linguistic correlates, with natural language expressing state transitions through alternative mechanisms our patterns miss. This indicates bottom-up approaches from linguistic phenomena may prove more effective than top-down logical operator mapping. Practically, Pattern 8's dominance shows conditional structures form temporal specification backbones, while high pattern co-occurrence (30,219 instances) demonstrates compositionality requiring multi-pattern analysis. The probabilistic relationships argue for machine learning approaches over deterministic rules.

5.4 Future Work

Future work can leverage these findings through: (1) **In-context learning** using our patterns with labeled NL-UDR-STL triplets for enhanced zero-shot translation; and (2) **Fine-tuning approaches** producing novel datasets to train specialized architectures. Research could integrate UD representations into transformer embeddings, using syntactic patterns as inductive biases for improved temporal logic translation.

6 Limitations

Our approach makes several assumptions that merit explicit discussion. First, we assume compositional semantics – that complex temporal expressions can be systematically decomposed into our twelve patterns – which may not hold for idiomatic or context-dependent expressions in natural language. Second, our evaluation relies on SpaCy's UD parsing accuracy; errors in syntactic analysis propagate through our pipeline, and parser performance on technical specifications may differ from its training on general web text. The patterns themselves were derived theoretically rather than empirically, potentially missing naturalistic expression patterns that emerge from actual usage.

Our empirical validation is limited to a single English dataset (`circuit_total_refined`) with 7,003 sentences from the circuit/hardware domain, which may not generalize to other temporal specification domains such as medical protocols, legal contracts, or aerospace requirements. The dataset's focus on STL may not extend to other temporal logics (LTL, CTL, MTL), and our binary pattern matching approach cannot capture gradient membership or probabilistic relationships between syntax and semantics. Furthermore, we conducted a single evaluation run without cross-validation, limiting our understanding of result variance.

These limitations suggest important directions for future work: expanding to multilingual datasets, incorporating probabilistic pattern matching, validating across diverse domains, and developing methods robust to parser errors. We emphasize that our findings establish a baseline understanding of syntax-semantics relationships rather than a complete solution, providing a foundation for hybrid approaches that can address these limitations through integration with semantic understanding and contextual reasoning.

References

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language.

Patrick Charollais. 2017. ECMA-404, 2nd edition, December 2017.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. 2023. NL2TL: Transforming Natural Languages to Temporal Logics using Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15880–15903, Singapore. Association for Computational Linguistics.

Noam Chomsky. 1965. *Aspects of the theory of syntax*, 20. print edition. Number 11 in Special technical report / Massachusetts Institute of Technology, Research Laboratory of Electronics. MIT Press, Cambridge, Mass.

Marie-Catherine De Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, pages 1–54.

Alexandre Donzé, Thomas Ferrere, and Oded Maler. 2013. Efficient robust monitoring for STL. In *International conference on computer aided verification*, pages 264–279. Springer.

David R. Dowty. 1979. *Word meaning and Montague grammar: the semantics of verbs and times in generative semantics and in Montague's PTQ*. Number v. 7 in Synthese language library. D. Reidel Pub. Co, Dordrecht ; Boston.

E. Allen Emerson and Joseph Y. Halpern. 1986. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178.

Yue Fang, Zhi Jin, Jie An, Hongshen Chen, Xiaohong Chen, and Naijun Zhan. 2025. Enhancing Transformation from Natural Language to Signal Temporal Logic Using LLMs with Diverse External Knowledge. ArXiv:2505.20658 [cs].

Jie He, Ezio Bartocci, Dejan Ničković, Haris Isakovic, and Radu Grosu. 2022. DeepSTL: from english requirements to signal temporal logic. In *Proceedings of the 44th International Conference on Software Engineering*, pages 610–622, Pittsburgh Pennsylvania. ACM.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Laurence R. Horn. 2001. *A natural history of negation*. The David Hume series. CSLI, Stanford, Calif.

Sascha Konrad and Betty H. C. Cheng. 2005. Real-time specification patterns. In *Proceedings of the 27th international conference on Software engineering - ICSE '05*, page 372, St. Louis, MO, USA. ACM Press.

Angelika Kratzer. 1991. Modality. In Arnim von Stechow and Dieter Wunderlich, editors, *Handbuch Semantik*, pages 639–50.

Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M.

Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Yassine Lakhnech, and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, volume 3253, pages 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.

Yuchen Mao, Tianci Zhang, Xu Cao, Zhongyao Chen, Xinkai Liang, Bochen Xu, and Hao Fang. 2024. NL2STL: Transformation from Logic Natural Language to Signal Temporal Logics using Llama2. In *2024 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, pages 469–474, Hangzhou, China. IEEE.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology - HLT '94*, page 114, Plainsboro, NJ. Association for Computational Linguistics.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In *Approaches to natural language: Proceedings of the 1970 Stanford workshop on grammar and semantics*, pages 221–242. Springer.

Joakim Nivre. 2020. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. *Proceedings of the Twelfth Language Resources and Evaluation Conference*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajicˇ, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection.

Barbara H. Partee. 1984. Nominal and temporal anaphora. *Linguistics and Philosophy*, 7(3):243–286.

Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, Providence, RI, USA. IEEE.

Guido Van Rossum and Fred L Drake Jr. 2009. *The Python Language Reference Manual*. Network Theory Ltd.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. *Proceedings of the 5th International Workshop on Semantic Evaluation*.

A Appendix

A.1 Formal Preliminaries

A.1.1 Signal Temporal Logic

STL formulas are defined recursively using the following grammar (He et al., 2022; Mao et al., 2024; Chen et al., 2023):

$$\begin{aligned}\phi ::= \quad & \pi^\mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \\ \mid \quad & \mathcal{F}_{[a,b]}\phi \mid \mathcal{G}_{[a,b]}\phi \mid \phi_1 \mathcal{U}_{[a,b]}\phi_2\end{aligned} \quad (2)$$

Where: (i) π^μ represents atomic predicates (e.g., $x \sim \mu$ where x is a variable, $\sim$ is a comparison operator, and μ is a value); (ii) ϕ, ϕ_1, ϕ_2, ... ϕ_n are STL formulas, and (iii) $[a, b]$ represents time intervals where $a, b \in \mathbb{R}$ and $a \leq b$.

A.1.2 Atomic Predicates

An atomic predicate π^μ in STL is a basic comparison of the form $x \sim \mu$, where x is a real-valued signal variable, $\sim \in \{<, \leq, =, \geq, >, \neq\}$ is a comparison operator, and $\mu \in \mathbb{R}$ is a constant, representing the simplest testable condition that can be evaluated as true or false at any given time instant (Maler and Nickovic, 2004). In the context of cyber-physical systems, atomic predicates typically express constraints on sensor readings (e.g., $\textit{temperature} > 25$), actuator states (e.g., $\textit{valve\_position} = 1$), or derived signals (e.g., $\textit{velocity} \leq 50$).

A.1.3 Logical Operators

The logical operators within STL operate on STL formulas ϕ, ϕ_1, and ϕ_2 as follows: (i) $\neg\phi$, meaning the negation of formula ϕ, (ii) $\phi_1 \wedge \phi_2$, meaning the conjunction (and) of formulas ϕ_1 and ϕ_2, (iii) $\phi_1 \vee \phi_2$, meaning the disjunction (or) of formulas ϕ_1 and ϕ_2, (iv) $\phi_1 \Rightarrow \phi_2$, meaning the implication from ϕ_1 to ϕ_2, and (v) $\phi_1 \Leftrightarrow \phi_2$, meaning the equivalence between ϕ_1 and ϕ_2 (Maler and Nickovic, 2004). These operators preserve their classical Boolean semantics at each time point, enabling compositional specification of complex logical relationships between temporal properties.

A.1.4 Temporal Operators

The temporal operators within STL are: (i) $\mathcal{F}_{[a,b]}\phi$ (Eventually/Finally), meaning the formula ϕ must be true at least once within the time interval $[a, b]$, formally $\exists t \in [a, b] : \phi(t)$; (ii) $\mathcal{G}_{[a,b]}\phi$ (Always/Globally), meaning the formula ϕ must be true throughout the entire time interval $[a, b]$, formally $\forall t \in [a, b] : \phi(t)$; and (iii) $\phi_1 \mathcal{U}_{[a,b]}\phi_2$ (Until), meaning ϕ_1 must hold until ϕ_2 becomes true within the time interval $[a,b]$, formally $\exists t \in [a,b] : \phi_2(t) \wedge \forall t' \in [a,t) : \phi_1(t')$ (Maler and Nickovic, 2004; Donzé et al., 2013). The bounded time intervals enable precise specification of real-time constraints essential for cyber-physical systems.

A.1.5 Extended Operators

For practical applications, STL often includes derived operators that can be expressed using the core grammar: (i) Rise operator $\uparrow \phi \equiv \neg\phi\mathcal{U}\phi$, detecting positive edges; (ii) Fall operator $\downarrow \phi \equiv \phi\mathcal{U}\neg\phi$, detecting negative edges; (iii) Weak Until $\phi_1\mathcal{W}\phi_2 \equiv \mathcal{G}\phi_1 \vee (\phi_1\mathcal{U}\phi_2)$, where ϕ_1 holds indefinitely or until ϕ_2; and (iv) Release $\phi_1\mathcal{R}\phi_2 \equiv \neg(\neg\phi_1\mathcal{U}\neg\phi_2)$, the dual of Until (Pnueli, 1977).

A.1.6 Universal Dependencies

Universal Dependencies (Nivre et al., 2016) represents syntax as directed graphs $G = (V, E, \ell)$ with word nodes V, dependency edges E, and labels $\ell : E \to R$ from 37 universal relations including core arguments (`nsubj`, `obj`), modifiers (`advmod`, `nmod`), and function words (`mark`, `det`).

A.1.7 Dependency Relations

The 37 universal relations are organized into several categories: (i) **Core arguments**: `nsubj` (nominal subject), `obj` (object), `iobj` (indirect object); (ii) **Non-core dependents**: `obl` (oblique), `vocative`, `expl` (expletive), `dislocated`; (iii) **Nominal dependents**: `nmod` (nominal modifier), `appos` (apposition), `nummod` (numeric modifier); (iv) **Clausal dependents**: `advcl` (adverbial clause), `acl` (clausal modifier), `ccomp` (clausal complement); (v) **Modifier words**: `advmod` (adverbial modifier), `amod` (adjectival modifier); (vi) **Function words**: `aux` (auxiliary), `cop` (copula), `mark` (marker), `det` (determiner), `case` (case marking) (De Marneffe et al., 2021).

A.1.8 Dependency Patterns

A dependency pattern $P = (r, h, d, C)$ consists of relation r, head/dependent constraints h/d, and contextual constraints C. Pattern P matches edge (u, v) when all constraints are satisfied. For temporal expressions, key patterns include: (i) `advmod` patterns for temporal adverbs (e.g., *always*, *eventually*); (ii) `mark` + `nummod` patterns for bounded expressions (e.g., *within 5 seconds*); (iii) `advcl` patterns for subordinate temporal clauses (e.g., *until X happens*); and (iv) compound patterns combining multiple relations for complex expressions.

A.2 Detailed Methodology

A.2.1 Dataset

We utilize the `circuit_total_refined` lifted NL-STL pairs dataset constructed by Chen et al. (2023) evaluation[1], containing 7,003 natural language sentences paired with their corresponding Signal Temporal Logic (STL) formulas. Each entry consists of six fields: (i) **ID**: unique sentence identifier; (ii) **Sentence**: natural language expressing temporal properties; (iii) **LTL**: temporal logic formula with English operators (e.g., "always", "eventually"); (iv) **Logic Sentence**: sentence with marked atomic propositions; (v) **Logic LTL**: formula with propositions as word spans (Span i, j); and (vi) **Propositions**: list of atomic propositions. The dataset provides comprehensive coverage of temporal operators (**G**, **F**, **U** and bounded variants), diverse syntactic structures, varying proposition complexity (0-4 propositions per sentence), and expert-annotated ground truth formulas for evaluation.

A.2.2 Computational Infrastructure

All evaluations and analyses were conducted on NVIDIA Saturn Cloud utilizing 2×NVIDIA A100 GPUs, 32 CPU cores, 512GB RAM, and 5TB disk storage over approximately 514 hours of compute time. The implementation employed SpaCy 3.8.0 with the `en_core_web_lg` model (Honnibal and Montani, 2017) for dependency parsing following Universal Dependencies v2 guidelines (Nivre, 2020), selected for its state-of-the-art accuracy and comprehensive syntactic coverage. The software stack comprised Python 3.x (Van Rossum and Drake Jr, 2009) with core libraries including JSON for data serialization (Charollais, 2017), regular expressions for pattern matching, difflib for similarity computation, and explicit memory management through garbage collection. The SpaCy pipeline was configured with tokenization preserving exact positions for ground truth alignment, part-of-speech tagging using Penn Treebank tagset (Marcus et al., 1994), dependency parsing, named entity recognition, and lemmatization. This infrastructure supported a multi-stage processing pipeline encompassing data loading from the `circuit_total_refined` dataset (Chen et al., 2023), preprocessing for token alignment, linguistic analysis via dependency parsing, UD pattern extraction, STL formula construction, and com-

[1] This dataset originally came from He et al. (2022), which was then transformed by Chen et al. (2023) in their evaluation.

prehensive evaluation through similarity metrics. The substantial computational resources enabled efficient batch processing and in-memory caching strategies, ensuring both reproducibility and computational efficiency across the 7,003 sentence dataset.

A.2.3 Universal Dependencies Parsing

Each sentence undergoes preprocessing to maintain word span alignment with ground truth annotations. SpaCy's pipeline transforms text into annotated linguistic structures through tokenization (preserving position information), part-of-speech tagging (providing grammatical categories), and dependency parsing (constructing directed graphs with 37 UD relation types). The parser produces dependency trees capturing both local and long-distance relationships crucial for temporal expressions. This structured representation enables identification of systematic correspondences between syntactic patterns and temporal logic operators.

A.2.4 Pattern-Based Extraction Framework

The pattern extraction framework implements twelve core patterns designed to map Universal Dependencies structures to Signal Temporal Logic operators. These patterns, defined in a structured dictionary, capture temporal relationships through specific dependency configurations as featured in Table 1. The extraction pipeline processes sentences through four integrated components: pattern detection traverses dependency trees matching against canonical patterns and lexical variants; temporal component extraction maps keywords like "always," "eventually," "within," and "for" to STL operators ($\mathbf{G}$, $\mathbf{F}$, $\mathbf{F}_{[0,k]}$, $\mathbf{G}_{[0,k]}$) with associated bounds; logical operator extraction identifies connectives ($\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$) through compound phrase detection and token analysis; and atomic proposition extraction analyzes dependency subtrees to identify signal references and comparison operators. Throughout this process, the system maintains precise word span alignment, with each extracted component storing its token span, operator type, textual representation, and Universal Dependencies evidence, enabling compositional STL formula construction while preserving linguistic provenance for error analysis and interpretability.

A.2.5 STL Formula Construction

The STL formula construction employs a compositional approach through the `construct_stl_formula_independent` function, which processes extracted components in three stages. First, semantic role extraction analyzes the Universal Dependencies parse to identify whether propositions serve as conditions, assertions, or temporal bounds based on dependency markers (*mark*, *aux*, *advcl*) and modal expressions. Second, logical structure building determines operator precedence and scope by analyzing the syntactic hierarchy, with temporal operators applied according to their position in the dependency tree and logical connectives maintaining their syntactic scope. Third, the system constructs the final STL formula by recursively combining atomic propositions with temporal operators ($\mathbf{G}$, $\mathbf{F}$, $\mathbf{U}$) and logical connectives ($\wedge$, $\vee$, $\neg$, $\rightarrow$), applying bounded intervals where numeric modifiers are present. The construction process handles nested temporal expressions through depth-first traversal of the logical structure, ensuring correct operator precedence (negation $>$ temporal $>$ conjunction $>$ disjunction $>$ implication) while preserving the semantic relationships encoded in the dependency parse. Formula normalization standardizes spacing, operator symbols, and parenthesization to enable consistent comparison with ground truth annotations.

A.2.6 Evaluation of UDR and STL Relationships

This study evaluates the proposed formal relationship between Universal Dependencies Relations and Signal Temporal Logic through comprehensive empirical analysis of 7,003 sentence-STL pairs. The evaluation framework employs multiple metrics: exact match accuracy using advanced formula normalization handling operator equivalences and structural variations; similarity scores via `SequenceMatcher` providing gradient correctness measures (0-1); component-wise accuracy separately assessing temporal operators, logical connectives, atomic propositions, and temporal bounds; and error severity classification into five levels based on similarity thresholds (complete failure < 0.2, major errors $0.2 - 0.4$, moderate errors $0.4 - 0.6$, minor errors $0.6 - 0.8$, near matches > 0.8).

The evaluation procedure follows eight systematic steps: (i) dataset loading and SpaCy initialization; (ii) preprocessing to maintain word span alignment; (iii) dependency parsing to generate UD trees; (iv) pattern matching against twelve defined patterns; (v) temporal/logical component extraction; (vi) compositional STL formula construction; (vii) comparison with ground truth using evaluation metrics; and (viii) statistical analysis of pattern occurrences, error distributions, and complexity correlations. Within this framework, the evaluation focuses on three key analyses of UDR-STL relationships: (i) Pattern-to-STL mapping analysis exploring the relationship between specific UDR patterns and STL operator combinations; (ii) Pattern variation analysis examining extensive syntactic diversity by identifying sentences where expected STL operators appear without canonical pattern matches, revealing variants such as "continuously," "constantly," and "throughout" for the "always" operator, and quantifying coverage impact to determine which patterns exhibit the most variation; and (iii) Co-occurrence analysis between individual UDR patterns and STL components to identify systematic associations between syntactic structures and logical operators. This systematic approach ensures reproducibility while capturing both overall performance and detailed insights into the pattern-based approach's strengths and limitations.

A.2.7 Statistical Analysis

Evaluation employs multiple metrics including exact match accuracy using advanced formula normalization handling operator equivalences and structural variations; similarity scores via `SequenceMatcher` providing gradient correctness measures (0-1); component-wise accuracy separately assessing temporal operators, logical connectives, atomic propositions, and temporal bounds; and error severity classification into five levels based on similarity thresholds (complete failure < 0.2, major errors $0.2 - 0.4$, moderate errors $0.4 - 0.6$, minor errors $0.6 - 0.8$, near matches > 0.8).

We conduct three primary analyses to evaluate the UDR-STL relationships: (i) **Pattern-to-STL mapping analysis** exploring the relationship between specific UDR patterns and STL operator combinations, quantifying how frequently each pattern correctly maps to its expected operator versus alternative mappings, and identifying systematic deviations from theoretical predictions; (ii) **Pattern variation analysis** examining extensive syntactic diversity by identifying sentences where expected STL operators appear without canonical pattern matches, revealing variants such as "continuously," "constantly," and "throughout" for the "always" operator, and quantifying coverage impact to determine which patterns exhibit the most variation; and (iii) **Co-occurrence analysis** between individual UDR patterns and STL components using lift analysis to identify systematic associations between syntactic structures and logical operators, revealing which pattern combinations most strongly predict specific temporal logic constructs.

A.3 Tables

No.	Pattern	UD Pattern	Logic	Variation	Cnt	%
1	Any Time Global	`det("any")+ nmod("time")`	G	every time	1,079	1.16
				det(every, time)	704	0.76
				Subtotal	**1,783**	**1.91**
2	Become/ Change Rise	`compound ("become"| "change")`	$\uparrow \Phi$	becomes	0	0.00
				Subtotal	**0**	**0.00**
3	Bounded After	`mark("after")+ nummod`	$F[k,\infty]$	mark(after, instant)	17	0.02
				mark(after, be)	3	0.00
				mark(after, is)	3	0.00
				All Others	9	0.01
				Subtotal	**32**	**0.03**
4	Always	`advmod ("always")`	G	advmod(always, is)	1,704	1.83
				during the first	904	0.97
				during the next	472	0.51
				All Others	7,813	8.38
				Subtotal	**10,893**	**11.69**
5	Eventually	`advmod("event- ually")`	F	at most	1,840	1.97
				advmod(eventually, be)	743	0.80
				advmod(finally, be)	631	0.68
				All Others	6,967	7.47
				Subtotal	**10,181**	**10.92**
6	Bounded For	`mark("for")+ nummod`	$G[0,k]$	mark(for, :)	5	0.01
				mark(for, ,)	4	0.00
				mark(for, needs)	1	0.00
				Subtotal	**10**	**0.01**
7	Bounded Within	`mark("within") +nummod`	$F[0,k]$	within	3,000	3.22
				within the first	1,046	1.12
				within the coming	463	0.50
				All Others	1,731	1.86
				Subtotal	**6,240**	**6.69**
8	If-Then Impl.	`advmod("then")+ mark("if")`	$\rightarrow$	then	16,098	17.27
				it is always	5,543	5.95
				when	5,229	5.61
				All Others	36,208	38.85
				Subtotal	**63,078**	**67.68**
9	Negated Always	`advmod ("never")`	$\neg G$	never	206	0.22
				Subtotal	**206**	**0.22**
10	No Longer Fall	`advmod("no longer")`	$\downarrow \Phi$	NA	0	0.00
				Subtotal	**0**	**0.00**
11	Until	`mark("until")+ advcl`	U	mark(before, ends)	467	0.52
				mark(until, ends)	160	0.17
				mark(until, is)	49	0.05
				All Others	86	0.09
				Subtotal	**782**	**0.84**
12	When First Rise	`mark("when")+ advmod("first")`	$\uparrow \Phi$	NA	0	0.00
				Subtotal	**0**	**0.00**
Grand Total					**93,205**	**100.00**

Table 3: Pattern variation analysis with logic operators