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

Christopher Rashidian
Purdue Polytechnic Institute
Purdue University
West Lafayette, IN 47907, USA
crashid@purdue.edu

Sabine Brunswicker
Purdue Polytechnic Institute, AIDA[3]
Purdue University
West Lafayette, IN 47907, USA
sbrunswi@purdue.edu

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 17 theoretically-motivated patterns, evaluated on the NL2TL benchmarking dataset of 7,002 expert-annotated sentence-STL pairs, and an additional cross-domain analysis. We built a parser guided by this grammar to explore the formal deterministic relationship between UDR Compositions and STL Operators, achieving ∼99% sentence coverage, ∼54% exact matches (and ∼97% similarity). Sentence-level regression analyses predict STL statements and STL Operator classes, considering the co-occurance of UDR substructures (UDR components) with an accuracy of more than ∼74% and ∼81%, respectively. They uncover a new logical grammatical link between temporal NL and formal logic, that is conditioned by the sentence-level context, and provide insights into how linguistic theory unfolds in practice through temporal linguistic expressions.

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, understanding how natural language (NL) temporal expressions systematically correspond to formal temporal logic operators of artificial languages like Linear Temporal Logic (LTL) or Signal Temporal Logic (STL) raises fundamental questions for computational linguistics (Maler and Nickovic, 2004; Sistla and Clarke, 1985). 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 $\mathcal{G}_{[2,46]}(\mathit{signal} \geq 8.9)$. This challenge has motivated emerging research in recent NLP research: Researchers fine-tune LLMs or train new translation models (e.g. DeepSTL (He et al., 2022), NL2TL (Chen et al., 2023), GraFT (English et al., 2025)) to translate NL to STL accurately. However, these approaches provide insufficient insight into how syntactic structures of NL systematically encode temporal logic operators of artificial languages like STL, leaving the fundamental linguistic mechanisms unexplained.

In this paper, we argue that Universal Dependencies (UD) (De Marneffe et al., 2021; Nivre, 2020) enable a systematic mapping between natural language and temporal logic operators of LTL or STL. Specifically, we ask: *Is there a relationship between UDR components and STL Operator composition?* Rather than developing another system for NL-STL Translation, we propose and empirically investigate a translational grammar that maps a set of "UDR Compositions" to Signal Temporal Logic, leveraging universal grammatical principles. UDR Compositions combine UD syntax substructures (such as the adverbial modifiers "*always*", "*eventually*", and "*not*") and associated semantics (e.g., temporal words associated with certain structures) to capture the building blocks that constitute temporal structures in natural language.

Our investigation focuses on empirically identifying and analyzing this grammar to enhance our understanding of how the structural theory of UDR for natural language relates to STL. Our paper makes two unique contributions:

First, we introduce a *new grammar* that defines a (probabilistic) relationship between UDR Compositions and STL Operators. We introduce 17 theoretically-motivated Universal Dependency Relationship Compositions ("*UDR Composition(s)*")

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1852–1866

defined independently based on UD syntactic theory, ranging from single-relation Compositions signaling operator categories to multi-relation configurations indicating specific operators. We programmed a parser to automatically extract these UDR Compositions and map them to STL Operators.

Second, we empirically examine this translational grammar based on two datasets, namely the `circuit_total_refined` dataset with 7,002 NL-STL pairs used in prior research (He et al., 2022; Chen et al., 2023) and 100 handcrafted natural language sentence-STL pairs from the TimeBank 1.2 Corpus (referred to as "*NL2TL*" "*TimeBank 1.2*" and in this paper)(Pustejovsky et al., 2003, 2006). In a first step, we examined our grammar assuming a rule-based deterministic mapping between UDR Compositions and STL Operators, using a rule-based parser, with an exact match translation of 53.84% into full STL sentences, and an average similarity of complete STL statements of 96.70% [1]. In addition, we performed sentence-level statistical analysis to examine how the seventeen core UDR Compositions predict and explain the STL Operators, as well as complete STL statements, considering the co-occurrence of UDR-Cs in a sentence. Logistic regression achieved an accuracy of ∼74.3% accuracy in translating UDR Compositions into complete STL statements. Additional multinomial logistic regression mapped 17 UDR Compositions into 10 STL Operators classes, and successfully predicted STL Operators with an accuracy of about ∼81% confirming a probabilistic relationship between multiple syntax substructures in predicting STL, caused by variability in language expression. For example, certain syntactic substructures like the UDR Composition "*Conjunction*" (∧) increase prediction accuracy as they are used in multiple contexts, and with different meanings to express temporality.

Our research has important implications for future research at the intersection of linguistic theory as well as deep-learning-inspired NLP. It sets the foundation for new theory-guided research on the nature of temporal language expression, and also guides future research on grammar-inspired NL-STL translation in the realm of GraFT (English et al., 2025).

[1]Here we assumed a 1:1 relationship between UDR Compositions and complete STL statements

2 Theoretical Foundations

2.1 Signal Temporal Logic

Signal Temporal Logic (STL) serves as the formal target language for temporal specifications in cyber-physical systems. Formally, STL extends First Order Logic, which itself extends propositional logic with quantifiers and predicates, by introducing temporal operators that quantify over bounded dense-time intervals. This extension accommodates real-valued signals and continuous-time dynamics, enabling the mathematical precision required for verification of safety-critical systems while addressing quantitative constraints ubiquitous in real-world applications (Donze et al., 2013; Maler and Nickovic, 2004). The bounded temporal operators with explicit time windows address a fundamental limitation of classical temporal logics when applied to systems with strict timing requirements. 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 appear in Appendix A.1.

2.2 Universal Dependencies

Universal Dependencies (De Marneffe et al., 2021; Nivre et al., 2016), grounded in Tesnière's dependency grammar theory (1959), developed to formally describe syntactic structures of natural language. It 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 Composition to STL Operators

We propose seventeen patterns (Table 1) that provide comprehensive STL coverage for cyber-physical system specifications. Each pattern consists of two key components:

the *UDR Composition* specifies the precise syntactic structure using Universal Dependencies relations (e.g., `advmod("always")`, `mark("if")+advcl+mark("then")`) that triggers pattern detection in natural language text, whereas the *STL Operator* defines the formal symbolic representation (e.g., G, F, U, $\rightarrow$) used in the resulting temporal logic formula. This distinction enables systematic mapping from linguistic surface forms to formal Signal Temporal Logic representations through dependency parsing. 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, demonstrating that dependency structures encode temporal semantics at multiple levels of abstraction, from atomic temporal operators to propositional-level and hierarchical formula compositions: `advmod` relations encode temporal quantification (Pattern Nos. 1, 2, 6, 7, 8), `mark+nummod` capture bounded constraints (Pattern Nos. 10, 11, 12, 13), lexical patterns identify state transitions (Pattern Nos. 9, 13, 14, 15, 16, 17), and logical relationships (Pattern Nos. 4, 5, 6).

No.	Pattern Name	UDR Composition	STL Op.
1	Always	advmod("always")	$\mathcal{G}$
2	Eventually	advmod("eventually")	$\mathcal{F}$
3	Until	mark("until")+advcl	$\mathcal{U}$
4	Conjunction	cc("and")+conj	$\wedge$
5	Disjunction	cc("or")+conj	$\vee$
6	Propositional Negation	advmod("not")	$\neg$
7	Negated Always	advmod("never")	$\neg\mathcal{G}$
8	If-Then Implication	mark("if")+advmod("then")	$\rightarrow$
9	Become/Change Rise	compound("become"/"change")	$\uparrow \phi$
10	Bounded After	mark("after")+nummod	$\mathcal{F}_{[k,\infty]}$
11	Bounded For	mark("for")+nummod	$\mathcal{G}_{[0,k]}$
12	Bounded Within	mark("within")+nummod	$\mathcal{F}_{[0,k]}$
13	When First Rise	mark("when")+advmod("first")	$\uparrow \phi$
14	Universal Quantification	det(specifications, "all")+nsubj(hold)	$\forall\varphi_i$
15	Existential Quantification	det(constraint, "a")+acl:relcl	$\exists\varphi_j$
16	Conditional Formula Selection	advcl(apply, holds)+mark("if")	ctx $\Rightarrow \varphi_i$
17	Specification Conjunction	obj(require, both)+nsubj(system)	$\bigwedge_i \varphi_i$

Table 1: Systematic correspondences between UDR Compositions and STL Operators.

The patterns exploit systematic correspondences between linguistic quantification and temporal logic. Determiners (*any*, *every*) parallel universal quantification ($\mathbf{G}$), while indefinites map to existential quantification ($\mathbf{F}$) (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 that demonstrate how syntactic subordination encodes semantic scope (Emerson and Halpern, 1983).

Pattern design leverages formal semantic principles: negation duality ($\neg\mathbf{G}_{[a,b]}\phi \equiv \mathbf{F}_{[a,b]}\neg\phi$) motivates Pattern No. 7's *never* encoding (Horn, 2001); material conditional correspondence justifies Pattern No. 8's if-then mapping (Kratzer, 1991); and edge operators ($\uparrow \phi \equiv \neg\phi\mathbf{U}\phi$) 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, which is essential for verification tasks (Konrad and Cheng, 2005).

3 Methodology

We used SpaCy 3.8.7 (Honnibal and Montani, 2017) to develop a Universal Dependencies parser and extract binary features for 17 theoretically motivated Universal Dependency Construction Classes (Table 1), along with 10 distinct STL Operator classes, as well as full STL Statements. To examine the quality of our parser, we implement a manual accuracy check on a 2% subsample of NL2TL and the 100 sentences from TimeBank 1.2 (see tables 7 and 17 in Annex for this). We used this parser to examine a deterministic 1:1 relationship between UDR Compositions and STL Operators, along with an exact STL statement. We evaluate the NL-STL translation based on exact matches (string-edit distance of 0 or 1), and similarity scores based on regex pattern matching (Chapman and Stolee, 2016).

To examine the relationship between UDR Compositions and STL Operators, we performed a binary true/false sentence-level (N = 7,002 sentences, 80/20 train-test split) logistic regression (Nick and Campbell, 2007), focused on UDR to complete STL statements, followed by a sentence-level multinomial (N = 7,002 sentences, 80/20 train-test split) logistic regression (Long, 1997) to examine the UDR Composition-STL Operator relationships considering multiple UDR Components jointly in a sentence. We used the Scikit-learn library and implemented the regression with the `lbfgs` solver and regularization (Lee et al., 2006; Minka, 2003; Moritz et al., 2016; Pedregosa et al., 2011). We report both the coefficients and odds ratios to discuss the importance of different UDR Components for predicting STL sentence, as

well as STL Operator classes considering UDR co-occurrences (Brunswicker and Haefliger, 2025). Complete methodological details appear in Appendix A.2.

4 Results

4.1 Descriptives

We analyzed 7,002 natural language sentences paired with STL formulas from NL2TL, and an additional 100 hand-crafted pairs from TimeBank 1.2. Table 2 provides an overview of the results of our analysis. For the NL2TL dataset, we identified 20,410 STL statements across 9 STL classes, and 42,428 UDR statements across 17 UDR classes. Our composition detection analysis achieved 99.94% coverage (6,998 out of 7,002 sentences). We achieved 53.84% exact matches and 96.70% string similarity.

Dataset Characteristics	NL2TL	TimeBank 1.2
Domain	Technical specs	Narrative discourse
Sentences analyzed	7,002	100
Num. STL Statements analyzed	20,410	185
STL Classes	9	4
Sentences with no UDR Compositions detected	4	23
Types of UDR Compositions detected	17	13
Total UDR Compositions detected	42,428	137
Sentence level coverage	99.94%	77.00%
Exact matches complete sentence-level STL Statements	3,770 (53.84%)	1 (1.00%)
Average similarity complete sentence-level STL statement	96.70%	95.8%

Table 2: Basic Descriptives: NL2TL and TimeBank 1.2.

We performed a spot test on the quality of our parser using a 2% subsample of the NL2TL dataset and 100% of The TimeBank 1.2 revealed substantial variation in composition detection performance. It suggests that our parser has sufficient accuracy in terms of the detection of UDR Compositions and STL Operators (we achieved an average parser accuracy of 86.28%). More details on this manual evaluation for both datasets can be found in Tables 7 and 17 in the Annex.

Pattern No.	Pattern Name	UDR Comp.	No. of Inst.	%
1	Always	advmod("always")	3,178	7.49
2	Eventually	advmod("eventually")	658	1.55
3	Until	mark("until")+advcl	3,879	9.14
4	Conjunction	cc("and")+conj	6,208	14.63
5	Disjunction	cc("or")+conj	1,540	3.63
6	Prop. Negation	advmod("not")	3,986	9.40
7	Negated Always	advmod("never")	102	0.24
8	If-Then Impl.	mark("if")+advmod("then")	727	1.71
9	Become/Change	compound("become"/"change")	1,185	2.79
10	Bounded After	mark("after")+nummod	4,064	9.58
11	Bounded For	mark("for")+nummod	802	1.89
12	Bounded Within	mark("within")+nummod	3,061	7.22
13	When First	mark("when")+advmod("first")	2,915	6.87
14	Universal Quant.	det("all")+nsubj	1,856	4.37
15	Existential Quant.	det("a")+acl:relcl	206	0.49
16	Cond. Formula Sel.	advcl+mark("if")	7,109	16.76
17	Spec. Conjunction	obj("both")+nsubj	952	2.24
Total			**42,428**	**100.00%**

Table 3: **NL2TL:** Composition frequency. Instances indicate of UDR Composition occurrences.

Table 3 describes our NL2TL's 42,428 UDR Composition instances and provides insight into the variability in how syntax structures constitute STL, and suggests a prominence of certain UDR Compositions in expressing temporality. *Conditional Formula Selection* (16.76%), followed by *Conjunction* (14.63%), and *Until* (9.14%). The lowest are *Existential Quantification* (0.49%), *Negated Always* (0.24%), and *Eventually* (1.55%). An analysis of our second dataset, the TimeBank 1.2's 137 instances (Table 15) exhibits a distinct distribution, suggesting that the domain may impact the dominance of certain UDRs for expressing STL in natural language. For example, in narrative discourse, Bounded For (12.41%), Conjunction (24.82%), and Propositional Negation (14.60%) dominate, while several compositions present in NL2TL are entirely absent. This distributional divergence reflects fundamental differences in temporal expression strategies between technical circuit specifications and narrative discourse.

4.2 Statistical Analyses of Patterns: UDR Composition - STL Relationships

We examined our proposed patterns in multiple ways. First, we explored the co-occurrence of UDR Compositions and STL Operators per sentence in a heatmap, then performed regression analysis to examine the probabilistic relationship between UDR Compositions and STL at the sentence level, considering UDR Composition co-occurrences. We first predicted full ground-truth STL StatementsWe use the ground truth label from the NL2TL dataset from the 17 UDR Components, using logistic regressions. Then, we use a multinominal logistic (MNL) regression to predict a particular STL Operator considering all 17 UDR Components occurring in a sentence assuming a deterministic relationship between UDR-STL Components based on our parser.

4.2.1 UDR Composition and STL Operator Co-occurence

UDR Composition and STL Operators co-occurrences in NL2TL (Table 22) reveal distinct frequency patterns and distributional characteristics. The dataset shows substantial variation across the 17×10 matrix, with frequencies ranging from 1 (UDR-C No. 15 with $\downarrow$) to 4,424 (UDR-C8 with $\rightarrow$). UDR-C No. 8 demonstrates the highest single operator association at 4,424 instances with $\rightarrow$, followed by UDR-C12 with $\rightarrow$ at 3,377 instances,

and UDR-C No. 8 with $\mathcal{F}_{[k,\infty]}$ at 3,259 instances. The top frequency concentrations show UDR-C No. 8's strong performance across multiple operators: 4,424 with $\rightarrow$, 3,259 with $\mathcal{F}_{[k,\infty]}$, 2,720 with G, and 1,966 with $\mathcal{G}_{[0,k]}$. UDR-C No. 12 also exhibits high frequencies, reaching 3,377 with $\rightarrow$, 2,692 with G, 2,596 with F, and 2,418 with $\mathcal{F}_{[k,\infty]}$. UDR-C No. 11 shows notable concentration in bounded always $\mathcal{G}_{[0,k]}$ with 3,006 instances, while UDR-C No. 1 peaks with G at 3,014 instances and UDR-C No. 2 reaches 3,129 with F. Lower frequency patterns include UDR-C No. 7, showing consistent low values across all operators (ranging from 2 to 206), and UDR-C No. 15 with minimal frequencies (1 to 97 across all operators). The $\rightarrow$ operator consistently shows the lowest frequencies across all compositions, with most values under 100. In contrast, $\rightarrow$ shows high activity across multiple compositions, appearing with frequencies above 1,000 in eight different UDR Compositions.

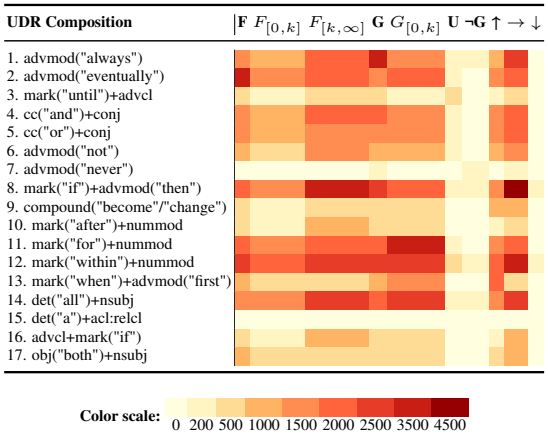

Table 4: **NL2TL:** UDR Compositions × STL Operators Co-occurrence heatmap. Color intensity represents co-occurrence frequency from light yellow (low) to dark red (high, max=4424).

TimeBank (Table 23) presents a strikingly different pattern with extreme sparsity. Of the 170 possible cells (17×10), only 22 contain non-zero values. The highest frequency is UDR-C No. 11 with $\mathcal{G}_{[0,k]}$ at 17 instances, followed by UDR-C No. 4 and UDR-C No. 6, each showing 4 instances with $\mathcal{G}_{[0,k]}$, and the rest showing only 1 to 3 cases.

4.2.2 Regression Analyses

	Precision	Recall	F-1 Score	Support
False (No Match)	0.716	0.736	0.726	647
True (Match)	0.768	0.749	0.758	754
Accuracy			0.743	1,401
Macro Average	0.742	0.743	0.742	1,401
Weighted Average	0.744	0.743	0.743	1,401

Table 5: **NL2TL:** Logistic regression test classification report (n=1,401).

Our sentence-level logistic regression analyses (Table 5) achieved an overall weighted accuracy of ∼74% (and a precision of true matches of ∼76%). An examination of the odds ratio (see Table 10) suggests that there are certain UDR Compositions, such as Conjunction ($\wedge$) and If-Then Implications($\rightarrow$), that are less likely to predict a match, potentially because there is variability in how they co-occur with other UDR Compositions. The confusion matrix can be found in Table 9 and the ROC Curve can be found in Figure 1.

The multinomial logistic regression, also performed at the sentence-level (N=7,002 for the NL2TL dataset), achieves 82.49% and 81.37% overall accuracy on the training and test set of the NL2TL dataset (Tables 6 and 12), with TimeBank 1.2 (Table 21) achieving a higher 90.00% accuracy. For the NL2TL dataset, we see variability in the STL Operator level classification accuracy with F and $F_{[k,\infty]}$ achieving the highest, and U and $\neg G$ achieving the lowest performance. F operator maintains perfect classification on NL2TL.

The primary confusion matrices (Tables 11 and 19) shows that confusion was highest between $F_{[k,\infty]}$ and G followed by $F_{[0,k]}$ and $G_{[0,k]}$, caused by how the different UDR Components, the syntatic substructures are combined.

STL Operator	Precision	Recall	F1	Support
F*	1.00	1.00	1.00	626
$F_{[0,k]}$	0.58	0.50	0.54	150
$F_{[k,\infty]}$	0.92	0.66	0.77	295
G	0.66	0.60	0.63	179
$G_{[0,k]}$	0.44	0.89	0.59	56
None	0.61	0.71	0.65	24
U	0.41	0.94	0.57	17
$\neg G$	0.60	1.00	0.75	3
$\uparrow$	0.54	0.93	0.68	15
$\rightarrow$	0.63	1.00	0.77	36
Macro Avg	0.64	0.82	0.70	1,401
Weighted Avg	0.84	0.81	0.82	1,401
Overall Accuracy: 81.37%				

Table 6: **NL2TL:** Test set performance (n=1,401). Per-operator classification metrics on held-out data. Training-test gap: 1.11% (82.49% vs 81.37%), indicating strong generalization without overfitting. *F serves as the baseline reference category.

We also analyzed the odds ratios and the coefficient magnitudes (see Table 10) that indicate how the likelihood of observing a particular UDR Composition in a sentence, increase the prediction of a particular STL Operator, holding other UDR Components constant: For example, as expected, the UDR-C No. 2 (Eventually) may be highly predictive for the F operator, while UDR-C No. 7 (Negated Always) is more likely to translate into $\neg G$, in the realms of a 1:1 mapping assumed by our grammar, rather independent from UDR-Cs. However, for $F_{[k,\infty]}$, we find that that UDR-C No. 12 (Bounded Within), with the highest coef-

ficient/odds ratio, is closely followed by UDR-C No. 1 (Always) with the second highest coefficient/odds value, suggesting they may be also be impactful in predicting the $F_{[k,\infty]}$ (potentially because of co-occurrence). Further details on this probabilistic relationship at the UDR Composition and STL Operator level can be found in Table 14 in the annex.

5 Discussion

Our empirical investigation reveals new insights into how syntactic theory for temporality translates into formal temporal logic, defined by mathematical artificial language. The striking contrast between high pattern detection (99.94% coverage on NL2TL, 77% on TimeBank 1.2) and variable accuracy rates (ranging from 0.00% to 100.00% across patterns, with 81.37% overall accuracy on NL2TL) shows a critical insight: The mapping between the building blocks of temporal syntax, the UDR components to STL Operators and full STL statements is inherently probabilistic, not only because of the domain context. Most importantly, we find that the co-occurrence of UDR components at the sentence level impacts the success of predicting STL operators. In other words, while there is some underlying logical relationship, the variability is caused by both the combination of syntactic structures and variability in semantic expression.

Our patterns demonstrate statistical tendencies caused by the way UDR-Components are combined: The mapping of UDR components to STL Operator is conditioned by structural context and linguistic variation. What is important is that the variability is not the same for all UDR-STL relationships, indicating that rich linguistic phenomena like temporal structures require the consideration of temporal structural building blocks, structural context, and semantics.

With that, we provide guidance to ongoing research on NL-STL translation: Our findings reveal new, valuable insights into how natural language systematically varies across sentences and domains, even when the data is decontextualized and the system description is lifted. In our research, we followed prior research (Chen et al., 2023) and used lifted STL statements to examine the relationship between syntactic structures and formal logic. However, even if we perform such lifting, we observe unique probabilistic relationships and variability caused by the combination of UDR-C at the sentence level. Technical domains exhibit highly systematized temporal encoding, which our UDR Compositions successfully capture, and precise temporal constraints because engineering discourse has evolved specialized syntactic conventions for unambiguous temporal specification. The high similarity scores (96.70% and 95.80%) across domains indicate that our patterns successfully identify core temporal structures, while our regression analysis provides insights into the underlying probabilistic relationship that causes variability.

Our findings address a key limitation in research on how NL relates to STL by offering a new grammar that provides a foundation for a logical mapping between syntax structure and STL Operators. Our empirical analysis uncovers a new probabilistic relationship between syntactic structures and temporal logic operators that can improve future theory development in computational linguistics. Further, it may also set the stage for the design of neurosymbolic frameworks like the one proposed by (English et al., 2025) that encode a grammar into the architecture of a neural network to improve not only the accuracy but also the interpretability of NL-STL translation models, based on the building blocks of temporal syntax structures, the UDR Components.

6 Limitations

Our approach relies on several assumptions worth discussing. We assume that UDR Compositions can systematically connect to STL Operators through our seventeen patterns, and examine how the NL maps to STL, considering the co-occurrence of the structural building blocks of temporal syntactic structures. We do not perform a granular STL-Operator level statistical analysis to examine the variability in the direct mapping further. Additional empirical analysis, including mixed effect regression models, and Bayesian approaches may offer new ways to examine the multi-level and nested relationship between the substructures and high-level sentences, as we as STL operators and STL sentence statements. Such research can also inspired deep-learning NLP research, that seeks to consider such findings when building deep learning and neural models like GraFT (English et al., 2025).

We examined our grammar using two English language datasets: NL2TL with 7,002 sentence pairs and 100 samples from TimeBank. We find

differences between domains in terms of the performance of our parser (53.84% complete matches versus 1.00%), TimeBank showing only 137 UDR Compositions compared to our full pattern set, and the probabilistic relationships explored with regression analyses. This suggests that our grammar needs to be studied and refined across additional datasets and contexts to increase its rigor. We invite others to explore our grammar from additional domains, such as probabilistic grammar development using our statistical relationships to assign confidence scores rather than fixed mappings, domain-adaptive composition discovery to address the significant performance gap between technical specifications and narrative text

Finally, hybrid interpretable systems integrating UDR Compositions into deep learning and neural approaches like GraFT for improved explainability (English et al., 2025).

References

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

Sabine Brunswicker and Stefan Haefliger. 2025. Is there collaboration in open collaboration? The role of producers and corporate users in Open Source software development. *Technovation*, 148:103325.

Carl Chapman and Kathryn T Stolee. 2016. Exploring regular expression usage and context in Python. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pages 282–293.

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

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.

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

Alexandre Donze, 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*, repr. with new preface edition. Number 7 in Synthese language library. Reidel, Dordrecht.

E. Allen Emerson and Joseph Y. Halpern. 1983. "Sometimes" and "not never" revisited: on branching versus linear time (preliminary report). In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '83*, pages 127–140, Austin, Texas. ACM Press.

William English, Dominic Simon, Sumit Kumar Jha, and Rickard Ewetz. 2025. Grammar-Forced Translation of Natural Language to Temporal Logic using LLMs. *International Conference on Machine Learning*.

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.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. 2006. Efficient l 1 regularized logistic regression. In *Aaai*, volume 6, pages 401–408.

J. Scott Long. 1997. *Regression models for categorical and limited dependent variables*, nachdr. edition. Number 7 in Advanced quantitative techniques in the social sciences. Sage Publ, Thousand Oaks, Calif.

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.

Thomas P Minka. 2003. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, pages 1–18.

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.

Philipp Moritz, Robert Nishihara, and Michael Jordan. 2016. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial intelligence and statistics*, pages 249–258. PMLR.

Todd G Nick and Kathleen M Campbell. 2007. Logistic regression. *Topics in biostatistics*, pages 273–301. Publisher: 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 Hajic˘, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

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

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12(null):2825–2830. Publisher: JMLR.org.

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.

James Pustejovsky, Jose Castano, Robert Ingria, Roser Sauri, Robert Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir Radev. 2003. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *New Directions in Question Answering Volume*, volume 3, pages 28–34.

James Pustejovsky, Marc Verhagen, Roser Sauri, Jessica Littman, Robert Gaizauskas, Graham Katz, Inderjeet Mani, Robert Knippen, and Andrea Setzer, editors. 2006. *TimeBank 1.2*. Linguistic Data Consortium. Lead Discovery Center LDC, Massachusetts.

A. P. Sistla and E. M. Clarke. 1985. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749.

Lucien Tesnière. 1959. *Éléments de Syntaxe Structurale*. Centre National de la Recherche Scientifique, Paris, France.

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

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):

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

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., *temperature* > 25), actuator states (e.g., *valve_position* $= 1$), or derived signals (e.g., *velocity* ≤ 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; Donze et al., 2013). Bounded time intervals enable precise specification of real-time constraints, which are 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 (De Marneffe et al., 2021; Nivre et al., 2016), grounded in Tesnière's dependency grammar theory (1959), 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 two datasets spanning technical specifications and narrative discourse domains. The primary dataset is the `circuit_total_refined` lifted NL-STL pairs dataset constructed by Chen et al. (2023) evaluation[2, 3], containing 7,002 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 ($\mathbf{G}$, $\mathbf{F}$, $\mathbf{U}$ and bounded variants), diverse syntactic structures, varying proposition complexity (0 to 4 propositions per sentence), and expert-annotated ground truth formulas for evaluation.

The TimeBank 1.2 corpus (Pustejovsky et al., 2003, 2006) is a widely used temporal event annotation resource developed at Brandeis University, containing 183 news articles annotated with temporal information according to the TimeML specification. TimeBank 1.2 provides comprehensive temporal annotations, including TIMEX3 temporal expressions (dates, times, durations), EVENT tags marking eventive predicates, and temporal relations (TLINK) capturing event ordering and temporal anchoring. The corpus represents narrative discourse from the news domain, fundamentally distinct from technical specifications in temporal expression conventions: narrative text encodes temporality through aspectual morphology, syntactic subordination, and implicit temporal ordering rather than the explicit temporal quantifiers characteristic of formal specifications.

For cross-domain validation, we constructed a handcrafted TimeBank 1.2 NL-STL dataset through systematic sampling and annotation. Dataset construction proceeded through four stages: (1) **TimeML Text Extraction** parsed the TimeBank 1.2 corpus to extract natural language sentences while removing temporal relation annotations (TLINK, SLINK, ALINK tags) and preserving EVENT and TIMEX3 content; (2) **Automated Cleanup** removed malformed annotations, HTML entities, metadata patterns, and filtered sentences through length and capitalization heuristics; (3) **Random Sampling** selected 100 sentences through random extraction from 2,624 total corpus sentences; and (4) **Handcrafted NL-STL Pairs** created temporal logic specifications for each sentence, establishing handcrafted sentence-STL pairs for narrative discourse.

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 3.8.7 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 NL2TL dataset (Chen et al., 2023), preprocessing for token alignment, linguistic analysis via dependency parsing, UDR Composition extraction, STL formula construction, and comprehensive 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 7,002 sentence dataset.

A.2.3 Universal Dependencies Parsing

Each sentence undergoes preprocessing to maintain word-span alignment with the ground truth

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

[3]Processed datasets and codes are available at: `https://github.com/Purdue-AIDA3/NL2UD2TL`

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 17 UD relation types). The parser produces dependency trees that capture both local and long-distance relationships crucial to temporal expressions. This structured representation enables the identification of systematic correspondences between syntactic structures and temporal-logic operators.

A.2.4 Universal Dependencies Feature Extraction

The feature extraction pipeline employs SpaCy 3.8.7 to generate Universal Dependencies parses from which we extract binary presence/absence indicators for the 35 theoretically-defined syntactic Classes. Each sentence undergoes dependency parsing to construct directed graphs with 17 UD relation types, capturing both local and long-distance syntactic relationships. The Class detection system traverses dependency trees to identify structural configurations corresponding to temporal expressions (e.g., "always," "eventually," "within"), logical connectives ($\wedge$, $\vee$, $\neg$, $\rightarrow$), and temporal-logical structures. We apply frequency filtering (1% occurrence threshold) to retain 17 Classes for statistical modeling, avoiding numerical instability from rare Classes. This approach encodes which syntactic structures are present in each sentence as binary feature vectors, where 1 indicates the presence of a syntactic structure and 0 indicates its absence, allowing the statistical model to discover probabilistic correspondences between syntactic features and temporal operators through data-driven learning.

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, the logical structure-building process determines operator precedence and scope by analyzing the syntactic hierarchy: temporal operators are applied according to their position in the dependency tree, and logical connectives maintain 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 by traversing the logical structure depth-first, 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 Manual Evaluation

To validate system performance, we conducted a manual evaluation on stratified samples from both datasets. For NL2TL, we performed a 2% stratified subsample from each of the 17 retained Composition classes, yielding 860 instances for comprehensive human review. For TimeBank 1.2, we conducted a 100% manual evaluation of all 100 sentences. Two expert annotators independently verified that each generated STL operator correctly captured the intended temporal semantics. Annotators also rated their confidence for each evaluation using a 3-point inter-rater reliability scale (1=low confidence, 2=moderate confidence, 3=high confidence), providing a measure of annotator certainty regarding the correctness of both the operator assignment and the complete formula generation.

A.3 Cross-Domain Validation

Cross-domain validation employs the handcrafted TimeBank 1.2-STL dataset to assess whether syntax-semantics correspondences learned from technical specifications generalize to narrative discourse. We developed a two-stage STL lifting pipeline that emulates the methodology of Chen et al. (2023) to transform narrative temporal expressions into formal STL formulas via systematic predicate abstraction. The pipeline processes the `TimeBank 1.2_NL-STL.json` dataset containing 100 handcrafted sentence-STL pairs: (1) **Predicate Name Extraction** uses regex pattern matching to identify all unique predicate function names from ground truth STL formulas, extracting predicate identifiers while discarding argument structures; (2)

Lifted Formula Construction performs systematic abstraction to propositional variables through deterministic mapping (predicate $i \mapsto$ `prop_`i), creating predicate-to-proposition mappings while preserving complete STL Operator structure, temporal bounds, and logical composition. The pipeline outputs lifted formulas with UTF-8 encoded Unicode operators alongside tokenized sentence representations for downstream parsing analysis.

Following lifting, the TimeBank 1.2 sample underwent identical processing to the NL2TL dataset: UD parsing via SpaCy 3.8.7 with the same 17-Composition taxonomy, binary feature vector extraction, and co-occurrence matrix computation. Cross-domain evaluation applies keyword-based pattern-detection rules to TimeBank 1.2 for composition detection and operator generation, testing whether NL2TL-derived correspondences transfer to narrative discourse. Separately, in-domain multinomial logistic regression training exclusively on TimeBank 1.2 data enables comparison of statistical correspondences across domains, revealing whether narrative temporal structures encode operators through fundamentally different Composition-Operator mappings. This parallel methodology isolates domain transfer effects from procedural differences, enabling direct comparison of syntax-semantics correspondences across technical and narrative temporal reasoning domains while testing whether the Universal Dependency Composition taxonomy encodes temporal logic operators independently of domain-specific vocabulary or discourse structure. We designated 'None' (the absence of a temporal operator) as the reference category, serving as the natural baseline for coefficient interpretation in multinomial logistic regression.

A.4 Evaluation of UDR and STL Relationships

The evaluation framework employs multiple complementary metrics to assess UD-STL correspondences. Exact match accuracy uses advanced formula normalization and handles operator equivalences and structural variations, providing a binary assessment of formula correctness. Similarity scores via `SequenceMatcher` provide gradient correctness measures (0 to 1), enabling fine-grained analysis of partial matches. Component-wise evaluation separately assesses temporal operators, logical connectives, atomic propositions, and temporal bounds, isolating specific aspects of the syntax-semantics mapping. Error severity classification categorizes results 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 ($0.6 - 0.8$), and near matches (> 0.8).

A.4.1 Regression Analyses

To examine probabilistic relationships between UDR Compositions and STL Operators, we applied two complementary sentence-level logistic regression approaches (N = 7,002 sentences, 80/20 train-test split). First, binary logistic regression with a One-vs-Rest (OvR) strategy created separate models for each STL Operator class, treating each as an individual classification problem against all other operators (Nick and Campbell, 2007). Second, multinomial logistic regression (MNL) simultaneously modeled all operator classes within a single framework, following established protocols for nominal categorical outcomes (Long, 1997; Brunswicker and Haefliger, 2025).

Both models used identical feature sets: binary vectors encoding the presence or absence of each of the 17 Classes per sentence. Target variables consisted of STL Operator classes extracted from ground truth annotations for the binary approach (5 classes) and from our STL output for the multinomial model (10 classes). Model implementation relied on scikit-learn's `LogisticRegression` with the `lbfgs` solver (Moritz et al., 2016), 1,000 maximum iterations, and balanced class weights to address substantial operator frequency imbalances (Lee et al., 2006; Minka, 2003; Moritz et al., 2016; Pedregosa et al., 2011). The binary model specified `multi_class='ovr'`, while the multinomial model used `multi_class='multinomial'`.

Dataset partitioning followed an 80/20 train-test split with fixed random seeds for reproducibility. The binary approach removed stratification due to rare classes with insufficient samples, while the multinomial model maintained stratified sampling to preserve class distributions. Rather than relying on predetermined Class-operator mappings, both approaches discovered probabilistic relationships through coefficient estimation. We report coefficients and odds ratios where positive values indicate increased operator probability when the pattern is present, while negative values indicate decreased probability (Brunswicker and Haefliger, 2025). For the multinomial model, we designated F (Eventually) as the reference category given its frequency dominance (44.7%, N=2,503), provid-

ing a natural baseline for coefficient interpretation across all other operators.

A.4.2 Additional Tables

UDR Comp. No.	Inst.	Cor. Op.	Acc. (%)	Conf.
1	64	64	100.00	2.61
2	82	65	79.27	2.75
3	17	13	76.47	2.43
4	62	53	85.48	2.63
5	59	55	93.22	2.62
6	38	36	94.74	2.47
7	5	4	80.00	2.52
8	143	124	86.71	2.75
9	20	19	95.00	2.56
10	14	11	78.57	2.33
11	78	68	87.18	2.64
12	125	104	83.20	2.80
13	31	29	93.55	2.87
14	80	62	77.50	2.79
15	3	2	66.67	2.48
16	15	11	73.33	2.59
17	24	22	91.67	2.87
Total	**860**	**742**	**86.28%**	**2.63**

Table 7: **NL2TL:** Composition generation accuracy analysis on a $\sim$2.00% stratified subsample on all occurrences. *Note:* Comp. No. = Composition Number; Inst. = Instances; Cor. Op. = Correct Operators; Acc. = Accuracy (%); Conf. = Confidence (inter-rater score of 1 to 3).

STL Operator	Count	%
F (Eventually)	663	3.25
$F_{[0,k]}$ (Bounded Within)	2,477	12.14
G (Always)	5,267	25.81
$G_{[0,k]}$ (Bounded For)	2,623	12.85
U (Until)	345	1.69
$\rightarrow$ (If-Then Implication)	4,424	21.68
$\wedge$(Conjunction)	3,331	16.32
$\vee$ (Disjunction)	1,280	6.27
Total	**20,410**	**100.0%**

Table 8: **NL2TL:** STL Operator distribution in dataset (N=20,410)

		Predicted False	Predicted True	Total
Actual	**False**	476	171	647
	True	189	565	754
	Total	665	736	**1,401**

Table 9: **NL2TL:** Logistic regression confusion matrix (n=1,401).

Composition	Coefficient	Predictive Direction
Compositions Favoring Match (Positive Coefficients)		
Become/Change Rise	+0.317	Higher presence; more likely match
Specification Conjunction	+0.134	Higher presence; more likely match
Universal Quantification	+0.108	Higher presence; more likely match
When First Rise	+0.038	Higher presence; more likely match
Propositional Negation	+0.034	Higher presence; more likely match
Negated Always	+0.014	Higher presence; more likely match
Compositions Favoring No Match (Negative Coefficients)		
Conjunction	-2.673	Higher presence; less likely match
If-Then Implication	-2.526	Higher presence; less likely match
Until	-0.934	Higher presence; less likely match
Eventually	-0.775	Higher presence; less likely match
Bounded For	-0.755	Higher presence; less likely match
Bounded Within	-0.629	Higher presence; less likely match
Disjunction	-0.466	Higher presence; less likely match
Bounded After	-0.431	Higher presence; less likely match
Existential Quantification	-0.353	Higher presence; less likely match
Always	-0.215	Higher presence; less likely match
Conditional Formula Selection	-0.067	Higher presence; less likely match

Table 10: **NL2TL:** Logistic regression coefficients. Composition effects on exact match probability. Positive coefficients increase match likelihood; negative coefficients decrease it. Ordered by coefficient magnitude within each group.

Actual	Predicted F^*	$F_{[0,k]}$	$F_{[k,\infty]}$	G	$G_{[0,k]}$	None	U	$\neg G$	$\leftarrow$	$\uparrow$
F^*	**626**	0	0	0	0	0	0	0	0	0
$F_{[0,k]}$	0	**75**	13	12	38	1	0	0	7	4
$F_{[k,\infty]}$	0	31	**196**	43	5	0	2	1	2	15
G	0	20	3	**107**	21	9	14	1	3	1
$G_{[0,k]}$	0	4	2	0	**50**	0	0	0	0	0
None	0	0	0	0	0	**17**	7	0	0	0
U	0	0	0	0	0	1	**16**	0	0	0
$\neg G$	0	0	0	0	0	0	0	**3**	0	0
$\uparrow$	0	0	0	0	0	0	0	0	**14**	1
$\rightarrow$	0	0	0	0	0	0	0	0	0	**36**

Table 11: **NL2TL:** Multinomial logistic regression confusion matrix (Test Set, n=1,401). Rows represent actual STL Operators, columns represent predicted operators. Diagonal values (bold) indicate correct predictions. $*F$ serves as the baseline reference category in the multinomial logistic regression model.

STL Operator	Precision	Recall	F1	Support
F^*	1.00	1.00	1.00	2,503
$F_{[0,k]}$	0.63	0.55	0.59	598
$F_{[k,\infty]}$	0.93	0.67	0.78	1,179
G	0.72	0.65	0.68	715
$G_{[0,k]}$	0.49	0.95	0.64	223
None	0.49	0.62	0.55	97
U	0.45	0.94	0.61	69
$\neg G$	0.45	1.00	0.62	14
$\uparrow$	0.48	0.81	0.60	62
$\rightarrow$	0.56	0.99	0.71	141
Macro Avg	0.62	0.82	0.68	5,601
Weighted Avg	0.85	0.82	0.83	5,601
Overall Accuracy: 82.49%				

Table 12: **NL2TL:** Training set performance (n=5,601). Per-operator classification metrics. $*F$ serves as the baseline reference category. Weighted averages account for class imbalance.

STL Operator	Pos. UDR-C No. (Coef. | OR)	Neg. UDR-C No. (Coef. | OR)
F^*	2 (9.1029 | 8,981.2458)	12 (-0.1770 | 0.8378)
$F_{[0,k]}$	12 (3.4892 | 32.7597)	3 (-1.7528 | 0.1733)
$F_{[k,\infty]}$	8 (3.4732 | 32.2397)	2 (-1.6619 | 0.1898)
G	1 (4.9611 | 142.7503)	16 (-1.5512 | 0.2120)
$G_{[0,k]}$	11 (6.2521 | 519.0996)	1 (-1.9186 | 0.1468)
U	3 (5.2661 | 193.6585)	8 (-1.7446 | 0.1747)
$\neg G$	7 (7.3607 | 1,572.9294)	1 (-2.2216 | 0.1084)
$\uparrow$	13 (3.4939 | 32.9140)	11 (-2.6857 | 0.0682)
$\rightarrow$	8 (4.8546 | 128.3289)	9 (-3.7228 | 0.0242)
None (No Temporal Operator)	14 (0.8475 | 2.3338)	8 (-4.9292 | 0.0072)

Table 13: **NL2TL:** Top and Bottom Ranked Composition Predictors per STL Operator (Coefficient | Odds Ratio). *The $F\downarrow$ was used as the baseline. $*F$ Represents the baseline measurement for the MNL.

Rank	F^* (Coef. | OR)	$F_{[0,k]}$ (Coef. | OR)	$F_{[k,\infty]}$ (Coef. | OR)
1	UDR-C No. 2 (+9.10 | 8,981)	UDR-C No. 12 (+3.49 | 32.76)	UDR-C No. 8 (+3.47 | 32.24)
2	1 (+1.70 | 5.46)	1 (+3.28 | 26.59)	16 (+3.42 | 30.59)
3	10 (+1.29 | 3.63)	16 (+2.48 | 11.99)	14 (+2.86 | 17.51)
4	16 (+1.25 | 3.50)	10 (+2.46 | 11.71)	1 (+2.58 | 13.20)
5	11 (+1.21 | 3.35)	14 (+2.21 | 9.08)	10 (+1.72 | 5.58)
6	13 (+0.59 | 1.80)	11 (+1.76 | 5.82)	6 (+0.33 | 1.39)
7	9 (+0.56 | 1.76)	9 (+0.25 | 1.29)	9 (+0.31 | 1.37)
8	14 (+0.48 | 1.61)	7 (+0.10 | 1.11)	17 (+0.18 | 1.20)
9	15 (+0.25 | 1.28)	4 (+0.09 | 1.10)	7 (+0.07 | 1.07)
10	17 (+0.17 | 1.18)	17 (+0.23 | 1.25)	15 (-0.02 | 0.98)
11	8 (+0.10 | 1.11)	15 (+0.31 | 1.36)	4 (-0.04 | 0.96)
12	3 (+0.00 | 1.00)	5 (-0.14 | 0.87)	5 (-0.03 | 0.97)
13	4 (+0.03 | 1.03)	13 (-0.30 | 0.74)	11 (-0.20 | 0.82)
14	5 (-0.00 | 1.00)	6 (-0.35 | 0.71)	13 (-0.24 | 0.78)
15	6 (+0.01 | 1.01)	8 (-1.41 | 0.25)	3 (-0.11 | 0.90)
16	7 (-0.09 | 0.91)	3 (-1.75 | 0.17)	12 (-1.50 | 0.22)
17	12 (-0.18 | 0.84)	2 (-1.69 | 0.18)	2 (-1.66 | 0.19)
Rank	G (Coef. | OR)	$G_{[0,k]}$ (Coef. | OR)	U (Coef. | OR)
1	UDR-C No. 1 (+4.96 | 142.8)	UDR-C No. 11 (+6.25 | 519.1)	UDR-C No. 3 (+5.27 | 193.7)
2	11 (+1.87 | 6.50)	13 (+1.70 | 5.46)	14 (+0.93 | 2.54)
3	13 (+0.89 | 2.43)	9 (+0.62 | 1.87)	12 (+0.46 | 1.59)
4	3 (+0.75 | 2.12)	7 (+0.25 | 1.28)	5 (+0.38 | 1.46)
5	7 (+0.39 | 1.47)	6 (+0.14 | 1.15)	17 (+0.02 | 1.02)
6	9 (+0.10 | 1.10)	4 (+0.12 | 1.13)	15 (-0.01 | 0.99)
7	4 (+0.22 | 1.25)	17 (-0.02 | 0.98)	10 (-0.26 | 0.77)
8	17 (+0.00 | 1.00)	5 (-0.08 | 0.93)	16 (-0.30 | 0.74)
9	5 (+0.03 | 1.03)	15 (-0.12 | 0.88)	4 (-0.23 | 0.79)
10	6 (-0.08 | 0.92)	8 (-0.27 | 0.77)	9 (-0.36 | 0.70)
11	15 (-0.29 | 0.75)	12 (-0.63 | 0.53)	2 (-0.28 | 0.76)
12	12 (-0.73 | 0.48)	14 (-0.73 | 0.48)	6 (-0.97 | 0.38)
13	14 (-0.74 | 0.48)	10 (-0.92 | 0.40)	13 (-0.79 | 0.45)
14	2 (-1.05 | 0.35)	3 (-1.06 | 0.35)	C (-0.84 | 0.43)
15	8 (-1.23 | 0.29)	16 (-1.03 | 0.36)	11 (-0.88 | 0.41)
16	10 (-1.52 | 0.22)	2 (-1.13 | 0.32)	1 (-1.37 | 0.25)
17	16 (-1.55 | 0.21)	1 (-1.92 | 0.15)	8 (-1.74 | 0.17)
Rank	$\neg G$ (Coef. | OR)	$\uparrow$ (Coef. | OR)	$\rightarrow$ (Coef. | OR)
1	UDR-C No. 7 (+7.36 | 1,573)	UDR-C No. 13 (+3.49 | 32.91)	UDR-C No. 8 (+4.85 | 128.3)
2	6 (+0.53 | 1.70)	9 (+2.91 | 18.33)	4 (+0.53 | 1.70)
3	9 (+0.19 | 1.21)	8 (+1.06 | 2.88)	6 (+0.13 | 1.14)
4	8 (+0.09 | 1.09)	5 (+0.85 | 2.35)	15 (-0.02 | 0.98)
5	15 (-0.01 | 0.99)	6 (+0.37 | 1.45)	12 (-0.08 | 0.93)
6	10 (-0.07 | 0.93)	17 (+0.30 | 1.35)	17 (-0.16 | 0.85)
7	5 (-0.26 | 0.77)	4 (-0.51 | 0.60)	2 (-0.43 | 0.65)
8	4 (-0.38 | 0.68)	12 (-0.09 | 0.91)	5 (-0.62 | 0.54)
9	2 (-0.31 | 0.73)	10 (-0.74 | 0.48)	3 (-0.62 | 0.54)
10	3 (-0.05 | 0.95)	16 (-1.31 | 0.27)	13 (-0.70 | 0.50)
11	12 (-0.64 | 0.53)	2 (-1.35 | 0.26)	10 (-0.94 | 0.39)
12	17 (-0.69 | 0.50)	14 (-1.57 | 0.21)	16 (-1.94 | 0.14)
13	16 (-0.90 | 0.41)	7 (-2.16 | 0.12)	14 (-2.25 | 0.11)
14	13 (-1.12 | 0.33)	3 (-2.27 | 0.10)	7 (-2.56 | 0.08)
15	14 (-2.03 | 0.13)	1 (-2.51 | 0.08)	1 (-3.10 | 0.05)
16	11 (-1.89 | 0.15)	14 (-1.57 | 0.21)	11 (-2.37 | 0.09)
17	1 (-2.22 | 0.11)	11 (-2.69 | 0.07)	9 (-3.72 | 0.02)
Rank	**None** (Coef. | OR)		
1	UDR-C No. 14 (+0.85 | 2.33)		
2	4 (+0.17 | 1.18)		
3	15 (-0.02 | 0.98)		
4	17 (-0.03 | 0.97)		
5	12 (-0.11 | 0.89)		
6	16 (-0.12 | 0.88)		
7	6 (-0.13 | 0.88)		
8	5 (-0.13 | 0.88)		
9	3 (-0.15 | 0.86)		
10	9 (-0.87 | 0.42)		
11	10 (-1.01 | 0.36)		
12	2 (-1.21 | 0.30)		
13	1 (-1.41 | 0.24)		
14	7 (-2.52 | 0.08)		
15	11 (-3.06 | 0.05)		
16	13 (-3.50 | 0.03)		
17	8 (-4.93 | 0.01)		

Table 14: **NL2TL:** Ranked UDR Composition predictors per STL Operator (Coefficient | Odds Ratio). Note: Coef. = coefficient (log-odds); OR = odds ratio (exponentiated coefficient). Positive coefficients (OR $>$ 1) increase operator probability; negative coefficients (OR $<$ 1) decrease it.

Pattern No.	Pattern Name	UDR Comp.	No. of Inst.	%
1	Always	advmod("always")	5	3.65
2	Eventually	advmod("eventually")	0	N/A
3	Until	mark("until")+advcl	1	0.73
4	Conjunction	cc("and")+conj	34	24.82
5	Disjunction	cc("or")+conj	9	6.57
6	Prop. Negation	advmod("not")	20	14.60
7	Negated Always	advmod("never")	0	N/A
8	If-Then Impl.	mark("if")+advmod("then")	3	2.19
9	Become/Change	compound("become"/"change")	1	0.73
10	Bounded After	mark("after")+nummod	3	2.19
11	Bounded For	mark("for")+nummod	17	12.41
12	Bounded Within	mark("within")+nummod	27	19.71
13	When First	mark("when")+advmod("first")	0	N/A
14	Universal Quant.	det("all")+nsubj	8	5.84
15	Existential Quant.	det("a")+acl:relcl	1	0.73
16	Cond. Formula Sel.	advcl+mark("if")	0	N/A
17	Spec. Conjunction	obj("both")+nsubj	8	5.84
Total			**137**	**100.00%**

Table 15: **TimeBank 1.2:** UDR Composition frequency and summary

STL Operator	Count	%
$\wedge$ (Conjunction)	68	36.76
$G_{[0,k]}$ (Bounded For)	44	23.78
$\rightarrow$ (If-Then Implication)	19	10.27
$\neg$ (Negation)	18	9.73
$F_{[0,k]}$ (Bounded Within)	16	8.65
$F_{[k,\infty]}$ (Bounded After)	10	5.41
$\vee$ (Disjunction)	9	4.86
U (Until)	1	0.54
$\uparrow$ (Rise)	0	0.00
$\downarrow$ (Fall)	0	0.00
Total	**185**	**100.00%**

Table 16: **TimeBank 1.2:** STL Operator distribution in dataset (N=185)

UDR Comp. No.	Inst.	Cor. Op.	Acc. (%)	Conf.
1	5	5	100.00	3.00
2	0	0	N/A	N/A
3	1	1	100.00	3.00
4	34	19	55.88	2.37
5	9	7	77.78	2.09
6	20	0	0.00	2.45
7	0	0	N/A	N/A
8	3	3	100.00	2.67
9	1	0	0.00	3.00
10	3	0	0.00	2.67
11	17	0	0.00	2.76
12	27	0	0.00	2.44
13	0	0	N/A	N/A
14	8	1	12.50	2.50
15	1	0	0.00	3.00
16	0	0	N/A	N/A
17	8	2	25.00	1.88
Total	**137**	**38**	**27.74%**	**2.60**

Table 17: **TimeBank 1.2:** Composition-Operator pair accuracy across 100 sentences.

Rank	$F_{[k,\infty]}$ (Coef. | OR)	G (Coef. | OR)	$G_{[0,k]}$ (Coef. | OR)
1	UDR-C No. 8 (+2.43 | 11.35)	UDR-C No. 1 (+2.64 | 13.94)	UDR-C No. 9 (+2.72 | 15.13)
2	3 (+0.79 | 2.21)	2 (0.00 | 1.00)	5 (+0.38 | 1.47)
3	2 (0.00 | 1.00)	12 (-0.02 | 0.98)	13 (+0.19 | 1.21)
4	12 (-0.01 | 0.99)	7 (-0.02 | 0.98)	7 (+0.16 | 1.18)
5	4 (-0.03 | 0.97)	4 (-0.06 | 0.94)	11 (+0.14 | 1.15)
6	7 (-0.04 | 0.96)	13 (-0.11 | 0.90)	2 (0.00 | 1.00)
7	13 (-0.09 | 0.91)	11 (-0.11 | 0.89)	12 (-0.03 | 0.97)
8	11 (-0.15 | 0.86)	5 (-0.26 | 0.77)	4 (-0.07 | 0.93)
9	5 (-0.27 | 0.77)	6 (-0.32 | 0.73)	6 (-0.16 | 0.85)
10	9 (-0.38 | 0.68)	3 (-0.40 | 0.67)	10 (-0.19 | 0.82)
11	6 (-0.40 | 0.67)	8 (-0.48 | 0.62)	8 (-0.41 | 0.67)
12	1 (-0.48 | 0.62)	9 (-0.49 | 0.61)	1 (-0.49 | 0.61)
13	10 (-0.62 | 0.54)	10 (-0.58 | 0.56)	3 (-0.50 | 0.61)
Rank	$\rightarrow$ (Coef. | OR)	**None*** (Coef. | OR)	
1	UDR-C No. 6 (+2.03 | 7.61)	UDR-C No. 5 (+0.34 | 1.40)	
2	10 (+1.35 | 3.85)	11 (+0.26 | 1.29)	
3	3 (+0.75 | 2.12)	4 (+0.22 | 1.24)	
4	2 (0.00 | 1.00)	13 (+0.15 | 1.16)	
5	12 (-0.00 | 1.00)	12 (+0.05 | 1.06)	
6	7 (-0.02 | 0.98)	10 (+0.05 | 1.05)	
7	4 (-0.06 | 0.94)	2 (0.00 | 1.00)	
8	11 (-0.13 | 0.88)	7 (-0.08 | 0.92)	
9	13 (-0.14 | 0.87)	3 (-0.65 | 0.52)	
10	5 (-0.20 | 0.82)	6 (-1.15 | 0.32)	
11	1 (-0.27 | 0.76)	8 (-1.17 | 0.31)	
12	8 (-0.37 | 0.69)	1 (-1.39 | 0.25)	
13	9 (-0.43 | 0.65)	9 (-1.42 | 0.24)	

Table 18: **TimeBank 1.2:** Ranked UDR Composition predictors per STL Operator (Coefficient | Odds Ratio). Note: Coef. = coefficient (log-odds); OR = odds ratio (exponentiated coefficient). Positive coefficients (OR $>$ 1) increase operator probability; negative coefficients (OR $<$ 1) decrease it.

	Predicted				
Actual	None*	$F_{[k,\infty]}$	$G_{[0,k]}$	U	$\rightarrow$
None*	**12**	0	0	0	0
$F_{[k,\infty]}$	0	**2**	0	0	0
$G_{[0,k]}$	0	0	**4**	0	0
U	1	0	0	**0**	0
$\rightarrow$	1	0	0	0	**0**

Table 19: **TimeBank 1.2:** Confusion matrix (Test Set, n=20). Rows represent actual STL Operators, columns represent predicted operators. Diagonal values (bold) indicate correct predictions. *None serves as the baseline reference category in the multinomial logistic regression model. Only 4 operators appear in test set (N/A indicates not present).

STL Operator	Precision	Recall	F1	Support
None*	1.00	1.00	1.00	62
$F_{[k,\infty]}$	1.00	1.00	1.00	1
G	1.00	1.00	1.00	3
$G_{[0,k]}$	1.00	1.00	1.00	13
$\rightarrow$	1.00	1.00	1.00	1
Macro Avg	1.00	1.00	1.00	80
Weighted Avg	1.00	1.00	1.00	80
Overall Accuracy: 100.00%				

Table 20: **TimeBank 1.2:** Training set performance (n=80). Per-operator classification metrics. *None serves as the baseline reference category. Only 4 operators appear in training set (N/A indicates not present). Perfect accuracy indicates model memorization of training patterns.

STL Operator	Precision	Recall	F1	Support
None*	0.86	1.00	0.92	12
$F_{[k,\infty]}$	1.00	1.00	1.00	2
$G_{[0,k]}$	1.00	1.00	1.00	4
U	0.00	0.00	0.00	1
$\rightarrow$	0.00	0.00	0.00	1
Macro Avg	0.57	0.60	0.58	20
Weighted Avg	0.81	0.90	0.85	20
Overall Accuracy: 90.00%				

Table 21: **TimeBank 1.2:** Test set performance (n=20). Per-operator classification metrics on held-out data. Training-test gap: 10.00% (100.00% vs 90.00%), indicating potential overfitting. *None serves as the baseline reference category. Only 4 operators appear in test set (N/A indicates not present).

A.4.3 Heatmaps

UDR Composition	F	$F_{[0,k]}$	$F_{[k,\infty]}$	G	$G_{[0,k]}$	U	¬G	↑	→	↓
1. advmod("always")	1169	798	1909	3014	1425	231	104	888	2580	75
2. advmod("eventually")	3129	1092	1727	1251	1527	233	48	1145	1993	69
3. mark("until")+advcl	431	131	408	515	197	449	31	156	535	18
4. cc("and")+conj	1310	853	1618	1659	1409	222	84	1033	2030	57
5. cc("or")+conj	1262	701	1456	1466	1168	183	79	1009	1911	86
6. advmod("not")	746	424	1023	944	728	102	50	625	1311	45
7. advmod("never")	48	41	96	121	91	11	206	72	136	2
8. mark("if")+advmod("then")	1993	1099	3259	2720	1966	288	136	1222	4424	129
9. compound("become"/"change")	515	233	595	469	398	61	25	935	815	10
10. mark("after")+nummod	549	226	658	356	384	54	25	159	508	15
11. mark("for")+nummod	1527	1190	1565	1706	3006	121	91	1470	1966	61
12. mark("within")+nummod	2596	1840	2418	2692	2397	430	154	1811	3377	110
13. mark("when")+advmod("first")	718	579	329	626	1146	25	47	1540	426	22
14. det("all")+nsubj	1456	1289	2232	2003	2489	175	111	1402	2566	77
15. det("a")+acl:relcl	97	43	57	52	50	14	3	34	75	1
16. advcl+mark("if")	331	159	725	453	320	44	23	195	727	16
17. obj("both")+nsubj	612	335	589	590	495	83	49	403	724	22

Table 22: **NL2TL:** UDR Compositions × STL Operators co-occurrence heatmap

UDR Composition	F	$F_{[0,k]}$	$F_{[k,\infty]}$	G	$G_{[0,k]}$	U	¬G	↑	→	↓
1. advmod("always")	0	0	0	3	0	0	0	0	0	0
2. advmod("eventually")	0	0	0	0	0	0	0	0	0	0
3. mark("until")+advcl	0	0	0	0	0	1	0	0	0	0
4. cc("and")+conj	0	0	1	1	4	1	0	0	1	0
5. cc("or")+conj	0	0	0	0	0	1	0	0	0	0
6. advmod("not")	0	0	0	0	4	1	0	0	1	0
7. advmod("never")	0	0	0	0	0	0	0	0	0	0
8. mark("if")+advmod("then")	0	0	0	0	1	0	0	0	3	0
9. compound("become"/"change")	0	0	0	0	1	0	0	0	0	0
10. mark("after")+nummod	0	0	3	0	0	0	0	0	0	0
11. mark("for")+nummod	0	0	0	0	17	0	0	0	1	0
12. mark("within")+nummod	0	0	1	0	2	0	0	0	1	0
13. mark("when")+advmod("first")	0	0	0	0	0	0	0	0	0	0
14. det("all")+nsubj	0	0	0	0	1	0	0	0	0	0
15. det("a")+acl:relcl	0	0	0	0	0	0	0	0	0	0
16. advcl+mark("if")	0	0	0	0	0	0	0	0	0	0
17. obj("both")+nsubj	0	0	0	0	2	0	0	0	0	0

Table 23: **TimeBank 1.2:** UDR Compositions × STL Operators co-occurrence heatmap (Subset, Raw Counts)

A.4.4 Figures

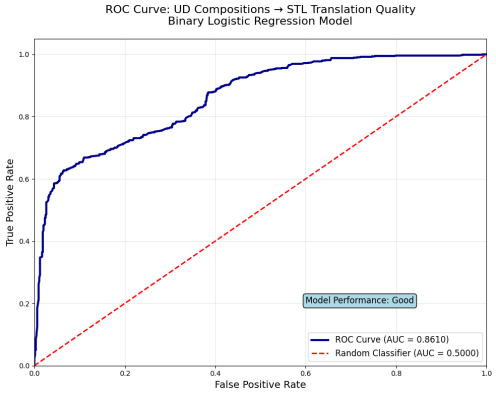

Figure 1: **NL2TL:** ROC Curve. Logistic regression model discriminating exact matches (AUC = 0.86) based on UDR Composition features. Strong performance indicates syntactic patterns are informative predictors of translation quality.

A.5 Acknowledgments

We would like to thank the expert annotators who provided invaluable manual evaluations that validated our system's performance and contributed to the creation of STL annotations for the TimeBank 1.2 corpus sample, enabling our cross-domain analysis. We are grateful to the faculty, staff, researchers, students, and collaborators of AIDA[3] at Purdue University for their ongoing support and insightful discussions that enriched this research. Special appreciation goes to the computational linguistics community for providing the foundational datasets that made this work possible.