

DeepSTL - From English Requirements to Signal Temporal Logic

Authors: Jie He, Ezio Bartocci, Dejan Ničković, Haris Isakovic, Radu Grosu

Conference: ICSE 2022, Pittsburgh



Presented By - Arpita Santra (242110602); Pankaj Kumar Barman (242040404);
Ashish Singh (242110603); Manas Dhaketa (251110046); Amitesh Patra (251110010)

Introduction

- Formal methods use **logic-based specifications** to ensure safety and correctness in **cyber-physical systems (CPS)** through techniques like model checking and runtime verification.
- A major challenge is **translating informal, natural-language requirements** into **formal logic expressions**, which requires significant expertise.
- **Natural Language Processing (NLP)** offers potential to automate this translation, similar to language translation systems.
- However, this faces two key challenges:
 - a. Challenge 1: Lack of available training data.
 - b. Challenge 2: Lack of systematic translation frameworks between natural language and formal logic.

Major contribution

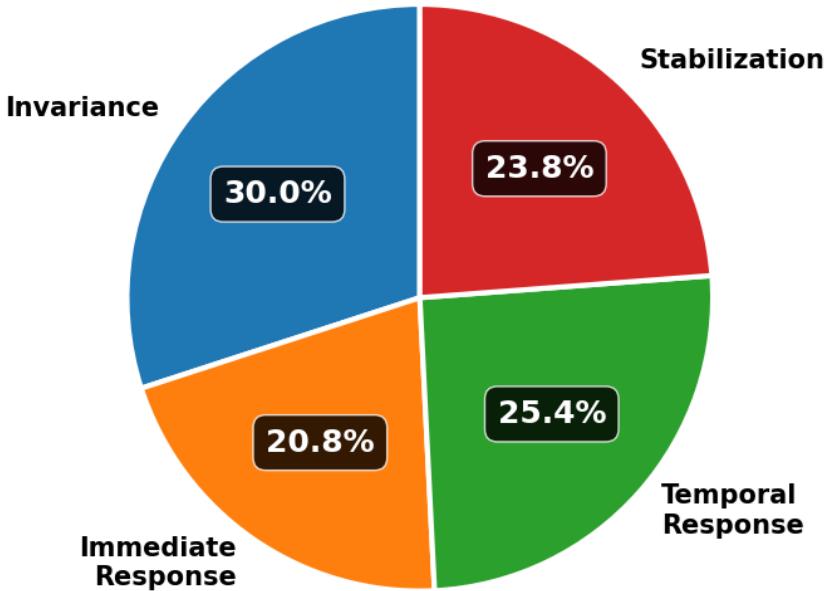
- The paper introduces **DeepSTL**, a method and tool that use **NLP** to translate **English CPS requirements** into **Signal Temporal Logic (STL)** — a formal specification language for CPS
- To develop Deep-STL they address the following five research questions (RQ), the solutions of which are also our main contributions.
 1. What are the characteristics of STL formulas in existing literature?
 2. How can synthetic STL examples be generated?
 3. Can DeepSTL effectively learn from synthetic data?
 4. How well does it generalize to real-world data?
 5. How do different neural architectures compare to Transformers?

Analysis of STL Statistics

Formulas Follow Templates : Most real-world STL specifications are not random, but complex formulas. Instead, over 90% STL specifications fall into four common templates:

- Invariance/Reachability
- Immediate Response
- Temporal response
- Stabilization/Recurrence

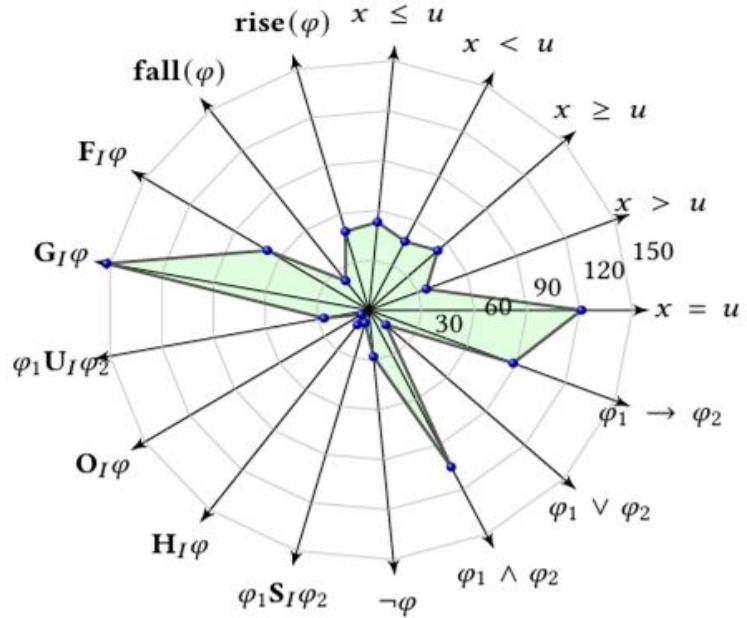
STL-Templates Distribution



Analysis of STL Statistics

Operator Frequency is Skewed: The use of operators is not uniform.

- The **G (Globally/Always)** operator is dominant, appearing in 86.2% of all specifications, as it forms the basis for most invariance and response properties.
- The **F (Finally/Eventually)** operator is the second most common.
- **Future-tense temporal operators** (G, F, U) are used far more frequently than their past-tense counterparts (H, O, S).



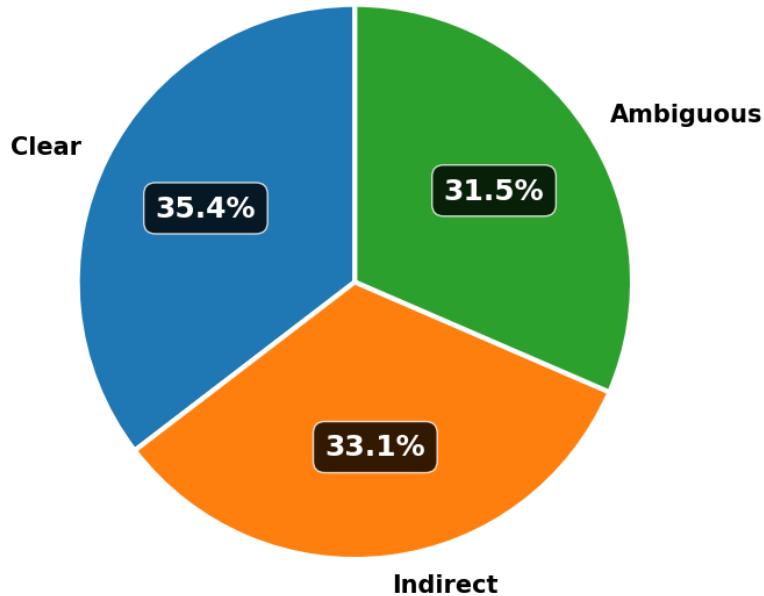
Frequency Distribution of STL Operators

Analysis of NL Specifications

Language Quality is a Major Challenge: The authors found that real-world English requirements are often messy and imprecise. They classified the requirements into three categories:

- **Clear:** A direct, unambiguous translation to STL is possible.
- **Indirect:** Requires an expert to infer implicit context not written in the sentence.
- **Ambiguous:** Lacks key information that must be found in external sources (like diagrams, tables, or by asking an expert).

Classification of English Requirements



Corpus Construction

Restricted STL Fragment:

Not using the full expressive power of Signal Temporal Logic (STL) – instead, we are working with a simplified subset (a *fragment*) of STL formulas that still captures the main behaviors

Simple Phrase (SP)

$$SP := \alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha$$

$$\alpha := x \circ u \mid \neg(x \circ u) \mid \text{rise}(x \circ u) \mid \text{fall}(x \circ u) \mid$$

$$\neg\text{rise}(x \circ u) \mid \neg\text{fall}(x \circ u)$$

$$\circ \in \{\text{<}, \leq, =, \geq, \text{>}\}$$

Basic logical or signal condition

Temporal Phrase (TP)

$$TP := TP' \mid \neg TP' \mid \text{rise } TP' \mid \text{fall } TP' \mid \neg\text{rise } TP' \mid \neg\text{fall } TP'$$

$$TP' := \text{UTO}_I(\alpha) \mid (\alpha)\text{BTO}_I(\alpha)$$

$$\text{UTO} \in \{\text{F, G, O, H}\}, \text{BTO} \in \{\text{U, S}\}, I \in \{t_1, t_2\}$$

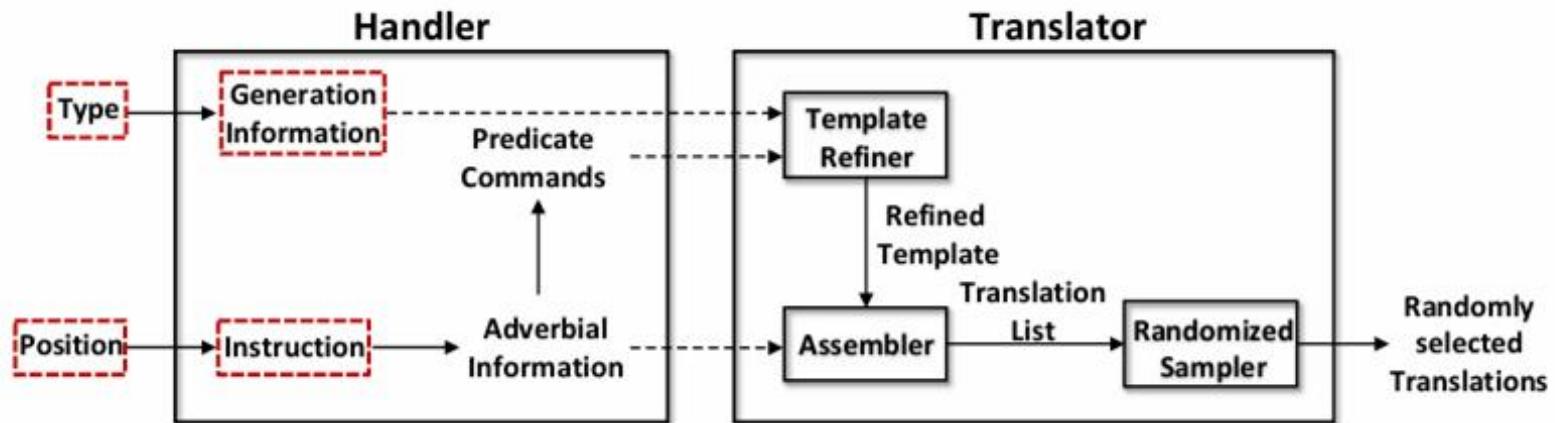
Time-based behavior using temporal operators

Nested Temporal Phrase (NTP)

$$NTP := F_I G_I(\alpha) \mid G_I F_I(\alpha)$$

Combination of temporal operators for complex timing

Corpus Construction



Handler → interprets instructions and creates logical building blocks.

Translator → refines those blocks into full STL formula templates and generates random variations.

Corpus Construction

Translate Temporal Phases

1. x-axis → Temporal Operator

2. y-axis → Temporal Phrase Variations

3. z-axis → Tense

Formal STL formulas are expanded step by step:

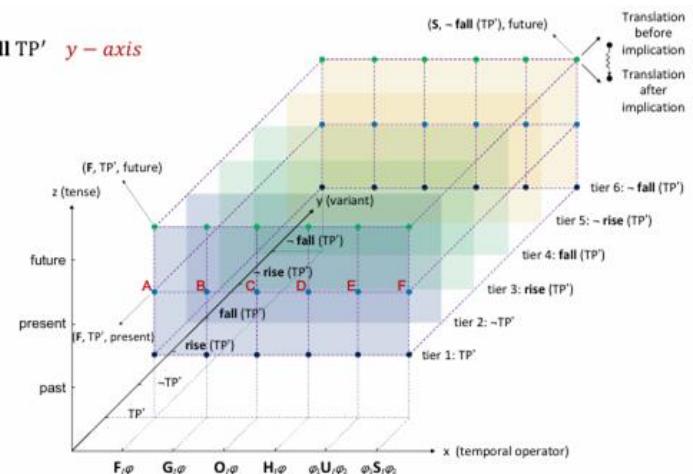
$$TP := TP' \mid \neg TP' \mid \text{rise } TP' \mid \text{fall } TP' \mid \neg \text{rise } TP' \mid \neg \text{fall } TP' \quad y - axis$$

$$TP' := \text{UTO}_I(\alpha) \mid (\alpha) \text{BTO}_I(\alpha) \quad x - axis$$

$$\text{UTO} \in \{\text{F, G, O, H}\}, \text{BTO} \in \{\text{U, S}\}, I \in \{t_1, t_2\}$$

General Strategy:

- Reuse translations of atomic propositions
- Add temporal adverbial modifiers
- Enrich verb tenses

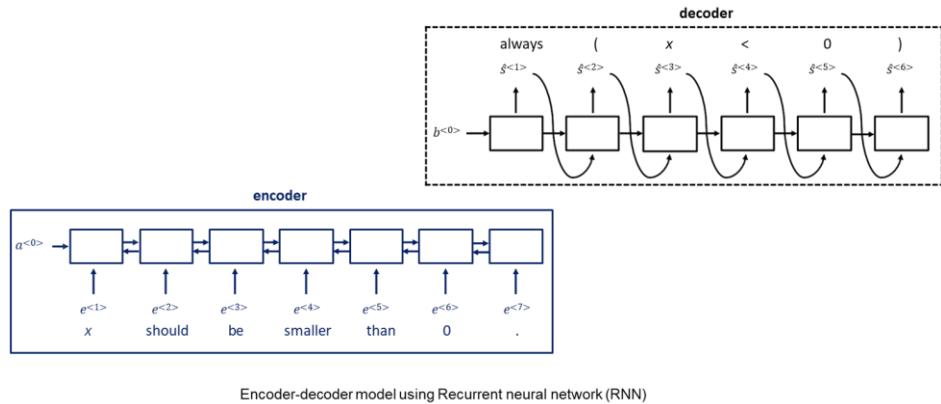


From logic operators

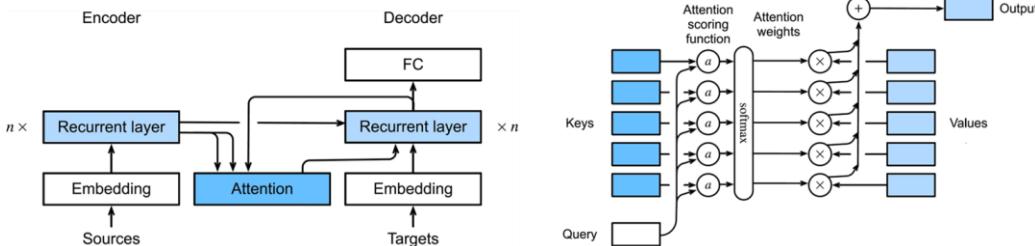
To signal trends

To natural English
with proper tense.

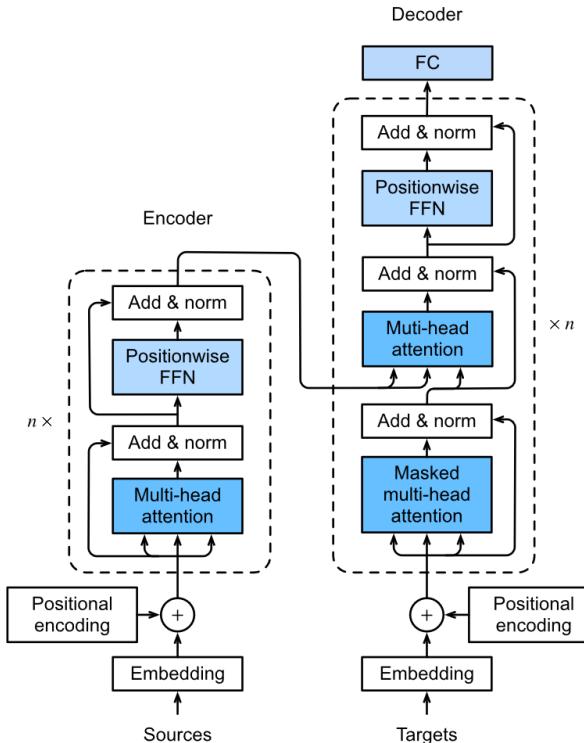
Models



Seq2seq model

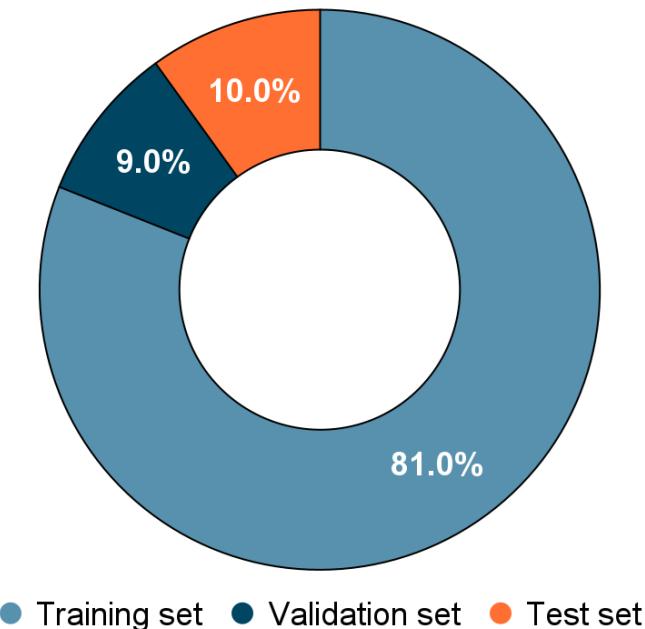


Attention Seq2seq model



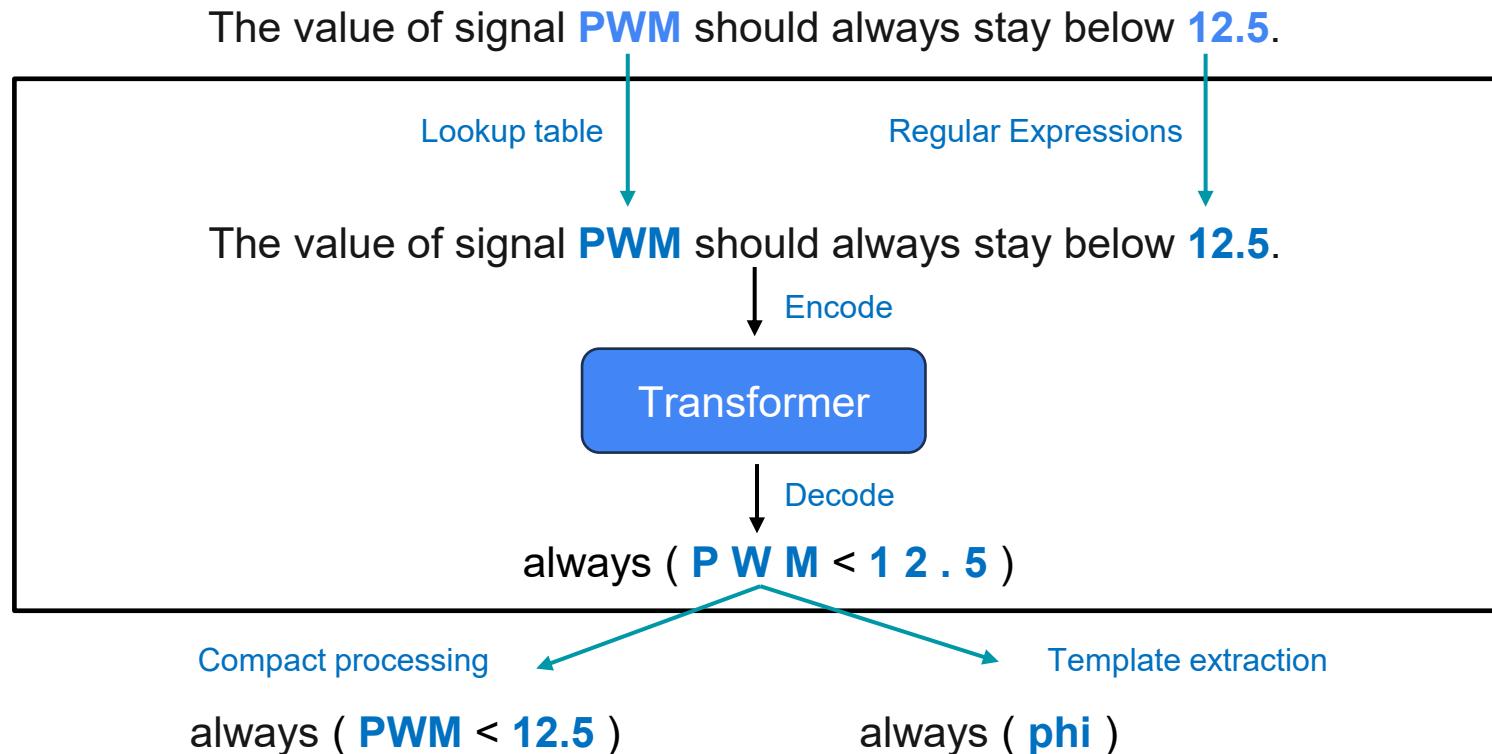
Transformer model

Data Split



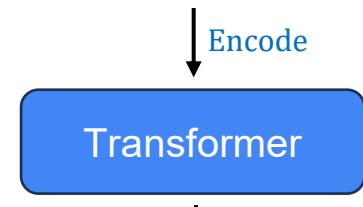
Training set	92,700 English-STL pairs
Validation set	10,800 English-STL pairs
Test set	12,000 English-STL pairs
Total	120,000 English-STL pairs

Training Pipeline



Sample Output

The value of signal PWM should always stay below 12.5.



always (P W M < 1 2 . 5)

Reference sequence:

always (P W M < 1 2 . 5)

Output sequence:

always (P W M < 1 2 . 6)

Formula accuracy: 10/11

Reference template:

always (phi)

Output template:

always (phi)

Template accuracy: 1

DeepSTL Results and Key Observations

Synthetic Test Set

Model	Formula Acc.	Template Acc.	BLEU
Seq2Seq	0.071 ± 0.039	0.207 ± 0.087	0.092 ± 0.036
Att-Seq2Seq	0.977 ± 0.006	0.980 ± 0.006	0.996 ± 0.001
Transformer (DeepSTL)	0.987 ± 0.003	0.995 ± 0.001	0.998 ± 0.001

Input (English): “Whenever Op_Cmd changes to Passive, then in response Spd_Act changes to 0 after at most 500 time units.”

Expected STL Formula: $G(\text{rise}(\text{Op_Cmd} = \text{Passive}) \rightarrow F_{[0-500]}(\text{Spd_Act} = 0))$

DeepSTL (Transformer) Output:  always(rise(Op_Cmd == Passive) → eventually [0:500] (Spd_Act == 0))

DeepSTL Results and Key Observations

Extrapolation (Real English Requirements)

Model	Formula Acc.	Template Acc.
Seq2Seq	0.05	0.16
Att-Seq2Seq	0.56	0.74
Transformer (DeepSTL)	0.71	0.90**

Input (English): “Whenever V_Mot enters the range [1, 12] then, after at most 100 time units, Spd_Act must be in the range [100, 1000].”

Reference STL Formula: $G(\text{rise}(1 \leq V_{\text{Mot}} \leq 12) \rightarrow F_{[0..100]}(100 \leq \text{Spd_Act} \leq 1000))$

DeepSTL (Transformer) Output: always (rise (V_Mot ≥ 1 and V_Mot ≤ 12) → eventually [0:100] (Spd_Act ≥ 100 and Spd_Act ≤ 1000))

Att-Seq2Seq Output: always (rise (V_Mot ≥ 1 and V_Mot ≤ 12) → not (eventually [0:100] (Spd_Act ≥ 100 and Spd_Act ≤ 1000)))

Conclusions and Future Work

- ❖ **Conclusion:** DeepSTL accurately translates English requirements to STL using Transformers—achieving near-perfect accuracy on synthetic data and strong results on real cases.
- ❖ **Future Work:**
Expand real datasets, add context-aware translation, and integrate DeepSTL into industrial design workflows and more domain-specific tasks like healthcare etc.

Thank You!