

西安电子科技大学
XIDIAN UNIVERSITY

Bridging Natural Language and Formal Specification–

Automated Translation of Software Requirements to LTL via Hierarchical Semantics Decomposition Using LLMs

Zhi Ma[1], Cheng Wen[2], Zhexin Su[1], Xiao Liang[1], Cong Tian[1*] , Shengchao Qin[1] and Mengfei Yang[3]
[1]School of Computer Science and Technology, Xidian University, Xi'an, China
[2]Guangzhou Institute of Technology of Xidian University, Guangzhou, China
[3]China Academy of Space Technology, Beijing, China
{mazhi, wencheng, qinshengchao}@xidian.edu.cn, {23031212487, xiaoliang}@stu.xidian.edu.cn,
ctian@mail.xidian.edu.cn, yangmf@bice.org.cn

Background & Motivation

Background

- Natural language requirements are the foundation of aerospace/embedded development.
- Formal specifications (e.g., LTL) are essential for verification and model checking.
- Manual translation from NL to LTL is costly and error-prone.





Background & Motivation

Challenges in Engineers. Requirement is more complex!

- Implicit temporal semantics are hard for LLMs to infer.
- Requirements contain deep nested logic.
- LLMs easily produce "silent failures" without structural validation.

Zero-Shot Prompt: You are an expert in the field of formal verification, proficient in the rules and usage of LTL formulas. Please translate the following natural language requirement into LTL formulas.

Req01: In the inertial navigation valid mode, the output will be set to reverse navigation computation, or the output will be set to low velocity low altitude navigation computation, or the output will be set to high velocity navigation computation, or the output will be set to high altitude navigation computation.	**Req02**: Globally, if the navigation system computes angles using dual inertial navigation modules, the input attitude angle shall be the weighted average of angles from module 1 and module 2 (weights 0.6 and 0.4), unless the system switches to single module or GPS signal is restored.
LTL01 by GPT-4o: G (valid_mode → (output = reverse_nav ∨ output = lowVlowH_nav ∨ output = highV_nav ∨ output = highH_nav))	**LTL02 by GPT-4o:** G ((Dual_INU_Active ∧ ¬Single_Module ∧ ¬GPS_Restored) → (Input_Attitude_Angle = 0.6 * Angle_Module1 + 0.4 * Angle_Module2))
Ground Truth: G (valid_mode → (F output = reverse_nav ∨ F output = lowVlowH_nav ∨ F output = highV_nav ∨ F output = highH_nav))	Ground Truth: G (Dual_INU_Active → (Input_Attitude_Angle = 0.6 * Angle-Module1+ 0.4 * Angle_Module2) U (Single_Module ∨ GPS_Restored))

Fig. 1. **Examples showcasing the discrepancy between LLM generated LTL and ground truth for complex requirements.** The red-circle indicates errors in the LLM's output due to missing details, contrasted with the manually corrected specification.

Contributions

Key Idea - REQ2LTL. Completed in two phases: NL → OnionL → LTL

- OnionL: a structured intermediate representation.
- Req2OnionL: LLM-guided hierarchical semantic decomposition.
- OnionL2LTL: deterministic rule-based LTL synthesis.
- Achieves 88.4% semantic accuracy and 100% syntactic correctness on real aerospace requirements.

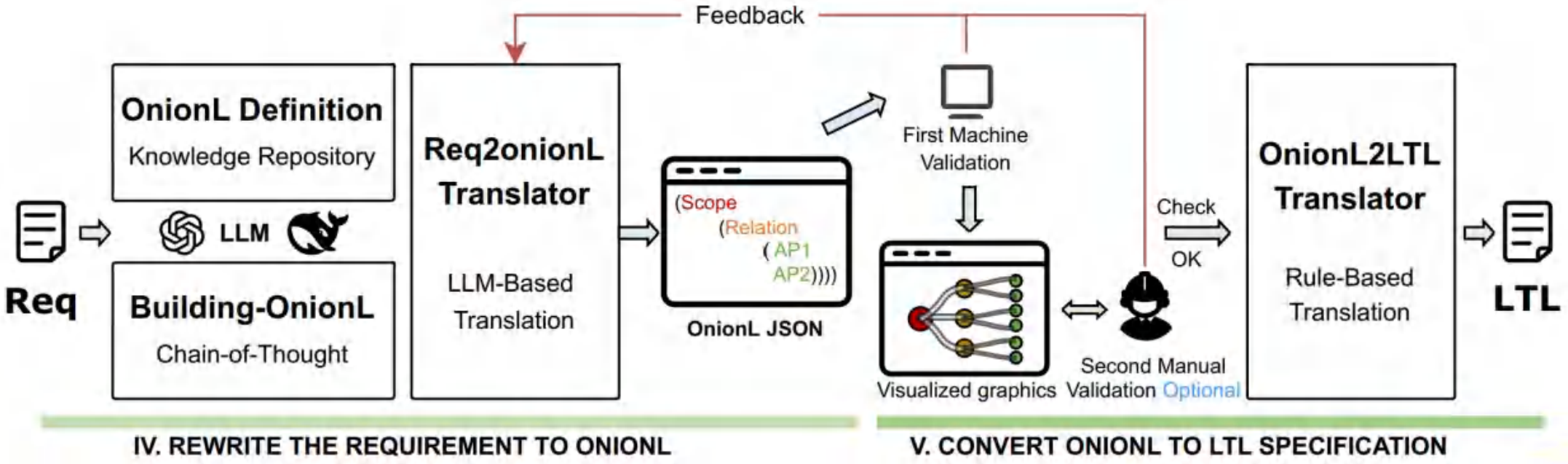
Overview of REQ2LTL

Req2OnionL (LLM Hierarchical Semantic Decomposition)

- Identify global temporal scope
- Recursively decompose logical structures
- Extract and normalize atomic propositions

OnionL2LTL (Regularized Formal Transformation)

- Structure Verification (Machine + Optional Human)
- Regularized Composition Unambiguous LTL



Why is OnionL?

Problem: LLMs extract local semantics well but cannot reliably construct global logical structure, causing missing operators and broken nesting in NL→LTL.

- Industrial NL often hides temporal cues ("unless", "will be set").

→ **LLM frequently misses F / U / X.**

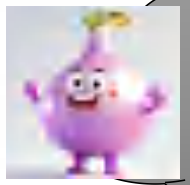
- Requirements contain multi-layer conditions and mode scopes.

→ **Autoregressive generation cannot maintain tree structure.**

- NL is linear; LTL is a compositional tree.

→ **LLM lacks structural validation → "silent failures".**

Why is OnionL?



Onions have layers. You get it? We both have layers.



- **OnionL exposes hidden semantics, introduces explicit hierarchical structure, and enables deterministic LTL synthesis.**
- **OnionL turns uncontrolled free-text generation into a verifiable structured semantic tree.**

(Base case)	$\varphi ::= \varphi_{AP} \in AP$
(Scope application)	$\varphi ::= \varphi_{SC}(\varphi_1)$
(Relational composition)	$\varphi ::= \varphi_1 \varphi_{RE} \varphi_2$
(Nested combination)	$\varphi ::= \varphi_{SC}(\varphi_1) \varphi_{RE} \varphi_{SC}(\varphi_2)$

Illustrative Example. Consider the requirement: "*In valid mode, if the temperature exceeds 50, eventually the warning light is turned on.*" Its corresponding *OnionL* expression is:

```
(Globally
    (Implies
        (Atomic: "workmode = valid")
        (Eventually
            (Implies
                (Atomic: "temperature > 50")
                (Atomic: "warning = ON")))))
```

- **The BUILDING-ONIONL Algorithm is a two-stage, prompt-driven process:**

➢ **Stage I (Macro-Structure Extraction):**

Identifies the main semantic scope (Temporal or Mode-based) to build the root node.

➢ **Stage II (Recursive Decomposition):**

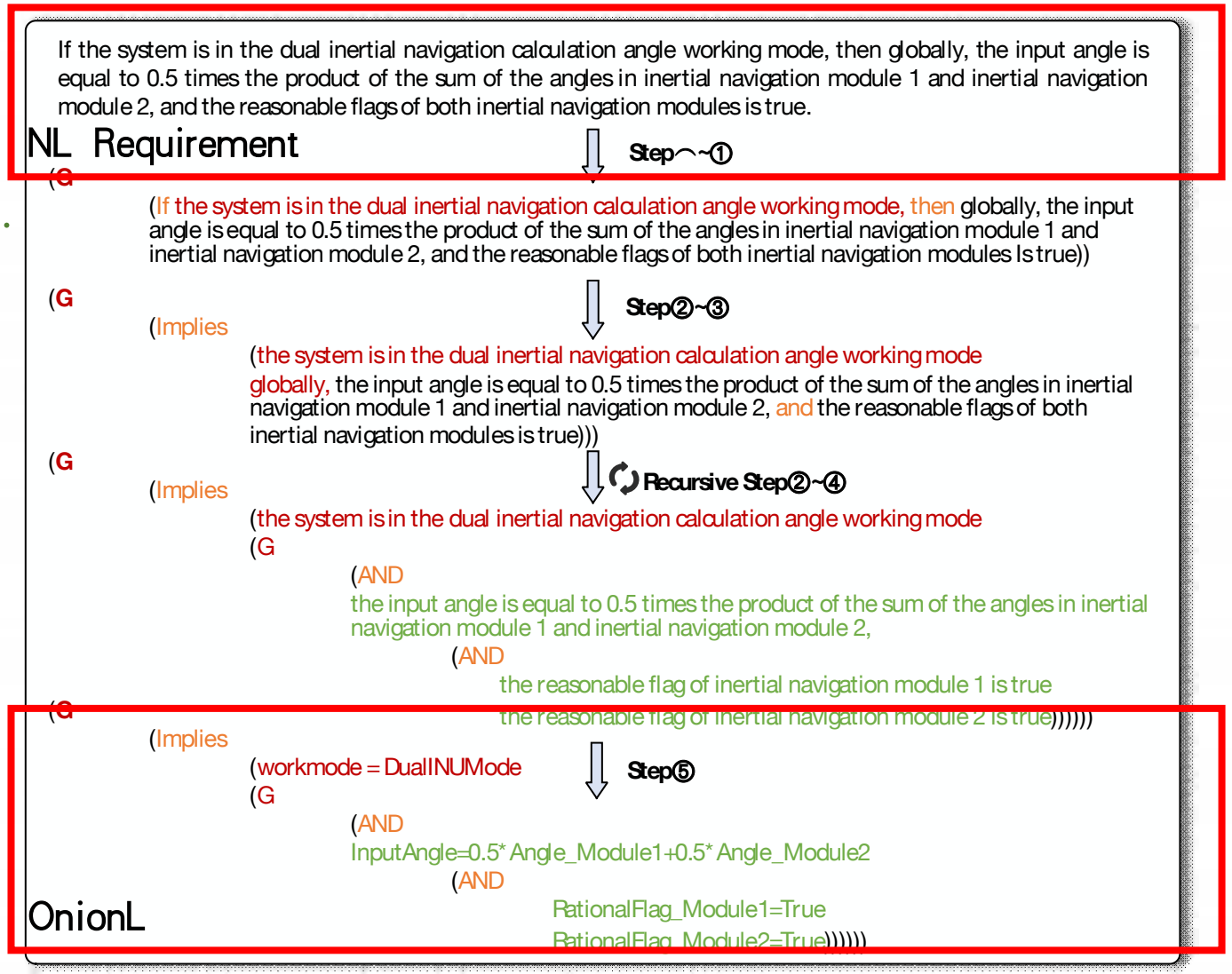
Parses the clause, extracts operators, and normalizes atomic propositions.

Algorithm 1: BUILDING-ONIONL

Input: Natural language requirement $\mathcal{R}$
Output: Structured *OnionL* expression φ
// Stage I: Macro-Structure Extraction
// Step 1: Scope Identification and Clause Separation
scope, clause ← EXTRACTSCOPE($\mathcal{R}$);
if scope *is Temporal* **then**
  // Step 2: Temporal scope → root node
  if ISATOMICCLAUSE(clause) **then**
    // Step 6: Atomic clause, normalize and terminate
    return NORMALIZEAP(clause)
  φ ← scope(DECOMPOSECLAUSE(clause));
else
  if scope *is Mode-based* **then**
    // Step 2: Mode scope → G(antecedent → consequent)
    $\varphi \leftarrow \mathbf{G}$(scope → DECOMPOSECLAUSE(clause));
  else
    // Step 2 (fallback): Wrap clause with G
    $\varphi \leftarrow \mathbf{G}$(DECOMPOSECLAUSE(clause));
// Stage II: Recursive Clause Decomposition and AP Normalizatio
Function DecomposeClause(c):
  if HASUNARYSCOPE(c) **then**
    // Step 3: Unary scope → parent
    scope, c' ← EXTRACTUNARYSCOPE(c);
    return scope(DecomposeClause(c'))
  if HASBINARYRELATION(c) **then**
    // Step 4: Binary relation → subtrees
    c_1, c_2, rel ← EXTRACTBINARYRELATION(c);
    return DecomposeClause(c_1) rel DecomposeClause(c_2)
  if ISATOMICCLAUSE(c) **then**
    return NORMALIZEAP(c) *// Step 5: Atomic clause, stop*
  refined ← REFINEIMPLICITSTRUCTURE(c);
  return DecomposeClause (refined) *// Step 5 (fallback): Refine*

Hierarchical Semantic Decomposition. Two-stage, six-step pipeline:

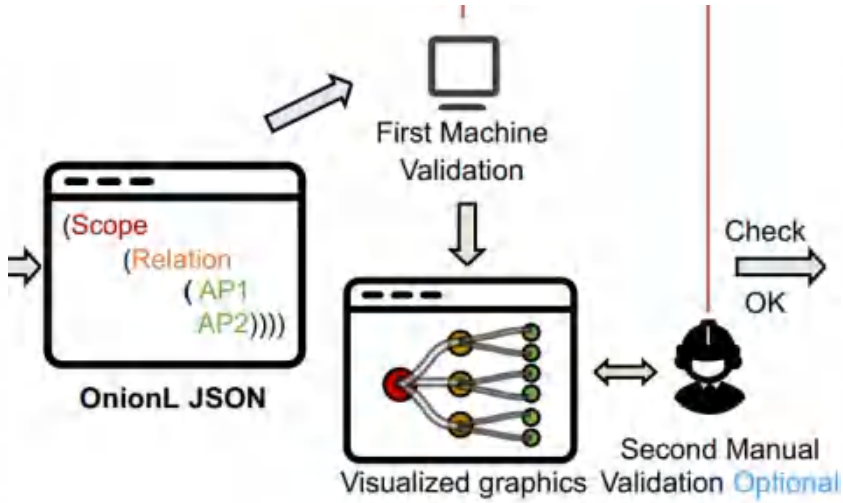
① Extract global scope.
② Build root.
③ Extract unary operators.
④ Decompose binary relations.
⑤ Check atomicity.
⑥ Normalize APs.



OnionL2LTL - From Structured Semantics to LTL Synthesis.

Check 1. Syntactic & Structural Validation.

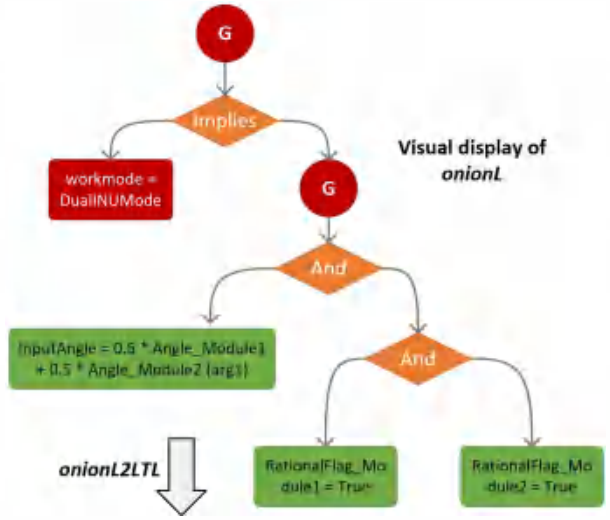
① Node validity checks (e.g., a U node must have two child nodes).
② Operation validity checks.
③ Report all errors and prevent translation execution if necessary.
④ Check for redundant expressions, such as G(G(p)).



Not perfect yet!

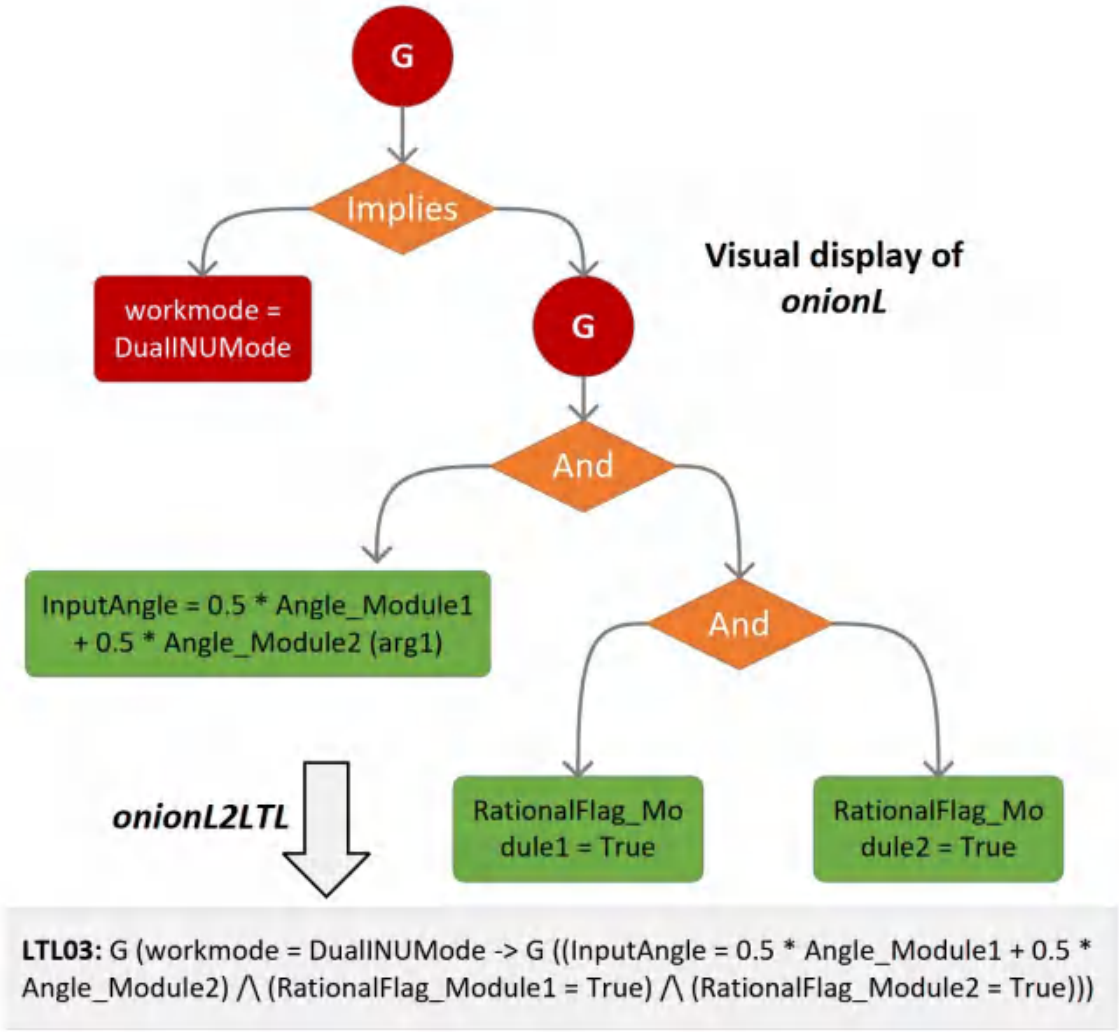
Check 2. Mermaid Graph Visualization (Human-in-the-loop)

① Renders OnionL trees as structure diagrams for engineers to view.
② Visualizes error paths for lightweight, interactive correction.
③ Enhances project interpretability and trustworthiness.



OnionL2LTL - Post-order Tree Traversal Synthesis.
Each OnionL tree node is deterministically converted into a valid LTL operator expression via post-order traversal.

- Construct formulas layer by layer upwards from the leaf nodes (AP), according to logical types
- Atomic propositions → Predicate combinations (e.g., Comp = "PowerSys", Var = "Voltage" → PowerSys.Voltage ≥ Vth)
- Logical relation nodes → Using standard connectors (/\, \/, OR, U, F, G, →, etc.)
- Scope nodes → Enclosing the entire subtree, such as G(...), F(...), X(...)



EVALUATION-Setup & Metrics

Datasets:

- **Academic. Circuit / Navigation / Office Email, 1500 sentences each.**
- **Industrial. 112 aerospace requirement statements derived from two satellite control systems.**

Differences between the two datasets:

Natural Language	LTL Formula
Once red, the light cannot become green next.	$G(\text{red} \rightarrow X\neg\text{green})$
Once the light is red, it must remain red until it turns yellow.	$G(\text{red} \rightarrow \text{red}\, U\, \text{yellow})$
If b holds, next c holds until a holds or always c holds.	$G(\text{b} \rightarrow X((\text{c}\, U\, \text{a}) \vee G\text{c}))$
If a holds then c is true until b.	$G(\text{a} \rightarrow (\text{c}\, U\, \text{b}))$
Navigate to the green room while avoiding landmark 1.	$(F\ \text{green}) \wedge G(\neg\ \text{landmark 1})$
Swing by landmark 1 before ending up in the red room.	$F(\text{landmark 1} \wedge F\ \text{red})$

NL in Academic datasets. Very simple! Obtaining the correct LTL specification is relatively easy for LLM.

EVALUATION-Setup & Metrics

Datasets:

- **Academic. Circuit / Navigation / Office Email, 1500 sentences each.**
- **Industrial. 112 aerospace requirement statements derived from two satellite control systems.**

Differences between the two datasets:

Req01: In the inertial navigation valid mode, the output will be set to reverse navigation computation, or the output will be set to low velocity low altitude navigation computation, or the output will be set to high velocity navigation computation, or the output will be set to high altitude navigation computation.

Req02: Globally, if the navigation system computes angles using dual inertial navigation modules, the input attitude angle shall be the weighted average of angles from module 1 and module 2 (weights 0.6 and 0.4), unless the system switches to single module or GPS signal is restored.

NL in Industrial datasets. Very difficult! Resulting in low accuracy in LLM recognition.

EVALUATION-Setup & Metrics

Baselines:

- ✓ **Method & Tools. Zero-shot, NL2LTL, NL2SPEC, NL2TL.**
- ✓ **LLM. GPT-4o、DeepSeek-V3.**

On Academic Datasets. Existing methods generally achieve high recognition accuracy, but our method is inferior to NL2TL on certain datasets.

TABLE II
PERFORMANCE COMPARISON ON ACADEMIC BENCHMARKS (LIFTED)

Method	Domain	Binary Acc. (%)	BLEU Score
Req2LTL	Circuit	95.30	**0.97**
	Navigation	94.50	**0.97**
	Office Email	**96.70**	**0.98**
NL2TL	Circuit	**96.20**	0.96
	Navigation	**96.50**	**0.97**
	Office Email	96.00	0.96
NL2LTL	Circuit	90.50	0.93
	Navigation	89.90	0.93
	Office Email	90.80	0.93
NL2SPEC	Circuit	90.80	0.93
	Navigation	89.70	0.93
	Office Email	91.20	0.93

EVALUATION-Setup & Metrics

On Industrial Datasets. Our method achieves higher accuracy, 88.4%!

90 out of 113 statements were converted correctly, but 13 Reqs is still wrong!

Setting	Exact Match (%)	Syntax Validity (%)	AP Recall (%)	BLEU Score
Prompt Strategy Comparison				
GPT-4o + Zero-Shot	43.8	89.3	98.5	0.76
GPT-4o + NL2LTL	55.4	91.5	98.4	0.77
GPT-4o + NL2SPEC	56.3	91.9	98.5	0.78
GPT-4o + NL2TL	65.2	94.6	98.1	0.85
GPT-4o + Req2LTL	**88.4**	**100.0**	**99.5**	**0.96**
DeepSeek-V3 + Zero-Shot	34.8	87.5	97.5	0.73
DeepSeek-V3 + NL2LTL	54.0	91.2	98.6	0.78
DeepSeek-V3 + NL2SPEC	54.5	91.9	98.7	0.79
DeepSeek-V3 + NL2TL	58.0	92.9	99.1	0.82
DeepSeek-V3 + Req2LTL	86.6	**100.0**	99.2	**0.96**
Ablation Study on Req2LTL				
D Full Version	**88.4**	**100.0**	**99.5**	**0.96**
A w/o Structured OnionL	65.2	98.2	98.9	0.86
B w/o Stage-wise Decomposition	58.9	94.6	98.1	0.85
C w/o Verification Feedback	84.8	90.2	**99.5**	0.91

EVALUATION-Setup & Metrics

Example Req 04 in these 13 errors, although REQ2LTL has excellent overall performance, it may fail in certain industrial applications due to the inherent uncertainty of natural language.

Example, Req04: *"The control system should as soon as possible initiate the heading adjustment function upon receiving a verified ARINC 429 waypoint command, ultimately reducing the deviation angle to less than 2 degrees."*

LTL04: $G($ WaypointCmd $=$ True $\rightarrow$ $(F($HeadingFun $=$ True$) \wedge F($DevAngleLow $< 2)))$



Both sub-goals were translated as *F (eventually)* by REQ2LTL, losing the semantic meaning of "as soon as possible."

The root of the error is that "as soon as possible" in natural language does not provide a formalizable time constraint.

EVALUATION-Setup & Metrics

Example Req 04 in these 13 errors, although REQ2LTL has excellent overall performance, it may fail in certain industrial applications due to the inherent uncertainty of natural language.

Example, Req04: *"The control system should as soon as possible initiate the heading adjustment function upon receiving a verified ARINC 429 waypoint command, ultimately reducing the deviation angle to less than 2 degrees."*

LTL04: $G($ WaypointCmd $=$ True $\rightarrow$
$(F\,($HeadingFun $=$ True$) \wedge F\,($DevAngleLow $< 2)))$

- This is why we provide a visual manual inspection component in the REQ2LTL process.
- Engineers can use OnionL's visual tree structure to more easily correct ambiguous requirement statements.
- Our method is quite practical.

LTL04: $G($ WaypointCmd $=$ True $\rightarrow$
$(X\,($HeadingFun $=$ True$) \wedge F\,($DevAngleLow $< 2)))$

Examp ... **ance, it may fail in** ... **ge.**

Exam

as poss
receivin
reducing

- Th ... ins
- En ... str ... req
- Ou

Req2LTL

nl to onionl | onionl to ltl

Rewrite natural language to ONIONL language

In the inertial navigation valid mode, the output will be set to reverse navigation computation, or the output will be set to low velocity low altitude navigation computation, or the output will be set to high velocity navigation computation, or the output will be set to high altitude navigation computation.

NL TO ONIONL

ONIONL grammar check

```
(Globally
  (Implies
    (Atomic: "Work_Mode = InertialNavigationValidMode")
    (OR
      (F (Atomic: "Output_Mode = ReverseNavigation"))
      (OR
        (F (Atomic: "Output_Mode = LowVelocityLowAltitudeNavigation"))
        (OR
          (F (Atomic: "Output_Mode = HighVelocityNavigation"))
          (F (Atomic: "Output_Mode = HighAltitudeNavigation"))
        )
```

CHECK | FEEDBACK

ONIONL tree check

MERMAID | FEEDBACK

Req2LTL

nl to onionl | onionl to ltl

Rewrite natural language to ONIONL language

In the inertial navigation valid mode, the output will be set to reverse navigation computation, or the output will be set to low velocity low altitude navigation computation, or the output will be set to high velocity navigation computation, or the output will be set to high altitude navigation computation.

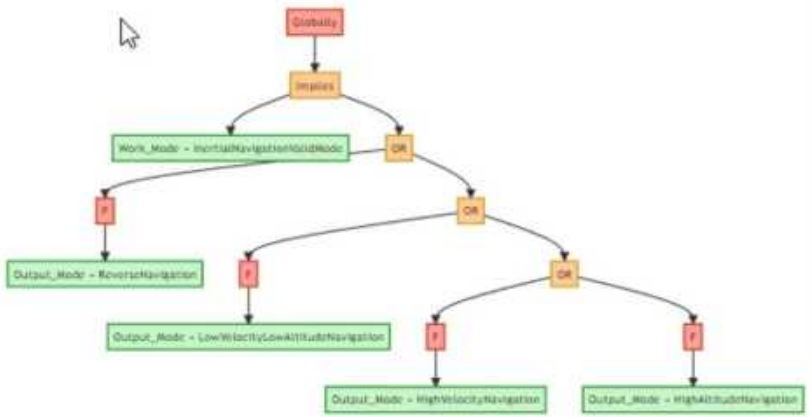
NL TO ONIONL

ONIONL grammar check

The expression syntax is correct!

CHECK FEEDBACK

ONIONL tree check



MERMAID FEEDBACK

Examp ance, it may
fail in ge.

Exan
as poss
receivin
reducing

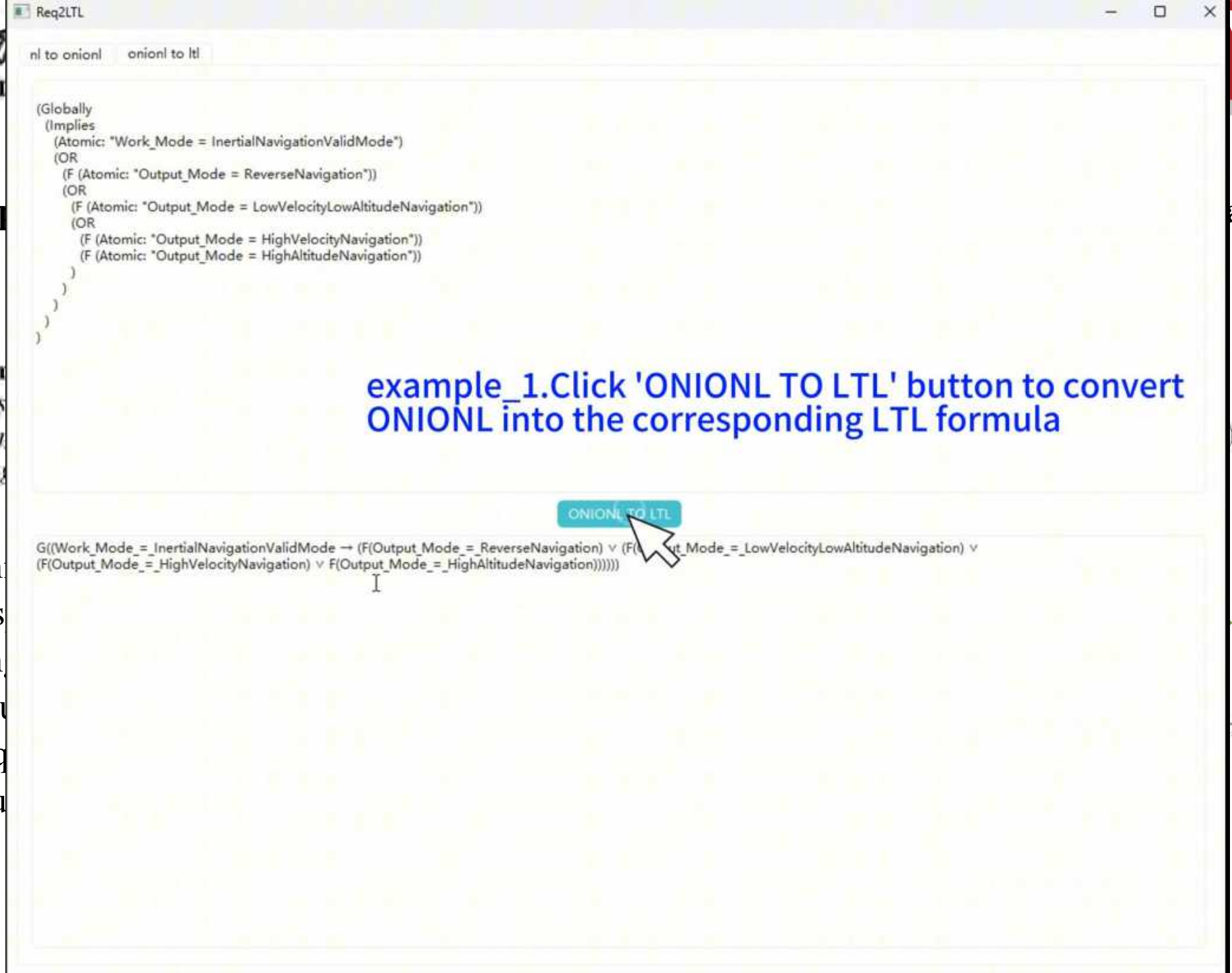
- Th
  ins
- En
  str
  req
- Ou

Low < 2)))

Examp ... **ance, it may fail in** ... **ge.**

Exan ... *as poss* ... *receivin* ... *reducin* ...))) ... Low < 2)))

- Th ... ins
- En ... stru ... req
- Ou



Conclution

- We introduced REQ2LTL, a modular framework for translating NL requirements to LTL. Its core is the novel OnionL intermediate representation, which successfully combines the semantic decomposition power of LLMs with the precision of deterministic, rule-based translation. On real-world aerospace requirements, REQ2LTL achieves 88.4% semantic accuracy and 100% syntactic correctness, significantly outperforming prior methods. It provides a practical and robust foundation for the scalable adoption of formal methods in safety-critical systems.

- **Future work:** We plan to extend the REQ2LTL framework to support more expressive temporal logics, such as Signal Temporal Logic (STL) and Metric Temporal Logic (MTL). This extension will enable the framework to handle the explicit quantitative timing constraints (e.g., "respond within 5 seconds") that are common and critical in industrial requirements.

Code & Data

WeChat

Thanks for listening