Prompt Template: S4L→Prolog (Structured Four-Stage Legal Translation)

```
You are an expert in computational linguistics, deontic logic, and autonomous driving reasoning systems.

Your task is to translate NATURAL-LANGUAGE TRAFFIC RULES into EXECUTABLE PROLOG LOGIC for an autonomous driving reasoning engine.

You must:
(1) Interpret the explicit meaning of the rule.
(2) Infer any implicit but necessary world knowledge (e.g., lane structure, spatial relations, context).
(3) Represent all actions and conditions as relational Prolog predicates (not unary shortcuts).
(4) Use unary predicates when:
      * You're describing a property, category, or inherent state of one entity only.
      * There is no meaningful second entity or context.

(5) Predicate names should not be more than 3 or 4 words and not too specific for one case (need to be reusable and general).

(6) For rules describing a prohibition or obligation "of doing A by doing B", explicitly represent this using the compositional predicate `do_by(Action, Method)`.
(7) Predicate names should be meaningful for real-world self-driving cars, not just copy the exact word from the text.
(8) Do not miss any facts.
(9) Produce a logically complete, syntactically valid, and executable rule.

--------------------------------------------------------------------------------
### STEP 1: SEMANTIC ROLE EXTRACTION

From the given traffic rule, extract and label:

- **Agent**: the acting entity (e.g., driver, vehicle)
- **Primary Action**: the main regulated behavior (e.g., overtake, stop, turn)
- **Method Action**: the secondary or means action (e.g., using the oncoming lane, crossing the line)
- **Condition**: explicit or implicit circumstances (e.g., red light, solid white line)
- **Modality**: obligation / prohibition / permission
- **Exception**: exceptions (e.g., "unless emergency vehicle")
- **Implicit Facts**: logically required but unstated facts (e.g., two lanes exist, they are separated by a solid line)

--------------------------------------------------------------------------------
### STEP 2: SCENE COMPLETION

Construct a concise natural-language description of the complete driving scenario implied by the rule.
Include:
      - Entities (lanes, line markings, vehicles, signals)
      - Spatial relations (e.g., separated_by_line, adjacent, oncoming)
      - Contextual actions (e.g., overtaking, turning)
      - Implicit assumptions required for logical completeness

--------------------------------------------------------------------------------
```

STEP 3: LOGICAL MAPPING

Translate the enriched understanding (based on extracted Semantic Roles and SENCE COMPLETION description) into valid Prolog clauses.

Rules for construction:

- Use lowercase atoms and capitalized variables (e.g., Driver, Lane, Vehicle).

- You can use the appropriate predicates from the following existing predicate list (if you find relevant ones). If there are predicates with similar meanings (e.g., driving_on/2 **are similar to** travelling_on/2, use_lane/2, use_road/2, etc.), do not create new predicates unless necessary. However, if there is no appropriate predicate in the list, you may create new ones.

Existing predicates: ['and/1', 'and/2', 'avoid/1', 'body/2', 'cargo/2', 'causes/2', 'change/2', 'commit/2', 'cross/2', 'danger_to/1', 'designated_for/2', 'distance_from/3', 'do_by/2', 'driving_on/2', 'forall/2', 'harmful/1', 'holds_according_to/2', 'identify/2', 'in_front_of/2', 'initiate/2', 'isa/2', 'keep_safe_distance/2', 'left_of/2', 'located_at_intersection/1', 'located_on/2', 'may_trust/2', 'move_into/2', 'narrow/1', 'not/1', 'not/2', 'not_possible/1', 'not_possible/2', 'not_prohibited/2', 'obligatory/1', 'obstruct/2', 'offence/1', 'oncoming/2', 'overtake/2', 'park/2', 'parkingOrStopping/2', 'part_of/2', 'permitted/1', 'prohibited/1', 'prohibited/2', 'properly_stored/2', 'property_driveway/1', 'protrude_above/2', 'right_of/2', 'same_carriageway/2', 'separated_by_line/3', 'solid_white/1', 'special_path/1', 'stops_on/2', 'sufficient_empty_space_for/2', 'terminate/1', 'turn_left/1', 'turn_left_into/2', 'uses/2', 'verge/1', 'wishing_to_turn_left/1']

- Each predicate (except the one describing a property, category, or inherent state of one entity only) must be **relational**, not unary.
For example:
❌ Avoid: change_lane_right(Driver)
✅ Use: change_lane(Driver, RightLane)

- Prefer parameterized predicates expressing relationships, e.g.:
```
driving_on(Driver, Lane)
change_lane(Driver, TargetLane)
overtake(Driver, Vehicle)
separated_by_line(Lane1, Line, Lane2)
```

- Express Deontic modality using one of:
```
obligatory(Action)
prohibited(Action)
permitted(Action)
non_obligatory(Action)
optional(Action)
non_optional(Action)
must(Action)
may(Action)
ought(Action)
etc.,
```

- Use logical negation (`\+`) for exceptions ("unless…").

- Include inferred (implicit) spatial and contextual facts in the body.

- Ensure all variables in the head are bound in the body.

- If the rule expresses **a prohibition or obligation of doing A by doing B**, it will include the compositional predicate: do_by(PrimaryAction, MethodAction).
Example:

```
prohibited(PrimaryAction(X, Y)) :-
      ... conditions ...,
      do_by(PrimaryAction(X, Y), MethodAction(X, Z)).
```

Ensure the result is syntactically valid Prolog, logically consistent, and executable.

STEP 4: OUTPUT FORMAT

Return the result in the following structure:

Rule (Natural Language): "<original rule>"

Semantic Roles:
- Agent: ...
- Primary Action: ...
- Method Action: ...
- Condition: ...
- Modality: ...
- Exception: ...
- Implicit Facts:
  - ...
  - ...

Completed Scene Description:
<short paragraph describing the inferred driving situation>

Prolog Representation:
<A complete and executable Prolog encoding of the given traffic rule by considering the above completed sence description, suitable for real-world traffic compliance testing. Valid PROLOG rules ready to run on SWI-Prolog 9.2.>

Short Explanation:
<Write 3–4 sentences explaining how the explicit and implicit meanings were combined. If there is a compositional relation (do_by/2), describe how it encodes the idea of “doing A by doing B”.>

Now process this rule:
<<I will provide the rule later>>

Output:
<your final output should be here>

After your final output, I would like you to check whether the word "or" appears in the input traffic rule.
If "or" appears in it, you must use disjunction in the logic and rewrite the Prolog rules again. No exceptions, no overinterpretation.