Human Expert Evaluation:

Prolog programs cannot use logical negation like in predicate logic formulas. Therefore, we provide the built-in predicate “not” as “negation as Failure”. It seems there are places in the resulting Prolog program where “not” is being used as logical negation. In such cases, it might be better not to use [not].

Below are comments on what human expert noticed for each rule.

Rule-1

LE to Prolog: Correct
NL to Prolog: Correct.
S4L to Prolog: Best.

Rule-2

LE to Prolog: Correct
NL to Prolog: Incorrect? Need to confirm if *and([straight_ahead, turn_left])* should be *or([straight_ahead, turn_left]).*
S4L to Prolog: Correct.

Rule-3

LE to Prolog: Incorrect? Is it okay that *wishing_to_turn_left(vehicle)* is missing?
NL to Prolog: Correct.
S4L to Prolog: Correct

Rule-4

LE to Prolog: Correct.
NL to Prolog: Correct.
S4L to Prolog: Best

Rule-5

LE to Prolog: Incorrect. It states, “cannot overtake” without specifying “because the road is narrow.”

NL to Prolog: Incorrect. The causal relationship between “the road being narrow” and “cannot overtake” is not expressed.

S4L to Prolog: Incorrect. The causal relationship between “the road being narrow” and “unable to overtake” is not expressed.

Rule-6

Article 5, Paragraph 4 states “Overtaking is prohibited unless sufficient space is available”. When another article is referenced like this, we must decide whether to execute the Prolog program for Article 5, Paragraph 4, or to append the conditions stated in Article 5, Paragraph 4 to this Rule 6. Here, Article 5, Paragraph 4 is not applied and is treated as plain text, but this is still not an error.

LE to Prolog: Incorrect.
NL to Prolog: Correct. Article 5, Paragraph 4 is not written, but this is not an error.
S4L to Prolog: Correct.

Rule-7
LE to Prolog: Correct.
NL to Prolog: Correct.
S4L to Prolog: Correct.

Rule-8
LE to Prolog: Correct.
NL to Prolog: Correct
S4L to Prolog: Correct

Rule-9
LE to Prolog: Correct
NL to Prolog: Incorrect. The built-in predicate "not" is an argument to "causes", but this may not be a valid Prolog program.
S4L to Prolog: Correct. It needs to be confirmed whether *isa(Driver,driver)* is the same as *driver(Driver).*

Rule-10
Section 127(1) of the German Code of Criminal Procedure is a rule allowing arrest in flagrante delicto. Rule10 seems to be a rule stating that even for an arrest in flagrante delicto, the driver must not cross the white line; in other words, arrest in flagrante delicto does not constitute an exception. Such rules are very difficult to write in Prolog because they must describe the priority relationship between the main rule and the exception rule. Probably, the following three rules are needed for accuracy, but just the third one might suffice. Expert thinks S4L Prolog program expresses this third one.

If A then P (main rule)
If B then negation_of_P (exception rule)
If A and B then P (main rule has priority over exception rule)

LE to Prolog: Correct.
NL to Prolog: Correct.
S4L to Prolog: Correct.

Rule-11
LE to Prolog: Correct.
NL to Prolog: Incorrect. It does not state that stopping causes a fault.
S4L to Prolog: Incorrect. It does not state that stopping causes a fault.

Rule-12
LE to Prolog: Correct.
NL to Prolog: Correct.
S4L to Prolog: Correct.

Rule-13
LE to Prolog: Incorrect. Does not describe [Parking].
NL to Prolog: Incorrect. Missing information that space is available on the "right side" of the roadway. Should [parkingOrStopping] be split into [parking] and [stopping]?
S4L to Prolog: Correct.

Rule-14
LE to Prolog: Correct
NL to Prolog: Correct.
S4L to Prolog: Correct

Rule-15
LE to Prolog: The rule head should be permitted, not prohibited. Exceptional_case is not described.
NL to Prolog: Correct.
S4L to Prolog: Correct.

Rule-16
LE to Prolog: Incorrect. It does not describe that overtaking is already being performed.
NL to Prolog: Correct.
S4L to Prolog: Best.

Rule-17
This rule only states "it does not constitute overtaking," so it does not mean "Cross" is permitted. Should we call the Prolog program for StVO Section 5, or add Section 5's conditions to Rule17?

LE to Prolog: Incorrect.
NL to Prolog: Incorrect.
S4L to Prolog: Incorrect.

Rule-18
LE to Prolog: Incorrect. The final condition contains a not operator. If this does not signify "negation as failure," the Prolog program cannot execute.
NL to Prolog: Incorrect. The final condition contains a not operator. If this does not signify "negation as failure," the Prolog program cannot execute.
S4L to Prolog: Correct.

Rule-19
LE to Prolog: Correct.
NL to Prolog: Correct.
S4L to Prolog: Incorrect. What is prohibited *is "cross",* not *"park"*.

Rule-20
LE to Prolog: Incorrect. The built-in predicate "not" is used as an argument to the head predicate *"may_trust"*.
NL to Prolog: Incorrect. The built-in predicate "not" is an argument of the head predicate *"may_trust"*.
S4L to Prolog: Incorrect. Using "Permitted" as the head predicate is odd. *"may_trust"* would be better.