本体推理规则引擎热插拔设计

1 概论

总体目标：构建本体推理系统，用于交规和防御驾驶安全规则的合规检验。

本体推理应用的最大挑战是实时性，没有专门支持车载的推理引擎，桌面系统运算多层大型推理网络时最慢的时候能耗时一分钟以上。十年前出现过大量的本体推理的 AD 应用研究，代表性的活动是欧盟组织的联合攻关活动。但是在近年来端到端性能加速提升的背景下，本体应用逐渐被忽略，主要原因如下，

1） 应用领域：从本体的特性来看，在低层（感知/规划）上 基本用途不上，在高层（决策/验证/解释） 用武之地，尤其是交通规则合规检查、防御驾驶推理领域，未来应用空间最大。
2） 应用规模：如果将本体用来建设大型世界模型，变量和运算量的指数型增长使实时性变得不可能实现

因此，开发主要路线如下：

1） 主要应用在高层决策和合规检验
2） 采用乐高积木的单元概念，保证模型能根据环境采用最小化运行。每条规则都是一个插件单元，可以随时进行 USB 那样的热插拔，将推理周期保证到 100ms 以内

本案根据《加州 2024～2025 驾驶员手册》为例，对交规进行本体描述。因为采用了主板+插件架构，所以可以轻松转移到其他任何一个国家/地区的法规表达，也可以转换成防御驾驶规则表达。

用户体验：

1） 可以利用开发工具包和插件编写规范自己编写规则

2） 针对一次事故，可以事后调用运行日志，记录可显示出事发时调用了哪条规则，规则逻辑是否正确，推理的输入量是否被正确传感，哪辆车违规在先等等，总之，对于追责和保险都能提供一手证据

3） 对不满意的场景行为，用户可以单独抽取规则模块进行修改，而对原系统不造成忽扰。可以依据这个系统进行全面行为设计

本文以“无灯路口路权”判断为例，说明主板、规则插件、测试包的开发方法。暂不包括后续上车应用部分，将随后进行。整个系统开发完成以后应当包括四大部分：规则表达、本体建模、逻辑推理与环境感知对接。

第一部分：交通规则表达与抽象（规则定义）

目标：将“无灯路口”的路权规则用自然语言和形式逻辑清晰表达出来。

关键工作：

1. **收集与总结**无灯路口常见场景（如 T 字型、四向停车、同时到达等）；
2. 将规则用自然语言**归纳整理为规则列表**（如“先到先行”、“右侧优先”等）；
3. 抽象出**影响路权的要素**：到达顺序、方向（直行/左转）、位置（右侧/左侧）等；
4. 为后续建模提供清晰的语义边界。

第二部分：本体设计与规则建模（Ontology + SWRL）

目标：构建支持推理的 OWL 本体并用 SWRL 表达推理规则。

关键工作：

1. **定义本体核心类举例**：
   - Vehicle（车辆）
   - Intersection（交叉口）

- ApproachDirection（接近方向）
- ArrivalTime（到达时间）
- IntendedAction（意图：直行/左转/右转）
- hasRightOfWay（ObjectProperty）

2. **构造个体之间的优先关系**：
   - hasPriorityOver
   - 使用 SWRL 规则表达如："如果车 A 早到于 B，则 A hasPriorityOver B"
3. **设置推理支持**：
   - 定义 hasPriorityOver 为 Transitive 和 Asymmetric
   - 使用 HermiT 或 Pellet 启用 SWRL 推理

第三部分：环境建模与输入映射（感知 → 本体）

目标：将自动驾驶车辆的传感器数据转化为本体中的输入。

关键工作：

1. **设计传感输入映射格式**（JSON/XML → OWL）：
   - 如：车 A 到达时间 3.2s，方向"north"，意图"左转"
2. 将感知数据实例化为 OWL 中的个体（使用 API 或自动转化工具）：
   - VehicleA rdf:type Vehicle ; hasArrivalTime "3.2" ; hasApproachDirection "north" ; ...
3. 与本体规则关联，触发推理。

第四部分：推理结果应用与接口设计（自动驾驶系统集成）

目标：将推理结果转化为行动建议或约束指令给自动驾驶控制模块。

关键工作：

1. **输出格式设计**：

    - 如：VehicleA hasRightOfWay → action: proceed；
    - VehicleB 无优先权 → action: wait
2. **构建可视化/调试接口**（可选）：
    - 前端网页模拟交叉口推理（React/Three.js）；
    - 显示传感信息、规则决策路径、当前行动建议。
3. **集成控制策略**（与自动驾驶系统对接）：
    - 使用 ROS Bridge / Python API / C++ 节点 读取推理结果
    - 实时约束行为生成器（如：wait, go, yield）

最终结果包括：

- 交规本体（.owl） 文件
- SWRL 规则与推理机制（可在 Protégé 中验证）
- 感知数据转化模块（Python）
- 行为决策接口（供自动驾驶调用）
- 可视化调试页面（交叉口模拟场景）

本文专门论述第一、第二部分。

2 交规理解

将交规手册整理成规则手册，剔除不适用的部分，比如驾驶证管理程序，等等。

本文以加州 2025 驾驶员手册为例，专门解决路权判断问题。所以，将于此相关的规则抽象出 26 条。具体见“26 条规则结构化表格.csv”。

3 本体推理架构设计

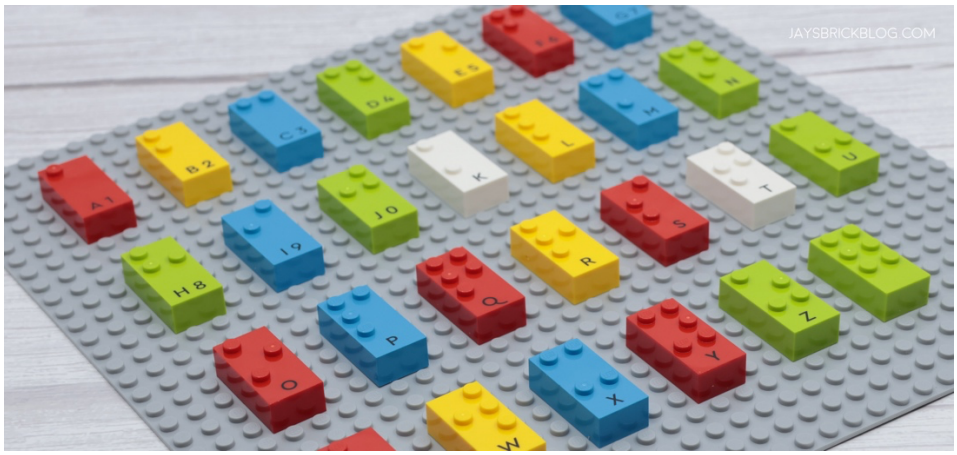

图 1

整体架构采用乐高积木形式。对 USB 式热插拔的要求是：

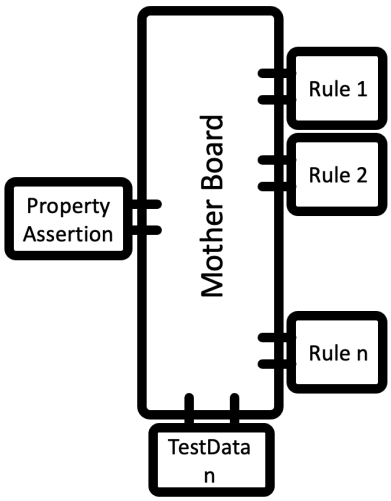

图 2

1） 对主板要求

预埋推理所必要的 Object Properties；不允许包含 SWRL 规则，也就是说，如果所有的规则插件都拔除，那么 SWRLTab 内容为空

2） 规则插件要求

所有的规则都包含在规则插件里，均采用 SWRL 进行编辑。规则插件通过 import 功能导入主板，不得采用 merge ontology 方式导入

3） 根据动态场景调用相应的规则

在特定场景下调用相关规则，阻隔无关规则，降低计算维度。场景大概分布计划：

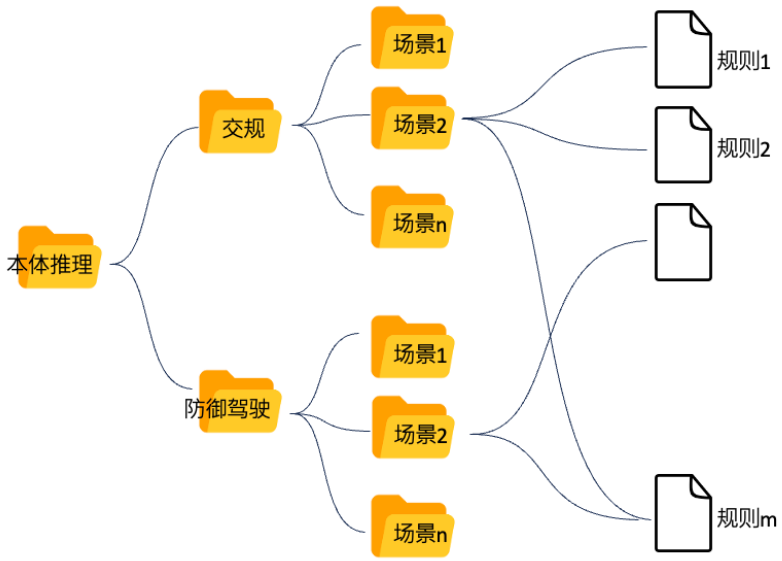

图 3

4） 断言输入 Property Assertion

Individual 定义和 individual 的 property assertion 全部由 PA 单元输入。主板和规则插件里面不得含有个体断言内容。个体断言只能代表纯粹的动态环境传感结果。

4 本体推理建模

本案以“无灯路口”规则为例，开发本体推理模型演示样板。

“无灯路口”在图 3 中算作一个场景。驶入这个场景以后自动驾驶系统就可以调用这个场景的相应规则。无灯路口规则暂时规定为四条（以后可随时增加）：

- 先到先行
- 右手优先
- 左传让直行
- 辅道让主道

开发目标：自动驾驶系统能判断优先权，并且能处理规则冲突。

4.1 主板开发

初始类结构设计举例

类名	描述
Vehicle	表示交通参与车辆
RightOfWayRule	所有路权规则的通用抽象类
IntersectionScenario	表示一个交叉口情境实例
TrafficSign	信号灯、停止标志、让行标志等
RoadCondition	表示道路或特殊场景属性（如坡道、障碍）

对象属性设计举例

属性名	类型	描述
mustYieldTo	ObjectProperty	A 车必须让行于 B 车
hasPriorityOver	ObjectProperty（Asymmetric+Transitive）	A 规则优先于 B 规则（用于规则推理）
appliesToScenario	ObjectProperty	规则适用于某个交通情境

involvesVehicle	ObjectProperty	某个情境中涉及的车辆

主板、插件、测试包的开发需要满足严格的命名空间规定，否则无法构成互相引用关系。

主板建模质量标准：

项目	要求
命名空间	http://www.auto-driving.org/ont/masterboard#
SWRL 规则	不含任何规则（SWRLTab 为空）
结构要素	
兼容性	必须能成功导入插件文件（插件中的规则与主板属性结构匹配）
显示要求	打开主板时必须显示正确 IRI（不能为 Anonymous）

主板模型示例见 *masterboard.owl*。该文件可用 protégé打开编辑，也可以用文本编辑器直接改写。

4.2 规则插件开发

插件模块建模质量标准（ruleN_xxx.owl）

项目	要求
命名空间	如：http://www.auto-driving.org/ont/rule1rightpriority#
导入路径	应为清晰的本地路径或相对路径，不使用 Anonymous
结构完整性	包含规则所需类、对象属性等定义（不能依赖主板外定义）
SWRL 格式要求	- 变量必须是 ?x、?y 显式变量格式； - 不能为 _autogen；
插件可视化	导入主板后，SWRLTab 中应能显示规则内容
规则名称规范	建议规则命名格式为 S1：right-hand rule、S2：first-arrival rule 等，清晰明了

规则插拔件开发的注意事项见 *SWRL 规则插件构建规范 v1.2*。

规则插件示例：*rule1rughthand.owl*，*rule2arrivefirst.owl*，*rule3straightpriority.owl*，*rule4mainroadpriority1*。

使用方法：打开主板以后用 import ontology 功能导入上述规则插件以后，规则就会显示到 SWRLTab 区域。见图 4 中的 S1～S4 内容。

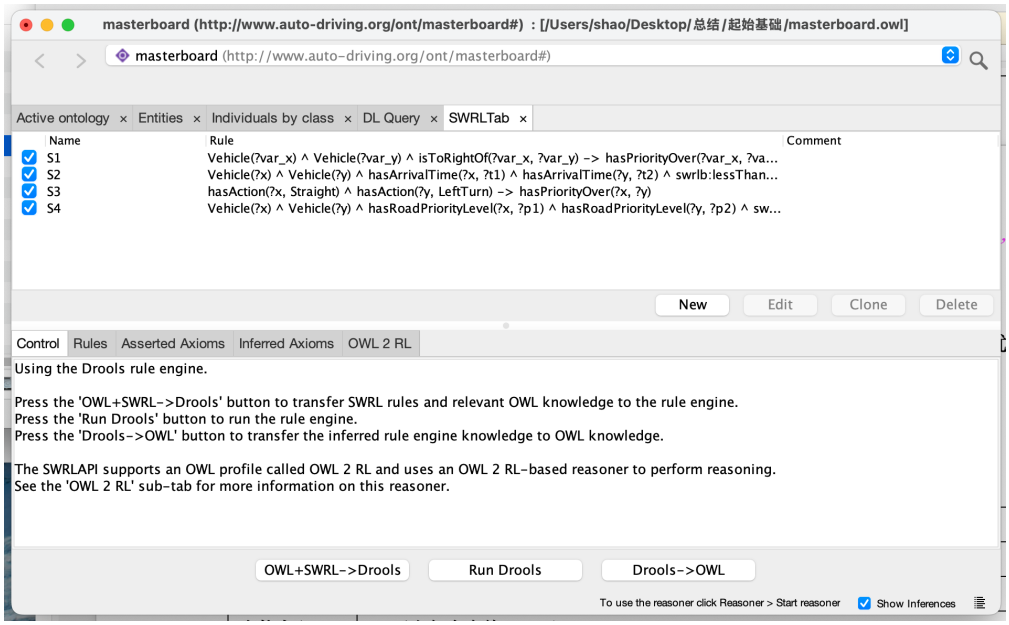

图 4

关于避免 autogen 错误的方法，见文档 *怎样消除 autogen 变量* 。

4.3 测试包插件开发

测试包相当于 Abox，也就是在环境中看到了什么。每个测试包专门用来检验如果个体断言施加无误（传感正确），推理系统是否能得出正确结论。

测试包构建质量要求（testdata_ruleN_vX.owl）

项目	要求

命名空间	如：`http://www.auto-driving.org/ont/testdata_rule1_v23#`
导入状态	能被主板成功导入，无错误提示
个体定义（举例）	- 至少包含个体 `v1` 和 `v2`； - 类型声明应为 `Vehicle`
属性断言	- 至少包含一个核心断言（如 `v2 hasRightSide v1`） - 该断言应在 `Property Assertions` 中可见，不应作为 `annotation` 显示
触发验证	在 `SWRL` 推理器运行时，规则应能被激活、触发结论（如 `hasPriorityOver(v1, v2)`）
导入方式	使用 `Import` 手动导入，不能用 `merge`，避免污染主板

生成规范和注意事项见 *热插拔测试包生成规范 v1*。

测试包示例见文件 *testdata1.owl*、*testdata2_v3_v4.owl*、*testdata_v5_v6.owl* 。

使用方法：在主板模型中，导入规则 1～4 以后，再用 import 功能导入测试包。启动推理机 pellet、Hermit，或者 Drool 就能看到推理结果。示例见图 5

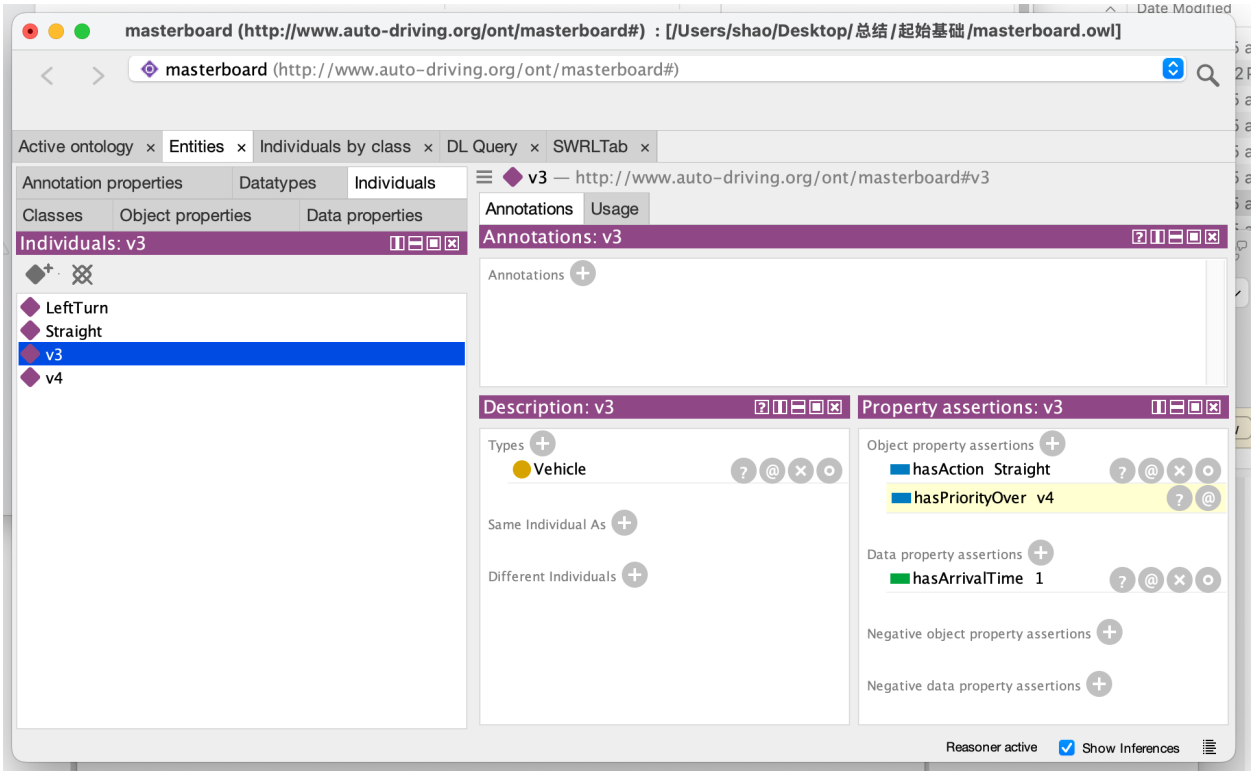

图 5 黄色背景就是 pellet 推理结果

注意，不同的操作系统、不同的 Protégé版本可能会有不同的界面和结果。

5 目前及下一步

1） 逐渐扩展场景和相应规则，增加模型的稳定性和运算速度。

2） 车载接口设计

3） 其他为开展工作见文档 *概念说明*。