Veritrace

Authoring di regole con LLM, revisione umana e tracciabilità simbolica in Jason

Lorenzo Leoni

Intelligent Software Engineering – University of Bologna

Presentazione progetto

Contesto

- Gli LLM sono utili per l'*authoring*: estrarre claim, proporre strutture, sintetizzare spiegazioni.
- Il problema è far entrare questi output in un *runtime* senza perdere controllo e tracciabilità.
- Veritrace usa una catena esplicita: human review, compilation, Jason MAS e trace finale.

Scelta progettuale

L'LLM può proporre artefatti candidati, ma il comportamento eseguibile passa solo da review, validation e runtime simbolico.

Problema

Gli LLM possono produrre regole plausibili, ma non dovrebbero diventare runtime behaviour in modo implicito.

Authoring ambiguo

Dal testo bisogna ottenere claim, condizioni, conclusioni e fonti.

Decisioni ispezionabili

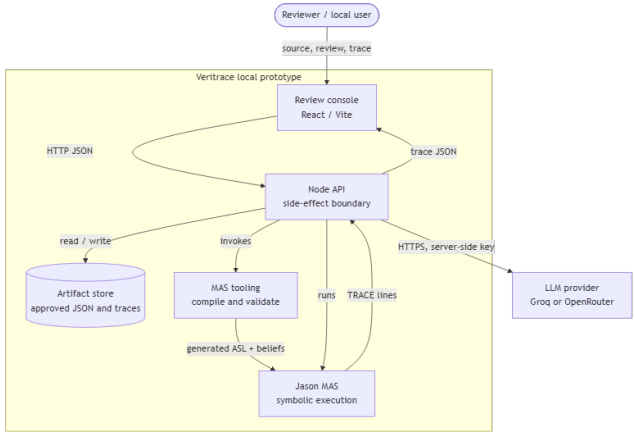
Serve sapere quale rule ha attivato quale evidence.

Autorità controllata

Il modello propone, ma il gate resta umano e simbolico.

Idea della soluzione

Una gated pipeline da source text a trace Jason.



Pipeline degli artefatti

Source → **Claim** → **Rule JSON** →
testo | significato candidato | bozza simbolica

Review → **AgentSpeak** → **Trace**
human gate | compiled logic | risultato ispezionabile

Punto chiave

Ogni passaggio lascia un artefatto leggibile: approved JSON, file `.asl`, belief file, trace JSON.

Due domini dimostrativi

Cardiac

- Utile per mostrare l'authoring.
- Source $\rightarrow$ claim $\rightarrow$ candidate rule.
- Esempio: soglia sul CMR Mass Score.

GDPR

- Utile per mostrare il runtime.
- Regole e piani già approvati.
- Casi: base giuridica mancante, categorie speciali, notifica data breach.

Dal JSON reviewato ad AgentSpeak

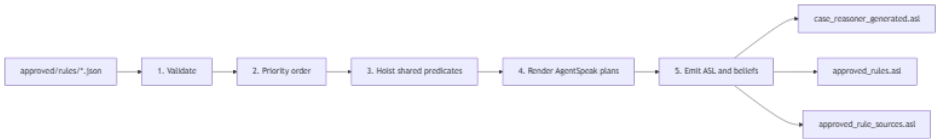
Approved artifact

```
"conditions": [
  "score(Case, cmr_mass_score, Score)",
  "cutoff(cmr_mass_score, Cutoff)",
  "Score >= Cutoff"
],
"conclusions": [
  "risk(Case, high)",
  "decision(Case, cmr_driven_high_suspicion
     )"
]
```

Generated plan

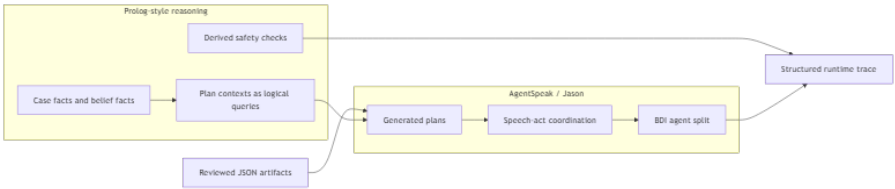
```
+!evaluate_case(Case)
  : score(Case, cmr_mass_score, Score)
    & cutoff(cmr_mass_score, Cutoff)
    & Score >= Cutoff
    & approved_rule(
     cmr_mass_score_above_cutoff)
  <- !emit_conclusion(...);
     !emit_evidence(...);
     .send(runtime_coordinator, tell,
           reasoner_done(Case)).
```

Compilation pipeline



- Valida solo approved rules strutturalmente corrette.
- Ordina le rules per priority per ottenere precedenza deterministica.
- Emette AgentSpeak e belief file con gate `approved_rule(...)` e mapping alle fonti.

AgentSpeak e Prolog nel progetto



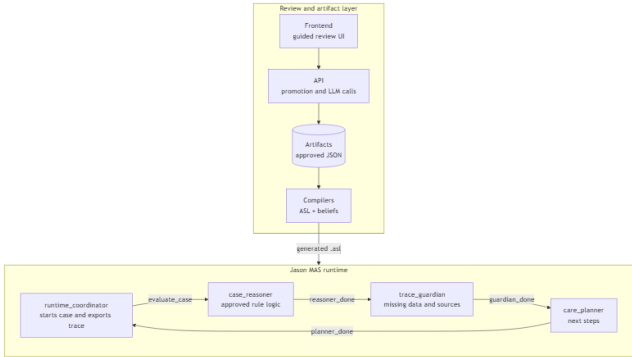
Prolog-style

- belief e case facts;
- plan contexts come query logiche;
- helper predicate come `usable_case_data/1`.

AgentSpeak/Jason

- events e plans;
- actions e goals controllati;
- coordinamento multi-agente con `.send`.

Jason MAS runtime



case_reasoner	applica le generated approved rules;
trace_guardian	controlla dati mancanti, approvazione, fonti e conflitti;
care_planner	aggiunge next step e piani approvati;
runtime_coordinator	orchestra ed esporta le linee `TRACE_*`.

Trace guardian

- Missing data espliciti.
- Activated rules ancora verificate con `approved_rule(...)`.
- Source mapping per ogni rule activation.
- Conflicting risks segnalati come human-review flags.

```
+!guard_trace(Case)
  <- !record_missing_data(Case);
     !check_runtime_review(Case);
     !check_rule_approval(Case);
     !emit_sources_for_activated_rules(
     Case);
     !check_conflicts(Case);
     !check_low_risk_safety(Case);
     .send(runtime_coordinator, tell,
           guardian_done(Case)).
```

LLM boundary

L'LLM è utile, ma non ha runtime authority.

1. Claim extraction

Estrae claim candidati dal source text.

2. Rule drafting

Produce candidate JSON da revieware.

3. Trace verbalization

Verbalizza trace già calcolati da Jason.

Non approva, non promuove, non compila e non esegue runtime knowledge.

Garanzie implementative

- **Promotion rollback**: se compilation o validation falliscono, l'artifact non resta promosso.
- **Deterministic generation**: AgentSpeak e belief files sono rigenerabili e diffabili.
- **Isolated runtime workspace**: i custom facts non modificano i sorgenti MAS.
- **Trace parser**: stdout Jason viene trasformato in trace JSON strutturato.
- **Integration tests**: verificano i flussi principali senza dipendere da un provider LLM.

Validation

- JSON validators per rules e plans;
- compilation consistency;
- golden cases Jason;
- trace contracts;
- Playwright integration tests della console.

```
npm test
npm run validate:rules
npm run validate:plans
npm run validate:compilation
npm run test:mas
npm run test:e2e:smoke
```

Takeaway

Veritrace non è un LLM che prende decisioni.

È un MAS simbolico che esegue regole approvate da un umano, con supporto LLM confinato ad authoring e spiegazione.

- artefatti tracciabili;
- compilazione verso AgentSpeak;
- ragionamento Prolog-style nei contesti;
- trace strutturato come fonte di verità.