Inference Engine

Short Overview

- Open World Assumption
- Forward Chaining Engine
- Backward Chaining Engine
- Hybrid Rule Engine
- Generic Rule Reasoner

Open World Assumption

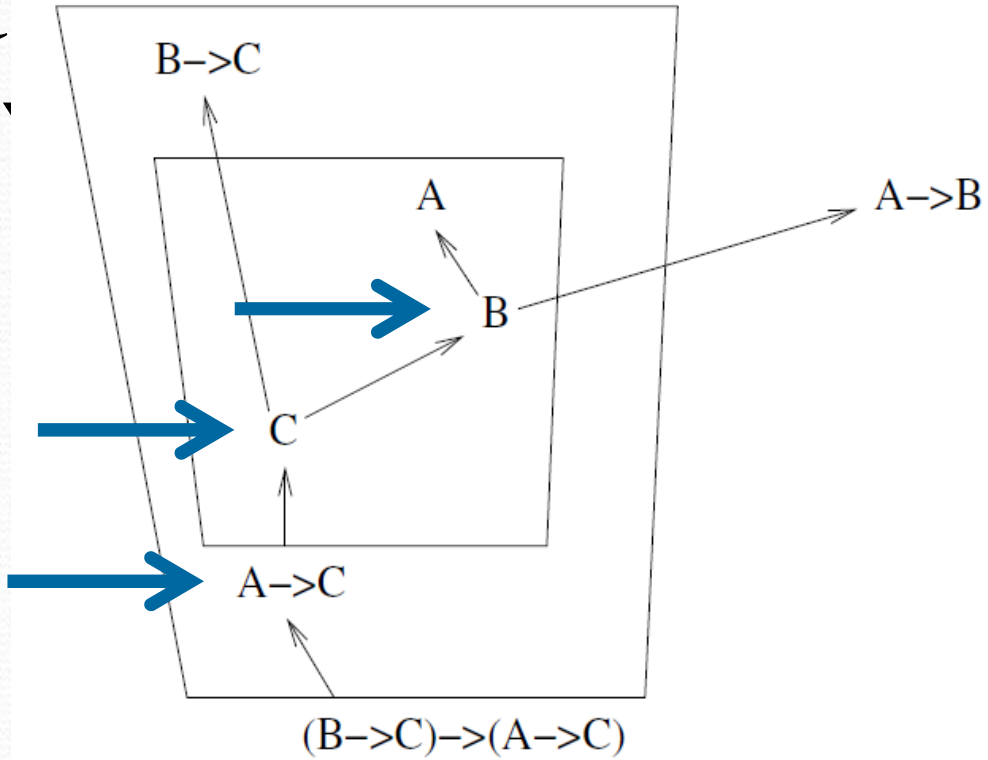
- Every non explicit knowledge is unknown

```
Rule       :=   bare-rule .
           or   [ bare-rule ]
                or   [ ruleName : bare-rule ]

bare-rule  :=   term, ... term -> hterm, ... hterm    // forward rule
           or   bhterm <- term, ... term    // backward rule

hterm      :=   term
           or   [ bare-rule ]

term       :=   (node, node, node)            // triple pattern
           or   (node, node, functor)         // extended triple pattern
           or   builtin(node, ... node)       // invoke procedural primitive

bhterm       :=   (node, node, node)            // triple pattern
```

Forward Chaining

- called Data-driven C
- Deduction Graph G
  for firing rules

- Based on
  RETE – Algorithm

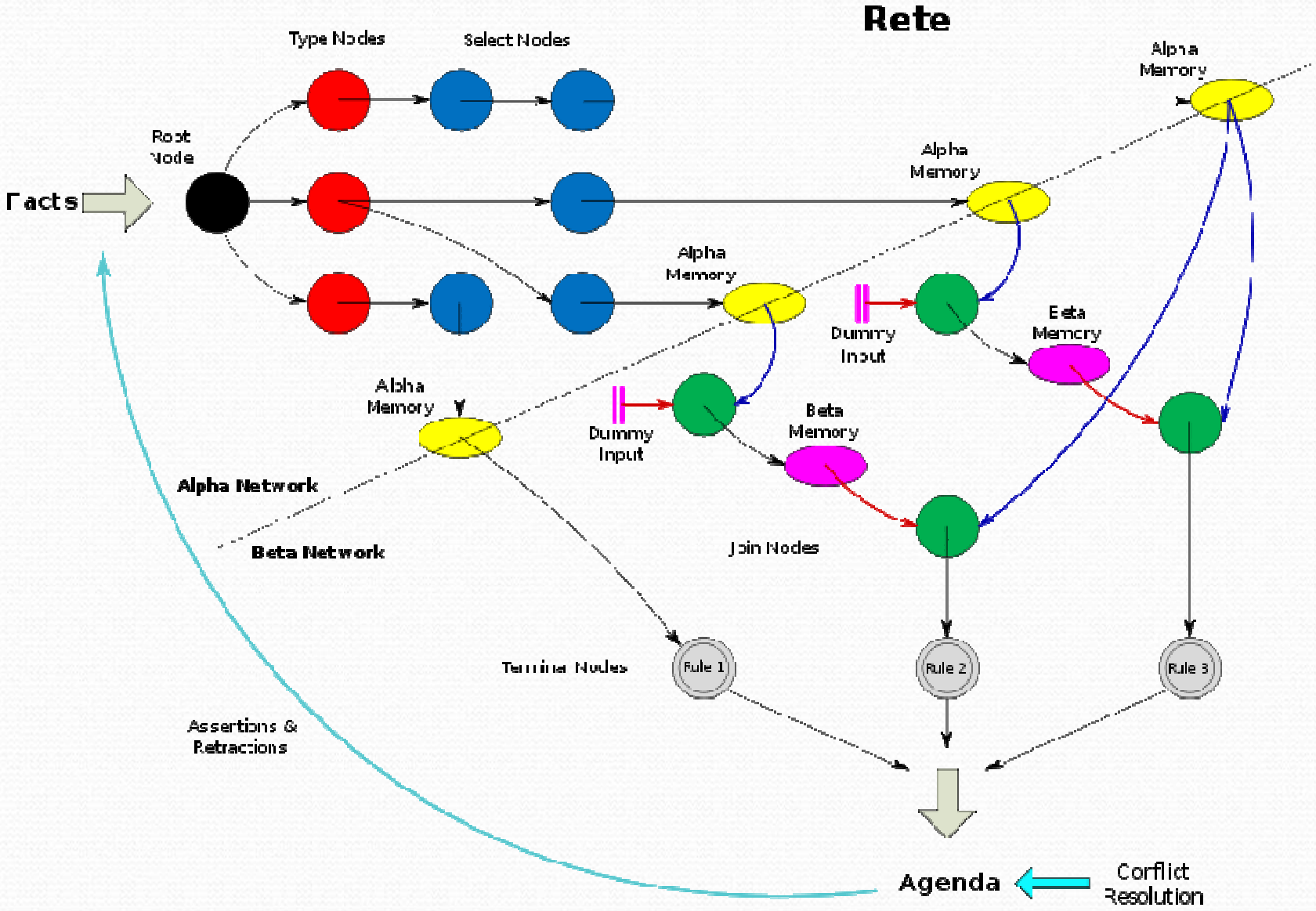


Forwarding Rule

RETE Algorithm

- Matches Tuples (= Facts)against Productions (=Rules)
- Produces a DAG, Graph splitted into
  - Alpha Network: contain Selection Nodes
  - Beta Network: contain Condition Nodes
- Conflict Resolution (*salience, recently used..)*

Example RETE - DAG

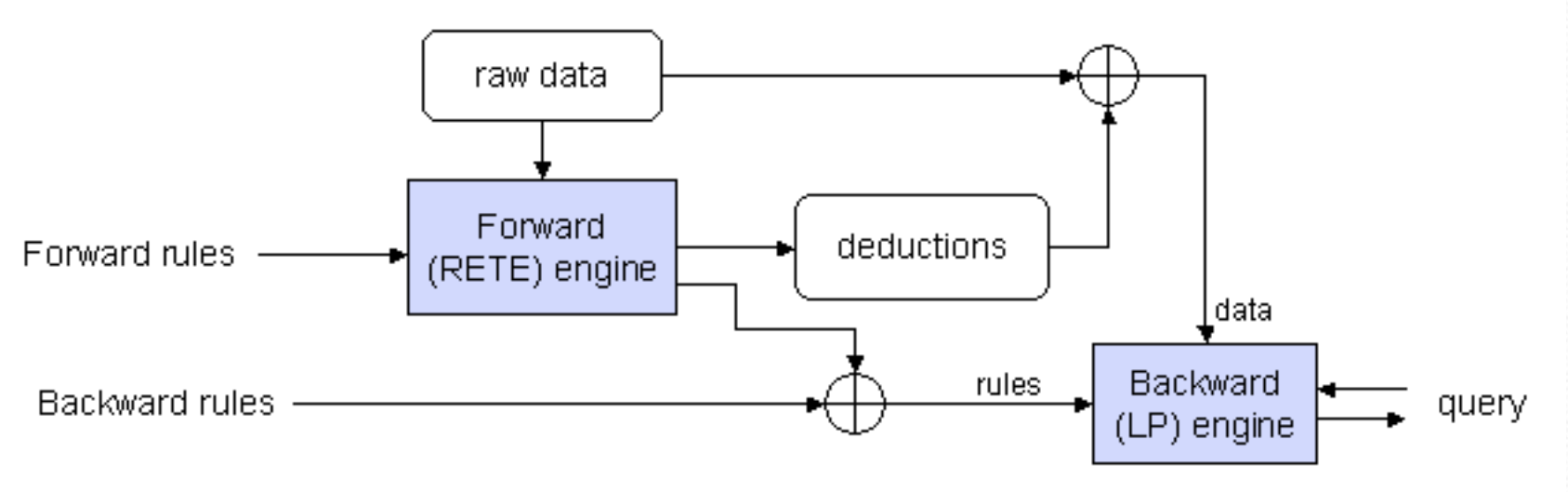


Backward Chaining

- Called Goal Orientated C.
- Logic Programming (Prolog)
- Results are cached (Called Tabling)
  - Inference Model is updated → Cache reset !
- Based on SLD Resolution
  - Special form of Resolution (Input Set is Horn !)

Hybrid Rule Engine (HRE)

- Combination of both Engines



Hybrid Rule Engine (HRE)

- Forward Part of HRE:
  - Pass an Instance of Rule to the Backward Engine
- Backward Part of HRE
  - Answers Queries
- Kind of Intelligence
  - Jena uses only those Part that is required

- Examples:
- https://sourceforge.net/p/semwebtech/code-0/14/tree/