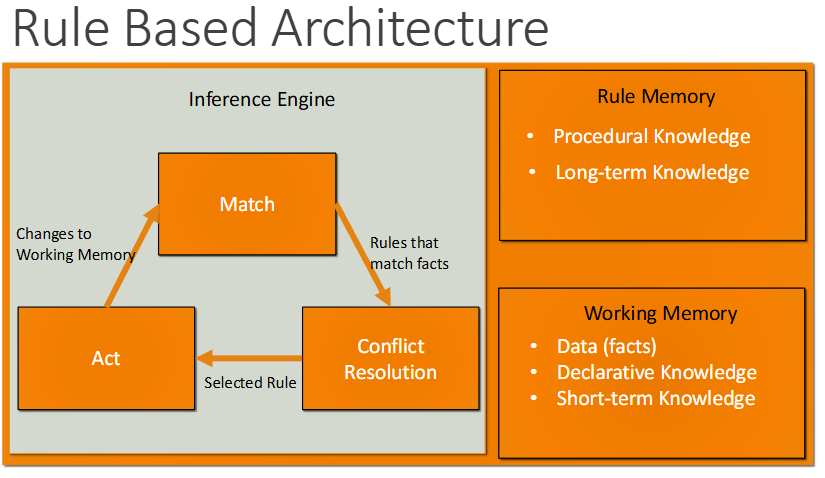
**NOTES**

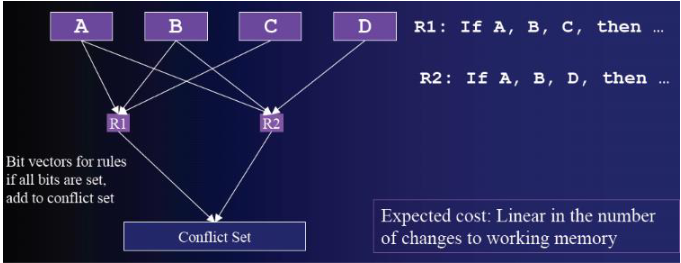
**Requirements for a rule based architecture**

* rule-base
* Stores information about subject domain
* Rules are the main way of expressing this knowledge
* If then format, IF <condition> THEN <conclusion>

**Rule based architecture**

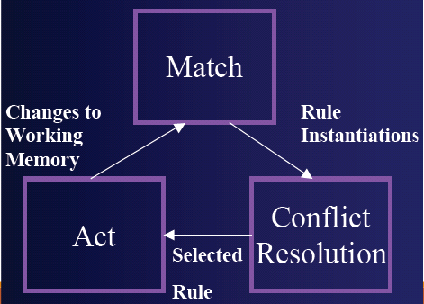


* Picking the next rule to fire
  + If only simple tests in conditions, compile rules into match net



**Conflict resolution**

* Which match rule should fire
* Which instantiation of a rule should fire?
  + Separate instantiation for every match of variables in rules



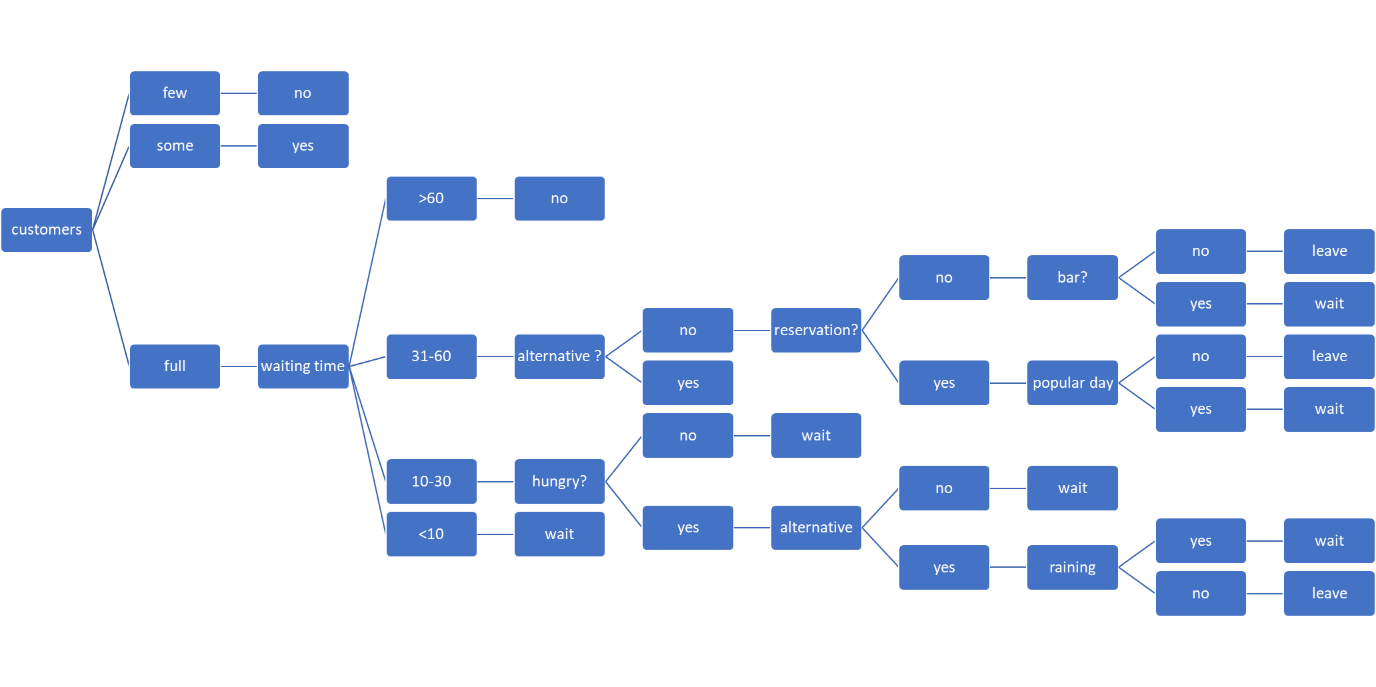
* Conflict resolution filters
  + Refractory Inhibition
    - Don’t fire same instantiation that has already fired
  + Data Recency:
    - Select instantiations that match most recent data
  + Specificity:
    - Select instantiations that match more working memory elements
  + Random
    - Select randomly between the remaining instantiations
* Other conflict res strats
  + Rule order
    - Pick first rule that matches
      * Makes order of loading important – bad for big systems
  + Rule importance
    - Pick rule with highest priority
      * Right 80% of time, forces total order on rules

**RBS Pros and Cons ->**

* Advantages
  + Corresponds to way people think of knowledge
  + Expressive
  + Modular knowledge
  + Easy to write and debug compared to decision tress
  + More concise than FSM
* Disadvantages
  + Memory intensive at times
  + Computationally intensive at times
  + Difficult to debug at times

**Rules**

* Consists of two parts
  + - If part also known as antecedent
    - Then part called the consequent
    - IF <antecedent> THEN <consequent>



    Whether to wait for a table can be a subjective exercise.  If this tree were to be adapted to help businesses determine whether punters are liable to wait, and to adjust elements of their business practice to optimise the number of customers, how would this be done?