Knowledge-Based Agents

- logical agents are always definite – each proposition is either true/false or unknown (agnostic).
- **knowledge representation language (KRL) –** expresses world knowledge.
  - o **declarative approach –** language is designed to be able to easily express knowledge for the world the language is being implemented for.
  - o **procedural approach –** encodes desired behaviors directly in code.
- **sentence –** a statement expressing a truth about the world in the KRL.
- **knowledge base (KB) –** a set of KRL sentences describing the world.
  - o **background knowledge –** initial knowledge in the KB
  - o **knowledge level –** we only need to specify what the agent knows and what its goals are in order to specify its behavior
  - o **Tell(P) –** function that adds knowledge P to the KB.
  - o **Ask(P) –** function that queries the agent about the truth of P.
- **inference –** the process of deriving new sentences from the knowledge base.
  - o *When the agent draws a conclusion from available information, it is guaranteed to be correct if the available information is correct.*

Logic

- **syntax –** description of a KRL in terms of well-formed sentences of the language.
- **semantics –** defines the truth of statements in the KRL w.r.t. each possible world.
- **model –** the "possible world" that is described by a KB.
  - o **model checking –** enumeration of all possible models to ensure that α is true in all models in which KB is true.
- **logical inference –** the process of using entailment to derive conclusions
- **logical entailment –** the concept of 1 sentence following from another sentence:

  $\alpha \models \beta$ if α is true, then β must also be true.

  Note: while similar to the notion of implication, entailment is a meta-statement, not a part of the language itself. That is, statements using entailment are used to describe other logical statements.
  - o <u>Monotonicity</u> – a set of entailed sentences can only *increase* in information as information is added to the knowledge base.

  $$KB \models \alpha \quad \Rightarrow \quad KB \wedge \beta \models \alpha$$
- **derivation –** if an inference procedure *i* can derive α from KB,

  $$KB \vdash_i \alpha$$
- **sound (truth-preserving) inference –** an inference procedure that derives only entailed sentences.
  - o *if KB is true in the real world, the any sentence α derived from KB by a sound inference procedure is also true in the real world.*
- **complete inference –** an inference procedure that can derive all entailed sentence.
- **grounding –** the connection, if any, between the logical reasoning processes and the real environment.

Propositional Logic

- **atomic sentence –** indivisible syntactic elements consisting of a single **propositional symbol**. *True* and *False* have fixed meaning.
- **complex sentence –** sentence constructed from other sentences joined by logical connectives:
  - **logical connectives:**
    - **not** $\neg$ **–** negation, **and** $\wedge$ **–** conjunction, **or** $\vee$ **–** disjunction
    - **implies** $\Rightarrow$ **–** implication ($(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$) *Note: if α is false, $\alpha \Rightarrow \beta$ says nothing about β.*
    - **if and only if** $\Leftrightarrow$ **–** biconditional
  - **order of operations (high->low):** $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- *Every known inference algorithm for propositional logic has a worst-case complexity exponential in the size of the input.*
- **logical equivalence –** two sentences α and β are logically equivalent if they are true in the same set of models.

  $$\alpha \equiv \beta \quad \Leftrightarrow \quad \alpha \models \beta \wedge \beta \models \alpha$$
- **validity –** a sentence is valid if it is true in all models.
  - **tautology –** sentences that are necessarily true.
- **Deduction Theorem –** *For any sentences α and β, $\alpha \models \beta$ if and only if the sentence $\alpha \Rightarrow \beta$ is valid.*
- **Satisfiablility –** a sentence is satisfiable if it is true in some model.
  - *Determining satisfiablity in propositional logic is NP-complete.*
  - **Proof by contradiction (refutation):** $\alpha \models \beta$ *if and only if the sentence* $\neg(\alpha \Rightarrow \beta)$ or rather $(\alpha \wedge \neg\beta)$ is unsatisfiable.
- **inferentially equivalent** – two sentences α and β are inferentially equivalent if the satisfiablity of α implies the satisfiablity of β and vice versa.

Reasoning Patterns in Propositional Logic

Common Patterns

	Modus Pones	And Eliminate	Bidirectional	Resolution
Premises	$\alpha \Rightarrow \beta, \quad \alpha$	$\alpha \wedge \beta$	$\alpha \Leftrightarrow \beta$	$\ell_1 \vee \ldots \vee \ell_k, \quad \neg\ell_i$
Conclusion	β	α	$(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$	$\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k$

Full Resolution Rule

$$\frac{\ell_1 \vee \ldots \vee \ell_k \quad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{l+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_i and m_j are complementary literals

- **conjunctive normal form (CNF)** – every sentence of propositional logic is *logically equivalent* to a conjunction of disjunctions of literals.

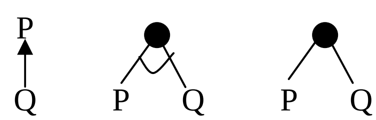
  $$\left(l_{1,1} \vee \ldots \vee l_{1,n_1}\right) \wedge \ldots \wedge \left(l_{m,1} \vee \ldots \vee l_{m,n_m}\right)$$

1. Eliminate biconditionals: $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Eliminate implications $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
3. Move $\neg$ inwards
4. Distribute $\wedge$ over $\vee$.

- A *complex sentence* can always be represented in CNF.
    1. literal – atomic sentence (positive) or negated atomic sentence (negative).
    2. clause – a disjunction of literals
    3. sentence – a conjunction of clauses.
- *Definite Clauses* – disjunction of literals of which exactly one is positive.

$$\neg n_1 \vee \ldots \vee \neg n_m \vee p \equiv \underbrace{n_1 \wedge \ldots \wedge n_m}_{body} \Rightarrow \underbrace{p}_{head}$$

    - **head** – the positive literal.
    - **body** – the negative literals; the premises.
    - **fact** – a definite clause with no negative literals.
    - **Horn clause** (*integrity constraint*) – a disjunction of literals at most one of which is positive. Horn clauses have the following advantages:
        - Inference can be done with forward/backward chaining.
        - Deciding entailment is *linear* in the size of the KB.
- **resolution** – a *sound* inference algorithm based on the resolution rule.
    - *By applying the only the resolution rule, any complete search algorithm can derive any conclusion entailed by any KB in propositional logic.*
    - **refutation completeness** – resolution can be used to confirm or refute any sentence, but it cannot enumerate all true sentences.
    - **resolution algorithm**
        - to show $KB \models \alpha$ we will show that $KB \wedge \neg\alpha$ is unsatisfiable.
        - $KB \wedge \neg\alpha$ is converted into CNF… a sequence of clauses
        - The resolution rule is applied to resulting clauses… each pair with complementary literals is resolved into a new clause.
            - if no new clauses can be added, α is not entailed.
            - if the *empty clause* {} is derived, α is entailed.
- **forward chaining** – a *sound* and *complete* inference algorithm (for Horn clauses) that is essentially Modus Pones. This algorithm is *data-driven reasoning*; reasoning which starts from the known data.
    - **AND-OR graph** – represents the derivation by a graph of literals. Disjunctions are represented by converging links and conjunctions are represented by multiple links joined by an arc.



$Q \rightarrow P$ $P \wedge Q$ $P \vee Q$

- **backward chaining** – a *sound* and *complete* inference algorithm (for Horn clauses) based on Modus Pones. This algorithm is *goal-directed reasoning*; reasoning that works backward from the goal.

Satisfiability

- **Davis-Putnam algorithm –** an algorithm for checking satisfiability based on the fact that satisfiability is commutative. Essentially, it is a DFS method of *model checking*.
    - Heuristics
        - **early termination –** short-circuit logical evaluations. A clause is true if *any* literal is true. A sentence is false if *any* clause is false.
        - **pure symbol heuristic –** a symbol that appears with the same sign in all clauses of a sentence (all positive literals or negative ones).
            - Making these literals *true* can never make a clause *false*. Hence, pure symbols are fixed respectively.
        - **unit clause heuristic –** assignment of true to unit clauses.
            - **<u>unit clause</u>** – a clause in which all literals but one have been assigned false → 1 way to make clause true.
            - <u>unit propagation</u> – assigning one unit clause creates another causing a cascade of forced assignments.
- **WalkSAT –** a local search algorithm based on the idea of a random walk that randomly alters the current assignment based on a *min-conflicts* heuristic.
    - If a satisfying assignment exists, it will be found, eventually.
    - WalkSAT can not guarantee a sentence is unsatisfiable.
- *Hard Satisfiablility*
    - Let *m* be the number of clauses and *n* be the number of symbols.
    - The ratio *m/n* is indicative of the difficulty of the problem.
        - **underconstrained –** relatively small *m/n* thus making the expected number of satisfying assignments high.
        - **overconstrained –** relatively high *m/n* thus making the expected number of satisfying assignments low.
        - **critical point –** value of *m/n* such that the problem is nearly satisfiable and nearly unsatisfiable. Thus, the most difficult cases for satifiablity algorithms

Propositional Logic Agents

- **inference-based agent** – an agent that maintains a knowledge base of propositions and uses the inference procedures described above for reasoning.
    - It is beyond the power of propositional logic to efficiently express statements that are true for sets of objects – FOL.
    - A proliferation of clauses occurs due to the fact that a different set of clauses is needed for each step in time.
- **circuit-based agent –** a reflex agent in which percepts are inputs to a sequential circuit – a network of gates (logical connectives) and registers (store truth value of a single proposition)
    - **dataflow** – at each time step, the inputs are set for that time step and signals propagate through the circuit.
    - **delay line** – implements internal state by feeding output of a register back into the register as input at the next time step. The delay is represented as a triangular gate.
    - Circuits can only ascribe *true/false* values to a variable; no unknowns.
        - requires each variable be represented by 2 *knowledge propositions*; 1 if the variable is known and the other for the value if known.
    - **locality** – the property of models in which the truth of each proposition can be determined by a constant number of other propositions.
    - **acyclicity** – a circuit such that every cyclical path has a time delay; a requirement for physical implementation.
    - Circuit agents have trouble representing interlocking dependencies ➔ incomplete.
- Tradeoffs:
    - *Conciseness* – circuit agents do not need separate copies of knowledge at each point in time whereas inference agents do.
    - *Computational Efficiency* – In worst case, inference is exponential in the number of symbols whereas circuit executes linearly in its size.
    - *Completeness* – An inferential agent is complete whereas a complete circuit-based agent becomes exponentially large in the worst case.
    - *Ease of Construction* – Building small, acyclic, not-too-incomplete circuits is relatively hard to building a declarative description.
- **Hybrid agent** – tries to get the best of both worlds by implementing reflexes with circuit agents and performing inference when needed for more difficult reasoning.