

David Keil, CSCI 400 Artificial Intelligence
Framingham State University

Topic 3: Knowledge representation and inference

1. Knowledge and beliefs
2. Concepts and instances
3. Logical inference
4. Expert systems and resolution proof

Inquiry

- What are ways in which knowledge is represented?
- How can knowledge be added to a knowledge base?

Objectives

- 3a. Use inference in propositional and predicate logic
- 3b. Describe methods of representing and using knowledge
- 3c. Distinguish *knowledge-based* from *goal-driven* agents

1. Knowledge and beliefs

- Unobservable parts of a state may be inferred by combining percepts with general rules of knowledge
- Knowledge-based agents are more flexible than problem-solving agents
- Knowledge and inference may be represented in the language of logic
- *Planning* is defining a sequence of actions to achieve a goal

Knowledge-based agents

- Key component of an agent: *knowledge base* (KB), composed of sentences in knowledge representation language
- Sentences may be added to KB by *Tell* operation and queried by *Ask* operation
- KB may include background knowledge and percept-based knowledge
- KB may be queried for an action to perform

Knowledge engineering

- *Definition:* creation of a formal representation of a knowledge base of rules from human domain knowledge
- *Steps:*
 - Identify task
 - Assemble domain knowledge
 - Choose vocabulary (ontology) of domain
 - Encode general domain knowledge
 - Encode problem instances
 - Pose queries
 - Debug

Situation calculus

- *Situation*: a state of the environment
- All situations after the initial one (s_0) are results of actions, $Result(action, situation)$
- Result of a *sequence* of actions starting from a state s : $Result(seq, s) =$
 - $\{ s \text{ if } seq = \lambda$
 - $\{ Result(rest(seq), Result(first(seq), s)$
 - $\{ \text{ otherwise}$
- *Projection* is a task of deducing the result of a sequence of actions, enabling *planning*

Events, intervals, places

- *Event calculus* accounts for occurrences in space-time
- Events (e.g., WWII) may contain sub-events
- Events occur in *time intervals* and *places*
- Categories such as walking are *processes* or *liquid events*
- *State*: a process of continuous nonchange

Beliefs and mental events

- *Propositional attitudes*: relations such as *Believes, Knows, Wants*
- *Approaches*: modal logic, or syntactic theory that uses *strings* to represent beliefs
- *Denotation*("Clark") = Superman, *Name*(Superman) = "Clark"
- *Logical agents* are deemed able to make inferences from beliefs
- *Knowledge*: justified true belief

Planning

- *Classical planning units*: fully observable, deterministic, finite state, discrete
- Much planning relies on partial decomposition of problems
- Actions occur when preconditions are met
- *Plan construction* consists of search in space of possible actions for such a sequence
- *Challenge*: the *frame problem*, which consists of deciding what exactly are the changes in the situation caused by an action

2. Concepts and instances

- An *ontology* stores the vocabulary and meaning for a problem domain
- Real-world *objects* are instances of abstract *categories* or *concepts* that may have *measures* (length, weight, etc.)
- *Disjoint* categories *partition* larger categories
- *Concepts* and relationships among them may be represented by *semantic networks* or *case frames*

Ontological engineering

- *Ontology*: a collection of facts about the world chosen to be in a KB
- Knowledge is about things, actions, relationships, beliefs
- Ontological engineering is part of knowledge engineering and addresses the representation of abstract real-world concepts
- *Upper ontology* addresses most abstract concepts, as opposed to details and exceptions

Relationships among categories

- *Inheritance* (subset relation) helps organize categories in *taxonomies*
- *Containment* relationships define *PartOf* hierarchies for composite objects
- PIC

Semantic networks

- Binary relations are denoted by edges between category or object nodes
- Multiway (n -ary) relations are denoted using *reified* links – relationships made into objects
- Edges may be labeled with *attributes*, e.g., a flight has an origin and a destination
- Categories have default values; e.g., a person has two legs by default
- PIC

Associationist theories of meaning

- The meaning of an expression may derive from a set of associations with other things
- Associationist theories rely not on formal logic but on how humans relate concepts
- Humans seem to organize knowledge in inheritance hierarchies, e.g., canary → bird → animal
- *Semantic networks* can represent these associations in graphs

Case frames

- Research in 1970s sought to capture deep semantic aspects of language
- Fundamental relationships:
“Sarah fixed the chair with glue”
- *Agent*: Sarah; *Object*: chair; *Time*: past;
Instrument: glue
- Relationships are independent of sentences or language
- These relationships are part of the *formalism* rather than the *domain knowledge*

Frames

- A representational scheme to express implicit connections of concepts in a domain
- *Example:* a hotel room frame has a bed, bathroom, chair, phone, room number
- Corresponds to the notions of a *class*, a *database table*
- Early MIT research in frames led to object-oriented paradigm

Fuzzy logic and sets

- L. Zadeh, 1983, proposed theory of vagueness
- Rejects excluded middle: $x \in S \rightarrow x \notin (U - S)$
- Set characteristic function maps to 0 .. 1, denotes possibility or confidence measure
- *Examples:* $Few = \{1 (1.0), 2(1.0), 3 (0.9), \dots\}$
 $Tall\text{-}persons \cap Med\text{-}size\text{-}persons \neq \emptyset$
- Fuzziness is not uncertainty; *truth*, not *belief*, is quantified at between 0 and 1

3. Logical inference

- *Inference rule*: A validity-maintaining procedure for deriving sentences from other sentences
- *Proof procedure*: An inference rule and an algorithm for applying it to a set of sentences to yield a new sentence
- Inference is a way to add knowledge to a knowledge base monotonically

Propositional logic

- *Propositional logic* (propositional calculus) is a Boolean algebra with the values *True* and *False* and the operations \neg , \wedge , \vee , \rightarrow
- It is a *language of assertions* that each evaluate to *true* or *false*, including
 - *true*, *false*
 - Symbols (p , q , r , ...) that may stand for assertions such as “roses are red”
 - Negation: $\neg p$
 - Conjunction: $p \wedge q$
 - Disjunction: $p \vee q$
 - Implication: $p \rightarrow q \Leftrightarrow \neg q \vee p$

Predicate logic

- Predicate calculus (first-order logic, FOL) extends propositional calculus by enabling *quantifiers* and *predicates* (functions)
- A *predicate* is a Boolean function, i.e., it returns a truth value; it is a *property*
- *Examples:*
 - $(\forall x) x < x + 1$ universal quantifier
 - $prime(5)$ predicate denoting a property
 - $(\exists x) x > 1$ existential quantifier
- *Arity* of a predicate is its number of parameters

Applications of predicate logic

- Theorems are expressed and proven in FOL
- Algorithm verification uses predicate logic; e.g., $Sorted(A) \Leftrightarrow (\forall i < |A|) A[i] \leq A[i + 1]$
- Theory of formal languages and automata uses FOL
- Artificial intelligence uses FOL to express knowledge and to implement automated reasoning

Diagnostic vs. model-based reasoning

- *Diagnostic rules* enable inference from observations to hidden causes of the observed facts
- *Causal rules* for model-based reasoning form a model of the system

Variables in predicate logic

- In unquantified expressions $sum(x, 5)$, $odd(x)$, $prime(x)$, x is an *unbound placeholder* that stands for an unknown *constant*
- “ $sum(x, 5) = 8$ ”, “ $odd(x) = true$ ” are meaningless without having a value for x
- “ $x = 3 \rightarrow sum(x, 5) = 8$ ”, “ $x = 1 \rightarrow odd(x) = true$ ” are meaningful and true statements
- *Quantifiers* (\forall , \exists) bind variables: $(\forall x) P(x)$ means that every value x has property P
- *Boolean variables* are names for assertions

Inference and entailment

- Truth of a sentence with variables is relative to its *model* (set of variable assignments)
- *Entailment*: $\alpha \models \beta$ (in every model where α is true, β is true)
- *Inference* is finding a desired instance of entailment
- Expression x *logically follows* from a set S of sentences iff every interpretation that satisfies S also causes x to evaluate to *true*

Quantifiers

- For sentence S ,
 - $(\exists x) S$ is true iff some assignment under I has an assignment to x s.t. S is true
 - $(\forall x) S$ is true iff S is true for all assignments of values to x under I
- *Correspondence between quantifiers*:
 - $(\exists x) P(x) \Leftrightarrow (\neg \forall x) \neg P(x)$
 - $(\forall x) P(x) \Leftrightarrow (\neg \exists x) \neg P(x)$

Quantifiers and proofs

- *Rule of universal instantiation*: If P is a property of all of a set's elements, then it is a property of any particular element
- *Universal modus ponens*:
 $((\forall x) (P(x) \Rightarrow Q(x)) \wedge P(a)) \Rightarrow Q(a)$
- *Universal modus tollens* (used in proof by contradiction):
 $((\forall x) (P(x) \Rightarrow Q(x)) \wedge \neg Q(a)) \Rightarrow P(a)$

Examples

- If it doesn't rain, Rose will go to the park
 $\neg \text{weather}(\text{rain}) \rightarrow \text{go}(\text{Rose}, \text{park})$
- Spark is a Spaniel and a good dog
 $\text{isa}(\text{Spark}, \text{Spaniel}) \wedge \text{good-dog}(\text{Spark})$
- All basketball players are tall
 $(\forall x) (\text{basketball-player}(x) \rightarrow \text{tall}(x))$
- Nobody likes taxes
 $\neg (\exists x) (\text{likes}(x, \text{taxes}))$

Rules of inference in FOL

- *Universal instantiation*
 $(\forall x) P(x) \Rightarrow P(y)$ if y exists
- *Existential instantiation*
 $(\exists x) P(x) \Rightarrow P(y)$ for some y not defined already
- *Generalized Modus Ponens* (substitution lifting)
 $(\text{King}(\text{John}), \text{greedy}(y), (\text{King}(x) \wedge \text{greedy}(x)) \Rightarrow \text{evil}(x)) \Rightarrow \text{Evil}(\text{John})$
(Substituting *John* for x)
- *Unification*

Soundness and completeness

- *Sound*: an inference rule that produces from set S only sentences that logically follow from S
- *Complete*: a rule that can infer from S every sentence that logically follows from S
- *Examples*:
 - *Modus Ponens*: $(p \wedge (p \rightarrow q)) \rightarrow q$
 - *Modus Tollens*: $((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$
 - *And* elimination, *and* introduction, universal instantiation

Satisfiability (SAT)

- Given a formula ϕ in propositional logic (logic with \neg , \wedge , \vee , \Rightarrow , no predicates, no quantifiers), does a set of variable assignments exist that satisfies ϕ (makes ϕ true)?
- *Examples:*
 - (a) $p \wedge q \wedge r$
 - (b) $(p \wedge q \vee \neg q)$
 - (c) $p \wedge \neg(q \vee \neg q)$

Satisfiability of formulas

- Let $SAT = \{ \phi \mid (\exists \rho) \phi_\rho = \text{true} \}$ where ϕ is a formula in logic and ρ is a set of truth assignments
- *Example:* $\rho = \{ (p, \text{true}), (q, \text{true}), (r, \text{false}) \}$
- Then SAT is the set of formulas that are *satisfiable*
- Thus SAT is a *language*
- Equivalently, SAT is the *decision problem* of telling whether a formula is satisfiable (whether a formula is in the language SAT)
- We assign SAT to a *complexity class*

Sample proofs

Assuming these axioms, plus inference rules:

- a. p is true
- b. q is false
- c. $p \rightarrow r$

Then

- $p \rightarrow q$ (vacuously, using (a), (b))
- r is true (modus ponens, using (a) and (c))

Proof methods and their foundations

- **Induction**: Helps prove correctness
- **Contradiction** $(p \Rightarrow \neg p) \Rightarrow \neg p$
 - Subcategory: diagonal proofs (see www.framingham.edu/faculty/dkeil/diag-proof.pdf)
- **Construction**
 - Example*: We prove that a problem is in a certain complexity class by constructing an algorithm of that complexity
 - Subcategory: proof by **reduction**

Proof by induction

- Uses the *induction principle*: For a set A of natural numbers, if:
 - $0 \in A$, and
 - $(x \in A)$ implies $(x + 1) \in A$...then A is the set of all natural numbers
- To show that predicate P is true for all natural numbers, an inductive proof shows that P is true for 0 and that $P(x) \Rightarrow P(x + 1)$
- Here, P may be the claim that algorithm A works for a given natural-number input

Unification

- An algorithm for determining what substitutions are needed to make two sentences match
- *Cases for unify* (E_1, E_2):
 - $E_1 = E_2$: Return $\{ \}$ (no substitution)
 - E_1 is variable: return $\{E_2 / E\}$
 - E_2 is variable: return $\{E_1 / E_2\}$
 - Otherwise decompose E_1, E_2 , return a composition of unifications of the components

Unification

- *Definition:* finding substitutions that make different sentences look alike
- Formally: $Unify(p, q) = \theta$ where $Subst(\theta, p) = Subst(\theta, q)$
- *Example:*
 $Unify(Knows(John, x), Knows(John, Jane))$
 $= \{x \mid Jane\} = \theta$

Backward and forward chaining

- Forward chaining is data driven; applies Modus Ponens; is sound and complete
- Forward chaining generates all possible entailments
- Backward chaining is goal driven; works backward from query
- Deciding entailment with Horn clauses by forward chaining is $O(n)$ in size of knowledge base

4. Expert systems and resolution proof

Expert systems:

- Separate knowledge from control
- *Make inferences* from heuristic *knowledge base*
- Support updating of knowledge base
- Can report reasoning process
- Store knowledge as *inference rules*
- Use resolution proof method

Expertise

- Is inaccessible to conscious mind
- Knows *what to do* but not necessarily the internal mechanism of effect
- Is based on valid or invalid *models*
- Is subject to changes in domain area
- Can be represented in first-order logic

Resolution theorem proving

- Resolution begins by putting premises or axioms in *clause form*, a normal form using disjunctions of literals (atomic expressions or their negations)
- *Resolution rule*:
$$((p \vee q_1 \vee q_2 \vee \dots \vee q_m) \wedge (\neg p \vee r_1 \vee r_2 \vee \dots \vee r_n)) \Rightarrow q_1 \vee q_2 \vee \dots \vee q_m \vee r_1 \vee r_2 \vee \dots \vee r_n$$
- Complementary literals are used to generate a disjunction out of two disjunctions
- *Intuition*: If we know that in one case at least one of a list of assertions holds, and otherwise at least one of a different list of assertions holds, then we know that at least one of the entire list of assertions holds

Example of resolution

- *Goal*: To prove that Fido will die
- *Predicate form*
 $(\forall x) \text{dog}(x) \rightarrow \text{animal}(x)$
 $\text{dog}(\text{fido})$
 $(\forall y) \text{animal}(y) \rightarrow \text{die}(y)$
 $\therefore \text{die}(\text{fido})$ by *modus ponens*
- *Clause form (resolution)*:
 $\neg \text{dog}(x) \vee \text{animal}(x)$
 $\text{dog}(\text{fido})$
 $\neg \text{animal}(y) \vee \text{die}(y)$
Negate the assertion to be proven: $\neg \text{die}(\text{fido})$
This leads to a clash after substitutions lead to $\text{die}(\text{fido})$

Rule-based expert systems

- *If...then* rules may be written as *premise-conclusion* productions
- Rules “fire” when premise is satisfied
- Rules may be applied in forward (data driven) or goal-directed way
- Rule firing is equivalent to search of an *and/or* graph
- [pic of and-or graph, see Luger]

The Prolog language

- Programming in Logic (first-order predicate logic), U of Edinburgh, 1970s
- Interpreter makes inferences from assertions that comprise a program
- Central tool: resolution proof method
- Declarative programming constructs program as constraints rather than steps

Prolog control structure

- *To prove an assertion as a goal*
 - Try to unify goal with a fact
 - Rule out conjunction or return composition of unifications
 - Attempt to prove disjunction
 - Unify goal with conclusions
- This is the control structure of Prolog

Concepts

| | | | |
|-------------------|-------------------|-----------------|-------------------|
| arity | expert system | logic program | propositional |
| belief | first-order logic | model-based | calculus |
| belief revision | formula | reasoning | resolution proof |
| case frame | implication | modus ponens | satisfiability |
| case-based | inference rule | modus tollens | script |
| reasoning | inheritance | negation | semantic network |
| causal rule | interpretation | planning | situation |
| closed-world | knowledge | predicate | soundness |
| assumption | knowledge base | calculus | state |
| conjunction | knowledge | premise- | truth maintenance |
| diagnostic rules | engineering | conclusion | truth table |
| disjunction | knowledge | production | universal |
| domain knowledge | representation | Prolog | quantifier |
| embodied approach | knowledge- | proof procedure | unsatisfiable |
| entailment | based agent | | validity |

References

George Luger. *Artificial Intelligence*. Addison
Wesley, 2005.

Stuart Russell and Peter Norvig. *Artificial
Intelligence: A Modern Approach, 2nd ed.*,
Prentice Hall, 2003.