Knowledge Representation

CS 486/686: Introduction to Artificial Intelligence
Fall 2013

Outline

- Knowledge-based agents
- Logics in general
- Propositional Logic
- Reasoning with Propositional Logic
- First Order Logic

Introduction

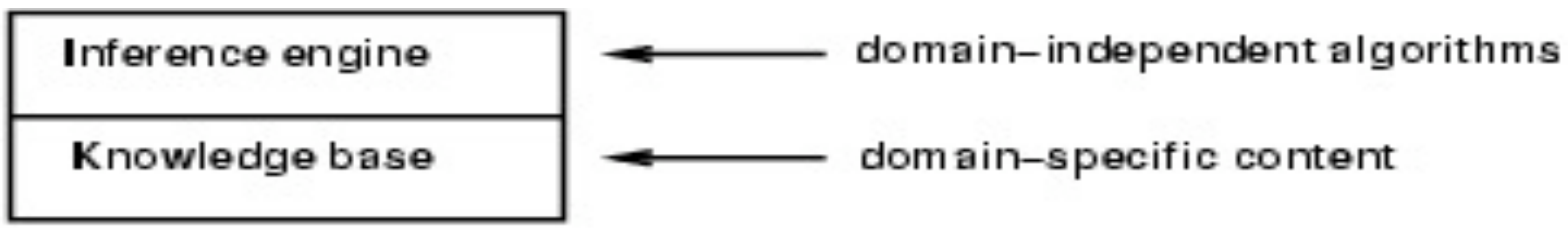
- So far we have taken the following approach
  - Figure out exactly what the problem is (problem definition)
  - Design an algorithm to solve the problem (search algorithm)
  - Execute the program

Knowledge-Based Agents

- An alternative approach
  - Identify the knowledge needed to solve the problem
  - Write down this knowledge in some language
  - Use logical consequences to solve the problem

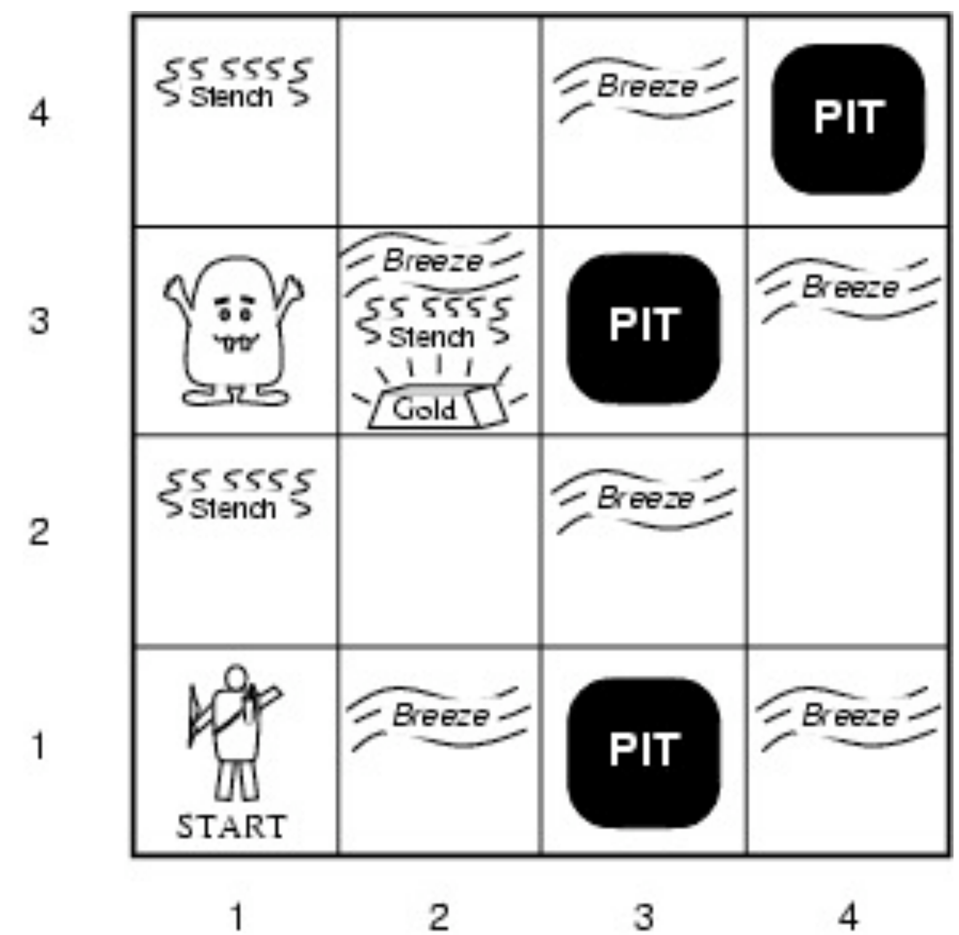
Knowledge-Based Agents

- Ideally
  - We tell the agent what it needs to know
  - The agent infers what to do and how to do it
- Agent has two parts
  - **Knowledge base**: Set of facts expressed in a formal standard language
  - **Inference engine**: Rules for deducing new facts

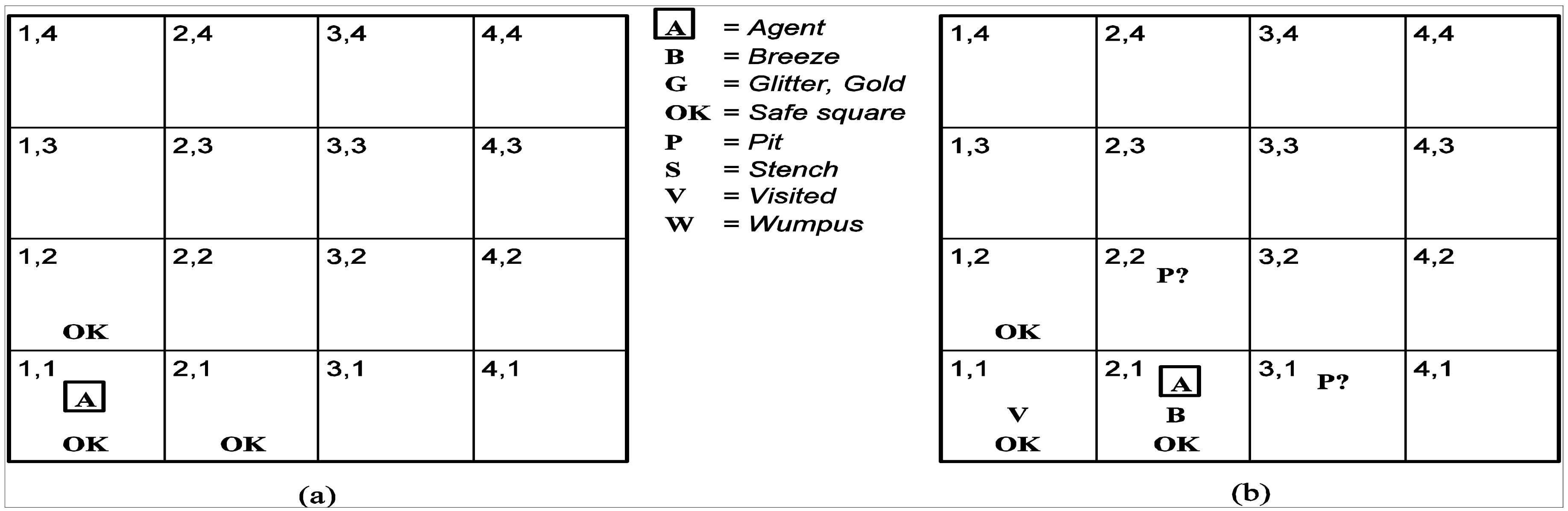


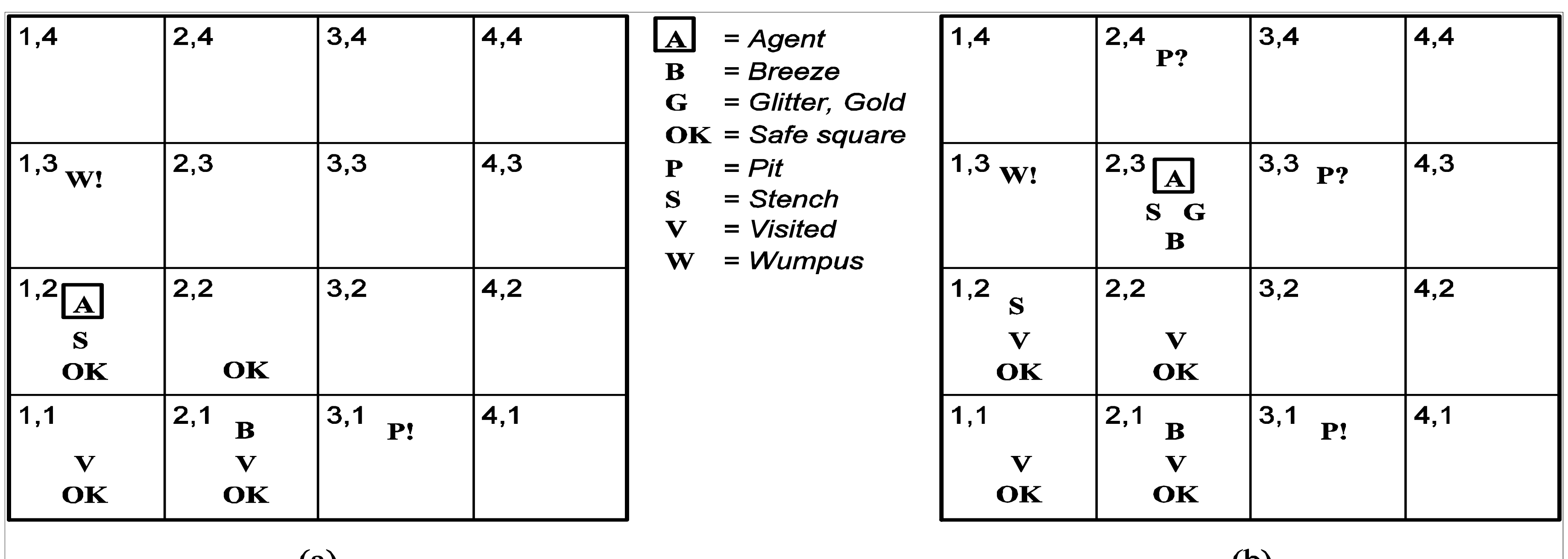
An Example: Wumpus World

- **Goal**:
  - Get gold back to start without falling into a pit or getting eaten by the wumpus
- **Environment**
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- **Sensors**: Stench, Breeze, Glitter, Bump, Scream
- **Actuators**: Left turn, Right turn, Forward, Grab, Release, Shoot



Wumpus World





What Is A Logic?

- Logic
  - A formal language for representing information so that conclusions can be drawn
- Logics have 2 components
  - **Syntax**: defines the sentences of the language
  - **Semantics**: defines the meaning of the sentences

Entailment

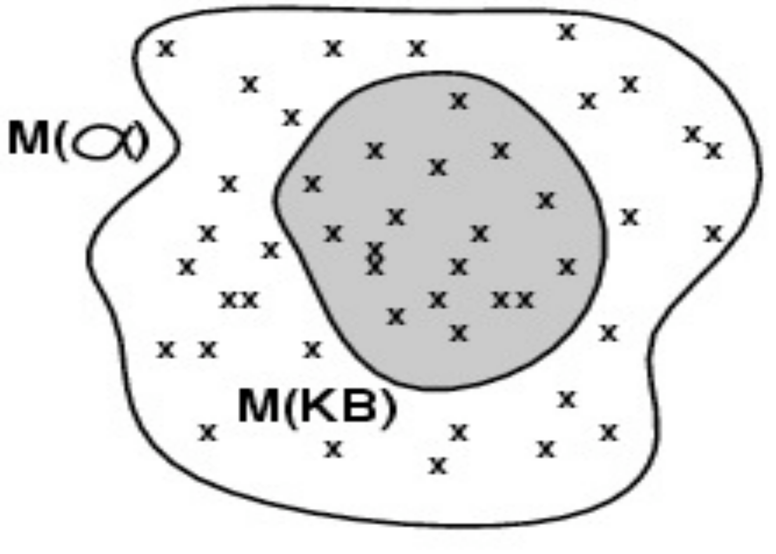
- Entailment means that "one thing follows from another"
    - KB |= α
- Knowledge base (KB) **entails** sentence α if and only if α is true in all possible worlds where KB is true
- Example:
    - KB: I finished the AI assignment. I am happy
    - α: I finished the AI assignment and I am happy.

Models

- A model is a formal "possible world" where a sentence can be evaluated
  - m is a model of sentence α if α is true in m
- M(α) is the set of all models of α
- KB l=α if and only of M(KB)⊆M(α)

KB: I finished the AI homework and I did not sleep last night

α : I finished the AI homework



Inference

- Given a KB, we want to be able to draw conclusions from it
- **Inference procedure:** KB $\vdash_i \alpha$
  - Sentence α can be derived from KB by inference algorithm
- Desired properties:
  - **Soundness:** the procedure only infers true statements
    - If KB $\vdash_i \alpha$ then KB $\models \alpha$
  - **Completeness**: the procedure can generate all true statements
    - IF KB $\models \alpha$ then it is true that KB $\vdash_i \alpha$

Propositional Logic

- **Atomic Symbols**: P, Q, R,...
  - Each symbol stands for a proposition that can be either True or False
- **Logical Connectives**
  - $\neg$ (negation)
  - $\vee$ (or)
  - $\wedge$ (and)
  - $\Rightarrow$ (implies)
  - $\Leftrightarrow$ (if and only if, equivalence)

Propositional Logic: Syntax

- **Grammar rules:**
  - Sentence → AtomicSentence | ComplexSentence
  - Atomic Sentence → True | False | Symbol
  - Symbol → P | Q | R | …
  - ComplexSentence → Sentence |¬ Sentence
    | (Sentence ∨ Sentence)
    |(Sentence ∧ Sentence)
    | (Sentence ⇒ Sentence)
    |(Sentence ⇔ Sentence)

Propositional Logic: Semantics

- The semantics of propositional logic are defined by a **truth table**
  - Symbols are mappings to an element in domain {0,1}

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Example: Propositional Logic

- Note that $P \Rightarrow Q$ is the same as $\neg P \vee Q$

P	Q	$\neg P$	$\neg P \vee Q$	$P \Rightarrow Q$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Exercise: Show that $P \Leftrightarrow Q$ is the same as $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

Entailment: Propositional Logic

- Let
  - $KB=(P \vee R) \wedge (Q \vee \neg R)$
  - $\alpha=P \vee R$

- Does $KB \models \alpha$?

Entailment

- Check all possible models
  - α must be true when ever KB is true

P	Q	R	$P \vee R$	$Q \vee \neg R$	KB	α
0	0	0	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Inference: Propositional Logic

- Using truth tables is
  - **Sound**: direct definition of entailment
  - **Complete**: works for any KB and α and always terminates
- But...
  - Really inefficient
  - If there are n symbols, then there are 2^n models

More Terminology

- Sentences α and β are **logically equivalent** if they are true in the same set of models
  - $\alpha \Leftrightarrow \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$
- **Deduction Theorem:**
  - For any sentences α and β, $\alpha \models \beta$ if and only $(\alpha \Rightarrow \beta)$ is valid
- Useful Result (**Proof by Contradiction**):
  - $\alpha \models \beta$ if and only if the sentence $(\alpha \wedge \neg \beta)$ is unsatisfiable

Logical Equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Inference Rules

- Given a KB we want to derive conclusions
  - Proof: sequence of inference rule applications

Modus Ponens

$$\frac{\alpha, \alpha \Rightarrow \beta}{\beta}$$

Resolution

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

And Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Unit Resolution

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Inference

- Modus Ponens and And-Elimination together are sound

Example: KB= "If you are in AI class then you are happy and paying attention", "You are in AI class"

Modus Ponens: "You are happy and you are paying attention"

And-Elimination: "You are happy"

Resolution

- Resolution is a **sound** and **complete** inference rule
  - Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.

Resolution

- Resolution is a **sound** and **complete** inference rule
  - Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.

Caveat: Given that α is true, we can not automatically generate $\alpha \vee \beta$ is true. However, we can find the answer to the question "Is $\alpha \vee \beta$ true".

Conjunctive Normal Form

- Resolution is applied to clauses of the form $\alpha \vee \beta \vee \ldots \vee \gamma$
- Any clause in propositional logic is logically equivalent to a clause in CNF
  - conjunction of disjunctions
  - eg. $(P \vee \neg Q \vee R) \wedge (\neg Q \vee A \vee B) \wedge \ldots$

Converting to CNF

1. Eliminate $\Leftrightarrow$, replacing $P \Leftrightarrow Q$ with $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$
2. Eliminate $\Rightarrow$, replacing $P \Rightarrow Q$ with $\neg P \vee Q$
3. Move "$\neg$" inwards, using $\neg(\neg P) = P$, $\neg(P \wedge Q) = \neg P \vee \neg Q$ and $\neg(P \vee Q) = \neg P \wedge \neg Q$
4. Distribute $\vee$ over $\wedge$ where possible

Resolution Algorithm

- Recall: To show KB$\models\alpha$, we show that (KB$\wedge\neg\alpha$) is unsatisfiable
- **Resolution Algorithm:**
  - Convert (KB$\wedge\neg\alpha$) to CNF
  - For every pair of clauses that contain complementary literals
    - Apply resolution to produce a new clause
    - Add new clause to set of clauses
    - Continue until
      - No new clauses are being added (KB does not entail α) or
      - Two clauses resolve to produce empty clause (KB$\models\alpha$)

Complexity of Inference

- Inference for propositional logic is NP-complete
- If all clauses are **Horn clauses**, then inference is linear in size of KB!
  - Horn clause: Disjunction of literals where at most one literal is positive
    - $\neg P \vee Q \vee \neg R$ is a Horn clause
    - $P \vee Q \vee R$ is not a Horn clause
  - Every Horn clauses establishes exactly one new fact
    - $\neg P \vee Q \vee \neg R \Leftrightarrow (P \wedge R) \Rightarrow Q$
    - We add all new facts in n passes

Forward Chaining

- When a new sentence α is added to the KB
  - Look for all sentences that share literals with α
  - Perform resolution
  - Add new sentence to KB and continue
- Forward chaining is
  - Data-driven
  - Eager: new facts are inferred as soon as possible

Backward Chaining

- When a query q is asked of the KB
  - If q is in the KB, return True
  - Otherwise, use resolution for q with other sentences in the KB and continue from result
- Backward chaining is
  - Goal driven: Centers reasoning around query being asked
  - Lazy: new facts are inferred only when needed

Forward vs Backward

- Which is better? That depends!
- Backward Chaining:
  - Does not grow the KB as much
  - Focused on proof so is generally more efficient
  - Does nothing until a question is asked
  - Typically used in proofs by contradiction

Forward vs Backward

- Forward Chaining
  - Extends the KB and improves understanding of the world
  - Typically used in tasks where the focus is on providing a model of the world

First Order Logic

- New elements
  - Predicates
    - Define objects, properties, relationships
  - Quantifiers
    - $\forall$ (for all), $\exists$ (there exists) are used in statements that apply to a class of objects
- Example: $\forall x \; On(x, Table) \Rightarrow Fruit(x)$

Sentences

- Terms
  - Constants, variables, function($term_1$,...,$term_n$)
- Atomic Sentences
  - Predicate($term_1$,$term_2$), $term_1$=$term_2$
- Complex Sentences
  - Combine atomic sentences with connectives
  - Likes(Alice, IceCream)$\wedge$Likes(Bob, IceCream)

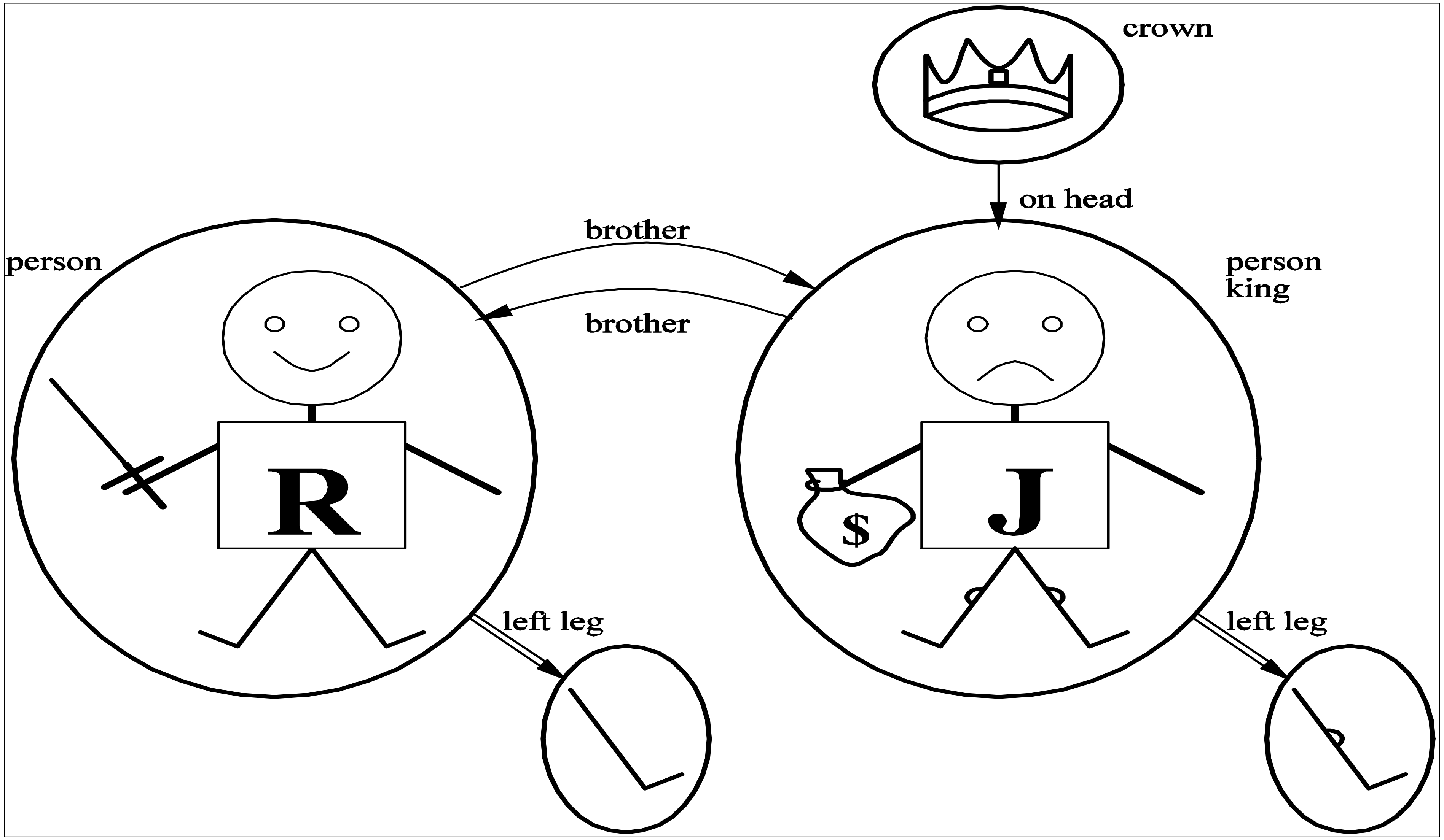
Semantics

- Sentences are true with respect to their interpretation
  - Model contains objects and relations among them
  - Interpretation specifies referents for
    - Constant symbols (objects)
    - Predicate symbols (relations)
    - Function symbols (functional relations)

Semantics

- Atomic sentence Predicate($term_1$,...,$term_n$) is true if and only if the relation referred to by Predicate holds for objects $term_1$,...$term_n$

Semantics



crown
on head
brother
brother
person
person
king
R
J
$
left leg
left leg

Universal Quantification

- Form: ∀<variables> <sentence>
  - Everyone taking AI is smart
  - $\forall x$ Taking(x,AI)⇒Smart(x)
- Equivalent to
  - (Taking(Alice, AI)⇒Smart(Alice))∧(Taking(Bob, AI)⇒Smart(Bob))...
- Example: $\forall x$ Taking(x,AI) ∧Smart(x)
  - Why is this unlikely not what you mean?
- **Typically ⇒ is the main connector!**

Existential Quantification

- Form: $\exists x$<variables><sentence>
  - Someone is taking AI is smart
  - $\exists x\ Taking(x,AI) \wedge Smart(x)$
- Equivalent to
  - $(Taking(Alice, AI) \wedge Smart(Alice)) \vee (Taking(Bob,AI) \wedge Smart(Bob)) \vee \ldots$
- Example: $\exists x\ Taking(x,AI) \Rightarrow Smart(x)$
  - Why is this unlikely to be what you want?
- **Typically $\wedge$ is the main connector**

Properties of Quantifiers

- Basic Rules
  - $\forall x \forall y$ is the same as $\forall y \forall x$
  - $\exists x \exists y$ is the same as $\exists y \exists x$
  - $\forall x \exists y$ is not the same as $\exists y \forall x$
- Example
  - $\exists x \forall y$ Loves (x,y)
  - $\forall y \exists x$ Loves(x,y)

Quantifier Duality

- Each quantifier can be expressed using the other quantifier and negation
  - $\forall x$ Likes(x,Broccoli)
  - $\neg\exists x \neg$Likes(x,Broccoli)

  - $\exists x$ Likes(x,Broccoli)
  - $\neg\forall x \neg$Likes(x,Broccoli)

Inference and FOL

- We know how to do inference in Propositional Logic: find α such that KBl =α
  - Is it possible to use these techniques for FOL?
  - Have to handle quantifiers, predicates, functions, ...

Universal Instantiation

- Given sentence $\forall x\ P(x) \wedge Q(x) \Rightarrow R(x)$ then we want to infer $P(John) \wedge Q(John) \Rightarrow R(John)$ and $P(Anne) \wedge P(Anne) \Rightarrow R(Anne)$ and ...

<u>Universal Instantiation (UI)</u>

v is a variable

α is a sentence

$$\frac{\forall v \alpha}{\mathsf{SUBST}(\{v/g\}\alpha)}$$

Substitute g for all occurrences of v in α

g is a ground term*

Universal Instantiation

- A **ground term** is a term without variables
- SUBST(θ,α) is the result of applying substitution θ to sentence α
- Example
  - SUBST({x/John},$\forall$ x P(x)$\wedge$Q(x)$\Rightarrow$ R(x)) = P(John)$\wedge$ Q(John)$\Rightarrow$R(John)
  - SUBST({x/Father(John)},$\forall$ x P(x)$\wedge$Q(x)$\Rightarrow$ R(x))
    =P(Father(John))$\wedge$Q(Father(John))$\Rightarrow$R(Father(John))

Existential Instantiation

- For any sentence α, variable v and constant symbol K **that does not appear anywhere in the KB**

Skolum Constant

$$\frac{\exists v \alpha}{\mathrm{SUBST}(\{x/K\}, \alpha)}$$

<u>Example</u>

∃ x Crown(x) yields

Crown(C_1) (C_1 is a new constant)

Reduction to Propositional Inference

- Suppose the KB contained the following
  - $\forall x\ Cat(x) \wedge Orange(x) \Rightarrow Cute(x)$
  - Orange(Kitty)
  - Cat(Kitty)
  - Sister(Kitty, Katy)
- Instantiating the universal sentence in all possible ways we have a new KB:
  - $Cat(Kitty) \wedge Orange(Kitty) \Rightarrow Cute(Kitty)$
  - $Cat(Katy) \wedge Orange(Katy) \Rightarrow Cute(Katy)$
  - Cat(Kitty)
  - Sister(Kitty, Katy)
- The new KB is in propositional form. The symbols are
  - Cat(Kitty), Cat(Katy), Orange(Kitty), Cute(Katy), Sister(Kitty,Katy), …

Example

- KB: Bob is a buffalo. Pat is a pig. Buffalos are faster than pigs.

Reduction Continued

- Every FOL KB can be propositionalized
  - Transformed into propositional logic
- This preserves entailment
  - A ground sentence is entailed by the new KB if and only if it was entailed in the original KB
- Thus we can apply resolution (sound and complete) and return the result?

Reduction Continued

- Problem: Functions
  - The set of possible ground substitutions can be infinite
  - Example: Assume the KB contains function Mother(x)
    - SUBST({x|John},Mother(x))=Mother(John)
    - SUBST({x|Mother(John)},Mother(x))=Mother(Mother(John))
    - SUBST({x| Mother(John)},Mother(Mother(x))=Mother(Mother(Mother(John)))

Reduction Continued

- **Theorem** (Herbrand 1930): If a sentence is entailed by a FOL KB, then it is entailed by a finite subset of the propositionalized KB.
- **Idea**: for n=0 to ∞
  - Create a propositional KB by instantiating with depth n terms
  - Check if α is entailed by this KB. If yes, then stop.

Reduction Continued

- **Problem**: Works if α is entailed by the KB but it loops forever if α is not entailed
- **Theorem**: (Turing 1936, Church 1936) Entailment in FOL is semi-decidable.
  - Algorithms exist that say yes to every entailed sentence
  - No algorithm exists that says no to every unentailed sentence

Problems with Propositionalization

- Problem is with universal instantiation
  - Generates many irrelevant sentences due to substitutions
- **Idea**: Find a substitution that makes different logical statements look identical
  - Unification

Unification

- Unify algorithm
  - Takes two sentences and returns a unifier if one exists
  - Unify(p,q)=θ where SUBST(θ,p)=SUBST(θ,q)
    - θ is the Unifier

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,Paul)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,Paul)	

Generalized Modus Ponens

- **Conditions**: Atomic sentences p_i, p_i' and q where there is a substitution θ such that SUBST(θ,p_i)=SUBST(θ,p_i')

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Rightarrow q}{\mathsf{SUBST}(\theta, q)}$$

Caveats:

GMP used with a KB of definite clauses (exactly one positive literal).

All variables are assumed to be universally quantified

Inference Algorithms

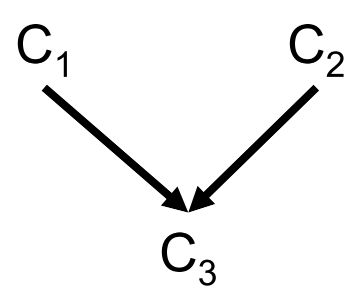
- You can now use
  - Forward chaining
  - Backward chaining
  - Resolution

Forward Chaining Example

Backward Chaining Example

Resolution Review

- Resolution is a refutation procedure
  - To prove KB |=α show that KB∧¬α is unsatisfiable
- Resolution used KB, ¬α in CNF
- Resolution inference rule combines two clauses to make a new one



Inference continues until an empty clause is derived (contradiction)

Resolution

$$\frac{l_1 \vee \cdots \vee l_k, m_1 \vee \cdots m_n}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{i-1} \vee m_{i+1} \vee \cdots \vee m_n)\theta}$$

Where Unify(l_i,: m_i)=θ

The two clauses, l_i and m_i, are assumed to be standardized apart so that they share no variables

<u>Example</u>

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(John)}{Unhappy(John)}$$

Unhappy(John) with θ = {x/John}

Converting to CNF

- Example $\forall x[\forall y\, A(y) \Rightarrow L(x,y)] \Rightarrow [\exists y\, L(y,x)]$
- Eliminate $\Leftrightarrow$ and $\Rightarrow$
  - $\forall x[\neg\forall y \neg A(y) \vee L(x,y)] \vee [\exists y\, L(y,x)]$
- Move $\neg$ inwards
  - $\forall x[\exists y A(y) \wedge \neg L(x,y)] \vee [\exists y\, L(y,x)]$
- Standardize variables
  - $\forall x[\exists y A(y) \wedge \neg L(x,y)] \vee [\exists z\, L(z,x)]$
- Skolemize
  - $\forall x[A(F(x)) \wedge \neg L(x,F(x))] \vee [\, L(G(x),x)]$
- Drop universal quantifiers
  - $[A(F(x)) \wedge \neg L(x,F(x))] \vee [\, L(G(x),x)]$
- Distribute $\vee$ over $\wedge$
  - $[A(F(x)) \vee L(G(x),x)] \wedge [\neg L(x,F(x)) \vee L(G(x),x)]$

Resolution Example

- Marcus is a person
- Marcus is a Pompeian
- All Pompeians are Roman
- Caesar is a ruler
- All Romans are either loyal to Caesar or hate Caesar
- Everyone is loyal to someone
- People only try to assassinate rulers they are not loyal to
- Marcus tries to assassinate Caesar
- Query: Does Marcus hate Caesar?

Conclusion

- Syntax, semantics, entailment and inference
- Propositional logic and FOL
- Understand how forward-chaining, backward-chaining and resolution work