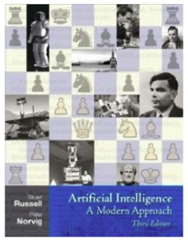
Logic and AI

- Logical Agents
- Propositional Logic
  - Syntax
  - Semantics
  - Resolution
- First Order Logic
  - Syntax
  - Semantics
  - Unification
  - Skolem
  - Resolution

Material from
Russell & Norvig,
chapters 7-9



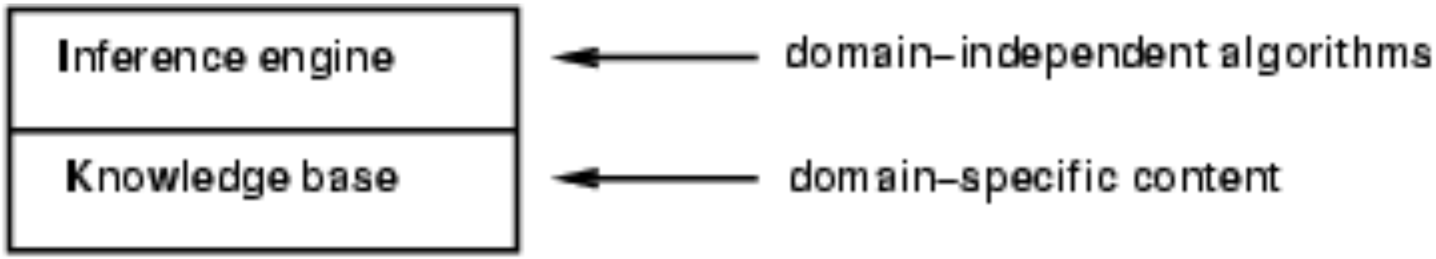
Many slides based on
Russell & Norvig's slides
[Artificial Intelligence:
A Modern Approach]

Some based on Slides by
Vincent Conitzer and Sriraam Natarajan

Logic and AI

- Would like our AI to have knowledge about the world, and logically draw conclusions from it
- Search algorithms generate successors and evaluate them, but do not “understand” much about the setting
- Example question: is it possible for a chess player to have 8 pawns and 2 queens?
  - Search algorithm could search through tons of states to see if this ever happens, but…

Knowledge bases



- Knowledge base = set of sentences in a formal language
- Declarative approach to building an agent (or other system):
  - `Tell` it what it needs to know
- Then it can `Ask` itself what to do - answers should follow from the KB
- Agents can be viewed at the knowledge level
  i.e., what they know, regardless of how implemented

A simple knowledge-based agent

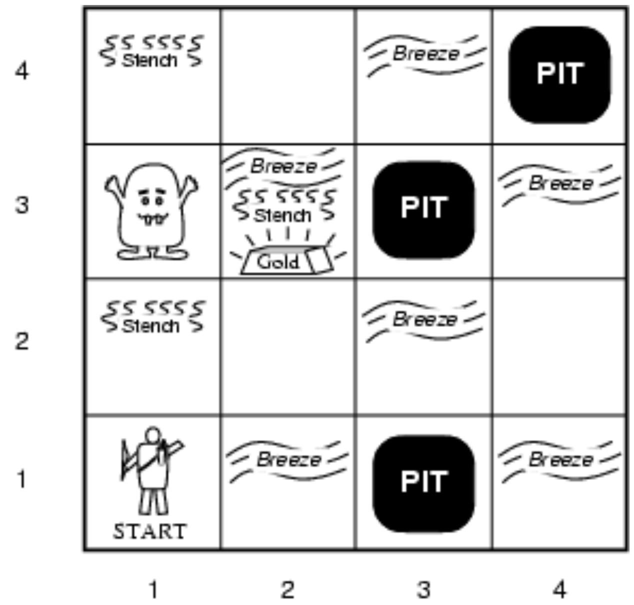
```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Wumpus World PEAS description

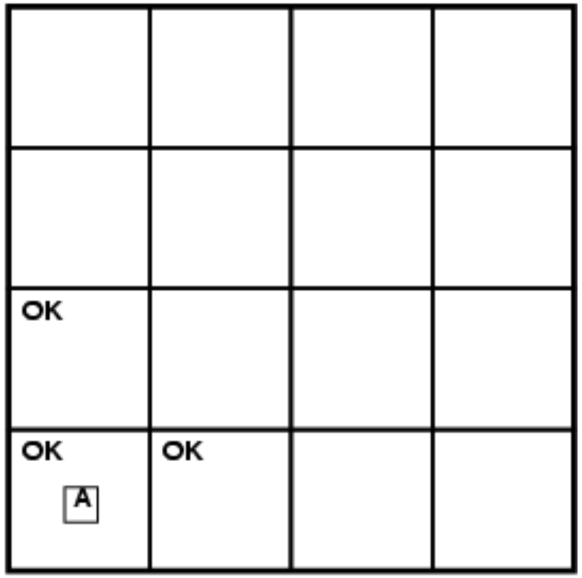
- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the 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



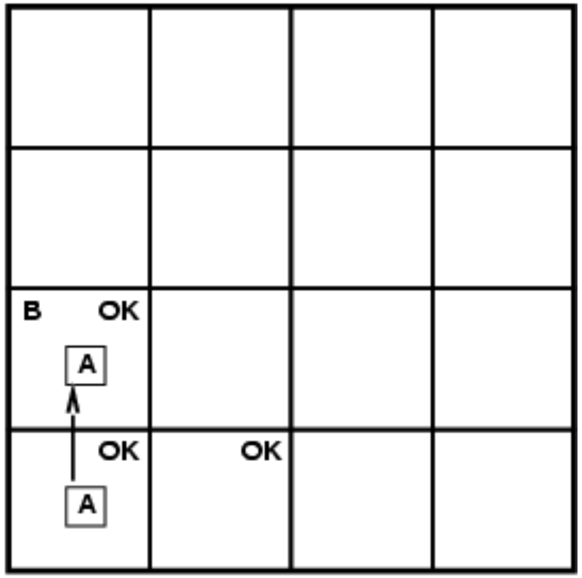
Wumpus world characterization

- Fully Observable No – only local perception
- Deterministic Yes – outcomes exactly specified
- Episodic No – sequential at the level of actions
- Static Yes – Wumpus and Pits do not move
- Discrete Yes
- Single-agent? Yes – Wumpus is essentially a natural feature

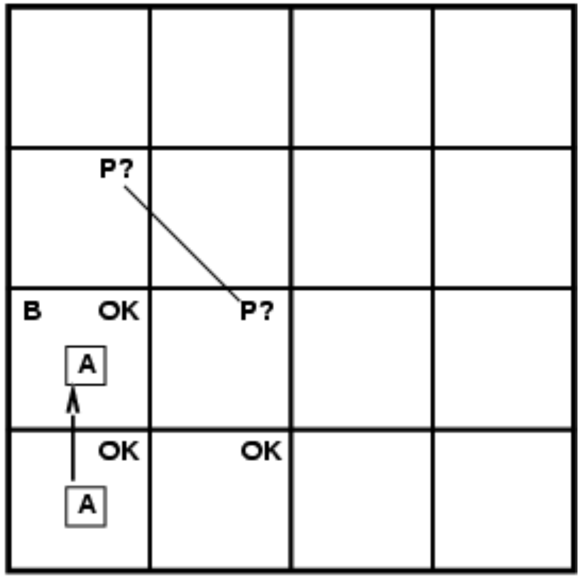
Exploring a wumpus world



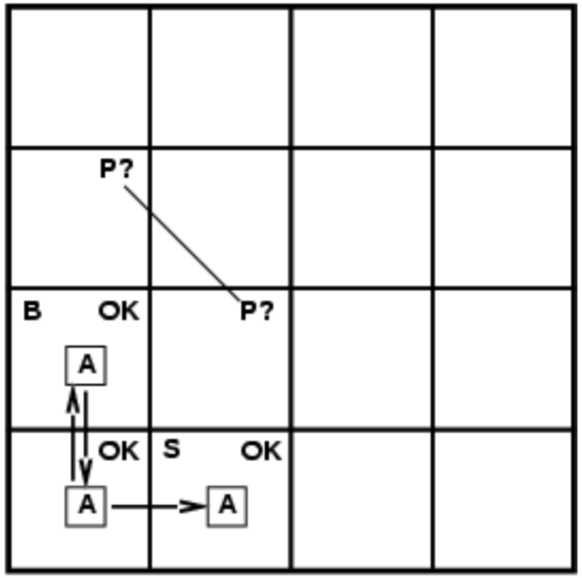
Exploring a wumpus world



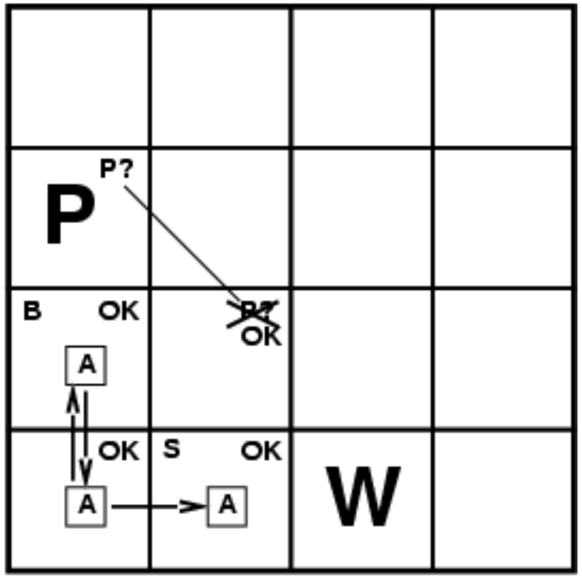
Exploring a wumpus world



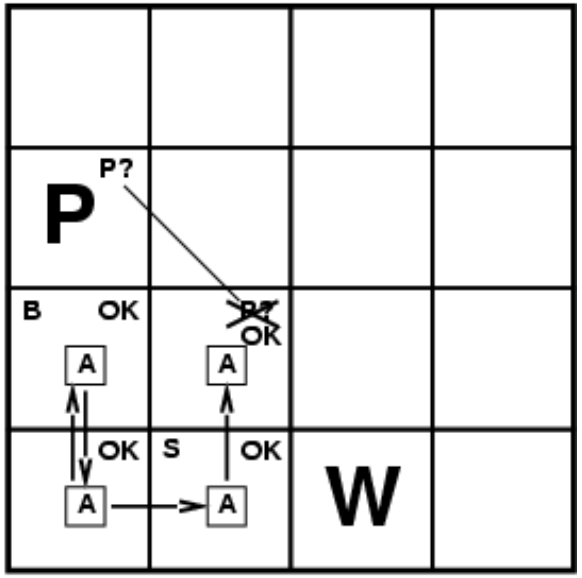
Exploring a wumpus world



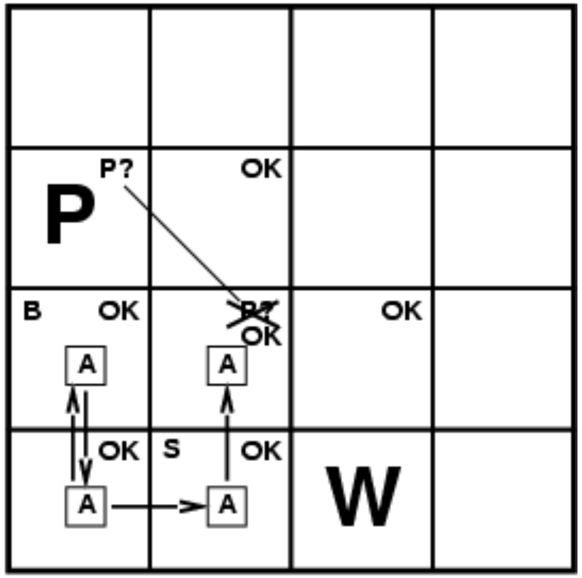
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



A "roommate" story

- You roommate comes home; he/she is completely wet
- You know the following things:
  - Your roommate is wet
  - If your roommate is wet, it is because of rain, sprinklers, or both
  - If your roommate is wet because of sprinklers, the sprinklers must be on
  - If your roommate is wet because of rain, your roommate must not be carrying the umbrella
  - The umbrella is not in the umbrella holder
  - If the umbrella is not in the umbrella holder, either you must be carrying the umbrella, or your roommate must be carrying the umbrella
  - You are not carrying the umbrella
- **Can you conclude that the sprinklers are on?**
- **Can AI conclude that the sprinklers are on?**

Knowledge base for the roommate story

- RoommateWet
- RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
- RoommateWetBecauseOfSprinklers => SprinklersOn
- RoommateWetBecauseOfRain => NOT(RoommateCarryingUmbrella)
- UmbrellaGone
- UmbrellaGone => (YouCarryingUmbrella OR RoommateCarryingUmbrella)
- NOT(YouCarryingUmbrella)

Syntax

- What do well-formed sentences in the knowledge base look like?
- A BNF grammar:
- *Symbol* → P, Q, R, …, RoommateWet, …
- *Sentence* → True | False | *Symbol* | NOT(*Sentence*) | (*Sentence* AND *Sentence*) | (*Sentence* OR *Sentence*) | (*Sentence* => *Sentence*)
- We will drop parentheses sometimes, but formally they really should always be there

Semantics

- A model specifies which of the proposition symbols are true and which are false
- Given a model, I should be able to tell you whether a sentence is true or false
- Truth table defines semantics of operators:

a	b	NOT(a)	a AND b	a OR b	a => b
false	false	true	false	false	true
false	true	true	false	true	true
true	false	false	false	true	false
true	true	false	true	true	true

- Given a model, can compute truth of sentence recursively with these

Caveats

- TwoIsAnEvenNumber OR ThreeIsAnOddNumber
  is true (not exclusive OR)
- TwoIsAnOddNumber => ThreeIsAnEvenNumber
  is true (if the left side is false it's always true)

All of this is assuming those symbols are assigned their natural values…

Tautologies

- A sentence is a tautology if it is true for any setting of its propositional symbols

P	Q	P OR Q	NOT(P) AND NOT(Q)	(P OR Q) OR (NOT(P) AND NOT(Q))
false	false	false	true	true
false	true	true	false	true
true	false	true	false	true
true	true	true	false	true

- (P OR Q) OR (NOT(P) AND NOT(Q)) is a tautology

Is this a tautology?

- (P => Q) OR (Q => P)

Logical equivalences

- Two sentences are logically equivalent if they have the same truth value for every setting of their propositional variables

P	Q	P OR Q	NOT(NOT(P) AND NOT(Q))
false	false	false	false
false	true	true	true
true	false	true	true
true	true	true	true

- P OR Q and NOT(NOT(P) AND NOT(Q)) are logically equivalent
- Tautology = logically equivalent to True

Famous logical equivalences

they can be used for rewriting and simplifying rules

- (a OR b) ≡ (b OR a) *commutatitvity*
- (a AND b) ≡ (b AND a) *commutatitvity*
- ((a AND b) AND c) ≡ (a AND (b AND c)) *associativity*
- ((a OR b) OR c) ≡ (a OR (b OR c)) *associativity*
- NOT(NOT(a)) ≡ a *double-negation elimination*
- (a => b) ≡ (NOT(b) => NOT(a)) *contraposition*
- (a => b) ≡ (NOT(a) OR b) *implication elimination*
- NOT(a AND b) ≡ (NOT(a) OR NOT(b)) *De Morgan*
- NOT(a OR b) ≡ (NOT(a) AND NOT(b)) *De Morgan*
- (a AND (b OR c)) ≡ ((a AND b) OR (a AND c)) *distributitivity*
- (a OR (b AND c)) ≡ ((a OR b) AND (a OR c)) *distributitivity*

Wumpus world sentences

- Let $P_{i,j}$ be true if there is a pit in [i, j].
- Let $B_{i,j}$ be true if there is a breeze in [i, j].

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

- "Pits cause breezes in adjacent squares"

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Inference

- We have a knowledge base of things that we know are true
  - RoommateWetBecauseOfSprinklers
  - RoommateWetBecauseOfSprinklers => SprinklersOn
- Can we conclude that SprinklersOn?
- We say SprinklersOn is entailed by the knowledge base if, for every setting (models) of the propositional variables for which the knowledge base is true, SprinklersOn is also true

RWBOS	SprinklersOn	Knowledge base
false	false	false
false	true	false
true	false	false
true	true	true

- SprinklersOn is entailed!

Wumpus Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

Simple algorithm for inference

- Want to find out if sentence a is entailed by knowledge base…
- *Go through the possible settings of the propositional variables,*
  - *If knowledge base is true and a is false, return false*
- *Return true*
- Not very efficient: $2^{\#\text{propositional variables}}$ settings

Inconsistent knowledge bases

- Suppose we were careless in how we specified our knowledge base:
  - PetOfRoommateIsABird => PetOfRoommateCanFly
  - PetOfRoommateIsAPenguin => PetOfRoommateIsABird
  - PetOfRoommateIsAPenguin => NOT(PetOfRoommateCanFly)
  - PetOfRoommateIsAPenguin
- It entails both PetOfRoommateCanFly and NOT(PetOfRoommateCanFly)
- Therefore, technically, this knowledge base implies anything: The Moon Is Made Of Cheese

Reasoning patterns

- Obtain new sentences directly from some other sentences in knowledge base according to reasoning patterns
- If we have sentences a and a => b , we can correctly conclude the new sentence b
  - This is called modus ponens
- If we have a AND b , we can correctly conclude a
- All of the logical equivalences from before also give reasoning patterns

Formal proof that the sprinklers are on

1) RoommateWet
2) RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
3) RoommateWetBecauseOfSprinklers => SprinklersOn
4) RoommateWetBecauseOfRain => NOT(RoommateCarryingUmbrella)
5) UmbrellaGone
6) UmbrellaGone => (YouCarryingUmbrella OR RoommateCarryingUmbrella)
7) NOT(YouCarryingUmbrella)

Knowledge Base

---

8) YouCarryingUmbrella OR RoommateCarryingUmbrella *(modus ponens on 5 and 6)*
9) NOT(YouCarryingUmbrella) => RoommateCarryingUmbrella *(equivalent to 8)*
10) RoommateCarryingUmbrella *(modus ponens on 7 and 9)*
11) NOT(NOT(RoommateCarryingUmbrella) *(equivalent to 10)*
12) NOT(NOT(RoommateCarryingUmbrella)) => NOT(RoommateWetBecauseOfRain) *(equivalent to 4 by contraposition)*
13) NOT(RoommateWetBecauseOfRain) *(modus ponens on 11 and 12)*
14) RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers *(modus ponens on 1 and 2)*
15) NOT(RoommateWetBecauseOfRain) => RoommateWetBecauseOfSprinklers *(equivalent to 14)*
16) RoommateWetBecauseOfSprinklers *(modus ponens on 13 and 15)*
17) SprinklersOn *(modus ponens on 16 and 3)*

Reasoning about penguins

1) PetOfRoommateIsABird => PetOfRoommateCanFly
2) PetOfRoommateIsAPenguin => PetOfRoommateIsABird
3) PetOfRoommateIsAPenguin => NOT(PetOfRoommateCanFly)
4) PetOfRoommateIsAPenguin
5) PetOfRoommateIsABird *(modus ponens on 4 and 2)*
6) PetOfRoommateCanFly *(modus ponens on 5 and 1)*
7) NOT(PetOfRoommateCanFly) *(modus ponens on 4 and 3)*
8) NOT(PetOfRoommateCanFly) => FALSE *(equivalent to 6)*
9) FALSE *(modus ponens on 7 and 8)*
10) FALSE => TheMoonIsMadeOfCheese *(tautology)*
11) TheMoonIsMadeOfCheese *(modus ponens on 9 and 10)*

Getting more systematic

- Any knowledge base can be written as a single formula in conjunctive normal form (CNF)
  - CNF formula: (… OR … OR …) AND (… OR …) AND …
  - … can be a symbol x, or NOT(x) (these are called literals)
  - Multiple facts in knowledge base are effectively ANDed together

RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)

becomes

(NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)

Converting story problem to conjunctive normal form

- RoommateWet
  - RoommateWet
- RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
  - NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
- RoommateWetBecauseOfSprinklers => SprinklersOn
  - NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
- RoommateWetBecauseOfRain => NOT(RoommateCarryingUmbrella)
  - NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
- UmbrellaGone
  - UmbrellaGone
- UmbrellaGone => (YouCarryingUmbrella OR RoommateCarryingUmbrella)
  - NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
- NOT(YouCarryingUmbrella)
  - NOT(YouCarryingUmbrella)

Unit resolution

If we have

- l_1 OR l_2 OR … OR l_k

and

- NOT(l_i)

we can conclude

- l_1 OR l_2 OR … l_{i-1} OR l_{i+1} OR … OR l_k
- Basically modus ponens

Applying resolution to story problem

1) RoommateWet
2) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
3) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
4) NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
5) UmbrellaGone
6) NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
7) NOT(YouCarryingUmbrella)
8) NOT(UmbrellaGone) OR RoommateCarryingUmbrella (6,7)
9) RoommateCarryingUmbrella (5,8)
10) NOT(RoommateWetBecauseOfRain) (4,9)
11) NOT(RoommateWet) OR RoommateWetBecauseOfSprinklers (2,10)
12) RoommateWetBecauseOfSprinklers (1,11)
13) SprinklersOn (3,12)

Limitations of unit resolution

- P OR Q
- NOT(P) OR Q
- Can we conclude Q?

(General) resolution

if we have

- l_1 OR l_2 OR … OR l_k
  and
- m_1 OR m_2 OR … OR m_n
  where for some i,j, l_i = NOT(m_j)

we can conclude

- l_1 OR l_2 OR … l_{i-1} OR l_{i+1} OR … OR l_k OR m_1 OR m_2 OR … OR m_{j-1} OR m_{j+1} OR … OR m_n
- Same literal may appear multiple times; remove those

Applying resolution to story problem (more clumsily)

1) RoommateWet
2) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
3) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
4) NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
5) UmbrellaGone
6) NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
7) NOT(YouCarryingUmbrella)
8) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR SprinklersOn (2,3)
9) NOT(RoommateCarryingUmbrella) OR NOT(RoommateWet) OR SprinklersOn (4,8)
10) NOT(UmbrellaGone) OR YouCarryingUmbrella OR NOT(RoommateWet) OR SprinklersOn (6,9)
11) YouCarryingUmbrella OR NOT(RoommateWet) OR SprinklersOn (5,10)
12) NOT(RoommateWet) OR SprinklersOn (7,11)
13) SprinklersOn (1,12)

Systematic inference?

- General strategy: if we want to see if sentence a is entailed, add NOT(a) to the knowledge base and see if it becomes inconsistent (we can derive a contradiction)
- CNF formula for modified knowledge base is satisfiable if and only if sentence a is not entailed
  - Satisfiable = there exists a model that makes the modified knowledge base true = modified knowledge base is consistent

Resolution algorithm

- Given formula in conjunctive normal form, repeat:
- Find two clauses with complementary literals,
- Apply resolution,
- Add resulting clause (if not already there)
- If the empty clause results, formula is not satisfiable
  - Must have been obtained from P and NOT(P)

Example

Our knowledge base:

- 1) RoommateWetBecauseOfSprinklers
- 2) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn

Can we infer SprinklersOn?

- We add:
  - 3) NOT(SprinklersOn)
- From 2) and 3), get
  - 4) NOT(RoommateWetBecauseOfSprinklers)
- From 4) and 1), get empty clause

Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$
$\neg W_{1,1}$
$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$
$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$
$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$
$\neg W_{1,1} \vee \neg W_{1,2}$
$\neg W_{1,1} \vee \neg W_{1,3}$
…

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Limitations of propositional logic

- Some English statements are hard to model in propositional logic:

  "If your roommate is wet because of rain, your roommate must not be carrying **any** umbrella"
- Pathetic attempt at modeling this:

  RoommateWetBecauseOfRain =>
  (NOT(RoommateCarryingUmbrella0) AND
  NOT(RoommateCarryingUmbrella1) AND
  NOT(RoommateCarryingUmbrella2) AND …)

Limitations of propositional logic

- No notion of objects
- No notion of relations among objects
- RoommateCarryingUmbrella0 is instructive **to us**, suggesting
  - there is an object we call Roommate,
  - there is an object we call Umbrella0,
  - there is a relationship Carrying between these two objects
- Formally, none of this meaning is there
  - Might as well have replaced RoommateCarryingUmbrella0 by P

Elements of first-order logic

- Objects: can give these names such as Umbrella0, Person0, John, Earth, …
- Relations: Carrying(., .), IsAnUmbrella(.)
  - Carrying(Person0, Umbrella0), IsUmbrella(Umbrella0)
  - Relations with one object = unary relations = properties
- Functions: Roommate(.)
  - Roommate(Person0)
- Equality: Roommate(Person0) = Person1

Things to note about functions

- It could be that we have a separate name for Roommate(Person0)
- E.g., Roommate(Person0) = Person1
- … but we do not **need** to have such a name

- A function can be applied to any object
- E.g., Roommate(Umbrella0)

Reasoning about many objects at once

- **Variables**: x, y, z, … can refer to multiple objects
- New operators “for all” and “there exists”
  - **Universal quantifier** and **existential quantifier**
- for all x: CompletelyWhite(x) => NOT(PartiallyBlack(x))
  - Completely white objects are never partially black
- there exists x: PartiallyWhite(x) AND PartiallyBlack(x)
  - There exists some object in the world that is partially white and partially black

Practice converting English to first-order logic

- “John has Jane’s umbrella”
- Has(John, Umbrella(Jane))
- “John has an umbrella”
- there exists y: (Has(John, y) AND IsUmbrella(y))
- “Anything that has an umbrella is not wet”
- for all x: ((there exists y: (Has(x, y) AND IsUmbrella(y))) => NOT(IsWet(x)))
- “Any person who has an umbrella is not wet”
- for all x: (IsPerson(x) => ((there exists y: (Has(x, y) AND IsUmbrella(y))) => NOT(IsWet(x))))

More practice converting English to first-order logic

- "John has at least two umbrellas"
- there exists x: (there exists y: (Has(John, x) AND IsUmbrella(x) AND Has(John, y) AND IsUmbrella(y) AND NOT(x=y))
- "John has at most two umbrellas"
- for all x, y, z: ((Has(John, x) AND IsUmbrella(x) AND Has(John, y) AND IsUmbrella(y) AND Has(John, z) AND IsUmbrella(z)) => (x=y OR x=z OR y=z))

Even more practice converting English to first-order logic…

- “TUDa’s basketball team defeats any other basketball team”
- for all x: ((IsBasketballTeam(x) AND NOT(x=BasketballTeamOf(TUDa))) => Defeats(BasketballTeamOf(TUDa), x))
- “Every team defeats some other team”
- for all x: (IsTeam(x) => (there exists y: (IsTeam(y) AND NOT(x=y) AND Defeats(x,y))))

More realistically…

- "Any basketball team that defeats TUDa's basketball team in one year will be defeated by TUDa's basketball team in a future year"
- for all x,y: (IsBasketballTeam(x) AND IsYear(y) AND DefeatsIn(x, BasketballTeamOf(TUDa), y)) => there exists z: (IsYear(z) AND IsLaterThan(z,y) AND DefeatsIn(BasketballTeamOf(TUDa), x, z))

Relationship between universal and existential

- for all x: a
- is equivalent to
- NOT(there exists x: NOT(a))

Something we cannot do in first-order logic

- We are **not** allowed to reason in general about relations and functions
- The following would correspond to higher-order logic (which is more powerful):
- “If John is Jack’s roommate, then any property of John is also a property of Jack’s roommate”
- (John=Roommate(Jack)) => for all p: (p(John) => p(Roommate(Jack)))
- “If a property is inherited by children, then for any thing, if that property is true of it, it must also be true for any child of it”
- for all p: (IsInheritedByChildren(p) => (for all x, y: ((IsChildOf(x,y) AND p(y)) => p(x))))

Axioms and theorems

- Axioms: basic facts about the domain, our “initial” knowledge base
- Theorems: statements that are logically derived from axioms

SUBST

- SUBST replaces one or more variables with something else
- For example:
  - SUBST({x/John}, IsHealthy(x) => NOT(HasACold(x))) gives us
  - IsHealthy(John) => NOT(HasACold(John))

Instantiating quantifiers

- From
- for all x: a
- we can obtain
- SUBST({x/g}, a)

- From
- there exists x: a
- we can obtain
- SUBST({x/k}, a)
- where k is a constant that does not appear elsewhere in the knowledge base (Skolem constant)
- Don't need original sentence anymore

Instantiating existentials after universals

- for all x: there exists y: IsParentOf(y,x)
- WRONG: for all x: IsParentOf(k, x)
- RIGHT: for all x: IsParentOf(k(x), x)
- Introduces a new function (Skolem function)
- … again, assuming k has not been used previously

Generalized modus ponens

- for all x: Loves(John, x)
  - John loves every thing
- for all y: (Loves(y, Jane) => FeelsAppreciatedBy(Jane, y))
  - Jane feels appreciated by every thing that loves her
- Can infer from this:
- FeelsAppreciatedBy(Jane, John)

- Here, we used the substitution {x/Jane, y/John}
  - Note we used different variables for the different sentences
- General UNIFY algorithms for finding a good substitution

Keeping things as general as possible in unification

- Consider EdibleByWith
  - e.g., EdibleByWith(Soup, John, Spoon) – John can eat soup with a spoon
- for all x: for all y: EdibleByWith(Bread, x, y)
  - Anything can eat bread with anything
- for all u: for all v: (EdibleByWith(u, v, Spoon) => CanBeServedInBowlTo(u,v))
  - Anything that is edible with a spoon by something can be served in a bowl to that something
- Substitution: {x/z, y/Spoon, u/Bread, v/z}
- Gives: for all z: CanBeServedInBowlTo(Bread, z)
- Alternative substitution {x/John, y/Spoon, u/Bread, v/John} would only have given CanBeServedInBowlTo(Bread, John), which is not as general

Resolution for first-order logic

- for all x: (NOT(Knows(John, x)) OR IsMean(x) OR Loves(John, x))
  - John loves everything he knows, with the possible exception of mean things
- for all y: (Loves(Jane, y) OR Knows(y, Jane))
  - Jane loves everything that does not know her
- What can we unify? What can we conclude?
- Use the substitution: {x/Jane, y/John}
- Get: IsMean(Jane) OR Loves(John, Jane) OR Loves(Jane, John)
- Complete (i.e., if not satisfiable, will find a proof of this), **if** we can remove literals that are duplicates after unification
  - Also need to put everything in canonical form first

Notes on inference in first-order logic

- Deciding whether a sentence is entailed is semidecidable: there are algorithms that will eventually produce a proof of any entailed sentence
- It is **not** decidable: we cannot always conclude that a sentence is not entailed

(Extremely informal statement of) Gödel's Incompleteness Theorem

- First-order logic is not rich enough to model basic arithmetic
- For any consistent system of axioms that is rich enough to capture basic arithmetic (in particular, mathematical induction), there exist true sentences that cannot be proved from those axioms

A more challenging exercise

- Suppose:
  - There are exactly 3 objects in the world,
  - If x is the spouse of y, then y is the spouse of x (spouse is a function, i.e., everything has a spouse)
- Prove:
  - Something is its own spouse

More challenging exercise

- there exist x, y, z: (NOT(x=y) AND NOT(x=z) AND NOT (y=z))
- for all w, x, y, z: (w=x OR w=y OR w=z OR x=y OR x=z OR y=z)
- for all x, y: ((Spouse(x)=y) => (Spouse(y)=x))
- for all x, y: ((Spouse(x)=y) => NOT(x=y)) *(for the sake of contradiction)*
- *Try to do this on the board…*

Umbrellas in first-order logic

- You know the following things:
  - You have exactly one other person living in your house, who is wet
  - If a person is wet, it is because of the rain, the sprinklers, or both
  - If a person is wet because of the sprinklers, the sprinklers must be on
  - If a person is wet because of rain, that person must not be carrying any umbrella
  - There is an umbrella that “lives in” your house, which is not in its house
  - An umbrella that is not in its house must be carried by some person who lives in that house
  - You are not carrying any umbrella
- Can you conclude that the sprinklers are on?

Applications

- Some serious novel mathematical results proved
- Verification of hardware and software
  - Prove outputs satisfy required properties for all inputs
- Synthesis of hardware and software
  - Try to prove that there exists a program satisfying such and such properties, in a constructive way