Artificial Intelligence

CS 165A

May 23, 2022

Instructor: Prof. Yu-Xiang Wang

Today

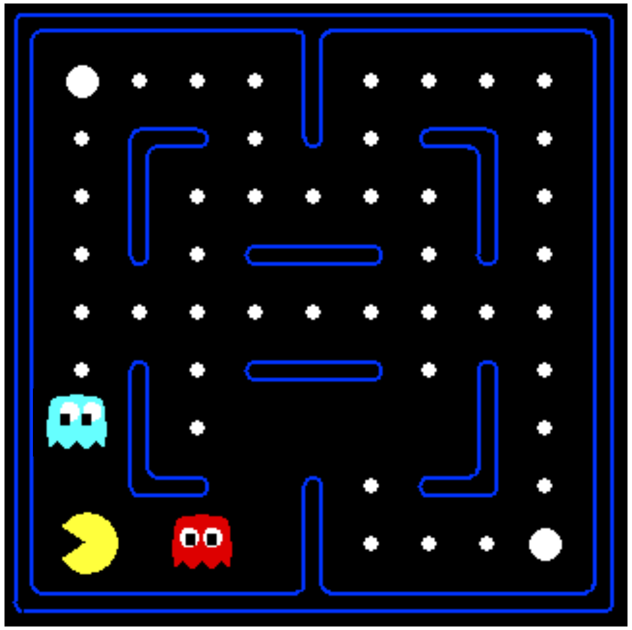
- → Logical Agents
- → Propositional Logic

Announcement

- ESCI survey is online (due June 3)
  - Please submit your feedback there
  - We have a small class and I am aiming for 100% response rate
- Please continue to work on Project 3
  - Everything is covered.
  - If you are stuck, consider getting help early!
- A few of you came to OH to discuss your midterm
  - It is a great way of learning.

Recap: Why not use an evaluation function? A Feature-Based Representations

- Solution: describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
    - Is it the exact state on this slide?
  - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Recap: Linear Value Functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:
  - $V_{\mathbf{w}}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$
  - $Q_{\mathbf{w}}(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$
- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but actually be very different in value!

Recap: Updating a linear value function

- Original Q learning rule tries to reduce prediction error at s, a:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

- Instead, we update the weights to try to reduce the error at s, a:

$$w_i \leftarrow w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, \partial Q_{\mathbf{w}}(s,a)/\partial w_i$$
$$= w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, f_i(s,a)$$

Recap: Policy gradient method

- Parametrize the policy instead

- Apply SGD (from Lecture ML3):
  - What is the objective function to optimize (minimize or maximize?)

  - How to construct the Stochastic Gradient (i.e., an unbiased estimator of the gradient)?

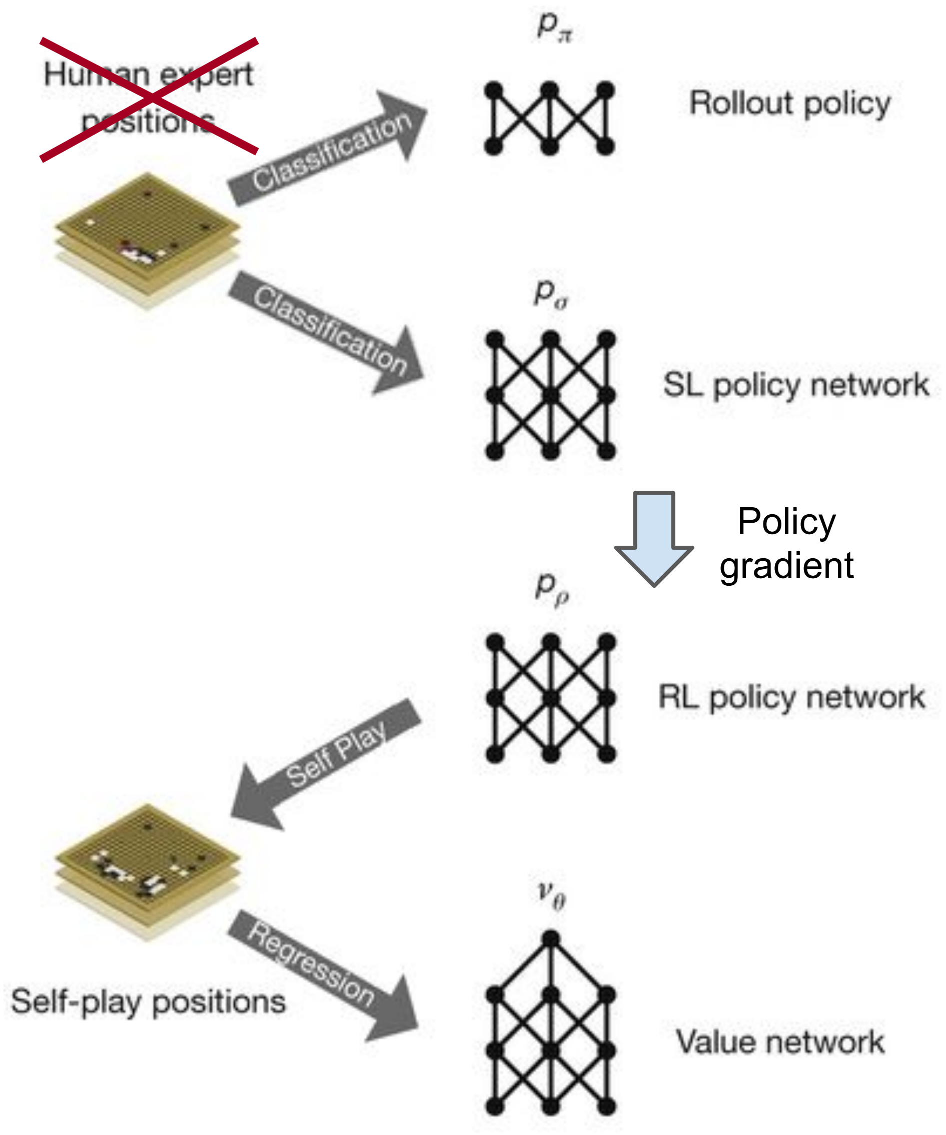
Elements of State-of-the-Art Reinforcement Learning

- Use a deep neural network to parameterize Q-function
- Use a deep neural network to parameterize the policy \pi
- Run a combination of Q-learning and Policy Gradient.
  - Actor-Critics, A3C, etc…
- Heuristic-based exploration: curiosity, reward shaping, etc..
- Experience replay to generate more data from existing data.
- Multi-agent RL: modeling your opponents

Alpha-Go and Alpha-Zero

- Parameterize the policy networks with CNN
- ~~Supervised learning initialization~~
- RL using Policy gradient
- Fit Value Network (This is a heuristic function!)
- Monte-Carlo Tree Search

https://www.youtube.com/watch?v=4D5yGiYe8p4



D. Silver. Mastering the game of Go with Deep Neural Networks and Tree Search. Nature, vol. 529 issue 7587

D. Silver, et al. "Mastering the game of go without human knowledge." *Nature* 550.7676 (2017): 354-359.

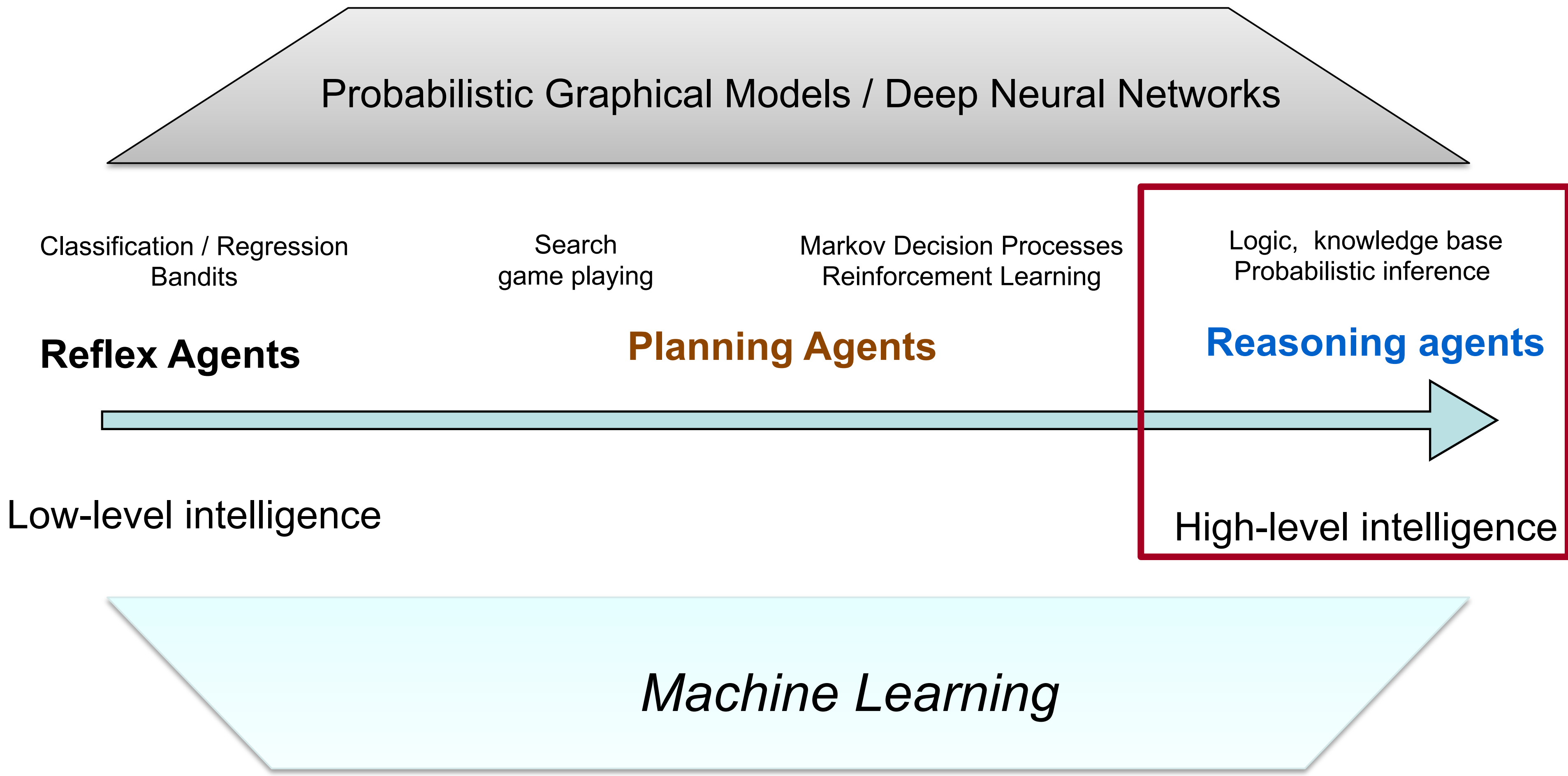
Recap: Summary of RL algorithms

- Model-based:
  - Policy iteration / Value iteration
  - Need to estimate the dynamics (MDP parameters)
- Model-free: (no need to “explicitly” estimate dynamics)
  - TD learning: SARSA, Q-learning
  - Function approximation (Share information across states)
- Absolutely model-free (do not even need an MDP model)
  - Policy gradient
- Modern RL methods combine all these (also with search)

Modeling-Inference-Learning paradigm

- **Modeling**: MDP, POMDP, Bandits, Contextual Bandits
- **Inference**: Dynamic programming / Simulating Bellman equations
- **Learning**: online learning, exploration, regret. Estimating MDP parameters vs learning value function directly.
- Q-learning and SARSA combine learning and inference!

High-level intelligence and logical inference



The final lecture series on “logic”

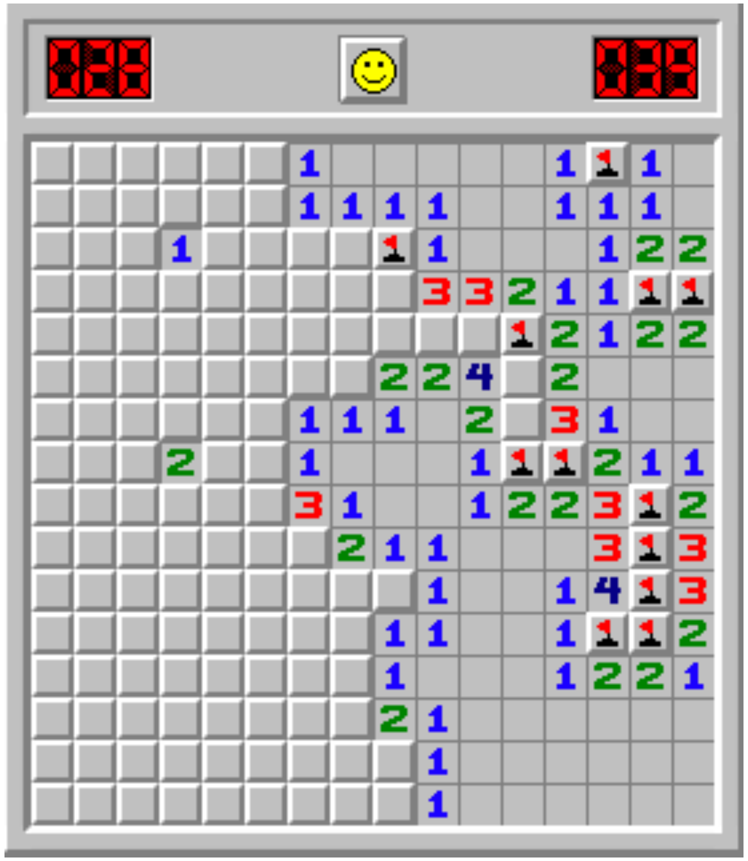
- So far:
  - Reflex agents (classifiers)
  - Problem solving / planning / game solving agents (Search)
  - Planning meets utility-maximizing agents (MDPs)

- They can:
  - Quantify uncertainty
  - Make rational decisions
  - Learn from experience

- What’s missing?
  - Knowledge, reasoning, logical deduction
  - (Arguably PGM does a bit of this, but our focus was to use PGM for modeling the world...)

Why do we care?

- Minesweeper



- Imagine how you would solve this?
- Imagine how an RL agent would solve this?

Knowledge Base:

- Encode the rules.
- Encode the observations so far.

What does a knowledge base do?

- **TELL** operation: add evidence.
- **ASK** operation: check if a tile has a mine under it, or not, or undetermined.

Knowledge and reasoning

- We want powerful methods for
  - Representing *Knowledge* – general methods for representing facts about the world and how to act in world
  - Carrying out *Reasoning* – general methods for deducing additional information and a course of action to achieve goals
  - Focus on *knowledge and reasoning* rather than *states and search*
    - Note that *search* is still critical

- This brings us to the idea of ***logic,*** but….
  - How to define logic formally?
  - How to represent / manipulate knowledge / inference at scale?
  - How to systematically use knowledge / inference by an agent?
  - What are the strengths and limitations of logical agents?

Example

- A certain country is inhabited by people who always tell the truth or always tell lies and who will respond only to yes/no questions.

  A tourist comes to a fork in the road where one branch leads to a restaurant and one does not.

  No sign indicating which branch to take, but there is an inhabitant Mr. X standing on the road.

  With a single yes/no question, can the hungry tourist ask to find the way to the restaurant?

Example (cont.)

- Answer: Is exactly one of the following true:
  1. you always tell the truth
  2. the restaurant is to the left

- Truth Table:

X is truth teller;	restaurant is to left;	response
true;	true;	no
true;	false;	yes
false;	true;	no
false;	false;	yes

Another Example (1 min discussion)

- Bob looks at Alice. Alice looks at George.
  Bob is married. George is unmarried.
  Does a married person ever look at an unmarried one;
  yes, no, cannot be determined?

Another Example (cont.)

- Amarried or ~Amarried
  BlooksA and AlooksG

  BlooksA ^ AlooksG =
  BlooksA ^ AlooksG ^ Amarried or BlooksA ^ AlooksG ^ ~Amarried

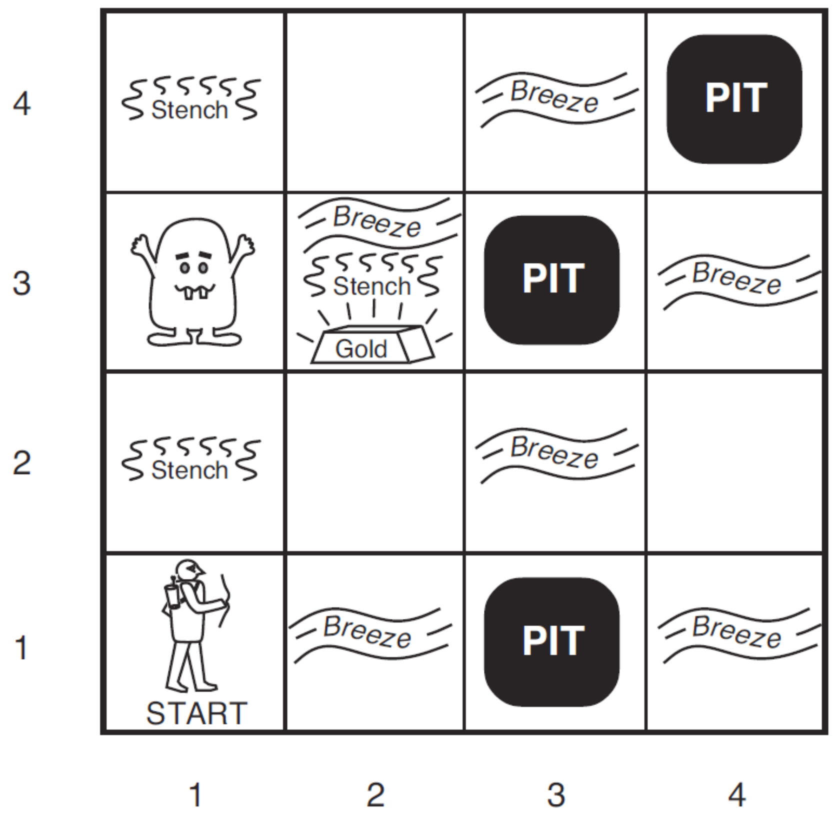
- Case 1: Amarried = true, then BlooksA ^ AlooksG ^ Amarried satisfies conclusion

  Case 2: Amarried = false, then BlooksA ^ AlooksG ^ ~Amarried satisfies conclusion

Wumpus World

- Logical Reasoning as a CSP

  - B_{ij} = breeze felt
  - S_{ij} = stench smelt
  - P_{ij} = pit here
  - W_{ij} = wumpus here
  - G = gold



http://thiagodnf.github.io/wumpus-world-simulator/

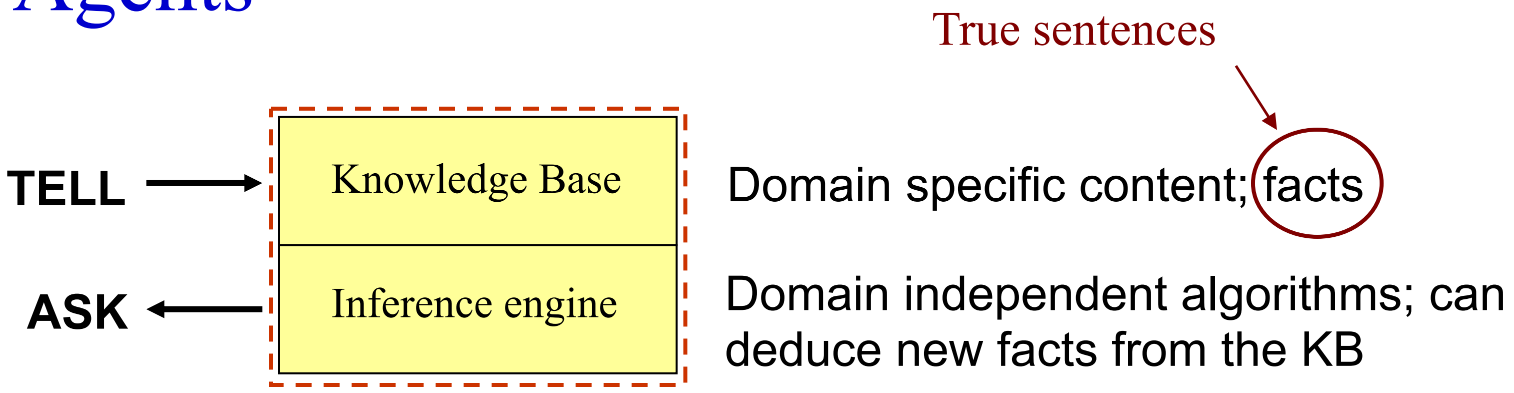
*The agent can only observe blocks that she has visited.

*Cannot observe the state directly. So cannot solve offline with search.

Knowledge-based agents

- A *knowledge-based agent* uses reasoning based on **prior** and **acquired** knowledge in order to achieve its goals
- Two important components:
  - Knowledge Base (KB)
    - Represents facts about the world (the agent's environment)
      - Fact = "sentence" in a particular knowledge representation language (KRL)
    - KB = set of sentences in the KRL
  - Inference Engine – determines what follows from the knowledge base (what the knowledge base *entails*)
    - Inference / deduction
      - Process for deriving new sentences from old ones
        - » *Sound* reasoning from facts to conclusions

KB Agents


True sentences
TELL
ASK
Knowledge Base
Inference engine
Domain specific content; facts
Domain independent algorithms; can deduce new facts from the KB

```
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
```

KB Agents need to TELL, ASK with a language and the KB needs to understand.

- How about using natural languages?
  - Example from Lecture 1.

They ate the pie with ice cream.

They ate the pie with rhubarb.

They ate the pie with paper plates.

They ate the pie with cold milk.

They ate the pie with friends.

They ate the pie with dinner.

They ate the pie with enthusiasm.

They ate the pie with spoons.

They ate the pie with napkins.

Ambiguities!!!

from Dr. Douglas Lenatand Dr. Michael Witbrock

Fundamental Concepts of Logical Language Representation and Concepts

- **Syntax**
  - Grammar / rules to follow for form a well-defined sentence
  - $x + y = 4$ is a valid sentence in "arithmetics", x4y+= is not.
- **Semantics**
  - The meaning of sentences. Truth of each sentence w.r.t. each possible world.
  - Possible World 1: x=3, y=1. Possible World 2: x=1, y=1.
- **Model** (Possible world, a.k.a. "interpretations" in some text)
  - Each model is an assignment of values to variables.
  - Each model fixes the truth value of all sentences.
  - If sentence α is true in Model m, we say: Model m satisfies sentence α, or m is a model of α, or $m \in M(\alpha)$,

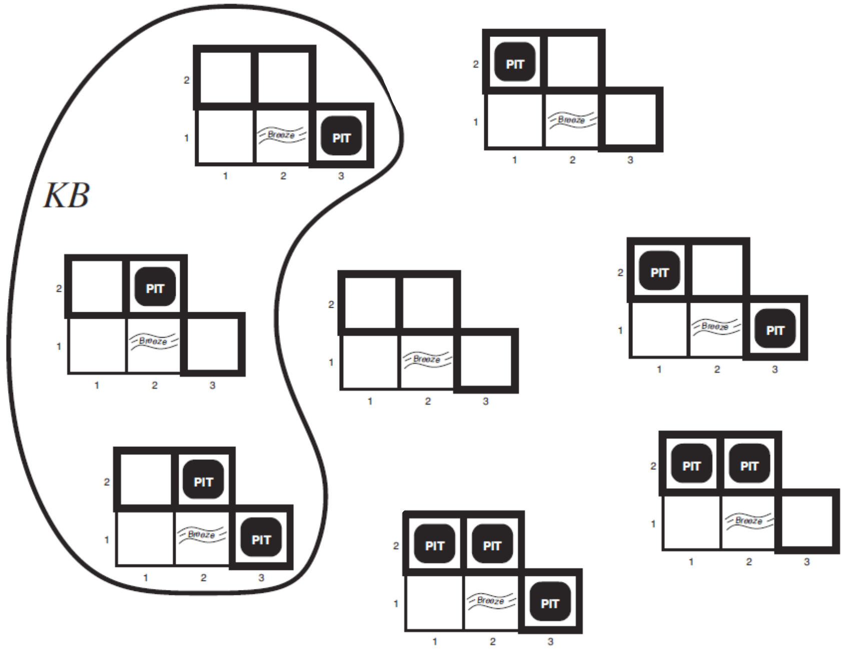
Fundamental Concepts of Logical Language Representation and Concepts

- **Entailment**
  - Sentence β logically follows from Sentence α
  - Denoted by $\alpha \models \beta$
  - α entails β *if an only if* $M(\alpha) \subseteq M(\beta)$
  - If all models of α are also models of β

- **Logical Inference**
  - The procedure of checking whether a sentence is entailed by a given a knowledge base
  - Simplest algorithm for logical inference: **Model checking**
  - Enumerate all models in $\mathrm{M}(\alpha)$, check whether they are in $\mathrm{M}(\beta)$.
  - We will come back to logical inference!

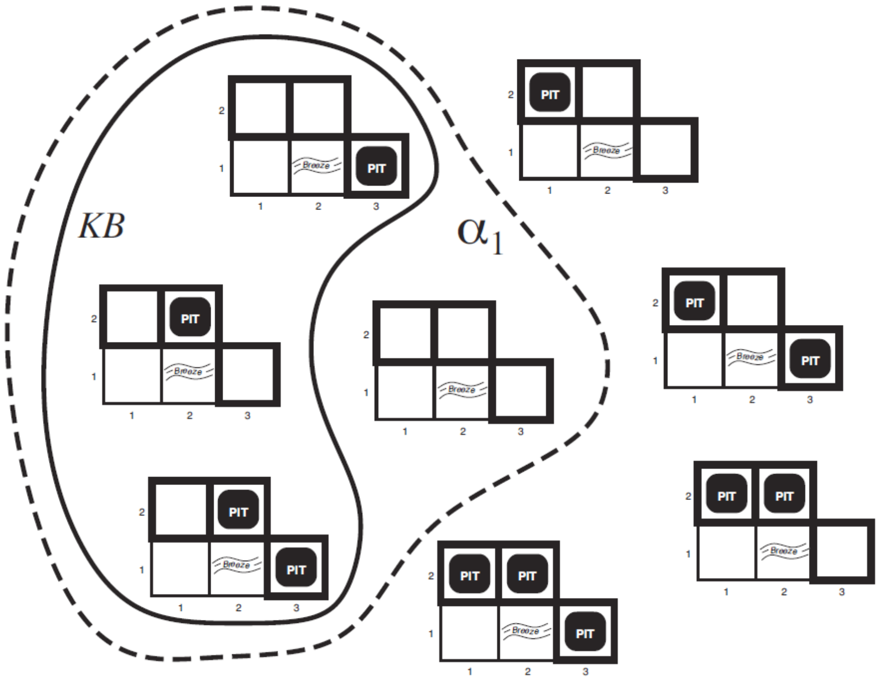
Example: Wumpus World

- Possible Models
  - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
  - Knowledge base
    - Nothing in [1,1]
    - Breeze in [2,1]



Example: Wumpus World

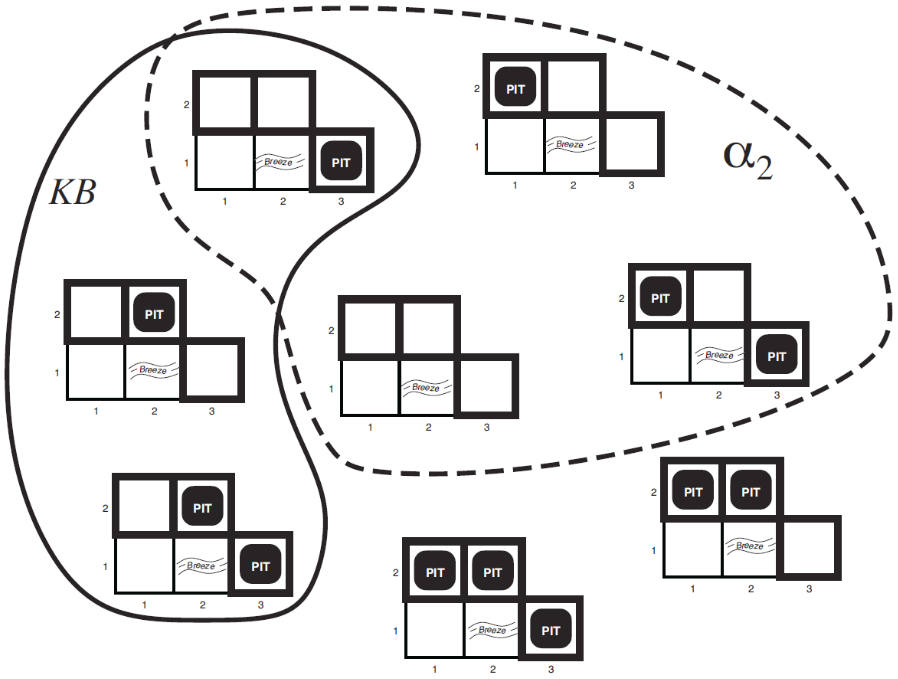
- Possible Models
  - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
  - Knowledge base
    - Nothing in [1,1]
    - Breeze in [2,1]
  - Query α_1:
    - No pit in [1,2]



*Question: Does KB entails α_1?

Example: Wumpus World

- Possible Models
    - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
    - Knowledge base
        - Nothing in [1,1]
        - Breeze in [2,1]
    - Query α_2:
        - No pit in [2,2]



*Question: Does KB entails α_2?

Inference and Entailment

- Given a set of (true) sentences, ***logical inference*** generates new sentences
  - Sentence α follows from sentences $\{ \beta_i \}$
  - Sentences $\{ \beta_i \}$ ***entail*** sentence α
  - The classic example is *modus ponens*: **P ⇒ Q** and **P** entail what?

- A knowledge base (KB) *entails* sentences α

$$\boxed{KB \models \alpha}$$

- An **inference procedure *i*** can derive α from KB

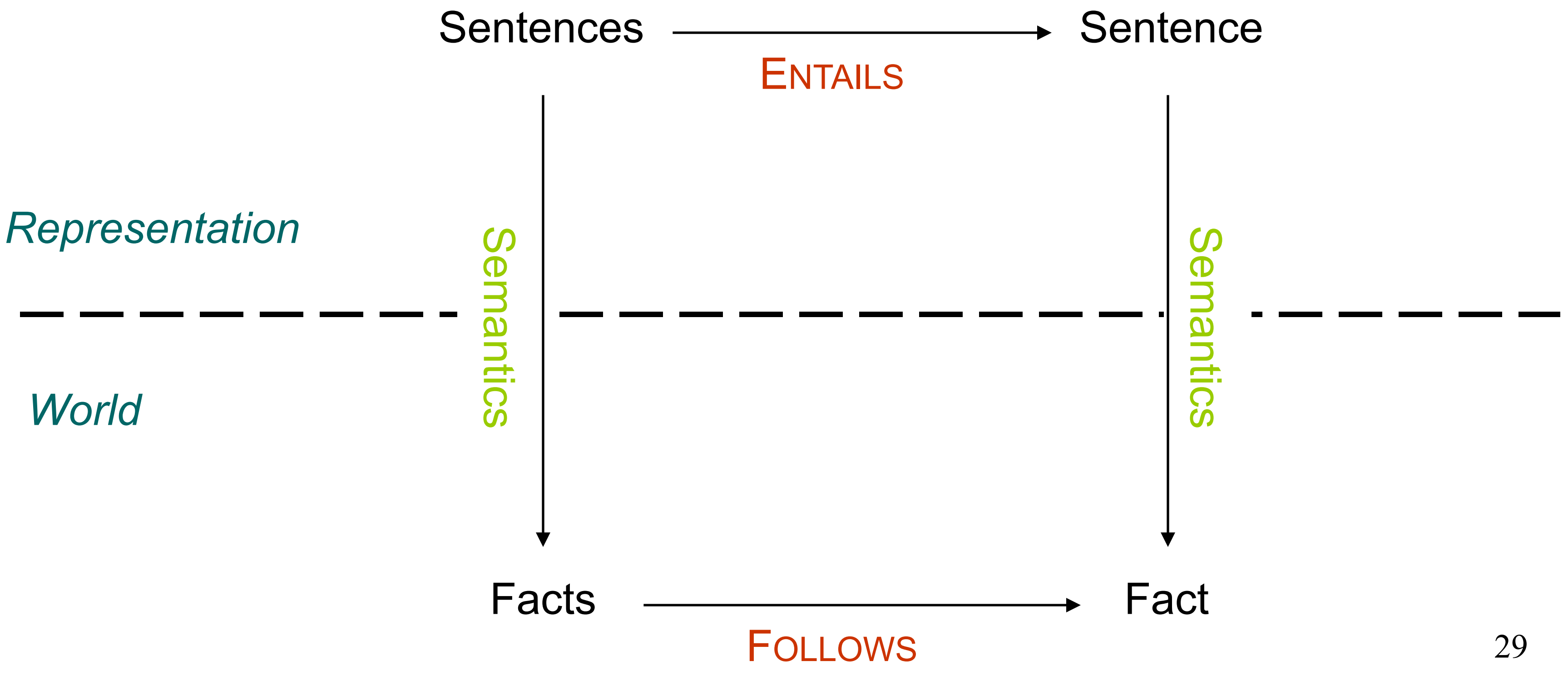
$$\boxed{KB \vdash_i \alpha}$$

Inference and Entailment (cont.)

Inference *(n.):*

a. The act or process of deriving logical conclusions from premises known or assumed to be true.

b. The act of reasoning from factual knowledge or evidence.



Inference engine

- An inference engine is a program that applies inference rules to knowledge
  - Goal: To infer new (and useful) knowledge

- Separation of
  - Knowledge
  - Rules
  - Control

Inference engine

Which rules should we apply when?

Inference procedures

- An inference procedure
  - Generates new sentences α that purport to be entailed by the knowledge base

    …or...
  - Reports whether or not a sentence α is entailed by the knowledge base
- Not every inference procedure can derive all sentences that are entailed by the KB
- A *sound* or *truth-preserving* inference procedure generates only entailed sentences
- Inference derives valid conclusions *independent of the semantics* (i.e., independent of the models)

Inference procedures (cont.)

- Soundness of an inference procedure
  - *i* is **sound** if whenever KB $\vdash_i \alpha$, it is also true that KB $\models \alpha$
  - i.e., the procedure only generates entailed sentences
- Completeness of an inference procedure
  - *i* is **complete** if whenever KB $\models \alpha$, it is also true that KB $\vdash_i \alpha$
  - i.e., the procedure can find a proof for any sentence that is entailed
- The derivation of a sentence by a sound inference procedure is called a ***proof***
  - Hence, the ***proof theory*** of a logical language specifies the reasoning steps that are sound

So far, we have defined the jargon and notation of a generic logic language

- Syntax
- Semantics
- Models
- Entailment
- Inference
- Soundness and completeness

- Make sure you know / understand these definitions!

Logics (Specify Syntax, Semantics, Inference procedures and so on…)

- We will soon define a logic which is expressive enough to say most things of interest, and for which there exists a sound and complete inference procedure
  - I.e., the procedure will be able to derive anything that is derivable from the KB
  - This is ***first-order logic***
  - But first, let's review ***propositional logic,*** which you've already learned from CS40

Propositional (Boolean) Logic

- Symbols represent **propositions** (statements of fact, sentences)
  - *P* means "San Francisco is the capital of California"
  - *Q* means "It is raining in Seattle"

- Sentences are generated by combining proposition symbols with Boolean (logical) connectives

Propositional Logic

- Syntax
  - *True, false,* propositional symbols
  - ( ) , $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies), $\Leftrightarrow$ (equivalent)

- Examples of sentences in propositional logic

P_1, P_2, etc. (propositions)	$S_1 \Rightarrow S_2$
(S_1)	$S_1 \Leftrightarrow S_2$
$\neg S_1$	$true$
$S_1 \wedge S_2$	$P_1 \wedge true \wedge \neg(P_2 \Rightarrow false)$
$S_1 \vee S_2$	$P \wedge Q \Leftrightarrow Q \wedge P$

Entailment and equivalence

- What is the meaning of α entails β, or $\alpha \models \beta$
- $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$
- Examples of logical equivalences
  - Commutativity of $\wedge$, $\vee$
  - Associativity of $\wedge$, $\vee$
  - Distributive laws
    - A and (B or C)=(A and B) or (A and C)
  - De Morgan’s laws
    - NOT (P OR Q) = (NOT P) AND (NOT Q)
    - NOT (P AND Q) = (NOT P) OR (NOT Q)
  - $P \Rightarrow Q \equiv \neg P \vee Q$

Precedence of operators (logical connectives)

- Levels of precedence, evaluating left to right
  1. ¬ (NOT)
  2. ∧ (AND, conjunction)
  3. ∨ (OR, disjunction)
  4. ⇒ (implies, conditional)
  5. ⇔ (equivalence, biconditional)
- $P \land \neg Q \Rightarrow R$
  - $(P \land (\neg Q)) \Rightarrow R$
- $P \lor Q \land R$
  - $P \lor (Q \land R)$
- $P \Leftrightarrow Q \quad \land \quad R \Rightarrow S$
  - $P \Leftrightarrow ((Q \land R) \Rightarrow S)$

Satisfiability and Validity

- Is this true: $(P \wedge Q)$?
  - It depends on the values of P and Q
  - This is a **satisfiable** sentence – there are some **models** for which it is true and others for which it is false
- Is this true: $(P \wedge \neg P)$?
  - No, it is never true
  - This is an **unsatisfiable** sentence (self-contradictory) – there is no **models** for which it is true
- Is this true: $(((P \vee Q) \wedge \neg Q) \Rightarrow P)$?
  - Yes, independent of the values of P and Q
  - This is a **valid** sentence – it is true under all possible **models** (a.k.a. a **tautology**)

Things to know!

- What is a <u>sound</u> inference procedure?
  - The procedure only generates entailed sentences
- What is a <u>complete</u> inference procedure?
  - The procedure can find a proof for any sentence that is entailed
- What is a <u>satisfiable</u> sentence?
  - There are some models for which it is true
- What is an <u>unsatisfiable</u> sentence?
  - There is no model for which it is true
- What is a <u>valid</u> sentence?
  - It is true under all possible models

Propositional (Boolean) Logic (cont.)

- Semantics
  - Defined by clearly interpreted symbols and straightforward application of truth tables
  - Rules for evaluating truth: Boolean algebra
  - Simple method: truth tables

Propositions / Variables

Sentences

Models

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

2^N rows (models) for N propositions

Knowledge are constraints that eliminate rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

- KB: Nothing

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

Knowledge are constraints that eliminates rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

- KB: Nothing
- KB: $[(P \wedge \neg Q) \vee (Q \wedge \neg P)] \Rightarrow R$

Knowledge are constraints that eliminates rows

- Adding a sentence to our knowledge base constrains the
- number of possible models:

Possible Models

P	Q	R
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

- KB: Nothing
- KB: $[(P \land \neg Q) \lor (Q \land \neg P)] \Rightarrow R$
- KB: R, $[(P \land \neg Q) \lor (Q \land \neg P)] \Rightarrow R$

Sherlock Entailment

- "When you have eliminated the impossible, whatever remains, however improbable, must be the truth" – *Sherlock Holmes via Sir Arthur Conan Doyle*

- Knowledge base and inference allow us to remove impossible models, helping us to see what is true in all of the remaining models

Logical Inference in Propositional Logic

- A simple algorithm for checking: KB entails α
  - Enumerate M(KB)
  - Check that it is contained in M(α)

- This inference algorithm is **sound** and **complete**.

- Are there other ways to do logical inference?

- Are they sound / complete?

Using propositional logic: rules of inference

- Inference rules capture patterns of sound inference
  - Once established, don't need to show the truth table every time
  - E.g., we can define an inference rule: $((P \lor H) \land \neg H) \vdash P$ for variables P and H

- Alternate notation for inference rule $\alpha \vdash \beta$:

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

"If we know α, then we can conclude β"

(where α and β are propositional logic sentences)

Inference

- We're particularly interested in

$$\frac{\textbf{KB}}{\beta} \quad \text{or} \quad \frac{\alpha_1, \alpha_2, \ldots}{\beta}$$

- Inference steps

$$\frac{\textbf{KB}}{\beta_1} \rightarrow \frac{\textbf{KB}, \beta_1}{\beta_2} \rightarrow \frac{\textbf{KB}, \beta_1, \beta_2}{\beta_3} \rightarrow \ldots$$

So we need a mechanism to do this!

Inference rules that can be applied to sentences in our KB

Important Inference Rules for Propositional Logic

- **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

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

- **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

- **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \ \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

- **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

- **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

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

- **Resolution**: (This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \qquad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Example (using inference rules)

KB

$Q \rightarrow \neg S$

$P \vee \neg W$

R

P

$P \rightarrow Q$

What can we infer ($\vdash$) if we add this sentence with no inference rules?

$P \rightarrow Q$

Nothing

What can we infer ($\vdash$) if we then add this inference procedure:

$$(\alpha \rightarrow \beta) \wedge \alpha \ \vdash \ \beta$$

$$\frac{(\alpha \rightarrow \beta),\ \alpha}{\beta}$$

Q and $\neg S$

Resolution Rule: one rule for all inferences

$$\frac{p \vee q, \quad \neg q \vee r}{p \vee r}$$

Propositional calculus resolution

Remember: $\boldsymbol{p \Rightarrow q} \Leftrightarrow \boldsymbol{\neg p \vee q}$, so let's rewrite it as:

$$\frac{\neg p \Rightarrow q, \quad q \Rightarrow r}{\neg p \Rightarrow r} \qquad \text{or} \qquad \frac{a \Rightarrow b, \quad b \Rightarrow c}{a \Rightarrow c}$$

Resolution is really the "chaining" of implications.

Soundness:

Show that $(\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \Rightarrow (\alpha \vee \gamma)$

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha\vee\beta \wedge \neg\beta\vee\gamma$	$\alpha \vee \gamma$
0	0	0	0	1	0	0
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

This is always true for all propositions α, β, and γ, so we can make it an inference rule

Soundness:

Show that $(\neg\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \gamma) \Rightarrow (\neg\alpha \Rightarrow \gamma)$

α	β	γ	$\neg\alpha \Rightarrow \beta$	$\beta \Rightarrow \gamma$	$\neg\alpha \Rightarrow \beta \wedge \beta \Rightarrow \gamma$	$\neg\alpha \Rightarrow \gamma$
0	0	0	0	1	0	0
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

This is always true for all propositions α, β, and γ, so we can make it an inference rule

Conversion to Conjunctive Normal Form: CNF

- Resolution rule is stated for conjunctions of disjunctions
- Question:
  - Can every statement in PL be represented this way?
- Answer: Yes
  - Can show every sentence in propositional logic is equivalent to conjunction of disjunctions
    - Conjunctive normal form (CNF)
- Procedure for obtaining CNF
  - Replace $(P \Leftrightarrow Q)$ with $(P \Rightarrow Q)$ and $(Q \Rightarrow P)$
  - Eliminate implications: Replace $(P \Rightarrow Q)$ with $(\neg P \vee Q)$
  - Move $\neg$ inwards: $\neg\neg$, $\neg(P \vee Q)$, $\neg(P \wedge Q)$
  - Distribute $\wedge$ over $\vee$, e.g.: $(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$ [What about $(P \vee Q) \wedge R$?]
  - Flatten nesting: $(P \wedge Q) \wedge R$ becomes $P \wedge Q \wedge R$

Complexity of reasoning

- Validity
  - NP-complete
- Satisfiability
  - NP-complete
- α is valid iff $\neg \alpha$ is unsatisfiable
- Efficient decidability test for validity iff efficient decidability test for satisfiability.
- To check if $KB \models \alpha$, test if $(KB \wedge \neg \alpha)$ is unsatisfiable.
- For a restricted set of formulas (Horn clauses), this check can be made in linear time.
  - Forward chaining
  - Backward chaining

Propositional logic is quite limited

- Propositional logic has simple syntax and semantics, and limited expressiveness
  - Though it is handy to illustrate the process of inference
- However, it only has one representational device, the proposition, and cannot generalize
  - Input: facts; Output: facts
  - Result: Many, many rules are necessary to represent any non-trivial world
  - It is impractical for even very small worlds
- The solution?
  - **First-order logic**, which can represent propositions, objects, and relations between objects
  - Worlds can be modeled with many fewer rules

Next lecture

- First order logic
- Read Chapter 7 and Chapter 8 of AIMA textbook.