Day 11: Semantic Networks

- Why do you need knowledge representation?
  - The world is more complicated than a game
  - Understanding language is complicated
  - Understanding complex systems is hard
- In the real world, there's:
  - Lots of information
  - Complex interactions
  - Partial state and ambiguity
  - Unanswerable questions
  - Changing facts
- How to evaluate knowledge representations:
  - How do we get the knowledge in?
  - Is it general or domain-specific?
  - How can we use and manipulate it?
  - Does it work well? How flexible is it?
- Representational Adequacy: is the representation adequate?
  - To represent 2^k things, you need k bits
    - But how many ideas are there? For somethings (ex. NLP) there are ∞
  - OTOH, is it detailed enough to actually represent whatever you need?
    - This depends on the task at hand, sometime's it's OK to oversimplify if the difference isn't relevant.
- Knowledge Accessibility:
  - How can stored knowledge be indexed, manipulated, etc.?

- Inferential Ability
  - How can you make inferences?
  - How efficient will it be?
  - Is it easy to control the inference to limit combinatorial expansion?
- Common Knowledge Representation Systems:
  - Procedural Representations (ex. programs)
    - Adequate because of Turing equivalence
    - Not accessible because it's compiled
    - Can't be used for inferences
    - Translatable, but only by a highly-skilled programmer
  - Feature-Value Systems
    - Each object is a vector, with each position in that vector representing a "feature" (attribute), and the number in each position representing the value of that attribute.
    - Good at organizing data, but not composable or flexible
    - Limited inferencing
  - Databases
    - Transaction-based table representation
    - Arbitrarily complex, but tables get large
    - Changes to representations can require complicated changes and downtime
    - Limited inferencing (basically, `JOIN`s)
    - Very scalable
  - First-Order Logic
  - Semantic Networks (aka Associative Network)
    - Uses a graph of labeled nodes (words) and labeled, directed arcs between nodes (relationships) to encode knowledge

- Usually: many types of nodes, few types of links
  - Tendency for combinatorial expansion of link types to fully describe a concept, which results in them being too specific for useful inferences
  - Often a desire to separate categories of things (ex. `dogs`) and specific instances of things (ex. `Fido`)
- Often used with a set of accessing procedures which perform reasoning
- Often not that expressive
- If you know the meaning of all nodes and links that a node is linked to, then you understand that node
- Advantages:
  - Easy to visualize
  - Formal definitions exist
  - Space-efficient
  - Related knowledge is clustered
- Inheritance: the main kind of reasoning done in semantic networks
  - `is-a` relationship is used to link a class to its superclass
  - Some types of links (ex. `has-part`) are inherited along `is-a` boundaries
  - Semantics of a semantic network can be informal or formal, and are often defined by the implementation
- Disadvantages:
  - Inheritance (especially multiple and exceptions) can cause problems
    - Ex ("Nixon Diamond"): `Republicans are pro-war. Quakers are pacifists. Richard Nixon was a Republican and a Quaker. Was he a pacifist?`

        - These sorts of problems are unsolved, because you need sophisticated logic for resolving conflicts
        - Fundamentally, `is-a` is far too vague. (Five things it can mean: member of a set, generalization, concept containment, description, instantiation)
    - Some semantic networks don't inherit infinitely far back. Most let closer ancestors override farther ancestors (like classes in programming).
  - Facts placed wrong can cause problems
  - No standard node/arc types
- How to handle non-binary/multilateral relationships?
  - Ex: `Mary gave Sally a book`
  - Turn the arc into its own node, ex: `give(giver: Mary, recipient: Sally, object: a book)`
- First implementation: Quillian:
  - English words defined in a dictionary-esque way, very circularly
- Example link types: is-a, has-part
- Graph Algorithms used for semantic networks:
  - Marker passing:
    - Each node has a unique marker
    - When a node is activated, it sends a copy of its marker to all of its neighbors (following outgoing links only)
    - Any node that receives a marker sends it on to all of its connections
    - If two different markers arrive at the same node, then you've found a connection between the two originating nodes
  - Spreading activation:

        - A node sends labeled activations (a numerical value) divided amongst its neighbors by some weighting mechanism
        - Each node reduces the activation by a bit
        - The amount of activation a node receives is a measure of the strength of its connection (or relatedness) with the originating node
- Representational Adequacy
    - Proved equivalent to first-order logic
    - Graph representation may limit expansion/explosion
    - Sometimes the networks is too complicated/messy/meaningless
- Types of inferences done in semantic networks:
    - Inheritence
    - Associational links
    - Relationships between words (connection strength, common concepts/ancestors, path between two nodes that intersects this node, etc.)