Contents

Notes

Contributions are, as always, welcome!

A lot of this document came from wikipedia and Artificial Intelligent: Foundations of Computation Agents

italics are from Dan's notes

> Quotes are from various smart people

Bold is possible answers to this nonsense

Symbol System Hypothesis (2016, 2014, 2011)

A physical symbol system (also called a formal system) takes physical patterns (symbols), combining them into structures (expressions) and manipulating them (using processes) to produce new expressions.

> "A physical symbol system has the necessary and sufficient means for general intelligent action."

- Implies human thinking is a kind of symbol manipulation
  - A symbol system is necessary for intelligence
  - Action
- *The environment can be represented by symbols and that action can be represented by symbol manipulation.*

Non-determinism (2016, 2015, 2014, 2011)

A nondeterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviors on different runs.

- *Symbol manipulation is action but non-determinism is the fact that intelligent action requires making choices.*
- *Paradox of intelligent AI is that intelligence requires choices but that computation is deterministic.*

Search (2016, 2015)

In computational complexity theory, nondeterministic algorithms are ones that, at every possible step, can allow for multiple continuations (imagine a man walking down a path in a forest and, every time he steps further, he must pick which fork in the road he wishes to take). These algorithms do not arrive at a solution for every possible computational path; however, they are guaranteed to arrive at a correct solution for some path (i.e., the man walking through the forest may only find his cabin if he picks some combination of "correct" paths). The choices can be interpreted as guesses in a search process.

Don't care vs don't know (2012, 2011)

Don't care non-determinism: the choice can be made arbitrarily. In terms of the logic programming computation model, any goal reduction will lead to a solution, and it does not matter which particular solution is found. **select**

If I say to a class of students "One of you must clean the board before the lesson" then I only want ONE of them to do it, and I don't CARE who.

Don't know non-determinism: the choice matters but the correct one is not known at the time the choice is made. **choose**

If I say to a class of students "Write all your names on this piece of paper" then I know that I want ALL the names, but I don't KNOW which ORDER they will be written in.

Cantor's theorem (2016, 2014)

In elementary set thoery, Cantor's theorem is a fundamental result that states that, for any set A, the set of all subsets of A (i.e. the power set) has a strictly greater cardinality than A itself.

For finite sets, Cantor's theorem can be seen to be true by simple enumeration of the number of subsets. Counting the empty subset, a set with n members has 2^n subsets, and the theorem holds since $2^n > n$ for $n = 0, 1, 2, \ldots$

As a particularly important instance, the power set of a countably infinite set, a set with the same cardinality as the natural numbers, is uncountably infinite and has the same cardinality as the real numbers (cardinality of the continuum)

There are too many infinite bit strings to be searched one at a time. There is no function from the natural numbers onto the set of infinite bit strings.

The problem SAT of Boolean satisfiability (2016, 2015)

The problem of determining if there exists an interpretation that satisfies a given Boolean formula, i.e. can the variables of a given Boolean formula be consistently replaced by the values `TRUE` or `FALSE` in such a way that the formula evaluates to `TRUE`. If this is the case, the formula is satisfiable.

Finite bit strings (2016)

With n boolean variables there are 2^n different possible bit strings of length n. (Hence the search space is exponential) In principle this can be searched but is very computationally expensive. As it's exponential a brute force solution takes worse than polynomial time but there may exist more efficient ways to solve the problem.

Binary Decision Diagram (2016)

A data structure that is used to represent a Boolean function. On a more abstract level, BDDs can be considered as a compressed representation of sets or relations.

A rooted, directed, acyclic graph. Terminal nodes 0 and 1, other nodes branching depending on boolean variable value.

- Ordered: Variables appear in the same order along all paths from the root to the leaves
- Reduced: *Low and high children of the node cannot be the same*
- Satisfiability: *When it is not equal to 0*

Church-Turing thesis (2014, 2012, 2011)

A hypothesis about the nature of computable functions. It states that a function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing Machine.

Turing machine (2015)

An abstract machine that manipulates symbols on a strip of tape according to a table of rules.

An agent acting intelligently can be represented as a Turing Machine. The tape of symbols is the environment.

Halting Problem (2015, 2012)

The problem in determining from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever. The halting problem is undecidable over Turing machines.

P vs NP Problem (2015)

Does every problem whose solution can be quickly verified by a computer also be quickly solved by a computer.

Example: Given a set of integers, does some nonempty subset of them sum up to 0? For instance, $S = \{-2, -3, 15, 14, 7, -10\}$. This is quickly verifiable ($\{-2, -3, -10, 15\}$ sum to 0), but there is no known algorithm to solve this in polynomial time.

If $P = NP$ then problems that can be verified in polynomial time can also be solved in polynomial time. However, if $P \neq NP$ then that would mean there are problems in NP (such as NP-complete problems) that are harder to compute than to verify.

An aside, complexity classes:

- P: Contains all decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time
- NP: Contains all decision problems that can be solved by a theoretical nondeterministic Turing machine using a polynomial amount of computation time

SAT (2015)

SAT is one of the first problems proven to be NP-complete. The belief that there is no algorithm to efficiently solve each SAT problem has not been proven mathematically, and resolving the question whether SAT has a polynomial-time algorithm is equivalent to the P vs NP problem.

Unifier (2011)

A substitution σ is a unifier of expressions e_1 and e_2 if $e_1\sigma$ is identical to $e_2\sigma$. That is, a unifier of two expressions is a substitution that when applied to each expressions results in the same expressions.

Constraint Satisfaction Problem (2016, 2015, 2014, 2012)

A mathematical problem defined as a set of objects whose state must satisfy a number of contraints of limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods.

SAT, SMT and ASP can be roughly thought of as certain forms of the constraint satisfaction problem.

Examples:

- Eight queens
- Map colouring
- Sudoku

Some worked examples on Tim's website

Generate-and-test (2016, 2015)

A simple algorithm that guarantees to find a solution if done systematically and there exists a solution.

1. Generate a possible solution
2. Test to see if this is the expected solution
3. If the solution is not found, go back to step 1.

STRIPS (2011)

Standford **R**esearch **I**nstitute **P**roblem **S**olver is an automated planner. The same name was later used to refer to the formal language of the inputs to this planner. This language is the base for most of the languages for expressing automated planning problem instances in use today, called action languages.

Situation Calculus (2011)

A logic formalism designed for representing and reasoning about dynamical domains.

Searches and Graphs

```
search(Node) :- goal(Node).
search(Node) :- arc(Node, Next), goal(Next).
```

Depth-first (2015, 2014, 2012)

Treates the frontier as a stack. Always selects one of the last elements added to the frontier.

Determinised `arc(Node, Next)` to `arcD(NodeList, NextList)`

```
goalD(NodeList) :- member(Node, NodeList), goal(Node).
arcD(NodeList, NextList) :- setof(Next, arcLN(NodeList, Next), NextList).
arcLN(NodeList, Next) :- member(Node, NodeList), arc(Node, Next).
```

Bounded

```
search(Node, s(B)) :- goal(Node).
search(Node, s(B)) :- arc(Node, Next), search(Next, B).
```

Iterative Deepending

```
itSearch(Node) :- bound(B), search(Node, B).

bound(0).
bound(s(B)) :- bound(B).
```

Breadth-first (2016)

Treats the frontier as a queue. Always selects the earliest element added to the frontier.

```
goalL(H|T) :- goal(H); goalL(T).
arcL(NodeList, NextList) :- findAll(X, (member(N, NodeList), arc(N, X)), NextList).
```

Best-first (2016)

Treats the frontier as a priority queue ordered by h. Always selects the element in the frontier with the lowest h-value

A-star (2016, 2015, 2014, 2013, 2012)

Treats the frontier as a priority queue ordered by $f(n) = cost(n) + h(n)$. Always selects the element in the frontier with the lowest estimated distance from the start to a goal node.

Heuristic (2016, 2015, 2014, 2013, 2012, 2011)

$h(n)$ is an estimate of the cost of the shortest path from node n to a goal node. It uses readily obtainable information about the node. It is an underestimate if there is no path from n to a goal that has path length less than $h(n)$.

Admissible (2016, 2015, 2014, 2013, 2012, 2011)

If the heuristic function never overestimates the actual minimal cost of reaching the goal (i.e. if it's admissible) then A* is itself admissible.

A admissible if it will find a solution if there is one, and it will have the minimal f value.*

It is admissible if

- the branching factor is finite
- arc costs are bounded above zero
- $h(n)$ is an underestimate of the length of the shortest path from n to a goal node

Knowledge Bases

g is a logical consequence of KB ($KB \models g$) if g is true in every model of KB. I.e. $KB \models g$ if there is no interpretation in which KB is true and g is false.

g can be derived from KB ($KB \vdash g$) given a proof procedure.

Soundness (2016, 2014)

Every atom g that a proof procedure derives follows logically from KB, i.e. $KB \vdash g$ implies $KB \models g$

Completeness (2016, 2015, 2014)

Everything that logically follows from KB is derived, i.e. $KB \vDash g$ implies $KB \vdash g$

Representation and Reasoning System (2016)

A RRS is made up of:

- Formal language: specifies the legal sentences
- Semantics: specifies the meaning of the symbols
- Reasoning theory or proof procedure: nondeterministic specification of how an answer can be produced

Implementation

- Language parser: maps sentences of the language into data structures
- Reasoning procedure: implementation of reasoning theory and search strategy

Logical consequence (2016, 2015, 2014, 2013, 2012, 2011)

A KB is true in intepretation I if and only if every clause in KB is true in I, in which case I is a model of KB.

g, a conjunction of atoms, is a logical consequence of a KB (written $KB \models g$) if g is true in every model of KB.

Interpretation (2014)

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where:

- D, the domain, is a nonempty set of real world elements to be represented
- ϕ is a mapping from constants to D (as D can't necessarily be represented digitally)
- π is a mapping from each possible predicate to its boolean outcome (eg: $greater(a, b) \rightarrow true$)

If an interpretation assigns the value true to a sentence, the interpretation is called a model of that sentence

Bottom up (2012, 2011)

The bottom-up proof procedure has only one rule of derivation, where if "$h \leftarrow b_1 \wedge \cdots \wedge b_m$" is a clause in the KB and all b_i have been derived, then h can be derived.

This holds when $m = 0$.

- Derives exactly the atoms that logically follow form KB
  - Sound
  - Complete
- Efficient
  - Linear in the number of clauses in KB

Top down (2012, 2011)

The top-down proof procedure starts from a query to determine if it is a logical consequence of KB. It starts with a query "$yes \leftarrow a_1 \wedge \cdots \wedge a_m$" and attempts to reduce it to an answer "$yes \leftarrow$." by replacing the a_i with their bodies in KB.

- 1:1 correspondence between top-down and bottom-up proofs
  - Sound
  - Complete

Satisfiable (2016)

- A formula is satisfiable if and only if it has a model
- A formula is unsatisfiable if and only if it has no models
- A formula is counter-satisfiable if and only if its negation has a model
- A formula is valid if and only if every interpretation is a model (i.e. a tautology)
- A formula is a logical consequence of an axiom set if every model of the axion set is also a model of the formula

Horn Clause (2015, 2013)

A Horn clause is either a definite clause or an integrity clause. Negations ($KB \models \neg c$) and disjunctions ($KB \models \neg c \vee \neg d$) can follow from KBs containing Horn clauses.

Integrity Constraint (2015, 2013)

An integrity constraint is a clause of the form:

$false \leftarrow a_1 \wedge \cdots \wedge a_2$

Definite Clause (2015, 2013)

A definite clause is any regular clause (prolog, less cuts) such as:

$r \leftarrow a_1 \wedge \cdots \wedge a_2$

Conflicts (2016, 2015, 2013, 2012)

A conflict of a KB is a set of assumables (an atom that can be assumed in a proof by contradiction) that make it imply false. A minimal conflict is a conflict such that no subset of it is also a conflict.
Note that there can be multiple minimal conflicts with different numbers of assumables.

Example: if $\{c, d, e, f, g, h\}$ are the assumables

KB:

- $false \leftarrow a \wedge b.$
- $a \leftarrow c.$
- $b \leftarrow d.$
- $b \leftarrow e.$

Either c is $false$ or d is $false$ in every model of KB. If they were both $true$ in some model I of KB, both a and b would be $true$ in I, so the first clause would be $false$ in I, a contradiction to I being a model of KB. Similarly, either c is false or e is false in every model of KB. So

- $\{c, d\}$ and $\{c, e\}$ are minimal conflicts
- $\{c, d, e, h\}$ is also a conflict, but not a minimal one

Complete Knowledge Assumption (2015, 2014, 2013, 2012)

The CKA lets you assume that a database of facts contains all possible information about the system modelled - all facts not listed are assumed to be false.

This is called the closed-world assumption. It can be contrasted with the open-world assumption, which is that the agent does not know everything and so cannot make any conclusions from a lack of knowledge.

Non-Monotonic (2016, 2015, 2014, 2013)

- A definite clause RSS is monotonic: adding clauses doesn't invalidate a previous conclusion
- With the complete knowledge assumption (CKA), the system is nonmonotonic: a conclusion can be invalidated by adding more clauses

Negative-as-failure (2016)

A nonmonotonic inference rule. Used to derive $\neg p$ from failure to derive p.

Abduction and Deduction (2016)

Abduction and deduction are similar processes starting from opposite ends of a proof.

Abduction starts with statements we know to be true in a given knowledge base and seeks to find the simplest clauses that make the statement true (finding an explanation for the statement, in Tim's words).

Deduction starts with a knowledge base and seeks to deduce as many things as possible.

Inductive reasoning is reasoning in which the premises are viewed as supplying strong evidence for the truth of the conclusion. While the conclusion of a deductive argument is certain, the truth of the conclusion of an inductive argument is probable, based upon the evidence given.

There is some debate over whether abductive and inductive reasoning are different.

Consistency (2015)

A knowledge base KB is consistent if and only if its negation is not a tautology, i.e. if there exists a model.

Example of an inconsistent knowledge base: $KB = \{a, \neg a\}$