Artificial Intelligence: Introduction and History

Definition of Artificial Intelligence

Nilsson: the study of intelligent behavior (perception, reasoning, learning, communicating, acting in complex environments) in artifacts.

Ginsberg: the enterprise of constructing an intelligent artifact (a "physical symbol system" that can pass the Turing test)

Disciplines:
software engineering + computer science + philosophy = cognitive science

In reality:
neat explorations into advanced (theoretical, formal) programming techniques

Neats vs Scruffies:
formal logic and proof (the Stanford approach)
vs
experimental programming (the MIT approach)
neural nets, genetic algorithms, simulation

AI Subject Matter (logic + graphs)

Knowledge Representation

formal logic
proof theory
functional and declarative programming
the "Knowledge Level"

Search

non-polynomial algorithms (NP-complete)
state space
brute force (depth first, breadth first)
heuristics
objective functions (hill climbing, best first)
adversary search (taking turns)

History of AI

Aristotle	first formal symbol system	c325 BC
Copernicus	Earth is not the center of the universe	1543 AD
Descartes	Mind =/= Body	1641
Galileo	Math is a world model	1642
Euler	state space	1735
Boole	formal logic	1847
Babbage	analytic engine	1854

Frege	formal computation	1879
Russell & Whitehead	symbolic math foundation	1910
Tarski	theory of reference and meaning	1944
Turing	test of computational intelligence	1950

Formal Knowledge

A **conceptual model** consists of

discrete objects, presumed to exist: the Universe of Discourse
interrelations between objects
functions: compound names for objects and for unnamed objects
relations: truth statements about objects

No matter how the world is conceptualized, there are other conceptualizations that are just as useful.

Information Processing Systems

Information = representation + transformation
(program = data structure + processes)

All representations are content free. A theory of meaning must link representation to reality.

The Physical Symbol System Hypothesis

Newell and Simon, 1972: To resolve the mind-body problem of Descartes:

Minds and computers are physical systems which manipulate symbols.

The ***knowledge level*** is an abstraction layer for Computer Science which unites symbolic computation with world modeling

hardware
assembly, microcode, machine instructions
programming languages
algorithms and data-structures
symbol level
knowledge level

Principles of Representation

Symbol systems = patterns + process

Qualitative (symbolic) information
not numerical (although this is just a stylistic difference)

Inference of "new" knowledge from a base of facts and rules

General principles of representation
  variables, quantification, dynamic binding

Interaction and semantics
  inheritance, context, theory of meaning

Meta-reasoning
  knowledge about what is known or unknown

New control structures
  learning from examples, explanation

Declarative Style

An ***AI program*** consists of
- a set of objects
- a set of functions (names for compound objects)
- a set of relations (facts)
- a set of permissible transformations

Objects and relations form a ***state***.
Transformations move between states.
Algorithms explore/search the ***state space***.
Programmers control the search.

State Space

The collection of facts (the database) at one given time defines the **state** of the world.
All possible state configurations define the **state space**.
To move from one state to another, apply a permitted **transformation rule.**
The state space and the moves between states from a **graph**.

Predicate Calculus

A *general purpose* language for describing objects, facts, and transformations for particular domains. Also called **First Order Logic**.

logic	{and, or, if, not, iff} inference, proof
object domains	{<unique atoms>}
quantification	{all.x, exists.x}
predicates	classes and properties
relations	connections between objects
functions	indirect names

Knowledge Representation

Objects: *Names* are intended to point ot actual concrete finite objects in the application domain. The choice of names is a part of interface/documentation. The actual names don't matter.

Variables: The *patterns* of linkage between variables with the same name determines the meaning of the variable.

Functions: Functions are *indirect names*. They name objects by telling how to get to them. Functions are compound names.

Knowledge Representation Labels

Constants:

names of specific objects:	John, Tuesday, My-Phone-Number
names of specific functions:	House-of[x], Phone-of[x], Truth-of[p]
names of specific relations:	Likes[Mary, Tom], Phone-Number[Tom, x]

Variables:

refer to sets/classes/domains of objects
always scoped/introduced by a quantifier

Knowledge Representation Atoms

Named objects	(object constants)
Indirect/compound named objects	(functions)
Relations between objects	(facts)

Logical connectives (and, or, not, if, iff) connect atoms.
They cannot be used inside atoms.

yes:	eyes-of[John] AND hair-of[John}
no:	(eyes-of AND hair-of)[John]
no:	hair-of[John AND Mary]
yes:	hair-of[John] AND hair-of[Mary]

Knowledge Representation Quantification

Variables name **classes of objects** (all.x) and **arbitrary objects** from a class (exists.x).

Variables are bound to specific objects by the act of **instantiation.**

Quantification provides a mechanism to refer to entire classes and to arbitrary objects.

all.x P[x]	every x for which P[x] is True
exists.x P[x]	some arbitrary x for which P[x] is True

Model Theory

Given an object domain and a collection of functions and relations on objects in that domain, a ***model*** of the domain is defined by the facts:

all atoms (atom-names) are True
all atoms not in the domain are False

Eg: Domain = {Mary, Tom, John}   Relation: {Likes}

all possible states:	*all possible models:*	
Likes[Mary, Tom]	1	empty (no relations true)
Likes[Mary, John]	6	one Likes relation isTrue
Likes[John, Mary]	15	two Likes relations True
Likes[Tom, Mary]	20	three True
Likes[John, Tom]	15	four True
Likes[Tom, John]	6	five True
	1	six True
	64	possible models in total

Entailment (implication) and Computability

P |= Q   (double turnstile)

All models for which the collection of facts in P are True imply that the collection of facts in Q are True for every model.

P |- Q   (single turnstile)

Using the rules of a given system, we can compute Q from P.

The central issue (1920-1970): Just because we can compute it (P |- Q), does that mean that what we compute matches our model (P |= Q), the ways in which the world can actually be?

Correctness

Sound: if P |- Q, then P |= Q
A sound computation always supports the world model.

Complete: if P |= Q, then P |- Q

A complete computation always generates all possible models.

Sound and Complete: P |- Q = P |= Q
The model and the computation represent the same world.

Decidability

Universal: if it can be stated formally, then it can be stated in First Order Logic.

Decidable: the computational procedure will terminate with a Yes/No result.

Semi-decidable: The computation might halt, but you don't know when. It may never halt if you ask the wrong knd of question. What we can't do is ask if questions which depend on the **failure** to prove something:

No: "Check to see if nothing is wrong"
No: "Prove that this search will fail to find X"