AI Exotics

Conventional AI Topics in 1999

knowledge representation and articulation
commonsense and uncertain reasoning, belief structures
inference and deduction, expert systems
search (heuristics, strategies, competitive)
blackboard systems
learning and adaptation
deliberation, planning, and acting
situated modeling
speech and language processing
image understanding and synthesis
manipulation and locomotion (robotics)
autonomous agents and robots
multiagent systems
cognitive modeling
mathematical foundations

Probable Directions of Research over the next few years

representing the environment as messy, changing, and hostile
huge knowledge bases, data mining
interaction with human collaborators
system self-understanding
self-motivated learning and agents
more agents and multiagent systems (the next big thing)

Exotic Topics in AI

neural networks (1985 old, has own technical societies)
multiagent systems (becoming mainstream)
autonomous agents, artificial life
data mining, particularly in dynamic web environmnets
fuzzy logics (1990 old)
constraint reasoning (1980 old)
chaotic systems, cellular automata (1980 old)
genetic algorithms (1990 old)
machine evolution (1996)
reactive machines (1990 old)
robot perception (1980 old but little progress)
diagrammatic systems (1990)
web intelligence

Neural Networks

Densely connected networks with input, middle, and output layers
Inherently parallel architecture
Good for pattern-recognition after training
Sub-symbolic (no map between representation and meaning)
Knowledge (and programming) is in the strength of connections between network units
Each middle unit "averages" inputs, eg: outK@t = Sum(weightJ * inJ@t) + biasK
Weighted summation is a discrete model of differential calculus
Logistic summation: out@t = 1/(1 + e^(sum(weightJ*inJ)))
    simulates perceptual threshhold model
Training is accomplished by modifying weights to meet a performance objective
Programming is reflexive,
    ie the system reprograms its own weights through error feedback
Open issues:
    How to learn to solve a problem without training in the exact problem?
    Are there other useful architectures (compared to three layer approach)?
    Can training be less extensive fro hard problems; does training scale up?
    How can a network generalize to a class of problem when trained in a subset?
    How can you know the state of the system's knowledge? (validation issue)
Experience cautions:
    The training examples must sufficiently constrain the problem.
    Selection of appropriate input data is a design problem.
    Known symmetries must be included in the training set.
    The task needs a probabilistic model.
    Don't train with binary vectors (errors are fatal)
    Course coding (information blurring) is good for pre-training

Cellular Automata

"discrete dynamical systems whose behavior is completely specified in terms of a local relation, much as is the case for a large class of continuous dynamical systems defined by partial differential equations" Tommaso & Toffoli

Simple example is domino runs.
Highly parallel recursive technique
Complex behavior (or is it?) from simple rules
Primary example of self-generating infinite systems
Most simple automata are Turing equivalent
Outcome is not "predictable", since running the automata is the only computational model
Main result: Complexity occurs at phase transitions between simple and random

Agents

Software agents are programs with "attitude", or disposition
Agents act autonomously in support of a user's requirement
Programming techniques for agents include

- reactivity (selective activity depending on context)
- autonomomy (goal directed and self-initializing activity)
- collaboration (attunement to work with other agents)
- knowledge-based (use rules for communication protocols)
- inferential (deduce consequences from inputs)
- temporal (persistence of state and identity)
- personality (believable interface with emotional simulations)
- adaptivity (self-changing over time)
- mobility (migrate to different environments, usually in search)

Purposes of agent approaches
- simplify distributed computing
  - automated interoperability
  - resource management
- improve user interface
  - simulate more human activity
  - relieve burden of direct interface
  - indirect management (vague specification)
- system architecture

Genetic and Evolutionary Algorithms

Genetic algorithms are designed using "natural selection" and random permutation.
Code fragments or bit-strings are modified by
- selection (those meeting a criteria are strengthened)
- crossover (two fragments swap pieces at a cut point)
- mutation (parts of a fragment are changed, usually randomly)

Millions of cycles are required for algorithm growth or evolution
Art is a good application.

Artificial Life

Self-replicating programs (viruses)
True autonomy ("Sorry, Dave, I can't let you do that.")
Graphics for Hollywood
Animal construction kits
Cellular autonoma, chaos, and fractal mathematics
Issues:
- physical grounding hypothesis (representation is not needed)
- What is autonomy? What is emergent behavior?
- What is the relationship between behavior and environment?