Automated Program Learning for AGI

Moshe Looks
madscience@google.com

Outline

- Formulations of program learning & current approaches
  - What distinguishes program learning from ML?
- Some achievements so far
- What program learning can't do
- What program learning *can* do for AGI
- Future

What are Programs?

- Well-specified
- Compact
- Combinatorial
- Hierarchical

What is Program Learning?

- Classical induction
  - `f([a, b, c], 2) = c`
    `f([x, y], 0) = x`
    `f = ?`
- Probabilistic induction
  - Maximize `P(D|H) + P(H)` over all H in some program space
  - Harder: learn the distribution over program space
  - Related: learning algorithms for first-order probabilistic models
- Optimization
  - Maximize `f(x) : X → R` over program space `X`
  - Learn to maximize reward (i.e. reinforcement learning)

What are Program Spaces?

- Functions of some type in a pure fragment of Lisp/ML/etc.
  - E.g. `List of Symbols, Nat → Symbol`
- Untyped treelike structure (s-exprs)
- Arbitrary typed functions
- Arbitrary typed functions + core operations

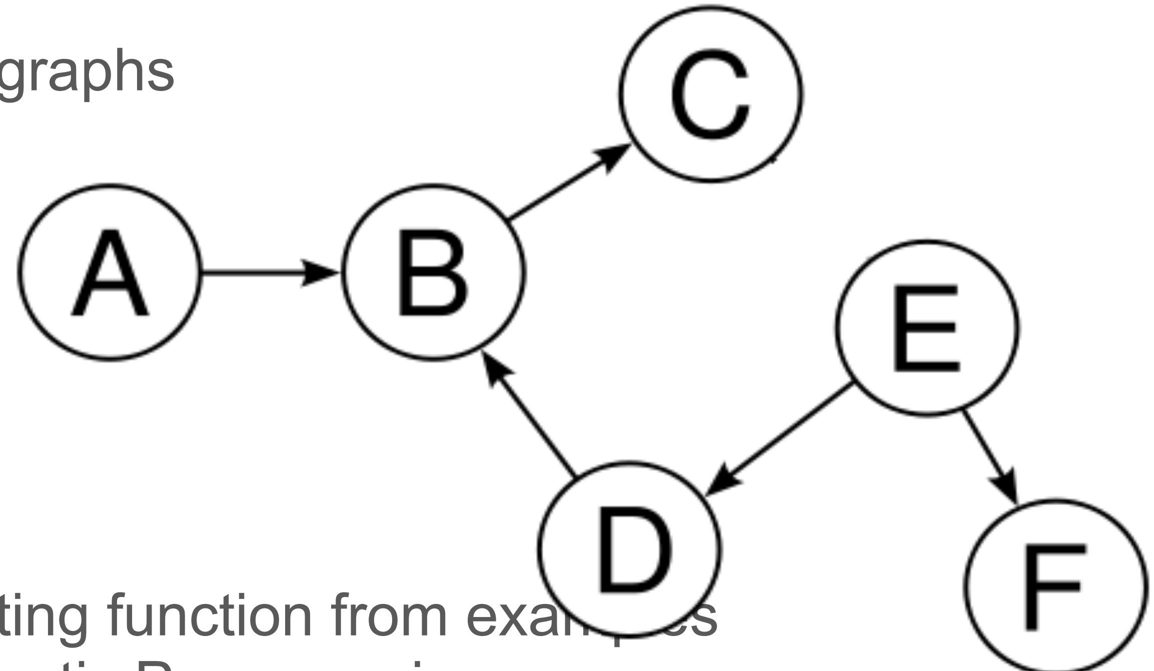
Approaches

- Analytical/Synthetic
  - Summers' synthesis method
  - Some ILP systems
- Generate & Test
  - Local Search
  - Evolutionary
  - Brute-Force

- Hybrid

What's Been Done

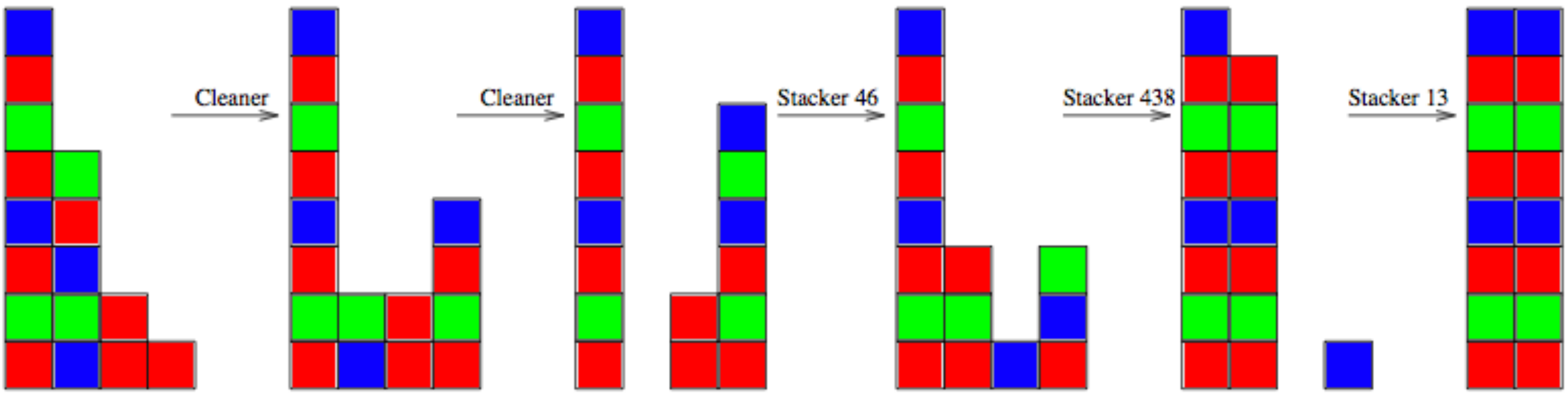
- Path finding in directed graphs
  - ADATE
  - Olsson, 1999



- General O(n*log(n)) sorting function from examples
  - Object Oriented Genetic Programming
  - Agapitos & Lucas, 2006
- Recursive pure functions on lists (*append, reverse, length,etc.)*
  - ADATE, Igor, Igor2 (also handles mutual recursion), MagicHaskeller,etc.

What's Been Done

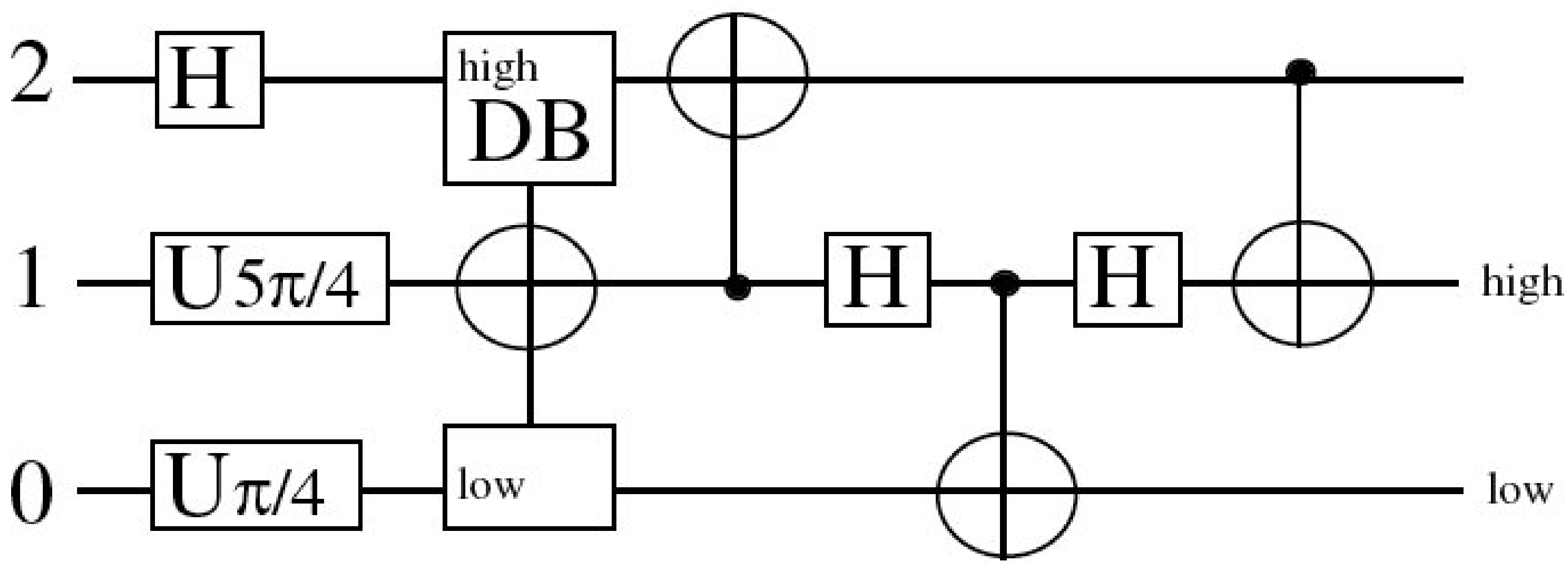
- Block Stacking
  - Hayek-4
  - Baum & Durdanovic, 2000



- Towers of Hanoi
  - Optimal Ordered Problem Solver
  - Schmidhuber, 2006

What's Been Done

- Numerous patentable "human competitive" innovations
  - Quantum algorithms (Spector et al.)
  - Circuits (Koza et al.)
  - More at http://www.genetic-programming.com/humancompetitive.html



"Quantum Computing Applications of Genetic Programming" (Spector, Barnum, and Bernstein 1999).

What's Been Done

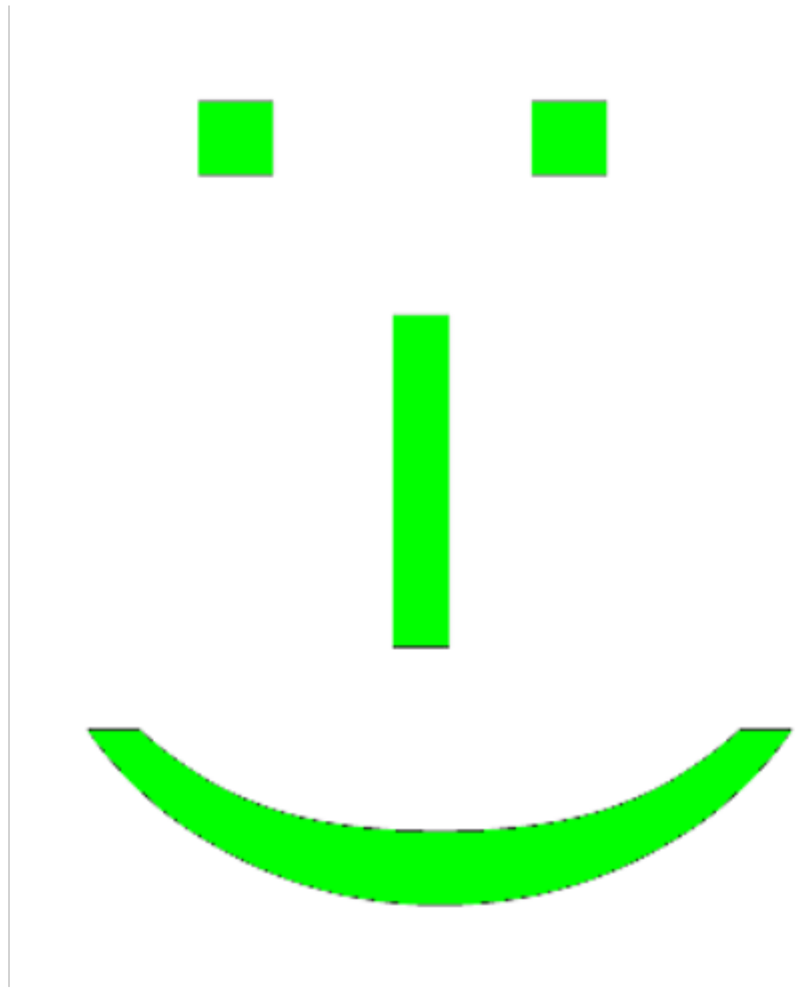
- Unsupervised rule discovery
  - E.g. mining the National Longitudinal Survey of Youth
- Reinforcement learning for agents
  - E.g. Novamente virtual pets

What Program Learning Can't Do

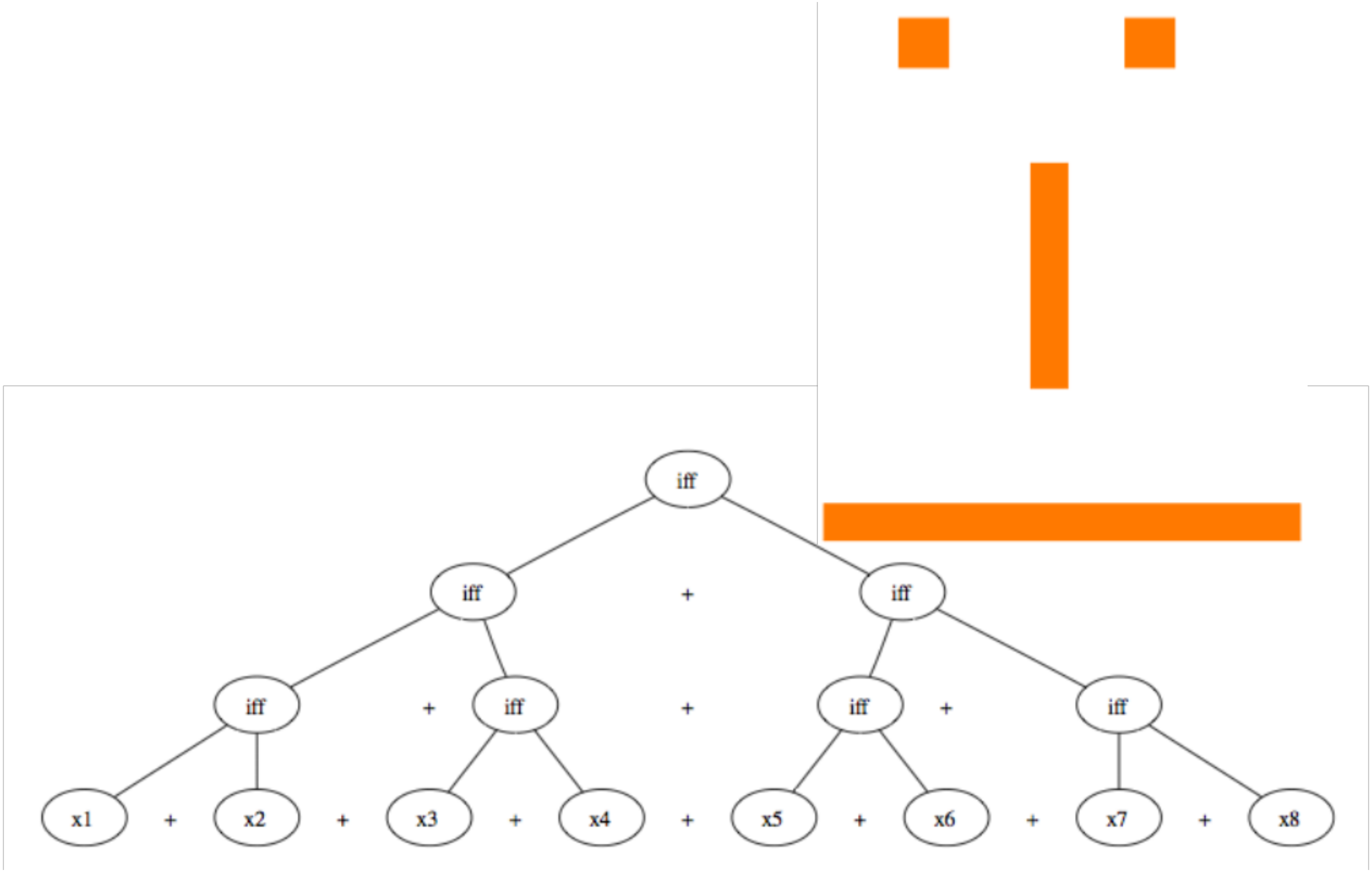
- Can't overrule no-free-lunch
  - Learning is intractable
  - *Averaged over all possible scoring functions* ...

- Can't learn to model "arbitrary" Turing machines
  - Near-decomposability (Simon)
- Can't scale up to large programs
  - Without external guidance
  - *Or* strong (structural) inductive bias
  - *Or* relatedness to past problems

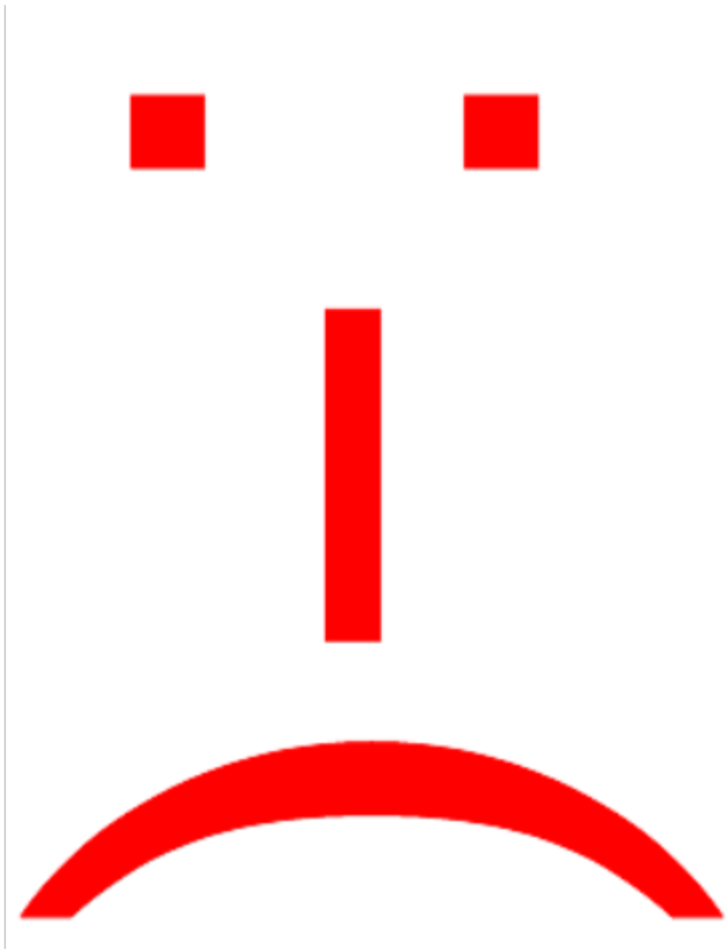
What Program Learning Can't Do

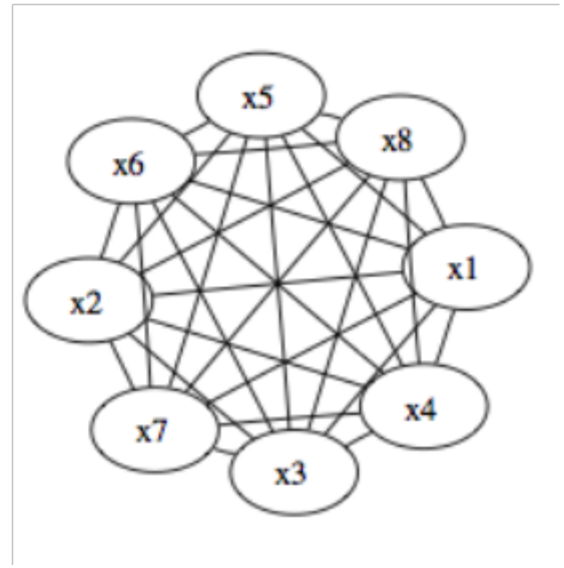


What Program Learning Can't Do



What Program Learning Can't Do





Program Learning for AGI – Two Viewpoints

- Modeling human programmers
  - AM (Artificial Mathematician)
- Modeling human programming
  - Building integrative systems
  - Program learning as one component

"*One understands a problem when one has mental programs that can solve it and many naturally occurring variations*"
- Eric Baum, A working hypothesis for general intelligence

Program Learning for AGI – Desiderata

- Noise tolerance
- “Clear box” evaluation model
- Decent anytime performance
- Handle a full range of types (incl. side effects) & control structures
- *Probabilistic/uncertain semantics for background knowledge*

Warning: Self-Promotion

- MOSES – Meta-Optimizing Semantic Evolutionary Search
  - designed with AGI in mind
- Noise tolerant - can even cope with changes in scoring function
- “Clear box” evaluation model
  - Exploits a core set of functions with known properties
- Decent anytime performance
  - Was generational, transitioning to incremental
- Handle a full range of types (incl. side effects) control structures
  - Working on it; see AGI-09 paper “Program Representation for General Intelligence” (Looks & Goertzel)
- Probabilistic/uncertain semantics for background knowledge
  - Incorporates probabilistic models over program subspaces
  - Working to incorporate models over substructures & functions

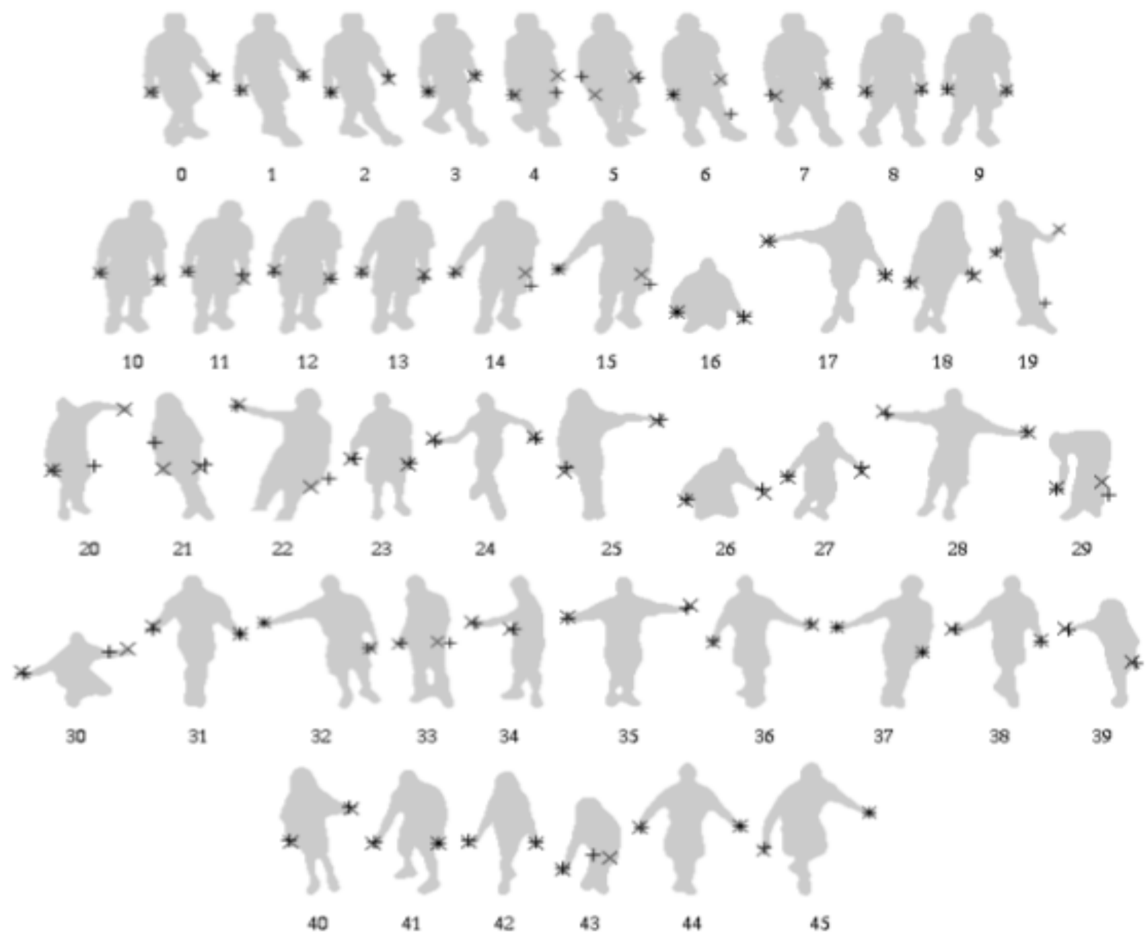
(Uncertain) Logical Inference Rules

- Logical inference (small steps) vs. program learning (big steps)

- Logical inference helps program learning
  - Infer which subfunctions are likely to be useful
    - based on past learning tasks
    - or explict declarative knowledge
  - Infer which programs are worth actually executing
- Program learning helps logical inference
  - Complementary forms of abstraction
  - E.g. compressing/generalizing logical knowledge
  - Tries to validate hypotheses directly
    - i.e. logical inference provides a scoring function
- A major plank of the Novamente design...

Perception and Action

- In some cases more specialized learning algos may be appropriate
- Some success in learning visual routines with GP
    - Johnson, "Evolving Visual Routines"
- Unsupervised learning also possible based on reasoning or interestingness functions



Space, Time, & Language

- Calvin & Bickerton
  - Evolutionary learning in cortical columns
  - Sentences, rock throwing etc.
  - These are programs!
- Computational substrate (Cassimatis et al.)
  - Set of core cognitive mechanisms based on understanding of
    - space & time
    - causality
    - social relations / theory of mind
- Translated to program learning terms
  - Given programs for solving problems in these domains
  - And mechanisms for adapting to solve variations
  - ... and many other domains will fall out quickly

Applying Bruce-Force

- Many approaches to program induction are embarrassingly parallel
- If you can't solve a problem, try doubling the # of machines
- If a problem is of long-term interest, apply unused resources to it

Reliability of Learned Programs

- PAC assurance - compact programs generalize well
  - What if this is not good enough?
- In the general case, can't prove properties of programs
  - Of course particular programs are different
  - Speculation: "learnable" programs will be easier
- Theorem-provers such as ACL2 (A Computational Logic for Applicative Common Lisp) are quite expressive
  - But not very efficient...
  - Recent work on learning over proofs
    - Generalization, Lemma Generation, and Induction in ACL2 (Erickson, 2008)

- Program learning makes theorem-proving more efficient
- Theorem-proving makes (some) learned programs more reliable

Stability Under Self-Modification

- Eventually, want to adapt/improve AGI's source code
  - How can we ensure stability?
  - Do we want to?
- Empirical methods:
  - Important to avoid opacity as much as possible
  - Clear-box program learning helps here...
- Formal methods:
  - Prove invariant properties as self-modifications are introduced
  - Hard problem: prove that such properties hold to begin with
  - What sort of properties?
    - no currency leaks (rationality)
    - no resource leaks (efficiency)
    - properties of goals (very hard problem)

Thank You!

Q&A

Moshe Looks
madscience@google.com