Program Representation for General Intelligence

Moshe Looks
madscience@google.com

Ben Goertzel
ben@novamente.net

Intro: The Importance of Program Representation

- What are programs?
  - well-specified
  - compact
  - combinatorial
  - hierarchical

- Why use programs for AGI
  - all of the above
  - compression = understanding
    - expressiveness allows for compression

- Why program representation is important for AGI
  - not in the limit
  - in practice, matters a great deal ...

Program Spaces are Not Nice

- Open-endeness
  - programs vary in shape and size
- Over-representation
  - syntax != semantics
- Chaotic execution
  - similar syntax !→ similar semantics
- High resource-variance
  - programs vary in memory and space requirements

Solution?

Solution?

More Knowledge!

Definitions

- Let *S* be a space of programmatic functions of the same type
  - e.g. λ-expressions mapping from lists to numbers
  - typically, the distance metric is implied
    - by the set of allowed transformations to program trees
- Let *B* be a corresponding space of program behaviors
  - e.g vectors of sample outputs
    - or probability distributions over such
  - distance reflects our preferences over behaviors
- Let *P* be probability distribution over *B*
  - describes what sorts of problems we expect
- Let $R(n) = \{s \in S \mid size(s) \leq n\}$

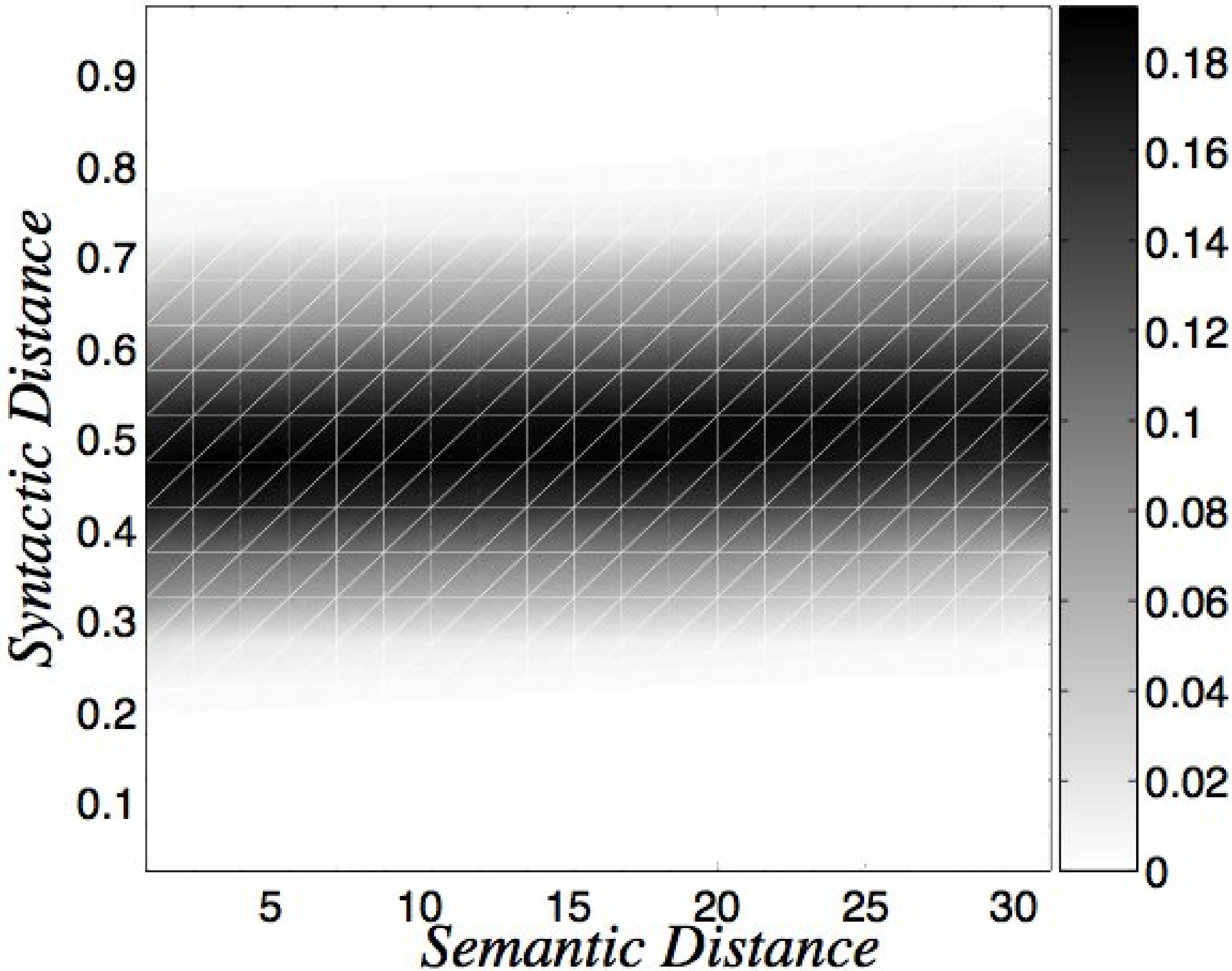
One More Definition

- $R(n)$ d-covers (B, P) to extent p if:
  - for a random behavior $b \in B$ c
  - chosen according to P
  - there is some program in $R(n)$ ($\subseteq S$)
    - with behavior within distance d of b
  - with probability p

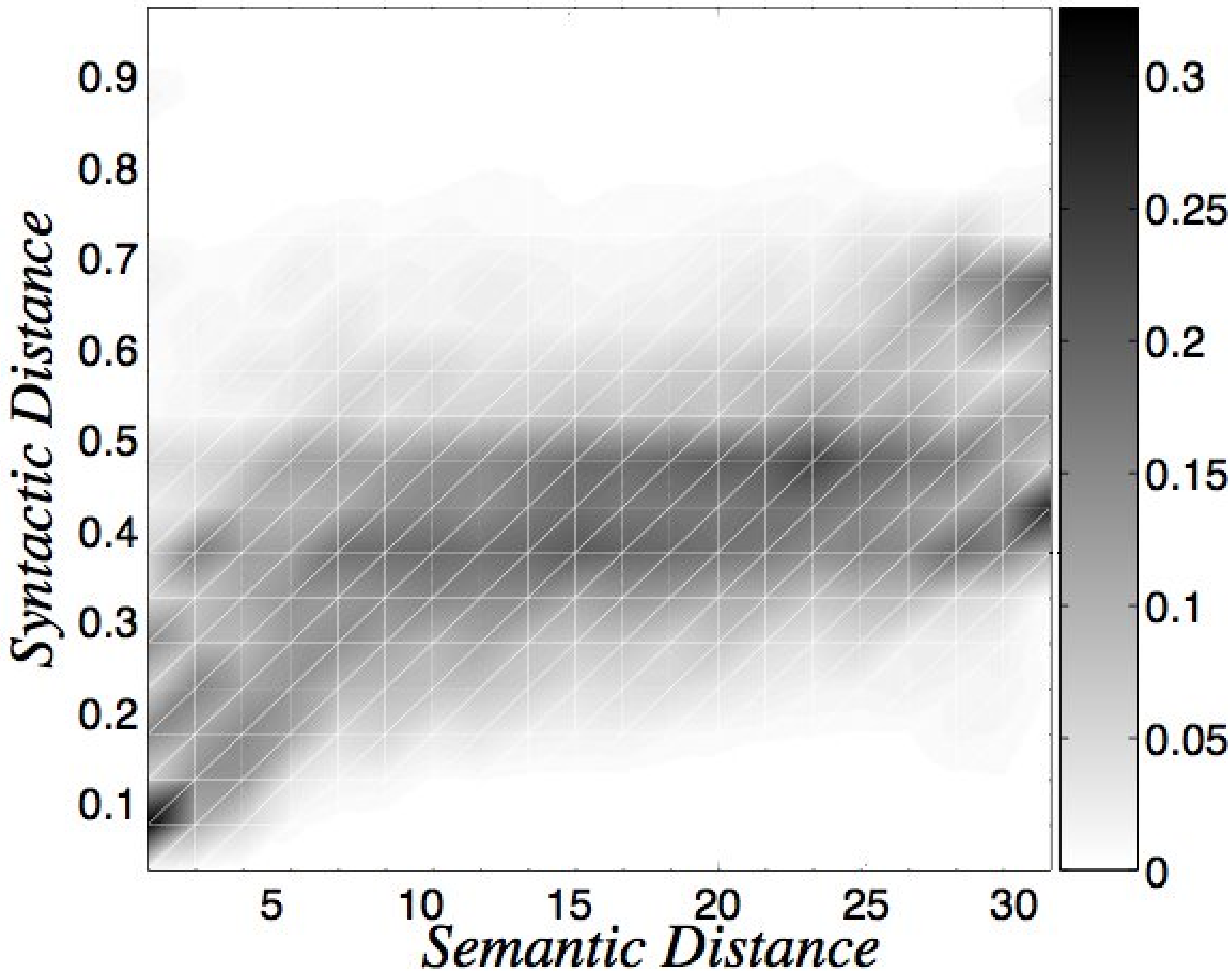
Tractable Representations

- *$R(n)$ d*-covers *(B, P)* to extent p if:
  - for a random behavior $b \in B$ c
  - chosen according to P
  - there is some program in $R(n)$ ($\subseteq S$)
    - with behavior within distance d of b
  - with probability p
- S is tractable if:
  - for fixed d, $p \to 1$ as $n \to \infty$
  - for fixed p, $d \to 0$ as $n \to \infty$
  - for fixed d and p, the needed n is minimized
  - distances in S (syntax) and B (semantics) are highly correlated
    - weighted by P

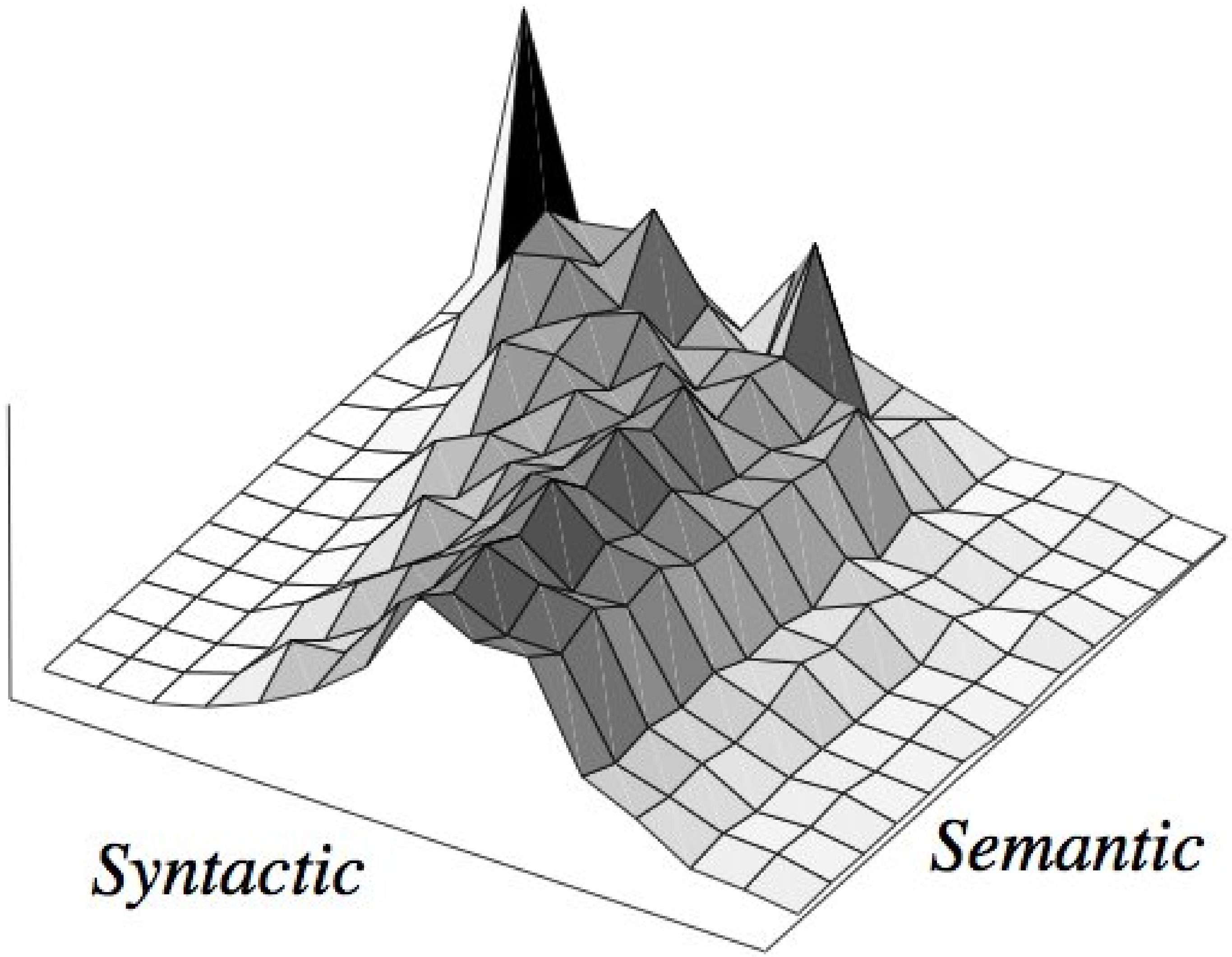
Empirical Demonstration: Before



Empirical Demonstration: After



Empirical Demonstration: After (3D)



Read the Paper to Find Out

- Types
  - Boolean
    - `true, false`
  - Number
    - `-3, 12.34`
  - Lists
    - `[true, true, false], [1, 0, 3]`
  - Tuples
    - `<2, true>, <-3.2, false>`
  - Enums
    - `foo, bar, baz`
  - Functions
    - `f(x,y,z) := if x then y*z else y+z+2`
  - Action Results
    - `and (go-left,grab,go-right,drop)`
    - explains how to handle side-effects

Read the Paper to Find Out

- Reductions
  - `x && y && x →x && y`
  - compressive abstractions
    - introduce new functions to shrink programs
- Rationale
  - reduce the size of the space (n)
  - increase the correlation between distances in S and B

Read the Paper to Find Out

- Neutral Transformactions (via Olsson's ADATE via Kimura)
  - abstraction
    - `x+3*y*z → f(a):=3*y*a, x+f(z)`
  - distribution
    - `x+y*(`**`if`**` p `**`then`**` z `**`else`**` 42) →`
      `if p then x+y*z else x+y*42`
  - arity broadening
    - `f(x,y) → f(x,y,z)`
  - list broadening
    - `f(x) → f([x])`
  - conditional insertion
    - `foo → `**`if`**` true `**`then`**` foo `**`else`**` goo`
- Rationale
  - speed convergence of *p* → 1 (for fixed *d*)
  - speed convergence of *d* → 0 (for fixed *p*)

Read the Paper to Find Out

- Non-neutral Transformations
  - most are specialized by type
  - *`fold`* functions for iteration
    - `fold(+,[2, 3, 4]) = 2 + 3 + 4`
- Scalability
  - ways to heuristically prune transformations
    - reduces the search space
  - might use Sinot's Director Strings
    - distribute function arguments
- Conclusion

Thank You!

Moshe Looks
madscience@google.com

Ben Goertzel
ben@novamente.net