Innate concepts as specialized programs?

Chung-chieh Shan
Rutgers University

Cornell workshop on grammar induction
Commentary on Noah Goodman's talk
'Concept learning as probabilistic program induction'
May 16, 2010

Marr

I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

Question

How to base algorithmic accounts of human performance on Noah's computational models?

- Initial hypothesis: Church's general inference
- Eventual hypotheses: hand-coded special inference

Complaint

Why not discard Church model eventually?
Especially if special inference is approximate. . .

Suggestion

Custom code generation—compile model into inference!

Marr

I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

Question

How to base algorithmic accounts of human performance on Noah's computational models?

- Initial hypothesis: Church's general inference
- Eventual hypotheses: hand-coded special inference

Complaint

Why not discard Church model eventually?
Especially if special inference is approximate. . .

Suggestion

Custom code generation—compile model into inference!

Marr

I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

Question

How to base algorithmic accounts of human performance on Noah's computational models?

- Initial hypothesis: Church's general inference
- Eventual hypotheses: hand-coded special inference

Complaint

Why not discard Church model eventually?
Especially if special inference is approximate. . .

Suggestion

Custom code generation—compile model into inference!

Marr

I represent knowledge in (probabilistic) programming languages for human communication and machine execution.

- Separate what from how
- Reconcile generality with specialization

Question

How to base algorithmic accounts of human performance on Noah's computational models?

- Initial hypothesis: Church's general inference
- Eventual hypotheses: hand-coded special inference

Complaint

Why not discard Church model eventually?
Especially if special inference is approximate. . .

Suggestion

Custom code generation—compile model into inference!

Parnas

‘Domain-general’, ‘language-specific’ are properties of modules.

A module is a part of a description of a system.

- Modularity should be invariant under physically entangled emulation with dye pack.
- Modularity makes a theory more concise, comprehensible.

Organizing principle: reuse in the face of change

Parnas

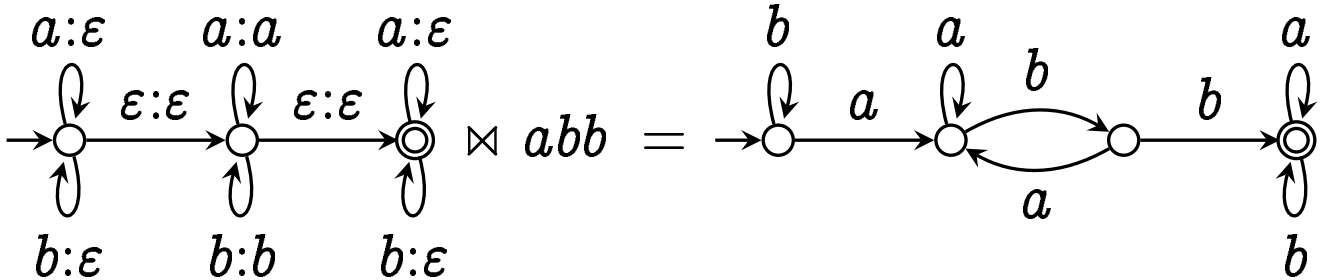
‘Domain-general’, ‘language-specific’ are properties of modules.

A module is a part of a description of a system.

- Modularity should be invariant under physically entangled emulation with dye pack.
- Modularity makes a theory more concise, comprehensible.

Organizing principle: reuse in the face of change

general description $\xrightarrow{\text{specialize}}$ specific machinery



Futamura

Computation: $\lambda x.\ x^8$

Futamura

Computation: $\lambda x.\ x^8$

Algorithm: $\lambda x.\ ((x^2)^2)^2$

Futamura

Computation: $\lambda x.\ x^8$

Algorithm: $\lambda x.\ ((x^2)^2)^2$

Algorithm: $\lambda x.\ f(3)$ where $f(0) = x$

$$f(k+1) = f(k)^2$$

Futamura

Computation: $\lambda x.\ x^8$

Algorithm: $\lambda x.\ ((x^2)^2)^2$

Algorithm: $\lambda x.\ f(3)$ where $f(0) = x$

$$f(k+1) = f(k)^2$$

Algorithm generator: ‘$\lambda x.$’ $f(3)$ where $f(0) =$ ‘x’

$$f(k+1) = f(k)^{\text{`}2\text{'}}$$

Futamura

Computation: $\lambda x.\ x^8$

Algorithm: $\lambda x.\ ((x^2)^2)^2$

Algorithm: $\lambda x.\ f(3)$ where $f(0) = x$

$f(k+1) = f(k)^2$

Algorithm generator: ‘$\lambda x.$’ $f(3)$ where $f(0) = \text{‘}x\text{’}$

$f(k+1) = f(k)^{\text{‘}2\text{’}}$

Computation: $\lambda x.\ x^{10}$

Algorithm: $\lambda x.\ ((x^2)^2 \times x)^2$

Algorithm generator: ‘$\lambda x.$’ $g(10)$ where $g(1) = \text{‘}x\text{’}$

$g(2n) = g(n)^{\text{‘}2\text{’}}$

$g(2n+1) = g(2n)\ \text{‘}\times x\text{’}$

Summary

A module is a part of a description of a system.

$$\text{general description} \xrightarrow{\text{specialize}} \text{specific machinery}$$

‘Compile time’ includes evolution.