Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Probabilistic Programming for Scientific Discovery

Lecture 1

Ludger Paehler

Lviv Data Science Summer School

July 29, 2020

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Table of Contents

Outline

Course Outline

Example Applications of Probabilistic Programming

- ETALUMIS: Bringing Probabilistic Programming to Scientific Simulators at Scale
- DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning

Why Do We even Need Probabilistic Programming?

Underlying Theoretical Ideas

Different Types of Probabilistic Programming Systems

Course Outline

- 4 Lectures
  1. Foundational Knowledge
  2. Inference Engines & Introduction to Turing.jl
  3. Hierarchical Bayesian Approaches & Bayesian Deep Learning
  4. The Connection to Scientific Problems
- 3 Tutorials for Self-Paced Consumption
  1. In-Depth Introduction to Probabilistic Programming Systems with Turing.jl
  2. Bayesian Approaches in Probabilistic Programming
     - Bayesian Deep Learning
     - Hierarchical Bayesian Modelling
  3. Machine-Learning Based Design with Probabilistic Programming

Lecture 1

- Example Applications of Probabilistic Programming
  1. *ETALUMIS: Bringing Probabilistic Programming to Scientific Simulators at Scale*
  2. *DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning*
- Why do we even need Probabilistic Programming?
- Underlying Theoretical Ideas
- Different Types of Probabilistic Programming Systems

Lecture 2

- Approaches to Inference - the Inference Engine
- Practical Introduction to a Probabilistic Programming Framework
- Extending our learned ideas to a more complex example

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Lecture 3

- Bayesian Hierarchical Approaches
- Bayesian Deep Learning, including but not limited to
    - Inference Networks
    - Uncertainty Quantification
- Marrying Deep Learning Frameworks with Probabilistic Programming for Type 2 Machine Learning

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Lecture 4

- Interaction with Scientific Simulators
  - What types of simulators would I want to link to?
  - What are the hidden pitfalls?
- Areas of application
  - Robotics
  - Physics
  - Engineering
  - Machine-Learning Based Design
- Extensive Machine-Learning Based Design Example

Outline

Course Outline

Example Applications of Probabilistic Programming

- ETALUMIS: Bringing Probabilistic Programming to Scientific Simulators at Scale
- DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning

Why Do We even Need Probabilistic Programming?

Underlying Theoretical Ideas

Different Types of Probabilistic Programming Systems

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

ETALUMIS

Bringing Probabilistic Programming to Scientific Simulators at Scale

- Blub

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

DreamCoder [1]

Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning

- Constructs domain-specific languages (DSLs) for scientific problems combined with a neural network, which embodies a learned domain-specific search strategy
  - Learns both the system prior and the needed inference algorithm
- Practically constructs a library of symbolic abstractions in a wake-sleep manner and applies said library to the solving of the chosen problem at hand
- Utilizes wake-sleep learning
  - During *sleep* the system consolidates its abstractions from the programs found during *wake* and improves upon the neural network recognition model by imagining new samples
  - During *wake* the generative model is exploited on the problem domain to find the programs with the highest posterior probability

[1] Ellis, K., Wong, C., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A. and Tenenbaum, J.B., 2020. DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. arXiv preprint arXiv:2006.08381.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

DreamCoder

- Knowledge is accumulated in a multilayered hierarchy with knowledge and skills being successively learned over time, i.e. the knowledge is bootstrapped from very simple examples to ever more complex cases
- Can be broken down to a probabilistic inference procedure, i.e. observing task X and inferring program ρ_x to solve task $x \in X$ combined with a prior distribution over program, which migh solve tasks in the domain

$$\rho_x = \underset{\substack{\rho: \\ Q(\rho|x) \text{ is large}}}{\arg\max} \; P[\rho|x, L] \propto P[x|\rho]P[\rho|L], \text{ for each task } x \in X \qquad \textit{Wake}$$

$$L = \arg\max_L P[L] \prod_{x \in X} \max_{\rho \text{ a refactoring of } \rho_x} P[x|\rho]P[\rho|L] \qquad \textit{Sleep : Abstraction}$$

$$\text{Train } Q(\rho|x) \approx P[\rho|x, L], \text{ where } x \sim X \text{ ('replay') or } x \sim L \text{ ('fantasy')} \qquad \textit{Sleep : Dreaming}$$

DreamCoder

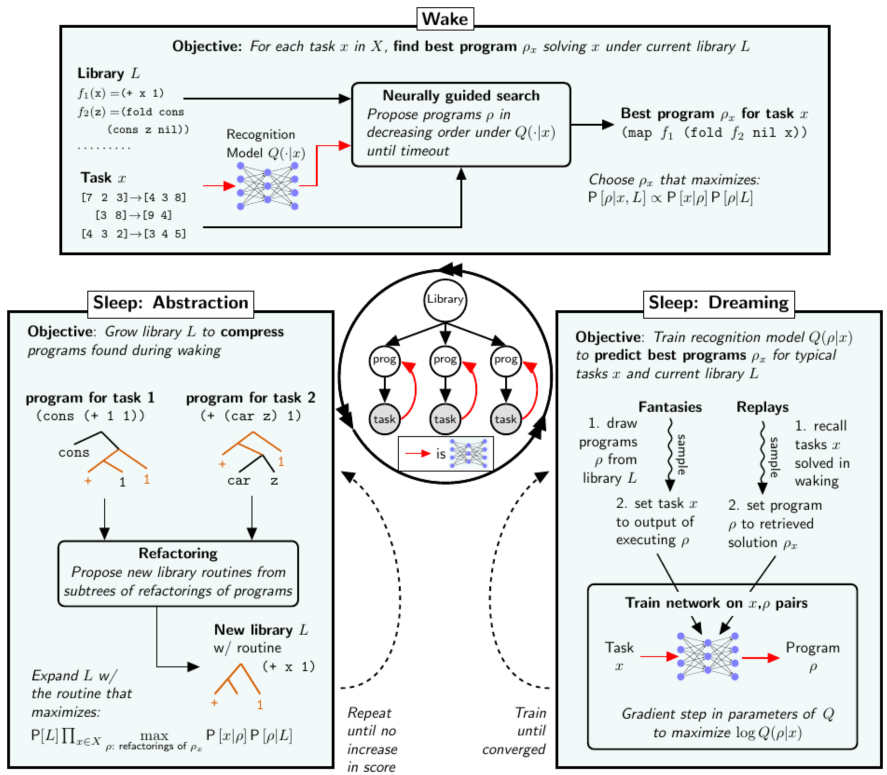


Figure: Algorithm Cycle of DreamCoder

DreamCoder

Algorithm 1 Full DreamCoder algorithm

function DreamCoder(D, X):
Input: Initial library functions D, tasks X
Output: Infinite stream of libraries, recognition models, and beams
Hyperparameters: Batch size B, enumeration timeout T, maximum beam size M
$\theta \leftarrow$uniform distribution
$\mathcal{B}_x \leftarrow \varnothing, \forall x \in X$ ▷ Initialize beams to be empty
while true **do** ▷ Loop over epochs
    shuffle $\leftarrow$ random permutation of X ▷ Randomize minibatches
    while shuffle is not empty **do** ▷ Loop over minibatches
        batch $\leftarrow$ first B elements of shuffle ▷ Next minibatch of tasks
        shuffle $\leftarrow$ shuffle with first B elements removed
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in \text{enumerate}(\mathrm{P}[\cdot|D, \theta], T) \text{ if } \mathrm{P}[x|\rho] > 0\}$ ▷ Wake
        Train $Q(\cdot|\cdot)$ to minimize $\mathcal{L}^{\text{MAP}}$ across all $\{\mathcal{B}_x\}_{x \in X}$ ▷ Dream Sleep
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in \text{enumerate}(Q(\cdot|x), T) \text{ if } \mathrm{P}[x|\rho] > 0\}$ ▷ Wake
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow$ top M elements of $\mathcal{B}_x$ as measured by $\mathrm{P}[\cdot|x, D, \theta]$ ▷ Keep top M programs
        $D, \theta, \{\mathcal{B}_x\}_{x \in X} \leftarrow \text{ABSTRACTION}(D, \theta, \{\mathcal{B}_x\}_{x \in X})$ ▷ Abstraction Sleep
        yield (D, θ), Q, $\{\mathcal{B}_x\}_{x \in X}$ ▷ Yield the updated library, recognition model, and solutions found to tasks
    end while
end while

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

DreamCoder

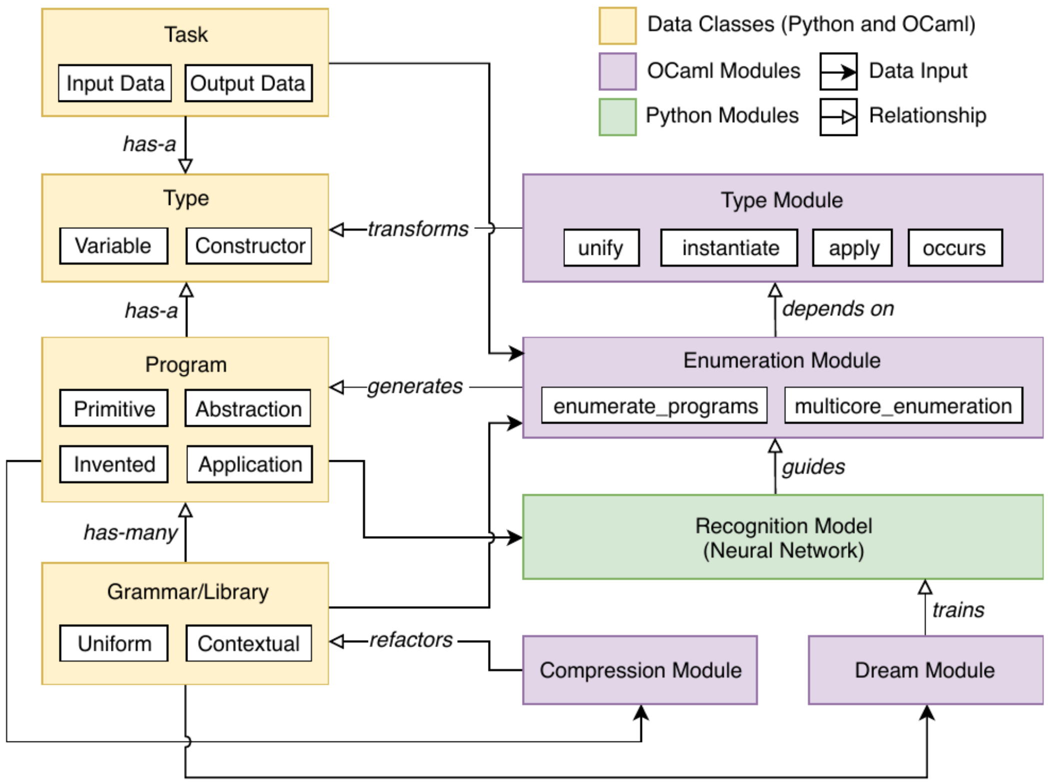


Figure: Different data-classes in DreamCoder.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

DreamCoder

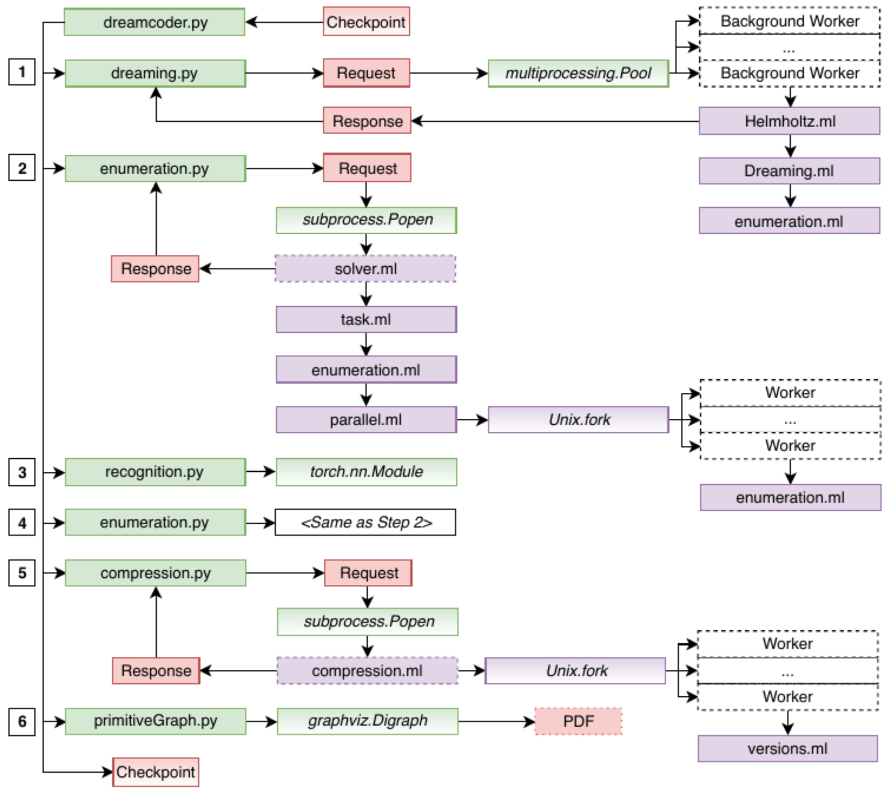


Figure: Program Flowchart: Phase 1, Dreaming. Phase 2, 1st Program Enumeration. Phase 3, Recognition Model Training. Phase 4, 2nd Program Enumeration. Phase 5, Abstraction (Compression). Phase 6, Library Visualization.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Recap: Helmholtz Machine [2] [3]

- Blub

[2] Graves, A., Wayne, G. and Danihelka, I., 2014. Neural turing machines. arXiv preprint arXiv:1410.5401.
[3] YouTube: DeepMind x UCL | Deep Learning Lectures | 8/12 | Attention and Memory in Deep Learning

Recap: Helmholtz Machine

- Blub 2

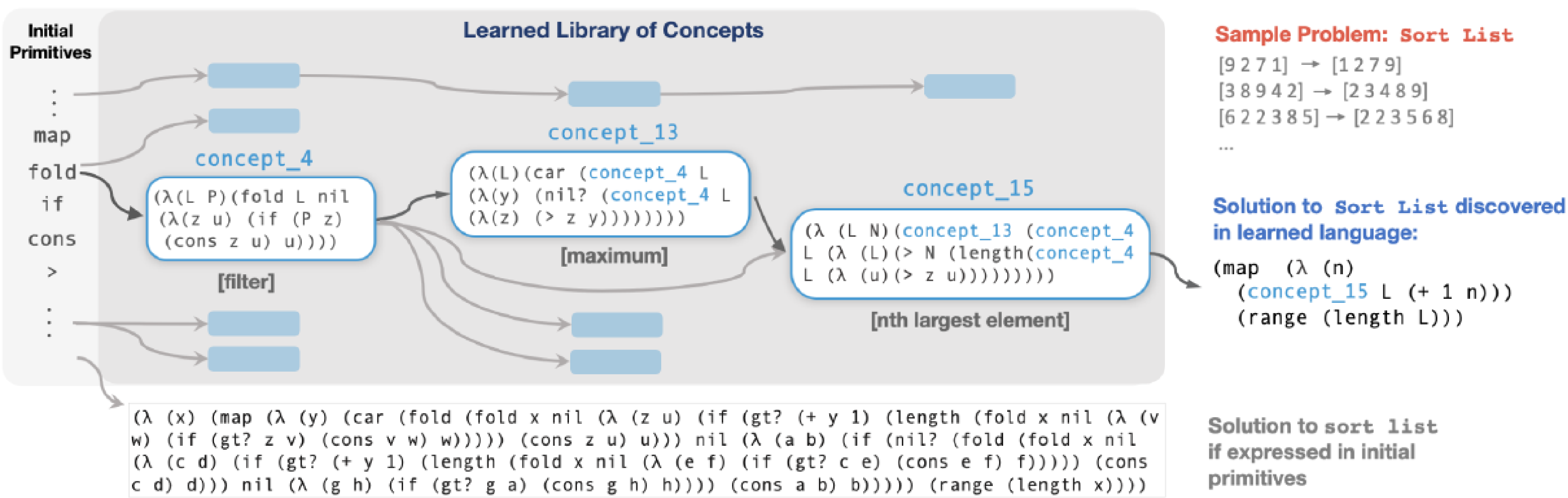
Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Recap: Helmholtz Machine

- Blub 3

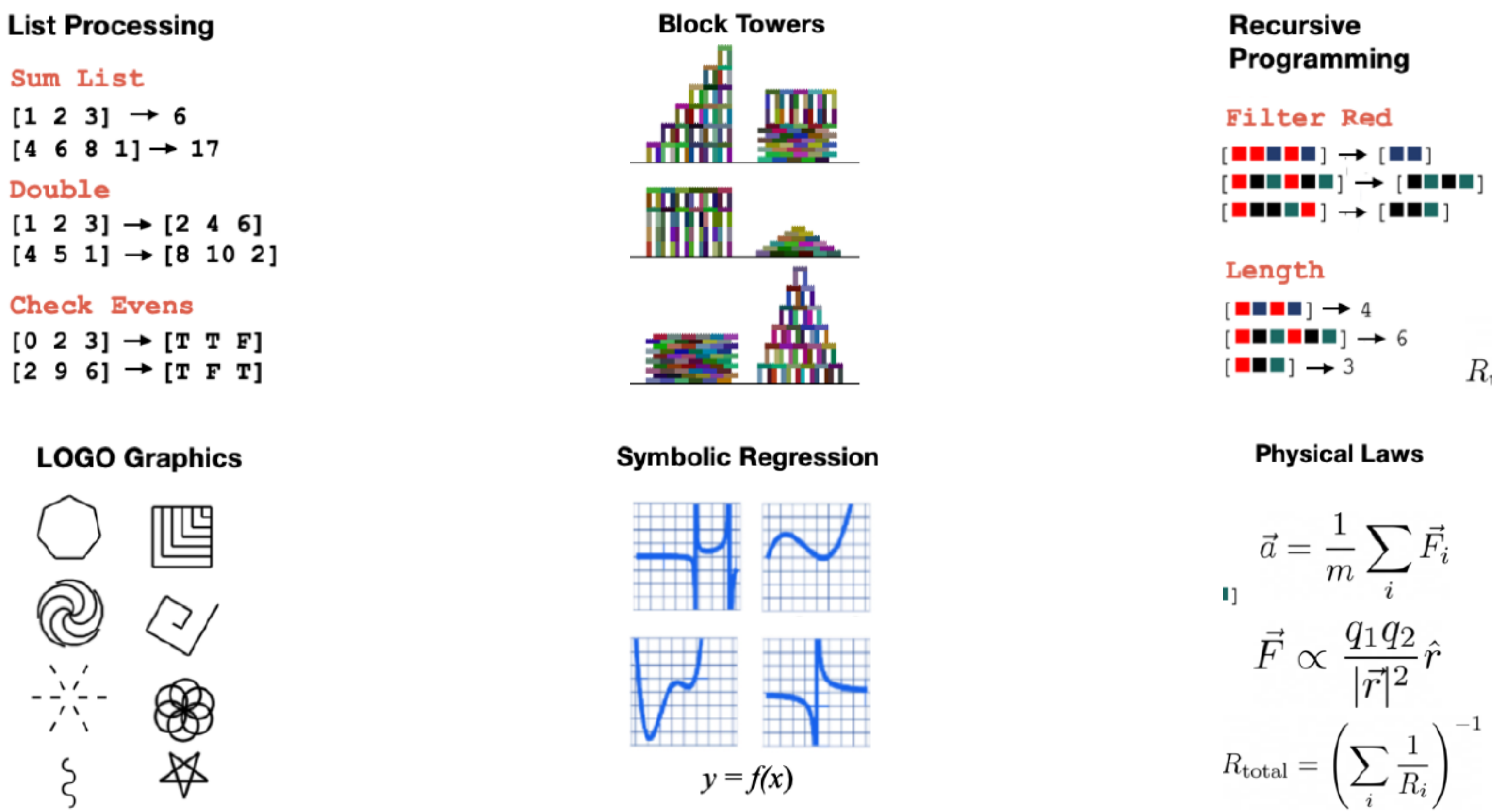
DreamCoder

- Due to its compositional nature, representations of problems can be bootstrapped from earlier, simpler version of the scientific task to more and more complex settings



DreamCoder

Applications



DreamCoder

Applications

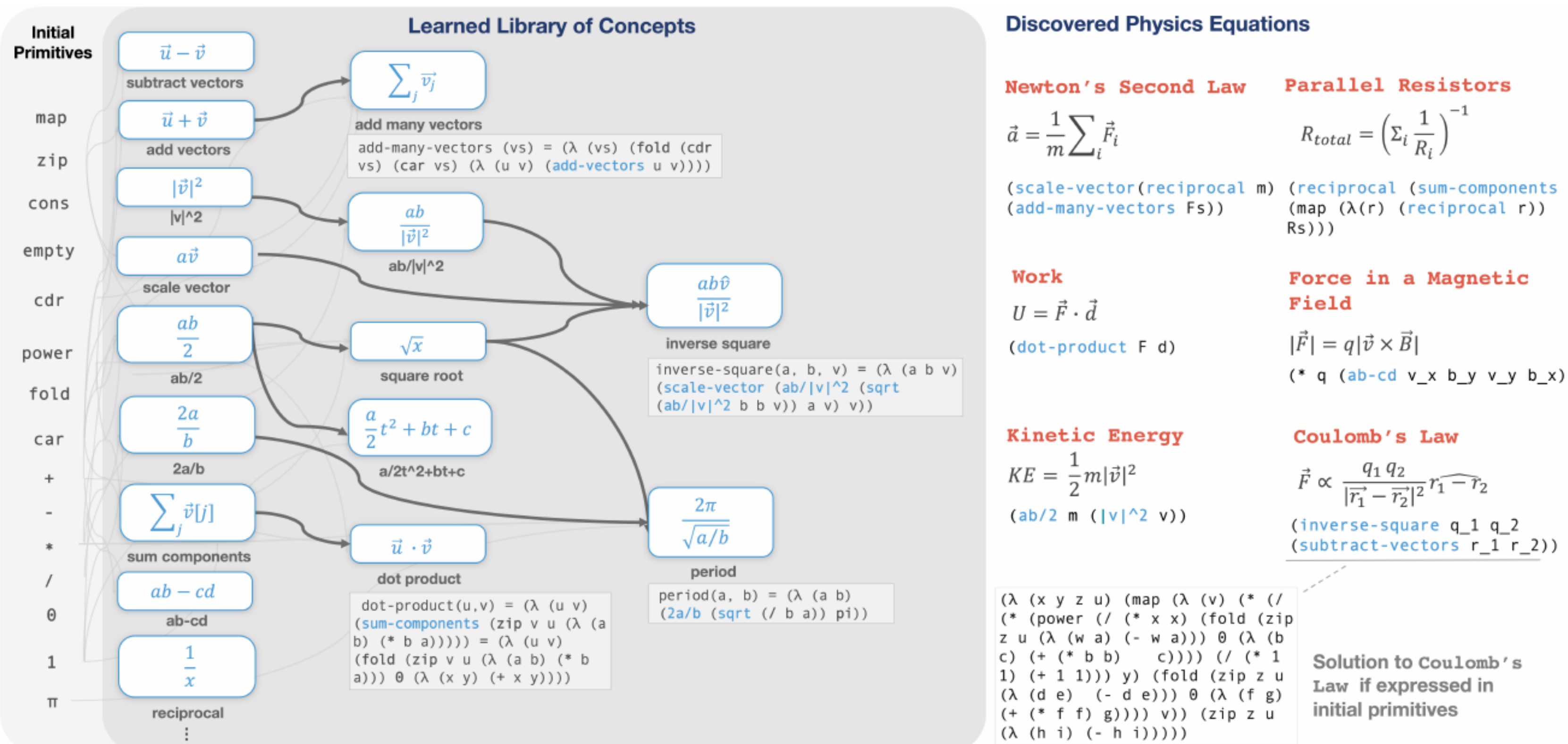


Figure: Learned library for physics equations.

DreamCoder

Applications

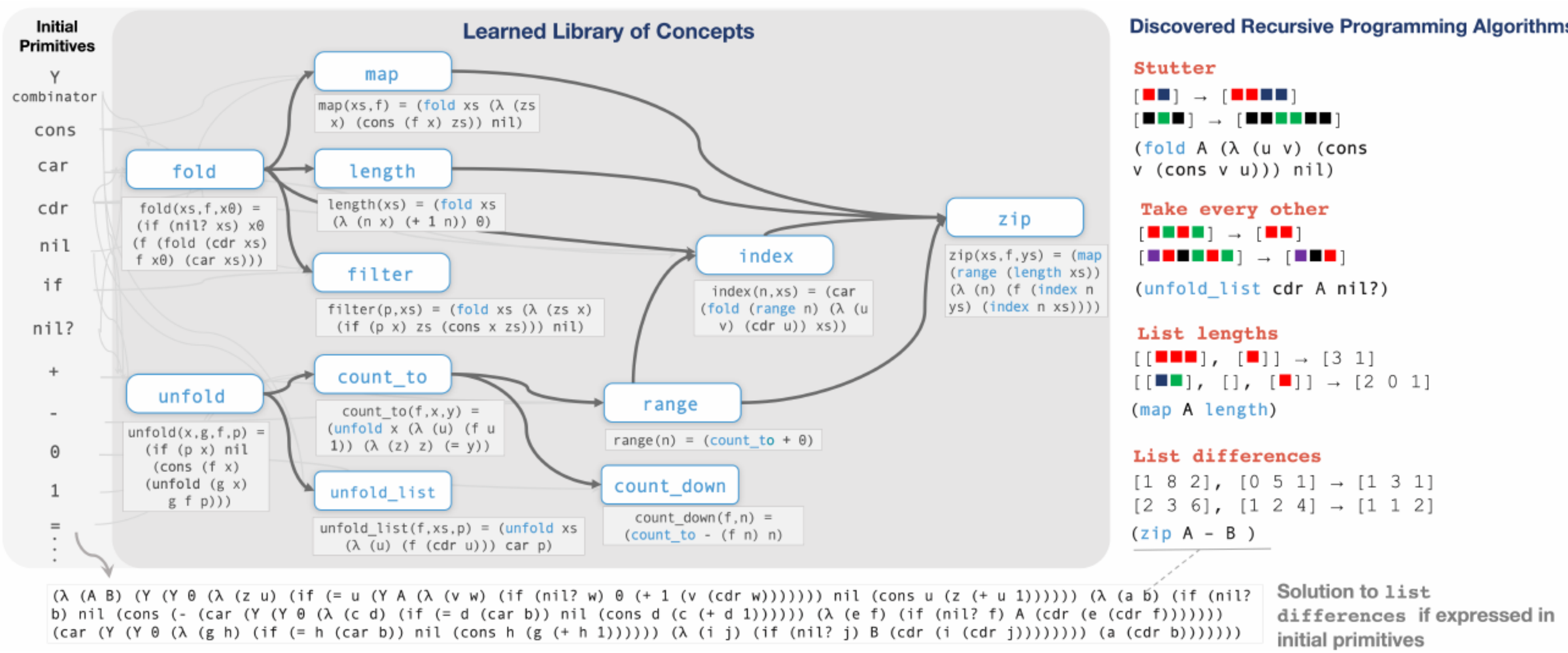


Figure: Learned library for recursive programming algorithm.

DreamCoder

Applications

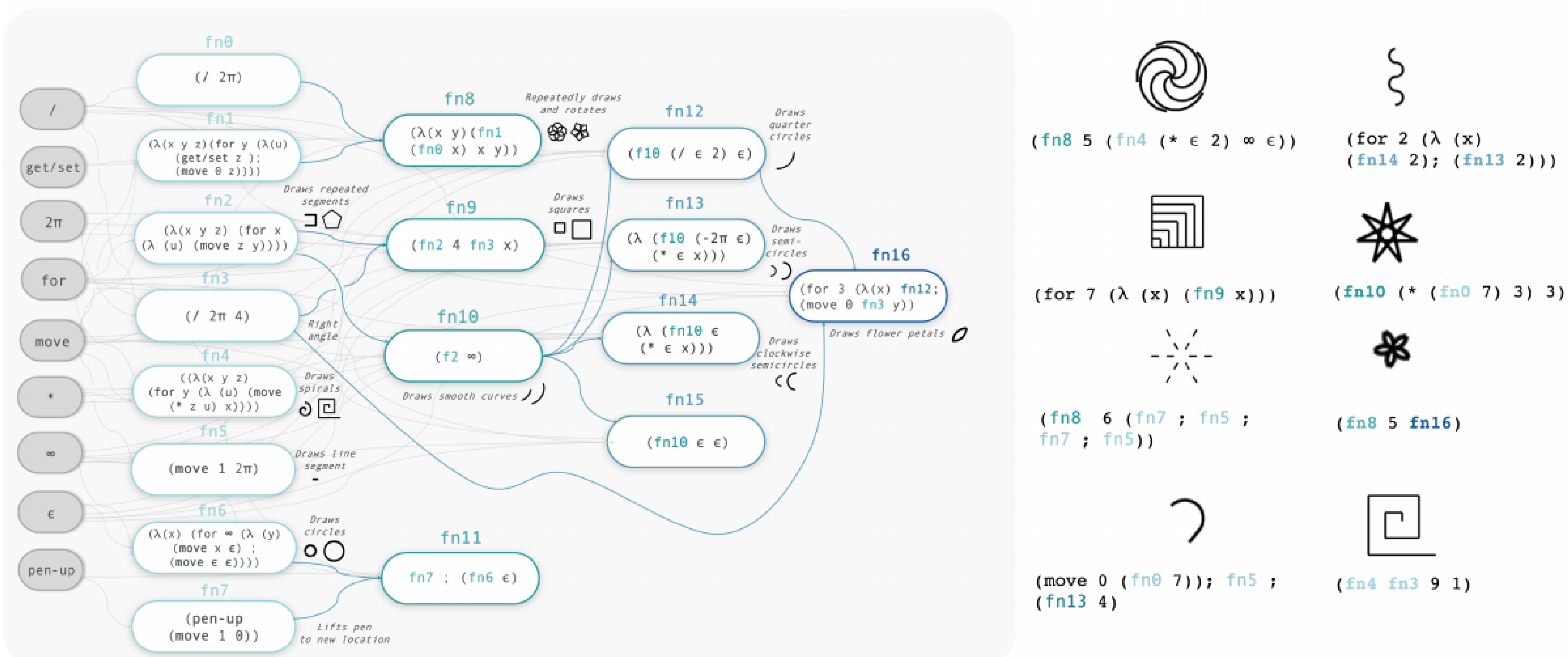


Figure: Learned library for LOGO graphics.

DreamCoder

Applications

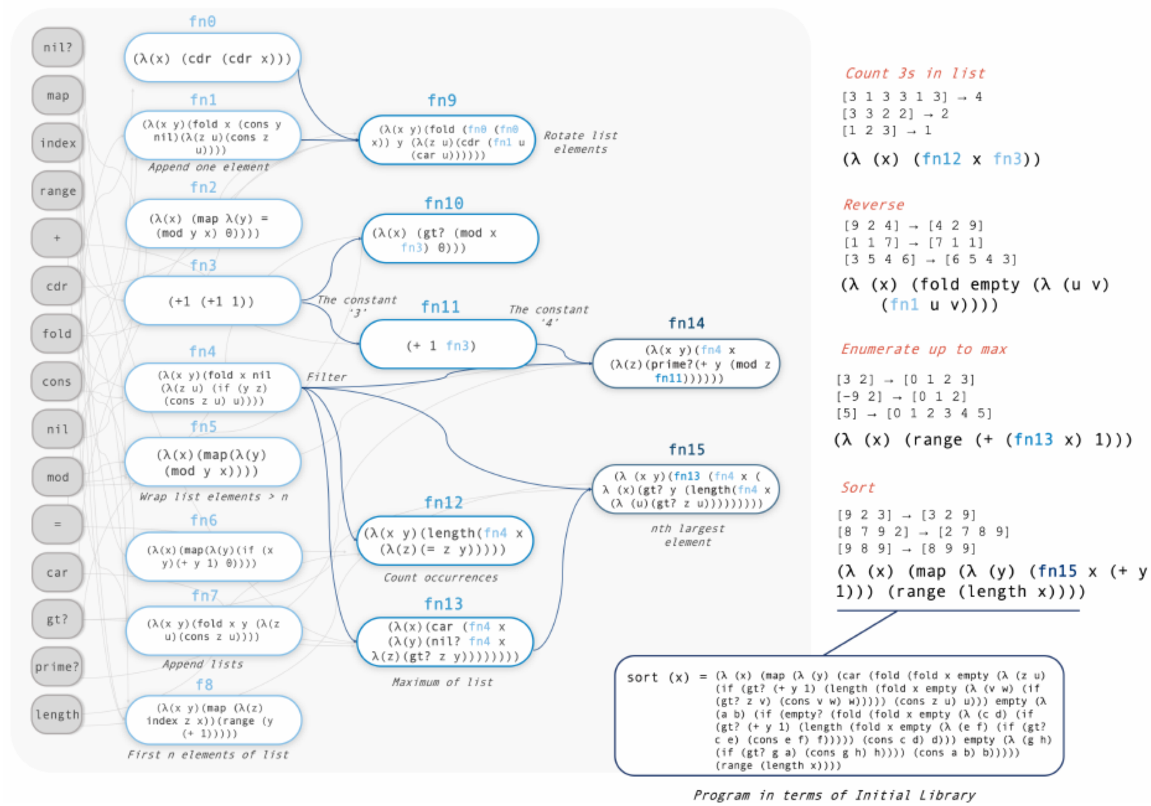


Figure: Learned library for list processing.

DreamCoder

Take-Aways

- Combining probabilistic programming with a DSL-learning procedure and novel probabilistic inference procedure to iteratively learn to represent a problem's domain allows one to gain the ability to solve a problem
- Such applications require highly complex codebase structures across multiple languages
- For more complex examples reliant on a highly efficient inference procedure

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Outline

Course Outline

Example Applications of Probabilistic Programming
- ETALUMIS: Bringing Probabilistic Programming to Scientific Simulators at Scale
- DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning

Why Do We even Need Probabilistic Programming?

Underlying Theoretical Ideas

Different Types of Probabilistic Programming Systems

Outline

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Outline