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

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

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

Lecture 2

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

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

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

Etalumis

Bringing Probabilistic Programming to Scientific Simulators at Scale [2]

- Large scale inverse problem, where a particle simulator is inverted by probabilistically inferring all choices in the simulator given the desired outputs
  - Developed in the context of particle simulations at CERN
- First large-scale application of probabilistic programming to physical simulators in the quest to potentially unearth new physics [1]
- Largest-scale posterior inference with 25000 latent variables at the time
- Amount of compute required highly dependent on the specific approach to inference and the nature of the simulator, i.e. latent dimensionality and intensity of the compute routine
  - Only set to improve with the impeding exascale-era

[1] Cranmer, K., Brehmer, J. and Louppe, G., 2020. The frontier of simulation-based inference. Proceedings of the National Academy of Sciences.

[2] Baydin, A.G., Shao, L., Bhimji, W., Heinrich, L., Meadows, L., Liu, J., Munk, A., Naderiparizi, S., Gram-Hansen, B., Louppe, G. and Ma, M., 2019, November. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-24).

Etalumis

- Proposes a direct linking of particle physics simulators with probabilistic programming systems to trace the internal structure of the simulator
  - Probabilistic programming system controls the random number draws of the simulator akin to samples from prior distributions in Bayesian statistics
- Utilizing inference compilation with three-dimensional convolutional LSTMs to guide the inference procedure and amortize the high computational costs of training
  - Dynamic compilation, in which a core gets expanded with further neural network components as inference compilation proceeds
- Utilizes importance sampling in conjunction with inference compilation as approach to inference

Etalumis

Simulators as Probabilistic Programs

- A simulator execution is viewed as an execution trace, a single sample to the probabilistic programming system
  - I.e. sampling is taking place in the space of execution traces
- Abstracting the simulator in this way enables the following analysis
  - Compute likelihoods
  - Learn/construct surrogate models
  - Generate training data for inference compilation
  - Introduce other generative approaches into the loop
- Enables us to guide the simulation in an intelligent fashion, using the inference network, which acts as a kind of oracle

Etalumis

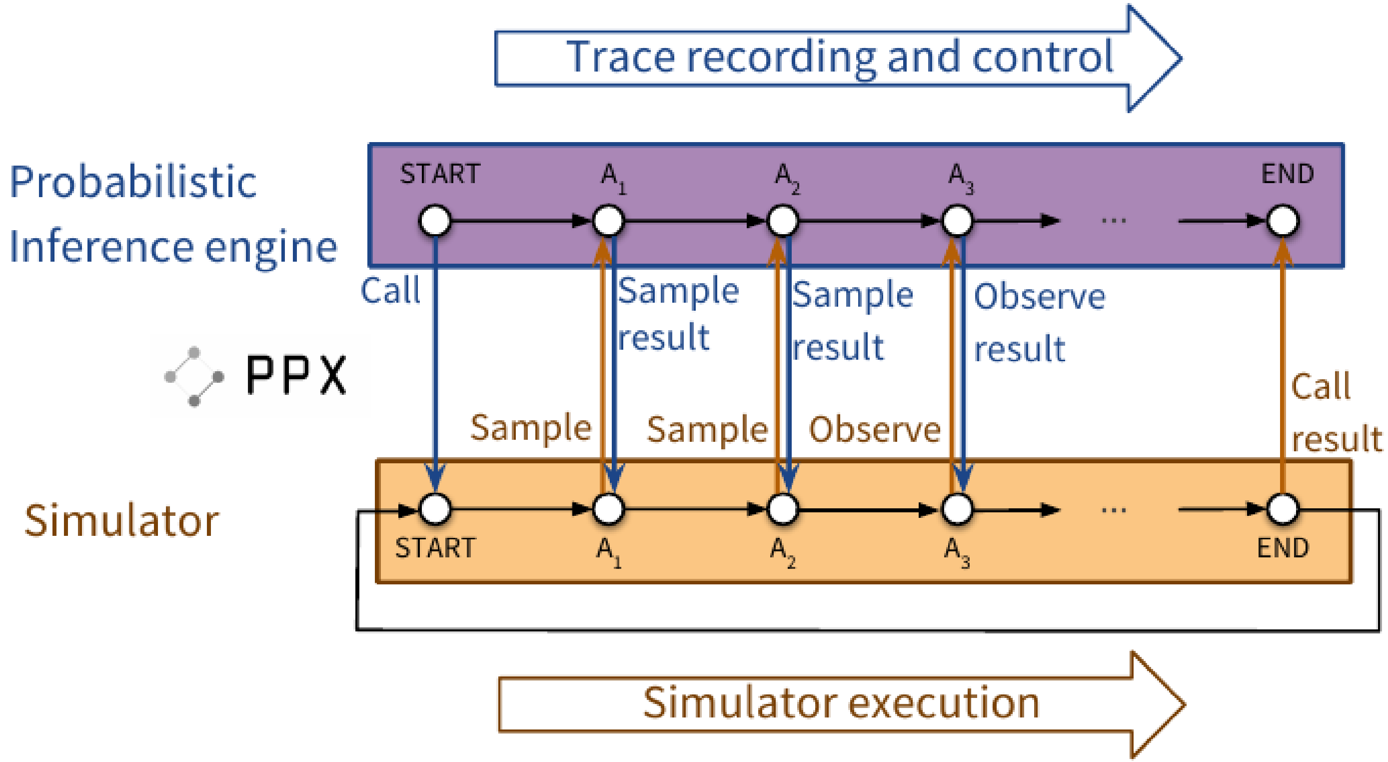


Figure: High-level view from the perspective of the probabilistic programming execution protocol (PPX).

Etalumis

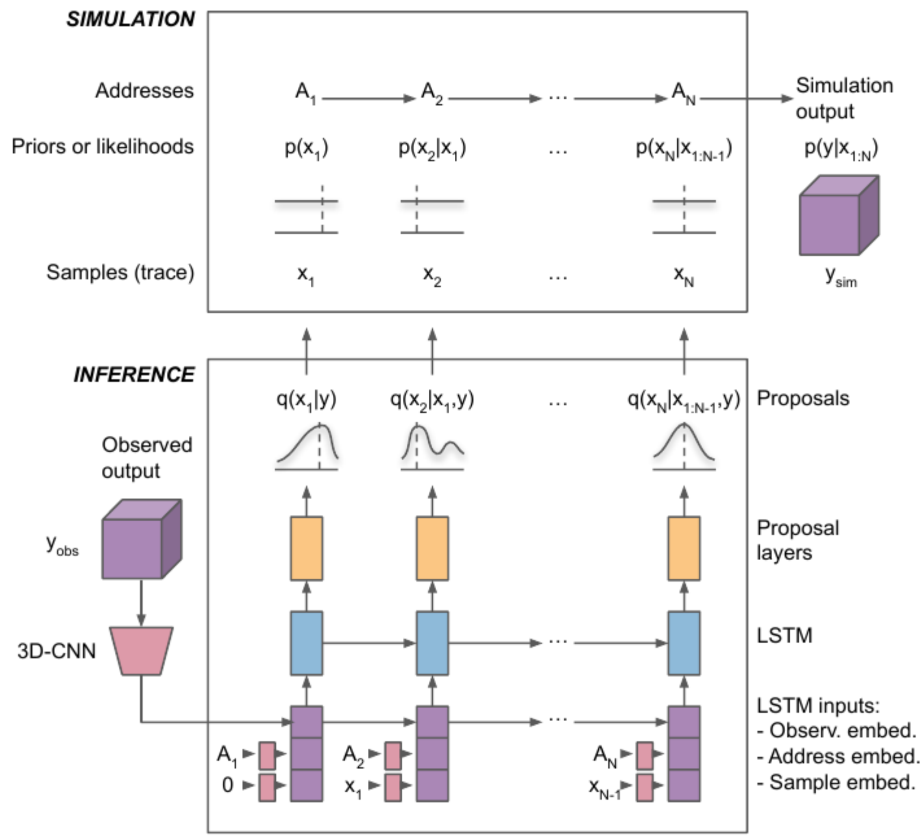


Figure: Detailed connection between simulation and inference in the probabilistic programming-based approach.

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

Etalumis

Recap: Amortized Inference [3] [4]

- Blub 1

[3]Gershman, S. and Goodman, N., 2014. Amortized inference in probabilistic reasoning. In Proceedings of the annual meeting of the cognitive science society (Vol. 36, No. 36).

[4]Ritchie, D., Horsfall, P. and Goodman, N.D., 2016. Deep amortized inference for probabilistic programs. arXiv preprint arXiv:1610.05735.

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

Etalumis

Recap: Amortized Inference

- Blub 2

Etalumis

Recap: Amortized Inference

- Blub 3

Etalumis

Outlook: Inference Compilation [5]

- Uses neural networks to construct a surrogate model for the probabilistic generative model, which is subsequently used at inference time as a custom proposal distribution to avoid sampling from the actual generative model
- Intuition is that the cost of constructing the surrogate can be amortized at inference time and be lower inference from the underlying generative model
  - Need to watch out for sample diversity and out-of-distribution samples
- Proposes adaptive neural network architecture with a recurrent core and embedding and proposal layers specified by the probabilistic program
- Approach is model-agnostic

[5] Le, T.A., Baydin, A.G. and Wood, F., 2017, April. Inference compilation and universal probabilistic programming. In Artificial Intelligence and Statistics (pp. 1338-1348).

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

Etalumis

Outlook: Inference Compilation

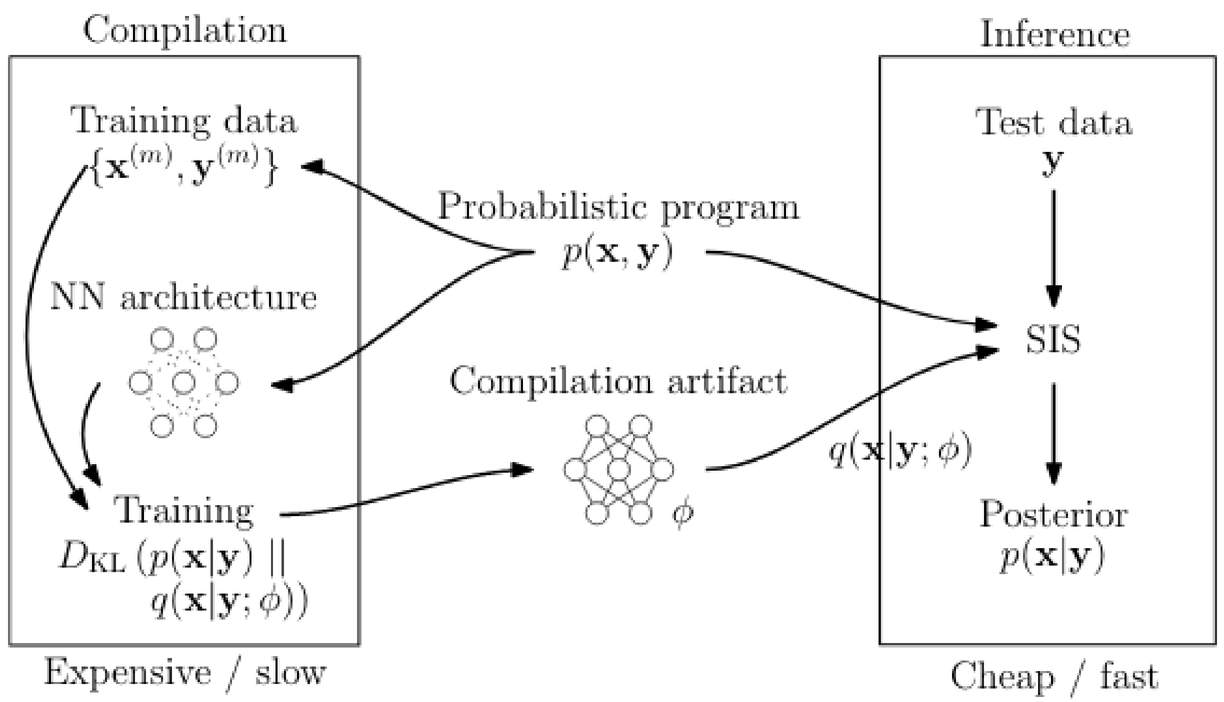


Figure: Automatic construction of a neural network surrogate, which is then trained with data generated by the probabilistic program to eventually act as the proposal distribution at inference time.

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

Etalumis

Outlook: Inference Compilation

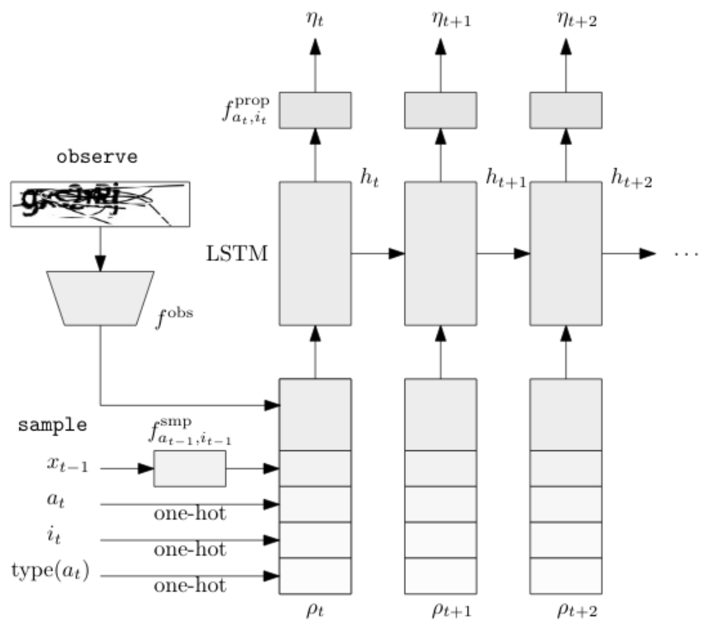


Figure: Example application for captcha solving based on probabilistic generative models for the captchas. With the LSTM at its core, required embeddings for the respective are attached adaptively.

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

Etalumis

Algorithm 2 Distributed training with MPI backend. $p(\boldsymbol{x}, \boldsymbol{y})$ is the simulator and $\hat{G}(\boldsymbol{x}, \boldsymbol{y})$ is an offline dataset sampled from $p(\boldsymbol{x}, \boldsymbol{y})$

Require: OnlineData {True/False value}
Require: B {Minibatch size}
  Initialize inference network $q_\phi(\boldsymbol{x}|\boldsymbol{y})$
  $N \leftarrow$ number of processes
  for all $n \in \{1, \ldots, N\}$ **do**
    while Not Stop **do**
      if OnlineData **then**
        Sample $\mathcal{D}_n = \{(\boldsymbol{x}, \boldsymbol{y})_1, \ldots, (\boldsymbol{x}, \boldsymbol{y})_B\}$ from $p(\boldsymbol{x}, \boldsymbol{y})$
      else
        Get $\mathcal{D}_n = \{(\boldsymbol{x}, \boldsymbol{y})_1, \ldots, (\boldsymbol{x}, \boldsymbol{y})_B\}$ from $\hat{G}(\boldsymbol{x}, \boldsymbol{y})$
      end if
      Synchronize parameters (ϕ) across all processes
      $\mathcal{L}_n \leftarrow -\frac{1}{B} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_n} \log q_\phi(\boldsymbol{x}|\boldsymbol{y})$
      Calculate $\nabla_\phi \mathcal{L}_n$
      Call `all_reduce` s.t. $\nabla_\phi \mathcal{L} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \nabla_\phi \mathcal{L}_n$
      Update ϕ using $\nabla_\phi \mathcal{L}$ with e.g. ADAM, SGD, LARC, etc.
    end while
  end for

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

Etalumis

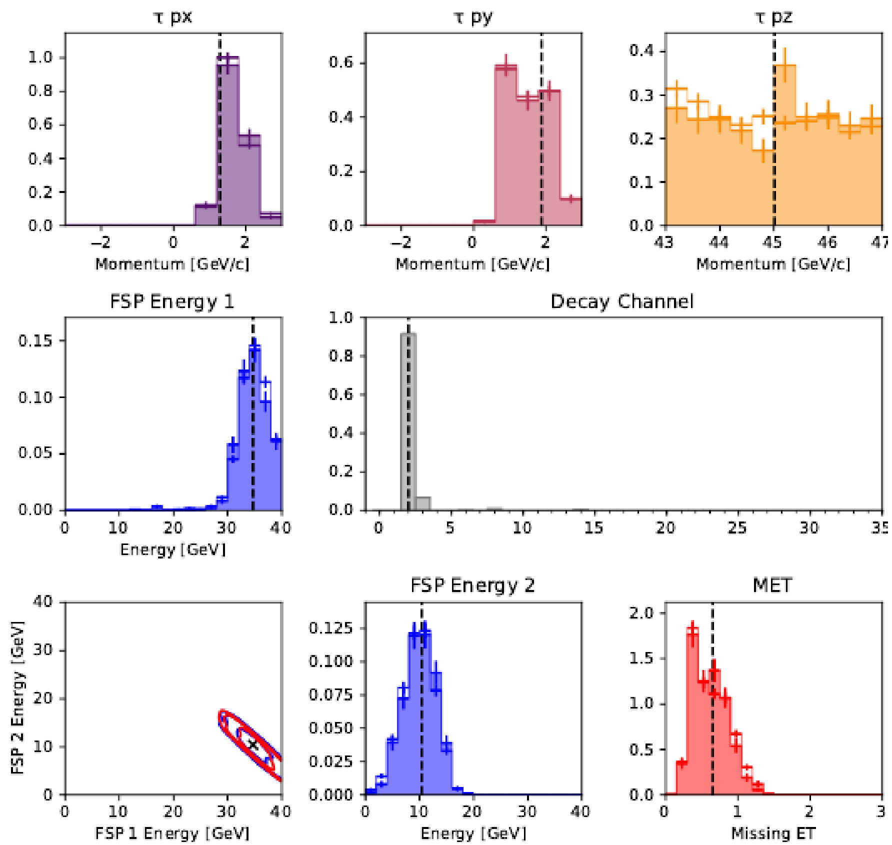
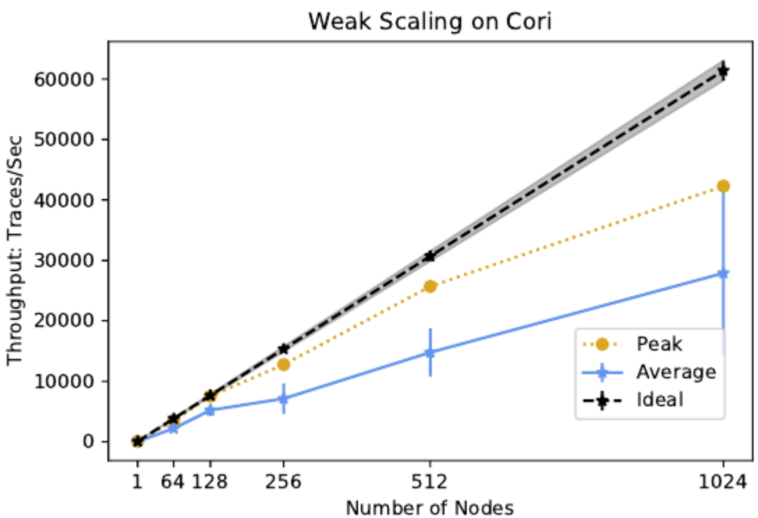
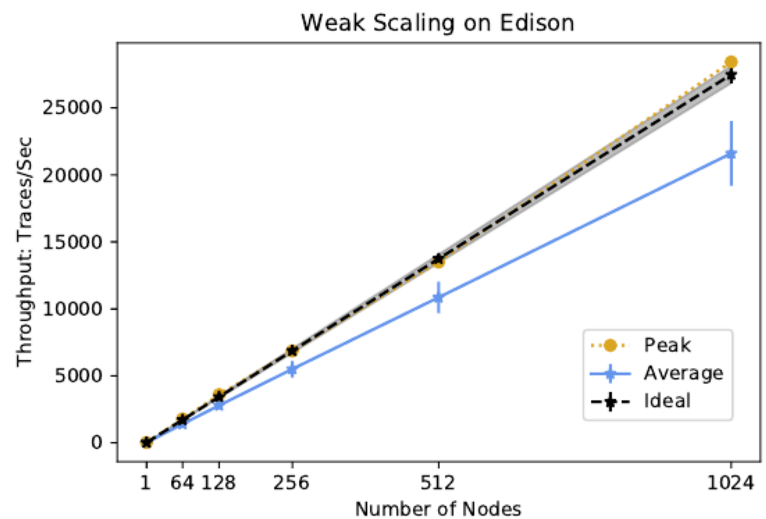


Figure: Posteriors obtained with random-walk Metropolis Hasting (filled histograms) and inference compilation (outline histograms) and ground truth values (dashed vertical lines).

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

Etalumis





- Moving to supercomputer scale for massive-scale inference to be able to generate the necessary number of simulations for the inference compilation to be successful.

Etalumis

- First probabilistic programming system to link to scientific simulators at scale to enable large-scale posterior inference on a supercomputing-scale
  - Can only ever capture the processes encapsulated in the simulator
- Introduces a probabilistic programming execution protocol to link to scientific simulators
- To make inference tractable it needs to rely on techniques, such as amortization through inference compilation, which essentially constructs an oracle
- Pushes the frameworks to the extreme with communication requirements and data exchange between computing instances

DreamCoder [6]

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
- 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

[6] 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.

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

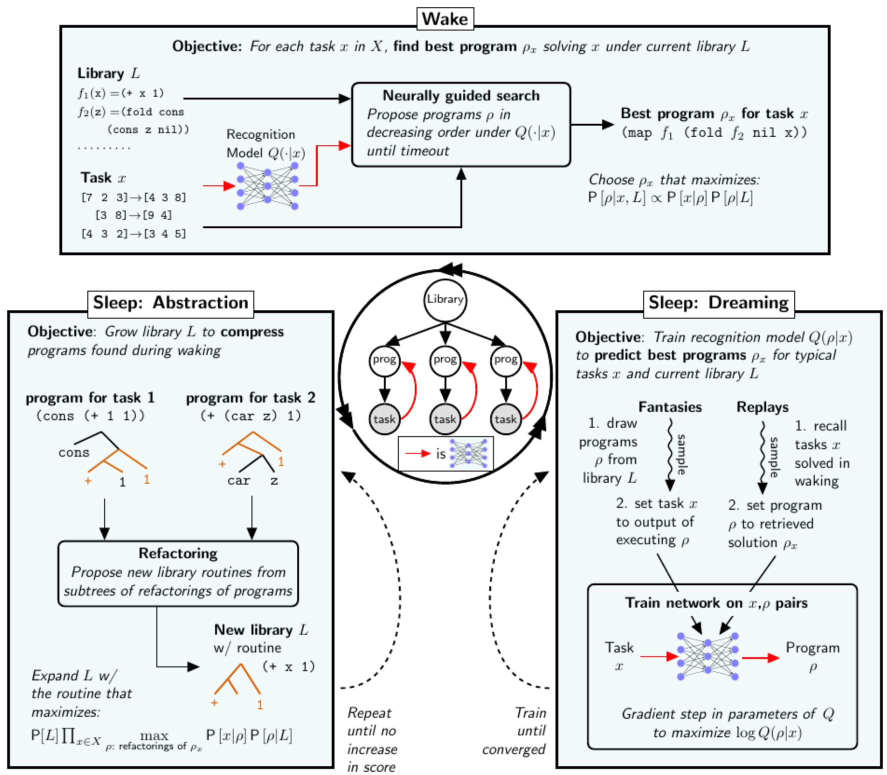


Figure: Algorithm Cycle of DreamCoder

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

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: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in$ enumerate$(\mathrm{P}[\cdot|D, \theta], T)$ 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: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in$ enumerate$(Q(\cdot|x), T)$ 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
        $D, \theta, \{\mathcal{B}_x\}_{x \in X} \leftarrow$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

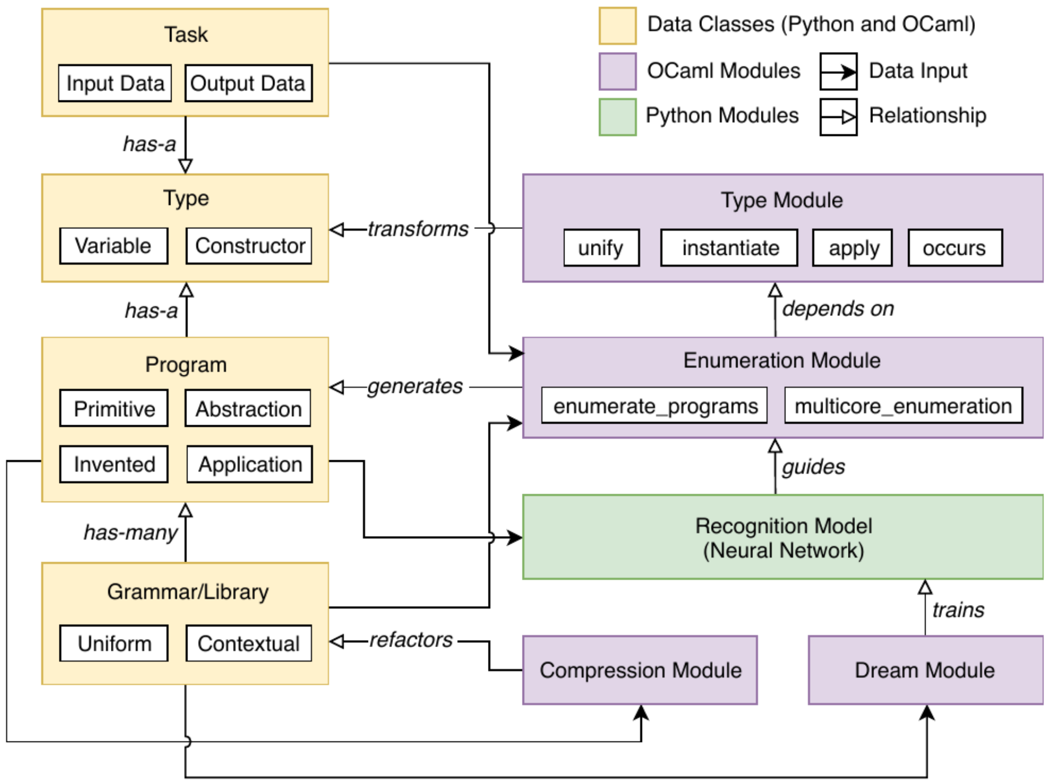


Figure: Different data-classes in DreamCoder.

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

DreamCoder

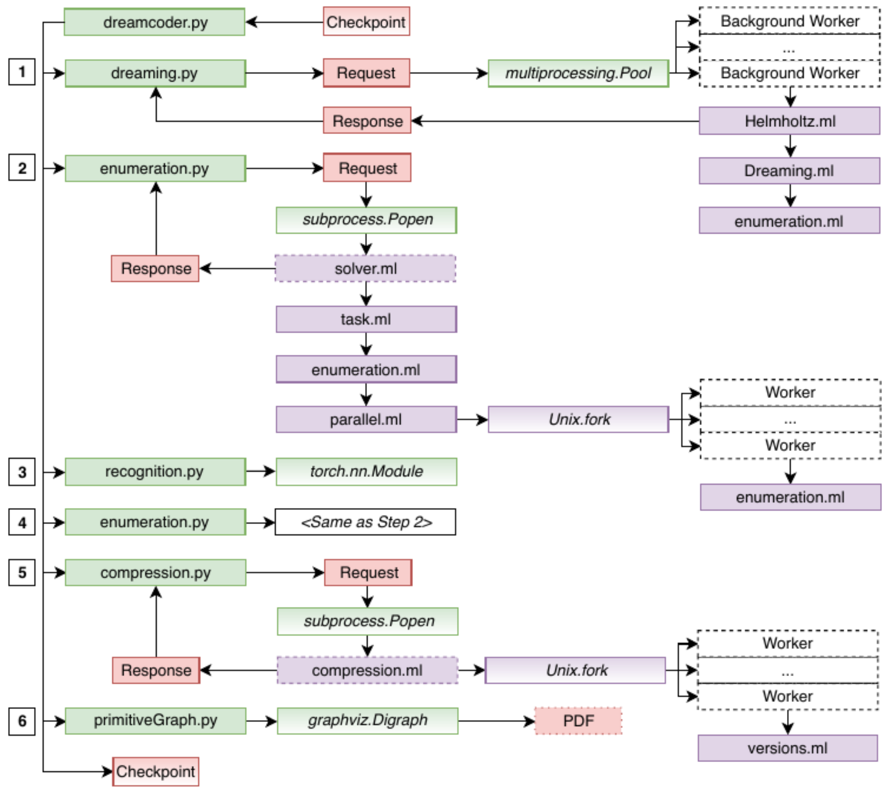
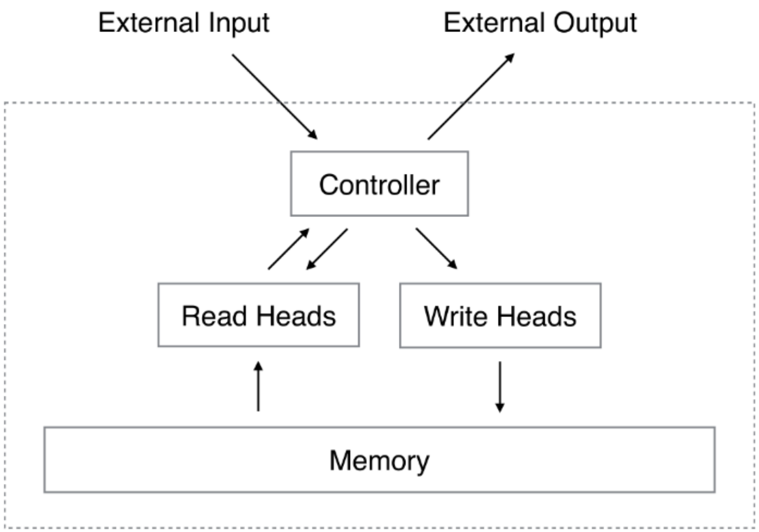


Figure: Program Flowchart: Phase 1, Dreaming. Phase 2, 1st Program Enumeration. Phase 3, Recognition Model Training. Phase 4, 2nd Program Enumeration. Phase 5, Abstraction (Compression). Phase 6, Library Visualization.

Recap: Helmholtz Machine [7] [8]

- Couples neural networks with external memory, which is accessed through an internal attention mechanism
  - Iteratively modify the state of the network through its memory mechanism
- Can infer simple algorithms from data
- Structure
  - Controller is a neural network
  - Heads select the part of the memory to access
  - Memory is essentially a large matrix



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

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

Recap: Helmholtz Machine

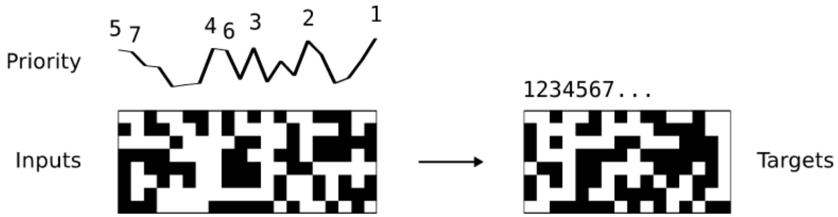
- Heavily relies on selection attention, whee the controller emits a distribution over the memory matrix, which then defines content- and location-based attention mechanisms [9]
- Content-based:
  - A key vector is compared to each memory location
- Location-based:
  - Use a shift kernel in conjunction with the weighting to shift to a new location in memory
- Results in three different interaction modes:
  1. Content key only
  2. Content and location
  3. Location only

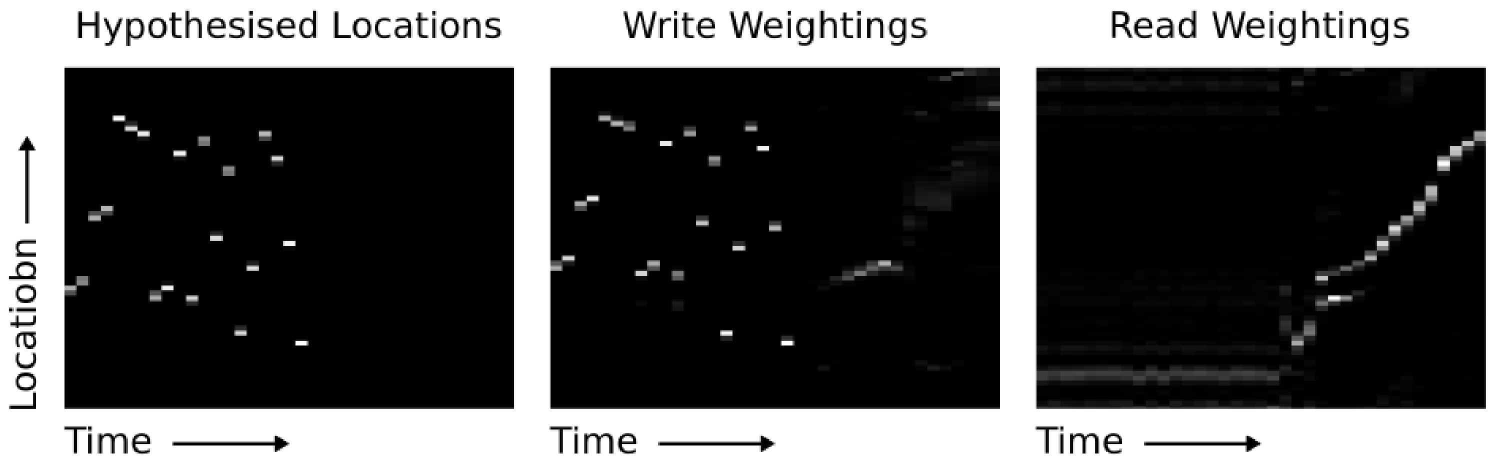
[9] Lilian Weng's review of attention: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Recap: Helmholtz Machine

Example: Priority Sort

- Learns algorithm to sort data from a sequence of random binary vectors and their respective scalar priority rating

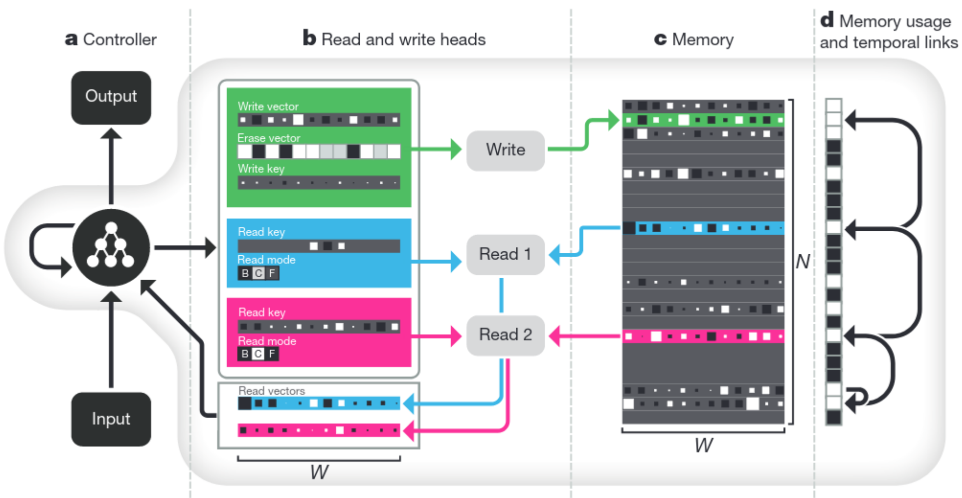




Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Outlook: Differentiable Neural Computer [10]

- Successor architecture to the neural turing machine with new attention mechanisms
- Specifically geared towards applications in graphs ⟶ more on this later!

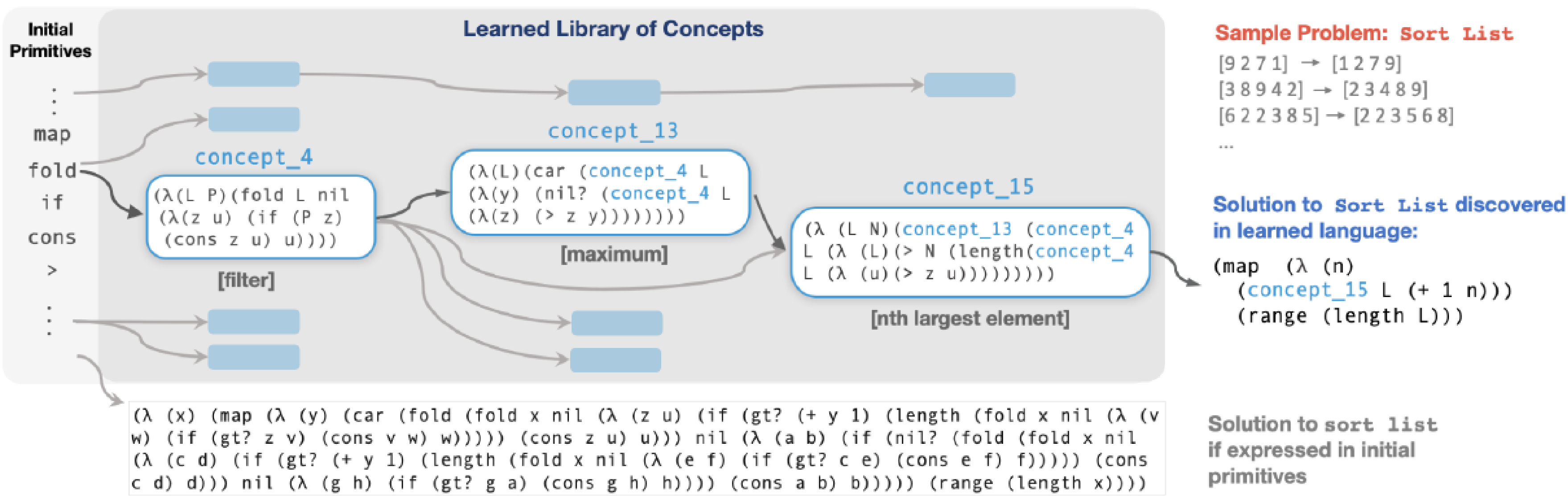


[10] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J. and Badia, A.P., 2016. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), pp.471-476.

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

DreamCoder

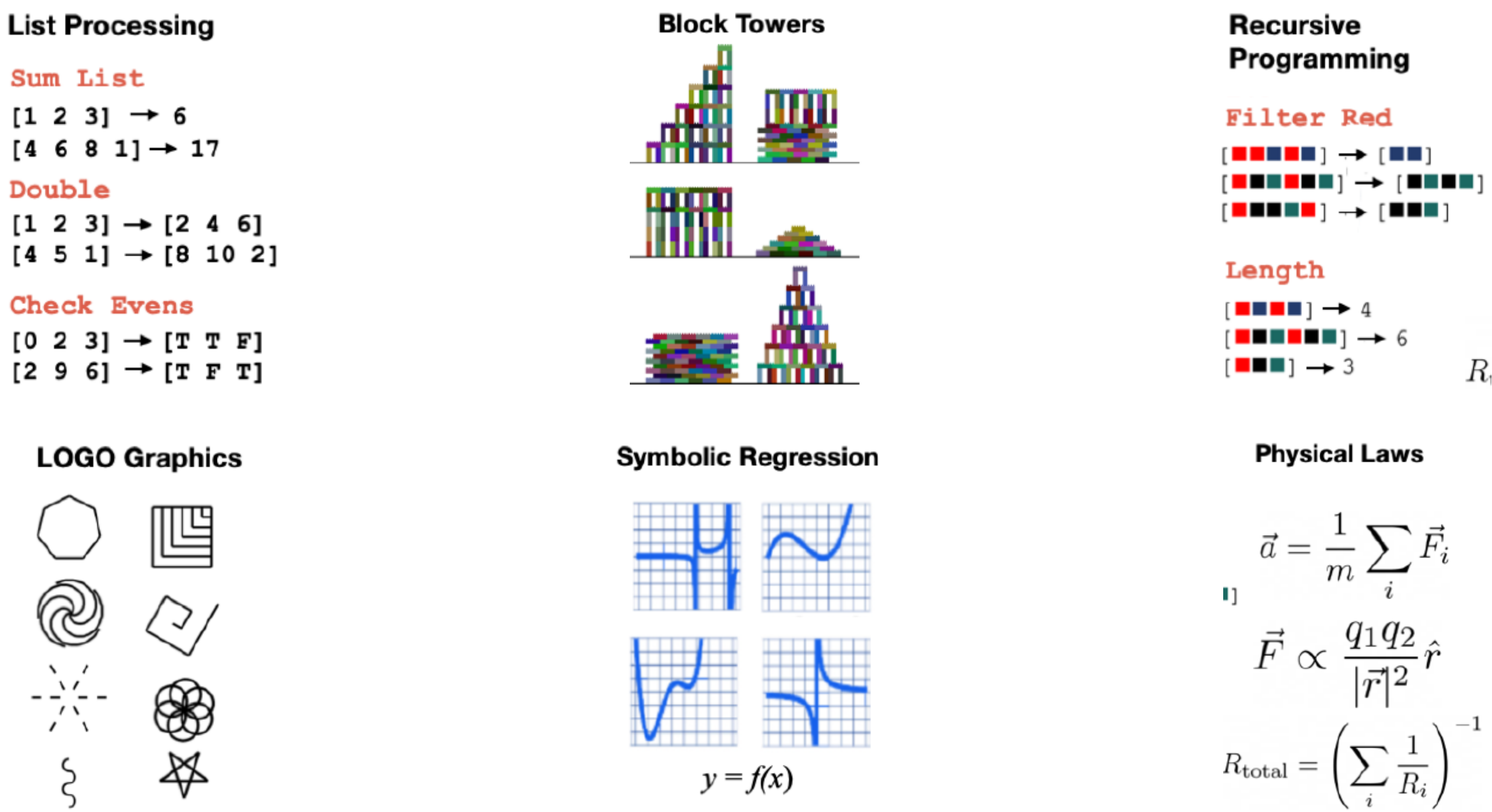
- Due to its compositional nature, representations of problems can be bootstrapped from earlier, simpler version of the scientific task to more and more complex settings



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

DreamCoder

Applications



DreamCoder

Applications

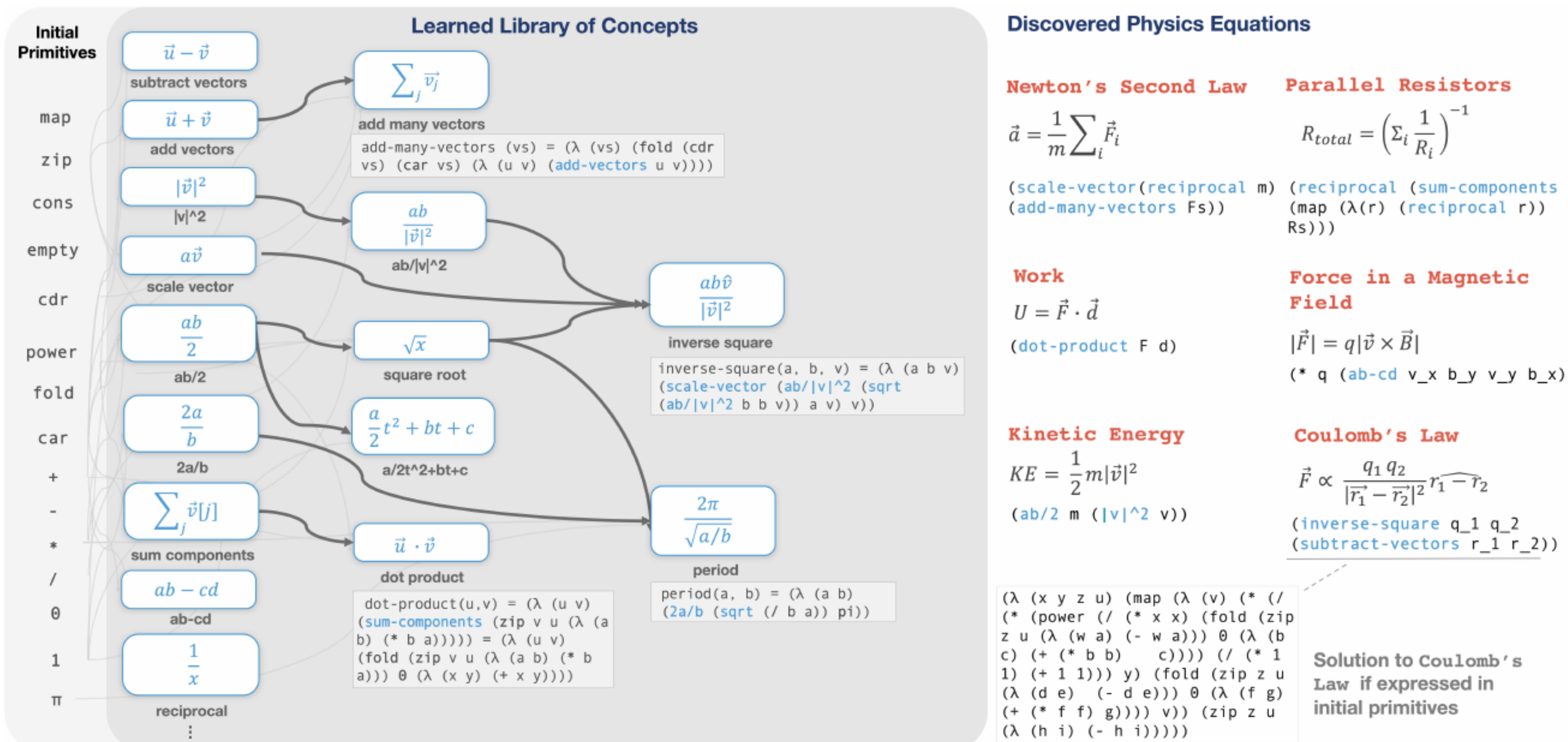


Figure: Learned library for physics equations.

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

DreamCoder

Applications

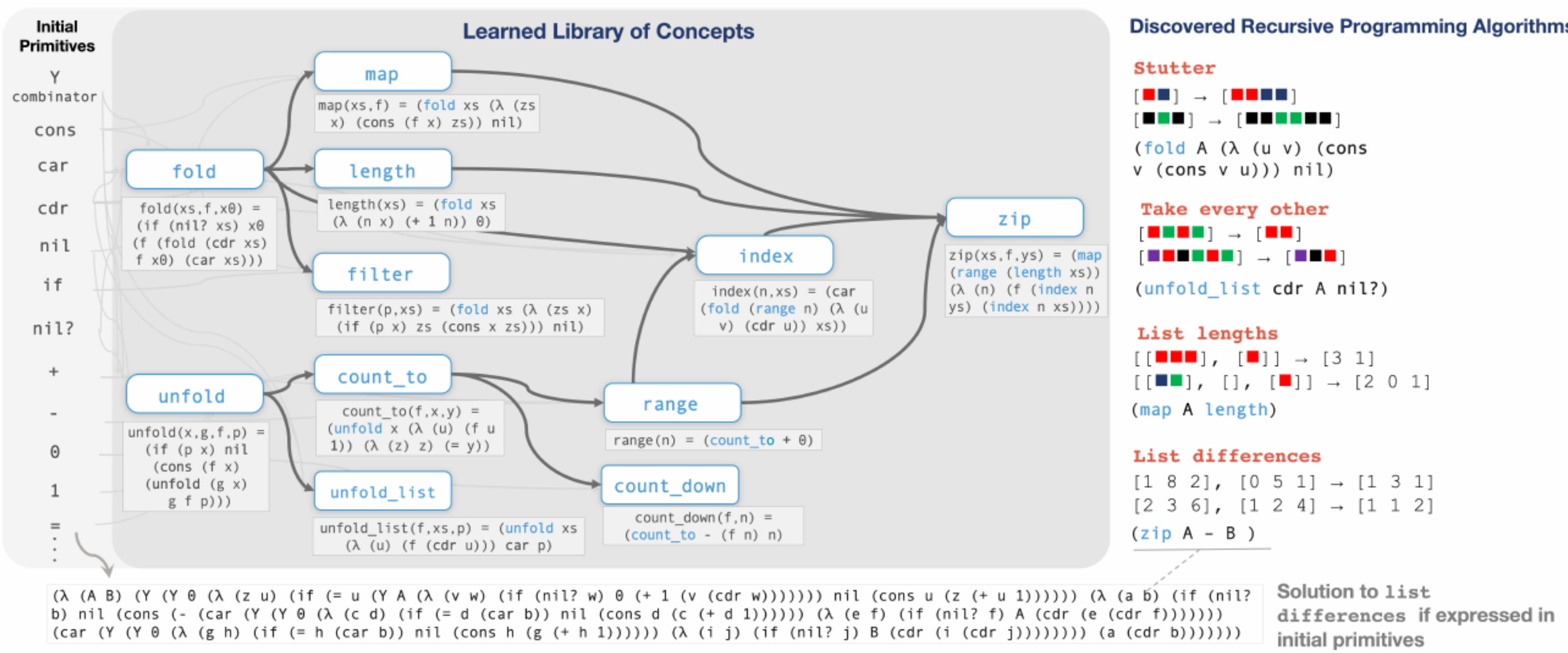


Figure: Learned library for recursive programming algorithm.

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

DreamCoder

Applications

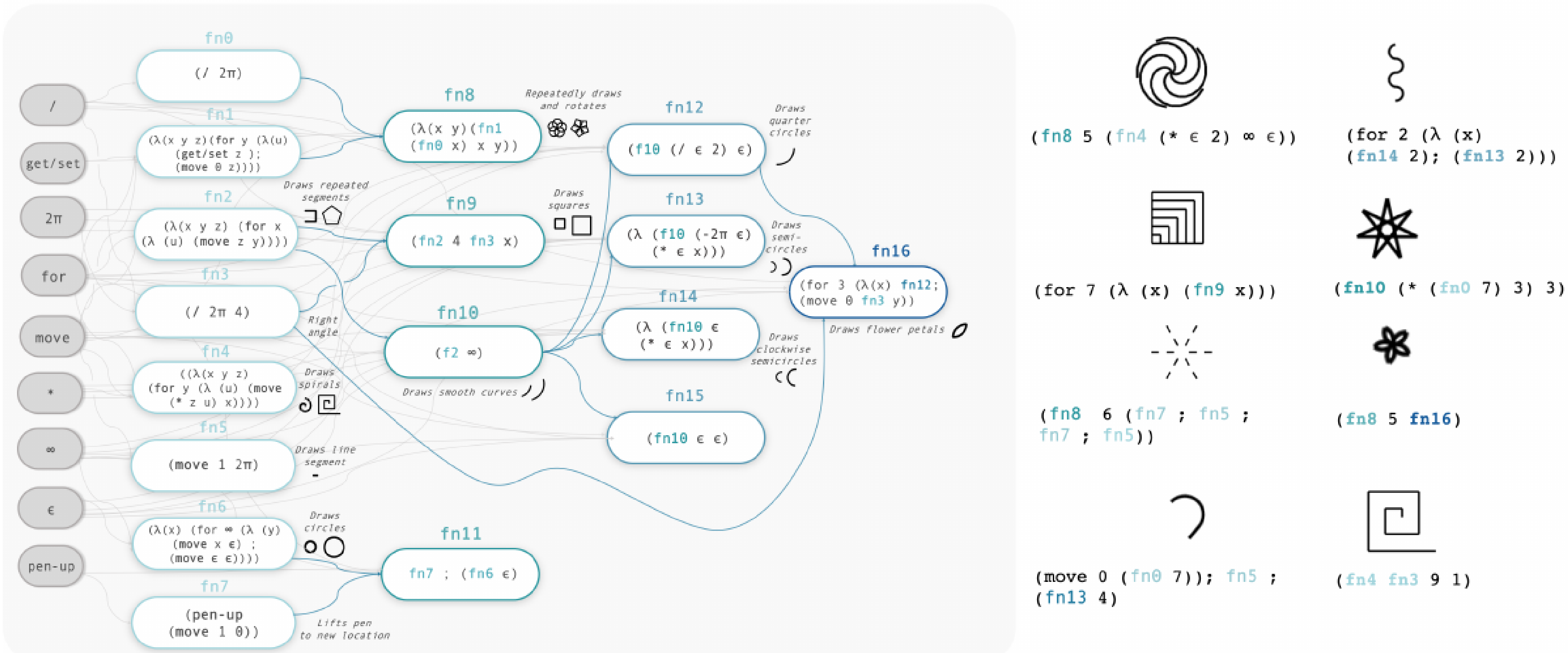


Figure: Learned library for LOGO graphics.

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

DreamCoder

Applications

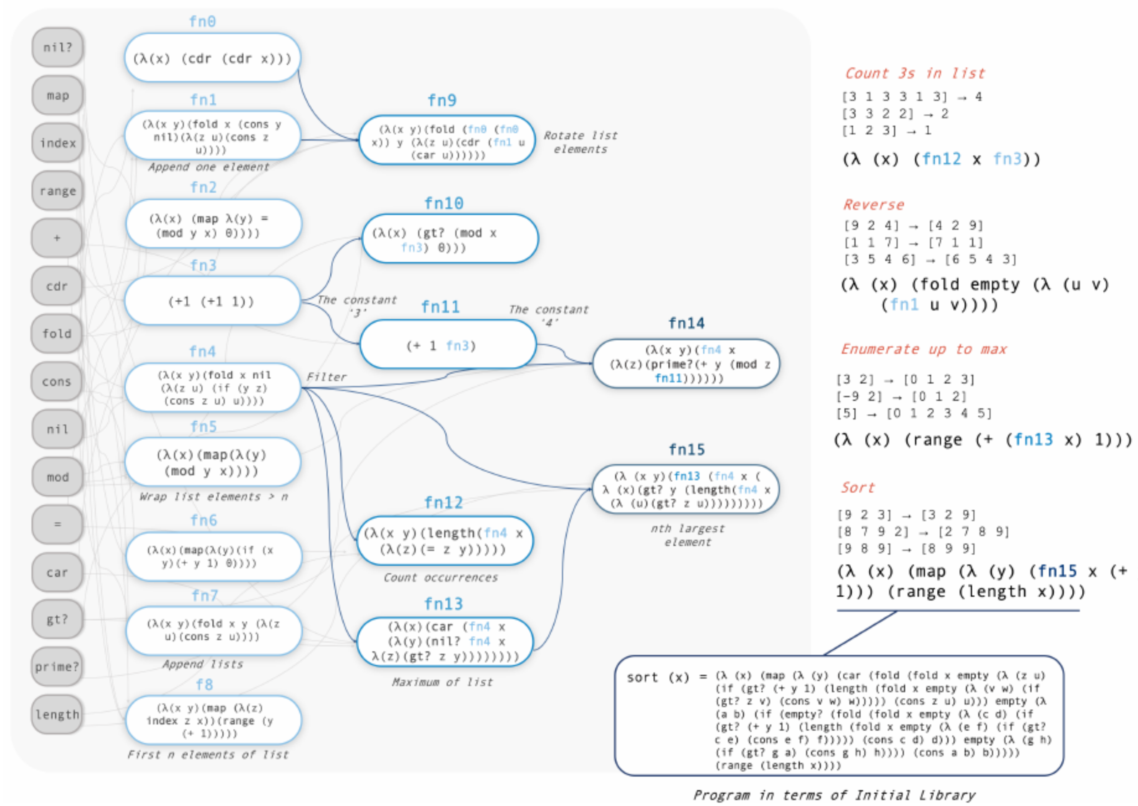


Figure: Learned library for list processing.

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

Why Do We Need Probabilistic Programming?

- Blub 1

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

Why Do We Need Probabilistic Programming?

- Blub 2

Why Do We Need Probabilistic Programming?

- Blub 3

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

Why Do We Need Probabilistic Programming?

- Blub 4

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

Why Do We Need Probabilistic Programming?

- Blub 5

Why Do We Need Probabilistic Programming?

- Blub 6

Why Do We Need Probabilistic Programming?

- Blub 7

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

Why Do We Need Probabilistic Programming?

- Blub 8

Outline