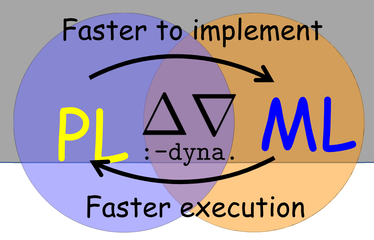
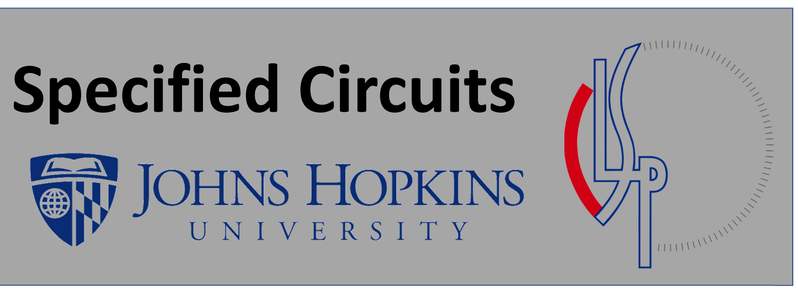
Treating Machine Learning Applications As Declaratively Specified Circuits

Jason Eisner, Nathaniel Wesley Filardo,
Matthew Francis-Landau, Tim Vieira





PL Helping ML: The Abstraction Challenge

COMPLEXITY IN ML SYSTEMS: The complexity of modern ML systems interferes with research, development, and education. It is a truism that an experiment that is casually suggested by a research advisor, and seems to be straightforward, may cost six months before an efficient and (hopefully) bug-free implementation is actually running.

ABSTRACTION IS KEY: Textbook algorithms may appear relatively simple because they can be written at an abstract level — e.g., as update rules on a small set of nicely notated mathematical quantities. However, applying such an algorithm to a real-world problem means instantiating those abstract quantities in terms of problem-specific data structures that must be efficiently and correctly manipulated.

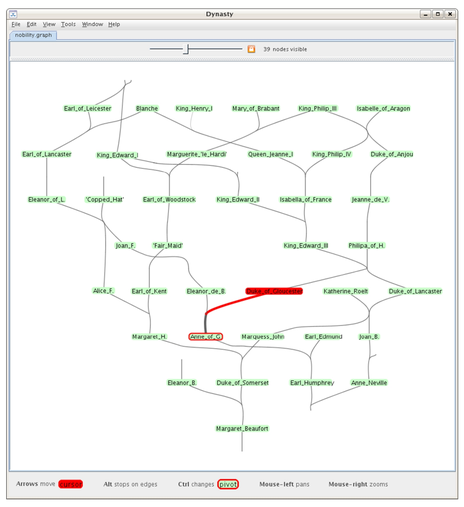
EFFICIENCY AND PORTABILITY: Worse, a typical applied ML system combines several of these techniques, so that many types of quantities are interacting. Not only does this increase complexity, but it creates a pressure to optimize across the abstraction boundaries in order to maintain speed. Choosing among possible optimizations is challenging and time-consuming, involving questions such as multiple-use data structures, time-space tradeoffs, loop orders, and use of specialized libraries and hardware. Implementing these optimizations further increases the complexity and risks bugs, particularly as the system evolves during research and development.

Dyna

Dyna is a high-level "circuit programming" language
Declarative semantics: fixpoint of a circuit

- can be infinitely wide, infinitely deep
- allows cycles
- dynamic data-dependent structure

- Prolog-like rules … very concise!
- Innovative typing and inheritance/modularity features



2-3 lines	Dijkstra's shortest-path algorithm
4 lines	Feed-forward neural network
11 lines	Bigram language model with Good-Turing backoff smoothing
6 lines	Arc-consistency constraint propagation
+6 lines	With backtracking search
+6 lines	With branch-and-bound
6 lines	Loopy belief propagation
3 lines	Probabilistic context-free parsing
+3 lines	Earley's algorithm
+7 lines	Conditional log-linear model of grammar weights (toy example)
+10 lines	Coarse-to-fine A* parsing
4 lines	Value computation in a Markov Decision Process
5 lines	Weighted edit distance
3 lines	Markov chain Monte Carlo (toy example)

Prolog-like Rules Define a Dynamic Computation Graph

```
a = b * c.    % equation
```

Reactive: **a** keeps up to date with **b** and **c**

```
b += x.
b += y.    equivalent to b = x + y. (almost)

c += z(1).
c += z(2).
c += z(3).
c += z("foo").
```

Shorthand

```
c += z(N).
```

Variable

```
a(I) = b(I) * c(I).   % elem-wise multiplication

a += b(I) * c(I).              % dot product (sparse)

a(I,K) += b(I,J) * c(J,K).   % matrix mult (sparse)
```

Real-World Examples

Shortest path:

```
distance(V) min= distance(U) + edge(U,V).
```

Probabilistic Context-free parsing

```
phrase(X,I,K) += word(W,I,K) * rewrite(X,W).
phrase(X,I,K) += phrase(Y,I,J) * phrase(Z,J,K)
                 * rewrite(X,Y,Z).
result = phrase("s", 0, sentence_length).
```

General Neural Network

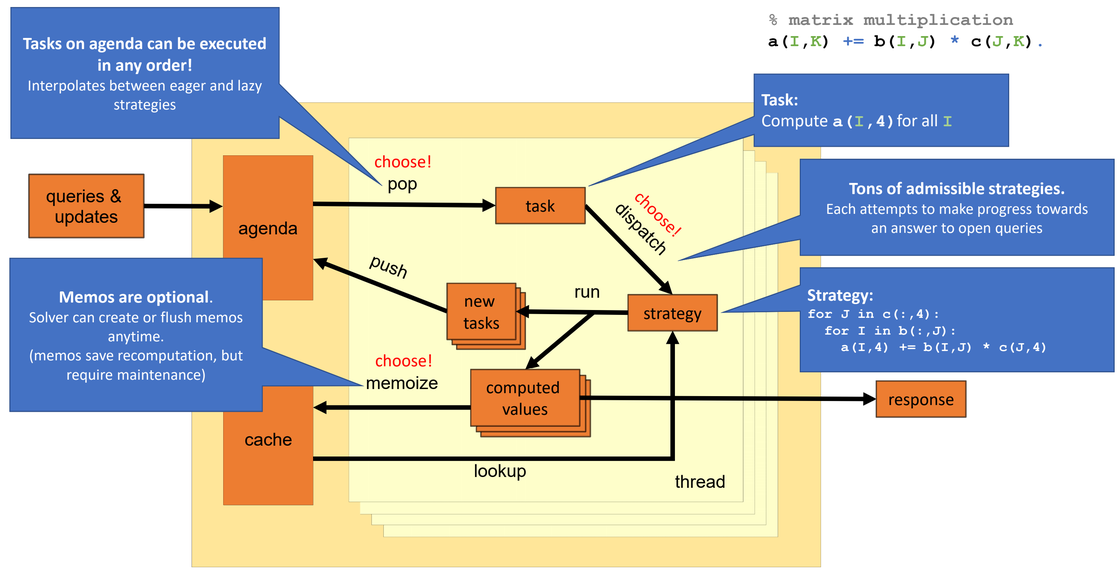
```
σ(X) = 1 / (1 + exp(-X)).
out(J) = σ(in(J)).
in(J) += out(I) * edge(I,J).
loss += (out(J) - target(J)) ** 2.
```

```
% Convolutional layer in the neural network
edge(input(X,Y),hidden(X+DX,Y+DY)) = convWeight(DX,DY).
convWeight(DX,DY) := random(*,-1,1) for DX:-1..1, DY:-1..1.
```

ML Helping PL: The Systems Challenge

- We have two protoypes of earlier versions of the language
  - Dyna 1 prototype (2005) was used for 17 dynamic programming research papers
  - Dyna 2 prototype (2013) was used for teaching an NLP course to linguists with no programming background.
  - Both were inefficient because they used too many one-size-fits-all strategies.
- Can we build an auto-tuning system? Or can you?
  - Fulfills the promise of declarative program specification!

Flexible Solver Architecture



Strategy options

Solver should systematize all the reasonable implementation tricks that programmers might use and make them work together correctly.

- **Parallelizing independent computations**
- **Ordering dependent computations**
  - Join strategies
  - Forward vs. backward chaining (update-driven vs. query-driven)
  - Dynamically identify unnecessary computation
    - Short-circuiting, branch-and-bound/A*, watched variables
- **Consolidating related work**
  - Static or dynamic batching (consolidating similar tasks, including GPU)
  - Inlining depth (consolidating caller-callee)
- **Storage**
  - Memoization policy; choose low-level data structures
- **Hardware**
  - Partitioning the problem across heterogeneous devices (GPUs, distributed)

Reinforcement Learning Objective

Total cost knob setting
Average latency on workload

Encourage earlier jobs to finish first

urgency

$$\rho(\pi) = \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \lambda_i \,\mathrm{latency}(i)\right]$$

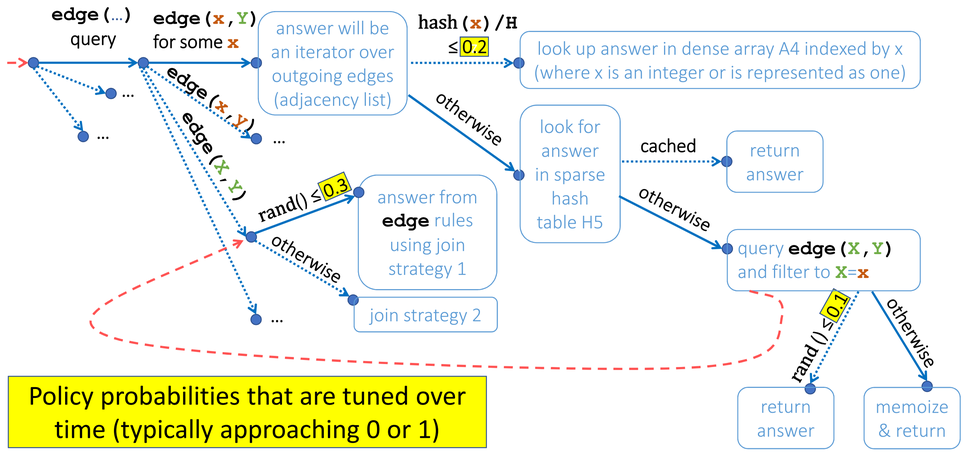
knob settings

$$= \mathbb{E}\left[\sum_{t=1}^{\infty} \mathrm{load}(t)\cdot(\mathrm{clock}(t{+}1) - \mathrm{clock}(t))\right]$$

$$= \sum_{i \in \mathcal{O}(t)} \gamma^i \lambda_i$$

Train by actor-critic with fast RLDT actor

Running the Policy



References and Further Reading

- Tim Vieira, Matthew Francis-Landau, Nathaniel Wesley Filardo, Farzad Khorasani, and Jason Eisner. 2017. Dyna: Toward a Self-Optimizing Declarative Language for Machine Learning Applications. MAPL Workshop.
- Jason Eisner and Nathaniel W. Filardo. 2011. Dyna: Extending Datalog For Modern AI. In Datalog Reloaded.
- Jason Eisner, Eric Goldlust, and Noah A Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In Proc. of EMNLP.
- Nathaniel Wesley Filardo and Jason Eisner. 2012. A Flexible Solver for Finite Arithmetic Circuits. In International Conference on Logic Programming LIPICS.
- Nathaniel Wesley Filardo. 2017. Dyna 2: Towards a General Weighted Logic Language. PhD Thesis.