The Geometry of Computation-Graph Abstraction

Koko Muroya
(University of Birmingham & RIMS, Kyoto University)

Steven W. T. Cheung
Dan R. Ghica
(University of Birmingham)
TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs

**Idealised TensorFlow**

*a calculus* with higher-order with computation graphs
Background: TensorFlow ([https://www.tensorflow.org/](https://www.tensorflow.org/))

an EDSL, for machine learning, with computation graphs

data-flow networks with *changeable* components

- constant
- operation
- “input”
- “variable” with a tentative value
Background: TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs

data-flow networks with *changeable* components

- constant
- operation
- “input”
- “variable” “cell” with a tentative value
Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs

equation: simple linear regression $f(x) = W \times x + b$

two cells with tentative values `W` and `b`

input `x`
Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs

by plugging in values to inputs

by updating values of cells using (input & desired output) data
Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs

- build
- use
  - by plugging in values to inputs
- train
  - cells automatically collected at run-time
  - by updating values of cells using (input & desired output) data
an EDSL, for machine learning, with computation graphs

\[
W = \text{tf.Variable}(...) \\
b = \text{tf.Variable}(...) \\
y = W \times x\_data + b
\]

\[
x\_data = ... \\
sess = \text{tf.Session()} \\
sess\_run(init) \\
y\_initial\_values = sess\_run(y)
\]

\[
x\_data = ... \\
y\_data = ... \\
sess = \text{tf.Session()} \\
sess\_run(init) \\
sess\_run(train)
\]
TensorFlow (https://www.tensorflow.org/) 
an EDSL, for machine learning, with computation graphs 
✓ automatic collection of cells 
✓ automatic differentiation 
● limited integration with a host language 
● imperative update of cells
TensorFlow (https://www.tensorflow.org/)
an EDSL, for machine learning, with computation graphs

✓ automatic collection of cells
✓ automatic differentiation
● limited integration with a host language
● imperative update of cells

Idealised TensorFlow

a calculus with higher-order computation graphs

✓ automatic collection of cells
✓ automatic differentiation
✓ full integration
➢ functional update of cells
Proposed Computation in *Idealised TensorFlow*

*a calculus with higher-order computation graphs*

**model**

- build

**output**

- use

(higher-order computation graph
- input as function argument
- cell as it is)
Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs

![Diagram](image)

- **build**
- **use**
- **train**

by function application
Proposed Computation in *Idealised TensorFlow*

*a calculus with higher-order computation graphs*

[Diagram showing the process of building, using, and possibly training a model.]
Proposed Computation in *Idealised TensorFlow*

A *calculus* with higher-order computation graphs

*Model* build

*Output* use

*Decoupled model*

*Graph abstraction*

*Train*

*Use*
Graph abstraction

run-time operation to automatically collect cells
Graph abstraction

run-time operation to automatically collect cells

via language construct ‘Abs’

1. *pure* function, representing parameterised model
2. *opaque* vector
   - size determined at run-time
   - order of elements obscured

cells may be contributed by free variables
Functional, not imperative, training
Functional, not imperative, training

New values of cells computed, yielding (again) an opaque vector

... with parameterised model unchanged

Decoupled model

Train
Proposed Computation in *Idealised TensorFlow*

A calculus with higher-order computation graphs

- **Model**
  - Use
  - Build

- **Output**
  - Use

**Decoupled Model**

- **Graph Abstraction**
  - Use
  - Train
Proposed Programming in *Idealised TensorFlow*

- built a model using cells

\[
\text{let } (\text{model}', p) = \text{abs} (\lambda x. \{1\} \times x + \{0\})
\]

\[
\text{let } y = \text{model}' (\text{optimiser data } p \text{ model}' \text{ loss_function}) 7
\]
Proposed Programming in *Idealised TensorFlow*

- built a model using **cells**
- get a *decoupled* model using **graph abstraction**

\[
\text{let } (\text{model}', p) = \text{abs} (\lambda x.\{1\} \times x + \{0\})
\]

\[
\text{let } y = \text{model}' \ (\text{optimiser data p model'} \ \text{loss\_function})
\]
Proposed Programming in *Idealised TensorFlow*

- built a model using **cells**
- get a *decoupled* model using **graph abstraction**
- train a model, by calculating new values of cells

\[
\begin{align*}
\text{let } (\text{model}', p) &= \text{abs } (\lambda x. \{1\} \times x + \{0\}) \\
\text{let } y &= \text{model}' (\text{optimiser data } p \text{ model' loss_function}) 
\end{align*}
\]
Proposed Programming in *Idealised TensorFlow*

- built a model using *cells*
- get a *decoupled* model using *graph abstraction*
- train a model, by calculating new values of cells
- use the trained model, yielding output

```
let (model', p) = abs (λx. {1} × x + {0})
```

```
let y = model' (optimiser data p model' loss_function) 7
```
Proposed Programming in *Idealised TensorFlow*

simply-typed \( \lambda \)-calculus, extended by:

- cells … containing field elements
- graph abstraction `\texttt{Abs}` … in implication form `\texttt{A}`
- opaque vectors … with operations

\[
\begin{align*}
A & \vdash \Gamma, T \\
\frac{A \vdash \Gamma, x : T, \Delta \vdash x : T}{A \vdash \Gamma, x : T, \Delta} & \frac{A \vdash \Gamma, x : T' \vdash p \vdash t : T}{A \vdash \Gamma, x : T' \vdash p \vdash \lambda x^{T'}. t : T' \rightarrow T} \\
P \in \mathbb{F} & \frac{A \vdash \Gamma \vdash p \vdash t : T' \rightarrow T}{A \vdash \Gamma, \overline{p}, \overline{q} \vdash tu : T} \\
A \vdash \Gamma \vdash p \vdash t_1 : T_1 & \frac{A \vdash \Gamma \vdash q \vdash t_2 : T_2}{A \vdash \Gamma, p \vdash t_1 \# t_2 : T} \\
P \in \mathbb{F} & \frac{A, a \vdash \Gamma, f : V_a \rightarrow T', x : V_a \vdash p \vdash t : T}{A \vdash \Gamma, T', T} \\
\frac{A \vdash \Gamma \vdash p \vdash A^{T'}_a (f, x). t : T' \rightarrow T}{A \vdash \Gamma \vdash \{p\} : \mathbb{F}}
\end{align*}
\]
Proposed Semantics of *Idealised TensorFlow*

a “dynamic Geometry of Interaction machine” [M. & Ghica ‘17], a *token-guided graph-rewriting abstract machine*

- *redex searching* by moving the “token”
- *rewriting* by replacing a sub-graph
- *observing value* by looking at data carried by the “token”

implementing call-by-value evaluation,
Proposed Semantics of *Idealised TensorFlow*

a “dynamic Geometry of Interaction machine” [M. & Ghica ‘17], a *token-guided graph-rewriting abstract machine*

- *redex searching* by moving the “token”
- *rewriting* by replacing a sub-graph
- *observing* value by looking at data carried by the “token”

implementing call-by-value evaluation,
Proposed Semantics of *Idealised TensorFlow* a “dynamic Geometry of Interaction machine” [M. & Ghica ‘17], a *token-guided graph-rewriting abstract machine*

- *redex searching* by moving the “token”
- *rewriting* by replacing a sub-graph
- *observing value* by looking at data carried by the “token”

implementing call-by-value evaluation,

... to prove

- type-soundness
- program equivalence
  - restricted call-by-value beta-law
  - garbage collection
TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs

✓ automatic collection of cells
✓ automatic differentiation
• limited integration with a host language
• imperative update of cells

Idealised TensorFlow

a calculus with higher-order computation graphs

✓ automatic collection of cells
✓ automatic differentiation
✓ full integration
➢ functional update of cells
TensorFlow (https://www.tensorflow.org/) is an EDSL, for machine learning, with computation graphs:

- ✓ automatic collection of cells
- ✓ automatic differentiation
- ● limited integration with a host language
- ● imperative update of cells

Idealised TensorFlow offers a calculus with higher-order computation graphs:

- ✓ automatic collection of cells
- ✓ automatic differentiation
- ✓ full integration
- ➢ functional update of cells

An online visualiser (https://cwtsteven.github.io/GoI-TF-Visualiser/CBV-with-CBN-embedding/index.html) is available to see Idealised TensorFlow in full action!
TensorFlow (https://www.tensorflow.org/) is an EDSL, for machine learning, with computation graphs:

- ✓ automatic collection of cells
- ✓ automatic differentiation
- ● limited integration with a host language
- ● imperative update of cells

Idealised TensorFlow is a calculus with higher-order computation graphs:

- ✓ automatic collection of cells
- ✓ automatic differentiation
- ✓ full integration
- ➢ functional update of cells

DecML, implementation(s) as an OCaml PPX extension:

https://github.com/DecML/decml-ppx
https://github.com/reubenrowe/ocaml-decml

[Cheung, Darvariu, Ghica, M. & Rowe, FLOPS ‘18]
Conclusion

*Calculi* with higher-order computation graphs?

- *token-guided graph rewriting* (“dynamic Geometry of Interaction"),
  powerful & flexible operational semantics

✓ automatic cell collection in TensorFlow

- language support for the bureaucracy of model parameters in machine learning
- automatic differentiation in TensorFlow
- dependency between cells à la self-adjusting computation