The Geometry of Computation-Graph Abstraction

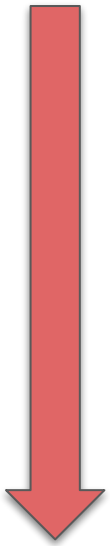
Koko Muroya
(University of Birmingham
& RIMS, Kyoto University)

Steven W. T. Cheung
Dan R. Ghica
(University of Birmingham)

LICS 2018 (Oxford), 12 July 2018

TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs



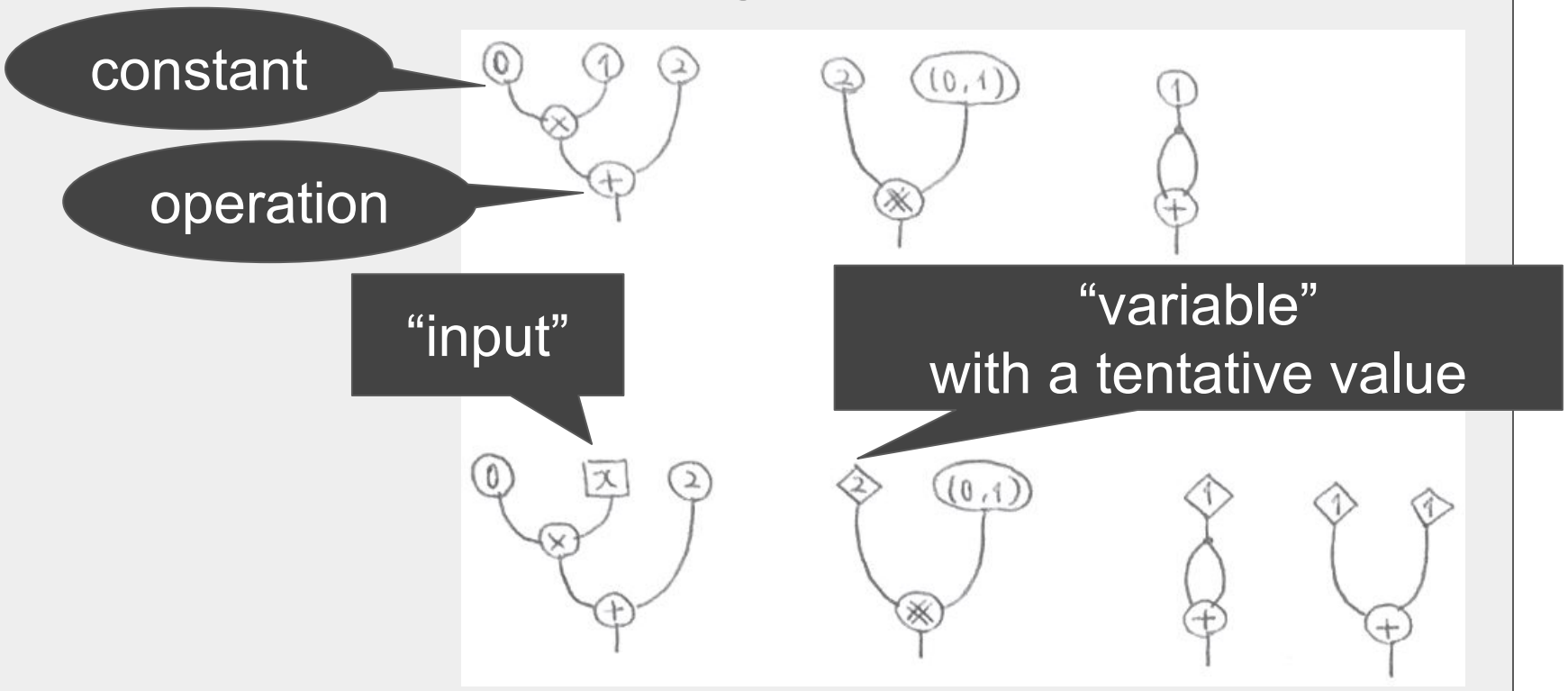
Idealised TensorFlow

a *calculus* with higher-order with computation graphs

Background: TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs

data-flow networks with *changeable* components



Background: TensorFlow (https://www.tensorflow.org/)

an EDSL, for machine learning, with computation graphs

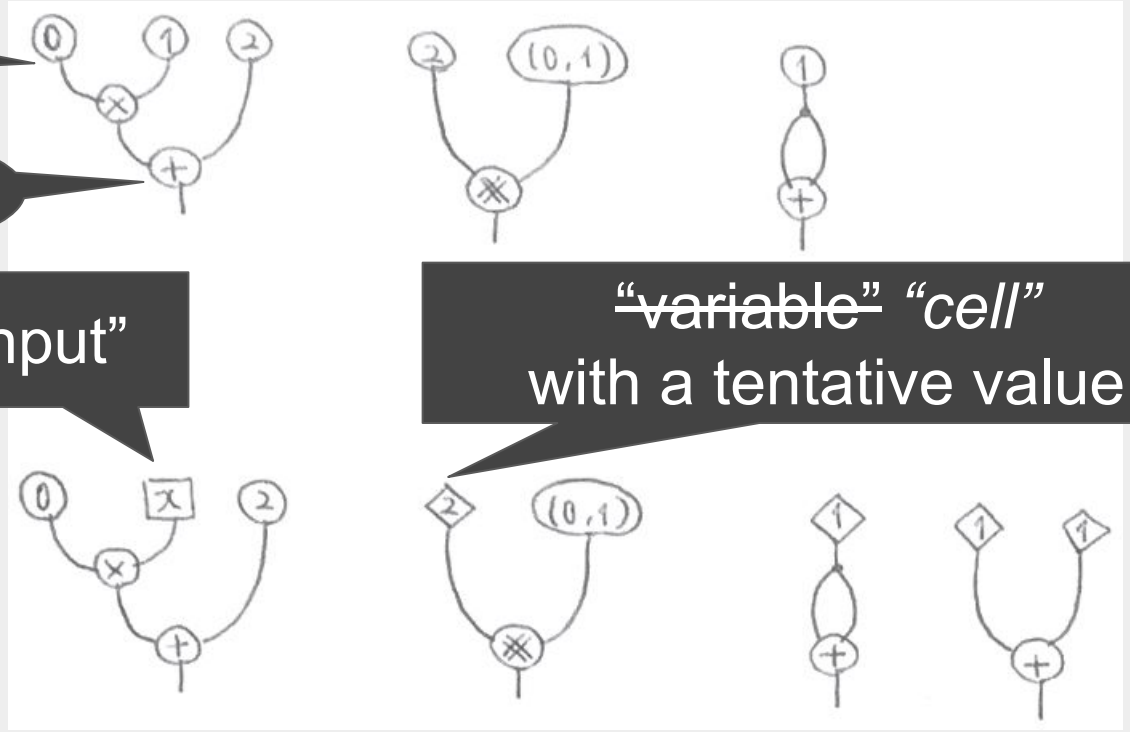
data-flow networks with *changeable* components

constant

operation

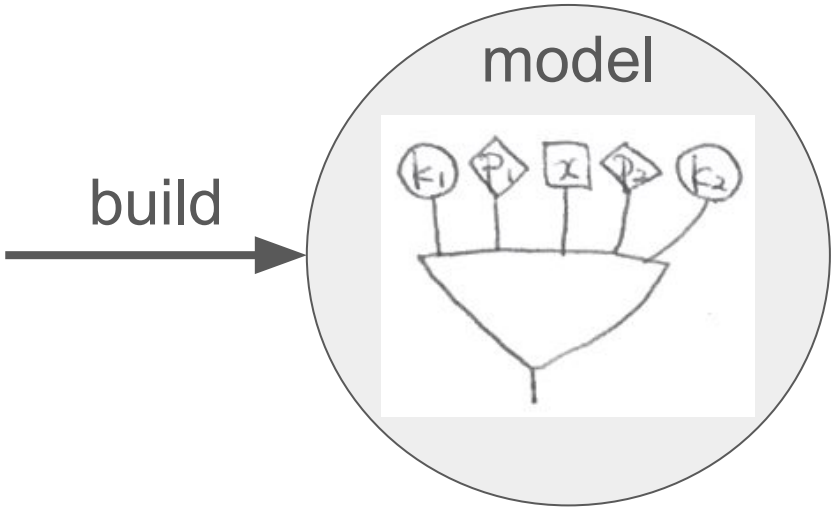
"input"

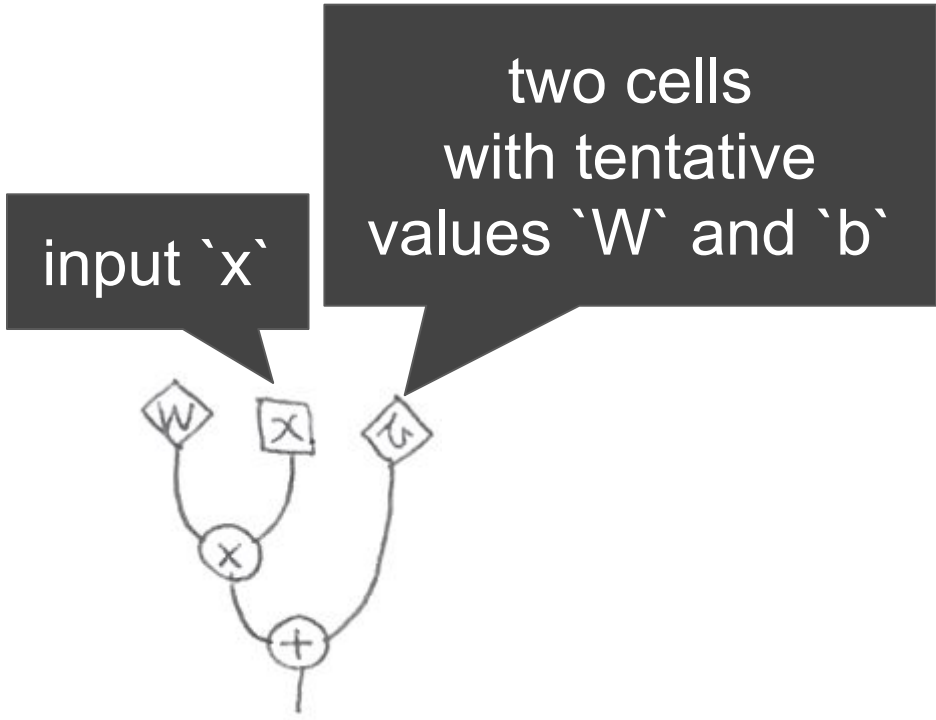
~~"variable"~~ *"cell"* with a tentative value



Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs

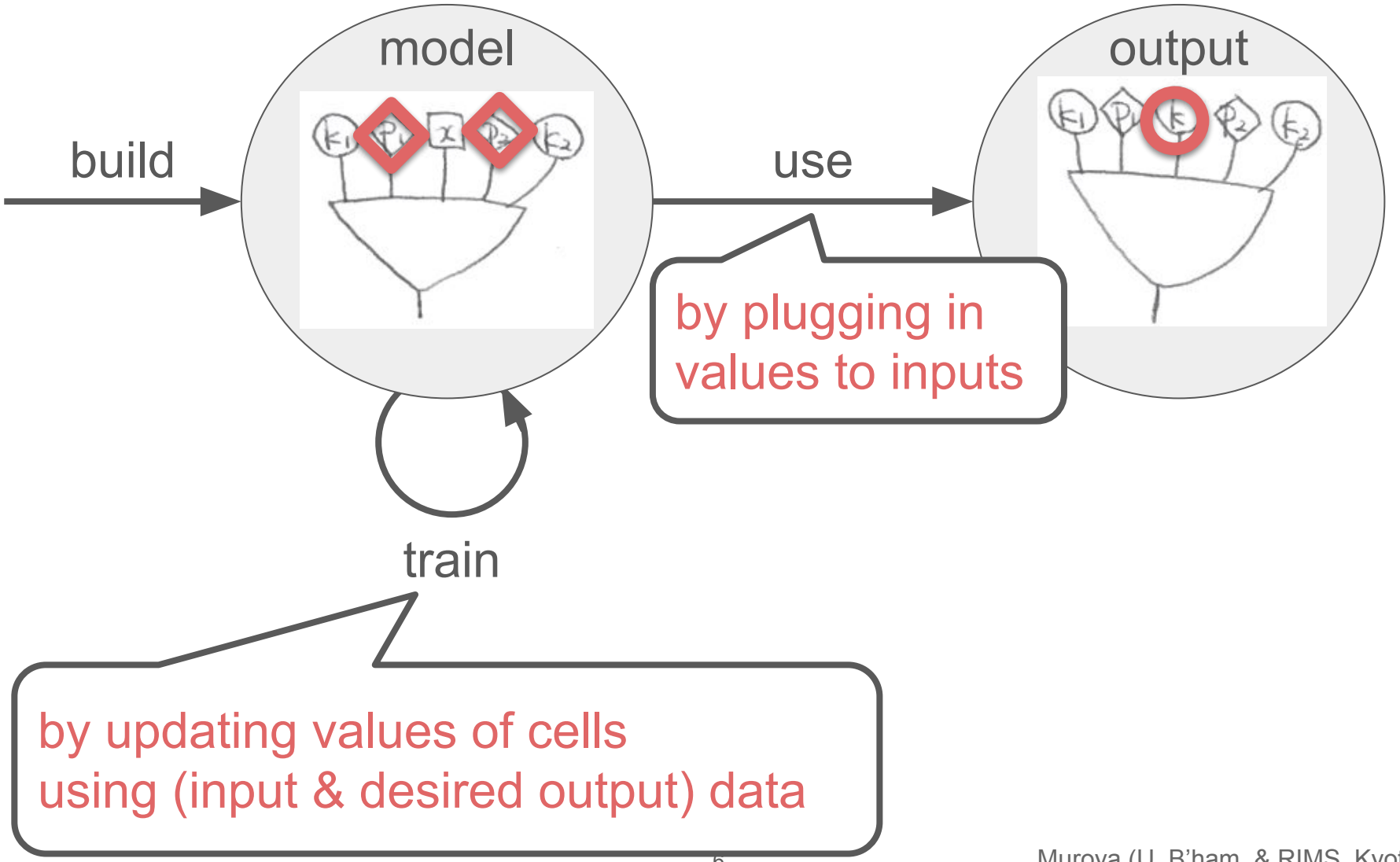




example: simple linear regression f(x) = W * x + b

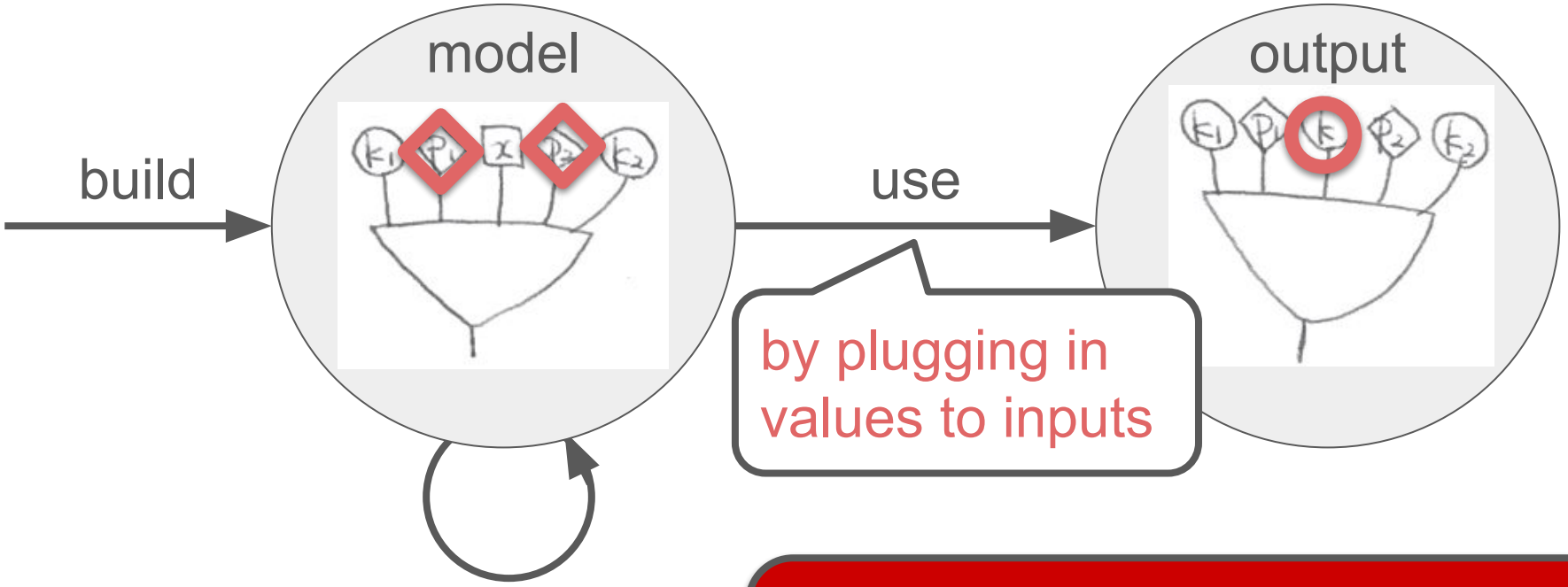
Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs



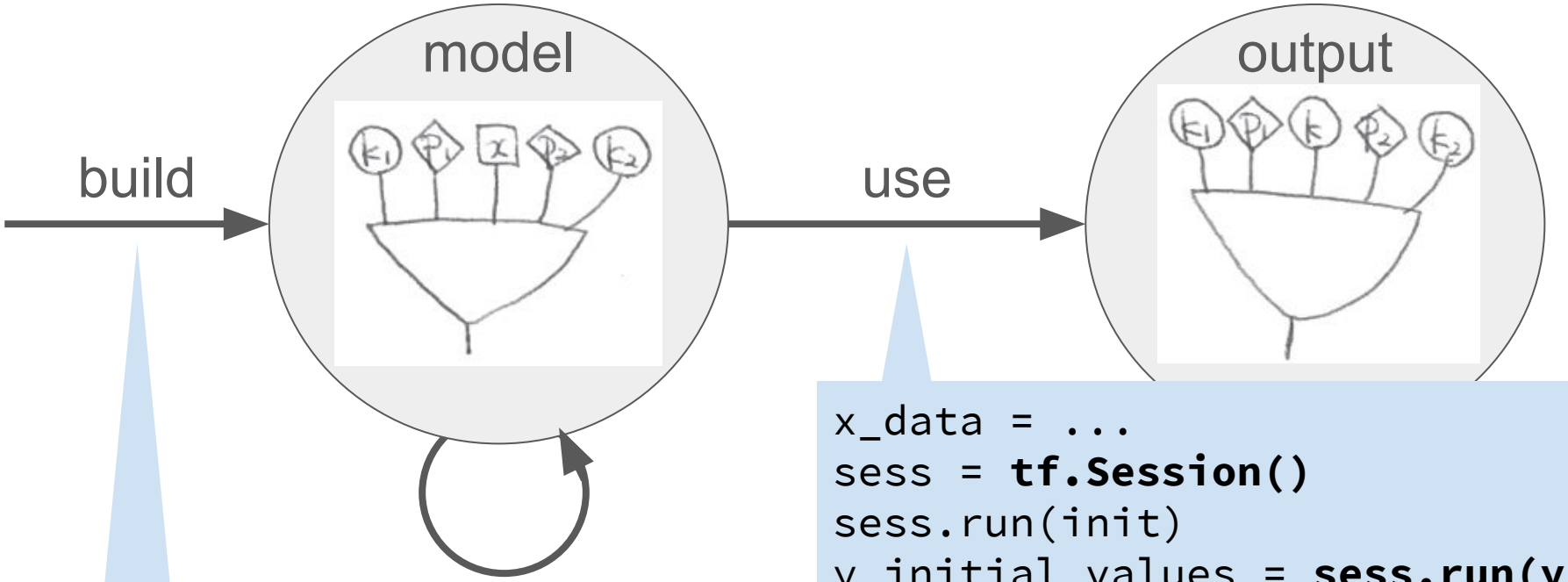
Background: Computation in TensorFlow

an EDSL, for machine learning, with computation graphs



Background: Programming in TensorFlow

an EDSL, for machine learning, with computation graphs




```
W = tf.Variable(...)
b = tf.Variable(...)
y = W * x_data + b
```

```
x_data = ...
sess = tf.Session()
sess.run(init)
y_initial_values = sess.run(y)
```

```
x_data = ...
y_data = ...
sess = 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

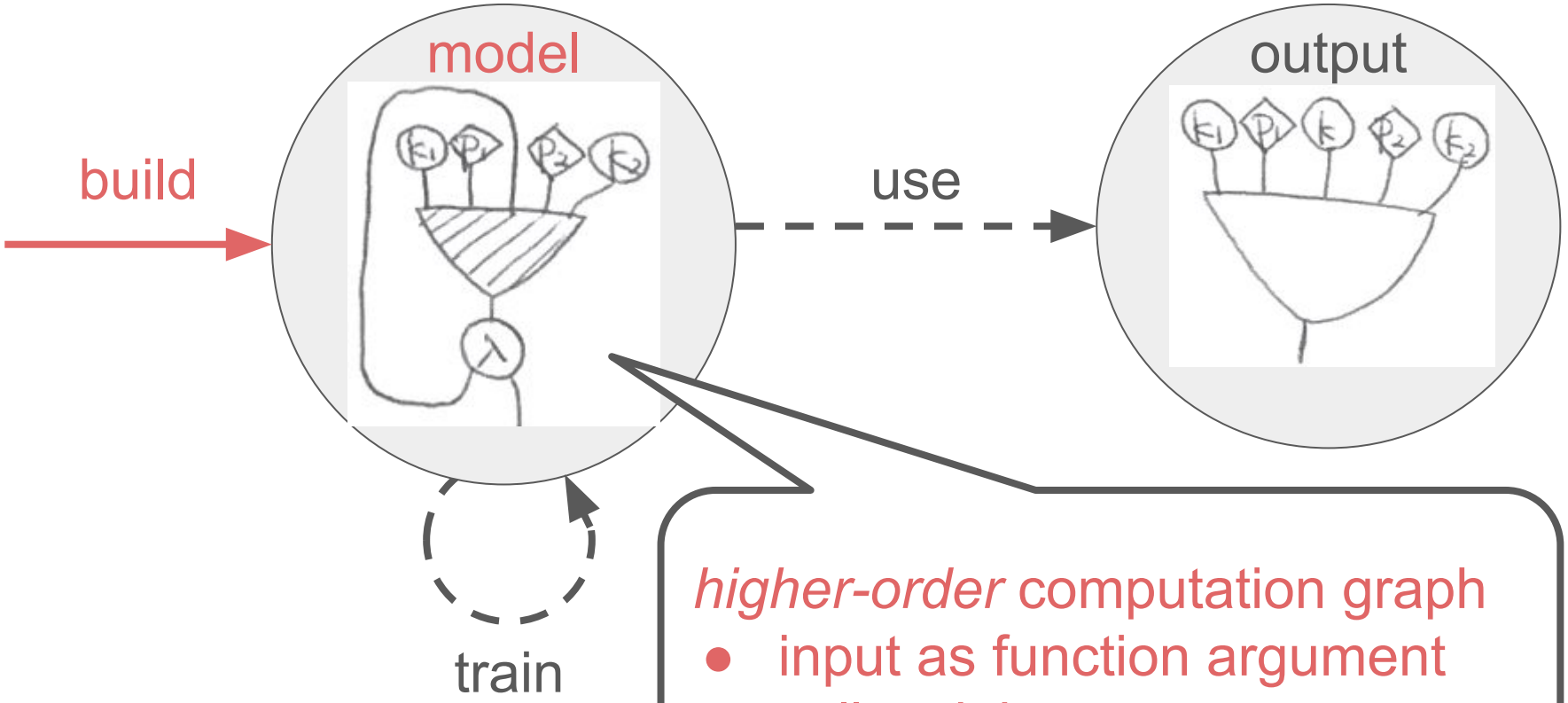
Idealised TensorFlow

a *calculus* with higher-order computation graphs

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

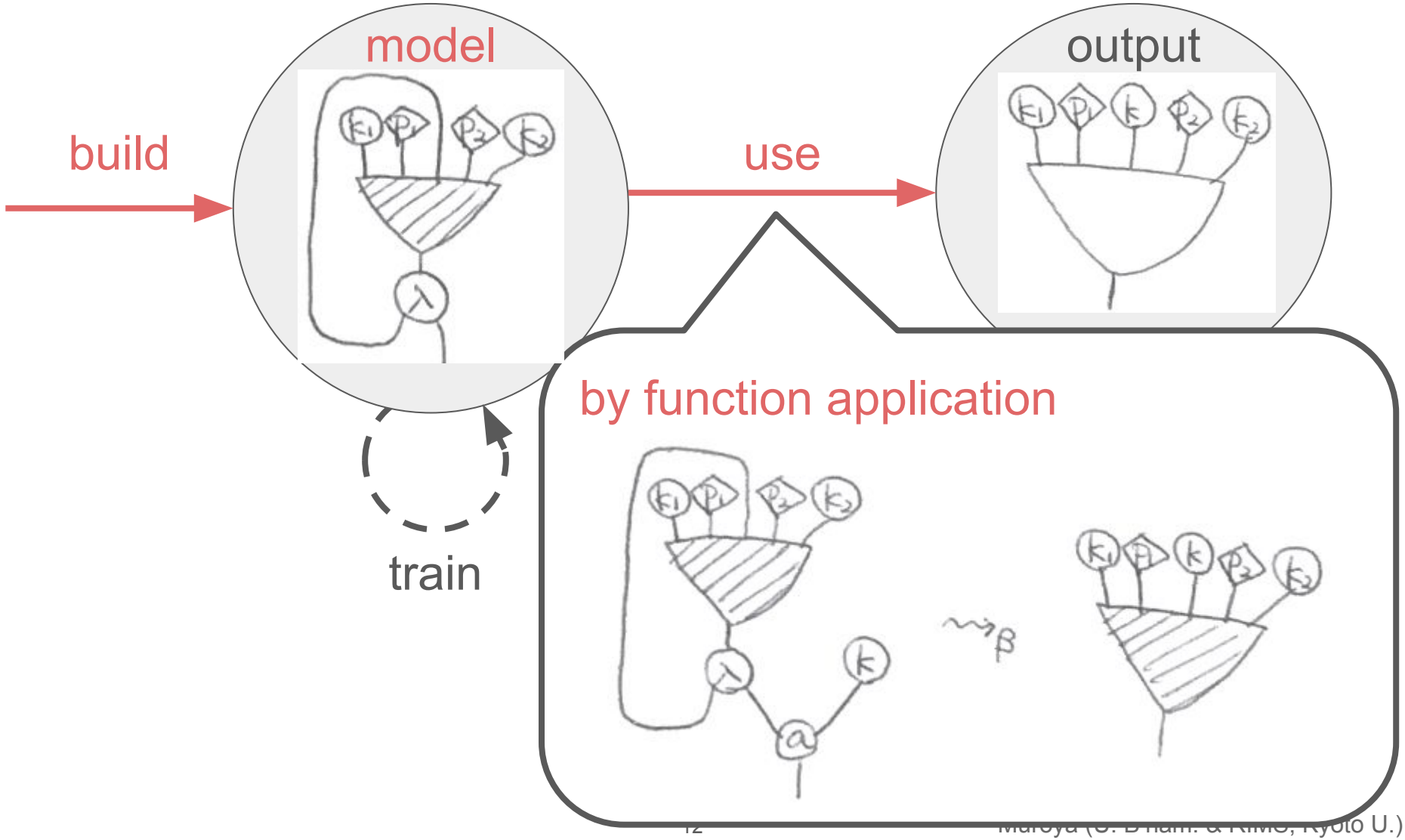
Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs



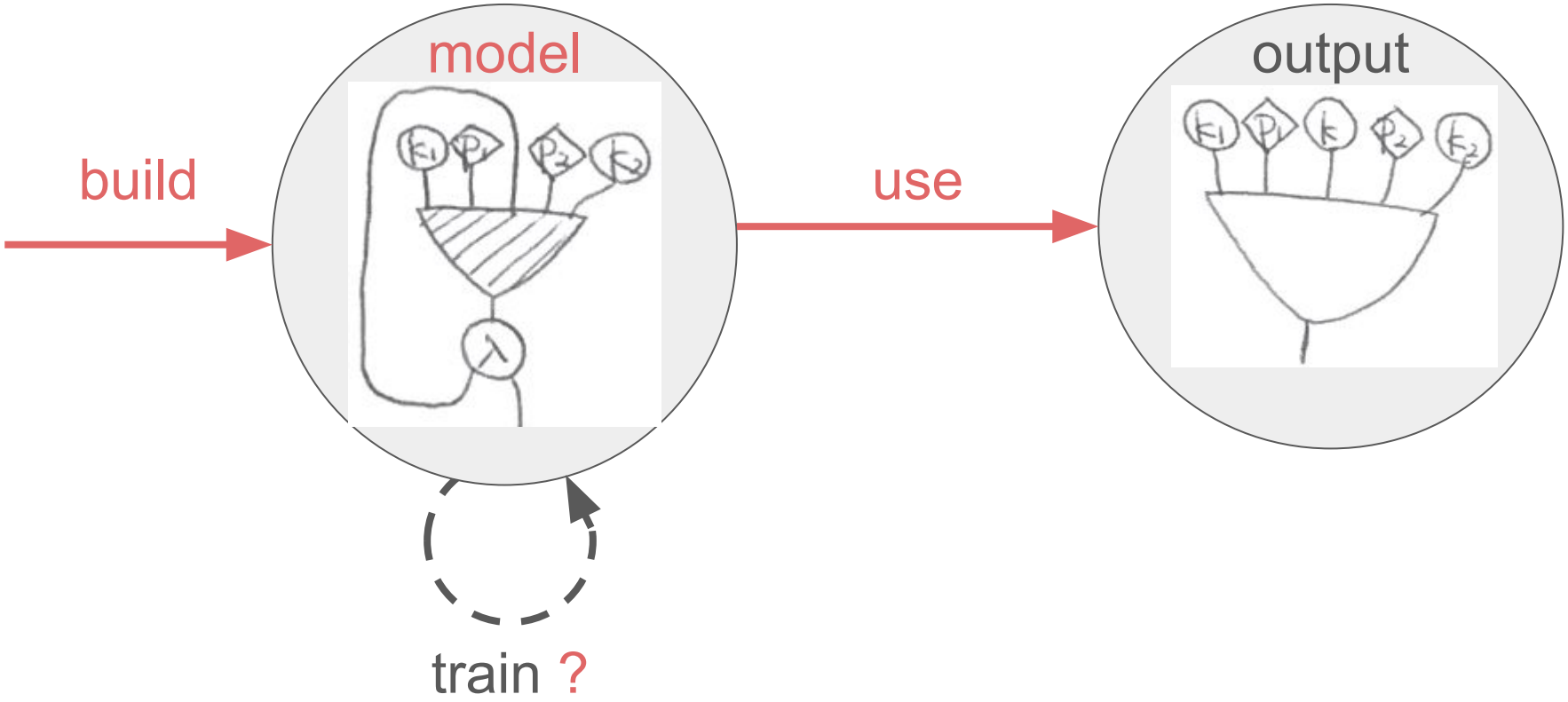
Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs



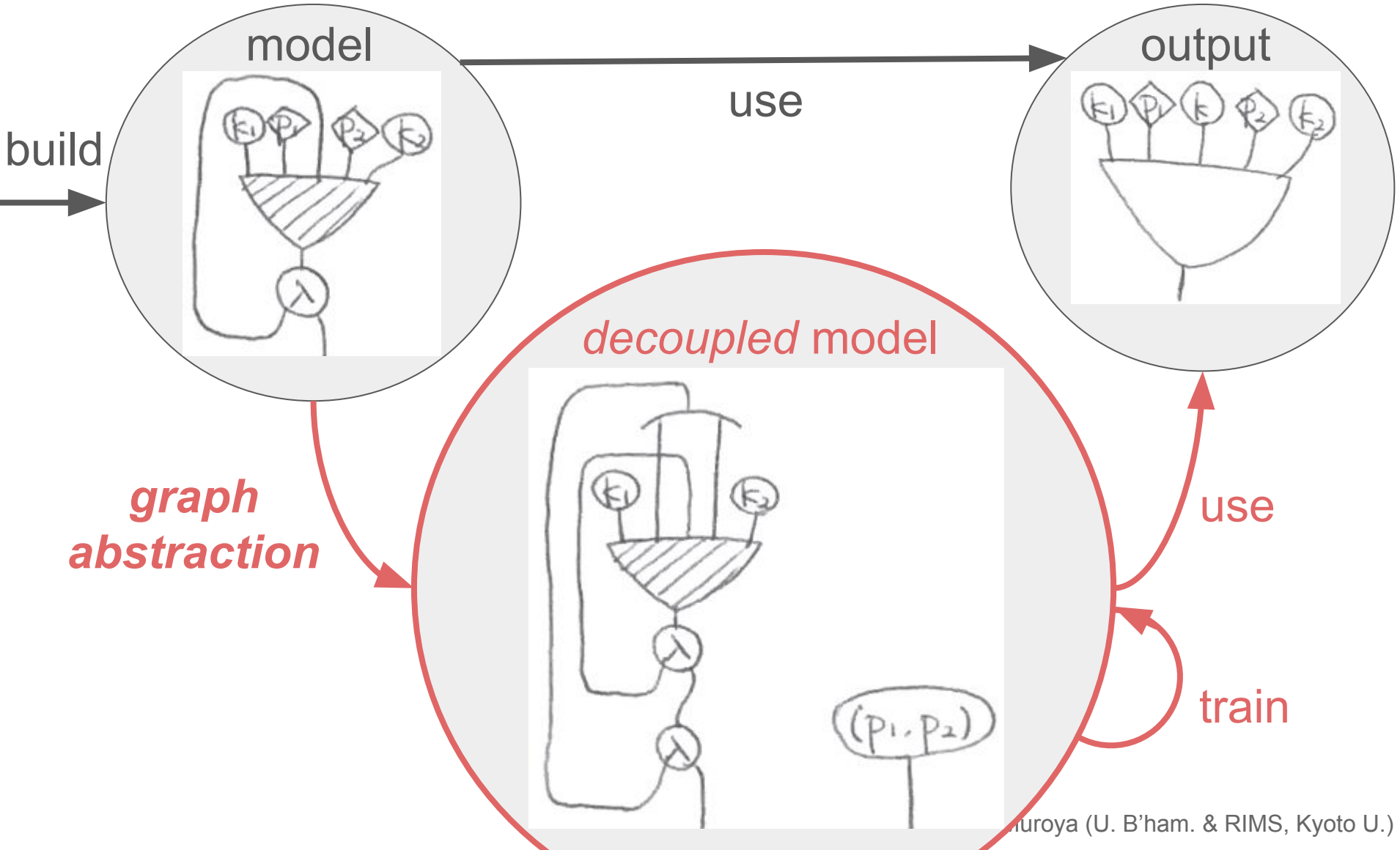
Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs



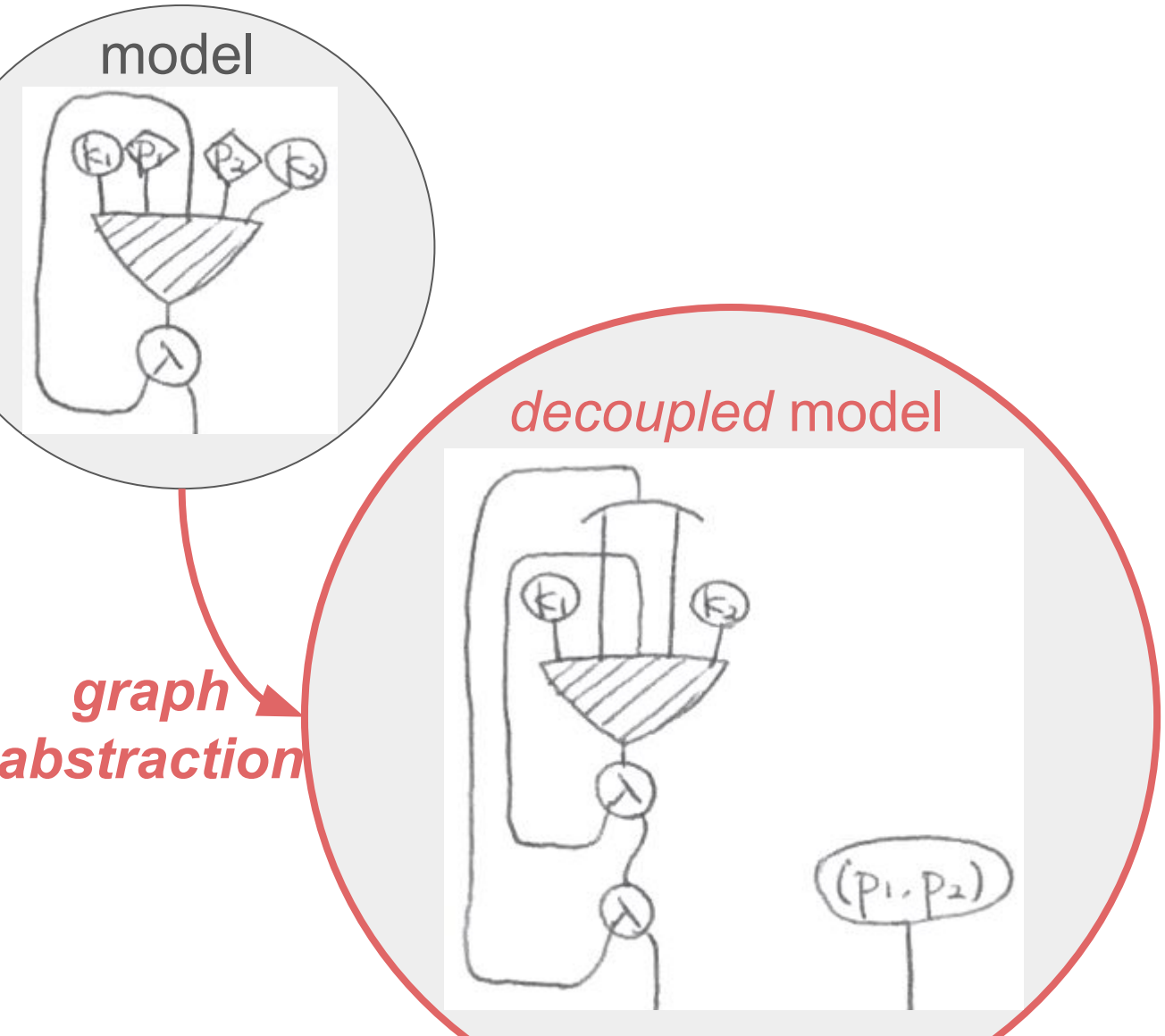
Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs



Graph abstraction

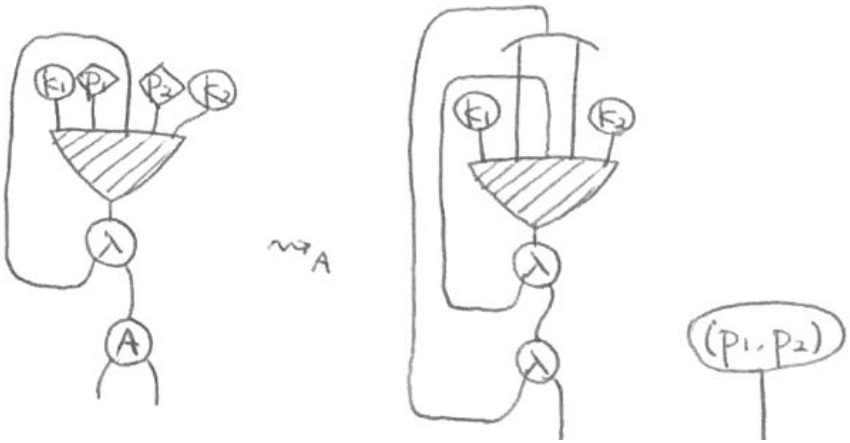
run-time operation to automatically collect cells



Graph abstraction

run-time operation to automatically collect cells

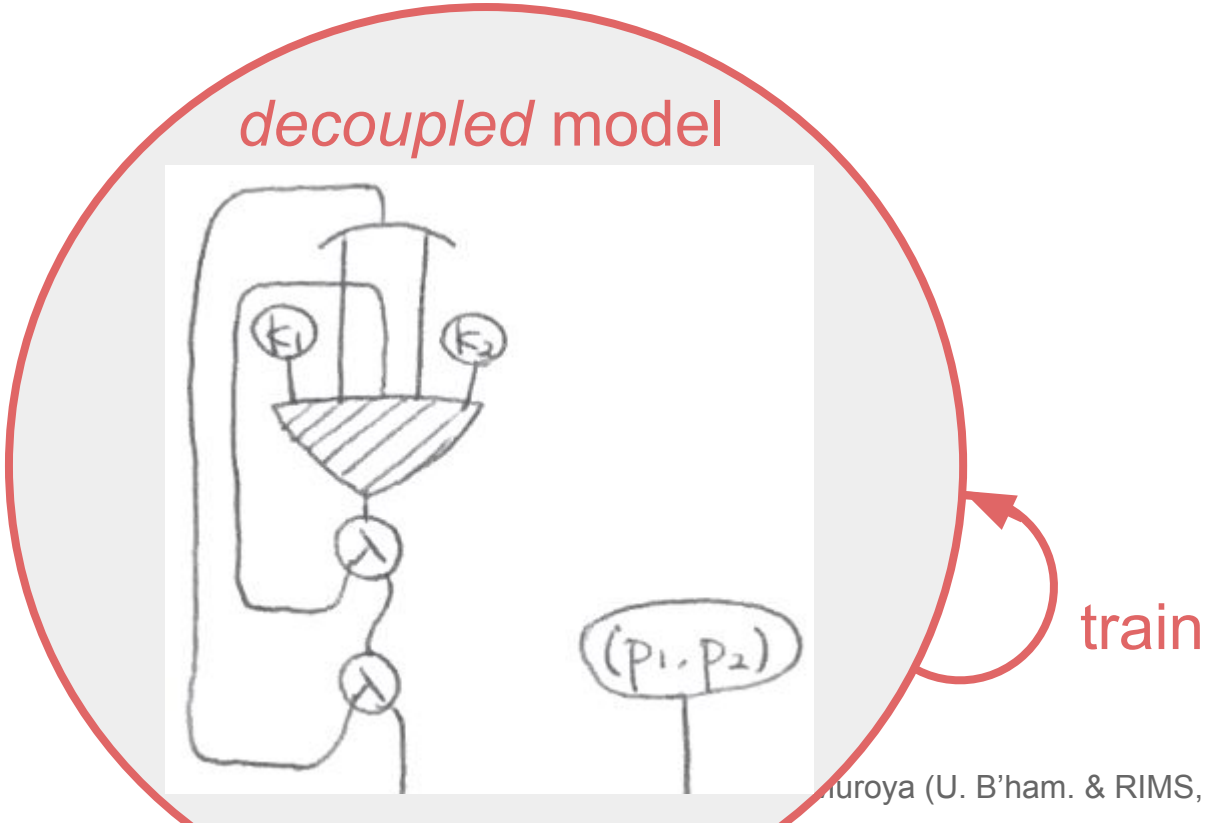
via language construct `Abs`



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

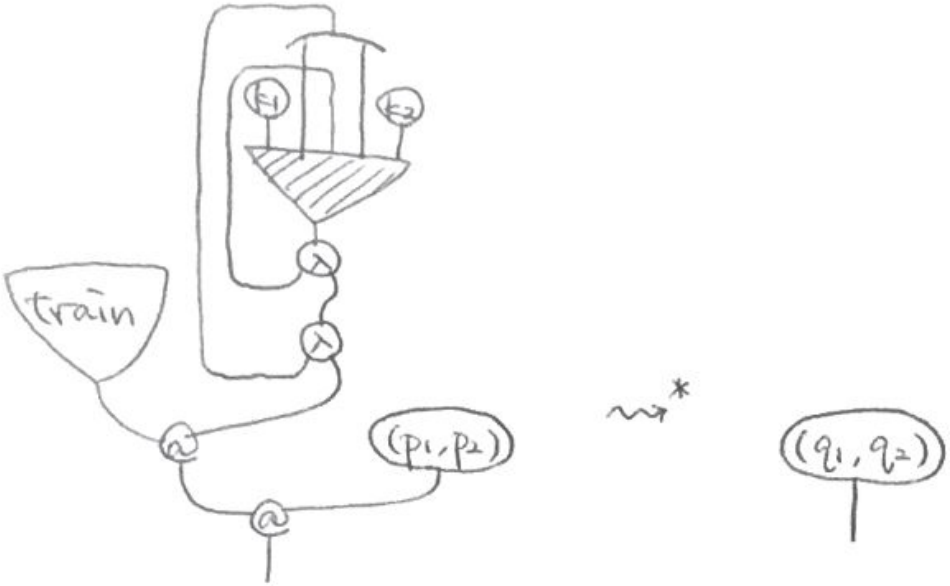
cells may be contributed by free variables

Functional, not imperative, training



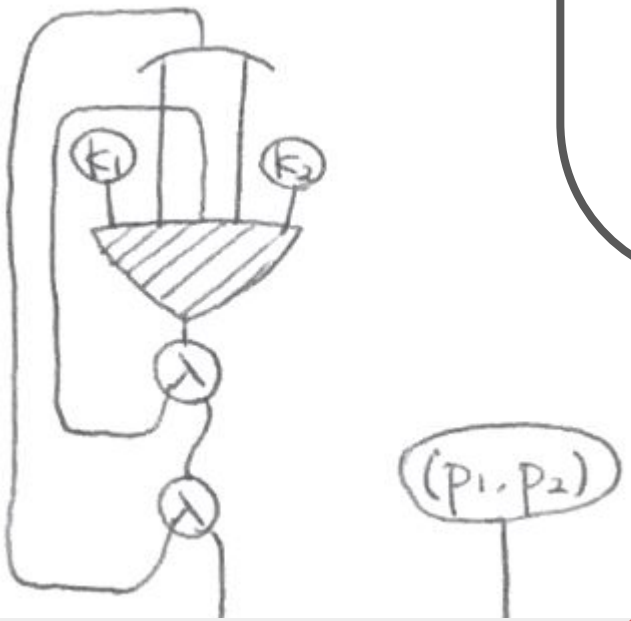
Functional, not imperative, training

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



… with parameterised model unchanged

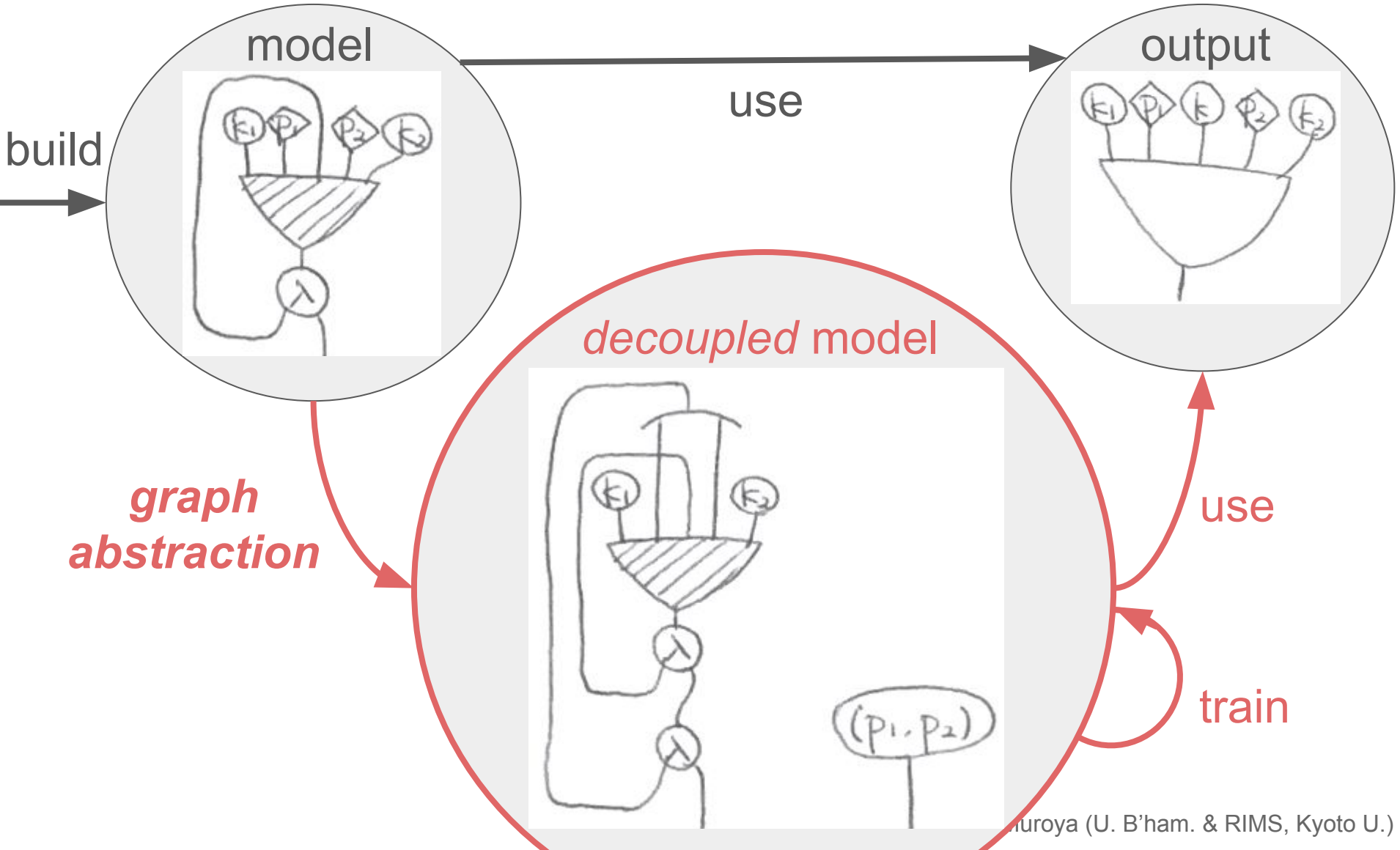
decoupled model



train

Proposed Computation in *Idealised TensorFlow*

a *calculus* with higher-order computation graphs



Proposed Programming in *Idealised TensorFlow*

- built a model using cells

$$let\ (model', p) = \mathrm{abs}\ (\lambda x.\{1\} \times x + \{0\})$$

$$let\ y = model'\ (optimiser\ data\ p\ model'\ loss\_function)\ 7$$

Proposed Programming in *Idealised TensorFlow*

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

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

$$let\ y = model'\ (optimiser\ data\ p\ model'\ loss\_function)\ 7$$

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

$$let\ (model', p) = \mathrm{abs}\ (\lambda x.\{1\} \times x + \{0\})$$

$$let\ y = model'\ (optimiser\ data\ p\ model'\ loss\_function)\ 7$$

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

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

$$let\ y = model'\ (optimiser\ data\ p\ model'\ loss\_function)\ 7$$

Proposed Programming in *Idealised TensorFlow*

simply-typed λ-calculus, extended by:

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

$$\frac{A \vdash \Gamma, T}{A \mid \Gamma, x : T, \Delta \mid - \vdash x : T} \qquad \frac{A \mid \Gamma, x : T' \mid \vec{p} \vdash t : T}{A \mid \Gamma \mid \vec{p} \vdash \lambda x^{T'}.t : T' \to T}$$

$$\frac{p \in \mathbb{F}}{A \mid \Gamma \mid - \vdash p : \mathbb{F}} \qquad \frac{A \mid \Gamma \mid \vec{p} \vdash t : T' \to T \quad A \mid \Gamma \mid \vec{q} \vdash u : T}{A \mid \Gamma \mid \vec{p}, \vec{q} \vdash t\ u : T}$$

$$\frac{A \mid \Gamma \mid \vec{p} \vdash t_1 : T_1 \quad A \mid \Gamma \mid \vec{q} \vdash t_2 : T_2 \quad \# : T_1 \to T_2 \to T}{A \mid \Gamma \mid \vec{p}, \vec{q} \vdash t_1 \ \# \ t_2 : T}$$

$$\frac{p \in \mathbb{F}}{A \mid \Gamma \mid p \vdash \{p\} : \mathbb{F}} \qquad \frac{A, a \mid \Gamma, f : V_a \to T', x : V_a \mid \vec{p} \vdash t : T \quad A \vdash \Gamma, T', T}{A \mid \Gamma \mid \vec{p} \vdash \mathrm{A}_a^{T'}(f, x).t : T' \to T}$$

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 sea*
- *rewriting*
- *observing*
  “token”

computation with computation graphs

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 sea*
- *rewriting*
- *observing*
  “token”

computation with computation graphs

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/)

an EDSL, for machine learning, with computation graphs

- ✓ autom
- ✓ autom
- limited
- imper

online visualiser
https://cwtsteven.github.io/GoI-TF-Visualiser/CBV-with-CBN-embedding/index.html

… to see *Idealised Tensorflow* in full action!

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/)

an EDSL, for machine learning, with computation graphs

- ✓ autom
- ✓ autom
- limited
- imper

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]

Idealised TensorFlow

a *calculus* with higher-order computation graphs

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

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
-