Proof synthesis and differential linear logic

Daniel Murfet

based on joint work with James Clift

- J. Clift and D. Murfet, *Derivatives of Turing Machines in Linear Logic*, arXiv: 1805.11813 (gradient descent on TMs in Section 7).
- J. Clift and D. Murfet, *Encodings of Turing Machines in Linear Logic*, arXiv: 1805.10770.
- J. Clift and D. Murfet, *Cofree coalgebras and differential linear logic*, Mathematical Structures in Computer Science (arXiv 2017).
- D. Murfet, *On Sweedler's cofree cocommutative coalgebra,* J. Pure and Applied Algebra, 219 (2015) 5289-5304.
- D. Murfet, *Logic and linear algebra: an introduction*, arXiv: 1407.2650.

therisingsea.org

Theoretical Computer Science 50 (1987) 1-102
North-Holland

LINEAR LOGIC*

Jean-Yves GIRARD
Équipe de Logique Mathématique, UA 753 du CNRS, UER de Mathématiques, Université de Paris VII, 75251 Paris, France

V.5. The exponentials

Already in the well-known case of lambda-calculus, there are two traditions:
- the tradition of identifying the variables, which comes from beta-conversion, when we substitute u for all occurrences of x;
- the tradition coming from the implementation, which tries to repress or control substitution.

The first tradition rests on safe logical grounds, whereas the second one is a kind of *bricolage* with hazardous justifications.

(i) As long as linear logic was resting on quantitative ideas, the principle (inspired on the way Krivine handled beta-conversion by restricting substitution to headvariables, and on the fact that a term is linear in its headvariable) suggested to use a linear 'first-order development', based on the identification between $!A$ and $1 + A.!A$. The operations of identification could be seen as formal derivation or formal primitive. The interest of this approach was to propose, at the theoretical level, to replace brutal beta-conversion by iterated linear conversions.

Linear Logic (sketch)

Types: $A, B, C, \ldots, \; !A, \; \underset{\wedge}{A \otimes B}, \; \underset{\Rightarrow}{A \multimap B}, \ldots, \; \underline{int}_A$ (integers), $\underline{bint}_A$ (binary integers), ...

Proofs: $\underline{n} : \underline{int}_A$ (Church numerals), for $n \geq 0$ an integer

$\underline{S} : \underline{bint}_A$ for any $S \in \{0,1\}^*$ e.g. $S = 001$.

$\underline{repeat} : \; !\underline{bint}_A \multimap \underline{bint}_A$

Cut-elimination is an equivalence relation on proofs

$$\dfrac{\dfrac{\begin{matrix}\underline{S}\\ \vdots \\ \vdash \underline{bint}\end{matrix}}{\vdash !\underline{bint}}\,\text{prom} \qquad \begin{matrix}\underline{repeat}\\ \vdots \\ !\underline{bint} \vdash \underline{bint}\end{matrix}}{\vdash \underline{bint}} \quad \sim \quad \begin{matrix}\underline{SS}\\ \vdots \\ \vdash \underline{bint}\end{matrix}$$

i.e. $\underline{repeat}(\underline{S}) = \underline{SS}$

How does this refine the standard encodings in λ-calculus?

ELSEVIER

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®
Theoretical Computer Science 309 (2003) 1–41

Theoretical
Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

The differential lambda-calculus

Thomas Ehrhard*, Laurent Regnier

Institut de Mathématiques de Luminy, CNRS-UPR 9016, 163 Avenue de Luminy, F-13288 Marseille, France

Abstract

We present an extension of the lambda-calculus with differential constructions. We state and prove some basic results (confluence, strong normalization in the typed case), and also a theorem relating the usual Taylor series of analysis to the linear head reduction of lambda-calculus.

Keywords: Lambda-calculus; Linear logic; Denotational semantics; Linear head reduction

Differential Linear Logic (sketch)

$$\cfrac{\begin{array}{c}\underline{T}\\ \vdots\\ \vdash \text{bint}\end{array} \qquad \cfrac{\cfrac{\begin{array}{c}\underline{S}\\ \vdots\\ \vdash \text{bint}\end{array}}{\vdash {!}\text{bint}}\,\text{prom} \qquad \cfrac{\begin{array}{c}\underline{\text{repeat}}\\ \vdots\\ {!}\text{bint} \vdash \text{bint}\end{array}}{{!}\text{bint}, \text{bint} \vdash \text{bint}}\,\partial}{\text{bint} \vdash \text{bint}}}{\vdash \text{bint}} \quad \sim \quad \begin{array}{c}\underline{ST} + \underline{TS}\\ \vdots\\ \vdash \text{bint}\end{array}$$

cut-elimination in diff. linear logic

Let $\varepsilon^2 = 0$ be an infinitesimal, then

$$\begin{aligned}\text{repeat}(S + \varepsilon T) &= (S + \varepsilon T)(S + \varepsilon T)\\ &= SS + \varepsilon ST + \varepsilon TS + \varepsilon^2 TT\\ &= SS + \varepsilon(ST + TS)\end{aligned}$$

__Claim__ (Ehrhard-Regnier) The derivative of __repeat__ at __S__ in the direction __T__ is __ST__+__TS__.

?

Deduction rules for (intuitionistic, first-order) linear logic

$$\text{(Dereliction):} \quad \frac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{der}$$

$$\text{(Contraction):} \quad \frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{ctr}$$

$$\text{(Weakening):} \quad \frac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{weak}$$

$$\text{(Axiom):} \quad \frac{}{A \vdash A} \qquad \text{(Cut):} \quad \frac{\Gamma \vdash A \qquad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B}\,\text{cut} \qquad \text{(Promotion):} \quad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\text{prom}$$

$$\text{(Left }\multimap\text{):} \quad \frac{\Gamma \vdash A \qquad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \multimap B, \Delta \vdash C}\,\multimap L \qquad \text{(Left }\otimes\text{):} \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\,\otimes\text{-}L$$

$$\text{(Right }\multimap\text{):} \quad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\,\multimap R \qquad \text{(Right }\otimes\text{):} \quad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,\otimes\text{-}R$$

Deduction rules for (intuitionistic, first-order) linear logic

(Dereliction): $$\frac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\ \text{der}$$

(Contraction): $$\frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\ \text{ctr}$$

(Weakening): $$\frac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\ \text{weak}$$

$\underline{001}$: $\mathbf{bint}_A = !(A \multimap A) \multimap (!(A \multimap A) \multimap (A \multimap A))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A, A \multimap A \vdash A}\multimap L}{A \multimap A, A \multimap A, A \multimap A \vdash A \multimap A}\multimap R}{\color{red}{!(A \multimap A)}, \color{red}{!(A \multimap A)}, \color{blue}{!(A \multimap A)} \vdash A \multimap A}\ 3\times\ \text{der}}{\color{red}{!(A \multimap A)}, \color{blue}{!(A \multimap A)} \vdash A \multimap A}\ \text{ctr}}{\vdash \mathbf{bint}_A}\ 2\times \multimap R$$

(Axiom): $$\frac{}{A \vdash A}$$

(Cut): $$\frac{\Gamma \vdash A \qquad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B}\ \text{cut}$$

(Promotion): $$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ \text{prom}$$

(Left $\multimap$): $$\frac{\Gamma \vdash A \qquad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \multimap B, \Delta \vdash C}\multimap L$$

(Left $\otimes$): $$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\ \otimes\text{-}L$$

(Right $\multimap$): $$\frac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\multimap R$$

(Right $\otimes$): $$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{-}R$$

$$\mathbf{bint}_A = !(A \multimap A) \multimap (!(A \multimap A) \multimap (A \multimap A)).$$

$$E = A \multimap A$$

$$\underline{\mathrm{repeat}} : !\mathbf{bint}_A \multimap \mathbf{bint}_A$$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$\textcolor{red}{!E} \vdash !E$}
\AxiomC{}
\UnaryInfC{$\textcolor{blue}{!E} \vdash !E$}
\AxiomC{}
\UnaryInfC{$\textcolor{red}{!E} \vdash !E$}
\AxiomC{}
\UnaryInfC{$\textcolor{blue}{!E} \vdash !E$}
\AxiomC{$\underline{\mathrm{comp}}^2_A$}
\noLine
\UnaryInfC{$\vdots$}
\noLine
\UnaryInfC{$E, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$\textcolor{blue}{!E}, !E \multimap E, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$\textcolor{red}{!E}, \textcolor{blue}{!E}, \mathbf{bint}_A, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$\textcolor{blue}{!E}, \textcolor{red}{!E}, \textcolor{blue}{!E}, \mathbf{bint}_A, !E \multimap E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$\textcolor{red}{!E}, \textcolor{blue}{!E}, \textcolor{red}{!E}, \textcolor{blue}{!E}, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\RightLabel{ctr}
\UnaryInfC{$\textcolor{red}{!E}, \textcolor{blue}{!E}, \textcolor{blue}{!E}, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\RightLabel{ctr}
\UnaryInfC{$\textcolor{red}{!E}, \textcolor{blue}{!E}, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\doubleLine
\RightLabel{$2\times \multimap R$}
\UnaryInfC{$\mathbf{bint}_A, \mathbf{bint}_A \vdash \mathbf{bint}_A$}
\doubleLine
\RightLabel{$2\times$ der}
\UnaryInfC{$!\mathbf{bint}_A, !\mathbf{bint}_A \vdash \mathbf{bint}_A$}
\RightLabel{ctr}
\UnaryInfC{$!\mathbf{bint}_A \vdash \mathbf{bint}_A$}
\end{prooftree}
$$

Sweedler semantics

$$[\![-]\!] : \mathcal{LL} \longrightarrow \mathrm{Vect}_k$$

(k an algebraically closed field)

$$[\![A \multimap B]\!] = \mathrm{Hom}_k([\![A]\!], [\![B]\!])$$

If $\dim [\![A]\!] = m$, $\dim [\![B]\!] = n$ this is the space of $n \times m$ matrices.

$$[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$$

Dimension mn, spanned by a basis $u_i \otimes v_j$ where $(u_i)_{i=1}^{m}$, $(v_j)_{j=1}^{n}$ are bases for $[\![A]\!]$, $[\![B]\!]$.

$$[\![!A]\!] = \bigoplus_{P \in [\![A]\!]} \mathrm{Sym}([\![A]\!])$$

Cofree cocommutative coalgebra over $[\![A]\!]$, studied by Sweedler.

Cut = composition

Contraction = comultiplication

Weakening = counit

$$\left[\!\!\left[\dfrac{\begin{array}{c}\pi \\ \vdots \\ !\Gamma \vdash A\end{array}}{!\Gamma \vdash !A}\,\mathrm{prom} \right]\!\!\right] = \text{unique morphism of coalgebras } \varphi \text{ making the diagram below commute}$$

$$\begin{tikzcd}
{[\![!\Gamma]\!]} \arrow[r, dashed, "\varphi"] \arrow[dr, bend right, "{[\![\pi]\!]}"'] & {[\![!A]\!]} \arrow[d, "\text{universal}"] \\
& {[\![A]\!]}
\end{tikzcd}$$

(see "Cofree coalgebras and differential LL")

Differentiating Turing Machines

language of closed symmetric monoidal categories with cofree coalgebras

$$\begin{tikzcd}
\{\text{Turing machines}\} \arrow[r, "\text{encode}"] & \{\text{linear logic proofs}\} \arrow[d, "\text{add dual numbers}"] \arrow[dr, bend left, "[\![-]\!]"] & \\
& \{\text{differential linear logic proofs}\} \arrow[r, "[\![-]\!]"] \arrow[loop below, "\partial"] & \mathrm{Vect}_k
\end{tikzcd}$$

based on work of Girard

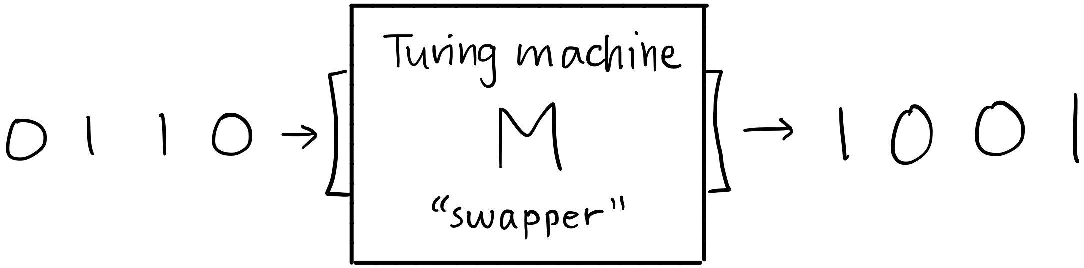
Ehrhard-Regnier derivative

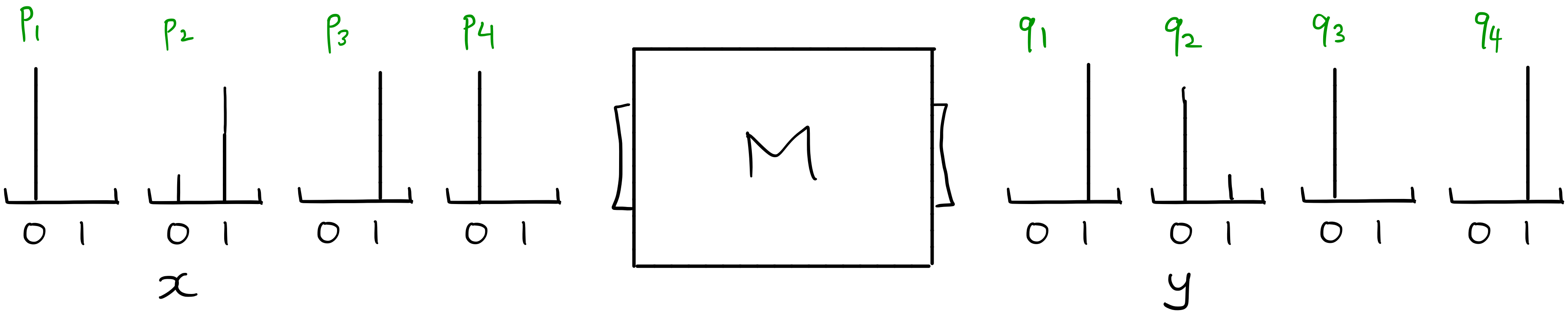
Sweedler semantics (based on cofree coalgebras)

QUESTION : What does the derivative of a Turing Machine compute?

Differentiating Turing Machines

How to make infinitesimal changes to discrete inputs?





$$\frac{\partial y}{\partial x}\Big|_{x=0} = \frac{\partial}{\partial x}(1-x)\Big|_{x=0} = -1$$

If we propagate uncertainty using standard probability, answers are independent of the <u>algorithm</u> and therefore meaningless.

Differentiating Turing Machines

How to propagate uncertainty through an algorithm?

Use the Sweedler semantics!

$$X_1 \; X_2 \; X_3 \; X_4 \rightarrow \left[\!\left[\boxed{\begin{array}{c}\text{Naive probabilistic} \\ \text{extension of} \\ M\end{array}} \right]\!\right] \rightarrow Y_1 \; Y_2 \; Y_3 \; Y_4 \qquad (t \text{ steps})$$

$$x = P(X_2 = 0) \qquad\qquad y = P(Y_2 = 0)$$

$\left.\dfrac{\partial y}{\partial x}\right|_{x=0}$ is computed by the Ehrhard-Regnier derivative of the t-step function of the encoding of M.

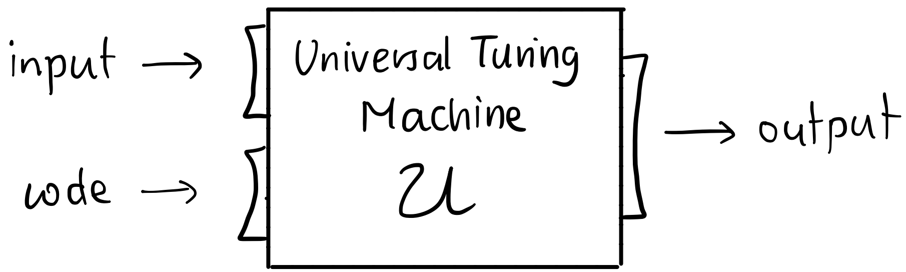
<u>QUESTION</u> : What does the derivative of a Turing Machine compute?

<u>ANSWER</u> : Rates of change of naive probability.

Proof Synthesis / TM synthesis

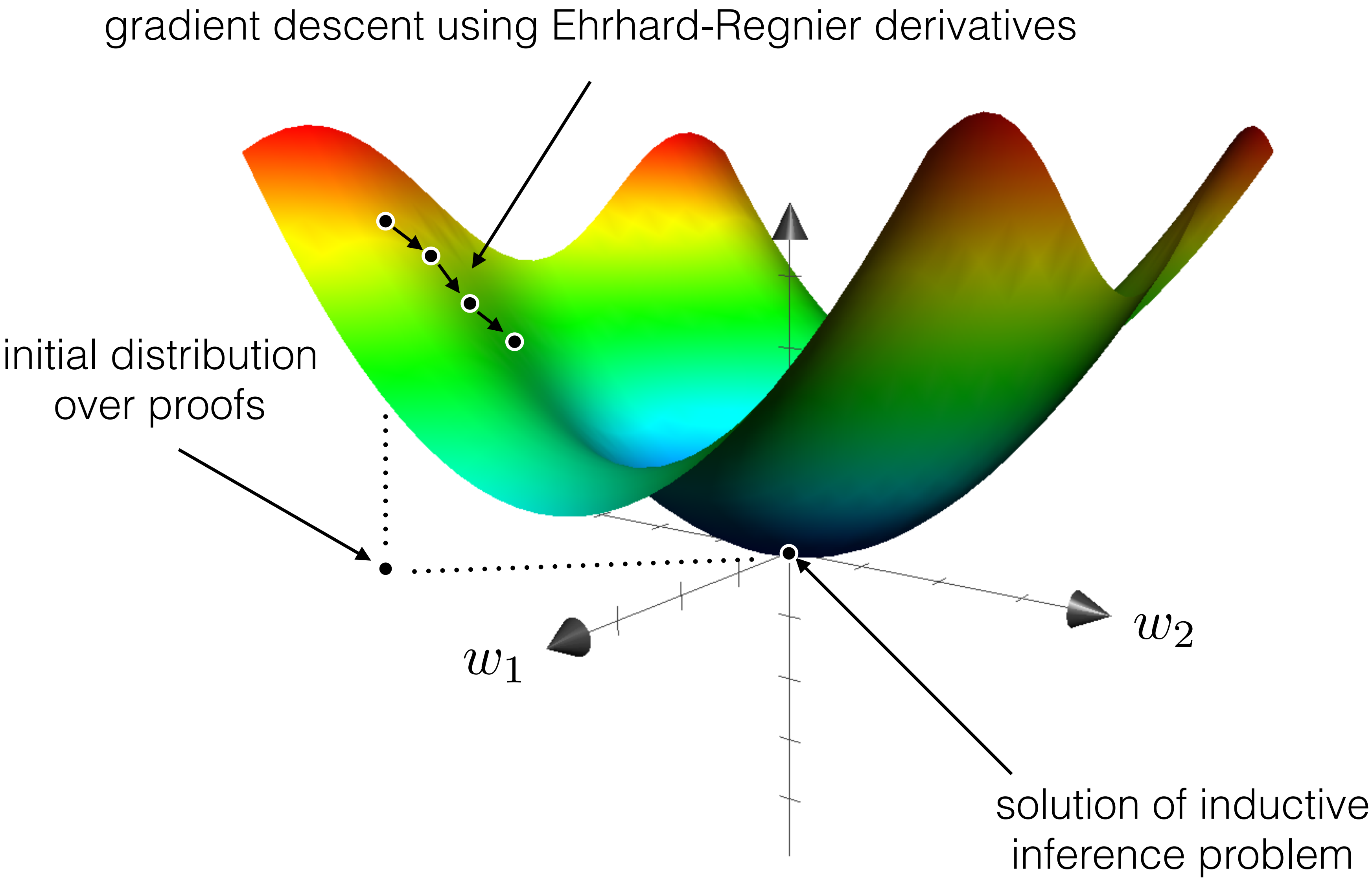
PROBLEM Given $u : A$ and $v : B$ find $\pi : !A \multimap B$ s.t. $\pi(u) = v$.

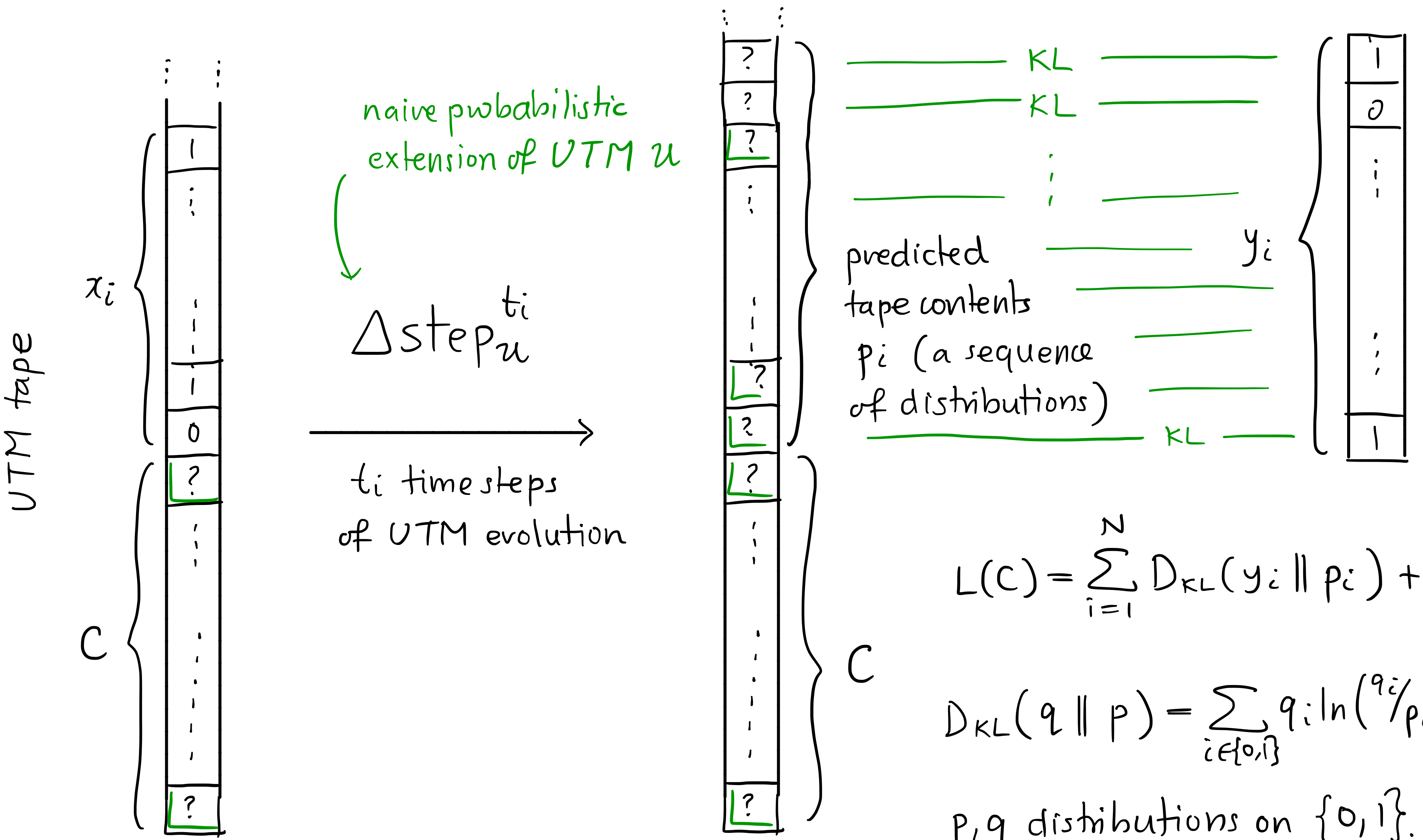
(usually for multiple pairs, subject to "regularisation" i.e. simplicity of π)



Turing Machine synthesis by gradient descent :

Vary distributions over code bits $\in [0,1]^N$ to minimise a smooth loss function $L : [0,1]^N \longrightarrow \mathbb{R}$.

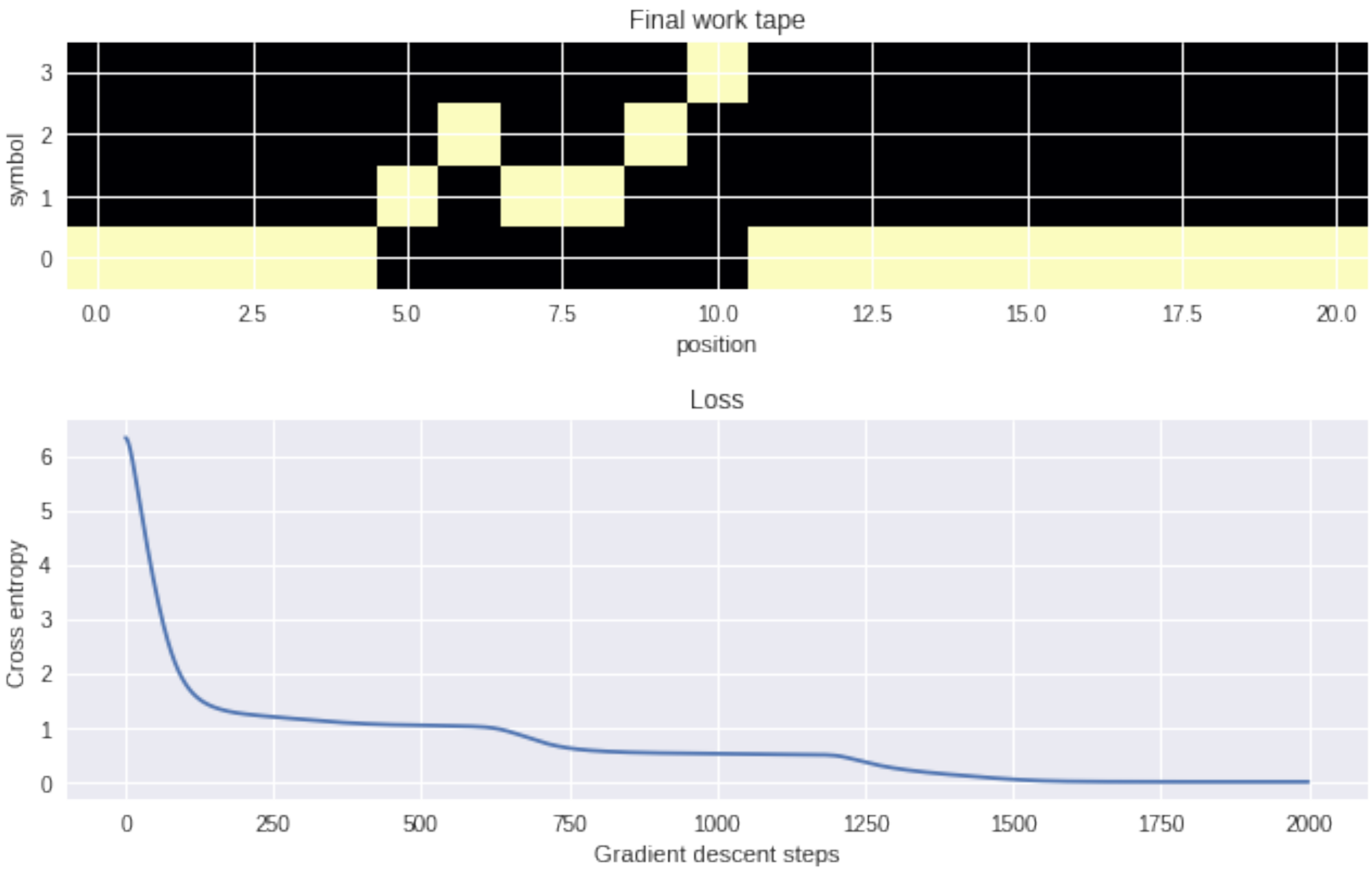

gradient descent using Ehrhard-Regnier derivatives
initial distribution
over proofs
w_1
w_2
solution of inductive
inference problem

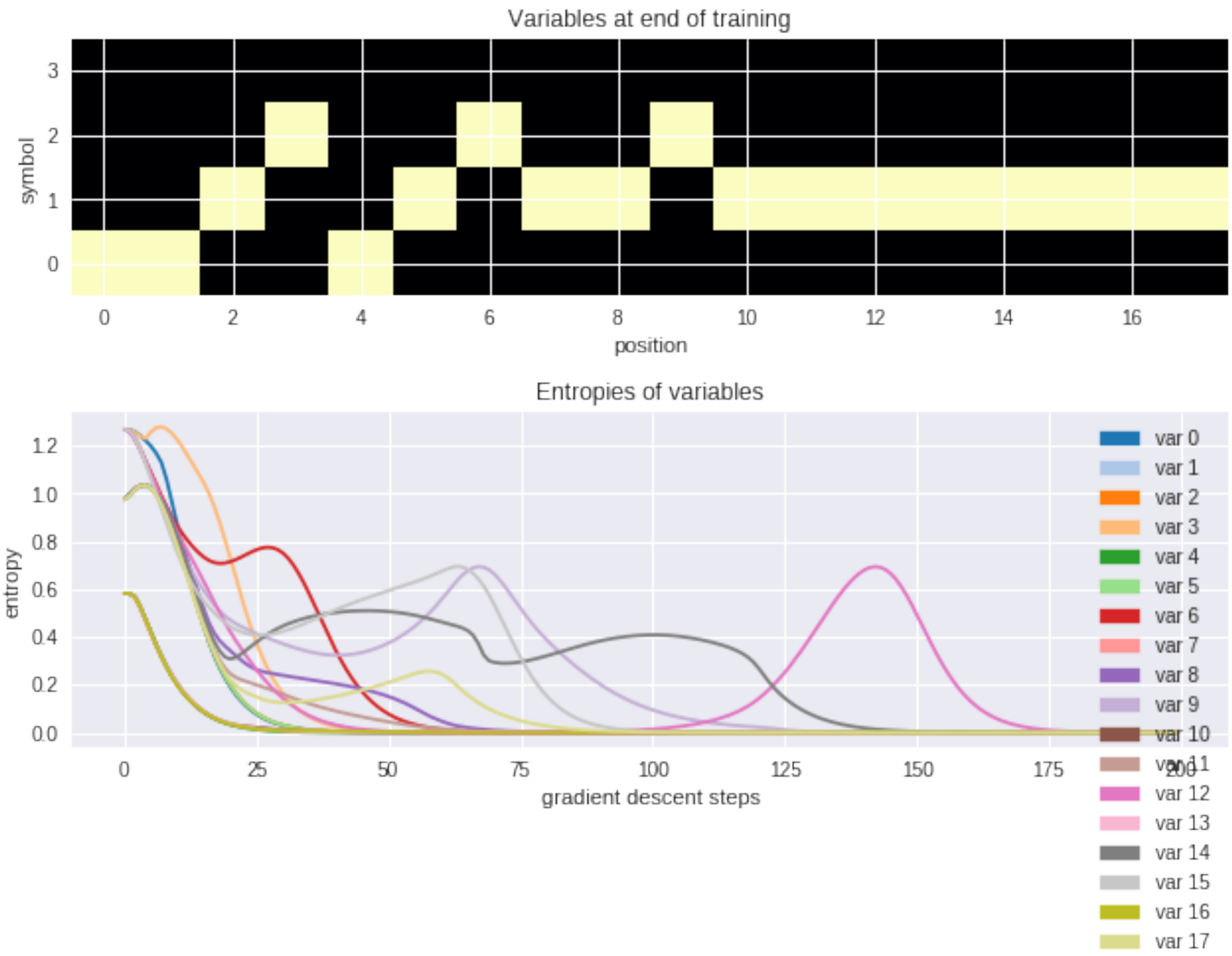


$$L(C) = \sum_{i=1}^{N} D_{KL}(y_i \,\|\, p_i) + \cdots$$

$$D_{KL}(q \,\|\, p) = \sum_{i \in \{0,1\}} q_i \ln\left(q_i / p_i\right)$$

p, q distributions on $\{0,1\}$.


Final work tape
symbol
3
2
1
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
position
Loss
Cross entropy
6
5
4
3
2
1
0
0
250
500
750
1000
1250
1500
1750
2000
Gradient descent steps


Variables at end of training
symbol
3
2
1
0
0
2
4
6
8
10
12
14
16
position
Entropies of variables
entropy
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
gradient descent steps
var 0
var 1
var 2
var 3
var 4
var 5
var 6
var 7
var 8
var 9
var 10
var 11
var 12
var 13
var 14
var 15
var 16
var 17

- Synthesis by gradient descent works in toy examples, but is <u>unlikely to work</u> in nontrivial examples (explosion of local minima, it seems Occam's razor is not a sufficiently strong prior in the continuous regime).

- Similar methods of propagating uncertainty through algorithms have arisen (in an ad hoc way) in the machine learning literature:
    - (DeepMind) R. Evans, E. Grefenstette "Learning explanatory rules from noisy data" Journal of AI Research 61 (2018) 1-64.
    - (Microsoft) A. Gaunt et. al "TerpreT: a probabilistic programming language for program induction" 2016.

Conclusion

1. Proofs in linear logic admit a <u>derivative</u> (Ehrhard-Regnier)

2. These derivatives compute (for certain proofs) rates of change of <u>naive probability</u> (Clift-M).

3. One can do <u>proof synthesis</u> by <u>gradient descent</u>, using loss functions defined on spaces of distributions over proofs (although this is not currently practical!).

therisingsea.org