Mathematical Structures in Computer Science (2020), **30**, pp. 379–415
doi:10.1017/S0960129520000109

CAMBRIDGE UNIVERSITY PRESS

PAPER

Encodings of Turing machines in linear logic

James Clift and Daniel Murfet*

Department of Mathematics, University of Melbourne, Melbourne, Australia
*Corresponding author. Email: d.murfet@unimelb.edu.au

(Received 8 November 2018; revised 8 January 2020; accepted 18 April 2020)

Abstract

The Sweedler semantics of intuitionistic differential linear logic takes values in the category of vector spaces, using the cofree cocommutative coalgebra to interpret the exponential and primitive elements to interpret the differential structure. In this paper, we explicitly compute the denotations under this semantics of an interesting class of proofs in linear logic, introduced by Girard: the encodings of step functions of Turing machines. Along the way we prove some useful technical results about linear independence of denotations of Church numerals and binary integers.

Keywords: linear logic, differential linear logic, Turing machines

1. Introduction

The Sweedler semantics is a denotational semantics of first-order intuitionistic linear logic in the category of vector spaces over an algebraically closed field k of characteristic zero (Murfet 2014; Clift and Murfet 2017). The $\otimes$ and & connectives of linear logic are interpreted by the usual tensor product and direct sum of vector spaces, while the exponential ! is interpreted by Sweedler's cofree cocommutative coalgebra (Sweedler 1969; Murfet 2015). The virtue of the Sweedler semantics is that it is amenable to computing examples, as we demonstrate in this paper with an exploration of the denotational semantics of Girard's encoding (Girard 1995) of Turing machines into linear logic, based on Turing's encoding of his machines as λ-terms (Turing 1937, Appendix).

Here is a basic motivating question: given a Turing machine M what is the information content of the denotation under the Sweedler semantics of the proof

$$\begin{array}{c}\pi\\ \vdots\\ !\mathbf{bool} \vdash \mathbf{bool}\end{array}$$

which computes the content of the tape square directly under the head after p steps of M? For simplicity we provide only a single input symbol to the machine. The denotation is a linear map

$$[\![\pi]\!] : ![\![\mathbf{bool}]\!] \longrightarrow [\![\mathbf{bool}]\!]$$

where $[\![\mathbf{bool}]\!]$ is a vector space containing denotations $[\![\underline{0}]\!], [\![\underline{1}]\!]$ of $\underline{0}, \underline{1} : \mathbf{bool}$ and $![\![\mathbf{bool}]\!]$ is a cocommutative coalgebra. A vector $g \in ![\![\mathbf{bool}]\!]$ is called *group-like* if $\Delta(g) = g \otimes g$ where Δ is the co-multiplication, and there is a group-like element $|\emptyset\rangle_v$ canonically associated with any

$v \in [\![\mathbf{bool}]\!]$ (see Clift and Murfet 2017, Section 2.2). Using the methods developed in this paper, it is easy to compute a polynomial $f(x) \in k[x]$ such that

$$[\![\pi]\!]|\emptyset\rangle_{(1-x)[\![\underline{0}]\!]+x[\![\underline{1}]\!]} = (1-f(x))[\![\underline{0}]\!] + f(x)[\![\underline{1}]\!]\,.$$

The values of the polynomial $f(x)$ at $x=0$ and $x=1$ encode the input–output behaviour of running the machine M for p steps, since if the output is τ on input σ then by construction $[\![\pi]\!]|\emptyset\rangle_{[\![\underline{\sigma}]\!]} = [\![\underline{\tau}]\!]$. However, the more interesting observation is that the values of the *derivative* of $f(x)$ are also meaningful: they are computed by the denotation in the Sweedler semantics of the Ehrhard–Regnier derivative (Ehrhard and Regnier 2003; Ehrhard 2016) of π. The details are in the sequel (Clift and Murfet 2018) but as that work motivates the results here, we include a sketch.

The bridge between the ordinary derivative of $f(x)$ and the Ehrhard–Regnier derivative of π is the concept of a primitive element in the theory of coalgebras, which is an equivalent but dual point of view on the concept of tangent vectors; see Clift and Murfet (2017), Section 2.3. A vector $z \in ![\![\mathbf{bool}]\!]$ is called *primitive* with respect to a group-like element g if $\Delta(z) = g \otimes z + z \otimes g$, and given $u, v \in [\![\mathbf{bool}]\!]$ there is a canonically associated primitive element $|u\rangle_v \in ![\![\mathbf{bool}]\!]$. This construction is linear in u but not in v. One should think of the primitive element

$$\big|\alpha[\![\underline{0}]\!] + \beta[\![\underline{1}]\!]\big\rangle_{[\![\underline{0}]\!]} = \alpha\big|[\![\underline{0}]\!]\big\rangle_{[\![\underline{0}]\!]} + \beta\big|[\![\underline{1}]\!]\big\rangle_{[\![\underline{0}]\!]} \in ![\![\mathbf{bool}]\!]$$

as a tangent vector with coefficients (α, β) at the point $[\![\underline{0}]\!]$ of the two-dimensional subspace of $[\![\mathbf{bool}]\!]$ spanned by $[\![\underline{0}]\!]$, $[\![\underline{1}]\!]$, or more conceptually, as an infinitesimal variation of the symbol 0 in the direction $\alpha\frac{\partial}{\partial 0} + \beta\frac{\partial}{\partial 1}$. These tangent vectors are the subject of the language of differential linear logic, in the sense that the primitive elements $|[\![\underline{\sigma}]\!]\rangle_{[\![\underline{\tau}]\!]}$ for $\sigma, \tau \in \{0, 1\}$ are the denotations under the Sweedler semantics of the proofs in differential linear logic which cut the Ehrhard–Regnier derivative of π against the pair $\underline{\tau}, \underline{\sigma}$.

To give a concrete example,

$$[\![\pi]\!]\big|[\![\underline{1}]\!] - [\![\underline{0}]\!]\big\rangle_{[\![\underline{0}]\!]} = -f'(0)[\![\underline{0}]\!] + f'(0)[\![\underline{1}]\!] \tag{1}$$

where $|[\![\underline{1}]\!] - [\![\underline{0}]\!]\rangle_{[\![\underline{0}]\!]}$ represents an infinitesimal variation of the symbol 0 in the direction that points from 0 to 1, and the right-hand side of (1) gives the corresponding infinitesimal variation in the output distribution over 0, 1. The information content of $[\![\pi]\!]$ therefore includes, in addition to the input–output behaviour of M after p steps, information about how the output varies when the input is varied 'infinitesimally'. Turing machines M, M' which compute the same output on the same input when executed for p steps may nonetheless have different behaviour with respect to infinitesimal variations in their input, and differential linear logic provides a logic of these differences in behaviour.

In this paper, we explain the encoding of the step function of a Turing machine into linear logic in a form that is amenable to Ehrhard–Regnier's derivative, we calculate the denotations of these encodings under the Sweedler semantics and we provide the technical foundations for interpreting these denotations as polynomials, in order to set the stage for the discussion in Clift and Murfet (2018) of derivatives that we have previewed above.

Outline of the paper. The encoding of the step function of a Turing machine M as a proof in linear logic is not unique, and indeed the paper is broadly organised around *four* different variants, each with different tradeoffs:

- *The Girard encoding* (Section 4) encodes the state of the tape as a pair of binary integers, and is a modified form of the encoding in Girard (1995) with a more conservative use of second-order quantifiers. The step function of M is encoded as a proof of

$$!\mathbf{bint}_{A^3} \otimes !\mathbf{bint}_{A^3} \otimes !\mathbf{bool}_{A^3} \vdash !\mathbf{bint}_A \otimes !\mathbf{bint}_A \otimes !\mathbf{bool}_A \tag{2}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

for any type A, where $\mathbf{bint}_A$ denotes the type of binary integers and $\mathbf{bool}_A$ denotes the type of booleans (see Section 2.2). The denotation of this encoding in the Sweedler semantics is given in Lemma 4.19.

- *The Boolean version of the Girard encoding* (Section 4.2) encodes the state of the tape as a sequence of booleans, and is defined by converting such a sequence into a pair of binary integers, running the Girard encoding for some number of steps and then converting the resulting pair of binary integers representing the tape back into a sequence of booleans. The step function of M is encoded by a family of proofs of

$$a\,!\mathbf{bool}_B \otimes b\,!\mathbf{bool}_B \otimes !_n\mathbf{bool}_B \vdash \left(!\mathbf{bool}_A\right)^{\otimes c} \otimes \left(!\mathbf{bool}_A\right)^{\otimes d} \otimes !_n\mathbf{bool}_A \tag{3}$$

with $a, b, c, d \geq 1$ giving the bounds of the initial and final tape, and where B is A^g for some function g of c, d and the number of time steps.

- *The direct Boolean encoding* (Section 5) also encodes the state of the tape using booleans and it comes in two flavours, one in which the tape contents are encoded *relative* to the head position, and one in they are encoded in *absolute* coordinates. In the relative case the step function of M is encoded by proofs of

$$!_s\mathbf{bool}_A^{\otimes 2h+1} \otimes !_n\mathbf{bool}_A \vdash !_s\mathbf{bool}_A^{\otimes 2h+3} \otimes !_n\mathbf{bool}_A \tag{4}$$

while in the absolute case it is encoded by proofs of

$$!_s\mathbf{bool}_A^{\otimes h} \otimes !_n\mathbf{bool}_A \otimes !_h\mathbf{bool}_A \vdash !_s\mathbf{bool}_A^{\otimes h} \otimes !_n\mathbf{bool}_A \otimes !_h\mathbf{bool}_A \tag{5}$$

where ${}_s\mathbf{bool}_A$ is the type of s-booleans, used to encode tape symbols, and ${}_h\mathbf{bool}_A$ is used to track the head position. The denotations of these encodings are given in Remarks 5.6 and 5.13 respectively.

These encodings all belong to a special class of proofs in linear logic which we call the *component-wise plain proofs*. We introduce and give the basic properties of this class in Section 3. Finally in Appendix A we study the denotations of integers and binary integers under the Sweedler semantics, and prove some basic linear independence results that are needed in the main text to establish the legitimacy of our description of denotations using polynomials (and which may be of independent interest).

Let us now briefly explain why we introduce four different encodings, and why Girard's original encoding in Girard (1995) is not suitable, in its original form, for our applications in differential linear logic. In Girard's original encoding he gives a proof which, when cut against an integer n and an encoding of the initial configuration of the Turing machine, returns the configuration of the Turing machine after n steps. This works by *iterating* an encoding of the one-step transition function, and this iteration requires second-order quantifiers. From the point of view of derivatives the use of second-order is problematic, because it is not clear how derivatives in linear logic should interact with second order. Our solution was to find a variation of Girard's encoding which introduces the second-order quantifiers only at the very bottom of the proof tree. This is the encoding given in Section 4. In Clift and Murfet (2018), we study the Ehrhard–Regnier derivative of the first-order proof which stops just before the use of second-order quantifiers.

Differentiating proofs involves making infinitesimal variations in inputs[1] and in the case of a proof encoding the step function of a Turing machine, it seems that the most natural infinitesimal variations to make are those describing the contents of an *individual* tape square. This is somewhat orthogonal to the approach of the Girard encoding, which uses two monolithic binary integers to encode the state of the tape. For this reason we were driven to develop the other encodings given above using sequence of booleans.

Related work. The work of Roversi (1999) fixes an error in Girard's original encoding, but from our point of view this error is irrelevant, since it concerns whether or not Girard's encoding is

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

typable in *light* linear logic. A different encoding of Turing machines into linear logic is given by Mairson and Terui (2003, Theorem 5) which uses booleans based on tensors rather than additives.

2. Background

Throughout k is an algebraically closed field of characteristic zero, and all vector spaces and coalgebras are defined over k. Coalgebras are all coassociative, counital and cocommutative (Sweedler 1969). We write $\mathrm{Prim}\,(C)$ for the set of primitive elements in a coalgebra C.

2.1 Linear logic and the Sweedler semantics

As introductory references for linear logic we recommend (Melliès 2009; Benton et al. 1992) and the survey (Murfet 2014). In this paper *linear logic* will always mean *first-order intuitionistic linear logic* with connectives $\otimes$, &, $\multimap$, ! and the corresponding introduction rules and cut-elimination transformations from Melliès (2009) and Benton et al. (1992). A *proof* is always a proof of a sequent in linear logic.

The semantics of intuitionistic linear logic in vector spaces using the cofree coalgebra to interpret the exponential, which we call the *Sweedler semantics* since Sweedler was the first to give a detailed study of the cofree coalgebra (Sweedler 1969), was introduced as an example by Hyland and Schalk (2003) and revisited in Murfet (2014) and Clift and Murfet (2017) with a focus on explicit formulas for the involved structures based on the results of Murfet (2015). Denotations of formulas and proofs will throughout be denoted by $[\![-]\!]$. Briefly, given formulas (we also use *type* as a synonym for formula) A, B the denotations are determined by the rules

$$\begin{aligned}
[\![A \otimes B]\!] &= [\![A]\!] \otimes [\![B]\!] \\
[\![A \,\&\, B]\!] &= [\![A]\!] \oplus [\![B]\!] \\
[\![A \multimap B]\!] &= \mathrm{Hom}_k\,([\![A]\!], [\![B]\!]) \\
[\![!A]\!] &= !\,[\![A]\!]
\end{aligned}$$

and a choice of vector spaces $[\![x]\!]$ for atomic formulas x, where $!V$ denotes the universal cocommutative coassociative counital coalgebra mapping to V. The universal morphism is usually denoted $d_V : !V \longrightarrow V$ or just d. We review here the description of $!V$ and this universal map; for more details see Clift and Murfet (2017, Section 2.4), Murfet (2015) and Murfet (2014, Section 5.2).

For any vector space V we define

$$!V = \bigoplus_{P \in V} \mathrm{Sym}_P\,(V) \tag{6}$$

where $\mathrm{Sym}_P\,(V) = \mathrm{Sym}\,(V)$ is the symmetric coalgebra. If $e_1, \ldots, e_n$ is a basis for V, then as a vector space $\mathrm{Sym}\,(V) \cong k[e_1, \ldots, e_n]$. Given $v_1, \ldots, v_s \in V$, the corresponding tensor in $\mathrm{Sym}_P\,(V)$ is written using a ket

$$|v_1, \ldots, v_s\rangle_P := v_1 \otimes \cdots \otimes v_s \in \mathrm{Sym}_P\,(V)\,.$$

And in particular, the identity element of $\mathrm{Sym}_P\,(V)$ is denoted by a vacuum vector

$$|\emptyset\rangle_P := 1 \in \mathrm{Sym}_P\,(V)\,.$$

With this notation the universal map $d : !V \longrightarrow V$ is defined for $P \in V$ by

$$d|\emptyset\rangle_P = P, \quad d|v\rangle_P = v, \quad d|v_1, \ldots, v_s\rangle_P = 0 \quad s > 1\,.$$

Since any vector in $!V$ is, by definition of the coproduct in (6), a finite linear combination of kets, these three formulas suffice to completely describe the linear map d. Similarly the comultiplication on $!V$ is defined by

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$\Delta|v_1,\ldots,v_s\rangle_P = \sum_{I\subseteq\{1,\ldots,s\}} |v_I\rangle_P \otimes |v_{I^c}\rangle_P$$

where I ranges over all subsets including the empty set, and for a subset $I=\{i_1,\ldots,i_p\}$ we denote by v_I the sequence $v_{i_1},\ldots,v_{i_p}$, and I^c is the complement of I. In particular

$$\Delta|\emptyset\rangle_P = |\emptyset\rangle_P \otimes |\emptyset\rangle_P\,.$$

The counit $!V \longrightarrow k$ is defined by $|\emptyset\rangle_P \mapsto 1$ and $|v_1,\ldots,v_s\rangle_P \mapsto 0$ for $s>0$.

We write $A^n = A \,\&\, \cdots \,\&\, A$ where there are n copies of A. Given a type A, $\pi : A$ means that π is a proof of $\vdash A$. Given a proof π of $!\Gamma \vdash B$, we write prom (π) for the proof obtained by applying the promotion rule to π to obtain a proof of $!\Gamma \vdash !B$.

Whenever we talk about a set of proofs $\mathcal{P}$ of a formula A in linear logic, we always mean a set of proofs *modulo the equivalence relation of cut-elimination*. Given a set of proofs $\mathcal{N}$ we write $[\![\mathcal{N}]\!]$ for $\{[\![v]\!]\}_{v\in\mathcal{N}}$. If proofs π,π' are equivalent under cut-elimination, then $[\![\pi]\!]=[\![\pi']\!]$, so the function $[\![-]\!]:\mathcal{P}\longrightarrow[\![A]\!]$ extends uniquely to a k-linear map

$$k\mathcal{P} \xrightarrow{\quad[\![-]\!]\quad} [\![A]\!] \tag{7}$$

where $k\mathcal{P}$ is the free k-vector space generated by the set $\mathcal{P}$. If ψ is a proof of $!A_1,\ldots,!A_r \vdash B$ and α_i is a proof of A_i for $1\le i\le r$, then $\psi(\alpha_1,\ldots,\alpha_r):B$ denotes the (cut-elimination equivalence class of) proof obtained by cutting ψ against the promotion of each α_i.

2.2 Encoding data as proofs

Given a base type A, we define

$$\begin{aligned}
\mathbf{bool}_A &= (A \,\&\, A) \multimap A\,,\\
{}_n\mathbf{bool}_A &= A^n \multimap A\,,\\
\mathbf{int}_A &= !(A\multimap A)\multimap(A\multimap A)\,,\\
\mathbf{bint}_A &= !(A\multimap A)\multimap\big(!(A\multimap A)\multimap(A\multimap A)\big)\,.
\end{aligned}$$

The encodings of booleans, integers and binary integers as proofs in linear logic go back to Girard's original paper Girard (1987). For each integer $n\ge 0$ there is a corresponding proof $\underline{n}_A$ of $\mathbf{int}_A$ (Clift and Murfet 2017, Section 4.1) and for $S\in\{0,1\}^*$ there is a corresponding proof $\underline{S}_A$ of $\mathbf{bint}_A$ (Clift and Murfet 2017, Section 4.2).

The two values of a boolean correspond to the following proofs $\underline{0}_A$ and $\underline{1}_A$ of $\mathbf{bool}_A$:

$$\dfrac{\dfrac{\overline{A\vdash A}}{A\,\&\,A\vdash A}\,\&L_0}{\vdash \mathbf{bool}_A}\multimap R \qquad \dfrac{\dfrac{\overline{A\vdash A}}{A\,\&\,A\vdash A}\,\&L_1}{\vdash \mathbf{bool}_A}\multimap R$$

whose denotations are projection onto the zeroth and first coordinates respectively. Note that we are using the convention that the left introduction rules for & are indexed by 0 and 1, rather than by the more conventional choice of 1 and 2, in order to be consistent with the usual assignment of 0 as 'false' and 1 as 'true'.

The n values of an n-boolean correspond to the projection maps $\mathrm{proj}_i:[\![A^n]\!]\to[\![A]\!]$, where $i\in\{0,\ldots,n-1\}$. We denote by $\underline{i}_A$ the proof

$$\dfrac{\dfrac{\overline{A\vdash A}}{A^n\vdash A}\,\&L_i}{\vdash {}_n\mathbf{bool}_A}\multimap R$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

whose denotation is proj_i. Here, by $\&L_i$ $(0 \leq i \leq n-1)$ we mean the rule which introduces $n-1$ new copies of A on the left, such that the original A is at position i, indexed from left to right.

2.3 Cartesian products of coalgebras

Let $(C, \Delta_C, \varepsilon_C)$ and $(D, \Delta_D, \varepsilon_D)$ be coalgebras. Then $C \otimes D$ is naturally a coalgebra (Sweedler 1969, p. 49), and it is the Cartesian product in the category of coalgebras (Sweedler 1969, p. 65). The following calculations are standard but will play such an important conceptual role in this paper and its sequel that it is worth reproducing them here.

Let $\mathbf{Coalg}_k$ denote the category of k-coalgebras. The following result is well known:

Lemma 2.1. *Let $\pi_C : C \otimes D \longrightarrow C$ and $\pi_D : C \otimes D \longrightarrow D$ to be the composites*

$$C \otimes D \xrightarrow{1 \otimes \varepsilon_D} C \otimes k \cong C \ , \qquad C \otimes D \xrightarrow{\varepsilon_C \otimes 1} k \otimes D \cong D \ . \tag{8}$$

These are coalgebra morphisms, and the tuple $(C \otimes D, \pi_C, \pi_D)$ is the Cartesian product of C, D in the category of coalgebras.

In particular, the group-like (Sweedler 1969, p. 57) elements decompose according to

$$\begin{aligned} G(C \otimes D) &\cong \mathbf{Coalg}_k(k, C \otimes D) \\ &\cong \mathbf{Coalg}_k(k, C) \times \mathbf{Coalg}_k(k, D) \\ &\cong G(C) \times G(D) \, . \end{aligned}$$

This isomorphism sends a pair (c, d) of group-like elements to the group-like element $c \otimes d$ in $C \otimes D$. Similarly for the primitive elements

$$\begin{aligned} \mathrm{Prim}\,(C \otimes D) &\cong \mathbf{Coalg}_k((k[\varepsilon]/\varepsilon^2)^*, C \otimes D) \\ &\cong \mathbf{Coalg}_k((k[\varepsilon]/\varepsilon^2)^*, C) \times \mathbf{Coalg}_k((k[\varepsilon]/\varepsilon^2)^*, D) \\ &\cong \mathrm{Prim}\,(C) \times \mathrm{Prim}\,(D) \, . \end{aligned}$$

A pair of primitive (Sweedler 1969, p. 199) elements x over c in C and y over d in D correspond to a pair of morphisms of coalgebras

$$\begin{aligned} \widetilde{x} &: (k[\varepsilon]/\varepsilon^2)^* \longrightarrow C \, , \quad 1^* \mapsto c, \ \varepsilon^* \mapsto x \\ \widetilde{y} &: (k[\varepsilon]/\varepsilon^2)^* \longrightarrow C \, , \quad 1^* \mapsto d, \ \varepsilon^* \mapsto y \, . \end{aligned}$$

And the corresponding primitive element in $C \otimes D$ is the image of ε^* under the map

$$\begin{gathered} (k[\varepsilon]/\varepsilon^2)^* \xrightarrow{\Delta} (k[\varepsilon]/\varepsilon^2)^* \otimes (k[\varepsilon]/\varepsilon^2)^* \xrightarrow{\widetilde{x} \otimes \widetilde{y}} C \otimes D \\ \varepsilon^* \longmapsto 1^* \otimes \varepsilon^* + \varepsilon^* \otimes 1^* \longmapsto c \otimes y + x \otimes d \, . \end{gathered}$$

Remark 2.2. A morphism of coalgebras sends primitive elements to primitive elements. In particular, if $\gamma : X \longrightarrow C \otimes D$ is a morphism of coalgebras, then the restriction gives a map $\mathrm{Prim}\,(\gamma) : \mathrm{Prim}\,(C) \longrightarrow \mathrm{Prim}\,(C \otimes D)$, and it is clear that the diagram

$$\begin{array}{ccc} \mathrm{Prim}\,(X) & \xrightarrow{\begin{pmatrix} \mathrm{Prim}\,(\gamma_C) \\ \mathrm{Prim}\,(\gamma_D) \end{pmatrix}} & \mathrm{Prim}\,(C) \times \mathrm{Prim}\,(D) \\ & \searrow^{\mathrm{Prim}\,(\gamma)} & \downarrow \cong \\ & & \mathrm{Prim}\,(C \otimes D) \end{array}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

commutes, where the vertical map is the canonical isomorphism described above. Thus the action of γ on primitive elements can be understood component-by-component.

3. Plain Proofs

In this section, we introduce the class of *plain proofs*. Our encodings of Turing machines in Section 4 will be component-wise plain.

Definition 3.1. *A proof of a sequent* $!A_1, \ldots, !A_r \vdash B$ *for* $r \geq 0$ *is* plain *if it is equivalent under cut-elimination to*

$$\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\end{array}}{n_1\, A_1, \ldots, n_r\, A_r \vdash B}\, der}{n_1\, !A_1, \ldots, n_r\, !A_r \vdash B}\, ctr/wk \qquad !A_1, \ldots, !A_r \vdash B$$

for some proof π *and tuple of non-negative integers* $\mathbf{n} = (n_1, \ldots, n_r)$*, where for* $n_i > 1$ *in the final step there is a corresponding contraction, and if* $n_i = 0$ *the final step involves a weakening. We write* nA *in a sequent to stand for* n *occurrences of* A*. We refer to the integer* n_i *as the* A_i-degree *and* $\mathbf{n}$ *as the* degree vector*.*

Example 3.2. Binary integers (Clift and Murfet 2017, Section 4.2) give plain proofs of the sequent $2!(A \multimap A) \vdash A \multimap A$.

Remark 3.3. If ψ is a plain proof as above, then:

- Suppose $A_i = A_j$ for some $i \neq j$ and let ρ be the proof obtained from ψ by contraction on $!A_i, !A_j$. Then ρ is plain.
- Let ρ be the proof of $!A, !A_1, \ldots, !A_r \vdash B$ obtained from ψ by weakening in the $!A$. Then ρ is plain.

Lemma 3.4. *Suppose that* $\theta_1, \ldots, \theta_r$ *are plain proofs with conclusions* $A_1, \ldots, A_r$ *and that* ψ *is a plain proof of* $!A_1, \ldots, !A_r \vdash B$*. Then the cut of* ψ *against the promotions of the* θ_i *is a plain proof with conclusion* B*.*

Proof. In the special case $r = 1$ the cut

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\end{array}}{n\, A \vdash B}\, n\times \text{der}}{n\, !A \vdash B}\, n-1\times \text{ctr}}{!A \vdash B}\, \text{prom} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\rho\\ \vdots\end{array}}{m\, B \vdash C}\, m\times \text{der}}{m\, !B \vdash C}\, m-1\times \text{ctr}}{!B \vdash C}}{!A \vdash !B}\, \text{cut}}{!A \vdash C}$$

is equivalent under cut-elimination (Melliès 2009, Section 3.9.3) to a proof of the form

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$\dfrac{\dfrac{\dfrac{\vdots}{mn\,A \vdash C}\, mn\times \text{der}}{mn\,!A \vdash C}\, mn-1\times \text{ctr}}{!A \vdash C}$$

which is plain. The general case is similar. □

Definition 3.5. *A proof*

$$\psi : !A_1, \ldots, !A_r \vdash !B_1 \otimes \cdots \otimes !B_s$$

is component-wise plain *if there are plain proofs*

$$\psi_i : !A_1, \ldots, !A_r \vdash B_i$$

for $1 \leq i \leq s$ *such that* ψ *is equivalent under cut-elimination to the proof*

$$\dfrac{\begin{array}{c}\otimes_{i=1}^{s} \text{prom}\,(\psi_i) \\ \vdots \\ s!A_1, \ldots, s!A_r \vdash !B_1 \otimes \cdots \otimes !B_s\end{array}}{!A_1, \ldots, !A_r \vdash !B_1 \otimes \cdots \otimes !B_s}\, ctr$$

We refer to the ψ_i *as the* components *of* ψ.

Observe that in the context of the definition the linear map $[\![\text{prom}\,(\psi_i)]\!]$ is a morphism of coalgebras, and the denotation of the proof ψ is precisely the morphism of coalgebras induced by the $[\![\text{prom}\,(\psi_i)]\!]$ into the tensor product of the $![\![B_i]\!]$ viewed as the Cartesian product in the category of coalgebras (Section 2.3). At the syntactic level this means in particular that the components ψ_i of a component-wise plain proof ψ may be recovered (of course, up to cut-elimination) by cutting against a series of weakenings and a dereliction.

This class of component-wise plain proofs is closed under composition:

Proposition 3.6. *Suppose given two component-wise plain proofs*

$$\psi : !A_1, \ldots, !A_r \vdash !B_1 \otimes \cdots \otimes !B_s\,,$$
$$\phi : !B_1, \ldots, !B_s \vdash !C_1 \otimes \cdots \otimes !C_t\,.$$

Then the proof

$$\dfrac{\dfrac{\begin{array}{c}\psi \\ \vdots \\ !A_1, \ldots, !A_r \vdash !B_1 \otimes \cdots \otimes !B_s\end{array} \qquad \dfrac{\begin{array}{c}\phi \\ \vdots \\ !B_1, \ldots, !B_s \vdash !C_1 \otimes \cdots \otimes !C_t\end{array}}{!B_1 \otimes \cdots \otimes !B_s \vdash !C_1 \otimes \cdots \otimes !C_t}\, \otimes L}{!A_1, \ldots, !A_r \vdash !C_1 \otimes \cdots \otimes !C_t}\, cut$$

which we denote $\phi \mid \psi$*, is component-wise plain.*

Proof. By hypothesis ψ, ϕ are equivalent under cut-elimination to the tensor products of promotions of components ψ_i, ϕ_j, and so $\phi \mid \psi$ is equivalent to a proof

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$\dfrac{\dfrac{\begin{array}{c}\otimes_i \text{prom}\,(\psi_i)\\ \vdots\\ s!A_1,\ldots,s!A_r \vdash !B_1 \otimes \cdots \otimes !B_s\end{array}}{!A_1,\ldots,!A_r \vdash !B_1 \otimes \cdots \otimes !B_s}\,\text{ctr} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\otimes_j \text{prom}\,(\phi_j)\\ \vdots\\ t!B_1,\ldots,t!B_s \vdash !C_1 \otimes \cdots \otimes !C_t\end{array}}{!B_1,\ldots,!B_s \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{ctr}}{!B_1 \otimes \cdots \otimes !B_s \vdash !C_1 \otimes \cdots \otimes !C_t}\,\otimes L}{!A_1,\ldots,!A_r \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{cut}$$

The cut-elimination rules of Melliès (2009, Section 3.10.2) apply to transform this to a proof

$$\dfrac{\dfrac{\begin{array}{c}\otimes_i \text{prom}\,(\psi_i)\\ \vdots\\ s!A_1,\ldots,s!A_r \vdash !B_1 \otimes \cdots \otimes !B_s\end{array} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\otimes_j \text{prom}\,(\phi_j)\\ \vdots\\ t!B_1,\ldots,t!B_s \vdash !C_1 \otimes \cdots \otimes !C_t\end{array}}{!B_1,\ldots,!B_s \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{ctr}}{!B_1 \otimes \cdots \otimes !B_s \vdash !C_1 \otimes \cdots \otimes !C_t}\,\otimes L}{s!A_1,\ldots,s!A_r \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{cut}}{!A_1,\ldots,!A_r \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{ctr}$$

which by Melliès (2009, Section 3.8.1) is equivalent under cut-elimination to cutting all the prom (ψ_i) against ϕ and then performing the contractions. But by Melliès (2009, Section 3.9.3) the resulting proof is equivalent under cut-elimination to cutting t copies of each prom (ψ_i) against the proof

$$\begin{array}{c}\otimes_j \text{prom}\,(\phi_j)\\ \vdots\\ t!B_1,\ldots,t!B_s \vdash !C_1 \otimes \cdots \otimes !C_t\end{array}$$

and then performing the contractions

$$\dfrac{\begin{array}{c}\vdots\\ st!A_1,\ldots,st!A_r \vdash !C_1 \otimes \cdots \otimes !C_t\end{array}}{!A_1,\ldots,!A_r \vdash !C_1 \otimes \cdots \otimes !C_t}\,\text{ctr}$$

Each of these cuts of prom (ψ_i) against $\otimes_j$ prom (ϕ_j) has as the final rule in the left branch a promotion and as the final rule in the right branch a right tensor introduction. The rules (Melliès 2009, Sections 3.11.1 and 3.11.2) transform the proof into a tensor product of sub-proofs κ_j where a fixed prom (ϕ_j) is cut against prom $(\psi_1),\ldots,$ prom (ψ_s) to obtain a proof $\kappa_j : s!A_1,\ldots,s!A_r \vdash !C_j$. Then $\kappa_1 \otimes \cdots \otimes \kappa_t$ is subject to the contractions as above.

Using the (*Promotion*, *Promotion*)-rule[2] (Benton et al. 1992, Section 5.2) the proof κ_j is equivalent under cut-elimination to the promotion of the proof κ'_j which results from cutting of all the prom (ψ_i) against ϕ_j. This κ'_j is by Lemma 3.4 a plain proof. We have now shown that $\phi \mid \psi$ is equivalent under cut-elimination to the tensor product over $1 \leq j \leq t$ of promotions of plain proofs $\kappa'_j : s!A_1,\ldots,s!A_r \vdash C_j$ followed by the contractions above. Hence $\phi \mid \psi$ is component-wise plain. □

3.1 Denotations

For the rest of this section suppose given a plain proof

$$\psi : !A_1,\ldots,!A_r \vdash B$$

constructed from π as in Definition 3.1. The denotation of ψ is a linear map

$$[\![\psi]\!] : ![\![A_1]\!] \otimes \cdots \otimes ![\![A_r]\!] \longrightarrow [\![B]\!]\,.$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Suppose given finite sets of proofs $\mathcal{P}_i$ of A_i and $\mathcal{Q}$ of B, such that

$$\left\{\pi(X_1,\ldots,X_r) \mid X_i \in \mathcal{P}_i^{n_i}\right\} \subseteq \mathcal{Q} \tag{9}$$

where the n_i are as in Definition 3.1. Given a set of proofs $\mathcal{N}$ we write $[\![\mathcal{N}]\!]$ for $\{[\![\nu]\!]\}_{\nu\in\mathcal{N}}$. We assume throughout that $\{[\![\nu]\!]\}_{\nu\in\mathcal{Q}}$ is linearly independent in $[\![B]\!]$. In our examples B is one of the standard datatypes and Appendix A gives us a supply of linearly independent proof denotations. Let us first of all examine the polynomials that arise in evaluating $[\![\pi]\!]$.

Given a function $\gamma : \{1,\ldots,n_i\} \longrightarrow \mathcal{P}_i$ for some i, we write

$$\rho_\gamma = \gamma(1)\otimes\cdots\otimes\gamma(n_i) : A_i^{\otimes n_i} \tag{10}$$

$$[\![\rho_\gamma]\!] = [\![\gamma(1)]\!]\otimes\cdots\otimes[\![\gamma(n_i)]\!] \in [\![A_i]\!]^{\otimes n_i}\,. \tag{11}$$

Observe that for $\lambda_\rho^{ij} \in k$, and with γ_i ranging over all functions $\{1,\ldots,n_i\} \longrightarrow \mathcal{P}_i$,

$$\begin{aligned}
[\![\pi]\!]\Big(\bigotimes_{i=1}^{r}\bigotimes_{j=1}^{n_i}\sum_{\rho\in\mathcal{P}_i}\lambda_\rho^{ij}[\![\rho]\!]\Big) &= [\![\pi]\!]\Big(\bigotimes_{i=1}^{r}\sum_{\gamma_i}\Big\{\prod_{j=1}^{n_i}\lambda_{\gamma_i(j)}^{ij}\Big\}[\![\rho_{\gamma_i}]\!]\Big)\\
&= \sum_{\gamma_1,\ldots,\gamma_r}\Big\{\prod_{i=1}^{r}\prod_{j=1}^{n_i}\lambda_{\gamma_i(j)}^{ij}\Big\}[\![\pi]\!]\big([\![\rho_{\gamma_1}]\!]\otimes\cdots\otimes[\![\rho_{\gamma_r}]\!]\big)\\
&= \sum_{\gamma_1,\ldots,\gamma_r}\Big\{\prod_{i=1}^{r}\prod_{j=1}^{n_i}\lambda_{\gamma_i(j)}^{ij}\Big\}[\![\pi\left(\rho_{\gamma_1},\ldots,\rho_{\gamma_r}\right)]\!]\\
&= \sum_{\tau\in\mathcal{Q}}\Big\{\sum_{\gamma_1,\ldots,\gamma_r}\delta_{\tau=\pi(\rho_{\gamma_1},\ldots,\rho_{\gamma_r})}\prod_{i=1}^{r}\prod_{j=1}^{n_i}\lambda_{\gamma_i(j)}^{ij}\Big\}[\![\tau]\!]\,.
\end{aligned}$$

Let us introduce variables $\{x_\rho^{ij}\}_{1\le i\le r, 1\le j\le n_i, \rho\in\mathcal{P}_i}$ so that with

$$\mathrm{Sym}\,(k\mathcal{P}_1^{n_1}\oplus\cdots\oplus k\mathcal{P}_r^{n_r}) \cong k\Big[\{x_\rho^{ij}\}_{i,j,\rho}\Big]$$

we may define an element of this algebra by:

Definition 3.7. *Given $\tau\in\mathcal{Q}$ set*

$$g_\pi^\tau := \sum_{\gamma_1,\ldots,\gamma_r}\delta_{\tau=\pi(\rho_{\gamma_1},\ldots,\rho_{\gamma_r})}\prod_{i=1}^{r}\prod_{j=1}^{n_i}x_{\gamma_i(j)}^{ij}\,.$$

In summary, we may compute $[\![\pi]\!]$ by these polynomials using the formula

$$[\![\pi]\!]\Big(\bigotimes_{i=1}^{r}\bigotimes_{j=1}^{n_i}\sum_{\rho\in\mathcal{P}_i}\lambda_\rho^{ij}[\![\rho]\!]\Big) = \sum_{\tau\in\mathcal{Q}} g_\pi^\tau\Big|_{x_\rho^{ij}=\lambda_\rho^{ij}}[\![\tau]\!]\,. \tag{12}$$

Now let us turn to calculating $[\![\psi]\!]$ using the morphism of k-algebras

$$C : \mathrm{Sym}\,\Big(\bigoplus_{i=1}^{r}k\mathcal{P}_i^{n_i}\Big) \cong k\Big[\{x_\rho^{ij}\}_{i,j,\rho}\Big] \longrightarrow k\Big[\{x_\rho^{i}\}_{i,j,\rho}\Big] \cong \mathrm{Sym}\,\Big(\bigoplus_{i=1}^{r}k\mathcal{P}_i\Big) \tag{13}$$

$$x_\rho^{ij} \mapsto x_\rho^i\,.$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

With $\omega_i = \sum_{\rho \in \mathcal{P}_i} \lambda^i_\rho [\![\rho]\!]$, we calculate

$$\begin{aligned}
[\![\psi]\!]\Big(|\emptyset\rangle_{\omega_1} \otimes \cdots \otimes |\emptyset\rangle_{\omega_r}\Big) &= [\![\pi]\!]\Big(\bigotimes_{i=1}^{r} d^{\otimes n_i} \Delta^{n_i - 1} |\emptyset\rangle_{\omega_i}\Big) \\
&= [\![\pi]\!]\Big(\bigotimes_{i=1}^{r} d^{\otimes n_i} |\emptyset\rangle_{\omega_i}^{\otimes n_i}\Big) \\
&= [\![\pi]\!]\Big(\bigotimes_{i=1}^{r} \omega_i^{\otimes n_i}\Big) \\
&= \sum_{\tau \in \mathcal{Q}} C(g^\tau_\pi)\Big|_{x^i_\rho = \lambda^i_\rho} [\![\tau]\!] \,. \qquad (14)
\end{aligned}$$

Let ι denote the function

$$\iota : \prod_{i=1}^{r} k\mathcal{P}_i \longrightarrow \bigotimes_{i=1}^{r} ![\![A_i]\!] \,,$$
$$\iota(\omega_1, \ldots, \omega_r) = \bigotimes_{i=1}^{r} |\emptyset\rangle_{[\![\omega_i]\!]}$$

where $k\mathcal{P}$ is the free vector space on $\mathcal{P}$.

Proposition 3.8. *There is a unique function F_ψ making the diagram*

$$\begin{array}{ccc}
![\![A_1]\!] \otimes \cdots \otimes ![\![A_r]\!] & \xrightarrow{\quad [\![\psi]\!] \quad} & [\![B]\!] \\
\uparrow{\scriptstyle \iota} & & \uparrow{\scriptstyle [\![-]\!]} \\
k\mathcal{P}_1 \times \cdots \times k\mathcal{P}_r & \xrightarrow[\quad F_\psi \quad]{} & k\mathcal{Q}
\end{array}$$

commute. Moreover this function is computed by a polynomial, in the sense that it is induced by a morphism of k-algebras

$$f_\psi : \mathrm{Sym}\,(k\mathcal{Q}) \longrightarrow \mathrm{Sym}\,(k\mathcal{P}_1 \oplus \cdots \oplus k\mathcal{P}_r) \,.$$

More precisely, if we present the symmetric algebras as polynomial rings in variables

$$\{y_\tau\}_{\tau \in \mathcal{Q}} \,, \qquad \{x^i_\rho\}_{1 \le i \le r, \rho \in \mathcal{P}_i}$$

respectively, then the polynomial $f^\tau_\psi := f_\psi(y_\tau)$ is given by the formula

$$f^\tau_\psi = \sum_{\gamma_1, \ldots, \gamma_r} \delta_{\tau = \pi(\rho_{\gamma_1}, \ldots, \rho_{\gamma_r})} \prod_{i=1}^{r} \prod_{j=1}^{n_i} x^i_{\gamma_i(j)} \,, \qquad (15)$$

where γ_i ranges over all functions $\{1, \ldots, n_i\} \longrightarrow \mathcal{P}_i$ and we use the notation of (10).

Proof. Since by hypothesis $\{[\![\nu]\!]\}_{\nu \in \mathcal{Q}}$ is linearly independent, the right-hand vertical map (from (7)) is injective and so the map F_ψ is unique if it exists. Existence follows from (14), and moreover this also shows that $F_\psi(\mathbf{a})_\tau = \mathrm{eval}_{x^i_\rho = a^i_\rho}(f^\tau_\pi)$ for all $\mathbf{a}$ in $\prod_i k\mathcal{P}_i$. □

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

4. The Girard Encoding

To fix the notation, we briefly recall the definition of a Turing machine from Arora and Barak (2009) and Sipser (2006).

Definition 4.1. *A* Turing machine $M = (\Sigma, Q, \delta)$ *is a tuple where* Q *is a finite set of states,* Σ *is a finite set of symbols called the* tape alphabet *and*

$$\delta : \Sigma \times Q \to \Sigma \times Q \times \{\mathit{left}, \mathit{right}\}$$

is a function, called the transition function.

The set Σ is assumed to contain some designated blank symbol $\square$ which is the only symbol that is allowed to occur infinitely often on the tape. If M is a Turing machine, a *Turing configuration* of M is a tuple $\langle S, T, q\rangle$, where $S, T \in \Sigma^*$ and $q \in Q$. This is interpreted as the instantaneous configuration of the Turing machine in the following way. The string S corresponds to the non-blank contents of the tape to the left of the tape head, including the symbol currently being scanned. The string T corresponds to a *reversed* copy of the contents of the tape to the right of the tape head, and q stores the current state of the machine. The reason for T being reversed is a matter of convenience, as we will see in the next section. The *step function*

$${}_\delta\mathrm{step} : \Sigma^* \times \Sigma^* \times Q \longrightarrow \Sigma^* \times \Sigma^* \times Q$$

sends the current configuration $\langle S, T, q\rangle$ to the configuration ${}_\delta\mathrm{step}(S, T, q)$ after one step.

The eventual goal of this section will be to present a method of encoding of Turing machines in linear logic. This is heavily based on the work by Girard (1995), which encodes Turing configurations via a variant of second-order linear logic called *light* linear logic. The encoding does not use light linear logic in a crucial way, but requires second order in many intermediate steps, making it incompatible with differentiation. We modify this encoding so that it is able to be differentiated (see Remark 4.15), while also filling in some of the details omitted from Girard (1995).

Definition 4.2. *Fix a finite set of states* $Q = \{0, ..., n-1\}$*, and a tape alphabet*[3] $\Sigma = \{0, 1\}$*, with* 0 *being the blank symbol. The type of* Turing configurations *on* A *is:*

$$\boldsymbol{Tur}_A = !\boldsymbol{bint}_A \otimes !\boldsymbol{bint}_A \otimes !_n\boldsymbol{bool}_A.$$

The configuration $\langle S, T, q\rangle$ *is represented by the element*

$$[\![\langle S, T, q\rangle]\!] = |\emptyset\rangle_{[\![\underline{S}_A]\!]} \otimes |\emptyset\rangle_{[\![\underline{T}_A]\!]} \otimes |\emptyset\rangle_{[\![\underline{q}_A]\!]} \in [\![\boldsymbol{Tur}_A]\!].$$

Our aim is to simulate a single transition step of a given Turing machine M as a proof ${}_\delta\underline{\mathrm{step}}_A$ of $\mathbf{Tur}_B \vdash \mathbf{Tur}_A$ for some formula B which depends on A, in the sense that if said proof is cut against a Turing configuration of M at time t, the result will be equivalent under cut-elimination to the Turing configuration of M at time step $t+1$. This will be achieved in Theorem 4.13. Following Girard (1995) the strategy will be as follows. Let $\langle S\sigma, T\tau, q\rangle$ be the (initial) configuration of the given Turing machine.

(1) Decompose the binary integers $S\sigma$ and $T\tau$ to extract their final digits, giving S, T, σ and τ. Note that σ is the symbol currently under the head, and τ is the symbol immediately to its right.
(2) Using the symbol σ together with the current state $q \in Q$, compute the new symbol σ', the new state q' and the direction to move d.
(3) If $d =$ right, append $\sigma'\tau$ to S. Otherwise, append $\tau\sigma'$ to T; remember that the binary integer T is the *reversal* of the contents of the tape to the right of the tape head. This is summarised in Figure 1.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

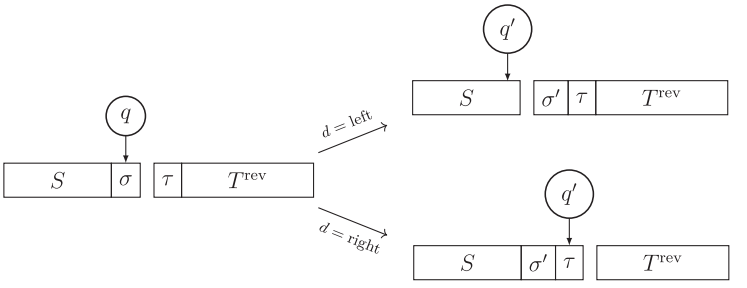


Figure 1. A single transition step of a Turing machine.

For simplicity we present the encoding where the head of the Turing machine must either move left or right at each time step, and leave to the reader the minor changes necessary to allow the head to also remain stationary.

4.1 The encoding

In order to feed the current symbol into the transition function, it is necessary to extract this digit from the binary integer which represents the tape. To do this we must decompose a binary integer $S' = S\sigma$ of length $l \geq 1$ into two parts S and σ, the former being a **bint** consisting of the first $l - 1$ digits (the *tail*) and the latter being a **bool** corresponding to the final digit (the *head*).

Proposition 4.3. *There exists a proof* $\underline{\text{head}}_A$ *of* $\boldsymbol{bint}_{A^3} \vdash \boldsymbol{bool}_A$ *which encodes the function* $[\![\underline{S\sigma}_{A^3}]\!] \mapsto [\![\underline{\sigma}_A]\!]$.

Proof. The construction we will use is similar to that in Girard (1995, Section 2.5.3). Let π_0, π_1 be the (easily constructed) proofs of $A^3 \vdash A^3$ whose denotations are $[\![\pi_0]\!](x, y, z) = (x, y, x)$ and $[\![\pi_1]\!](x, y, z) = (x, y, y)$ respectively. Similarly let ρ be the proof of $A^2 \vdash A^3$ with denotation $[\![\rho]\!](x, y) = (x, y, x)$. Define by $\underline{\text{head}}_A$ the following proof:

$$\dfrac{\dfrac{\dfrac{\pi_0}{\dfrac{A^3 \vdash A^3}{\vdash A^3 \multimap A^3}\multimap R}\text{prom}}{\vdash\, !(A^3 \multimap A^3)} \qquad \dfrac{\dfrac{\pi_1}{\dfrac{A^3 \vdash A^3}{\vdash A^3 \multimap A^3}\multimap R}\text{prom} \qquad \dfrac{\rho \qquad \dfrac{\dfrac{}{A \vdash A}}{A^3 \vdash A}\&L_2}{A^2 \vdash A^3}\multimap L}{\vdash\, !(A^3 \multimap A^3) \qquad A^3 \multimap A^3, A^2 \vdash A}\multimap R}{A^3 \multimap A^3 \vdash \mathbf{bool}_A}\multimap L}{\mathbf{int}_{A^3} \vdash \mathbf{bool}_A}\multimap L}{\mathbf{bint}_{A^3} \vdash \mathbf{bool}_A}\multimap L$$

where the rule $\&L_2$ introduces two new copies of A on the left, such that the original copy is at position 2 (that is, the third element of the triple).

We now show that $[\![\underline{\text{head}}_A]\!]([\![\underline{S\sigma}_{A^3}]\!]) = [\![\underline{\sigma}_A]\!]$ as claimed. Recall that the denotation $[\![\underline{S\sigma}_{A^3}]\!]$ of a binary integer is a function which, given inputs α and β of type $A^3 \multimap A^3$ corresponding to the digits zero and one, returns some composite of α and β. The effect of the two leftmost branches of $\underline{\text{head}}_A$ is to substitute $[\![\pi_0]\!]$ for α and $[\![\pi_1]\!]$ for β in this composite, giving a linear map $\varphi : [\![A^3]\!] \to [\![A^3]\!]$. The rightmost branch then computes $\text{proj}_2 \circ \varphi \circ [\![\rho]\!] : [\![A^2]\!] \to [\![A]\!]$, giving a boolean.

In other words, $[\![\underline{\text{head}}_A]\!]([\![\underline{S\sigma}_{A^3}]\!])$ is the element of $[\![\mathbf{bool}_A]\!]$ given by:

$$(a_0, a_1) \mapsto \text{proj}_2 \circ \varphi \circ [\![\rho]\!](a_0, a_1) = \text{proj}_2 \circ \varphi(a_0, a_1, a_0),$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

where φ is the composite of $[\![\pi_0]\!]$ and $[\![\pi_1]\!]$ as above. Note, however, that repeated applications of the functions $[\![\pi_i]\!]$ only serve to update the final digit of the triple, and thus only the final copy of $[\![\pi_i]\!]$ determines the output value. Hence the above simplifies to

$$\mathrm{proj}_2 \circ \varphi(a_0, a_1, a_0) = \mathrm{proj}_2 \circ [\![\pi_\sigma]\!](a_0, a_1, a_0) = \mathrm{proj}_2\,(a_0, a_1, a_\sigma) = a_\sigma.$$

Thus $[\![\underline{\mathrm{head}}_A]\!]([\![\underline{S\sigma}_{A^3}]\!]) = \mathrm{proj}_\sigma$, which is indeed the boolean corresponding to σ.

Lastly, we consider the special case when $S\sigma$ is the empty list. In this case

$$\mathrm{proj}_2 [\![\rho]\!](a_0, a_1) = \mathrm{proj}_2\,(a_0, a_1, a_0) = a_0$$

which captures the fact that any symbols outside the working section of the tape are assumed to be the blank symbol, 0. □

Proposition 4.4. *There exists a proof* $\underline{\mathit{tail}}_A$ *of* $\boldsymbol{bint}_{A^3} \vdash \boldsymbol{bint}_A$ *which encodes the function* $[\![\underline{S\sigma}_{A^3}]\!] \mapsto [\![\underline{S}_A]\!]$.

Remark 4.5. This could also be encoded as a proof of $\mathbf{bint}_{A^2} \vdash \mathbf{bint}_A$. However, it will be much more convenient later if the sequents proven by $\underline{\mathrm{head}}_A$ and $\underline{\mathrm{tail}}_A$ have the same premise, since we will need to apply them both to two copies of the same binary integer.

Proof. This is largely based on the predecessor for $\mathbf{int}_A$. Define π to be the proof

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A \vdash A^3}\,\&R \qquad \dfrac{\dfrac{}{A \vdash A}}{A^3 \vdash A}\,\&L_0}{A, A^3 \multimap A^3 \vdash A}\,{\multimap}L}{A^3 \multimap A^3 \vdash A \multimap A}\,{\multimap}R$$

which has denotation:

$$[\![\pi]\!](\varphi)(a) = \mathrm{proj}_0\,(\varphi(a, a, a)).$$

Define ρ to be the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}}{A^3 \vdash A}\,\&L_2}{A^3, !(A \multimap A) \vdash A}\,\mathrm{weak} \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A}}{A^3 \vdash A}\,\&L_2}{A^3, !(A \multimap A) \vdash A}\,\mathrm{weak} \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\multimap}L}{A^3, A \multimap A \vdash A}\,\&L_2}{A^3, !(A \multimap A) \vdash A}\,\mathrm{der}}{A^3, !(A \multimap A) \vdash A^3}\,\&R}{!(A \multimap A) \vdash A^3 \multimap A^3}\,{\multimap}R$$

The denotation $[\![\rho]\!]$ is

$$[\![\rho]\!](|\alpha_1, ..., \alpha_s\rangle_\gamma)(a_0, a_1, a_2) = \begin{cases} (a_2, a_2, \gamma a_2) & s = 0 \\ (a_2, a_2, \alpha_1 a_2) & s = 1 \\ (a_2, a_2, 0) & s > 1. \end{cases}$$

Finally, define $\underline{\mathrm{tail}}_A$ to be the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\rho}{\vdots}}{!(A \multimap A) \vdash A^3 \multimap A^3}\,\mathrm{prom}}{!(A \multimap A) \vdash !(A^3 \multimap A^3)} \qquad \dfrac{\dfrac{\dfrac{\rho}{\vdots}}{!(A \multimap A) \vdash A^3 \multimap A^3}\,\mathrm{prom}}{!(A \multimap A) \vdash !(A^3 \multimap A^3)} \qquad \dfrac{\dfrac{\pi}{\vdots}}{A^3 \multimap A^3 \vdash A \multimap A}\,{\multimap}L}{!(A \multimap A), \mathbf{bint}_{A^3} \vdash A \multimap A}\,{\multimap}L}{!(A \multimap A), !(A \multimap A), \mathbf{bint}_{A^3} \vdash A \multimap A}\,2\times{\multimap}R}{\mathbf{bint}_{A^3} \vdash \mathbf{bint}_A}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Evaluated on the binary integer S, this gives a binary integer T which if fed two vacuum vectors $|\emptyset\rangle_\gamma$ and $|\emptyset\rangle_\delta$ (corresponding to the digits 0, 1) will return the composite $[\![A]\!] \to [\![A]\!]$ obtained by substituting $[\![\rho]\!]|\emptyset\rangle_\gamma$ and $[\![\rho]\!]|\emptyset\rangle_\delta$ for each copy of the digits 0 and 1 respectively in S, and then finally keeping the 0th projection by the left introduction of π.

As an example, suppose that the binary integer S is 0010. Then the corresponding linear map $[\![A]\!] \to [\![A]\!]$ is

$$a \mapsto \mathrm{proj}_0\,(\tilde{\gamma} \circ \tilde{\delta} \circ \tilde{\gamma} \circ \tilde{\gamma}(a, a, a))$$

where $\tilde{\gamma} = [\![\rho]\!]|\emptyset\rangle_\gamma$, which is the morphism $(a_0, a_1, a_2) \mapsto (a_2, a_2, \gamma a_2)$, and similarly for $\tilde{\delta}$. Thus, we have:

$$\begin{aligned}
\mathrm{proj}_0\,(\tilde{\gamma} \circ \tilde{\delta} \circ \tilde{\gamma} \circ \tilde{\gamma}(a, a, a)) &= \mathrm{proj}_0\,(\tilde{\gamma} \circ \tilde{\delta} \circ \tilde{\gamma}(a, a, \gamma(a))) \\
&= \mathrm{proj}_0\,(\tilde{\gamma} \circ \tilde{\delta}(\gamma(a), \gamma(a), \gamma\gamma(a))) \\
&= \mathrm{proj}_0\,(\tilde{\gamma}(\gamma\gamma(a), \gamma\gamma(a), \delta\gamma\gamma(a))) \\
&= \mathrm{proj}_0\,(\delta\gamma\gamma(a), \delta\gamma\gamma(a), \gamma\delta\gamma\gamma(a)) \\
&= \delta\gamma\gamma(a). \qquad \square
\end{aligned}$$

When fed through the decomposition steps, the base type of the binary integers changes from A^3 to A. We therefore also need to modify the base type of the n-boolean representing the state, in order to keep the base types compatible.

Lemma 4.6. *There exists a proof ${}_n\underline{\mathrm{booltype}}_A$ of ${}_n\mathbf{bool}_{A^3} \vdash {}_n\mathbf{bool}_A$ which converts an n-boolean on A^3 to the equivalent n-boolean on A; that is, it encodes $[\![\underline{i}_{A^3}]\!] \mapsto [\![\underline{i}_A]\!]$.*

Proof. For $i \in \{0, ..., n-1\}$, let π_i be the proof of $A^n \vdash A^3$ whose denotation is $(a_0, ..., a_{n-1}) \mapsto (a_i, a_i, a_i)$. Define ${}_n\underline{\mathrm{booltype}}_A$ as the proof:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi_0 \\ \vdots \\ A^n \vdash A^3\end{matrix} \quad \cdots \quad \begin{matrix}\pi_{n-1} \\ \vdots \\ A^n \vdash A^3\end{matrix}}{A^n \vdash (A^3)^n}\,\&R \qquad \dfrac{A \vdash A}{A^3 \vdash A}\,\&L_0}{A^n, {}_n\mathbf{bool}_{A^3} \vdash A}\,{\multimap}L}{{}_n\mathbf{bool}_{A^3} \vdash {}_n\mathbf{bool}_A}\,{\multimap}R$$

The denotation of ${}_n\underline{\mathrm{booltype}}_A$ is the function

$$[\![{}_n\underline{\mathrm{booltype}}_A]\!](\varphi)(a_0, ..., a_{n-1}) = \mathrm{proj}_0 \circ \varphi((a_0, a_0, a_0), ..., (a_{n-1}, a_{n-1}, a_{n-1})),$$

and hence $[\![{}_n\underline{\mathrm{booltype}}_A]\!]([\![\underline{i}_{A^3}]\!]) = [\![\underline{i}_A]\!]$. $\square$

We next encode the transition function $\delta : \Sigma \times Q \to \Sigma \times Q \times \{\mathrm{left}, \mathrm{right}\}$ of a given Turing machine.

Lemma 4.7. *Given any function $f : \{0, ..., n-1\} \to \{0, ..., m-1\}$, there exists a proof F of ${}_n\mathbf{bool}_A \vdash {}_m\mathbf{bool}_A$ which encodes f.*

Proof. Let F be the following proof:

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}}{A^m \vdash A}\,{\&L_{f(1)}} \quad \ldots \quad \dfrac{\dfrac{}{A \vdash A}}{A^m \vdash A}\,{\&L_{f(n)}}}{A^m \vdash A^n}\,{\&R} \qquad \dfrac{}{A \vdash A}}{\dfrac{A^m, {}_n\mathbf{bool}_A \vdash A}{{}_n\mathbf{bool}_A \vdash {}_m\mathbf{bool}_A}\,{\multimap R}}\,{\multimap L}$$

The denotation of F is, for $\varphi \in [\![{}_n\mathbf{bool}_A]\!]$ and $(a_0, \ldots, a_{m-1}) \in [\![A^m]\!]$:

$$[\![F]\!](\varphi)(a_0, \ldots, a_{m-1}) = \varphi(a_{f(0)}, \ldots, a_{f(n-1)}).$$

In particular, this means that $[\![F]\!]([\![\underline{i}_A]\!])(a_0, \ldots, a_{m-1}) = \mathrm{proj}_i\,(a_{f(0)}, \ldots, a_{f(n-1)}) = a_{f(i)}$, and hence $[\![F]\!]([\![\underline{i}_A]\!]) = \mathrm{proj}_{f(i)} = [\![\underline{f(i)}_A]\!]$ as desired. □

Proposition 4.8. *Given a transition function $\delta : \Sigma \times Q \to \Sigma \times Q \times \{left, right\}$, write δ_i for the component $\mathrm{proj}_i \circ \delta$ $(i \in \{0, 1, 2\})$. Then there exists proofs*

$$\begin{aligned}
&{}^0_\delta\underline{trans}_A : \mathbf{bool}_A, {}_n\mathbf{bool}_A \vdash \mathbf{bool}_A\\
&{}^1_\delta\underline{trans}_A : \mathbf{bool}_A, {}_n\mathbf{bool}_A \vdash {}_n\mathbf{bool}_A\\
&{}^2_\delta\underline{trans}_A : \mathbf{bool}_A, {}_n\mathbf{bool}_A \vdash \mathbf{bool}_A
\end{aligned}$$

which encode δ_i, for $i = 0, 1, 2$. We are using the convention that left $= 0$ and right $= 1$.

Proof. Given $i \in \{0, 1, 2\}$ and $j \in \Sigma = \{0, 1\}$, let $\Delta_{i,j}$ be the proof obtained from Lemma 4.7 corresponding to the function $\delta_i(j, -)$, omitting the final $\multimap R$ rule. Define ${}^i_\delta\underline{\mathrm{trans}}_A$ as the following proof, where $m = n$ if $i = 1$ and $m = 2$ otherwise:

$$\dfrac{\dfrac{\begin{array}{c}\Delta_{i,0}\\ \vdots\end{array} \qquad \begin{array}{c}\Delta_{i,1}\\ \vdots\end{array}}{A^m, {}_n\mathbf{bool}_A \vdash A \qquad A^m, {}_n\mathbf{bool}_A \vdash A}\,{\&R}}{A^m, {}_n\mathbf{bool}_A \vdash A^2} \qquad \dfrac{}{A \vdash A}}{\dfrac{A^m, \mathbf{bool}_A, {}_n\mathbf{bool}_A \vdash A}{\mathbf{bool}_A, {}_n\mathbf{bool}_A \vdash {}_m\mathbf{bool}_A}\,{\multimap L}}\,{\multimap L}$$

Then $[\![{}^i_\delta\underline{\mathrm{trans}}_A]\!]$ is the function

$$[\![{}^i_\delta\underline{\mathrm{trans}}_A]\!](\psi \otimes \varphi)(a_0, \ldots, a_{m-1}) = \psi(\varphi(a_{\delta_i(0,0)}, \ldots, a_{\delta_i(0,n-1)}), \varphi(a_{\delta_i(1,0)}, \ldots, a_{\delta_i(1,n-1)})),$$

and thus we have $[\![{}^i_\delta\underline{\mathrm{trans}}_A]\!]([\![\underline{\sigma}]\!] \otimes [\![\underline{q}]\!]) = \mathrm{proj}_{\delta_i(\sigma,q)} = [\![\underline{\delta_i(\sigma, q)}]\!]$. □

Once the new state, symbol and direction have been computed, our remaining task is to recombine the symbols with the binary integers representing the tape.

Example 4.9. Let $E = A \multimap A$. The proof $\underline{\mathrm{concat}}_A$ is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{!E \vdash !E}}{!E, !E, !E, !E, \mathbf{bint}_A, \mathbf{bint}_A \vdash E}\,{2\times \mathrm{ctr}}}{!E, !E, \mathbf{bint}_A, \mathbf{bint}_A \vdash E}\,{2\times \multimap R}}{\mathbf{bint}_A, \mathbf{bint}_A \vdash \mathbf{bint}_A}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{!E \vdash !E}}{!E, !E, !E, \mathbf{int}_A, \mathbf{bint}_A \vdash E}\,{\multimap L}}{!E, !E, !E, \mathbf{bint}_A \vdash E}\,{\multimap L}}{!E, !E, E, \mathbf{bint}_A \vdash E}\,{\multimap L}}{!E, E, \mathbf{int}_A \vdash E}\,{\multimap L}}{E, E \vdash E}\,{\multimap L}}{A, E, E \vdash A}\,{\multimap R}}{\underline{\mathrm{comp}}^2_A}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

where $\underline{\mathrm{comp}}^2_A$ encodes composition (Clift and Murfet 2017, Definition 4.2). The colours indicate which copies of $!E$ are contracted together. When cut against the proofs of binary integers $\underline{S}_A$ and $\underline{T}_A$, the resulting proof will be equivalent under cut-elimination to $\underline{ST}_A$. We write $\underline{\mathrm{concat}}(S, -)$ for the proof of $\mathbf{bint}_A \vdash \mathbf{bint}_A$ obtained by cutting a binary integer $\underline{S}_A$ against $\underline{\mathrm{concat}}_A$ such that the first $\mathbf{bint}_A$ is consumed, meaning that $\underline{\mathrm{concat}}(S, -)$ prepends by S. Similarly define $\underline{\mathrm{concat}}(-, T)$ as the proof which appends by T.

Lemma 4.10. *Let $W_{00}, W_{01}, W_{10}, W_{11}$ be fixed binary integers, possibly empty. There exists a proof $\pi(W_{00}, W_{01}, W_{10}, W_{11})$ of $\boldsymbol{bint}_A, \boldsymbol{bool}_A, \boldsymbol{bool}_A \vdash \boldsymbol{bint}_A$ which encodes*

$$(S, \sigma, \tau) \mapsto SW_{\sigma\tau}.$$

Proof. Let $E = A \multimap A$. We give a proof corresponding to the simpler function $(S, \sigma) \mapsto SW_\sigma$, where W_0, W_1 are fixed binary sequences:

$$\dfrac{\dfrac{\dfrac{\underline{\mathrm{concat}}_A(-, W_0)}{\vdots}\quad \dfrac{\underline{\mathrm{concat}}_A(-, W_1)}{\vdots}}{\mathbf{bint}_A, !E, !E, A \vdash A \quad \mathbf{bint}_A, !E, !E, A \vdash A}\,\&R \qquad \dfrac{}{A \vdash A}}{\mathbf{bint}_A, !E, !E, A \vdash A \& A}\,{\multimap}L}{\mathbf{bint}_A, \mathbf{bool}_A, !E, !E, A \vdash A}\,3\times {\multimap}R}{\mathbf{bint}_A, \mathbf{bool}_A \vdash \mathbf{bint}_A}$$

The required proof $\pi(W_{00}, W_{01}, W_{10}, W_{11})$ is an easy extension of this, involving two instances of the $\&R$ and $\multimap L$ rules rather than one. □

Proposition 4.11. *There exist proofs ${}^0\underline{\mathrm{recomb}}_A$ and ${}^1\underline{\mathrm{recomb}}_A$ of*

$$\boldsymbol{bint}_A, 3\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}_A$$

which encode the functions

$$(S, \tau, \sigma, d) \mapsto \begin{cases} S & \text{if } d = 0 \text{ (left)} \\ S\sigma\tau & \text{if } d = 1 \text{ (right)} \end{cases} \quad \text{and} \quad (T, \tau, \sigma, d) \mapsto \begin{cases} T\tau\sigma & \text{if } d = 0 \text{ (left)} \\ T & \text{if } d = 1 \text{ (right)} \end{cases}$$

respectively.

Proof. Define $\pi(-, -, -, -)$ as described in Lemma 4.10, omitting the final $\multimap R$ rules. The desired proof ${}^0\underline{\mathrm{recomb}}_A$ is:

$$\dfrac{\dfrac{\dfrac{\pi(\emptyset, \emptyset, \emptyset, \emptyset)}{\vdots}\quad \dfrac{\pi(00, 10, 01, 11)}{\vdots}}{\mathbf{bint}_A, 2\,\mathbf{bool}_A, !E, !E, A \vdash A \quad \mathbf{bint}_A, 2\,\mathbf{bool}_A, !E, !E, A \vdash A}\,\&R \qquad \dfrac{}{A \vdash A}}{\mathbf{bint}_A, 2\,\mathbf{bool}_A, !E, !E, A \vdash A \& A}\,{\multimap}L}{\mathbf{bint}_A, 3\,\mathbf{bool}_A, !E, !E, A \vdash A}\,3\times {\multimap}R}{\mathbf{bint}_A, 3\,\mathbf{bool}_A \vdash \mathbf{bint}_A}$$

and ${}^1\underline{\mathrm{recomb}}_A$ is the same, with the leftmost branch replaced by $\pi(00, 01, 10, 11)$ and the second branch replaced by $\pi(\emptyset, \emptyset, \emptyset, \emptyset)$. □

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Proposition 4.12. *There exist proofs*

$$
\begin{aligned}
{}_\delta\underline{left}_A &: 3\,\boldsymbol{bint}_{A^3},\quad \boldsymbol{bint}_{A^3}, 2_n\boldsymbol{bool}_{A^3} \vdash \boldsymbol{bint}_A\\
{}_\delta\underline{right}_A &: 2\,\boldsymbol{bint}_{A^3}, 2\,\boldsymbol{bint}_{A^3}, 2_n\boldsymbol{bool}_{A^3} \vdash \boldsymbol{bint}_A\\
{}_\delta\underline{state}_A &: \quad \boldsymbol{bint}_{A^3}, \qquad\qquad {}_n\boldsymbol{bool}_{A^3} \vdash {}_n\boldsymbol{bool}_A
\end{aligned}
$$

which, if fed the indicated number of copies of S, T and q corresponding to a Turing configuration, update the left part of the tape, the right part of the tape and the state.

Proof. We simply compose (using cuts) the proofs from Propositions 4.3 through 4.11; the exact sequence of cuts is given in Figures 2, 3 and 4. The verification that the proofs perform the desired tasks is made clear through the following informal computations. Here $\langle S\sigma, T\tau, q\rangle$ is the configuration of the Turing machine at time t, and $\langle S', T', q'\rangle$ is its configuration at time $t+1$. In other words, we have $\delta(\sigma, q) = (\sigma', q', d)$, and

$$
(S', T') = \begin{cases} (S, T\tau\sigma') & d = 0 \text{ (left)}\\ (S\sigma'\tau, T) & d = 1 \text{ (right).}\end{cases}
$$

$$
\begin{aligned}
{}_\delta\underline{\text{left}}_A \text{ is :}\quad & (S\sigma)^{\otimes 3} \otimes (T\tau) \otimes q^{\otimes 2} & \\
\longmapsto\ & S \otimes \sigma^{\otimes 2} \otimes \tau \otimes q^{\otimes 2} & (\underline{\text{tail}}_A \otimes \underline{\text{head}}_A^{\otimes 3} \otimes {}_n\underline{\text{booltype}}_A^{\otimes 2})\\
\longmapsto\ & S \otimes \tau \otimes (\sigma \otimes q)^{\otimes 2} & (\text{exchange})\\
\longmapsto\ & S \otimes \tau \otimes \sigma' \otimes d & (\text{id}^{\otimes 2} \otimes {}^0_\delta\underline{\text{trans}}_A \otimes {}^2_\delta\underline{\text{trans}}_A)\\
\longmapsto\ & S' & ({}^0\underline{\text{recomb}}_A)
\end{aligned}
$$

$$
\begin{aligned}
{}_\delta\underline{\text{right}}_A \text{ is :}\quad & (S\sigma)^{\otimes 2} \otimes (T\tau)^{\otimes 2} \otimes q^{\otimes 2} & \\
\longmapsto\ & \sigma^{\otimes 2} \otimes \tau \otimes T \otimes q^{\otimes 2} & (\underline{\text{head}}_A^{\otimes 3} \otimes \underline{\text{tail}}_A \otimes {}_n\underline{\text{booltype}}_A^{\otimes 2})\\
\longmapsto\ & T \otimes \tau \otimes (\sigma \otimes q)^{\otimes 2} & (\text{exchange})\\
\longmapsto\ & T \otimes \tau \otimes \sigma' \otimes d & (\text{id}^{\otimes 2} \otimes {}^0_\delta\underline{\text{trans}}_A \otimes {}^2_\delta\underline{\text{trans}}_A)\\
\longmapsto\ & T' & ({}^1\underline{\text{recomb}}_A)
\end{aligned}
$$

$$
\begin{aligned}
{}_\delta\underline{\text{state}}_A \text{ is :}\quad & (S\sigma) \otimes q & \\
\longmapsto\ & \sigma \otimes q & (\underline{\text{head}}_A \otimes {}_n\underline{\text{booltype}}_A)\\
\longmapsto\ & q'. & ({}^1_\delta\underline{\text{trans}}_A)
\end{aligned}
$$

□

Theorem 4.13. *There exists a proof ${}_\delta\underline{step}_A$ of $\boldsymbol{Tur}_{A^3} \vdash \boldsymbol{Tur}_A$ which encodes a single transition step of a given Turing machine.*

Proof. The desired proof ${}_\delta\underline{\text{step}}_A$ is given in Figure 5. □

By cutting the above construction against itself, we obtain the following:

Corollary 4.14. *For each $p \geq 1$, there exists a proof ${}^p_\delta\underline{step}_A$ of $\boldsymbol{Tur}_{A^{3^p}} \vdash \boldsymbol{Tur}_A$ which encodes p transition steps of a given Turing machine.*

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

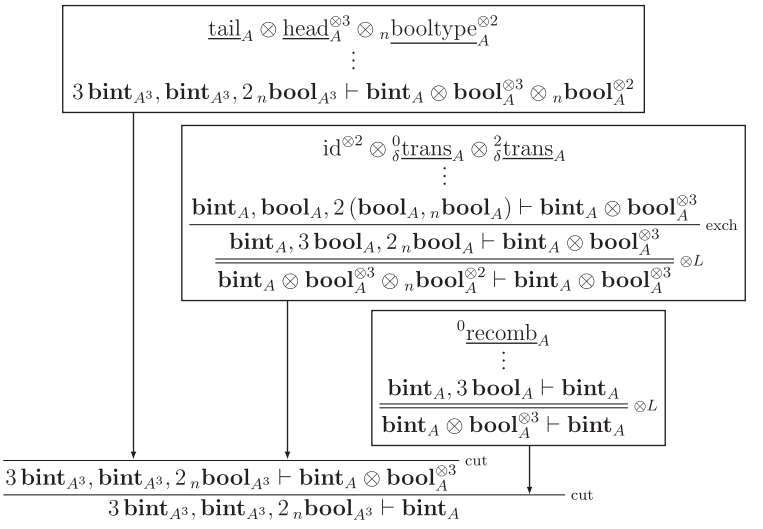


Figure 2. The proof ${}_\delta\underline{\mathrm{left}}_A$.

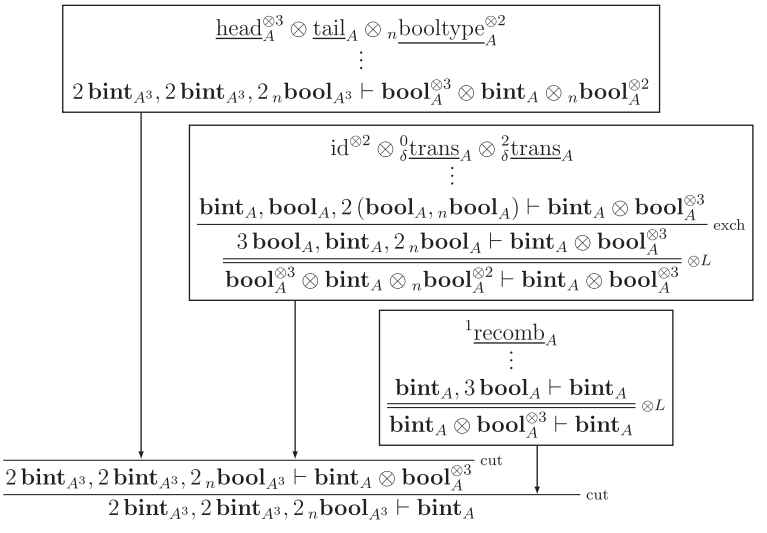


Figure 3. The proof ${}_\delta\underline{\mathrm{right}}_A$.

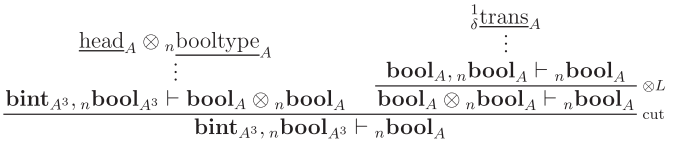


Figure 4. The proof ${}_\delta\underline{\mathrm{state}}_A$.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

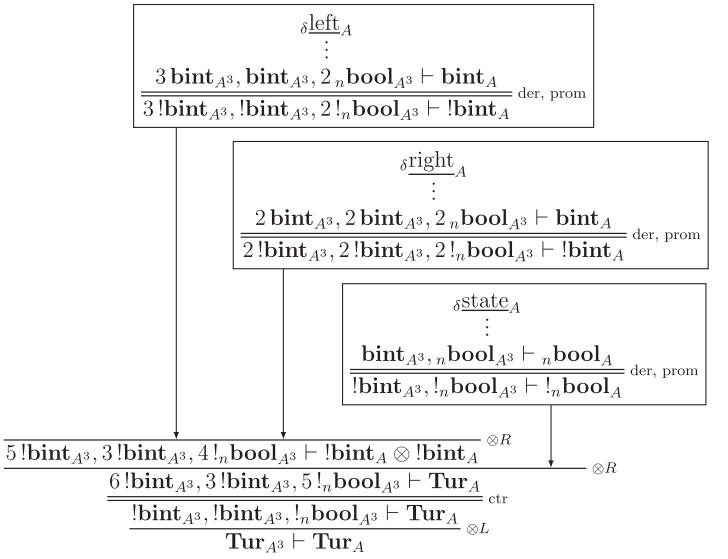
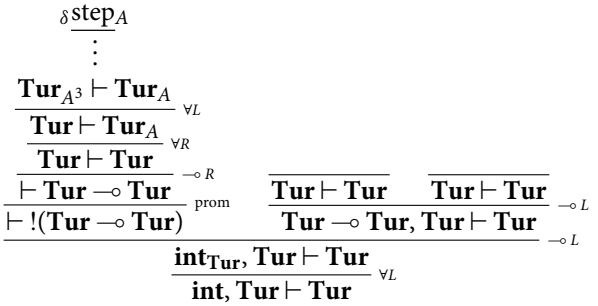


Figure 5. The proof $_\delta\underline{\text{step}}_A$.

Note that this iteration must be performed 'by hand' for each p; we cannot iterate for a variable number of steps. By this we mean that it is not possible to devise a proof of $\mathbf{int}_B, \mathbf{Tur}_C \vdash \mathbf{Tur}_A$ (for suitable types B, C) which simulates a given Turing machine for n steps when cut against the Church numeral $\underline{n}_B$. The fundamental problem is that iteration using $\mathbf{int}_B$ only allows iteration of *endomorphisms* $B \multimap B$, and so the fact that our base type changes in each iteration of $_\delta\underline{\text{step}}_A$ makes this impossible.

Remark 4.15. If one is willing to use second order, then iteration in the above sense becomes possible via the following proof, where $\mathbf{Tur} = \forall A.\mathbf{Tur}_A$:



In the original encoding given by Girard (1995), the use of second-order quantifiers is present throughout the encoding. In contrast, the above encoding only involves the $\forall R, \forall L$ rules at the very bottom of the proof tree, which makes it more suitable for the study of Turing machines in the context of differential linear logic in Clift and Murfet (2018).

Next we observe that $_\delta\underline{\text{step}}_A$ is component-wise plain in the sense of Definition 3.5.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Definition 4.16. *We write* $_\delta\underline{left}^\dagger_A$ *for the proof*

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}{}_\delta\underline{left}_A\\ \vdots\\ 3\,\boldsymbol{bint}_{A^3}, \boldsymbol{bint}_{A^3}, 2\,{}_n\boldsymbol{bool}_{A^3} \vdash \boldsymbol{bint}_A\end{array}}{3\,!\boldsymbol{bint}_{A^3}, !\boldsymbol{bint}_{A^3}, 2\,!_n\boldsymbol{bool}_{A^3} \vdash \boldsymbol{bint}_A}\,der}{!\boldsymbol{bint}_{A^3}, !\boldsymbol{bint}_{A^3}, !_n\boldsymbol{bool}_{A^3} \vdash \boldsymbol{bint}_A}\,ctr$$

and similarly for $_\delta\underline{right}^\dagger_A$ *and* $_\delta\underline{state}^\dagger_A$.

It is clear that:

Lemma 4.17. *The proofs* $_\delta\underline{left}^\dagger_A$, $_\delta\underline{right}^\dagger_A$, $_\delta\underline{state}^\dagger_A$ *are plain.*

Proposition 4.18. *The proofs* $_\delta\underline{step}_A$ *and* $^p_\delta\underline{step}_A$ *are component-wise plain.*

Proof. The second claim follows from Proposition 3.6. □

The denotation of the proof $_\delta\underline{\text{step}}_A$ is a linear map

$$[\![{}_\delta\underline{\text{step}}_A]\!] : !{[\![\mathbf{bint}_{A^3}]\!]}^{\otimes 2} \otimes !{[\![\mathbf{bool}_{A^3}]\!]} \longrightarrow !{[\![\mathbf{bint}_A]\!]}^{\otimes 2} \otimes !{[\![\mathbf{bool}_A]\!]}$$

and we compute the value of this map on vectors of the form

$$\Psi = |\emptyset\rangle_\alpha \otimes |\emptyset\rangle_\beta \otimes |\emptyset\rangle_\gamma,$$

where

$$\alpha = \sum_{i=1}^{s} a_i [\![\underline{S_i\sigma_i}]\!], \quad \beta = \sum_{i=1}^{t} b_i [\![\underline{T_i\tau_i}]\!], \quad \gamma = \sum_{i=1}^{r} c_i [\![\underline{q_i}]\!].$$

Note that

$$[\![{}_\delta\underline{\text{step}}_A]\!](\Psi) = |\emptyset\rangle_{[\![{}_\delta\underline{\text{left}}_A]\!}(\alpha^{\otimes 3}\otimes\beta\otimes\gamma^{\otimes 2})} \otimes |\emptyset\rangle_{[\![{}_\delta\underline{\text{right}}_A]\!}(\alpha^{\otimes 2}\otimes\beta^{\otimes 2}\otimes\gamma^{\otimes 2})} \otimes |\emptyset\rangle_{[\![{}_\delta\underline{\text{state}}_A]\!}(\alpha\otimes\gamma)},$$

and so the problem reduces to computing the values of each of $[\![{}_\delta\underline{\text{left}}]\!]$, $[\![{}_\delta\underline{\text{right}}]\!]$ and $[\![{}_\delta\underline{\text{state}}]\!]$ on the appropriate number of copies of α, β, γ. Write $(\hat\sigma_i^j, \hat q_i^j, \hat d_i^j) = \delta(\sigma_i, q_j)$. We have:

- $[\![{}_\delta\underline{\text{left}}_A]\!](\underline{[\![S_i\sigma_i]\!]} \otimes \underline{[\![S_j\sigma_j]\!]} \otimes \underline{[\![S_k\sigma_k]\!]} \otimes \underline{[\![T_l\tau_l]\!]} \otimes \underline{[\![q_m]\!]} \otimes \underline{[\![q_n]\!]}) = \delta_{\hat d_k^m=0}\underline{[\![S_i]\!]} + \delta_{\hat d_k^m=1}\underline{[\![S_i\hat\sigma_j^m\tau_l]\!]}$,
- $[\![{}_\delta\underline{\text{right}}_A]\!](\underline{[\![S_j\sigma_j]\!]} \otimes \underline{[\![S_k\sigma_k]\!]} \otimes \underline{[\![T_l\tau_l]\!]} \otimes \underline{[\![T_i\tau_i]\!]} \otimes \underline{[\![q_m]\!]} \otimes \underline{[\![q_n]\!]}) = \delta_{\hat d_k^m=0}\underline{[\![T_i\tau_l\hat\sigma_j^m]\!]} + \delta_{\hat d_k^m=1}\underline{[\![T_i]\!]}$,
- $[\![{}_\delta\underline{\text{state}}_A]\!](\underline{[\![S_i\sigma_i]\!]} \otimes \underline{[\![q_j]\!]}) = \underline{[\![\hat q_i^j]\!]}$,

where δ on the right is the Kronecker delta. Using this, we compute

Lemma 4.19. *The vector* $[\![{}_\delta\underline{left}_A]\!](\alpha^{\otimes 3} \otimes \beta \otimes \gamma^{\otimes 2})$ *is equal as an element of* $[\![\boldsymbol{bint}_A]\!]$ *to*

$$\left(\sum_{i=1}^{s}\sum_{j=1}^{r} a_i c_j \delta_{\hat d_i^j=0}\right)\left(\sum_{i=1}^{s} a_i\right)\left(\sum_{i=1}^{t} b_i\right)\left(\sum_{i=1}^{r} c_i\right)\left(\sum_{i=1}^{s} a_i \underline{[\![S_i]\!]}\right)$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$+\left(\sum_{i=1}^{s}\sum_{j=1}^{r} a_i c_j \delta_{\hat{d}_i^j=1}\right)\left(\sum_{i=1}^{s}\sum_{j=1}^{s}\sum_{k=1}^{t}\sum_{l=1}^{r} a_i a_j b_k c_l [\![\underline{S_i \hat{\sigma}_j^l \tau_k}]\!]\right). \tag{16}$$

The vector $[\![\delta \underline{right}_A]\!](\alpha^{\otimes 2} \otimes \beta^{\otimes 2} \otimes \gamma^{\otimes 2})$ *is equal as an element of* $[\![\mathbf{bint}_A]\!]$ *to*

$$\left(\sum_{i=1}^{s}\sum_{j=1}^{r} a_i c_j \delta_{\hat{d}_i^j=1}\right)\left(\sum_{i=1}^{s} a_i\right)\left(\sum_{i=1}^{t} b_i\right)\left(\sum_{i=1}^{r} c_i\right)\left(\sum_{i=1}^{t} b_i [\![\underline{T_i}]\!]\right)$$
$$+\left(\sum_{i=1}^{s}\sum_{j=1}^{r} a_i c_j \delta_{\hat{d}_i^j=0}\right)\left(\sum_{i=1}^{t}\sum_{j=1}^{s}\sum_{k=1}^{t}\sum_{l=1}^{r} b_i a_j b_k c_l [\![\underline{T_i \hat{\sigma}_j^l \tau_k}]\!]\right). \tag{17}$$

The vector $[\![\delta \underline{state}_A]\!](\alpha \otimes \gamma)$ *is equal as an element of* $[\![\mathbf{bool}_A]\!]$ *to*

$$\sum_{i=1}^{s}\sum_{j=1}^{r} a_i c_j [\![\underline{\hat{q}_i^j}]\!]. \tag{18}$$

Remark 4.20. These formulas only describe the values of $[\![\delta \underline{step}_A]\!]$ on certain group-like elements, but the values on more general kets can be derived from these formulas by taking derivatives; see Clift and Murfet (2018, Corollary 4.5).

Remark 4.21. From the calculations in the lemma we can extract polynomial functions, as explained in Section 3.1. To do this we restrict the domain to sequences of bounded length, and identify such sequences with proofs. By Remark A.11 as long as we choose our type A such that $\dim[\![A]\!] > c/2$ the set of denotations

$$\left\{[\![\underline{S}_A]\!]\right\}_{S \in \Sigma^{\leq c}}$$

is a linearly independent set in $[\![\mathbf{bint}_A]\!]$, and in particular these denotations are all distinct, and we may identify $\Sigma^{\leq c}$ with the set of proofs $\mathcal{P}_c = \{\underline{S}_A\}_{S \in \Sigma^{\leq c}}$. Then by Proposition 3.8 if we fix integers a, b and then choose A such that $\dim([\![A]\!]) > \frac{1}{2}\max\{a+1, b+1\}$, there is a unique function $F_{\delta \underline{\mathrm{left}}_A}$ making the diagram

$$\begin{array}{ccc} !\mathbf{bint}_{A^3}\!] \otimes\, !\mathbf{bint}_{A^3}\!] \otimes\, !\mathbf{bool}_{A^3}\!] & \xrightarrow{[\![\delta \underline{\mathrm{left}}_A]\!]} & [\![\mathbf{bint}_A]\!] \\ \uparrow \iota & & \uparrow [\![-]\!] \\ k\Sigma^{\leq a} \times k\Sigma^{\leq b} \times kQ & \xrightarrow[F_{\delta \underline{\mathrm{left}}_A}]{} & k\Sigma^{\leq a+1} \end{array} \tag{19}$$

commute. The formula for the function $F_{\delta \underline{\mathrm{left}}_A}$ is given by reading the formula (16) with $[\![\underline{S_i}]\!]$ replaced by S_i and $[\![\underline{S_i \hat{\sigma}_j^l \tau_k}]\!]$ replaced by $S_i \hat{\sigma}_j^l \tau_k$, and similarly for $F_{\delta \underline{\mathrm{right}}_A}$ and $F_{\delta \underline{\mathrm{state}}_A}$.

4.2 The Boolean version

Our ultimate purpose in giving the encodings in this paper is to understand the derivatives of Turing machines in Clift and Murfet (2018), and we are therefore interested in encodings which allow us to differentiate the encoding with respect to the contents of individual tape squares. One natural way to obtain such an encoding is to derive it from $\underline{step}$ and that is our goal in this section. In Section 5 we give encodings obtained directly, without going via $\underline{step}$.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

The aim in this section is to construct a proof ${}^{p}\underline{\text{boolstep}}$ of

$$a\ !\mathbf{bool}_B, b\ !\mathbf{bool}_B, !_n\mathbf{bool}_B \vdash \left(!\mathbf{bool}_A\right)^{\otimes c} \otimes \left(!\mathbf{bool}_A\right)^{\otimes d} \otimes !_n\mathbf{bool}_A \tag{20}$$

for some power $B = A^{g(c,d,p)}$ such that if initially the state is q and the tape reads

$$\ldots, \square, x_1, \ldots, \underline{x_a}, y_1, \ldots, y_b, \square, \ldots$$

with the underline indicating the position of the head, then running the Turing machine for p steps on this input yields

$$\ldots, \square, x'_1, \ldots, \underline{x'_c}, y'_1, \ldots, y'_d, \square, \ldots$$

The algorithm ${}^{p}\underline{\text{boolstep}}$ returns this state, as a sequence of booleans, in the given order. In this section, to avoid an explosion of notation we drop the subscript δ and write $\underline{\text{boolstep}}$ when we should more correctly write ${}_{\delta}\underline{\text{boolstep}}$.

Lemma 4.22. *There is a proof* $\underline{cast}_A$ *of* $\mathbf{bool}_A \vdash \mathbf{bint}_A$ *which converts a boolean to the equivalent binary sequence; that is, it encodes* $[\![\underline{i}_A]\!] \mapsto [\![\underline{i}_A]\!]$ *for* $i \in \{0, 1\}$.

Proof. Let $E = A \multimap A$. The proof is:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\underline{0}_A \\ \vdots \\ !E, !E, A \vdash A\end{array} \qquad \begin{array}{c}\underline{1}_A \\ \vdots \\ !E, !E, A \vdash A\end{array}}{!E, !E, A \vdash A \& A}\,\&R \qquad \dfrac{}{A \vdash A}}{\mathbf{bool}_A, !E, !E, A \vdash A}\,\multimap L}{\mathbf{bool}_A \vdash \mathbf{bint}_A}\,3\times \multimap R$$

where the incoming proofs at the top are the binary integers 0, 1. □

Lemma 4.23. *There is a proof* $\underline{read}^j_A$ *of* $\mathbf{bint}_{A^{3^j+2}} \vdash \mathbf{bool}_A$ *for* $j \geq 0$ *which 'reads' the symbol at the position j from the right in a binary integer; that is, it encodes*

$$[\![\underline{s_r s_{r-1} \cdots s_1 s_0}_{A^{3^j+2}}]\!] \longmapsto [\![\underline{s_j}_A]\!].$$

Proof. We use the proofs $\underline{\text{head}}_A : \mathbf{bint}_{A^3} \vdash \mathbf{bint}_A$ of Lemma 4.3 and $\underline{\text{tail}}_A : \mathbf{bint}_{A^3} \vdash \mathbf{bint}_A$ of Lemma 4.4, cut together as follows (where $\psi \mid \phi$ denotes the cut of ψ, ϕ)

$$\underline{\text{head}}_{A^{3^{j+1}}} \mid \underline{\text{tail}}_{A^{3^j}} \mid \cdots \mid \underline{\text{tail}}_{A^3} \mid \underline{\text{tail}}_A : \mathbf{bint}_{A^{3^{j+2}}} \vdash \mathbf{bool}_A$$

where there are j copies of $\underline{\text{tail}}$ at various base types. Note that if we apply $\underline{\text{read}}^j$ to the empty binary sequence, or to any sequence of length $\leq j$, we obtain the boolean $\underline{0}$. □

Lemma 4.24. *There is a proof* $\underline{multread}^a_A$ *of* $!\mathbf{bint}_{A^{3^{a+1}}} \vdash \left(!\mathbf{bool}_A\right)^{\otimes a}$ *which reads off a symbols from the right end of a binary integer; that is, it encodes*

$$[\![\underline{s_r s_{r-1} \cdots s_1 s_0}_{A^{3^j+2}}]\!] \longmapsto \left([\![\underline{s_{a-1}}_A]\!], \ldots, [\![\underline{s_0}_A]\!]\right).$$

Proof. We use the proofs

$$\underline{\text{read}}^{a-1}_A : \mathbf{bint}_{A^{3^{a+1}}} \vdash \mathbf{bool}_A$$
$$\underline{\text{read}}^{a-2}_{A^3} : \mathbf{bint}_{A^{3^{a+1}}} \vdash \mathbf{bool}_{A^3}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

$$\underline{\mathrm{read}}^{a-3}_{A3^2} : \mathbf{bint}_{A3^{a+1}} \vdash \mathbf{bool}_{A3}$$

$$\vdots$$

$$\underline{\mathrm{read}}^{0}_{A3^{a-1}} : \mathbf{bint}_{A3^{a+1}} \vdash \mathbf{bool}_{A3^{a-1}}$$

to read off the relevant symbols, and then fix the base types by cutting against an appropriate number of copies of the proof ${}_n\underline{\mathrm{booltype}}_B$ of Lemma 4.6 with various base types B. This leaves us with a proofs of $\mathbf{bint}_{A3^{a+1}} \vdash \mathbf{bool}_A$ which we then derelict on the left and promote on the right, and then tensor together to obtain a proof of

$$a\,!\mathbf{bint}_{A3^{a+1}} \vdash \left(!\mathbf{bool}_A\right)^{\otimes a}.$$

A series of contractions then gives the desired proof $\underline{\mathrm{multread}}^a_A$. □

Definition 4.25. *For $c, d \geq 1$ the proof $\underline{unpack}^{c,d}_A$ of*

$$\boldsymbol{Tur}_{A3^{\max\{c,d\}+1}} \vdash \left(!\boldsymbol{bool}_A\right)^{\otimes c} \otimes \left(!\boldsymbol{bool}_A\right)^{\otimes d} \otimes !_n\boldsymbol{bool}_A$$

is defined as follows: let $e = \max\{c, d\}$ and consider the proofs

$$\underline{multread}^c_{A3^{e-c}} : !\boldsymbol{bint}_{A3^{e+1}} \vdash \left(!\boldsymbol{bool}_{A3^{e-c}}\right)^{\otimes c}$$

$$\underline{multread}^d_{A3^{e-d}} : !\boldsymbol{bint}_{A3^{e+1}} \vdash \left(!\boldsymbol{bool}_{A3^{e-d}}\right)^{\otimes d}.$$

We cut these proofs against an appropriate number of promoted copies of ${}_n\underline{booltype}_B$ for various base types B, to obtain proofs of sequents with conclusion $\left(!\boldsymbol{bool}_A\right)^{\otimes c}, \left(!\boldsymbol{bool}_A\right)^{\otimes d}$. These are tensored with the identity on $!_n\boldsymbol{bool}_A$ and with the proof that reverses the output of the $\underline{multread}$ corresponding to the right-hand part of the tape.

Definition 4.26. *The proof $\underline{pack}^{a,b}_A$ of $a\,!\boldsymbol{bool}_A, b\,!\boldsymbol{bool}_A, !_n\boldsymbol{bool}_A \vdash \boldsymbol{Tur}_A$ is given by the tensor product of the identity on $!_n\boldsymbol{bool}_A$ and the proofs*

$$\dfrac{\dfrac{\dfrac{\underline{cast}^{\otimes a}_A}{\vdots}}{a\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}^{\otimes a}_A} \qquad \dfrac{\underline{concat}}{\vdots}}{\dfrac{a\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}_A}{a\,!\boldsymbol{bool}_A \vdash !\boldsymbol{bint}_A}\,der,prom}\,cut \qquad \boldsymbol{bint}^{\otimes a}_A \vdash \boldsymbol{bint}_A$$

and

$$\dfrac{\dfrac{\dfrac{\dfrac{\underline{cast}^{\otimes b}_A}{\vdots}}{b\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}^{\otimes b}_A}\,ex \qquad \dfrac{\underline{concat}}{\vdots}}{b\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}^{\otimes b}_A \qquad \boldsymbol{bint}^{\otimes b}_A \vdash \boldsymbol{bint}_A}\,cut}{\dfrac{b\,\boldsymbol{bool}_A \vdash \boldsymbol{bint}_A}{b\,!\boldsymbol{bool}_A \vdash !\boldsymbol{bint}_A}\,der,prom}$$

where the exchange rule reverses the order of the inputs.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

To simplify the formulas in the next definition, let us write

$$\Gamma_A^{a,b} = a\,!\mathbf{bool}_A, b\,!\mathbf{bool}_A, !_n\mathbf{bool}_A\,,$$

$$X_A^{c,d} = \left(!\mathbf{bool}_A\right)^{\otimes c} \otimes \left(!\mathbf{bool}_A\right)^{\otimes d} \otimes !_n\mathbf{bool}_A\,.$$

Definition 4.27. *For $a, b, c, d, p \geq 1$ the proof ${}^p\underline{boolstep}_A^{a,b,c,d}$ is*

$$\dfrac{\dfrac{\underline{pack}_{A^{3^{e+1}}}^{a,b} \quad \vdots}{\Gamma_{A^{3^{p+e+1}}}^{a,b} \vdash \boldsymbol{Tur}_{A^{3^{p+e+1}}}} \qquad \dfrac{{}^p_\delta\underline{step}_{A^{3^{e+1}}} \quad \vdots}{\boldsymbol{Tur}_{A^{3^{p+e+1}}} \vdash \boldsymbol{Tur}_{A^{3^{e+1}}}}}{\Gamma_{A^{3^{p+e+1}}}^{a,b} \vdash \boldsymbol{Tur}_{A^{3^{e+1}}}}\,cut \qquad \dfrac{\underline{unpack}_A^{c,d} \quad \vdots}{\boldsymbol{Tur}_{A^{3^{e+1}}} \vdash X_A^{c,d}}}{\Gamma_{A^{3^{p+e+1}}}^{a,b} \vdash X_A^{c,d}}\,cut$$

where $e = \max\{c, d\}$.

Lemma 4.28. *The proof ${}^p\underline{boolstep}_A^{a,b,c,d}$ is component-wise plain.*

Proof. By Propositions 3.6 and 4.18, it suffices to argue that pack and unpack are component-wise plain, but this is obvious. □

5. Direct Boolean Encodings

In the previous section, we developed an encoding of the step function as an operation on a sequence of booleans, rather than a pair of binary integers. It is natural to ask if such an encoding can be designed directly, without going via step.

Ideally, one could define a family of proofs of

$$a\,!\mathbf{bool}_A, b\,!\mathbf{bool}_A, !_n\mathbf{bool}_A \vdash !\mathbf{bool}_A^{\otimes c} \otimes !\mathbf{bool}_A^{\otimes d} \otimes !_n\mathbf{bool}_A$$

indexed by a, b, c, d, as in Section 4.2. However, there is an issue with this approach, in that it cannot be easily iterated. More specifically, we cannot simply cut multiple one-step proofs against one another, since the computation of all steps beyond the first must know where the tape head has moved to, which is invisible when encoding the tape contents as simply a string of booleans.

In this section, we present two possible remedies for this issue. The first is to increase the number of booleans representing the tape after each step, in such a way to ensure that the head is always in the centre of the working section of the tape. We call this the *relative encoding* and denote the proof relstep (see Section 5.1). The second solution is to simply keep track of the head position through the use of an additional h-boolean. We call this the *absolute encoding* and denote the proof absstep (see Section 5.2). While this has the advantage that the positions of each tape square do not change after each step, it is limited by the fact that one must remain in a section of the tape of length h.

Fix a finite set of states $Q = \{0, ..., n-1\}$ and a finite tape alphabet $\Sigma = \{0, ..., s-1\}$, and let $\delta : \Sigma \times Q \to \Sigma \times Q \times \{\text{left}, \text{right}\}$ be the transition function. For $i \in \{0, 1, 2\}$, we write $\delta_i = \text{proj}_i \circ \delta$ for the ith component of δ. Throughout A is a fixed type, and we write **bool** for $\mathbf{bool}_A$. We give the versions of relstep and absstep in which the head of the Turing machine must either move left or right in each time step; the modification to allow the head to remain stationary is routine.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Definition 5.1. *Given $n \in \mathbb{N}$, we write ${}^n\underline{eval}$ for the following proof, whose denotation is the evaluation map.*

$$\frac{\overline{A^n \vdash A^n} \qquad \overline{A \vdash A}}{{}_n\mathbf{bool}, A^n \vdash A} \multimap L$$

Definition 5.2. *Given two proofs π, ρ of $\Gamma \vdash A$, we write $\pi \ \& \ \rho$ for the following proof.*

$$\frac{\begin{array}{cc}\pi & \rho \\ \vdots & \vdots \\ \Gamma \vdash A & \Gamma \vdash A\end{array}}{\Gamma \vdash A^2} \&R$$

As for formulas, we write π^n for the proof $\pi \ \& \ \ldots \ \& \pi$, where there are n copies of π.

We will frequently need to discard unwanted booleans without the use of exponentials. This can be achieved with the following family of proofs.

Definition 5.3. *Let π be a proof of $\Gamma \vdash A$. We recursively define a family of proofs*

$${}_n\underline{boolweak}(\pi, k) : \Gamma, k\ {}_n\mathbf{bool} \vdash A$$

for $k \geq 0$ by defining ${}_n\underline{boolweak}(\pi, 0) = \pi$, and defining ${}_n\underline{boolweak}(\pi, k+1)$ as the proof

$$\frac{\begin{array}{c}{}_n\underline{boolweak}(\pi, k)^n \\ \vdots \\ \Gamma, k\ {}_n\mathbf{bool} \vdash A^n\end{array} \qquad \overline{A \vdash A}}{\Gamma, (k+1)\ {}_n\mathbf{bool} \vdash A} \multimap L$$

5.1 Relative step

The goal of this section is to construct a component-wise plain proof

$${}_h\underline{\text{relstep}} : !_s\mathbf{bool}^{\otimes 2h+1} \otimes !_n\mathbf{bool} \vdash !_s\mathbf{bool}^{\otimes 2h+3} \otimes !_n\mathbf{bool}$$

which encodes a single step transition of a Turing machine, such that the position of the head remains in the middle of the sequence of s-booleans representing the tape. For this to be possible, we introduce two new copies of the blank symbol, to be appended to the left end of the tape if the head moves left, and likewise for right. As an example, if the tape contents were initially

$$\sigma_{-h} \ldots \sigma_{-1}\sigma_0\sigma_1 \ldots \sigma_h$$

then $\underline{\text{relstep}}$ would produce the sequence

$$\square\,\square\,\sigma_{-h} \ldots \sigma_{-1}\sigma_0'\sigma_1 \ldots \sigma_h$$

if the Turing machine moved left, where σ_0' is the newly written symbol. Note that the new symbol in the centre of the tape is σ_{-1}. If instead the Turing machine moved right, then $\underline{\text{relstep}}$ would produce the sequence

$$\sigma_{-h} \ldots \sigma_{-1}\sigma_0'\sigma_1 \ldots \sigma_h\,\square\,\square,$$

thus causing σ_1 to become the new symbol in the centre of the tape.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Lemma 5.4. *For $-h-1 \leq m \leq h+1$, there exists a proof ${}^{m}\underline{\text{symbol}}$ which computes the new symbol in position m, relative to the new tape head position.*

Proof. In order to compute the symbol in relative position m, the proof ${}^{m}\underline{\text{symbol}}$ will require a copy of the symbol in position $m-1$ to be used if we move left, and the symbol in position $m+1$ if we move right. We will also need to provide each proof ${}^{m}\underline{\text{symbol}}$ with a copy of the state and the currently scanned symbol, in order to compute the direction to direction to move. Define π as the following proof.

$$\dfrac{\dfrac{{}^{2}_{\delta}\underline{\text{trans}}}{\vdots}\quad \dfrac{\dfrac{\dfrac{\underline{\text{boolweak}}({}^{s}\underline{\text{eval}},1)}{\vdots}}{{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},A^{s}\vdash A}\quad \dfrac{\dfrac{\underline{\text{boolweak}}({}^{s}\underline{\text{eval}},1)}{\vdots}}{{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},A^{s}\vdash A}\,\text{exch}}{{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},A^{s}\vdash A^{2}}\,\&R\quad \dfrac{}{A\vdash A}}{{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},\mathbf{bool},A^{s}\vdash A}\,\multimap L}{{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},\mathbf{bool}\vdash {}_{s}\mathbf{bool}}\,\multimap R}{{}_{s}\mathbf{bool},{}_{n}\mathbf{bool}\vdash \mathbf{bool}}\quad \text{cut}$$
$$\qquad\qquad {}_{s}\mathbf{bool},{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},{}_{n}\mathbf{bool}\vdash {}_{s}\mathbf{bool}$$

If $\alpha,\beta,\sigma\in\Sigma$ and $q\in Q$, then we have

$$[\![\pi]\!](\alpha,\beta,\sigma,q)=\begin{cases}\alpha & \delta_2(\sigma,q)=0\ \text{(left)}\\ \beta & \delta_2(\sigma,q)=1\ \text{(right)}.\end{cases}$$

For the symbols not near the tape head ($m\neq\pm1$) or the ends of the tape ($m\neq\pm h,\pm(h+1)$), we can simply define ${}^{m}\underline{\text{symbol}}=\pi$, with the understanding that the inputs to ${}^{m}\underline{\text{symbol}}$ will be the s-booleans corresponding to relative position $m-1, m+1$ and 0 respectively. For $m=\pm h$, $\pm(h+1)$, we define ${}^{m}\underline{\text{symbol}}$ as the cut of the blank symbol $\underline{0}$ against π, in order to introduce new blank symbols at the ends of the tape. This produces a proof of ${}_{s}\mathbf{bool},{}_{s}\mathbf{bool},{}_{n}\mathbf{bool}\vdash{}_{s}\mathbf{bool}$. Finally for $m=\pm1$, we must also compute a copy of the new symbol, which can be achieved by cutting ${}^{0}_{\delta}\underline{\text{trans}}$ against π, producing a proof of ${}_{s}\mathbf{bool},{}_{n}\mathbf{bool},{}_{s}\mathbf{bool},{}_{s}\mathbf{bool},{}_{n}\mathbf{bool}\vdash{}_{s}\mathbf{bool}$. □

Proposition 5.5. *For $h\geq 0$, there is a component-wise plain proof*

$${}_{h}\underline{\text{relstep}}:{!}_{s}\boldsymbol{bool}^{\otimes 2h+1}\otimes{!}_{n}\boldsymbol{bool}\vdash{!}_{s}\boldsymbol{bool}^{\otimes 2h+3}\otimes{!}_{n}\boldsymbol{bool}$$

which encodes a single transition step of a given Turing machine, using relative tape coordinates.

Proof. For $-h-1\leq m\leq h+1$ we derelict the hypotheses of ${}^{m}\underline{\text{symbol}}$ and then promote the result, do the same with ${}^{1}_{\delta}\underline{\text{trans}}$, and then we tensor all these promoted proofs together, and perform contractions. In total, it is necessary to create two copies of each ${}_{s}\mathbf{bool}$ other than the current symbol, along with $2k+6$ copies of the current symbol and state. These copies are used to produce two copies of the new symbol, one copy of the new state and $2k+3$ copies of the direction to move; one for each of the proofs ${}^{m}\underline{\text{symbol}}$. □

Remark 5.6. We now compute the polynomials associated with ${}_{h}\underline{\text{relstep}}$:

$$\alpha^{m}=\sum_{i\in\Sigma}a_i^{m}[\![\underline{i}]\!],\qquad \beta=\sum_{q\in Q}b_q[\![\underline{q}]\!].$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

We compute the value of $[\![_h\underline{\text{relstep}}]\!]$ on $|\emptyset\rangle_{\alpha^{-h}} \otimes \ldots \otimes |\emptyset\rangle_{\alpha^h} \otimes |\emptyset\rangle_\beta$. The image of this element is a tensor of the form

$$|\emptyset\rangle_{\theta^{-h-1}} \otimes \ldots \otimes |\emptyset\rangle_{\theta^{h+1}} \otimes |\emptyset\rangle_\mu$$

where the individual vectors θ^m, μ are described as follows. Let $(\hat{\sigma}_i^q, \hat{q}_i^q, \hat{d}_i^q) = \delta(i,q)$. For $m \neq \pm 1, \pm h, \pm(h+1)$ the vector θ^m is given by

$$\begin{aligned}
&[\![^m\underline{\text{symbol}}]\!](\alpha^{m-1} \otimes \alpha^{m+1} \otimes \alpha^0 \otimes \beta) \\
&\quad = \sum_{i\in\Sigma}\sum_{j\in\Sigma}\sum_{k\in\Sigma}\sum_{q\in Q} a_i^{m-1} a_j^{m+1} a_k^0 b_q (\delta_{\hat{d}_k^q=0}[\![\underline{i}]\!] + \delta_{\hat{d}_k^q=1}[\![\underline{j}]\!]) \\
&\quad = \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=0}\right)\left(\sum_{i\in\Sigma} a_i^{m-1}[\![\underline{i}]\!]\right)\left(\sum_{j\in\Sigma} a_j^{m+1}\right) \\
&\qquad + \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=1}\right)\left(\sum_{i\in\Sigma} a_i^{m-1}\right)\left(\sum_{j\in\Sigma} a_j^{m+1}[\![\underline{j}]\!]\right).
\end{aligned}$$

For $m = h, h+1$ the vector θ^m is given by

$$\begin{aligned}
&[\![^m\underline{\text{symbol}}]\!](\alpha^{m-1} \otimes \alpha^0 \otimes \beta) \\
&\quad = \sum_{i\in\Sigma}\sum_{k\in\Sigma}\sum_{q\in Q} a_i^{m-1} a_k^0 b_q (\delta_{\hat{d}_k^q=0}[\![\underline{i}]\!] + \delta_{\hat{d}_k^q=1}[\![\underline{0}]\!]) \\
&\quad = \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=0}\right)\left(\sum_{i\in\Sigma} a_i^{m-1}[\![\underline{i}]\!]\right) \\
&\qquad + \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=1}\right)\left(\sum_{i\in\Sigma} a_i^{m-1}\right)[\![\underline{0}]\!].
\end{aligned}$$

For $m = -h, -h-1$ the vector θ^m is given by

$$\begin{aligned}
&[\![^m\underline{\text{symbol}}]\!](\alpha^{m+1} \otimes \alpha^0 \otimes \beta) \\
&\quad = \sum_{i\in\Sigma}\sum_{k\in\Sigma}\sum_{q\in Q} a_i^{m+1} a_k^0 b_q (\delta_{\hat{d}_k^q=0}[\![\underline{0}]\!] + \delta_{\hat{d}_k^q=1}[\![\underline{i}]\!]) \\
&\quad = \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=0}\right)\left(\sum_{i\in\Sigma} a_i^{m+1}\right)[\![\underline{0}]\!] \\
&\qquad + \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=1}\right)\left(\sum_{i\in\Sigma} a_i^{m+1}[\![\underline{i}]\!]\right).
\end{aligned}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Finally θ^1 is

$$\begin{aligned}
&[\![^1\underline{\text{symbol}}]\!](\alpha^0 \otimes \beta \otimes \alpha^2 \otimes \alpha^0 \otimes \beta) \\
&\quad = \sum_{i\in\Sigma}\sum_{p\in Q}\sum_{j\in\Sigma}\sum_{k\in\Sigma}\sum_{q\in Q} a_i^0 b_p a_j^2 a_k^0 b_q (\delta_{\hat{d}_k^q=0}[\![\underline{\hat{\sigma}_i^p}]\!] + \delta_{\hat{d}_k^q=1}[\![\underline{j}]\!]) \\
&\quad = \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=0}\right)\left(\sum_{i\in\Sigma}\sum_{p\in Q} a_i^0 b_p [\![\underline{\hat{\sigma}_i^p}]\!]\right)\left(\sum_{j\in\Sigma} a_j^2\right) \\
&\qquad + \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=1}\right)\left(\sum_{i\in\Sigma} a_i^0\right)\left(\sum_{p\in Q} b_p\right)\left(\sum_{j\in\Sigma} a_j^2 [\![\underline{j}]\!]\right).
\end{aligned}$$

and θ^{-1} is

$$\begin{aligned}
&[\![^{-1}\underline{\text{symbol}}]\!](\alpha^{-2} \otimes \alpha^0 \otimes \beta \otimes \alpha^0 \otimes \beta) \\
&\quad = \sum_{i\in\Sigma}\sum_{j\in\Sigma}\sum_{p\in Q}\sum_{k\in\Sigma}\sum_{q\in Q} a_i^{-2} a_j^0 b_p a_k^0 b_q (\delta_{\hat{d}_k^q=0}[\![\underline{i}]\!] + \delta_{\hat{d}_k^q=1}[\![\underline{\hat{\sigma}_j^p}]\!]) \\
&\quad = \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=0}\right)\left(\sum_{i\in\Sigma} a_i^{-2}[\![\underline{i}]\!]\right)\left(\sum_{j\in\Sigma} a_j^0\right)\left(\sum_{p\in Q} b_p\right) \\
&\qquad + \left(\sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q \delta_{\hat{d}_k^q=1}\right)\left(\sum_{i\in\Sigma} a_i^{-2}\right)\left(\sum_{j\in\Sigma}\sum_{p\in Q} a_j^0 b_p [\![\underline{\hat{\sigma}_j^p}]\!]\right).
\end{aligned}$$

The vector μ is computed by

$$[\![^1_\delta\underline{\text{trans}}]\!](\alpha^0 \otimes \beta) = \sum_{k\in\Sigma}\sum_{q\in Q} a_k^0 b_q [\![\underline{\hat{q}_k^q}]\!].$$

Remark 5.7. We have now given two encodings of the step function of a Turing machine which use booleans to encode the contents of the tape squares, and which index these tape squares relative to the head, namely $\underline{\text{boolstep}}$ (in Section 4.2) and $\underline{\text{relstep}}$. These are not proofs of the same sequents but there is a natural way to compare them when $s = 2$, and the result of this comparison is *negative*: the encodings are genuinely different.

Recall that ${}^p\underline{\text{boolstep}}_A^{a,b,c,d}$ is a proof of

$$a\ !\mathbf{bool}_B, b\ !\mathbf{bool}_B, !_n\mathbf{bool}_B \vdash !\mathbf{bool}_A^{\otimes c} \otimes !\mathbf{bool}_A^{\otimes d} \otimes !_n\mathbf{bool}_A \tag{21}$$

where $B = A^{3^{p+e+1}}$ with $e = \max\{c, d\}$. In general, the diagram

$$\begin{array}{ccc}
!\llbracket\mathbf{bool}_B\rrbracket^{\otimes 2h+1} \otimes !\llbracket{}_n\mathbf{bool}_B\rrbracket & \xrightarrow{\llbracket^1\underline{\text{boolstep}}^{h+1,h,h+2,h+1}\rrbracket} & !\llbracket\mathbf{bool}_A\rrbracket^{\otimes 2h+3} \otimes !\llbracket{}_n\mathbf{bool}_A\rrbracket \\
\Big\downarrow {\scriptstyle !\llbracket\underline{\text{booltype}}\rrbracket^{\otimes 2h+1}\otimes !\llbracket{}_n\underline{\text{booltype}}\rrbracket} & & \Big\uparrow {\scriptstyle =} \\
!\llbracket\mathbf{bool}_A\rrbracket^{\otimes 2h+1} \otimes !\llbracket{}_n\mathbf{bool}_A\rrbracket & \xrightarrow[\llbracket{}_h\underline{\text{relstep}}\rrbracket]{} & !\llbracket\mathbf{bool}_A\rrbracket^{\otimes 2h+3} \otimes !\llbracket{}_n\mathbf{bool}_A\rrbracket
\end{array}$$

does not commute. However, the two encodings do give rise to the same naive probabilistic extension of the step function; see Clift and Murfet (2018, Appendix D).

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

5.2 Absolute step

Suppose we are confined to a region of the tape which is h symbols long, encoded by a sequence of ${}_s\mathbf{bool}$'s which appear in the same order as they occur on the tape. The goal of this section is to construct a component-wise plain proof

$$ {}_h\underline{\text{absstep}} : !_s\mathbf{bool}^{\otimes h} \otimes !_n\mathbf{bool} \otimes !_h\mathbf{bool} \vdash !_s\mathbf{bool}^{\otimes h} \otimes !_n\mathbf{bool} \otimes !_h\mathbf{bool} $$

which encodes a single step transition of a Turing machine, where the purpose of the h-boolean is to keep track of the head position, and thus it decrements if we move left, and increments if we move right. The valid positions are $\{0, \ldots, h-1\}$ which we identify with $\mathbb{Z}/h\mathbb{Z}$. All positions are read modulo h, and so if the head is currently in position $h-1$ and it moves right, it moves to position 0, and similarly if the head is at position 0 and moves left, it moves to position $h-1$.

Lemma 5.8. *For $0 \leq m \leq h-1$, there exists a proof*

$$ {}^m\underline{symbol} : {}_s\boldsymbol{bool}, {}_n\boldsymbol{bool}, {}_h\boldsymbol{bool} \vdash {}_s\boldsymbol{bool} $$

which encodes the function

$$ (\sigma, q, i) \mapsto \begin{cases} \sigma & i \neq m \\ \delta_0(\sigma, q) & i = m. \end{cases} $$

Proof. Let $\pi : {}_s\mathbf{bool}, {}_n\mathbf{bool}, A^s \vdash A$ be the proof ${}_n\underline{\text{boolweak}}({}^s\underline{\text{eval}}, 1)$. Then ${}^m\underline{\text{symbol}}$ is the following proof.

$$ \dfrac{\dfrac{\pi^m \,\&\, {}^0_\delta\underline{\text{trans}} \,\&\, \pi^{h-m-1}}{\vdots}}{\dfrac{\dfrac{{}_s\mathbf{bool}, {}_n\mathbf{bool}, A^s \vdash A^h \qquad \overline{A \vdash A}}{{}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, A^s \vdash A}\,{\multimap L}}{{}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool} \vdash {}_s\mathbf{bool}}\,{\multimap R}} $$

□

Lemma 5.9. *There exists a proof*

$$ \underline{state} : h\ {}_s\boldsymbol{bool}, {}_n\boldsymbol{bool}, {}_h\boldsymbol{bool} \vdash {}_n\boldsymbol{bool} $$

which encodes the function

$$ (\sigma_0, \ldots, \sigma_{h-1}, q, i) \mapsto \delta_1(\sigma_i, q). $$

Proof. For $0 \leq i \leq h-1$, define π_i be the proof as the proof

$$ {}_s\underline{\text{boolweak}}({}^1_\delta\underline{\text{trans}}, h-1) : h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, A^n \vdash A, $$

where the original ${}_s\mathbf{bool}$ from the proof ${}^1_\delta\underline{\text{trans}}$ is in the ith position on the left of the turnstile in π_i. Then $\underline{\text{state}}$ is the following proof.

$$ \dfrac{\dfrac{\pi_0 \,\&\, \ldots \,\&\, \pi_{h-1}}{\vdots}}{\dfrac{\dfrac{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, A^n \vdash A^h \qquad \overline{A \vdash A}}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, A^n \vdash A}\,{\multimap L}}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool} \vdash {}_n\mathbf{bool}}\,{\multimap R}} $$

□

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Lemma 5.10. *There exists a proof*

$$\underline{tapehead} : h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, {}_h\mathbf{bool} \vdash {}_h\mathbf{bool}$$

which encodes the function

$$(\sigma_0, ..., \sigma_{h-1}, q, i, j) \mapsto \begin{cases} i-1 & \delta_2(\sigma_j, q) = left \\ i+1 & \delta_2(\sigma_j, q) = right. \end{cases}$$

Proof. Let $\tau_L : {}_h\mathbf{bool}, A^h \vdash A$ be the proof which converts the h-boolean $\underline{i}$ into the h-boolean $\underline{i-1}$ modulo h, defined as follows:

$$\dfrac{\dfrac{\dfrac{\overline{A \vdash A}}{A^h \vdash A}\,{\scriptstyle \&L_{h-1}} \quad \dfrac{\overline{A \vdash A}}{A^h \vdash A}\,{\scriptstyle \&L_0} \quad \ldots \quad \dfrac{\overline{A \vdash A}}{A^h \vdash A}\,{\scriptstyle \&L_{h-2}}}{A^h \vdash A^h}\,{\scriptstyle \&R} \qquad \overline{A \vdash A}}{{}_h\mathbf{bool}, A^h \vdash A}\,{\scriptstyle \multimap L}$$

Similarly define a proof τ_R which encodes $\underline{i} \mapsto \underline{i+1}$ modulo h.

Given $0 \leq i \leq h-1$, let ρ_i be the following proof, where as in Lemma 5.9 the i indicates which ${}_s\mathbf{bool}$ in the leftmost branch is the original copy from the proof ${}^2\underline{\text{trans}}$.

$$\dfrac{\dfrac{{}_s\underline{\text{boolweak}}({}^2_\delta\underline{\text{trans}}, h-1)}{\vdots}}{\dfrac{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, A^2 \vdash A}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool} \vdash \mathbf{bool}}\,{\scriptstyle \multimap R}} \qquad \dfrac{\dfrac{\tau_L \,\&\, \tau_R}{\vdots}}{\dfrac{{}_h\mathbf{bool}, A^h \vdash A^2 \qquad \overline{A \vdash A}}{{}_h\mathbf{bool}, \mathbf{bool}, A^h \vdash A}\,{\scriptstyle \multimap L}}$$

$$\dfrac{\qquad\qquad}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, A^h \vdash A}\,{\scriptstyle \text{cut}}$$

Then $\underline{\text{tapehead}}$ is:

$$\dfrac{\dfrac{\dfrac{\rho_0 \,\&\, \ldots \,\&\, \rho_{h-1}}{\vdots}}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, A^h \vdash A^h \qquad \overline{A \vdash A}}\,{\scriptstyle \multimap L}}{h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, {}_h\mathbf{bool}, A^h \vdash A}\,{\scriptstyle \multimap R}$$

$$h\ {}_s\mathbf{bool}, {}_n\mathbf{bool}, {}_h\mathbf{bool}, {}_h\mathbf{bool} \vdash {}_h\mathbf{bool}$$

□

Proposition 5.11. *There is a component-wise plain proof*

$${}_h\underline{absstep} : !_s\mathbf{bool}^{\otimes h} \otimes !_n\mathbf{bool} \otimes !_h\mathbf{bool} \vdash !_s\mathbf{bool}^{\otimes h} \otimes !_n\mathbf{bool} \otimes !_h\mathbf{bool}$$

which encodes a single transition step of a given Turing machine, using absolute tape coordinates.

Proof. Clear from the above. In total, the contraction steps at the bottom of the tree produce three copies of each tape ${}_s\mathbf{bool}$, $h+2$ copies of the state ${}_n\mathbf{bool}$ and $h+3$ copies of the ${}_h\mathbf{bool}$ which keeps track of the head position. □

Remark 5.12. One interesting feature of the absolute encoding is that it can be iterated for a variable number of steps through the use of an **int**, since the denotation of ${}_h\underline{absstep}$ is an endomorphism. Note that this is impossible for the relative encoding, since the type changes after each step due to the growth of the tape.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Remark 5.13. We now compute the denotation of $_h\underline{\text{absstep}}$. Let $H = \{0, ..., h-1\}$, and define

$$\alpha^m = \sum_{i_m \in \Sigma} a^m_{i_m} [\![\underline{i_m}]\!], \qquad \beta = \sum_{q \in Q} b_q [\![\underline{q}]\!], \qquad \gamma = \sum_{k \in H} c_k [\![\underline{k}]\!].$$

We compute the value of $[\![_h\underline{\text{absstep}}]\!]$ on $|\emptyset\rangle_{\alpha^0} \otimes ... \otimes |\emptyset\rangle_{\alpha^{h-1}} \otimes |\emptyset\rangle_\beta \otimes |\emptyset\rangle_\gamma$. Let $(\hat{\sigma}^q_i, \hat{q}^q_i, \hat{d}^q_i) = \delta(i, q)$. Note that:

$$[\![^m\underline{\text{symbol}}]\!](\alpha^m \otimes \beta \otimes \gamma)$$
$$= \sum_{i_m \in \Sigma} \sum_{q \in Q} a^m_{i_m} b_q \left(c_m [\![\underline{\hat{\sigma}^q_{i_m}}]\!] + \sum_{\substack{k \in H \\ k \neq m}} c_k [\![\underline{i_m}]\!] \right),$$

$$[\![\underline{\text{state}}]\!](\alpha^0 \otimes ... \otimes \alpha^{h-1} \otimes \beta \otimes \gamma)$$
$$= \sum_{i_0 \in \Sigma} \cdots \sum_{i_{h-1} \in \Sigma} \sum_{q \in Q} \sum_{k \in H} a_{i_0} \ldots a_{i_{h-1}} b_q c_k [\![\underline{\hat{q}^q_{i_k}}]\!]$$
$$= \sum_{k \in H} c_k \left[\left(\sum_{i_k \in \Sigma} \sum_{q \in Q} a^k_{i_k} b_q [\![\underline{\hat{q}^q_{i_k}}]\!] \right) \prod_{\substack{m \in H \\ m \neq k}} \left(\sum_{i_m \in \Sigma} a^m_{i_m} \right) \right], \text{ and}$$

$$[\![\underline{\text{tapehead}}]\!](\alpha^0 \otimes ... \otimes \alpha^{h-1} \otimes \beta \otimes \gamma^{\otimes 2})$$
$$= \sum_{i_0 \in \Sigma} \cdots \sum_{i_{h-1} \in \Sigma} \sum_{q \in Q} \sum_{k \in H} \sum_{l \in H} a_{i_0} \ldots a_{i_{h-1}} b_q c_k c_l \left(\delta_{\hat{d}^q_{i_l}=0} [\![\underline{k-1}]\!] + \delta_{\hat{d}^q_{i_l}=1} [\![\underline{k+1}]\!] \right)$$
$$= \sum_{l \in H} c_l \left[\left(\sum_{i_l \in \Sigma} \sum_{q \in Q} \sum_{k \in H} a^l_{i_l} b_q c_k \left(\delta_{\hat{d}^q_{i_l}=0} [\![\underline{k-1}]\!] + \delta_{\hat{d}^q_{i_l}=1} [\![\underline{k+1}]\!] \right) \right) \prod_{\substack{m \in H \\ m \neq l}} \left(\sum_{i_m \in \Sigma} a^m_{i_m} \right) \right],$$

where $k \pm 1$ is computed modulo h.

Notes

1 More precisely, the denotational semantics of Ehrhard–Regnier's derivative in the Sweedler semantics, and other similar semantics, involves such variations.

2 This rule appears to be missing from the cut-elimination transformations in Melliès (2009).

3 It is straightforward to modify what follows to allow larger tape alphabets. At any rate, any Turing machine can be simulated by one whose tape alphabet is {0, 1}, so this is really not as restrictive as it might seem (Rogozhin 1996).

References

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*, Cambridge University Press.

Benton, N., Bierman, G., de Paiva, V. and Hyland, M. (1992). *Term Assignment for Intuitionistic Linear Logic*, Technical report 262, Computer Laboratory, University of Cambridge.

Clift, J. and Murfet, D. (2017). *Cofree Coalgebras and Differential Linear Logic*, arXiv preprint [arXiv:1701.01285].

Clift, J. and Murfet, D. (2018). *Derivatives of Turing machines in Linear logic*, arXiv preprint [arXiv:1805.11813].

Drensky, V. and Formanek, E. (2004). *Polynomial Identity Rings*, Birkhäuser.

Ehrhard, T. and Regnier, L. (2003). The Differential λ-Calculus. *Theoretical Computer Science* **309** 1–41.

Ehrhard, T. (2016). *An Introduction to Differential Linear Logic: Proof-Nets, Models and Antiderivatives*, arXiv preprint [arXiv:1606.01642].

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science* **50** (1) 1–102.
Girard, J.-Y. (1995). Light linear logic. *Information and Computation* **143** (2) 175–204.
de la Harpe, P. (2000). *Topics in Geometric Group Theory*, The University of Chicago Press.
Hyland, M. and Schalk, A. (2003). Glueing and orthogonality for models of linear logic. *Theoretical Computer Science* **294** 183–231.
Mairson, H. and Terui, K. (2003). On the computational complexity of cut-elimination in linear logic. In: Blundo, C. and Laneve, C. (eds.) *Theoretical Computer Science. ICTCS 2003. Lecture Notes in Computer Science*, vol. 2841, Springer, Berlin, Heidelberg.
Melliès, P.-A. (2009). Categorical semantics of linear logic. In: *Interactive Models of Computation and Program Behaviour, Panoramas et Synthèses*, vol. 27, Société Mathématique de France.
Murfet, D. (2015). On Sweedler's cofree cocommutative coalgebra. *Journal of Pure and Applied Algebra* **219** 5289–5304. arXiv preprint [arXiv:1406.5749].
Murfet, D. (2014). *Logic and Linear Algebra: An Introduction*, arXiv preprint, [arXiv:1407.2650].
Rogozhin, Y. (1996). Small universal Turing machines. *Theoretical Computer Science* **168** 215–240.
Roversi, L. (1999). A P-time completeness proof for light logics. In: Flum, J. and Rodriguez-Artalejo, M. (eds.) *Computer Science Logic. CSL 1999. Lecture Notes in Computer Science*, vol. 1683, Springer, Berlin, Heidelberg, 469–483.
Sipser, M. (2006). *Introduction to the Theory of Computation*, Thomson Course Technology.
Sweedler, M. (1969). *Hopf Algebras*, W. A. Benjamin.
Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* **2** (1) 230–265.

Appendix A. Ints, Bints and Bools

In this appendix, we develop the basic properties of denotations of integers, binary integers and booleans in the Sweedler semantics. The denotations of integers and booleans are easily seen to be linearly independent, but binary integers are more interesting.

Proposition A.1. *Let A be a formula with* $\dim[\![A]\!] > 0$. *Then*

(i) The set $\{[\![\underline{n}_A]\!]\}_{n\geq 0}$ is linearly independent in $[\![\boldsymbol{int}_A]\!]$.
(ii) The set $\{[\![\underline{i}_A]\!]\}_{i=0}^{n-1}$ is linearly independent in $[\![_n\boldsymbol{bool}_A]\!]$.

Proof. For (i) suppose that $\sum_{i=0}^{n} c_i[\![\underline{i}]\!] = 0$ for some scalars $c_0, ..., c_n \in k$. Let $p \in k[x]$ be the polynomial $p = \sum_{i=0}^{n} c_i x^i$. For $\alpha \in [\![A \multimap A]\!]$ we have

$$p(\alpha) = \sum_{i=0}^{n} c_i \alpha^i = \sum_{i=0}^{n} c_i [\![\underline{i}]\!] |\emptyset\rangle_\alpha = 0.$$

The minimal polynomial of α therefore divides p. Since this holds for any linear map $\alpha \in [\![A \multimap A]\!]$, it follows that p is identically zero, as k is characteristic zero. The claim in (ii) is clear since the denotations $[\![\underline{i}]\!]_A$ are projections from $[\![A]\!]^{\oplus n}$. □

Corollary A.2. *The function $\mathbb{N} \to [\![\boldsymbol{int}_A]\!]$ given by $n \mapsto [\![\underline{n}_A]\!]$ is injective.*

We now investigate the question of whether binary integers have linearly independent denotations. Surprisingly this does not hold in general. In fact, for an atomic formula A it is impossible to choose $[\![A]\!]$ finite-dimensional such that all binary integers are linearly independent in $[\![\mathbf{bint}_A]\!]$.

For a binary integer T, one can write the value of $[\![\underline{T}_A]\!]$ on arbitrary kets as a sum of its values on vacuum vectors. This will simplify the task of checking whether binary integers have linearly dependent denotations; at least in the case where we have a fixed number of zeroes and ones, we only need to check linear dependence after evaluating on vacuum vectors. From this point onward let A be a fixed type and write $[\![\underline{T}]\!]$ for $[\![\underline{T}_A]\!]$.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Proposition A.3. *For $T \in \{0,1\}^n$, let t_i be the ith digit of T and let $T_0 = \{i \mid t_i = 0\}$ and $T_1 = \{i \mid t_i = 1\}$. Then*

$$[\![\underline{T}_A]\!](|\alpha_1, ..., \alpha_k\rangle_\gamma \otimes |\beta_1, ..., \beta_l\rangle_\delta) = \sum_{f \in \mathrm{Inj}\,([k], T_0)} \sum_{g \in \mathrm{Inj}\,([l], T_1)} \Gamma_n^{f,g} \circ \cdots \circ \Gamma_1^{f,g},$$

where

$$\Gamma_i^{f,g} = \begin{cases} \alpha_{f^{-1}(i)} & i \in \mathrm{Im}\,(f) \\ \beta_{g^{-1}(i)} & i \in \mathrm{Im}\,(g) \\ \gamma & i \in T_0 \setminus \mathrm{Im}\,(f) \\ \delta & i \in T_1 \setminus \mathrm{Im}\,(g). \end{cases}$$

In particular, $[\![\underline{T}_A]\!]$ vanishes on $|\alpha_1, ..., \alpha_k\rangle_\gamma \otimes |\beta_1, ..., \beta_l\rangle_\delta$ if $k > |T_0|$ or $l > |T_1|$.

Proof. This follows in the same way as Clift and Murfet (2017, Lemma 4.11). □

Proposition A.4. *Let $m, n \geq 0$, and let $T \in \{0,1\}^*$ contain exactly m zeroes and n ones. Then for $\alpha_i, \beta_i, \gamma, \delta \in [\![A \multimap A]\!]$ we have:*

$$[\![\underline{T}]\!](|\alpha_1, ..., \alpha_m\rangle_\gamma, |\beta_1, ..., \beta_n\rangle_\delta) = \sum_{I \subseteq [m]} \sum_{J \subseteq [n]} (-1)^{m-|I|}(-1)^{n-|J|} [\![\underline{T}]\!] \left(|\emptyset\rangle_{\sum_{i \in I} \alpha_i}, |\emptyset\rangle_{\sum_{j \in J} \beta_j} \right). \tag{$*$}$$

Note that the right-hand side does not depend on γ and δ.

Proof. A single term of the double sum on the right-hand side of ($*$) corresponds to replacing (in the reversal of T) each 0 with $\sum_i \alpha_i$ and each 1 with $\sum_j \beta_j$. After expanding these sums, consider the coefficient of a particular noncommutative monomial p which contains only the variables $\alpha_{i_1}, ..., \alpha_{i_k}, \beta_{j_1}, ..., \beta_{j_l}$, where $\{i_1, ..., i_k\} \subseteq [m]$ and $\{j_1, ..., j_l\} \subseteq [n]$. The terms in the right-hand side of ($*$) which contribute to this coefficient are precisely those for which the set I contains each of $i_1, ..., i_k$, and J contains each of $j_1, ..., j_l$. Hence the coefficient of p is

$$\begin{aligned} \sum_{\substack{\{i_1,...,i_k\} \subseteq I \subseteq [m] \\ \{j_1,...,j_l\} \subseteq J \subseteq [n]}} (-1)^{m-|I|}(-1)^{n-|J|} &= \sum_{i=k}^{m} \sum_{j=l}^{n} (-1)^{m-i}(-1)^{n-j} \binom{m-k}{i-k} \binom{n-l}{j-l} \\ &= \sum_{i=k}^{m} (-1)^{m-i} \binom{m-k}{i-k} \sum_{j=l}^{n} (-1)^{n-j} \binom{n-l}{j-l} \\ &= \begin{cases} 1 & m = k \text{ and } n = l \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

So the only monomials p with non-zero coefficient are those which contain each α_i and each β_j exactly once, which agrees with the left-hand side of ($*$) by Proposition A.3. □

In order to make the content of the above proposition clear, we give a concrete example involving a specific binary sequence.

Example A.5. Let $T = 0010$. The left-hand side of ($*$) is therefore

$$\sum_{\sigma \in S_3} \alpha_{\sigma(1)} \beta_1 \alpha_{\sigma(2)} \alpha_{\sigma(3)}$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

while the right-hand side is

$$(\alpha_1+\alpha_2+\alpha_3)\beta_1(\alpha_1+\alpha_2+\alpha_3)(\alpha_1+\alpha_2+\alpha_3)-(\alpha_1+\alpha_2)\beta_1(\alpha_1+\alpha_2)(\alpha_1+\alpha_2)-$$
$$(\alpha_1+\alpha_3)\beta_1(\alpha_1+\alpha_3)(\alpha_1+\alpha_3)-(\alpha_2+\alpha_3)\beta_1(\alpha_2+\alpha_3)(\alpha_2+\alpha_3)+$$
$$\alpha_1\beta_1\alpha_1\alpha_1+\alpha_2\beta_1\alpha_2\alpha_2+\alpha_3\beta_1\alpha_3\alpha_3.$$

After expanding the above, consider for example the monomial $\alpha_1\beta_1\alpha_1\alpha_1$:

$$(\alpha_1+\alpha_2+\alpha_3)\beta_1(\alpha_1+\alpha_2+\alpha_3)(\alpha_1+\alpha_2+\alpha_3)-(\alpha_1+\alpha_2)\beta_1(\alpha_1+\alpha_2)(\alpha_1+\alpha_2)-$$
$$(\alpha_1+\alpha_3)\beta_1(\alpha_1+\alpha_3)(\alpha_1+\alpha_3)-(\alpha_2+\alpha_3)\beta_1(\alpha_2+\alpha_3)(\alpha_2+\alpha_3)+$$
$$\alpha_1\beta_1\alpha_1\alpha_1+\alpha_2\beta_1\alpha_2\alpha_2+\alpha_3\beta_1\alpha_3\alpha_3.$$

As expected, the coefficient is $\binom{2}{2}-\binom{2}{1}+\binom{2}{0}=1-2+1=0$.

Corollary A.6. *Let $m, n \geq 0$, let $T \in \{0,1\}^*$ contain exactly m zeroes and n ones, and let $0 \leq k \leq m$ and $0 \leq l \leq n$. Then*

$$[\![\underline{T}]\!](|\alpha_1,...,\alpha_k\rangle_\gamma, |\beta_1,...,\beta_l\rangle_\delta) = \sum_{I\subseteq[m]}\sum_{J\subseteq[n]} \frac{(-1)^{m-|I|}(-1)^{n-|J|}}{(m-k)!(n-l)!}[\![\underline{T}]\!]\left(|\emptyset\rangle_{\sum_{i\in I}\alpha_i}, |\emptyset\rangle_{\sum_{j\in J}\beta_j}\right),$$

where on the right-hand side we define $\alpha_i = \gamma$ for $i > k$ and $\beta_j = \delta$ for $j > l$.

Proof. Note that by Proposition A.3

$$[\![\underline{T}]\!](|\alpha_1,...,\alpha_k\rangle_\gamma, |\beta_1,...,\beta_l\rangle_\delta) = \frac{[\![\underline{T}]\!](|\alpha_1,...,\alpha_k,\gamma,...,\gamma\rangle_\gamma, |\beta_1,...,\beta_l,\delta,...,\delta\rangle_\delta)}{(m-k)!(n-l)!},$$

where the kets on the right have length m and n respectively. Then apply the previous proposition. □

Lemma A.7. *Let $T_1,...,T_r \in \{0,1\}^*$ each contain exactly m zeroes and n ones, and let $c_1,...,c_r \in k \setminus \{0\}$. Then $\sum_s c_s[\![\underline{T_s}]\!] = 0$ in $[\![\mathbf{bint}_A]\!]$ if and only if $\sum_s c_s[\![\underline{T_s}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) = 0$ in $[\![A \multimap A]\!]$ for all $\alpha, \beta \in [\![A \multimap A]\!]$.*

Proof. ($\Rightarrow$) is immediate. For ($\Leftarrow$), suppose that $\sum_{s=1}^r c_s[\![\underline{T_s}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) = 0$ for all α, β. It follows from Corollary A.6 that

$$\begin{aligned}
&\sum_{s=1}^r c_s[\![\underline{T_s}]\!](|\alpha_1,...,\alpha_k\rangle_\gamma, |\beta_1,...,\beta_l\rangle_\delta)\\
&= \sum_{I\subseteq[m]}\sum_{J\subseteq[n]} \frac{(-1)^{m-|I|}(-1)^{n-|J|}}{(m-k)!(n-l)!}\sum_{s=1}^r c_s[\![\underline{T_s}]\!]\left(|\emptyset\rangle_{\sum_{i\in I}\alpha_i}, |\emptyset\rangle_{\sum_{j\in J}\beta_j}\right)\\
&= \sum_{I\subseteq[m]}\sum_{J\subseteq[n]} \frac{(-1)^{m-|I|}(-1)^{n-|J|}}{(m-k)!(n-l)!}\cdot 0\\
&= 0,
\end{aligned}$$

where we again define $\alpha_i = \gamma$ for $i > k$ and $\beta_j = \delta$ for $j > l$. □

Suppose that A is a formula with $\dim[\![A]\!] = n < \infty$. By the above lemma, the existence of distinct binary integers with linearly dependent denotations reduces to the task of finding a non-zero noncommutative polynomial $t_n(x,y)$ such that $t_n(\alpha,\beta) = 0$ for all $n \times n$ matrices $\alpha, \beta \in [\![A \multimap A]\!]$. To describe such a polynomial, we will require the following theorem.

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

Theorem A.8. ***(Amitsur–Levitzki Theorem).*** *For $n \in \mathbb{N}$, let $k\langle x_1, ..., x_n\rangle$ denote the ring of non-commutative polynomials in n variables, and let $s_n \in k\langle x_1, ..., x_n\rangle$ be the polynomial*

$$s_n = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) x_{\sigma(1)} \cdots x_{\sigma(n)}.$$

Then for all $\alpha_1, ..., \alpha_{2n} \in M_n(k)$, we have $s_{2n}(\alpha_1, ..., \alpha_{2n}) = 0$. Furthermore, $M_n(k)$ does not satisfy any polynomial identity of degree less than $2n$.

Proof. See Drensky and Formanek (2004, Theorem 3.1.4). □

Corollary A.9. *For $n \in \mathbb{N}$, there exists a non-zero polynomial $t_n \in k\langle x, y\rangle$ such that for all $\alpha, \beta \in M_n(k)$ we have $t_n(\alpha, \beta) = 0$.*

Proof. The polynomial $t_n(x, y) = s_{2n}(x, xy, ..., xy^{2n-1})$ is non-zero and satisfies the desired property. □

Proposition A.10. *For any formula A such that $\dim[\![A]\!] < \infty$, there exist distinct binary integers $T_1, ..., T_r \in \{0, 1\}^*$ such that $[\![\underline{T_1}]\!], ..., [\![\underline{T_r}]\!]$ are linearly dependent in $[\![\boldsymbol{bint}_A]\!]$.*

Proof. Let $n = \dim[\![A]\!]$, so that $[\![A \multimap A]\!] \cong M_n(k)$. For $1 \leq i \leq 2n$, let R_i be the binary integer $1^{i-1}0$. Note that for all $\alpha, \beta \in M_n(k)$ we have

$$\sum_{\sigma \in S_{2n}} \operatorname{sgn}(\sigma) [\![\underline{R_{\sigma(2n)} \cdots R_{\sigma(1)}}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) = t_n(\alpha, \beta) = 0.$$

Hence $\sum_{\sigma \in S_{2n}} \operatorname{sgn}(\sigma) [\![\underline{R_{\sigma(2n)} \cdots R_{\sigma(1)}}]\!] = 0$ by Lemma A.7. □

Remark A.11. Note that despite the above proposition, if we have a *particular* finite collection of binary integers $T_1, ..., T_r$ in mind it is always possible for A atomic to choose $[\![A]\!]$ such that $[\![\underline{T_1}]\!], ..., [\![\underline{T_r}]\!]$ are linearly independent in $[\![\mathbf{bint}_A]\!]$. To see this, let d denote the maximum length of the T_s, and note that a linear dependence relation between the $[\![\underline{T_s}]\!]$ gives rise to a polynomial identity for $M_n(k)$ of degree d, where $n = \dim[\![A]\!]$. By Amitsur–Levitzki we must therefore have $d \geq 2n$, so if we choose $\dim[\![A]\!] > d/2$ then $[\![\underline{T_1}]\!], ..., [\![\underline{T_r}]\!]$ must be linearly independent.

In addition, while linear independence does not always hold for an arbitrary collection of binary integers, it turns out that we do have linear independence for any *pair* of distinct binary integers, so long as $\dim[\![A]\!]$ is at least 2.

Proposition A.12. *Let A be a formula with $\dim[\![A]\!] \geq 2$. The function $\{0, 1\}^* \to [\![\boldsymbol{bint}_A]\!]$ which maps S to $[\![\underline{S}]\!]$ is injective.*

Proof. Let $n = \dim[\![A]\!]$. For simplicity of notation we suppose that n is finite, as the case where n is infinite is similar. Consider the subgroup G of $GL_n(k)$ generated by

$$\alpha = \begin{bmatrix} 1 & 2 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{and} \quad \beta = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 2 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

https://doi.org/10.1017/S0960129520000109 Published online by Cambridge University Press

It is well known that G is freely generated by α and β; see de la Harpe (2000, Section II.B). Suppose that $[\![\underline{S}]\!] = [\![\underline{T}]\!]$, so that in particular we have $[\![\underline{S}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) = [\![T]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta)$. In other words, the composite obtained by substituting α for zero and β for one into the digits of S is equal to the corresponding composite for T. Since α and β generate a free group, it follows that $S = T$. □

Proposition A.13. *Let A be a formula with $\dim[\![A]\!] \geq 2$, and let $S, T \in \{0,1\}^*$ with $S \neq T$. The denotations $[\![\underline{S}]\!], [\![\underline{T}]\!]$ are linearly independent in $[\![\boldsymbol{bint}_A]\!]$.*

Proof. Suppose we are given $S, T \in \{0,1\}^*$ such that $a[\![\underline{S}]\!] + b[\![\underline{T}]\!] = 0$ for some $a, b \neq 0$. With α, β as above, it follows that

$$[\![\underline{S}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) \circ [\![\underline{T}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta)^{-1} = -\frac{b}{a} I$$

So $[\![\underline{S}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta) \circ [\![\underline{T}]\!](|\emptyset\rangle_\alpha, |\emptyset\rangle_\beta)^{-1}$ is in the center of G, which is trivial since G is free of rank 2, and hence $a = -b$. It follows that $[\![\underline{S}]\!] = [\![\underline{T}]\!]$ and therefore $S = T$ by the previous proposition. □

Cite this article: Clift J and Murfet D (2020). Encodings of Turing machines in linear logic. *Mathematical Structures in Computer Science* **30**, 379–415. https://doi.org/10.1017/S0960129520000109