On the Proof Theory of Program Transformations

Martin C. Henson
Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester, Essex, England
email: hensm@uk.ac.sx, fax: +44 206 872788

§1 Abstract

We provide an intensional semantics for certain elementary program transformations by describing a translation from these transformations to the derivations of a simple theory of operations and types and we show that this semantics is intensionally faithful. Our objective is to understand more precisely the intensional structure of a class of semi-formal program derivations.

Keywords: *constructive set theory, program transfromation, intensional semantics*

§2 Introduction

This paper continues our study of the proof theory of certain elementary program derivations: those obtained by the techniques of *transformational programming* (*e.g.* [BuD77] [Bir84] [Hen88]) from functional programs. In our earlier work [Hen93] we concentrated exclusively on transformations over the natural numbers. In this paper we wish to extend this work towards algebraic types in general. These data-types are, in their full generality, significantly more problematic than the almost pathologically simple special case of the natural numbers. In view of this we shall be restricting ourselves, in this paper, to transformations from unary operations on binary trees. This datatype appears to require us to introduce all the difficulties necessary for dealing with the general case: difficulties which were not at all apparent in [Hen93].

§2.1 General background

Transformational programming, like program derivation in a constructive type theory [Con86] [Bac89] [HaN87] [Hen89*a*] [Tho91], is a methodology in which programs are obtained by reasoning from specifications. However, it is well known that the 'calculus' of transformations is unsound. Thus, strictly speaking, every transformation induces a correctness proof obligation. Some attempts to characterise correct transformations have been undertaken, [Kot78] [Kot85] for example, but these do not give much insight into the *structure* of transformations as putative proofs of equivalence. What is less well known is that transformations can effect proof-theoretic non-trivialities, for example, one can obtain programs whose termination proof requires Π^0_2-induction from programs whose termination proof requires only Σ^0_1- induction apparently by *equational manipulation!* Furthermore, transformations are arguments couched in a language conceptually very close to the programs themselves and are evidently logic-free. These observations, combined in particular with similar examples of *proof theoretic sleight of hand*, go some way to explain why program transformation is such a success story of functional languages and of software science in general.

The overall objective of our research, of which this paper is one contribution, is to investigate the proof theoretic structure of program transformations. The purpose is two-fold. Firstly, a proof theoretic semantics provides a mechanism for guaranteeing the correctness of transformations and such a mechanism can be incorporated within a computer-aided program development environment [San93]. Secondly, and much more importantly, such a semantics helps to uncover the logical properties underlying a regime of *semi-formal reasoning*. Understanding this, we hope,

will help us to bridge the *formality gap* between practical program development on the one hand and distressingly complex formal proofs within some foundational theory on the other [Hen91].

§2.2 Organisation of the paper

The plan of the paper is as follows. In the next section we introduce a simple theory of operations and types, $\mathcal{EOB}$, into which we may express functional programs and transformations. §4 is devoted to the intensional semantics of transformations within $\mathcal{EOB}$ and the proof of the main theorem which demonstrates that the computational content of the semantics of a transformation is the computational content of the transformation itself, up to intensional equality. §5 provides a simple but illustrative example of the translation. Finally, in §6, we make some concluding remarks.

§3 Programs, transformations and the theory $\mathcal{EOB}$

§3.1 The elementary theory of operations and trees

In [Hen93] we worked with $\mathcal{EON}$ of [Bee85], a theory of operations based on the logic of partial terms[1]. In this context we shall require a modification of this. The theory $\mathcal{EOB}$ (Elementary theory of Operations and Binary trees) can be obtained from $\mathcal{EON}$ simply by replacing the introduction rules and elimination rule for natural numbers with the following rules for binary trees[2].

$$\frac{}{\Gamma \vdash \mathsf{Leaf} \in B} \qquad \frac{\Gamma \vdash t_0 \in B \quad \Gamma \vdash t_1 \in B}{\Gamma \vdash (\mathsf{Node}\ t_0\ t_1) \in B}$$

$$\frac{\Gamma \vdash \varphi(\mathsf{Leaf}) \quad \Gamma,\ t_1 \in B,\ t_2 \in B,\ \varphi(t_1),\ \varphi(t_2) \vdash \varphi(\mathsf{Node}\ t_1\ t_2)}{\Gamma,\ x \in B \vdash \varphi(x)}$$

There are other, minor, differences in our presentation: we write $x \in B$ and $f \in B \to B$ (*etc.*) whereas $B(x)$ and $(\forall x)(B(x) \supset B(f\,x))$ would be more in the spirit of [Bee85]. Also, we present the logic as a system of natural deduction in sequent form (whereas we have a Hilbert system in [Bee85]). We also assume, as in [Bee85], that lambda abstraction is defined *à la* Curry. Finally, we let lower case greek letters range over individual formulae and upper case greek letters over sequences of formulae. Of special interest will be sequences composed of formulae each with exactly one free variable. In this case $\Psi(z)$ will denote the sequence $\psi_1(z_1) \ldots \psi_n(z_n)$ *etc.*

The most important property of $\mathcal{EOB}$ in connection with the formalisation of functional programs is the recursion theorem.

Theorem 3.1.1 *There is a term R such that for any term f:* $\vdash R\,f\!\downarrow \wedge (\forall x)(R\,f\,x \approx f\,(R\,f)\,x)$. •

We use δ (suitably decorated with diacriticals) to range over ***Der***($\mathcal{EOB}$), the *derivations* of $\mathcal{EOB}$[3].

---

1 In partial term logic one writes $t\!\downarrow$ for the atomic formula which asserts that the term t denotes. Rules for existential introduction and universal elimination are modified, by adding premises in an obvious way, to ensure that the terms they exhibit actually denote (see [Bee85], for example, for more detail).

2 In fact one would more usefully *add* these to rules to $\mathcal{EON}$ and allow the trees to carry natural numbers at the leaves. This we will assume in §5 but for the technical development it would simply complicate the presentation without conceptual benefit.

3 These need not be closed derivations; so, for example, every sequent belongs to ***Der***($\mathcal{EOB}$).

§3.2 Programs

Programs[4] are *ensembles*, $f\,\boldsymbol{P} \approx_{\text{def}} \boldsymbol{e}$, of *recursion equations*[5] where f is an *operation name*, $\boldsymbol{P}$ is an $n \times m$-matrix of *patterns* with typical element $p = \boldsymbol{P}_{ij}$ and typical row sequence $\boldsymbol{p} = \boldsymbol{P}_i$, when $i \in n$, $j \in m$, whilst $\boldsymbol{e}$ is an m-ary sequence of terms with typical element $\boldsymbol{e}_i$. We let B range over ensembles and β over recursion equations. We will write $\wedge$B for conjunction of the equations comprising B. Patterns are terms which are built up from Leaf, Node and variables by application. If x occurs in a pattern, p, we may write $p(x)$ (*etc.*). $\mathcal{V}(e)$ denotes the set of variables occurring free in e (*etc.*). In an equation $f\,\boldsymbol{p} \approx e$ we require $\mathcal{V}(e) \subseteq \mathcal{V}(\boldsymbol{p})$.

An ensemble, B, is *well-typed* when there exists a derivation in the following type assignment system with B ***wellformed*** as conclusion[6]. In what follows Γ is a *context*, a set of *typings* of the form $x \,\mathrm{E}\, T$ (with each such x distinct) where $T ::= B \mid T \to T$.

$$\frac{(x \,\mathrm{E}\, T) \in \Gamma}{\Gamma \vdash x \,\mathrm{E}\, T}\ (var) \qquad \frac{}{\Gamma \vdash \mathsf{Leaf} \in B}\ (leaf) \qquad \frac{\Gamma \vdash e_1 \,\mathrm{E}\, T_0 \to T_1 \quad \Gamma \vdash e_2 \,\mathrm{E}\, T_0}{\Gamma \vdash (e_1\, e_2) \,\mathrm{E}\, T_1}\ (app)$$

$$\frac{}{\Gamma \vdash \mathsf{Node} \in B \to B \to B}\ (node) \qquad \frac{\Gamma \vdash e_1 \,\mathrm{E}\, T \quad \Gamma \vdash e_2 \,\mathrm{E}\, T}{\Gamma \vdash e_1 \approx e_2 \,\mathrm{E}\, T}\ (eqn)$$

$$\frac{f \,\mathrm{E}\, T_0, \Gamma_0 \vdash f\,\boldsymbol{P}_0 \approx \boldsymbol{e}_0 \,\mathrm{E}\, T_1 \ \ldots\ f \,\mathrm{E}\, T_0, \Gamma_{n\text{-}1} \vdash f\,\boldsymbol{P}_{n\text{-}1} \approx \boldsymbol{e}_{n\text{-}1} \,\mathrm{E}\, T_1}{\vdash f\,\boldsymbol{P} \approx_{\text{def}} \boldsymbol{e}\ \textit{wellformed}}\ (sys)$$

We shall always assume that we are working with well-typed equations, consequently we shall write $e \,\mathrm{E}\, T$ to indicate that the expression (*etc.*) is assigned the type T in the appropriate well-typing.

We need to classify a particularly well behaved subset of the ensembles: those which exhaust their domains of definitions without overlap. To do this we first require the following.

Let $\theta \in \mathit{SUBST}$ where $\theta ::= \{[x \leftarrow e]\}^*$. Simultaneous substitution (with respect to all free occurrences) is denoted $e\theta$ (*etc.*). We shall write $e_0 \leq_\theta e_1$ when $e_0 = e_1\theta$ and $e_0 \leq e_1$ when $e_0 \leq_\theta e_1$ for some θ. We write $\theta_0{*}\theta_1$ for substitution concatenation.

Definition 3.2.1 Let $f\,\boldsymbol{P} \approx_{\text{def}} \boldsymbol{e}$ be an ensemble with $f \,\mathrm{E}\, \boldsymbol{T} \to T$. $f\,\boldsymbol{P} \approx_{\text{def}} \boldsymbol{e}$ is

(*i*) *complete iff* for every $\boldsymbol{v} \in \boldsymbol{T}$ there exists a $j \in m$ such that, for all $i \in n$, $\boldsymbol{v}_i \leq \boldsymbol{P}_{ij}$

(*ii*) *non-overlapping iff* for every $\boldsymbol{v} \in \boldsymbol{T}$, if whenever there exist $j, k \in m$ such that, for all $i \in n$ $\boldsymbol{v}_i \leq \boldsymbol{P}_{ij}$ and $\boldsymbol{v}_i \leq \boldsymbol{P}_{ik}$, then $j = k$.

(*iii*) a *partition iff* it is complete and non-overlapping. •

We shall insist, from now on, that our ensembles are *always* partitions as this is a necessary (but by no means sufficient) condition that they specify total operations of $\mathcal{EOB}$.

---

4 Our notation is similar to that of Miranda (a trademark of Research Software Limited) [TuD85].

5 We use *weak* equality (that is $t_0 \approx t_1$ *iff def* $t_0{\downarrow} \vee t_1{\downarrow} \supset t_0 = t_1$) because we are allowing general recursion here. In fact we will study termination preserving transformations from systems of equations known to specify *total* functions. Thus, we will shift to strong equality in the sequel when the context permits.

6 We have, unfortunately, *three* membership relations to contend with in this paper: $\in$ is the membership in $\mathcal{EOB}$, ε is membership in the meta-language, and the relation, E, introduced here is (in due course) a defined, partial membership in $\mathcal{EOB}$.

Proposition 3.2.2 *For every ensemble* B *there is an* $\mathcal{EOB}$ *term* w *such that* $\vdash \wedge B[f \leftarrow w]$ *and whenever* B *contains the equation* $f\,\boldsymbol{p} \approx_{\text{def}} e$ *we have* $w\,\boldsymbol{p} \rightarrow_\beta^* e[f \leftarrow w]$.

Proof. This is a consequence of Theorem 3.1.1, standard techniques for translating recursion equations into combinatory algebra [TuD79] [Hen89*b*] [TuR91] and the fact that B is non-overlapping. •

Note that Proposition 3.2.2 is stated so that the witnessing term captures the intension of the ensemble B. This is important as our results concern equalities preserved up to intension and it is clear that $\mathcal{EOB}$ supports many witnesses for the equations which comprise B which are, themselves, not intensionally equal. We can also, when requiring the intension of the ensemble B, utilise B itself as a system of rewrite rules and thus avoid having to deal explicitly with the precise mechanism implicit in the proof above.

Corollary 3.2.3 *Let* $\mathcal{R}$ *be a collection of ensembles. The theory* $\mathcal{EOB} + \mathcal{R}$ *is conservative over* $\mathcal{EOB}$.

Proof. Indeed, Proposition 3.2.2 assures that $\mathcal{EOB} + \mathcal{R}$ is just a definitional extension of $\mathcal{EOB}$. •

If we take $t \in T \Leftrightarrow_{\text{def}} t{\downarrow} \supset t \in T$ (a formula of $\mathcal{EOB}$) we have:

Proposition 3.2.4 *The type assignment system is derivable in* $\mathcal{EOB}$.

Proof. An easy induction on the structure of the derivations of the type assignment system. •

§3.3 Term assignment for $\mathcal{EOB}$

We now introduce a term assignment version, $\mathcal{EOB}^{\mathcal{TA}}$, of $\mathcal{EOB}$. This is given as a system of natural deduction in sequent form and it incorporates a notion of *information loss* [Hen90] for Harrop formulae [Har56] (denoted φ_H). The judgements of the system have the form: $\Upsilon \vdash t : \varphi$ or $\Upsilon \vdash \varphi_H$ where Υ, the *assignment context*, is a set of *variable assignments*. A variable assignment is either a Harrop formula or has the form $x : \varphi$ for some variable x and non-Harrop formula φ. Since Υ is a set we do not need rules of exchange or contraction. We will write Γ_Υ for the $\mathcal{EOB}$ context formed from Υ by removing the variable prefixes. Similarly, given an $\mathcal{EOB}$ context Γ, we write Υ_Γ for the assignment context consisting of the Harrop formulae in Γ and variable assignments $x : \varphi$ for each non Harrop φ in Γ (with each such x distinct). We shall use μ (suitably decorated) to range over $\boldsymbol{Der}(\mathcal{EOB}^{\mathcal{TA}})$. There is a evidently a bijection (up to renaming of variables) between closed derivations in $\boldsymbol{Der}(\mathcal{EOB})$ and $\boldsymbol{Der}(\mathcal{EOB}^{\mathcal{TA}})$ which we will denote like this: $\delta \mapsto \mu_\delta$, $\mu \mapsto \delta_\mu$. We will write $t : \delta$ when t is assigned to the root sequent of the derivation μ_δ. When space prevents the convenient display of derivation fragments in full we will sometimes write typing judgements in a context, for example $x \in B$, as x, the variable alone.

We shall not display the system $\mathcal{EOB}^{\mathcal{TA}}$ in full as, for the most part, it can be viewed as an explicit proof of soundness for an abstract realizability interpretation similar to that given for $\mathcal{EON}$ in [Bee85]. However, since this interpretation incorporates a notion of information loss we will provide the rules governing the existential quantifier, the treatment of Harrop antecedents and the induction principle since these are central to our project. In these rules a formula of the form φ (that is: *not* subscripted with H) is assumed to be *non*-Harrop.

$$\frac{\Upsilon \vdash t_1 : \varphi_H(x \leftarrow t_2) \quad \Upsilon \vdash t_2\downarrow}{\Upsilon \vdash t_2 : (\exists x)\varphi_H} \qquad \frac{\Upsilon \vdash t_1 : (\exists x)\varphi_H \quad \Upsilon, \varphi_H[x\leftarrow y] \vdash t_2 : \eta}{\Upsilon \vdash t_2[y \leftarrow t_1] : \eta}$$

$$\frac{\Upsilon \vdash t_1 : (\forall x)(\varphi_H \supset \psi) \qquad \Upsilon \vdash \varphi_H[x\leftarrow t_2]}{\Upsilon \vdash t_1\ t_2 : \psi[x\leftarrow t_2]}$$

$$\frac{\Upsilon \vdash t_1 : \varphi(\mathsf{Leaf}) \quad \Upsilon, s_1 \in B, s_2 \in B, \varphi(s_1), \varphi(s_2) \vdash t_2 : \varphi(\mathsf{Node}\ s_1\ s_2)}{\Upsilon, x \in B \vdash \textit{trec}\ (\lambda n.t_1)(\lambda s_1 s_2.\lambda v_1 v_2.t_2)\ x : \varphi(x)}\ (B\text{-}elim)^7$$

In this last rule the term ***trec*** satisfies the equations:

$\textit{trec}\ z\ g\ \mathsf{Leaf} = z$

$\textit{trec}\ z\ g\ (\mathsf{Node}\ s_1\ s_2) = g\ s_1\ s_2\ (\textit{trec}\ z\ g\ s_1)\ (\textit{trec}\ z\ g\ s_2)$

and is constructed as in the proof of Proposition 3.2.2.

$\mathcal{EOB}$is a *non-extensional* theory; that is to say its primitive equality is the intensional equality of the underlying theory of operations. We shall need a defined notion of *extensional* equality.

Definition 3.3.1

(*i*) $\Gamma \vdash t_0 \equiv_B t_1$ *iff* $\Gamma \vdash t_0 \in B \wedge t_1 \in B \wedge t_0 = t_1$

(*ii*) $\Gamma \vdash t_0 \equiv_{T_0\to T_1} t_1$ *iff* $\Gamma \vdash (\forall x_0, x_1 \in T_0)(x_0 \equiv_{T_0} x_1 \supset t_0\ x_0 \equiv_{T_1} t_1\ x_1)$ •

We need to use this to define the notion of an *extensional specification*. In this paper we are only concerned with *unary* operations.

Definition 3.3.2 Let ψ be a specification of the form: $(\forall x \in B)(\exists y \in B)\varphi_H(x, y)$.

ψ is an *extensional specification iff whenever* $\Upsilon \vdash t_0 : \psi$ *and* $\Upsilon \vdash t_1 : \psi$ *then* $\Gamma_\Upsilon \vdash t_0 \equiv_{B\to B} t_1$. •

§3.4 Transformations

For the sequel we shall take $\mathcal{R}$to be some, fixed, collection of ensembles. Let $E_{\mathcal{R}}$ denote that subset of all expressions which are composed solely of variables, data-constructors and operation names occurring in $\mathcal{R}$and closed under application. We write $\mathcal{L}$ for the collection of equations, $e_1 \approx e_2$, where $e_1, e_2 \in E_{\mathcal{R}}$ and whose universal closures are consequences of $\mathcal{EOB} + \mathcal{R}$. Note in particular that the universal closures of the equations in $\mathcal{R}$are in $\mathcal{L}$.

Central to transformational programming is the ability to perform certain substitutions upon recursion equations. First we need: $e[e_0 \Leftarrow e_1]$, the *replacement of specified occurrences of* e_0 *in* e *by* e_1. Such replacements are well-formed only when $\mathcal{V}(e_1 \mathrm{E}\ T) \subseteq \mathcal{V}(e_0 \mathrm{E}\ T)$. The notion of a *specified occurrence* of an expression only serves to label it as such, thus we refrain from labouring this issue any further.

Next we need the relation $e_1 \subseteq e_2$ (e_1 *is a subexpression of* e_2) which is simply the reflexive transitive closure of (*i*) $e_0 \subseteq e_1\ e_2$ *when* $e_0 \subseteq e_1$ (*ii*) $e_0 \subseteq e_1\ e_2$ *when* $e_0 \subseteq e_2$.

We are now in a position to introduce transformations which are to be certain tree structures over the programming notation. The rules for building these trees are as follows.

---

7 We will refer to the variable x as the *induction variable* and the variables s_1 and s_2 as the *eigen variables* of the instance of this rule.

Definition 3.4.1 (Prime Transformations)[8]

(*i*)
$$\frac{(f\ p(x) \approx e)[x \leftarrow \mathsf{Leaf}] \qquad (f\ p(x) \approx e)[x \leftarrow \mathsf{Node}\ s_0\ s_1]}{f\,p(x \,\mathrm{E}\, B) \approx e} \quad (ins)$$

where s_0 and s_1 are fresh variables. A useful, horizontal notation for this is:

$$f\,p(x) \approx e \rightarrowtail_{\mathrm{I}} (f\,p(x) \approx e)[x \leftarrow \mathsf{Leaf}], (f\,p(x) \approx e)[x \leftarrow \mathsf{Node}\ s_0\ s_1]$$

or, more generally: $\beta \rightarrowtail_{\mathrm{I}} \mathrm{B}$ (when the rest of the data is understood).

(*ii*)
$$\frac{e_0 \subseteq e \qquad e_1 \approx e_2\, \varepsilon\, \mathcal{L} \qquad e_0 \leq_\theta e_1 \qquad f\,p \approx e[e_0 \Leftarrow e_2\theta]}{f\,p \approx e} \quad (law)$$

Horizontally: $f\,p \approx e \rightarrowtail_{\mathrm{L}} f\,p \approx e[e_0 \Leftarrow e_2\theta]$ or generally: $\beta_1 \rightarrowtail_{\mathrm{L}} \beta_2$ (when the other data is understood).

(*iii*)
$$\frac{e_0 \subseteq e \qquad g\,\boldsymbol{q} \approx d\, \varepsilon\, \mathcal{L} \qquad e_0 \leq_\theta g\,\boldsymbol{q} \qquad f\,p \approx e[e_0 \Leftarrow d\theta]}{f\,p \approx e} \quad (unf)$$

Horizontally: $f\,p \approx e \rightarrowtail_{\mathrm{U}} f\,p \approx e[e_0 \Leftarrow d\theta]$ or generally: $\beta_1 \rightarrowtail_{\mathrm{U}} \beta_2$ (when the other data is understood).

(*iv*)
$$\frac{e_0 \subseteq e \qquad d \approx g\,\boldsymbol{q}\, \varepsilon\, \mathcal{L} \qquad e_0 \leq_\theta d \qquad f\,p \approx e[e_0 \Leftarrow g\,\boldsymbol{q}\theta]}{f\,p \approx e} \quad (fld)$$

We need to distinguish between two classes of fold. Firstly, the *benign* folds when g and f are distinct operation names. Secondly, *serious* folds when g and f are the same operation name. Our horizontal notation will be $\beta_1 \rightarrowtail_{\mathrm{B}} \beta_2$ for benign folds and $\beta_1 \rightarrowtail_{\mathrm{F}} \beta_2$ for serious folds, when, as usual, the other data is understood.

We will refer to the root equation in each of these prime transformations as the *subject* of the transformation. •

We note that (*unf*) and (*fld*) are, in fact just special cases of (*law*). In fact they correspond to distinctive transformational steps and we have therefore distinguished between them. Moreover, our semantics treats serious instances of (*fld*), in particular, in a very different way to (*law*).

A transformation, π, is a tree in the system of rules given in Definition 3.4.1 from a single equation $f\,x = e_r$ where f must be a *fresh* name for a unary operation over B. We will write $\pi(\mathrm{B})$ to indicate the terminal ensemble, B, of π, $\pi(\beta)$ to distinguish one equation, β, among B, $\pi(\beta \rightarrowtail_{\mathrm{X}} \mathrm{B})$ to indicate an X-transformation out of the terminal equation β of $\pi(\beta)$, and $(\beta)\pi$ to indicate the root equation of π. We will write $\mathcal{PT}_B$ for the set of all transformations.

We now briefly elaborate some basic results regarding transformations. A more comprehensive treatment of this material is contained in [Hen93].

---

8 Note that these are *not* proof rules and the statements above the line are *not* premises. Our trees simply provide a useful mechanism for organising the relationship between the equations and auxiliary data which constitute transformations.

Lemma 3.4.2 *If* $\beta \rightarrow_X B$ *then* $\vdash \beta = B$ (where $X \in \{U, B, L, I\}$) •

Lemma 3.4.3 *For every* $(B_0)\pi(B_1)$, *if* B_0 E T *then* B_1 E T. •

Next we impose an order structure upon the prime transformation subscripts: $I < U = B = L < F$

Definition 3.4.4

(*i*) A transformation π is in *canonical form iff* whenever $\beta_0 \rightarrow_{X_0} \cdots \rightarrow_{X_n} \beta_n$ is a path in π, $X_0 \ldots X_n$ is an ascending chain in the subscript order.

(*ii*) $(B_0)\pi_0(B_1)$ *and* $(C_0)\pi_1(C_1)$ *are equivalent iff* $B_0 = C_0$ and $B_1 = C_1$.

(*iii*) $(f\,x \approx e)\pi(B)$ is *correct iff* $f\,x \approx e \equiv_{B \rightarrow B} B$ •

Proposition 3.4.5 *Every transformation* π *can be put into an equivalent canonical form.* •

We will assume from now on that, unless otherwise indicated, our transformations are in canonical form.

§4 Intensional semantics for transformations

In this section we provide a translation: $[\ \_\ ] \in \mathcal{PT}_B \rightarrow \textbf{\textit{Der}}(\mathcal{EOB}) + \{\textbf{\textit{fail}}\}$. When we write $[\ \pi\ ]$, there is an implicit assumption that $[\ \_\ ]$ is not ***fail*** at π. Such assumptions will be taken as read in the sequel to simplify the presentation. For similar reasons of simplicity it is useful to stipulate at the outset that the translation has the following property: *If the translation denotes* ***fail*** *at* π *then it does so at any transformation* π_0 *which extends* π. Our translation, when it denotes an element of ***Der***$(\mathcal{EOB})$ on a transformation π, will guarantee that π is correct[9].

§4.1 Properties of the translation

We need some preliminary notions. For the rest of this section $\varphi[\ \_\ ]$ will always mean the formula $(\exists y \in B)(y = \_)$[10]. We characterise certain special formulae and contexts which arise in the image of the translation.

Definition 4.1.1

(*i*) (*a*) $\supset \varphi$ *is* φ

(*b*) $(\Psi, \psi) \supset \varphi$ *is* $\Psi \supset (\psi \supset \varphi)$

(*ii*) (*a*) $\mathcal{F}(\varphi[e(x)])$ *iff* $\mathcal{V}(\varphi[e(x)]) = \{x\}$.

(*b*) $\mathcal{F}((\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(x, z)]))$ *iff* $\mathcal{F}(\Psi(z)) \wedge \mathcal{V}((\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(x, z)])) = \{x\}$.

We shall call the subformula of the form $\varphi[e]$ the *existential component.*

(*iii*) (*a*) $\mathcal{C}(x \in B)$.

(*b*) $\mathcal{C}(z \in B, \Psi(z))$ *iff* $\mathcal{F}(\Psi(z))$. •

Fact 4.1.2 *A context satisfying* $\mathcal{C}$ *may be written* $z \in \boldsymbol{B}$, $x \in B$, $\Psi(\boldsymbol{u})$ *where* $\boldsymbol{u}$ *is empty when* z *is empty and* $\boldsymbol{u}$ *is* z, x *otherwise.* •

Whenever we have need of Fact 4.1.2 we shall assume, without comment, the relationship between the variable x and the sequences $\boldsymbol{u}$ and z, as described above.

We shall need to be able to measure the complexity of formulae which satisfy the predicate $\mathcal{F}$.

---

9 Hence, in particular, if we begin with equations which denote a *total* function the transformation will preserve that property. We shall now assume that we undertake transformations only with such operations and so we utilise strong equality in our analysis henceforth.

10 $\varphi(x)$ will, as usual, distinguish x among the free variables of φ.

Definition 4.1.3 We define a map *rank* from formulae to natural numbers as follows:
If $\mathcal{F}((\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(x, z)]))$ then $rank((\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(x, z)) =$
$1+\boldsymbol{max}\{rank(\psi_i(z_i)) \mid i \in n\}$ where $\Psi(z)$ is the sequence of formulae $\psi_0(z_0) \ldots \psi_{n-1}(z_{n-1})$. •
We note that $rank(\varphi) = rank(\varphi\theta)$ for any substitution θ and will make use of this fact without further comment.

There is a constraint that we must impose upon the translation: it must satisfy the following lemma. We shall, then, construct the translation and prove this lemma by simultaneous induction over the structure of transformations.

Lemma 4.1.4 (*Coordination Lemma*)
Let $\pi(\mathrm{B})$ *be a transformation. There exists a unique, one-one correspondence between the open sequents,* $\Gamma \vdash \varphi[e_0]$, *of* $[\ \pi\ ]$ *and the equations,* $f\,p = e$, *in* B *such that* $C(\Gamma)$, $\mathcal{V}(p) = \mathcal{V}(\Gamma)$ *and* $e \leq_\theta e_0$ *for some* θ. *Furthermore, if* π *is free of serious folds then* θ *is the identity.* •
This can be extended to establish an injection, ι, from the equations in a transformation π into the sequents of the derivation $[\ \pi\ ]$. We fix a canonical inverse for ι by setting $\iota^{-1}(s) = \beta$ where $\iota(\beta) = s_0$ and $s \ldots s_0 \ldots s_r$ is the path in the derivation $[\ \pi\ ]$ from s to the root sequent s_r and there is no sequent s_1, *including s itself*, such that $s \ldots s_1 \ldots s_0 \ldots s_r$ $(s_1 \neq s_0)$ with s_1 in the range of ι.

§4.2 Translating transformations

Definition 4.2.1 (*Intensional semantics*)
We define a function $[\ \_\ ] \in \mathcal{PT}_B \rightarrow \boldsymbol{Der}(\mathcal{EOB}) + \{\boldsymbol{fail}\}$ by induction over the structure of transformations and we simultaneously prove Lemma 4.1.4. In the derivation fragments which follow we shall often collapse several steps into one (iterated eliminations of implication or the universal quantifier for example), we omit the occasional minor premise (in substitutions and elimination of implication in particular) and we omit altogether certain trivial steps (such as certain operations on the context like weakenings).

Base Case: The transformation π consists solely of the eureka equation $f\,x = e_r$. We set $[\ \pi\ ] =_{\mathrm{def}}$ $x \in B \vdash \varphi[e_r]$. It is immediate that the corresponding base case of Lemma 4.1.4 is established. In particular note that the eureka equation is a transformation free of serious folds and the substitution required to mediate between the equation and the sequent is indeed the identity.

For the remaining cases we can assume that π is $\pi_0(\beta_i \rightarrow_{\mathrm{X}} \mathrm{B})$ for some X and equation β_i (say $f\,p \approx e$) which is the subject of the prime transformation X. *Ex hypothesi* (Lemma 4.1.4) we have an unsupported sequent $\Gamma \vdash \varphi[e_1]$ (for some Γ and φ) in $[\ \pi_0\ ]$ corresponding to β_i in π_0 which satisfies $C(\Gamma)$, $\mathcal{V}(p) = \mathcal{V}(\Gamma)$ and $e \leq_\zeta e_1$ for some substitution ζ.

Ad (*unf*):

$$\left[\ \frac{e_0 \subseteq e \qquad g\,\boldsymbol{q} = d \in \mathcal{L} \qquad e_0 \leq_\theta g\,\boldsymbol{q} \qquad f\,p = e[e_0 \Leftarrow d\theta]}{\ldots\ldots\ \dfrac{f\,p = e}{\pi_0}\ \ldots\ldots}\ \right]$$

$=_{\mathrm{def}}$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall(g\ \boldsymbol{q} = d)}}{\Gamma \vdash g\ \boldsymbol{q}\theta = d\theta} \qquad \Gamma \vdash \varphi[e_1][e_0 \Leftarrow d\theta]}{\cdots\cdots \quad \dfrac{\Gamma \vdash \varphi[e_1]}{[\,\pi_0\,]} \quad \cdots\cdots} \textit{(ex hypothesi)}$$

Note that $g\ \boldsymbol{q}\theta$ is e_0 since $e_0 \leq_\theta g\ \boldsymbol{q}$.

The corresponding case of Lemma 4.1.4 follows easily: there is a unique open sequent to place in correspondence with the new equation in π. The context of this sequent is the context of the open sequent from which this derivation fragment extends so $C(\Gamma)$ is immediate. Since p is not changed we also have $\mathcal{V}(p) = \mathcal{V}(\Gamma)$ immediately. Since the transformation is in canonical form and the prime transformation we are considering is an unfold it follows that π_0 is a transformation free of serious folds. Hence we may conclude that ζ is the identity and $e_1 = e$. Hence $\varphi[e_1][e_0 \Leftarrow d\theta] = \varphi[e][e_0 \Leftarrow d\theta] = \varphi[e[e_0 \Leftarrow d\theta]]$ as required.

Cases (*law, benign fld*): Similarly.

Case (*ins*): Since the transformation is in canonical form and the prime transformation we are considering is an instantiation it follows that π_0 is free of serious folds and hence ζ is the identity and $e_1 = e$. We may write the equation as $f\ p(z, x) = e$ where x is the variable we are instantiating. The the corresponding sequent has the form: $z \in \boldsymbol{B}, x \in B, \Psi(\boldsymbol{u}) \vdash \varphi[e(z, x)]$ *ex hypothesi* Lemma 4.1.4 and by Fact 4.1.2.

$$\left[\ \dfrac{(f\ p(z, x) = e)[x \leftarrow \mathsf{Leaf}] \qquad (f\ p(z, x) = e)[x \leftarrow \mathsf{Node}\ t_0\ t_1]}{\cdots\cdots \quad \dfrac{f\ p(z, x) = e}{\pi_0} \quad \cdots\cdots}\ \right]$$

$=_{\text{def}}$

$$\dfrac{\dfrac{z \in \boldsymbol{B}, \Psi(z) \vdash \varphi[e(z, \mathsf{Leaf})]}{\vdash \kappa(\mathsf{Leaf})} \qquad \dfrac{z \in \boldsymbol{B}, t_1 \in B, t_2 \in B, \Psi(z), \kappa(t_1), \kappa(t_2) \vdash \varphi[e(z, \mathsf{Node}\ t_1\ t_2)]}{t_1 \in B, t_2 \in B, \kappa(t_1), \kappa(t_2) \vdash \kappa(\mathsf{Node}\ t_1\ t_2)}\ \textit{(B-elim)}}{\cdots\cdots \quad \dfrac{x \in B \vdash (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])}{z \in \boldsymbol{B}, x \in B, \Psi(\boldsymbol{u}) \vdash \varphi[e(z, x)]}\ (\forall \text{ and } \supset \textit{- elims, weakening}) \quad \cdots\cdots}$$

where $\kappa(x)$ is the formula $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])$.

We place the leftmost (rightmost) open sequent in correspondence with the leftmost (rightmost) equation in the fragments above. This is clearly the only correspondence which satisfies the variable condition. This correspondence requires no new substitutions on the component expressions and since the substitution was, *ex hypothesi*, the identity it remains so in both cases here. This is important because the extended transformation is free of serious folds. Finally, we

note that $C(z \in \boldsymbol{B}, \Psi(z))$ and $C(z \in \boldsymbol{B}, t_1 \in B, t_2 \in B, \Psi(z), \kappa(t_1), \kappa(t_2))$ follow *ex hypothesi* Lemma 4.1.4 from $C(z \in \boldsymbol{B}, x \in B, \Psi(\boldsymbol{u}))$. We have, then, verified Lemma 4.1.4 for this case.

Case (serious fld): Let ζ be the substitution *ex hypothesi* Lemma 4.1.4 such that $e \leq_\zeta e_1$ where the sequent $\Gamma \vdash \varphi[e_1]$ is in correspondence with the equation $f\,p = e$ on which the fold takes place.

$$\left[\begin{array}{c} e_0 \subseteq e \qquad d = g\,\boldsymbol{q} \in \mathcal{L} \qquad e_0 \leq_\theta d \qquad f\,p = e[e_0 \Leftarrow g\,\boldsymbol{q}\theta] \\ \hline \ldots\ldots \quad f\,p = e \quad \ldots\ldots \\ \hline \pi_0 \end{array}\right]$$

$=_{\text{def}}$

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma^- \vdash \Psi(z\xi_0)\end{array} \qquad \dfrac{\dfrac{}{\Gamma \vdash (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_2(z,x)])}}{\Gamma \vdash \Psi(z\xi_0) \supset \varphi[e_2(z\xi_0, x)]}}{\Gamma \vdash \varphi[e_0]} \qquad \dfrac{\Gamma \vdash \varphi[e_1[\,e_0 \Leftarrow w]] \qquad \dfrac{}{\Gamma, w = e_0 \vdash w = e_0}}{\Gamma, w = e_0 \vdash \varphi[e_1]}$$

$$\dfrac{\ldots\ldots \quad \dfrac{\Gamma \vdash \varphi[e_1]}{[\,\pi_0\,]} \quad \ldots\ldots}{\text{(ex hypothesi)}}$$

where the assumption $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_2(z, x)])$ is chosen such that $e_0 \leq_{\xi_0} e_2 \leq_{\xi_1} e_r$[11] and Γ^- is the context Γ with this assumption removed. This derivation is then completed, for each of the formulae comprising $\Psi(z\xi_0)$, as follows.

If $(\forall \boldsymbol{v} \in \boldsymbol{B})(\Phi(\boldsymbol{v}) \supset \varphi[e_3(\boldsymbol{v}, z\xi_0)])$ occurs in Γ^-, then we have:

$$\dfrac{}{\Gamma^- \vdash (\forall \boldsymbol{v} \in \boldsymbol{B})(\Phi(\boldsymbol{v}) \supset \varphi[e_3(\boldsymbol{v}, z\xi_0)])}$$

Otherwise:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ (\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v}))^- \vdash \Xi(w\zeta)\end{array} \qquad \dfrac{\dfrac{}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v}) \vdash (\forall w \in \boldsymbol{B})(\Xi(w) \supset \varphi[e_4(w, y)])}}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v}) \vdash \Xi(w\zeta) \supset \varphi[e_3(\boldsymbol{v}, z\xi_0)]}}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v}) \vdash \varphi[e_3(\boldsymbol{v}, z\xi_0)]}}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{B} \vdash \Phi(\boldsymbol{v}) \supset \varphi[e_3(\boldsymbol{v}, z\xi_0)]}}{\Gamma^- \vdash (\forall \boldsymbol{v} \in \boldsymbol{B})(\Phi(\boldsymbol{v}) \supset \varphi[e_3(\boldsymbol{v}, z\xi_0)])}$$

choosing the assumption $(\forall w \in \boldsymbol{B})(\Xi(w) \supset \varphi[e_4(w, y)])$ so that $e_3 \leq_\zeta e_4$. Similarly, the context $(\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v}))^-$ is the context $\Gamma^-, \boldsymbol{v} \in \boldsymbol{B}, \Phi(\boldsymbol{v})$ with the assumption:

---

11 Note that this forces $\xi_0 * \xi_1 = \theta$ and that no assumption need exist with this property. In this case the translation denotes ***fail***.

$(\forall w \in \boldsymbol{B})(\Xi(w) \supset \varphi[e_4(w, y)])$ removed. The process is still incomplete but the task at hand (completing the derivation above each component formula of $\Xi(w\zeta)$ is solved by the same strategy.

It remains to verify the corresponding case of Lemma 4.1.4. Naturally the single new equation introduced in the passage from π_0 to π is placed in correspondence with the single new sequent introduced in the passage from $[\ \pi_0\ ]$ to $[\ \pi\ ]$. The substitution we evidently require is:
$\zeta * [w \leftarrow g\ q\theta]$. •

Fact 4.2.2 *For each* π, *the final sequent of* $[\ \pi\ ]$ *is extensional.* •

We must ensure that Definition 4.2.1 is well-defined, in particular that the iterative process required in the last case terminates. We begin by characterising the non-trivial assumptions which are created by instantiations.

Proposition 4.2.3 *Recall the derivation fragment introduced by the translation in the case for instantiations*:

$$\dfrac{\dfrac{\dfrac{\dfrac{z \in \boldsymbol{B}, \Psi(z) \vdash \varphi[e(z, \mathsf{Leaf})]}{\vdash \kappa(\mathsf{Leaf})} \quad \dfrac{z \in \boldsymbol{B}, t_1 \in B, t_2 \in B, \Psi(z), \kappa(t_1), \kappa(t_2) \vdash \varphi[e(z, \mathsf{Node}\ t_1\ t_2)]}{t_1 \in B, t_2 \in B, \kappa(t_1), \kappa(t_2) \vdash \kappa(\mathsf{Node}\ t_1\ t_2)}}{x \in B \vdash (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])}}{\ldots\ldots \qquad \dfrac{z \in \boldsymbol{B}, x \in B, \Psi(u) \vdash \varphi[e(z, x)]}{[\ \pi_0\ ]} \qquad \ldots\ldots}$$

(*i*) *The sequence of formulae* $\Psi(z), \kappa(t_1), \kappa(t_2)$ *has length* $rank(\kappa(x))+1$.
(*ii*) *If* ψ_0, ψ_1 *and* ψ_2 *are any three formulae drawn from* $\Psi(z), \kappa(t_1), \kappa(t_2)$ *with* $rank(\psi_0) = rank(\psi_1) = rank(\psi_2)$ *then* $\psi_i = \psi_j$, *with* $i \neq j$, *for some* $i, j\ \varepsilon\ 3$.

Proof. By induction on the structure of the transformation. We note that (*ii*) states that there are, at most, two formulae at any rank in the sequence $\Psi(z), \kappa(t_1), \kappa(t_2)$. *Base case*: If the sequences $z \in \boldsymbol{B}$ and $\Psi(u)$ are empty then we create a list of assumptions $t_1 \in B$, $t_2 \in B$, $\kappa(t_1), \kappa(t_2)$ where both $\kappa(t_1)$and $\kappa(t_2)$ have rank 1. Furthermore, $\kappa(t_1), \kappa(t_2)$ has length 2 as required for (*i*). (*ii*) follows immediately. *Induction case*: By Fact 4.1.2 the sequence $\boldsymbol{u}$ is z, x. *Ad* (*i*): We may assume *ex hypothesi* that $\Psi(z, x)$ has length n+1 where the maximum rank of a formula in $\Psi(z, x)$ is n. It is then clear that $\Psi(z), \kappa(t_1), \kappa(t_2)$ has length n+2. *Ad* (*ii*): We may also assume *ex hypothesi* that $\Psi(z, x)$ satisfies condition (*ii*). But then so must $\Psi(z)$ which has *fewer* members. Since $rank(\kappa(x))$ $> \psi$ for any ψ occurring in $\Psi(z)$ we may conclude that condition (*ii*) holds for $\Psi(z), \kappa(t_1), \kappa(t_2)$ as required. •

Corollary 4.2.4 *If* $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])$ *has rank n then* $\Psi(z)$ *is a sequence of formulae of length n*-1 *satisfying condition* (*ii*) *of Proposition 4.2.3*.

Proof. Consider the list of assumptions in which this formula first appears. It must have the form $z \in \boldsymbol{B}, x \in B, y \in B, \Psi(z), (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)]), (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, y)])$. By Proposition 4.2.3(*i*) the length of the sequence $\Psi(z), (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)]), (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, y)])$ is n+1. Thus the length of $\Psi(z)$ must be n-1. Since $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])$ has rank n, the formulae $\Psi(z)$ comprise not more than 2 formulae at any rank $m < n$ by Proposition 4.2.3(*ii*). •

We now extend the notion of rank to contexts.

Definition 4.2.5

(*i*) $rank() = 0$

(*ii*) $rank(z \in B, \Gamma) = rank(\Gamma)$

(*ii*) $rank(\kappa, \Gamma) = 3^{rank(\kappa)} + rank(\Gamma)$ where $\mathcal{F}(\kappa)$ •

Proposition 4.2.6 *Definition 4.2.1 is well-defined.*

Proof. We note that *rank*, extended to contexts, is a map into the ordinals below ω. We may, then, proceed by induction to ω. In the rest of the proof we will refer to the sequents occurring in the derivation fragments for the case of serious folding given in Definition 4.2.1. It is sufficient to demonstrate that the process outlined in that case reduces the rank of the context from which the assumptions are drawn at each stage. The rank of the context Γ, from which our first assumption $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_2(z, x)])$ is drawn, is given by $rank(\Gamma) = 3^{rank(\kappa_0)} + rank(\Gamma^-)$ where κ_0 is $(\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_2(z, x)])$. On the other hand the rank of the context $\Gamma^-, v \in \boldsymbol{B}, \Phi(v)$, from which the subsequent assumption $(\forall w \in \boldsymbol{B})(\Xi(w) \supset \varphi[e_4(w, y)])$ is drawn, is given by $rank(\Gamma^-, v \in S, \Phi(v)) = rank(\Gamma^-) + rank(\Phi(v))$ so we must show that $rank(\Phi(v)) < 3^{rank(\kappa_0)}$. We know that $rank(\kappa_1) < rank(\kappa_0)$ where κ_1 is $(\forall v \in \boldsymbol{B})(\Phi(v) \supset \varphi[e_3(v, y)])$ because $\kappa_1 \varepsilon \Psi(z\xi_0)$, and $\Psi(z)$ occurs in κ_0, hence $3^{rank(\kappa_1)} < 3^{rank(\kappa_0)}$ and similarly $rank(\kappa_2) < rank(\kappa_1)$ for each $\kappa_2 \varepsilon \Phi(v)$ hence $rank(\Phi(v)) < 3^{rank(\kappa_1)}$ by Proposition 4.2.3, its corollary and the fact that: $3^n > 2(3^{n-1} + \ldots + 3^0)$. Thus $rank(\Phi(w)) < 3^{rank(\kappa_0)}$ as required. •

In an initial equation of the form $f\,x = e_r$ we know that x (and no other variable) can occur free in e_r. Hence we will introduce no ambiguity if we write $e_r(e)$ for $e_r[x \leftarrow e]$.

Lemma 4.2.7 *Suppose that* $(f\,x = e_r)\pi$ *is a transformation consisting solely of instantiations*[12]. *Each equation in* π *has the form* $f\,p(z) = e_r(p(z))$ *for some pattern* $p(z)$.

Proof. By induction on the structure of π. *Base Case*: π is the equation $f\,x = e_r$ and evidently $e_r = e_r[x \leftarrow x]$ as required. *Induction Case*: π has the form $\pi_0(\beta_i \rightarrow_1 \beta_{i0}, \beta_{i1})$. *Ex hypothesi* β_i has the form $f\,p(z, n) = e_r(p(z, n))$. By the definition of the prime transformation of instantiation β_{i0} and β_{i1} have the forms $f\,p(z, \mathsf{Leaf}) = e_r(p(z, \mathsf{Leaf}))$ and $f\,p(z, \mathsf{Node}\ t_0\ t_1) = e_r(p(z, \mathsf{Node}\ t_0\ t_1))$as required. •

Corollary 4.2.8 *Suppose that* $(f\,x = e_r)\pi$ *is an i-transformation. Each sequent of* [π] *which lies in the range of* ι *has the form*: $z \in \boldsymbol{B}, y \in B, \Psi(u) \vdash \varphi[e_r(p(z, y))]$.

Proof. This follows immediately from Lemma 4.2.7, Lemma 4.1.4 and Fact 4.1.2. •

Lemma 4.2.9 *Consider a transformation* $(f\,x = e_r)\pi$ *and an arbitrary sequent of* [π] *in the range of* ι (*which by Fact 4.1.2, must have the form* $z \in \boldsymbol{B}, y \in B, \Psi(u) \vdash \varphi[e(z, y)]$).
Let $\psi(u)$ *be a member of* $\Psi(u)$ *and s be the conclusion sequent of the instance of B-elim in* [π] *at which* $\psi(u)$ *is discharged*:

(*i*) $\psi(u)$ *has the form* $(\forall v \in \boldsymbol{B})(\Xi(v) \supset \varphi[e_r(p(v, u))])$.

(*ii*) *s has the form* $n \in B \vdash (\forall v \in \boldsymbol{B})(\Xi(v) \supset \varphi[e_r(p(v, n))])$.

Proof. Inspection of Definition 4.2.1 is sufficient to demonstrate that these formulae are *only* introduced via an instance of *B-elim*. The variable u must be one of the two eigen variables of the instance of *B-elim* at which $\psi(u)$ is discharged. Let $\iota^{-1}(s) = \beta$. Suppose, by Corollary 4.2.8, that the

---

12 We will refer to such transformations as *i*-transformations in the sequel.

sequent $\iota(\beta)$ has the form: $v \in B, n \in B, \Xi(v) \vdash \varphi[e_r(p(v, n))]$. Thus s has the form: $n \in B \vdash (\forall v \in B)(\Xi(v) \supset \varphi[e_r(p(v, n))])$ as required for (*ii*). Both assumptions introduced by this rule instance have the form: $(\forall v \in B)(\Xi(v) \supset \varphi[e_r(p(v, n))])$ with $n = u$ in one case, verifying (*i*). •

For convenience it is useful at this point to set some notation for the next few results. In what follows $(f\,x = e_r)\pi$ is an *i*-transformation. By Corollary 4.2.8 each open sequent of $[\,\pi\,]$ has the form $z \in B, y \in B, \Psi(u) \vdash \varphi[e_r(p(z, y))]$. For each such open sequent let $t(z, y, w)$ be a term such that: $z \in B, y \in B, w : \Psi(u) \vdash t(z, y, w) : \varphi[e_r(p(z, y))]$ in $\mathcal{EOB}^{\mathcal{IA}}$. This uniquely determines an $\mathcal{EOB}^{\mathcal{IA}}$ derivation, $\mu_{[\pi]}$ with conclusion sequent: $x \in B \vdash f_0\,x : \varphi[e_r]$ for some term f_0. •

Definition 4.2.10 Given a context $z \in B, u : \Psi(z)$ occurring in $\mu_{[\pi_0]}$ we define the sequence $\mathfrak{I}_{\Psi(z)}$ to be $\mathfrak{I}_{\psi_1(z_1)} \cdots \mathfrak{I}_{\psi_n(z_n)}$ and each $\mathfrak{I}_{\psi_i(z_i)}$ to be $h_i\ z_i$ where h_i is chosen such that the conclusion sequent of that instance of *B-elim* in $[\,\pi\,]$ at which $\psi_i(z_i)$ is discharged corresponds in the $\mathcal{EOB}^{\mathcal{IA}}$ derivation $\mu_{[\pi]}$ to the sequent with assigned term $h_i\ n$ form some n. Furthermore, we define the substitution $\theta_{(u:\Psi(z))} =_{\text{def}} [u \leftarrow \mathfrak{I}_{\Psi(z)}]$. •

Fact 4.2.11 $u\theta_{(u:\Psi(z))} = \mathfrak{I}_{\Psi(z)}$. •

Proposition 4.2.12 *f_0 converts to the system of equations corresponding to the open sequents of $[\,\pi\,]$ with typical component $f_0\,p(z, y) = t(z, y, \mathfrak{I}_{\Psi(u)})$ corresponding to typical open sequent: $z \in B, y \in B, \Psi(u) \vdash \varphi[e_r(p(z, y))]$.*

Proof. By induction on the structure of the *i*-transformation $(f\,x = e_r)\pi$.

Base Case: The transformation is the eureka equation $f\,x = e_r$. The derivation $[\,\pi\,]$ is, simply the sequent $x \in B \vdash \varphi[e_r]$. Let t be a term such that $x \in B \vdash t\,x : \varphi[e_r]$ in $\mathcal{EOB}^{\mathcal{IA}}$. Since this single open sequent is itself the conclusion sequent in $[\,\pi\,]$ the uniquely determined term f_0 is t. This, being the single open sequent in $[\,\pi\,]$, establishes the single equation $f_0\,x = t\,x$ (there being no substitutions as the sequence $\Psi(u)$ is, in this case, empty) and the result follows immediately.

Induction Case: π has the form: $\pi_0(\beta_i \rightarrow_I \beta_{i0}, \beta_{i1})$. By Lemma 4.2.7, β_i has the form $f\,p(z, y) = e_r(p(z, y))$ for some pattern $p(z, y)$ and, by Definition 3.4.1 and Lemma 4.2.7, if the variable y is instantiated in the final step the equations β_{i0} and β_{i1} will have the form: $f\ p(z, \mathsf{Leaf}) = e_r(p(z, \mathsf{Leaf}))$ and $f\,p(z, \mathsf{Node}\ s_0\ s_1) = e_r(p(z, \mathsf{Node}\ s_0\ s_1))$. The open sequent $\iota(\beta_i)$ in $[\,\pi_0\,]$ has, by Corollary 4.2.8, the form: $z \in B, y \in B, \Psi(z, y) \vdash \varphi[e_r(p(z, y))]$ and so, by Definition 4.2.1, the open sequents $\iota(\beta_{i0})$ and $\iota(\beta_{i1})$ will have the form $z \in B,\ \Psi(z) \vdash \varphi[e_r(p(z, \mathsf{Leaf}))]$ and $z, s_0, s_1 \in B, \kappa(s_0), \kappa(s_1), \Psi(z) \vdash \varphi[e_r(p(z, \mathsf{Node}\ s_0\ s_1))]$ where $\kappa(u)$ is the formula: $(\forall z \in B)(\Psi(z) \supset \varphi[e_r(p(z, u))])$. Let us suppose that the terms assigned to the open sequents of $[\,\pi\,]$ include, in particular, terms $t_{\mathsf{L}}(z, w)$and $t_{\mathsf{N}}(s_0, s_1, z, v_0, v_1, w)$ such that:
$z \in B, w : \Psi(z) \vdash t_{\mathsf{L}}(z, w) : \varphi[e_r(p(z, \mathsf{Leaf}))]$ and:
$z, s_0, s_1 \in B, v_0 : \kappa(s_0), v_1 : \kappa(s_1), w : \Psi(z) \vdash t_{\mathsf{N}}(s_0, s_1, z, v_0, v_1, w) : \varphi[e_r(p(z, \mathsf{Node}\ s_0\ s_1))]$.
So in $\mu_{[\,\pi\,]}$ the term, f_0, assigned to the root sequent is uniquely determined and the term assigned to the sequent $\iota(\beta_i)$ is calculated by contemplating the appropriate derivation fragment given in Definition 4.2.1:

$$\dfrac{\dfrac{z, \overline{w} : \Psi(z) \vdash t_{\mathsf{L}} : \varphi[e_r(p(z, \mathsf{L}))]}{\vdash \lambda z \overline{w}.t_{\mathsf{L}} : \kappa(\mathsf{Leaf})} \quad \dfrac{z, s_0, s_1, \overline{w} : \Psi(z), v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash t_{\mathsf{N}} : \varphi[e_r(p(z, \mathsf{N}\ s_0\ s_1))]}{s_0, s_1 \in B, v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash \lambda z \overline{w}.t_{\mathsf{N}} : \kappa(\mathsf{Node}\ s_0\ s_1)}}{\dfrac{y \in B \vdash \textit{trec}\ (\lambda z \overline{w} t_{\mathsf{L}})\ (\lambda s_0 s_1 v_0 v_1 z \overline{w}.t_{\mathsf{N}})\ y : (\forall z)(\Psi(z) \supset \varphi[e_r(p(z, y))]}{\dfrac{\ldots\ldots \qquad z, y \in B, w : \Psi(u) \vdash \textit{trec}\ (\lambda z \overline{w}.t_{\mathsf{L}})\ (\lambda s_0\ s_1 v_0 v_1 z \overline{w}.t_{\mathsf{N}})\ y\ z\ \overline{w} : \varphi[e_r(p(z, y))] \qquad \ldots\ldots}{\mu_{[\pi_0]}}}}$$

where $\overline{w}$ is the sequence of variables w with w, if $w : \psi(y)$ occurs in $\Psi(u)$, removed.

Ex hypothesi we may assume that f_0 converts to that system of equations corresponding to the open sequents of $[\ \pi_0\ ]$ and were we have, in particular, that: $f_0\ p(z, y)$ converts to $\textit{trec}\ (\lambda z \overline{w}.t_{\mathsf{L}})\ (\lambda s_0\ s_1 v_0 v_1 z \overline{w}.t_{\mathsf{N}})\ z\ \mathfrak{I}_{\Psi(z)}$. But this may be specialised by further conversions, writing h for the term $\textit{trec}\ (\lambda z \overline{w}.t_{\mathsf{L}})\ (\lambda s_0\ s_1 v_0 v_1 z \overline{w}.t_{\mathsf{N}})$, as follows: $f_0\ p(z, \mathsf{Leaf}) = h\ \mathsf{Leaf}\ z\ \mathfrak{I}_{\Psi(z)} = (\lambda z \overline{w}.t_{\mathsf{L}})\ z_0\ \mathfrak{I}_{\Psi(z)} = t_{\mathsf{L}}(z, \mathfrak{I}_{\Psi(z)})$ and $f_1\ p(z, \mathsf{Node}\ s_0\ s_1) = (h\ (\mathsf{Node}\ s_0\ s_1)\ z\ \mathfrak{I}_{\Psi(z)}) = (\lambda s_0\ s_1 v_0 v_1 z \overline{w}.t_{\mathsf{N}})\ s_0\ s_1\ (h\ s_0)\ (h\ s_1)\ z\ \mathfrak{I}_{\Psi(z)} = t_{\mathsf{N}}(s_0, s_1, z, (h\ s_0), (h\ s_1), \mathfrak{I}_{\Psi(z)}) = t_{\mathsf{N}}(s_0, s_1, z, \mathfrak{I}_{(\Psi(z),\ \kappa(s0),\ \kappa(s1))})$ and these are precisely the two new equations required to correspond to the two new open sequents which result in extending the derivation from $[\ \pi_0\ ]$ to $[\ \pi\ ]$. •

Our next result, links the intermediate, *local, recursions* (the operations like those of the form h in the proof above) to the final, *global, recursion* f_0.

Corollary 4.2.13 *If* $y \in B \vdash h\ y : (\forall z)(\Psi(z) \supset \varphi[e_r(p(z, y))]$ *is the conclusion sequent of an instance of B-elim in* $\mu_{[\ \pi\ ]}$ *then* $f_0\ p(z, y) = h\ y\ z\ \mathfrak{I}_{\Psi(z)}$.

Proof. There are two cases to consider. When z is empty the conclusion sequent in question is: $x \in B \vdash f_0\ x : \varphi[e_r(x)]$ and the result is immediate. When z is non-empty consider the derivation fragment in $\mu_{[\pi]}$ corresponding to the instance of *B-elim* in question:

$$\dfrac{\begin{array}{c}\vdots\\ \vdots\end{array} \qquad\qquad \begin{array}{c}\vdots\\ \vdots\end{array}}{\dfrac{y \in B \vdash h\ y : (\forall z)(\Psi(z) \supset \varphi[e_r(p(z, y))]}{\dfrac{\ldots\ldots \qquad z, y \in B, w : \Psi(z, y) \vdash h\ y\ z\ \overline{w} : \varphi[e_r(p(z, y))] \qquad \ldots\ldots}{\mu_{[\pi_0]}}}}$$

Consider the derivation $\mu_{[\pi_0]}$ with its open sequent $z, y \in B, w : \Psi(z, y) \vdash h\ y\ z\ \overline{w} : \varphi[e_r(p(z, y))]$. By Proposition 4.2.12 we have the equation $f_0\ p(z, y) = h\ y\ z\ \overline{w}\theta_{(w:\Psi(z,y))}$ and this is $h\ y\ z\ \overline{w}\theta_{(\overline{w}:\Psi(z))}$ since y does not occur free in $\overline{w}$. Hence, by Fact 4.2.11, $f_0\ p(z, y) = h\ y\ z\ \mathfrak{I}_{\Psi(z)}$ as required. •

Lemma 4.2.14 *Suppose that* $(f\ x = e_r)\pi$ *is an i-transformation. Consider any instantiation in* π. *If the conclusion sequent of the instance of B-elim in* $\mu_{[\ \pi\ ]}$ *which corresponds to this instantiation is* $y \in B \vdash h\ y : (\forall z)(\Psi(z) \supset \varphi[e_r(p(z, y))]$ *then*:

$h\ \mathsf{Leaf} = \lambda z \overline{w}.t_{\mathsf{L}}(z, \overline{w})$

$h\ (\mathsf{Node}\ s_0\ s_1) = \lambda z \overline{w}.t_{\mathsf{N}}(z, s_0, s_1, \overline{w}, \mathfrak{I}_{\kappa(s_0)}, \mathfrak{I}_{\kappa(s_1)})$

for terms t_{L} and t_{N} such that:

$f_0\, p(z, \mathsf{L}) = t_{\mathsf{L}}(z, \mathfrak{I}_{\Psi(z)})$

$f_0\, p(z, \mathsf{N}\, s_0\, s_1) = t_{\mathsf{N}}(z, s_0, s_1, \mathfrak{I}_{\Psi(z)}, \mathfrak{I}_{\kappa(s_0)}, \mathfrak{I}_{\kappa(s_1)})$

Proof. Consider the following fragment corresponding to an instantiation in $\mu_{[\,\pi\,]}$.

$$\dfrac{\dfrac{z, \overline{w} : \Psi(z) \vdash t_{\mathsf{L}} : \varphi[e_r(p(z, \mathsf{L}))]}{\vdash \lambda z\overline{w}.t_{\mathsf{L}} : \kappa(\mathsf{Leaf})} \qquad \dfrac{z, s_0, s_1, \overline{w} : \Psi(z), v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash t_{\mathsf{N}} : \varphi[e_r(p(z, \mathsf{N}\, s_0\, s_1))]}{s_0, s_1 \in B, v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash \lambda z\overline{w}.t_{\mathsf{N}} : \kappa(\mathsf{Node}\, s_0\, s_1)}}{\dfrac{\dfrac{y \in B \vdash h\, y : (\forall z)(\Psi(z) \supset \varphi[e_r(p(z, y))])}{z, y \in B, w : \Psi(u) \vdash h\, y\, z\, \overline{w} : \varphi[e_r(p(z, y))]} \quad \ldots\ldots}{\mu_{[\pi_0]}}}$$

The equations then follow by inspection and the definition of ***trec***. That the terms t_{L} and t_{N} satisfy their equations follows by Proposition 4.2.12. •

Proposition 4.2.15 *Let* π *be an arbitrary transformation. If the following derivation fragment*[13] *occurs in a subderivation of* $\mu_{[\pi]}$ *which corresponds to a serious fold in* π *then* $(r\, z\xi\, f)\theta_{(r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}))} = f_0\, p(z\xi, y)$.

$$\dfrac{\begin{array}{c}\vdots\\ \boldsymbol{r} : \Delta(\boldsymbol{y}) \vdash f : \Psi(z\xi)\end{array} \qquad \dfrac{}{r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash r : (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_r(p(z, y))])}}{r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash\ r : \Psi(z\xi) \supset \varphi[e_r(p(z\xi, y))]}$$

$$r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash\ r\, z\xi\, f : \varphi[e_r(p(z\xi, y))]$$

Proof. We proceed by induction on the structure of such derivations.
Case z is empty. The the fragment has the simplified form:

$$\dfrac{}{r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash r : \varphi[e_r(y)]}$$

and $r\theta_{(r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}))} = \mathfrak{I}_{\varphi[e_r(y)]} = f_0\, y$ as required.

Induction Case: The fragment has the form:

$$\dfrac{\begin{array}{c}\vdots\\ \boldsymbol{r} : \Delta(\boldsymbol{y}) \vdash f : \Psi(z\xi)\end{array} \qquad \dfrac{}{r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash r : (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_r(p(z, y))])}}{r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash\ r : \Psi(z\xi) \supset \varphi[e_r(p(z\xi, y))]}$$

$$r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}) \vdash\ r\, z\xi\, f : \varphi[e_r(p(z\xi, y))]$$

We note that $(r\, z\xi\, f)\theta_{(r, \boldsymbol{r} : \Delta(y, \boldsymbol{y}))} = (\mathfrak{I}_{\delta(y)}\, z\xi\, f)\theta_{(\boldsymbol{r} : \Delta(\boldsymbol{y}))}$ and $\mathfrak{I}_{\delta(y)}$ is $h\, y$ for some suitable h (see Definition 4.2.10).Thus we have to show that $h\, y\, z\xi\, f\theta_{(\boldsymbol{r} : \Delta(\boldsymbol{y}))} = f_0\, p(z\xi, y)$. Now by Corollary

13 We have omitted the variable assignments for clarity.

4.2.13 we know that $h\ y\ z\ \Im_{\Psi(z)} = f_0\ p(z, y)$ so $h\ y\ z\xi\ \Im_{\Psi(z\xi)} = f_0\ p(z\xi, y)$ since this is just a substitution instance. It remains to show that $f\theta_{(r\,:\,\Delta(y))} = \Im_{\Psi(z\xi)}$. This follows by induction on the length of f if we can show that $f\theta_{(r\,:\,\Delta(y))} = \Im_{\psi(z\xi)}$ holds for an arbitrary f in f and the corresponding $\psi(z\xi)$ in $\Psi(z\xi)$. Note that $\Im_{\psi(z\xi)} = h_1\ z\xi$ for some suitable h_1, so it remains to show that $f\theta_{(r\,:\,\Delta(y))} = h_1\ z\xi$. There are two cases to consider.

Case ξ is trivial on z. The derivation has the form:

$$\frac{}{r:\ \Delta(y) \vdash r:\ (\forall v \in B)(\Phi(v) \supset \varphi[e_r(p(v, z))])}$$

and $r\theta_{(r:\Delta(z))} = r\theta_{(r:\delta(z))} = \Im_{\delta(z)} = h_1\ z$, as required.

Case otherwise. The derivation has the form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(r, u : \Delta(y), \Phi(v)) \vdash g : \Xi(w\zeta)} \quad \dfrac{\dfrac{}{r, u : \Delta(y), \Phi(v) \vdash u : (\forall w \in B)(\Xi(w) \supset \varphi[e_r(q(w, y))])}}{r, u : \Delta(y), \Phi(v) \vdash u\ w\zeta : \Xi(w\zeta) \supset \varphi[e_r(p(v, z\xi))]}}{r, u : \Delta(y), \Phi(v) \vdash u\ w\zeta\ g : \varphi[e_r(p(v, z\xi))]}}{r : \Delta(y) \vdash \lambda u.u\ w\zeta\ g : \Phi(v) \supset \varphi[e_r(p(v, z\xi))]}}{r:\ \Delta(y) \vdash \lambda vu.u\ w\zeta\ g:\ (\forall v \in B)(\Phi(v) \supset \varphi[e_r(p(v, z\xi))])}$$

Ex hypothesi we may conclude that $(u\ w\zeta\ g)\theta_{(\Delta(y),\ \Phi(v))} = f_0\ p(v, z\xi)$ but then consider the fragment:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{v, u : \Phi(v) \vdash t_{\mathsf{L}} : \varphi[e_r(p(v, \mathsf{L}))]}}{\vdash \lambda vu.t_{\mathsf{L}} : \kappa(\mathsf{Leaf})} \quad \dfrac{\dfrac{\dfrac{}{v, s_0, s_1, u : \Phi(v), v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash t_{\mathsf{N}} : \varphi[e_r(p(v, \mathsf{N}\ s_0\ s_1))]}}{s_0, s_1 \in B, v_0 : \kappa(s_0), v_1 : \kappa(s_1) \vdash \lambda vu.t_{\mathsf{N}} : \kappa(\mathsf{Node}\ s_0\ s_1)}}{y \in B \vdash h_1\ z : (\forall v)(\Phi(v) \supset \varphi[e_r(p(v, z))])}}{v, y \in B, u : \Phi(v) \vdash h_1\ z\ v\ u : \varphi[e_r(p(v, z))]} \quad \ldots\ldots}{\mu_{[\pi_0]}}$$

where $\kappa(x) = (\forall v)(\Phi(v) \supset \varphi[e_r(p(v, z))])$. We proceed by cases on the substitution ξ.

Case $\xi = [z \leftarrow \mathsf{Leaf}]$: By Proposition 4.2.12 we know that $f_0\ p(v, z\xi) = t_{\mathsf{L}}(v, \Im_{\Phi(v)})$ and thus $(u\ w\zeta\ g)\theta_{(\Delta(y),\ \Phi(v))} = t_{\mathsf{L}}(v, \Im_{\Phi(v)})$. Hence $\lambda vu.t_{\mathsf{L}} = (\lambda vu.u\ w\zeta\ g)\theta_{\Delta(y)}$ and the result follows by Lemma 4.2.14.

Case $\xi = [z \leftarrow \mathsf{Node}\ p_0\ p_1]$: Again, by Proposition 4.2.12 we know that $f_0\ p(v, z\xi) = t_{\mathsf{N}}(p_0, p_1, v, \Im_{(\Psi(z),\ \kappa(s0),\ \kappa(s1))})$ and thus $(u\ w\zeta\ g)\theta_{(\Delta(y),\ \Phi(v))} = t_{\mathsf{N}}(p_0, p_1, v, \Im_{(\Psi(z),\ \kappa(s0),\ \kappa(s1))})$. Hence $\lambda vu.t_{\mathsf{N}} = (\lambda vu.u\ w\zeta\ g)\theta_{\Delta(y)}$ and the result follows by Lemma 4.2.14. •

Definition 4.2.16 Let $\delta\ \varepsilon\ \mathit{Der}(\mathcal{EOB})$ where every open sequent has the form $\Gamma \vdash \varphi[e]$. We construct the derivation $\mathit{Close}(\delta)\ \varepsilon\ \mathit{Der}(\mathcal{EOB})$ from δ by treating each of its open sequents as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdots}}{\Gamma \vdash e \in B}\,(mem\text{-}\downarrow)}{\Gamma \vdash e\!\downarrow}\,(refl)}{\Gamma \vdash e = e}\,(\exists\text{-}intro)}{\ldots\ldots \quad \Gamma \vdash \varphi[e] \quad \ldots\ldots}$$
$$\delta$$

The ellipsis can be filled in by induction over the structure of e via:

$$\frac{}{\Gamma \vdash x \in B}\,(ass) \qquad \frac{}{\Gamma \vdash \mathsf{Node} \in B \to B \to B} \qquad \frac{}{\Gamma \vdash \mathsf{Leaf} \in B}$$

$$\frac{\Gamma \vdash e_0 \in T_0 \to T_1 \quad \Gamma \vdash e_0 \in T_0}{\Gamma \vdash e_0\, e_1 \in T_1} \qquad \frac{}{\Gamma \vdash f \in T}\,(con)$$

Regarding (*con*): In $\mathcal{EOB}$ + $\mathcal{R}$an operation name f is a constant of fixed type. •

Fact 4.2.17 *Let* $\delta \in \mathbf{Der}(\mathcal{EOB})$. *If* $\Gamma \vdash \varphi[e]$ *is among the open sequents of* δ *then the corresponding sequent of* $\mu_{Close(\delta)} \in \mathbf{Der}(\mathcal{EOB}^{\mathcal{TA}})$ *is* $\Upsilon_\Gamma \vdash e : \varphi[e]$. *Moreover, if* $u : \psi(z)$ *occurs in* Υ_Γ *then* u *is not free in* e. •

Theorem 4.2.18 *For every* $\pi(\mathrm{B}) \in \mathcal{PT}_B(\omega)$, *if* $t : Close([\,\pi\,])$ *then* $\vdash t = \mathrm{B}$.

Proof. We proceed by induction on the length of the transformation π. We will, in each case, utilise the same notation for the data as is used in the corresponding case of Definition 4.2.1.

Base Case: $[\,\pi\,]$ is the single open sequent $x \in B \vdash \varphi[e_r]$. The root sequent of the derivation $\mu_{Close([\,\pi\,])}$ is, by Fact 4.2.17, $x \in B \vdash e_r : \varphi[e_r]$ and this evidently yields the term $\lambda x.e_r$ which is equal to $f\,x = e_r$ as required.

For the remaining cases we may assume that the transformation has the form: $\pi = (\beta)\pi_0(\beta_i \to_{\mathrm{X}} \mathrm{B})$ for some X and equation β_i. We can do some preliminary work for several of these cases. *Ex hypothesi*, we may assume that if $t : Close([\,\pi_0(\mathrm{B}_0)\,])$ then $\vdash t = \mathrm{B}_0$. Pictorially we have, in $\mathcal{EOB}$, the following situation:

$$\left.\begin{array}{c}\dfrac{}{\vdots}\\ \dfrac{}{\ldots\ldots \quad \Gamma \vdash \varphi[e] \quad \ldots\ldots}\end{array}\right\}\, Close$$
$$[\,\pi_0\,]$$

and, corresponding to this, in $\mathcal{EOB}^{\mathcal{TA}}$:

$$\dfrac{\dfrac{\dfrac{}{\vdots}}{\Gamma \vdash e : \varphi[e]}\ \ldots\ldots}{B_0 : \mu_{[\,\pi_0\,]}}$$

Without loss of generality we may write t as $t_0[x \leftarrow e]$. Now each of the following cases will extend the sequent $\Gamma \vdash \varphi[e]$ and then close the derivation so formed. As a result the term assigned to this sequent in $\mathcal{EOB}^{\mathcal{TA}}$ will be some e_0 and thus we will have:
$t_0[x \leftarrow e_0] : \mathit{Close}([\pi_0(\beta_i \twoheadrightarrow_X B)])$. If we suppose that $\Gamma \vdash e = e_0$ then, since intensional equality is a congruence, $\Gamma \vdash t_0[x \leftarrow e] = t_0[x \leftarrow e_0]$ which is just $\Gamma \vdash B_0 = t_0[x \leftarrow e_0]$. We require: $\Gamma \vdash B = t_0[x \leftarrow e_0]$ and this will follow from $\Gamma \vdash B_0 = B$. But these ensembles are identical up to the one equation β_i of B_0. If the transformation out of β_i is an unfold, a benign fold, a law application or an instantiation then we are done (by Lemma 3.4.2). So in each of those cases it simply remains to demonstrate that $\Gamma \vdash e = e_0$.

Ad (*unf*): We must consider the derivation $\mathit{Close}([\,\pi\,])$ whose operative component is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall(g\ \boldsymbol{q} = d)}}{\Gamma \vdash g\ \boldsymbol{q}\theta = d\theta} \qquad \dfrac{\dfrac{}{\vdots}}{\Gamma \vdash \varphi[e_1][e_0 \Leftarrow d\theta]}}{\ldots\ldots \quad \Gamma \vdash \varphi[e_1] \quad \ldots\ldots}}{[\,\pi_0\,]}$$

In $\mathcal{EOB}^{\mathcal{TA}}$ we will have:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Upsilon_\Gamma \vdash \forall(g\ \boldsymbol{q} = d)}}{\Upsilon_\Gamma \vdash g\ \boldsymbol{q}\theta = d\theta} \qquad \dfrac{\dfrac{}{\vdots}}{\Upsilon_\Gamma \vdash e_1[e_0 \Leftarrow d\theta] : \varphi[e][e_0 \Leftarrow d\theta]}}{\ldots\ldots \quad \Upsilon_\Gamma \vdash e_1[e_0 \Leftarrow d\theta] : \varphi[e] \quad \ldots\ldots}}{\mu_{[\,\pi_0\,]}}$$

and we have by the previous analysis to show that $e_1 = e_1[e_0 \Leftarrow d\theta]$ but that is immediate from the data.

Ad (*law*): Similarly.

Ad (*benign fld*): Similarly.

Ad (*ins*): We must consider the derivation $\mathit{Close}([\,\pi\,])$; pictorially:

$$
\dfrac{
\dfrac{\dfrac{\dfrac{\overline{\quad\quad}\\ \vdots}{z \in \boldsymbol{B}, \Psi(\boldsymbol{u})^{-} \vdash e(\mathsf{L}) = e(\mathsf{L})}}{z \in \boldsymbol{B}, \Psi(\boldsymbol{u})^{-} \vdash \varphi[e(\mathsf{Leaf})]}}{\vdash \kappa(\mathsf{Leaf})}
\qquad
\dfrac{\dfrac{\dfrac{\overline{\quad\quad}\\ \vdots}{z \in \boldsymbol{B}, t_1 \in B, t_2 \in B, \Psi(\boldsymbol{u})^{-}, \kappa(t_1), \kappa(t_2) \vdash e(\mathsf{N}\ t_1\, t_2) = e(\mathsf{N}\ t_1\, t_2)}}{z \in \boldsymbol{B}, t_1 \in B, t_2 \in B, \Psi(\boldsymbol{u})^{-}, \kappa(t_1), \kappa(t_2) \vdash \varphi[e(\mathsf{Node}\ t_1\, t_2)]}}{t_1 \in B, t_2 \in B, \kappa(t_1), \kappa(t_2) \vdash \kappa(\mathsf{Node}\ t_1\, t_2)}
}{
\dfrac{x \in B \vdash (\forall z \in \boldsymbol{B})(\Psi(\boldsymbol{u})^{-} \supset \varphi[e(x)])}{\cdots\cdots \quad z \in \boldsymbol{B}, x \in B, \Psi(\boldsymbol{u}) \vdash \varphi[e(x)] \quad \cdots\cdots}
}
$$

$$[\,\pi_0\,]$$

In $\mathcal{EOB}^{\mathcal{TA}}$ we have (omitting some variable assignments in contexts for layout reasons):

$$
\dfrac{
\dfrac{\dfrac{\overline{\quad\quad}\\ \vdots}{\boldsymbol{v} : \Psi(z) \vdash e(z, \mathsf{L}) : \varphi[e(z, \mathsf{L})]}}{\vdash \lambda z\boldsymbol{v}.e(z, \mathsf{L}) : \kappa(\mathsf{Leaf})}
\qquad
\dfrac{\dfrac{\overline{\quad\quad}\\ \vdots}{\boldsymbol{v} : \Psi(z), w_1 : \kappa(t_1), w_2 : \kappa(t_2) \vdash e(z, \mathsf{N}\ t_1\, t_2) : \varphi[e(z, \mathsf{N}\ t_1\, t_2)]}}{w_1 : \kappa(t_1), w_2 : \kappa(t_2) \vdash \lambda z\boldsymbol{v}.e(z, \mathsf{N}\ t_1\, t_2) : \kappa(\mathsf{Node}\ t_1\, t_2)}
}{
\dfrac{x \in B \vdash \boldsymbol{trec}\ (\lambda z\boldsymbol{v}.e(z, \mathsf{L}))\ (\lambda t_1 t_2 w_1 w_2 z\boldsymbol{v}.e(z, \mathsf{N}\ t_1\, t_2).)\ x : (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e(z, x)])}{\cdots\cdots \quad z, x, \boldsymbol{v} : \Psi(\boldsymbol{u}) \vdash \boldsymbol{trec}\ (\lambda z\boldsymbol{v}.e(z, \mathsf{L}))\ (\lambda t_1 t_2 w_1 w_2 z\boldsymbol{v}.e(z, \mathsf{N}\ t_1\, t_2))\ x\ z\ \boldsymbol{v} : \varphi[e(z, x)] \quad \cdots\cdots}
}
$$

$$[\,\pi_0\,]$$

Our prior analysis requires us to demonstrate that:
$\boldsymbol{trec}\ (\lambda z\boldsymbol{v}.e(z, \mathsf{Leaf}))\ (\lambda t_1 t_2 w_1 w_2 z\boldsymbol{v}.e(z, \mathsf{Node}\ t_1\, t_2))\ x\ z\ \boldsymbol{v} = e(x)$. Let us write f for the expression $\boldsymbol{trec}\ (\lambda z\boldsymbol{v}.e(z, \mathsf{Leaf}))\ (\lambda t_1 t_2 w_1 w_2 z\boldsymbol{v}.e(z, \mathsf{Node}\ t_1\, t_2))$. Proceeding by cases, and noting that, by Fact 4.2.17, the variables $\boldsymbol{v}$, w_1 and w_2 do not occur free in either $e(z, \mathsf{Leaf})$ or $e(z, \mathsf{Node}\ t_1\, t_2)$:
$f\,\mathsf{Leaf}\ z\ \boldsymbol{v} = (\lambda z\boldsymbol{v}.e(z, \mathsf{Leaf}))\ z\ \boldsymbol{v} = e(z, \mathsf{Leaf})$ and
$f\,(\mathsf{Node}\ s_1\, s_2)\ z\ \boldsymbol{v} = (\lambda t_1 t_2 w_1 w_2 z\boldsymbol{v}.e(z, \mathsf{Node}\ t_1\, t_2))\, s_1\, s_2\, (f\, s_1)\ (f\, s_2)\ z\ \boldsymbol{v} = e(z, \mathsf{Node}\ s_1\, s_2)$ as required.

Ad (*serious fld*): We must consider the derivation $\mathcal{C}lose$([π]) whose operative component is:

$$
\dfrac{
\dfrac{\dfrac{\vdots}{\Gamma^{-} \vdash \Psi(z\xi_0)} \qquad \dfrac{\dfrac{\overline{\quad\quad}}{\Gamma \vdash (\forall z \in \boldsymbol{B})(\Psi(z) \supset \varphi[e_r(p(z, y))])}}{\Gamma \vdash \Psi(z\xi_0) \supset \varphi[e_r(p(z\xi_0, y))]}}{\Gamma \vdash \varphi[e_r(p(z\xi_0, y))]}
\qquad
\dfrac{\dfrac{\overline{\quad\quad}\\ \vdots}{\Gamma \vdash \varphi[e_1[e_0 \Leftarrow w]]} \quad \dfrac{}{\Gamma, w = e_0 \vdash w = e_0}}{\Gamma, w = e_0 \vdash \varphi[e_1]}
}{
\cdots\cdots \quad \Gamma \vdash \varphi[e_1] \quad \cdots\cdots
}
$$

$$[\,\pi_0\,]$$

In $\mathcal{EOB}^{\mathcal{TA}}$ we have (omitting the equality premise for layout reasons):

$$\dfrac{\dfrac{\vdots}{\Upsilon_{\Gamma-} \vdash f : \Psi(z\xi_0)} \quad \dfrac{\dfrac{}{\Upsilon_\Gamma \vdash v : (\forall z \in B)(\Psi(z) \supset \varphi[e_r(p(z,y))])}}{\Upsilon_\Gamma \vdash v\, z\xi_0 : \Psi(z\xi_0) \supset \varphi[e_r(p(z\xi_0,y))]}}{\dfrac{\Upsilon_\Gamma \vdash v\, z\xi_0 f : \varphi[e_r(p(z\xi_0,y))]}{\dfrac{\cdots\cdots \quad \Upsilon_\Gamma \vdash e_1[e_0 \Leftarrow v\, z\xi_0 f] : \varphi[e_1] \quad \cdots\cdots}{\mu_{[\pi_0]}}}} \qquad \dfrac{\dfrac{}{\vdots}}{\dfrac{\Upsilon_\Gamma \vdash e_1[e_0 \Leftarrow w] : \varphi[e_1[e_0 \Leftarrow w]]}{\Upsilon_\Gamma, w = e_0 \vdash e_1[e_0 \Leftarrow w] : \varphi[e_1]}}$$

The equation β in π such that $\iota(\beta) = \Gamma \vdash \varphi[e_1]$ is $f\,q = e$ with $e \leq_\zeta e_1$ for some ζ by Lemma 4.1.4. By the definition of serious folding, Definition 3.4.1(*iv*), the equation β_0 such that $\iota(\beta) = \Gamma \vdash \varphi[e_1[e_0 \Leftarrow w]]$ is $f\,q = e[e_0 \Leftarrow f\,x\theta]$. By Proposition 4.2.15 we may conclude that $v\,z\xi_0\,f = f_0\,p(z\xi_0, y)$. By Definition 4.2.1 we know that ξ_1 is $[x \leftarrow p(z, y)]$ thus θ, which is $\xi_0 * \xi_1$, is $[x \leftarrow p(z\xi_0, y)]$. Thus $v\,z\xi_0\,f = f_0\,x\theta$ and thus the equation for f_0 corresponding to the sequent: $\Gamma \vdash \varphi[e_1]$ (Proposition 4.2.12) is $f_0\,q = e[e_0 \Leftarrow f_0\,x\theta]$. Since all other equations for f in the transformation and all other equations for f_0 given by Proposition 4.2.12 remain the same we can conclude that $f_0 = \mathrm{B}$ as required. •

Theorem 4.2.19 *(Soundness)*[14]
Let $\pi(\mathrm{B})$ *be a transformation from the equation* $f\,x = e_r$. *If* $[\,\pi\,] \in$ ***Der***$(\mathcal{EOB})$ *then* π *is correct.*

Proof. Assume that $[\,\pi\,] \in$ ***Der***$(\mathcal{EOB})$. The conclusion of $[\,\pi\,]$ is the sequent $x \in B \vdash \varphi[e_r]$. We can verify immediately (in fact it is the base case of Theorem 4.2.18) that $(f\,x = e_r) : x \in B \vdash \varphi[e_r]$. By Theorem 4.2.18 we know that $\mathrm{B} : x \in B \vdash \varphi[e_r]$. But, by Fact 4.2.2, $f = \mathrm{B}$. Hence is π correct. •

§4.3 Completeness

There remains the question of completeness. Is it the case that every correct transformation is translated into a derivation in ***Der***$(\mathcal{EOB})$ by our translation? In fact it *is* possible to construct (rather pathological) counterexamples to completeness but this is perhaps not too surprising. After all, systems like the calculus of transformations, which are, in general, capable of unsound arguments, are very underconstrained and thus, conversely, it is not unlikely that they permit some valid arguments for what are, essentially, *accidental reasons*. There is, however, an interesting and fruitful investigation of this topic which begins with certain observations which already exist in the literature. There we see hints and suggestions for the modification of the straightforward transformational tradition, motivated by the well-known problems of unrestricted folding, which restrict the use of folding to circumstances in which well-orderings play an explicit rôle. The work of Bird (for example [Bir84]) is particularly noteworthy and our own remarks in [Hen87] are in this spirit too. The advice one obtains from these sources is that the calculus of transformations can be restricted, without prejudicing its expressibility, so that folding is *only* allowed on immediate predecessors in some well-ordering. So far as we can tell from the available literature this advice has not been formalised to a point where a proof that this restriction ensures that transformations are correct in general could be provided. In [Hen93] we proved completeness for such a calculus of transformations over the natural numbers by examining a hierarchy of transformations classified

14 We take *validity* of a transformation to be given by the (extensional) notion of *correctness*. This explains our use of the terminology.

according to a complexity measure which was based upon the number of instantiations they involve. In the context of our current work we would be inclined to begin a similar investigation with the following definition.

Definition 4.3.1

(*i*) $\mathcal{PT}_B(n) = \{\pi \mid \pi \textit{ contains not more than } n \textit{ occurrences of } (ins)\}$

(*ii*) $\mathcal{PT}_B(\omega) = \bigcup_{n \in \mathbf{N}} \mathcal{PT}_B(n)$. •

We can then, at least, establish the following quite easily.

Theorem 4.3.2 *(Mini Completeness)*
Let $\pi \in \mathcal{PT}_B(1)$ *and subject to the extra requirement that folds may take place only on immediate predecessors. If* π *is correct then* $[\,\pi\,] \in \textbf{\textit{Der}}(\mathcal{EOB})$.

Proof. Suppose for a contradiction that $[\,\pi\,] = \textbf{\textit{fail}}$. Inspection of Definition 4.2.1 shows that this is the result of an un-translatable fold step. From this it follows that the fold is not undertaken on an immediate predecessor since an assumption for those values are present in the context. But this is in contradiction with the assumption. •

We should like to continue with this analysis which would require a thorough investigation of the operation schemata which are utilised by the transformations of greater complexity (similar to our work in [Hen93]) but, at this time, all the issues are not clear to us. The main problem would be to determine precisely the well-orderings which are appropriate for each complexity class. In [Hen93] we showed that the ordering ω^n was appropriate for the transformations in $\mathcal{PT}_N(n)$ but the extra complexity in the inductive assumptions we require in the translation of instantiations given in this paper makes the task of generalising the approach of [Hen93] somewhat daunting and we must, therefore, leave this for future research.

§5 An illustrative example

We shall finally illustrate the technical development with one, rather typical, example. This example, which involves a double instantiation, does show quite graphically the hidden complexity underlying what appears superficially to be a reasonably simple transformation. It illustrates, in particular, the iterative process which is required in Definition 4.2.1 for the interpretation of serious folding. The transformation yields a linear operation from a (worst case) quadratic one. The function we deal with takes an arbitrary element of B^{15} and yields another which has the same fringe of leaves but is left-linear. That is, it satisfies the specification:

$$(\forall s_0 \in B)(\exists s_1 \in B)(\textit{left-linear}(s_1) \wedge \textit{eq-fringe}(s_0, s_1))$$

with the predicates given by:

$$\frac{}{\textit{left-linear}(\mathsf{Leaf}\ n)} \qquad \frac{\textit{left-linear}(s)}{\textit{left-linear}(\mathsf{Node}\ (\mathsf{Leaf}\ n)\ s)} \qquad \frac{\textit{eq-fringe}(s_0, s_1)}{\textit{fringe}(s_0) = \textit{fringe}(s_1)}$$

where:

---

15 As we mentioned in §3.1 we utilise the more realistic definition of B which allows numerals to be carried in the Leaf case.

$$
\begin{array}{lcl}
\mathit{fringe}\ s & = & \mathit{frg}\ s\ \mathsf{Nil} \\
\mathit{frg}\ (\mathsf{Leaf}\ n)\ l & = & \mathsf{Cons}\ n\ l \\
\mathit{frg}\ (\mathsf{Node}\ s_0\ s_1)\ l & = & \mathit{frg}\ s_0\ (\mathit{frg}\ s_1\ l)
\end{array}
$$

We begin with the initial operation:

$$
\begin{array}{lcl}
\mathit{rotate}\ (\mathsf{Leaf}\ n) & = & \mathsf{Leaf}\ n \\
\mathit{rotate}\ (\mathsf{Node}\ s_0\ s_1) & = & \mathit{join}\ (\mathit{rotate}\ s_0)\ (\mathit{rotate}\ s_1)
\end{array}
$$

where:

$$
\begin{array}{lcl}
\mathit{join}\ (\mathsf{Leaf}\ n)\ s & = & \mathsf{Node}\ (\mathsf{Leaf}\ n)\ s \\
\mathit{join}\ (\mathsf{Node}\ s_0\ s_1)\ s_2 & = & \mathsf{Node}\ s_0\ (\mathit{join}\ s_1\ s_2)
\end{array}
$$

which may be derived in the term assignment system $\mathcal{TK}^{\mathcal{IA}}$ by induction over B given the function *join* and the properties that *join* preserves linearity and the fringes of its arguments.

We adopt the very simple eureka definition: *rote* s = *rotate s* and the transformation then proceeds:

$$
\begin{array}{lcll}
\mathit{rotate}\ (\mathsf{Leaf}\ n) & = & \mathit{rote}\ (\mathsf{Leaf}\ n) & \textbf{Instantiate} \\
 & = & \mathsf{Leaf}\ n. & \textbf{Unfold} \\
\mathit{rote}\ (\mathsf{Node}\ (\mathsf{Leaf}\ n)\ s) & = & \mathit{rotate}\ (\mathsf{Node}\ (\mathsf{Leaf}\ n)\ s) & \textbf{Instantiate} \\
 & = & \mathit{join}\ (\mathit{rotate}\ (\mathsf{Leaf}\ n))\ (\mathit{rotate}\ s_1) & \textbf{Unfold} \\
 & = & \mathit{join}\ (\mathsf{Leaf}\ n)\ (\mathit{rotate}\ s_1) & \textbf{Unfold} \\
 & = & \mathsf{Node}\ (\mathsf{Leaf}\ n)\ (\mathit{rotate}\ s_1) & \textbf{Unfold} \\
 & = & \mathsf{Node}\ (\mathsf{Leaf}\ n)\ (\mathit{rote}\ s_1). & \textbf{SFold} \\
\mathit{rote}\ (\mathsf{Node}\ (\mathsf{Node}\ s_0\ s_1)\ s_2) & = & \mathit{rotate}\ (\mathsf{Node}\ (\mathsf{Node}\ s_0\ s_1)\ s_2) & \textbf{Instantiate} \\
 & = & \mathit{join}\ (\mathit{rotate}\ (\mathsf{Node}\ s_0\ s_1))\ (\mathit{rotate}\ s_1) & \textbf{Unfold} \\
 & = & \mathit{join}\ (\mathit{join}\ (\mathit{rotate}\ s_0)\ (\mathit{rotate}\ s_1))\ (\mathit{rotate}\ s_1) & \textbf{Unfold} \\
 & = & \mathit{join}\ (\mathit{rotate}\ s_0)\ (\mathit{join}\ (\mathit{rotate}\ s_1)\ (\mathit{rotate}\ s_1)) & \textbf{Law} \\
 & = & \mathit{join}\ (\mathit{rotate}\ s_0)\ (\mathit{rotate}\ (\mathsf{Node}\ s_1\ s_2)) & \textbf{BFold} \\
 & = & \mathit{rotate}\ (\mathsf{Node}\ s_0\ (\mathsf{Node}\ s_1\ s_2)) & \textbf{BFold} \\
 & = & \mathit{rote}\ (\mathsf{Node}\ s_0\ (\mathsf{Node}\ s_1\ s_2)). & \textbf{SFold}
\end{array}
$$

yielding the final operation given by the system of equations:

$$
\begin{array}{lcl}
\mathit{rotate}\ s & = & \mathit{rote}\ s \\
\mathit{rote}\ (\mathsf{Leaf}\ n) & = & \mathsf{Leaf}\ n \\
\mathit{rote}\ (\mathsf{Node}\ (\mathsf{Leaf}\ n)\ s) & = & \mathsf{Node}\ (\mathsf{Leaf}\ n)\ (\mathit{rote}\ s) \\
\mathit{rote}\ (\mathsf{Node}\ (\mathsf{Node}\ s_0\ s_1)\ s_2) & = & \mathit{rote}\ (\mathsf{Node}\ s_0\ (\mathsf{Node}\ s_1\ s_2))
\end{array}
$$

If we call this transformation π then [π] is not ***fail*** and, indeed, [π] is reproduced as Figure 1[16].

§6 General remarks and future research

We have been able to provide an interpretation for a class of program transformations as derivations in a simple theory of operations and types which is faithful to the intension of the transformations. Our objective was to extend the work described in [Hen93] to cover arbitrary algebraic data types. In fact, as we remarked in §2, we have dealt with a typical example of such a type and we have restricted ourselves to transformations from unary operations. The reason for our

---

16 Figure 1 is presented in natural deduction form rather than the less compact natural deduction in sequent form which we have found most convenient to present the material in the main body of the paper.

approach here is simply one regarding clarity of presentation: the derivations and the analysis is, even so, rather combinatorial and, as we sketch below, would be more so if we had attempted to present the arbitrary case with arbitrary arity operations. Our justification for claiming to have dealt with the general case here concerns the issues we have covered explicitly. There are significant differences between the analysis here and the analysis of [Hen93]. This is because the data type N is rather atypical as an abstract data type. Particularly relevant to our application is the fact that *re-instantiations* of arguments of type N are trivial and reduce to a case analysis. This is *not* true of B and not true in general for the algebraic types. Indeed, the example we displayed in §5 is a good example of a non-trivial re-instantiation. In this respect, then, B is a good representative of the algebraic types and the redevelopment of the analysis in the more general setting simply complicates the presentation without introducing any new conceptual difficulties or interest. Similarly we covered quite thoroughly in [Hen93] the complications which accrue from allowing transformations from arbitrary arity operations and we would have gained no new insight from allowing them here. The above notwithstanding, we shall sketch some of the details which establish the full generalisation in §6.1 in order to at least demonstrate what would be at stake. Then in §6.2 we briefly mention some possible avenues for future investigation.

§6.1 The general case

We need to work in an underlying theory of operations and types which is rich enough to deal with the full range of types. The algebraic types are those constructed by the disjoint union of cartesian products of type variables, constants and positive recursion. We give the general case and some examples utilising a notation similar to Miranda [TuD85]:

$$\begin{array}{lll} B & ::= & \mathsf{Leaf}\ N \mid \mathsf{Node}\ B\ B \\ L & ::= & \mathsf{Nil} \mid \mathsf{Cons}\ N\ L \\ N & ::= & \mathsf{Zero} \mid \mathsf{Succ}\ N \\ T & ::= & \mathsf{DC}_0\ \boldsymbol{T}_0 \mid \ldots \mid \mathsf{DC}_n\ \boldsymbol{T}_n \end{array}$$

where each $\boldsymbol{T}_i = T_{i0} \ldots T_{im_i}$ and where T and each T_{ij} are type variables.

Our new theory should include rules which capture the least fixpoints of positive type operations[17].

$$\frac{\Gamma \vdash z \in B(\Xi(\lambda X.B))}{\Gamma \vdash z \in \Xi(\lambda X.B)}\ (\Xi\text{–}intro) \qquad \frac{\Gamma \vdash B(T) \subseteq T}{\Gamma \vdash \Xi(\lambda X.B) \subseteq T}\ (\Xi\text{-}elim)$$

Intuitively, $\Xi(\lambda X.B)$ is the smallest type closed under the operation B. It is also convenient to add rules for comprehension types:

$$\frac{\Gamma \vdash z \in \{x \mid \varphi(x \leftarrow z)\}}{\Gamma \vdash \varphi(x \leftarrow z)} \qquad \frac{\Gamma \vdash \varphi(x \leftarrow z)}{\Gamma \vdash z \in \{x \mid \varphi(x \leftarrow z)\}}$$

Small adjustments are also required to the rules governing definedness of the underlying partial logic but these are not central in this context and we omit the details. The new theory is, in fact, the theory $\mathcal{TK}$of, for example, [Hen89*a*].

---

17 These significantly extend the algebraic types but are syntactically much easier to manage.

The algebraic types now are special cases formed by careful choice of the operation B. Taking $B(X) =_{\text{def}} \{\mathsf{DC}_0\} \times \Pi(T_0[T \leftarrow X]) + \ldots + \{\mathsf{DC}_n\} \times \Pi(T_n[T \leftarrow X])$[18] we obtain the expected rules for T as special cases of Ξ–*intro* and Ξ-*elim* including, in particular:

$$\frac{x_0 \in T_0, \Psi_0(\mathbf{y}_0) \vdash \psi(\mathsf{DC}_0\, x_0) \ \ldots \ x_n \in T_n, \Psi_n(\mathbf{y}_n) \vdash \psi(\mathsf{DC}_n\, x_n)}{x \in T \vdash \psi(x)} \qquad (T\text{-}elim)$$

where each $\boldsymbol{x}_i = x_{i0} \ \ldots \ x_{im_i}$ and where $\Psi_i(\mathbf{y}_i) = \psi(y_{i0}), \ \ldots, \ \psi(y_{ik_i})$ with the y_{ij} distinct variables among the $\boldsymbol{x}_i$ such that, if $y_{ij} = x_{pq}$ then $T_{pq} = T$ and if $T_{pq} = T$ then $y_{ij} = x_{pq}$ for some y_{ij}.

In the term assignment system $\mathcal{TK}^{\mathcal{IA}}$ we have, among others, the rule:

$$\frac{\Upsilon \vdash f : B(T) \subseteq T}{\Upsilon \vdash \boldsymbol{irec}\, f : \Xi(\lambda X.B) \subseteq T}$$

where ***irec*** satisfies the equation: $\boldsymbol{irec}\, f\, x = f\, x\, (B\ (\boldsymbol{irec}\, f)\, x)$. In other respects the theory $\mathcal{TK}^{\mathcal{IA}}$ follows the pattern of $\mathcal{EOB}$.

With this much in place we can sketch part of the translation $[\ \_\ ] \in \mathcal{PT} \rightarrow \boldsymbol{Der}(\mathcal{TK}) + \{\boldsymbol{fail}\}$. In the main this is not different in any regard to §4. We give two cases for illustration. Let $\boldsymbol{z}$ be the variables occurring in the patterns $\boldsymbol{p}$, p and $\boldsymbol{q}$ other than x. Let $\boldsymbol{u}$ be a sequence of variables occurring in the sequence $\boldsymbol{z}x$

$$\left[\ \begin{array}{c} \ldots \quad \dfrac{(f\,\boldsymbol{p}, p(x), \boldsymbol{q} = e)[x \leftarrow (\mathsf{DC}_i\, \boldsymbol{x}_i)]}{} \quad \ldots \\ \ldots\ \ldots \quad \dfrac{f\,\boldsymbol{p}, p(x \in T), \boldsymbol{q} = e}{} \quad \ldots\ \ldots \\ \pi_0 \end{array}\ \right]$$

$=_{\text{def}}$

$$\frac{\begin{array}{l} \ldots \quad \dfrac{\boldsymbol{x}_i \in \boldsymbol{T}_i, z \in \boldsymbol{T}, \Theta_0(\boldsymbol{u}_0), \Psi_i(\mathbf{y}_i) \vdash \varphi[e[x \leftarrow \mathsf{DC}_i\, \boldsymbol{x}_i]])}{} \quad \ldots \\ \ldots \quad \boldsymbol{x}_i \in \boldsymbol{T}_i, \Psi_i(\mathbf{y}_i) \vdash (\forall z \in \boldsymbol{T})(\Theta_0(\boldsymbol{u}_0) \supset \varphi[e[x \leftarrow \mathsf{DC}_i\, \boldsymbol{x}_i]]) \quad \ldots \end{array}}{\dfrac{x \in T \vdash (\forall z \in \boldsymbol{T})(\Theta_0(\boldsymbol{u}_0) \supset \varphi[e])}{\ldots\ldots \quad \dfrac{z \in \boldsymbol{T}, x \in T, \Theta(\boldsymbol{u}) \vdash \varphi[e]}{} \quad \ldots\ldots}}$$

$$[\ \pi_0\ ]$$

where $\Theta_0(\boldsymbol{u}_0)$ is the sequence $\Theta(\boldsymbol{u})$ with a formula of the form $\zeta(x)$ *should it occur* removed. As in the development of §4 the formulae $\Theta(\boldsymbol{u})$ must satisfy (a suitable generalisation of) our predicate $\mathcal{F}$. Things become more complicated when we examine in detail the case of serious folding. The definition in this case has the form:

---

18 Π denotes iterated cartesian product.

$$\left[\ \frac{e_0 \subseteq e \qquad f\,\boldsymbol{x} = e_r \in \mathcal{L} \qquad e_0 \leq_\theta e_r \qquad f\,\boldsymbol{p} = e[e_0 \Leftarrow (f\,\boldsymbol{x})\theta]}{\ldots\ldots f\,\boldsymbol{p} = e \ldots\ldots}\ \right]$$

$$\pi_0$$

$=_{\text{def}}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma^- \vdash \Theta(\boldsymbol{u})\xi_0} \qquad \dfrac{}{\Gamma \vdash \Theta(\boldsymbol{u})\xi_0 \supset \varphi[e_0]}}{\Gamma \vdash \varphi[e_0]} \qquad \dfrac{\Gamma \vdash \varphi[\,e_0 \Leftarrow w] \qquad \Gamma, w = e_0 \vdash w = e_0}{\Gamma, w = e_0 \vdash \varphi[e]}}{\ldots\ldots \quad \Gamma \vdash \varphi[e] \quad \ldots\ldots}}{[\,\pi_0\,]}$$

where the assumption $(\forall z \in \boldsymbol{T})(\Theta(\boldsymbol{u}) \Rightarrow \varphi[e_1])$ is chosen such that $e_0 \leq_{\xi_0} e_1 \leq_{\xi_1} e_r$ and Γ^- is the context Γ with the assumption $(\forall z \in \boldsymbol{T})(\Theta(\boldsymbol{u}) \Rightarrow \varphi[e_1])$ removed. The derivation above is then completed, for each of the formulae comprising $\Theta(\boldsymbol{u})\xi_0$, as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(\Gamma^-, \boldsymbol{v} \in \boldsymbol{S}, \Phi(\boldsymbol{w}))^- \vdash \Phi_0(\boldsymbol{w}_0)\theta_0} \qquad \dfrac{}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{S}, \Phi(\boldsymbol{w}) \vdash \Phi_0(\boldsymbol{w}_0)\theta_0 \supset \varphi[e_3\theta_0]}}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{S}, \Phi(\boldsymbol{w}) \vdash \varphi[e_2]}}{\Gamma^-, \boldsymbol{v} \in \boldsymbol{S} \vdash \Phi(\boldsymbol{w}) \supset \varphi[e_2]}}{\Gamma^- \vdash (\forall \boldsymbol{v} \in \boldsymbol{S})(\Phi(\boldsymbol{w}) \supset \varphi[e_2])}$$

choosing the assumption $(\forall \boldsymbol{v}_0 \in \boldsymbol{S}_0)(\Phi_0(\boldsymbol{w}_0) \supset \varphi[e_3])$ so that $e_2 \leq_{\theta_0} e_3$. As expected the context $(\Gamma^-, \boldsymbol{v} \in \boldsymbol{S}, \Phi(\boldsymbol{w}))^-$ is the context $\Gamma^-, \boldsymbol{v} \in \boldsymbol{S}, \Phi(\boldsymbol{w})$ with the assumption: $(\forall \boldsymbol{v}_0 \in \boldsymbol{S}_0)(\Phi_0(\boldsymbol{w}_0) \supset \varphi[e_3])$ removed. As before the ellipsis is filled in by iterating the process as necessary. The termination of this is now more complex since, in general, the selection of a formula of rank k may result in contexts with an unbounded number of formulae of rank k-1. This is a consequence of the general definition of algebraic types which (in the definition of type T) might contain a summand of the form: DC $T_0 \ldots T_m$ with $m \in \omega$ of which any n ($n \in m$) of the T_i may be T. The rank of formulae satisfying the predicate $\mathcal{F}$ is given by: $rank((\forall z \in \boldsymbol{T})(\Theta(\boldsymbol{u}) \supset \varphi[e])) = 1 + \boldsymbol{max}\{rank(\zeta_i(u_i)) \mid i \in n\}$ where $\Theta(\boldsymbol{u})$ is a sequence of formulae $\zeta_0(u_0) \ldots \zeta_{n\text{-}1}(u_{n\text{-}1})$. We then extend this to contexts: $rank(z \in T, \Gamma) = rank(\Gamma)$, $rank(\kappa, \Gamma) = \omega^{rank(\kappa)} + rank(\Gamma)$ when $\mathcal{F}(\kappa)$ and we note that $lim\{rank(\Gamma) \mid \Gamma$ *is a standard context*$\} = \omega^\omega$ so the analogy to Proposition 4.2.6 follows by transfinite induction to ω^ω. From this point on all the analogous results (Lemma 4.2.7 through Theorem 4.2.19) go through, with differences only resulting from the considerably greater combinatorial complexity which results from the general definition of types.

§6.2 Future work

Possible extensions to the approach which would require substantial new analysis might include the generalisation to transformations which require *type simulations*. This approach has its intellectual roots in the work of Wand [Wan80] and was worked out in [Hen88]. The difficulty here resides in the need to generalise substantially the notion of canonical sequent. Similarly one would require a major generalisation of these sequents to handle a calculus of program development based upon pre/post condition specifications rather than simple equations. This, however, would be a particularly interesting line of work.

§7 Acknowledgements

I would like to thank the members of the *Constructive set theory in programming group* at the University of Essex, in particular Ray Turner and Mike Sanderson. I also benefited from the discussions with the audiences at seminars at the University of Kent and the *Applied Logic Colloquium*, University of London, where an early formulation of this material was first presented.

§8 References

[Bac89] Backhouse, R. C. *et al*, *Do-it-yourself type theory*, Formal Aspects of Computing, 1, pp 19-84, 1989.

[Bee85] Beeson, M., *Foundations of Constructive Mathematics*, Springer Verlag, 1985.

[Bir84] Bird, R., *The promotion and accumulation strategies in transformational programming*, ACM. Trans. Programming Languages and Systems, 6, pp 487-504, 1984.

[BuD77] Burstall, R. & Darlington, J., *A transformation system for developing recursive programs*, J. ACM, 24, pp 44-67, 1977.

[Con86] Constable, R. *et al*, *Implementing mathematics with the NuPRL proof development system*, Prentice-Hall, 1986.

[HaN87] Hayashi, S. & Nakano, H., *The PX system – A computational logic*, Publications of the Research Institute for Mathematical Sciences, Kyoto University, Tokyo, 1987.

[Har56] Harrop, R., *On disjunctions and existential statements in intuitionistic systems of logic*, Math. Ann. 132, pp 347-361, 1956.

[Hen87] Henson, M. C., *Elements of functional languages*, Blackwell Scientific Publications, 1987.

[Hen88] Henson, M. C., *Higher order accumulation and type simulations*, Computer Journal, 31(6), pp517-524, 1988.

[Hen89*a*] Henson, M. C., *Program development in the programming logic TK*, Formal Aspects of Computing, 1, pp173-192, 1989.

[Hen89*b*] Henson, M. C., *Realizability models for program construction*, Proc. Conf. on Mathematics of Program Construction, Gröningen, LNCS 375, pp 256-272, Springer, 1989.

[Hen90] Henson, M. C., *Information loss in the programming logic TK*, Programming concepts and methods, (ed: Broy, M. and Jones, C.), pp 509-545, Elsevier, 1990.

[Hen91] Henson, M. C., *Bridging the formality gap: proofgramming*, Proc. 12th Computer Conf., pp 145-156, Dunedin, New Zealand, August, 1991.

[Hen93] Henson, M. C., *An intensional semantics for elementary program transformation*, submitted, J. Logic and Computation, 1993.

[Kot78] Kott, L., *About transformations system: a theoretical study*, in: Program transformations, (ed: Robinet, B.), pp 232-247, Dunod, 1978.

[Kot85] Kott, L., *Unfold/fold program transformations*, in: Algebraic methods in semantics, (eds: Nivat. M. & Reynolds, J. C.), pp 411-434, Cambridge University Press, 1985.

[San93] Sanderson, M. T., *Private communication*, 1993.

[Tho91] Thompson, S., *Type theory and functional programming*, Addison Wesley, 1991.

[TuD79] Turner, D. A., *A new implementation technique for applicative languages*, Software Practice and Experience, 9, pp 31-49, 1979.

[TuD85] Turner, D. A., *Miranda - A non-strict functional language with polymorphic types*, in: Proc. IFIP Int. Conf. on functional programming languages and computer architecture, Nancy, LNCS 201, Springer Verlag, pp 445-472, 1985.

[TuR91] Turner, R., *Constructive foundations for functional programming*, McGraw-Hill, 1991.

[Wan80] Wand, M., *Continuation based transformation strategies*, J. ACM, 27(1), pp 164-180, 1980.

Figure 1

$z_0 \in N$

Leaf $z_0 \in B$

Leaf $z_0\downarrow$ *close*

Leaf z_0 = Leaf z_0

$z_0 \in N$

Leaf $z_0 \in B$

$\exists x \in B.x$ = Leaf z_0 *unf*

$z_0 \in N$

$\exists x \in B.x = \textbf{\textit{Rotate}}$(Leaf z_0) *ins*

$\exists x \in B.x = \textbf{\textit{Rotate}}\ z_1$

$z_0 \in N$

Leaf $z_0 \in B$

$w \in B$

Node(Leaf z_0) $w \in B$

Node(Leaf z_0) $w\downarrow$ *close*

Node(Leaf z_0) w = Node(Leaf z_0) w

$z_0 \in N$

Leaf $z_0 \in B$

$w \in B$

Node(Leaf z_0) $w \in B$

$\exists x \in B.x$ = Node(Leaf z_0) w *sfld*

$w = \textbf{\textit{Rotate}}\ z_1$

$\exists x \in B.x$ = Node(Leaf z_0)($\textbf{\textit{Rotate}}\ z_1$)

$\exists x \in B.x$ = Node(Leaf z_0)($\textbf{\textit{Rotate}}\ z_1$) *unf*

$z_0 \in N$

$z_1 \in B$

$\textbf{\textit{Rotate}}\ z_1 \in B$

$\exists x \in B.x = \textbf{\textit{Join}}$(Leaf z_0)($\textbf{\textit{Rotate}}\ z_1$) *unf*

$z_0 \in N$

$\exists x \in B.x = \textbf{\textit{Join}}(\textbf{\textit{Rotate}}$(Leaf z_0))($\textbf{\textit{Rotate}}\ z_1$) *unf*

$z_0 \in N$

Leaf $z_0 \in B$

$z_1 \in B$

$\exists x \in B.x = \textbf{\textit{Rotate}}$(Node(Leaf z_0) z_1) *ins*

$(\exists x \in B.x = \textbf{\textit{Rotate}}\ z_1) \supset \exists x \in B.x = \textbf{\textit{Rotate}}$(Node(Leaf z_0) z_1)

$\forall x \in B.((\exists x \in B.x = \textbf{\textit{Rotate}}\ x) \supset \exists x \in B.x = \textbf{\textit{Rotate}}$(Node(Leaf z_0) x))

$\forall x \in B.((\exists x \in B.x = \textbf{Rotate}\ x) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_2\ x))$
$z_3 \in B$
$z_4 \in B$
$\text{Node}\ z_3\ z_4 \in B$
$(\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_3\ z_4)) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_2(\text{Node}\ z_3\ z_4))$
$\forall x \in B.((\exists x \in B.x = \textbf{Rotate}\ x) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_3\ x))$
$z_4 \in B$
$(\exists x \in B.x = \textbf{Rotate}\ z_4) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_3\ z_4)$
$\exists x \in B.x = \textbf{Rotate}\ z_4$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_3\ z_4)$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_2(\text{Node}\ z_3\ z_4))$

$w \in B$
$w\downarrow$ *close*
$w = w$
$w \in B$
$\exists x \in B.x = w$ *sfld*
$w = \textbf{Rotate}(\text{Node}\ z_2(\text{Node}\ z_3\ z_4))$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_2(\text{Node}\ z_3\ z_4))$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_2(\text{Node}\ z_3\ z_4))$ *bfld*
$\exists x \in B.x = \textbf{Join}(\textbf{Rotate}\ z_2)(\textbf{Rotate}(\text{Node}\ z_3\ z_4))$ *bfld*
$\exists x \in B.x = \textbf{Join}(\textbf{Rotate}\ z_2)(\textbf{Join}(\textbf{Rotate}\ z_3)(\textbf{Rotate}\ z_4))$ *law*
$z_2 \in B$
$\textbf{Rotate}\ z_2 \in B$
$z_3 \in B$
$\textbf{Rotate}\ z_3 \in B$
$z_4 \in B$
$\textbf{Rotate}\ z_4 \in B$
$\exists x \in B.x = \textbf{Join}(\textbf{Join}(\textbf{Rotate}\ z_2)(\textbf{Rotate}\ z_3))(\textbf{Rotate}\ z_4)$ *unf*
$z_2 \in B$
$z_3 \in B$
$\exists x \in B.x = \textbf{Join}(\textbf{Rotate}(\text{Node}\ z_2\ z_3))(\textbf{Rotate}\ z_4)$ *unf*
$z_2 \in B$
$z_3 \in B$
$\text{Node}\ z_2\ z_3 \in B$
$z_4 \in B$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}(\text{Node}\ z_2\ z_3)\ z_4)$ *ins*
$(\exists x \in B.x = \textbf{Rotate}\ z_4) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}(\text{Node}\ z_2\ z_3)\ z_4)$
$\forall x_1 \in B.((\exists x \in B.x = \textbf{Rotate}\ x_1) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}(\text{Node}\ z_2\ z_3)\ x_1))$
$\forall x_0 \in B.\forall x_1 \in B.((\exists x \in B.x = \textbf{Rotate}\ x_1) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ x_0\ x_1))$ *ins*
$z_0 \in B$
$\forall x \in B.((\exists x \in B.x = \textbf{Rotate}\ x) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_0\ x))$
$z_1 \in B$
$(\exists x \in B.x = \textbf{Rotate}\ z_1) \supset \exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_0\ z_1)$
$\exists x \in B.x = \textbf{Rotate}\ z_1$
$\exists x \in B.x = \textbf{Rotate}(\text{Node}\ z_0\ z_1)$
$\forall x \in B.\exists y \in B.y = \textbf{Rotate}\ x$ *eur*