Parametrised bar recursion: A unifying framework for realizability interpretations of classical dependent choice

Thomas Powell

November 4, 2014

Abstract

During the last twenty years or so a wide range of realizability interpretations of classical analysis have been developed. In many cases, these are achieved by extending the base interpreting system of primitive recursive functionals with some form of bar recursion, which realizes the negative translation of either countable or countable dependent choice. In this work we present the many variants of bar recursion used in this context as instantiations of a general, parametrised recursor, and give a uniform proof that under certain conditions this recursor realizes a corresponding family of parametrised dependent choice principles. From this proof, the soundness of most of the existing bar recursive realizability interpretations of choice, including those based on the Berardi-Bezem-Coquand functional, modified realizability and the more recent products of selection functions of Escardó and Oliva, follows as a simple corollary. We achieve not only a uniform framework in which familiar realizability interpretations of choice can be compared, but show that these represent just simple instances of a large family of potential interpretations of dependent choice principles.

1 Introduction

One of the central problems of mathematical logic and computer science is to understand the constructive meaning of non-constructive proofs. From the early 20th century onwards a rich variety of constructive interpretations of classical logic have been developed, together with techniques for extracting computational content from proofs. The majority of these techniques initially deal with proofs in formal systems of classical predicate logic or weak subsystems of mathematics such as Peano arithmetic. In order to interpret stronger non-constructive proofs from subsystems of mathematical analysis which contain principles such as the axiom of dependent choice,

\[ \forall n, x^X \exists y^X A_n(x, y) \rightarrow \exists f^{N \rightarrow X} \forall n A_n(f(n), f(n+1)), \]

these techniques must typically be adapted and extended - a process which tends to be highly non-trivial.

In this paper we focus on just one method of giving a computational interpretation to proofs: namely classical modified realizability. A long established way of extending this technique to deal with choice principles is to introduce some kind of bar recursion to the usual interpreting system of primitive recursive functionals. Bar recursive interpretations of choice principles are among the most widely known and studied methods for giving a computational interpretation to classical analysis, and numerous instances of this method can be found in the literature. However, both the particular variant of bar recursion used and the form of choice that it realizes differs from case to case, and it is not well-understood how the these variants compare as realizers. This is compounded by the fact that each variant was typically devised in a distinct setting, and both their existence and correctness often established using a slightly different method of proof, obscuring the inherent similarities between them all.

The main contribution of this paper is to construct a general bar recursive term which contains several free parameters. We show that whenever these parameters obey certain conditions, the term can be used to realize the negative translation of a corresponding variant of countable dependent choice, and that moreover essentially all of the known variants of bar recursion used to extend modified realizability to classical analysis appear as simple instantiations of the parameters.

Our motivation for this is twofold. Firstly, we obtain a clear, unifying framework in which the differences between existing variants of bar recursion and their correctness proofs are essentially reduced...
to simple structures on the natural numbers, and thus their behaviour as realizers is more easily compared. Secondly, in doing this we greatly generalise the interpretation of classical analysis by giving not one but a whole class of realizers of choice principles, which can be freely chosen between to suit the problem at hand. Having a flexible choice in this way means that in practise we are able extract computational content which is potentially more efficient and semantically meaningful.

1.1 Realizability interpretations of classical analysis: A brief overview

This work is designed to be self-contained as possible, and in particular no prior knowledge of bar recursion is assumed. However, because we are partly motivated by the desire to unify existing interpretations, we provide here a very short summary of some of the best known variants of bar recursion that we aim to bring together. We simply state without explanation their defining equations: the purpose here is not to treat in any detail the exact meaning of these objects, but to allow the reader to see certain key features of the recursion that will be dealt with in a more general setting later.

The idea of interpreting countable choice with bar recursion was first established not for realizability interpretations, but for Gödel’s more subtle Dialectica interpretation, in a fundamental paper of Spector in 1962 [27]. Here, bar recursion was given as the schema

$$\text{SBR}(Y, G, H, s^*) = \begin{cases} G(s) & \text{if } Y(\hat{s}) < |s| \\ H(s, \lambda x. \text{SBR}(Y, G, H, s \ast x)) & \text{otherwise.} \end{cases}$$

(1)

The key details here are that $s$ is a finite sequence, $s \ast x$ denotes the extension of $s$ with the object $x$, $\hat{s}$ its canonical extension as an infinite sequence, and the output type of $Y$ is a natural number. The parameter $Y$ is responsible for terminating the recursion, based on the assumption that the underlying tree barred by sequences satisfying $Y(\hat{s}) < |s|$ is well-founded - a fact that is true in all continuous models and also in non-continuous structures such as the strongly majorizable functionals [9].

An early adaptation of this idea to realizability was given by Berardi et al. [3], in which an interpretation for dependent choice broadly similar but somewhat simpler than Spector’s variant of bar recursion was established. This realizer was later reformulated in the more standard framework of modified realizability in [6], where it was given its now familiar name modified bar recursion and is defined as

$$\text{MBR}(Y, H, s) =_N Y(s \oplus \lambda n. H(s, \lambda x. \text{MBR}(Y, H, s \ast x))).$$

(2)

Here $\oplus$ is the overwrite operation and the outcome type of $Y$ is a natural number. This restriction on the type of $Y$ ensures that the tree $Y(s \oplus \ldots) = Y(s \oplus 0)$ is well-founded in continuous models because $Y$ only looks at a finite amount of information from its input sequence. Thus, like Spector’s variant of bar recursion [1], modified bar recursion is carried out over some underlying well-founded tree, although unlike Spector’s bar recursion, this tree is not computable [4].

In addition to what later became modified bar recursion, [3] contains a striking realizer for countable choice, which took from Spector only the basic idea of backward recursion, replacing the sequential bar recursive calls of (1) and (2) with a symmetric updating of finite partial functions. This realizer was again put into a standard realizability framework in [4] [5], and will be referred to here as the Berardi-Bezem-Coquand, or BBC-functional:

$$\text{BBC}(Y, H, u) =_N Y(u \oplus \lambda n. H(u, \lambda x. \text{BBC}(u^n)).$$

(3)

Here, $u$ is a finite partial function and $u^n$ is the domain-theoretic extension of $t$ with value $x$ at input $n$. As with modified bar recursion, BBC terminates by continuity of $Y$, although proving this is somewhat technical and as shown in [5] is most elegantly done with Zorn’s lemma in the form of open induction. It has been argued that [3] provides a computational interpretation of countable choice that is superior to standard bar recursion in that it is ‘demand driven’, in the sense that to compute an approximation for the $n$th point in a choice sequence it is not automatically necessary to compute approximations for $1, \ldots, n-1$ first [3].

Much more recently, a family of new variants of bar recursion known as products of selection functions were developed and explored by Escardó and Oliva, beginning in [13]. One of these, the so-called implicit

\footnote{Indeed, a closely related fact proved in [17] is that (4) is not S1-S9 computable in the total continuous functionals and is therefore strictly stronger than (3).}
product of selection functions, was shown in [15] to realize not only countable dependent choice but more generally a dependent version of the so called J-shift arising from the Pierce translation. The product of selection functions is distinguished by the fact that it incorporates course-of-values recursion into its bar recursive calls, and can be formulated as

\[ \text{IPS}(Y, H, s) = \text{N} \left( s \oplus \lambda n. H(t_n, \lambda x . \text{IPS}(Y, H, t_n * x)) \right) \]  

(4)

where \( t_n \) is the sequence of length \( n \) primitive recursively defined by

\[ (t_n)_i = \begin{cases} s_i & \text{if } i < |s|, \\ H(t_i, \lambda x . \text{IPS}(Y, H, t_i * x)) & \text{for } i < n. \end{cases} \]

In addition to solving the modified realizability interpretation of choice, this form of bar recursion has deep links to game theory as a functional that computes optimal strategies in a class of higher-type, continuously well-founded sequential games [14]. This provides a highly illuminating bridge between the computational content of the axiom of choice and the world of game theory, and is exploited in e.g. [23] to give a game-theoretic constructive interpretation of Ramsey’s theorem.

Of course, this list is by no means exhaustive, and several further variants of (2)-(4) have been devised for realizing choice principles. For example, a realizer of the refined A-translation of a sequential variant of dependent choice is given in [26], and is used in [25] to extract a realizer for Higman’s lemma. Similarly, forms of bar recursion closely related to (2) have been used for realizing dependent choice in a range of settings, including Parigot’s λµ-calculus [10] and in the context of realizability toposes [28].

Nevertheless, all of these recursors share two important features in common:

1. they take as input some partial object \( t \), and make recursive calls over extensions of this object;
2. they terminate by continuity of \( Y \).

In fact, one could go as far as to say that (2)-(4) along with their many other variants are essentially the same realizer, with the single exception that recursive calls are made using slightly different patterns. In this paper we make this idea precise, and show that the basic recipe used to form the realizers mentioned above can be used to construct, in a completely uniform way, an infinite class of bar recursive functionals, each one of which can be used to interpret a particular set of variants of the axiom of dependent choice.

Note that we restrict our attention to realizability interpretations of analysis, and in particular bar recursors of the form (1) which arise from the Dialectica interpretation do not fit into our framework. Nevertheless, we believe that the basic ideas behind this work could be readily lifted to the Dialectica interpretation (see for example the author’s recent article with P. Oliva [22] on a symmetric form of (1) which updates partial functions similarly to the BBC-functional), and may also be relevant to more direct computational interpretations of analysis, such as the learning-based realizabilities of e.g. [1, 2].

1.2 Outline of the paper

After setting up our basic formal systems in Section 2 we define in Section 3 a general principle of backward induction which will be used throughout the paper in order to prove the correctness of our realizers. Backward induction replaces principles such as bar induction or dependent choice which are more typically used on the meta-level to prove correctness of realizers, and is preferred here due to its far greater flexibility. We also define an analogous schema of backward recursion which will be used to construct our parametrised realizers. This section is strongly influenced by Berger [5], and our formulation of backward induction is similar to but stronger than his update induction.

Sections 4 and 5 form the core of the paper, in which we define our parametrised realizer and use it to realize dependent choice. In Section 4 we restrict ourselves to the simpler double negation shift: While this section’s main result, Theorem 4.4, is eventually subsumed by the later Theorem 5.3 it is instructive to first focus on the double negation shift so that the main ideas of Section 5 can be appreciated. In the sequel we introduce a family of dependent choice principles parametrised by a well-founded ordering \( \langle \) on the natural numbers, and show that the negative translation of each of these principles can be realized by our parametrised bar recursor under certain conditions. Theorem 5.3 gives essentially all the results listed in Section 1.1 as a Corollary, and vastly generalises all of them.

We conclude on a more informal level, and in Section 6 discuss a potential semantic interpretation of our parametrised realizer, inspired by the game-theoretic semantic reading of the Berardi-Bezem-Coquand functional in [3].
2 Preliminaries

Throughout this paper we work in an entirely standard calculus $\mathcal{T}$ of recursive functionals in all finite types, which is essentially the extensional Peano arithmetic $\mathbb{E}$-$\mathbb{PA}$ as defined in [17, 29], but based on a slightly richer type system.

2.1 The finite types

For us, the finite types consist of base types $\text{nat}$ and $\text{bool}$ for natural numbers and booleans, and are build from the formation of function types $\rho \rightarrow \tau$, product types $\rho \times \tau$, and finite sequence types $\rho^*$. We sometimes use the abbreviation $\tau^\rho$ for $\rho \rightarrow \tau$.

A discrete type is any type $\tau$ which can be encoded in $\text{nat}$, and so in all standard models $\text{bool}$, $\text{nat}$, $\text{bool} \times \text{nat}$, $\text{nat}^*$ etc. are discrete types but $\text{nat} \rightarrow \text{nat}$ is not.

2.2 The theory $\mathcal{T}$

Terms of $\mathcal{T}$ are typed lambda terms formed by application and abstraction, and include variables for each type, the usual constructors and deconstructors for product and sequence types, the arithmetic constants $0$: $\text{nat}$ and $s$: $\text{nat} \rightarrow \text{nat}$, and finally recursors $\mathbb{R}_\rho$: $\rho \rightarrow (\text{nat} \rightarrow \rho \rightarrow \rho) \rightarrow \text{nat} \rightarrow \rho$ of each type.

Equations in $\mathcal{T}$ are formed using basic symbols $=_{\text{bool}}$ and $=_{\text{nat}}$ for equality of type $\text{bool}$ and $\text{nat}$, while formulas of $\mathcal{T}$ are built from $=_{\text{nat}}$ together with the usual logical connectives and quantifiers for all types. Equality at arbitrary types is defined inductively, so for example $f =_{\rho \rightarrow \tau} g$ abbreviates $\forall x (f x =_\tau g x)$.

The axioms and rules of $\mathcal{T}$ are the standard axioms and rules of classical logic in all finite types, along with those of the typed lambda calculus, defining equations for all the constants, induction for arbitrary formulas, and finally full extensionality:

$$\forall f^\rho \rightarrow^\tau \forall x^\rho, y^\rho (x =_\rho y \rightarrow f x =_\tau f y).$$

In what follows we often consider extensions of $\mathcal{T}$ with new recursively defined functionals $\mathcal{F}$, in which case by $\mathcal{T} + \mathcal{F}$ we mean the extension of $\mathcal{T}$ with new constants $\mathcal{F}$ and their associated defining axioms.

We use, throughout, the following notation and abbreviations.

- $0_\rho$ is the inductively defined 0 term of type $\rho$.
- $\pi_i$ and $\langle \ , \ \rangle$ denote the projection and pairing operations. For a sequence $\alpha$: $(\rho_0 \times \rho_1)^{\text{nat}}$ we sometimes write $\alpha_i$: $\rho^{\text{nat}}$ for $\lambda n.\pi_i(\alpha(n))$.
- For a finite sequence $s$: $\rho^*$, $|s|$ denotes the length of $s$, while $s \ast t := \langle s_0, \ldots, s_{m-1}, t_0, \ldots, t_{n-1} \rangle$ denotes the concatenation $s$ and $f$. We also use $s \ast x$ to denote $s \ast \langle x \rangle$, and $s \ast \alpha$ for the concatenation of $s$ with the infinite sequence $\alpha$.
- $s \oplus \alpha$: $\rho^{\text{nat}}$ denotes the overwriting of $\alpha$ with the finite sequence $s$ i.e.

$$\{s \oplus \alpha\}(n) := \begin{cases} s_n & \text{if } n < |s| \\ \alpha(n) & \text{if } n \geq |s| \end{cases}.$$  

- $[\alpha](n) := \langle \alpha(0), \ldots, \alpha(n - 1) \rangle$ denotes the finite initial segment of length $n$ of $\alpha$: $\rho^{\text{nat}}$, while $\hat{s}$ denotes the canonical extension $s \oplus 0_{\rho^{\text{nat}}}$ of the finite sequence $s$: $\rho^*$.
- For a decidable predicate $P(x)$, the term `$y^\rho$ if $P(x)'$ of type $\rho$ is shorthand for

$$\begin{cases} y & \text{if } P(x) \\ 0_\rho & \text{otherwise} \end{cases}.$$  

- For a decidable predicate $P(n)$ on $\text{nat}$, the term $\mu i \leq n.P(i)$: $\text{nat}$ is the least $i \leq n$ satisfying $P(i)$, and just $n$ if no such $i$ exists.
For us, the theory \( T \) acts a standard lambda calculus equipped with a robust meta-theory for reasoning about terms. The exact details of \( T \) are not particularly important: while if the reader prefers they can simply assume we are working in a variant of \( \mathbb{EHA}^\omega \), we believe that everything which follows can be easily lifted to alternative settings. For example, a slightly different approach would be to work a weaker, quantifier-free term calculus \( T \) and to do all the reasoning in an unspecified meta-theory, as in [3]. Alternatively we could work in a theory of partial continuous functional as in [4], taking our base type to represent the flat domain \( \text{nat}_\bot \).

2.3 Models of \( T \)

In order to prove both the existence and correctness of our realizers, it is necessary to work in a constructive interpretation \( (T_\rho) \) of \( T \) which satisfies some variant of the following two properties:

1. \( (T_\rho) \) contains arbitrary choice sequences, in other words \( T_{\text{nat}\to\rho} \) contains all sequences \( \mathbb{N} \to T_\rho \) and so in particular \( (T_\rho) \) validates dependent choice;

2. Whenever \( \tau \) is a discrete type, functionals of type \( \rho^{\text{nat}} \to \tau \) satisfy the following continuity principle:

\[
\text{Cont} : \forall \alpha \in \rho^{\text{nat}} \exists n \forall \beta (\alpha(n) = \beta(n) \rightarrow F\alpha = F\beta).
\]

Both of these principles are satisfied automatically by the Kleene-Kreisel continuous functionals \( C^\omega [16, 18] \), whereas for term models such as the theory \( \mathcal{P} \) of [3], infinite choice sequences are added explicitly. Here we do not choose any particular interpretation of \( T \), rather we simply add principles such as \( \text{Cont} \) and dependent choice to our meta-theory whenever they are required.

3 Backward induction and recursion

We now develop some of the crucial background theory that will be required in order to prove our main results. In particular, we formalize a general principle of backward induction and define an associated backward recursor, both of which will be used to construct and verify the correctness of our parametrised realizer. In simple terms, backward induction is induction over domain-theoretic extensions of partial sequences. Special cases of backward induction include bar induction, and update induction as defined in [5]. In fact, this section is largely inspired by [5] in that we formulate backward induction as an instance of the more general principle of open induction.

3.1 Open induction

Open induction, first considered by Raoult in [24], is an extension of well-founded (or Noetherian) induction to chain-complete partial orders. Recall that a partial order \( (X, \leq) \) is chain-complete if every non-empty chain \( \gamma \) in \( X \) has a least upper bound \( \bigvee \gamma \). A predicate \( B \) on \( X \) is open if it satisfies

\[
B(\bigvee \gamma) \rightarrow \exists x \in \gamma B(x)
\]

for every non-empty chain in \( X \), and the principle of open induction over \( X \) is given by

\[
\text{OI}_{(X, \leq)} : \forall x (\forall y > x B(y) \rightarrow B(x)) \rightarrow \forall x B(x)
\]

where \( B \) ranges over open predicates. Note that open induction implies well-founded induction since whenever \( > \) is well-founded \( (X, \leq) \) is trivially chain-complete and all predicates are automatically open. However, in general \( > \) need not be well-founded, in which case openness becomes a non-trivial property.

**Theorem 3.1** (Raoult [24]). Any chain-complete partial order satisfies open induction.

**Proof.** This is a direct consequence of Zorn’s lemma. Suppose that the open predicate \( B \) satisfies the premise of open induction, which is classically equivalent to

\[
\forall x (\neg B(x) \rightarrow \exists y > x \neg B(y)), \quad (5)
\]
We define $u$ by assigning $\pi$ non-empty. We show that every chain in $S$ has an upper bound in $S$. For the empty chain this is trivial since $S$ contains at least one element. On the other hand, if $\gamma$ is non-empty, then it has an upper bound $\bigvee \gamma$ in $X$ by completeness, and moreover $\bigvee \gamma \in S$ since $\forall x \in \gamma B(x) \rightarrow B(\bigvee \gamma)$ by openness of $B$. Therefore by Zorn’s lemma $S$ contains a maximal element, which contradicts (5). Thus $S$ must be empty.

### 3.2 Partial sequences in $T$

In our calculus $T$ we can simulate a partial sequence of type $\mathsf{nat} \rightarrow \rho$ as a total sequence of type $\mathsf{nat} \rightarrow \mathsf{bool} \times \rho$. The idea is that defined values can be represented as $(1, x)$, and undefined values as $(0, 0)$. Accordingly, an object $u$ of this type we say that $n$ is in the domain of $u$, or $n \in \text{dom}(u)$, if $\pi_0 u(n) \neq 0$, and $n \notin \text{dom}(u)$ otherwise. Membership of $\text{dom}(u)$ is a decidable property.

We imagine $\mathsf{bool} \times \rho$ as simulating a type $\rho \equiv \rho + 1$, and we write $u(n) =_\rho x$ instead of $u(n) =_{\mathsf{bool} \times \rho} (n, x)$ whenever $n \neq 0$, and $u(n) =_\rho \bot$ otherwise. Similarly, $u =_\rho v$ if for all $n$ we have either $\pi_0 u(n) = \pi_1 u(n) = 0$ or $\pi_0 u(n), \pi_0 v(n) \neq 0$ and $\pi_1 u(n) = \pi_1 v(n)$.

We extend the overwrite operation $\oplus$ given for finite sequences in Section 2.2 to partial functions by defining $u \oplus v : \rho^{\mathsf{nat}}$ by

$$(u \oplus v)(n) = \begin{cases} u(n) & \text{if } n \in \text{dom}(u) \\ v(n) & \text{if } n \notin \text{dom}(u). \end{cases}$$

We define $u \odot \alpha : \rho^{\mathsf{nat}}$ for total sequence $\alpha^{\mathsf{nat}}$ analogously. It will always be clear from the context which types the operator $\oplus$ takes as input.

We isolate as a special case the addition of a single value to $u$ whenever there is some $n \notin \text{dom}(u)$ such that $n \in \text{dom}(v)$. In particular $u^*_n \supseteq u$ when $n \notin \text{dom}(u)$.

### 3.3 Backward induction

We are now ready to define backward induction, which we simply take to be open induction over the partial order $\langle \mathsf{nat} \rightarrow \rho, \sqsubseteq \rangle$, or in other words, the schema

$$\text{backI} : \forall u^{\mathsf{nat} \rightarrow \rho}(\forall v \sqsubseteq u B(v) \rightarrow B(u)) \rightarrow \forall u B(u)$$

where $B$ ranges over open predicates. This formulation of backward induction makes sense and is valid in any interpretation of $T$ that admits arbitrary sequences, since for any chain $\gamma$ in $T_{\mathsf{nat} \rightarrow \rho}$,

$$(\bigvee \gamma)(n) := \begin{cases} x(n) & \text{if } n \in \text{dom}(x) \text{ for any } x \in \gamma \\ \bot & \text{otherwise}. \end{cases}$$

is a perfectly well-defined object of $\mathbb{N} \rightarrow T_{\rho} \equiv T_{\mathsf{nat} \rightarrow \rho}$ and is a least upper bound of $\gamma$ with respect to $\sqsubseteq$. However, since $\bigvee \gamma$ is not definable in $T$ itself, it will be convenient to have a model-independent definition of an open formula. The following (somewhat ad-hoc) definition of openness will be sufficient for everything that follows.

**Definition 3.2.** A formula $B(u)$ on partial sequences is open if it is of the form

$$B(u) : \equiv \forall n[n \in \text{dom}(u) \rightarrow A(n, u)] \rightarrow \forall n S([u](n)) \rightarrow \exists n P([u](n))$$

where $P$ and $S$ are arbitrary predicates on $\rho^*$, and $A$ is monotone in the following sense

$$u \sqsubseteq v \rightarrow A(n, u) \rightarrow A(n, v).$$

(6)
This is a generalisation of the model-independent notion of openness given in [5], where open formulas defined to be those of the form \( \exists n S([u](n)) \rightarrow \exists n P([u](n)) \). However, it should be pointed out that in [5] a more general form of open induction is treated, namely open induction over the lexicographic ordering on partial sequences, and as such a more restrictive notion of openness is required - indeed open formulas in the sense of Definition 3.2 are not necessarily open with respect to the lexicographic ordering.

Nevertheless, entirely analogously to [5], backward induction does not require the full strength of Zorn’s lemma, and is provable from just dependent choice, using a version of the minimal-bad-sequence argument due to Nash-Williams [19].

**Proposition 3.3.** The principle of backward induction for formulas open in the sense of Definition 3.2 is provable in \( T + DC \).

**Proof.** Suppose for contradiction that we have \( \forall u (\neg B(u) \rightarrow \exists v \subseteq u \neg B(v)) \) but there exists some partial sequence \( u_0 \) such that \( \neg B(u_0) \). Using dependent choice construct the sequence \( (u_n) \) as follows: Supposing that we have already constructed \( \langle u_0, \ldots, u_n \rangle \) for \( n \geq 0 \), define

(i) \( u_{n+1} := w \) if \( n \notin \text{dom}(u_n) \) and \( \neg B(w) \) holds for some \( w \supseteq u_n \) such that \( [w](n) = [u_n](n) \) and \( n \in \text{dom}(w) \);

(ii) \( u_{n+1} := u_n \) if either \( n \in \text{dom}(u_n) \) or \( n \notin \text{dom}(u_n) \) and no suitable \( w \) exists.

First, it is clear by definition that for all \( n \) we have \( \neg B(u_n) \) and \( [u_n](n) = [u_{n+1}](n) \) and \( u_n \subseteq u_{n+1} \). Define \( \hat{u} := \lambda n.u_{n+1}(n) \). Then it follows that for all \( n \) we have \( [\hat{u}](n) = [u_n](n) \) and \( u_n \subseteq \hat{u} \). The first of these is done by a simple induction. For the latter, take \( i \in \text{dom}(u_n) \). Then either \( i < n \) in which it is clear that \( \hat{u}(i) = u_n(i) \), or \( i \geq n \) and we obtain \( \hat{u}(i) = u_{n+1}(i) = u_n(i) \) by \( u_n \subseteq u_n \subseteq \hat{u} \).

Now we prove that \( \neg B(\hat{u}) \). Note that \( \neg B(u) \) is classically equivalent to

\[
\forall i([i \in \text{dom}(u) \rightarrow A(i, u)] \land S([u](i)) \land \neg P([u](i))),
\]

so taking an arbitrary \( n \) and setting \( u = u_{n+1} \) and \( i = n \) we get, by \( \neg B(u_{n+1}) \),

\[
[n \in \text{dom}(u_{n+1}) \rightarrow A(n, u_{n+1})] \land S([u_{n+1}](n)) \land \neg P([u_{n+1}](n)).
\]

But \( S([u_{n+1}](n)) \land \neg P([u_{n+1}](n)) \rightarrow S([\hat{u}](n)) \land \neg P([\hat{u}](n)) \), and furthermore \( n \in \text{dom}(\hat{u}) \) implies \( n \in \text{dom}(u_{n+1}) \) and hence \( A(n, u_{n+1}) \), so using monotonicity of \( A \) and the fact that \( u_{n+1} \subseteq \hat{u} \) we have \( A(n, \hat{u}) \) and hence we have established

\[
n \in \text{dom}(\hat{u}) \rightarrow A(n, \hat{u}).
\]

This establishes \( \neg B(\hat{u}) \). But now we know by the backward induction hypothesis that there exists some \( v \supseteq \hat{u} \) such that \( \neg B(v) \) holds. Let \( n \) be the least point such that \( n \in \text{dom}(v) \) but \( n \notin \text{dom}(\hat{u}) \). Then \( [v](n) = [\hat{u}](n) = [u_n](n) \) and \( n \notin \text{dom}(u_n) \) else we’d have \( n \in \text{dom}(\hat{u}) \) by \( u_n \subseteq \hat{u} \). In addition, \( u_n \subseteq \hat{u} \subseteq v \) implies \( u_n \subseteq v \). Then by the existence of at least one such \( v \) we’d have \( n \in \text{dom}(u_{n+1}) \) and thus \( n \in \text{dom}(\hat{u}) \), a contradiction. Therefore there is no initial sequence \( u_0 \) satisfying \( \neg B(u_0) \), and we’re done. \( \square \)

As we will see in Section 4 and especially Section 5 one of the key ideas in this paper is construct forms of recursion based on restricted, or relativised variants of backward recursion which take as input partial functions that are downward closed with respect to some relation on \( \text{nat} \).

**Proposition 3.4.** Let \( \triangleleft \) be some decidable relation on \( \text{nat} \), and define the predicate \( u \in D_{\triangleleft} \) by

\[
u \in D_{\triangleleft} \equiv \forall n \in \text{dom}(u)[\forall i \leq n(i \in \text{dom}(u))].
\]

Equivalently, we say that \( u \) is \( \triangleleft \)-closed. Then for any relation \( \sqsubset \) on \( \hat{\text{nat}} \) such that \( u \sqsubseteq v \rightarrow u \sqsubseteq v \), the following principle of relativised backward induction is provable from \( \text{backI} \):

\[
\forall u \in D_{\triangleleft}(\forall v \sqsubseteq u[v \in D_{\triangleleft} \rightarrow B(v)] \rightarrow B(u)) \rightarrow \forall u \in D_{\triangleleft} B(u).
\]
and prove that the recursor defines a total

In addition, if we define a founded recusor we are working in a structure such as PCF or the Scott continuous functionals, we can define a well-ordered variant of bar induction. In particular, if for some relation we have that relativised backward induction just becomes update induction in the sense of [5].

Example 3.5 (Update induction). If we define if \( v \) is an update of \( u \), and let \(<\) just be the empty relation, then relativised backward induction just becomes update induction in the sense of [5].

Example 3.6 (Bar induction). For the case that \(<\) is defined as \( u \sqsubseteq v \) if \( v \) is an update of \( u \), we have that relativised backward induction just becomes update induction in the sense of [5].

Proof. First note that \( u \in D_\triangleleft \) is of the form \( \forall n \in \text{dom}(u)D_0(u, u) \) with \( D_0(u, u) \) monotone in the sense of [5], therefore the predicate \( B'(u) \) is open for any open \( B(u) \). Thus we obtain

\[
\forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall uB'(u)
\]

the first implication following from the inclusion \( \subseteq \triangleleft \subseteq \) and the second from normal backward induction applied to \( B'(u) \). Rearranging this gives us relativised backward induction.

Example 3.5 (Update induction). If we define if \( v \) is an update of \( u \), and let \(<\) just be the empty relation, then relativised backward induction just becomes update induction in the sense of [5].

Example 3.6 (Bar induction). For the case that \(<\) is defined as \( u \sqsubseteq v \) if \( v \) is an update of \( u \), we have that relativised backward induction just becomes update induction in the sense of [5].

Proof. First note that \( u \in D_\triangleleft \) is of the form \( \forall n \in \text{dom}(u)D_0(u, u) \) with \( D_0(u, u) \) monotone in the sense of [5], therefore the predicate \( B'(u) \) is open for any open \( B(u) \). Thus we obtain

\[
\forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall uB'(u)
\]

the first implication following from the inclusion \( \subseteq \triangleleft \subseteq \) and the second from normal backward induction applied to \( B'(u) \). Rearranging this gives us relativised backward induction.

\[
\forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall u(\forall v \sqsubseteq uB'(v) \rightarrow B'(u)) \rightarrow \forall uB'(u)
\]

3.4 Backward recursion

The purpose of introducing backward induction was to give us a way to reason about backward recursion, which we define and discuss in this section. To begin with, let us consider as a comparison the entirely standard concept of well-founded recursion over some decidable well-founded relation \( \prec \) on \( \rho \). Assuming we are working in a structure such as PCF or the Scott continuous functionals, we can define a well-founded recusor \( \text{wR}_\triangleleft \) as the fixpoint of the following recursive equation

\[
\text{wR}_\triangleleft^\psi(x) =_\sigma \psi_x(\lambda y . \text{wR}_\triangleleft^\psi(y) \text{ if } y \prec x),
\]

and prove that the recursor defines a total functional for any outcome type \( \sigma \) using well-founded induction over \( \prec \):

\[
\text{wl} : \forall x(\forall y \prec xA(y) \rightarrow A(x)) \rightarrow \forall xA(x).
\]

We want to define a backward recusor in a similar way - although we have two problems: firstly the relation \( u \sqsubseteq v \) is not decidable, and secondly backward induction is only valid for open formulas. We avoid these issues by defining backward recursion to be the fixpoint of the following recursive equation

\[
\text{Back}^\psi_{\rho, \tau}(u) =_\tau \psi_u(\lambda n, v . \text{Back}^\psi_{\rho, \tau}(u \sqcup v) \text{ if } n \in \text{dom}(v) \setminus \text{dom}(u)),
\]

where \( \tau \) is restricted to being a discrete type, while \( n \in \text{dom}(v) \setminus \text{dom}(u) \) denotes the decidable predicate \( n \in \text{dom}(v) \wedge n \notin \text{dom}(u) \). Observe that any \( w \sqcup u \) is of the form \( w = u \sqcup v \) for some \( v \) which is defined at least at one point \( n \notin \text{dom}(u) \), and so \( \text{Back} \) makes recursive calls over all \( w \sqcup u \), although crucially it must always have access to a point \( n \in \text{dom}(w) \) such that \( n \notin \text{dom}(u) \). The necessity of the restriction on \( \tau \) is to ensure that totality of \( \text{Back}^\psi_{\rho}(u) \) is an open property on (domain-theoretically) total input \( u \): indeed for discrete \( \tau \) and total \( u \) we have

\[
\text{Back}^\psi_{\rho, \tau}(u) \text{ is total } \leftrightarrow \exists n \forall w(\text{Back}^\psi_{\rho, \tau}([u](n) \sqcup w) \text{ is total})
\]
assuming sequential continuity Cont. Therefore totality of Back is provable using backward induction. Note that alternatively, a direct proof via Zorn’s lemma that Back exists in the Scott partial continuous functionals (and therefore also the total continuous functionals since ‘$\forall^c$’ is the extensional collapse of total elements of the partial continuous functionals) can be carried out using the same manner as the proof of totality of the Berardi-Bezem-Coquand functional in [4].

As a final alternative, one can just show that backward recursion is definable from the slightly more general schema of open recursion on the lexicographic ordering considered in [5]. Open recursion is defined to be the fixpoint of the following recursive equation:

\[
\text{Open}^\psi_{\rho,\tau}(u) =_\tau \psi_u(\lambda n, v . \text{Open}^\psi_{\rho,\tau}([u](n) \oplus v) \text{ if } n \in \text{dom}(v) \setminus \text{dom}(u))
\]

where once again $\tau$ is discrete, and lexicographically open recursive functionals of the above form are shown to be total in [5, Proposition 5.1].

**Proposition 3.7.** Back is instance-wise primitive recursively definable from Open.

**Proof.** Primitive recursively define

\[
m_{n,u,v} := \text{least } i \leq n \text{ s.t. } i \in \text{dom}(v) \setminus \text{dom}(u), \text{ else } n,
\]

and set Back$^\psi_{\rho,\tau}(u) = \text{Open}^\psi_{\rho,\tau}(u)$ where

\[
\tilde{\psi}_u(f^{\text{nat} \times \rho^{\omega} \rightarrow \tau}) := \psi_u(\lambda n, v . f(m_{n,u,v}, u \oplus v) \text{ if } n \in \text{dom}(v) \setminus \text{dom}(u)).
\]

Then expanding definitions we have

\[
\text{Back}^\psi_{\rho,\tau}(u) = \tilde{\psi}_u(\lambda n, v . \text{Back}^\psi_{\rho,\tau}([u](n) \oplus v) \text{ if } n \in \text{dom}(v) \setminus \text{dom}(u))
\]

and

\[
\text{Back}^\psi_{\rho,\tau}(u) = \psi_u(\lambda n, v . \text{Back}([u](m_{n,u,v}) \oplus (u \oplus v)) \text{ if } n \in \text{dom}(v) \setminus \text{dom}(u))
\]

where for (a) we use $n \in \text{dom}(v) \setminus \text{dom}(u) \rightarrow m_{n,u,v} \in \text{dom}(u \oplus v) \setminus \text{dom}(u)$, and (b) follows by minimality of $m_{n,u,v}$. □

**Remark 3.8 (Bar recursion).** It is easy to see that update recursion as defined in [5] is a simple instance of backward recursion. However, it is instructive to pause for a moment to consider a natural instance of bar recursion that arises from backward recursion. Let us define $\text{Bar}(H, s^\psi) := \text{Back}^\psi(\hat{s})$ where

\[
\psi_u(f^{\text{nat} \times \rho^{\omega} \rightarrow \tau}) :=_\tau H(u, \lambda t . f(|t| - 1, \hat{t})).
\]

Then it is not too hard to show that $\text{Bar}$ satisfies

\[
\text{Bar}(H, s) = H(\hat{s}, \lambda t . \text{Bar}(H, s \oplus t) \text{ if } |t| > |s|),
\]

and this can be viewed as a ‘implicitly well-founded’ variant of Spector’s bar recursion [1]. The reason we highlight this is that while several such implicit variants of Spector’s so-called ‘special’ instance of bar recursion have been studied, including both modified bar recursion and the implicit product of selection functions, constructing a direct analogue to the general form is more complicated (for example, an implicit form of the so-called product of quantifiers is known not to exist [14]).

The subtle reason for this is that such variants of bar recursion must not be allowed to access the length of the input sequence $s$. For example, no object $\Phi$ can satisfy the slightly altered equation

\[
\Phi(H, s) = H(s, \lambda t . \Phi(H, s \oplus t) \text{ if } |t| > |s|)
\]

even for discrete output type, since we could just take $\tau = \text{nat}$ and define $H(s, g) := 1 + f(s \ast 0)$, and then $\Phi(\langle \rangle) = n + 1 + \Phi(H, [0](n + 1)) > n$ for all $n$, which cannot hold in any model of arithmetic. Indeed, trying to define this from Back with continuous $\psi_u$ is impossible, since we’d require a non-continuous unbounded search (and thus totality of the underlying instance of Back would no longer be an open predicate). Thus we overcome the difficulty with implicit variants of bar recursion by removing access to the length of the input. Note that this problem is not a feature of modified bar recursion and implicit products of selection functions (or indeed any of the realizers we define in the following sections), since these are defined ‘pointwise’, and make recursive calls only when we are accessing points already greater than the length of the input sequence.

\footnote{See [21] for the distinction between the special and generals forms of Spector’s bar recursor.}
4 A computational interpretation of the double negation shift

In this section we give a new, general realizability interpretation to the double negation shift. Ultimately, this will form a special case of the interpretation of full dependent choice given in the next section. However, by focusing first on the double negation shift we have an opportunity to present our main ideas in a slightly simplified setting.

4.1 Modified realizability interpretation of extensions of $\text{PA}^\omega$

We begin by briefly recalling how Kreisel’s modified realizability can be used in conjunction with the so-called Friedman trick to extract programs from $\Pi^2_1$-formulas. For every formula in the language of $\text{HA}^\omega$ we define the realizability relation $x \text{mr } A$ inductively by

\begin{align*}
() \text{ mr } A &\equiv A \text{ if } A \text{ is an atomic formula,} \\
x, y \text{ mr } (A \land B) &\equiv x \text{ mr } A \land y \text{ mr } B, \\
n^\text{nat}, x, y \text{ mr } (A \lor B) &\equiv (n = 0 \rightarrow x \text{ mr } A) \land (n \neq 0 \rightarrow y \text{ mr } B), \\
f \text{ mr } (A \rightarrow B) &\equiv \forall x (x \text{ mr } A \rightarrow fx \text{ mr } B), \\
x \text{ mr } \forall z A(z) &\equiv \forall z(z \text{ mr } A(z)), \\
x, y \text{ mr } \exists z A(x) &\equiv y \text{ mr } A(x).
\end{align*}

It is a standard result that whenever $\text{HA}^\omega \vdash A$ then $\text{HA}^\omega \vdash t \text{ mr } A$ where $t$ is some primitive recursive term extracted from the proof of $A$. The interpretation of classical logic, on the other hand, is more subtle. A simple combination of the negative translation with modified realizability fails to work since the atomic formula $\bot$ is realized by () and therefore all negated formulas are trivially interpreted. In particular, this method gives us no way of extracting realizers from $\Pi^2_1$-formulas $\forall x^\text{nat}\exists y^\text{nat} A(x, y)$.

One well-established way of overcoming this problem is to slightly alter the definition of modified realizability by regarding $x^\text{mr} \bot$ as an uninterpreted formula. Then, as discussed in e.g. [6, 8], whenever $\text{HA}^\omega + \Delta \vdash \Phi \text{ mr } \Gamma^N$, where $\Gamma^N$ denotes the negative translation of $\Gamma$, $\Delta$ is some set of axioms satisfying some natural closure properties with respect to $\bot$, and $\Phi$ is some closed term in the language of $\text{HA}^\omega + \Delta$, then from a classical proof $\text{PA}^\omega + \Gamma \vdash \forall y \exists x^\text{nat} A(y, x)$ one can extract a term $t$ in $\Phi$ such that $\text{HA}^\omega + \Delta \vdash \forall y A(y, t)$. This utilises the aforementioned Friedman trick of replacing the formula $x \text{ mr } \bot$ by the quantifier-free formula $A(y, x)$.

Thus we have a method that allows us to extract realizers for $\Pi^2_1$-formulas from any extension $\Gamma$ of Peano arithmetic whenever we can realize the negative interpretation $\Gamma^N$ of $\Gamma$. In the remainder of this paper we develop this idea and focus on constructing terms $\Phi$ such that $\Phi \text{ mr } A$, where $\Gamma$ is either countable or countable dependent choice, and $\Gamma^N$ is the adapted realizability interpretation. In fact, following [15] we generalise slightly and replace $\bot$ by some arbitrary formula $R$ whose type of realizers is a discrete type, although if the reader prefers they can just treat this as a relabelling and imagine $R = \bot$.

4.2 The $J$-shift and its variants

The axiom of countable choice is given by

$$AC : \forall n \exists x^\rho B_n(x) \rightarrow \exists \alpha^\text{nat} \forall n B_n(\alpha(n)).$$

It is well-known that the negative translation of $AC$,

$$\forall n ((\exists x B^N_n(x) \rightarrow R) \rightarrow R) \rightarrow ((\exists \alpha \forall n B^N_n(\alpha(n)) \rightarrow R) \rightarrow R),$$

(here with an arbitrary discretely-realized $R$ in place of $\bot$) is provable in intuitionistic logic from the simpler double-negation shift,

$$\text{DNS} : \forall n ((A(n) \rightarrow R) \rightarrow R) \rightarrow (\forall n A(n) \rightarrow R) \rightarrow R,$$

by setting $A(n) := \exists x B^N_n(x)$, and thus a realizability interpretation of countable choice follows directly from that of DNS. Note that this version of DNS for arbitrary $R$ is also called the $K$-shift in [15].
In order to successfully realize DNS one typically relies on a term \( h \) realizing \( \forall n(R \rightarrow A(n)) \), and so in practise one must work with a restricted class of formulas \( A(n) \) (for \( A(n) := \exists x B^N_n(x) \) one can trivially construct \( h \) even uniformly in \( n \)). This need for additional realizers and a corresonding restriction on formulas seems slightly inelegant, and so a cleaner presentation of DNS is given in [15] in which, rather than seperately assuming \( R \rightarrow A(n) \), adds this positive information directly to the premise of DNS, yielding

\[
\forall n((A(n) \rightarrow R) \rightarrow A(n)) \rightarrow (\forall n A(n) \rightarrow R) \rightarrow R.
\]

which in [15] is written in an equivalent form called the \( J \)-shift:

\[
J \text{-shift} : \forall n((A(n) \rightarrow R) \rightarrow A(n)) \rightarrow (\forall n A(n) \rightarrow R) \rightarrow \forall n A(n).
\]

Here, we adopt this convention of [15] in adding the positive information we need directly to the premise of the double negation shift, so that our interpretation is valid for all \( A(n) \). However, for notational reasons we interpret a slightly more flexible variant of the \( J \)-shift designed to match the family of realizers we construct.

**Definition 4.1.** We define the \( J_i \)-shifts for \( i = 1, 2 \) by

\[
J^*_1 \text{-shift} : \forall m, n((A(m) \rightarrow R) \rightarrow A(n)) \rightarrow (\forall n A(n) \rightarrow R) \rightarrow R
\]

\[
J^*_2 \text{-shift} : \forall m, n((A(m) \rightarrow R) \rightarrow A(n)) \rightarrow (\forall n A(n) \rightarrow R) \rightarrow \forall n A(n)
\]

where \( R \) has discrete realizing type.

**Proposition 4.2.** \( J^*_1 \)-shift \( \iff \) \( J^*_2 \)-shift \( \iff \) \( J \)-shift over minimal logic.

**Proof.** \( J^*_1 \)-shift \( \Rightarrow \) \( J^*_2 \)-shift follows from the observation that \( \forall m, n((A(m) \rightarrow R) \rightarrow A(n)) \rightarrow (R \rightarrow \forall n A(n)) \) while \( J^*_2 \)-shift \( \Rightarrow \) \( J \)-shift follows from \( \forall n ((A(n) \rightarrow R) \rightarrow A(n)) \rightarrow \forall m, n((A(m) \rightarrow R) \rightarrow A(n)) \).

The remaining directions are obvious. \( \square \)

The reasons that we highlight these essentially syntactic variants of the \( J \)-shift is that we want versions of the \( J \)-shift that are directly realized by our variants of bar recursion. As we will see, in most cases each variant will only use the premise of \( J^*_1 \)-shift for \( (m, n) \in I \subseteq \text{nat} \times \text{nat} \) for some \( I \), and so we can modify \( J^*_1 \)-shift again to give a version that corresponds directly to the realizer.

### 4.3 Realizing the \( J^*_i \)-shift

We focus on constructing a realizer for the \( J^*_1 \)-shift, then a realizer of the \( J^*_2 \)-shift comes out immediately. Suppose that the realizing types of \( A(n) \) and \( R \) are \( \rho \) and \( \tau \) respectively, where we assume that \( \tau \) is discrete. The \( J^*_1 \)-shift is realized by a term \( \Phi \) of type \( (\text{nat} \rightarrow \text{nat} \rightarrow (\rho \rightarrow \tau) \rightarrow \rho) \rightarrow (\rho^{\text{nat}} \rightarrow \tau) \rightarrow \tau \), which, given terms \( \varepsilon : \text{nat} \rightarrow \text{nat} \rightarrow (\rho \rightarrow \tau) \rightarrow \rho \) and \( q : \rho^{\text{nat}} \rightarrow \tau \) that satisfy

\[
\forall m, n, \rho^{\text{nat}}(\forall x : A(n) \rightarrow p(x) \rightarrow m \rightarrow R) \rightarrow \varepsilon_m, n(\rho) \rightarrow m \rightarrow A(n)
\]

\[
\forall \alpha^{\varepsilon_m}(\forall u : \alpha(n) \rightarrow m \rightarrow R) \rightarrow q(\alpha) \rightarrow m \rightarrow R
\]

returns a term \( \Phi(q) : \tau \) satisfying \( \Phi \vdash q \rightarrow R \).

The basic idea that unites all such existing realizers of double-negation shift principles is to form an auxiliary functional \( \Psi \) which performs a backward recursive loop that builds increasingly large partial realizers for \( \forall n A(n) \). Given a partial realizer \( u : \rho^{\text{nat}} \) which satisfies \( \forall n \in \text{dom}(u) \rightarrow (u(n) \rightarrow A(n)) \), we define \( \Psi^{\varepsilon \cdot q} u := q(u \oplus c(u)) \), with the aim that \( c(u) \) is a completion of the partial realizer of \( u \) and that therefore \( \Psi^{\varepsilon \cdot q} u \rightarrow m \rightarrow R \) for all \( u \). This completion \( c(u) \) is constructed using \( \varepsilon \) for \( n \notin \text{dom}(u) \) we define

\[
c(u)(n) := \varepsilon_m, n(\lambda x : A(n) \rightarrow \Psi^{\varepsilon \cdot q}(v^x_m))
\]

for some index \( m \) and partial realizer \( v \) of \( \forall n A(n) \). This is a realizer or \( A(n) \) because \( \Psi^{\varepsilon \cdot q}(v^x_m) \rightarrow m \rightarrow R \) whenever \( x \rightarrow m \rightarrow A(m) \). Of course, we have not specified what either \( m \) or \( v \) should be, but intuitively, provided \( u \subseteq v^x_m \) and \( v^x_m \) is a partial realizer for each recursive call, we can prove that \( \Psi^{\varepsilon \cdot q} u \rightarrow m \rightarrow R \) for all \( u \) by backward induction. Then setting \( \Phi = \Psi^{\varepsilon \cdot q} \) gives us a realizer for the \( J^*_1 \)-shift.

Such realizers are essentially the computational content of the following proof of the \( J^*_1 \)-shift by backward induction, each different variant corresponding to certain *mathematically* inessential choices made during the proof.
Proposition 4.3. The $J_1^*$-shift (and hence also the $J_2^*$-shift) is provable from backward induction over minimal logic.

Proof. Assume the premise of $J_1^*$-shift. We prove $R$ by backward induction over the formula

$$B(S) \equiv A(S) \rightarrow R,$$

where $S$ denotes a subset of the natural numbers encoded as a function $\text{nat} \rightarrow \text{bool}$, and $A(S)$ is shorthand for $\forall n \in S A(n)$. Clearly $B$ is an open property relative to set inclusion. To prove that it is progressive, assume $B(T)$ for all $T \supseteq S$ and that $A(S)$ holds.

Now, suppose that we have a well-founded relation $\prec_S$ on $\text{nat}$. We prove by well-founded induction that $\forall n A(n)$, and then $B(S)$ follows from $\forall n A(n) \rightarrow R$. For the induction step, suppose that $n \notin S$ and that $A(k)$ holds for all members of $T := S \cup \{k \mid k \prec_S n\}$. Pick some $m_{S,n} \notin S$. Then

$$A(m_{S,n}) \rightarrow A(T \cup \{m_{S,n}\}) \rightarrow R$$

the second step following from the backward induction hypothesis since $T \cup \{m_{S,n}\} \supseteq S$. Therefore by

$$(A(m_{S,n})) \rightarrow A(T \cup \{m_{S,n}\}) \rightarrow A(n)$$

we obtain $A(n)$. This completes the $\prec_S$-induction step, and we’re done.

In the proof of Proposition 4.3 the choice of $\prec_S$ and choice of $m_{S,n}$ is completely arbitrary. In particular we can set $\prec_S = \emptyset$ and $m_{S,n} = n$ to obtain the most simple version of this proof that avoids any auxiliary induction. In this case the proof essentially becomes that of [3, Proposition 3.5 (a)], the computational content of which is the symmetric Berardi-Bezem-Coquand functional. However, alternative choices of these parameters yield realizers with a different behaviour, and as we will see include the well-known bar recursive realizers of both [6] and [15].

Theorem 4.4. Suppose that $\rho$ and $\tau$ are the realizing types of $A(n)$ and $R$, with $\tau$ discrete, and that

$\prec: \rho^{\text{nat}} \rightarrow (\text{nat} \times \text{nat} \rightarrow \text{bool})$ and $m: \text{nat} \times \rho^{\text{nat}} \rightarrow \text{nat}$ satisfy

(i) $\prec_u$ is well-founded,

(ii) $n \notin \text{dom}(u) \rightarrow mn \notin \text{dom}(u)$.

Then there is a term $\Phi_{\prec, m}$ primitive recursive in $\text{Back} + \lambda u. \text{wR}_{\prec_u} \lambda n. \text{which realizes the J}_1^{\text{shift}},$ provably in $\text{E-HA}^+ \text{ + Cont + back} \text{ + Back + (wR}_{\prec_u} \text{ + } \lambda u. \text{wR}_{\prec_u}$.

Remark 4.5. Note that the term $\lambda u. \text{wR}_{\prec_u}$ is not properly defined - we are simply assuming here that there exists a function $F$ definable in $\mathcal{T}$ such that for all $u$, $F(u)$ satisfies the defining equation of $\text{wR}_{\prec_u}$. However, in all of the concrete examples we consider, $\prec$ will not depend on $u$ and $\text{wR}_{\prec}$ will always be trivially definable in $\mathcal{T}$, so this rather casual definition will not be problematic.

Proof of Theorem 4.4. We define an auxiliary realizer $\Psi_{\prec, m}^{\varepsilon}$ of $\rho^{\text{nat}} \rightarrow \tau$ as the fixpoint of the recursive equation

$$\Psi^{\varepsilon,q}(u) = q(u \equiv \lambda \alpha . \varepsilon_{\text{mn}},_u(\lambda x . \Psi^{\varepsilon,q}(u \equiv t\left[n \cup u\right]\alpha))$$

where $t\left[n \cup u\right]: \rho^{\text{nat}}$ is defined via $\text{wR}_{\prec_u}$ on $\alpha$ to be $\lambda \kappa. (\lambda \alpha (k) \text{ if } k \prec_u n)$. It is fairly easy to show that $\Psi$ can be constructed using $\text{Back}$ and the term $\lambda u. \text{wR}_{\prec_u}$, and is therefore well-defined.

To prove correctness, assume that $\varepsilon$ and $q$ realize the premise of the $J_1^*$-shift. Analogous to the proof of Proposition 4.3, we use a main backward induction and an auxiliary well-founded induction, the purpose of $\Psi$ being to realize $B(S)$. So now we define

$$B(u) \equiv \forall n \in \text{dom}(u)(u(n) \text{ mr } A(n)) \rightarrow \Psi^{\varepsilon,q}(u) \text{ mr } R.$$

This is an open formula since by $\text{Cont}$ we have $\Psi(u) \text{ mr } R \leftrightarrow \exists n \forall w \Psi([u][v] \cup w) \text{ mr } R$. Now, to prove progressiveness of $B$, assume that $u$ is a partial realizer of $\forall n A(n)$ and suppose that $B(v)$ holds for all $v \supseteq u$. 

12
We prove by \(\text{wl}_{\prec u}\) that \(\alpha_u\) realizes \(\forall n A(n)\). Suppose that \(n \notin \text{dom}(u)\) and that \(\alpha_u(m) \not\equiv A(k)\) for all \(k \prec_u n\), so in particular \(u \not\vDash t[u,n]A\) is a partializer of \(\forall n A(n)\). Therefore we have

\[ x \not\vDash A(mnu) \rightarrow \forall i \in \text{dom}(v)(v(i) \not\vDash A(i)) \rightarrow \Psi^{\vDash q}(v) \not\vDash R \]

for \(v := u \oplus t[u,n]^x_{mnu}\), the second step following from the main hypothesis since \(mnu \notin \text{dom}(u)\) and thus \(v \not\vDash u\). Therefore \(\forall x.\Psi(u \oplus t[u,n]^x_m) \not\vDash A(m) \rightarrow R\) and therefore

\[ \alpha_u(n) = \vDash_{mnu,n}(\lambda x.\Psi(u \oplus t[u,n]^x_m)) \not\vDash A(n). \]

This completes the auxiliary well-founded induction, giving us \(\forall n(\alpha_u(n) \not\vDash A(n))\), and therefore \(\Psi(u) = \vDash q(\alpha_u) \not\vDash R\), which completes the main backward induction step. Finally, then, we obtain \(\forall v B(u)\) by \(\text{backl}\) and so in particular defining \(\Phi_{(\vDash q,\vDash \lambda u.\not\vDash)} := \Psi_{(\vDash q,\vDash \lambda u.\not\vDash)}(\emptyset)\) we obtain \(\Phi_{\vDash q} \not\vDash R\) by \(B(\emptyset)\).

**Corollary 4.6.** There is a term \(\Phi_{(\vDash q)}\) primitive recursive in \(\text{Back} + \lambda u.\not\vDash u\) which realizes the \(J^2\)-shift, provably in \(\text{E-HA}^\omega + \text{Cont} + \text{backl} + \text{Back} + (\text{wl}_{\prec u}) + \lambda u.\not\vDash u\).

**Proof.** Keeping all the notation of Theorem 4.4 define \(\Psi_{\vDash q}(u) = \alpha_u\) so that it satisfies the recursive equation

\[ \Psi_{\vDash q}(u) = u \oplus \lambda n.\vDash_{mnu,n}(\lambda x.\vDash q(\Psi_{\vDash q}(u \oplus t[u,n]^x_{mnu}))). \]

Define \(\Psi_{\vDash q} := \Psi_{\vDash q}(\emptyset)\). Then it follows immediately from \(\forall v B(u)\) and \(\vDash q\)-induction that \(\Phi_{\vDash q} \not\vDash R\) \(\forall n A(n)\).

Let us now briefly consider some specific instantiations of the parameters of Theorem 4.4 (a more detailed discussion will be given in Section 5.2). Firstly, in the simple case that \(\prec_u = \emptyset\) and \(mnu = n\) for all \(u\), we have \(t[u,n] = \emptyset\) and, using the abbreviation \(\vDash n,\vDash \vDash q\) for \(\vDash n,\vDash \vDash q\) our realizer becomes

\[ \Psi_{\vDash q}(u) = q(u \oplus \lambda n.\vDash_{n}(\lambda x.\vDash q(\Psi_{\vDash q}(u \oplus t[u,n]^x_{mnu}))). \]

which is nothing more than a simple variant of the Berardi-Bezem-Coquand realizer of countable choice given in [3] and discussed in [5]. On the other hand, suppose that we still keep \(\prec_u = \emptyset\) for all \(u\), but define \(mnu := m \leq n(i \in \text{dom}(u))\). Then finite input for this variant of \(\Psi\) will be of the form \(\tilde{s}\) for some sequences \(s : \rho^\omega\). Defining \(\Psi_1(s) := \Psi_{(\vDash q,\vDash \lambda u.\not\vDash)}(\tilde{s})\) and observing that for \(n > |s|\) we have \(mnu = \emptyset\) we have \(\Psi_{\vDash q}(s) = q(\tilde{s} \oplus \lambda n.\vDash_{\vDash n}(\lambda x.\vDash q(\Psi_{\vDash q}(s \oplus t[u,n]^x_{mnu}))))\) which is just a (non-dependent) form of modified bar recursion. Finally, let us define \(\prec_u = \emptyset\) and \(mnu = n\) for all \(u\). Then setting \(\Psi_2(s) := \Psi_{(\vDash q,\vDash \lambda u.\not\vDash)}(\tilde{s})\), abbreviating \(\vDash_{\vDash q,\vDash \vDash q}\) by \(\vDash_{\vDash q,\vDash \vDash q}\) and observing that \(\tilde{s} \oplus t[u,n] := [\alpha_u](n)\) we obtain a realizing term satisfying

\[ \Psi_{\vDash q}(s) = q(\tilde{s} \oplus \lambda n.\vDash_{\vDash n}(\lambda x.\vDash q(\Psi_{\vDash q}(s \oplus t[u,n]^x_{mnu})))). \]

The corresponding variant \(\Psi_2\) which realizes \(J^2\)-shift is exactly the simple implicit product of selection functions of [12].

Thus three completely different modified realizability interpretations of countable choice appear as simple instances of Theorem 4.4. Moreover, in each instance we only require a restricted form of backward induction and well-founded recursion which corresponds exactly to the soundness proofs used in the original papers: for \(\Psi_0\) Theorem 4.4 reduced to the proof of the double negation shift using update recursion given in [3], while \(\Psi_1\) and \(\Psi_2\) require backward induction relativised to downward closed partial functions, which is entirely equivalent to the variants of bar induction used to prove their correctness in [6] and [12] respectively. Thus Theorem 4.4 doesn’t simply provide a parametrised framework with which different realizers can be compared, but also a framework in which their correctness proofs can be viewed in a uniform way as relativisations of backward induction.

The construction of such a framework is only partially motivated by the desire to compare existing interpretations. Theorem 4.4 generalises existing work in that one can use an arbitrary parameters to define new realizers of the \(J^2\)-shift that are automatically correct, giving an additional level of flexibility and power when it comes to extracting computational content from proofs in practise.

However, we do not discuss this in any more detail for now, instead proceeding straight to the generalisation of Theorem 4.4 to full dependent choice.
5 A computational interpretation of dependent choice

We give a parametrised realizability interpretation to the principle of countable dependent choice. Our formulation of dependent choice will be slightly more general that the usual sequential variants treated in e.g. [6, 15, 26] in that we parametrise the principle itself by a decidable well-founded relation \(<\) on \(\text{nat}\) which dictates the underlying dependency of the choice sequence.

To be more precise, given a decidable (strict) well-founded partial order \(<\) let us extend our type system with types \(\rho^s\) which represent the set \(\bigcup_{n \in \text{nat}} \rho^s_n\) where

\[
\rho^s_n \defeq \{ m \mid m < n \} \rightarrow \rho.
\]

We tacitly assuming that the types \(\rho^s\) can be smoothly incorporated into our system, and come equipped with a length function \(|\cdot| : \rho^s \rightarrow \text{nat}\) returning for each \(t : \rho^s\) a unique index \(|t|\) such that \(t \in \rho^{|t|}\).

For \(\langle\cdot, \cdot\rangle\), the type \(\rho^s\) is isomorphic to the type \(\rho^*\) of finite sequences over \(\rho\), in fact \(\rho^s\) is essentially a generalisation of the finite sequence type to arbitrary \(<\)-closed partial functions (which need not have finite domain, though). Note that \(\rho^p\) is isomorphic to \(\text{nat}\), and as we will see, for \(\langle\cdot, \cdot\rangle = \emptyset\) our parametrised choice principle collapses to normal countable choice.

Now, the principle of \(<\)-DC\(_\text{seq}\) is given by the schema

\[
\forall s, r \forall i \mid s.A_i(\{s\}(i), s(i)) \rightarrow \exists x . A_i|s|(s,x) \rightarrow \exists \alpha^n. \forall n . A_n(\{\alpha\}(n), \alpha(n))
\]

where \(\{\alpha\}(n) : \rho^s_n\) is just the \(<\)-initial segment of \(\alpha\) i.e. \(\lambda m \mid \alpha(m)\), and analogously for \(s\) (note that \(\{s\}(i)\) is well-defined for \(i < |s|\) by transitivity of \(<\)). Note that \(<\)-DC\(_\text{seq}\) follows from the full axiom of choice together with well-founded recursion over \(<\) as follows: by classical logic and full choice we have

\[
\forall s, r \forall i \mid s.A_i(\{s\}(i), s(i)) \rightarrow \exists x . A_i|s|(s,x) \rightarrow \exists \Theta . \forall s. \forall i \mid s.A_i(\{s\}(i), s(i)) \rightarrow A_i(s, \Theta s)).
\]

Now, recursively defining \(\alpha(n) \defeq \Theta(\{\alpha\}(n))\), we prove \(\forall n.A_n(\{\alpha\}(n), \alpha(n))\) by \(\text{wI}\), since from the assumption that \(\forall i \mid n.A_i(\{\alpha\}(n), \alpha(i))\) we obtain \(A_n(\{\alpha\}(n), \alpha(n))\) using that fact that \(\{\alpha\}(n)\} = \{\alpha\}(i)\) for \(i < n\).

5.1 Realizing \(<\)-DC\(_\text{seq}^J\)

As in the previous section, we realize a positive form of the negative translation of \(<\)-DC\(_\text{seq}\) which is somewhat analogous to the dependent \(J\)-shift of [15].

Definition 5.1. We define the translated principle \(<\)-DC\(_\text{seq}^J\) by

\[
\langle\rho^s\rangle \rightarrow \langle\rho^s \times \rho^s \times \rho\rangle
\]

where \(A\) is arbitrary formula over \(\text{nat} \times \rho^s \times \rho\) and the realizing type of \(R\) is restricted to being discrete.

If \(\sigma\) and \(\tau\) are the realizing types of \(A_n(s, x)\) and \(R\) respectively, then \(<\)-DC\(_\text{seq}^J\) is realized by a term \(\Phi\) of type \((\rho^s) \rightarrow (\rho \times \sigma)^s \rightarrow (\rho \times \sigma \rightarrow \tau) \rightarrow \rho \times \sigma) \rightarrow ((\rho \times \sigma)^{\text{nat}} \rightarrow \tau) \rightarrow \tau\) which given input \(\varepsilon : \rho^s \rightarrow (\rho \times \sigma)^s \rightarrow (\rho \times \sigma \rightarrow \tau) \rightarrow \rho \times \sigma\) and \(q : (\rho \times \sigma)^{\text{nat}} \rightarrow \tau\) that satisfy

\[
\forall \langle s(i) \rangle, \rho^{|s(i)|}, \rho^{|s(i)|} \rightarrow \tau
\]

returns a term \(\Phi_{\text{seq}^q} : \tau\) satisfying \(\Phi_{\text{seq}} \rightarrow \tau\). As in the previous section, we construct a family of realizing terms which follow the same basic principle of backward recursion on partial functions, and which represent the computational content of a proof of \(<\)-DC\(_\text{seq}^J\) by backward induction.
Proposition 5.2. \( \triangleleft \text{-DC}_{\text{seq}} \) is provable from backward induction.

Proof. The backward induction takes place over the formula

\[
B(u) : A(u) \rightarrow R
\]

where \( u \) has type \( \rho^\text{nat} \) and now

\[
A(u) \equiv \forall n \in \text{dom}(u)(\forall k \prec n(k \in \text{dom}(u)) \land \text{dom}_{\text{seq}}(\{u\}(n), u(n))).
\]

It is easy to see that \( B \) is open in the sense of Definition 3.2. To show progressiveness of \( B \) let us now assume that \( B(v) \) holds for all \( v \sqsubseteq u \) and that the premise of \( B(u) \) is true.

We want to establish \( R \), which we do by showing that there exists \( u \sqsubseteq \alpha \) such that \( \forall n \text{dom}_{\text{seq}}(\{\alpha\}(n), \alpha(n)) \).

We do this by well-founded induction on some relation \( \prec \), together with \( \triangleleft \). Take some \( n \notin \text{dom}(u) \) and assume that we have already defined \( \alpha \) for all \( k \prec n \).

Suppose that in addition \( \alpha \) has been defined correctly for \( k \prec u \).

Therefore, since \( \forall i \prec mnu \text{dom}_{\text{seq}}(\{\alpha\}(i), \alpha(i)) \) holds we have \( \text{dom}_{\text{seq}}(\{\alpha\}(n), \alpha(n)) \) which implies \( R \). This in turns completes the backward induction hypothesis we have

\[
\exists x \text{dom}_{\text{seq}}(\{\alpha\}(mnu), x) \rightarrow A(\alpha|T \oplus (mnu, x)) \rightarrow R
\]

and therefore since \( \forall i < mnu \text{dom}_{\text{seq}}(\{\alpha\}(i), \alpha(i)) \) holds we have \( \text{dom}_{\text{seq}}(\{\alpha\}(n), x) \) for some \( x \).

Setting \( \alpha(n) := x \) then completes the well-founded induction step, and thus \( \forall n \text{dom}_{\text{seq}}(\{\alpha\}(n), \alpha(n)) \).

As with the proof of Proposition 4.3, the choices of \( \prec \) and \( mnu \) are arbitrary with the exception of some additional conditions to ensure they are now compatible with \( \triangleleft \).

Theorem 5.3. Suppose that \( \sigma \) and \( \tau \) are the realizing types of \( A(n) \) and \( R \), with \( \tau \) discrete, and that \( \triangleleft_{\text{seq}} : (\rho \times \sigma) \rightarrow (\text{nat} \times \text{nat} \rightarrow \text{bool}) \) and \( m : \text{nat} \times (\rho \times \sigma) \rightarrow \text{nat} \) satisfy

(i) \( \prec \cup \triangleleft \) is well-founded,

(ii) \( n \notin \text{dom}(u) \rightarrow mnu \notin \text{dom}(u) \).

(iii) \( u \) is \( \triangleleft \)-closed \( \rightarrow u \cup \{k \mid k \prec u \} \cup \{mnu\} \) is \( \triangleleft \)-closed.

Then there is a term \( \Phi(\triangleleft_{\text{seq}}, m) \) primitive recursive in \( \text{Back} + \text{wR}_{\text{seq}} \cup \triangleleft \) which realizes \( \triangleleft \text{-DC}_{\text{seq}}^\text{nat} \), provably in \( \text{E-HA}^\omega + \text{Cont} + \text{back} + \text{Back} + (\text{wR}_{\text{seq}} \cup \triangleleft) + \text{laR}_{\text{seq}} \cup \triangleleft \).

Proof. As before, we define an auxiliary realizor \( \Psi^{\text{eq}}_{\triangleleft_{\text{seq}}, m} : (\rho \times \sigma) \rightarrow \tau \) as the fixpoint of the recursive equation

\[
\Psi^{\text{eq}}_{\triangleleft_{\text{seq}}, m}(u) = q\left[u \oplus \lambda n. \varepsilon_{\{\alpha\}(mnu), \alpha(n)}(\lambda x. \Theta^{\text{eq}}(u \oplus t[n, u]_x))\right]_{\alpha_u}
\]

where \( t[n, u] : (\rho \times \sigma) \rightarrow \text{nat} \) is defined as \( t[n, u] = \lambda k. (\alpha_u(k)) \) if \( k \prec u \).

Note that \( \alpha_u \) is well-defined by \( \prec \cup \triangleleft \)-recursion since in order to compute \( \alpha_u(n) \) we require \( \alpha_u(k) \) for either \( k \prec_u n \), \( n \prec \alpha_u \), or \( k \prec mnu \), and in the latter case by (iii) this implies that either \( k \in \text{dom}(u) \) or \( k \prec u n \).

Moreover, by (iii) and (ii), if \( n \notin \text{dom}(u) \) recursive calls on \( \Psi \) are always made on strict extensions \( u \).

Therefore \( \Psi^{\text{eq}} \) is definable in \( \text{Back} + \text{wR}_{\text{seq}} \cup \triangleleft \).

For correctness, we assume that \( \varepsilon \) and \( q \) realize the premise of \( \triangleleft \text{-DC}_{\text{seq}}^N \) and use backward induction on the formula

\[
B(u) : \equiv A(u) \rightarrow \Psi^{\text{eq}}(u) \rightarrow R.
\]

where

\[
A(u) : \equiv \forall n \in \text{dom}(u)(\forall k \prec n(k \in \text{dom}(u)) \land u(n)) \rightarrow A(\{u_0\}(n), u(n)).
\]
Again, \( B(u) \) is open by continuity of \( \Psi \) and the fact that its premise is of the form \( \forall n \in \text{dom}(u) \rightarrow A'(n, u) \) for monotone \( A'(n, u) \). To prove progressiveness of \( B \), assume that \( u \) is a \((\loopsof u\text{-closed})\) partial realizer i.e. \( A(u) \) holds, and that \( B(v) \) holds for all \( v \sqsupseteq u \).

We prove by \( \text{wl}_{\sqsubseteq} \cup \text{U} \sqsubset \) that

\[
\forall n(\alpha_u(n) \text{ mr } A_n(\{(\alpha_u)_0\}(n), \alpha_u(n)u))
\]

and then \( \Psi^{\varepsilon, q}(u) = q(\alpha_u) \text{ mr } R \), completing the backward induction step. So, for \( n \notin \text{dom}(u) \) \((C(n) \text{ clearly})\) holds for \( n \in \text{dom}(u) \) suppose that \( \forall k \preceq u nC(k) \) holds. Then by this, the fact that \( u \) is a partial realizer and that \( \text{dom}(u) \cup \{k \mid k \preceq u n\} \cup \{mn\} \) is \( \sqsubseteq \)-closed we have

\[
\forall x[1 \text{ mr } A_{\text{mn}}(\{(\alpha_u)_0\}(mn), x_0)] \rightarrow A(u \uplus t[n, u]^x_{n_{\text{mn}}}) \rightarrow \Psi(u \uplus t[n, u]^x_{n_{\text{mn}}}) \text{ mr } R].
\]

by the main induction hypothesis. Since we clearly have \( \forall k \preceq mn \alpha(k)_1 A_k(\{(\alpha_u)_0\}(k), \alpha(k)_0) \), then

\[
\alpha_u(n) = \varepsilon_{\{(\alpha_u)_0\}(mn), (\alpha_u)_0}(\lambda x. \Psi(u \uplus t[n, u]^x_{n_{\text{mn}}}))
\]

implies \( \alpha_u(n) \text{ mr } A_n(\{(\alpha_u)_0\}(n), \alpha_u(n)u) \text{ i.e. } C(n) \). This completes the well-founded induction step, and therefore \( q(\alpha_u) \text{ mr } R \), which completes the backward induction step. Therefore \( \forall u B(u) \) and hence by \( B(\emptyset) \) we have \( \Psi_{(\emptyset, \prec, m)q} := \Psi_{(\emptyset, \prec, m)}(\emptyset) \).

**Corollary 5.4.** There is a term \( \tilde{\Phi}_{(\prec, \prec, m)} \) primitive recursive in \( \text{Back} + \lambda u.\text{wR}_{\prec u} \) which realizes the \( \triangleleft_{-\text{DC}^2_{\text{seq}}} \)-closed \( \text{wR}_{\prec u} \) in \( \text{E-HA}^2 + \text{Cont} + \text{back}l + \text{Back} + \text{wl}_{\leq u} + \lambda u.\text{wR}_{\prec u} \).

**Proof.** Just as in the previous section, and keeping the notation of Theorem 5.3 define \( \tilde{\Psi}^{\varepsilon, q}(u) = \alpha_u \) so that it satisfies the recursive equation

\[
\tilde{\Psi}^{\varepsilon, q}(u) = u \uplus \lambda n. \varepsilon_{\{(\alpha_u)_0\}(mn), (\alpha_u)_0}(n)(\lambda x. q(\tilde{\Psi}^{\varepsilon, q}(u \uplus t[n, u]^x_{n_{\text{mn}}}))).
\]

Define \( \tilde{\Phi}_{sq} := \tilde{\Psi}^{\varepsilon, q}(\emptyset) \). Then it follows immediately from \( \forall u B(u) \) and \( \prec_0 \cup \triangleleft \)-induction that \( \tilde{\Phi}_{sq} = \alpha_0 \text{ mr } \exists x \forall n A(n) \) since \( \forall n \alpha_0(n) \text{ mr } A_n(\{(\alpha_0)_0\}(n), \alpha_0(1)) \).

### 5.2 Examples

We now show that essentially all of the solutions to the modified realizability interpretation choice principles given across [3, 6, 13, 26] appear as special cases of Theorem 5.3 given suitable instantiations of \( \sqsubseteq, \preceq, m \) and \( A_n(s, x) \).

#### 5.2.1 The Berardi-Bezem-Coquand functional \((\circ \equiv \circ \leq \emptyset = \emptyset)\)

It is easy to see that the principle \( \text{\text{2}{\text{DC}}_{\text{seq}}^2} \) is just countable choice, since \( \rho^0_n \) is just a singleton object of ‘length’ \( n \), and so \( \rho^0 \) is isomorphic to \( \text{nat} \). Setting \( A_n(n, x) := B_n(x) \) we obtain \( \text{AC} \) as defined in Section 4. In addition to this, Theorem 5.3 completely reduces to Theorem 4.4 for \( A(n) := \exists x B_n(x) \) once we eliminate \( \leq \). Therefore there is a direct correspondence between the realizers of the two theorems in this case.

For any function \( m \) satisfying \( n \notin \text{dom}(u) \rightarrow mn \notin \text{dom}(u) \), we can define a ‘generalised’ version of the Berardi-Bezem-Coquand functional as \( \text{BB}_m := \Psi_{(\emptyset, \emptyset, m)} \), which satisfies the defining equation

\[
\text{BB}_{(m)}^{\varepsilon, q}(u) = q(u \uplus \lambda n. \varepsilon_{mn, n}(\lambda x. \text{BB}_{(m)}^{\varepsilon, q}(u_{mn}^x))).
\]

Regardless of the choice of \( m \), this functional always gives a computational interpretation to countable choice. Now suppose that we move back into the more conventional setting of the double negation translation of countable choice, setting \( R = \perp \) and assuming \( B_n(x) \) is a negated formula, so that there exists a term \( h \) satisfying \( \forall n, x h \text{ mr } (\perp \rightarrow B_n(x)) \). Then if \( \phi \) satisfies

\[
\phi \text{ mr } \forall m((\exists x B_m(x) \rightarrow \perp) \rightarrow \perp)
\]
then $\varepsilon_{m,n}^\phi(p) := \rho \times \sigma \langle 0, \mu(h(\phi_m(p))) \rangle$ realizes the premise of $\emptyset$-$\text{DC}_\text{seq}^f$ and thus $\text{BBC}^\phi_{(m),1} := \text{BBC}^\phi_{(m)}$ which has defining equation

$$\text{BBC}^\phi_{(m),1}(u) = q(u \oplus \lambda n . \langle 0, h(\phi_{m,n}(\lambda x . \text{BBC}^\phi_{(m),1}(u_{m,n}))) \rangle).$$

realizes the negative translation of AC. In particular, setting $mmu = n$ we obtain

$$\text{BBC}^\phi_{(m)}(u) = q(u \oplus \lambda n . \langle 0, h(\phi_n(\lambda x . \text{BBC}^\phi_{(m)}(u_{n}))) \rangle)$$

which is just the BBC functional of [3] (more precisely, the variant of the Berardi-Bezem-Coquand functional for input with arbitrary domain considered in [3]).

5.2.2 Modified bar recursion ($\prec_n = \emptyset$)

Suppose that we retain the simplification $\prec_n = \emptyset$, but now allow $\prec$ to range over arbitrary decidable partial orders. Then we obtain a realizer which makes recursive calls over updates of its input, just like the BBC functional, but now $m$ must not only satisfy condition (iii) but also (iii):

$$u \prec i \rightarrow u \cup \{mmu\} \text{ is } \prec\text{-closed}$$

In this case, $\text{MBR}^\prec_{(\prec_n)} = \Psi^\prec_{(\prec_n,\emptyset,m)}$, which has defining equation

$$\text{MBR}^\prec_{(\prec_n)}(u) = q(u \oplus \lambda n . \varepsilon_{(\prec_n)}(\{mmu\},\{m\})(\lambda x . \text{MBR}^\prec_{(\prec_n)}(u_{m,n}))$$

forms a realizer for $\prec\text{-DC}_\text{seq}$. This can be viewed as a generalisation of modified bar recursion as first defined in [8]. All existing variants of MBR occur when $\prec$ is the usual ordering on nat, but MBR is perfectly well-defined in more unusual cases. For instance, suppose that we have a bijective encoding $c: \text{nat} \rightarrow \text{bool}^*$, and that

$$m \prec n := c(m) \text{ is a proper prefix of } c(n).$$

Then $\prec\text{-closed partial functions are precisely partial functions whose domain is a binary tree, and in this case there are many valid choices for } m, \text{ a canonical one being}$

$$mmu := n \text{ if } n \in \text{dom}(u) \text{ else } i \text{ where } c(i) \text{ is the least prefix of } c(n) \text{ whose prefixes are all in } \text{dom}(u).$$

This variant of MBR yields an intuitive realizer for dependent choice over binary trees.

Now, suppose that we do indeed have $\prec = \prec$. Then $\prec\text{-closed partial functions are either total or of the form } \hat{s} \text{ where } s \text{ is a finite sequence. Observing that in order to evaluate } \text{MBR}(\emptyset) \text{ we can restrict ourselves to input with finite domain we can redefine our realizer in this case as } \text{MBR}^{\prec}_{(\prec_n)}(s^\prec) = \text{MBR}^{\prec}_{(\prec_n)}(\hat{s}), \text{ setting } mmu := \mu i \leq n (i \notin \text{dom}(u)). \text{ Clearly such an } m \text{ satisfies (iii), and in particular } m\hat{s} = |s| \text{ for } \leq |s|. \text{ Therefore } \text{MBR}_1 \text{ has defining equation}$

$$\text{MBR}^\prec_{(\prec_n)}(s) = q(\hat{s} \oplus \lambda n . \varepsilon_{s,\{\alpha\}}(\alpha) ) (\lambda x . \text{MBR}^\prec_{(\prec_n)}(s \cdot x)))$$

This directly realizes $\prec\text{-DC}_\text{seq}$, which is isomorphic to

$$\forall s^\prec (\forall i < |s| A_i([s](i), s(i)) \rightarrow \exists x A_i([s](x, x)) \rightarrow \exists \alpha \forall n A_n([\alpha](n), \alpha(n)).$$

We can now easily redefine various concrete instances of MBR found in the literature which arise from setting $R = \sqcup$ and instantiating $A_i(s,x)$ by the correct formula. First, note that we immediately derive AC from $\prec\text{-DC}_\text{seq}$ by setting $A_n(s,x) := B_n(x)$, and a corresponding realizer for AC by defining

$$\text{MBR}^{\phi,h}_{\text{DC}} := \text{MBR}^{\phi,h}_{\text{DC}}(p) := \langle 0, \mu(h(\phi_{\text{DC}}(p))) \rangle, \text{ where } \phi \text{ and } h \text{ are as in Section 5.2.1.}\text{ This has defining equation}$

$$\text{MBR}^{\phi,h}_{\text{DC}}(s) = q(\hat{s} \oplus \lambda n . \langle 0, h(\phi_{\text{DC}}(\lambda x . \text{MBR}^{\phi,h}_{\text{DC}}(s \cdot x))) \rangle)$$

which is exactly the realizer of countable choice constructed in [6]. Note that the same realizer could have been constructed from $\text{BBC}^\phi_{(m),1}$ for suitable $m$.

Now suppose that $A_0(s,x) := B_0(x, x)$ and $A_n(s,x) := B_n(s_{[s]-1}, x)$ for $n > 0$. Then $\prec\text{-DC}_\text{seq}$ immediately implies the following, standard formulation of dependent choice:

$$\text{DC} : \forall n, y \exists x B_n(y, x) \rightarrow \forall x_0 \exists \alpha(0) = x_0 \land \forall n B_n(\alpha(n), \alpha(n + 1)).$$
The challenge for realizing DC is as follows: we must construct a realizer of $R$, given realizers

$$
\phi^{\text{nat} \rightarrow \rho \mapsto (\rho \times \sigma \rightarrow \tau) \mapsto \tau} \text{mr } \forall n, y (\exists x B_n(y, x) \rightarrow \bot) \rightarrow \bot)
$$

$$
Y^{(\rho \times \sigma) \mapsto \tau} \text{mr } \exists \alpha (\alpha(0) = x_0 \land \forall n B_n(\alpha(n), \alpha(n + 1)) \rightarrow \bot}
$$

and in addition assuming that $B$ is negated (thus guaranteeing the existence of a realizer $h$ of ex-falso quodlibet: $\forall n, y, x \text{ h mr } (\bot \rightarrow B_n(y, x))$). But in this case, we can easily define realizers of the premise of $\langle DC \rangle$ as

$$
e^{\text{nat}}_x (p) := (0, h(\phi_{|s|}(x_0 \ast n_0), (p))
$$

$$
q^Y(\alpha) := Y_{x_0}(\alpha) := Y((x_0 * \alpha_0, \alpha_1))
$$

and defining $MBR_3^{B,Y} := MBR_1^{x \mapsto q^Y}$ yields a realizer for DC satisfying

$$
MBR_3^{B,Y}(s) = Y_{x_0}(\hat{s} \oplus \lambda n \cdot (0, h(\phi_{|s|}(x_0 \ast n_0), (\lambda x . MBR_3^{B,Y}(s \ast x))))
$$

which is this time exactly the realizer of DC given in [6]. In an entirely analogous way, the bar recursive solution to the sequential variant of dependent choice considered in [20]:

$$
B(\langle \rangle) \rightarrow \forall x \mapsto (B(s) \rightarrow (\exists x B(s \ast x)) \rightarrow \exists \alpha \forall n B(\alpha(n))
$$

(9)

can be defined in terms of $MBR_1$, since [6] is easily implied by $\langle DC \rangle$ for $A_n(s, x) := B(s \ast x)$. Given realizers

$$
\phi^{\text{nat} \rightarrow \rho \mapsto (\rho \times \sigma \rightarrow \tau) \mapsto \tau} \text{mr } B(\langle \rangle)
$$

$$
Y^{(\rho \times \sigma) \mapsto \tau} \text{mr } \exists \alpha \forall n B(\alpha(n)) \rightarrow \bot
$$

and assuming the existence of a realizer $\forall s h \text{ mr } (R \rightarrow B(s))$, it is easy to see that

$$
e_{s,x}^{a_0,\rho}(p) := (0, h(\phi_{|s|}(a_0 \ast x_0), (p))
$$

$$
q_{s,Y}^{a_0} (\alpha) := Y_{a_0}(\alpha) := Y((a_0, G_0 * \alpha_1))
$$

realize the premise of $\langle DC \rangle$, and therefore the realizer we obtain is $MBR_3^{a_0,\rho,Y} := MBR_1^{x \mapsto q^{a_0}}$, which satisfies

$$
MBR_3^{a_0,\rho,Y}(s) = Y_{a_0}(\hat{s} \oplus \lambda n \cdot (0, h(\phi_{|s|}(a_0 \ast x_0), (\lambda x . MBR_3^{a_0,\rho,Y}(s \ast x))))
$$

which is exactly the term used to interpret dependent choice in [26] (and also extract an algorithm from Higman’s lemma in [24]).

5.2.3 Products of selection functions ($\text{mut} = n$, $\prec$ total)

We finally consider the case in which the functional $m$ is elimated by setting it as the identity function. Now, condition [3] is trivially satisfied, while [4] is reduced to $u \cup \{ k \mid k \leq u \}$ being $\prec$-closed whenever $u$ is. Clearly, the conditions are satisfied whenever $\langle \rangle \subseteq \prec u$, and in particular when $\leftarrow u \Rightarrow \langle$ the realizer simplifies to

$$
\Psi^{\prec \delta}(u) = q(u \oplus \lambda n \cdot e_{\langle \alpha \rangle}(\alpha)(\lambda x . \Psi^{\prec \delta}(u \oplus \langle \alpha \rangle(n \ast x)))
$$

Now, suppose that $\langle \rangle$ is a total order, and therefore constitutes an encoding of some countable ordinal $\xi$. It is always the case (for arbitrary parameters) by condition [5] that we can evaluate $\Psi(\langle \rangle)$ by restricting the input to being $\langle \rangle$-closed. However, when $\langle \rangle$ is total then the set of $\langle \rangle$-closed partial sequences is isomorphic to $\rho^\delta$, since if $u$ is $\langle \rangle$-closed then $\text{dom}(u) = \{ k \mid k \leq u \}$ where $n$ is the least element of the set of unordered elements of $u$. Let us suppose that $\langle \rangle$ comes equipped with a computable ordinal successor function $s : \text{nat} \rightarrow \text{nat}$ i.e. for each number $n$, $s(n)$ is the least number greater than $n$ with respect to $\prec$. Define $PS^{\prec \delta}(\langle \rangle) : \rho^\delta \rightarrow \rho^\text{nat}$ by $PS^{\prec \delta}(\langle \rangle) := \Psi^{\prec \delta}(\langle \rangle)$. Then $PS$ satisfies the equation

$$
PS^{\prec \delta}(\langle \rangle)(s) = \hat{s} \oplus \lambda n \cdot e_{\langle \alpha \rangle}(\alpha)(\lambda x . q(PS^{\prec \delta}(\langle \rangle)(\langle \alpha \rangle(n) \ast x)))
$$
where \( \{\alpha\}(n) \ast x : \rho_{\alpha(n)}^x \) is defined by
\[
(\{\alpha\}(n) \ast x)(m) := \begin{cases} 
\alpha(m) & \text{if } m < n \\
 x & \text{if } m = n.
\end{cases}
\]

This functional is a generalisation of the (implicitly well-founded) product of selection functions of Escardó and Oliva to arbitrary recursive ordinals, and by Corollary 5.4 is not only a realizer \(<\text{-DC}_{\text{seq}}\>\), but a direct realizer of the following transfinite \(J\)-shift principle:
\[
\begin{aligned}
\forall s(\forall i < |s| A_i(\{s\}(i), s(i)) \rightarrow (\exists x A_i(s, x) \rightarrow R) \rightarrow \exists x A_i(s, x)) \\
\rightarrow (\exists \alpha \forall n A_n(\{\alpha\}(n), \alpha(n)) \rightarrow R) \rightarrow \exists \alpha \forall n A_n(\{\alpha\}(n), \alpha(n)).
\end{aligned}
\]

For the particular case that \(\prec\) is the normal ordering on \(\text{nat}\) and \(A_n(s, x) := B_n(s \ast x)\) this becomes
\[
\begin{aligned}
\forall s^\prec (\forall i < |s| B_i([s][i + 1]) \rightarrow (\exists x B_i(s \ast x) \rightarrow R) \rightarrow \exists x B_i(s \ast x)) \\
\rightarrow (\exists \alpha \forall n B_n([\alpha](n + 1)) \rightarrow R) \rightarrow \exists \alpha \forall n B_n(\{\alpha\}(n + 1)).
\end{aligned}
\]

which is precisely the dependent \(J\)-shift of \([15]\), and our realizer becomes
\[
\text{PS}_{\prec,<}^\epsilon q(s) = \hat{s} \oplus \lambda n \cdot \epsilon_{\alpha(n)}(\lambda x \cdot q(\text{PS}_{\prec,<}^\epsilon q([\alpha](n) \ast x)))
\]
which is isomorphic to their dependent product of selection functions. Setting \(A_n(s, x) := B_n(x)\) as in Section 5.2.2 we see that the dependent \(J\)-shift implies the non-dependent \(J\)-shift:
\[
\forall n(\exists x B_n(x) \rightarrow R) \rightarrow \exists x B_n(x) \rightarrow (\exists \alpha \forall n B_n(\alpha(n)) \rightarrow R) \rightarrow \exists \alpha \forall n B_n(\alpha(n))
\]
and our realizer can be simplified to
\[
\text{PS}_{\prec,<}^\epsilon q(s) = \hat{s} \oplus \lambda n \cdot \epsilon_{\alpha(n)}(\lambda x \cdot q(\text{PS}_{\prec,<}^\epsilon q([\alpha](n) \ast x)))
\]
which is now isomorphic to the non-dependent product of selection functions of \([15]\).

**Summary**

We have demonstrated that the many variants of bar recursion used to interpret choice principles are just instances of the same basic combinatorial idea: more precisely computational interpretations of particular proofs of \(<\text{-DC}_{\text{seq}}\>\) by backward induction. Theorem 5.3 gives us a completely uniform framework in which to understand the variety of different realizers currently in the literature, and moreover in each case we are able to provide generalisations of these realizers to more complex orderings on \(\text{nat}\). We summarise all this in the table below (here \(m_1 nu := \mu i \leq n(i \notin \text{dom}(u))\) and \(m_2 nu = n\).

| \(m\) | \(\prec_u\) | \(\prec\) | \(A_n(s, x)\) | realizer |
|---|---|---|---|---|
| \(m_1\) | \(\emptyset\) | \(\emptyset\) | \(B_n(x)\) | Berardi-Bezem-Coquand functional \([4]\) |
| \(m_2\) | \(\emptyset\) | // \(\prec\) | \(B_n(x)\) | simple modified bar recursion \([6]\) |
| \(m_2\) | \(\emptyset\) | \(\prec\) | \(B_n(s[i-1], x)\) | dependent modified bar recursion \([6]\) |
| \(m_2\) | \(\emptyset\) | // \(\prec\) | \(B(s \ast x)\) | dependent modified bar recursion \([26]\) |
| \(m_1\) | \(\emptyset\) | // \(\prec\) | \(B_n(x)\) | simple product of selection functions \([12]\ [15]\) |
| \(m_1\) | \(\emptyset\) | // \(\prec\) | \(B_i(s \ast x)\) | dependent product of selection functions \([12]\ [15]\) |

However, these examples constitute only an extremely limited range of possibilities for \(\Psi_{(\prec_u, \prec)}\) based on very simple instantiations of its parameters. Theorem 5.3 gives us far more than just a unifying perspective for existing realizers: it gives us a very general recipe for devising new realizers for choice principles that can be tailored to the situation at hand. Thus, rather than simply giving a fixed interpretation of choice and extracting programs relative to this realizer, we have an additional level of flexibility which should allow us to extract much more efficient and meaningful programs.
6 Understanding the parametrised realizer

Having completed the main theoretic work of this paper, the purpose of this section is to give a somewhat informal graphical representation of the structure of our realizer and the proof of Theorem 5.3. In doing so we hope to give some insight into a potential semantic interpretation of our realizer.

Much work has been done in the last few decades on providing a computational interpretation of classical logic that can be understood on an intuitive level terms of learning - the basic idea being that realizers of classical principles typically carry out some kind of ‘learning procedure’ in order to construct an approximation to that principle.

In [3] a connection is suggested between realizers of negative translated formulas and winning strategies related to the Novikoff interpretation of classical formulas [11, 20]. In particular, an illuminating semantic interpretation of what we have called the Berardi-Bezem-Coquand functional is given, in which the functional represents a strategy for building an approximation of a choice functional which wins against any continuous opponent. We attempt to extend this interpretation to our generalised realizer of choice, and argue that the our parameters can be giving a clear meaning in this context.

Let us first very briefly recall the basic ideas of [3, 11] (the reader is referred to these papers for a proper treatment). In the Novikoff calculus, the formula \( \exists x \forall y A_0(x, y) \) is mapped to the propositional formula

\[
\bigvee_x \bigwedge_y A_0(x, y).
\]

The truth of such a formula is debated by two players \( \text{\textcursiveE} \text{loise} \) and \( \text{\textcursiveV} \text{belard} \), who support truth and falsity respectively. First, \( \text{\textcursiveE} \text{loise} \) selects some value \( x_0 \) for which she claims that \( \bigwedge_y A_0(x_0, y) \) is true, and \( \text{\textcursiveV} \text{belard} \) follows this by choosing some \( y_0 \) in an attempt to falsify \( A_0(x_0, y_0) \). The formula as a whole is intuitionistically valid if \( \text{\textcursiveE} \text{loise} \) has a winning strategy regardless of any choices made by \( \text{\textcursiveV} \text{belard} \).

Now, it is clear that such a correspondence does not work in the case of classical logic, since there are \( \Sigma_2 \)-formulas classically true but for which there is no effective strategy for \( \bigvee_x \bigwedge_y A_0(x, y) \). In this case, validity of \( \bigvee_x \bigwedge_y A_0(x, y) \) is interpreted as the existence of some \( x_0 \) such that

\[
A_0(x_0, y') \lor \bigvee_x \bigwedge_y A_0(x, y)
\]

is valid for all \( y' \). The idea here is that \( \text{\textcursiveE} \text{loise} \) picks a potential witness \( x_0 \), which is followed by an attempt at a counterexample \( y_0 \) from \( \text{\textcursiveV} \text{belard} \), and the game becomes \( A_0(x_0, y_0) \lor \bigvee_x \bigwedge_y A_0(x, y) \). In other words, either \( A_0(x_0, y_0) \) is true, in which case \( \text{\textcursiveE} \text{loise} \) wins, or it is false, and \( \text{\textcursiveE} \text{loise} \) can backtrack and start again, this time using falsity of \( A_0(x_0, y_0) \) as constructive information. In this way, \( \text{\textcursiveE} \text{loise} \) is allowed to ‘learn’ from \( \text{\textcursiveV} \text{belard} \)’s choices. Moreover, as demonstrated in [3], this notion of learning and backtracking is captured by the recursive functionals which realize the negative translation of the classical formula.

Let us now take this basic idea and consider how the truth of countable choice can be interpreted as a dialogue between \( \text{\textcursiveE} \text{loise} \) and \( \text{\textcursiveV} \text{belard} \) in which \( \text{\textcursiveE} \text{loise} \) eventually wins. Countable choice can be written as the disjunction

\[
\exists n \forall x \neg A_n(x) \lor \exists f \forall n. A_n(f(n)).
\]

We give this formula a rough interpretation along the lines of [3] as follows. \( \text{\textcursiveE} \text{loise} \) begins by attempting to realize the conclusion of \( \text{AC} \) with some default function \( f_0 = \lambda n.0 \) whose ‘domain’ of genuine constructive information is empty, and \( \text{\textcursiveV} \text{belard} \) responds by selecting some point \( m_0 \) such that this attempt fails i.e. we cannot provide a realizer for \( A_{m_0}(0) \). \( \text{\textcursiveE} \text{loise} \) responds by now attempting to falsify the premise at some \( m_0 \), to which \( \text{\textcursiveV} \text{belard} \) responds with a point \( x_0 \) and a realizer for \( A_{m_0}(x_0) \).

\( \text{\textcursiveE} \text{loise} \) has now been given some constructive information by \( \text{\textcursiveV} \text{belard} \), so she starts the whole process again, this time with the function \( f_1 := f_0[m_0 \mapsto x_0] \), which now has a domain of \( \{m_0\} \). This time, \( \text{\textcursiveV} \text{belard} \) picks a point \( n_1 \) at which \( f_1 \) fails. Either he picks \( n_1 = m_0 \) in which case he loses since by his own admission \( A_{m_1}(x_0) \) is true (see [3] for a formal translation of this logic into Novikoff strategies), or he chooses \( n_1 \notin \{m_0\} \). Then \( \text{\textcursiveE} \text{loise} \) responds with some \( m_1 \notin \{m_0\} \) which falsifies the premise of \( \text{AC} \), and again \( \text{\textcursiveV} \text{belard} \) responds with some \( x_1 \) and a realizer for \( A_{m_1}(x_1) \). \( \text{\textcursiveE} \text{loise} \) now updates her approximation again to some \( f_2 \) which includes this information, and continues as before. Roughly speaking, such a strategy should eventually result in success for \( \text{\textcursiveE} \text{loise} \) whenever \( \text{\textcursiveV} \text{belard} \) is ‘continuous’, because he will eventually be forced to pick \( n_1 \) in the domain of \( f_1 \).
We have been vague as to two details in this strategy: firstly how \( \exists \text{loise} \) decides on which \( m_i \) to choose in light of \( \forall \text{belard}'s \) original choice \( n_i \), and secondly in how she chooses to update her approximation each time. In the first instance it is clear that if \( \forall \text{belard} \) has a 'good' choice for \( n_i \) which is not in the domain \( f_i \) then \( \exists \text{loise} \) must also respond with \( m_i \) not in the domain of \( f_i \) if she has any chance of falsifying the premise of AC. Then, when it comes to updating \( f_i \) with the information \((m_i, x_i)\), she could either just add this directly to \( f_i \) and define \( f_{i+1} := f_i[m_i \rightarrow x_i] \), or she could potentially erase some of the existing elements in the domain of \( f_i \).

The most natural choice is of course to pick \( m_i = n_i \) each time and update directly without any erasing, and this is precisely the strategy given in [3]. However, in the case of dependent choice where the elements of the choice sequences are related in some way, it is crucial for \( \exists \text{loise} \) to update her approximation in a manner which is coherent with the underlying dependency required for the choice sequence.

Let us now move on to our realizer for the negative translation of countable dependent choice (in the form of the \( \lnot\text{DC}^\text{seq} \)). We start at \( \Phi \) and denote that \( \exists \text{loise} \) has already computed a partial realizer. For simplicity we now work in a concrete realizability setting with \( R = \bot \) and \( \varepsilon_{s,r}(p) = (0, h(\phi_s(p))) \) where

\[
\phi \equiv \forall s (\exists x A_{[s]}(s, x) \rightarrow \bot) \rightarrow \bot.
\]

and \( \forall n, r, x (h \equiv \forall (r \rightarrow A_n(r, x))) \), so in particular \( \varepsilon \) realizes the premise of \( \lnot\text{DC}^\text{seq} \).

Let us now run through the diagram, step by step. Each box roughly represents a stage in a game between \( \exists \text{loise} \) and \( \forall \text{belard} \), and an arrow represents a reverse implication in the proof of Theorem 5.3.

We start at \( A \) with the assumption that \( \exists \text{loise} \) has already computed a partial realizer \( u \) that is correct wherever it’s defined. At step \( B \), \( \exists \text{loise} \) plays an approximation \( (\alpha_u)_0 := u_0 \oplus 0 \), and in response, \( \forall \text{belard} \) selects some \( n \) and challenges \( \exists \text{loise} \) to realise \( A_n((\alpha_u)_0(n), \alpha_u(n)) \). If \( n \in \text{dom}(u) \) then \( \exists \text{loise} \) wins, so we assume the contrary and move on to step \( C \). \( \exists \text{loise} \) responds to the challenge in the next
step $D$ by claiming that the premise is false at point $mnu$. \vbelard is now forced to produce $x$ such that $x_1 m r A_{mnu}(\{(\alpha_u)_0\}(mnu), x_0)$ - if he fails then $\exists\loise$ wins, and if he succeeds then $\exists\loise$ is given constructive information and takes this as an assumption.

We now come to the subtle part: the manner in which $\exists\loise$ updates $(\alpha_u)_0$ to reflect this new information and repeat the loop. From a proof theoretic perspective, $E'$ is implied by $A$ and $F'$ by the cut rule. We can interpret this semantically as follows: $\exists\loise$ forms the updated function $\alpha_0[mnu \mapsto x_0]$, but states that if in future $\forall\belard$ queries this realizer for any $k \prec u n$ with $k \notin \text{dom}(u)$, she will ignore any subsequent information received and revert to stage $C$ as if $\forall\belard$ had chosen $k$ instead of $n$. This is reflected in the definition of $\Psi$, since it makes a recursive call on the partial function $u \odot t[n, u]^{mnu}$, but the fact that this is a partial realizer for $k \prec u n$ relies on nested recursive calls of the form $\Psi(u \odot t[k, u]^{nku})$, which in particular forgets the value of $x$.

Therefore at stage $E$ there are two possibilities. Either at some point in the future $\forall\belard$ does query $k \prec n$ for $k \notin \text{dom}(u)$, in which case all subsequent information is deemed irrelevant and we make an auxiliary $\prec$-recursive loop back to $C$, or $\forall\belard$ never queries $k \prec n$, in which case we can treat $u \odot t[n, u]^{mnu}$ as a partial realizer and make a backward recursive loop back to stage $A$. In this way, the game corresponding to dependent choice can be seen as a path through Figure 1. By the combination of $\prec$-induction and backward induction, using the fact that $\prec$ is well-founded and $\forall\belard$'s choice of $n$ at step $B$ is based on a continuous strategy, the proof of Theorem 5.3 essentially says that there is no infinite path starting from 0, and therefore $\exists\loise$ has a winning strategy in the game as a whole.

Now let us think about our concrete examples. For the Berardi-Bezem-Coquand functional we have $t[u, n] = \emptyset$ and so the `forgetful' subloop is completely avoided. In this case $\exists\loise$ simply responds to $\forall\belard$'s choice of $n$ directly, and so when $\forall\belard$ plays $(1, x_0), (4, x_1), (3, x_2)$ we get the following sequence of updates:

\[
[0, 0, 0, 0, 0] \mapsto [0, x_0, 0, 0, 0] \mapsto [0, x_0, 0, x_0, x_1] \mapsto [0, x_0, 0, x_2, x_1] \mapsto \ldots
\]

This semantic interpretation of the BBC-functional is of course completely analogous to the one given in [3]. For modified bar recursion, whenever $\forall\belard$ makes a sensible choice $n \notin \text{dom}(u)$, $\exists\loise$ switches instead the least element not already in the domain. Again we avoid the forgetful subloop, but this time $\exists\loise$'s updates are done in sequence - the same three choices from $\forall\belard$ results in the following response:

\[
[0, 0, 0, 0, 0] \mapsto [x_0, 0, 0, 0, 0] \mapsto [x_0, x_1, 0, 0, 0] \mapsto [x_0, x_1, x_2, 0, 0] \mapsto \ldots
\]

Finally, for the product of selection functions, $\exists\loise$ never changes $\forall\belard$'s choice of $n$, but now assumes a policy of forgetting everything above the point being updated. This time the result is

\[
[0, 0, 0, 0, 0] \mapsto [0, x_0, 0, 0, x_1] \mapsto [0, x_0, 0, x_2, 0] \mapsto \ldots
\]

the point $x_1$ being forgotten as a result of $\forall\belard$ choosing $3 < 4$ as his third move.

It now becomes clear why the strategy related to the BBC-functional fails for dependent choice. It is assumed that $\forall\belard$ always picks $x_i$ such that $A_m([u \odot 0](m), x_i)$ holds, and so e.g. we have $A_4([0, x_0, 0], x_1)$. However, if ever in the future he gives $\exists\loise$ some new information $x_{i+j}$ for $k < m$, then $x_i$ is no longer valid - for instance we would require $A_4([0, x_0, 0, x_2], x_1)$ to hold, which is not necessarily true.

Modified bar recursion and the product of selection functions represent two methods of overcoming this. For the former, $\exists\loise$ ensures that $\forall\belard$ always gives her constructive information in sequence, while for the latter she is happy to take information for any point $\forall\belard$ chooses, but whenever updating she erases everything above, relying on future moves to regain this information.

## 7 Concluding remarks

There are several directions in which the work presented here could be developed. We have already demonstrated that Theorem 5.3 provides a uniform soundness proof through which most of the existing variants of backward recursion used to interpret dependent choice can be derived. The most obvious next step would be to explore the use of new variants of bar recursion which arise from giving more interesting values to the parameters of $\Psi(m, \prec, \odot)$.

Take, for example, the following classical statement:
For any function \( f : \text{bool}^* \to \text{nat} \), there exists a function \( g : \text{bool}^* \to \text{nat} \) such that for any branch \( s : \text{bool}^* \) we have \( f(g(s)) \leq f(t) \) whenever \( t \) has \( s \) as a prefix.

This is a direct consequence of the axiom of countable choice, and so any instantiations of \((m, \prec)\) will yield a realizer capable of building an approximation to the choice function \( g \). However, it would seem most sensible to choose the parameters so that the recursion is carried out over the natural tree structure \( s \prec t \) iff \( s \) is a prefix of \( t \) (relative to some encoding of \( \text{bool}^* \) into \( \text{nat} \)) - using, for example, the variant of modified bar recursion sketched out the beginning of Section 5.2.2. Then if an approximation to \( g \) is defined at \( s \) then this information could be used to extend the approximation to extensions \( t \) of \( s \).

It would be interesting to examine in general how such realizers, tailored to the situation at hand, compare to those built from the existing forms of bar recursion. One would expect an advantage in terms of both algorithmic efficiency and the syntactic expressiveness of the extracted program.

Another interesting application of our parametrised form of bar recursion would be to extend the work of Escardó and Oliva on Nash-equilibria of unbounded games to the transfinite case. In [14, 15] it is shown that the infinite products of selection functions corresponding to the functional \( \mathcal{PS}_{\langle \rangle} \) defined in Section 5.2.3 computes optimal strategies in infinite sequential games over the natural numbers. However, \( \mathcal{PS}_{\langle \rangle} \) is well-defined for any computable well ordering on \( \text{nat} \), and in particular it is not too difficult to show that in these cases Spector’s equations:

\[
\alpha(n) = \varepsilon_{\{\alpha\}(n)}(p_n) \\
p_n(\varepsilon_{\{\alpha\}(n)}(p_n)) = q(\alpha)
\]

can be solved in \( \varepsilon \) and \( q \) for arbitrary \( \prec \) by setting \( \alpha = \mathcal{PS}^{\prec,q}(\emptyset) \) and \( p_n(x) := q(\mathcal{PS}^{\prec,q}(\{\alpha\}(n)\star x)) \), entirely analogously to the normal ordering on \( \text{nat} \). However, it would be useful to formalise this properly and to investigate whether there are any interesting applications of higher-type transfinite games.

Finally, to the author’s knowledge all known realizability interpretations of choice are restricted to countable or dependent choice principles, and it is not known how to extend these to choice over function spaces, for example:

\[
\forall f : \text{nat} \to \text{nat} \exists x \forall^\alpha B_f(x) \to \exists f : \text{nat} \to \text{nat} \forall x \forall^\alpha f B_f(F(f)).
\]

One possibility would be to build an approximation to \( F \) over some countable basis \((c_i)_{i \in \text{nat}}\) for the space \( \text{nat} \to \text{nat} \). However, in this case updates to the approximation must be made in a coherent way, since the values of two distinct elements \( c_1 \) and \( c_2 \) must be compatible with their intersection. Therefore the ideas behind our parametrised realizer could be helpful here, the aim being to assign an ordering to the basis and make sure that the updates respect this ordering. However, in the author’s opinion it is likely that some additional ingenuity would be required here, as the challenge posed by giving a computational interpretation non-countable choice seems to be a significant one.

References

[1] F. Aschieri. Learning, Realizability and Games in Classical Arithmetic. PhD thesis, Universita degli Studi di Torino and Queen Mary, University of London, 2011.

[2] J. Avigad. Update procedures and the 1-consistency of arithmetic. Mathematical Logic Quarterly, 48(1):3–13, 2002.

[3] S. Berardi, M. Bezem, and T. Coquand. On the computational content of the axiom of choice. Journal of Symbolic Logic, 63(2):600–622, 1998.

[4] U. Berger. The Berardi-Bezem-Coquand functional in a domain-theoretic setting. Unpublished results, available from author’s webpage at http://www.cs.swan.ac.uk/~csulrich/recent-papers.html, 2002.

[5] U. Berger. A computational interpretation of open induction. In F. Titsworth, editor, Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science, pages 326–334. IEEE Computer Society, 2004.

[6] U. Berger and P. Oliva. Modified bar recursion and classical dependent choice. Lecture Notes in Logic, 20:89–107, 2005.
[7] U. Berger and P. Oliva. Modified bar recursion. *Mathematical Structures in Computer Science*, 16(2):163–183, 2006.

[8] U. Berger and H. Schwichtenberg. Program extraction from classical proofs. In D. Leivant, editor, *Logic and Computational Complexity workshop (LCC’94)*, volume 960 of *Lecture Notes in Computer Science*, pages 77–97. 1995.

[9] M. Bezem. Strongly majorizable functionals of finite type: A model for bar recursion containing discontinuous functionals. *Journal of Symbolic Logic*, 50:652–660, 1985.

[10] V. Blot and C. Riba. On bar recursion and choice in a classical setting. In *Proceedings of APLAS 2013*, volume 8301 of *Lecture Notes in Computer Science*, pages 349–364, 2013.

[11] T. Coquand. An analysis of Ramsey’s theorem. *Information and Computation*, 110(2):297–304, 1994.

[12] M. Escardó and P. Oliva. Computational interpretation of analysis via products of selection functions. In *Proceedings of CiE 2010*, volume 6158 of *LNCS*, pages 141–150. 2010.

[13] M. Escardó and P. Oliva. Selection functions, bar recursion and backward induction. *Mathematical Structures in Computer Science*, 20(2):127–168, 2010.

[14] M. Escardó and P. Oliva. Sequential games and optimal strategies. *Royal Society Proceedings A*, 467:1519–1545, 2011.

[15] M. Escardó and P. Oliva. The Peirce translation. *Annals of Pure and Applied Logic*, 163(6):681–692, 2012.

[16] S. C. Kleene. Countable functionals. In A. Heyting, editor, *Constructivity in Mathematics*, pages 81–100. North-Holland, Amsterdam, 1959.

[17] U. Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Monographs in Mathematics. Springer, 2008.

[18] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland, Amsterdam, 1959.

[19] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Proceedings of the Cambridge Philosophical Society*, 59:833–835, 1963.

[20] P.S. Novikoff. On the consistency of certain logical calculi. *Matematicheskij Sbornik*, 12(54):230–260, 1943.

[21] P. Oliva and T. Powell. On Spector’s bar recursion. *Mathematical Logic Quarterly*, 58:356–365, 2012.

[22] P. Oliva and T. Powell. Bar recursion over finite partial functions. 2014. Preprint, available from http://arxiv.org/abs/1410.6361.

[23] P. Oliva and T. Powell. A constructive interpretation of Ramsey’s theorem via the product of selection functions. To appear: *Mathematical Structures in Computer Science*, 2015. Preprint available at http://arxiv.org/abs/1204.5631.

[24] J.-C. Raoult. Proving open properties by induction. *Information Processing Letters*, 29:19–23, 1988.

[25] M. Seisenberger. *On the Constructive Content of Proofs*. PhD thesis, Ludwig Maximilians Universität München, 2003.

[26] M. Seisenberger. Programs from proofs using classical dependent choice. *Annals of Pure and Applied Logic*, 153(1-3):97–110, 2008.
[27] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In F. D. E. Dekker, editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island, 1962.

[28] T. Streicher. A classical realizability model arising from a stable model of untyped lambda calculus. Preprint, available from http://arxiv.org/abs/1407.1547, 2014.

[29] A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer, Berlin, 1973.