Extracting Non-Deterministic Concurrent Programs

Ulrich Berger

Swansea University, U.K.

Abstract

We introduce an extension of intuitionistic fixed point logic by a modal operator facilitating the extraction of non-deterministic concurrent programs from proofs. We apply this extension to program extraction in computable analysis, more precisely, to computing with Tsuiki's infinite Gray code for real numbers.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic

Keywords and phrases Proof theory, realizability, program extraction, non-determinism, concurrency, computable analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.26

1 Introduction

The fact that proofs in constructive systems carry computational content is known as the Brouwer-Heyting-Kolmogorov interpretation or the Curry-Howard correspondence. It is the origin of various methods to automatically extract certified programs from formal proofs which are implemented in proof systems such as Nuprl [8], PX [9], Coq [12], Isabelle [5], Agda [7], Minlog [3]. The extracted programs are usually functional; other programming paradigms, such as non-determinism or concurrency, are hardly covered by this methodology. This may be considered a weakness of program *extraction* compared with existing program *verification* techniques which *do* cover these programming paradigms. This paper aims to narrow the gap between program extraction and verification by introducing *Concurrent Fixed Point Logic* (CFP), an intuitionistic theory of inductive and coinductive definitions extended by a modal operator enabling the extraction of non-deterministic concurrent programs.

The development of CFP was triggered by an example from computable analysis, *Tsuiki's infinite Gray code* for real numbers [14], which encodes a real number $x \in \mathbb{I} = [-1, 1]$ [1] by the itinerary of x along the *tent map*

$$\mathrm{t} : \mathbb{I} \to \mathbb{I}, \quad \mathrm{t}(x) = 1 - 2|x|.$$

More precisely, x is encoded by the stream $a_0 : a_1 : a_2 : \ldots$ where the head of the stream, a_0, equals 0, 1 or $\perp$ (= undefined) depending on whether x is less, greater, or equal to 0, and the tail of the stream, $a_1 : a_2 : \ldots$, encodes $\mathrm{t}(x)$. Since $\mathrm{t}(0) = 1$ and $\mathrm{t}(1) = \mathrm{t}(-1) = -1$, at most one a_i can be undefined, and in that case $a_{i+1} = 1$ and $a_k = 0$ for all $k > i + 1$.

Infinite Gray code stands out from other encodings of real numbers by the fact that it is at the same time *unique* (every real number in $\mathbb{I}$ has exactly one Gray code) and *computable*

[1] Tsuiki considers reals in the interval $[0, 1]$, but we find it convenient to work with an interval that is symmetric around 0.

(it is computably equivalent to admissible encodings such as the Cauchy- or the signed digit representation). In contrast, other computable representations of the reals are highly redundant. For example, a signed digit representation of $x \in \mathbb{I}$ is any stream $d_0 : d_1 : d_2 : \ldots$ of signed digits $d_i \in \mathrm{SD} = \{-1, 0, 1\}$ such that $x \in \mathbb{I}_{d_0} := [d_0/2 - 1/2, d_0/2 + 1/2]$ and $d_1 : d_2 : \ldots$ is a signed digit representation of $2x - d_0$. It is easy to see that every non-dyadic real in $\mathbb{I}$ has continuum many signed digit representations. The infinite Gray code's unlikely combination of uniqueness and computability is, of course, only possible because it is not total: dyadic rationals in $]-1, 1[$ have an infinite Gray code with one undefined digit.

Tsuiki defines the following function transforming infinite Gray code into signed digit representation (`a0:a1:...:an:s` stands for a stream that begins with the digits `a0:a1:...:an` and continues with the stream `s`; we also use the function `swap 0 = 1; swap 1 = 0`):

```
gtos (0:s)     = -1 : gtos s
gtos (1:b:s)   =  1 : gtos (swap b : s)
gtos (a:1:c:s) =  0 : gtos (a : swap c : s)
```

Since the pattern `a:1:c:s` in the third line overlaps with those in the first and second line, this definition cannot be executed in a deterministic functional language, but should rather be viewed as a system of rewrite rules from left to right. Tsuiki introduced (and implemented in Prolog) two-head Turing machines that are able to execute definitions as the one above: Initially, the machine's first head reads the first input digit and its second head reads the second input digit. If the computation of the first input digit terminates first, the first or second line fires, if the computation of the second input digit terminates first, the third line fires and the second head moves one position to the right (while the first head continues to wait for the computation of the first input digit to terminate, which may or may not happen eventually). This is an example of a non-deterministic concurrent computation with potentially incompatible results of the different threads. One can show that no continuous function can translate Gray code into signed digit representation. The closest to infinite Gray code one seems to get with traditional program extraction is a program that works on so-called pre Gray code, i.e., streams representing constructions of infinite Gray code [4].

In this paper, we extend the method of program extraction developed in [1] in order to be able to extract concurrent programs such as `gtos` above from proofs of their specification. Our basic formal system is essentially the one considered in [1] and is called IFP in this paper (Intuitionistic Fixed Point Logic). It comprises intuitionistic first-order logic with inductive and coinductive definitions and a realizability interpretation (Sect. 2). An important feature of program extraction in IFP is the fact that proofs can be carried out in any non-computational theory (Sect. 2). The correctness of the extracted programs will then be proven in the same theory. This makes it possible to extract programs from proofs in an abstract axiomatic setting.

Our leading example is the structure $\mathbb{R}$ of the real numbers with 0, 1, addition and multiplication, which we specify in IFP by the disjunction-free axioms of Archimedean ordered fields (replacing $\vee$ by $\neg \wedge \neg$ if required). The set of natural numbers $\mathbb{N}$ can be defined in IFP as the least subset of the reals that contains 0 and is closed under the $+1$ function. Hence IFP includes Heyting Arithmetic. We define predicates C and G as the largest predicates on $\mathbb{I}$ satisfying

$$\begin{aligned} \mathrm{C}(x) &\leftrightarrow \bigvee_{d \in \mathrm{SD}} x \in \mathbb{I}_d \wedge \mathrm{C}(2x - d) \\ \mathrm{G}(x) &\leftrightarrow (x \neq 0 \to x < 0 \vee x > 0) \wedge \mathrm{G}(\mathrm{t}(x)) \end{aligned}$$

The signed digit representations of x will be the realizers of $\mathrm{C}(x)$, and the Gray code of x will be a realizer of $\mathrm{G}(x)$. Hence, in order to extract a program that translates Gray code into signed digit representation, we may attempt to prove $\mathrm{G} \subseteq \mathrm{C}$. Since there can be no deterministic program accomplishing the transformation, this proof cannot be carried out in IFP alone, but some extra principle is needed. The following *Disjunction Principle*

$$\text{(DP)} \quad (A \overset{P}{\vee} B) \wedge (P \overset{Q}{\vee} C) \to (A \vee B \vee C)$$

suffices, where $A \overset{P}{\vee} B$ is shorthand for $(P \to A \vee B) \wedge (\neg P \to A \wedge B)$ and P, Q, A, B, C range over non-computational formulas. Using (DP) (with $P := x \neq 0$, $Q := x \neq 0$, $A := x \in \mathbb{I}_{-1}$, $B := x \in \mathbb{I}_1$, and $C := x \in \mathbb{I}_0$), we show $\mathrm{G} \subseteq \mathrm{C}$ (Theorem 12). In order to extract a program from the proof of $\mathrm{G} \subseteq C$ one needs a realizer of (DP) which, unfortunately, does not exist in IFP. To solve this problem, we embed, in Sect. 5, IFP into Concurrent Fixed Point Logic (CFP), where (DP) *is* realizable (Lemma 30). CFP extends the language of IFP by a modal operator S and extends programs by families $\mathrm{Fam}\,\varphi$ of programs $\varphi(i)$ (where i ranges over a countable index set $\mathcal{I}$) that are to be executed non-deterministically and concurrently. This is expressed by an operational semantics (Sect. 6) that allows $\mathrm{Fam}\,\varphi$ to reduce to $\varphi(i)$ for every $i \in \mathcal{I}$. A family $\mathrm{Fam}\,\varphi$ realizes a formula of the form $\mathrm{S}(A)$, where A is computational, if (i) $\varphi(i)$ yields a result for at least one $i \in \mathcal{I}$ and (ii) for any $i \in \mathcal{I}$, if $\varphi(i)$ yields a result b, then b realizes A (Sect. 5). The proof rules for CFP given in Sect. 5 endow the modality S with the structure of a strong monad which can be seen as a proof-theoretic analogue of Moggi's monadic computational lambda-calculus [10]. With some coding effort the countably infinite non-determinism represented by the construct $\mathrm{Fam}\,\varphi$ can be simulated by binary non-deterministic choice as considered, for example, in [6]. The denotational semantics introduced in Sect 6 could be simplified accordingly. In order not to distract the readers attention from the main issues of this paper, we refrain from carrying out these simplifications.

There exists a rich literature on modelling and verifying non-deterministic and concurrent programs (e.g. [13, 11, 6] and many others). The novelty of our work lies in the fact that we extract these programs from ordinary mathematical proofs together with a formal certificate of their correctness.

2 Intuitionistic Fixed Point Logic

We briefly recall (and slightly improve) the system IFP of Intuitionistic Fixed Point Logic and its realizability interpretation as defined in [1].

IFP is intuitionistic first-order predicate logic with inductive and coinductive definitions given as least and greatest fixed points of strictly positive predicate transformers. The *formulas* of IFP, are $P(\vec{t})$, $X(\vec{t})$, $A \wedge B$, $A \vee B$, $A \to B$, $\forall x\, A$, $\exists x\, A$, $(\mu\,\Phi)(\vec{t})$ (inductive definitions) and $(\nu\,\Phi)(\vec{t})$ (coinductive definitions), where P ranges over predicate constants, X ranges over predicate variables, $\vec{t}$ ranges over tuples of first-order terms of a given signature, x ranges over object variables and Φ ranges over strictly positive predicate transformers. The latter are of the form $\lambda X \lambda \vec{x}\,.\, A$ where $\vec{x}$ and $\vec{t}$ have the same lengths and A is strictly positive in X i.e. X does not occur free in any premise of a subformula of A which is an implication [2]. Predicate constants and variables have fixed arities. We assume that there is a 0-ary predicate constant $\bot$ for falsity and an equality predicate.

A *predicate* is either a predicate constant P, a predicate variable X, an abstraction $\lambda \vec{x}.A$, an inductive predicate $\mu\,\Phi$, or a coinductive predicate $\nu\,\Phi$. The application, $\mathcal{P}(\vec{t})$, of

[2] In [2] it is shown that this condition can be weakened to provable monotonicity.

a predicate $\mathcal{P}$ to a list of terms $\vec{t}$ is a primitive syntactic construct, except when $\mathcal{P}$ is a an abstraction, $\lambda\vec{x}\,A$, in which case $\mathcal{P}(\vec{t})$ stands for $A[\vec{t}/\vec{x}]$. We will use abbreviations such as $\mathcal{P} \subseteq \mathcal{Q}$ for $\forall\vec{x}\,.\,\mathcal{P}(\vec{x}) \to \mathcal{Q}(\vec{x})$ and will sometimes write $\{x \mid A\}$ for $\lambda x\,A$, etc. We will also write $\mathcal{P}(\vec{x}) \stackrel{\mu}{=} A[\mathcal{P}/X]$ for $\mathcal{P} = \mu\,\lambda X \lambda\vec{x}\,A$ and similarly for ν.

The *proof rules* of IFP are the usual ones of intuitionistic predicate calculus with equality augmented by rules expressing that $\mu\,\Phi$ and $\nu\,\Phi$ are the least and greatest fixed points of the operator Φ (as is well-known, the fixed point property can be replaced by inclusions):

$$\frac{}{\Gamma \vdash \Phi(\mu\,\Phi) \subseteq \mu\,\Phi}\ \text{Closure} \qquad \frac{\Gamma \vdash \Phi(\mathcal{P}) \subseteq \mathcal{P}}{\Gamma \vdash \mu\,\Phi \subseteq \mathcal{P}}\ \text{Induction}$$

$$\frac{}{\Gamma \vdash \nu\,\Phi \subseteq \Phi(\nu\,\Phi)}\ \text{Coclosure} \qquad \frac{\Gamma \vdash \mathcal{P} \subseteq \Phi(\mathcal{P})}{\Gamma \vdash \mathcal{P} \subseteq \nu\,\Phi}\ \text{Coinduction}$$

Realizability is formalized in an extension RIFP of IFP by an extra sort of realizers and *program terms* (*programs for short*) of this new sort. Programs are untyped λ-terms with pairing, injections and recursion, more precisely, variables $a, b, c, d, e, f, g, \ldots$, the constant nil, and the composite terms $\langle M, N\rangle$, $\mathrm{inl}(M)$, $\mathrm{inr}(M)$, $\lambda a.M$, $\pi_i(M)$ $(i = 1, 2)$, case M of $\{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\}$, $(M\,N)$, $\mathrm{rec}\,a\,.\,M$. The free variables of a program are defined as usual (the constructs λa, $\mathrm{rec}\,a$ and $\mathrm{inl}(a) \to$, $\mathrm{inr}(a) \to$ in a case term bind the variable a). In order to keep programs readable, we will use pattern matching in a slightly liberal way by allowing wildcards, nested patterns and possibly omitting patterns in which case realizers matching an omitted pattern are mapped to the default value nil. For example, case M of $\{\mathrm{inl}(\mathrm{inr}(\_)) \to N\}$ stands for the nested case analysis case M of $\{\mathrm{inl}(a) \to$ case a of $\{\mathrm{inl}(a_0) \to \mathrm{nil}\,;\,\mathrm{inr}(a_1) \to N\}\,;\,\mathrm{inr}(b) \to \mathrm{nil}\}$ where a and a_1 are not free in N. Closed programs built from nil by pairing $\langle\cdot,\cdot\rangle$ and the injections $\mathrm{inl}(\cdot)$, $\mathrm{inr}(\cdot)$ are called *data*. All axioms and rules for IFP, including closure, induction, coclosure and coinduction and the rules for equality, are extended to RIFP. In addition, we add the equations

$$\begin{aligned}
\pi_i(\langle M_1, M_2\rangle) &= M_i \qquad (i = 1, 2)\\
\text{case } \mathrm{inl}(M) \text{ of } \{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\} &= L[M/a]\\
\text{case } \mathrm{inr}(M) \text{ of } \{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\} &= R[M/b]\\
(\lambda a.M)N &= M[N/a]\\
\mathrm{rec}\,a\,.\,M &= M[\mathrm{rec}\,a\,.\,M/a]
\end{aligned}$$

We define simultaneously *non-computational* and *faithful* formulas (abbreviated *nc* and *ff*). nc formulas without free predicate variables will be called *ncc* formulas.

- The class of nc formulas contains all atomic formulas (i.e. $P(\vec{t})$ and $X(\vec{t})$) and is closed under all logical operators except disjunction (but including inductive and coinductive definitions). In addition, if A is ff and B is nc, then $A \to B$ is nc.
- The class of ff formulas contains all ncc formulas and is closed under all logical operators except implication, universal quantification and coinductive definitions.

Note that all disjunction-free formulas without free predicate variables (which is the class of formulas called non-computational in [1]) are ncc.

Realizability assigns to every IFP-formula A a unary RIFP-predicate $\mathbf{r}(A)$. Intuitively, the RIFP-formula $\mathbf{r}(A)(a)$, which we will usually write $a\,\mathbf{r}\,A$, states that a "realizes" A. The definition of $a\,\mathbf{r}\,A$ is relative to a fixed one-to-one mapping from IFP-predicate variables X to RIFP-predicate variables $\tilde{X}$ with one extra argument place of the new sort. If the formula A has the free predicate variables $X_1, \ldots, X_n$, then the formula $a\,\mathbf{r}\,A$ has the free predicate

variables $\tilde{X}_1, \ldots, \tilde{X}_n$. For $\Phi = \lambda X \lambda \vec{x}\, A$ we set $\mathbf{r}(\Phi) = \lambda \tilde{X} \lambda (b, \vec{x})\, b\, \mathbf{r}\, A$.

$$
\begin{array}{rrcl}
 & a\,\mathbf{r}\,X(\vec{t}) & = & \tilde{X}(a,\vec{t}) \\
 & a\,\mathbf{r}\,P(\vec{t}) & = & P(\vec{t}) \\
a\,\mathbf{r}\,(A \wedge B) = a\,\mathbf{r}\,(B \wedge A) & & = & (a\,\mathbf{r}\,A) \wedge B \quad \text{if } B \text{ is ncc} \\
 & b\,\mathbf{r}\,(A \to B) & = & A \to b\,\mathbf{r}\,B \quad \text{if } A \text{ is ncc} \\
\text{Otherwise} & c\,\mathbf{r}\,(A \wedge B) & = & \pi_1(c)\,\mathbf{r}\,A \wedge \pi_2(c)\,\mathbf{r}\,B \\
 & c\,\mathbf{r}\,(A \vee B) & = & \exists a\,(c = \mathrm{inl}(a) \wedge a\,\mathbf{r}\,A) \vee \exists b\,(c = \mathrm{inr}(b) \wedge b\,\mathbf{r}\,B) \\
 & f\,\mathbf{r}\,(A \to B) & = & \forall a\,(a\,\mathbf{r}\,A \to (f\,a)\,\mathbf{r}\,B) \\
 & a\,\mathbf{r}\,\diamond x\,A & = & \diamond x\,(a\,\mathbf{r}\,A) \quad (\diamond = \forall, \exists) \\
 & a\,\mathbf{r}\,(\diamond\,\Phi)(\vec{t}) & = & (\diamond\,\mathbf{r}(\Phi))(a,\vec{t}) \quad (\diamond = \mu, \nu)
\end{array}
$$

▶ **Lemma 1.** *If A is an ncc formula, then $a\,\mathbf{r}\,A$ is equivalent to A, provably in* RIFP.

Proof. See Appendix. ◀

▶ **Theorem 2** (Soundness)**.** *From a proof in* IFP *of $\Delta, \Gamma \vdash A$, where Δ consists of ncc formulas, one can extract a program term M such that* RIFP *proves $\Delta, \vec{a}\,\mathbf{r}\,\Gamma \vdash (M\,\vec{a})\,\mathbf{r}\,A$.*

Proof. The proof is as in [1]. ◀

In the following we will apply the Soundness Theorem with Δ being the axioms of Archimedean ordered fields, written as ncc formulas, and Γ a set of formulas for which we have (or construct) programs realizing them. We are free to add stability, $\neg\neg A \to A$, for ncc formulas A to Δ, since, clearly, $\neg\neg A \to A$ is ncc if A is.

3 Cauchy and signed digit representation of real numbers

In this section we introduce real numbers in IFP. We define predicates A and C which correspond, via realizability, to the (fast) Cauchy representation and the signed digit representations, respectively, and prove their equivalence.

We let $(\mathbb{R}, 0, 1, +, \cdot, \leq)$ be the structure of real numbers, of which we will only use that it is an Archimedean ordered field. In order for the axioms to be ncc we write the Dichotomy axiom as $\neg x < y \wedge \neg y < x \to x = y$. As explained at the end of the previous section, we may assume stability of ncc and hence atomic formulas, that is, $\neg\neg x < y \to x < y$, etc. The Archimedean principle, written as an ncc formula, is

$$\text{(AP)} \qquad \forall x\,.\,(\forall n \in \mathbb{N}\,|x| \leq 2^{-n}) \to x = 0$$

where the set of natural numbers, $\mathbb{N} \subseteq \mathbb{R}$, is defined inductively as

$$\mathbb{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbb{N}(x-1)$$

(recall that this stands for $\mathbb{N} = \mu\,\lambda X \lambda x\,.\,x = 0 \vee X(x-1)$). The integers and the rational numbers, $\mathbb{Z}, \mathbb{Q} \subseteq \mathbb{R}$ are defined by

$$\mathbb{Z} = \{x \mid x \in \mathbb{N} \vee -x \in \mathbb{N}\}, \qquad \mathbb{Q} = \{q \mid \exists k \in \mathbb{N}\,.\,k > 0 \wedge kq \in \mathbb{Z}\}$$

The function $\max : \mathbb{R} \to \mathbb{R}$ is introduced (i.e. added to the signature) and axiomatized by

$$\max(x,y) \leq z \leftrightarrow x \leq z \wedge y \leq z$$

and the absolute value is defined by $|x| := \max(x, -x)$.

The set of real numbers in $\mathbb{I}$ that can be approximated by rational numbers is defined by

$$\mathrm{A} = \{x \in \mathbb{I} \mid \forall n \in \mathbb{N}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\,.\, |x - q| \leq 2^{-n}\}\,.$$

A realizer of "$x \in \mathrm{A}$" is a fast rational Cauchy-sequence converging to x.

Let $\mathrm{SD} = \{-1, 0, 1\}$ be the set of *signed digits* (hence $d \in \mathrm{SD}$ means $d = -1 \vee d = 0 \vee d = 1$) and set $\mathbb{I}_d := [d/2 - 1/2, d/2 + 1/2] \subseteq \mathbb{I}$. We define $\mathrm{C} \subseteq \mathbb{R}$ coinductively by

$$\mathrm{C} \overset{\nu}{=} \{x \in \mathbb{R} \mid \exists d \in \mathrm{SD}\,.\, x \in \mathbb{I}_d \wedge 2x - d \in \mathrm{C}\}\,.$$

Classically, $\mathrm{C} = \mathbb{I}$. A realizer of "$x \in \mathrm{C}$" is a stream of signed digits $d_0, d_1, \ldots$ such that

$$x = \sum_{i=0}^{\infty} d_i 2^{-(i+1)}\,.$$

Therefore, C corresponds to the signed digit representation of real numbers in $\mathbb{I}$.

▶ **Lemma 3.**
(a) *If $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ for some $a, b \in \mathbb{Q}$, then $y \in \mathrm{A}$.*
(b) *If $x \in \mathrm{C}$ and $b \in \mathbb{Q}$ is such that $x + b \in \mathbb{I}$, then $x + b \in \mathrm{C}$.*
(c) *If $x \in \mathrm{C}$ and $2x + b \in \mathbb{I}$, where $b \in \mathbb{Q}$, then $2x + b \in \mathrm{C}$.*
(d) *If $x \in \mathrm{C}$, then $-x \in \mathrm{C}$.*

Proof. See Appendix. ◀

▶ **Theorem 4.** $\mathrm{C} = \mathrm{A}$.

Proof. "$\mathrm{C} \subseteq A$": We show $\forall n \in \mathbb{N} \forall x \in \mathrm{C}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\,.\, |x - q| \leq 2^{-n}$ by induction on n. If $n = 0$, set $q = 0$. For $n+1$ assume $x \in \mathrm{C}$. Let $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in \mathrm{C}$. By i.h. we have $q \in \mathbb{Q} \cap \mathbb{I}$ with $|(2x - d) - q| \leq 2^{-n}$. Hence $|x - (q + d)/2| = |(2x - d) - q|/2 \leq 2^{-(n+1)}$.

For "$A \subseteq \mathrm{C}$" we use coinduction. Assume $x \in A$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in A$. Since $x \in \mathrm{A}$ we have $q \in \mathbb{Q}$ such that $|x - q| \leq 1/4$. If $q \leq -1/2$ we know $x \in \mathbb{I}_{-1}$, and from Lemma 3 (a) it follows that $2x + 1 \in A$. Similarly, if $q \geq 1/2$ we know $x \in \mathbb{I}_1$ and $2x - 1 \in A$. Otherwise $x \in \mathbb{I}_0$ and $2x \in A$. ◀

4 Infinite Gray code

In this section we introduce infinite Gray code via a coinductive predicate G and prove its equivalence to the Cauchy and signed digit representation, using the (in IFP not realizable) Disjunction Principle to prove $\mathrm{G} \subseteq \mathrm{C}$.

Let $\mathrm{t}(x) = 1 - 2|x|$ be the *tent map*, which maps $\mathbb{I}$ onto itself, and set

$$\mathrm{D}(x) := x \neq 0 \to x \leq 0 \vee x \geq 0.$$

We define $\mathrm{G} \subseteq \mathbb{R}$ coinductively by

$$\mathrm{G}(x) \overset{\nu}{=} |x| \leq 1 \wedge \mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))\,.$$

Classically, we clearly have $\mathrm{G} = \mathbb{I}$. A realizer of "$x \in \mathrm{G}$" is a stream of partial Booleans representing the itinerary of x along the tent map. Such a realizing stream can have at most one undefined item, namely if $\mathrm{t}^n(x) = 0$, in which case the item with index n is undefined. In this case, $\mathrm{t}^{(n+1)}(x) = 1$ and $\mathrm{t}^m(x) = 0$ for $m > n + 1$, hence the items after the undefined

one are $1, 0, 0, 0, \ldots$. In the proof of Theorem 5 below we use the axiom of countable choice for rational numbers (AC^ω) and Markov's principle (MP), which are both realizable.

$$(\mathrm{AC}^\omega) \qquad (\forall n \in \mathbb{N}\, \exists q \in \mathbb{Q}\, A(n,q)) \to \exists f : \mathbb{N} \to \mathbb{Q}\, \forall n \in \mathbb{N}\, A(n, f(n)) \text{ where } A \text{ is ncc.}$$

$$(\mathrm{MP}) \qquad (\forall n \in \mathbb{N}\,.\, A(n) \vee \neg A(n)) \wedge (\neg\neg \exists n \in \mathbb{N}\, A(n)) \to \exists n \in \mathbb{N}\, A(n) \text{ where } A \text{ is ncc.}$$

It is easy to see that (AC^ω) is realized by the identity function, $\lambda a\,.\, a$, and (MP) is realized by unbounded search through the (unary representations of) natural numbers, which can be coded as $\lambda f\,.\,(\mathrm{rec}\, g\,.\,\lambda a\,.\,\mathrm{case}\ f\, a\ \mathrm{of}\ \{\mathrm{inl}(\_) \to a\,;\, \mathrm{inr}(\_) \to g\,(\mathrm{inr}(a))\})\,(\mathrm{inl}(\mathrm{nil}))$.

▶ **Theorem 5** ($\mathrm{AP}, \mathrm{AC}^\omega, \mathrm{MP}$)**.** $\mathrm{A} \subseteq \mathrm{G}$

Proof. By coinduction. Assume $\mathrm{A}(x)$. We have to show $\mathrm{D}(x)$ and $A(\mathrm{t}(x))$. The latter is easy: To show $A(\mathrm{t}(x))$, fix $n \in \mathbb{N}$. Let $q \in \mathbb{Q} \cap \mathbb{I}$ such that $|x - q| \le 2^{-(n+1)}$. Then $\mathrm{t}(q) \in \mathbb{Q} \cap \mathbb{I}$ and $|\mathrm{t}(x) - \mathrm{t}(q)| = 2||x| - |q|| \le 2|x - q| \le 2^{-n}$.

Now we show $\mathrm{D}(x)$. Assume $x \neq 0$. By (AC^ω), there exists a sequence of rational numbers q_n such that $\forall n \in \mathbb{N}\, |x - q_n| \le 2^{-n}$. We show $\neg\forall n \in \mathbb{N}\, |q_n| \le 2^{1-n}$. Assume $\forall n \in \mathbb{N}\, |q_n| \le 2^{1-n}$. Then $\forall n \in \mathbb{N}\, |x| \le 2|q_n| \le 2^{2-n}$. By (AP) it follows $x = 0$, contradicting the assumption $x \neq 0$. By (MP) it follows that there exists $n \in \mathbb{N}$ such that $|q_n| > 2^{1-n}$. If $q_n > 0$, then we have $q_n \ge 2^{1-n}$, which, together with $|x - q_n| \le 2^{-n}$, implies $x \ge 2^{-n} > 0$. Similarly, if $q_n < 0$, then we have $q_n \le -2^{1-n}$, which, together with $|x - q_n| \le 2^{-n}$, implies $x \le -2^{-n} < 0$. ◀

In Lemma 6 below we use the Disjunction Principle discussed in the Introduction:

$$(\mathrm{DP}) \qquad (A \overset{P}{\vee} B) \wedge (P \overset{Q}{\vee} C) \to A \vee B \vee C\,.$$

▶ **Lemma 6** (DP)**.** *If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.*

Proof. Assume $x \in \mathrm{G}$. Then $\mathrm{D}(x)$ and $\mathrm{D}(\mathrm{t}(x))$. In order to apply (DP), set $A := x \in \mathbb{I}_{-1}$, $B := x \in \mathbb{I}_1$, $C := x \in \mathbb{I}_0$, $P := x \neq 0$, and $Q := \mathrm{t}(x) \neq 0$. Hence, it suffices to show that the premises of (DP) hold. $P \to A \vee B$ is $\mathrm{D}(x)$. To show $\neg P \to A \wedge B$, assume $\neg(x \neq 0)$, that is, $x = 0$. Then clearly $x \in \mathbb{I}_{-1}$ and $x \in \mathbb{I}_1$. To show $Q \to C \vee P$, assume $\mathrm{t}(x) \neq 0$. Then $\mathrm{t}(x) \le 0 \vee \mathrm{t}(x) \ge 0$, since $\mathrm{D}(\mathrm{t}(x))$. If $\mathrm{t}(x) \le 0$, then $|x| \ge \frac{1}{2}$, hence $x \neq 0$. If $\mathrm{t}(x) \ge 0$, then $x \in \mathbb{I}_0$. Finally, to show $\neg Q \to C \wedge P$, assume $\neg(\mathrm{t}(x) \neq 0)$, that is, $\mathrm{t}(x) = 0$. Then $|x| = \frac{1}{2}$, hence $x \in \mathbb{I}_0$ and $x \neq 0$. ◀

▶ **Lemma 7.** *If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$ and $|x| \in \mathrm{G}$.*

Proof. Assume $x \in \mathrm{G}$. Then $\mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))$. Since $\mathrm{t}(x) = \mathrm{t}(-x) = \mathrm{t}(|x|)$ it easily follows $\mathrm{D}(-x) \wedge \mathrm{G}(\mathrm{t}(-x))$ and also $\mathrm{D}(-x) \wedge \mathrm{G}(\mathrm{t}(|x|))$. Hence $\mathrm{G}(-x)$ and $\mathrm{G}(|x|)$. ◀

▶ **Lemma 8.** *If $|x| \le 1$ and $\mathrm{G}(\frac{x+1}{2})$, then $\mathrm{G}(x)$. Equivalently, if $0 \le x \le 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x - 1)$.*

Proof. Assume $|x| \le 1$ and $\mathrm{G}(\frac{x+1}{2})$. Then $\mathrm{G}(\mathrm{t}(\frac{x+1}{2}))$, that is, $\mathrm{G}(-x)$, since the assumption $|x| \le 1$ implies $\mathrm{t}(\frac{x+1}{2}) = -x$. Hence $G(x)$, by Lemma 7. ◀

▶ **Lemma 9.** *If $\mathrm{G}(x)$, then $\mathrm{G}(\mathrm{t}(x))$.*

Proof. Obvious. ◀

▶ **Lemma 10.** *If $0 \le x \le 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1 - x)$.*

Proof. Assume $0 \leq x \leq 1$ and $\mathrm{G}(x)$. Since $0 \leq 1 - x \leq 1$, it suffices to show $\mathrm{G}(\mathrm{t}(1-x))$. The assumption $x \leq 1$ implies that $\mathrm{t}(1-x) = 2x - 1$. Hence $\mathrm{G}(2x-1)$, by Lemma 8. ◀

▶ **Lemma 11.** *If* $|x| \leq 1$ *and* $\mathrm{G}(\frac{x}{2})$*, then* $\mathrm{G}(x)$*. Equivalently, if* $-\frac{1}{2} \leq x \leq \frac{1}{2}$ *and* $\mathrm{G}(x)$*, then* $\mathrm{G}(2x)$*.*

Proof. Assume $|x| \leq 1$ and $\mathrm{G}(\frac{x}{2})$. Hence $\mathrm{G}(\mathrm{t}(\frac{x}{2}))$. Since $\mathrm{t}(\frac{x}{2}) = 1 - |x|$, we have $\mathrm{G}(1-|x|)$ and therefore $\mathrm{G}(|x|)$, by Lemma 10 and since $0 \leq 1 - |x| \leq 1$. Hence $\mathrm{G}(\mathrm{t}(x))$ (since $\mathrm{t}(|x|) = \mathrm{t}(x)$), by Lemma 9. Furthermore $\mathrm{D}(\frac{x}{2})$ which implies $\mathrm{D}(x)$. It follows $\mathrm{G}(x)$. ◀

▶ **Theorem 12** (DP)**.** $\mathrm{G} \subseteq \mathrm{C}$.

Proof. By coinduction. Assume $\mathrm{G}(x)$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $\mathrm{G}(2x-d)$. By Lemma 6 (which was proven using DP), there exists $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$. By the Lemmas 9 (note that if $x \in \mathbb{I}_{-1}$, then $2x+1 = \mathrm{t}(x)$), 8, and 11 we have $\mathrm{G}(2x-d)$. ◀

5 Concurrent Fixed Point Logic

In this section we introduce Concurrent Fixed Point Logic, CFP, extending IFP.

CFP extends the language of IFP by a modal operator S. The proof calculus of IFP is extended by the rules

$$\frac{\Gamma \vdash A}{\Gamma \vdash \mathrm{S}(A)}\ (\mathrm{S}^+) \qquad \frac{\Gamma \vdash \mathrm{S}(A) \qquad \Gamma, A \vdash \mathrm{S}(B)}{\Gamma \vdash \mathrm{S}(B)}\ (\mathrm{S}^-) \qquad \frac{\Gamma \vdash \mathrm{S}(A)}{\Gamma \vdash A}\ (\mathrm{S}^{\mathrm{nc}}) \text{ if } A \text{ is ncc}$$

which will be justified by the Soundness Theorem 16. The rules (S^+) and (S^-) state that S is a strong monad (see [10] for an analogous construction for a computational lambda-calculus). They immediately imply monotonicity, $(A \to B) \to \mathrm{S}(A) \to \mathrm{S}(B)$, and idempotency, $\mathrm{S}(\mathrm{S}(A)) \leftrightarrow \mathrm{S}(A)$, and that S interacts nicely with the logical operators, for example, $\mathrm{S}(A \wedge B) \leftrightarrow \mathrm{S}(A) \wedge \mathrm{S}(B)$, $\mathrm{S}(A \to B) \to \mathrm{S}(A) \to \mathrm{S}(B)$ and $\mathrm{S}(\forall x\, A) \to \forall x\, \mathrm{S}(A)$.

The modality S can be viewed as a predicate transformer by setting $\mathrm{S}(\mathcal{P}) = \lambda \vec{x}\,.\, S(\mathcal{P}(\vec{x}))$. In particular, $\mathrm{S}(\lambda \vec{x}\,.\, A) = \lambda \vec{x}\,.\, S(A)$. Hence, if $\Phi = \lambda X \lambda \vec{x}\,.\, A$ is a monotone predicate transformer, we can form the composition $\Phi \circ S = \lambda X\,.\, \Phi(\mathrm{S}(X)) = \lambda X \lambda \vec{x}\,.\, A[\mathrm{S}(X)/X] = \lambda X \lambda \vec{x}\,.\, A[\lambda \vec{y}\,.\, \mathrm{S}(X(\vec{y}))/X]$. We call a predicate variable *guarded* in a formula A if every free occurrence of X in A is within a disjunctive or existential subformula of A. We call a predicate transformer $\Phi = \lambda X \lambda \vec{x}\,.\, A$ guarded if X is guarded in A.

We embed IFP into CFP by defining for every IFP-formula A a CFP-formula A^{S} (if $\Phi = \lambda X \lambda \vec{x}\,.\, A$, we set $\Phi^{\mathrm{S}} = \lambda X \lambda \vec{x}\,.\, A^{\mathrm{S}}$):

$$\begin{aligned}
A^{\mathrm{S}} &= A \quad \text{if } A \text{ is atomic, i.e. of the form } P(\vec{t}) \text{ or } X(\vec{t}) \\
(A \wedge B)^{\mathrm{S}} &= A^{\mathrm{S}} \wedge B^{\mathrm{S}} \\
(A \to B)^{\mathrm{S}} &= A^{\mathrm{S}} \to B^{\mathrm{S}} \\
(A \vee B)^{\mathrm{S}} &= \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}}) \\
(\forall x\, A)^{\mathrm{S}} &= \forall x\, A^{\mathrm{S}} \\
(\exists x\, A)^{\mathrm{S}} &= \mathrm{S}(\exists x\, A^{\mathrm{S}}) \\
((\nu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\nu\, \Phi^{\mathrm{S}})(\vec{t}) \\
((\mu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\mu\, \Phi^{\mathrm{S}})(\vec{t}) \quad \text{if } \Phi \text{ is guarded} \\
((\mu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S}))(\vec{t}) \quad \text{if } \Phi \text{ is not guarded}
\end{aligned}$$

The guardedness condition for $\Phi = \lambda X \lambda \vec{a} A$ in $\mu\,\Phi$ is satisfied in all inductive definitions given by two or more "rules" (for example the natural numbers are defined by two rules) since then the predicate variable X occurs only within a disjunction.

▶ **Lemma 13.** CFP *proves* $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$ *(hence* $\mathrm{S}(A^{\mathrm{S}}) \leftrightarrow A^{\mathrm{S}}$*) for all formulas* A *without free predicate variables.*

Proof. See Appendix. ◀

▶ **Theorem 14** (Concurrent embedding). *If* $\Gamma \vdash_{\mathrm{IFP}} A$*, then* $\Gamma^{\mathrm{S}} \vdash_{\mathrm{CFP}} A^{\mathrm{S}}$ *for all formulas* A *without free predicate variables.*

Proof. See Appendix. The proof depends crucially on Lemma 13. ◀

Realizability for CFP is formalized in a system RCFP that extends both CFP and RIFP. We extend the programming language introduced in Sect. 2 by constructs for concurrent computation, where the latter is modeled by a family of computations indexed by a set $\mathcal{I}$ which is the least set containing the constant $*$ and with i, j the elements $\mathrm{L}(i)$, $\mathrm{R}(i)$ and (i, j). We denote elements of $\mathcal{I}$ by *index terms*, denoted $s, t, \ldots$, which are first-order terms built from index variables $\alpha, \beta, \ldots$ and the constant $*$ using the constructors $\mathrm{L}(\_)$, $\mathrm{R}(\_)$, and $(\_,\_)$. Closed index terms can be identified with elements of $\mathcal{I}$ and will be called *indices* and denoted $i, j, k, \ldots$. *Concurrent program terms* are defined like the program terms in Sect. 2, but with the extra constructs of *non-deterministic choice* $\langle\alpha\rangle M$ (binding α in M), *index application* $M \cdot s$, and *pattern matching on indices*

$$\text{case } s \text{ of } \{* \to K\,;\, \mathrm{L}(\alpha) \to L\,;\, \mathrm{R}(\alpha) \to M\,;\, (\alpha, \beta) \to N\}$$

(binding α in L and M and α, β in N). In a pattern matching we may omit some of the clauses in which case the omitted clauses have the default value nil. For example, case s of $\{(\alpha, \beta) \to M\}$ stands for case s of $\{* \to \mathrm{nil}\,;\, \mathrm{L}(\_) \to \mathrm{nil}\,;\, \mathrm{R}(\_) \to \mathrm{nil}\,;\, (\alpha, \beta) \to M\}$. We use the abbreviation $\langle\alpha, \beta\rangle M$ for the term $\lambda\gamma\,.\,\text{case } \gamma \text{ of } \{(\alpha, \beta) \to M\}$ and define, for later use, the terms

$$\begin{aligned} \text{return} &:= \lambda a\, \langle\alpha\rangle \mathrm{inr}(a) \\ \text{bind} &:= \lambda c \lambda f\, \langle\alpha, \beta\rangle \text{case } c \cdot \alpha \text{ of } \{\mathrm{inl}(a) \to (f\, a) \cdot \beta\,;\, \mathrm{inr}(\_) \to \mathrm{nil}\} \end{aligned}$$

The logical language of RCFP extends the language of RIFP by a sort of indices and quantification over indices. The specification of programs is extended in RCFP by the equations

$$\begin{aligned} (\langle\alpha\rangle M) \cdot s &= M[s/\alpha] \\ \text{case } * \text{ of } \{\ldots\} &= K \\ \text{case } \mathrm{L}(s) \text{ of } \{\ldots\} &= L[s/\alpha] \\ \text{case } \mathrm{R}(s) \text{ of } \{\ldots\} &= M[s/\alpha] \\ \text{case } (s, t) \text{ of } \{\ldots\} &= N[s, t/\alpha, \beta] \end{aligned}$$

where $\{\ldots\} = \{* \to K\,;\, \mathrm{L}(\alpha) \to L\,;\, \mathrm{R}(\alpha) \to M\,;\, (\alpha, \beta) \to N\}$.

Realizability for formulas of the form $\mathrm{S}(A)$ is defined as

$$c\,\mathbf{r}\,\mathrm{S}(A) \quad = \quad \exists \alpha, a\,(c \cdot \alpha = \mathrm{inl}(a)) \wedge \forall \alpha, a\,(c \cdot \alpha = \mathrm{inl}(a) \to a\,\mathbf{r}\,A)$$

We do *not* stipulate closure of nc or ff formulas under S. Therefore, the sets of nc and ff formulas remains unchanged and the Lemma 1 remains valid for RCFP:

▶ **Lemma 15.** *If A is an ncc formula in* CFP*, then $a \,\mathbf{r}\, A$ is equivalent to A.*

▶ **Theorem 16** (Soundness for CFP)**.** *From a proof in* CFP *of $\Delta, \Gamma \vdash A$, where Δ consists of ncc formulas, one can extract a concurrent program term M such that* RCFP *proves $\Delta, \vec{a} \,\mathbf{r}\, \Gamma \vdash (M\,\vec{a}) \,\mathbf{r}\, A$.*

Proof. It suffices to verify the realizability of the new proof rules (S^+), (S^-), (S^{nc}), that is, to find realizers of the formulas
(a) $A \to \mathrm{S}(A)$,
(b) $\mathrm{S}(A) \to (A \to \mathrm{S}(B)) \to \mathrm{S}(B)$,
(c) $\mathrm{S}(A) \to A$ if A is ncc.
It is easy to see that return realizes (a) and bind realizes (b). We show that $\lambda c\,.\,\mathrm{nil}$ realizes (c). Assume $c \,\mathbf{r}\, \mathrm{S}(A)$. We have to show $\mathrm{nil} \,\mathbf{r}\, A$, that is A, by Lemma 15. Since $c \,\mathbf{r}\, \mathrm{S}(A)$, there exist α and a such that $c \cdot \alpha = \mathrm{inl}(a)$ and $a \,\mathbf{r}\, A$. Hence A, by Lemma 15. ◀

A program *a concurrently realizes* an IFP-formula A, written $a \,\mathbf{cr}\, A$, if a realizes A^{S}, that is, $a \,\mathbf{r}\, A^{\mathrm{S}}$. Combining the Embedding Theorem (Thm. 14), the Soundness Theorem for CFP (16) and the fact that realizability of A^{S} is equivalent to A for non-computational formulas (Lemma 1), one obtains:

▶ **Theorem 17** (Concurrent Soundness)**.** *From a proof of $\Delta, \Gamma \vdash_{\mathrm{IFP}} A$, where Δ is non-computational, one can extract a concurrent program M that maps concurrent realizers of Γ to a concurrent realizer of A, that is $\Delta, \vec{a} \,\mathbf{cr}\, \Gamma \vdash_{\mathrm{CFP}} (M \cdot \vec{a}) \,\mathbf{cr}\, A$.*

In order to understand what kind of extracted program we can expect from the proof of $(\mathrm{G} \subseteq \mathrm{C})^{\mathrm{S}}$, which will be obtained from the proof of $\mathrm{G} \subseteq \mathrm{C}$ (Theorem 12) using Theorem 14 and the realizer of (DP^{S}) (Lemma 30), let us write out this formula:

$$\begin{aligned}
(\mathrm{G} \subseteq \mathrm{C})^{\mathrm{S}} &= \forall x\,.\,\mathrm{G}^{\mathrm{S}}(x) \to \mathrm{C}^{\mathrm{S}}(x) \\
\mathrm{C}^{\mathrm{S}}(x) &\overset{\nu}{=} \mathrm{S}\Big(\bigvee_{d \in \mathrm{SD}} x \in \mathbb{I}_d \wedge \mathrm{C}^{\mathrm{S}}(2x - d)\Big) \\
\mathrm{G}^{\mathrm{S}}(x) &\overset{\nu}{=} (x \neq 0 \to \mathrm{S}(x < 0 \vee x > 0)) \wedge \mathrm{G}^{\mathrm{S}}(\mathrm{t}(x))
\end{aligned}$$

One sees that the predicates C^{S} and G^{S} are almost the same as the original C and G except that the digits of the streams realizing C^{S} or G^{S} can now be computed non-deterministically and concurrently. The extracted program will be able consume and produce such streams.

6 Semantics of concurrent programs and program extraction for CFP

In this section we define an operational big-step semantics for concurrent program terms and show that it fits with a domain-theoretic semantics (Adequacy Theorem 21). Combined with a denotational Soundness Theorem (Theorem 18) and a Faithfulness Theorem (Theorem 20) we obtain that from a proof of a data-formula A (see below) from assumptions Γ one can extract a concurrent program that computes from concurrent realizers of Γ (non-concurrent) data realizing A.

The denotational model of realizers for CFP is the Scott-domain D defined by the recursive domain equation

$$D = 1 + D + D + D \times D + (D \to D) + D^{\mathcal{I}}$$

where 1 is the one-point domain, $+$ denotes the separated sum, $\times$ denotes the topological product, $(D \to D)$ denotes the continuous function space, and $D^{\mathcal{I}}$ denotes the $\mathcal{I}$-fold

topological product of D. Only the last component $D^{\mathcal{I}}$ is new, the rest is as in [1] Sect 5. The components of the sum on the right-hand side of the equation above are embedded into D via the constructors $\mathrm{Nil} : D$, $\mathrm{In}_0, \mathrm{In}_1 : D \to D$, $\mathrm{Pair} : D \times D \to D$, $\mathrm{Fun} : (D \to D) \to D$, $\mathrm{Fam} : D^{\mathcal{I}} \to D$. Every concurrent program term M has an obvious denotation $[\![M]\!]\xi \in D$ in any given environment ξ that maps all its free index variables to elements of $\mathcal{I}$ and all its free object variables to elements of D. For example, $[\![\lambda a\, M]\!]\xi = \mathrm{Fun}(\lambda a' \in D\,.\,[\![M]\!]\xi[a \mapsto a'])$, $[\![\langle\alpha\rangle M]\!]\xi = \mathrm{Fam}(\lambda i \in \mathcal{I}\,.\,[\![M]\!]\xi[\alpha \mapsto i])$, $[\![(M\,N)]\!]\xi = f([\![N]\!]\xi)$ if $[\![M]\!]\xi = \mathrm{Fun}(f)$, and $[\![(M \cdot s)]\!]\xi = \varphi([\![s]\!]\xi)$ if $[\![M]\!]\xi = \mathrm{Fam}(\varphi)$, otherwise these terms have value $\bot$. If M is closed (i.e. has neither free index nor object variables), we omit the environment.

▶ **Theorem 18** (Denotational soundness). *If* RCFP *proves* $\Gamma \vdash B$*, then* B *holds in every model of* Γ *that interprets the sort of realizers as* D*. In particular, if* B *is of the form* $M\,\mathbf{r}\,A$*, then* $[\![M]\!] \in D$ *realizes* A *in that model.*

We call a CFP-formula a *parametric data formula* if every subformula of the form $A \to B$ or $\nu\,\Phi(t)$ is non-computational. A data formula is a parametric data formula without free predicate variables. Furthermore, *data terms* are defined, as in [1], as the terms built from Nil by injections and pairing. As in [1] we identify data terms with the corresponding elements in D and call them simply *data*

Our main result is analogous to the Program Extraction Theorem in [1] and refers to a big-step operational semantics. First we introduce *closures* which are inductively defined as pairs (M, η) where M is a term and η is a finite mapping from object variables to closures. A *value* is a closure (M, η) where M is an *intro term*, that is, either Nil or an injection or a pair or a λ-abstraction or a choice term, $\langle\alpha\rangle M$. The *bigstep reduction relation* $c \longrightarrow v$ between closures c and values v is inductively defined as in [1], but with additional five rules:

$$\frac{(M,\eta) \longrightarrow (\langle\alpha\rangle M_0, \eta') \qquad (M_0[s/\alpha], \eta') \longrightarrow v}{(M\,s, \eta) \longrightarrow v}$$

In the remaining four rules we use the abbreviation

$$\vec{C} := * \to M_* \,;\, \mathrm{L}(\alpha) \to M_0 \,;\, \mathrm{R}(\alpha) \to M_1 \,;\, (\alpha, \beta) \to M$$

$$\frac{(M_*, \eta) \longrightarrow v}{(\text{case } * \text{ of } \{\vec{C}\}, \eta) \longrightarrow v} \qquad \frac{(M_0[s/\alpha_p], \eta) \longrightarrow v}{(\text{case } \mathrm{L}(s) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v} \qquad \frac{(M_1[s/\alpha_p], \eta) \longrightarrow v}{(\text{case } \mathrm{R}(s) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v}$$

$$\frac{(M[s,t/\alpha,\beta], \eta) \longrightarrow v}{(\text{case } (s,t) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v}$$

Note that neither the denotational semantics nor the bigstep operational semantics exhibit any kind of non-determinism or concurrency. These features come into play only through the *printing relation*, $c \Longrightarrow d$, between closures c and data d, defined below. It is inductively defined as in [1], but with five additional rules concerned with non-deterministic choice:

$$\frac{c \longrightarrow (\langle\alpha\rangle M, \eta) \qquad (M, \eta) \Longrightarrow d}{c \Longrightarrow d}$$

In the remaining four rules (where $p = 0, 1$) α must occur free in c and β must be fresh:

$$\frac{c[*/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[\mathrm{L}(\alpha)/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[\mathrm{R}(\alpha)/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[(\alpha,\beta)/\alpha] \Longrightarrow d}{c \Longrightarrow d}$$

If M is a closed term, then we write $M \Longrightarrow d$ instead of $(M, \emptyset) \Longrightarrow$ where $\emptyset$ is the empty mapping. Essentially, these rules allow to reduce a closure c with a choice parameter α to $c[i/\alpha]$ for any $i \in \mathcal{I}$. The way the rules are set up, one can do the instantiation $c[i/\alpha]$ lazily and incrementally, by only specifying the outer shape of i in one step.

▶ **Theorem 19** (Program Extraction). *From a proof of a data formula A in* CFP *from concurrently realizable assumptions one can extract a concurrent program term M such $M \Longrightarrow d$ for some data d provably realizing A.*

The main building blocks for the proof of the Program Extraction Theorem are the Soundness Theorem (Theorem 16), the Computational Adequacy Theorem and the Faithfulness Theorem below. The latter two refer to the relation $d \in \mathrm{data}(a)$, for $a \in D$ and data d, which is defined inductively as follows:

(i) $\mathrm{Nil} \in \mathrm{data}(\mathrm{Nil})$.
(ii) If $d \in \mathrm{data}(a)$, then $\mathrm{In}_p\, d \in \mathrm{data}(\mathrm{In}_p\, a)$ for $p = 0, 1$.
(iii) If $d_p \in \mathrm{data}(a_p)$ for $p = 0, 1$, then $\mathrm{Pair}\, d_1\, d_1 \in \mathrm{data}(\mathrm{Pair}\, a_0\, a_1)$.
(iv) If $d \in \mathrm{data}(\varphi\, i)$ for some $i \in \mathcal{I}$, then $d \in \mathrm{data}(\mathrm{Fam}\, \varphi)$.

▶ **Theorem 20** (Faithfulness). *If $a \in D$ concurrently realizes a data formula A, then* $\mathrm{data}(a)$ *is nonempty and all $d \in \mathrm{data}(a)$ realize A, provably in* IFP.

Proof. See Appendix. ◀

▶ **Theorem 21** (Computational Adequacy). *If $d \in \mathrm{data}([\![M]\!])$, then $M \Longrightarrow d$.*

The proof of Computational Adequacy is quite involved and will occupy the rest of this section.

Proof of the Program Extraction Theorem from Soundness, Computational Adequacy and Faithfulness. From a proof of a data formula A in CFP from realizable assumptions, one obtains by Soundness a concurrent program term M realizing A. More precisely, $[\![M]\!]$ realizes A. By Faithfulness, there is a data $d \in \mathrm{data}([\![M]\!])$ such that d provably deterministically realizes A. By Computational Adequacy, $M \Longrightarrow d$. ◀

Now we develop the necessary machinery to prove Computational Adequacy. For a closure c we let $\overline{c}$ be the closed term represented by c, that is,

$$\overline{(M, \eta)} = M[\overline{\eta(x)}/x \mid x \in \mathrm{FV}(M)].$$

The proof is done through the following series of lemmas whose proofs can be found in the Appendix.

▶ **Lemma 22** (Correctness).
(a) *If $c \longrightarrow v$, then $\vdash \overline{c} = \overline{v}$.*
(b) *If $c \Longrightarrow d$, then $\vdash d \in \mathrm{data}([\![\overline{c}]\!])$.*

▶ **Lemma 23** (Instantiation).
(a) *If $c[i/\alpha] \Longrightarrow d$, then $c \Longrightarrow d$.*
(b) *If $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \Longrightarrow d$, then $c \Longrightarrow d$.*

▶ **Lemma 24** (Irreducibility of values). *For values v, v'*

$$v \longrightarrow v' \quad \textit{iff} \quad v = v'\,.$$

Let D_0 be the set of compact elements of D. Every $a \in D_0$ has a natural *rank*, $\mathbf{rk}(a) \in \mathbb{N}$, satisfying properties $\mathbf{rk}1 - \mathbf{rk}4$. The first three properties are as in [1], the fourth is
rk4 If $\mathrm{Fam}\, \varphi$ is compact, then for every $i \in \mathcal{I}$, $\varphi\, i$ is compact with $\mathbf{rk}(\varphi\, i) < \mathbf{rk}(\mathrm{Fam}\, \varphi)$.

To every $a \in D_0$ we assign a set of closures $\mathrm{Cl}(a)$ by recursion on $\mathbf{rk}(a)$. The definition is as in [1], with the extra clause

$$\mathrm{Cl}(\mathrm{Fam}\,\varphi) = \{c \mid \exists \alpha, M, \eta\,.\, c \longrightarrow (\langle\alpha\rangle M, \eta) \wedge \forall i \in \mathcal{I}\,.\,(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)\}$$

A similar assignment of closures to the compact elements of a semantic domain is used in [15].

▶ **Lemma 25** (Monotonicity). *If a, b are compact elements in D such that $a \sqsubseteq b$, then* $\mathrm{Cl}(a) \supseteq \mathrm{Cl}(b)$.

▶ **Lemma 26** (Printing of data). *If $c \in \mathrm{Cl}(a)$ and $d \in \mathrm{data}(a)$, then $c \Longrightarrow d$.*

▶ **Lemma 27** (Reducibility of closures). *$c \in \mathrm{Cl}(a)$ iff $c \longrightarrow v$ for some $v \in \mathrm{Cl}(a)$.*

We write $\eta \in \mathrm{Cl}(\xi)$ if η and ξ have the same domain and $\eta(x) \in \mathrm{Cl}(\xi(x))$ for every object variable x and $\eta(\alpha) = \xi(\alpha)$ for every index variable α in the common domain.

▶ **Lemma 28** (Approximation). *If $\eta \in \mathrm{Cl}(\xi)$, $a \in D_0$ and $a \sqsubseteq [\![M]\!]\xi$, then $(M, \eta) \in \mathrm{Cl}(a)$.*

▶ **Lemma 29.** *If $d \in \mathrm{data}(a)$, then $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq a$.*

Proof of the Adequacy Theorem (Theorem 21). Assume $d \in \mathrm{data}([\![M]\!])$ where M is closed. By Lemma 29, $d \in \mathrm{data}(a)$ for some compact $a \sqsubseteq [\![M]\!]$. By Lemma 28, $(M, \emptyset) \in \mathrm{Cl}(a)$. By Lemma 26, $M \Longrightarrow d$. ◀

7 Realizing the Disjunction Principle

In this Section we show that the Disjunction Principle can be concurrently realized.

Recall that in Theorem 12 we proved in IFP that $\mathrm{G} \subseteq \mathrm{C}$, with the help of (DP). By Concurrent Soundness (Theorem 16), we can extract from this proof a concurrent program transforming infinite Gray code into signed digit representation, provided we can concurrently realize (DP).

▶ **Lemma 30.** *The Disjunction Principle can be concurrently realized.*

Proof. The embedding of the Disjunction Principle, $(\mathrm{DP})^{\mathrm{S}}$, is

$$(A \overset{P}{\vee} B)^{\mathrm{S}} \wedge (P \overset{Q}{\vee} C)^{\mathrm{S}} \to (A \vee B \vee C)^{\mathrm{S}}$$

where $(A \overset{P}{\vee} B)^{\mathrm{S}} = (P \to \mathrm{S}(A \vee B)) \wedge (\neg P \to A \wedge B)$, $(P \overset{Q}{\vee} C)^{\mathrm{S}} = (Q \to \mathrm{S}(P \vee C)) \wedge (\neg Q \to P \wedge C)$ and $(A \vee B \vee C)^{\mathrm{S}} = \mathrm{S}(\mathrm{S}(A \vee B) \vee C)$. Since CFP proves that $\mathrm{S}(\mathrm{S}(A \vee B) \vee C)$ is equivalent to $\mathrm{S}((A \vee B) \vee C)$ (easy exercise), it suffices to realize the formula

$$(*) \qquad (A \overset{P}{\vee} B)^{\mathrm{S}} \wedge (P \overset{Q}{\vee} C)^{\mathrm{S}} \to \mathrm{S}((A \vee B) \vee C)$$

where P, Q, A, B, C are ncc formulas. The following program realizes $(*)$: $f_{\mathrm{DP}} = \lambda c\,.$

$$\begin{aligned} \langle\gamma\rangle\,.\,\mathrm{case}\ \gamma\ \mathrm{of} \quad & \{\mathrm{L}(\alpha) \to \mathrm{case}\ \pi_1(c) \cdot \alpha\ \mathrm{of}\ \{\mathrm{inl}(\mathrm{inl}(\_)) \to \mathrm{inl}(\mathrm{inl}(\mathrm{inl}(\mathrm{nil})))\,; \\ & \qquad\qquad \mathrm{inl}(\mathrm{inr}(\_)) \to \mathrm{inl}(\mathrm{inl}(\mathrm{inr}(\mathrm{nil})))\}\,; \\ & \mathrm{R}(\beta) \to \mathrm{case}\ \pi_2(c) \cdot \beta\ \mathrm{of}\ \{\mathrm{inl}(\mathrm{inr}(\_)) \to \mathrm{inl}(\mathrm{inr}(\mathrm{nil}))\}\} \end{aligned}$$

In order to show that f_{DP} realizes $(*)$, we assume $a := \pi_1(c)$ realizes $(A \overset{P}{\vee} B)^{\mathrm{S}}$ and $b := \pi_2(c)$ realizes $(P \overset{Q}{\vee} C)^{\mathrm{S}}$. We show that $f_{\mathrm{DP}}\, c$ realizes $\mathrm{S}((A \vee B) \vee C)$. The assumptions mean:

(1) If P, then a concurrently realizes $A \vee B$, i.e.
  (1.1) $a \cdot i_0 = \mathrm{inl}(a_0)$ for some i_0, a_0,
  (1.2) If $a \cdot i = \mathrm{inl}(a')$, then $a' \,\mathbf{r}\, (A \vee B)$, that is, $a' = \mathrm{inl}(\_)$ and A, or $a' = \mathrm{inr}(\_)$ and B.
(2) If Q, then b concurrently realizes $P \vee C$, i.e.
  (2.1) $b \cdot j_0 = \mathrm{inl}(b_0)$ for some j_0, b_0,
  (2.2) If $b \cdot j = \mathrm{inl}(b')$, then $b' \,\mathbf{r}\, (P \vee C)$, that is, $b' = \mathrm{inl}(\_)$ and P, or $b' = \mathrm{inr}(\_)$ and C.
(3) If $\neg P$, then A and B.
(4) If $\neg Q$, then P and C

In the proof that $f_{\mathrm{DP}}(a, b)$ realizes $\mathrm{S}((A \vee B) \vee C)$ we argue classically, admitting case analysis on P and Q.

The first condition holds since, if P holds, then, by (1), $f_{\mathrm{DP}}(a, b) \cdot \mathrm{L}(i_0)$ is of the form $\mathrm{inl}(\_)$. If P does not hold, then Q holds, by (4), and therefore $b \cdot j_0 = \mathrm{inl}(b_0)$, by (2.1). By (2.2), $b_0 = \mathrm{inr}(\_)$, since P does not hold. Hence $f_{\mathrm{DP}}(a, b) \cdot \mathrm{R}(j_0)$ is of the form $\mathrm{inl}(\_)$.

To verify the second condition, assume $f_{\mathrm{DP}}(a, b) \cdot k = \mathrm{inl}(c)$. We have to show that c realizes $(A \vee B) \vee C$. By the definition of $f_{\mathrm{DP}}(a, b)$, k is of the form $\mathrm{L}(i)$ or $\mathrm{R}(j)$.

If $k = \mathrm{L}(i)$, then either $a \cdot i = \mathrm{inl}(\mathrm{inl}(\_))$ and $c = \mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$, or else $a \cdot i = \mathrm{inl}(\mathrm{inr}(\_))$ and $c = \mathrm{inl}(\mathrm{inr}(\mathrm{nil}))$. If P holds, then a concurrently realizes $A \vee B$, by (1). Hence, if $a \cdot i = \mathrm{inl}(\mathrm{inl}(\_))$, then A holds, hence $\mathrm{inl}(\mathrm{nil})$ realizes $A \vee B$ and consequently $c = \mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$ realizes $(A \vee B) \vee C$. The case that $a \cdot i = \mathrm{inl}(\mathrm{inr}(\_))$ is similar. If P does not hold, then A and B hold, by (3), hence $\mathrm{inl}(\mathrm{nil})$ and $\mathrm{inr}(\mathrm{nil})$ both realize $A \vee B$. It follows in any case that c realizes $(A \vee B) \vee C$.

If $k = \mathrm{R}(j)$, then $b \cdot j = \mathrm{inl}(\mathrm{inr}(\_))$ and $c = \mathrm{inr}(\mathrm{nil})$. If Q holds then C holds by (2.2). If Q does not hold, then C holds by (4). In either case c realizes $((A \vee B) \vee C$. ◀

In order to understand f_{DP} we express its behaviors in terms of overlapping defining equations that ignore the indices in $(\in \mathcal{I})$ labeling the different choices. Ignoring also the leading $\mathrm{inl}(\cdot)$ of the inputs and outputs, which only flag up a valid result, we obtain (writing `(a,b)` for $\langle a, b\rangle$)

```
fDP (inl(_),b) = inl(inl(nil))
fDP (inr(_),b) = inl(inr(nil))
fDP (a,inr(_)) = inr(nil)
```

These equation must be interpreted as non-deterministic rewrite rules, similar to the program `gtos` in the introduction.

8 Extracting programs for infinite Gray code

We conclude by extracting the programs from the proofs of the Lemmas 6-11 in Sect. 4, and assembling them, via program extraction from the proof of Theorem 12, to yield the main program transforming infinite Gray to signed digit representation. Since the proofs are so short it is easy to read off the extracted programs from the proof "by hand".

For signed digit streams (that is, realizers of $\mathrm{C}(x)$) we display the possible digits $\mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$, $\mathrm{inl}(\mathrm{inr}(\mathrm{nil}))$, $\mathrm{inr}(\mathrm{nil})$ as `-1,1,0`, respectively. For infinite Gray codes (that is, realizers of $\mathrm{G}(x)$) we display the possible digits $\mathrm{inl}(\mathrm{nil})$, $\mathrm{inr}(\mathrm{nil})$ as `0,1`, respectively. Note that with this display the program `fDP` (which will be used in the next program) reads

```
fDP (0,b) = -1
fDP (1,b) = 1
fDP (a,1) = 0
```

Furthermore, we display nested pairs like $\langle a, \langle b, s\rangle\rangle$ as `a:b:s`.

Lemma 6. If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

```
f6 (a:b:s) = fDP (a,b)
```

Lemma 7. (a) If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$. (b) If $x \in \mathrm{G}$, then $|x| \in \mathrm{G}$.

```
f7a (a:s) = swap a : s    where {swap 0 = 1;  swap 1 = 0}
f7b (a:s) = 1:s
```

Lemma 8. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x-1)$.

```
f8 (a:s) = f7a s
```

Lemma 9. If $\mathrm{G}(x)$, then $\mathrm{G}(\mathrm{t}(x)$.

```
f9 (a:s) = s
```

Lemma 10. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1-x)$.

```
f10 s = 1 : f8 s
```

Lemma 11. If $-\frac{1}{2} \leq x \leq \frac{1}{2}$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x)$.

```
f11 (a:s) = a : f9 (f10 s)
```

Theorem 12. $\mathrm{G} \subseteq \mathrm{C}$.

```
f12 s = let { d = f6 s } in
   d : case d of {-1 -> f12 (f9 s); 0 -> f12 (f11 s); 1 -> f12 (f8 s)}
```

Hence

```
f12 (0:s) = -1 : f12 s
f12 (1:a:s) = 1 : f12 (swap a : s)
f12 (a:1:c:s) = 0 : f12 (a : swap c : s)
```

Again, the equations above should be read as overlapping rewrite rules. Observe that the equations for `f12` correspond exactly to the equations given for the program `gtos` shown in the introduction.

9 Conclusion

We introduced a logic and realizability interpretation for the extraction of non-deterministic concurrent programs and applied it to extract Tsuiki's program converting infinite Gray code for real numbers into signed digit representation. Through the Soundness and Computational Adequacy Theorems, extracted programs come with formal proofs of their correctness and termination.

Although we are still far from a fully fledged method of certified code generation for non-deterministic and concurrent programs, we believe that our application to infinite Gray code (which was done on paper) can be viewed as a proof of concept that makes it worthwhile to implement this method in a suitable proof system.

Regarding further applications, computable analysis holds plenty of other inherently non-deterministic problems (for example, root finding or inversion of matrices with real valued entries) to which our method can be applied.

References

1 U. Berger. Realisability for induction and coinduction with applications to constructive analysis. *Jour. Universal Comput. Sci.*, 16(18):2535–2555, 2010.

2 U. Berger and T. Hou. A realizability interpretation of Church's simple theory of types. *Mathematical Structures in Computer Science*, 2016. To appear.

3 U. Berger, K. Miyamoto, H. Schwichtenberg, and M. Seisenberger. Minlog – a tool for program extraction for supporting algebra and coalgebra. In *CALCO-Tools*, volume 6859 of *LNCS*, pages 393–399. Springer Verlag, Berlin, Heidelberg, New York, 2011. doi:10.1007/978-3-642-22944-2_29.

4 U. Berger, K. Miyamoto, H. Schwichtenberg, and H. Tsuiki. Logic for Gray-code computation. In *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, Ontos Mathematical Logic 6. de Gruyter, 2016. To appear.

5 S. Berghofer. Program Extraction in simply-typed Higher Order Logic. In *Types for Proofs and Programs (TYPES'02)*, volume 2646 of *LNCS*, pages 21–38. Springer Verlag, Berlin, Heidelberg, New York, 2003.

6 A. Bucciarelli, T. Ehrhard, and G. Manzonetto. A relational semantics for parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7):918–934, 2012.

7 C. M. Chuang. *Extraction of Programs for Exact Real Number Computation Using Agda*. PhD thesis, Swansea University, 2011.

8 R.L. Constable. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice–Hall, New Jersey, 1986.

9 S. Hayashi and H. Nakano. *PX: A Computational Logic*. MIT Press, 1988.

10 Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

11 C.-H.L. Ong. Non-determinism in a functional setting. In *Proc. of LICS'93*, pages 275–286, 1993.

12 C. Paulin-Mohring. Inductive definitions in the system Coq; rules and properties. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, pages 328–345. LNCS Vol. 664, 1993.

13 G.D. Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5(3):452–487, 1976.

14 H. Tsuiki. Real Number Computation through Gray Code Embedding. *Theoretical Computer Science*, 284(2):467–485, 2002.

15 G. Winskel. *The Formal Semantics of Programming Languages*. Foundations of Computing Series. The MIT Press, Cambridge, Massachusetts, 1993.

A Appendix: Proofs

Lemma 1. If A is an ncc formula, then $a\,\mathbf{r}\,A$ is equivalent to A, provably in RIFP.

Proof. For a predicate $\mathcal{P}$ let $\mathrm{K}\,\mathcal{P} = \lambda(a,\vec{x})\,\mathcal{P}(\vec{x})$ (that is, $\mathcal{P}$ is extended by an extra dummy first argument ranging over realizers). For a predicate $\mathcal{Q}$ with first argument ranging over realizers let $\exists\,\mathcal{Q} = \lambda\vec{x}\,\exists a\,\mathcal{Q}(a,\vec{x})$ and $\forall\,\mathcal{Q} = \lambda\vec{x}\,\forall a\,\mathcal{Q}(a,\vec{x})$. Note that $\exists$ is left adjoint to K, that is $\exists\,\mathcal{Q} \subseteq \mathcal{P}$ iff $\mathcal{Q} \subseteq \mathrm{K}\,\mathcal{P}$, and $\forall$ is right adjoint to K, that is $\mathcal{P} \subseteq \forall\,\mathcal{Q}$ iff $\mathrm{K}\,\mathcal{P} \subseteq \mathcal{Q}$. Hence $\mathrm{K}\,(\forall\,\mathcal{Q}) \subseteq \mathcal{Q} \subseteq \mathrm{K}\,(\exists\,\mathcal{Q})$. In addition, if $\mathcal{Q} = \mathrm{K}\,\mathcal{P}$, then $\forall\,\mathcal{Q} = \exists\,\mathcal{Q} = \mathcal{P}$. For an IFP-formula A, let

$$\begin{aligned} \mathrm{K}_A &= \{\tilde{X} = \mathrm{K}\,X \mid X \text{ free in } A\} \\ \exists_A &= \{\exists\,\tilde{X} = X \mid X \text{ free in } A\} \end{aligned}$$

We show, more generally, that RIFP proves

(a) $\mathrm{K}_A \vdash A \leftrightarrow a\,\mathbf{r}\,A$ for nc formulas A.
(b) $\exists_A \vdash A \leftrightarrow \exists a\,a\,\mathbf{r}\,A$ for ff formulas A.
The proof is by simultaneous induction on A.

For (a), the only non-obvious cases are implication as well as inductive and coinductive definitions.

Consider $A \to B$ where A is ffc and B is nc. W.l.o.g. let us assume that A is not ncc. Hence $c\,\mathbf{r}\,(A \to B)$ is $\forall a(a\,\mathbf{r}\,A \to (c \cdot a)\,\mathbf{r}\,B)$. By induction hypothesis (a) this is equivalent to $\forall a(a\,\mathbf{r}\,A \to B)$ and hence, by induction hypothesis (b) and since $\mathrm{K}_A \vdash \exists_A$, to $A \to B$.

Consider $(\mu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $(\mu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\mu\,\Phi)(\vec{t})$. We show more generally $\mu\,\mathbf{r}(\Phi) = \mathrm{K}\,(\mu\,\Phi)$. We show the inclusion $\mu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\mu\,\Phi)$ by induction. Hence, we have to show $\mathbf{r}(\Phi)(\mathrm{K}\,(\mu\,\Phi)) \subseteq \mathrm{K}\,(\mu\,\Phi)$, that is, $a\,\mathbf{r}\,A \to A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\mu\,\Phi)$ and $X = \mu\,\Phi$. But the extra assumptions imply $\tilde{X} = \mathrm{K}\,X$. Hence, induction hypothesis (a) applies. The other inclusion, $\mathrm{K}\,(\mu\,\Phi) \subseteq \mu\,\mathbf{r}(\Phi)$, is equivalent to $\mu\,\Phi \subseteq \forall(\mu\,\mathbf{r}(\Phi))$. We show $\mu\,\Phi \subseteq \forall\,(\mu\,\mathbf{r}(\Phi))$ by induction. Hence, we have to show $\Phi(\forall\,(\mu\,\mathbf{r}(\Phi))) \subseteq \forall\,(\mu\,\mathbf{r}(\Phi))$, that is, $A \to a\,\mathbf{r}\,A$ under the extra assumptions $X = \forall\,(\mu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mu\,\mathbf{r}(\Phi)$. Since $\mathrm{K}\,(\forall\,(\mu\,\mathbf{r}(\Phi))) \subseteq \mu\,\mathbf{r}(\Phi)$ and $a\,\mathbf{r}\,A$ is monotone in $\tilde{X}$ (and A is independent of $\tilde{X}$), it suffices to show that $A \to a\,\mathbf{r}\,A$ follows form the extra assumptions $X = \forall\,(\mu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mathrm{K}\,(\forall\,(\mu\,\mathbf{r}(\Phi)))$. But this is guaranteed by induction hypothesis (a).

For coinduction, the proof is obtained by dualization, that is, by inverting all inclusions and implications and replacing μ, "induction", $\forall$ by ν, "coinduction", $\exists$, respectively. More precisely, consider $(\nu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $(\nu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\nu\,\Phi)(\vec{t})$. We show more generally $\nu\,\mathbf{r}(\Phi) = \mathrm{K}\,(\nu\,\Phi)$. We show the inclusion $\mathrm{K}\,(\nu\,\Phi) \subseteq \nu\,\mathbf{r}(\Phi)$ by coinduction. Hence, we have to show $\mathrm{K}\,(\nu\,\Phi) \subseteq \mathbf{r}(\Phi)(\mathrm{K}\,(\nu\,\Phi))$, that is, $A \to a\,\mathbf{r}\,A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\nu\,\Phi)$ and $X = \nu\,\Phi$, which holds by induction hypothesis (a). The other inclusion, $\nu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\nu\,\Phi)$, is equivalent to $\exists(\nu\,\mathbf{r}(\Phi)) \subseteq \nu\,\Phi$, where for a predicate $\mathcal{P}$, $\exists\,\mathcal{P} = \lambda \vec{x} \exists a\ \mathcal{P}(a, \vec{x})$. We show $\exists(\nu\,\mathbf{r}(\Phi)) \subseteq \nu\,\Phi$ by coinduction. Hence, we have to show $\exists\,(\nu\,\mathbf{r}(\Phi)) \subseteq \Phi(\exists\,(\nu\,\mathbf{r}(\Phi)))$, that is, $a\,\mathbf{r}\,A \to A$ under the extra assumptions $X = \exists\,(\nu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \nu\,\mathbf{r}(\Phi)$. Since $\nu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\exists\,(\mu\,\mathbf{r}(\Phi)))$ and $a\,\mathbf{r}\,A$ is monotone in $\tilde{X}$, it suffices to show that $a\,\mathbf{r}\,A \to A$ follows form the extra assumptions $X = \exists\,(\nu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mathrm{K}\,(\exists\,(\nu\,\mathbf{r}(\Phi)))$. But this is guaranteed by induction hypothesis (a).

For (b), the only non-obvious case is induction.

Consider $(\mu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $\exists a\,(\mu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\mu\,\Phi)(\vec{t})$. We show more generally $\exists\,(\mu\,\mathbf{r}(\Phi)) = \mu\,\Phi$. The inclusion $\exists\,(\mu\,\mathbf{r}(\Phi)) \subseteq \mu\,\Phi$ is equivalent to $\mu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\mu\,\Phi)$. We use induction. Hence, we have to show $\mathbf{r}(\Phi)(\mathrm{K}\,(\mu\,\Phi)) \subseteq \mathrm{K}\,(\mu\,\Phi)$, that is, $\exists a\,a\,\mathbf{r}\,A \to A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\mu\,\Phi)$ and $X = \mu\,\Phi$. But the extra assumptions imply $\tilde{X} = \mathrm{K}\,X$ and hence $\exists\,\tilde{X} = X$. Hence, induction hypothesis (b) applies. The other inclusion, $\mu\,\Phi \subseteq \mu\,\exists\,(\mu\,\mathbf{r}(\Phi))$, can be shown by induction. Hence, we show $\Phi(\exists\,(\mu\,\mathbf{r}(\Phi))) \subseteq \exists\,(\mu\,\mathbf{r}(\Phi))$. Since $\exists\,(\mu\,\mathbf{r}(\Phi)) = \exists\,(\mathbf{r}(\Phi)(\mu\,\mathbf{r}(\Phi)))$, this equivalent to $X = \exists\,(\mu\,\mathbf{r}(\Phi)), \tilde{X} = \mu\,\mathbf{r}(\Phi) \vdash A \to \exists a\,a\,\mathbf{r}\,A$. Since the assumptions in this sequent imply $X = \exists\,\tilde{X}$, this is implied by induction hypothesis (b). ◀

Lemma 3
(a) If $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ for some $a, b \in \mathbb{Q}$, then $y \in \mathrm{A}$.
(b) If $x \in \mathrm{C}$ and $b \in \mathbb{Q}$ is such that $x + b \in \mathbb{I}$, then $x + b \in \mathrm{C}$.
(c) If $x \in \mathrm{C}$ and $2x + b \in \mathbb{I}$, where $b \in \mathbb{Q}$, then $2x + b \in \mathrm{C}$.
(d) If $x \in \mathrm{C}$, then $-x \in \mathrm{C}$.

Proof.

(a) Let $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ where $a, b \in \mathbb{Q}$. Let $n \in \mathbb{N}$. Let $k \in \mathbb{N}$ such that $|a| \leq 2^k$ Since $x \in A$ there is $q \in \mathbb{Q}$ such that $|x - q| \leq 2^{-(n+k)}$. Hence

$$|ax + b - (aq + b)| = |a||x - q| \leq 2^k / 2^{n+k} = 2^{-n}$$

Let $q' = -1$ if $aq + b < -1$, $= 1$ if $aq + b > 1$, and $= aq + b$ otherwise. Then $q' \in \mathbb{Q} \cap \mathbb{I}$ and $|ax + b - q'| \leq 2^{-n}$.

(b) Define $P := \{x \in \mathbb{I} \mid \exists b \in \mathbb{Q}\,.\, x + b \in \mathrm{C}\}$. We show $P \subseteq C$ by coinduction. Let $x \in P$, that is $x \in \mathbb{I}$ and $x + b \in \mathrm{C}$ for some $b \in \mathbb{Q}$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in P$. Since $x + b \in \mathrm{C}$ we find $d_0, d_1 \in \mathrm{SD}$ such that $2(x + b) - d_0 \in \mathrm{C}$ and $4(x + b) - 2d_0 - d_1 \in \mathrm{C}$. Hence $|4(x + b) - 2d_0 - d_1| \leq 1$, i.e. $|x - a| \leq 1/4$ where $a := d_0/2 + d_1/4 - b$. Therefore $x \in \mathbb{I} \cap [a - 1/4, a + 1/4]$. Choose $d \in \mathrm{SD}$ such that $\mathbb{I} \cap [a - 1/4, a + 1/4] \subseteq \mathbb{I}_d$. Then $x \in \mathbb{I}_d$. With $c := 2b + d - d_0 \in \mathbb{Q}$ we have $2x - d + c = 2(x + b) - d_0 \in \mathrm{C}$, hence $2x - d \in P$.

(c) If $x \in \mathrm{C}$, then $2x - e \in \mathrm{C}$ for some $e \in \mathrm{SD}$. Hence $2x + b \in \mathrm{C}$, by part (b), provided $2x + b \in \mathbb{I}$.

(d) Let $P(x) = -x \in \mathrm{C}$. We show $P \subseteq \mathrm{C}$, by coinduction. Assume $-x \in \mathrm{C}$. Let $d \in \mathrm{SD}$ such that $-x \in \mathbb{I}_d$ and $2(-x) - d \in \mathrm{C}$. Then $x \in \mathbb{I}_{-d}$ and $-(2x - (-d)) \in \mathrm{C}$, i.e. $P(2x - (-d)$. ◀

Lemma 13. CFP proves $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$ (hence $\mathrm{S}(A^{\mathrm{S}}) \leftrightarrow A^{\mathrm{S}}$) for all formulas A without free predicate variables.

Proof. For a formula A we set $\Gamma_A = \{\mathrm{S}(X) \subseteq X \mid X \text{ is not guarded in } A\}$. Note that if A has no free predicate variables, then $\Gamma_A = \emptyset$.

We show more generally $\Gamma_A \vdash_{\mathrm{CFP}} \mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$, by induction on A.

The *cases where* A^{S} *is of the form* $\mathrm{S}(\ldots)$, that is, $A \vee B$ and $\exists x\, A$, are trivial since the modality is idempotent.

For the *case* $P(\vec{t})$ the assertion holds by the rule $(\mathrm{S}^{\mathrm{nc}})$.

For the *case* $X(\vec{t})$ the assertion holds since X is not guarded in $X(\vec{t})$.

Cases $A \wedge B$ *and* $A \to B$. First note that $\Gamma_{A \wedge B} = \Gamma_{A \to B} = \Gamma_A \cup \Gamma_B$. $\mathrm{S}((A \wedge B)^{\mathrm{S}})$ is $\mathrm{S}(A^{\mathrm{S}} \wedge B^{\mathrm{S}})$, which implies $\mathrm{S}(A^{\mathrm{S}}) \wedge \mathrm{S}(B^{\mathrm{S}})$. By induction hypothesis, this is equivalent to $A^{\mathrm{S}} \wedge B^{\mathrm{S}}$, i.e. $(A \wedge B)^{\mathrm{S}}$. For implication the argument is similar.

Cases $\forall x\, A$. $\mathrm{S}((\forall x\, A)^{\mathrm{S}})$ is $\mathrm{S}(\forall x\, A^{\mathrm{S}})$, which implies $\forall x\, \mathrm{S}(A^{\mathrm{S}})$. By induction hypothesis and the rule (S^{+}), this is equivalent to $\forall x\, A^{\mathrm{S}}$, i.e. $(\forall x\, A)^{\mathrm{S}}$.

Case $(\nu\, \Phi)(\vec{t})$. It suffices to show $\mathrm{S}(\nu\, \Phi^{\mathrm{S}}) \subseteq \nu\, \Phi^{\mathrm{S}}$. Define $\mathcal{P} := \nu\, (\Phi^{\mathrm{S}} \circ S)$. Assuming $\Phi = \lambda X \lambda \vec{x}\,.\, A$, we have $\Gamma_A \subseteq \Gamma_{\nu\, \Phi} \cup \{\mathrm{S}(X) \subseteq X\}$. Hence, by induction hypothesis, $\Gamma_{\nu\, \Phi}, \mathrm{S}(X) \subseteq X \vdash_{\mathrm{CFP}} \mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$. In the following, we reason in CFP and assume $\Gamma_{\nu\, \Phi}$. Since $\mathrm{S}(\mathrm{S}(X)) \subseteq \mathrm{S}(X)$, it follows $\mathrm{S}(A^{\mathrm{S}}[\mathrm{S}(X)/X]) \to A^{\mathrm{S}}[\mathrm{S}(X)/X]$, i.e. $\mathrm{S}((\Phi^{\mathrm{S}} \circ \mathrm{S})(X)) \subseteq (\Phi^{\mathrm{S}} \circ \mathrm{S})(X)$. In particular, for $X := \mathcal{P}$ we obtain $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$, since $(\Phi^{\mathrm{S}} \circ \mathrm{S})(\mathcal{P}) = \mathcal{P}$. Therefore, it suffices to show that $\nu\, \Phi^{\mathrm{S}} = \mathcal{P}$. By (S^{+}), $\Phi^{\mathrm{S}} \subseteq \Phi^{\mathrm{S}} \circ S$, hence $\nu\, \Phi^{\mathrm{S}} \subseteq \mathcal{P}$, by the monotonicity of the greatest fixed point operator. Since $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$ we have $\mathcal{P} = \Phi^{\mathrm{S}}(\mathrm{S}(\mathcal{P})) \subseteq \Phi^{\mathrm{S}}(\mathcal{P})$, by the monotonicity of Φ^{S}. Hence $\mathcal{P} \subseteq \nu\, \Phi^{\mathrm{S}}$, by coinduction.

Case $(\mu\, \Phi)(\vec{t})$ *where* Φ *is not guarded.* $\mathrm{S}(\mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S})) \subseteq \mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S})$ is shown in a similar way as $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$ was shown in the previous case, since there we used only the fixed point property.

Case $(\mu\, \Phi)(\vec{t})$ *where* Φ *is guarded.* We reason in CFP assuming $\Gamma_{\mu\, \Phi}$. Let Φ be $\lambda X \lambda \vec{x}\,.\, A$. Since X is guarded in A, we have $\Gamma_{\mu\, \Phi} = \Gamma_A$. Hence, by induction hypothesis, $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$, i.e. $\mathrm{S}(\Phi^{\mathrm{S}}(X)) \subseteq \Phi^{\mathrm{S}}(X)$. Consequently, $\mathrm{S}(\mu\, \Phi^{\mathrm{S}}) = \mathrm{S}(\Phi^{\mathrm{S}}(\mu\, \Phi^{\mathrm{S}})) \subseteq \Phi^{\mathrm{S}}(\mu\, \Phi^{\mathrm{S}}) = \mu\, \Phi^{\mathrm{S}}$. ◀

Theorem 14 (Concurrent embedding). If $\Gamma \vdash_{\mathrm{IFP}} A$, then $\Gamma^{\mathrm{S}} \vdash_{\mathrm{CFP}} A^{\mathrm{S}}$ for all formulas A without free predicate variables.

Proof. Induction on derivations.

The assumption rule is trivial and the rules for the connectives where the embedding is defined homomorphically, that is, conjunction, implication, universal quantification as well as induction and coinduction, are straightforward using the induction hypothesis.

Disjunction introduction.

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash A^{\mathrm{S}}$. Hence $\Gamma^{\mathrm{S}} \vdash A^{\mathrm{S}} \vee B^{\mathrm{S}}$ by disjunction introduction. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}})$ by the rule (S^+)

Disjunction elimination.

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}})$, $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash C^{\mathrm{S}}$, and $\Gamma^{\mathrm{S}}, B^{\mathrm{S}} \vdash C^{\mathrm{S}}$. By the last two sequents, the rule (S^+) and disjunction elimination, we have $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vee B^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$, by the rule (S^-). With Lemma 13 it follows $\Gamma^{\mathrm{S}} \vdash C^{\mathrm{S}}$.

Existence introduction.

$$\frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x\, A}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash (A[t/x])^{\mathrm{S}}$. By existence introduction and since $(A[t/x])^{\mathrm{S}}$ is the same as $A^{\mathrm{S}}[t/x]$ it follows $\Gamma^{\mathrm{S}} \vdash \exists x\, A^{\mathrm{S}}$. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(\exists x\, A^{\mathrm{S}})$, by the rule (S^+).

Existence elimination.

$$\frac{\Gamma \vdash \exists x\, A \qquad \Gamma, A \vdash C}{\Gamma \vdash C}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(\exists x\, A^{\mathrm{S}})$ and $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash C^{\mathrm{S}}$. Applying the rule (S^+) to the second sequent yields $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$ and furthermore $\Gamma^{\mathrm{S}}, \exists x\, A^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$, using existence elimination (since the embedding and the modality don't introduce new free variables). With the rule (S^-) it follows $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$ and hence $\Gamma^{\mathrm{S}} \vdash C^{\mathrm{S}}$, using Lemma 13. ◀

Theorem 20 (Faithfulness). If $a \in D$ concurrently realizes a data formula A, then $\mathrm{data}(a)$ is nonempty and all $d \in \mathrm{data}(a)$ realize A, provably in IFP.

Proof. For an $n+1$-ary predicate P whose first argument is of type D we define a predicate P' of the same arity by $P'(a, \vec{x}) := \forall d \in \mathrm{data}(a)\, P(d, \vec{x})$, and extend this to predicate substitutions by setting $\theta'(\tilde{X}) := (\theta(\tilde{X}))'$. We show that for a parametric data formula A

$$\mathbf{r}(A^{\mathrm{S}})\theta' \subseteq (\mathbf{r}(A)\theta)'$$

In particular, for a data formula A we have $\mathbf{r}(A^{\mathrm{S}}) \subseteq \mathbf{r}(A)'$, i.e. if a realizes A^{S}, then $\mathbf{r}(A)'(a)$ holds, that is, d realizes A for all $d \in \mathrm{data}(a)$. The proof is by induction on A. We only look at the case $A = \mu\, \Phi\, \vec{t}$, since the other cases are easy. We show by induction $\mu\, (\mathbf{r}(\Phi^{\mathrm{S}})\, \theta') \subseteq P'$ where $P := \mu\, (\mathbf{r}(\Phi)\, \theta)$. Hence we have to show $(\mathbf{r}(\Phi^{\mathrm{S}})\, \theta')\, P' \subseteq P'$:

$$(\mathbf{r}(\Phi^{\mathrm{S}})\, \theta')\, P' = (\mathbf{r}(\Phi^{\mathrm{S}})\, \tilde{X})\, (\theta'[\tilde{X} := P])' \overset{\text{i.h.}}{\subseteq} (\mathbf{r}(\Phi)\, \tilde{X})\, (\theta'[\tilde{X} := P])' = (\mathbf{r}(\Phi)\, \theta)\, P)' = P'\,. \blacktriangleleft$$

Lemma 22 (Correctness).
(a) If $c \longrightarrow v$, then $\vdash \overline{c} = \overline{v}$.
(b) If $c \Longrightarrow d$, then $\vdash d \in \mathrm{data}([\![\overline{c}]\!])$.

Proof. Easy, by induction along the definitions of $c \longrightarrow v$ and $c \Longrightarrow d$. ◀

Lemma 23 (Instantiation).
(a) If $c[i/\alpha] \Longrightarrow d$, then $c \Longrightarrow d$.
(b) If $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \Longrightarrow d$, then $c \Longrightarrow d$.

Proof. Part (a) is proved by induction on i. Part (b) follows immediately from (a). ◀

Lemma 24 (Irreducibility of values). For values v, v'

$$v \longrightarrow v' \quad \text{iff} \quad v = v' .$$

Proof. This follows immediately from the rules of the big-step reduction relation. ◀

Lemma 25 (Monotonicity). If a, b are compact elements in D such that $a \sqsubseteq b$, then $\mathrm{Cl}(a) \supseteq \mathrm{Cl}(b)$.

Proof. Induction on the maximum of $\mathbf{rk}(a)$ and $\mathbf{rk}(b)$. In the case $a = \mathrm{Fam}\,\varphi \sqsubseteq \mathrm{Fam}\,\psi$ with $\varphi\, i \sqsubseteq \psi\, i$ for all $i \in \mathcal{I}$, we have $\mathrm{Cl}(\varphi\, i) \supseteq \mathrm{Cl}(\psi\, i)$, by induction hypothesis. Let $c \in \mathrm{Cl}(b)$. We show $c \in \mathrm{Cl}(a)$. Let $c \longrightarrow (\langle\alpha\rangle M, \eta)$ such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\psi\, i)$ for all $i \in \mathcal{I}$. Hence $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$, which proves that $c \in \mathrm{Cl}(a)$.

The other cases are as in [1], Lemma 12. ◀

Lemma 26 (Printing of data). If $c \in \mathrm{Cl}(a)$ and $d \in \mathrm{data}(a)$, then $c \Longrightarrow d$.

Proof. Induction on the definition of $\mathrm{data}(a)$. In the case $a = \mathrm{Fam}\,\varphi$, the hypotheses of the lemma imply that we have $d \in \mathrm{data}(\varphi\, i_0)$ for some $i_0 \in \mathcal{I}$ and $c \longrightarrow (\langle\alpha\rangle M, \eta)$ such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Therefore, by induction hypothesis, $(M[i_0/\alpha], \eta) \Longrightarrow d$. By Lemma 23, $c \Longrightarrow d$.

The other cases are easy. ◀

Lemma 27 (Reducibilty of closures). $c \in \mathrm{Cl}(a)$ iff $c \longrightarrow v$ for some $v \in \mathrm{Cl}(a)$.

Proof. Induction on $\mathbf{rk}(a)$. We only consider the case $a = \mathrm{Fam}\,\varphi$ since the other cases are as in [1], Lemma 13.

If $c \in \mathrm{Cl}(a)$, then $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Set $v := (\langle\alpha\rangle M, \eta)$, which is a value. By rule (i), $v \longrightarrow v$. Hence $v \in \mathrm{Cl}(a)$, by the above.

Conversely, if $v \in \mathrm{Cl}(a)$, then $v \longrightarrow (\langle\alpha\rangle M, \eta)$ for some α, M, η such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Since $(\langle\alpha\rangle M, \eta)$ is a value, it follows, by Lemma 24, that $v = (\langle\alpha\rangle M, \eta)$. Hence, if $c \longrightarrow v$, then $c \in \mathrm{Cl}(a)$. ◀

Lemma 28 (Approximation). If $\eta \in \mathrm{Cl}(\xi)$, $a \in D_0$ and $a \sqsubseteq [\![M]\!]\xi$, then $(M,\eta) \in \mathrm{Cl}(a)$.

Proof. As in [1], Lemma 15, we replace in the statement of the lemma the value $[\![M]\!]\xi$ by its nth approximation $[\![M]\!]^n\xi$ and show,

$$(+) \qquad \text{for all } n \in \mathbb{N},\ \text{if } \eta \in \mathrm{Cl}(\xi), a \in D_0 \text{ and } a \sqsubseteq [\![M]\!]^n\xi,\ \text{then } (M,\eta) \in \mathrm{Cl}(a),$$

by induction on n. Since $([\![M]\!]^n\xi)_{n\in\mathbb{N}}$ is an increasing sequence with $[\![M]\!]\xi$ as its supremum, it follows that for compact a, $(+)$ is equivalent to the statement of the lemma.

We assume $a \neq \bot$ (for $a = \bot$ the statement is trivial) and look only at the cases where M is formed by one of the new constructs. Since $[\![M]\!]^0\xi = \bot$ we are in the step of the induction. Hence we assume $a \sqsubseteq [\![M]\!]^{n+1}\xi$.

Case $M = \langle\alpha\rangle N$. Then $[\![M]\!]^{n+1}\xi = \mathrm{Fam}\,\varphi$ where $\varphi\, i = [\![N[i/\alpha]]\!]^n\xi$. Since $a \neq \bot$ we have $a = \mathrm{Fam}\,\varphi_0$ where $\varphi_0\, i \sqsubseteq \varphi\, i$ and $\varphi_0\, i$ is compact for all $i \in \mathcal{I}$. Since $v := (M,\eta)$ is a value, we have $v \longrightarrow v$, and, by induction hypothesis, $(N[i/\alpha],\eta) \in \mathrm{Cl}(\varphi_0\, i)$. Hence $(M,\eta) \in \mathrm{Cl}(a)$.

Case $M = \mathrm{case}\ k\ \mathrm{of}\ \{* \to N_*\,;\, \mathrm{L}(\alpha) \to N_0\,;\, \mathrm{R}(\alpha) \to N_1\,;\, (\alpha,\beta) \to N\}$.

Subcase $k = *$. Then $[\![M]\!]^{n+1}\xi = [\![N_*]\!]^n\xi$. By induction hypothesis $(N_*,\eta) \in \mathrm{Cl}(a)$. By Lemma 27, $(N_*,\eta) \longrightarrow v$ for some value $v \in \mathrm{Cl}(a)$. Hence $(M,\eta) \longrightarrow v$.

Subcase $k = \mathrm{L}(i)$ for some $i \in \mathcal{I}$. Then $[\![M]\!]^{n+1}\xi = [\![N_p]\!]^n\xi[\alpha \mapsto i]$. By induction hypothesis $(N_0,\eta[\alpha \mapsto i]) \in \mathrm{Cl}(a)$. By Lemma 27 (implication from left to right), $(N_0,\eta[\alpha \mapsto i]) \longrightarrow v$ for some value $v \in \mathrm{Cl}(a)$. Hence $(M,\eta) \longrightarrow v$, By Lemma 27 (implication from right to left).

Subcase $k = \mathrm{R}(i)$ for some $i \in \mathcal{I}$. Similar to the case above.

Subcase $k = (i,j)$ for some $i,j \in \mathcal{I}$. Similar to the previous case.

Case $M = N\, i_0$. By assumption, $a \sqsubseteq [\![M]\!]^{n+1}\xi = [\![N]\!]^n\xi \cdot i_0$. Since $a \neq \bot$, $[\![N]\!]^n\xi = \mathrm{Fam}\,\varphi$, for some $\varphi \in D^{\mathcal{I}}$. Define $\varphi_0 \in D^{\mathcal{I}}$ by $\varphi_0(i_0) := a$ and $\varphi_0(i) := \bot$ for $i \neq i_0$. Then $\mathrm{Fam}\,\varphi_0$ is compact and $\mathrm{Fam}\,\varphi_0 \sqsubseteq [\![N]\!]^n\xi$. By induction hypothesis, $(N,\eta) \in \mathrm{Cl}(\mathrm{Fam}\,\varphi_0)$. Hence $(N,\eta) \longrightarrow (\langle\alpha\rangle N_0,\eta) \in \mathrm{Cl}(\varphi_0\, i_0) = \mathrm{Cl}(a)$. By Lemma 27, $N_0[i_0/\alpha] \longrightarrow v \in \mathrm{Cl}(a)$ and consequently $N\, i_0 \longrightarrow v$. Therefore $N\, i_0 \in \mathrm{Cl}(a)$. ◀

Lemma 29. If $d \in \mathrm{data}(a)$, then $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq a$.

Proof. Easy induction on the definition of $d \in \mathrm{data}(a)$. We only look at rule (iv), that is, $d \in \mathrm{Fam}\,\varphi$ because $d \in \mathrm{data}(\varphi\, i_0)$ for some $i_0 \in \mathcal{I}$. By induction hypothesis, $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq \varphi\, i_0$. Define $\varphi_0 \in D^{\mathcal{I}}$ by $\varphi_0\, i_0 := a$ and $\varphi_0\, i := \bot$ for $i \neq i_0$. Then $\mathrm{Fam}\,\varphi_0$ is compact and $\mathrm{Fam}\,\varphi_0 \sqsubseteq \mathrm{Fam}\,\varphi$, and, by rule (iv), $d \in \mathrm{data}(\mathrm{Fam}\,\varphi_0)$. ◀