Formalised Inductive Reasoning in the Logic of Bunched Implications

James Brotherston*

Dept. of Computing, Imperial College London

Abstract. We present a framework for inductive definitions in the logic of bunched implications, BI, and formulate two sequent calculus proof systems for inductive reasoning in this framework. The first proof system adopts a traditional approach to inductive proof, extending the usual sequent calculus for predicate BI with explicit induction rules for the inductively defined predicates. The second system allows an alternative mode of reasoning with inductive definitions by *cyclic proof*. In this system, the induction rules are replaced by simple case-split rules, and the proof structures are cyclic graphs formed by identifying some sequent occurrences in a derivation tree. Because such proof structures are not sound in general, we demand that cyclic proofs must additionally satisfy a *global trace condition* that ensures soundness. We illustrate our inductive definition framework and proof systems with simple examples which indicate that, in our setting, cyclic proof may enjoy certain advantages over the traditional induction approach.

1 Introduction

The mechanised verification of properties of computer programs — for example, properties expressing safety, liveness, or correctness — is an important and very challenging problem currently attracting considerable interest in the computer science research community. A source of inconvenience, though, is the tendency of real-life computer programs to be written in low-level languages employing pointer arithmetic and similar operations that directly alter data stored in shared mutable structures, such as the heap. Because the (potentially dangerous) effects of these operations are hard to analyse, programs written using such languages have so far proven very much less amenable to formal reasoning than those written in, e.g., high-level functional programming languages, which are typically more well-behaved from a mathematical standpoint. The logic of *bunched implications* (BI), formulated by O'Hearn and Pym [19], addresses this problem by offering a convenient formalism for expressing properties of programs that access and modify some shared resource [16]. In this paper, we extend BI with a framework for *inductive definitions*, and formulate sequent calculus proof systems for formal reasoning in this extension.

* This research is supported by EPSRC grant EP/E002536/1. My thanks also to Richard Bornat, Cristiano Calcagno, Peter O'Hearn, David Pym, and Hongseok Yang for helpful discussions, references and advice.

Inductive definitions are an important and well-established tool for representing many structures commonly used in the specification of computer programs, such as linked lists and binary trees. For any inductively defined structure, there are naturally associated inductive proof principles allowing us to reason about the structure by exploiting the recursion in its definition. Most often, these principles are encoded as inference rules or axioms in the native reasoning framework; but one can also reason with inductive definitions via a natural mode of *cyclic proof* [8, 9]. In contrast to the usual finite tree proofs, cyclic proofs are regular infinite trees — represented as a finite (cyclic) graph — satisfying a global condition ensuring soundness. For inductively defined relations, this soundness condition is manifested as a generalisation of the principle of infinite descent *à la* Fermat [10].

By considering particular models of BI, one can obtain logics suitable for carrying out verification in specific programming languages. One very successful such logic is the *separation logic* of O'Hearn, Reynolds and Yang, which is suitable for reasoning about C-like languages [21]. Separation logic has to date been used in the verification of several non-trivial programs involving pointer arithmetic, including (but not limited to) a copying garbage collector [6], a DAG duplication program [7] and the Schorr-Waite graph marking algorithm [25]. It has also fruitfully been employed in local shape analysis [13, 14], program termination analysis [4], and automated program verification (see e.g. [3, 2]).

However, as has been noted [5, 22], static analysis in separation logic (and other analysis based upon BI) has so far typically relied upon *ad hoc* extensions of the core logic by the particular inductive definitions needed for the development. It is thus of clear interest to develop a formal extension of BI in which one can define and reason about general inductive structures (over some suitable class of definitions). We provide one such extension, which could form a basis for theorem proving support to check logical implications in BI involving arbitrary inductive predicates (as needed, e.g., to accelerate fixed-point computations in shape analysis [13], or to check properties of predicates generated by inductive recursion synthesis [15] or used in automated verification [18]). Furthermore, our notion of *cyclic proof* for reasoning with inductive predicates appears to offer a new and potentially advantageous approach to certain static analysis applications (which we discuss later). In this paper, however, we confine ourselves to providing the foundations necessary to develop such applications.

In Section 2, we extend first-order predicate BI with a framework for (possibly mutual) inductively defined relations, based on simple "productions" in the style of Martin-Löf [17], in which the multiplicative connectives of BI may occur in the premises of definitions. This framework, though relatively simple, appears nonetheless powerful enough to express the inductive definitions that have arisen in practice in existing applications of separation logic to program verification. In Section 3 we extend the usual Gentzen-style proof system for BI to obtain a proof system supporting induction in the extended logic $\mathrm{BI}_{\mathrm{ID}}$ by adding left- and right-introduction rules for atomic formulas involving the inductive predicates of the theory. Following the approach taken in [8–10], the right introduction rules for an inductive predicate P are merely sequent versions of the productions defining P, while the left-introduction rule for P embodies the natural principle of rule induction over the definition of P. However, there is also a natural notion of cyclic proof for the logic, for which we introduce a second proof system in Section 4. In this system, the induction rules of the first system are replaced by simple *case-split* rules. *Pre-proofs* in the system are “unfinished” derivation trees in which every node to which no proof rule has been applied is identified with a syntactically identical interior node; pre-proofs can thus straightforwardly be understood as cyclic graphs. In general, pre-proofs are not sound, so to ensure soundness we impose a *global trace condition* stipulating, essentially, that for each infinite path in the pre-proof, some inductive definition is unfolded infinitely often along the path. By appealing to the well-foundedness of our inductive definitions, all such paths can be disregarded, whereby the remaining portion of proof is finite and hence sound for standard reasons. Finally, in Section 5, we identify the main directions for future work.

2 First-order predicate BI with inductive definitions

In this section we give the syntax and semantics of our logic, $\mathrm{BI}_{\mathrm{ID}}$, obtained by extending first-order predicate BI à la Biering *et al* [5] with a framework for (possibly mutual) inductive definitions.

A brief comment on some of our mathematical and notational conventions is in order. We often use vector notation to abbreviate sequences, e.g. $\mathbf{x}$ for $(x_1, \ldots, x_n)$; for any $n \in \mathbb{N}$ and $i \leq n$ we define the ith projection function π_i^n on n-tuples of sets by $\pi_i^n(X_1, \ldots, X_n) = X_i$; for any $n \in \mathbb{N}$ we extend set union, intersection and inclusion to n-tuples of sets by their corresponding pointwise definitions; and we write $\mathrm{Pow}(X)$ for the powerset of a set X.

Our languages are the standard (countable) first-order languages — containing arbitrarily many constant, function, and predicate symbols — except that we designate finitely many of the predicate symbols as *inductive*. A predicate symbol that is not inductive is called *ordinary*. For the rest of this paper, we shall consider a fixed language Σ containing exactly n inductive predicates $P_1, \ldots, P_n$, and use $Q_1, Q_2, \ldots$ for ordinary predicates. We also assume the existence of a denumerably infinite set $\mathcal{V}$ of variables, which is disjoint from Σ.

The elements of Σ are interpreted by a structure, as in first-order logic, with the difference here that our structures include a notion of a set of possible resource states or “worlds”, given by a partial commutative monoid. The interpretation of predicates is parameterised by the elements of this monoid: in other words, the set of (tuples of) objects in the domain of which a given predicate is true depends on the current resource state. (However, the interpretations of the constant and function symbols are resource-independent.)

Definition 2.1 (BI-structure). A BI-*structure* for Σ is a tuple:

$$M = (D, \langle R, \circ, e \rangle, \mathbf{c}^M, \mathbf{f}^M, \mathbf{Q}^M, \mathbf{P}^M)$$

where D is a set (called the *domain* of M), $\langle R, \circ, e\rangle$ is a partial commutative monoid and:

- $c^M \in D$ for each constant Σ-symbol $c \in \{\mathbf{c}\}$;
- $f^M : D^k \to D$ for each function Σ-symbol $f \in \{\mathbf{f}\}$ of arity k;
- $Q^M \subseteq R \times D^k$ for each ordinary predicate Σ-symbol $Q \in \{\mathbf{Q}\}$ of arity k;
- $P^M \subseteq R \times D^k$ for each inductive predicate Σ-symbol $P \in \{\mathbf{P}\}$ of arity k;

If $\mathbf{X}$ is an n-tuple of sets satisfying $\pi_i^n(\mathbf{X}) \subseteq R \times D^{k_i}$, where k_i is the arity of P_i, for all $i \in \{1, \ldots, n\}$, then we write $M[\mathbf{P} \mapsto \mathbf{X}]$ to mean the structure defined as M except that $\mathbf{P}^{M[\mathbf{P}\mapsto\mathbf{X}]} = \mathbf{X}$.

Our structures interpret the inductive predicate symbols of Σ only for technical convenience: we shall only be interested later in those structures in which the interpretation of the inductive predicates coincides with the standard interpretation, which is determined by a fixed set of inductive definitions.

The *terms* of Σ are defined as usual; we write $t[u/x]$ to denote the term obtained by substituting the term u for all occurrences of the variable x in the term t. We write $t(x_1, \ldots, x_n)$ for a term t all of whose variables occur in $\{x_1, \ldots, x_n\}$, where $x_1, \ldots, x_n$ are distinct, and in such cases write $t(t_1, \ldots, t_n)$ to denote the term obtained by substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$ respectively in t. Also, if M is a structure with domain D, then $t^M(x_1, \ldots, x_k) : D^k \to D$ is obtained by replacing every constant symbol c by c^M and every function symbol f by f^M in $t(x_1, \ldots, x_n)$.

The formulas of $\mathrm{BI}_{\mathrm{ID}}$ are just the standard formulas of predicate BI[1], given by the following grammar:

$$\begin{aligned} F ::= \;& \top \mid \bot \mid I \mid Q(t_1, \ldots, t_k)\ (k = \text{arity of } Q) \mid t_1 = t_2 \mid \\ & F \wedge F \mid F \vee F \mid F \to F \mid F * F \mid F \mathrel{-\!\!*} F \mid \exists x F \mid \forall x F \end{aligned}$$

where Q ranges over all the predicate symbols of Σ (both inductive and ordinary), x ranges over $\mathcal{V}$ and $t_1, \ldots, t_k$ range over terms of Σ. We use the standard precedences on the logical connectives, with $*$ and $\mathrel{-\!\!*}$ having the same logical precedence as $\wedge$ and $\to$ respectively, and use parentheses to disambiguate where necessary. We write $\neg F$ to abbreviate the formula $F \to \bot$.

As in first-order logic, we interpret variables as elements of the domain D of a BI-structure using environments $\rho : \mathcal{V} \to D$; we extend environments to all terms of Σ in the usual way and write $\rho[x \mapsto d]$ for the environment defined exactly as ρ except that $\rho[x \mapsto d](x) = d$. The formulas of $\mathrm{BI}_{\mathrm{ID}}$ are then interpreted by the following satisfaction (a.k.a. "forcing") relation:

Definition 2.2 (Satisfaction relation for BI**).** Let $M = (D, \langle R, \circ, e\rangle, \ldots)$ be a BI-structure for the language Σ, let $r \in R$ and let ρ be an environment for M. We define the satisfaction relation $M, r \models_\rho F$ on formulas by:

[1] As in [5], our "predicate BI" is propositional BI extended with the usual additive quantifiers $\forall$ and $\exists$, as opposed to propositional BI extended with both additive and multiplicative versions of the quantifiers, as in e.g. [19].

$$\begin{array}{rcl}
M, r \models_\rho \top & \iff & \text{true} \\
M, r \models_\rho \bot & \iff & \text{false} \\
M, r \models_\rho I & \iff & r = e \\
M, r \models_\rho Q\mathbf{t} & \iff & Q^M(r, \rho(\mathbf{t})) \quad (Q \text{ ordinary or inductive}) \\
M, r \models_\rho t_1 = t_2 & \iff & \rho(t_1) = \rho(t_2) \\
M, r \models_\rho F_1 \wedge F_2 & \iff & M, r \models_\rho F_1 \text{ and } M, r \models_\rho F_2 \\
M, r \models_\rho F_1 \vee F_2 & \iff & M, r \models_\rho F_1 \text{ or } M, r \models_\rho F_2 \\
M, r \models_\rho F_1 \rightarrow F_2 & \iff & M, r \models_\rho F_1 \text{ implies } M, r \models_\rho F_2 \\
M, r \models_\rho F_1 * F_2 & \iff & r = r_1 \circ r_2 \text{ and } M, r_1 \models_\rho F_1 \text{ and } M, r_2 \models_\rho F_2 \\
 & & \text{for some } r_1, r_2 \in R \\
M, r \models_\rho F_1 \mathrel{-\!\!*} F_2 & \iff & \text{for all } r' \in R,\ M, r' \models_\rho F_1 \text{ and } r' \circ r \text{ defined} \\
 & & \text{implies } M, r' \circ r \models_\rho F_2 \\
M, r \models_\rho \forall x F & \iff & M, r \models_{\rho[x \mapsto d]} F \text{ for all } d \in D \\
M, r \models_\rho \exists x F & \iff & M, r \models_{\rho[x \mapsto d]} F \text{ for some } d \in D
\end{array}$$

(Informally, $M, r \models_\rho F$ means: "the formula F is true in M in the resource state r and under the environment ρ".)

We now give our schema for (possibly mutual) inductive definitions, which extends the framework used in [8–10] and, like that framework, is based on Martin-Löf's "productions" [17]. Our schema allows the multiplicative connectives of BI to occur in the premises of definitional clauses:

Definition 2.3 (Inductive definition set). An *inductive definition set* for Σ is a set of *productions*, which are rules of the form:

$$\frac{C(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})} \quad i \in \{1, \dots, n\}$$

where $C(\mathbf{x})$ is an *inductive clause* given by the following grammar:

$$\begin{array}{rl}
C(\mathbf{x}) ::= & \top \mid I \mid Q\mathbf{t}(\mathbf{x}) \mid P_j\mathbf{t}(\mathbf{x})\ (j \in \{1, \dots, n\}) \mid t_1(\mathbf{x}) = t_2(\mathbf{x}) \mid \\
 & C(\mathbf{x}) \wedge C(\mathbf{x}) \mid C(\mathbf{x}) * C(\mathbf{x}) \mid \hat{F}(\mathbf{x}) \rightarrow C(\mathbf{x}) \mid \hat{F}(\mathbf{x}) \mathrel{-\!\!*} C(\mathbf{x}) \mid \forall x C(\mathbf{x})
\end{array}$$

where Q ranges over the ordinary predicate symbols of Σ and $\hat{F}(\mathbf{x})$ ranges over all formulas of BI in which no inductive predicate symbols occur and whose free variables are contained in $\{\mathbf{x}\}$.

The productions whose conclusions feature an inductive predicate P should be read as disjunctive clauses of the definition of P, whose free variables are implicitly existentially quantified. For some readers the following, equivalent notation for definitions may be more familiar:

$$P\mathbf{y} =_{def} (\exists \mathbf{x_1}.\, \mathbf{y} = \mathbf{t_1}(\mathbf{x_1}) \wedge C_1(\mathbf{x_1})) \vee \ldots \vee (\exists \mathbf{x_k}.\, \mathbf{y} = \mathbf{t_k}(\mathbf{x_k})) \wedge C_k(\mathbf{x_k}))$$

where $\{\mathbf{y}\} \cap \{\mathbf{x_1}, \dots, \mathbf{x_k}\} = \emptyset$ and $C_1(\mathbf{x_1}), \dots, C_k(\mathbf{x_k})$ are inductive clauses. It is trivial to convert from either form to the other.

As usual, the standard interpretation of the inductive predicate symbols of Σ is obtained by taking the least fixed point of a monotone operator constructed from the definition set Φ:

Definition 2.4 (Definition set operator). Let $M = (D, \langle R, \circ, e\rangle, \ldots)$ be a BI-structure for Σ, let Φ be an inductive definition set for Σ and, for each $i \in \{1, \ldots, n\}$, let k_i be the arity of the inductive predicate symbol P_i. We partition Φ into disjoint subsets $\Phi_1, \ldots, \Phi_n \subseteq \Phi$ by defining each Φ_i to be the set of productions in Φ in whose conclusion P_i occurs. We then index each definition set Φ_i by j, with $j \in \{1, \ldots, |\Phi_i|\}$, and from each production $\Phi_{i,j} \in \Phi_i$, say:

$$\frac{C(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})}$$

we obtain a corresponding n-ary function $\varphi_{i,j} : (\mathrm{Pow}(R \times D^{k_1}) \times \ldots \times \mathrm{Pow}(R \times D^{k_n})) \to \mathrm{Pow}(R \times D^{k_i})$ as follows:

$$\varphi_{i,j}(\mathbf{X}) = \{(r, \mathbf{t}^M(\mathbf{d})) \mid M[\mathbf{P} \mapsto \mathbf{X}], r \models_{\rho[\mathbf{x}\mapsto\mathbf{d}]} C(\mathbf{x})\}$$

(Note that any variables occurring in the right hand side but not the left hand side of the set expression in the definition of $\varphi_{i,j}$ above are, implicitly, existentially quantified over the entire right hand side of the expression.) Then the *definition set operator* for Φ is the operator φ_Φ, with domain and codomain $\mathrm{Pow}(R \times D^{k_1}) \times \ldots \times \mathrm{Pow}(R \times D^{k_n})$, defined by:

$$\varphi_\Phi(\mathbf{X}) = (\bigcup_j \varphi_{1,j}(\mathbf{X}), \ldots, \bigcup_j \varphi_{n,j}(\mathbf{X}))$$

Proposition 2.5. *The operator φ_Φ is monotone (with respect to $\subseteq$).*

Proof. (Sketch) Assuming that $\mathbf{X} \subseteq \mathbf{Y}$, where $\mathbf{X}$ and $\mathbf{Y}$ are n-tuples of sets of the appropriate type, one can prove by structural induction on $C(\mathbf{x})$ that $M[\mathbf{P} \mapsto \mathbf{X}], r \models_{\rho[\mathbf{x}\mapsto\mathbf{d}]} C(\mathbf{x})$ implies $M[\mathbf{P} \mapsto \mathbf{Y}], r \models_{\rho[\mathbf{x}\mapsto\mathbf{d}]} C(\mathbf{x})$. It follows that $\varphi_{i,j}(\mathbf{X}) \subseteq \varphi_{i,j}(\mathbf{Y})$ for any i and j, and thus $\varphi_\Phi(\mathbf{X}) \subseteq \varphi_\Phi(\mathbf{Y})$ as required. □

Example 2.6. Let Φ_N be the inductive definition set consisting of the following productions for a unary inductive predicate N:

$$\frac{\top}{N0} \qquad \frac{Nx}{Nsx}$$

Then the definition set operator for Φ_N is defined by:

$$\varphi_{\Phi_N}(X) = \{(r, 0^M) \mid r \in R\} \cup \{(r, s^M d) \mid (r, d) \in X\}$$

In structures M in which all "numerals" $(s^M)^k 0^M$ for $k \geq 0$ are distinct, the predicate N corresponds to the property of being a natural number.

Example 2.7. Let $\mapsto$ be an ordinary, binary predicate symbol (written infix), and let $\Phi_{\texttt{ls}}$ be the inductive definition set consisting of the following productions for a binary inductive predicate $\texttt{ls}$:

$$\frac{I}{\texttt{ls}\,x\,x} \qquad\qquad \frac{x \mapsto x' * \texttt{ls}\,x'\,y}{\texttt{ls}\,x\,y}$$

Then the definition set operator for $\Phi_{\texttt{ls}}$ is defined by:

$$\begin{aligned}\varphi_{\Phi_{\texttt{ls}}}(X) = {}& \{(e,(d,d)) \mid d \in D\} \\ & \cup \{(r_1 \circ r_2,(d,d')) \mid (r_1,(d,d'')) \in {\mapsto^M} \text{ and } (r_2,(d'',d')) \in X\}\end{aligned}$$

where d'' in the second set comprehension is, implicitly, existentially quantified. In separation logic, where the resource states are heaps and $x \mapsto y$ is true of a heap h if h is a single-celled heap in which x is a pointer to y, the predicate $\texttt{ls}$ is used to represent (possibly cyclic) segments of singly-linked lists, so that $\texttt{ls}\,x\,y$ is true of a heap h if h represents a linked list whose first element is pointed to by x and whose last element contains the pointer y.

It is a standard result for inductive definitions that the least n-tuple of sets closed under the productions in Φ is the least prefixed point of the operator φ_Φ (see e.g. [1]), and that this least prefixed point can be approached in iterative *approximant* stages, as follows:

Definition 2.8 (Approximants). Let Φ be an inductive definition set for Σ, and define a chain of ordinal-indexed sets $(\varphi^\alpha_\Phi)_{\alpha\geq 0}$ by transfinite induction: $\varphi^\alpha_\Phi = \bigcup_{\beta<\alpha} \varphi_\Phi(\varphi^\beta_\Phi)$ (note that this implies $\varphi^0_\Phi = (\emptyset,\ldots,\emptyset)$). Then for each $i \in \{1,\ldots,n\}$, the set $P^\alpha_i = \pi^n_i(\varphi^\alpha_\Phi)$ is called the α^{th} *approximant of* P_i.

Definition 2.9 (Standard model). Let Φ be an inductive definition set for Σ. Then a BI-structure M for Σ is said to be a *standard model* for (Σ,Φ) if $P^M_i = \bigcup_\alpha P^\alpha_i$ for all $i \in \{1,\ldots,n\}$.

Definition 2.9 thus fixes within a BI-structure a standard interpretation of the inductive predicate symbols of Σ that is uniquely determined by the other components of the structure.

Proposition 2.10. *For any inductive definition set Φ* not *employing universal quantification, φ^ω_Φ is a prefixed point of φ_Φ and thus in a standard model of (Σ,Φ) we have $P^M_i = P^\omega_i$ for all $i \in \{1,\ldots,n\}$. If Φ does feature universal quantification, the closure ordinal is $>\omega$ in general.*

Proof. (Sketch) One can show by structural induction on $C(\mathbf{x})$ that, if $C(\mathbf{x})$ contains no occurrences of $\forall$, then $M[\mathbf{P} \mapsto \varphi^\omega_\Phi], r \models_{\rho[\mathbf{x}\mapsto\mathbf{d}]} C(\mathbf{x})$ implies $M[\mathbf{P} \mapsto \varphi^k_\Phi], r \models_{\rho[\mathbf{x}\mapsto\mathbf{d}]} C(\mathbf{x})$ for some $k \in \mathbb{N}$. It follows that $\varphi_{i,j}(\varphi^\omega_\Phi) \subseteq \bigcup_{k\in\mathbb{N}} \varphi_{i,j}(\varphi^k_\Phi)$ for any i and j, and thus that $\varphi_\Phi(\varphi^\omega_\Phi) \subseteq \varphi^\omega_\Phi$ as required.

For the second part of the proposition, consider a BI-structure M with domain $\mathbb{N}$ and in which the Peano axioms hold, and the inductive definition set Φ

consisting of the productions for N in Example 2.6 together with a production with premise $\forall x N x$ and conclusion $P0$ (where P is a unary inductive predicate symbol). Then one can easily verify that the least prefixed point of φ_Φ is $\varphi_\Phi^{\omega+1}$. □

3 A proof system for induction in $\mathrm{BI}_{\mathrm{ID}}$

In this section we give a Gentzen-style proof system suitable for formalising traditional proof by induction in our logic $\mathrm{BI}_{\mathrm{ID}}$. We fix an inductive definition set Φ for Σ, partitioned into $\Phi_1, \ldots, \Phi_n$ as in Defn. 2.4. Our starting point will be the standard sequent calculus for BI (cf. [20]). We write *sequents* of the form $\Gamma \vdash F$, where F is a formula and Γ is a *bunch*, given by the following definition:

Definition 3.1 (Bunch). A *bunch* is a tree whose leaves are labelled by formulas of $\mathrm{BI}_{\mathrm{ID}}$ and whose internal nodes are labelled by ';' or ',' (denoting respectively additive and multiplicative combination).

As our sequents have at most one formula occurring on the right hand side, our proof system is intuitionistic. This is not for ideological reasons but for technical convenience; the formulation of a classical (multiple-conclusion) sequent calculus for BI would necessitate the use of a multiplicative disjunction (for details see [20]).

We write $\Gamma(\Delta)$ to mean that Γ is a bunch of which Δ is a subtree (also called a "sub-bunch"), and write $\Gamma(\Delta')$ for the bunch obtained by replacing the considered instance of Δ by Δ' in $\Gamma(\Delta)$.

Definition 3.2 (Coherent equivalence for bunches). Define $\equiv$ to be the least relation on bunches satisfying:

1. commutative monoid equations for ';' and $\top$;
2. commutative monoid equations for ',' and I;
3. congruence: if $\Delta \equiv \Delta'$ then $\Gamma(\Delta) \equiv \Gamma(\Delta')$.

The usual sequent calculus rules for our version of predicate BI, plus rules for equality and an explicit substitution rule, are given in Figure 1. Our proof system, called $\mathrm{LBI}_{\mathrm{ID}}$, is obtained from this system by adding rules for introducing atomic formulas of the form $P_i\mathbf{t}$, where P_i is an inductive predicate symbol, on the left and right of sequents.

First, for each $i \in \{1, \ldots, n\}$ and each production $\Phi_{i,j} \in \Phi_i$, we obtain a right-introduction rule $(P_i R_j)$ for the predicate P_i as follows:

$$\frac{C(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})} \quad \Longrightarrow \quad \frac{\Gamma \vdash C(\mathbf{u})}{\Gamma \vdash P_i\mathbf{t}(\mathbf{u})}\,(P_i R_j)$$

Before giving the rules for introducing inductive predicates on the left of sequents, we first give a formal definition of what it means for two inductive predicates to have a mutual definition in Φ (repeated from [8]):

Structural rules:

$$\frac{}{F \vdash F}\,(\mathrm{Id}) \qquad \frac{\Gamma(\Delta) \vdash F}{\Gamma(\Delta; \Delta') \vdash F}\,(\mathrm{Weak}) \qquad \frac{\Gamma(\Delta; \Delta) \vdash F}{\Gamma(\Delta) \vdash F}\,(\mathrm{Contr})$$

$$\frac{\Gamma' \vdash F}{\Gamma \vdash F}\;\Gamma \equiv \Gamma'\;\;(\mathrm{Equiv}) \qquad \frac{\Delta \vdash G \qquad \Gamma(G) \vdash F}{\Gamma(\Delta) \vdash F}\,(\mathrm{Cut}) \qquad \frac{\Gamma \vdash F}{\Gamma[\theta] \vdash F[\theta]}\,(\mathrm{Subst})$$

Propositional rules:

$$\frac{}{\bot \vdash F}\,(\bot\mathrm{L}) \qquad \frac{\Gamma(F_1) \vdash F \quad \Gamma(F_2) \vdash F}{\Gamma(F_1 \vee F_2) \vdash F}\,(\vee\mathrm{L}) \qquad \frac{\Gamma(F_1; F_2) \vdash F}{\Gamma(F_1 \wedge F_2) \vdash F}\,(\wedge\mathrm{L})$$

$$\frac{}{\vdash \top}\,(\top\mathrm{R}) \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2}\;\; i \in \{1,2\}\;(\vee\mathrm{R}) \qquad \frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}\,(\wedge\mathrm{R})$$

$$\frac{\Delta \vdash F_1 \quad \Gamma(F_2) \vdash F}{\Gamma(\Delta, F_1 \mathrel{-\!\!*} F_2) \vdash F}\,(\mathrel{-\!\!*}\mathrm{L}) \qquad \frac{\Delta \vdash F_1 \quad \Gamma(\Delta; F_2) \vdash F}{\Gamma(\Delta; F_1 \to F_2) \vdash F}\,(\to\mathrm{L}) \qquad \frac{\Gamma(F_1, F_2) \vdash F}{\Gamma(F_1 * F_2) \vdash F}\,(*\mathrm{L})$$

$$\frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \mathrel{-\!\!*} F_2}\,(\mathrel{-\!\!*}\mathrm{R}) \qquad \frac{\Gamma; F_1 \vdash F_2}{\Gamma \vdash F_1 \to F_2}\,(\to\mathrm{R}) \qquad \frac{\Gamma \vdash F_1 \qquad \Delta \vdash F_2}{\Gamma, \Delta \vdash F_1 * F_2}\,(*\mathrm{R})$$

Quantifier rules:

$$\frac{\Gamma(G[t/x]) \vdash F}{\Gamma(\forall x G) \vdash F}\,(\forall\mathrm{L}) \qquad \frac{\Gamma \vdash F}{\Gamma \vdash \forall x F}\;\; x \notin FV(\Gamma)\;(\forall\mathrm{R})$$

$$\frac{\Gamma(G) \vdash F}{\Gamma(\exists x G) \vdash F}\;\; x \notin FV(\Gamma \cup \{F\})\;(\exists\mathrm{L}) \qquad \frac{\Gamma \vdash F[t/x]}{\Gamma \vdash \exists x F}\,(\exists\mathrm{R})$$

Equality rules:

$$\frac{}{\Gamma \vdash t = t}\,(=\mathrm{R}) \qquad \frac{\Gamma(\top)[u/x, t/y] \vdash F[u/x, t/y]}{\Gamma(t = u)[t/x, u/y] \vdash F[t/x, u/y]}\,(=\mathrm{L})$$

Fig. 1. Sequent calculus proof rules for predicate BI with equality.

Definition 3.3 (Mutual dependency). Define the binary relation Prem on the inductive predicate symbols $\{P_1, \ldots, P_n\}$ of Σ as the least relation satisfying: $\mathit{Prem}(P_i, P_j)$ holds whenever P_j occurs in the premise of some production in Φ_i. Also define Prem^* to be the reflexive-transitive closure of Prem. Then we say two predicate symbols P and Q are *mutually dependent* if both $\mathit{Prem}^*(P, Q)$ and $\mathit{Prem}^*(Q, P)$ hold.

Now to obtain an instance of the induction rule for any inductive predicate P_j, we first associate with every inductive predicate P_i a tuple $\mathbf{z_i}$ of k_i distinct variables (called *induction variables*), where k_i is the arity of P_i. Furthermore, we associate to every predicate P_i that is mutually dependent with P_j a formula (called an *induction hypothesis*) H_i, possibly containing some of the induction variables. Next, define the formula G_i for each $i \in \{1, \ldots, n\}$ by: $G_i = H_i$ if P_i and P_j are mutually dependent, and $G_i = P_i\mathbf{z_i}$ otherwise. For convenience, we shall write $G_i\mathbf{t}$ for $G_i[\mathbf{t}/\mathbf{z_i}]$, where $\mathbf{t}$ is a tuple of k_i terms. Then an instance of the induction rule (Ind P_j) for P_j has the following schema:

$$\frac{\text{minor premises} \qquad \Gamma(H_j\mathbf{t}) \vdash F}{\Gamma(P_j\mathbf{t}) \vdash F} \ (\text{Ind } P_j)$$

where the premise $\Gamma(H_j\mathbf{t}) \vdash F$ is called the *major premise* of the rule, and for each predicate P_i that is mutually dependent with P_j, we obtain a *minor premise* from every production in Φ_i as follows:

$$\frac{C(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})} \qquad \Longrightarrow \qquad C_H(\mathbf{x}) \vdash H_i\mathbf{t}(\mathbf{x}) \quad (\forall x \in \mathbf{x}.\, x \notin FV(\Gamma))$$

where $C_H(\mathbf{x})$ is the formula obtained by replacing every formula of the form $P_k\mathbf{t}(\mathbf{x})$ (for P_k an inductive predicate) by $G_k\mathbf{t}(\mathbf{x})$ in the inductive clause $C(\mathbf{x})$.

Example 3.4. The induction rule for the predicate N from Example 2.6 is:

$$\frac{\top \vdash H0 \quad Hx \vdash Hsx \quad \Gamma(Ht) \vdash F}{\Gamma(Nt) \vdash F} \ (\text{Ind } N)$$

where H is the induction hypothesis associated with N and x is suitably fresh.

Example 3.5. The induction rule for the predicate `ls` from Example 2.7 is:

$$\frac{I \vdash Hxx \quad x \mapsto x' * Hx'y \vdash Hxy \quad \Gamma(Htu) \vdash F}{\Gamma(\mathtt{ls}\, t\, u) \vdash F} \ (\text{Ind } \mathtt{ls})$$

where H is the induction hypothesis associated with `ls` and x, x', y are fresh.

Definition 3.6 (Validity). Let M be a standard model for (Σ, Φ). Then a sequent $\Gamma \vdash F$ is said to be *true in* M if $M, r \models_\rho \phi_\Gamma$ implies $M, r \models_\rho F$ for all environments ρ and resource states r, where ϕ_Γ is the formula obtained by replacing every occurrence of ';' by $\wedge$ and every occurrence of ',' by $*$ in the bunch Γ. $\Gamma \vdash F$ is said to be *valid* if it is true in all standard models.

By a *derivation tree*, we mean a finite tree of sequents in which each parent sequent is obtained as the conclusion of an inference rule with its children as premises. We distinguish between "leaves" and "buds" in the tree. By a *leaf* we mean an axiom, i.e., the conclusion of a 0-premise inference rule. By a *bud* we mean any sequent occurrence in the tree that is not the conclusion of a proof rule. An $\mathrm{LBI}_{\mathrm{ID}}$ *proof* is then, as usual, a finite derivation tree constructed according to the proof rules that contains no buds. The following proposition is a straightforward consequence of the local soundness of our proof rules.

Proposition 3.7 (Soundness of $\mathbf{LBI}_{\mathbf{ID}}$). *If there is an LBI_{ID} proof of $\Gamma \vdash \Delta$ then $\Gamma \vdash \Delta$ is valid.*

Example 3.8. We give an $\mathrm{LBI}_{\mathrm{ID}}$ proof that the predicate N from Example 2.6 admits multiplicative weakening, i.e. that $F, Nx \vdash Nx$:

$$\dfrac{\dfrac{\dfrac{}{F \vdash N0}\,(NR_1)}{\vdash F \mathbin{-\!\!*} N0}\,(\mathbin{-\!\!*}\mathrm{R}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{F \vdash F}\,(\mathrm{Id}) \quad \dfrac{}{Ny \vdash Ny}\,(\mathrm{Id})}{F, F \mathbin{-\!\!*} Ny \vdash Ny}\,(\mathbin{-\!\!*}\mathrm{L})}{F, F \mathbin{-\!\!*} Ny \vdash Nsy}\,(NR_2)}{F \mathbin{-\!\!*} Ny \vdash F \mathbin{-\!\!*} Nsy}\,(\mathbin{-\!\!*}\mathrm{R}) \qquad \dfrac{\dfrac{}{F \vdash F}\,(\mathrm{Id}) \quad \dfrac{}{Nx \vdash Nx}\,(\mathrm{Id})}{F, F \mathbin{-\!\!*} Nx \vdash Nx}\,(\mathbin{-\!\!*}\mathrm{L})}{F, Nx \vdash Nx}\,(\mathrm{Ind}\ N)$$

Note that in the application of (Ind N) in this proof we associate the induction variable z and the induction hypothesis $F \mathbin{-\!\!*} Nz$ with the inductive predicate N. We remark that one can easily see that this example demonstrates the need for generalisation of induction hypotheses in this setting (at least for cut-free proofs): it is clear that no subformula of the root sequent is sufficiently strong as an induction hypothesis to enable us to prove the major premise of the induction.

4 A cyclic proof system for $\mathbf{BI}_{\mathbf{ID}}$

We now define a second proof system $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ for $\mathrm{BI}_{\mathrm{ID}}$ which admits a notion of cyclic proof. *Pre-proofs* are finite derivation trees together with a function assigning to every bud in the tree a syntactically identical interior node (a *companion* for the bud), and thus can be viewed as cyclic graphs. Since pre-proofs are not sound in general, we impose a *global trace condition* on pre-proofs, corresponding to an infinite descent principle for our inductive definitions, to ensure soundness.

The proof rules of the system $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ are the rules of $\mathrm{LBI}_{\mathrm{ID}}$ described in Section 3, except that for each inductive predicate P_j of Σ, the induction rule (Ind P_j) of $\mathrm{LBI}_{\mathrm{ID}}$ is replaced by the *case-split rule*:

$$\dfrac{\text{case distinctions}}{\Gamma(P_j\mathbf{u}) \vdash F}\,(\mathrm{Case}\ P_j)$$

where we obtain a case distinction from each production in Φ_j as follows:

$$\frac{C(\mathbf{x})}{P_j\mathbf{t}(\mathbf{x})} \qquad \Longrightarrow \qquad \Gamma(\mathbf{u} = \mathbf{t}(\mathbf{x}); C(\mathbf{x})) \vdash F \quad (\forall x \in \mathbf{x}.\, x \notin FV(\Gamma \cup \{F\}))$$

Example 4.1. The case-split rule for N from Example 2.6 is:

$$\frac{\Gamma(t = 0; \top) \vdash F \qquad \Gamma(t = sx; Nx) \vdash F}{\Gamma(Nt) \vdash F}\,(\text{Case } N)$$

Example 4.2. The case-split rule for `ls` from Example 2.7 (modulo applying the equality rule to eliminate some generated equalities) is:

$$\frac{\Gamma(t = u; I) \vdash F \qquad \Gamma(t \mapsto x * \mathtt{ls}\, x\, u) \vdash F}{\Gamma(\mathtt{ls}\, t\, u) \vdash F}\,(\text{Case } \mathtt{ls})$$

Definition 4.3 (Companion). Let B be a bud of a derivation tree $\mathcal{D}$. A non-bud sequent C in $\mathcal{D}$ is said to be a *companion* for B if $C = B$.

By assigning a companion to each bud node in a finite derivation tree, one obtains a finite representation of an associated (regular) infinite tree:

Definition 4.4 ($\mathbf{CLBI}^{\omega}_{\mathbf{ID}}$ pre-proof). A *$CLBI^{\omega}_{ID}$ pre-proof* of a sequent $\Gamma \vdash \Delta$ is a pair $\mathcal{P} = (\mathcal{D}, \mathcal{R})$, where $\mathcal{D}$ is a derivation tree constructed according to the proof rules of $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ given above and whose root is $\Gamma \vdash \Delta$, and $\mathcal{R}$ is a function assigning a companion to every bud of $\mathcal{D}$.

We consider $\mathcal{D}$ to have a directed edge from the conclusion of each rule instance to each of its premises, whence the *graph* of $\mathcal{P}$ is the directed graph $\mathcal{G}_{\mathcal{P}}$ obtained from $\mathcal{D}$ by identifying each bud node B in $\mathcal{D}$ with its companion $\mathcal{R}(B)$.

We observe that the local soundness of our proof rules is not sufficient to guarantee that pre-proofs are sound, due to the (possible) cyclicity evident in their graph representations. In order to give a criterion for soundness, we formulate the notion of a *trace* following a path in a pre-proof graph, similar to that used in [8–10, 24] but more complex due to our richer induction schema and use of bunches in sequents:

Definition 4.5 (Trace). Let $\mathcal{P}$ be a $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ pre-proof and let $(\Gamma_i \vdash F_i)_{i\geq 0}$ be a path in $\mathcal{G}_{\mathcal{P}}$. A *trace following* $(\Gamma_i \vdash F_i)_{i\geq 0}$ is a sequence $(\tau_i)_{i\geq 0}$ such that, for all i, τ_i is a leaf of Γ_i (we write F_{τ_i} to mean the formula labelling τ_i in Γ_i.) Furthermore, for each i, one of the following conditions must hold:

1. $\Gamma_i \vdash F_i$ is the conclusion of one of the following inferences, τ_i is the leaf of Γ_i indicated by the underlined formula in the conclusion and τ_{i+1} is one of the leaves of Γ_{i+1} indicated by the underlined formulas in the appropriate premise:

$$\frac{\Gamma(\underline{F_1}; \underline{F_2}) \vdash F}{\Gamma(\underline{F_1 \wedge F_2}) \vdash F}\,(\wedge\text{L}) \quad \frac{\Gamma(\underline{F_1}, \underline{F_2}) \vdash F}{\Gamma(\underline{F_1 * F_2}) \vdash F}\,(*\text{L}) \quad \frac{\ldots \Gamma(\mathbf{u} = \mathbf{t}(\mathbf{x}); \underline{C(\mathbf{x})}) \vdash F \ldots}{\Gamma(\underline{P_j\mathbf{u}}) \vdash F}\,(\text{Case } P_j)$$

$$\frac{\Delta \vdash F_1 \quad \Gamma(\underline{F_2}) \vdash F}{\Gamma(\Delta, \underline{F_1 \mathrel{-\!\!*} F_2}) \vdash F}\ (\mathrel{-\!\!*}\mathrm{L}) \qquad \frac{\Delta \vdash F_1 \quad \Gamma(\Delta; \underline{F_2}) \vdash F}{\Gamma(\Delta; \underline{F_1 \to F_2}) \vdash F}\ (\to\mathrm{L}) \qquad \frac{\Gamma(\underline{G[t/x]}) \vdash F}{\Gamma(\underline{\forall x G}) \vdash F}\ (\forall\mathrm{L})$$

In the case where τ_i and τ_{i+1} are the leaves indicated by the underlined formulas in the displayed instance of (Case P_j) above, i is said to be a *progress point* of the trace. An *infinitely progressing trace* is a trace having infinitely many progress points.

2. τ_{i+1} is the leaf in Γ_{i+1} corresponding to τ_i in Γ_i, modulo any splitting of Γ_i performed by the rule applied with conclusion $\Gamma_i \vdash F_i$. (Thus $F_{\tau_{i+1}} = F_{\tau_i}$, modulo any substitution performed by the rule.) E.g. if $\Gamma_i \vdash F_i$ is the conclusion of the inference:

$$\frac{\Delta \vdash F_1 \quad \Gamma(F_2) \vdash F}{\Gamma(\Delta, F_1 \mathrel{-\!\!*} F_2) \vdash F}\ (\mathrel{-\!\!*}\mathrm{L})$$

then, if $\Gamma_{i+1} \vdash F_{i+1}$ is the left hand premise, then τ_{i+1} and τ_i are the same leaf in Δ and, if $\Gamma_{i+1} \vdash F_{i+1}$ is the right hand premise, then τ_{i+1} and τ_i are the same leaf in $\Gamma(-)$.

Informally, a trace follows (a part of) the construction of an inductively defined predicate occurring in some part of the bunches occurring on a path in a pre-proof. These predicate constructions never become larger as we follow the trace along the path, and at progress points, they actually decrease. This property is encapsulated in the following lemma and motivates the subsequent definition of a *cyclic proof*:

Lemma 4.6. *Let $\mathcal{P}$ be a $CLBI^{\omega}_{ID}$ pre-proof of $\Gamma_0 \vdash F_0$, and let M be a standard model such that $\Gamma_0 \vdash F_0$ is false in M in the resource state r_0 and environment ρ_0 (say). Then there we can construct an infinite path $(\Gamma_i \vdash F_i)_{i \geq 0}$ in $\mathcal{G}_{\mathcal{P}}$ and infinite sequences $(r_i)_{i \geq 0}$ and $(\rho_i)_{i \geq 0}$ such that:*

1. *for all i, $\Gamma_i \vdash F_i$ is false in M_i in the resource state r_i and environment ρ_i;*
2. *if there is a trace $(\tau_i)_{i \geq n}$ following some tail $(\Gamma_i \vdash F_i)_{i \geq n}$ of $(\Gamma_i \vdash F_i)_{i \geq 0}$, then there exists a second sequence of resource states, $(r'_i)_{i \geq n}$, such that $M, r'_i \models_{\rho_i} F_{\tau_i}$ for all $i \geq n$ and the sequence $(\alpha_i)_{i \geq n}$ of ordinals defined by $\alpha_i =$ least α s.t. $M[\mathbf{P} \mapsto \varphi^{\alpha}_{\Phi}], r'_i \models_{\rho_i} F_{\tau_i}$, is non-increasing. Furthermore, if j is a progress point of $(\tau_i)_{i \geq n}$ then $\alpha_{j+1} < \alpha_j$.*

Proof. (Sketch) To construct the required infinite sequences satisfying property 1 of the lemma just requires us to use the fact that the proof rules of $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ are locally sound (i.e., falsifiability of the conclusion of a rule instance implies falsifiability of one of its premises).

For part 2 of the lemma, we suppose there is a trace $(\tau_i)_{i \geq n}$ following the tail $(\Gamma_i \vdash F_i)_{i \geq n}$ of the constructed infinite path. Since $\Gamma_n \vdash F_n$ is false in M under ρ_n and r_n by property 1, it follows that there is a suitable substate r'_n of r_n ("suitability" being given by a formal relation capturing the property that the relationship between r'_n and r_n reflects the position of τ_n in Γ_n) such that $M, r'_n \models_{\rho_n} F_{\tau_n}$, and thus there is a least ordinal α such that $M[\mathbf{P} \mapsto \varphi^{\alpha}_{\Phi}], r'_n \models_{\rho_n} F_{\tau_n}$. Now, given any edge $(\Gamma_k \vdash F_k, \Gamma_{k+1} \vdash F_{k+1})$ in the tail and a suitable substate r'_k of r_k satisfying $M, r'_k \models_{\rho_k} F_{\tau_k}$, one can find a suitable substate r'_{k+1} of r_{k+1} satisfying $M, r'_k \models_{\rho_k} F_{\tau_k}$. Moreover, we have:

$$\text{least } \alpha \text{ s.t. } M[\mathbf{P} \mapsto \varphi^{\alpha}_{\Phi}], r'_{k+1} \models_{\rho} F_{\tau_{k+1}} \leq \text{least } \alpha \text{ s.t. } M[\mathbf{P} \mapsto \varphi^{\alpha}_{\Phi}], r'_k \models_{\rho} F_{\tau_k}$$

Furthermore, if k is a progress point of the trace, then this inequality holds strictly. This property, which can be proven by a lengthy case analysis on the rule with conclusion $\Gamma_k \vdash F_k$ enables us to construct the required sequences $(r'_i)_{i \geq n}$ and $(\alpha_i)_{i \geq n}$.

Definition 4.7 (CLBI$^{\omega}_{\text{ID}}$ proof). A CLBI$^{\omega}_{\text{ID}}$ pre-proof $\mathcal{P} = (\mathcal{D}, \mathcal{R})$ is a *CLBI$^{\omega}_{ID}$ proof* if, for every infinite path in $\mathcal{D}$, there is an infinitely progressing trace following some tail of the path.

Proposition 4.8 (Soundness). *If there is a CLBI$^{\omega}_{ID}$ proof of $\Gamma \vdash \Delta$ then $\Gamma \vdash \Delta$ is valid.*

Proof. If $\Gamma \vdash F$ has a CLBI$^{\omega}_{\text{ID}}$ proof $\mathcal{P}$ but is false in some standard model M then we can use property 1 of Lemma 4.6 to construct an infinite path π in $\mathcal{G}_{\mathcal{P}}$ together with a sequence of environments and resource states that falsify each sequent along the path. Since $\mathcal{P}$ is a proof, there is an infinitely progressing trace following some tail of π. Thus we can invoke property 2 of Lemma 4.6 to create a monotonically decreasing chain of ordinals which, since the trace progresses infinitely often, must decrease infinitely often. This contradicts the well-foundedness of the ordinals, so $\Gamma \vdash F$ must indeed be valid. □

Example 4.9. The following is a CLBI$^{\omega}_{\text{ID}}$ proof of the sequent $F, Nx \vdash Nx$ (recall we gave an LBI$_{\text{ID}}$ proof in Example 3.8):

$$\cfrac{\cfrac{\cfrac{}{F \vdash N0}\,(NR_1)}{F, x = 0 \vdash Nx}\,(=\text{L}) \qquad \cfrac{\cfrac{\cfrac{F, \underline{Nx} \vdash Nx \;\; (\dagger)}{F, \underline{Ny} \vdash Ny}\,(\text{Subst})}{F, \underline{Ny} \vdash Nsy}\,(NR_2)}{F, (x = sy; \underline{Ny}) \vdash Nx}\,(=\text{L})}{F, \underline{Nx} \vdash Nx \;\; (\dagger)}\,(\text{Case } N)$$

We use (†) to indicate the pairing of a suitable companion with the only bud in this pre-proof. To see that it is indeed a CLBI$^{\omega}_{\text{ID}}$ proof, observe that any infinite path π in the pre-proof graph necessarily has a tail consisting of repetitions of the path from the companion to the bud in this pre-proof, and there is a progressing trace following this path, denoted by the underlined formulas (with a progress point at the displayed application of (Case N)). Thus by concatenating copies of this trace we can obtain an infinitely progressing trace on a tail of π as required.

We remark that, unlike the situation for LBI$_{\text{ID}}$ (cf. Example 3.8), we do not require generalisation in this proof, i.e., the invention of new formulas in the proof is not necessary.

Example 4.10. The following is a $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ pre-proof of $\texttt{ls}\,x\,x', \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{ls}\,x\,y \vdash \texttt{ls}\,x\,y}\,(\text{Id})}{I, \texttt{ls}\,x\,y \vdash \texttt{ls}\,x\,y}\,(\equiv)}{(x'=x;I), \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y}\,(=\text{L}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x\mapsto z \vdash x \mapsto z}\,(\text{Id}) \quad \dfrac{(\dagger)\ \underline{\texttt{ls}\,x\,x'}, \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y}{\underline{\texttt{ls}\,z\,x'}, \texttt{ls}\,x'\,y \vdash \texttt{ls}\,z\,y}\,(\text{Subst})}{x\mapsto z, \underline{\texttt{ls}\,z\,x'}, \texttt{ls}\,x'\,y \vdash x\mapsto z * \texttt{ls}\,z\,y}\,({*}\text{R})}{x\mapsto z, \underline{\texttt{ls}\,z\,x'}, \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y}\,(\texttt{ls}R_2)}{\underline{x\mapsto z * \texttt{ls}\,z\,x'}, \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y}\,({*}\text{L})}{(\dagger)\ \underline{\texttt{ls}\,x\,x'}, \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y}\,(\text{Case }\texttt{ls})$$

The pairing of a suitable companion with the only bud in this pre-proof is again denoted by (†). A trace from the companion to the bud is denoted by the underlined formulas, with a progress point at the displayed application of (Case $\texttt{ls}$). As in the previous example, there is only one infinite path, and one can easily observe that the required infinitely progressing trace is obtained by concatenating copies of the displayed trace. So this pre-proof is indeed a proof.

We remark that the standard $\mathrm{LBI}_{\mathrm{ID}}$ proof of $\texttt{ls}\,x\,x', \texttt{ls}\,x'\,y \vdash \texttt{ls}\,x\,y$ proceeds by induction on $\texttt{ls}\,x\,x'$ using the induction variables z, z' (say) and the induction hypothesis $\texttt{ls}\,z'\,y \mathrel{-\!\!*} \texttt{ls}\,z\,y$, thus requiring a generalisation similar to that needed in Example 3.8.

Proposition 4.11. *It is decidable whether a $\mathit{CLBI}^{\omega}_{\mathit{ID}}$ pre-proof is a proof.*

Proof. (Sketch) The property of every infinite path posessing an infinitely progressing trace along a tail is an ω-regular property, and hence reducible to the emptiness of a Büchi automaton. A full proof (for a general notion of trace) appears in [9]; a similar argument appears in [24]. □

5 Conclusions and future work

In this paper, we extend BI with a fairly general class of inductive definitions, and develop sequent calculus proof systems for formal reasoning in the resultant extension $\mathrm{BI}_{\mathrm{ID}}$, as is needed in order to develop proper theorem proving support for inductive reasoning in separation logic. We hope that the formal framework(s) we present here will be of use to researchers in static analysis in providing a sound foundation for logical reasoning in future program verification applications employing inductively defined predicates (in a BI / separation logic context). In a technical sense, our contribution is a reasonably straightforward extension of the framework for inductive definitions and corresponding proof systems given in our previous work for first-order logic with inductively defined relations [8–10]. Thus one might reasonably hope that the key proof-theoretic results from that work, including appropriate completeness and cut-elimination theorems, will also extend to the systems we consider here. Certainly we expect that our cyclic proof system $\mathrm{CLBI}^{\omega}_{\mathrm{ID}}$ subsumes the induction system $\mathrm{LBI}_{\mathrm{ID}}$ (although we have not checked this in detail), with the question of their equivalence presenting similar difficulties to those discussed in [8–10]; in the setting of first-order logic with inductively defined relations, we have conjectured but not yet proven the equivalence of the two proof styles.

It is worth remarking that our logic $\mathrm{BI}_{\mathrm{ID}}$ and the corresponding proof systems should be straightforwardly extensible to a more powerful definitional framework than the one we give here, for example by allowing inductive predicates to occur "negatively" in inductive definitions, subject to an appropriate stratification of predicates to ensure monotonicity as in iterated inductive definitions (cf. [17]).

One particularly promising avenue for further development is the development of static analysis applications based upon cyclic proof in separation logic. For example, our current work with Calcagno and Bornat develops a calculus for giving cyclic proofs of program termination in a (very) simple programming language, based upon a proof system of Hoare judgements which express termination from a given program point and under a given precondition [11]. We hope that it will also be possible to directly formulate cyclic proof systems for the verification of other properties. For example, such systems might use appropriate cyclic proof principles to establish invariants for the looping constructs in programs, or, given such invariants, to prove appropriate postconditions. An important factor related to such developments is the potential of cyclic proof for automated proof search. We have seen simple examples in which cyclic proof avoids the generalisation apparently necessary in the corresponding inductive proof (Examples 4.9 and 4.10); more generally, cyclic proof should offer a "least-commitment" approach to proof search, whereby the induction schema, variables and hypotheses are not chosen at the beginning of the proof, as in traditional inductive theorem proving, but are eventually selected implicitly via the satisfaction of the soundness condition. It would be interesting to examine the implications of these phenomena for proof search; we have previously given proof-theoretic machinery for analysing and manipulating the structure of general cyclic proofs [8, 9] which may be of assistance in such investigations.

References

1. Peter Aczel. An introduction to inductive definitions. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, 1977.
2. Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. Smallfoot: Modular automatic assertion checking with separation logic. In *Proceedings of FMCO 2005*, volume 4111 of *LNCS*, pages 115–137, 2005.
3. Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. Symbolic execution with separation logic. In *Proceedings of APLAS 2005*, volume 3780 of *LNCS*, pages 52–68. Springer, 2005.
4. Josh Berdine, Byron Cook, Dino Distefano, and Peter W. O'Hearn. Automatic termination proofs for programs with shape-shifting heaps. In *Proceedings of 18th CAV*, volume 4144 of *LNCS*, pages 386–400. Springer, 2006.
5. Bodil Biering, Lars Birkedal, and Noah Torp-Smith. BI hyperdoctrines and separation logic. In *Proceedings of ESOP'05*, pages 233–247. Springer-Verlag, 2005.
6. L. Birkedal, N. Torp-Smith, and J.C. Reynolds. Local reasoning about a copying garbage collector. In *Proceedings of POPL'04*, pages 220–231, 2004.

7. Richard Bornat, Cristiano Calcagno, and Peter O'Hearn. Local reasoning, separation and aliasing. In *Proceedings of SPACE'04*, January 2004.
8. James Brotherston. Cyclic proofs for first-order logic with inductive definitions. In *Proceedings of TABLEAUX 2005*, volume 3702 of *LNAI*, pages 78–92. Springer-Verlag, 2005.
9. James Brotherston. *Sequent Calculus Proof Systems for Inductive Definitions*. PhD thesis, University of Edinburgh, November 2006.
10. James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. To appear in *Proceedings of LICS-22*, 2007.
11. James Brotherston, Cristiano Calcagno and Richard Bornat. Cyclic proofs of termination in separation logic. Forthcoming.
12. Alan Bundy. The automation of proof by mathematical induction. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 13, pages 845–911. Elsevier Science, 2001.
13. Cristiano Calcagno, Dino Distefano, Peter W. O'Hearn, and Hongseok Yang. Beyond reachability: Shape abstraction in the presence of pointer arithmetic. In *Proceedings of SAS-13*, volume 4134 of *LNCS*, pages 182–203. Springer, 2006.
14. Dino Distefano, Peter W. O'Hearn, and Hongseok Yang. A local shape analysis based on separation logic. In *Proceedings of TACAS-12*, volume 3920 of *LNCS*, pages 287–302, 2006.
15. Bolei Guo, Neil Vachharajani and David I. August. Shape analysis with inductive recursion synthesis. To appear in *Proceedings of PLDI'07*, June 2007.
16. Samin Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In *Proceedings of POPL'01*, January 2001.
17. Per Martin-Löf. Haupstatz for the intuitionistic theory of iterated inductive definitions. In J.E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 179–216. North-Holland, 1971.
18. Huu Hai Nguyen, Cristina David, Shengchao Qin and Wei-Ngan Chin. Automated verification of shape and size properties via separation logic. *Proceedings of VMCAI'07*, January 2007.
19. P.W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
20. David Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series. Kluwer, 2002.
21. John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of LICS-17*, 2002.
22. Élodie-Jane Sims. Extending separation logic with fixpoints and postponed substitution. *Theoretical Computer Science*, 351(2):258–275, 2006.
23. Christoph Sprenger and Mads Dam. On the structure of inductive reasoning: circular and tree-shaped proofs in the μ-calculus. In *Proceedings of FOSSACS 2003*, volume 2620 of LNCS, pages 425–440, 2003.
24. Christoph Sprenger and Mads Dam. A note on global induction mechanisms in a μ-calculus with explicit approximations. *Theoretical Informatics and Applications*, July 2003.
25. Hongseok Yang. An example of local reasoning in BI pointer logic: the Schorr-Waite graph marking algorithm. In *Proceedings of SPACE 2001*, 2001.