Proof Systems for Retracts in Simply Typed Lambda Calculus

Citation for published version:

Digital Object Identifier (DOI):
10.1007/978-3-642-39212-2_36

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Automata, Languages, and Programming

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Proof Systems for Retracts in Simply Typed Lambda Calculus

Colin Stirling
School of Informatics
University of Edinburgh
cps@inf.ed.ac.uk

Abstract. This paper concerns retracts in simply typed lambda calculus assuming \( \beta\eta \)-equality. We provide a simple tableau proof system which characterises when a type is a retract of another type and which leads to an exponential decision procedure.

1 Introduction

Type \( \rho \) is a retract of type \( \tau \) if there are functions \( C : \rho \to \tau \) and \( D : \tau \to \rho \) with \( D \circ C = \lambda x.x \). This paper concerns retracts in the case of simply typed lambda calculus [1]. Various questions can be asked. The decision problem is: given \( \rho \) and \( \tau \), is \( \rho \) a retract of \( \tau \)? Is there an independent characterisation of when \( \rho \) is a retract of \( \tau \)? Is there an inductive method, such as a proof system, for deriving assertions of the form “\( \rho \) is a retract of \( \tau \)”? If so, can one also construct (inductively) the witness functions \( C \) and \( D \)?

Bruce and Longo [2] provide a simple proof system that solves when there are retracts in the case that \( D \circ C = \beta \lambda x.x \). The problem is considerably more difficult if \( \beta \)-equality is replaced with \( \beta\eta \)-equality. De Liguoro, Piperno and Statman [3] show that the retract relation with respect to \( \beta\eta \)-equality coincides with the surjection relation: \( \rho \) is a retract of \( \tau \) iff for any model there is a surjection from \( \tau \) to \( \rho \). They also provide a proof system for the affine case (when each variable in \( C \) and \( D \) occurs at most once) assuming a single ground type. Regnier and Urzyczyn [9] extend this proof system to cover multiple ground types. The proof systems yield simple inductive nondeterministic algorithms belonging to NP for deciding whether \( \rho \) is an affine retract of \( \tau \). Schubert [10] shows that the problem of affine retraction is NP-complete and how to derive witnesses \( C \) and \( D \) from the proof system in [9]. Under the assumption of a single ground type, decidability of when \( \rho \) is a retract of \( \tau \) is shown by Padovani [8] by explicit witness construction (rather than by a proof system) of a special form.

More generally, decidability of the retract problem follows from decidability of higher-order matching in simply typed lambda calculus [13]: \( \rho \) is a retract of \( \tau \) iff the equation \( \lambda z^\rho. x_1^{\rho \to \tau}(x_2^\rho \to z) =_{\beta\eta} \lambda z^\rho. z \) has a solution (the witnesses \( D \) and \( C \) for \( x_1, x_2 \)). Since the complexity of matching is non-elementary [15] this decidability result leaves open whether there is a better algorithm, or even a proof

---

1 For a full version see http://www.homepages.inf.ed.ac.uk/cps/ret.pdf
system, for the problem. In the case of $\beta$-equality matching is no guide to solvability: the retract problem is simply solvable whereas $\beta$-matching is undecidable [4].

In this paper we provide an independent solution to the retract problem. We show it is decidable by exhibiting sound and complete tableau proof systems. We develop two proof systems for retracts, one for the (slightly easier) case when there is a single ground type and the other for when there are multiple ground types. Both proof systems appeal to paths in terms. Their correctness depend on properties of such paths. We appeal to a dialogue game between witnesses of a retract to prove such properties: a similar game-theoretic characterisation of $\beta$-reduction underlies decidability of matching.

In Section 2 we introduce retracts in simply typed lambda calculus and fix some notation for terms as trees and for their paths. The two tableau proof systems for retracts are presented in Section 3 where we also briefly examine how they generate a decision procedure for the retract problem. In Section 4 we sketch the proof of soundness of the tableau proof systems (and completeness and further details are provided in the full version).

2 Preliminaries

Simple types are generated from ground types using the binary function operator $\to$. We let $a, b, a, \ldots$ range over ground types and $\rho, \sigma, \tau, \ldots$ range over simple types. Assuming $\to$ associates to the right, so $\rho \to \sigma \to \tau$ is $\rho \to (\sigma \to \tau)$, if a type $\rho$ is not a ground type then it has the form $\rho_1 \to \ldots \to \rho_n \to a$. We say that $a$ is the target type of $a$ and of any type $\rho_1 \to \ldots \to \rho_n \to a$. Simply typed terms in Church style are generated from a countable set of typed variables $x^a$ using lambda abstraction and function application [1]. We write $S^\sigma$, or sometimes $S : \sigma$, to mean term $S$ has type $\sigma$. The usual typing rules hold: if $S^\tau$ then $\lambda x^\sigma.S^\sigma : \sigma \to \tau$; if $S^{\sigma \to \tau}$ and $U^\sigma$ then $(S^{\sigma \to \tau}U^\sigma) : \tau$. In a sequence of unparenthesised applications we assume that application associates to the left, so $SU_1 \ldots U_k$ is $((\ldots (SU_1) \ldots)U_k)$. Another abbreviation is $\lambda z_1 \ldots z_m$ for $\lambda z_1 \ldots \lambda z_m$. Usual definitions of when a variable occurrence is free or bound and when a term is closed are assumed.

We also assume the usual dynamics of $\beta$ and $\eta$-reductions and the consequent $\beta\eta$-equivalence between terms (as well as $\alpha$-equivalence). Confluence and strong normalisation ensure that terms reduce to (unique) normal forms. Moreover, we assume the standard notion of $\eta$-long $\beta$-normal form (a term in normal form which is not an $\eta$-reduct of some other term) which we abbreviate to lnf. The syntax of such terms reflects their type: a lnf of type $a$ is a variable $x^a$, or $x U_1 \ldots U_k$ where $x^{\rho_1 \ldots \rho_k \to \sigma}$ and each $U_i^{\rho_i}$ is a lnf; a lnf of type $\rho_1 \to \ldots \to \rho_n \to a$ has the form $\lambda x_1^{\rho_1} \ldots x_n^{\rho_n}S$, where $S^a$ is a lnf.

The following definition introduces retracts between types [2, 3].

Definition 1. Type $\rho$ is a retract of type $\tau$, written $\models \rho \preceq \tau$, if there are terms $C : \rho \to \tau$ and $D : \tau \to \rho$ such that $D \circ C =_{\beta\eta} \lambda x^\rho.x$. 
The witnesses $C$ and $D$ to a retract can always be presented as lns. We can think of $C$ as a “coder” and $D$ as a “decoder” [9]. Assume $\rho = \rho_1 \to \ldots \to \rho_1 \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$: in a retract the types must share target type [9]. We instantiate the bound $\rho_1$ variables in a decoder $D$ to $D(z_1^0, \ldots, z_n^\omega)$, often abbreviated to $D(\overline{x})$, and the bound variable of type $\rho$ in $C$ to $C(x^\rho)$: so, $\models \rho \leq \tau$ if $D(z_1^\omega, \ldots, z_n^\omega)(C(x^\rho)) = \beta_\rho x z_1 \ldots z_l$. From [9], we can restrict a decoder to be of the form $\lambda f^\tau. f S_1^\tau \ldots S_n^\tau$ with $f$ as head variable and a coder $C(x)$ has the form $\lambda y_1^\tau \ldots y_n^\tau. H(x T_l^\rho \ldots T_l^\rho)$.

**Definition 2.** We say that the decoder $D(z_1, \ldots, z_l) = \lambda f^\tau. f S_1^\tau \ldots S_l^\tau$ and the coder $C(x) = \lambda y_1^\tau \ldots y_n^\tau. H(x T_l^\rho \ldots T_l^\rho)$ are canonical witnesses for $\rho \leq \tau$ if $D(\overline{x}) (C(x)) = \beta_\rho x z_1 \ldots z_l$ and they obey the following properties:

1. variables $f, z_1, \ldots, z_l$ occur only once in $D(\overline{x})$,
2. $x$ occurs only once in $C(x)$,
3. $H$ is $\epsilon$ if $\rho$ and $\tau$ are constructed from a single ground type,
4. if $T_i^\rho$ contains an occurrence of $y_j$ then it is the head variable of $T_i^\rho$, $z_i$ occurs in $S_i^\tau$ and $T_i^\rho$ contains no other occurrences of any $y_k$, $1 \leq k \leq n$.

The next result follows from observations in [3, 9].

**Proposition 1.** $\models \rho \leq \tau$ iff there exist canonical witnesses for $\rho \leq \tau$.

So, if there is only a single ground type then $C(x)$ can be restricted to have the form $\lambda y_1^\tau \ldots y_n^\tau. x T_1^\rho \ldots T_l^\rho$ with $x$ as head variable [3].

**Example 1.** From [3]. Let $\rho = \rho_1 \to \rho_2 \to o$ where $\rho_1 = \rho_2 = \sigma \to o$ and let $\tau = \tau_1 \to o$ where $\tau_1 = \sigma \to (o \to o \to o) \to o$ and $\sigma$ is arbitrary. It follows that $\models \rho \leq \tau$. A decoder $D(z_1^\omega, z_2^\omega)$ is $\lambda f^\tau. f (\lambda u^\sigma v^{\omega^\sigma o^o} w(z_1 u) (z_2 u))$ and a coder $C(x)$ is $\lambda y_1^\tau \ldots y_n^\tau. H(x T_l^\rho \ldots T_l^\rho)$; so, $(D(z_1, z_2)) C(x) \to^{\beta_\rho} x (\lambda w^\sigma. z_1 w) \to^{\beta_\rho} x \to^{\beta_\rho} x z_1 z_2$. $\square$

**Example 2.** From [9] with multiple ground types. Let $\rho = \rho_1 \to \rho_2 \to a$ where $\rho_1 = b \to a$, $\rho_2 = a$ and let $\tau = \tau_1 \to a$ where $\tau_1 = b \to (a \to o \to a) \to a$. A decoder is $D(z_1^\omega, z_2^\omega) = \lambda f^\tau. f (\lambda w_1^\delta w_2^{\omega^\delta o^o} u z_1 u_1 z_2)$ and a coder $C(x)$ is $\lambda y^\tau \ldots y^\tau. H(x (\lambda w_1^\delta w_2^\omega. x (\lambda v^\sigma. y (\lambda w_1^\delta w_2^\omega. w)) w))$; so, $(D(z_1, z_2)) C(x) \to^{\beta_\rho} x (\lambda v^\omega. z_1 v) z_2 \to^{\beta_\rho} x z_1 z_2$. $\square$

Terms are represented as special kinds of tree (that we call binding trees in [12, 14]) with dummy lambdas and an explicit binding relation. A term of the form $y^o$ is represented as a tree with a single node labelled $y^o$. In the case of $y U_1 \ldots U_k$, when $y^o_1 \ldots y^o_k \to a$, we assume that a dummy lambda with the empty sequence of variables is placed directly above any subterm $U_i$ in its tree representation if $\rho_i$ is a ground type. With this understanding, the tree for $y U_1 \ldots U_k$ consists of a root node labelled $y^o_1 \ldots y^o_k \to a$ and $k$-successor trees representing $U_1, \ldots, U_k$. We also use the abbreviation $\lambda y$ for $\lambda y_1 \ldots y_m$ for $m \geq 0$, so $\lambda y$ is possibly the empty sequence of variables in the case of a dummy lambda. The
tree representation of $\lambda f. S : \rho_1 \rightarrow \ldots \rightarrow \rho_k \rightarrow a$ consists of a root node labelled $\lambda f$ and a single successor tree for $S^a$. The trees for $C(x)$ and $D(z_1, z_2)$ of Example 1, where we have omitted the types, are in Figure 1.

We say that a node is a lambda (variable) node if it is labelled with a lambda abstraction (variable). The type (target type) of a variable node is the type (target type) of the variable at that node and the type (target type) of a lambda node is the type (target type) of the subterm rooted at that node.

The other elaboration is that we assume an extra binary relation ↓ between nodes in a tree that represents binding; that is, between a node labelled $\lambda y_1 \ldots y_n$ and a node below it labelled $y_j$ (that it binds). A binder $\lambda y$ is such that either $y$ is empty and therefore is a dummy lambda and cannot bind a variable occurrence or $y = y_1 \ldots y_k$ and $\lambda y$ can only then bind variable occurrences of the form $y_i$, $1 \leq i \leq k$. Consequently, we also employ the following abbreviation $n \downarrow m$ if $n \downarrow m$ and $n$ is labelled $\lambda y_1 \ldots y_k$ and $m$ is labelled $y_i$. In Figure 1 we have not included the binding relation; however, for instance, $(2) \downarrow (7)$.

**Definition 3.** Lambda node $n$ is a descendant ($k$-descendant) of $m$ if either $m \downarrow m'$ ($m \downarrow_k m'$), $n$ is a successor of $m'$ for some $m'$ and the target types of $m, m'$ and $n$ are the same, or $n'$ is a descendant ($k$-descendant) of $m$ and $n$ is a descendant of $n'$ for some $n'$.
We assume a standard presentation of nodes of a tree as sequences of integers: an initial sequence, typically e, is the root node; if n is a node and m is the ith successor of n then m = ni. For the sake of brevity we have not followed this approach in Figure 1 where we have presented each node as a unique integer (i).

**Definition 4.** A path of the tree of a term of type σ is a sequence of nodes \( \pi = n_1, \ldots, n_k \) where \( n_1 \) is the root of the tree, each \( n_{i+1} \) is a successor of \( n_i \) and if \( n_j \) is a variable node then for some \( i < j \), \( n_i \downarrow n_j \) (hence a closed path).

For paths \( \pi = m_1, \ldots, m_i \) and \( \pi = n_1, \ldots, n_k \) of type σ we write \( \pi \sqsubseteq \pi \) if for some \( i > 0 \), for all \( h \leq 2i \), \( m_h = n_h \), \( m_{2i+1} = m_{2i}p \), \( n_{2i+1} = n_{2i}q \) and \( p < q \).

A (closed) subtree of a tree of a term of type σ is a set of paths P of type σ such that if \( \pi, \bar{\pi} \) are distinct paths in P then \( \pi \sqsubseteq \bar{\pi} \) or \( \pi \sqsubset \bar{\pi} \).

A path \( \pi = n_1, \ldots, n_k \) is a k-minimal provided that for each binding node \( n_i \) there are at most \( k \) distinct nodes \( n_j, i < j \leq l \), such that \( n_i \downarrow n_j \). A subtree P is k-minimal if each path in P is k-minimal.

Not every path or subtree is useful in a term. So, we define when a path or subtree is realisable meaning that their nodes are “accessible” [7] or “reachable” [6] in an applicative context.

**Definition 6.** Assume \( \pi = n_1, \ldots, n_l \) is a path of odd length of a closed term \( T \) of type σ, \( m \) is the node below \( n_l \) in \( T \) and \( T' \) is the term \( T \) when the variable \( u^r \) at node \( m \) is replaced with a fresh free variable \( z^r \). We say that \( \pi \) is realisable if there is a closed term \( U = \lambda y^r yS_1 \ldots S_k \) such that \( UT' = \beta_\eta \lambda x.\bar{z}W_1 \ldots W_q \) for some \( q \geq 0 \).

**Definition 7.** Assume \( P \) is a subtree of closed term \( T \) of type σ where each path has even length, \( m_1, \ldots, m_q \) are the leaves of \( P \) and \( T_l, 1 \leq i \leq q \), is the term \( T \) when the variable \( u^i \) at \( m_i \) is replaced with a fresh free variable \( z^i \). We say that \( P \) is realisable if there is a closed term \( U = \lambda y^i yS_1 \ldots S_k \) such that for each \( i, UT_i = \beta_\eta \lambda x.\bar{z}_iW_1 \ldots W_q \), for \( q_i \geq 0 \).

Next we define two useful operations on paths, restriction relative to a suffix and the subtype after a prefix.

**Definition 8.** Assume that \( \pi = n_1, \ldots, n_p \) is a path, \( \sigma = \sigma_1 \rightarrow \ldots \rightarrow \sigma_k \rightarrow a \), \( n_i \) is a lambda node of type \( \sigma \) and \( w = n_1, \ldots, n_p \) is a suffix of \( \pi \).

1. The suffix \( w \) admits \( \sigma_j \), \( 1 \leq j \leq k \), if either there is no \( n_q \), \( i \leq q \leq p \), such that \( n_i \downarrow n_q \) or there is a j-descendant \( n_q \) of \( n_i \) whose type is \( \tau_l \rightarrow \ldots \rightarrow \tau_r \rightarrow a \) and for some \( r \) there is not a \( t : q < t \leq p \) such that \( n_q \downarrow n_t \) and \( a \) is the target type of \( \tau_r \).
2. The restriction of \( \sigma \) to \( w \), \( \sigma \mid w \), is defined as \( \sigma_w \) where
- \( \sigma_w = \alpha \),
- \( (\sigma_1 \rightarrow \ldots \rightarrow \sigma_k \rightarrow \alpha)_w = \text{if } w \text{ admits } \sigma_j \text{ then } \sigma_j \rightarrow (\sigma_{j+1} \rightarrow \ldots \rightarrow \sigma_k \rightarrow \alpha)_w \) else \( (\sigma_{j+1} \rightarrow \ldots \rightarrow \sigma_k \rightarrow \alpha)_w \).

**Definition 9.** Assume that \( \pi = n_1, \ldots, n_p \) is a path of type \( \sigma \). For a prefix \( w \) of \( \pi \) we define the subtype of \( \sigma \) after \( w \), \( w(\sigma) \):
- if \( w = \varepsilon \) (the empty prefix) then \( \sigma \),
- if \( w = n_1, \ldots, n_q \), \( q \leq p \), then the type of node \( n_q \).

We also define a canonical presentation of a (prefix or suffix of a) path \( \pi = n_1, \ldots, n_k \) of type \( \sigma \) as a word \( w \). If \( w \) is the empty prefix we write \( w = \varepsilon \). Otherwise, \( w = (w_1, \ldots, w_j) \), \( j \leq k \), where for each \( i \geq 0 \), \( w_{2i+1} = n_{2i+1} \) and if \( n_k \downarrow_i \downarrow_{n_2} \), then \( w_{2i} = n_k m \). Thus, we distinguish between \( w = \varepsilon \) (the empty word) and \( w = (\varepsilon) \) the prefix of length 1 consisting of the root node. Also, we can present a subtree as a set of words. Words will occur in our proof systems as presentations of paths. For example, \( w = (\varepsilon, 1, 11, 112, 1112) \) of type \( \tau \) as in Figure 1 represents the path labelled \( \lambda f, f, \lambda uw, v, \lambda \) of \( D(z_1, z_2) \) in Figure 1 when its root is \( \varepsilon \). To illustrate Definitions 8 and 9, for the prefix \( w' = (\varepsilon, 1, 11) \) and the suffix \( w'' = (11, 112, 1112) \) of \( w \) we have \( w' = \tau_1 \) where \( \tau_1 = \sigma \rightarrow (o \rightarrow o \rightarrow o) \rightarrow o \) as in Example 1 and \( \tau_1 \mid w'' = \sigma \rightarrow \alpha \). Word \( w'' \) of type \( \tau_1 \) has labelling \( \lambda u^o v^o a^o \rightarrow u, v, \lambda \); so, \( w'' \) admits the first component \( \sigma \) of \( \tau_1 \) but not the second \( (o \rightarrow o \rightarrow o) \). The final element of \( w' \) is the same as the first element of \( w'' \); in such a case we define their concatenation to be \( w \).

**Definition 10.** The concatenation of (a prefix) \( v \) and (a suffix) \( w \), \( v \cdot^w \), is: \( \varepsilon \cdot^w = w \); if \( v_k = w_1 \) then \( v_1, \ldots, v_k, w_1, \ldots, w_n = v_1, \ldots, v_k, w_2, \ldots, w_n \).

## 3 Proof Systems for Retracts

We now develop goal directed tableau proof systems for showing retracts. By inverting the rules one has more classical axiomatic systems: we do it this way because it thereby provides an immediate nondeterministic decision procedure for deciding retracts. We present two such proof systems: a slightly simpler system for the restricted case when there is a single ground type and one for the general case.

### 3.1 Single Ground Type

Assertions in our proof system are of two kinds. First is \( \rho \sqsubseteq \tau \) with meaning \( \rho \) is a retract of \( \tau \). The second has the form \([\rho_1, \ldots, \rho_k] \sqsubseteq \tau \) which is based on the “product” as defined in [3]. We follow [9] in allowing reordering of components of types since \( \rho \rightarrow \sigma \rightarrow \tau \) is isomorphic to \( \sigma \rightarrow \rho \rightarrow \tau \). Instead we could include explicit rules for reordering (as with the axiom in [3]). Moreover, we assume that \([\rho_1, \ldots, \rho_k]\) is a multi-set and so elements can be in any order.
The proof rules are given in Figure 2. There is a single axiom $I$, identity, a weakening rule $W$, a covariance rule $C$, and two product rules $P_1$ and $P_2$. The rules are goal directed: for instance, $C$ allows one to decompose the goal $\delta \to \rho \leq \sigma \to \tau$ into the two subgoals $\delta \leq \sigma$ and $\rho \leq \tau$. $I$, $W$ and $C$ (or their variants) occur in the proof systems for affine retracts (when variables in witnesses can only occur at most once) [3, 9]. The new rules are the product rules: $P_2$ appeals to $k$-minimal realisable paths (presented as words), and the restriction operator of Definition 8. The proof system does not require the axiom A4 of [3], $\sigma \leq (\sigma \to a) \to a$: all instances are provable using $W$ and $C$.

**Definition 11.** A successful proof tree for $\rho \leq \tau$ is a finite tree whose root is labelled with the goal $\rho \leq \tau$, the successor nodes of a node are the result of an application of one of the rules to it, and each leaf is labelled with an axiom. We write $\triangleright \rho \leq \tau$ if there is a successful proof tree for $\rho \leq \tau$.

For some intuition about the product rules assume $\rho = \rho_1 \to \ldots \to \rho_l \to a$ and $\tau = \tau_1 \to \ldots \to \tau_n \to a$. Now, $\triangleright \rho \leq \tau$ if there are canonical, Definition 2, witnesses $D(z_{1}^{\rho_1}, \ldots, z_l^{\rho_l}) = \lambda f . f S_{1}^{\tau_1} \ldots S_{n}^{\tau_n}$. Since we can reorder components of $\rho$ and $\tau$ we can assume that $z_1$ is in $S_{1}^{\tau_1}$. Suppose $z_1, \ldots, z_k$, where $k > 1$, are in $S_{1}^{\tau_1}$ and so $\rho_1$ must occur in $T_{1}^{p_1}, \ldots, T_{1}^{p_l}$. Therefore, there is a common coder $S_{1}^{\tau_1}(x_1/z_1, \ldots, x_k/z_k)$ and $k$ decoders $T_i(z_i)$ where $z_i = z_{i1}, \ldots, z_{il_i}$, and $\rho_1, \ldots, \rho_l$, are the components of $\rho_i$ such that $T_i(z_i)(S_{1}^{\tau_1}(x_1, \ldots, x_k)) = \beta g x_i z_i$ (which is similar to the product in [3]). In $S_{1}^{\tau_1}(x_1/z_1, \ldots, x_k/z_k)$ there are distinct odd length paths $w_1, \ldots, w_k$ of type $\tau_1$ to the lambda nodes above $x_1, \ldots, x_k$. These paths are realisable, Definition 6, because each $x_i$ belongs to the normal form of $T_i(z_i)(S_{1}^{\tau_1}(x_1, \ldots, x_k))$. Using a combinatorial argument, see the full version, $S_{1}^{\tau_1}$ can be chosen so that these words are $k$-minimal and
(σ → o) → (σ → o) → o ≤ (σ → (o → o → o) → o) → o

\[\begin{align*}
[σ → o, σ → o] & \leq σ → (o → o → o) → o \quad o ≤ o \\
σ → o & \leq σ → o \\
σ → o & \leq σ → o
\end{align*}\]

Fig. 3. A proof tree for Example 1

by reordering ρ’s components $w_1 \subset \ldots \subset w_k$. We may not be able to reduce to the subgoals $ρ_1 \leq τ_1, \ldots, ρ_k \leq τ_1$ as $w_i$ may prescribe the form of $T_i(z_i)$: if $T_i(z_i) = λf^1 · fS_1^i \ldots S_m^i$ then path $w_i$ may prevent $S_i^j$ containing elements of $τ_i$, so, this may restrict the possible distribution of $z_i$ within the subterms $S_1^i, \ldots, S_m^i$ which is captured using $τ_i \upharpoonright w_i$.

An example proof tree is in Figure 3 for the retract of Example 1 (which is not affine). Rule $P_1$ is applied to the root and then $P_2$ to the first subgoal where $w_1 = (ε, 2, 21)$ and $w_2 = (ε, 2, 22)$. Let $σ' = σ → (o → o → o) → o$. Now, $σ' \upharpoonright w_1 = σ → o = σ' \upharpoonright w_2$; in both cases only the first component of $σ'$ is admitted.

3.2 Multiple Ground Types

We extend the proof system to include multiple ground types. Again, assertions are of the two kinds $ρ \leq τ$ and $[ρ_1, \ldots, ρ_k] \leq τ$. However, we now assume that to be a well-formed assertion $ρ \leq τ$ both $ρ$ and $τ$ must share the same target type (which is guaranteed when there is a single ground type). The rules for this assertion are as before the axiom $I$, weakening $W$, covariance $C$ and the product rule $P_1$ in Figure 2: however, $C$ carries the requirement that the target types of $δ$ and $σ$ coincide. The other product rule $P_2'$, presented in Figure 4, is different: the arity of $ρ_1 → \ldots → ρ_n → a$ is the maximum of $n$ and the arities of each $ρ_i$ where a gound type $a$ has arity 0.

\[\begin{align*}
P_2' & \frac{[ρ_1, \ldots, ρ_k] \leq σ}{ρ_1 \leq v_1(σ) \mid w_1 \ldots ρ_k \leq v_k(σ) \mid w_k}
\end{align*}\]

where

- $k'$ is the maximum of $k$ and $h^2$ where $h$ is the arity of $σ$
- there is a $k'$-minimal realisable subtree $U$ of type $σ$ where each path has even length (which can be $\emptyset$),
- each $v_i$ is $ε$, a prefix of a path in $U$ of odd length or the extension of a path in $U$ with a single node,
- $v_i^1w_1 \subset \ldots \subset v_i^hw_h$ and each $v_i^hw_i$ is a $k'$-minimal realisable path of type $σ$ of odd length and if $U \neq \emptyset$, $v_i^hw_i$ extends some path in $U$.

Fig. 4. Product proof rule for multiple gound types
(b → a) → a → a ≤ (b → (a → o → a) → a) → a

\[
\frac{[b → a, o] ≤ b → (a → o → a) → a}{b → a ≤ b → a \quad o ≤ o}
\]

Fig. 5. A proof tree for Example 2

In \([p_1, \ldots, p_k] ≤ \sigma\) it is not required that \(p_j\) and \(\sigma\) share the same target type. Instead rule \(P_2^k\) requires that \(p_i\) and \(v_i(\sigma)\), see Definition 9, do share target types: for the concatenation \(v^i_1 w_i\), see Definition 10. The specialisation to the case of the single ground type is when \(U = \emptyset\) and \(v = \varepsilon\).

Let \(ρ = p_1 → \ldots → p_1 → a\) and \(τ = τ_1 → \ldots → τ_n → a\). So, \(|= ρ ≤ τ\) iff there are canonical witnesses \(D(z^1_{p_1}, \ldots, z^{\varepsilon}_{p_1}) = λf^\tau.fS^1_1 \cdots S^\varepsilon_n\) and \(C(x) = λy^1_1 \cdots y^\varepsilon_n.H(xT^1_1 \cdots T^\varepsilon_n)\). Assume \(z_1, \ldots, z_k\), where \(k ≥ 1\), occur in \(S^1_1\). There is a path \(v\) in \(C(x)\) to the node above \(x\) which determines a subtree \(U\) of \(S^1_1\). The head variable in \(T^1_1\) bound in \(v\) has the same target type as \(p_i\). There are distinct paths \(v^1_1 w_1, \ldots, v^k_1 w_k\) of odd length to the lambda nodes above \(z_1, \ldots, z_k\) in \(S^1_1\): \(v_i\) is decided by the meaning of the head variable in \(T^1_1\); so, \(v_i(τ_1)\) has the same target type as \(p_i\). The rest of the path is the tail of \(w_i\); so we need to consider whether \(|= p_i ≤ v_i(τ_1) \vDash w_i\).

Figure 5 is the proof tree for the retract in Example 2. There is an application of \(P_1\) followed by \(P_2^k\). In the application of \(P_2^k\) the subtree \(U = \{(ε, 2)\}\), \(v_1 = ε\), \(w_1 = (ε, 2, 21) = v^1_1 w_1\), \(v_2 = (ε, 2, 22) = v^2_2 w_2\) when \(w_2 = (22)\). So, \(v_1(b → (a → o → a) → a) \vDash w_1 = b → a\) as the first component is admitted (unlike the second); and \(v_2(b → (a → o → a) → a) = o = o \vDash w_2\).

3.3 Complexity

The proof systems provide nondeterministic decision procedures for checking retracts. Each subgoal of a proof rule has smaller size than the goal. Hence, by focussing on one subgoal at a time a proof witness can be presented in \(\text{PSPACE}\). However, this does not take into account checking that a subgoal obeys the side conditions in the case of the product rules. Given any type \(\sigma\), there are boundedly many realisable \(k\)-minimal paths (with an upper bound of \(k^n\) where \(n\) is size of \(\sigma\)). So, this means that overall the decision procedure requires at most \(\text{EXPSPACE}\).

4 Soundness and Completeness

To show soundness and completeness of our proof systems, we define a dialogue game \(G(D(\overline{\pi}), C(x))\) played by a single player \(\forall\) on the trees of potential witnesses for a retract that characterises when \((D(\overline{\pi}))C(x) = βη \pi \overline{\sigma}\), similar to game semantics [5]. The game is defined in the full version of the paper.
To provide intuition for the reader we briefly describe $G(D(z_1, z_2), C(x))$ where these terms are from Figure 1. Play starts at node (0), the binder $\lambda f$ at that node is associated with $C(x)$ rooted at (12); so, the next position is at node (1) and therefore jumps to (12); the binder at (12) $\lambda y$ is associated with node (2) (the successor of (1)). Play proceeds to (13) and $\wedge$ chooses to go left or right; suppose it is left, so play is then at (14); nodes (13) and (14) are part of the normal form, see Definition ??). Play descends to (15) and, therefore, jumps to (2); so, with the binder at (2), $u$ is associated with the the subtree at (16) and $v$ with the subtree at (18). Play proceeds to (3) and so jumps to (18); now, $s$ is associated with (4) and $t$ with (8). Play proceeds to (19) and so jumps to (4), descends to (5) and then to (6) and then to (7) and jumps to (16) before finishing at (17). This play captures the path $x\lambda w.z w$ of the normal form.

Some of the key properties, defined in the full version, we appeal to in the correctness proofs below associate subtrees with realisable paths and vice versa. For instance, as illustrated in the play above the path rooted at (0) dowonto (7) is associated with the subtree rooted at (12) and with leaves (17) and (19). Let $\rho = \rho_1 \rightarrow \ldots \rightarrow \rho_t \rightarrow a$ and $\tau = \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow a$ and let $\tau_1 = \sigma = \sigma_1 \rightarrow \ldots \rightarrow \sigma_m \rightarrow b$.

Theorem 1. (Soundness) If $\vdash \rho \leq \tau$ then $\models \rho \leq \tau$.

Proof. By induction on the depth of a proof. For the base case, the result is clear for a proof that uses the axiom $I$. So, assume the result for all proofs of depth $< d$. Consider now a proof of depth $d$. We proceed by examining the first rule that is applied to show $\vdash \rho \leq \tau$. If it is $W$ or $C$ the result follows using the same arguments as in [3]. Assume the rule is $W$ and suppose $\models \rho \leq \tau$. Therefore there are terms $D_1$ and $C_1$ such that $D_1 \lnot \rho (C_1^{\rho \rightarrow \tau}) = \beta \eta \ x$. Now $D_1(\rho \rightarrow \tau) - \rho = \lambda f \sigma \rightarrow \tau \ y. D_1(f y)$ and $C_1^{\rho \rightarrow \tau} x = \lambda \delta \sigma. C_1(x)$ are witnesses for $\models \rho \leq \delta \rightarrow \sigma \rightarrow \tau$. Assume that the rule is $C$, so $\models \delta \leq \sigma$ and $\models \rho \leq \tau$. So there are terms $D_1, C_1, D_2, C_2$ such that $D_1^{\rho \rightarrow \sigma} (C_1^{\rho \rightarrow \sigma}) = \beta \eta \ x$ and $D_2^{\rho \rightarrow \sigma} (C_2^{\rho \rightarrow \sigma}) = \beta \eta \ x$. Now $D_1(\rho \rightarrow \delta \rightarrow \sigma) - \rho = \lambda x y. C_2(x(D_1 y))$ and $C_1(\beta \eta \ x, D_2(\rho(z_1 y)))$ are witnesses for $\models \delta \rightarrow \rho \leq \sigma \rightarrow \tau$.

Consider next that the first rule is $P_1$. So after $P_1$ there is either an application of $P_2$ or $P_2^2$; in the former case, there are k-minimal realisable paths $w_1 \subseteq \ldots \subseteq w_k$ of odd length of type $\sigma$ such that $\vdash \rho_i \leq \sigma \rightarrow w_i$; in the latter case, there is a $k'$-minimal realisable subtree $U$ of type $\sigma$ where each path has even length; and there are paths $v_1^i w_1 \subseteq \ldots \subseteq v_k^i w_k$ where each element is a $k'$-minimal realisable path of type $\sigma$ of odd length and if $U \neq \emptyset$, it extends some path in $U$ and where each $v_i$ is $\varepsilon$, a prefix of a path in $U$ of odd length path or an extension of a path in $U$ with a single node and $\vdash \rho_i \leq v_i(\sigma) \rightarrow w_i$; where $k'$ is the maximum of $k$ and the square of the arity of $\sigma$. So, by the induction hypothesis there are terms $D_i(z_i)$ and $C_i(x_i)$ such that $D_i(z_i)(C_i(x_i)) = \beta \eta \ x_i z_i$, witnesses for $\rho_i \leq \sigma \rightarrow w_i$ or $\rho_i \leq v_i(\sigma) \rightarrow w_i$, and terms $D'(z_{k+1}, \ldots, z_i)$ and $C'(x')$ such that $D'(z_{k+1}, \ldots, z_i)(C'(x')) = \beta \eta \ x' z_{k+1} \ldots z_i$, witnesses for $\rho_{k+1} \rightarrow \ldots \rightarrow \rho_t \rightarrow a \leq \tau'$ where $\tau' = \tau_2 \rightarrow \ldots \rightarrow \tau_n \rightarrow a$. We assume that all these terms are canonical witnesses. The term $D'(z_{k+1}, \ldots, z_i)$
is \( \lambda f'^{\tau}.fS_2^\tau \ldots S_n^\tau \) and \( C'(x') \) is \( \lambda y_2^\tau \ldots y_n^\tau.H'(x'T_{k+1}^{\rho_k+1} \ldots T_1^{\rho_1}) \) where \( H' = \varepsilon \) if the rule applied was \( P_2 \).

We need to show that there are terms \( D(z_1, \ldots, z_l) \) and \( C(x) \) that are witnesses for \( \models \rho \leq \tau \). \( D(\overline{z}) \) will have the form \( \lambda f^{\tau}.fS_1^\tau \ldots S_n^\tau \) and \( C(x) \) the form \( \lambda y_1^\tau \ldots y_n^\tau.H(xT_1^{\rho_1} \ldots T_1^{\rho_1}) \) where \( H = \varepsilon \) in the case of a single ground type. All that remains is to define \( S_1^\tau \) so it contains \( z_1, \ldots, z_k, T_1^{\rho_1}, \ldots, T_k^{\rho_k} \) and \( H \) (as an extension of \( H' \)). If \( U = \emptyset \) then \( H = H' \). Otherwise, let \( u \) be an odd length path such that \( U \) is associated with (so, its head variable is \( y_1^\tau \)). \( H \) consists of the suffix of \( u \) followed by the subtree \( H' \). The head variable of each \( T_i^{\rho_i} \) is \( y_1 \) in the case of the single ground type and \( g_i^{\tau}(x) \) in the general case (which is either \( y_1 \) or bound in \( u \)).

We assume that \( S'_1 \) is the subterm of \( S_1^\tau \) that is rooted at the initial vertex of the path \( w_i \); which is \( S_1^\tau \) itself in the single ground type. To complete these terms we require that \( T_i^{\rho_i}(S'_1(z_1, \ldots, z_k)) = \beta_n z_i \). Therefore, removing lambda abstraction over variables \( z_i \) and changing \( z_i \) to \( x_i \), we require that \( T_i(\overline{z}_i)(S'_1(x_1, \ldots, x_k)) = \beta_n x_i \overline{z}_i \). We construct a term \( C''(x_1) \) that occurs after the path \( w_i \) in \( S'_1 \) (and which has root \( x_i \) when there is a single ground type). We also complete \( T_i(\overline{z}_i) \) whose initial part is the tree \( U_i \) associated with the path \( w_i \).

First, we examine the single ground type case. So, \( S_1^\tau \) will have the form \( \lambda u_1 \ldots u_m.S'_1, C''(x_1) \) the form \( x_1 C''_{i_1} \ldots C''_{i_p} \) and \( T_i(\overline{z}_i) \) the form \( \lambda f^{\tau_1}.fV_{i_1}^{\tau_1} \ldots V_{i_p}^{\tau_p} \). Assume \( D_i(\overline{z}_i) \) is \( \lambda q^{\nu_i}g_i W_i^{\nu_i} \ldots W_i^{\nu_i} \) and \( C_i(x_1) \) is \( \lambda u_1 \ldots u_l x_i C_{i_1} \ldots C_{i_p} \). Assume \( w_i \) admits \( \sigma_i \); therefore, for some \( r : 1 \leq r \leq m, i_j = r \) (so, \( W_i^{\nu_i} \) may contain occurrences of variables in \( \overline{z}_i \)). If \( w_i \) does not occur in the path \( w_i \) then we set \( V_i = W_i^{\nu_i} \). Otherwise, there is a non-empty subpath \( w_{ir} \) of \( w_i \) generated by \( u_r \), and a subtree \( V_{i_r}^{\nu_i} \) of \( V_i^{\nu_i} \) associated with \( w_{ir} \). Each \( C_{i_r}^{\nu_i} \) contains a single \( u_i \) (as head variable). Assume \( C_{i_r}^{\nu_i} \) contains \( u_r \). Assume that the path in \( W_i^{\nu_i} \) to the lambda node above \( z_{is} \) is \( w_{is}' \). If we can build the same path in \( W_i^{\nu_i} \) (by copying nodes of \( C_{i_r}^{\nu_i} \) to \( C_{is}^{\nu_i} \)) then we are done (letting \( V_{is}^{\nu_i} \) include this path followed by the subterm of \( W_i^{\nu_i} \) rooted at \( z_{is} \)). Otherwise, we initially include \( w_{ir} \) in \( C_{is}^{\nu_i} \) and then try to build \( w_{is}' \) in \( V_{is}^{\nu_i} \) by copying nodes of \( C_{i_r}^{\nu_i} \) to \( C_{is}^{\nu_i} \); in \( V_{is}^{\nu_i} \) and, therefore in \( U_i^{\nu_i} \), there is a path whose prefix except for its final variable vertex is the same as a prefix of \( w_{is}' \) and then differ. In the game \( G(C_{is}^{\nu_i}, V_{is}^{\nu_i}) \), play jumps from that variable in \( V_{is}^{\nu_i} \) to a lambda node in \( w_{is} \). By definition of admits, there is a binder \( n' \) labelled \( \lambda \tau \) in \( w_{is} \) such that for some \( q \) \( q(n' \downarrow q n'_l) \) for all nodes \( n'_l \) after \( n' \) in \( w_i \) (and in \( w_{is} \)). Therefore, we add a variable node labelled \( v_q \) to the end of \( w_{is} \) in \( C_{is}^{\nu_i} \); so play jumps to a lambda node in \( V_{is}^{\nu_i} \) which is a successor of a leaf of \( U_i^{\nu_i} \); below this node, we build the path \( w_{is}' \) except for its root node (by adding further nodes to \( C_{is}^{\nu_i} \) and add the subtree rooted at \( z_{is} \) in \( W_i^{\nu_i} \) to \( V_{is}^{\nu_i} \)).

For the general case, assume \( v_1(\sigma) = \sigma_1' \rightarrow \ldots \rightarrow \sigma_m' \rightarrow b \). So, \( S_{i_r}^{\nu_i}(\sigma) \) will have the form \( \lambda u_1 \ldots u_m.S'_1, C''(x_1) \) the form \( H_i(x_1 C''_{i_1} \ldots C''_{i_p}) \) and \( T_i(\overline{z}_i) \) the form \( \lambda f^{\tau_1}.fV_{i_1}^{\tau_1} \ldots V_{i_p}^{\tau_p} \). Assume \( D_i(\overline{z}_i) \) is \( \lambda q^{\nu_i}g_i W_i^{\nu_i} \ldots W_i^{\nu_i} \) and \( C_i(x_1) \) is \( \lambda u_1 \ldots u_l .H_i(x_1 C_{i_1} \ldots C_{i_p}) \). We set \( H_i = H_i' \). Then we proceed in a similar fashion to the single base type case. If some \( u_i' \) does not occur in the path \( w_i \) then
$V_i = W_i$; otherwise we need to build similar paths to $z_is$ in $W_i$ in $V_i$ (by copying vertices from $C_is$ to $C''_is$ and using that $w_i$ admits $(v_i(σ))_i$. □

The proof of completeness (by induction on the size of $ρ$) is easier.

**Theorem 2.** *(Completeness)* If $|= ρ ⊆ τ$ then $⊢ ρ ⊆ τ$.

5 Conclusion

We have provided tableau proof systems that characterise when a type is a retract of another type in simply typed lambda calculus (with respect to $βη$-equality). They offer a a nondeterministic decision procedure for the retract problem in EXPSPACE: it may be possible to improve on the rather crude $k$-minimality bounds used on paths within the proof systems. Given the constructive proof of correctness, we also expect to be able to extract witnesses for a retract from a successful tableau proof tree (similar in spirit to [10]).

**References**