The Inhabitation Problem for Intersection Types

M W Bunder[1]

[1] School of Mathematics and Applied Statistics
University of Wollongong
Wollongong NSW 2522 AUSTRALIA
Email: mbunder@uow.edu.au

Abstract

In the system $\lambda\wedge$ of intersection types, without ω, the problem as to whether an arbitrary type has an inhabitant, has been shown to be undecidable by Urzyczyn in [10]. For one subsystem of $\lambda\wedge$, that lacks the $\wedge$-introduction rule, the inhabitation problem has been shown to be decidable in Kurata and Takahashi [9]. The natural question that arises is: What other subsystems of $\lambda\wedge$, have a decidable inhabitation problem?

The work in [2], which classifies distinct and inhabitation-distinct subsystems of $\lambda\wedge$, leads to the extension of the undecidability result to $\lambda\wedge$ without the (η) rule. By new methods, this paper shows, for the remaining six (two of them trivial) distinct subsystems of $\lambda\wedge$, that inhabitation is decidable. For the latter subsystems inhabitant finding algorithms are provided.

Keywords: Lambda Calculus, Type Theory, Intersection Types, Inhabitation.

1 Introduction

In simple (Curry-style) type theory (see for example Hindley [8]), not every closed lambda term (or combinator) has a type. Coppo and Dezani-Ciancaglini in [7] extended simple type theory to include intersection types and the universal type ω, in their system all $\lambda-$terms have types.

We consider the type assignment system $TA_{\lambda\wedge}$ (or simply $\lambda\wedge$), which is that of [7], without ω, in which all closed $\lambda-$terms with normal form have types. We will be interested in the inhabitation problem which asks if it can be decided whether, for a type α, there is a term X such that $\vdash X : \alpha$ in a given type theory. For $\lambda\wedge$ the inhabitation problem was shown to be undecidable by Urzyczyn in [10]. For a subsystem of $\lambda\wedge$, that lacks the $\wedge$-introduction rule, the inhabitation problem has been shown to be decidable in Kurata and Takahashi in [9]. We detemine which, of the other natural subsystems of $\lambda\wedge$, as identified in [2], have a decidable inhabitation problem, in some cases this follows easily from the work in [9] and [10]. We also provide algorithms which allow us to find an inhabitant X for a type α, in the decidable systems.

Before doing this we need to detail the type systems and list some results from [2].

1.1 Definition (Types)

The set of types T is given by:

1. $a, b, c, ...$, atoms (or type variables) are types.

2. If α and β are types so are $(\alpha \to \beta)$ and $(\alpha \wedge \beta)$.

A type $\alpha \to \beta$ is called an $\to$-type. A type $\alpha \wedge \beta$ is called an $\wedge$-type.

The usual bracketing rules of logic will apply.

1.2 Definition (Statements)

If M is a λ-term and α a type, $M : \alpha$ is a statement.

1.3 Definition (Judgements)

If Δ is a set of statements $\{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$ where $x_1, \ldots, x_n$ are distinct variables and $M : \alpha$ is a statement, $\Delta \vdash M : \alpha$ is a judgement.

1.4 Definition (Postulates for the Type Assignment System $TA_{\lambda\wedge}$)

(Var) $\qquad \Delta, x : \alpha \vdash x : \alpha$

$(\to E) \qquad \dfrac{\Delta \vdash M : \alpha \to \beta \qquad \Delta \vdash N : \alpha}{\Delta \vdash MN : \beta}$

$(\to I) \qquad \dfrac{\Delta, x : \alpha \vdash M : \beta}{\Delta \vdash \lambda x.M : \alpha \to \beta}$

$(\wedge I) \qquad \dfrac{\Delta \vdash M : \alpha \qquad \Delta \vdash M : \beta}{\Delta \vdash M : \alpha \wedge \beta}$

$(\wedge E) \qquad \dfrac{\Delta \vdash M : \alpha \wedge \beta}{\Delta \vdash M : \alpha} \qquad \dfrac{\Delta \vdash M : \alpha \wedge \beta}{\Delta \vdash M : \beta}$

$(\eta) \qquad \dfrac{\Delta \vdash \lambda x.Nx : \alpha \qquad x \notin FV(N)}{\Delta \vdash N : \alpha}$

$\Delta \vdash M : \alpha$ (or more formally $\Delta \vdash_{\lambda\wedge} M : \alpha$) will represent: $\Delta \vdash M : \alpha$ can be derived from the above postulates.

The system $TA_{\lambda\wedge}$ will usually be abbreviated to $\lambda\wedge$.

An alternative formulation of $\lambda\wedge$ uses a preorder $\leq$ on T.

1.5   Definition ($\leq$)

Axioms

(1) $\alpha \leq \alpha$

(2) $\alpha \leq \alpha \wedge \alpha$
(3) $\alpha \wedge \beta \leq \alpha$
(4) $\alpha \wedge \beta \leq \beta$
(5) $(\alpha \to \beta) \wedge (\alpha \to \gamma) \leq \alpha \to \beta \wedge \gamma$

Rules

(6) $\alpha \leq \beta \leq \gamma \quad \Rightarrow \quad \alpha \leq \gamma$
(7) $\alpha \leq \alpha' \ \& \ \beta \leq \beta' \quad \Rightarrow \quad \alpha \wedge \beta \leq \alpha' \wedge \beta'$
(8) $\alpha \leq \alpha' \ \& \ \beta \leq \beta' \quad \Rightarrow \quad \alpha' \to \beta \leq \alpha \to \beta'$.

In Definition 1.4 the $(\wedge E)$ and (η) rules can be replaced by:

$$(\leq) \qquad \frac{\Delta \vdash M : \alpha \qquad \alpha \leq \beta}{\Delta \vdash M : \beta}$$

We will be interested in the following subsystems of $\lambda \wedge$.

1.6   Definition (Notation for Type Systems)

We will denote the system involving the judgements of Definition 1.3 for types, with postulates (Var), $(\to E)$ and $(\to I)$, by $\lambda(\ )$ and provability in this system by $\vdash$. Systems with additional rules will be denoted by $\lambda(\wedge I), \lambda(\wedge I, \eta)$ etc and the corresponding provability by $\vdash_{\wedge I}, \vdash_{\wedge I, \eta}$ etc. Clearly $\lambda \wedge$ is $\lambda(\wedge I, \wedge E, \eta)$ or $\lambda(\wedge I, \leq)$.

We will use λ and $\vdash_\lambda$ for Curry's simple type theory. This is $\lambda(\ )$ and $\vdash$ without the use of $\wedge$ in Definition 1.1(iii) and $(\wedge I)$ and $(\wedge E)$ in Definition 1.4.

We will write $A, B, C, \ldots$ for arbitrary type systems.

1.7   Definition (Inhabitation)

If A is one of the type systems of Definition 1.6, we say that a type α is inhabited if $(\exists M) \vdash_A M : \alpha$.

Note that α being inhabited does not imply that there is any algorithm that guarantees to find an inhabitant of α.

1.8   Definition (Inhabitation Problem)

The question as to whether, in a system A, it can be decided if an arbitrary type is inhabited or not is called the inhabitation problem of A.

Urzyczyn showed in [10] that the inhabitation problem for $\lambda \wedge$ is undecidable. Kurata and Takahashi have shown in [9] that the problem is decidable for $\lambda(\leq)$. Note that their method does not include an algorithm for finding an inhabitant for a given type.

In [2] we studied and classified the various subsystems of $\lambda \wedge$. We found that some of the subsystems A and B were equivalent in the sense that:

$$(\forall \alpha, M) [\vdash_A M : \alpha \quad \Leftrightarrow \quad \vdash_B M : \alpha] \qquad (1)$$

This is denoted by $A \approx_1 B$.

Additional systems A and B had equivalent inhabitation problems in that

$$(\forall \alpha) [(\exists N) \vdash_A N : \alpha \quad \Leftrightarrow \quad (\exists N) \vdash_B N : \alpha] \qquad (2)$$

This is denoted by $A \approx_2 B$.

Any pair of systems satisfying (2) that we found also satisfied

$$(\forall \alpha, M) [\vdash_A M : \alpha \quad \Rightarrow \quad \vdash_B M : \alpha] \vee \qquad (3)$$
$$(\forall \alpha, M) [\vdash_B M : \alpha \quad \Rightarrow \quad \vdash_A M : \alpha]$$

Work in [2] showed that systems equivalent in the (2) - (3) sense come in the following groups (or inhabitation equivalence classes).

(1) $\lambda \wedge [\equiv \lambda(\wedge I, \wedge E, \eta) \approx_1 \lambda(\wedge I, \leq)], \ \lambda(\wedge I, \wedge E)$
(2) $\lambda(\wedge I), \lambda(\wedge I, \eta)$
(3) $\lambda(\leq) [\approx_1 \lambda(\leq, \wedge E, \eta) \approx_1 \lambda(\leq, \wedge E)]$
(4) $\lambda(\wedge E), \lambda(\wedge E, \eta)$
(5) $\lambda(\ ) [\approx_1 \lambda(\eta)]$.

Note that $\lambda(\leq)$ and $\lambda(\wedge E, \eta)$ are distinct systems that are both "$\lambda \wedge$ without $(\wedge I)$". ($a \wedge b \to b \wedge a$ is inhabited in $\lambda(\leq)$ but not in $\lambda(\wedge E, \eta)$.)

Urcyczyn's work in [10] for $\lambda \wedge$ and the inhabitation equivalence of the systems in Group 1 lead to:

1.9   Theorem

The inhabitation problems for the systems $\lambda \wedge, \lambda(\wedge I, \leq)$ and $\lambda(\wedge I, \wedge E)$ are undecidable.

The work of Kurata and Takahashi in [9] shows that $\lambda(\leq)$ is decidable. As the systems in Group 3 are equivalent it follows that:

1.10   Theorem

The inhabitation problems for $\lambda(\leq), \lambda(\leq, \wedge E, \eta)$ and $\lambda(\leq, \wedge E)$ are decidable.

The system considered by Kurata and Takahashi was actually $\lambda(\leq)$ with (ω), but the addition or deletion of (ω) does not affect the result.

We will show below, using generation lemmas proved in [2], that inhabitation problems for the systems in Groups 2, 4 and 5 are also decidable. We will in fact provide algorithms to find inhabitants for arbitrary types in these systems.

Note that in systems that do not have both $(\wedge E)$ and $(\wedge I)$ or the full strength of $(\leq)$, we may have

$$\Delta \vdash M : \alpha \wedge \beta$$
but not $\qquad \Delta \vdash M : \beta \wedge \alpha$
and $\qquad \Delta \vdash M : \alpha \wedge (\beta \wedge \alpha)$
but not $\qquad \Delta \vdash M : (\alpha \wedge \beta) \wedge \alpha$.

Notation We write $\alpha_1 \wedge \ldots \wedge \alpha_n$ to represent one of the possible bracketings of $\alpha_1 \wedge \ldots \wedge \alpha_n$.

Of course, via the formulas as types isomorphism, a type in a type system can be considered as a theorem of a logic and its inhabitant as a proof of that theorem. The logics corresponding to the intersection type systems however, are not particularly simple (see Venneri [11] and Bunder [5] and [6]) and it is easier to examine decidability for the type theories rather than for the corresponding logics.

2 Inhabitation for $\lambda(\ )$

It is easy to show that any valid judgement $\Gamma \vdash \alpha$ in $\lambda(\ )$ can be transformed into a valid judgement $\Gamma' \vdash \alpha'$ in λ by replacing all distinct $\wedge$-types in Γ and α by distinct atoms. Hence as $\lambda(\ ) \approx_2 \lambda(\eta)$:

2.1 Theorem

The inhabitation problems for the systems $\lambda(\ )$ and $\lambda(\eta)$ are decidable.

If α is a type, an inhabitant of α, or a guarantee that there is none in $\lambda(\ )$ and $\lambda(\eta)$, can be provided by an inhabitant finding algorithm, such as that in [3], for λ, applied to the α'. (The methods used in [3] are a simplified version of the Ben-Yelles algorithm (see [4] and Hindley [8]).)

2.2 Example

$\tau = (a \wedge b \rightarrow c) \rightarrow a \wedge b \rightarrow (a \wedge b \rightarrow c \rightarrow (a \wedge b) \wedge d) \rightarrow (a \wedge b) \wedge d$

Let $\quad \tau' = (e \rightarrow c) \rightarrow e \rightarrow (e \rightarrow c \rightarrow f) \rightarrow f$.

Using the algorithm of [3] for λ:

$x_1 : e \rightarrow c, \ x_2 : e$ and $\ x_3 : e \rightarrow c \rightarrow f$, give $x_1 x_2 : c, \ x_3 x_2(x_1 x_2) : f$.

So $\quad \vdash \lambda x_1 x_2 x_3 . x_3 x_2(x_1 x_2) : \tau'$ and $\vdash \lambda x_1 x_2 x_3 . x_3 x_2(x_1 x_2) : \tau$.

Our proof of the decidability of the inhabitation problem for $\lambda(\wedge E)$ requires some additional notation and a number of preliminary lemmas.

3 Notation

3.1 Definition (Long Subterms)

An occurrence of a subterm N of a term M is said to be <u>long in M</u> if (i) $N \equiv x_i X_1 \ldots X_n$ and the occurrence is not part of $N X_{n+1}$ or (ii) if $N \equiv \lambda x_1 \ldots x_k . Q$ and the occurrence is not part of $\lambda x_0 . N$.

3.2 Definition (Positive and Negative Subtypes)

1. τ is a positive subtype of τ.

2. If $\alpha \rightarrow \beta$ is a positive (negative) subtype of τ then α is a negative (positive) subtype of τ and β is a positive (negative) subtype of τ.

3. If $\alpha \wedge \beta$ is a positive (negative) subtype of τ, α and β are positive (negative) subtypes of τ.

3.3 Definition (Long Subtypes)

An occurrence of a subtype α of a type τ is said to be a long $\rightarrow$-subtype of τ if the occurrence is not the α in a $\beta \rightarrow \alpha$ in τ.

An occurrence of a subtype α in τ is a long $\wedge$-subtype of τ if the occurrence is not the α in an $\alpha \wedge \beta$ or $\beta \wedge \alpha$ in τ.

3.4 Example

$\tau = (a \wedge b \rightarrow (c \rightarrow d) \rightarrow e) \wedge (f \rightarrow g)$.

τ and c are long positive $\rightarrow$ and $\wedge$ subtypes of τ ($\rightarrow \wedge$-subtypes).

$a \wedge b, c \rightarrow d$ and f are long negative $\rightarrow \wedge$-subtypes of τ.

$a \wedge b \rightarrow (c \rightarrow d) \rightarrow e$ and $f \rightarrow g$ are long positive $\rightarrow$-subtypes of τ.

a and b are long negative $\rightarrow$-subtypes of τ.

c, e and g are long positive $\wedge$-subtypes of τ.

d is a long negative $\wedge$-subtype of τ.

3.5 Definition (Nontrivial Intersections)

A nontrivial intersection is any one other than one of the form $\alpha \wedge \ldots \wedge \alpha$.

4 The Generation Lemma for $\lambda(\wedge E)$

The Generation Lemma follows directly from the work in [2], modified using Definition 1.11 and Lemma 4.3(v) of [2].

4.1 Lemma Generation Lemma for $\lambda(\wedge E)$

If

$$\Delta \vdash_{\wedge E} M : \alpha \qquad (4)$$

then one of the following holds:

1. $M \equiv x, \ (\exists \beta) \ x : \beta \ \epsilon \ \Delta \ \& \ \beta \equiv \beta_1 \wedge \ldots \wedge \alpha \wedge \ldots \wedge \beta_n$.

2. $M \equiv PQ, \ (\exists \beta, \gamma) \ \Delta \vdash_{\wedge E} P : \gamma \rightarrow \beta$
    $$\Delta \vdash_{\wedge E} Q : \gamma$$
    where the derivations are shorter than those of (4) and $\beta \equiv \beta_1 \wedge \ldots \wedge \alpha \wedge \ldots \wedge \beta_n$.

3. $M \equiv \lambda x . N, \ (\exists \beta, \gamma) \ \Delta, x : \beta \vdash_{\wedge E} N : \gamma$

where the derivation is shorter than that of (4) and $\alpha \equiv (\beta \rightarrow \gamma)$.

5 The Main Lemma for $\lambda(\wedge E)$

A derivation is said to have a <u>cut</u> if it has a use of $(\rightarrow E)$, as in Definition 1.4, where $\Delta \vdash M : \alpha \rightarrow \beta$ is derived by $(\rightarrow I)$, or a use of $(\wedge I)$ followed immediately by a use of $(\wedge E)$ (or an equivalent use of $(\leq)$). A derivation is <u>normalised</u> if it has no cuts.

It is well known (see [1]) that all derivations in $\lambda \wedge$ with ω can be normalised. All terms appearing in such derivations are in normal form. This result clearly applies to $\lambda \wedge$ and its subsystems as well.

5.1 Definition

The type of a variable x_m in a derivation of

$$x_1 : \tau_1, \ldots, x_n : \tau_n \vdash_A N : \alpha$$

will be defined to be τ_m, (i) if $1 \leq m \leq n$, or, (ii) if $\lambda x_m . M$ is introduced into N by $(\rightarrow I)$ from

$$x_1 : \tau_1, \ldots, x_m : \tau_m \vdash_A M : \beta.$$

In the derivation β is defined to be the type of the occurrence of M in N.

For systems without $(\wedge I)$ the type of a variable and the type of a term introduced, as a subterm, into a normalised derivation are uniquely defined. For systems with $(\wedge I)$ an occurrence of a term may have a finite set of types in a derivation.

5.2 Lemma

If $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash_{\wedge E} N : \alpha$, there exists a term M in β-normal form such that no two distinct variables of M have the same type and also:

1.
$$x_{j_1} : \tau_{j_1}, \ldots, x_{j_\ell} : \tau_{j_\ell} \vdash_{\wedge E} M : \alpha \qquad (5)$$

where $\{j_1, \ldots, j_\ell\} \subseteq \{1, \ldots, n\}$.

2. For every occurrence of a long subterm P of M with
$$FV(P) = \{x_{i_1}, \ldots, x_{i_k}\}$$

there are types $\tau_{i_1}, \ldots, \tau_{i_k}$ and β such that

$$x_{i_1} : \tau_{i_1}, \ldots, x_{i_k} : \tau_{i_k} \vdash_{\wedge E} P : \beta \qquad (6)$$

where:

(I) $\tau_{i_1}, \ldots, \tau_{i_k}$ are long negative $\rightarrow \wedge$-subtypes of $\tau = \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \alpha$.

(II) If P is of the form $\lambda x_r ... x_t . R$, β has a long positive $\rightarrow \wedge$-occurrence in α or a long negative $\rightarrow \wedge$-occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$.

(III) If P is of the form $x_r P_1 \ldots P_t, (t \geq 0)$, β has a long positive $\wedge$-occurrence in α or a long negative $\wedge$-occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$. Also β has a negative occurrence in α or a positive occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$. .

Proof (i) If N' is the β-normal form of N there is a normalised derivation of

$$x_1 : \tau_1, \ldots, x_n : \tau_n \vdash_{\wedge E} N' : \alpha. \qquad (7)$$

If in (7) a long subterm $x_s Q_1 \ldots Q_t$ of N' has type $\alpha_1 \rightarrow \ldots \rightarrow \alpha_u \rightarrow \gamma$, where γ is an atom or an intersection, this is replaced by $\lambda x_q \ldots x_{q+u-1} . x_s Q_1 \ldots Q_t x_q \ldots x_{q+u-1}$, where $x_q, \ldots x_{q+u-1}$ are variables not in N', $q > n$ and $\tau_{q+i-1} = \alpha_i$ for $q = 1, \ldots, u$. When all such changes to N' are made call the result N''.

Next free or bound variables x_p and x_q, with the same type in N'', are identified. For example $\lambda x_q . C_1[\lambda x_p . C[x_q, x_p]]$ becomes $\lambda x_p . C_1[\lambda x_p . C[x_p, x_p]]$. If $1 \leq p, q \leq n$, x_q can be omitted from the left hand side of the $\vdash$ in (7). None of these changes alter any subtypes of τ.

When all such changes have been made we have M and (7) becomes (5).

(ii) Let the variables in M other than $x_1, \ldots, x_n$ be $x_{n+1}, \ldots, x_m$ and their types be $\tau_{n+1}, \ldots, \tau_m$.

Case 1 $M \equiv P$.

In this case $\beta \equiv \alpha$ and (6) is (5) with any $x_j \notin FV(M)$ omitted. (This can always be done in a normalised derivation). (I) and (II) clearly hold and if P is of the form $x_r P_1 ... P_t$ it follows from Lemma 4.1(ii) that β has a positive occurrence in τ_r, so (III) holds.

We now prove the remaining cases by induction on M.

Case 2 $M \equiv x_i M_1 \ldots M_p. (p \geq 0)$ and P is, or is in, an M_j.

By Lemma 4.1(ii) applied p times we have:
$$x_{j_1} : \tau_{j_1}, \ldots, x_{j_\ell} : \tau_{j_\ell} \vdash_{\wedge E} x_i : \alpha_1 \rightarrow \ldots \rightarrow \alpha_p \rightarrow \xi$$
and
$$x_{j_1} : \tau_{j_1}, \ldots, x_{j_\ell} : \tau_{j_\ell} \vdash_{\wedge E} M_j : \alpha_j$$

where $\xi = \xi_1 \wedge \ldots \wedge \alpha \wedge \ldots \wedge \xi_t$ and $1 \leq j \leq p$.

We have (6) and (I) by the induction hypothesis, after leaving out variables not free in P. Also by the induction hypothesis, if P is of the form $\lambda x_r ... x_t . R$ and β does not have a long negative $\rightarrow \wedge$-occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$, it has a long positive $\rightarrow \wedge$-occurrence in α_j and so a long negative one in τ_i and a long positive one in α. Thus (II) holds.

If P is of the form $x_r P_1 ... P_t$, β has a long positive $\wedge$-occurrence in α_j (and so a long negative one in τ_i) or a long negative $\wedge$-occurrence in one of $\tau_{i_1}, ..., \tau_{i_k}$. Also β has a negative occurrence in α_j (and so a positive one in τ_i) or a positive occurrence in one of $\tau_{i_1}, ..., \tau_{i_k}$. Thus (III) holds.

Case 3 $M \equiv \lambda x_{n+1} . Q$, where P is, or is in, Q.

By Lemma 4.1 (iii)

$$x_{j_1} : \tau_{j_1}, \ldots, x_{j_\ell} : \tau_{i_\ell}, \ x_{n+1} : \tau_{n+1} \vdash_{\wedge E} Q : \xi$$

where $\alpha \equiv \tau_{n+1} \rightarrow \xi$.

By the induction hypothesis and the omission of variables that are duplicated or not free in Q, (6) and (I) hold. If P is of the form $\lambda x_r ... x_t . R$, β has a long positive $\wedge$-occurrence in ξ, and so in α, or a long negative $\wedge$-occurrence in one of $\tau_1, ..., \tau_{n+1}$. If this is in τ_{n+1} it has a long positive $\wedge$-occurrence in α. (Note that P can't be Q, in this case, as then P is not long in M.) Thus (II) holds.

If P is of the form $x_r P_1 ... P_t$, β has a long positive $\wedge$-occurrence in ξ (and so in α) or a long negative $\wedge$-occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$. If this is in τ_{n+1} this is a positive $\wedge$-occurrence in α. Also β has a negative occurrence in ξ (and so in α) or a positive occurrence in one of $\tau_{i_1}, \ldots, \tau_{i_k}$. If this is τ_{n+1}, this is negative in α. Thus (III) holds.

Note 1. In many modern trteatments of $\lambda-$calculus, clashes of bound variables, as introduced in part (i) of the proof, though strictly allowed, are avoided. The identification of variables with the same types, in this proof, and that of Lemma 8.2, simplifies the proof and leads to finitely bounded inhabitation search algorithms for $\lambda(\wedge E)$ and $\lambda(\wedge I)$ in Sections 6 and 9.
2. In the lemma corresponding to 5.2 in [3] (and also in the Ben-Yelles algorithm in Hindley [8]), we could assume that M was in long normal form, which meant that every long subterm of M, formed by application had an atomic type. Here we can only assume that M must have an atomic or an intersection type. For example $M = x_1 x_2$ in:

$$x_1 : a \rightarrow (b \rightarrow c) \wedge (e \rightarrow f), x_2 : a,$$

$$x_3 : (b \rightarrow c) \wedge (e \rightarrow f) \rightarrow g \vdash_{\wedge E} x_3(x_1 x_2) : g$$

cannot be expanded to $\lambda x_4 . x_1 x_2 x_4$ where $x_1 x_2 x_4$ has an atomic type.

5.3 Lemma

If

$$\Delta \vdash_{\wedge E} M : \tau, \qquad (8)$$

N appears in M and is introduced into the derivation of (8) by

$$\Delta' \vdash_{\wedge E} N : \alpha \qquad (9)$$

where $\Delta \subseteq \Delta'$, then if

$$\Delta' \vdash_{\wedge E} P : \alpha \qquad (10)$$

where $FV(P) \subseteq FV(N)$, we have

$$\Delta \vdash_{\wedge E} [P/N]M : \tau \qquad (11)$$

where in $[P/N]M$ only the given occurrence of N with type α, introduced in (9), is replaced by P.

Proof By induction, on M.

Case 1 $M \equiv N$ then $\Delta' \equiv \Delta$ and (11) is (10).

Case 2 $M \equiv RQ$ where N is (in) R or Q
By Lemma 4.1 (ii)

$$\Delta \vdash R : \gamma \to \beta$$

$$\Delta \vdash Q : \gamma$$

where $\beta \equiv \beta_1 \wedge \ldots \wedge \tau \wedge \ldots \wedge \beta_n$.
By the induction hypothesis we have

$$\Delta \vdash_{\wedge E} [P/N]R : \gamma \to \beta$$
$$\text{or} \qquad \Delta \vdash_{\wedge E} [P/N]Q : \gamma$$

and $(\to E)$ and $(\wedge E)$ gives (11).

Case 3 $M \equiv \lambda x.R$ where N is (in) R
By Lemma 4.1 (iii) we have

$$\Delta, x : \beta \vdash_{\wedge E} R : \gamma$$

where $\beta \to \gamma \equiv \tau$.
The result follows by the induction hypothesis and $(\to I)$.

6 The Type Inhabitant Search Algorithm for $\lambda(\wedge E)$

Aim Given a $\to \wedge$-type τ, to find a λ-term M such that

$$\vdash_{\wedge E} M : \tau$$

Step 1 To each distinct long negative $\to \wedge$-subtype of τ assign a distinct variable, giving a list:

$$x_1 : \tau_1, \ldots, x_m : \tau_m$$

Step 2 For each type $\tau_i \equiv \alpha \wedge \beta$ from Step 1 write $x_i : \alpha, x_i : \beta$, repeat the procedure if α or β are intersections. Identical types for the same x_i may be omitted, identical types may also be obtained for distinct $x_i s$.

Step 3 For each set $A \subseteq \{x_1, \ldots, x_m\}$ and for each β that has both a long positive $\wedge$- and a negative occurrence in τ find an N, by application and $(\wedge E)$, such that $FV(N) \subseteq A$ and $N : \beta$, if there is not already such an N.

Step 4 For each set $A \subseteq \{x_1, \ldots, x_m\}$ and each β which has both a long positive and a long negative $\to \wedge$-occurrence in τ, if possible, find a term N by abstraction with respect to some or all of the variables found in Step 1 such that $FV(N) \subseteq A$ and $N : \beta$, if there isn't already such an N. (More than one abstraction with respect to the same variable is allowed.)
If after a Step 4 a closed $M : \tau$ is found stop. If not continue with further applications of Steps 3 and 4 until no new terms are created. If that happens, without forming $M : \tau$, τ has no inhabitants. This same algorithm, without Step 2, can be used to find inhabitants in $\lambda( )$ of λ.

6.1 Example

$$\tau \equiv (a \to b \wedge (c \to d)) \to a \wedge c \to d$$

Step 1 $x_1 : a \to b \wedge (c \to d), \ x_2 : a \wedge c$

Step 2 $x_2 : a, \ x_2 : c$

Step 3 $x_1 x_2 : b \wedge (c \to d), \ x_1 x_2 : c \to d, \ x_1 x_2 x_2 : d$

Step 4 $\lambda x_1 x_2.x_1 x_2 x_2 : \tau$.

6.2 Example

$$\tau = [(a \to b \to c) \wedge a \to b \to b \to c]$$

Step 1 $x_1 : (a \to b \to c) \wedge a, \ x_2 : b$

Step 2 $x_1 : a \to b \to c, \ x_1 : a$

Step 3 $x_1 x_1 : b \to c, \ x_1 x_1 x_2 : c$

Step 4 $\lambda x_1 x_2.x_1 x_1 : \tau$ and $\lambda x_1 x_2 x_2.x_1 x_1 x_2 : \tau$.

6.3 Theorem

Given a type τ, the Type Inhabitant Search Algorithm for $\lambda(\wedge E)$ will produce an inhabitant in β-normal form for τ in $\lambda(\wedge E)$ and $\lambda(\wedge E, \eta)$, or show that there is no such inhabitant in either system.

Proof By Lemma 5.2, if τ has an inhabitant, it has one M in β-normal form with no two variables of the same type.
Also by Lemma 5.2, Step 1 of the algorithm provides us with a finite set of typed variables which is the largest set that need appear in M. Step 2 provides each of these variables with a finite (possibly empty) set of additional types which includes all the types these variables need take in M.
Lemma 5.2 also provides us with all the composite types subterms of M can have and these again form a finite set. By Lemma 5.3, once we have a subterm for M with a certain set of free variables, there is no need to look for another with a superset of this set of free variables. Hence the number of terms that can be formed is finite and these are systematically constructed by Steps 3 and 4.
As the inhabitant Search Algorithm for $\lambda(\wedge E)$ is inherently finite, if it terminates without having found an inhabitant for τ, then τ has none.
Given that $\lambda(\wedge E) \approx_2 \lambda(\wedge E, \eta)$ and that $\lambda(\wedge E)$ is a subsystem of $\lambda(\wedge E, \eta)$, this algorithm also finds inhabitants of τ in $\lambda(\wedge E, \eta)$ or shows that there are none.

7 The Generation Lemma for $\lambda(\wedge I)$

The Generation Lemma follows directly from the work in [2], modified using Definition 1.11 and Lemma 4.3(iv) of [2].

7.1 Lemma Generation Lemma for $\lambda(\wedge I)$
If

$$\Delta \vdash_{\wedge I} M : \alpha \qquad (12)$$

where M is in normal form, then one of the following holds:

1. $M \equiv x, \ (\exists \beta) \ x : \beta \ \in \Delta \ \& \ \alpha \equiv \beta \wedge \ldots \wedge \beta$.

2. $M \equiv PQ$, $(\exists \beta_1, \alpha_1, \ldots \beta_k, \alpha_k)\Delta \vdash_{\wedge I} P : \beta_i \to \alpha_i$

$$\Delta \vdash_{\wedge I} Q : \beta_i.$$

for $1 \le i \le k$, where $\alpha \equiv \alpha_1 \wedge \ldots \wedge \alpha_k$ and the derivations, together, are shorter than those of (12).

3. $M \equiv \lambda x.N$, $(\exists \beta_1, \gamma_1, \ldots, \beta_k, \gamma_k)$ $\Delta, x : \beta_i \vdash_{\wedge I} N : \gamma_i$

for $1 \le i \le k$, where $\alpha \equiv (\beta_1 \to \gamma_1) \wedge \ldots \wedge (\beta_k \to \gamma_k)$ and each derivation is shorter than that of (12).

8   The Main Lemmas for $\lambda(\wedge I)$

The two lemmas below are generalisations of two lemmas used in [3] to prove that the inhabitation finding algorithm for simple type theory, that appears there, is valid. The situation here is little more complex because one occurrence of a subterm of X in $\Delta \vdash_{\wedge I} X : \tau$ can have more than one type in the derivation.

For example with $x : \beta$ we might prove:

$$\Delta \vdash_{\wedge I} \lambda x.M : \beta \to \alpha$$

and with $x : \gamma$, we might prove:

$$\Delta \vdash_{\wedge I} \lambda x.M : \gamma \to \delta,$$

and so by $(\wedge I)$:

$$\Delta \vdash_{\wedge I} \lambda x.M : (\beta \to \alpha) \wedge (\gamma \to \delta).$$

So not only $\lambda x.M$, but also x, can have more than one type in a derivation.

8.1   Lemma

If X is in normal form, U is an occurrence of a subterm of X which has types $\alpha_1 \ldots \alpha_k$ in the derivation of $\Delta \vdash_{\wedge I} X : \tau$, then if V, with $FV(V) \subseteq FV(U)$, can also be assigned types $\alpha_1, \ldots, \alpha_k$, given Δ, and if the types of $FV(V)$ are the same as they were for $FV(U)$, then

$$\Delta \vdash_{\wedge I} X[U := V] : \tau.$$

Proof By induction on X, as in Lemma 5.3.

8.2   Lemma

If

$$x_1 : \tau_1, \ldots, x_\ell : \tau_\ell \vdash_{\wedge I} Z : \tau \qquad (13)$$

then there is an $X =_\beta Z$ in β-normal form such that:

1. No two distinct variables of X have the same set of types.

2. For some $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, \ell\}$,

$$x_{i_1} : \tau_{i_1}, \ldots, x_{i_k} : \tau_{i_k} \vdash_{\wedge I} X : \tau \qquad (14)$$

3. Each type of each variable x_{i_j} in $\lambda x_{i_1} \ldots x_{i_k}.X$ that is used in a derivation of (14), will have a long negative $\to$ - occurrence in $\tau' = \tau_{i_1} \to \ldots \to \tau_{i_k} \to \tau$.

4. Each occurrence of a composite subterm Y of X, that is, in the derivation of (14), an abstraction, or is long and formed by application, will have a type which is a long positive $\wedge$-occurrence in τ'.

Proof (i), (ii). Theorem 4.13 of [1] proves that every Z satisfying (13), with $\lambda \wedge$ for $\lambda(\wedge I)$, has a β-normal form. Clearly this also holds for $\lambda(\wedge I)$. Subject reduction can be proved for $\lambda(\wedge I)$, by standard means, so (13) holds with $nf(Z)$ for Z.

If $nf(Z)$ contains two variables x_p and x_q, with the same set of types, we can change, by Lemma 8.1, all occurrences of x_q to x_p, without altering any types. Also we can drop one of two, now identical $x_p : \tau_p$ and $x_q : \tau_q$ from $x_1 : \tau_1, \ldots, x_\ell : \tau_\ell$. We then have (14), so (i) and (ii) hold.

(iii), (iv) By induction on the length of X.

If X, which is in normal form, is formed by application, $X \equiv x_{i_t} X_1 \ldots X_m$ then by Lemma 7.1(ii) and (i) we have:

$$x_{i_1} : \tau_{i_1}, \ldots, x_{i_k} : \tau_{i_k} \quad \vdash_{\wedge I} \quad x_{i_t} : \beta_1 \to \ldots \to \beta_m \to \beta$$
$$x_{i_1} : \tau_{i_1}, \ldots, x_{i_k} : \tau_{i_k} \quad \vdash_{\wedge I} \quad X_r : \beta_r \qquad (15)$$

for $1 \le r \le m$, where $\tau = \beta \wedge \ldots \wedge \beta$, $1 \le t \le k$ and $\tau_{i_t} = \beta_1 \to \ldots \to \beta_m \to \beta$.

If $x_{i_j} = x_{i_t}, \tau_{i_j} = \tau_{i_t}$ has a long negative $\to$ - occurrence in τ', so (iii) holds.

If x_{i_j} is in X_r, for some r, then, by the induction hypothesis (iii), τ_{i_j} has a long negative $\to$ - occurrence in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to \beta_r$.

If this occurrence is in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to$, this is a long negative $\to$ - occurrence in τ'. If it is in β_r, then τ_{i_j} has a long positive $\to$ - occurrence in τ_{i_t} and so a long negative $\to$ - occurrence in τ'. Hence (iii) holds.

If Y is a long composite subterm of X formed by application, it may be X itself, in which case (iv) holds with τ as the type of Y.

Otherwise Y is (in) an X_r. By the induction hypothesis (iv), applied to (15), we have that this occurrence of Y has a type with a long positive $\wedge$ - occurrence in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to \beta_r$. If the occurrence is in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to$, it is also one in τ'. If the occurrence is in β_r, as β_r has a long negative $\to$ - occurrence in τ_{i_t} and so a positive one in τ', each type of Y has a long positive $\wedge$ - occurrence in τ', i.e. (iv) holds.

If X is formed by abstractions, $X = \lambda x_{i_{k+1}}.V$, then by Lemma 7.1 (iii)

$$x_{i_1} : \tau_{i_1}, \ldots, x_{i_k} : \tau_{i_k}, x_{i_{k+1}} : \tau^s \vdash_{\wedge I} V : \beta^s \qquad (16)$$

for $1 \le s \le u$ and $\tau = (\tau^1 \to \beta^1) \wedge \ldots \wedge (\tau^u \to \beta^u)$.

As the derivation of (14) can come via (16), for $1 \le s \le u$, and $(\wedge I)$, any types τ_{i_j} of the variable x_{i_j} used in the derivation of (14) , must be used in the derivation of (16) for at least one value of s.

By the induction hypothesis (iii), applied to (16), we have that each τ_{i_j} has a long negative $\to$ - occurrence in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to \tau^s \to \beta^s$ and so in τ'. Thus (iii) holds.

If Y is formed by application and is long in X, it is also long in V, so, by the induction hypothesis (iv), each occurrence of Y, in the derivation of (16), for some $s, 1 \le s \le u$, has a type with a long positive $\wedge$ - occurrence in $\tau_{i_1} \to \ldots \to \tau_{i_k} \to \tau^s \to \beta^s$ and so in τ'. So (iv) holds.

If Y is formed by abstraction and is (in) V, the result (iv) holds by induction hypothesis (iv).

If Y is X, (iv) holds as the type of Y is τ.

The $\lambda(\wedge I)$ Inhabitant Search Algorithm, given below, is a generalised version of the algorithm for λ given in [4]. The latter, in turn, is a simplified version of the Ben-Yelles Algorithm for λ (see Hindley [8]).

9 The $\lambda(\wedge I)$ Inhabitant Search Algorithm

Aim To find a λ-term X such that, for all $i, 1 \le i \le k$

$$x_1 : \tau_1^i, \ldots, x_n : \tau_n^i \vdash_{\wedge I} X : \delta_i \qquad (17)$$

where $X x_1 \ldots x_n$ has (i) a minimal number of distinct variables and (ii) a minimal total number of occurrences of these variables.

Notes 1. To find an inhabitant X of a type τ, we only need to solve (17) for $n = 0, k = 1$, and $\delta_1 = \tau$, but we require to solve more general versions of (17) in the process.

2. For no value of m and ℓ $(1 \le m < \ell \le n)$ is $\tau_\ell^i = \tau_m^i$ for all $i, 1 \le i \le k$, as otherwise $X[x_\ell := x_m]$ will be a solution of (17) where $X[x_\ell := x_m] x_1 \ldots x_{\ell-1} x_{\ell+1} \ldots x_n$ has fewer distinct variables than $X x_1 \ldots x_n$.

3. We assume that no two instances of (17), for distinct values of i, are identical.

4. We will call a set of judgements, such as (17), for $1 \le i \le k$, with a common set of variables on the left of the $\vdash$ and a common unknown λ-term, such as X, on the right of the $\vdash$, a **simultaneous set of judgements (ssj)**.

5. An ssj of the form

$$x_1 : \tau_1^i, \ldots, x_n : \tau_n^i \vdash_{\wedge I} Y : \delta_i$$

for $1 \le i \le k$ is said to be **equivalent** to the ssj (17).

6. In the algorithm we will construct a tree where the nodes are ssj's, with (17) at the root.

Step 1 If in (17) $\tau_j^i = \delta_i$ for some j $(1 \le j \le n)$ and all $i, (1 \le i \le k)$, then the tree consists only of the root, the ssj (17), and the algorithm stops with $X = x_j$.

Otherwise, if $\tau_j^i = \beta_1^i \to \ldots \to \beta_r^i \to \delta_i$ for all i, $1 \le i \le k$, construct a group of ssj's

$$x_1 : \tau_1^i, \ldots, x_n : \tau_n^i \vdash_{\wedge I} X_{t_1} : \beta_{t_1}^i \qquad (18)$$

each with $1 \le i \le k$, provided there is no ssj equivalent to (18) for any $t_1 (1 \le t_1 \le r)$ in the branch from (17) to the root of the tree. If there is such an equivalent ssj, (17) has no solution.

For each τ_j^i of the appropriate form, that is not excluded in this way, there is a group of ssj's of the form (18) with $1 \le t_1 \le r$, that appears in the tree directly below (17).

If all of the ssj's in a group have a solution, found by going back to Step 1, then (17) has as solution $X = x_j X_1 \ldots X_r$.

If there is no solution of (18) for any t_1, there is no solution of (17) for that value of j, and the tree is not extended below the ssj's in the group with that value of j.

If there is no solution of (17) for any value of j or no τ_j^i is of the right form and if $\delta_i = \tau_{n+1}^i \to \gamma_i$ for all i $(1 \le i \le k)$, go to Step 2. If $\delta_i = \alpha_i \cap \gamma_i$, for some i, go to Step 3.

Step 2 If $\tau_{n+1}^i \ne \tau_j^i$ for some i and all j the ssj

$$x_1 : \tau_1^i, \ldots, x_{n+1} : \tau_{n+1}^i \vdash_{\wedge I} X' : \gamma_i \qquad (19)$$

with $1 \le i \le k$, appears directly below (17) in the tree (as a singleton group), provided no equivalent ssj appears in the branch from (17) to the root of the tree. (19) is then solved by returning to Step 1. If there is a solution then the solution to (17) is $\lambda x_{n+1}.X'$.

If $\tau_{n+1}^i = \tau_j^i$ for some j and all $i, 1 \le i \le k$, the ssj

$$x_1 : \tau_1^i, \ldots, x_n : \tau_n^i \vdash_{\wedge I} X' : \gamma_i \qquad (20)$$

appears directly below (17) in the tree, provided no equivalent ssj appears in the branch from (17) to the root.

(20) is solved by returning to Step 1. If there is a solution, then $X = \lambda x_j.X'$.

Step 3 We assume that (17) is ordered so that this $i = k$. In each of the four cases below we add a new ssj directly below (17) in the tree (as a singleton group), provided no equivalent ssj has appeared in the branch from (17) to the root.

1. If $\tau_1^k, \ldots, \tau_n^k, \alpha_k$ and $\tau_1^k, \ldots, \tau_n^k, \gamma_k$ are both distinct from each $\tau_1^j, \ldots, \tau_n^j, \delta_j$ for $1 \le j < k$, and from each other the ssj is

$$x_1 : \tau_1^i, \ldots, x_n : \tau_n^i \vdash_{\wedge I} X : \delta_i \qquad (21)$$

   for $1 \le i \le k+1$, where $\delta_k = \alpha_k$, $\delta_{k+1} = \gamma_k$, and $\tau_t^k = \tau_t^{k+1}$ for $1 \le t \le n$.

2. If $\tau_1^k, \ldots, \tau_n^k, \alpha_k$ is distinct from each $\tau_1^j, \ldots, \tau_n^j, \delta_j$ but $\tau_1^k, \ldots, \tau_n^k, \delta_k, \gamma_k \equiv \tau_1^j, \ldots, \tau_n^j, \delta_j$ for some $j, (1 \le j \le k)$ or $\equiv \tau_1^k, \ldots, \tau_n^k, \alpha_k$ the new ssj is similar to (17) but with α_k instead of δ_k.

3. If $\tau_1^k, \ldots, \tau_n^k, \gamma_k$ is distinct from each $\tau_1^j, \ldots, \tau_n^j, \delta_j$, but $\tau_1^k, \ldots, \tau_n^k, \alpha_k \equiv \tau_1^j, \ldots, \tau_n^j, \delta_j$ for some $j, (1 \le j \le k)$ the new ssj is similar to (17) but with γ_k for δ_k.

4. If $\tau_1^k, \ldots, \tau_n^k, \gamma_k \equiv \tau_1^r, \ldots, \tau_n^r, \delta_r$ and $\tau_1^k, \ldots, \tau_n^k, \alpha_k \equiv \tau_1^j, \ldots, \tau_n^j, \delta_j$ for some j and $r, 1 \le j, r \le k$, the new ssj is similar to (17), but with $1 \le i \le k - 1$.

In each case now go back to Step 1.

We now prove that the algorithm is effective.

9.1 Theorem

The $\lambda(\wedge I)$ Inhabitation Search Algorithm provides a solution X for (17), for all $i (1 \le i \le k)$ or a guarantee that there is no solution, by generating a tree with an ssj at each node and (17) at the root. No node will have more than $rs + 1$ nodes directly below it and no branch is longer than $2^{s(r-n)!}$ where r is the number of long negative $\to$-subtypes of

$$\alpha = (\tau_1^1 \to \ldots \to \tau_n^1 \to \delta_1) \wedge \ldots \wedge (\tau_1^k \to \ldots \to \tau_n^k \to \delta_k)$$

and s is the number of long positive $\wedge$-subtypes of α.

Proof In the algorithm, Step 1 considers all the possible ways in which (17) can be derived by (Var) or using $(\to E)$ as the final step, Step 2 considers the ways in which (17) can be derived by $(\to I)$ as a final step and Step 3 the ways in which (17) is derived using $(\wedge I)$ as a final step. Any X satisfying (17), for $1 \le i \le k$, must be found in an indefinitely extended tree.

Coming down any branch of the tree from the root, the algorithm has each unknown λ-term as a subterm of the ones above it. If the step above it is Step 1 (other than $\tau_j^i = \delta_i$ for $1 \le i \le k$) or Step 2, it will be

a proper subterm of the terms higher in the branch. If on a branch there is an ssj

$$x_1 : \tau_1^i, \ldots, x_p : \tau_p^i \vdash_{\wedge I} X_{t_1 \ldots t_s} : \beta_{t_1 \ldots t_s}^i \qquad (22)$$

for $1 \leq i \leq q$, appearing below an equivalent ssj

$$x : \tau_1^i, \ldots, x_p : \tau_p^i \vdash_{\wedge I} X_{t_1 \ldots t_v} : \beta_{t_1 \ldots t_v}^i \qquad (23)$$

(i.e. $\beta_{t_1 \ldots t_s}^i = \beta_{t_1 \ldots t_v}^i$ for $1 \leq i \leq q$), then $X_{t_1 \ldots t_s}$ is a proper part of $X_{t_1 \ldots t_v}$, as there must be at least one nontrivial Step 1 or a Step 2 between the ssj's. This however means that $X_{t_1 \ldots t_s}$ is a shorter solution than $X_{t_1 \ldots t_v}$ of the ssj (23), so the branch from (23) to (22) does not lead to an X satisfying (i) and (ii). So the algorithm rightly does not search branches below (22).

We now show that the tree must be finite.

By Lemma 7.1 (17) holds for $1 \leq i \leq k$, if and only if

$$\vdash_{\wedge I} \lambda x_1 \ldots x_n . X : \alpha.$$

By Lemma 8.2 the variables in $\lambda x_1 \ldots x_n . X$, and so in X, have types which are long negative $\rightarrow$-subtypes of α. Let there be r of these.

In a node (i.e. an ssj)

$$x_1 : \tau_1^i, \ldots, x_m : \tau_m^i \vdash_{\wedge I} Y : \beta_i \qquad (24)$$

for $1 \leq i \leq \ell$, as by the algorithm the types of different variables are distinct and $n < m \leq r$, there can be at most $(r - n)!$ different sequences $\tau_{n+1}, \ldots, \tau_m$.

Also by Lemma 8.2, each β_i is a long positive $\wedge$-subtype of α. Let there be s of these. Then there can be at most $s(r - n)!$ distinct judgements, of the form (24) in the tree generated by the algorithm and as each node (i.e. each ssj) in the tree consists of a subset of this set of judgements, there can be no more than $2^{s(r-n)!}$ distinct nodes (ssj's) in the tree.

Given that there can be no two equivalent ssj's on any branch, no branch can be longer than $2^{s(r-n)!}$.

Below any ssj, such as (24), there can be at most m groups of ssj's resulting from Step 1, where $m \leq r$. Each group can have no more than s members. Also below (24) there can be an ssj stemming from Step 2 or 3. Thus there can be no more than $rs + 1$ nodes below any node in the tree.

9.2 Corollary

The inhabitation problem for $\lambda(\wedge I)$ is decidable.

9.3 Example

To find X such that

$$\vdash_{\wedge I} X : (a \rightarrow a \rightarrow a \wedge a) \wedge ((a \rightarrow b) \rightarrow (b \rightarrow a) \rightarrow a \rightarrow b)$$
$$|$$
$$\text{Step 3}$$
$$|$$
$$\vdash_{\wedge I} X : a \rightarrow a \rightarrow a \wedge a$$
$$\vdash_{\wedge I} X : (a \rightarrow b) \rightarrow (b \rightarrow a) \rightarrow a \rightarrow b$$
$$|$$
$$\text{Step 2} \quad (X = \lambda x_1 . X')$$
$$|$$
$$x_1 : a \vdash_{\wedge I} X' a \rightarrow a \wedge a$$
$$x_1 : a \rightarrow b \vdash_{\wedge I} X'(b \rightarrow a) \rightarrow a \rightarrow b$$
$$|$$

$$\text{Step 2} \quad (X' = \lambda x_2 . X'')$$
$$|$$
$$x_1 : a, \; x_2 : a \vdash_{\wedge I} X'' : a \wedge a$$
$$x_1 : a \rightarrow b, \; x_2 : b \rightarrow a \vdash_{\wedge I} X'' : a \rightarrow b$$
$$|$$
$$\text{Step 3}$$
$$|$$
$$x_1 : a, \; x_2 : a \vdash_{\wedge I} X'' : a$$
$$x_1 : a \rightarrow b, \; x_2 : b \rightarrow a \vdash_{\wedge I} X'' : a \rightarrow b$$
$$|$$
$$\text{Step 1} \quad X'' = x_1$$
$$\text{So} \quad X = \lambda x_1 x_2 . x_1.$$

10 Conclusion

The inhabitation problem is decidable for the systems $\lambda(\leq), \lambda(\leq, \wedge E, \eta), \lambda(\leq, \wedge E), \lambda(\wedge I), \lambda(\wedge I, \eta), \lambda(\wedge E), \lambda(\wedge E, \eta), \lambda()$, and $\lambda(\eta)$ and undecidable for $\lambda \wedge, \lambda(\wedge I, \leq)$ and $\lambda(\wedge I, \wedge E)$.

We have given inhabitant finding algorithms for $\lambda(), \lambda(\eta), \lambda(\wedge E), \lambda(\wedge E, \eta), \lambda(\wedge I)$ and $\lambda(\wedge I, \eta)$.

References

Barendregt, H.P., Coppo, M., Dezani, M. (1983), 'A filter lambda model and the completeness of type assignment', *Journal of Symbolic Logic* **48**, 931–940.

Bunder, M.W. (2002), 'A classification of intersection type system', *Journal of Symbolic Logic* **67**, 353–362.

Bunder, M.W. (2000), 'Proof finding algorithms for implicational logics', *Theoretical Computer Science* **232**, 165-186.

Bunder, M.W. (1995), 'Ben-Yelles-type algorithms and the generation of proofs in implicational logics', *University of Wollongong, Department of Mathematics Preprint Series* **3/95**.

Bunder, M.W. (2002), 'Intersection type for lambda terms and combinators and their logics', *Journal of the Interest Group in Propositional Logic* **10**, 357–378.

Bunder, M.W. (2003), 'Intersection type systems and logics related to the Meyer-Routley system B+', *Australasian Journal of Logic* **1**, 43–55.

Coppo, M. & Dezani, M. (1978), 'A new type-assignment for lambda terms', *Archiv Math. Logik* **19**(2), 139–156.

Hindley, J.R. (1987), *Basic Simple Type Theory*, Cambridge University Press.

Kurata, T. & Takahashi, M. (1995), Mining association rules between sets of items in large databases, *in* 'Lecture notes in Computer Science', Vol. 902, TLCA '95. M. Dezani and G. Plotkin (eds) pp. 297–311.

Urzyczyn, P. (1999), 'The emptiness problem for intersection types', *Journal of Symbolic Logic* **64**, 1195–1215.

Venneri, B. (1994), 'Intersection types as logical formulae', *Journal of Logic and Computation* **4**, 109–124.