Online appendix for the paper

Typed Answer Set Programming Lambda Calculus Theories and Correctness of Inverse Lambda Algorithms with respect to them

published in Theory and Practice of Logic Programming

Chitta Baral, Juraj Dzifcak, Marcos A. Gonzalez and Aaron Gottesman
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ

submitted 25 March 2012; revised 11 June 2012; accepted 18 June 2012

Appendix A

Before presenting the theorems and proofs, we will present several necessary definitions.

Definition 1 (*applicator term*)
An applicator term of a typed λ-calculus formula is an expression of the form $x@J_1@\cdots@J_n$ where x is a variable and J_i are typed λ-calculus formulas.

Definition 2 (*sub-formula*)
A sub-formula of a typed λ-calculus formula F is a sub-term of F which is a typed λ-calculus formula.

Lemma 1
Given typed λ-calculus formulas H, G and F, if $G@F$ in β-normal form is equal to H, then the sub-terms in H must be the same as the sub-terms in $G@F$.

Proof
The definition of typed ASP λ-calculus formulas presented in previous sections states that all variables that appear in a formula are bound. Therefore, when one formula is applied to another, none of the sub-terms of those formulas will be modified by the application due to the definition of application in lambda calculus. $\square$

Lemma 2
Given typed λ-calculus formulas H and G, if G is a sub-term of H, then there is always an F such that $H = F@G$.

Proof
Let $G = g_1, g_2 ..., g_n$. (where $g_1, ..., g_n$ are λ-elements)
We know that G occurs in H. Thus,
Let $H = h_1, h_2, \cdots, h_i, g_1, \cdots, g_n, h_{i+1}, \cdots, h_n$.
Let F be λv.$h_1, h_2, \cdots, h_i, v, h_{i+1}, \cdots, h_n$.
Then, $H = h_1, h_2, \cdots, h_i, g_1, g_2, \cdots, g_n, h_{i+1}, \cdots, h_n = \lambda v.h_1, h_2, \cdots, h_i, v, h_{i+1}, \cdots, h_n$ @ $g_1, g_2, \cdots, g_n = F@G$. $\square$

Theorem 1 (*Soundness of Inverse$_L$*)
Given two typed λ-calculus formulas H and G in β-normal form, if *Inverse*$_L(H, G)$ returns a non-null value F, then $H = F$ @ G.

Proof
Let G and H be two typed λ-calculus formulas, let $F = Inverse_L(H, G)$.

1. If G is $\lambda v \cdot v$
   - $F = \lambda v \cdot (v@H)$

   By the above condition of the algorithm for this case, $H = H$ and $G = \lambda v \cdot v$. Then, $F@G = \lambda v \cdot (v@H)$ @ $\lambda v \cdot v = (\lambda v \cdot v)@H = H$.
2. If G is a sub-term of H
   - $F = \lambda v \cdot H(G : v)$

   Let E_1 and E_2 be possibly empty series of λ-elements. By the above condition of the algorithm for this case, $H = E_1 G E_2$ and $G = G$. Then, $F@G = (\lambda v \cdot E_1 v E_2)$ @ $G = E_1 G E_2 = H$.
3. G is not $\lambda v \cdot v$, $(J^1(J^1_1, \cdots, J^1_m), J^2(J^2_1, \cdots, J^2_m), \cdots, J^n(J^n_1, \cdots, J^n_m))$ are sub-terms of H and, $\forall J^i \in H$, G is $\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})$ with $1 \leq s \leq m$ and $\forall p$, $1 \leq k_p \leq s$.
   - $F = \lambda w \cdot H((J^1 : (w@J^1_{k_1}@ \cdots @J^1_{k_m}), \cdots, J^n : (w@J^n_{k_1}@ \cdots @J^n_{k_m})))$ where each J_{k_p} maps to a different v_{k_p} in G.

   Let $E_1, E_2, \cdots, E_{n+1}$ be (possibly empty) series of λ-elements. By the above condition of the algorithm for this case, $G = \lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})$ and $H = E_1 J^1 E_2 J^2 \cdots E_n J^n E_{n+1}$. Then, $F@G =$
   $(\lambda w \cdot E_1 w@J^1_{k_1}, @ \cdots @J^1_{k_m} E_2 w@J^2_{k_1} @ \cdots @J^2_{k_m} \cdots E_n w@J^n_{k_1} @ \cdots @J^n_{k_m} E_{n+1})$
   @ $\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})$
   $= E_1(\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})@J^1_{k_1} @ \cdots @J^1_{k_m})E_2 \cdots E_n$
   $(\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})@J^n_{k_1} @ \cdots @J^n_{k_m})E_{n+1}$
   $= E_1 J^1(J^1_1, \cdots, J^1_m)E_2, \cdots, J^n(J^n_1, \cdots, J^n_m)E_{n+1} = H$.
   J^i contains all common λ-elements to all J^i formulas from H, which sub-terms of each formula had been substituted by variables. Thus, when the formulas $J^i_{k_1}, \cdots, J^i_{k_m}$ belonging to a given formula are placed in the variables $v_{k_1}, \cdots, v_{k_m}$, one obtained the specific J^i formula back.
   The reason why it is stated in G that each J_{k_p} maps to a different v_{k_p} is due to the last step of the proof. For each occurrence of a specific variable v_{k_p} in G, one needs one formula J_{k_p} that will be placed on all the variables v_{k_p} during the application.

4. H is $\lambda v_1, \cdots, v_i \cdot J$ and $J^1(J^1_{i+1}, \cdots, J^1_s)$ is a sub-term of J, G is $\lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@\cdots@J^1_{k_s})$ with $\forall p,\ i+1 \leq k_p \leq s$.
   - $F = \lambda w \lambda v_1, \cdots, v_i \cdot (w@\lambda v_{i+1}, \cdots, v_s \cdot (J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s})))$

   Let E_1, E_2 be (possibly empty) series of λ-elements. By the above condition of the algorithm for this case, $H = \lambda v_1, \cdots, v_i \cdot E_1 J^1(J^1_{i+1}, \cdots, J^1_s)E_2$ and $G = \lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@, \cdots, @J^1_{k_s})$. Thus, $F@G = \lambda w \lambda v_1, \cdots, v_i \cdot (w@\lambda v_{i+1}, \cdots, v_s \cdot J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s}))@\lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@\cdots@J^1_{k_s}) = \lambda v_1, \cdots, v_i \cdot (\lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@\cdots@J^1_{k_s})@\lambda v_{i+1}, \cdots, v_s \cdot J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s}))$ $= \lambda v_1, \cdots, v_i \cdot (J(\lambda v_{i+1}, \cdots, v_s \cdot J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s}))@J^1_{k_1}@\cdots@J^1_{k_s})$ $= \lambda v_1, \cdots, v_i \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s)) = \lambda v_1, \cdots, v_i \cdot E_1(J^1(J^1_{i+1}, \cdots, J^1_s))E_2 = H$.

□

Theorem 2 (*Soundness Inverse$_R$*)
Given two typed λ-calculus formulas H and G in β-normal form, if $\mathit{Inverse}_R(H, G)$ returns a non-null value F, then $H = G @ F$.

Proof
Let G and H be two typed λ-calculus formulas, let $F = \mathit{Inverse}_R(H, G)$.

1. If G is $\lambda v \cdot v@J$
   - $F = \mathit{Inverse}_L(H, J)$

   One knows by the previous proof that if $\mathit{Inverse}_L(H, J)$ returns a non-null value K, then $H = K @ J$. By the given condition of the algorithm for this case, G is $\lambda v \cdot v@J$. Therefore, $G@F = \lambda v \cdot v@J @ K = K @ J = H$.
2. If J is a sub-term of H and G is $\lambda v \cdot H(J : v)$
   - $F = J$

   Let E_1 and E_2 be possibly empty series of λ-elements. By the above condition of the algorithm for this case, $H = E_1 J E_2$ and G $= \lambda v \cdot E_1(J : v)E_2$. Then, $G@F = (\lambda v \cdot E_1(J : v)E_2) @ J = E_1 J E_2 = H$.
3. G is not $\lambda v \cdot v@J$, $(J^1(J^1_1, \cdots, J^1_m), J^2(J^2_1, \cdots, J^2_m), \cdots, J^n(J^n_1, \cdots, J^n_m))$ are sub-terms of H such that for all i, $J^i(J^i_1, \ldots, J^i_m) = J^1(J^1_1, \ldots, J^1_m : J^i_1, \ldots, J^i_m)$ and G is $\lambda w \cdot H(J^1(J^1_1, \cdots, J^1_m), \cdots, J^n(J^n_1, \cdots, J^n_m) : (w@J^1_{k_1}, \cdots, @J^1_{k_m}), \cdots, (w@J^n_{k_1}, \cdots, @J^n_{k_m}))$ for some permutation $\{k_1, \ldots, k_m\}$ of $\{1, \ldots, m\}$.
   - $F = \lambda v_{k_1}, \cdots, v_{k_m} \cdot J^1(J^1_1, \cdots, J^1_m : v_1, \cdots, v_m)$.

   Let $E_1, E_2, \cdots, E_{n+1}$ be (possibly empty) series of λ-elements. By the above condition of the algorithm for this case, $G = \lambda w \cdot H((J^1(J^1_1, \cdots, J^1_m) : w@J^1_{k_1}@\cdots@J^1_{k_m}), \cdots, (J^n(J^n_1, \cdots, J^n_m) : w@J^n_{k_1}@\cdots@J^n_{k_m}))$ and $H = E_1 J^1 E_2 J^2 \cdots E_n J^n E_{n+1}$. Thus, $G @ F = (\lambda w \cdot E_1 w@J^1_{k_1}@\cdots@J^1_{k_m} E_2 w@J^2_{k_1}@\cdots@J^2_{k_m} \cdots E_n w@J^n_{k_1}@\cdots@J^n_{k_m} E_{n+1}) @ (\lambda v_{k_1}, \cdots, v_{k_m} \cdot J^1(J^1_1, \cdots, J^1_m : v_1, \cdots, v_k)$ $= E_1 \lambda v_{k_1}, \cdots, v_{k_m} \cdot (J^1(J^1_1, \cdots, J^1_m : v_1, \cdots, v_k))@J^1_{k_1}@\cdots@J^1_{k_m} E_2 \ldots E_n \lambda v_{k_1}, \cdots, v_{k_m} \cdot (J^1(J^1_1, \cdots, J^1_m : v_1, \cdots, v_k))@J^n_{k_1}@\cdots@J^n_{k_m} E_{n+1} = E_1 J^1(J^1_1, \ldots, J^1_m : J^1_1, \ldots, J^1_m)E_2 \ldots E_n J^1(J^1_1, \ldots, J^1_m : J^n_1, \ldots, J^n_m)E_{n+1} = E_1 J^1 E_2 \ldots E_n J^n E_{n+1}$ $= H$.

4. H is $\lambda v_1, \cdots, v_i \cdot J$ and $J^1(J^1_{i+1}, \cdots, J^1_s)$ is a sub-term of J, G is $\lambda w \cdot \lambda v_1, \cdots, v_i \cdot (w@\lambda v_{i+1}, \cdots, v_s \cdot (J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s})))$ with $\forall p,\ i+1 \leq k_p \leq s$.
   - $F = \lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@\cdots@J^1_{k_s})$

   Let E_1, E_2 be (possibly empty) series of λ-elements. By the above condition of the algorithm for this case, $H = \lambda v_1, \cdots, v_i \cdot E_1 J^1(J^1_{i+1}, \cdots, J^1_s)E_2$ and $G = \lambda w \cdot \lambda v_1, \cdots, v_i \cdot (w@\lambda v_{i+1}, \cdots, v_s \cdot J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s})))$. Thus, $G@F = \lambda w \lambda v_1, \cdots, v_i \cdot (w@\lambda v_{i+1}, \cdots, v_s \cdot J^1(J^1_{i+1}, \cdots, J^1_s : v_{k_1}, \cdots, v_{k_s}))\ @\ \lambda w \cdot J(J^1(J^1_{i+1}, \cdots, J^1_s) : w@J^1_{k_1}@\cdots@J^1_{k_s}) = H$.

$\square$

Theorem 3 (*Completeness of Inverse$_L$*)
For any two typed λ-calculus formulas H and G in β-normal form, where H is of order two or less, and G is of order one or less, if there exists a set of typed λ-calculus formulas Θ_F of order two or less in β-normal form, such that $\forall F_i \in \Theta_F$, $H = F_i@G$, then $Inverse_L(H, G)$ will give an F where $F \in \Theta_F$.

Proof
In Table 4 of the paper we demonstrated that there are 6 possible combinations of H, G and F. None of the combinations will satisfy option 4 of the operator since F needs to be order 3. Recall that the inverse operator has 4 possible cases, which we will refer to as "options". The assumption from the condition of the theorem is that $H = F@G$.

Proof by Contradiction.

- Case 1: Typed λ-calculus formulas H and G have order zero and F has order one.
  1. Assume that $Inverse_L(H, G) =$ null.
  2. Options 1 and 3 of the operator are not satisfied since G is order zero.
  3. We are left with option number 2. Lets assume that a formula G is not a sub-term of H.
  4. From the previous point, one knows that a formula G is not a sub-term of H.
  5. Suppose G is not a sub-term of H.
     — By definition of the input of the operator, G is order zero. By Lemma 1,H is formed by the sub-terms of G and F. If G is placed in the outermost variable of F, it will not receive any modification since G is order zero. This outermost variable is a λ-term and thus by definition a sub-term. Therefore, G is a sub-term of H. Contradiction.
- Case 2: Typed λ-calculus formulas H and F have order one and G has order zero.
  1. Assume that $Inverse_L(H, G) =$ null.
  2. Options 1 and 3 of the operator are not satisfied since G is order zero.

3. We are left with option number 2. Lets assume that a formula G is not a sub-term of H.
4. In the exactly same way as in the case 1 above, we can show that G is a sub-term of H which leads to a contradiction.

- Case 3: Typed λ-calculus formulas H and F have order two and G has order zero.

1. Assume that $\mathit{Inverse}_L(H, G) = \text{null}$.
2. Options 1 and 3 of the operator are not satisfied since G is order zero.
3. We are left with option number 2. Lets assume that a formula G is not a sub-term of H.
4. In the exactly same way as in the case 1 above, we can show that G is a sub-term of H which leads to a contradiction.

- Case 4: Typed λ-calculus formula H has order zero, G has order one and F has order two.

1. Assume that $\mathit{Inverse}_L(H, G) = \text{null}$
2. Option 2 of the operator cannot be satisfied. Since F has as input an order one formula, and a zero order as output, it needs to have an application to reduce the order. H has zero order, therefore it does not have any application. Thus, F cannot be formed from H and be of second order.
3. Lets assume that options 1 and 3 of the operator are not satisfied.
4. Option 1 of the operator is not satisfied if G is not of the form specified. One has two possible scenarios:
   — If G is of the form considered in option 1, then it is satisfied and the operator returns the specified F. Contradiction.
   — If G is not of the form considered in option 1, option 1 is not satisfied and we are left with option number 3.
5. Lets assume option 3 of the operator is not satisfied. For this to happen, some condition of option 3 cannot be satisfied. Thus, G is $\lambda v \cdot v$ and/or the formulas J^i are not sub-terms of H and/or $\forall\ J^i$, G is not of the form $\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_{k_1}, \cdots, v_{k_m})$.
6. G is not $\lambda v \cdot v$; this was shown in point 4 above.
7. By definition of sub-term, a formula H has at least one sub-term which is H itself. Therefore, H has at least a sub-term J^0 which is itself.
8. G is order one, therefore, it will start with a list of lambda abstractors. By definition of Typed lambda-calculus formulas, after the list of the lambda abstractors, one will have a formula, call it J.
   When G is applied to F, it is placed on the outermost variable. If there are more than one occurrences of this variable in F, one will have the formula G several times in the resulting formula H.
   H, output of F, can be order one or zero, therefore each occurrence of the variable in F will be in an applicator term with the formulas found in H for each occurrence of G generating different versions of G in H. If H is order two, then G will be order one, and in order to reduce the order of

G to zero in F so that one obtains a valid formula H, one will also need applicator terms in F. Each of these versions of G in H can be identified as J^i.
These J^i sub-terms can all have common λ-components depending on the structure of G. The sub-formulas of these J^i are what differentiates one J^i from another in H. Each of them has m sub-formulas J_i that will be placed in the variables of G. All J^i have the same number of J_i since they were all originated from the application of G to F, which has a definite number of variables. These sub-formulas belong to the applicators terms in F.
The variables in G that are bound to the initial list of abstractors can have any order or repetition inside G since there is no restriction on the structure of G. All stated to this point can be expressed as $G = \lambda v_1, \cdots, v_s \cdot J^i(J_1^i, \cdots, J_m^i : v_{k_1}, \cdots, v_{k_m})$.
This is the second condition of option 3. Contradiction.

- Case 5: Typed λ-calculus formula H and G have order one, F has order two.
  1. Lets assume that $Inverse_L(H, G) = \text{null}$.
  2. Option 2 of the operator cannot be satisfied. Thus, G cannot be a sub-term of H. One has two possible scenarios:
     — Suppose F is a second order formula without applicator terms. When G is applied to F, G is placed in the outermost variable of F and it becomes a sub-term of H. Option 2 is satisfied. Contradiction.
     — Suppose F is a second order formula with applicator terms. When G is applied to F, λ-abstractors and bound variables of G are not present in H when they are substituted by the formulas of the applicator terms of F. Therefore, G is not a sub-term of H and option 2 is not satisfied. One proceeds considering other options of the operator.
  3. By point 1 above, one also has that options 1 and 3 of the operator are not satisfied.
  4. Option 1 of the operator is not satisfied if G is not of the form specified. One has two possible scenarios:
     — If G is of the form considered in option 1, then it is satisfied and the operator returns the specified F. Contradiction.
     — If G is not of the form considered in option 1, option 1 is not satisfied and one continues considering the last possible option.
  5. Option 3 cannot be satisfied. By point 4, one knows that G is not a sub-term of H. By point 5, one knows that G is not $\lambda v \cdot v$. For option 3 to not be satisfied, some condition of option 3 cannot be satisfied. Thus, G is λv.v and/or the formulas J^i are not sub-terms of H and/or $\forall$ J^i, G is not of the form $\lambda v_1, \cdots, v_s \cdot J^i(J_1^i, \cdots, J_m^i : v_p, \cdots, v_q)$. If this is the case, then option 3 is not satisfied.
  6. In this step, one can apply the same reasoning that was shown in the previous case (case 4), with the only difference being that in this situation,

since H is order one, the number of formulas in the applicators of F will be at least one less than the number of initial λ-abstractors in F or G. And G and F have to obey the rules of ASP λ-calculus formulas with respect to formulas with connectors so that H is a valid ASP λ-calculus formula after the application of F to G.

- Case 6: Typed λ-calculus formula H and F have order two, G has order one.
  1. Assume that $Inverse_L(H, G) = \text{null}$.
  2. Option 2 of the operator cannot be satisfied. Thus, G cannot be a sub-term of H. One has two possible scenarios:
     — Suppose F is a second order formula without applicator terms. When G is applied to F, G is placed in the outermost variable of F and it becomes a sub-term of H. Option 2 is satisfied. Contradiction.
     — Suppose F is a second order formula with applicator terms. When G is applied to F, λ-abstractors and bound variables of G are not present in H when they are substituted by the formulas of the applicator terms of F. Therefore, G is not a sub-term of H and option 2 is not satisfied. One proceeds considering other options of the operator.
  3. By point 1 above, one also has that options 1 and 3 of the operator are not satisfied.
  4. Option 1 of the operator is not satisfied if G is not of the form specified. One has two possible scenarios:
     — If G is of the form considered in option 1, then it is satisfied and the operator returns the specified F. Contradiction.
     — If G is not of the form considered in option 1, option 1 is not satisfied and one continues considering the last possible option.
  5. Option 3 cannot be satisfied. By point 4, one knows that G is not a sub-term of H. By point 5, one knows that G is not $\lambda v \cdot v$. For option 3 to not be satisfied, some condition of option 3 cannot be satisfied. Thus, G is $\lambda v \cdot v$ and/or the formulas J^i are not sub-terms of H and/or $\forall\ J^i$, G is not of the form $\lambda v_1, \cdots, v_s \cdot J^i(J^i_1, \cdots, J^i_m : v_p, \cdots, v_q)$. If this is the case, then option 3 is not satisfied.
  6. In this step, one can apply the same reasoning that was shown in case 4 above, with the difference being that in this situation, since H is order two, the result of applying the formula G to F will give a second order formula as output.

□

Theorem 4 (*Completeness of* $Inverse_R$)
For any two typed λ-calculus formulas H and G of order two or less in β-normal form, if there exists a set of typed λ-calculus formulas Θ_F of order one or less in β-normal form, such that $\forall F_i \in \Theta_F$, $H = G@F_i$, then $Inverse_R(H, G)$ will give an F, where $F \in \Theta_F$.

Proof
There are 6 possible combinations of H, G and F with different orders as discussed previously. None of the combinations will satisfy rule 4 of the operator since G needs to be order 3. As before, we will refer to the various cases of the inverse operator as "options". The assumption from the condition of the theorem is $H = G@F$.

Proof by Contradiction:

- Case 1: Typed λ-calculus formulas H and F have order zero and G has order one.
  1. Lets assume that $Inverse_R(H, G) =$ null.
  2. Options 1 and 3 of the operator are not satisfied since G has to be order two to satisfy the conditions.
  3. Lets assume that option 2 of the operator is not satisfied.
  4. This means that a formula J is not a sub-term of H or/and G is not of the form $\lambda v \cdot H(J : v)$.
  5. Suppose that there is no sub-term J of H.

     — By definition of sub-term, H is a sub-term of H. Thus contradiction.
  6. Suppose G is not of the form $\lambda v \cdot H(J : v)$.

     — G is a typed λ-calculus formula formed by λ-elements $g_1, g_2, \cdots, g_n$. H is formed by $h_1, h_2, \cdots, h_n$. By point 5, J is a sub-term of H, therefore H is formed by $h_1, h_2, \cdots, h_i, j_1, \cdots, j_n, h_{i+1}, \cdots h_n$ (λ-elements can be empty). By Lemma 2, G is λv.$h_1, h_2, \cdots, h_i, v, h_{i+1}, \cdots, h_n$, which is in fact $\lambda v \cdot H(J : v)$. Contradiction.
- Case 2: Typed λ-calculus formula F has order zero and H, G has order one.
  1. Assume that $Inverse_R(H, G) =$ null.
  2. Options 1 and 3 of the operator are not satisfied since G has to be order two to satisfy the conditions.
  3. Lets assume that option 2 of the operator is not satisfied.
  4. This means that a formula J is not a sub-term of H or/and G is not of the form $\lambda v \cdot H(J : v)$.
  5. As in the above case, it was shown that J is a sub-term of H and that G is of the form $\lambda v \cdot H(J : v)$. Therefore, contradiction.
- Case 3: Typed λ-calculus formula F has order zero and H, G has order two.
  1. Assume that $Inverse_R(H, G) =$ null
  2. Option 1 is not satisfied since F is order zero and it cannot be applied to G applicator term.
  3. Option 3 of the operator is not satisfied since G needs to have applicator terms and H has the same order as G. The outermost variable of G could have an occurrence not in an applicator term somewhere in the formula, but then this variable would have a different type.
  4. Lets assume that option 2 of the operator is not satisfied.
  5. This means that a formula J is not a sub-term of H or/and G is not of the form $\lambda v \cdot H(J : v)$.

6. In case 1 above, it was shown that J is a sub-term of H and that G is of the form $\lambda v \cdot H(J : v)$. Therefore, contradiction.

- Case 4: Typed λ-calculus formula H has order zero, F has order one and G has order two.

1. Assume that $Inverse_R(H, G) = \text{null}$.
2. Option 2 of the operator is not satisfied because G is order two and H is order zero. G cannot be formed by H.
3. We are left with options 1 and 3.
4. Option 1 of the operator is not satisfied if G is not of the form specified. One has two possible scenarios:
   — If G is of the form considered in option 1, then it is satisfied and the operator returns the specified F. Contradiction.
   — If G is not of the form considered in option 1, option 1 is not satisfied and one continues considering the last possible option.

   G is order two and therefore it is receiving a function as input which is F. Therefore it will have a λ-abstractor at the beginning that binds to the variable where F will be placed. And since G is order two and H is order zero, G needs to have an application to reduce the order in the output. One has that option 1 is not satisfied, thus the expression in the application to the variable of G, next to its abstractor, cannot be a formula and nothing else.
5. By point 3 above, Option 3 of the operator is not satisfied. For this to happen, some condition of option 3 cannot be satisfied. Thus, G is $\lambda v \cdot v@J$ and/or J^i are not sub-terms of H and/or G is not $\lambda w \cdot H((J^1(J_i, \cdots, J_k) : w@J_p, \cdots, @J_q), \cdots, (J^n(J_i, \cdots, J_k) : w@J_p, \cdots, @J_q))$. If this is the case, then option 3 is not satisfied.
6. G cannot be $\lambda v \cdot v@J$; this was shown in the previous point 4.
7. By definition of sub-term, a formula H has at least one sub-term which is H. Therefore H has at least a sub-term J^0 which is itself.
8. G is a second order formula, therefore it is receiving a formula of order one or less as input. In order to do this, it needs an abstractor at the beginning of the formula that binds to the variable where the input formula will be placed. Denote the lambda abstractor by λw.

   H, output of G, can be order one or zero, therefore G will have occurrences of applicator terms whose variable will be bound to the initial abstractor to reduce the order. If H is order two, then F will be order one. In order to reduce the order of F to zero, so that one obtains a valid formula H, one will also need applicator terms. The number of formulas in the applicator terms depends on the number of variables in G. They will be equal if H is order zero and different if H is order one. Every occurrence of w will be in an applicator term for G to remain as a valid typed formula, since all occurrences of w need to have the same type as their corresponding abstractor.

   These J_i formulas have to be order zero in order for G to remain being

second order. By point 4 above, we know that G is not of the form $\lambda v \cdot v @ J$, which is the smallest possible second order formula which reduces the order of its input. Therefore it must have more λ-terms which are added using connectors.
By Lemma 1, H is formed by the subterms of G and F. Thus, in this case, all subterms of G are present in H. One has that the difference between the formula H and G is that H has as sub-terms, the formula F after being applied to the applicator terms of G. F is order one, therefore, it will have a list of abstractors at the beginning of the formula which bind to variables in the formula. When F is applied to G, it will be placed in the variable w of each applicator term and it will generate a formula when its variables are substituted by the formulas in the applicator term. These formulas will be part of H and each of them can be identified as J^i. Each of them will have common elements that belong to F, since F is only changing by the formulas of G placed on its variables.
All applicator terms will have the same number of formulas since they are all applied to the same F which has a specific number of variables. And since the variables in G can be in any order or repetition, the formulas of the applicators have to be as well.
All stated to this point can be expressed as: $\lambda w \cdot H((J^1(J_1, \cdots, J_m) : w @ J_{k_1}, \cdots, @ J_{k_m}), \cdots, (J^n(J_1, \cdots, J_m) : w @ J_{k_1}, \cdots, @ J_{k_m}))$. This is the last condition of option 3. Contradiction.

- Case 5: Typed λ-calculus formulas F and H have order one, G has order two.
  1. Assume that $\mathit{Inverse}_R(H, G) = \text{null}$.
  2. Option 2 of the operator cannot be satisfied. One has two possible situations:
     — G is order two and H is order one. If G has no applicator terms, then G can be formed by H. And, as shown in case 1, this leads to the condition being satisfied and thus, contradiction.
     — If G has applicator terms, then it cannot be formed by an order one formula H. Option 2 is not satisfied and one proceeds with the next possible options.
  3. Lets assume options 1 and 3 of the operator are not satisfied.
  4. Option 1 of the operator is not satisfied if G is not of the form specified. One knows by point 2 that G has at least one applicator term. One has two possible scenarios:
     — If G is of the form considered in option 1, then it is satisfied and the operator returns the specified F. Contradiction.
     — If G is not of the form considered in option 1, option 1 is not satisfied and one continues considering the last possible option.
  5. Option 3 of the operator cannot be satisfied. For this to happen, some condition of option 3 cannot be satisfied. Thus, G is $\lambda v \cdot v @ J$ and/or J^i are not sub-terms of H and/or G is not $\lambda w \cdot H((J^1(J_i, \cdots, J_k) : w @ J_p, \cdot\cdot$

$\cdot, @J_q), \cdots, (J^n(J_i, \cdots, J_k) : x@J_p, \cdots, @J_q))$. If this is the case, then option 3 is not satisfied.

6. One can apply the same reasoning that was shown in the previous case (case 4) with the only difference being that in this case since H is order one, the number of formulas in the applicators of G needs to be at least one less than the number of initial λ-abstractors in F or G.

- Case 6: Typed λ-calculus formulas G and H have order two, F has order one.
  1. Assume that $Inverse_R(H, G) = \text{null}$.
  2. Option 1 of the operator cannot be satisfied. G is receiving as input an order one formula F to which a formula J will be applied. The output of G is H of order two. In order for J applied to F to return a formula of order two is if J was order two. However, then G would become order three.
  3. Option 2 of the operator cannot be satisfied. G is receiving as input the formula F of order one. In order to give a valid output of order two, G needs to have some additional sub-term of order two and reduce the order of the input F to zero so that the resulting expression is a formula. But, since F is placed in a variable and not in an applicator term, its order cannot be reduced.
  4. Option 3 cannot be satisfied. The reasoning for this option is the same as that in the previous case and it leads to a contradiction.

□

Theorem 5 (*Inverse$_L$ complexity*)
The *Inverse$_L$* Algorithm runs in exponential time in the number of variables in G and polynomial time in the size of the formulas H and G.

Proof
Case 3 gives the worst case complexity. Let k be the number of variables in G, then we need to find all permutations of length k, since they are the parts in H we need to substitute. In the worst case H is a big conjunction terms. First, we find all subterms of H (on the order of the size of H). Then we would try to check if those subterms have k subterms that we can substitute with the variables in G to end up with G (on the order of the size of G). There are then in the worst-case on the order of $|H|!/(|H|-k)!$ permutations we need to check per subterm. Thus, the worst case complexity of *Inverse$_L$* is $O((|H|!/(|H|-k)!)(|H|+2|G|))$. However, if k is small, as it is in most practical examples, then $|H|!/(|H|-k)!$ is approximately $O(|H|^k)$. □

Theorem 6 (*Inverse$_R$ complexity*)
The *Inverse$_R$* Algorithm runs in exponential time in the number of variables in G and polynomial time in the size of the formulas H and G.

Proof
Since Case 1 of $\mathit{Inverse}_R$ uses $\mathit{Inverse}_L$ its worst case complexity can be no better than the complexity of $\mathit{Inverse}_L$. All other cases of $\mathit{Inverse}_R$ run in polynomial time. Note case 3 of $\mathit{Inverse}_R$ has polynomial complexity since the formulas given as input to the applications in G can be used to find the subterms of H that can generate them. Therefore, we do not have to search all permutation of the subterms as in $\mathit{Inverse}_L$. Thus, the worst case complexity is given in Case 1 to be the same as $\mathit{Inverse}_L$, $O((|H|!/(|H|-k)!)(|H|+2|G|))$. $\square$