INVERSION FUNCTION OF MDS FOR SENTENCES ANALYSIS

Erqing Xu
Department of Mathematics, Cornell University, Ithaca, New York 14853, U.S.A.

Keywords: Natural language understanding, MDS, Inversion function, Proof tree.

Abstract: Traditional sentence analysis refers to finding the sentence structure for a given sentence. A question different from this is: given a sentence Curry-Horwad isomorphic with a type, can we establish the proof tree representing the sentence? Therefore, this paper combines the extensional Kripke interpretation and MDS (Minimalist Deductive System); derives the Kripke model of MDS; provides the applicable inversion function such that we are able to obtain the proof tree of typed λ-terms which represents sentence structure; and demonstrates that the product-free proof trees obtained with inversion function of MDS enjoy the property of Church-Rosser equality. Application examples demonstrate that our work is valid. The main difference between our work and traditional sentence analysis approach is that the objects of analysis are different. The object of our work is: Kripke model of MDS and type of sentence satisfied by assignment. But the object of traditional sentence analysis approach is sentence. This paper enlarges the range of application of sentence analysis, improves sentence analysis approach, enhances natural language understanding, and thus is meaningful. Our work has not been seen in literature.

1 INTRODUCTION

In natural language understanding, parsing as logic deduction has become one of the hot topics of research. Minimalist Deductive System is a late approach (Lecomte, 2004). In MDS calculus, a sentence is Curry-Horwad isomorphic with a type. The feature of sentence analysis with MDS is that the establishment of proof tree is type-driven. Then we may naturally have the question: for a given type of sentence, can we establish the proof tree representing the sentence? This question is meaningful for the improvement of sentence analysis and natural language understanding.

Coquand (2002) forwards inversion function of simple type λ-calculus. This inversion function is able to return typed λ-terms according to the given type. However, inversion function relies on specific Kripke model. The Kripke model of MDS has not been seen. Therefore, in order to obtain the inversion function of MDS, first we have to obtain the Kripke model of MDS. Now we already have Kripke model of intuitionnistic logic, and MDS is a fragment of partially commutative linear logic. Since the difference between linear logic and intuitionistic logic is the absence of contraction and weakening (Morrill, 1994), it is hopeful that Kripke model of intuitionnistic logic becomes the Kripke model of MDS.

The work of this paper is: 1. combining the extensional Kripke interpretation and MDS to derive the Kripke model of MDS; providing the applicable inversion function for MDS calculus of types. 2. forwarding the method of representing the result of inversion function, i.e. typed λ-terms as a proof tree. 3. demonstrating product-free proofs obtained by inversion function enjoys the property of strong normalization. For MDS, the above-mentioned work has not been seen in literature.

Comparison between the work of this paper and related work is as follows:

The main difference between our work and traditional sentence analysis approach is that the objects of analysis are different. The object of our work is: Kripke model of MDS and type of sentence satisfied by assignment. But the object of traditional sentence analysis approach is sentence.

The difference between our work and inversion function of simple type λ-calculus is: 1. The calculus is different. MDS calculus in this paper is linear logic calculus embodying the minimalist grammar, which is resource sensitive. Simple type λ-calculus is pure typed λ-calculus, which is intuitionnistic logic. Our work is applicable to Kripke model of

Xu E. (2010).
INVERSION FUNCTION OF MDS FOR SENTENCES ANALYSIS .
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Artificial Intelligence*, pages 151-156
DOI: 10.5220/0002590401510156

MDS and sentences satisfied by assignment, while the latter is applicable to pure typed semantic objects.

The organization of the rest of this paper is: 2. Preliminaries, 3. Kripke model and Inversion function for MDS, 4. Representing the result of inversion function as proof tree, 5. Church-Rosser equality of the result of inversion function, and 6. Conclusion.

2 PRELIMINARIES

Definition 1. (Type) (Hindley & Seldin 1986) Assume that we have been given some symbols called atomic types; then we define types as follows:

(a)each atomic type is a type;

(b)if α and β are types, then (α →β) is a type. Each type (α →β) is called a compound type.

Definition 2. (Typed λ-terms) (Hindley & Seldin 1986) For each type α, assume that we have infinitely: many variables v:α of type α, and perhaps some constants c:α of type α; then we define typed λ-terms as follows:

(a) each v: α and c: α is a typed λ-term of type α;

(b) if N: α →β and N: α are typed λ-terms of types α →β and α respectively, then MN:β is a typed λ-term of type β;

(c) if x: α is a variable of type α and M: β is a typed λ-term of type β, then (λx.M): α →β is a λ-term of type α →β.

Definition 3. (MDS) (Lecomte, 2004) MDS is composed of lexical entries and rules.

Generally speaking, a lexical entry consists in an axiom

$$\vdash w: T$$

where *T* is of the following type:

$((F_2\backslash (F_3\backslash\ldots(F_n\backslash(G_1\otimes G_2\otimes\ldots\otimes G_m\otimes A)/))))/F_1)$

where,

m and n can be any number greater than or equal to 0,

$F_1, \ldots, F_n$ are attractors,

$G_1, \ldots, G_n$ are features,

A is the resulting category type. (Lecomte, 2004)

There are nine rules in MDS, which are illustrated in Figure 1.

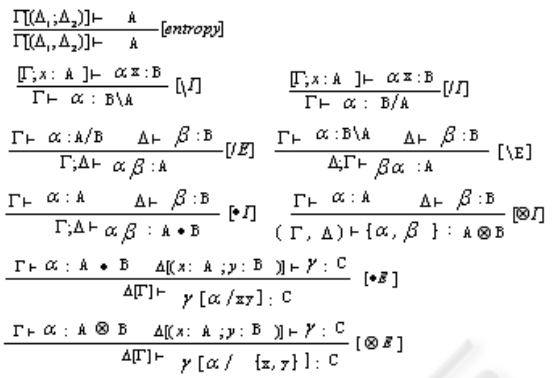

Figure 1: Rules of MDS.

Definition 4. (Ranta 1994) A context, in the technical sense of type theory, is a sequence of hypotheses of the form

$$x_1:A_1, x_2:A_2(x_1),\ldots, x_n:A_n(x_1,\ldots,x_{n-1}).$$

where the judgment x:A which introduces a variable, is a hypothesis.

Definition 5. (Coquand 2002) The set of semantic objects is defined as usual in Kripke semantics:

$$\frac{A\in T \quad \omega\in W}{\mathrm{Force}(\omega, A)\in \mathrm{Set}}$$

Force(ω, A)∈Set is written ω⊩ A, where T∈Set is the set of types and W is the set of possible worlds.

Note that Kripke interpretation is sometimes called Kripke model. (Wang 1997)

Definition 6. (Simpson 1992) The extensional Kripke interpretation is a sextuple:

$$\prec W,\ \leq, \{[\![A]\!]_\omega\}, \{[\![P]\!]_\omega\}, \{\mathcal{E}_\omega^{A,B}\}, \{\mathcal{E}_{\omega\omega'}^{A}\} \succ$$

where

•W is a set of possible worlds with a partial ordering, ≤.

•{ $[\![A]\!]_\omega$} is a family of sets, with $[\![A]\!]_\omega$, indexed by types, A, and possible worlds, ω.

•{ $[\![P]\!]_\omega$} is a family of relations, $[\![P]\!]_\omega \subseteq [\![A_1]\!]_\omega \times \ldots \times [\![A_n]\!]_\omega$, indexed by predicate symbols, P, with decorations, P: $<A_1,\ldots, A_n>$ and possible worlds, ω.

• $\{\mathcal{E}_\omega^{A,B}\}$ is a family of functions, $\mathcal{E}_\omega^{A,B}$: $[\![A\to B]\!]_\omega \times [\![A]\!]_\omega \to [\![B]\!]_\omega$.

• $\{\mathcal{E}_{\omega\omega'}^{A}\}$ is a family of functions, $\mathcal{E}_{\omega\omega'}^{A}$: $[\![A]\!]_\omega \to [\![A]\!]_{\omega'}$, indexed by types, A, and pairs of possible worlds, ω′≥ω.

The extensional Kripke interpretation is simply denoted as W.

An inversion function, given a semantic object in a particular Kripke model, returns a proof tree. The function is defined together with function ***val*** that intuitively takes a proof tree of the form of an variable applied to zero or more arguments. The definition is as follows.

In this paper, the extensional Kripke interpretation is taken as the particular Kripke model.

Definition 7. (Coquand 2002) Let a proof of type

$$M\in[\ ]\vdash A_1\to A_2\ldots\to A_n\to o$$

where A_i, i =1,2,…,n, is types. An inversion function, denoted as ***reify***, is defined as

$reify([[M]])\equiv\lambda(z_1{:}A_1)\ldots\lambda(z_n{:}A_n).\{[[M]]val(z_1)\ldots val(z_n)\}$

where $z_1\ldots z_n$ are fresh names, i.e.

$z_1\equiv gensym(\Gamma)$,

…,

$z_n\equiv gensym([\Gamma,z_1{:}A_1,\ldots, z_{n-1}{:}A_{n-1}])$

here $gensym\in(\Gamma\in C)$Name, C is set of contexts, Name is a countably infinite set, and

$$[[M]]val(z_1)\ldots val(z_n)$$

is a proof tree of type o, atomic type. If x is a variable of type $A_1\to\ldots\to A_k\to B$, then

$val(x)=\Lambda([v_1]\ldots\Lambda([v_k](x\ reify\ (v_1)\ldots(reify(v_n)))$

where Λ is the simplified interpretation of abstraction.

Note. The set of semantics objects $\omega\Vdash A$ in Definition 2 is the same as $\{\ [\![A]\!]_\omega\}$ in Definition 3. It is denoted as [[A]] in inverse function.

Next, Definitions 8-10 define sentences satisfied by the extensional Kripke model and assignment.

Definition 8. (Coquand 2002) Suppose C is the set of contexts. The set of environments is defined as

$$\frac{\Gamma\in C\quad \omega\in W}{Force\_env(\omega,\Gamma)\in Set}$$

where each variable in a context is associated with a semantic object. $Force\_env(\omega,\Gamma)\in Set$ is written $\omega\Vdash\Gamma$.

Note. Environment is sometimes called assignment.

Definition 9. (Coquand 2002) The interpretation for proof tree of types in a given environment is defined as:

$$[[\ ]]_{term}\in(\Gamma\vdash A;\ \omega\Vdash\Gamma)\omega\Vdash A.$$

Definition 10. (Wang 1997) It is inductively defined as follows that in the extensional Kripke interpretation of a formula of type, α, is satisfied by the environment $\omega\Vdash\Gamma$ at possible $\omega\in W$ (denoted by $\omega\vDash\alpha$):

(1)when α is $P(A_1,\ldots, A_n)$, where P is predicate variable, A_i , i =1,…,n, is a type, and $[[P]]_\omega$ is defined in Definition 2, for all $\omega'\geq\omega$,

$$\omega\vDash P(A_1,\ldots, A_n)\ \text{ iff }\ <[[A_1]]_{term},\ldots,[[A_n]]_{term}>\in[[P]]_{\omega'}$$

(2)when α is $\alpha_1\wedge\alpha_2$,

$$\omega\vDash\alpha_1\wedge\alpha_2\ \text{iff for all}\ \omega'\geq\omega,\ \omega\vDash\alpha_1\ \text{and}\ \omega\vDash\alpha_2.$$

(3)when α is $\alpha_1\to\alpha_2$,

$$\omega\vDash\alpha_1\to\alpha_2\ \text{iff for all}\ \omega'\geq\omega,\ \text{if}\ \omega'\vDash\alpha_1,\ \text{then}\ \omega'\vDash\alpha_2.$$

Definition 11. (Wang 1997; Coquand 2002) Suppose α be a type lambda formula. If $\omega\vDash\alpha$, then α is called K-satisfiable.

3 KRIPKE MODEL AND INVERSION FUNCTION FOR MDS

3.1 Kripke Model for MDS

It is composed of the following six components.

(1) Possible worlds. The world of mind can be seen as a possible world. (Jiang & Pan 1998) The possible world is denoted as w, and W={ w}. Context is denoted as Γ. Contexts in Definition 4 is taken as possible worlds in Definition 6, that is, w=Γ. The possible world includes the set of typed λ-terms representing words and sentences.

(2) If $\Gamma\subseteq\Gamma'$, then $w\leq w'$.

(3) The set of semantic objects, $[\![A]\!]_\omega$, in Definition 6 is λ-terms for lexical entries at w. And each variable in a context is associated with a semantic object. $\{\ [\![A]\!]_\omega\}$ is the set of $[\![A]\!]_\omega$ in all possible worlds.

(4) $[\![P]\!]_\omega$ is the products of types in the possible world w, and they occur in rule 7-9 of MDS. $\{\ [\![P]\!]_\omega\}$ is the set of $[\![P]\!]_\omega$ in all possible worlds.

(5) ε_ω^{AB} : $[\![A\to B]\!]_\omega\times[\![A]\!]_\omega\to[\![B]\!]_\omega$ means Definition 2 (c) at w.

(6) $\varepsilon_{\omega\omega'}^{A}$: $[\![A]\!]_\omega\to[\![A]\!]_{\omega'}$ means that if $[\![A]\!]$ holds at w, then $[\![A]\!]$ holds for all $w'\geq w$.

3.2 Inversion Function for MDS

Definition 7 including symbols with the following denotations leads to the inversion function for MDS.
1.[[M]].
Given the type of sentence is M. Let

$$M \in [\ ] \vdash A_1 \rightarrow A_2 \ldots \rightarrow A_n \rightarrow o.$$

For each A_i , i =1,…,n, its semantics object, [[A_i]] , is a variable z_i which is a λ-term with type in any context w′≥w. At any w′≥w, type M is mapped to its semantics object, [[M]], which is λ-expression with type for sentence.
2. reify([[M]])
reify([[M]]) ≡ λ(z_1:A_1)…λ(z_n:A_n).{[[M]]val(z_1)…val (z_n)}
is λ-expression with n bound variables z_1 ,…, z_n: of types A_1, …, A_n , respectively where val(x) ia as follows. If x is a variable of type $A_1 \rightarrow \ldots \rightarrow A_k \rightarrow B$, then there are k bound variables in x, v_i=[[A_i]], i =1,…,k at any w′≥w . Thus,

val(x)=Λ([v_1]…Λ([v_k](x reify (v_1)…(reify(v_n)))
is an variable x applied to k arguments v_1,…,v_k such that val(x) is semantics object at any w′≥w.

3.3 Application of Inverse Function for MDS

We take an example to show how the inverse function returns λ - expression with type representing a sentence.

Let M∈[ ]⊢(t→t)→t→t be a type of sentence. From lexical entries of MDS, a list of lexical entries appeared in the example is as follows.

$$\alpha_1 = \lambda v.\text{seem}(v): t \rightarrow t \quad (1)$$
$$\alpha_2 = \alpha_3\alpha_4 = \text{approach}(\text{mary}): t \quad (2)$$
$$\alpha_3 = \lambda u.u(\text{mary}): (e \rightarrow t) \rightarrow t \quad (3)$$
$$\alpha_4 = \lambda y.\text{approach}(y): e \rightarrow t \quad (4)$$
$$\alpha_5 = x: e \quad (5)$$
$$\alpha_4\alpha_5 = \text{approach}(x): t \quad (6)$$
$$\lambda\alpha_5.\ \alpha_4\alpha_5 = \lambda x.\text{approach}(x): e \rightarrow t \quad (7)$$
$$\alpha_3\lambda\alpha_5.\ \alpha_4\alpha_5 = \text{approach}(\text{mary}) \quad (8)$$
$$\alpha_1\alpha_4\alpha_5 = \text{seem}(\text{approach}(x)): t \quad (9)$$
$$\lambda\alpha_5.\alpha_1\alpha_4\alpha_5 = \lambda x.\text{seem}(\text{approach}(x)): e \rightarrow t \quad (10)$$
$$\alpha_3(\lambda\alpha_5.\alpha_1\alpha_4\alpha_5) = \text{seem}(\text{approach}(\text{mary})): t \quad (11)$$
$$\lambda\alpha_5.\ \alpha_4\alpha_5 = \lambda x.\text{approach}(x): e \rightarrow t \quad (12)$$

From 3.2,

reify([[M]]) ≡ λ(α_1:t→t).λ(α_2:t).[[$\alpha_1\alpha_2$]](α_1=val(α_1)α_2=val(α_2)) [[$\alpha_1\alpha_2$]]{α_1=val(α_1)α_2=val(α_2)}
≡app(val(α_1), val(α_2))
≡app(Λ[v](α_1 reify(v))), val(α_2))
≡ α_1 reify (val(α_2)) (13)

where ‘app’ is for application of λ-calculu, and last equation above is due to that Λ[v](α_1 reify(v))) is applied to val(α_2). Because (2),in α_2 ,there are two arguments, α_3 and α_4 , therefore

reify(val(α_2)) ≡ λ(α_3: (e→t)→t).λ(α_4: e→t).([[$\alpha_3\alpha_4$]] val(α_3).val(α_4))=[[$\alpha_3\alpha_4$]]{ α_3=val(α_3), α_4=val(α_4)}
≡app(val(α_3), val(α_4))
≡app(Λ[v](α_3 reify(v))), val(α_4))
≡ α_3 reify(val(α_4)) (14)

Because α_4, e→t, in (4) is a compound type, its range is t and its domain is one argument, e. ∴

reify(val(α_4))≡λ(α_5: e).app(val(α_4), val(α_5))
≡λ(α_5 :e)app(Λ([v] $\alpha_4$reify(v))), α_5)
≡λ(α_5: e). α_4 reify(α_5)
≡λ(α_5: e)($\alpha_4\alpha_5$) (15)

‘reify(val(α_4))’ in the result of (14) is replaced by (15), and ‘reify (val(α_2))’ in the result of (13) is replaced by (14), it is obtained that

Reify([[M]])= λ(α_1: t→t).λ(α_2: t).($\alpha_1\alpha_3$ λ(α_5:e).$\alpha_4\alpha_5$) (16)

α_1 to α_5 in (16) are replaced by (1)-(5),respectively, it is obtained that

reify([[M]])

=(λv.seem(v): t→t) λ(approach(mary): t) λv.seem(v) λu.u(mary) λ(x:e) λy.approach(y) x
=λ(λv.seem(v): t→t). λ(approach(mary): t) (λv.seem(v). approach(mary)) (16′)

The inverse function results in (16′), the typed λ-expression representing a sentence. (16′) is equivalent to proof tree of sentence. (16′) can take as the form of proof tree shown in the next section.

4 REPRESENTING THE RESULT OF INVERSION FUNCTION AS PROOF TREE

The method of representing the λ-terms obtained with inversion function as a proof tree is as follows: The λ-terms obtained in respective steps of the application of the inversion function are transformed into sub-proof trees. If the λ-terms obtained in a certain step are juxtaposition, then transform the result into a deductive sub-proof tree of application illustrated by Definition 2(b). If a certain step introduces a new variable, then transform the result

into a deductive sub-proof tree of abstraction illustrated by Definition 2(c). Combine all the sub-proof trees and we have the final proof tree.

We take（16）as an example to illustrate the process of the derivation of the proof tree.

(13)/(16):

$$\text{reify}([[M]]) \equiv (\alpha_1)\ \text{reify}(\text{val}(\alpha_2))$$

Its type is t. Since α_1 and reify $(\text{val}(\alpha_2))$ are juxtaposition, then transform them into a deductive sub-proof tree of application. We have deduction (17)

$$\frac{\alpha_1 : t \to t \qquad \text{reify}(\text{val}(\alpha_2)): t}{\alpha_1\alpha_3\ (\lambda\alpha_5.\ \alpha_4\alpha_5): t} \qquad (17)$$

From (14),

$$\text{reify}(\text{val}(\alpha_2)) = (\alpha_3)\text{reify}(\text{val}(\alpha_4))$$

Its type is e→t. Since α_3 and reify $(\text{val}(\alpha_4))$ are juxtaposition, we have similar situation. Then we have deduction (18)

$$\frac{\alpha_3 : (e \to t) \to t \qquad \text{reify}(\text{val}(\alpha_4): e \to t}{\text{reify}\ (\text{val}(\alpha_2)): t} \qquad (18)$$

Use $\text{reify}(\text{val}(\alpha_4)) = \lambda(\alpha_5: e)(\ \alpha_4\alpha_5)$ in (18), and we have deduction (19)

$$\frac{\alpha_3 : (e \to t) \to t \quad \lambda(\alpha_5: e)(\ \alpha_4\alpha_5) : e \to t}{\text{reify}\ (\text{val}(\alpha_2)): t} \qquad (19)$$

Use (19) to substitute $\text{reify}(\text{val}(\alpha_2))$ in (17), and we have (20).

$$\frac{\alpha_1 : t \to t \qquad \dfrac{\alpha_3 : (e \to t) \to t \quad \lambda\alpha_5.\ \alpha_4\alpha_5: e \to t}{\alpha_3\ (\lambda\alpha_5.\ \alpha_4\alpha_5): t}}{\alpha_1\alpha_3\ (\lambda\alpha_5.\ \alpha_4\alpha_5): t} \qquad (20)$$

$\lambda\alpha_5.\ \alpha_4\alpha_5: e \to t$ in （15） and （20） can be obtained in proof (21). In （21）, the upper deduction is application, and the lower deduction is λ-abstraction.

$$\frac{\dfrac{[\alpha_4: e \to t]^2 \qquad [\alpha_5{:}e]^1}{\alpha_4\alpha_5 : t}}{\lambda\alpha_5.\ \alpha_4\alpha_5: e \to t} \qquad (21)$$

Combine（17）-（21）, and we have (22）

$$\frac{\alpha_1 : t \to t \qquad \dfrac{\alpha_3 : (e \to t) \to t \quad \lambda\alpha_5.\ \alpha_4\alpha_5: e \to t}{\alpha_3\ (\lambda\alpha_5.\ \alpha_4\alpha_5): t}}{\alpha_1\alpha_3\ (\lambda\alpha_5.\ \alpha_4\alpha_5): t} \qquad (22)$$

Replace α_1 through α_5 with the actual λ -terms representing the lexical items, and we have:

$$\dfrac{\lambda v.\ \text{seem}(v) : t \to t \qquad \dfrac{\lambda u.\ u(\text{mary}) : (e \to t) \to t \qquad \dfrac{[\lambda y.\ \text{approach}(y) : e \to t]^2 \qquad [x:e]^1}{\text{approach}(x) : t}}{\text{approach}(\text{mary}) : t}}{\text{seem}\ (\text{approach}(\text{mary})) : t}$$

which can be represented as the following proof tree:

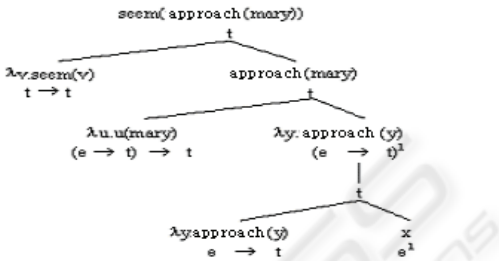

Figure 2: “It seems that Mary approaches”.

5 CHURCH-ROSSER EQUALITY OF THE RESULT OF INVERSION FUNCTION

Now refer to another proof which is equivalent in the sense of Church-Rosser equality.

$\beta_1 = \lambda u.u(\text{mary}): (e \to t) \to t$

$\beta_2 = \lambda\beta_6.\ \beta_3\beta_5\beta_6 = \text{seem}(\text{approach}(x)): e \to t$

$\beta_3 = \lambda v.\text{seem}(v): t \to t$

$\beta_4 = \beta_5\beta_6 = \text{approach}(x): t$

$\beta_5 = \lambda y.\text{approach}(y): e \to t$

$\beta_6 = x: e$

$\beta_3\beta_5\beta_6 = \text{seem}(\text{approach}(x)): t$

$\beta_1\ (\lambda\beta_6.\ \beta_3\beta_5\beta_6) = \text{seem}(\text{approach}(\text{mary})): t$

Let $M \in [\ ] \vdash ((e \to t) \to t) \to (e \to t) \to t$ be a proof tree of types.

Then,

Reify([[M]])

$\equiv \lambda(\beta_1: (e \to t) \to t)\ \lambda(\beta_2: e \to t)\ ([[\beta_1\beta_2]]\ \text{val}(\beta_1).\text{val}(\beta_2))$

$\equiv [[\ \beta_1\beta_2]]\{\ \beta_1 = \text{val}(\beta_1),\ \beta_2 = \text{val}(\beta_2)\}$

$\equiv \text{app}(\text{val}(\beta_1),\ \text{val}(\beta_2))$

$\equiv \text{app}(\Lambda[v](\ \beta_1\text{reify}(v))),\ \text{val}(\beta_2)) = \beta_1.\text{reify}(\text{val}(\beta_2)$

reify$(\text{val}(\beta_2))$

$\equiv \lambda(\beta_6{:}e).\text{app}(\text{val}(\beta_3), \text{val}(\beta_4))$

$\equiv \lambda(\beta_6{:}e)\text{app}(\Lambda([v]\beta_3\text{reify}(v))),\ \beta_4)$

$\equiv \lambda(\ \beta_6{:}\ e).\ \beta_3\ \text{reify}(\beta_4)$

$\therefore \text{reify}(\text{val}(\beta_4))$

$\equiv \lambda\ (\beta_5{:}\ e \to t)\ \lambda(\beta_6{:}\ e)\ [[\beta_5.\ \beta_6]]\ \{\beta_5 = \text{val}(\beta_5),\ \beta_6 = \text{val}(\beta_6)\}$

$\equiv \text{app}(\text{val}(\beta_5),\ \text{val}(\beta_6)) = \text{app}(\Lambda[v](\ \beta_5\ \text{reify}(v))),\ \text{val}(\beta_6))$

$\equiv \beta_5\ \text{reify}\ (\text{val}(\beta_6)) = \beta_5\beta_6$

∴reify[[M]]
≡ $\lambda(\beta_1: (e\rightarrow t)\rightarrow t)\ \lambda(\beta_2: e\rightarrow t)\ \beta_1\lambda(\beta_6{:}e)\ \beta_3\beta_5\beta_6$

Replace β_1 through β_6 with lexical items, and we have:

reify([[M]])
≡ $\lambda(\lambda u.u(mary){:}(e\rightarrow t)\rightarrow t).\lambda(seem(work(x)){:}e\rightarrow t).(\lambda u.u(mary)\lambda(x{:}e)\ \lambda v.seem(v)\ \lambda y.work(y)\ x)$

Now we transform the computational result of the inversion function into a proof tree.

From reify[[M]] ≡ $\beta_1.(reify(val(\beta_2)){:}\ e\rightarrow t)$, we have deduction (23)

$$\frac{\beta_1 : (e\rightarrow t)\rightarrow t \quad reify(val(\beta_2)){:}\ e\rightarrow t}{\beta_1\ (\lambda\beta_6.\ \beta_3\beta_5\beta_6){:}\ t} \qquad (23)$$

From $reify(val(\beta_2))= \lambda(\ \beta_6{:}\ e).\ \beta_3\ reify(val(\beta_4))$, we have deduction (26)

$$\frac{\beta_1 : (e\rightarrow t)\rightarrow t \quad \lambda\beta_6.\ \beta_3\ reify(val(\beta_4)){:}\ e\rightarrow t}{\beta_1\ (\lambda\beta_6.\ \beta_3\beta_5\beta_6){:}\ t} \qquad (24)$$

From $reify(val(\beta_4))= \beta_5\beta_6{:}\ t$ and (8), we have (25)

$$\frac{\beta_1 : (e\rightarrow t)\rightarrow t \quad \lambda(\ \beta_6{:}e).\ \beta_3\beta_5\beta_6{:}\ e\rightarrow t}{\beta_1\ (\lambda\beta_6.\ \beta_3\beta_5\beta_6){:}\ t} \qquad (25)$$

where $\lambda(\ \beta_6{:}e).\ \beta_3\beta_5\beta_6{:}\ e\rightarrow t$ can be derived from (26):

$$\dfrac{\dfrac{[\beta_5{:}\ e\rightarrow t]^2 \quad [\beta_6{:}e]^1}{\beta_3{:}\ t\rightarrow t \quad \beta_5\beta_6 : t}}{\dfrac{\beta_3\beta_5\beta_6 : t}{\lambda\beta_6.\ \beta_3\beta_5\beta_6{:}\ e\rightarrow t}} \qquad (26)$$

∴From（25） and （26） we have （27）

$$\dfrac{\beta_1 : (e\rightarrow t)\rightarrow t \quad \dfrac{\dfrac{[\beta_5{:}\ e\rightarrow t]^2 \quad [\beta_6{:}e]^1}{\beta_3{:}\ t\rightarrow t \quad \beta_5\beta_6 : t}}{\beta_3\beta_5\beta_6 : t}}{\beta_1\ (\lambda\beta_6.\ \beta_3\beta_5\beta_6){:}\ t} \qquad (27)$$

Replace β_1 through β_6 with the actual λ -terms representing the lexical items, and we have:

$$\dfrac{\lambda u.\ u(mary) : (e\rightarrow t)\rightarrow t \quad \dfrac{\lambda v.\ seem(v) : t\rightarrow t \quad \dfrac{[\lambda y.\ approach(y) : e\rightarrow t]^2 \quad [x{:}e]^1}{approach(x) : t}}{seem\ (approach(x)) : t}}{seem\ (approach(mary)) : t}$$

Figure 3: seem(approach(mary)): t.

which can be represented as the following proof tree:

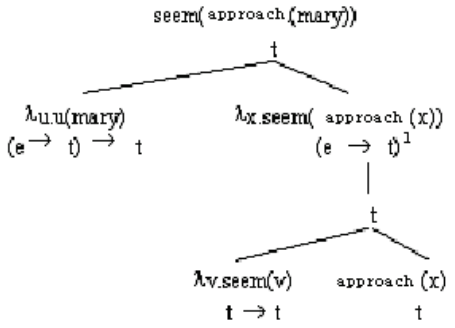

Figure 4: “Mary seems to approach”.

6 CONCLUSIONS

This paper realizes the establishment of the proof tree representing the sentence according to the sentence type with inversion function. Our work is applicable to Kripke model of MDS and types of sentences satisfied by assignment. Application examples demonstrate that our work is valid. This paper enlarges the range of application of sentence analysis, improves the approach of sentence analysis, and enhances natural language understanding. Our work is meaningful.

REFERENCES

Hindley, J. R., Seldin, J. P., 1986. *Introduction to combinators and λ-calculus*, Cambridge University Press. Cambridge.

Jiang, Y., Pan, H. H., 1998. *An introduction to formal semantics*, Chinese Social Science Press. Beijing.

Retore, C., 2005. The logic of categorical grammars. In *Rapport de recherche, No. 5703*. INRIA.

Lecomte, A., 2004. Rebuilding MP on a logical ground. In *Research on language and computation*. Vol 2, pp. 27-55.

Morrill, G. V., 1994. *Type logical grammar-categorical logic of signs*, Kluwer Academic Publishers.

Qu, Y. W., 1998. *Fundamentals and formal descriptions of formal semantics*, Science Press. Beijing.

Ranta, A., 1994. *Type-theoretical grammar*, Oxford University Press. Oxford.

Simpson, A. K., 1992. Kripke semantics for a logical framework. In *Paper presented at Workshop on Types for Proofs and Programs*. pp. 313-340, Baastad, Sweden.

Wang, H. P., 1997. *Mathematical Logic*, Peking University Press. Beijing.

Coquand, C., 2002. A formalized proof of soundness and completeness of a simply typed Lambda-calculus with explicit substitutions. In *High-order and Symbolic Computation*. Vol 15, pp. 57-90.