Acquisition of Concentrated Modification Structure from Logical Formula

Hiroshi Sakaki
Meisei University
2-590 Nagabuchi Ohme-shi Tokyo 198-8655 Japan
sakaki@ei.meisei-u.ac.jp

Summary

This paper aims to construct concentrated modification structure from logical formula. First the matrix expression is extracted from the logical formula. Then, the matrix is transformed into geometrical structure called dependency structure equivalent to the former. Finally modification structure in which the arbitrarily chosen central node is modified by the entire nodes in the dependency structure, is constructed.

1 Introduction

Dependency structure was first proposed by Tesnière(Tesnière 1959; Kodama 1987). Mel'čuk proposed the dependency structure having arcs with their names(Mel'čuk 1988). Kaplan and Bresnan proposed the concept of Lexical Functional Grammar, which is notational variance with dependency structure with named arcs. As dependency structure is more suitable for human recognition, dependency structure is used in this paper(Kaplan 1982, Bresnan 2001). The term representing these two will be typed feature structure. There is a good manipulation tool for typed feature structure(Makino 1997).

This paper aims to construct the "concentrated modification structure" from a meaning expression through the structural transformation of the input typed feature structure. The typed feature structure is produced from logical formula by conventional method. Minute treatment of input typed structure as it is, is not considered here. The concept of "concentrated modification structure" is first proposed in this paper although similar consideration has been made before by the author of this paper(Sakaki 2002).

2 Transformation from atomic formula to atomic dependency structure

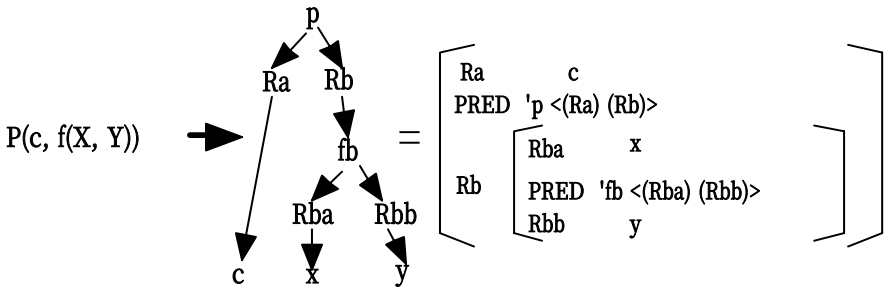

Figure 1: Transformation from atomic formula to atomic dependency structure

Here, transformation from atomic formula to the "atomic dependency structure" is treated.

Fig.1 is prototypical expression of the transformation. The portion existing to the left of the thick arrow is an atomic formula and the portion to the right illustrates the atomic dependency structure generated from the former. The rightmost portion is corresponding LFG expression.

Atomic dependency structure has two kinds of nodes, one of which is the concept node and the other of which is the role node. A concept node has the name beginning with a lower case letter and a role node has the name beginning with an upper case letter. A modification relation is defined only between concept node pairs. A concept node at the tip of an arc is defined to modify the node at the root of an arc.

The transformation rule designates that the name of a predicate at an atomic formula is transformed into the name of the topmost node at the atomic dependency structure after the upper case to lower case transformation at the first letter. Also the transformation rule generates role nodes not included in the atomic formula. The atomic formula in Fig.1 contains the function "fb". The method treated in this paper can handle functions.

3 Generation algorithm of concentrated modification structure

The “concentrated modification structure” is the construction where entire information of the input dependency structure is concentrated to the topmost node of an atomic dependency structure. This enables acquisition of entire information from the node. The algorithm of obtaining concentrated modification structure from logical formula consists of stage (1) “Generation of matrix of conjunctive canonical form”, stage (2) “Generation of matrix dependency structure” and stage (3) “Generation of concentrated modification structure”. They are respectively illustrated in sections with corresponding titles.

3.1 Generation of matrix of conjunctive canonical form

This stage or the stage (1) is conducted by conventional procedure shown in the appendix. This paper illustrates the operation of each stages for the logical formula input given as Eq.(A-2) generated from the English sentence in Eq.(A-1). The output of this stage is the matrix shown in Eq.(A-4). There is nearly automatic way to generate logical formula from English language expression(Lepore 2003).

3.2 Generation of matrix dependency structure

This stage or the stage (2) is divided into sub-stage (2-1) “Generation of atomic dependency structure” and sub-stage (2-2) “Connection by conjunction and disjunction constructions”. They are respectively illustrated in the subsections with corresponding titles.

3.2.1 Generation of atomic dependency structure

The sub-stage (2-1) generates atomic dependency structures using the correspondence

illustrated in Chapter 2. The atomic dependency structures shown in Fig.2 (a) though (c) are generated at sub-stage (2-1) from the atomic formulas existing in the matrix of Eq.(A-4).

The atomic formulas Plan(X), ～Suggest(mary, X), Understand(john, X) are respectively transformed into the atomic dependency structures of Fig.2(a), (b) and (c). The role nodes not existing in the atomic formula are introduced in the atomic dependency structure. The role node "Agt" shows that the concept node under it acts as agent or subject whereas the role node "Obj" shows that the node below acts as object or recipient of an action. The concept node "isa" shows

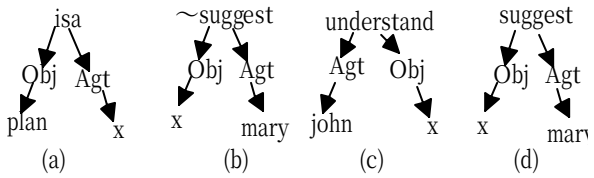

Figure 2: Atomic dependency structures

the node under the node "Agt" has the nature of the node under the node "Obj". The symbol "～" represents negation of the concept. When the structure in Fig.2(d) generates the value True, the structure in Fig.2(b) generate the value False and vice versa.

3.2.2 Connection by conjunction and disjunction constructions

At this sub-stage, the atomic dependency structures generated from atomic formulas of the same clause are first connected by disjunction constructions. A portion thus generated, together with an independent single predicate not influence by disjunction operation, is defined as a clause structure. The partial trees under the node "or" and “isa” in Fig.3 are clause structures.

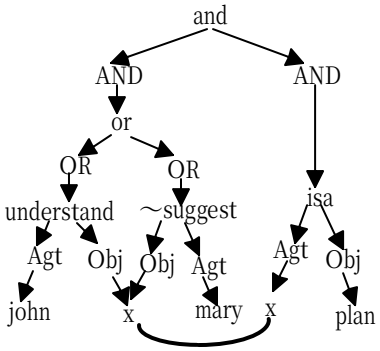

Figure 3: Matrix dependency structure

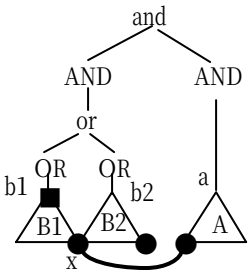

Figure 4: Abbreviated expression of matrix dependency structure

Successively, every clause structures are connected by conjunction constructions. A conjunction construction is built by the concept node "and" and the role nodes "AND" whereas a disjunction expression is built by the concept node "or" and the role nodes "OR". Fig.3 is the matrix dependency structure generated by the stage (2) from the matrix equation in Eq.(A-4).

For clearer explanation of the stage (3), the structure in Fig.3 is abbreviated into the structure of Fig.4. As the correspondence is obvious, no explanation is made.

3.3 Generation of concentrated modification structure

At the illustration of the stage, designation of "multi-tree node" is necessary. A node in the matrix dependency structure that belongs to more than one atomic dependency structures and that is not modified by other nodes is called multi-tree node. The node x in the matrix dependency structure in Fig.3 is a multi-tree node. The “central node” is defined as the node to which entire modification information in a matrix dependency structure is concentrated.

The input to this stage or stage (3) is the matrix dependency structure composed at previous stage that is exemplified by the figure in Fig.3. As duplication of input matrix dependency structure takes place in this stage, a duplicated result is called "unit structure".

Stage (3) is composed of the application of the operation "generation of concentrated modification structure" shown below. This operation is composed of 3 sub-operations respectively given as terms (3-1) through (3-3). Sub-operation (3-2) is the main sub-operation and recurrently used. Sub-operation (3-1) deals with initiation operation and sub-operation (3-3) deals with termination of the operation. The concepts of "active structure", "active multi-tree node" and "descendant node" are defined at the description of operations.

[Generation of concentrated modification structure]

(3-1) A unit structure is generated and the central node is designated in the unit structure. The atomic dependency structure (not clause structure) having the central node as its topmost node is designated as the active structure. The multi-tree nodes belonging to the active structure are selected as the active multi-tree node. The portion other than the active structure is eliminated from the unit structure. Under entire active multi-tree nodes, unit structures are connected respectively via role nodes "Lambda".

(3-2) Each unit structure generated at the former sub-operation (3-1) or sub-operation (3-2) has the “descendant node” defined by following expression. The node identical with the node in upper unit structure, to which the unit structure is connected, is defined as descendant node.

At each unit structure generated at the former sub-operation (3-1) or sub-operation (3-2), the clause structures including the descendant node are designated as active structures. A multi-tree node that belongs to an active structure and that has not the same node on the path to the central node, is selected as an “active multi-tree node”.

The portion other than the active structure is eliminated from the unit structure. Under entire active multi-tree nodes, unit structures are connected respectively via role nodes "Lambda". This sub-operation is recurrently applied as long as active multi-tree nodes exist.

(3-3) Operation (3) is terminated when no active multi-tree node exists. The resultant structure existing at the termination of the operation is the “concentrated modification structure” generated from the input matrix dependency structure.(end of the operation)

Here, the generation of concentration modification structure operation is applied to the matrix dependency structure in Fig.3. The node "understand" is designated as the central node. This

node corresponds to the node "b1" illustrated by a rectangle in abbreviated expression of Fig.4. Hereafter, the matrix dependency structure in Fig.4 is treated.

The generation process is shown in Fig.5(a). First, the description of the sub-operation (3-1) is given. At first, the “upper unit structure” is created. The node "b1" is designated as the central node. The atomic dependency structure "B1" is designated as the active structure. In Fig.5(a) the active structure is shown with thick lines. The multi-tree node "x" is selected as active multi-tree node. Then portions of upper unit structure drawn by thin lines are eliminated. Successively, the “lower unit structure” is connected under node "x" via the role node "Lambda".

Following is the description of the sub-operation (3-2). The descendant node in the lower unit structure is node "x". The entire atomic dependency structures are designated as active structures because all of them have the descendant node "x". As the node "x" has the same node on the path to the node "b1", this node is not selected as active multi-tree node.

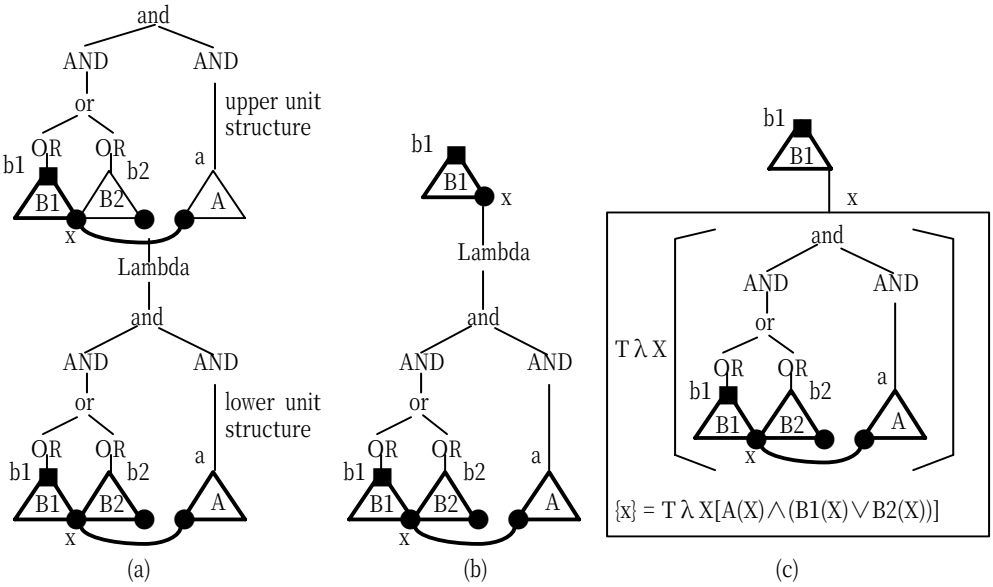

Figure 5: Generation of concentrated modification structure

Following is the description of the sub-operation (3-3). Since there is no active multi-tree node, the operation of the stage (3) is terminated. The structure existing at the end of the operation is shown in Fig.5(b). This is the concentrated modification structure generated.

3.4 Characteristics of the concentrated modification structure

The unit structure connected under a multi-tree node generates the set of values to be accommodated in the multi-tree node. This set of values is defined to be the set at which the unit structure under treatment takes the value of T(true).

At the case of concentrated modification structure in Fig.5(a), the set of values provided to the multi-tree node "x" is the set {x} given in following equations.

$\{x\} = X \mid A(X) \wedge (B1(X) \vee B2(X)) = T$ (1-1)

$\{x\} = X \mid \lambda X[A(X) \wedge (B1(X) \vee B2(X))] (X) = T$ (1-2)

$\{x\} = T\, \lambda X[A(X) \wedge (B1(X) \vee B2(X))]$ (1-3)

$B1(\ T\, \lambda X[A(X) \wedge (B1(X) \vee B2(X))]\ )$ (1-4)

Here, “X” is a variable representing value stored in the node “x”. Eq.(1-1) is plain expression of the definition. Eq.(1-2) is the expression using the lambda expression. Eq.(1-3) is defined as the abbreviated expression of Eq.(1-2). The role node "Lambda" is introduced to reflect the expression in Eq.(1-2) and Eq.(1-3). Fig.5(c) illustrates the situation where the value accommodated by the multi-tree node “x” is Eq.(1-3). Substituting the variable “X” in the equation B1(X) with {x} given by Eq.(1-3) generates Eq.(1-4) which represents concentration of information to the atomic dependency structure “B1” realizing the aim of the algorithm.

4 More complicated example

Here, an example generated by augmentation of previous example is treated.

Tom report that John understands whichever plan Mary suggests. (2)

The matrix dependency structure for the sentence of Eq.(2) is expressed in Fig.6.

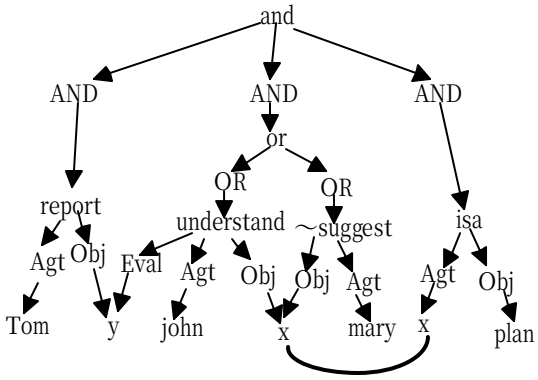

Figure 6: Matrix dependency structure

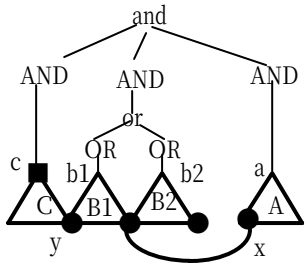

Figure 7: Abbreviated expression of matrix dependency structure

This is generated through successive application of sub-stage (1) shown in the appendix and sub-stage (2) expressed in section 3.2. The role node newly introduced is the node “Eval”. This node represents the evaluation given to the action represented by the topmost node of the atomic dependency structure to which the role node belongs. The structure in Fig.6 has the construction of Fig.3 but attached with an extra atomic dependency structure. This structure is abbreviated as shown in Fig.7. The correspondence between these two figures is obvious.

Application of stage (3) for the abbreviated expression in Fig.7 is descried. The node "c” corresponding the node “report” is designated as the central node.

Fig.8 shows the construction process of concentrated modification structure through the application of stage (3). First the top unit structure in Fig.8(a) is created and the atomic

dependency structure “C” depicted in thick line is selected as active structure. The multi-tree node "y" is selected as active multi-tree node. Then middle unit structure is connected under the node “y”. Among the configuration in the middle unit structure, the atomic dependency structures drawn in thick line are designated as active structure by sub-operation (3-2).

The active multi-tree node here is node “x”. The bottom unit structure is connected under the node “x”. Among the configuration in the middle unit structure, the atomic dependency structures drawn in thick line are active structure. As new multi-tree node is absent, the sub-operation (3-2) terminates. From the structure in Fig.8(a), the concentrated modification structure in Fig.8(b) is obtained.

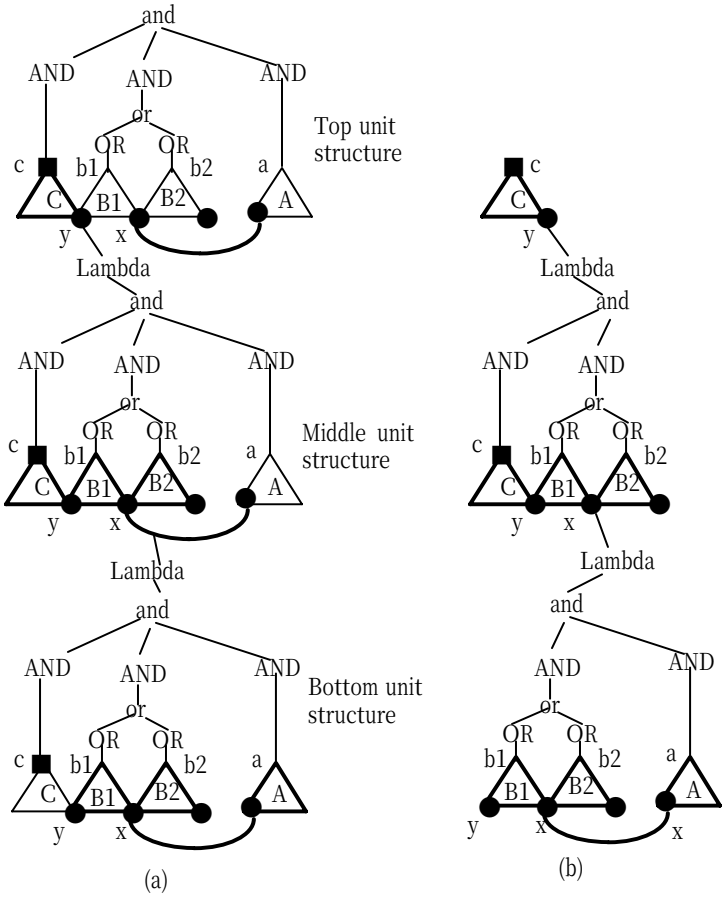

Figure 8: Generation of concentrated modification structure for a complex case

Here, the set of values provided to the multi-tree node "x" and “y” are respectively the sets {x} and {y} given in following equations Eq.(3-1) and Eq.(3-2). Here, the letters “X” and “Y” represent values stored respectively in nodes “x” and “y”. Substituting Eq.(3-1) into Eq.(3-2) generates Eq.(3-3) which has complete information of the value “y”. Substituting the value of Eq.(3-1) in the topmost atomic dependency structure gives rise to Eq.(3-4) which shows the concentration of information to the atomic dependency structure “C” at Fig.8(b).

$$\{x\} = T\,\lambda X[(B1(Y, X) \vee B2(X)) \wedge A(X)] \qquad (3\text{-}1)$$

$$\{y\} = T\lambda Y\,[C(Y)\wedge(B1(Y, X)\vee B2(X))] \quad (3\text{-}2)$$

$$\{y\} = T\lambda Y\,[C(Y)\wedge(B1(Y, T\lambda X[(B1(Y, X)\vee B2(X))\wedge A(X)]\,) \vee B2(\,T\lambda X[(B1(Y, X)\vee B2(X))\wedge A(X)]\,))] \quad (3\text{-}3)$$

$$\complement(T\lambda Y\,[C(Y)\wedge(B1(Y, T\lambda Y[(B1(Y, Y)\vee B2(X))\wedge A(X)]\,) \vee B2(\,T\lambda X[(B1(Y, X)\vee B2(X))\wedge A(x)]\,))]) \quad (3\text{-}4)$$

The equation Eq.(3-1) is the same as Eq.(1-3) except for the existence of variable “Y”. This shows the value of “X” is the handle of information of bottom portion of Fig.8(b) that is identical with lower portion of Fig.5(b).

5 Conclusion

This paper deals with the method of constructing concentrated modification structure from logical formula. The matrix of the logical formula is first generated. Then the matrix is transformed into equivalent dependency structure. Finally, the concentrated modification structure is obtained from the dependency structure.

Appendix Generation of matrix of conjunctive canonical form from logical formula

Conjunctive canonical form is generated through following sub-processes (1-1) through (1-6).

[Generation of conjunctive canonical form]

(1-1) Implication symbols are eliminated.
(1-2) Negation symbols are moved to the positions directly in front of atomic formulas.
(1-3) If necessary, bound variables are re-named.
(1-4) Existential quantifiers are eliminated by the introduction of Skolem function.
(1-5) Transformation to the matrix without quantifier symbols is accomplished.
(1-6) The matrix is transformed into conjunctive canonical form. (end of process)

The following shows the operation applied for the example used throughout this paper.

John understands whichever plan Mary suggests. (A-1)

$(\forall X)(Plan(X)\wedge(Suggest(mary, X)\rightarrow Understand(john, X))$ (A-2)

$=(\forall X)(Plan(X)\wedge(\sim Suggest(mary, X)\vee Understand(john, X)))$ (A-3)

$=Plan(x)\wedge(\sim Suggest(mary, X)\vee Understand(john, X))$ (A-4)

Eq.(A-1) is the original English sentence and Eq.(A-2) is corresponding logical formula composing the input of this generation operation. Eq.(A-3) is the result of sub-process (1-1) and also the prenex normal form obtained after sub-process (1-5). Eq.(A-4) is the matrix of conjunctive canonical form for the logical formula in Eq.(A-2). This forms the input for stage (2). “X” is a variable. "～" symbol expresses negation of the predicate attached with the symbol.

References

Bresnan, J: 2001m Lexical Functional Syntax, Blackwell

Fillmore, C.: 1968, The Case for Case in E.Bach and R. Harms (eds.), *Universals in Linguistic Theory*, Holt, Rinehart and Winston.

Kaplan, R. and Bresnan J.:1982, Lexical-Functional grammar: In Bresnan, J(ed) *The Mental Representation of Grammatical Relations,* MIT Press

Kodama, T.: 1987, *Research on Dependency Structures,* Kenkyusha Publishing, (in Japanese).

Lepore E. : 2003 *Meaning and Argument*, Blackwell

Makino T. et al,: 1997, *LiLFeS - Practical Programming Language For Typed Feature Structures***.** Proceedings of Natural Language Pacific Rim Symposium '97 (listed in http://www-tsujii.is.s.u-tokyo.ac.jp/lilfes/index-j.html)

Mel'čuk, I.: 1988, *Dependency Syntax,* State University of New York Press.

Sakaki, H. :2002, *Graphical Expression of Meanings Included in Various Kinds of Sentences*, Proc. Vol.XVII SCI2002 pp 146,153

Tesnière, L.: 1959, *Éléments de Syntaxe Structurale,* Kliensieck.