TOWARDS A MATHEMATICAL THEORY OF ANALOGY

Haraguchi, Makoto
Research Institute of Fundamental Information Science, Kyushu University

https://doi.org/10.5109/13366

出版情報：Bulletin of informatics and cybernetics. 21 (3/4), pp.29-56, 1985-03. Research Association of Statistical Sciences
バージョン：
権利関係：

Bulletin of Informatics and Cybernetics Vol. 21, No. 3~4, 1985

TOWARDS A MATHEMATICAL THEORY OF ANALOGY

By

Makoto HARAGUCHI*

Abstract

This paper presents a mathematical theory of analogy, which should be a basis in developing analogical reasoning by a computer. The analogy is a partial identity between two sets of facts. In order to compare several analogies, we introduce an ordering of analogies, and we define two types of optimal analogies, maximal analogies and greatest ones. We show a condition under which the greatest analogy exists, and also present a top-down procedure to find the maximal analogies.

1. Introduction

Analogical reasoning (AR, for short) is a kind of reasoning which derives conclusion from premises like other reasoning. The premise in AR is a statement that given two situations are similar in some respect, while the conclusion is one that the situations are similar in the other respects as well.

The AR or analogical problem solving is so important in the studies of artificial intelligence that many authors have investigated from various viewpoints [2, 4, 6, 14, 15]. In fact, Chapman [4] dealt with problem testing assistant by which some test programs are automatically coded from "layer correspondences" which are mappings between data types and programs. Tanwangson and Fu [14] also dealt with a robot planning problem by using a mapping between commands to the robot. The mappings in their studies are analogies, and their problem solving methods are based on AR. Winston [15] developed a system to answer a question by detecting a similarity between two English texts in question. Carbonel [2] proposed a general problem solver which can map the past solution for some problem into a solution for the analogous new problem.

Thus AR covers various problem domains. However no framework to unify the various AR techniques has been discussed. Furthermore neither methods to compare various analogies nor powers of AR have been discussed. It seems to the author that such unified and precise discussions would not be possible in the frameworks so far proposed. This should come from lack of mathematical theory of AR, unlike other reasoning such as deductive reasoning and inductive one. The deductive reasoning, as is well-known, has a firm basis of predicate logic, and has been implemented by using

* Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

theoretical results on the logic. On the other hand, the inductive reasoning has a mathematical basis of recursion theory, by which the power of inductive reasoning can be precisely discussed [1]. Recently Shapiro [12] has succeeded to give a unified theory of inductive reasoning. His theory is based on first order logic, and hence it has various applications such as logic program synthesis, grammatical inference and automatic debugging [13]. The observations above naturally motivate us to propose a mathematical theory of analogy.

1.1. The problems

The first problem to be answered in developing our mathematical theory of AR should be in what language we describe our theory and related concepts. The final goal of our studies on AR should be to design a knowledge information system which uniformally performs deductive reasoning and inductive reasoning as well as AR. Hence it is to be desired that one language can describe them. As we have observed in the previous paragraph, the first order language has been working as a description language common to deductive reasoning and inductive reasoning. It should be natural to expect the language to work successfully for our AR. Thus we choose the first order language as the description language of AR.

Our second problem is how to describe analogies in terms of the first order language. Generally an analogy is a partial agreement or likeness between two objects that are unlike in other ways. Therefore two analogous objects have two portions of these objects, one from each, which can be viewed identical in some way. In this sense, the two objects are partially indentical. Thus, we consider an analogy as a partial identity. Description of the partial identity depends on representations of the two objects in question. In this paper, we assume that an object is a system which consists of parts and relations between the parts. As is well-known, the first order language is suitable to represent the relations by facts, which are ground atoms in this paper. Then we can represent the system itself by a set of facts. Moreover, we can represent the identical portions by a formula, if we replace terms representing the system parts by some variables.

The third problem is to find an analogy, given two sets of facts. We call this problem an analogy detection. It is important to clarify the analogy detection. In fact, let us consider the following problem solving activities using AR:

(A1) we utilize the past experiences to solve the current problem by detecting an analogy between the past problem and the current one [2, 4, 6, 14, 15].

(A2) we learn a general law by detecting an analogy and then by identifying the analogous parts of the two situations [15].

Thus the key problem in AR is the analogy detection. There may exist several analogies for a pair of objects, because of the following facts:

(F1) Information about the objects depends on the amount of knowledges about the objects and also depends on the power of deductive reasoning.

(F2) Analogy often depends on viewpoints.

(F3) The partial identity is not always unique.

Any system which deals with the analogy detection should have three sub-systems. First one deduces enough information about objects to find useful analogy. The second

one extracts a necessary portion of these objects from some viewpoints. The third one finds analogies by using the information obtained in the two sub-systems above. We present, in this paper, an analogy detection method which is used in the third sub-system. Since several partial identities may exist for a given pair of objects, we should have an ordering of partial identities to compare them. We define such an ordering, and we also present a top-down method to find optimal partial identities with respect to the ordering. Although we can develop a bottom-up detection, we do not adopt it, since it may find many non-optimal partial identities before it finds the desired optimal ones.

The fourth problem is to derive conclusions in AR. For this purpose, it is necessary to set up relationships between analogous problems and the corresponding solutions. For instance, Plaisted [8] has mathematically clarified, in his theorem proving method, some relationships between sets of clauses and the corresponding resolution proofs. Although we do not discuss this problem in detail in this paper, it is possible to build a mathematical theory of AR by combining such a method as in the above and the theory we are presenting [5].

1.2. The outline

Analogy treated in this paper is based on Polya's clarified analogy [11]: Two systems are analogous if they agree in clearly definable relations of their respective parts. In Section 2, we make some assumptions on two objects with which AR is concerned and consider some examples on the clarified analogies, according to which we formalize analogies. Our formal analogy consists of a set of atoms and a pair of substitution, and it defines a partial identity between two sets of facts. By taking the formal analogy as a logical formula, the set of atoms and the pair of substitutions turn out to be a theorem and a pair of proofs, respectively. Finally we show that our formal analogy can define Winston's analogy [15].

In Section 3, to compare several analogies, we introduce an ordering of analogies which is a quasi-ordering in the sense of Plotkin [9]. The ordering relates to an improvement of analogies. We also present three improvement operators: *Refine*, *Merge* and *And*. These operators produce a better analogy from given analogies. Moreover, we show that two analogies are ordered if and only if one of them is constructed by applying the three operators to the other.

In Section 4, we discuss what is called canonical analogies. In general, our formal analogy may have some redundancy in defining the partial identity. Such a redundancy can be removed by unifications. We call the analogy thus obtained a canonical analogy that is better than the original one. In consequence, we have a finite search space of canonical analogies, which is called a canonical analogy structure. Here we should notice that our canonical analogy can be treated in nearly the same way as generalization of literals by Plotkin [9, 10]. This fact is mathematically simple, but it is important when we consider the analogy detection.

In Section 5, we define maximal and greatest analogies by using the ordering. We characterize the maximal analogy and greatest one in terms of the *Refine* and *And* operators. We also present a top-down method which finds the maximal analogies by using the canonical analogies.

2. Analogy as a Partial Identity

In this section, we define a formal analogy on clarified analogy, and we also show that Winston's analogy can be described in our formal analogy.

2.1. Clarified analogy

> "Looking in a natural history museum at the skeletons of various mammals, you may find them all frightening. If this is all the similarity you can find between them, you do not see much analogy. Yet you may perceive a wonderfully suggestive analogy if you consider the hand of a man, the paw of a cat, the foreleg of a horse, the fin of a whale, and the wing of a bat, these organs so differently used, as composed of similar parts similarly related to each other. This example illustrates the most typical case of clarified analogy; two systems are analogous, if they agree in clearly definable relations of their respective parts."
>
> G. Polya [11]: Induction and Analogy in Mathematics

Thus, to detect analogy between two systems, first we should find out some relations between the parts of each system. Such relations naturally depend on our viewpoints, interest, preconceptions and so forth. However in this paper we do not deal with the problem of how to find the relations, and instead we start with making an assumption as in system theory.

(A1) Objects of AR are systems, each of which consists of a set of parts and a set of relation instances which occur in the system. The relations are already defined so that they can describe relationships between the parts.

Then the next problem is how to decide whether the two systems agree in relations or not. For this purpose, we make another assumption.

(A2) Two systems agree in relations of their respective parts if there exists a *pairing*, which corresponds parts in one system to parts in the other system, and at least one pair of relation instances, one from each system, such that they are the same instance of a relation except for the pairing.

We call the above pair of relation instances an *agreement of relations* under a pairing of parts. Moreover, since there may exist several agreements under a pairing, we mean an *analogy* by a set of agreements under the pairing.

Under the assumptions A1 and A2, let us consider the following example in which parts of systems and relations are represented by individual constants and predicate symbols, respectively. Then each relation instance is represented by a fact, i. e., ground atom.

EXAMPLE 2.1. A family is a system of persons and relations such as "parent", "father", etc. Consider the following families:

$$F_1=(\{makoto,\ fumiko,\ soichiro\};\ \{parent(makoto,\ fumiko),\ mother(makoto,\ fumiko),\ father(fumiko,\ soichro),\ ground\text{-}father(makoto,\ soichiro)\}).$$

$$F_2=(\{james,\ frank,\ edward\};\ \{parent(james,\ frank),\ father(james,\ frank),\ father(frank,\ edward),\ ground\text{-}father(james,\ edward)\}).$$

Under the pairing $\{\langle makoto,\ james\rangle, \langle fumiko,\ frank\rangle\}$, the relations $parent\,(makoto,\ fumiko)$ and $parent\,(james,\ frank)$ can be viewed identical under the pairing. In other words, the above two facts are identified and paired by the pairing of constant symbols. Such a pair of identical facts is called an identity of facts (under a pairing of terms). Then the above identity of facts can be described in the following way:

There are some persons X and Y such that $parent\ (X,\ Y)$;

X is *makoto* in F_1 and *james* in F_2, and
Y is *fumiko* in F_1 and *frank* in F_2.

This statement is also described by an atom $parent\,(X,\ Y)$ and a pair $u=(u_1,\ u_2)$ of substitutions:

$$u_1=\{makoto/X,\ fumiko/Y\},$$
$$u_2=\{james/X,\ frank/Y\}.$$

Observe that the pairing of constants is described by the substitutions and variables. For instance, the pair $\langle Xu_1,\ Xu_2\rangle$ is $\langle makoto,\ james\rangle$. Thus $P_1=(\{parent\,(X,\ Y)\},\ u)$ defines the pairing and the identity of facts under the pairing. There are two more agreements of relations under the pairing $\{\langle makoto,\ james\rangle, \langle fumiko,\ frank\rangle, \langle soichiro,\ edward\rangle\}$. One is

$$\langle father\,(fumiko,\ soichiro),\ father\,(frank,\ edward)\rangle$$

and the other is

$$\langle ground\text{-}father\,(makoto,\ soichiro),\ ground\text{-}father\,(james,\ edward)\rangle.$$

These two agreements are certainly identities of facts under the pairing. Similarly to P_1, the former identity is defined by

$$P_2=(\{father\,(Y,\ Z)\},\ v),$$

and the latter identity is defined by

$$P_3=(\{ground\text{-}father\,(X,\ Z)\},\ v),$$

where
$$v=(v_1,\ v_2),$$
$$v_1=\{makoto/X,\ fumiko/Y,\ soichiro/Z\},$$
$$v_2=\{james/X,\ frank/Y,\ edward/Z\}.$$

By combining P_1, P_2 and P_3, we have a set of agreements

$$P=(\{parent(X,\ Y),\ father\,(Y,\ Z),\ ground\text{-}father\,(X,\ Z)\},\ v).$$

Note that, since each agreement is an identity of facts, the pair P defines a partial identity between two sets of facts under the pairing of constant symbols.

EXAMPLE 2.2. In the example above, constants as system parts are in correspondence with each other. Moreover we can naturally correspond constants to terms which describe some constructed parts. Consider two systems represented by the following Fig. 2.1,

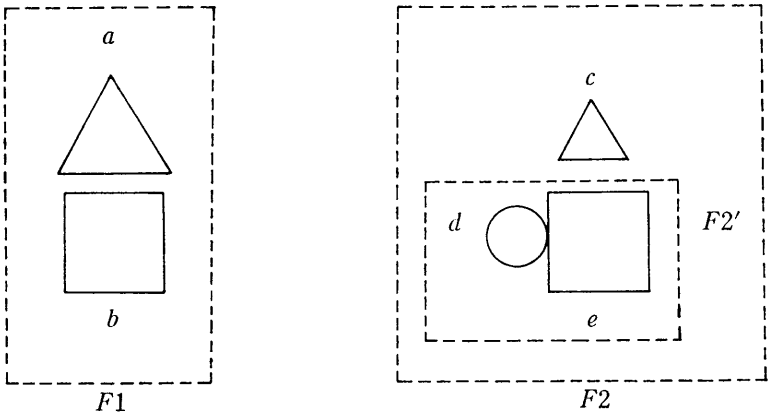

Fig. 2.1 Two systems.

where $F1$ is a system with

$$S1=\{above\,(a,\ b),\ triangle\,(a),\ square\,(b)\},$$

and $F2$ is a system with

$$S2=\{above\,(c,\ on\,(d,\ e)),\ square\,(e),\ circle\,(d),\ triangle\,(c)\}.$$

The function symbol "on" represents a figure constructor and that the predicate symbol "above" represents a relation. Between $F1$ and $F2$, we observe two agreements of relations:

There is a triangle X, and the triangle X is above Y;

X is a in $F1$ and c in $F2$, and
Y is b in $F1$ and $on\,(d,\ e)$ in $F2$.

In the last agreement, we correspond a constant symbol b to a term $on\,(d,\ e)$ which is a constructed part $F2'$. We allow such a pairing of terms. Similarly to Example 2.1, these agreements are identities of facts under the pairing $\{\langle a,\ c\rangle,\ \langle b,\ on(d,\ e)\rangle\}$. Moreover, the sets of these identities of facts is defined by $P=(W,\ u)$, where

$$W=\{triangle\,(X),\ above\,(X,\ Z)\},$$
$$u=\langle\{a/X,\ b/Z\},\ \{c/X,\ on(d,\ e)/Z\}\rangle.$$

These observations can be summed up as follows:

A set of agreements of relations under a paring of parts is a partial identity between two sets of facts. The pairing of parts is defined by a pair u of substitutions. The partial identity under the pairing is defined by the pair u and a set W of atoms. Thus $(W,\ u)$ defines our analogy as a set of agreements of relations.

2.2. Formal definition of analogy

Let S_1 and S_2 be finite sets of ground atoms with no common individual constants. Let u be a pair $(u_1,\ u_2)$ of substitutions. For a variable or an atom A, the pair $\langle Au_1,\ Au_2\rangle$ is denoted by Au. For a set W of variables or atoms, the set $\{Au:\ A\in W\}$ is denoted by Wu. $Var\,(W)$ denotes the set of all variables in W. Moreover, when there is no confusion, the association A with the pair Au is denoted by $A-Au$.

DEFINITION 2.1. Let S_1 and S_2 be sets as in the above.

(1) We say that a pair u of substitutions satisfies the *partial identity condition* (PIC, for short) for a set W of atoms if $Wu \subseteq S_1 \times S_2$, and both Wu and $Var(W)u$ are one-to-one.

(2) We call W an *analogy theorem* under S_1 and S_2 if there exists a pair of substitutions u which satisfies PIC for W. In this case we call (W, u) an *analogy*, Wu the *partial identity between two sets of facts* (PIF, for short) of (W, u), and $Var(W)u$ the *pairing of terms* (PT, for short) of (W, u). Moreover, we say that a variable x in W defines the pair xu of terms, and that an atom A in W defines the identity Au of facts. (The term "partial identity" is due to Klix [7].)

W and u_i in an analogy $(W=\{A_1, \cdots, A_n\}, u=(u_1, u_2))$ are taken as a formula ${}^{\exists}x_1 \cdots {}^{\exists}x_k[A_1 \wedge \cdots \wedge A_n]$ and an answer substitution of refutation of $S_i \rightarrow W$, respectively. In fact, we have

PROPOSITION 2.1. *For a set $S=\{B_1, \cdots, B_q\}$ of ground atoms, the following conditions are equivalent.*

(1) $B_1 \wedge \cdots \wedge B_q \rightarrow {}^{\exists}x_1 \cdots {}^{\exists}x_k[A_1 \wedge \cdots \wedge A_n]$ *is valid.*

(2) $\bar{A}_1 \vee \cdots \vee \bar{A}_n$ *u-subsumes* $\bar{B}_1 \vee \cdots \vee \bar{B}_q$ *for some substitution* u.

(3) $\{A_1, \cdots, A_n\}u \subseteq S$.

PROOF. Assume that $B_1 \wedge \cdots \wedge B_q \rightarrow {}^{\exists}x_1 \cdots {}^{\exists}x_k[A_1 \wedge \cdots \wedge A_n]$ is valid. Then $S'=\{B_1, \cdots, B_q, \bar{A}_1 \vee \cdots \vee \bar{A}_n\}$ is an unsatisfiable set of clauses. Since each B_j is positive, and since $\bar{A}_1 \vee \cdots \vee \bar{A}_n$ consists of negative literals, there is a linear refutation of S' with $\bar{A}_1 \vee \cdots \vee \bar{A}_n$ as top and $\{B_j\}$ as side clauses. According to "subsumption algorithm" (for instance, refer to [3]), $\bar{A}_1 \vee \cdots \vee \bar{A}_n$ u-subsumes $\bar{B}_1 \vee \cdots \vee \bar{B}_q$ for the answer substitution u of the linear refutation. The converse is trivial.

Thus an analogy (W, u) consists of a theorem under each S_i and a pair of proofs which satisfy PIC. For this reason, we call W an analogy theorem.

2.3. Extensible-relation representation

Now we show that Definition 2.1 is still available to define a set of agreements of relations about relations. Such relations are said to be extensible. In what follows, we simply call relation instances relations.

P. H. Winston [15] developed a system which reasons and learns by analogy. Inputs to his system are simple English like sentences which describe several facts about some situations such as Shakespeare's tragedies and scientific laws. Then the system translates them into kinds of networks called *extensible-relation representations* by using a frame structure. The extensible-relation representation consists of situation parts as nodes that are tied together with relations. In order to express a supplementary description for

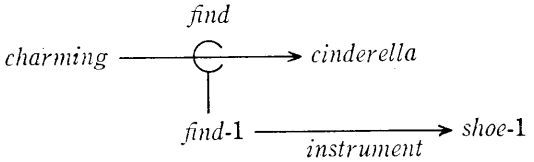

Fig. 2.2 Extensible-relation representation.

the relation itself, a new kind of node called reference node is created. Such a node is hanging from the relation to which the node refers. By using the reference nodes, we can treat the extensible-relations between relations and parts. For example, the system translates the input sentence "Charming finds Cinderella [instrument Shoe-1]" into the network structure illustrated by Fig. 2.2, where "cinderella", "charming", and "shoe-1" are situation parts, "find" and "instrument" are relation names. "find-1" is a reference node which refers the relation *find* (*charming, cinderella*). There is another relation *istrument* (*find*-1, *shoe*-1). Since "find-1" refers *find* (*charming, cinderella*), the last relation is represented by *instrument* (*find* (*charming, cinderella*), *shoe*-1) which is a true description in Cinderella Story (CS, for short). Thus the relation names are treated as function symbols when the situation as a system is represented in the first order language. In what follows, we use the predicate symbol "erel" to denote the true extensible relations in a given situation. For example, consider Romeo and Juliet Story (RJ, for short) which has "romeo" and "juliet" as situation parts and "love", "kiss" and "cause" as relations. By using the erel, a portion of RJ is discribed as follows:

$$\begin{aligned} S_{RJ} = \{ & erel\ (love\ (romeo,\ juliet)),\ erel\ (kiss\ (romeo,\ juliet)), \\ & erel\ (cause\ (love\ (romeo,\ juliet),\ kiss\ (romeo,\ juliet))), \\ & boy\ (romeo),\ girls\ (juliet),\ strong\ (romeo),\ beautiful\ (juliet)\}, \end{aligned}$$

where "boy", "girl", "strong" and "beautiful" are predicate symbols to denote the properties of situation parts. The extensible-relation representation which corresponds to S_{RJ} has two reference nodes. One of them refers *love* (*romeo, juliet*) and the others refers *kiss* (*romeo, juliet*).

Given two extensible-relation representations of situations, Winston's analogy is a pairing of parts with the agreements of relations which he calls "evidences" of the pairing. For example, consider an analogy between CS and RJ. A portion of CS is described by:

$$\begin{aligned} S_{CS} = \{ & prince\ (charming),\ strong\ (charming),\ beautiful\ (cinderella), \\ & erel\ (love\ (charming,\ cinderella)),\ erel\ (kiss\ (charming,\ cinderella)), \\ & erel\ (cause\ (love\ (charming,\ cinderella),\ kiss\ (charming,\ cinderella))), \\ & erel\ (instrument\ (find\ (charming,\ cinderella),\ shoe\text{-}1))\}. \end{aligned}$$

Similarly to S_{RJ}, the extensible-relation representation which corresponds to S_{CS} have three reference nodes to define cause and instrument relations. There are five evidences of the pairing

$$\{\langle romeo,\ charming\rangle,\ \langle juliet,\ cinderella\rangle\}.$$

For instance, S_{RJ} and S_{CS} agree in love and kiss relations. This establishes two evidences, which correspond two identities of facts:

$$erel\ (love\ (X,\ Y)),\ \text{and}\ erel\ (kiss\ (X,\ Y)),\ \text{where}$$
$$X\text{-}\langle romeo,\ charming\rangle\ \text{and}\ Y\text{-}\langle juliet,\ cinderella\rangle.$$

Then the reference nodes which refer the love relation are paired. Similarly, the reference nodes which refer the kiss relation are paired. These new pairs of reference

nodes are added to the pairing of parts. Then, under this new paring, S_{RJ} and S_{CS} agree in cause relation. This establishes an evidence of the original pairing. Note that, since the paired reference nodes refer the identical relations under the original paring, this evidence corresponds to the identity of fact:

$$erel\,(cause\,(love\,(X, Y),\ kiss\,(X, Y)),\ \text{where}$$
$$X\text{-}\langle romeo,\ charming\rangle \text{ and } Y\text{-}\langle juliet,\ cinderella\rangle.$$

Consequently, we have the set of five evidences. The set is defined by $P=(W, u)$, where

$$W=\{erel\,(love\,(X, Y)),\ erel\,(kiss\,(X, Y)),\ strong\,(X),\ beautiful\,(Y),$$
$$erel\,(cause\,(love\,(X, Y),\ kiss\,(X, Y)))\},$$
$$u=\langle\{romeo/X,\ juliet/Y\},\ \{charming/X,\ cinderella/Y\}\rangle.$$

Although we have not strictly described Winston's method to compute the evidences of a given pairing of parts, we have shown that a set of evidences under the paring of parts is described by a set of identities of facts. Hence Winston's analogy turns out to be a partial identity between two sets of facts, and hence it is defined by our formal analogy.

3. An Ordering of Analogies

In general, there may exist many analogies for a given pair of sets of facts. Hence we need to determine whether one analogy is superior to another or not. For this purpose, we also need an ordering of analogies. The orderings of analogies so far considered are numerical ones [14, 15], so that it is impossible to find structural relationships between analogies. Thus the ordering should be a structual one.

3.1. Definition of ordering of analogies

We need the following definition to consider the structural ordering of analogies.

DEFINITION 3.1. (Plotkin [9, 10]) For given clauses C_1 and C_2, if a clause C u_i-subsumes C_i for some substitution u_i $(i=1, 2)$, then C is called a *generalization* of C_1 and C_2. We call the pair (C, u) a *generalization diagram* of C_1 and C_2, where u is the pair (u_1, u_2).

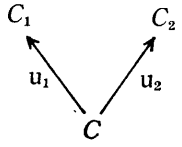

Fig. 3.1 Generalization diagram.

The following proposition is obvious.

PROPOSITION 3.1. *If* $(W=\{A_1, \cdots, A_n\}, u)$ *is an analogy under* $S_i=\{B_{i,1}, \cdots, B_{i,m(i)}\}$ $(i=1, 2)$, *then* $(A_1\vee\cdots\vee A_n, u)$ *is a generalization diagram of* $B_{i,1}\vee\cdots\vee B_{i,m(i)}$ $(i=1, 2)$.

From this proposition, we represent an analogy (W, u) by the corresponding gene-

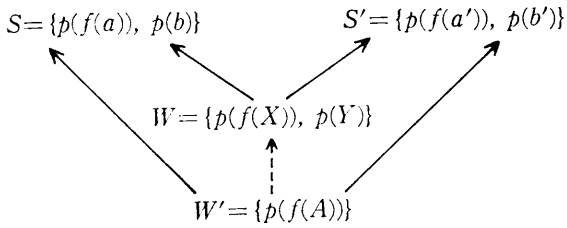

Fig. 3.2 Two generalization diagrams as analogies.

ralization diagram. Now consider the following example to define our ordering of analogies.

EXAMPLE 3.1. Fig. 3.2 are two generalization diagrams for two analogies (W, u) and (W', u') with $u=\{X-\langle a, a'\rangle, Y-\langle b, b'\rangle\}$ and $u'=\{A-\langle a, a'\rangle\}$, where the dotted arrow is a substitution $v=\{X/A\}$. The pair $\langle a, a'\rangle$ defined by the variable A in W' is also defined by the variable $X=Av$ in W. This implies that the above diagram is commutative and that PT of (W', u') is defined by PT of (W, u) and the substitution v. We require this commutativity to define our ordering. Moreover, since W' v-subsumes W, PIF of (W, u) includes PIF of (W', u'). Thus, (W, u) is better than (W', u') with respect to both PIF and PT.

DEFINITION 3.2. For given S_i $(i=1, 2)$, we say that an analogy (A, u) is superior to an analogy (B, v), if there exists a substitution s such that $Bs\subseteq A$ and that $Xv=Xsu$ for any variable X in B. In this case we denote $(A, u)\geqq(B, v)$, and say that $(A, u)\geqq(B, v)$ holds by the substitution s. We also define an equivalence $(A, u)-(B, v)$ if $(A, u)\geqq(B, v)$ and $(A, u)\leqq(B, v)$.

From the definition, we have

PROPOSITION 3.2.

(1) *If* $(A, u)\geqq(B, v)$ *then* $Au\supseteq Bv$.

(2) *The ordering* $\geqq$ *of analogies is a quasi-ordering, that is, a reflexive and transitive binary relation.*

In what follows, we identify (W, u) and (W', v) if $(W, u|_{Var(W)})$ and $(W', v|_{Var(W')})$ are the same except for a renaming of variables, where $u|_{Var(W)}=\{t/x\in u : x\in Var(W)\}$. In this case we denote $(W, u)=(W', v)$.

3.2. Analogy operators and characterizations of ordering

We have introduced the quasi-ordering $\leqq$ to compare analogies. Therefore, if we improve an analogy, then the improved analogy should be superior to the original one. we carry out such improvements by using the operators *Merge*, *Refine* and *And*. They take analogies under preconditions, and return a superior analogy if they succeed. Moreover we characterize the quasi-ordeing $\leqq$ in terms of these operators.

(O1) ***Merge* operator**: For an input analogy $P=(W, u)$,

precondition: there exists a non-empty set V of variables

$$x_i\ (1\leqq i\leqq m,\ 2\leqq m)\quad \text{with}\quad x_iu=x_ju\quad \text{for all}\quad i, j.$$

output: a pair $P'=(Ws, u)$, where $s=\{x_1/x_j: 2\leqq j\leqq m\}$, and the pair P' is denoted by *Merge* $[P; V]$.

From the definition, it is clear that P' is an analogy which is superior to P. *Merge* operator is said to be applicable to P if there exists a set V satisfying the precondition. Note tlhat *Merge* removes a redundancy in the uses of variabes which define the same pairing of terms.

For an analogy (W, u), assume that there exists a variable x which defines the pair $xu=\langle f(g(a), b), f(g(a'), b')\rangle$. Then it is natural to decompose the pair xu to pairs $\langle a, a'\rangle$ and $\langle b, b'\rangle$. For this purpose, new variables x_1 and x_2 are introduced, and the variable x is replaced by the term $f(g(x_1), x_2)$. Thus and so Refine simplifies the PT.

(O2) ***Refine* operator**: For an input analogy (W, u),

precondition: there exists a non-empty set V of variables
x_i in W $(1\leqq i\leqq m)$ with
$x_iu=\langle term_i(t_{i,1}, \cdots, t_{i,n(i)}), term_i(s_{i,1}, \cdots, s_{i,n(i)})\rangle$, where
$term_i(z_1, \cdots, z_{n(i)})$ is a proper term with variables $z_1, \cdots, z_{n(i)}$.

output pair: $P'=(Wv, w)$, where
$v=\{term_i(z_{i,1}, \cdots, z_{i,n(i)})/x_i: 1\leqq i\leqq m\}$,
$z_{i,j}$ $(1\leqq j\leqq n(i), 1\leqq i\leqq m)$ are new variables, and
$w=u|_{Var(W)\setminus V}\cup\{z_{i,j}-\langle t_{i,j}, s_{i,j}\rangle: 1\leqq j\leqq n(i), 1\leqq i\leqq m\}$.

From the definition, $u=vw$ on $Var(W)\setminus V$. For any variable x in V with $xu=\langle term(t_1, \cdots, t_n), term(s_1, \cdots, s_n)\rangle$, we have $xvw_1=term(t_1, \cdots, t_n)=xu_1$. Similarly, $xvw_2=xu_2$ holds. Hence $u=vw$ holds on $Var(W)$. Hence, if the output pair P' is an analogy, then $P'\geqq P$ by v and their PIFs are equal to each other. Since $Var(Wv)w$ may not be one-to-one, P' is not necessarily an analogy. Hence we define

$$Refine[P; V]=\begin{cases} P' & \text{if } P' \text{ is an analogy} \\ \perp \text{ (undefined)} & \text{otherwise.} \end{cases}$$

Moreover, when P' is an analogy, we say that we can refine P (with respect to V) and that Refine operator is applicable to P.

The third operator is *And*. *And* operator is the most important of the three. In a word, *And* operator tries to combine two analogies into one.

(O3) ***And* operator**:
For given two analogies $P=(W, u)$ and $P'=(W', u')$,

output: a pair $P''=(W\cup W', v)$, where
we rename the variables in W and W' so that $Var(W)\cap Var(W')=\phi$,
$v=\langle v_1, v_2\rangle$, and
$v_i=\{xu_i/x: x\in Var(W)\}\cup\{xu_i'/x: x\in Var(W')\}$.

And operator thus has no precondition. From the difinition, we have $(W\cup W')v=Wu\cup Wu'$ and $Var(W\cup W')v=Var(W)u\cup Var(W')u'$. Similarly to *Refine*, the output pair

P'' may not be an analogy. So we define

$$And[P;\ P'] = \begin{cases} P'' & \text{if } P'' \text{ is an analogy} \\ \perp & \text{otherwise.} \end{cases}$$

Moreover, we say that And operator is applicable to P and P' if P'' is an analogy.

EXAMPLE 3.2. Let us recall Example 2.1, where

$$P_1 = (\{parent\,(X,\ Y),\ father\,(Y,\ Z),\ ground\text{-}father\,(X,\ Z)\},\ \{X\text{-}\langle makoto,\ james\rangle,\ Y\text{-}\langle fumiko,\ frank\rangle,\ Z\text{-}\langle soichiro,\ edward\rangle\})$$

is an analogy. Also

$$P_2 = (\{father(K,\ L)\},\ K\text{-}\langle fumiko,\ james\rangle,\ L\text{-}\langle soichiro,\ frank\rangle\})$$

is an analogy. Since the set

$$\{\langle father\,(fumiko,\ soichiro),\ father\,(frank,\ edward)\rangle,\ \langle father\,(fumiko,\ soichiro),\ father\,(james,\ frank)\rangle\}$$

is not one-to-one, the output pair of *And* for P_1 and P_2 is not an analogy,

If $P = And[P_1;\ P_2] \neq \perp$ then $P_i \leqq P$ holds by the empty substitution ϕ or a renaming of variables. The converse does not hold in general.

EXAMPLE 3.3. Let

$$\begin{aligned} &S_1 = \{r_1(f(a)),\ r_2(f(a))\},\\ &S_2 = \{r_1(f(a')),\ r_2(b')\},\\ &P = (\{r_1(f(Y)),\ r_2(Z)\},\ \{Y - \langle a,\ a'\rangle,\ Z - \langle f(a),\ b'\rangle\}),\\ &P_1 = (\{r_1(X)\},\ \{X - \langle f(a),\ f(a')\rangle\}),\\ &P_2 = (\{r_2(Z)\},\ \{Z - \langle f(a'),\ b'\rangle\}). \end{aligned}$$

Then we have

$$P_1 \leqq P \text{ by } \{f(Y)/X\} \quad \text{and} \quad P_2 \leqq P \text{ by } \phi.$$

Since the set $\{\langle f(a),\ f(a')\rangle,\ \langle f(a),\ b'\rangle\}$ is not one-to-one, $And[P_1;\ P_2] = \perp$. However, by refining P_1, we have

$$And[Refine[P_1;\ \{X\}];\ P_2] \neq \perp.$$

Example 3.3 shows that *And* operator is applicable to two analogies by refining if necessary, when they have the same superior anology. In fact we have the following theorem which characterizes the superiority of one analogy over two analogies in terms of *Refine* and *And* operators.

THEOREM 3.1. *For analogies P_1, P_2 and P, $P_i \leqq P$ holds for $i = 1, 2$ if and only if there exists P_i', which is either P_i or $Refine[P_i;\ V_i] \neq \perp$ for some V_i, such that $P \geqq And[P_1';\ P_2'] \neq \perp$ holds.*

We need a proposition and a lemma to prove this theorem. In what follows, we call a substitution, which replaces variables by variables, v-substitution.

PROPOSITION 3.3. *Let $P_B = (B,\ s)$ and $P_i = (A_i,\ u_i)$ be analogies such that $P_B \geqq P_i$ by a v-substitution v_i. Then $P_B \geqq And[P_1;\ P_2] \neq \perp$ holds.*

PROOF. Since v_i replaces variables by variables, we have $Var(A_i)u_i \subseteq Var(B)s$. Also we have $A_iu_i \subseteq Bs$. Since both $Var(B)s$ and Bs are one-to-one, $Var(A_1)u_1 \cup Var(A_2)u_2$ and $A_1u_1 \cup A_2u_2$ are one-to-one. Hence $And[P_1; P_2] \neq \perp$. It is clear that $P_B \geqq And[P_1; P_2]$ holds by $v := v_1 \cup v_2$.

LEMMA 3.1. *Let $P_B=(B, s)$ and $P_A=(A, u)$ be analogies with $P_B \geqq P_A$ by w. If $V=\{x \in Var(A): xw$ is a proper term$\} \neq \phi$, then we have $Refine[P_A; V] \neq \perp$ and $P_B \geqq Refine[P_A; V]$ by a v-substitution.*

PROOF. Assume that V is not empty. For each x in V, let $xw=t(z_1, \cdots, z_n)$ for some term t and some variable z_j in B. The commutativity $u=ws$ implies

$$xu=\langle t(z_1s_1, \cdots, z_ns_1), t(z_1s_2, \cdots, z_ns_2)\rangle .$$

Hence, V satisfies the precondition of *Refine*. *Refine* operator produces the following output pair $P_C=(C, q)$:

$$C=Aw', \quad w'=\{t(x_1, \cdots, x_n)/x : x \in V\},$$
$$x_j \text{ is the new variable corresponding to } z_j \text{ (for each } x \in V),$$
$$q=u|_{Var(A)\setminus V} \cup \{x_j - \langle z_js_1, z_js_2\rangle\} .$$

For a variable y in $Var(A)\setminus V$, we have $yq=yu=yws \in Var(W)s$, since yw is a variable in B. For the new variable x_j for x, we have $x_jq=z_js \in Var(B)s$. Hence $Var(C)q \subseteq Var(B)s$. Since $w'q=u$ holds on $Var(A)$, $Cq=Aw'q=Au$ holds. Since $Var(B)s$ and Au are one-to-one, and since $w'q=u$ holds, (C, q) is an analogy which is superior to $P_A=(A, u)$ by w'. Let

$$w''=\{yw/y : y \in Var(A)\setminus V\}$$
$$\cup \{z_j/x_j : x_j \textit{ is the new variable for } x \textit{ in } V\} .$$

It is now clear that $P_C \leqq P_B$ by w''. Since yw is a variable whenever $y \in Var(A)\setminus V$, w'' is a v-substitution.

PROOF OF THEOREM 3.1. If-part is trivial.

Only-if part: Let $P_i=(A_i, u_i)$ and $P=(B, v)$ be analogies with $P_i \leqq P$ by w_i for i=1, 2, where $u_i=\langle u_{i1}, u_{i2}\rangle$. Since $A_1u_1 \cup A_2u_2 \subseteq Bv$, $And[P_1; P_2] \neq \perp$ iff $Var(A_1)u_1 \cup Var(A_2)u_2$ is one-to-one. Hence, we verify only the case that $Var(A_1)u_1 \cup Var(A_2)u_2$ is not one-to-one. Since each $Var(A_i)u_i$ is one-to-one. there exist variables x_i in $Var(A_i)$ $(i=1, 2)$ such that

$$\begin{aligned} &x_1u_{11}=x_2u_{21} \quad \text{and} \quad x_1u_{12} \neq x_2u_{22}, \quad \text{or} \\ &x_1u_{11} \neq x_2u_{21} \quad \text{and} \quad x_1u_{12}=x_2u_{22}. \end{aligned} \tag{3.1}$$

Let $V_i=\{x \in Var(A_i): xw_i$ *is a proper term*$\}$ $(i=1, 2)$. If both V_1 and V_2 are empty, then, for the variable x_i in (3.1), we have

$$\begin{aligned} &x_1w_1v_1=x_2w_2v_1 \quad \text{and} \quad x_1w_1v_2 \neq x_2w_2v_2, \quad \text{or} \\ &x_1w_1v_1 \neq x_2w_2v_1 \quad \text{and} \quad x_1w_1v_2=x_2w_2v_2. \end{aligned} \tag{3.2}$$

Since x_iw_i is a variable in P, (3.2) implies that P is not an analogy. Hence, at least one V_i is not empty. We verify only the case that $V_1 \neq \phi$ and $V_2=\phi$. Other cases are

similarly verifiable. From Lemma 3.1, we have $P_1'=(A_1', u_1')=Refine[P_1; V_1]\neq \perp$ and $P\geqq P_1'\geqq P_1$. Note that $P\geqq P_1'$ holds by some v-substitution w_1'. Moreover, since we assume that V_2 is empty, w_2 is also a v-substitution. Hence, according to Proposition 3.3, we completes the proof.

Theorem 3.1 characterizes the superiority of one analogy over two analogies, while the following theorem characterizes the superiority of an analogy over another one.

THEOREM 3.2. *For two analogies P_1 and P_2, $P_1\leqq P_2$ holds if and only if P_2 is obtained from P_1 by a finite number of applications of Refine, Merge and And operators.*

PROOF. Let $P_A=(A, u)\leqq P_B=(B, v)$ by w. P_B is constructed from P_A by the following steps.

Step 1: For the set $V=\{x\in Var(A): xv \text{ is a proper term}\}$, if V is empty then go to Step 2. Otherwise, according to Lemma 3.1, we have an analogy $P_C=(C, s)$ with

$$P_C=Refine[P_A; V]\neq \perp \quad \text{and} \quad P_A\leqq P_C\leqq P_B.$$

Especially, $P_C\leqq P_B$ holds by a v-substitution s'. Set $P_A:=P_C$ and $w:=s'$, respectively.

Step 2: We can merge two variables x and y in P_A with $xw=yw$, since $xu=xwv=ywv=yu$ holds. By the definition of *Merge*, clearly $Merge[P_A; \{x, y\}]\leqq P_B$ by w. Hence, by repeating the applications of *Merge* until such a pair of variables is removed from A, we have an analogy $P_{B'}=(B', v')$ with $P_A\leqq P_{B'}\leqq P_B$ by $w'=w|_{Var(B')}$, which is a renaming. Hence, without loss of generality, we can assume

$$B'\subseteq B \quad \text{and} \quad v'=v|_{Var(B')}. \tag{3.3}$$

Step 3: Since (3.3) holds, we have $P=And[P_{B'}; (B\backslash B', v)]\neq \perp$. From the definition of *And*, some variables in P_B may be realized by distinct variables in P. Hence, it suffices to apply *Merge* in order to obtain P_B from P. This completes the proof.

Theorem 3.1 is concerned with the comparability of analogies. Theorem 3.3 asserts that superior analogies are always constructed from their inferior analogies by applying analogy operators. Hence the theorems become important, when we consider optimal analogies, which will be discussed more in detail.

4. Canonical Analogy

Our formal analogy may have redundancies in defining a PIF under a PT. For instance, if there exist variables which define the same pair of terms, then one of the variables is redundant in defining the pair. However, we can remove this kind of redundancy by applying *Merge* operator, as we have discussed in Section 3. Similarly, if there exist atoms which define the same identity of facts, then one of the atoms is redundant. An analogy without this kind of redundancy is called canonical. In this section, we first show that we can always construct the canonical analogy by removing the redundancy from a given analogy. Moreover, we convert the canonical analogies into literals. Although this conversion is mathematically simple, it gives us an effective search space of analogies. Therefore, our canonical analogies play a key role to find optimal analogies in Section 5.

4.1 Canonical analogy and unification

Let D and D' be two atoms in an analogy (W, u). If the following condition (4.1) holds, then one of the atoms is redundant in defining PIF.

$$Du=D'u \tag{4.1}$$

Our canonical analogy (CA, for short) is defined so that (4.1) does not hold.

DEFINITION 4.1. An analogy (W, u) is called *canonical* if the mapping $\lambda_{D\in W}.\ Du: W\to S_1\times S_2$ is one-to-one.

Note that *Merge* operator can remove some redundancies in the sense of (4.1). For example, an analogy $(\{p(X), p(Y)\}, \{X-\langle a, a'\rangle, Y-\langle a, a'\rangle\})$ under $\{p(a)\}$ and $\{p(a')\}$ is not canonical. By merging X with Y, we have $(\{p(X)\}, \{X-\langle a, a'\rangle\})$, which is canonical. By combining the applications of *Merge* and *Refine*, we can remove more complicated redundancy. For instance,

$$P=(\{p(f(X), Y), p(Y, f(X))\}, \{X-\langle a, a'\rangle, Y-\langle f(a), f(a')\rangle\})$$

is a non-canonical analogy under $\{p(f(a), f(a))\}$ and $\{p(f(a'), f(a'))\}$. *Merge* is not applicable to P. However, by refining Y to $f(X_1)$, and then by merging X_1 with X, we have a CA $(\{p(f(X), f(X))\}, \{X-\langle a, a'\rangle\})$, which is superior to P.

The condition (4.1) implies unifiabilities of atoms. So we first show that each CA is constructed by unifications, and then show that it is also constructed by applying *Merge* and *Refine* operators.

LEMMA 4.1. *For any analogy P, there exists a CA P_C which is superior to P and whose PIF is exactly that of P.*

COROLLARY 4.1. *Let P and P_C be analogies stated in Lemma* 4.1. *Then P_C is obtained from P by applying Merge and Refine operators.*

PROOF OF LEMMA 4.1. Assume that (4.1) holds for some D and D' in $P=(W, u)$. Since $Du=\langle Du_1, Du_2\rangle=\langle D'u_1, D'u_2\rangle=D'u$, both u_1 and u_2 are unifiers of D and D'. Let s be the most general unifier (mgu, for short) of D and D'. Then there exists a pair $w=\langle w_1, w_2\rangle$ of substitutions with $sw_i=u_i$. Hence, if a pair $P'=(Ws, w)$ is an analogy, then $P'\geqq P$ holds, and the redundancy on D and D' is removed. Let us verify that P' is really an analogy. Remind the (standard) unification algorithm described as follows:

begin

$U_i:=D_i\ (i=1, 2)$; $v:=$*empty substitution*;
while $U_1\neq U_2$ **do**
begin
find disagreement $\{x, t\}$ such that
(4.2) x is a variable not occurring in t;
if such $\{x, t\}$ is not found **then** return "fail";
$U_i:=U_i\{t/x\}$;
$v:=v\{t/x\}$
end
return v as the desired mgu s

end

Assume $t/x \in s$. Then there exists a while-loop stage in which t'/x is added to $v=\{t_1/y_1, \cdots, t_n/y_n\}$ with $x \neq y_j$ $(1 \leqq j \leqq n)$. Since x does not appear in t' by the condition (4.2), x disappears from both U_1 and U_2 and v becomes $\{t_1q/y_1, \cdots, t_nq/y_n, t'/x\}$, where $q=\{t'/x\}$. Since t', t_jq, and U_i never contain the variable x, x does not appear in t'' whenever $t''/y \in s$ for some y. Hence, if $t/x \in s$ for some t then $x \notin Var(Ws)$. This implies $xs=x$ whenever $x \in Var(Ws)$. Since $Var(Ws) \subseteq Var(W)$, we have $xu_i=xsw_i=xw_i$ whenever $x \in Var(Ws)$. Hence $Var(Ws)w \subseteq Var(W)u$ holds. Clearly $(Ws)w=Wu$. Thus w satisfies PIC for Ws. When the condition (4.1) still holds for $P'=(Ws, w)$, we repeat the above process until the condition does not hold. The resulting analogy is clearly canonical, superior to P, and its PIF is that of P.

PROOF OF COROLLARY 4.1. Since $P_C=(B, v) \geqq P=(W, u)$, P can be improved to $P'=(B', v')$ by using *Refine* and *Merge* operators, where $B' \subseteq B$ and $v'=v|_{Var(B')}$ (refer to Step 1 and Step 2 in the proof of Theorem 3.2). Since *Refine* and *Merge* preserve PIF, $B'v'=Wu=Bv$ holds. Assume that there exists an atom D in $B \setminus B'$. Since $Wu=Bv$, there exists an atom D' in B' with $D'v=Dv$. This contradicts that (B, v) is canonical. Hence we have $B'=B$. This completes the proof.

4.2. Conversion of canonical analogies into literals

By Lemma 4.1, we can replace each analogy with the corresponding CA. We therefore consider CAs only. Note that the set of all CAs is at most finite up to the identical analogies. Thus we obtain a finite search space of our formal analogies. We covert them to literals instead of considering CAs directly.

DEFINITION 4.2.

(1) A pair of atoms is called *compatible* if they have the same predicate symbol.

(2) A set $S \subseteq S_1 \times S_2$ is called a *selection* if S is a one-to-one relation of compatible pairs of atoms.

The conversion of CA (W, u) to a literal is possible for any selection S which includes PIF Wu of (W, u). Arrange pairs in S in an arbitrary order:

$$\langle A_{11}, A_{21}\rangle, \cdots, \langle A_{1m}, A_{2m}\rangle, \text{ where } \langle A_{1j}, A_{2j}\rangle \in S \subseteq S_1 \times S_2. \tag{4.3}$$

Since (W, u) is canonical, if $\langle A_{1j}, A_{2j}\rangle \in Wu$ then there exists unique D_j in W such that $D_ju=\langle A_{1j}, A_{2j}\rangle$. We place the atom D_j at the j-th position. For a position j with $\langle A_{1j}, A_{2j}\rangle \notin Wu$, we place a reserved variable z_j. z_j is chosen so that $z_j \notin Var(W)$. In consequence, we have an ordered list $e_1, \cdots, e_n$ of atoms in W and the reserved variables. Define an extention u' of u by

$$u'=\langle u'_1, u'_2\rangle, \text{ where}$$
$$u'_i=u_i \cup \{A_{ij}/z_j : z_j \textit{ is the reserved variable}\}.$$

Clearly we have

$$\text{For each } j,\ e_ju'=\langle A_{1j}, A_{2j}\rangle. \tag{4.4}$$

The formula (4.4) is simply written as

$$Tu'=\langle T_1, T_2\rangle, \tag{4.5}$$

where

$$T_i=tup(A_{i1}, \cdots, A_{in}),$$

$$T=tup(e_1, \cdots, e_n),$$

and "*tup*" is the reserved predicate symbol to denote the list of atoms and variables. Note that we treat predicate symbols as function symbols when we use this tup notation. Then

(4.6) $Var(T)u'$ is one-to-one,

since $Var(T)u'=Var(W)u\cup\{\langle A_{1j}, A_{2j}\rangle : \langle A_{1j}, A_{2j}\rangle \notin Wu\}$, $Var(W)u$ is a set of paired terms, and $\langle A_{1j}, A_{2j}\rangle$ is a pair of atoms. The formula (4.5) is the desired literal for the CA (W, u).

We need following definitions to deal with the conversion systematically.

DEFINITION 4.3. (Plotkin [9]) For two literals L_1 and L_2, a relation $L_1\leqq L_2$ holds if there exists a substitution u such that $L_1u=L_2$. A literal L is called a generalization of literals L_1 and L_2 if $L\leqq L_i$ holds for $i=1, 2$. Moreover a generalization L of L_i is called a least generalization, if $L'\leqq L$ holds whenever L' is a generalization of L_i.

DEFINITION 4.4. For a given selection $S=\{\langle A_{1i}, A_{2i}\rangle : 1\leqq i\leqq n\}$, a structure $D(S)=(D, \leqq)$ is defined as follows: D consists of equivalence classes of generalizations of $T_i=tup(A_{i1}, \cdots, A_{in})$ $(i=1, 2)$, where the order of arrangement $\langle A_{11}, A_{21}\rangle, \cdots, \langle A_{1n}, A_{2n}\rangle$ is arbitrary, and the equivalence of literals is defined by renaming of variables. Let $[L]$ be the eqivalence class of an element L. Then $[L_1]\leqq[L_2]$ is defined by $L_1\leqq L_2$.

For literals L_1 and L_2 with $L_1u=L_2$ for some u, the substitution u is uniquely determined by restricting u to $Var(L_1)$. This is the reason why we convert CAs to literals. Moreover we can represent $L_1\leqq L_2$ by a diagram $L_1\rightarrow L_2$. Fig. 4.1 is an example of $D(S)$.

PROPOSITION 4.1. *The structure $D(S)$ is a finite lattice.*

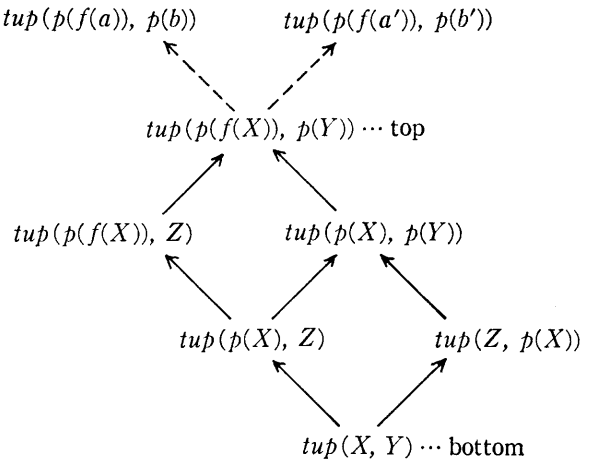

Fig. 4.1 Structure $D(S)$ for $S=\{\langle p(f(a)), p(f(a'))\rangle, \langle p(b), p(b')\rangle\}$.

This proposition is essentially due to Plotkin. Therefore we omit the proof. Note that $L_1 \vee L_2$ and $L_1 \wedge L_2$ are obtained by unification and least generalization, respectively. Moreover, the greatest element of $D(S)$ is the least generalization of T_1 and T_2. The following proposition is now obvious, since it restates the conversion (4.5) and its property (4.6) in terms of Definition 4.4.

PROPOSITION 4.2. *For a CA (W, u) and a selection $S=\{\langle A_{1j}, A_{2j}\rangle : 1 \leqq j \leqq n\}$ with $S \supseteq Wu$, there exists $T=tup(e_1, \cdots, e_n)$ in $D(S)$ such that $Var(T)u_T$ is one-to-one, and that $d(T)=(W, u)$ holds, where u_T denotes the unique pair of substitutions such that $Tu_T=\langle T_1, T_2\rangle$, $T_i=tup(A_{i1}, \cdots, A_{in})$ $(i=1, 2)$, and the function d is defined by*

$$d(T=tup(e_1, \cdots, e_n))=(\{e_j : e_j \text{ is not a variable}\}, u_T).$$

For T in $D(S)$, $Var(T)u_T$ is not necessarily one-to-one. For this reason, we define $V(S)$ to be the set of all literals T in $D(S)$ such that $Var(T)u_T$ is one-to-one. For T in $V(S)$, $d(T)$ is clearly a CA. Hence we define a structure $CAS(S)=(d(V(S)), \leqq)$ of CAs whose PIFs are included in S, where $\leqq$ is the quasi-order of analogies.

PROPOSITION 4.3. *$d(T) \leqq d(T')$ holds in $CAS(S)$ if and only if $T \leqq T'$ in $V(S)$.*

PROOF. If-part is trivial, from the definition of d.

Only-if part: Let T and T' be $tup(e_1, \cdots, e_n)$ and $tup(e'_1, \cdots, e'_n)$, respectively. Let

$$d(tup(e_1, \cdots, e_n))=(W, u) \leqq (W', u')=d(tup(e'_1, \cdots, e'_n)) \text{ by } s.$$

Then, for each e in W, es in W' and $eu=(es)u'=\langle A_{1h}, A_{2h}\rangle$ holds for some h. Let $e=e_i$ and $es=e'_j$. Since $e_iu=eu=(es)u'=e'_ju'=\langle A_{1h}, A_{2h}\rangle$ holds for some h, we have $i=j=h$, that is, $e_is=e'_i$ for $e_i \in W$. Since $e_j \notin W$ implies that e_j is a variable, we can define s' by $s'=s \cup \{e'_j/e_j : e_j \notin W\}$. Thus we have $tup(e_1, \cdots, e_n)s'=tup(e'_1, \cdots, e'_n)$. This completes the proof.

THEOREM 4.1. *For a selection S, $CAS(S)$ is a finite poset of all CAs whose PIFs are included in S, and d is an isomorphism between $CAS(S)$ and $V(S)$.*

PROOF. From Proposition 4.2 and the definition of $CAS(S)$, it is clear that $CAS(S)$ consists of all CAs whose PIFs are included in S. From Proposition 4.3. if $d(T) \leqq d(T')$ and $d(T) \geqq d(T')$ hold, then we have $T \leqq T'$ and $T \geqq T'$. Hence $T=T'$, by the definition of $D(S)$. Thus $\geqq$ is a partial order, if analogies are restricted to those in $d(V(S))$. Moreover d is an isomorphism between $CAS(S)$ and $V(S)$.

5. Maximal Analogy and Greatest Analogy

Our formal analogy is a partial identity between two sets of facts (PIF) under a pairing of terms (PT). Hence the problem of analogy detection requires to find

(D1) the PT as simple as possible, and

(D2) the PIF under the PT as large as possible.

We first define a maximal analogy (MA, for short) by the maximality with respect to our ordering of analogies. We show that a canonical analogy is maximal if and only if the operators cannot be applicable to it. Since the non-applicability of operators means that both (D1) and (D2) satisfied, our MAs are really what we desire.

The next problem is how to find such MAs. One way for this is to apply the

operators to trivial analogies, whose PIFs are singleton, until any operator is not applicable. The search strategy is bottom-up. We do not adopt this strategy, since it finds many non-optimal analogies until the desired optimal ones are found. Instead we present a top-down method to find all the MAs by using canonical analogies.

Generally several MAs can exist for a given pair of sets of facts. Since, for two distinct MAs P_1 and P_2, neither $P_1 \leqq P_2$ nor $P_2 \leqq P_1$ holds, we should have an additional criterion to compare them. We leave this as a future problem. On the other hand, if there exists the uniqui MA then it is greatest with respect to our ordering. We call such MA the greatest analogy, and also show a necessary and sufficient condition for the greatest analogy to exist.

5.1. Maximal analogy and its characterization

DEFINITION 5.1. An analogy (W, u) is called *maximal* if $(W, u) \leqq (W', u')$ then $(W, u) - (W', u')$ holds for any analogy (W', u').

In what follows, for two analogies P_1 and P_2, we define $P_1 < P_2$ by $P_1 \leqq P_2$ and $P_1 \ngeqq P_2$, and we say that P_2 is properly superior to P_1. From the definition, for any MA P, there is no analogy which is properly superior to P. Since non-canonical MA has redundancies in describing PIF under PT, we consider CAs only in order to characterize MAs by operators.

THEOREM 5.1. *A CA (W, u) is maximal if and only if* (1) *Merge and Refine are not applicable to (W, u) and* (2) *there exist no atom A and no substitution v such that*

$$Av \in S_1 \times S_2,\ Av \notin Wu, \quad \text{and} \quad And[(W, u);\ (\{A\}, v)] \neq \perp .$$

PROOF. Assume that *Merge* operator is applicable to (W, u). Let $(Wv, u) = Merge[(W, u);\ V]$, where V is a set of variables satisfying $xu = yu$ whenever x and y are in V, and $v = \{x_0/y : y \in V \setminus \{x_0\}\}$ for an arbitrary chosen variable x_0 in V. Let $(Wv, u) \leqq (W, u)$ by a substitution s. From the definition of v, $xvu = xu$ holds for any $x \in Var(W)$. Hence $Avsu = Avu = Au$ holds for any $A \in W$. Since $Avs \in W$ and (W, u) is canonical, we have $Avs = A$ for any $A \in W$. Hence $xvs = x$ for all $x \in Var(W)$. Hence, for distinct variables x and y in V, we have $x = xvs = x_0 s = yvs = y$. This is a contradiction. Thus $(Wv, u) > (W, u)$, therefore (W, u) is not maximal. For *Refine* operator, it is similarly proved that refined analogy is properly superior to (W, u) if *Refine* is applicable.

Conversely assume that (W, u) is not maximal. Then, according to Lemma 4.1, there exists a CA (W', u') with $(W, u) \leqq (W', u')$. Let $(W, u) \leqq (W', u')$ by a substitution v, and $V = \{x \in Var(W) : xv \text{ is a proper term}\}$. From the proof of Theorem 3.2, *Refine* is applicable if $V \neq \phi$, and *Merge* is applicable if $V = \phi$ and there exist distinct variables x and y in $Var(W)$ with $xv = yv$. Hence it suffices to verify only the case that v replaces distinct variables by distinct ones. Then, without loss of generality, W is a subset of W' and $u = u'|_{Var(W)}$. If $W = W'$ then $(W', u') \leqq (W, u)$ holds. Hence $W' \supsetneqq W$. Then, for an atom A in $W' \setminus W$, we have $And[(W, u);\ (\{A\}, u')] \neq \perp$.

The condition (1) means that we cannot observe the PT of (W, u) more simply. The condition (2) means that we cannot enlarge the PIF of (W, u) by adding a new identity of facts. Thus our maximal (canonical) analogies are optimal with respect to both PIF and PT.

5.2. A top-down detection of maximal analogies

Now let us consider the problem of finding MAs. We use CAs to find MAs. First we define a "global" structure CAS of CAs.

DEFINITION 5.2. Let C and $\leqq$ be the set of all CAs (except for the equality of analogies) and the quasi-order defined in Section 3, respectively. Then the structure $CAS=(C, \leqq)$ is called a canonical analogy structure.

PROPOSITION 5.1. *CAS is a finite poset.*

PROOF. Let (W, u) and (W', u') be two equivalent CAs. Clearly $Wu=W'u'$ holds. From Theorem 4.1, $(W, u)\leqq(W', u')$ and $(W', u')\leqq(W, u)$ holds in $CAS(Wu)$. Thus $(W, u)=(W', u')$.

THEOREM 5.2. *An analogy is maximal if and only if it is equivalent to a maximal element of CAS.*

PROOF. Let P be maximal in CAS, and $P'\geqq P$ for some P'. From Lemma 4.1, there exists a CA P'' with $P''\geqq P'\geqq P$. This implies $P''\geqq P$ in CAS. Since P is maximal, we have $P''=P$. Hence $P\geqq P'$. Thus P is an MA. Conversely let P be an MA. According to Lemma 4.1, there exists a CA P' with $P'\geqq P$. By the maximality of P, $P-P'$ holds, It is now trivial that P' is a maximal element of CAS.

From Theorem 5.2, in order to find all MAs, it only suffices to find all maximal elements of CAS (up to equivalence of analogies). For this purpose, we decompose the poset CAS to the poset $CAS(S)$. Note that, from Theorem 4.1, for two selections S and S' with $S\subseteq S'$, $CAS(S)\subseteq CAS(S')$ holds. Hence we consider maximal elements under a largest possible selection.

DEFINITION 5.3. A selection S is called *maximal* if there is no selection which includes S properly.

Since, for any selection S, $CAS(S)$ is a full subposet of CAS, the following proposition is obvious.

PROPOSITION 5.2. *A maximal element of CAS is also maximal in CAS(S) for some maximal selection S.*

Let S be a maximal selection. Then by Proposition 5.2, in order to find MAs, we find all the maximal elements of $CAS(S)$, and then reject them if they are not maximal in CAS. To find the maximal element of $CAS(S)$, we may only search $D(S)$ in a top-down manner until maximal elements of $V(S)$ are found, since $V(S)$ and $CAS(S)=d(V(S))$ are isomorphic by the function d (Theorem 4.1). The following definitions are necessary to state our top-down search in $D(S)$ strictly.

Let L_{top} be the greatest element of $D(S)$ for a given selection $S\subseteq S_1\times S_2$. Then L_{top} has no individual constants, since S_1 and S_2 have no common individual constants. For an occurrence a of a term t in the literal L_{top}, the term t is denoted by $exp(a)$. For two occurrences a and a' of terms. we define that a relation $a<a'$ holds if a occurs in a'. $a\leqq a'$ is defined to be $a=a'$ or $a<a'$. When we represent L_{top} by a tree, the occurrences correspond to nodes, and $a<a'$ means that a is a successor of a'. A set F of occurrences of terms in the literal L_{top} is called a frontier if

(1) for any a in F, $exp(a)$ is a proper term, and

(2) for any a and a' in F, $a<a'$ does not hold.

For two frontiers F and F', we define the relation $\leqq$ by

$F \leqq F'$ if, for any $a \in F$, there exists a' in F' with $a \leqq a'$.

Moreover, with each frontier F of L_{top}, we associate a literal $C[F]$, called a *contraction,* as follows:

Each occurrence a in F is replaced by a new variable $\langle exp(a) \rangle$ in L_{top}. Then the resulting literal is $C[F]$, and the variable $\langle exp(o) \rangle$ is called $\langle\ \rangle$*-variable.*

EXAMPLE 5.1. Let $L_{top}=tup(p(f(x)), q(y, f(x)))$, and $F=\{a\}$, where a is the rightmost occurrence of $f(x)$. Then

$$C[F]=tup(p(f(x)), q(y, \langle f(x) \rangle)).$$

PROPOSITION 5.3.

(1) $C[F] \leqq L_{top}$, *therefore* $C[F] \in D(S)$.

(2) $F \geqq F'$ *iff* $C[F] \leqq C[F']$.

(3) *For any literal* $L \in D(S)$, *there exists a contraction* $C[F]$ *of* L_{top} *such that* $L \leqq C[F]$. *Especially* $C[F] \in V(S)$ *whenever* $L \in V(S)$.

PROOF. (1) is trivial by considering a substitution

$$u=\{exp(a)/\langle exp(a) \rangle : a \in F\}.$$

Note that $xu=x$ holds for each variable x except $\langle\ \rangle$-variable.

(2) For occurrences $a, a_1, \cdots, a_k$ such that $a \geqq a_j$, $t(a; a_1, \cdots, a_k)$ denotes the term which is obtained by replacing each a_j in the term $exp(a)$ by the variable $\langle exp(a_j) \rangle$. Define a substitution v by

$$\begin{aligned} v=&\{exp(a)/\langle exp(a) \rangle : a \in F \text{ and there is no } a' \in F' \text{ such that } a \geqq a'\} \\ &\cup \{t(a; a'_1, \cdots, a'_k)/\langle exp(a) \rangle : a \in F \text{ and there are} \\ &\qquad\qquad a'_1, \cdots, a'_k \text{ in } F' (k>0) \text{ such that } a'_j \leqq a\} \end{aligned}$$

Then $C[F]v=C[F']$ whenever $F \geqq F'$. Conversely assume that

$$C[F]v=C[F'] \quad \text{and} \quad C[F]u=C[F']u'=L_{top}.$$

Let

$$C[F]=term(\langle exp(a_1) \rangle, \cdots, \langle exp(a_n) \rangle, x_1, \cdots, x_k),$$

where $term(y_1, \cdots, y_n, z_1, \cdots, z_k)$ is a term in which there are no multiple occurrences of variables, and $x_1, \cdots, x_k$ are all the non-$\langle\ \rangle$-variables in $C[F]$. Since $vu'=u$ holds, we have $x_j=x_ju=x_jvu'$. Therefore, x_jv is not a $\langle\ \rangle$-variable. Note that $\langle\ \rangle$-variables correspond to proper terms in L_{top}. Since $yu'=y$ whenever y is non-$\langle\ \rangle$-variable, we have $x_j=x_jvu'=x_jv$. Thus we have

$$C[F']=term(\langle exp(a_1) \rangle v, \cdots, \langle exp(a_n) \rangle v, x_1, \cdots, x_k).$$

Hence, for any a' in F', the occurrence of $\langle exp(a') \rangle$ occurs in the j-th argument $\langle exp(a_j) \rangle$ for some j. This implies $a' \leqq a_j$.

(3) Let L be a term $t(x_1, \cdots, x_m)$ with variables $x_1, \cdots, x_m$, and assume that $Lu=L_{top}$. Define a contraction L' and a substitution v as follows:

$$\begin{aligned} L'&=term([x_1u], \cdots, [x_mu]), \\ v&=\{[x_ju]/x_j : 1 \leqq j \leqq m\}, \end{aligned}$$

where

$$[x_ju]=if\ x_ju \text{ is a proper term then } \langle x_ju \rangle \text{ else } x_ju.$$

Clearly $Lv=L'$ and L' is a contraction. Moreover, since $\langle x_j u\rangle$ in L' is replaced by $x_j u$ in L_{top}, $L\in V(S)$ iff $L'\in V(S)$.

Let $VC(S):=V(S)\cap\{C[F]: F \textit{ is a frontier of } L_{top}\}$. Then, from Proposition 5.3 (3), maximal elements of $VC(S)$ are those of $V(S)$ and vice versa. Hence, it suffices to find maximal elements of $VC(S)$ in order to find maximal elements of $CAS(S)$. For this purpose, we consider a top-down generation of contractions. We frist define a *one-step contraction*. Let F and a be a frontier and an occurrence with $a\notin F$, respectively. Moreover let a_j be the occurrence of t_j in $exp(a)=f(t_1, \cdots, t_n)$. If $a_j\in F$ or $exp(a_j)=t_j$ is a variable, for all j, then we have a frontier $F'=F\setminus\{a_j: exp(a_j) \textit{ is a proper term}\}\cup\{a\}$. The literal $C[F']$ thus obtained is called a one-step contraction of $C[F]$. For example, there are two one-step contractions of $tup(p(f(x)), q(y, \langle f(x)\rangle))$:

$$tup(p(\langle f(x)\rangle), q(y, \langle f(x)\rangle)),$$
$$tup(p(f(x)), \langle q(y, f(x))\rangle).$$

Procedure 5.1. The set-valued function *level* (n) is defined by

$$level(0)=\{C[\phi]\},$$
$$level(n+1)=\{C[F']: C[F'] \textit{ is a one-step contraction of some } C[F] \textit{ in } level(n)\}.$$

In what follows, for two literals L_1 and L_2, we define $L_1<L_2$ by $L_1\leqq L_2$ and $L_2\not\leqq L_1$.

PROPOSITION 5.4.

(1) *For any L in level $(n+1)$, there exists L' in level(n) such that $L'>L$.*

(2) *For two distinct L and L' in level(n) if any, L and L' are incomparable, that is, neither $L\leqq L'$ nor $L'\leqq L$ holds.*

(3) *$L_1<L_2$ never holds whenever $L_i\in level(n_i)$ and $n_1<n_2$.*

PROOF. (1) Let $L=C[F]$, $L'=C[F']$, and $F=F'\setminus\{a_1, \cdots, a_n\}\cup\{a\}$. From the definition of one-step contraction, $a_j<a$ for $j=1, \cdots, n$. Hence $F'\leqq F$, therefore $C[F]=L\leqq C[F']=L'$ holds (Proposition 5.3 (2)). Now assume that $F\leqq F'$. Then, for any a in F', there exist $a'\in F$ and $a''\in F'$ such that $a\leqq a'\leqq a''$ holds. Since F' is a frontier, we have $a=a''$, therefore $a=a'$. This shows $F'\subseteq F$. Similarly, we have $F\subseteq F'$. This contradicts $F\neq F'$. Thus $F\not\leqq F'$. Hence $C[F']=L'\not\leqq L=C[F]$.

(2) We use the following function c:

(i) for a term t,

$$c(t)=\begin{cases} 1+\sum_{i=1}^{n} c(t_i) & \textit{if } t \textit{ is a proper term } f(t_1, \cdots, t_n) \\ 0 & \textit{otherwise}, \end{cases}$$

(ii) for an occurrence a of term, $c(a)=c(exp(a))$,

(iii) for a frontier F, $c(F)=\sum_{a\in F} c(a)$.

Then, by using induction, it is easily verified that

$$\text{if } C[F]\in level(n) \text{ then } c(F)=n \text{ holds.}$$

Let $C[F]$ and $C[F']$ be distinct contractions in $level(n)$ for some n. Since $F=F'$ iff $F\leqq F'$ and $F'\leqq F$, $F\neq F'$ implies (C1) or (C2):

(C1) There exists a_1 in F such that there is no a in F' with $a_1 \leqq a$,

(C2) $F' \leqq F$, therefore there exists a_2 in F' such that there is no a in F with $a_2 \leqq a$.

We verify (2) only for the case (C1), since it is similarly proved for the other case (C2). Assume that $F' \leqq F$. Let $d(a)=\{a' \in F' : a' \leqq a\}$, for each a in F. Then it is easy to verify

(5.1) $$d(a) \cap d(a') = \phi \text{ whenever } a \text{ and } a' \text{ are distinct,}$$

(5.2) $$c(a) \geqq \sum_{a' \in d(a)} c(a').$$

Note that the right-hand side of the inequality is defined to be 0 if $d(a)=\phi$, and also note that the equality of (5.2) holds iff $a \in F'$. From (C1), we have

(5.3) $$c(a_1) > \sum_{a' \in d(a)} c(a').$$

Since $F' \leqq F$, we have

(5.4) $$F' = \bigcup_{a \in F} d(a),$$

which is a disjoint union by (5.1). From (5.2) and (5.4), we have

(5.5) $$\begin{aligned} c(F') &= \sum_{a \in F'} c(a) = \sum_{a \in F} \sum_{a' \in d(a)} c(a') \\ &\leqq \sum_{a \in F} c(a) = c(F). \end{aligned}$$

However, from (5.3), the equalty of (5.5) does not hold. Thus we have $c(F') < c(F)$. This contradicts that both $C[F]$ and $C[F']$ are in $level(n)$. Hence we have $F \leqq F'$ and $F' \leqq F$.

(3): Assume that $L_1 \leqq L_2$ holds for L_i in $level(n_i)$ and $n_1 < n_2$. From (1), there exists L_3 in $level(n_1)$ such that $L_2 < L_3$. Hence we have $L_1 < L_3$. This contradicts (2).

From Proposition 5.4, $level(n_1) \cap level(n_2) = \phi$ whenever $n_1 \neq n_2$. Based on Proposition 5.4 and Procedure 5.1, a procedure to find all maximal elements of $VC(S)$ for a given selection S is described as follows:

Let $S=\{\langle A_{1,i}, A_{2,i}\rangle : 1 \leqq i \leqq n\}$, $L_{top}=tup(t_1, \cdots, t_n)$, and $t_j u = \langle A_{1,j}, A_{2,j}\rangle$ for each j. Then $tup(\langle t_1\rangle, \cdots, \langle t_n\rangle)$ is the bottom of $D(S)$, therefore the least element of $VC(S)$.

Procedure 5.2.

//"*level*" and "*succ*" are set-valued variables//

begin

level : = $\{L_{top}\}$;

succ : = ϕ ;

if $L_{top} \in V(S)$ **then** *return* (*d*(*level*)) ;

while *level* $\neq \phi$ **or** *level* $\neq \{tup(\langle t_1\rangle, \cdots, \langle t_n\rangle)\}$ **do**

begin

level : = *Generate* (*level*) ;

level : = *Prune* (*level*)

end

return (*d* (*succ*))

end

"Generate" in the procedure above is an operation which generates all possible one-step contractions of contractions in the set *level*. By the procedure "Prune", we can prune our search space. *Prune* behaves as follows: For each L in *level*, *Prune* first checks if $L \leqq L'$ for some L' in the set *succ*. If this occurs, L is removed from the set *level*. Then it checks if $L \in V(S)$ or not for each L in *level*. If so, L is removed from the set *level* and stored in the set *succ*.

THEOREM 5.3. *Procedure* 5.2 *finds every maximal element of* $CAS(S)$.

PROOF. Let $l(n)$ be the set *level* at the n-th stage of while loop. Then it is clear that $l(n) \subseteq level(n)$. Let $C[F_i] \in level(n_i)$ $(i=1, 2)$, $C[F_1] \neq C[F_2]$, and $C[F_1] \geqq C[F_2]$, that is, $C[F_1] > C[F_2]$. From Proposition 5.4, we have $n_1 \leqq n_2$ and

(*) $\quad n_1 < n_2$ whenever $C[F_1] > C[F_2]$ and $C[F_i] \in level(n_i)$.

Let $L = C[F] \in level(n)$ be a maximal element of $VC(S)$. Let $L < L' \in level(n')$. Then, from (*), $n' < n$ holds. By the maximality of L, we have $L' \notin VC(S)$. Hence each L' satisfying $L' > L$ is generated without removed from the set *level*. Therefore $L \in l(n)$, and L is stored in the set *succ*. Moreover, by the condition (*), any maximal element of $VC(S)$ is generated earlier than its inferior element of $VC(S)$. Hence such an inferior element is pruned by *Prune*. Thus the set *succ* contains only maximal elements of $VC(S)$.

Even if we consider maximal elements of $CAS(S)$ for a maximal selection S, they are not necessarily those of CAS. For instance, consider the following example.

EXAMPLE 5.2. Let

$$S_1 = \{r_2(a),\ r_1(f(a))\},$$
$$S_2 = \{r_2(b'),\ r_2(a'),\ r_1(f(a'))\}.$$

For a maximal selection

$$SEL1 = \{\langle r_2(a),\ r_2(b')\rangle,\ \langle r_1(f(a)),\ r_1(f(a'))\rangle\},$$
$$P_1 = (\{r_1(f(y))\}\ ;\ \{y - \langle a,\ a'\rangle\})$$

and

$$P_2 = (\{r_2(x),\ r_1(z)\}\ ;\ \{x - \langle a,\ b'\rangle,\ z - \langle f(a),\ f(a')\rangle\})$$

are maximal elements. There is another maximal selection

$$SEL2 = \{\langle r_2(a),\ r_2(a')\rangle,\ \langle r_1(f(a)),\ r_1(f(a'))\rangle\}.$$

For this selection, $L_{top} = tup(r_2(x),\ r_1(f(x)))$ is in $VC(S)$, thetefore $P_3 = (\{r_2(x),\ r_1(f(x))\}\ ;\ \{x - \langle a,\ a'\rangle\})$ is the unique maximal element of $CAS(SEL2)$. By taking union of

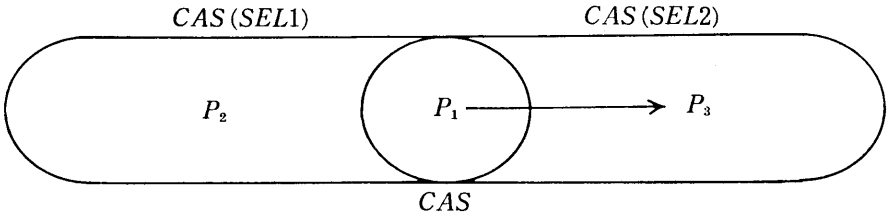

Fig. 5.1 A portion of CAS in Example 2.2.

$CAS(SEL1)$ and $CAS(SEL2)$, we obtain CAS, since maximal selections are just $SEL1$ and $SEL2$. Observe that P_1 is not maximal in CAS, since P_3 is properly superior to P_1.

In other words, we cannot enlarge PIF of P_1 under the constraint that PIF is included in $SEL1$. On the other hand, the enlargement of PIF is possible under $SEL2$. This is a reason why P_1 is not maximal in CAS. Formally we have the following proposition.

PROPOSITION 5.5. *Let a maximal element (W, u) of $CAS(S)$ be not maximal in CAS. Then there exists another selection S' and maximal element (W', u') of $CAS(S')$ such that $(W, u)<(W', u')$ and $Wu \subsetneqq W'u'$ hold.*

PROOF. Since (W, u) is not maximal in CAS, there exists a maximal element (W', u') of CAS with $(W', u')>(W, u)$. Clearly $W'u' \supseteq Wu$ holds. Assume $Wu=W'u' \subseteq S$. From Theorem 4.1, (W', u') is also an element of $CAS(S)$, therefore $(W', u')>(W, u)$ holds in $CAS(S)$. This contradicts the maximality of (W, u) in $CAS(S)$.

Even if $Wu \subsetneqq W'u'$ holds, it is possible that (W, u) is maximal in CAS.

EXAMPLE 5.3. Let

$$S_1=\{r_1(g(a)),\ r_2(a,\ g(a))\}\ ,$$
$$S_2=\{r_1(g(a')),\ r_2(f(b'),\ g(a')),\ r_2(b',\ g(a'))\}\ .$$

There exist two maximal selections:

$$SEL1=\{\langle r_1(g(a)),\ r_1(g(a'))\rangle,\ \langle r_2(a,\ g(a)),\ r_2(f(b'),\ g(a'))\rangle\}\ ,$$
$$SEL2=\{\langle r_1(g(a)),\ r_1(g(a'))\rangle,\ \langle r_2(a,\ g(a)),\ r_2(b',\ g(a'))\rangle\}\ .$$

For $SEL1$, there are two maximal elements P_1 and P_2:

$$P_1=(\{r_1(g(x))\}\ ;\ \{x-\langle a,\ a'\rangle\})\ ,$$
$$P_2=(\{r_1(x),\ r_2(y,\ x)\}\ ;\ \{x-\langle g(a),\ g(a')\rangle,\ y-\langle a,\ f(b')\rangle\})\ .$$

For $SEL2$, there exist maximal elements P_1 and P_3, where

$$P_3=(\{r_1(x),\ r_2(y,\ x)\}\ ;\ \{x-\langle g(a),\ g(a')\rangle,\ y-\langle a,\ b'\rangle\})\ .$$

In contrast with Example 5.2, P_1, P_2 and P_3 are incomparable, and the PIF of P_1 is included in that of P_i for $i=2, 3$.

Thus, for two maximal elements (W, u) and (W', u') under some selections, the condition $W'u' \supsetneqq Wu$ is not sufficient to reject (W, u). However, it can be used as precondition. From Proposition 4.3 and Proposition 5.5, it is clear that the following procedure 5.3 correctly reject (W, u) if it is not maximal in CAS.

Procedure 5.3.

for a given $P=(W, u)$,

begin
  for each maximal selection S' with $S' \neq S$ **do**
  for each maximal element $P'=(W', u')$ of $CAS(S')$
      such that $Wu \subsetneqq W'u' \subseteq S'$ **do**
    begin
      convert P and P' to literals T and T' in $V(S')$
          (such that $d(T)=P$ and $d(T')=P'$);

check if $T \leqq T'$ or not;
if $T \leqq T'$ holds **then** return ("P is not maximal")
end
return ("P is maximal")
end

5.3. Greatest analogy and its characterization

We have developed a top-down method to find all the maximal analogies by using canonical analogy structure. As we have observed, there exist several maximal analogies. Maximal analogies are optimal and distinct ones are incomparable. On the other hand, if the maximal analogy exists uniquely, then the problem to decide the best analogy becomes very easy.

DEFINITION 5.4. An analogy (W, u) is called greatest if $(W, u) \geqq (W', u')$ holds for any analogy (W', u').

The following proposition is obvious, so we omit the proof.

PROPOSITION 5.6. *For S_1 and S_2 for which at least one analogy exists, there exists the greatest analogy if and only if maximal analogies are unique up to equivalence of analogies.*

The existence of greatest analogy essentially depends on the superiority of one analogy over two analogies. Since this superiority is characterized by *And* and *Refine* operators in Section 3, the existence of greatest analogy relates to these analogy operators. In fact, we have the following theorem:

THEOREM 5.4. *For S_1 and S_2 for which at least one analogy exists, there exists the greatest analogy if and only if And operator is applicable for any two analogies, by refining if necessary.*

PROOF. Since any greatest analogy dominates all the analogies, only-if part is a direct consequence of Theorem 3.1. Conversely let $P_1, \cdots, P_n$ be all the maximal elements of *CAS*. By the assumption of if-part, there exists P_i' $(i=1, 2)$, which is either P_i or a refined version of P_i, such that

$$P_{12} := And[P_1'; P_2'] \neq \perp .$$

Similarly we have

$$P_{1j} := And[P_{1(j-1)}'; P_j'] \neq \perp, \ for \ 2 \leqq j \leqq n,$$

where

$$P_{1k}' \text{ is either } P_{1k} \text{ or its refined version.}$$

Clearly $P_{1n} \geqq P_i' \geqq P_i$ holds. It follows from Lemma 4.1 that P_{1n} is the greatest analogy.

6. Concluding Remarks

We have presented some formal definition of analogy and then we have formalized the problem of analogy detection which should play a key role in analogical reasoning. There still remain many problems to be solved.

(1) Formal analogy with an equivalence of terms.

By the present definition, our formal analogy is a syntactic partial identity of formal descriptions represented by sets of facts. Therefore, due to invalid description, we may

lose a good analogy. Moreover, for an analogy (W, u), pairings of terms may not be those of elements in a domain in question, since distinct terms may define the same element. In order to overcome such difficulties, it suffices to introduce an equivalence of terms and a partial identity condition for the eqivalence classes of terms.

(2) Use of deduction or abstraction.

In order to extract useful analogy, we should positively make use of information about system, which are related to the use of deduction.

(3) Analogical problem solving.

In general, problem solving needs a mechanical method to find solution in a solution space, given problems in a problem space. The study of analogy started with applying the concept of analogy to the problem solving. Now we have a mathematical framework of analogy, as we have developed in this paper. In order to cope with the analogical problem solving, based on our theory, there are two problems to be answered. First we must have a translator which transforms each problem in the problem space into our description. After this translation, we take a formal analogy between the problems, and then we obtain a partial solution of one problem, which corresponds that of another problem. Then our second problem arises. When problems in the problem space are analogous, are the corresponding solutions also analogous? This concerns the applicability of analogical problem solving. Some theoretical investigations are necessary to answer this problem and to utilize the analogical problem solving. Especially it is important to find classes of problem solving for which we positively answer our second problem.

Acknowledgements

The author would like to thank Prof. Seiiti Huzino for his constructive comments and encouragement. The author also wishes to express his sincere thanks to Prof. Setsuo Arikawa for many useful discussions on the problem of analogy.

References

[1] BLUM, L. and BLUM, M.: *Toward a mathematical theory of inductive inference*, Information and Control, **38** (1975), 125-155.

[2] CARBONELL, J.G.: *A computational model of analogical problem solving*, IJCAI-81 (1981), 147-152.

[3] CHANG, C. and LEE, R.C.: Symbolic logic and mechanical theorem proving, Academic Press, London (1973).

[4] CHAPMAN, D.: *A programming testing assistant*, CACM, **25** (1982), 625-634.

[5] HARAGUCHI, M. and ARIKAWA, S.: *An application of analogical reasoning to knowledge information system*, Proc. of the Summer Symposium on Knowledge Understanding Systems, IIASSIS (1984), 161-164 (Japanese).

[6] KLING, R.E.: *A paradigm for reasoning by analogy*, Artificial Intelligence, **2** (1971), 147-178.

[7] KLIX, R.E. and MEER, F.D.: *Analogical reasoning-an approach to mechanisms underlying human intelligence performances*, Human and Artificial Intelligence, North-Holland, New York (1979), 193-212.

[8] PLAISTED, D.A.: *Theorem proving with abstraction*, Artificial Intelligence, **16** (1981), 47-

108.
[9] PLOTKIN, G.D.: *A note on inductive generalization,* Machine Intelligence, **5** (1970), 153-216.
[10] PLOTKIN, G.D.: *A further note on inductive generalization,* Machine Intelligence, **6** (1971), 101-124.
[11] POLYA, G.: Induction and analogy in mathematics, Princeton University Press, Princeton (1954).
[12] SHAPIRO, E.Y.: *Inductive inference of theories from facts,* Research Report 192, Yale Univ. (1981).
[13] SHAPIRO, E.Y.: *Algorithmic program debugging,* Research Report 237, Yale Univ. (1982).
[14] TANGWONGSAN, S. and FU, K.S.: *An application of learning to robot planning,* Int. J. of Computer and Information Science, **8** (1979), 303-333.
[15] WINSTON, P.H.: *Learning and reasoning by analogy,* CACM, **23** (1980), 689-703.

Communicated by S. Arikawa
Received July 25, 1984
Revised September 29, 1984