ICOT Technical Report: TR-450

TR-450

Operation Presumption: Knowledge Acquisition by Induction

by
H. Taki & Y. Fujii

January, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Operation Presumption: Knowledge Acquisition by Induction

Hirokazu Taki and *Yuichi Fujii

ICOT Research Center, Institute for New Generation Computer Technology,
4-28, Mita 1-Chome, Minato-ku, Tokyo 108 Japan
Phone: +81-3-456-3192, Csnet:htaki%icot.jp@relay.cs.net

Abstract

This paper describes a knowledge acquisition support system which presumes knowledge fragments by induction. In this system, human problem solving knowledge is extracted as a set of operations. There are three knowledge acquisition phases in the knowledge acquisition process. The first is the expert model construction phase, in which problem solving knowledge is extracted as a set of operations and the type of each operation is defined. Operation types are prepared by this system. The second phase is the model instantiation phase, in which the system extracts detailed knowledge according to the operation types Sometimes, it is not easy to extract detailed information without knowledge seeds. Therefore, this system supports an inductive mechanism which makes operation presumptions in order to stimulate human experts to associate detailed knowledge. The third phase is the knowledge refinement phase, in which the system inference engine evaluates the knowledge base at the operation level to check whether the knowledge is sufficient. This paper introduces an interactive knowledge acquisition support system, EPSILON; knowledge representation, Expert Model; an interactive knowledge acquisition method, pre-post method; and an operation presumption method.

1. Introduction

There is a big problem in the building knowledge base of expert systems, called the knowledge acquisition bottle-neck. Knowledge acquisition support systems have been developed to solve this problem. There are three phases in the knowledge acquisition process; (1) the expert model modelling phase, (2) the expert model instantiation phase, and (3) the knowledge refinement phase. In the expert model modelling phase, a human problem solving model is extracted. In the expert model instantiation phase, knowledge, which is used in problem solving, is extracted. Knowledge consists of concepts, relations between concepts, and problem solving knowledge using concepts and their relations.

In the knowledge refinement phase, the knowledge acquisition system detects shortage and inconsistency of knowledge, and guides the human expert to modify the knowledge. Many knowledge acquisition support systems have assumed problem solving knowledge before knowledge acquisition. A problem solving strategy, which MORE [Kahn 85] assumes for expert systems, includes the usage of domain models for diagnosis tasks. A problem solving strategy, which ETS [Boose 84] assumes for expert systems, includes the classification method of items using knowledge in the form of a rating grid. EPSILON [Taki 87] has no problem solving strategy before knowledge acquisition. It extracts

---

* Current address is :
Engineering Strategy Planning Headquarters, Nippon Telegraph and Telephone Corporation, 1-6 Uchisaiwai-Cho, 1-Chome, Chiyoda-ku, Tokyo 100, Japan

problem solving knowledge first, then acquires other knowledge which is used by problem solving knowledge. These knowledge acquisition support systems use some methods to extract knowledge; ETS uses the personal construct theory to stimulate the human expert to associate constructs related to items. It also uses a refinement method which translates a rating grid into an implication graph and cluster trees to stimulate the human expert to find some shortage or inconsistency of knowledge. MORE uses a refine method which refines a domain model using eight interview strategies. EPSILON/One also supports a knowledge acquisition method, the pre-post method, to stimulate the human expert to remember operations of his expert jobs and to extract detailed knowledge of each operation using operation types.

However, it is difficult for us to prepare a general method to extract knowledge. To associate expert knowledge, we must prepare knowledge seeds. EPSILON/One asks the expert for detailed information of operations according to the operation type, but sometimes, the expert cannot answer or remember information. Humans can modify knowledge more easy than they can remember it. Therefore, to give EPSILON the capacity to extract operation information like this, we have been developing the operation presumption method, which supports making knowledge seeds to associate expert knowledge.

The following sections introduce knowledge representation for knowledge acquisition, Expert Model; a knowledge acquisition support system, EPSILON/One; and an operation presumption method using inductive inference.

2. Expert Model

This section explains the original idea of the Expert Model, and gives details of Expert Model structures.

2.1 Basic Idea of Expert Model

The Expert Model is based on two ideas: the simplified expert task model and analysis and grouping of diagnosis expert knowledge written in production rule form.

(1) Simplified Expert Task Model

Expert systems can be distinguished as different task types. According to "Building Expert Systems" [Hayes-Roth 83], there are 10 generic categories of knowledge engineering applications: interpretation, prediction, diagnosis, design, planning, monitoring, debugging, repair, instruction and control. We tried to define these categories as simple models, because simple expert task models provide the expert task images for the human expert to represent his knowledge. Two samples are introduced later.

Simple Diagnosis Task Model

Fig.1 shows a diagnosis task. Diagnosis tasks are represented in tree structures and their search algorithm in general AI architectures. However, the tree structure does not represent expert task images. A simpler model is a filter model. In Fig.1, G1 denotes the possible results of trouble positions, F1 is a filter to select real results, and G2 denotes the results of a diagnosis task. Fig.2 also shows a diagnosis task, but this model has some sub-

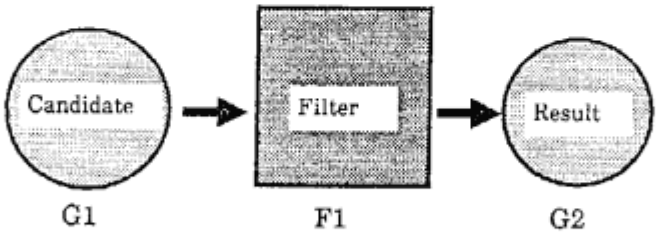

Fig.1 Simple Diagnosis Task Model

tasks which are the same as these of Fig.1.

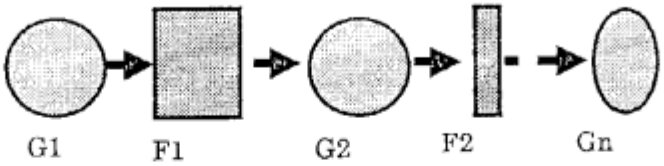

Fig.2 Simple Diagnosis Task Model(No.2)

Simple Design Task Model

Fig.3 shows a very simple design task. Design tasks are very complex; therefore, this model is actually a sub-task of a design task. It consists of a part modification phase, a part assembly phase and a good selection result phase. G4 and G5 are design-part groups, F4 is a modifier, G10 is a group of modified G4 parts, F5 is an assembly or combination operation, G11 is an interim result group, F6 is a selector, and G12 is a group of design task results. (G5 should not be modified.) We have considered simplified expert task models and created a model (operation) which is constructed with a source element-group, evaluators (e.g. select filter) and a destination element-group.

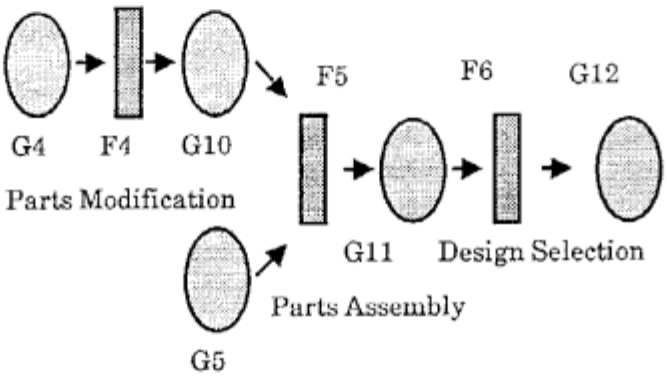

Fig.3 Simple Design Task Model

(2) Study from Production Rules of Diagnosis Expert Systems

A production rule is a general knowledge representation for expert systems. Therefore, knowledge engineers must have techniques of rule writing to represent many kinds of knowledge. We assumed that those techniques may appear in the form of rules. Seven operations have been discovered in production rule sets: SELECTION, CLASSIFICATION, ORDERING, COMBINATION, TRANSLATION, INPUT and OUTPUT. The combined result of both ideas is the generic operation.

2.2 Operation Types

The Expert Model is a specialized knowledge representation to collect expert task smoothly from human experts. It is not necessary for the knowledge acquisition process to provide a general knowledge representation, i.e. , production rule form. This section explains the components of the Expert Model.

2. 2. 1 Generic Operations

Generic operations are kernel knowledge representations in the Expert Model. The generic operations have been derived from simple expert task models and by analyzing production rules of diagnosis expert systems. The benefits of these frameworks are explained as knowledge acquisition methodology in section 3.

There are seven generic operations: SELECTION, CLASSIFICATION, ORDERING,

COMBINATION, TRANSLATION, INPUT and OUTPUT. Each operation has one or more source groups, destination groups and evaluator groups.

SELECTION Operation

The SELECTION operation picks up elements from the source group according to the selection conditions of the evaluators. Fig.4 shows an example of the SELECTION operation.

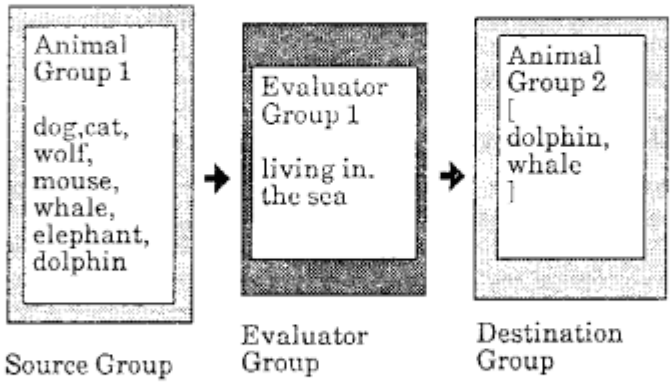

Fig.4 Example of SELECTION operation

CLASSIFICATION Operation

The CLASSIFICATION operation separates source elements into groups. The upper SELECTION operation is an example which is a special case of CLASSIFICATION. Fig.5 shows an example of the CLASSIFICATION operation.

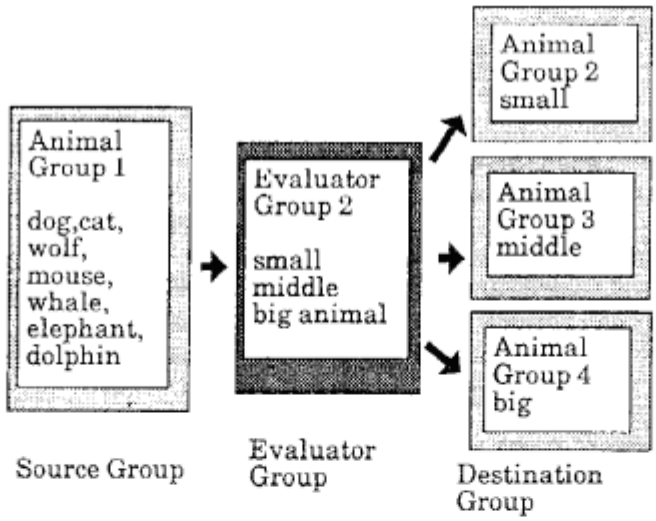

Fig.5.Example of CLASSIFICATION operation

ORDERING Operation

Elements of groups take turns to evaluate. The ORDERING operation sors elements of the source group and makes a destination group, in which elements take their turns in order of the ordering condition.

COMBINATION Operation

The COMBINATION operation assembles source elements into new elements. Generally, there are multiple sources, as shown in Fig.6.

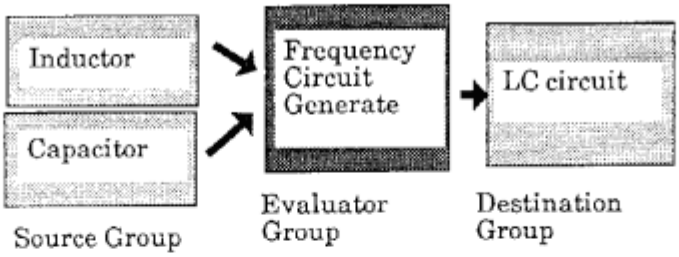

Fig.6.Example of COMBINATION operation

TRANSLATION Operation

The TRANSLATION operation is used for data abstraction and data interpretation. In this operation, elements of the source group are translated and the destination group is generated. New elements must be generated, or attributes of elements must be added or changed.

INPUT Operation and OUTPUT Operation

These are interaction operations between an expert system and a user (or a measuring device). The INPUT operation makes a new group without source groups. The OUTPUT operation does not make a destination group. It gives source group information to the user according to the output process.

2.2.2 Elements

Elements consist of items and their attributes. Elements are frame-based knowledge representations which have attribute-slots and an inheritance, but they do not have the attached procedures as demons. Each element must belong to one or more element groups. An element is accessed by one or more generic operations.

2.2.3 Evaluators

Evaluators consist of conditions and actions, and may include the procedural process. Sometimes, the evaluator group is a source group of a generic operation. For example, in a constraint relaxation problem of a planning task, a big evaluator group which has many conditions as constraints must be modified to a small evaluator group which has fewer conditions. This small evaluator group is used to solve the planning task.

2.2.4 Meta-Operation-Control: Meta-Script

The expert tasks have their scenario for solving problems, therefore, this meta-operation control is called Meta-Script in the Expert Model. It is a representation to describe meta-level knowledge. Fig.7 shows its location in the knowledge base.

The Expert Model is recognized as an object-oriented model, and a generic operation is an object. The communication between a generic operation and other operations is a message passing process, which has two types in the Expert Model: the inheritance and the post operation call.

3. Knowledge Acquisition Support System : EPSILON

3.1 EPSILON System Over View

The knowledge acquisition support system, EPSILON consists of three sub-systems. They are a knowledge elicitation system, an expert model inference engine and a knowledge refinement system.

The knowledge elicitation system interviews the expert directly, and extract knowledge in form of Expert Model. This sub-system uses pre-post method to acquire knowledge by interview.

The knowledge refinement system checks lacks and inconsistency of knowledge base.

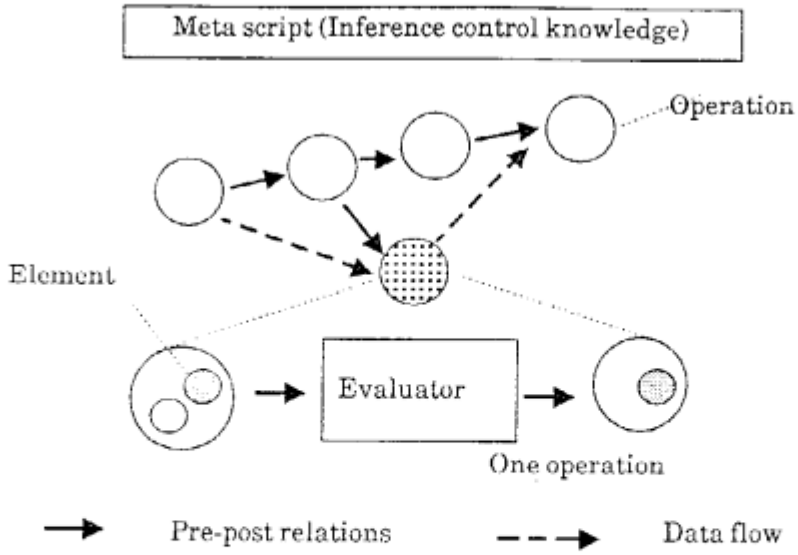

Fig.7 Hierarchy of the Expert Model

Extracted Expert Model is translated into logical knowledge representation and checked its inconsistency. Lacks of it is checked to compare Expert Model and other knowledge. For exmaple, if the Expert Model is knowledge for diagnosis of a machine, it is compared to the machine design information which represents its parts and the structure.

The expert model inference engine evaluates Expert Model. The expert can check his knowledge to observe behaviors of operations. EPSILON system structure is shown in Fig. 8.

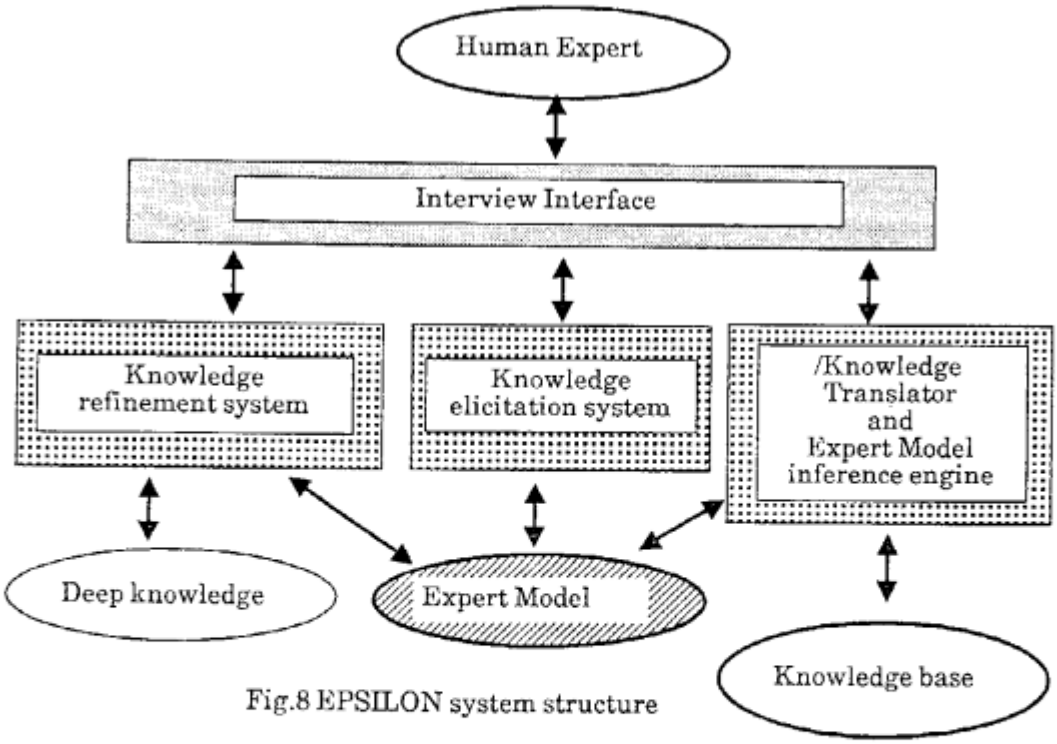

Fig.8 EPSILON system structure

3.2 Knowledge Acquisition Method : Pre-Post Method

The pre-post method is a problem solving model construction and model instantiation method. This method has a few model building strategies. The main strategy of this method stimulates a human expert to remember pre-post operations which are associated with a focal operation. The human expert can easily answer what operations are necessary before or after the operation. For example, when the car does not move, the human is asked *"What do you do before checking the engine ?"*. He can easily answer, *"I have to check the remaining gas" or "I have to check the battery"*. The following steps are knowledge acquisition steps based on the pre-post method.

Step 1: Collecting several human expert operations at random. It is not necessary for each operation to have any relation to the others. These operations are used as the starting points of knowledge acquisition.

Step 2: Making questions to obtain pre and post operations for each collected operation. This step continues until no new operations are extracted from the human expert. The same process is repeated for new operations.

Step 3: Many operations have been gathered and they have pre/post information which operations have to execute before or after operations. This step is the pre/post relation check step. One way of checking is to display pre/post relations graphically for the human expert.

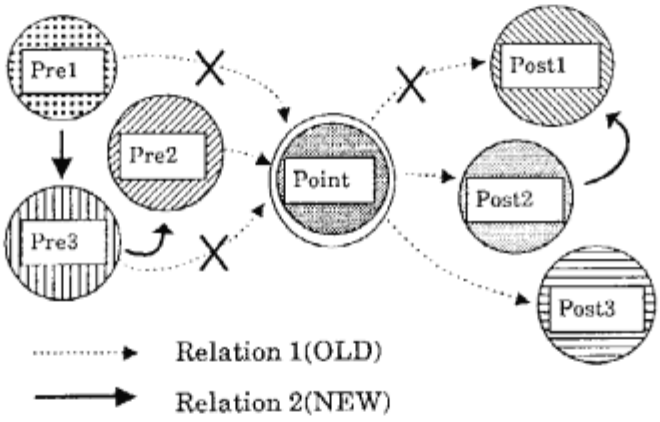

Fig.9 Pre-post relation example

Fig.9 shows pre-post relations. Relation1 shows only that Pre1 to Pre3 are pre-operations of a point operation and Post1 to Post3 are its post-operations. In relation2, there are new orders among pre-post operations defined by the expert. Pre1 is a pre-operation of Pre3. Pre2 is a post-operation of Pre3. Post1 is a post-operation of Post2. The more pre-post information collected, the clearer pre-post relations.

The sequence of some operations may be unclear. We treat the group of these operations as a weak sequence strategy group. When an operation of this group is finished, sometimes the result of the operation affects the other operations of this group. It is important for this group to obtain constraint information.

Step 4: This step is the instantiation step of each operation. The human expert must select a generic operation type to define the role of each operation. Sometimes, operations defined by the human expert are big tasks which contain several sub-operations. To distribute these big tasks, it is important to ask *"What sub-operations exist in XX (an operation name)? Please answer at random"*. After this question, pre-post operations must be extracted for sub-operations. Then, the Step 3 process is executed again.

Note: From step 1 to step 4, a problem solving model is made. From step 5, the details of operations in this model is extracted.

Step 5: This step merges operations. According to pre-post information and generic operation types, some operations are merged to one generic operation.

Step 6: Each operation has evaluators. This step defines these evaluators. It is not difficult to do this process, because the role of the operation is clear and its function has been identified by its generic operation type.

Step 7: The elements of the source group are defined in this step. Sometimes, it is difficult to decide the elements and group of a focal operation, because they are decided dynamically after its pre-operation has been executed. Therefore, the human expert must search for the candidates of the source group from the result (destination group) of its pre-operations.

Step 8: Attributes and attribute values of the elements are defined in this step. Each element must have the attribute evaluated by the evaluators. Generally, elements flow through a lot of operations, and need the attribute which is evaluated by the evaluators in the flow.

Other Steps: There are other steps in pre-post method. They are operation blocks making step to group operations with same inference control, and operation inference control definition step.

4. Operation Presumption

This operation presumption method makes the details of operation from input data, input element examples, output data, and output element examples of an operation inductively. This section explains the details of operation presumption for four operation types (selection, classification, translation and ordering). It is not necessary to make knowledge seeds to associate expert knowledge for the other operation types (input, output and combination).

4.1 Operation Presumption by Inductive Inference

Induction is an inference which makes more general rules or concepts from examples. It decides the generalization level of knowledge to limit a version space [Mitchell 78] from positive and negative examples. For example, in a selection operation, an input element group is a set of all examples. An output element group is a positive example set of this operation. Elements which are not selected by this operation are negative examples. An operation evaluator can be presumed from input and output elements. However, it is difficult to make a good generalization level of the operation evaluator from a few examples [Valiant 84]. Therefore, this system uses the results of induction for knowledge seeds. If the expert does not remember the details of an operation easily, this is a useful knowledge acquisition method.

4.2 Selection and Classification Criteria Presumption

To presume selection or classification criteria, attributes and criteria formulae (logical or mathematical formulae) must be decided. If attributes to be focused on are multiple, logical relations between criteria of these attributes must be decided. To explain the presumption examples, element groups (EGi: i-th element group) and elements (ELj: j-th element) are defined as follows.

An element group is shown as a set of elements.

$$EGi = \{ELj, ELj+1, ..., ELk\}$$

An element is shown as a list of pairs of an attribute (ATn: n-th attribute) and an attribute value (VAn: n-th attribute value).

$$ELj = (ATn = VAn) \wedge (ATn+1 = VAn+1) \wedge \ldots.$$

Then, a selection criterion includes pairs of an attribute and its value in positive examples, but does not include pairs of an attribute and its value in negative examples. We assume that a human expert uses simple and general criteria in preference to complex and special ones when making a selection. For this reason, the system begins with simple criterion generation. Candidates for selection criteria are generated as follows.

Step 1: Picking up attributes in all elements.

Step 2: Making two attribute value sets for each attribute in positive examples and negative examples. If an element has an attribute, the system regards the element which has a pair of the attribute and an attribute value as being "undefined". A value set made from positive examples is called a positive attribute value set (PVS) for an attribute. A value set made from negative examples is called a negative attribute value set (NVS) for an attribute.

Example 1: Positive examples:

$$EL1 = (AT1 = VA1) \wedge (AT2 = VA2)$$
$$EL2 = (AT1 = VA1) \wedge (AT3 = VA3)$$

Negative example:

$$EL3 = (AT2 = VA4) \wedge (AT3 = VA3)$$

PVS for AT1 = (VA1), NVS for AT1 = (undefined)
PVS for AT2 = (VA2 $\vee$ undefined), NVS for AT2 = (VA4)
PVS for AT3 = (undefined $\vee$ VA3), NVS for AT3 = (VA3)

Step 3: Searching attributes for which an intersection of PVS and NVS is an empty set. If there are attributes which satisfay this condtion, the system makes some candidates for selection criteria and shows them to the human expert as **Step 6.** In example 1, attributes AT1 and AT2 satisfy this condition, but attribute AT3 dose not. If there are no attributes which satisfy this condition, the system processes **Step 4.**

Step 4: Making two attribute value sets for multiple attributes in positive and negative examples. Elements of these sets are conjunctions which consist of a pair of an attribute and its value. A set made from positive examples is called a multiple positive attribute values set (MPVS). A set made from negative examples is called a multiple negative attribute values set (MNVS). Combinations of attributes begin with attributes which have the lowest number of "undefined" in PVS and NVS.

[Example 2] Positive example:

$$EL4 = (AT5 = VA5) \wedge (AT7 = VA7)$$
$$EL5 = (AT5 = VA5) \wedge (AT6 = VA6) \wedge (AT8 = VA8)$$

Negative example:

$$EL6 = (AT5 = VA5) \wedge (AT6 = VA9)$$
$$EL7 = (AT6 = VA6) \wedge (AT7 = VA10)$$

PVS for AT5 = (VA5), NVS for AT5 = (VA5 $\vee$ undefined)
PVS for AT6 = (undefined $\vee$ VA6), NVS for AT6 = (VA9 $\vee$ VA6)
PVS for AT7 = (VA7 $\vee$ undefined), NVS for AT7 = (undefined $\vee$ VA10)
PVS for AT8 = (undefined $\vee$ VA8), NVS for AT8 = (undefined)
AT5 and AT6 have one undefined values.
AT7 and AT8 have two undefined values.

Order of Combinations: AT5 ∧ AT6, AT5 ∧ AT7, AT6 ∧ AT7, AT5 ∧ AT8,
AT6 ∧ AT8, AT8 ∧ AT7, AT5 ∧ AT6 ∧ AT7, ...

MPVS for AT5 and AT6 = (AT5 = VA5)∧(AT6 = undefined))∨((AT5 = VA5)∧(AT6 = VA6)
MNVS for AT5 and AT6 = (AT5 = VA5)∧(AT6 = VA9))∨((AT5 = undefined)∧(AT6 = VA6)

Step 5: Checking whether an intersection of an MPVS and an MNVS is an empty set. If the MPVS and MNVS pair satisfies the condition, the system makes some candidates for selection criteria and show them to the human expert as **Step 7**. If the pair dosen't satisfy the condition, the system checks another pair.

Step 6: Generating three candidates from PVS and NVS for selection criteria. They are an expression of an attribute and its PVS, a negative expression of an attribute and its NVS, and a conjunction of these two expressions.

In example 1, they are shown as follows.

[Attribute AT1] Type 1: AT1 = VA1
Type 2: AT1 ≠ undefined
Type 3: (AT1 = VA1)∧(AT1 ≠ undefined)

[Attribute AT2] Type 1: AT2 = VA2 ∨ undefined
Type 2: AT2 ≠ VA4
Type 3: (AT2 = VA2 ∨ undefined)∧(AT2 ≠ VA4)

The system shows type 1 expression first. If the human expert requires another expression, the system shows type 2 expression second and type 3 expression third. If the human expert finds new selection criteria because of these hints, the expert modifies them directly. If the expert rejects the expression of this form expression, the system return to **Step 4** to make other candidates.

Step 7: Generating three candidates from MPVS and MNVS for selection criteria. They are an expression of an attribute and its MPVS, a negative expression of an attribute and its MNVS, and a conjunction of these two expressions.

In example 2, they are shown as follows.

[Attribute AT5 and AT6]
Type 1: ((AT5 = VA5)∧(AT6 = undefined))∨((AT5 = VA5)∧(AT6 = VA6))
That is: (AT5 = VA5)∧(AT6 = undefined ∨ VA6)
Type 2: ((AT5 ≠ VA5)∨(AT6 ≠ VA9))∧((AT5 ≠ undefined)∨(AT6 ≠ VA6))
Type 3: (AT5 = VA5)∧(AT6 = undefined ∨ VA6)∧
((AT5 ≠ VA5)∨(AT6 ≠ VA9))∧((AT5 ≠ undefined)∨(AT6 ≠ VA6))

The system shows these candidates as in **Step 6**. If the human expert rejects the expression of this form, the system return to **Step 4** to make other candidates.

This method is based on logical relation. So the current version of the selection operation presumption algorithm doesn't make an expression "AT1 >= 10" from an original expression "AT1 = 10, 20, 100, ...".

Example 3: *Animal selection operation Input and output element groups are defined as follows.*

Input EG = {lion, whale, penguin, salmon, dolphin, elephant}
Output EG = {whale, dolphin, salmon}

Elements and their contents are defined as follows.

lion = (habitat = plain)∧(legs = 4)∧(genus = mammalia)
whale = (habitat = sea)∧(has = fins)∧(genus = mammalia)
penguin = (habitat = seaside)∧(legs = 2)∧(has = wings)∧(genus = birds)
salmon = (habitat = sea)∧(has = fins)∧(genus = fish)
dolphin = (habitat = sea)∧(has = fins)∧(genus = mammalia)
elephant = (habitat = plain)∧(legs = 4)∧(genus = mammalia)

A normal logical selection criterion is derived from these examples.
It is :

(((habitat = sea) ∧ (has = fins) ∧ (genus = mammalia)) ∨
((habitat = sea) ∧ (has = fins) ∧ (genus = fishes)) ∨
((habitat = sea) ∧ (has = fins) ∧ (genus = mammalia))) ∧
¬(((habitat = plain) ∧ (legs = 4) ∧ (genus = mammalia)) ∧
((habitat = seaside) ∧ (legs = 2) ∧ (has = wings)) ∧ (genus = birds)) ∧
((habitat = plain) ∧ (legs = 4) ∧ (genus = mammalia))

An arranged criterion is :

(habitat = sea) ∧ (has = fins)∧ (genus = mammalia ∨ fish)∧
((habitat ≠ plain)∨(legs ≠ 4)∨(genus ≠ mammalia))∧
((habitat ≠ seaside)∨(legs ≠ 2)∨(has ≠ wings)∨(genus ≠ birds))

This logical formula is a candidate knowledge seed of this operation. However, the expert focuses on one or a few attributes to select or classify elements. This formula is most specialized criterion. So the system tries to propose more general ones.
In example 3, PVS and NVS are shown as follows.

PVS for habitat = (sea), NVS for habitat = (plain ∨ seaside)
PVS for legs = (undefined), NVS for legs = (2 ∨ 4)
PVS for has = (fins), NVS for has = (undefined ∨ wings)
PVS for genus = (mammalia ∨ fish)
NVS for genus = (mammalia ∨ birds)

Pairs of PVS and NVS for attributes "habitat", "legs" and "has" have no intersections. The pair of PVS and NVS for "habitat" has no undefined value, so the system makes following candidates.

(1) habitat = sea
(2) habitat ≠ plain and seaside
(3) (habitat = sea)∧(habitat ≠ plain ∧ seaside)

If the expert didn't use these criteria, the system generates the following candidates according to our method.

(4) legs = undefined
(5) legs ≠ 2 and 4
(6) (legs = undefined) ∧ (legs ≠ 2 ∧ 4)
:
(i) (habitat = sea)∧(legs = undefined)
(i+1) ((habitat ≠ plain) ∨ (legs ≠ 4))∧((habitat ≠ seaside)∨
(legs ≠ 2))
(i+2) ((habitat = sea)∧(legs = undefined))∧
((habitat ≠ plain) ∨ (legs ≠ 4))∧((habitat ≠ seaside)∨
(legs ≠ 2))
:

4.3 Translation Equation Presumption

A translation operation makes a new pair of an attribute and its value to refer to one or

more attribute values. In the following examples, referred attributes are assumed in the same elements. There are two kinds of translation operations: translating one symbol into another, and translating one number into another using numerical calculation.

Example 4: *Make a new attribute and its value*

(Make an area of a box)

Input EG = {box}

Output EG = {box}

box in input EG = (height = 20)∧(length = 5)

box in output EG = (height = 20)∧(length = 5)∧(area = 100)

If a human is given example 4, he compares the input element group and the output element group, and quickly makes "area = height * length", because he has knowledge about area. If he is given an example that makes (c = 100) from (a = 20) and (b = 5), he also makes "c = a * b" easily. There are many candidates which satisfy this relation.

area = height * length
area = height * 5
area = height * 6 - length * 4
area = length * 20
:

If the element "box" has (weight = 100), the human makes "area = weight". There are too many induction results to narrow down. Before knowledge acquisition, it is difficult for knowledge acquisition systems to know the relations between attributes (for example, relations among area, length and height). A set of attributes in an input element group related to the generated new attribute is first determined for presumption. The intersection of attributes of input elements must be shown for the expert. The expert selects focused attributes from this set. However, sometimes, the expert checks different attributes for each input element. Then, other attributes are also shown. In this example, after the expert selects attributes "height" and "length", he forms relations between these attributes. After attributes are selected, there are many translation formula candidates.

If the expert can define the form of a translation formula, this presumption is a problem which decides the formula parameters. If enough examples are given, this system solves simultaneous equations. The problem of deciding on a translation formula is difficult. Therefore, the current version of the presumption algorithm supports only the selection of possible candidates of attributes by forming on intersection of attribute sets of input elements when multiple examples are given.

Next, we discuss the problem of presuming symbol exchange. Focused attributes are also important in this case. In output elements, the system must focus on new attributes or attributes which have new values in output elements. There is a strong possibility that an attribute which has a new value is a candidate. This heuristic is useful for selecting attributes in this operation type.

Example 5(a): *Change attribute value*

Input EG = {box 1, box 2, box 3}

Output EG = {box 1, box 2, box 3}

box 1 in input EG = (area = 50)∧(place = on-table)
box 2 in input EG = (area = 60)∧(place = on-floor)
box 3 in input EG = (area = 100)∧(place = in-garden)

box 1 in output EG = (area = small)∧(place = on-table)
box 2 in output EG = (area = small)∧(place = on-floor)
box 3 in output EG = (area = large)∧(place = in-garden)

Example 5(a) shows a translation operation focusing on the attribute "area". The following logical formulae are generated easily in example 5.

(area = 50)$\vee$(area = 60) --> (area = small)
(area = 100) --> (area = large)

These formulae handle numbers for translation criteria. Using a heuristic in which one or more numbers select the translation criterion, these formulae can be generalized. The comparison result is :

(area = 50) < (area = 60) < (area = 100)

(area = 50) and (area = 60) are translated into the same value (area = small), which means that the necessary criterion is between 60 and 100. Then, EPSILON proposes the following candidates.

(area <= 80) --> (area = small)
(area > 80) --> (area = large)

The expert checks these translation formulae forms the following, for example.

(area <= 90) --> (area = small)
(area > 90) --> (area = large)

Example 5(b): *Change attribute value*

Input EG = {box 1, box 2, box 3}
Output EG = {box 1, box 2, box 3}

box 1 in input EG = (area = 50)$\wedge$(place = on-table)
box 2 in input EG = (area = 60)$\wedge$(place = on-floor)
box 3 in input EG = (area = 100)$\wedge$(place = in-garden)

box 1 in output EG = (area = 50)$\wedge$(place = in-house)
box 2 in output EG = (area = 60)$\wedge$(place = in-house)
box 3 in output EG = (area = 100)$\wedge$(place = out-of-house)

Example 5(b) handles symbol translation focusing on the attribute "place". The following logical translation formulae are generated, but cannot be generalized.

(place = on-table) or (place = on-floor) --> (place = in-house)
(place = in-garden) --> (place = out-of-house)

4.4 Ordering Criteria Presumption

There is a process order, which represents a list of elements, in each element group. An inference engine of EPSILON has two modes for problem solving: all solution search and partial solution search. The inference engine uses this order for partial solution search. The ordering operation sorts elements according to the mathematical or symbol order, for example, [short, middle, long]. The mathematical order may be ascending or descending.

Example 6: *Sorting using symbolic ordering*

Input EG = {box 1, box 2, box 3}
Output EG = {box 2, box 3, box 1}

box 1 = (size = small)
box 2 = (size = middle)
box 3 = (size = large)

The order of symbols of example 4 is the order list of values of an attribute (size).
Obtained criteria: size = [small, middle, large]

Example 7: Sorting using numerical ordering
Input EG = {box 1, box 2, box 3}
Output EG = {box 2, box 3, box 1}

box1 = (size = 150)
box2 = (size = 120)
box3 = (size = 130)

The order of example 7 is made from numerical comparison of an attribute (size).
Obtained criteria: Ascending order for size

If focused attributes are multiple, then a translation operation must be defined before this ordering operation to translate values of multiple attributes into one value of one attribute.

4.5 Knowledge Acquisition using Operation Presumption

There are many learning algorithms, but most of them are for special knowledge. It is therefore difficult to extract all knowledge of expert systems using only the learning algorithms which have been developed to date. However, these methods are useful for making knowledge seeds of interactive knowledge acquisition systems. It is important for us to apply learning methods to knowledge acquisition. In our method, if the induction result class is too wide (general), we must search for presumption heuristics. It is also important to ask good questions to limit the class of the criteria [Angluin 86]. Induction heristics are related to non-monotonic reasoning [Nicolas 89].

5. Summary

This paper introduced the knowledge acquisition support system, EPSILON, and a presumption method to make knowledge seeds. We proposed Expert Model as knowledge representation for knowledge acquisition, used in EPSILON. Expert Model consists of operations, so it is extracted in operation form. EPSILON supports an interview method and a presumption method to extract operation details. This operation presumption is a syntactical function, but we must support heuristics for effective presumptions. The currently implemented EPSILON is EPSILON/One, which supports the non-hierarchical Expert Model and pre-post method. EPSILON/One is implemented on a sequential inference machine (PSI-2), developed by ICOT, in an object oriented logic programming language ESP[Chikayama 84].

6. Acknowledgment

We wish to express our thanks to the member of 5th Laboratory at ICOT, Mr. Kazuhiro Tsubaki and Mr. Eiichi Horiuchi, the chief of 5th Laboratory, Mr. Kenji Ikoma, the members of the Knowledge base System Architecture - Knowledge Acquisition and Design - Sub Working Group, its head, Professor Riichiro Mizoguchi and Professor Norihiro Abe. We would like to thank the members of the Learning and Non-monotonic Reasoning System Research Group (LANRS) at ICOT and Dr. Koichi Furukawa. Finally, we must express our thanks to Dr. Kazuhiro Fuchi, Director of ICOT Research Center, who provided us with the opportunity to conduct this research in the Fifth Generation Computer Systems Project.

References

[Angluin 86] Angluin, D. :Types of queries for Concept Learning, TALEU/DCS/TR-479, June 1986

[Bennett 85] Bennett, J. : ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System, Journal of Automated Reasoning 1, 49-74, 1985

[Boose 84] Boose, J. : Personal Construct Theory and the Transfer of Human Expertise, in Proceedings of the National Conference on Artificial Intelligence, Austin, Texas, 1984

[Chandrasekaran 86] Chandrasekaran, B. : Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design, IEEE Expert, Fall 1986

[Chikayama 84] Chikayama, T. : Unique Features of ESP, International Conference on Fifth Generation Computer Systems, November 1984

[Clancey 85] Clancey, W. : Heuristic Classification, Artificial Intelligence 27, 1985

[Genesereth 86] Genesereth, M. , Nilsson, N.: Logical Foundations of Artificial Intelligence Los Altos, CA: Morgan-Kaufmann, 1986

[Hays-Roth 83] Hays-Roth, F. , Waterman, D. A. , Lenat,D. B. : Building Expert Systems, Addison-Wesley, 1983

[Iwashita 86] Iwashita, Y. , et al. : Knowledge Acquisition and Learning in Case Studies of Expert System Development, ICOT TR-0204, 1986

[Kahn 85] Kahn, G. , Nowlan, S. and McDermott, J. : Strategies for Knowledge Acquisition, IEEE transactions on Pattern Analysis and Machine Intelligence 7(5), 1985

[Nicolas 89] Nicolas, Helft, Induction as Nonmonotonic Inference, KR89,1989

[Miki 87] Miki, M. and Taki, H. : The Knowledge Acquisition System with Fault Tree and Diagnostic Flow Interface, Proceedings of International Workshop on Industrial Applications of Machine Vision and Machine Intelligence, Tokyo, 1987

[Mitchell 78] Mitchell, T.M. : Version Spaces: An Approach to Concept Learning, Stanford Technical Report STAN-CS-78-711, HPP-79-2

[Taki 87] Taki, H., Tsubaki, K. and Iwashita, Y. : EXPERT MODEL for Knowledge Acquisition, the Proceedings of IEEE Expert Systems in Goverment Conference, 1987

[Taki 88] Taki, H. : Knowledge Acquisition by Observation, Proceedings of Ineternatinal conference of Fifth Generation Computer Systems, 1988

[Valiant 84] Valiant, L.G. : A Theory of the Learnable, CACM, Vol. 27, No. 11, pp1134-1142, 1984