An Augmented EBL and its Application to the Utility Problem

Masayuki YAMAMURA, Shigenobu KOBAYASHI
my@sys.titech.ac.jp kobayasi@sys.titech.ac.jp
Dept. Intelligence Science,
Graduate School of Science and Engineering,
Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku Yokohama, 227 Japan.

Abstract

EBL can learn justified generalizations from only one example when the domain theory is perfect. However, it does not work when the domain theory is imperfect. Imperfectness of the domain theory can be classified into four levels, i.e. incomplete, intractable, inconsistent and non-operational ones. It is necessary to unify EBL and SBL to solve these problems.
In this paper, we propose a framework of an augmented EBL to handle plural examples simultaneously. We formalize it on logic program, and introduce a concept of least EBG to extract similarities from plural examples. We discuss on an approach to solve utility problem with the augmented EBL. Utility problem is a problem to learn more efficient description under complete, tractable, consistent but non-operational domain theories.
We define operationality criteria with maximizing usage degree and minimizing backtracking number, and show they increase partial monotonically by generalization. Since this partial monotinicity is not preferable to search operational generalizations, least EBGs are more operational than usual EBGs. We design a simple incremental learner based on least EBGs, and show its usefulness in recursive domain theories. We also discuss on other imperfect theory problems.

1 Introduction

EBL (Explanation-Based Learning) is a framework of deductive learning [Mitchell *et al.*, 1986]. Given an example of the goal concept, it builds an explanation, and learns a concept description as its generalization. It can learn justified generalizations from only one example when the domain theory is perfect, but it does not work when the domain theory is imperfect. Several types of imperfectness have been pointed out, and they are called imperfect theory problems.

SBL (Similarity-Based Learning) is a framework of inductive learning [Mitchell, 1982]. Given positive and negative examples, it extracts similarities, and learns a concept description as their generalization. It can learn something without any domain theories, but it requires numerous examples, and learned descriptions are no more justified.

EBL and SBL have complemental features, and can be placed at both extremes in continuum of imperfectness. In real applications, we cannot expect perfect theories nor fully imperfect ones. Therefore, it is necessary to unify EBL and SBL to resolve their difficulties.

This paper attempts to solve imperfect theory problems of EBL by augmenting it to handle plural examples simultaneously [Yamamura *et al.*, 1989a] [Yamamura, 1990a]. We augment EBL rather than SBL because SBL has no explicit domain theories and we want to avoid implicit ones such as inductive bias.

Mitchell et. al. have pointed out three types of imperfectness i.e. incomplete, intractable and inconsistent theories [Mitchell *et al.*, 1986]. We add non-operational one into them and look them as levels of imperfectness. This seems useful as an index of a class of learning. Upper part of figure 1 shows dependencies of these levels. SBL is placed at the most imperfect extreme. Tractability requires completeness since an incomplete theory is useless if it ignores "difficulties" of that domain. Consistency requires tractability since explanations of consistency may not be build if it is intractable. Utility problem of conventional EBLs is placed below them.

There are many works on unifying EBL and SBL. Their domain theory can be placed in some level of imperfectness. For example, IOE (Induction On Explanation) of [Flann *et al.*, 1989] specializes a domain theory which is complete, tractable but inconsistent. Chunking macros for such as 8 puzzle from primitive operators which is complete but intractable [Laird *et al.*, 1986]. [Bergadano *et al.*, 1988] use SBL when an explanation can not be constructed from domain theory which is incomplete.

Lower part of figure 1 summarizes our works on the augmented EBL. In this paper, we forcus on utility problem, then also discuss on other problems.

2 Augmentated EBL

2.1 Generalisation of Explanation Structures

In this paper, we assume logic program as a description language. A domain theory *D* is a finite set of defi-

	Domain Theory				
Problem Dependencies	*Complete*				*Incomplete*
	\ *Tractable*			*Intractable*	
	Consistent		*Inconsistent*		
	Operational	*Nonoperational*			
Unified Approaches	—	Description Operationalization with Least EBG of Plural Examples.	Consistency Maintainance with EBG Version Space.	Knowledge Compilation with Examples and Meta Theory.	Knowledge Transfer with Analogical Reasoning.
	(Perfect)	(Conventional EBL)	(Imperfect Theory Problems)		

Figure 1: imperfect thoery problems and augmented EBL approaches.

$$D = \left\{ \begin{array}{l} \mathbf{kill}(X,Y) \leftarrow \mathbf{hate}(X,Y) \wedge \mathbf{possess}(X,Z) \\ \qquad\qquad\qquad \wedge\, \mathbf{weapon}(Z), \\ \mathbf{hate}(W,W) \leftarrow \mathbf{depressed}(W), \\ \mathbf{possess}(U,V) \leftarrow \mathbf{buy}(U,V), \\ \mathbf{weapon}(\mathbf{gun}), \\ \mathbf{weapon}(\mathbf{knife}) \end{array} \right.$$

$$\begin{array}{ll} E_1 = & \{\mathbf{depressed}(\mathbf{john}),\ \mathbf{buy}(\mathbf{john},\mathbf{gun})\}, \\ G_1 = & \leftarrow \mathbf{kill}(\mathbf{john},\mathbf{john}). \\ & \\ E_2 = & \{\mathbf{depressed}(\mathbf{tom}),\ \mathbf{buy}(\mathbf{tom},\mathbf{knife})\}, \\ G_2 = & \leftarrow \mathbf{kill}(\mathbf{tom},\mathbf{tom}). \end{array}$$

Figure 2: a domain theory and examples.

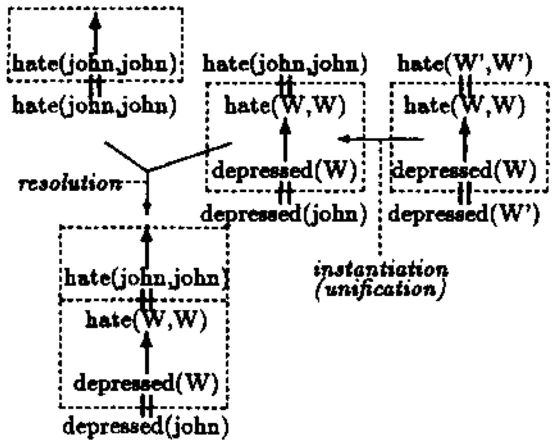

Figure 3: an inference with explanation structures.

nite clauses. An example E_i, is a finite set of ground unit clauses. A goal concept is a predicate, and is given as a ground negative clause $\leftarrow G$ for each examples. A goal concept is explained or proved iff $D \cup E_i \cup \{\leftarrow G\} \vdash \Box$.

In the goal regression method [Mitchell *et al,* 1986], a conclusion of an EBG (Explanation-Based Generalization) comes from the goal concept, and preconditions come from the facts. Constants are generalized into variables, and resolutions are undone. The following definition generalizes these operations.

Definition 1 (generalization of an explanation)
1. For a Horn clause $C = A \leftarrow B_1 \wedge \cdots \wedge B_n$, let its equivalent copy be $C' = A' \leftarrow B_1' \wedge \cdots \wedge B_n'$. Then, following expression is an explanation structure.

$$A' : (A \leftarrow (B_1 : B_1') \wedge \cdots \wedge (B_n : B_n')).$$

C and *C* are called its internal and external structure respectively.
2. For an explanation structure T, let C' be its external structure and *a* be a substitution. Then, the expression 7V, which has the same internal structure as *T* and has the external structure $C'\sigma$, is called an instantiation of *T.* Inversely, *T* is called an uninstantiation of $T\sigma$.
3. For two explanation structures 5 and T, let $A' \leftarrow B_1' \wedge \cdots \wedge B_i' \wedge \cdots \wedge B_n'$ and $B_i' \leftarrow C_1' \wedge \cdots \wedge C_m'$ be their external structures respectively. Then, the expression *U* connected 5 and *T* at B_i' is called a (binary) resolvent of 5 and *T.* Inversely, *S* is called an unresolution of U, and *T* is called its remainder.
4. An explanation structure *S* is more general than T, denoted by $S \preceq T$, iff there exists a sequence of uninstantiations and unresolutions from *T* to *S.*

For example, consider a domain theory *D* shown in figure 2. In SLD resolution, a negative clause and a definite clause are first unified then binarily resolved upon some literal. Figure 3 shows this inference scheme, where thick arrows denote rules and double lines denote ":" of definition 1. Dotted boxes denote internal structures of explanation structures. Internal structures holds the history of clause applications, and external structures holds the actual instantiation.

In conventional EBL on logic programming [Hirsh, 1987]lKedar-Cabelli *et* al., 1987]lPrieditis *et* al, 1987], a generalization is defined as an inverse of one inference, i.e a pair of an unresolution and an uninstantiation. We consider them separately in order to handle generalization of compound terms (or structured objects) because instantiations of heads of clauses affects inference efficiency under a pure-Prolog interpretor. Figure 4 shows a full explanation structure for *E\.*

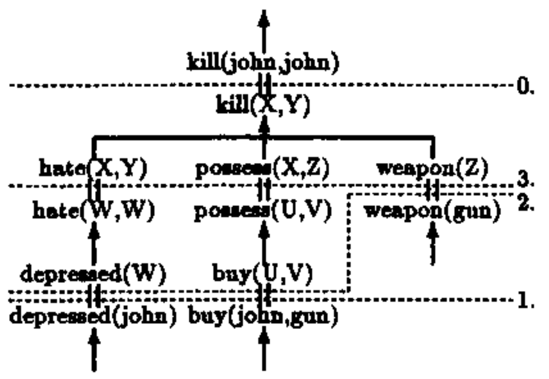

Figure 4: the explanation structure for E_1.

$$
\begin{aligned}
m_1 &= \text{kill(john, john)}.\\
m_2 &= \text{kill(john, john)} \leftarrow \text{depressed(john)} \land \text{buy(john, gun)}.\\
m_3 &= \text{kill}(W, W) \leftarrow \text{depressed}(W) \land \text{buy}(W, \text{gun}).\\
m_4 &= \text{kill(john, john)} \leftarrow \text{depressed(john)} \land \text{buy(john, gun)}\\
&\qquad \land \text{weapon(gun)}.\\
m_5 &= \text{kill}(W, W) \leftarrow \text{depressed}(W) \land \text{buy}(W, Z) \land \text{weapon}(Z).\\
m_6 &= \text{kill}(X, Y) \leftarrow \text{hate}(X, Y) \land \text{possess}(X, Z) \land \text{weapon}(Z).
\end{aligned}
$$

Figure 5: EBGs from E_1.

An EBG learned with an EBL system must be a theorem of that domain theory. For this constraint, we introduce a concept of an obliged generalization.

Definition 2 (EBG)

1. An explanation T is an obliged generalization, iff the internal structure of T does not include any facts of a certain example and its external structure is a definite clause.
2. Obliged generalizations of an explanation for some example are called EBG. For an EBG, its external structure is called an EBG macro.

Figure 5 shows some of EBG macros of the explanation of figure 4. m_1 to m_6 are all unresolved at line 0 to be definite clauses, mi is less general than the obliged generalization i.e it includes facts of E_1, then it is not valid in other examples, m_2 is the obliged generalization (additionally unresolved at line 1). m_3 is the most uninstantiation of m_2. m_4 is more unresolved than m_2 (line 2). m_5 is the most uninstantiation of m_4. m_6 is generalized until the domain theory (line 3). Remark there are a number of generalizations from only one explanation.

2.2 Generalization Space

Since various generalizations are obtained from one explanation, various sets of various generalizations are obtained from plural examples. We call such a class of all sets of generalizations a generalization space.

Definition 3 (generalization space)

1. For an explanation $T \in T$, let T_0 be an uninstantiation of T. Then $T' = (T - \{T\}) \cup \{T_0\}$ is an uninstantiation of T. Similarly, let T_0 be an unresolution and $T_1, \cdots, T_n$ are its remainders. Then $T' = (T - \{T\}) \cup \{T_0, \cdots, T_n\}$ is an unresolution of T.
2. An explanation set S is more general than T, denoted by $S < T$, iff there exists a sequence of uninstantiations and unresolutions from T to 5.
3. A class of all generalization of the obliged generalization of an example set is called its generalization space.

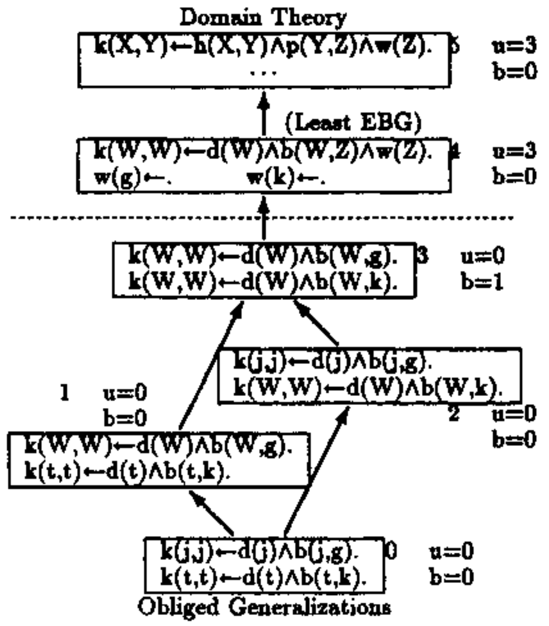

Figure 6: the generalization space on $\{E_1 \quad E_2\}$

For example, E_2 of figure 2 is the same as E_1 except the hero and the instrument. Figure 6 shows a part of the generalization space generated from $\{E_1, E_2\}$. Predicates and constants are simplified in their initial. 0 is the obliged generalization that is only unresolved at the goal and facts of examples. 1 and 2 uninstantiate "John" and "tom" respectively. 3 uninstantiates both. 4 unresolves the weapon, and the same structure is extracted in there, "weapon(knife)" and "weapon(gun)" in 4 are the remainder of that unresolution. 5 is the most general, i.e the domain theory.

Thus, a generalization space contains all candidates that are learnable from given examples. Learning problem in the augmented EBL is to search a good generalization with some criteria in a generalization space.

2.3 Least EBG

A common generalization such as 4 of figure 6 can be thought of as "simirality" in SBL. We call the most special common generalization the least EBG. It is known that least EBG is unique and computable incrementally regardless of example orderings.

Definition 4 (least EBG)

For EBGs $T_1, \cdots, T_n$ T is a common EBG iff $T \preceq T_1, \cdots, T \preceq T_n$. The least generalization in common EBGs is called the least EBG.

Theorem 1 (features of the least EBG)
1. There is the least EBG for any set of EBGs.
2. Let T be the least EBG of $T_1 \cdots, T_k$ and V be the least EBG of $T_{k+1},\cdots,T_n$. Then, the least EBG of T and T is the least EBG of $T_1,\cdots,T_n$.

Proofs are given in [Yamamura, 1990a]. To prove the uniqueness, we used a lemma that the most general unifier and the least uninstantiation commutes each other. If there are another minimal common generalization, they are equivalent by this lemma. Computability is a corollary of uniqueness. Thus, the uniqueness of the least EBG closely relates with the uniqueness of the most general unifier. m_5 of figure 5 is a least EBG.

From a viewpoint of extracting similarities, a generalization subspace that consists of least EBGs is interesting. We call such a subspace a least EBG space.

Definition 5 (least EBG space)
1. For $\mathcal{S} \subseteq \mathcal{T}$, let T_0 be the least EBG of S and $T_1,\cdots,T_n$ be its remainder. Then $\mathcal{T}' = (\mathcal{T} - \mathcal{S}) \cup \{T_0,\cdots,T_n\}$ is called a least EBG generalization of T.
2. S is more general by least EBG than T iff there exists a sequence of least EBG generalizations from T to 5.
3. A class of all least EBG generalizations of the obliged generalization of an example set is called its least EBG space.

0 and 4 of figure 6 are a least EBG generalization. In general, there are combinatorially many least EBG generalizations for more than three examples, and its size can much more increase if remainders are generalizable. However, the searching space is reduced from the raw generalization space.

3 An Approach to the Utility Problem

3.1 Operationally Criteria

Utility problem is a problem to learn more efficient program under complete, tractable, consistent but non-operational domain theories. A learner needs to improve performance without violating initial perfectness of the domain theory. We assume a pure-Prolog interpretor and learned EBGs are merely inserted before the domain theory.

Efficiency of EBGs are maesured with operationally criteria. There are several alternative levels to define operationally [Keller, 1988]. In many existing EBL, the most detailed level of CPU time is used. Here, we define them in more abstract level with maximizing usage degrees and minimizing backtrack numbers. They reflect on natural requirements to be an efficient program for pure-Prolog.

Definition 6 (operationally criteria)
1. For a generalization $\mathcal{S} = (\mathcal{T} - \{T\}) \cup \{T_0,\cdots,T_n\}$, its usage degree is a function s.t.

$$\mathcal{U}(\mathcal{S},\mathcal{T}) = \sum_{i=0}^{n} \begin{cases} |T_i| & \text{if } T_i \in \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

where $|T_i|$ denotes the number of clauses used in T_i In general, when $\mathcal{S} \preceq \mathcal{T}$, the usage degree sums up all usage degrees through some generalization path.

2. For an example set $\mathcal{E}$ and its generalization T, the backtrack number $\mathcal{B}(\mathcal{T},\mathcal{E})$ is the minimum of the sum of goal failures which occur during a macrotable of T reconstructs all explanations of $\mathcal{E}$.
3. For two generalizations S and T of an example set $\mathcal{E}$, S is more operational than T iff $\mathcal{U}(\mathcal{S},\mathcal{E}) \geq \mathcal{U}(\mathcal{T},\mathcal{E})$ and $\mathcal{B}(\mathcal{S},\mathcal{E}) \leq \mathcal{B}(\mathcal{T},\mathcal{E})$.

Usage degree measures an applicablity or a storage cost. It increases when one macro becomes commonly used in more than two examples. Backtrack number measures an efficiency or a searching cost. It counts backtrackings when a problem solver re-construct explanations for examples. These two measures show operationality/generality trade-off of [Segre, 1987].

In Figure 6, u denotes a usage degree and b denotes a backtracking number. The least EBG (4) is the most operational macrotable because its usage degree is the maximum and its backtracking number is the minimum.

Two measures of operationally criteria increases according to generalization as shown in figure 6 but it is not monotonic. Following fact holds in general.

Theorem 2 (partial monotonicity)
For $\mathcal{S} \preceq \mathcal{T} \preceq \mathcal{E}, \mathcal{U}(\mathcal{S},\mathcal{E}) \geq \mathcal{U}(\mathcal{T},\mathcal{E})$.
And if $\mathcal{U}(\mathcal{S},\mathcal{E}) = \mathcal{U}(\mathcal{T},\mathcal{E})$, **then** $\mathcal{B}(\mathcal{S},\mathcal{E}) \geq \mathcal{B}(\mathcal{T},\mathcal{E})$

Proof was given in [Yamamura, 1990a]. The total monotonicity of usage degree is clear. To show the partial monotonicity of backtracking number, we can construct a macrotable of T that has less or equal backtracking number by arranging a macrotable of S.

The reason why backtrack number incleases partial monotonically is for ambiguous macros that have the same head but different bodies as shown in figure 6. Such ambiguous macros increase according to generalization. However, when they are generalized until the same structure, i.e. when the usage degree increases, this ambiguity is often reduced.

The partial monotonicity is not preferable feature to search operational EBGs efficiently. However, since they increase monotonically in subspaces of the same usage degree, their minimal generalizations seems to be useful. The least EBG is such a generalization. Figure 7 shows tipical generalization space.

Theorem 3 (operationally of the least EBG)
A least EBG generalization is minimal in the subspace which consists of generalizatins of the same usage degree.

Proof is given in [Yamamura, 1990a]. To prove this, we have shown any generalization between the obliged generalization and a least EBG generalization have less usage degree. This theorem shows the utility of the least EBG.

3.2 Least EBG Learner

We have constructed a simple incremental learner to demonstrate the utility of least EBG. Figure 8 shows its top-level actions. (-) and (+) denotes input and output variables of predicates respectively. Examples are given incrementally through example/2. Explanations are generated with solve/3 by learner itself. In that time, cur-

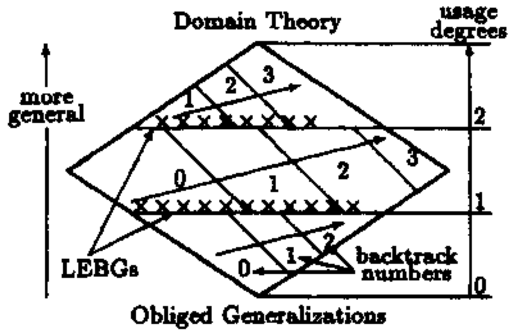

Figure 7: a typical generalization space.

```
learner(CurMT(−)) :-
        example(Goal(−), Facts(−)),
        solve(Goal(+), Facts(+), CurMT(+), Expl(−)),
        lebg_spece(CurMT(+), Expl(+), GenSp(−)),
        operational(GenSp(+), NxtMT(−)),
        !, learner(NxtMT(−)).
```

Figure 8: the top-level of the learner.

rent macrotable *CurMT* is inserted before the domain theory and used.

lebg-space/3 generates whole LEBG space *GenSp* from $CurMT \cup \{Expl\}$. operational/2 select the most operational EBG from *GenSp.* We assume that backtracking number has higher priority than usage degree. Learner proceeds to next example with selected *NxtMT* by tail recursion.

Least EBG learner simply generates the whole least EBG space. Its size exponentially increases with the number of explanations of the current least EBG set. Remark that it is not the number of examples. Examples are summarized into least EBGs and learner can keep its efficiency. This is a practice of the theorem 1.

Least EBG learner is implemented on SICStus Prolog on Sun4. In next section, we demonstrate least EBG learner in two extreme cases.

3,3 Learning in a Recursive Theory

3.3.1 Deterministic Recursion

Consider the well known linear recursive predicate member/2 shown in Figure 9. The learner is given n examples whose target element is at first, at second, •*•, at n-th place. In much of existing EBLs, its performance often becomes wrong because it generates the same number of macros as examples like $LEBG_1$ [Prieditis *et* a/., 1987]. On the other hand, least EBG learner learns original domain theory because it has no backtracking and maximum usage degree like *LEBG**. The usage degree of $LEBG_2$ becomes *n(n* - l)/2 - 2 for n examples. Offcourse *LEBG* also has no backtracking, but usage degree, which reflects someway matching efficiency, remains minimum.

It is important for a learner to learn nothing when there are nothing to learn. In fact, following theorem

$$D = \begin{cases} member(A,[A|X]). \\ member(A,[B|X]) \leftarrow member(A,X). \end{cases}$$

$$E = \begin{cases} \leftarrow member(a,[a,b,c]). \\ \leftarrow member(2,[1,2]). \\ \leftarrow member(f,[d,e,f,g]). \\ \vdots \end{cases}$$

$$LEBG_1 = \begin{cases} member(A,[A|X]). \\ member(B,[A,B|X]). \\ member(C,[A,B,C|X]). \\ \vdots \end{cases}$$

$$LEBG_2 = \begin{cases} member(A,[A|X]). \\ member(A,[B|X]) \leftarrow member(A,X). \end{cases}$$

Figure 9: least EBGs of a deterministic recursive theory.

holds.

Theorem 4 (learning a deterministic recursion)
If a domain theory solves any examples without backtracking, least EBG learner learns least EBG of no backtracking and of the maximum usage degree, regardless of example orderings.

Proof We show this by induction. It is clear for one example. For *k* example, assume the learner has *LEBGk* of no backtracking and of the maximum usage degree. For *(k* + l)-th example, we show contradiction if there does not exists $LEBG_{k+1}$ of no backtracking from the union of *LEBGk* and that example. The clauses that causes backtracking must have the same rules in their root of their explanation structures, because original domain theory have some backtrackings if they have different rules. Then, their least EBG generalization will reduce this ambiguity because original domain theory have no backtrackings and least EBGs are less general than it. Thus least EBG of no backtracking is found, and it has the maximum usage degree from theorem 2. Q.E.D.

This theorem is applicable for domain theories that contains some redundant non-recursive clauses, such as "hate(W,W) :- depressed(W)" in previous section. Much of existing speed-up learning pointed out that unwinding such redundant clauses contribute to improve efficiency(Subramanian *et* a/., 1990]. This theorem extends this facts into deterministic recursive theories.

3.3.2 Non-linear Recursion

We show another extreme. Consider ancestor/2 that search transitive relation shown in figure 10. The learner is given examples of all of six discendants of a. Original domain theory causes infinite branch
for "← ancestor(a,f) and "<- ancestor(a,g)".

Much of existing EBLs learn exhaustive cache of all example like $LEBG_1$. It is a kind of rote learning but the most operational least EBG because the slightest generalization causes backtracking like *LEBG**. Especially, seif-unwindings of transitive rule make situation

$$D = \begin{cases} \text{ancestor(a,b).} & \text{ancestor(a,c).} \\ \text{ancestor(6,d).} & \text{ancestor(6,e).} \\ \text{ancestor(c,f).} & \text{ancestor(c,g).} \\ \text{ancestor(X, } Z) & \\ \quad \leftarrow \text{ancestor(X, } Y) \wedge \text{ancestor(y, } Z). & \end{cases}$$

$$E = \begin{cases} \leftarrow \text{ancestor(a, } b). & \leftarrow \\ \leftarrow \text{ancestor(a, } d). & \leftarrow \\ \leftarrow \text{ancestor(a, /).} & \leftarrow \end{cases}$$

$$LEBG_1 = \begin{cases} \text{ancestor(a, 6).} & \text{ancestor(a, c).} \\ \text{ancestor (a, <f).} & \text{ancestor(a, e).} \\ \text{ancestor(a,/).} & \text{ancestor(a, } g). \end{cases}$$

$$LEBG_2 =s \begin{cases} \text{ancestor(a, 6).} & \text{ancestor(a, c).} \\ \text{ancestor(b, d).} & \text{ancestor (6, e).} \\ \text{ancestor(c, /).} & \text{ancestor(c, } g). \\ \text{ancestor(a, } Z) \leftarrow & \text{ancestor(6, } Z). \\ \text{ancestor(a, } Z) \leftarrow & \text{ancestor(c, } Z). \end{cases}$$

Figure 10: least EBGs of a non-linear recursive theory.

$$\begin{aligned} E_3 &= \{\mathbf{depressed(king), buy(king, cannon)}\}, \\ G_3 &= \leftarrow \mathbf{kill(king, king)}. \\ MSV &= \mathbf{kill}(W, W) \leftarrow \mathbf{depressed(W)} \wedge \mathbf{buy}(W, Z) \\ &\qquad \wedge \mathbf{small\text{-}w}(Z). \end{aligned}$$

Figure 12: a negative example and the MSV.

worse. [Subramanian *et al,* 1990] have pointed out there exists such a situation that the generalization-to-n in non-linear recursive theories misleads efficiency. The example of ancestor/2 supports this from a viewpoint of operationality.

Unfortunately, since our simple least EBG learner does not generate whole least EBG space of given examples, it is sensitive in example orderings in such a domain. However, it does not generate LEBGs that cause infinite branch, given such as {← ancestor(a,f), ← ancestor(a,d)}, and merely caches them. Therefore, least EBG learner is sufficiently useful in such a domain.

4 Approaches to Other Imperfect Theory Problems

4.1 Inconsistent Theory and EBG Version Space

Inconsistent theory is distinctive in analytic problems such as classification and diagnosis. It is natural to look inconsistency as failures in classification or diagnosis. We assume that training examples are given with judgements whether they are positive or negative, and define a domain theory is inconsistent iff some negative examples are provable as positive examples. A learner with the inconsistent theory must change its competence rather than performance. We assume a learner swaps rules which conclude the goal concept with EBGs. These assumptions are similar to [Flann *et* al., 1989].

A learning task is to find EBGs s.t. prove positive examples but never prove negative examples. In SBL, such generalizations make a range called a version space. We show an EBG space of this paper have the same structure as a version space, and that when the number of disjunctives, i.e the number of macros, are bounded to n, its MSV is a least EBG generalization [Yamamura *et al,* 1990b].

For example, consider a domain theory which have a tree structured conceptual hierarchy. Figure 11 shows such a domain theory as an and-or tree. This is a refinement of *D* of figure 2 where *small-w* denotes "small weapon" and so on. This domain theory is inconsistent. Figure 12 shows one of provable negative examples. E3 says that a king might not kill himself with a cannon if he were depressed. To exclude this example, a learner must specialize the domain theory to constrain instruments for self-destructions to small weapons. The MSV of the 1-bounded EBG version space is the least EBG as shown in figure 12 It includes all small weapons but excludes any big weapons.

IOE of [Flann *et* al., 1989] is the MSV of 1-bounded version space. Thus, the EBG version space is thought of as an extention of IOE to *n* disjunctives. IVSM of [Hirsh, 1989] also uses the concept of version space. His version space is defined on features output by EBL, and is used also for incomplete theory. Our version space is defined directory on an EBG space, and we use more knowledge intensive method for incomplete theory.

4.2 Intractable Theory and Knowledge Compilation

The intractable theory problem is distinctive in synthetic problems such as planning and design. For example, operator definitions of 8-puzzle are complete and consistent for any state transitions, but it is intractable to build a solution only with them. There often exists some strategies to solve any problems without search, such as a concept of serially decomposable subgoals in 8-puzzle [Korf, 1985]. A learning task for intractable theory problem is a knowledge compilation to find such strategies.

We have developed two kinds of approaches. One is to extract correct macros from instructed solution from a teacher under domain theory of primitive operators for problem solving domain that has serially decomposable subgoals such as 8 puzzle [Yamamura *et al.,* 1989b]. Least EBG is used to extract correct macros. However, it often falls into a difficulty of SBL that a learner requires numerous negative examples until convergence. It means that the computational load shifts from EBL to SBL to resolve the imperfectness of the domain theory.

Another is to extend the interpretor to consider decision level or control level domain theory like [Minton *et* al., 1987]. By this, intractable theory problem at object level can be regarded as another kind of imperfect theory problem at decision level. We have developed decision level domain theory for problem solving domain that has serially decomposable subgoals [Yamamura *et* al., 1990b]. This approach is a kind of knowledge compilation.

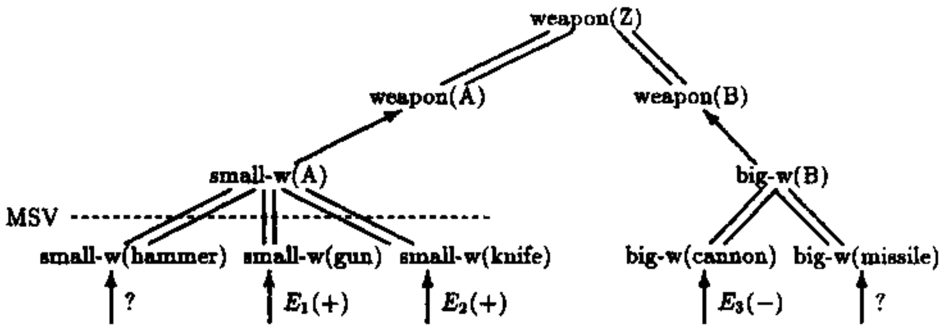

Figure 11: a tree structured hierarcy.

4.3 Incomplete Theory and Knowledge Transfer

For the incomplete theory problem, we have proposed an approach in classification problems using a background theory in addition to the domain theory and examples [Araki *et al.*, 1990]. A background theory consists of a commonsense knowledge such as text books or dictionaries. A learning task is to extend the domain theory by transferring knowledge from the background theory based on similarities of examples. The least EBG can be also used to extract similarities. Remark that without a background theory, the learning task is degenerated into SBL when the domain theory is the most incomplete one, i.e. null. We have given a solution only for a constrained class of domain theories. Otherwise, it soon falls into difficulties of SBL.

IVSM of [Hirsh, 1989] uses pure SBL method when the domain theory is incomplete. Our approach aims more knowledge intensive method since it seems more natural that there exists somewhat relevant knowledge in the background knowledge when the domain theory is incomplete.

5 Conclusion

In this paper, we proposed an augmentation of EBL to handle plural examples simultaneously under imperfect domain theories for unifying SBL and EBL. We presented solutions to utility problem with least EBGs which extract similarities of plural examples. There remain many problems in more imperfect theories, and the augmentation brings together intrinsic difficulties of SBL. However, in real application, useful expertise for learning seems often ignored because they do not fit with existing frameworks. Therefore, knowledge intensive approaches like the augmented EBL seems important.

References

[Araki *et al.*, 1990] Araki, Y. Yamamura, M. and Kobayashi, S. *An Incremental Learning of Domain Theory with a Background Theory. Knowledge Eng. Sympo.,* 1990 in Japanese.

[Bergadano *et al.*, 1988] Bergadano, F. and Giordana, A. *A Knowldge Intensive Approach to Concept Induction. IML88,* 305-317,1988.

[Flann *et ai,* 1989] Flann, N. S. and Dietterich, T. G. *A Study of Explanation-Baaed Methods for Inductive Learning. Machine Learning,* 4, 187-226 1989.

[Hirsh, 1987] Hirsh Haym. *Explanation-Based Generalization in a Logic-Programming Environment IJCAI87,* 221-227, 1987.

[Hirsh, 1989] Hirsh Haym. *Combining Empirical and Analytical Learning with Version spaces. IML89,* 29-33, 1989.

[Kedar-Cabelli *et al,* 1987] Kedar-CabeUi, S. T. and McCarty, L. T. *Explanation-Based Generalization as Resolution Theorem Proving. IML87,* 383-389, 1987.

[Keller, 1988] Keller, R. M. *Defining Operationality for Explanation-Based Learning. Artif Intell.,* 35, 227-241 1988.

[Korf, 1985] Korf, R. E. *Macro-Operators : A Weak Method for Learning. Artif. Intell,* 26, 35-77 1985.

[Laird *et al.,* 1986] Laird, J. E. Rosenbloom, P. S. and Newell, A. *Chunking in Soar ; the Anatomy of a General Learning Mechanism. Machine Learning,* 1, 11-46 1986.

[Minton et *al.*, 1987] Minton, S. and Carbonell, J. G. *Starategies for Learning Search Control Rules: An Explanation-based Approach. IJCAI87,* 228-235 1987.

[Mitchell, 1982] Mitchell, T. M. *Generalization as Search. Artif. Intell,* 18, 203-226 1982.

[Mitchell *et al,* 1986] Mitchell, T. M. Keller, R. M. and Kedar-Cabelli, S. T. *Explanation-Based Generalization: An Unifying View. Machine Learning,* 1, 47-80 1986.

[Prieditis *et al,* 1987] Prieditis, A. E. and Mostow, J. *PROLEARN : Towards A Prolog Interpreter that Learns. AAAI87,* 494-498 1987.

[Segre, 1987] Segre, A. M. *On the Operationality/Generality Trade-off in Explanation-Based Learning. IJCAI87,* 242-248 1987.

[Subramanian *et al,* 1990] Subramanian, S. and Feldman, R. *The Utility of EBL in Recursive Domain Theories. AAAI90,* 942-949 1990.

[Yamamura *et al,* 1989a] Yamamura, M. and Kobayashi, S. *An Augmentation of EBL on Plural Examples. JJSAI,* 4, 4, 389-397 1989 in Japanese.

[Yamamura *et al,* 1989b] Yamamura, M. and Kobayashi, S. *An Augmented EBL for Problem Solving Macrotable. SICE Annual Conf.,* 1377-1380 1989.

[Yamamura, 1990a] Yamamura, M. *A Theory of an Augmented EBL and its Application to Problem Solving. Ph.D Thesis,* Tokyo Inst, of Tech. 1990 in Japanese.

[Yamamura *et al,* 1990b] Yamamura, M. and Kobayashi, S. *Towards Unifying EBL and SBL to Solve Imperfect Theory Problems. PRICAI90,* 595-600 1990.