Chapter 6

ψ-RHB$^+$: ILP System that Learns Logic Programs based on ψ-terms

6.1 Overview of ψ-RHB$^+$

6.1.1 Framework

The framework of hierarchically sorted ILP based on ψ-terms that learns from only positive examples is defined as follows. In this chapter, we call literals and clauses based on ψ-terms just literals and clauses.

Let a signature be $\Sigma_{OSF} = \langle \mathcal{P}, \mathcal{S}, \preceq, \sqcap, \sqcup, \mathcal{L} \rangle$, where $\mathcal{P}$ is a finite set of predicates, $\mathcal{S}$ is a finite set of sort symbols, and $\mathcal{L}$ is a finite set of feature symbols and $\mathcal{V}$ be a finite set of variables. Example language L_E, background knowledge language L_B, and hypothesis language L_H are defined as follows.

- L_E: the set of all ground atomic formulae whose predicate symbols is an observation predicate symbol p.
- L_B: the set of all ground unit clauses without the predicate symbol p.
- L_H: the set of all definite clauses whose heads are atomic formulae with p and whose bodies consist of literals with predicates symbols

in background knowledge language L_B.

Let a finite set of positive examples be $E^+ \subseteq L_E$ and a finite set of background knowledge be $B \subseteq L_B$, where the following conditions are satisfied:

$$\forall e \in E^+ \ B \not\vdash_{OSF} e$$

Hierarchically sorted ILP based on ψ-terms is defined as φ that satisfies the following conditions with respect to hypotheses H.

$$H \subseteq L_H \ \ s.t. \ \ H = \varphi(L_E, L_B, L_H, B, E^+).$$
$$B \cup H \ is \ consistent.$$
$$For \ most \ of \ e \in E^+ \quad B \cup H \vdash_{OSF} e.$$
$$For \ most \ of \ e \in L_E - E^+ \quad B \cup H \not\vdash_{OSF} e.$$

Because our goal is to apply an ILP based on ψ-terms to the learning from real-world data with some noise, the second last condition is relaxed from "$\forall e \in E^+$" to "for most of $e \in E^+$". The last condition states that H is more preferable if it covers a smaller amount of examples in L_E other than E^+. We employ in this chapter the weighted informativity as a heuristics to estimate the appropriateness of H.

6.1.2 Characteristics of ψ-terms

For the sake of introducing features (or attributes) and sorts, ψ-terms enable the following advantages.

partial descriptions For example, term $name(first \Rightarrow peter)$ expresses the information of a person whose first name is known. This is equivalent to $name(first{\Rightarrow}peter, last{\Rightarrow}\top)$. In the process of unification [4], possibly other features can be added to the term.

dynamic generalization and specialization Sorts, placed at the positions of function symbols, can be dynamically generalized and specialized. For example, $person(id \Rightarrow name(first \Rightarrow person))$ is a generalized form of $man(id \Rightarrow name(first \Rightarrow Jack))$.

abstract representation Abstract representations of examples using sorts can reduce the amount of data. For example,

$$familiar(person(residence \Rightarrow France), French)$$

represents a number of ground instances, such as,

$$familiar(Serge(residence \Rightarrow France), French).$$

coreference Coreference enables the recursive representation of terms. For example, $X : person(spouse \Rightarrow person(spouse \Rightarrow X))$ refers to itself recursively [1] .

natural language processing applicability ψ-terms have the same representation power as *feature structures* [10] which are used for formally representing the syntax and semantics of natural language sentences.

6.1.3 Least General Generalization

In the definition of the *least general generalization (lgg)* [43], the part that defines the term *lgg* should be extended to the *lgg* of ψ-terms. The lgg of ψ-terms has already been described in [1]. The lgg of a subset of *description logics*, called the *least common subsumer (LCS)*, can be found in [12]. The lgg of *feature terms*, which are equivalent to ψ-terms, can be found in [42]. We present a definition of a least general generalization of two ψ-terms as an extension of Plotkin's lgg described in Section 2.2,

Now, we *operationally* define a least general generalization of two ψ-terms based on [55], using the following notations. a and b represent untagged ψ-terms. s, t, and u represent ψ-terms. f, g, and h represent sorts. X, Y, and Z represent variables in $\mathcal{V}$.

Definition 74 (lgg of ψ-terms)

1. $lgg(X : a, X : a) = X : a$.

[1] Variables are also used as coreference tags. This is one of the most elegant ways to represent coreference.

2. *$lgg(s,t) = u$, where $s \neq t$ and the tuple (s,t,u) is already in the history Hist.*

3. *If $s = X{:}f(l_1^s \Rightarrow s_1, .., l_n^s \Rightarrow s_n)$ and $t = Y{:}g(l_1^t \Rightarrow t_1, .., l_m^t \Rightarrow t_m)$, then $lgg(s,t){=}u$, where $L = \{l_1^s, ..., l_n^s\} \cap \{l_1^t, ..., l_m^t\}$ and for features $l_i \in L$, $u{=}Z{:}h(l_1 \Rightarrow lgg(s.l_1, t.l_1), ..., l_{|L|} \Rightarrow lgg(s.l_{|L|}, t.l_{|L|}))$ with $h = f \sqcup g$. Then, (s,t,u) is added to Hist.*

For example, the lgg of

$$injured(passenger(of\ \Rightarrow 10))$$

and

$$injured(men(of\ \Rightarrow 2))$$

is

$$injured(people(of\ \Rightarrow number)),$$

where $passenger \sqcup men = people$ and $10 \sqcup 2 = number$.

6.2 Algorithm of ψ-RHB$^+$

The positive examples are atomic formulae based on ψ-terms. The hypothesis language is a set of Horn clauses based on ψ-terms. The background knowledge also consists of atomic formulae. ψ-RHB$^+$ [2], a ψ-term capable ILP system, employs a bottom-up approach, like Golem [38].

Learning Algorithm

The learning algorithm of our ILP system is based on the Golem's algorithm extended for ψ-terms. The steps are shown in Algorithm 6.

In Step 2, we have to link sorts in the examples and the background knowledge because the OSF formalism [4] which underlies the formalism of ψ-terms is not formed under the *unique name assumption*. For

[2] In this thesis, ψ-RHB [55] is called ψ-RHB$^+$ in order to emphasize that it learns from only positive examples. It would be easy to modify ψ-RHB$^+$ so that it can learn from both positive and negative examples; however, this modification has not been investigated yet.

Algorithm 6 *Learning algorithm*

```
1. Given positive examples $E^+$, background knowledge $B$.
2. Link sorts which have the same names in $E^+$ and $B$.
3. A set of hypotheses $H = \{\}$.
4. Select $K$ pairs of examples $(A_i, B_i)$ as $EP$ $(0 \leq i \leq K)$.
5. Select sets of literals $AR_i$ and $BR_i$ as selected background knowl-
   edge according to the variable depth $D$.
6. Compute lggs of clauses $A_i{:}-\bigwedge AR_i$ and $B_i{:}-\bigwedge BR_i$.
7. Simplify the lggs by evaluating with weighted informativity $PWI$,
   which is the informativity used in RHB$^+$.
8. Select the best clause $C$, and add it to $H$ if the score of $C$ is better
   than the threshold $\delta$.
9. Remove covered examples from $E^+$.
10. If $E^+$ is empty then return $H$; otherwise, goto Step 4.
```

example, if we have two terms $f(t)$ and $g(t)$, t in $f(t)$ and t in $g(t)$ are not identical. The OSF formalism requires that they be represented as $f(X : t)$ and $f(X : t)$, if t is identical in both of two terms. Therefore, the same sort symbols in the examples and in the background knowledge are linked and will be treated as identical symbols in the later steps.

In Step 5, to speed up the learning process, literals related to each pair of examples are selected. At first, AR_i and BR_i are empty. Then, (1) select the background knowledge literals A_{sel} so that it has all literals whose sort symbols are identical to the sorts in A_i or AR_i, and select B_{sel} in the same manner using B_i or BR_i. (2) Add literals A_{sel} and B_{sel} to sets AR_i and BR_i, respectively. Repeat (1) and (2) n times when the predefined variable depth is n. This iteration creates sets of literals.

We use AR_i as the selected background knowledge for A_i, and BR_i for B_i in Step 6. What is computed in Step 6 is the following lgg of clauses.

$$lgg((A_i :- \bigwedge AR_i), (B_i :- \bigwedge BR_i)),$$

where $\bigwedge S$ is a conjunction of all of the elements in S. The *lggs* of clauses have the variable depth of at most n.

In Step 7, simplification of the lggs is achieved by checking all literals in the body as to whether removal of literals makes the score of the weighted informativity worse or not. For the purpose of informativity estimation, we use the concept of ground instances of atomic formulae based on ψ-terms. We call atomic formula A ground instance if all of the sorts appearing in A are constants. For example,

$$familiar(Serge(residence \Rightarrow France), French).$$

Moreover, literals in the body are checked as to whether they satisfy the input-output mode declarations of the predicates.

6.3 Discussion

ψ-RHB$^+$ employs the lgg of clauses whose heads are positive examples and whose bodies are literals selected from background knowledge. This approach is an extension of Golem's approach which is based on Muggleton's *relative least general generalization (rlgg)*. Our approach and Muggleton's approaches are not simple computation of Plotkin's rlgg.

As an rlgg of two positive examples e_1 and e_2, Golem computes $lgg(e_1 :- K, e_2 :- K)$, where K is a conjunction of the literals in B.

However, Lemma 3 in [44] clearly states that $lgg(e_1, e_2)$ is a least general generalization of e_1 and e_2, *relative to* Th. There is no need for Golem to add K to the bodies when computing an rlgg in a mathematical sense.

This can be explained as follows. Let $C_1 = lgg(e_1 :- K, e_2 :- K)$ and $C_2 = lgg(e_1, e_2)$. $C_1 \sim C_2(Th)$ but $C_1 \not\sim C_2$. Obviously, $C_2 \leq C_1$ when Th is not empty. What Golem and our algorithm mainly computes is a variation of clauses that are relatively equivalent to the lgg of two positives.

The reason why ψ-RHB$^+$ employs the lgg of clauses based on ψ-terms is simply that Golem has successfully generalized positive examples in various test example sets. It is of very interest to investigate the theoretical background of finding a most useful clause in $\{C|C \sim C_2(Th)\}$.

6.4 Summary

This chapter has described an algorithm of a ψ-term capable ILP and its application to information extraction. The lgg of logic terms was extended to the lgg of ψ-terms. The learning algorithm is based on the lgg of clauses with ψ-terms. Natural language processing relies on a vast variety of nouns relating to the sort hierarchy (or taxonomy) which plays a crucial role in generalizing data generated from the natural language. Therefore, the information extraction task will match the requirements of the ψ-term capable ILP.