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 $\Sigma_{OSF} = (\mathcal{P}, \mathcal{S}, \mathcal{Z}, \cap, \cup, \mathcal{L})$, 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 $L_B$, 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:

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

Hierarchically sorted ILP based on $\psi$-terms is defined as $\varphi$ that satisfies the following conditions with respect to hypotheses $H$.

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

Because our goal is to apply an ILP based on $\psi$-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 $\psi$-terms

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

**Partial descriptions** For example, term $\text{name}(first \Rightarrow peter)$ expresses the information of a person whose first name is known. This is equivalent to $\text{name}(first \Rightarrow peter, last \Rightarrow T)$. 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, $\text{person}(id \Rightarrow \text{name}(first \Rightarrow \text{person}))$ is a generalized form of $\text{man}(id \Rightarrow \text{name}(first \Rightarrow \text{Jack})))$. 

58
abstract representation Abstract representations of examples using sorts can reduce the amount of data. For example,

\textit{familiar(person(residence } \Rightarrow \text{ France}), French)}

represents a number of ground instances, such as,

\textit{familiar(Serge(residence } \Rightarrow \text{ France), French)}.

coreference Coreference enables the recursive representation of terms. For example, \( X : \text{person(spouse } \Rightarrow \text{ person(spouse } \Rightarrow X)) \) refers to itself recursively \(^1\).

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

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

Definition 74 (\textit{l gg of } \psi \text{-terms})


\(^1\) Variables are also used as coreference tags. This is one of the most elegant ways to represent coreference.
2. $\text{lgg}(s, t) = u$, where $s \neq t$ and the tuple $(s, t, u)$ is already in the history $\text{Hist}$.

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 $\text{lgg}(s, t) = u$, where $L = \{l_1^s, ..., l_n^s\} \cap \{l_1^t, ..., l_m^t\}$ and for features $l_i \in L$, $u = Z : h(l_i \Rightarrow \text{lgg}(s.l_i, t_l_i), ..., l_{|L|} \Rightarrow \text{lgg}(s.l_{|L|}, t.l_{|L|}))$

with $h = f \cup g$. Then, $(s, t, u)$ is added to $\text{Hist}$.

For example, the $\text{lgg}$ of

\[ \text{injured(passenger(of \Rightarrow 10))} \]

and

\[ \text{injured(men(of \Rightarrow 2))} \]

is

\[ \text{injured(people(of \Rightarrow number))}, \]

where $\text{passenger} \cup \text{men} = \text{people}$ and $10 \cup 2 = \text{number}$.

6.2 Algorithm of $\psi$-RHB$^+$

The positive examples are atomic formulae based on $\psi$-terms. The hypothesis language is a set of Horn clauses based on $\psi$-terms. The background knowledge also consists of atomic formulae. $\psi$-RHB$^+$ \footnote{In this thesis, $\psi$-RHB \cite{55} is called $\psi$-RHB$^+$ in order to emphasize that it learns from only positive examples. It would be easy to modify $\psi$-RHB$^+$ so that it can learn from both positive and negative examples; however, this modification has not been investigated yet.}, a $\psi$-term capable ILP system, employs a bottom-up approach, like Golem \cite{38}.

Learning Algorithm

The learning algorithm of our ILP system is based on the Golem's algorithm extended for $\psi$-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 \cite{4} which underlies the formalism of $\psi$-terms is not formed under the unique name assumption. For
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 knowledge according to the variable depth \( D \).
6. Compute \( lggs \) of clauses \( A_i \): \( \neg \bigwedge AR_i \) and \( B_i \): \( \neg \bigwedge BR_i \).
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.

$$\text{lgg}((A_i : - \wedge AR_i), (B_i : - \wedge BR_i)),$$

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

In Step 7, simplification of the lgg 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 $\psi$-terms. We call atomic formula $A$ ground instance if all of the sorts appearing in $A$ are constants. For example,

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

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

### 6.3 Discussion

$\psi$-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 $\text{lgg}(e_1 : - K, e_2 : - K)$, where $K$ is a conjunction of the literals in $B$.

However, Lemma 3 in [44] clearly states that $\text{lgg}(e_1, e_2)$ 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 $C_1 = \text{lgg}(e_1 : - K, e_2 : - K)$ and $C_2 = \text{lgg}(e_1, e_2)$. $C_1 \sim C_2(Th)$ but $C_1 \not\prec C_2$. 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 $\psi$-RHB$^+$ employs the lgg of clauses based on $\psi$-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 $\{C \mid C \simeq C_2(Th)\}$.

6.4 Summary

This chapter has described an algorithm of a $\psi$-term capable ILP and its application to information extraction. The lgg of logic terms was extended to the lgg of $\psi$-terms. The learning algorithm is based on the lgg of clauses with $\psi$-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 $\psi$-term capable ILP.