An Analytic Method of Seasonal Reference in Japanese Haiku-Poem†

Yamasaki, Susumu; Yokono, Hikaru

Abstract— *This paper makes an overall design by means of an analytic method to determine the season word of a given "haiku": In Japan, there has been a unique poem style known as the "haiku", which (1) takes three parts of 5, 7 and 5 syllables, (2) contains a fragment of 5 syllables followed by a phrase of 7 and 5 syllables, a phrase of 5 and 7 syllables followed by a fragment of 5 syllables, or unifies three parts without any fragment and phrase separation, and (3) involves a season word for a seasonal reference. The season word must be a key word in the haiku. However, there is a critical problem to identify the season word of a haiku, when it has got more than one season word. It has been traditionally determined by the intention of the writer with reference to the sense and sensibility. We organize a methodology to see seasonal reference with lexical and logical analyses of the given haiku in a text form. Not all haikus are logically analyzed, but some can be so that they are experimentally consistent with the sense.*

Keywords— *Application of logic to Japanese poem, Applied logic (ISY)*

1. INTRODUCTION

FOLLOWING the book ([1]), we see that there is a unique poem style renowned as a "haiku" in the Japanese language. The haiku takes the form:

(i) to take three parts of 5, 7 and 5 syllables, where one syllable consists of a consonant followed by a vowel, except the consonant n.
(ii) to contain a fragment (consisting of one part) and a phrase (consisting of two parts) or to unify 5, 7 and 5 syllables without any fragment and phrase separation.
(iii) to involve a season word for a seasonal reference.

In English, the poetry is enjoyed even at a primitive stage such that there is a starting set of rules:

(i) to write in three lines that are short, long, short without counting syllables.
(ii) to make sure that haiku has fragment and phrase.
(iii) to have some element of nature.
(iv) to use the verb in the present tense.
(v) to avoid capital letters or punctuation.
(vi) to avoid rhymes.

With the usage of capital letters or punctuation, there is another way to follow the 5, 7 and 5 syllables ([2]). For the illustration of haiku all in this paper, we refer to English translations of haikus, which Basho made, from the book ([2]). For example,

In this hush profound,
Into the very rock it seeps —
The cicada sound.

(A) Technical Interest of Haiku

With reference to the above rules (iii) and (iv), the season word in the Japanese haiku is free from the constraint at that stage of the book ([1]). However, we think of the season word as essential in the haiku written in Japanese, because Japanese likes the situation which is:

(a) to denote some instance or pause in an environment where the poem writer has been or is.
(b) to denote a time point in the period when natural phenomena were, are and would be changing.

In general, except the haiku where there is no season word, we are allowed to have the cases that:

(i) there is just one season word.
(ii) there are more than one season word.

We have example:

How many cloud shapes
Capped the peak before the moon
Rose on Moon Mountain?
(cloud shape: summer, moon: autumn)

(B) Problem of Haiku Season Word

If there is only one season word, then we can enjoy the haiku by it as a key word. To automate the process of detecting the season word or none for both of the above cases, we face a problem of how we could find it for a given haiku. From the view of arts, what word is seasonal in a haiku is concerned with sensibility of the writer and the readers, inspired by it. There is some Japanese explanation

Author is with Department of Computer Science,
Okayama University, Japan
(e-mail: yamasaki@momo.cs.okayama-u.ac.jp)

that the season word is determined by the writer's mind itself, but no theory to determine it from the context of the haiku. It may be relevant to haiku's content. That is the problem from the analytic view by which we can partly see semantics of the haiku, and with which we can enjoy the haiku more familiarly.

Hence we are interested in the analysis problem of the season word contained in a Japanese haiku. If there are more than one season word, the problem is mainly related to the preference of one word.

(C) Analytic Solution

With the implementable lexical analysis of the haiku text, we aim at a logical analysis to classify and identify the season word or none with negations (denoting negative information processing). This logical analysis is motivated by a structural way to prefer one word in a category among more than one candidate word.

The logical analysis presents a method to organize an automated system to find out the season word. Experiments can be made, if both lexical and logical analyses are combined well.

This problem and a system construction as some solution of it would be an interdisciplinary aspect between artistic preference in arts and logical analysis in computing.

(D) Paper Organization

The paper is organized as follows. Section 2 is concerned with category of season words and with a logical method to represent it. In Section 3, a logical analysis is presented for the preference of season word. In Section 4, the system design is shown in relation to the logical analysis of Section 3. A formal aspect of the interpreter is described for preference of season word in the haiku-processing. In Section 5, we have an outlook at the system implementation. In Section 6, concluding remarks and considerations are made.

2. Season WORD IN HAIKU

2.1 Category of Season Words

In this section, we present some attributes of the season word. It contains:

(i) a reference to a seasonal aspect.
(ii) a suggestion of a spatial aspect.
(iii) a relevance to an emblem of sense.

We have the corresponding examples:

(i) What a cool, summer breeze!
Here, I make myself at home,
Rest, and take my ease.

(ii) O'er wild ocean spray,
All the way to Sado Isle
Spreads the Milky Way!

(iii) The river Mogami
Has drowned the hot, summer sun
And sunk it in the sea!

If we take a logical analysis for detection of the season word, the third item cannot be captured, but the first two aspects may be understood. The attributes may be partially categorized. The season word is evidently classified as referring to one of seasons: spring, summer, autumn and winter. It is also regarded as referring to the one in 7 categories: weather, astronomical aspect, geography, event, life, animal, and plant.

The season word categorized in weather, astronomical aspect, geography, event or life can preferably refer to the seasonal decision, even if there are more than one season word.

2.2 Logical Method

When we have more than one season word in a haiku, we must prefer some one as a primary season word which definitely determines the situation of the haiku. Based on the category, the primary season word may be inferred such that the preference of a season word to other words may be logically represented.

The predicate to denote preference of m to n for the primary word is defined to be

$$pref(m,n) = \begin{cases} true & \text{if } m \text{ is a primary season word and} \\ & n \text{ is not a primary season word,} \\ & \text{or } m \text{ is a season word and} \\ & n \text{ is not a season word,} \\ false & \text{otherwise,} \end{cases}$$

where variables *m* and *n* stand for words, or none (which is a special "symbol" and not even a season word). Note the case that we may see the season of the given haiku by means of some interactive method between a designed system and a system user. By such an interaction, the season of the haiku may be supposedly identified for the purpose of knowing which is the primary season word. For simplicity, this paper does not deal with such an interactive way.

Regarding whether it is a primary season word, we make use of the predicate *primary(x)* for the word, which may contain negations in 2-valued or 3-valued logic.

The 3-valued logic contains three truth-values **t**, **u** and **f**, where the value **u** stands for the undefined. Three negations, *not* (default negation), $\sim_w$ (weak negation) and $\sim_s$ (strict negation) as in [3] may be relevant to the representation of the predicate *primary(x)*.

	not	$\sim_w$	$\sim_s$
t	**f**	**f**	**f**
u	**u**	**t**	**f**
f	**t**	**t**	**t**

Table 1. Negations.

Let *season_w(m)* be a predicate to state that the word *m* is a season word. Also let *categorized(m)* be a predicate to state that the word *m* may be categorized as a supposedly primary word for a haiku, with reference to category of season words. The predicate *categorized(m)* is sensitive to an association with the primary (season) word. It is not always logical, but often rather related to some sense of the writer or the reader. However, by means of the predicate, we represent preference or non-preference of a season word with reference to the role of the primary word.

We have preference in terms of strict and default negations among the above three. The negation *not* is used to state that it is not a primary season word. Unless a season word is categorized, it is regarded as negated for the role of the primary word. If a word is not a season word, it is strictly negated for a primary season word. The strict negation $\sim_s$ is used to state that it is not a season word, nor a primary word.

In the case that negations $\sim_s$ and *not* are adopted:

- If both *season_w(m)* and *categorized(m)* then *primary(m)*.
- If *not season_w(m)* then $\sim_w$ *primary(m)*.
- If *season_w(m)* and *not categorized(m)* then *not primary(m)*.

3. LOGICAL ANALYSIS TO IDENTIFY SEASONAL REFERENCE

We must prepare for the predicates:

(i) *Season(x,m)* stands for the case that the word m is the primary season one among the list of words *x*. If *x* is empty, "*none*" is assigned to *m*.
(ii) *pref(m,n)* is as in Section 2, to stand for a preference of *m* (as the primary word) to *n*.
(iii) The predicate *primary(m)* states that the word *m* is regarded as a primary season one.
(iv) The predicate *season_w(m)* states that the word *m* is a season one.
(v) The predicate *categorized(m)* states that the word may be acknowledged as a supposedly primary one, with reference to category of season words.

By means of the view of logical analysis to identify the primary word, we have logical relations among predicates, some of which are now re-stated by the same manner as in the previous section. *first(x)* is the first element of the list *x*, while rest(x) is the list obtained by removing *first(x)* from *x*.

(a) If the list *x* is empty then *Season(x,none)*.
(b) If *Season(rest(x),a)* and *pref(first(x),a)* then *Season(x,first(x))*.
(c) If *Season(rest(x),a)* and *pref(a,first(x))* then *Season(x,a)*.
(d) If *Season(rest(x),none)* and $\sim_s$ *primary(first(x))* then *Season(x,none)*.
If *primary(a)* and *not primary(b)*, or *season_w(a)* and $\sim_s$ *primary(b)*, or *season_w(a)* and *b* is *none*, then *pref(a,b)*.
(e) If *season_w(a)* and *categorized(a)* then *primary(a)*.
(f) If *not season_w(a)* then $\sim_s$ *primary(a)*.
If *season_w(a)* and *not categorized(a)* then *not primary(a)*.

Now take the haiku:

How cool the autumn air!
I'll peel them and enjoy them --
The melon and the pear.

(i) It is clear that *season_w(autumn air being cool)*.
From the categorical classification,
categorized(autumn air being cool)
so that *primary(autumn air being cool)*.
(ii) We also see that
season_w(melon) and *season_w(pear)*.
But we have the negated predicates
not categorized(melon) and
not categorized(pear).

The logical analysis is automated by means of an interpreter whose formal aspect is presented in what follows, so that the negated predicates

not primary(m) and $\sim_s$ *primary(m)*

may be available.

4. RULE-BASED DATABASE WITH PREFERENCE

4.1. Negation versus Preference

The logical analysis to identify the primary word can be implemented by the representation of rule-based database and its retrieval. The rule-based database may be expressed by logic programs with negations, where the negations are built in an expression of preference. Regarding the present method of preference, the backgrounds of logic programming with negation is given by [4], [5], [6], [7], [8].

The rule-based database consists of (a)-(f) statements as in Section 3. The relation can be stated by the (clausal) form:

$$A \leftarrow B_1, \ldots, B_l, not\, C_1, \ldots, not\, C_m, \sim_s D_1, \ldots, \sim_s D_n,$$

where:

(i) $A, B_i\ (1 \le i \le l)$, $C_j\ (1 \le j \le m)$, and $D_k\ (1 \le k \le n)$ are predicates.
(ii) The negations *not* and $\sim_s$ denote default and strict ones, respectively.

The form means that:

If B_1 and … and B_l and *not* C_1 and … and *not* C_m and $\sim_s D_1$ and … and $\sim_s D_n$ then A.

In accordance with the statements (a)-(f) in Section 3, we demonstrate the relations in the clausal form:

(a) $Season(nil, none) \leftarrow$
(b) $Season(x, first(x)) \leftarrow Season(rest(x), a), pref(first(x), a)$
(c) $Season(x, a) \leftarrow Season(rest(x), a), pref(a, first(x))$
(d) $pref(a,b) \leftarrow primary(a), not\ primary(b)$
$pref(a,b) \leftarrow season\_w(a), \sim_s primary(b)$

$pref(a, none) \leftarrow$

(e) $primary(a) \leftarrow season\_w(a), categorized(a)$

(f) $not\ primary(a) \leftarrow season\_w(a), not\ categorized(a)$

$\sim_s\ primary(a) \leftarrow not\ season\_w(a)$

Note that the rule-based database is represented by a set of clauses (clausal forms) as above, where the negations *not primary(a)* and $\sim_s$ *primary(a)* of the item (f) occur only in the right-hand sides (bodies) of other clausal forms, for a simpler procedure construction.

A query to ask whether
B_1 and … and B_l and *not* C_1 and … and *not* C_m
and $\sim_s D_1$ and … and $\sim_s D_n$
holds for the database is represented by the goal statement (which is the negation of a query) of the form:

$$\leftarrow B_1, \ldots, B_l, not\ C_1, \ldots, not\ C_m, \sim_s D_1, \ldots, \sim_s D_n.$$

It means that:
The conjunction of B_1 and … and B_l
and *not* C_1 and … and *not* C_m
and $\sim_s D_1$ and … and $\sim_s D_n$ is contradictory.

For example, a query of whether the predicate *Season(x,m)* holds, that is, the word is a primary word of the list *x* is represented by its negated form $\leftarrow Season(x, m)$ to obtain a substitution (calculation) of a value (term) for the variable m. For the given rule-based database and the goal statement, we have in [3] a procedure to see whether the query denoted by the goal statement is well answered with reference to the database.

4.2. Description of Procedure

As in [3], we recursively define the relations $suc_P,\ fail_P \subseteq Goal$ for the set *Goal* of goals, where *P* denotes the rule-based database of the ground version, that is, the rule-based database containing no variable: By the relations $suc_P(\leftarrow A)$ and $fail_P(\leftarrow A)$, we have the meanings that the goal (statement) $\leftarrow A$ succeeds, and that the goal $\leftarrow B$ fails, respectively. The relations are to be the least set satisfying the following rule closure, where two rules (a) and (b) are contained for the rule closure to be compositional. A (possibly empty) sequence of literals (predicates or their negations) is denoted by using letters like *G*, G_1, G_2 ,…, where a sequence of literal sequences is also regarded as a sequence of literals.

(a) $suc_P(\leftarrow G_1), suc_P(\leftarrow G_2) \Rightarrow suc_P(\leftarrow G_1, G_2).$

(b) $fail_P(\leftarrow G) \Rightarrow fail_P(\leftarrow G_1, G, G_2).$

(i) $suc_P(\square).$

(ii) $suc_P(\leftarrow G), (A \leftarrow G) \in P \Rightarrow suc_P(\leftarrow A).$

(iii) $fail_P(\leftarrow A) \Rightarrow suc_P(\leftarrow not\ A).$

(iv) $fail_P(\leftarrow A) \Rightarrow suc_P(\leftarrow \sim_s A).$

(v) no clause (in P) with A in head $\Rightarrow fail_P(\leftarrow A).$

(vi) for all clauses $A \leftarrow G,\ fail_P(\leftarrow G) \Rightarrow fail_P(\leftarrow A).$

(vii) $suc_P(\leftarrow A) \Rightarrow fail_P(\leftarrow not\ A).$

(viii) no relation $fail_P(\leftarrow A) \Rightarrow fail_P(\sim_s A).$

The procedure is in accordance with the set (which is not always the least) satisfying the above rule closure. The procedure is described for the database of the ground version. If the database involves variables, the procedure may be graded up to the first-order version, with negations containing no variable for the safe-rule (as surveyed in [9]).

In the sense that the following theorem holds, the procedure is consistent. The consistency is relevant to consistent reasonings as in [3], [10], [11], [12], [13], [14], [15], [16]. The proofs of Theorems 4.1 and 4.2 are verbally accounted in Appendix.

Theorem 4.1: Assume a predicate *A* for a rule-based database *P*. When there is a succeeding derivation from the goal $\leftarrow A$ for P, there is no failing derivation from the goal $\leftarrow A$.

We observe a model theoretic aspect of the succeeding and failing derivations.

Theorem 4.2: Assume a rule-based database *P* of the ground version with strict and default negations. Take the sets:

(i) $T = \{A \mid A \text{ is a predicate such that} \leftarrow A \text{ succeeds}\}.$

(ii) $F = \{B \mid B \text{ is a predicate such that} \leftarrow B \text{ fails}\}.$

The pair *(T,F)* is a 3-valued Herbrand model of *P*.

5. SYSTEM DESIGN TO SEE PRIMARY SEASON WORD

The automated system, which is implemented to infer a primary season word for a given haiku, contains the system modules of:
(i) lexical analysis of haiku texts and transformation of real haiku texts to lists of words.
(ii) rule-based database, which includes relations among season and primary words with the list *x*.
(iii) logical analysis interpreter to find a primary season word for a list of words, with reference to the rule-based database.

The methods of modules can be demonstrated by Table 2.

Modules	Methods
(i) Lexical analysis and Transformation to word	(i) Syntactic analysis
(ii) Rule-based database	(ii) Logic programming with negation
(iii) Interpreter	(iii) Succeeding and failing derivation

Table 2. Methods of System Modules.

5.1. Lexical Analysis

The Japanese haiku contains a sequence of 17, or less or more syllables without any separation pause or symbol between syllables. We must find a pause between words by lexical analysis so that the haiku can be a sequence of words.

Based on a sequence of words, we make a processing to get a list of words which may be season words, by excluding words or a sub-sequence of words which cannot be season words. Words included in the parts like pronoun, adverb and so on can be excluded in any category of season words. The system JUMAN, whose improved version appears in [17], is available for this system design.

- The exclusion rules are as follows.
- The word, which is a pronoun, an adverb, an adnominal, an auxiliary verb, a conjunction or an interjection, can be excluded.
- The word which is a conjunction, followed by another word, can be neglected such that the following one is examined.
- The word, which is just a suffix, can be neglected.
- The sequence consisting of a noun, "no" (in Japanese) and a noun is regarded as a word.
- Some sequence of a noun and a verb is regarded as a word.

5.2. Rule-Based Database

As demonstrated in Section 4, we may implement the system to infer the primary season word for a given haiku.

After the stages in lexical analysis and transformation of it to a list of words, the system executes the following routine, based on the rule-based database and implementation module (constructed by means of the formal method as in Section 4). Let *x* be the list of words, while the variable *m* contains a season word or *none*.

```
procedure $primary\_word\_detection$;
begin
$m := none$;
while $(x \neq nil)$ do
   begin
      if the goal $\leftarrow Season(x,\ first(x))$ succeeds
      then begin $m := first(x)$; $x := nil$ end
      else $x := rest(x)$
   end;
$return(m)$
end.
```

5.3. Illustration

We now consider the primary season words of the previous examples.

For the haiku:
How many cloud shapes
Capped the peak before the moon
Rose on Moon Mountain?

It is easy for the designed system to detect both *season_w(cloud shapes)* and *season_w(moon)*.
We can have both
categorized(cloud shapes) and
categorized(moon)
so that the system must fail, unless the word "moon" included in
"the moon" and "Moon Mountain"
are identified. If they are identified, then the word "moon" must suggest the mountain name so that we have *not primary(moon)*. In this case, *primary(cloud shapes)*.

6. CONCLUDING REMARKS

In this paper, we make use of strict and default negations for preference of the primary season word of the haiku which may involve more than one season word. The primary word must be both a season word and a categorized one. The acknowledgement of a category contains sensibility, but can be made by a classification of season words. Unless the word is seasonal, it may be regarded as strictly negated. Even if it is seasonal but not categorized, it is interpreted as default negation.

The database for the preference of primary words is incorporated in succeeding and failing derivations. The SLDNF resolution may be available.

(1) The well-founded model ([18], [19]) is acquired for the modified database (logic program) obtained by substituting default negation for strict one.
(2) The well-founded model may be even a model of the original database with strict and default negations. (The proof is omitted.)

However, the succeeding and failing derivations work well for the database with strict and default negations, they are more refined than the SLDNF resolution derivations such that the derivations are consistent and contain model theoretic aspect in the sense of Theorem 4.2. On the other hand, more specific and concrete treatments are not yet made, while the preference theory and practice have begun as in case of a conflict between defaults of [20].

From broader views, this paper's treatment is expected to be included in diagnosis as in [21] or in specification as in [22] for haiku analysis by means of semantic aspect. Interactive haiku analysis and synthesis systems are the subject of haiku-processing by applying agents (as functions) in [23]. Whether relation between logical analysis (in [24]) and Japanese sensibility is made clear is still a problem.

Appendix

(A) Proofs

(1) Proof of Theorem 4.1:

Assume that there is both a succeeding derivation from the goal $\leftarrow A$ and a failing one from the goal $\leftarrow A$. Because there is a succeeding derivation, by means of the rule (ii) with the clause $A \leftarrow G$ in *P*, we have a goal $\leftarrow G$. At the same time, a goal $\leftarrow G$ is obtained by means of the rule (vi), since there is a failing derivation from the goal $\leftarrow A$. By the same reasons for both succeeding and failing derivations, we reach a case without loss of generality that the goal $\leftarrow G$ succeeds and the goal $\leftarrow G$ fails, where *G* takes the form

$\leftarrow not\ B_1, \ldots, not\ B_m, \sim_s C_1, \ldots, \sim_s C_n.$

Because of the succeeding derivation from goal $\leftarrow G$ with reference to the rule (iii) or the rule (iv),

any goal among $\leftarrow B_1, \ldots, \leftarrow B_m$ fails and

any goal among $\leftarrow C_1, \ldots, \leftarrow C_n$ fails.

Because of the failing derivation from the goal $\leftarrow G$ with reference to the rule (vii) or the rule (viii),

some goal among $\leftarrow B_1, \ldots,$ and $\leftarrow B_m$ succeeds, or no goal among $\leftarrow C_1, \ldots,$ and $\leftarrow C_n$ fails.

It follows that there is some predicate B_j such that:

- the goal $\leftarrow B_j$ fails.
- the goal $\leftarrow B_j$ succeeds.

For the above reason, we recursively reach the assumption more than once that the goal $\leftarrow A'$ succeeds and the goal $\leftarrow A'$ fails. This is a contradiction, because the goal $\leftarrow A'$ cannot succeed, owing to the illegal recursion. Hence the first assumption is contradictory such that this concludes the proof. q.e.d.

(2) Proof of Theorem 4.2:

Note that the program contains no variable. By Theorem 4.1, the pair (T, F) is consistent, that is, a 3-valued Herbrand interpretation. For any clause

$A \leftarrow B_1, \ldots, B_l, not\ C_1, \ldots, not\ C_m, \sim_s D_1, \ldots, \sim_s D_n,$

we have the following cases:

(i)

$\forall B_i.[(1 \le i \le l) \Rightarrow B_i \in T]$ and
$\forall C_j.[(1 \le j \le m) \Rightarrow C_j \in F]$ and
$\forall D_k.[(1 \le k \le n) \Rightarrow D_k \in F]$
$\Rightarrow$
$\forall B_i.[(1 \le i \le l) \Rightarrow \leftarrow B_i$ succeeds] and
$\forall C_j.[(1 \le j \le m) \Rightarrow \leftarrow C_j$ fails] and
$\forall D_k.[(1 \le k \le n) \Rightarrow \leftarrow D_k$ fails]
$\Rightarrow \leftarrow A$ succeeds
$\Rightarrow A \in T$, that is, the clause is true.

(ii)

$\exists B_i.[(1 \le i \le l)$ and $B_i \in F]$ and
$\exists C_j.[(1 \le j \le m)$ and $C_j \in T]$ and
$\exists D_k.[(1 \le k \le n)$ and $D_k \notin F]$
$\Rightarrow$ the clause is true.

(iii)

$\forall B_i.[(1 \le i \le l) \Rightarrow B_i \notin F]$ and
$\exists B_i.[(1 \le i \le l)$ and $B_i \notin T]$ and
$\forall C_j.[(1 \le j \le m) \Rightarrow C_j \notin T]$ and
$\exists C_j.[(1 \le j \le m)$ and $C_j \notin F]$ and
$\forall D_k.[(1 \le k \le n) \Rightarrow D_k \in F]$
$\Rightarrow$
$\forall B_i.[(1 \le i \le l) \Rightarrow \leftarrow B_i$ does not fail] and
$\exists B_i.[(1 \le i \le l)$ and $\leftarrow B_i$ does not succeed] and
$\forall C_j.[(1 \le j \le m) \Rightarrow \leftarrow C_j$ does not succeed] and
$\exists C_j.[(1 \le j \le m)$ and $\leftarrow C_j$ does not fail] and
$\forall D_k.[(1 \le k \le n) \Rightarrow \leftarrow D_k$ fails]
$\Rightarrow \leftarrow A$ does not fail
$\Rightarrow A \notin F$, that is, the clause is true.

Hence any clause of the program *P* is true in the pair (T, F). That is, the pair is a model. This completes the proof. q.e.d.

References

[1] J. Reichhold, Writing and Enjoying Haiku. Tokyo: Kodansha International, 2002.
[2] D. Britton, A Haiku Journey: Basho's Narrow Road to a Fair Province. Tokyo: Kodansha International, 1980.
[3] S. Yamasaki, "Logic programming with default, weak and strict negations," Theory and Practice of Logic Programming, vol. 6, pp.737-749 , 2006.
[4] J. Minker (ed.), Foundations of Deductive Databases and Logic Programming. Los Altos: Morgan Kaufmann Publishers, 1987.
[5] P. Schmitt, "Computational aspects of three-valued logic," Lecture Notes in Computer Science, vol. 230, 1986, pp. 190–198.
[6] C. Ruiz and J. Minker, "Logic knowledge bases with two default rules," Annals of Mathematics and Artificial Intelligence, vol. 22, 1998, pp. 333–361.
[7] J. Shepherdson, Negation in Logic Programming, ser. Foundations of Deductive Databases and Logic Programming. Los Altos: Morgan Kaufmann Publishers, 1987, pp. 18–88.
[8] ——, "A sound and complete semantics for a version of negation as failure," Theoretical Computer Science, vol. 65, 1989, pp. 343–371.
[9] J. Lloyd, Foundations of Logic Programming, 2nd, Extended Edition. Berlin: Springer-Verlag, 1993.
[10] P. Dung, "Negation as hypothesis: An abductive foundation for logic programming," Proc. of 8th ICLP, 1991, pp. 2–17.

[11] ——, “An argumentation-theoretic foundation for logic programming,” Journal of Logic Programming, vol. 22, 1995, pp. 151–177.

[12] L. Giordano, A. Martelli, and M. Sapino, “Extending negation as failure by abduction: A three-valued stable model semantics,” Journal of Logic Programming, vol. 26, 1996, pp. 31–67.

[13] K. Kunen, “Signed data dependencies in logic programs,” Journal of Logic Programming, vol. 7, 1989, pp. 231–245.

[14] L. Pereira, N. Joaquim, and J. Alferes, “Derivation procedures for extended stable models,” Proc. of 12th IJCAI, 1991, pp. 863–868.

[15] S. Yamasaki and Y. Kurose, “A sound and complete proof procedure for a general logic program in no-floundering derivations with respect to the 3-valued stable model semantics,” Theoretical Computer Science, vol. 266, 2001, pp. 489–512.

[16] J.-H. You and L. Yuan, “On the equivalence of semantics for normal logic programs,” Journal of Logic Programming, vol. 22, 1995, pp. 211–222.

[17] T. Nakamura, Y. Matsumoto, S. Kurohashi, and M. Nagao, “Improvements of Japanese morphological analyzer JUMAN,” Proc. of International Workshop on Sharable Natural Language Resources, vol. 7, 1994, pp. 48–55.

[18] A. Van Gelder, “The alternating fixpoint of logic programs with negation,” Journal of Computer and System Sciences, vol. 47, 1993, pp. 185–221.

[19] A. Van Gelder, K. Ross, and J. Schlipf, “The well-founded semantics for general logic programs,” Journal of ACM, vol. 38, 1990, pp. 620–650.

[20] G. Brewka, “Well-founded semantics for extended logic programs with dynamic preference,” Journal of Artificial Intelligence Research, vol. 4, 1996, pp. 19–36.

[21] R. Reiter, “A theory of diagnosis from first-principles,” Artificial Intelligence, vol. 32, 1987, pp. 57–95.

[22] E. Burke and E. Foxley, Logic and Its Application. London: Prentice-Hall, 1996.

[23] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. London: Prentice-Hall, 1995.

[24] B. Russell, History of Western Philosophy. London: Routledge, 2000.