Study of Non-monotonic Reasoning with Negative Sort

Koki Mizuno

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 2001

Keywords: non-monotonic reasoning, rule generation, turn over positive-negative predication , hierarchical structure, exception processing, strong negation.

In this paper, we construct a system which generates rules from the given knowledge in the frame of non-monotonic reasoning. This system expresses hierarchical structure between rules. And, we study a method of how strong negation is expressed on this system.

Usually, horn-program or clause set program do to describe relation with inductive logic program. Although this method can not describe in knowledges on monotonic reasoning. Therefor we need frame of non-monotonic reasoning for possible to express exception processing and default reasoning.

System always has same knowledges then does not cause a problem. This problem is turning over positive-negative predication. This problem causes by changing knowledge base. If knowledge base does not change then positive example is always positive example. Actually this moment does not exist. If negative knowledges increase more than positive knowledges then this situation causes turning over positive-negative predication. We study about resolution of this situation. System does not have completed knowledge base. Therefor we consider turning over positive-negative predication. And we consider theorem for supplement of knowledge base defect. This theorem called close world assumption:CWA. CWA applies knowledge base restriction. Knowledges which do not include in system are treated as negative information. For example system has 'Jack is bird' and system does not have 'Jack flies' or 'Jack does not fly' then system adopts 'Jack does not fly' by CWA effect. Jack is an entity. Knowledge 'Jack is bird' includes background. Knowledge 'Jack flies'and 'Jack does not fly' are respectively positive example and negative example. Decision to negative example by CWA does not change this decision. Therefor generated rules cause contradiction by this decision. System should not

apply this decision by CWA for avoid contradiction. Open world specialize:OWS is able to treat non-decisionable states of positive or negative of example without system. Therefor we use this method for construct system owns non-decisionable state of examples. And system is able to turn over positive-negative predication.

We consider construction of hierarchical structure refer on order sorted logic. Hierarchical structure expresses a relation between rules. Hierarchical structure relates an exception processing. For example, generally birds fly. But exception exist. Ostrich or injured bird do not fly. View bird's property of 'fly' then we can see exception. Now system is generated rule 'a bird flies', next when system generate negative-rule 'a bird does not fly', this rule is included in 'bird flies' structure. Next bird is structure in animal. Exception of animal is bird. Bird's structure is included in animal structure. Therefor we consider construction of create multi hierarchical structure.

System tunes up with it is able to treat implicit negative predication. We apply strong negation as implicit negation predication. Strong negation has 'un-' on head of word. Strong negation is strong more than classical negation. '¬happy(b)' expresses b is not happy. 'unhappy(b)' expresses b is unhappy. Classical negation '¬' expresses not happy. But 'un-' expresses only unhappy. Difference of negation causes difference of designation objects. We consider this disagreement.

It is first-order predication that system has. This system constructs with example set and background set. Example set consists positive predications and negative predications. Generated rule type consists 'example $\Leftarrow$ background, not exception'. Background explain object. For example 'bird(Y)' explain 'Y is a bird'. Example set include positive predications and negative predications and respectively 'fly+(Z)','fly-(W)'express 'Z flies','W does not fly'. Generate rule from premise knowledge: 'bird flies' to

$$\text{fly+(X)} \Leftarrow \text{bird(X)}, \backslash\text{+ab(X)}$$

'ab(X)' is an exception. It expresses 'bird X does not fly'.

We proceed extension to system which is able to treat multi predication. And consider system generate which included strong negation information.