Automatic Ordering of Predicates by Metarules

Tristan Cazenave
LAFORIA-IBP
case 169
Université Pierre et Marie Curie
4, place Jussieu
75252 PARIS CEDEX 05, FRANCE
cazenave@laforia.ibp.fr

Abstract

I describe metarules which order predicates contained in first order logic rules. These metarules are applied to rules created by a learning Go program. After the ordering of the predicates, the rules are matched orders of magnitude faster than the original learned rules.

Key words : Metaprogramming, Metareasoning, First Order Logic, Game of Go, Machine Learning.

1 Introduction

One aspect of machine learning systems is the efficiency of the learned rules. This problem has been referenced as the utility problem [Minton 1988]. I have written a learning program which transforms its problem solving activity in the efficient matching of first order rules [Cazenave 1996]. A component of this system is the compilation of the learned rules. This paper describes how learned rules are compiled so as to match them orders of magnitude faster. This problem has already been adressed in papers such as [Ishida 1988]. My approach makes explicit use of metaprogramming, my system reasons on itself so as to improve itself.

The first part of the paper is concerned with the representation of rules and of metarules. A second part shows an example of predicate ordering of a rule created by the system. Then the possible extensions of the work are discussed and perspectives for future work are shown.

2 The rules and the Metarules

2.1 The rules

The rules are represented using a list of premises and a list of conclusions. A premise and a conclusion are composed of metapredicates, predicates, functions, variables and constants. The metapredicates used in rules are the four metapredicates : ‘present’, ‘absent’, ‘ajoute’ and ‘enleve’. Which respectively test if a fact is present in the working memory and instanciates the variables, test is a fact or a set of facts are absent of the working memory, append a fact to the working memory, retract a fact from the working memory. A variable always begins with a ‘ ?’. My system allows the use of integer and real variables and constants.

The working memory used to unify with my rules represents a position in the game of Go. The rules determine what are the consequences of some moves for some fixed goals of the

Game of Go. Efficiently matching these rules is crucial to a Go program. The more it can match rules, the more it understands the position and the better it plays.

Moreover, my program is a learning Go program. Thus the rules it learns do not have a good ordering of predicates. It is vital for him to reason about itself so has to order itself the predicates involved in its rules. Table 1 gives an example of a rule learned by my system.

```
( premises    (
            present ( Couleur ( ?c ) )
            present ( Couleur_opposees ( ?c1 ?c ) )
            present ( Couleur ( ?c1 ) )
            present ( Nombre_voisines_couleur_avant ( ?i + 4 ) )
            present ( Nombre_voisines ( ?i 4 ) )
            present ( Nombre_Blocs_voisins_avant ( ?i 0 ) )
            present ( Couleur_intersection_avant ( ?i + ) )
            present ( Voisine ( ?i ?i4 ) )
            present ( Couleur_intersection_avant ( ?i4 + ) )
            present ( Voisine ( ?i ?i6 ) )
            present ( Couleur_intersection_avant ( ?i6 + ) )
            intersections_differentes ( ?i4 ?i6 )
            present ( Voisine ( ?i ?i2 ) )
            present ( Couleur_intersection_avant ( ?i2 + ) )
            intersections_differentes ( ?i6 ?i2 )
            intersections_differentes ( ?i4 ?i2 )
            present ( Voisine ( ?i2 ?i1 ) )
            present ( Couleur_intersection_avant ( ?i1 + ) )
            present ( Nombre_voisines ( ?i1 4 ) )
            present ( Nombre_voisines_couleur_avant ( ?i1 ?c1 0 ) )
            present ( Voisine ( ?i4 ?i1 ) )
            intersections_differentes ( ?i1 ?i )
            present ( Nombre_Blocs_voisins_couleur_avant ( ?i1 ?c 0 ) )
            present ( Voisine ( ?i ?i7 ) )
            absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) )
            present ( Couleur_intersection_avant ( ?i7 + ) )
            intersections_differentes ( ?i4 ?i7 )
            intersections_differentes ( ?i6 ?i7 )
            intersections_differentes ( ?i2 ?i7 )
            present ( Voisine ( ?i1 ?i8 ) )
            absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) )
            present ( Couleur_intersection_avant ( ?i8 + ) )
            intersections_differentes ( ?i4 ?i8 )
            intersections_differentes ( ?i2 ?i8 )
            present ( Couleur_intersection_avant ( ?i8 ?c2 ) )
            present ( Voisine ( ?i1 ?i3 ) )
            absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) )
            present ( Couleur_intersection_avant ( ?i3 + ) )
            intersections_differentes ( ?i4 ?i3 )
            intersections_differentes ( ?i8 ?i3 )
            intersections_differentes ( ?i2 ?i3 )
            present ( Voisine ( ?i4 ?i5 ) )
            absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) )
            present ( Couleur_intersection_avant ( ?i5 + ) )
            intersections_differentes ( ?i ?i5 )
            intersections_differentes ( ?i1 ?i5 )
           )
  conclusions   (
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) )
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i1 ?i ?i1 GIII ) )
           )
)
```

Table 1

2.2 The Metarules

My system uses various kinds of metarules. I give in Table 2 two examples of metarules used to give a priority to predicates inside a list of premises. The metapredicate ‘regle’ instanciates in ?r all the rules of a list of rules. The metapredicate ‘condition’ look if its second argument unifies with a premise of the rule in its first argument, ?var and ?var1 are metavariables which can be instanciated in any variable. The metapredicate ‘priorite’ assigns to its third argument the priority corresponding to the premises of the rule in its first argument which unify with the premise in its second argument. The function ‘superieur_reel’ verifies than it first real argument is greater than its second real argument.

(nom (Metaregle_ordonne_1) premises (regle (?r) condition (?r present (Voisine (?var ?var1))) instanciee (?var) non_instanciee (?var1) priorite (?r present (Voisine (?var ?var1)) ?reel) superieur_reel (?reel 3.79)) conclusions (affecte_priorite (?r present (Voisine (?var ?var1)) 3.79)))	(nom (Metaregle_ordonne_2) premisses (regle (?r) condition (?r present (Nombre_voisines (?var 2))) instanciee (?var) priorite (?r present (Nombre_voisines (?var 2)) ?reel) superieur_reel (?reel 0.01)) conclusions (affecte_priorite (?r present (Nombre_voisines (?var 2)) 0.01)))

Table 2

The information contained in these rules are about the repartition of the facts in the working memory. They give the average number of instanciations of a variable when the premise instanciates variables. They can also give the probability of unifying a fact when all its arguments are instanciated.

3 Predicate Ordering

3.1 Gathering Informations on Unification

My system has the possibility to observe its behavior when unifying rules. It can collect the number of times it unifies a predicates. The information gathered on a 9x9 Go board working memory is given in Table 3. Each premise and conclusion is followed by the number of time it has been unified.

```
Number of Nodes 55315
Number of new facts deduced 208

( premises     (
              present ( Couleur ( ?c ) ) 1
              present ( Couleur_opposees ( ?c1 ?c ) ) 2
              present ( Couleur ( ?c1 ) ) 2
              present ( Nombre_voisines_couleur_avant ( ?i + 4 ) ) 2
              present ( Nombre_voisines ( ?i 4 ) ) 76
              present ( Nombre_Blocs_voisins_avant ( ?i 0 ) ) 76
              present ( Couleur_intersection_avant ( ?i + ) ) 76
              present ( Voisine ( ?i ?i4 ) ) 76
              present ( Couleur_intersection_avant ( ?i4 + ) ) 304
              present ( Voisine ( ?i ?i6 ) ) 304
              present ( Couleur_intersection_avant ( ?i6 + ) ) 1216
              intersections_differentes ( ?i4 ?i6 ) 1216
              present ( Voisine ( ?i ?i2 ) ) 912
              present ( Couleur_intersection_avant ( ?i2 + ) ) 3648
              intersections_differentes ( ?i6 ?i2 ) 3648
              intersections_differentes ( ?i4 ?i2 ) 2736
              present ( Voisine ( ?i2 ?i1 ) ) 1824
              present ( Couleur_intersection_avant ( ?i1 + ) ) 7044
              present ( Nombre_voisines ( ?i1 4 ) ) 6912
              present ( Nombre_voisines_couleur_avant ( ?i1 ?c1 0 ) ) 5724
              present ( Voisine ( ?i4 ?i1 ) ) 5562
              intersections_differentes ( ?i1 ?i ) 2676
              present ( Nombre_Blocs_voisins_couleur_avant ( ?i1 ?c 0 ) ) 852
              present ( Voisine ( ?i ?i7 ) ) 832
              absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 3328
              present ( Couleur_intersection_avant ( ?i7 + ) ) 328
              intersections_differentes ( ?i4 ?i7 ) 328
              intersections_differentes ( ?i6 ?i7 ) 222
              intersections_differentes ( ?i2 ?i7 ) 114
              present ( Voisine ( ?i1 ?i8 ) ) 108
              absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 432
              present ( Couleur_intersection_avant ( ?i8 + ) ) 170
              intersections_differentes ( ?i4 ?i8 ) 170
              intersections_differentes ( ?i2 ?i8 ) 166
              present ( Couleur_intersection_avant ( ?i8 ?c2 ) ) 112
              present ( Voisine ( ?i1 ?i3 ) ) 112
              absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 448
              present ( Couleur_intersection_avant ( ?i3 + ) ) 448
              intersections_differentes ( ?i4 ?i3 ) 448
              intersections_differentes ( ?i8 ?i3 ) 336
              intersections_differentes ( ?i2 ?i3 ) 224
              present ( Voisine ( ?i4 ?i5 ) ) 112
              absent ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 448
              present ( Couleur_intersection_avant ( ?i5 + ) ) 192
              intersections_differentes ( ?i ?i5 ) 174
              intersections_differentes ( ?i1 ?i5 ) 114
            )
  conclusions   (
              ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 104
              ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i1 ?i ?i1 GIII ) ) 104
            )
)
```

Table 3

3.2 Example of a learned rule

Table 4 gives an example of a rule learned by my system before ordering. The number of nodes involved in unifying the rule makes it impossible to unify it in reasonable times.

```
Number of Nodes 808206733
Number of new facts deduced 104

( premisses     (
            present ( Voisine ( ?i1 ?i3 ) ) 1
            present ( Couleur_intersection_avant ( ?i + ) ) 288
            present ( Voisine ( ?i ?i4 ) )  21600
            present ( Couleur_intersection_avant ( ?i8 + ) ) 76800
            present ( Couleur ( ?c ) ) 5836800
            present ( Voisine ( ?i4 ?i5 ) ) 11673600
            present ( Couleur_opposees ( ?c1 ?c ) ) 41558016
            present ( Couleur ( ?c1 ) ) 41558016
            present ( Nombre_voisines_couleur_avant ( ?i + 4 ) ) 41558016
            present ( Couleur_intersection_avant ( ?i2 + ) ) 41558016
            present ( Voisine ( ?i1 ?i8 ) ) 20009416
            present ( Nombre_voisines_couleur_avant ( ?i1 ?c1 0 ) ) 988120
            present ( Nombre_Blocs_voisins_avant ( ?i 0 ) ) 963722
            present ( Couleur_intersection_avant ( ?i4 + ) ) 963722
            present ( Couleur_intersection_avant ( ?i6 + ) ) 904232
            intersections_differentes ( ?i4 ?i6 ) 68721706
            present ( Voisine ( ?i ?i2 ) ) 65328042
            present ( Nombre_voisines ( ?i1 4 ) ) 3266402
            intersections_differentes ( ?i6 ?i2 ) 1572712
            intersections_differentes ( ?i4 ?i2 ) 1553296
            present ( Couleur_intersection_avant ( ?i7 + ) ) 1534120
            present ( Voisine ( ?i2 ?i1 ) ) 116593074
            intersections_differentes ( ?i4 ?i8 ) 115153652
            present ( Couleur_intersection_avant ( ?i1 + ) ) 113732002
            present ( Voisine ( ?i4 ?i1 ) ) 106711508
            present ( Voisine ( ?i ?i6 ) ) 5269704
            present ( Nombre_voisines ( ?i 4 ) ) 260232
            intersections_differentes ( ?i1 ?i ) 260232
            present ( Nombre_Blocs_voisins_couleur_avant ( ?i1 ?c 0 ) ) 257020
            present ( Voisine ( ?i ?i7 ) ) 222116
            intersections_differentes ( ?i4 ?i7 ) 10968
            present ( Couleur_intersection_avant ( ?i3 + ) ) 10834
            intersections_differentes ( ?i4 ?i3 ) 10164
            intersections_differentes ( ?i8 ?i3 ) 10040
            intersections_differentes ( ?i2 ?i3 ) 9916
            present ( Couleur_intersection_avant ( ?i5 + ) ) 9792
            intersections_differentes ( ?i ?i5 ) 9188
            intersections_differentes ( ?i1 ?i5 ) 9074
            present ( Couleur_intersection_avant ( ?i8 ?c2 ) ) 8962
            intersections_differentes ( ?i6 ?i7 ) 8962
            intersections_differentes ( ?i2 ?i7 ) 6812
            intersections_differentes ( ?i2 ?i8 ) 4912
          )
  conclusions   (
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII ) ) 3248
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i1 ?i ?i1 GIII ) ) 3248
          )
)
```

Table 4

3.3 Rule after ordering

Ordering the premises of the rule makes the unification much faster. Table 5 gives the number of nodes involved. It allows the rule to be unified rapidly. Thus my system is able to unify much more rules. Moreover another mechanism is used so as not to deduce many times the same conclusion using different paths in the unification graph. It consists in verifying the conclusion has not been already deduced when instanciating new variables. This is done by inserting ‘absent’ premises after premises instanciating variables. A priority is given to the instanciation of the variables present in conclusion in order to instanciate them as soon as possible in the unification of the rule. The sooner they are instanciated in the rules, the more savings are done.

```
Number of Nodes 111289
Number of new facts deduced 104

( premises    (
            present ( Couleur ( ?c )  ) 1
            present ( Couleur_opposees ( ?c1 ?c )  ) 2
            present ( Couleur ( ?c1 )  ) 2
            present ( Nombre_voisines_couleur_avant ( ?i + 4 )  ) 2
            present ( Nombre_voisines ( ?i 4 )  ) 76
            present ( Nombre_Blocs_voisins_avant ( ?i 0 )  ) 76
            present ( Couleur_intersection_avant ( ?i + )  ) 76
            present ( Voisine ( ?i ?i4 )  ) 76
            present ( Couleur_intersection_avant ( ?i4 + )  ) 304
            present ( Voisine ( ?i ?i6 )  ) 304
            present ( Couleur_intersection_avant ( ?i6 + )  ) 1216
            intersections_differentes ( ?i4 ?i6 ) 1216
            present ( Voisine ( ?i ?i2 )  ) 912
            present ( Couleur_intersection_avant ( ?i2 + )  ) 3648
            intersections_differentes ( ?i6 ?i2 ) 3648
            intersections_differentes ( ?i4 ?i2 ) 2736
            present ( Voisine ( ?i2 ?i1 )  ) 1824
            present ( Couleur_intersection_avant ( ?i1 + )  ) 7044
            present ( Nombre_voisines ( ?i1 4 )  ) 6912
            present ( Nombre_voisines_couleur_avant ( ?i1 ?c1 0 )  ) 5724
            present ( Voisine ( ?i4 ?i1 )  ) 5562
            intersections_differentes ( ?i1 ?i ) 2676
            present ( Nombre_Blocs_voisins_couleur_avant ( ?i1 ?c 0 )  ) 852
            present ( Voisine ( ?i ?i7 )  ) 832
            present ( Couleur_intersection_avant ( ?i7 + )  ) 3328
            intersections_differentes ( ?i4 ?i7 ) 3328
            intersections_differentes ( ?i6 ?i7 ) 2496
            intersections_differentes ( ?i2 ?i7 ) 1664
            present ( Voisine ( ?i1 ?i8 )  ) 832
            present ( Couleur_intersection_avant ( ?i8 + )  ) 3328
            intersections_differentes ( ?i4 ?i8 ) 3328
            intersections_differentes ( ?i2 ?i8 ) 2496
            present ( Couleur_intersection_avant ( ?i8 ?c2 )  ) 1664
            present ( Voisine ( ?i1 ?i3 )  ) 1664
            present ( Couleur_intersection_avant ( ?i3 + )  ) 6656
            intersections_differentes ( ?i4 ?i3 ) 6656
            intersections_differentes ( ?i8 ?i3 ) 4992
            intersections_differentes ( ?i2 ?i3 ) 3328
            present ( Voisine ( ?i4 ?i5 )  ) 1664
            present ( Couleur_intersection_avant ( ?i5 + )  ) 6656
            intersections_differentes ( ?i ?i5 ) 6576
            intersections_differentes ( ?i1 ?i5 ) 4912
           )
  conclusions   (
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i ?i1 ?i GIII )  ) 3248
            ajoute ( Coup_jeu_binaire_intersection_intersection_avant ( ?c Connecter ?i1 ?i ?i1 GIII )  ) 3248
           )
)
```

Table 5

The insertion of ‘absent’ premises approximately doubles the speed of the unification. The unification costs 111,289 node in Table 5 without the absent optimization. It only costs 55315 in Table 3 with the ‘absent’ premises inserted.

4 Conclusion

I have shown how to order predicates using metarules and metapredicates. This rules and metarules are used in a Go learning program playing at an international level [Pettersen 1994].

The method described in this paper allows speedups of 14,000 when matching rules on a working memory representing a 9x9 Go board. It can give even better speedups on larger working memories. This technique can be reused in domains where we know a priori the repartition of the facts in the working memory. This is the case for many domains. I actually apply predicate ordering to rules about other games and about the management of a firm. This work is a part of a longer goal which is to create autonomous self programming systems [Pitrat 1990]. The efficient unification of a great number of rules containing many condition is vital for deductive learning systems. The use of metarules for compiling rules offers a good way to enhance it greatly.

Bibliography

[Cazenave 1996] - T. Cazenave, *Learning to Forecast by Explaining the Consequences of Actions*, Workshop on Machine Learning, Forecasting and Optimization, Madrid, July 1996.

[Ishida 1988] - T. Ishida, *Optimizing Rules in Production System Programs*, AAAI 1988, pp 699-704, 1988.

[Minton 1988] - S. Minton, *Learning Search Control Knowledge - An Explanation Based Approach*, Kluwer Academics, Boston, 1988.

[Pettersen 1994] - E. Pettersen. *The Computer Go Ladder*. World Wide Web page: http://cgl.ucsf.edu/go/ladder.html, 1994.

[Pitrat 1990] - J. Pitrat, *Métaconnaissance - Futur de l'Intelligence Artificielle*, Hermès, Paris, 1990.