Two Results Regarding Refinement Operators

Fabrizio Riguzzi

Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1
44100 Ferrara, Italy,
friguzzi@ing.unife.it

Abstract. In this paper we present two results regarding refinement operators. The first is that it does not exist a refinement operator that is both complete and optimal for the θ-subsumption ordering and for the language of full clausal logic.
The second regards the properties of the refinement operator implemented in Aleph's code by predicate `auto_refine/2`. We think this operator is interesting for its simplicity and because it does not require the construction of a bottom-clause. In particular, the operator is useful in the cases where a bottom-clause can not be built, as for example in learning from interpretations. The properties of this operator are that it is locally finite, not proper nor complete but weakly complete. Moreover, the operator is also not optimal. However, it can be made complete by extending the specification of the language bias and by requiring that the language does not contain function symbols.

1 Preliminaries

A couple (G, R) of a set G and a relation R over $G \times G$ is a *quasi-ordered set* if R is reflexive and transitive. We will usually denote a relation R of a quasi-ordered set as $\geq$. We will write that $A > B$ iff $A \geq B$ but $B \ngeq A$ and $A \approx B$ iff $A \geq B$ and $B \geq A$.

The most common relation used in ILP is θ-subsumption. A clause C θ-subsumes a clause D iff there exists a substitution θ such that $C\theta \subseteq D$ and we write $C \geq_\theta D$. We will write that $A >_\theta B$ iff $A \geq_\theta B$ but $B \ngeq_\theta A$ and $A \approx_\theta B$ iff $A \geq_\theta B$ and $B \geq_\theta A$. θ-subsumption is reflexive and transitive so, if G is a set of clauses, $(G, \geq_\theta)$ is a quasi-ordered set.

Let us now define the concept of downward refinement operators for quasi-ordered sets. In the literature there exist two different definitions. The first, from [3] is: a *downward refinement operator* for a quasi ordered set $(G, \geq)$ is a function ρ such that $\rho(C) \subseteq \{D | C \geq D\}$ for every $C \in G$. The second, from [2], is: a *downward refinement operator* for a quasi ordered set $(G, \geq)$ is a function ρ such that $\rho(C) \subseteq \{D | C \geq D$ and D is a maximal specialization of $C\}$ for every $C \in G$, i.e. $\rho(C) \subseteq \{D | C \geq D$ and $\nexists E \in G$ such that $C > E$ and $E > D\}$ for every $C \in G$. We will use the latter definition.

Let us give also some properties regarding refinement operators. Let $(G, \geq)$ be a quasi-ordered set and let ρ a downward refinement operator for $(G, \geq)$ [3]:

- the sets of *one-step refinements*, *n-step refinements*, and *refinements* of some $C \in G$ are respectively:
  $\rho^1(C) = \rho(C)$
  $\rho^n(C) = \{D|$ there is an $E \in \rho^{n-1}(C)$ such that $D \in \rho(E)\}, n \geq 2$
  $\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup \rho^3(C) \ldots$
- a *ρ-chain* from C to D is a sequence $C = C_0, C_1, \ldots, C_n = D$, such that $C_i \in \rho(C_{i-1})$ for every $1 \leq i \leq n$
- ρ is *locally finite* iff, for every $C \in G$, $\rho(C)$ is finite and computable
- ρ is *complete* iff, for every $C, D \in G$ such that $C > D$, there is an $D' \in \rho^*(C)$ such that $D' \approx D$
- ρ is *weakly complete* iff $\rho^*(\text{false}) = G$ [1] (such an operator is called *complete* in [2, 6]),
- ρ is *proper* iff, for every $C \in G$, $\rho(C) \subseteq \{D|C > D\}$
- ρ is *ideal* iff it is locally finite, complete and proper
- ρ is *DD-optimal* iff $\forall D, C_1, C_2 \in G : D \in \rho^*(C_1) \cap \rho^*(C_2) \rightarrow C_1 \in \rho^*(C_2)$ or $C_2 \in \rho^*(C_1)$ (called simply optimal in [2])

We will use here a different definition of an optimal refinement operator. ρ is *R-optimal* iff $\forall D_1, D_2, C_1, C_2 \in G : D_1 \in \rho^*(C_1), D_2 \in \rho^*(C_2), D_1 \approx D_2 \rightarrow (\exists C_1' \in \rho^*(C_2)$ such that $C_1' \approx C_1)$ or $(\exists C_2' \in \rho^*(C_1)$ such that $C_2' \approx C_2)$.

The notions of DD-optimality and R-optimality are not related, i.e., DD-optimality does not imply R-optimality nor viceversa.

In [5] the non existence of DD-optimal refinement operators was proved for the $(G, \leq_\theta)$ where G is unrestricted. However this result does not carry over to R-optimality.

In the following we will use simply optimal in place of R-optimal.

Lemma 1. *In a proper and optimal refinement operator, if $D \in \rho^*(C)$ then there do not exist two distinct ρ-chains from C to clauses D_1 and D_2 respectively such that $D_1 \approx D$ and $D_2 \approx D$.*

Proof. The proof proceeds by contradiction. Consider a proper and optimal refinement operator ρ, suppose $D \in \rho^*(C)$ and that there are two different ρ-chains, one from C to D_1 and one from C to D_2 such that $D_1 \approx D$ and $D_2 \approx D$. Then there exist two clauses C_1 and C_2 on which the two ρ-chins split. Let us call E their parent, i.e., $C_1 \in \rho(E)$ and $C_2 \in \rho(E)$.

Let us now prove by contradiction that $\nexists C_1' \in \rho^*(C_2)$ such that $C_1' \approx C_1$. If such a C_1' existed, then $C_2 > C_1$ and thus C_1 would not belong to $\rho(E)$ because it is not a maximal specialization. In the same way we can prove that $\nexists C_2' \notin \rho^*(C_1)$ such that $C_2' \approx C_2$. This contradicts the assumption that ρ is optimal.

This means that the refinement graph becomes a tree, with $false$ as root and with equivalence classes of clauses as nodes.

The following corollary is a simple special case of the previous theorem.

Corollary 1 (A similar theorem is reported, without a proof, in [2]). *In a proper and optimal refinement operator, if $D \in \rho^*(C)$ then there do not exist two distinct ρ-chains from C to D.*

2 Non Existence of Optimal and Complete Operators

Theorem 1. *There does not exist a proper operator that is both optimal and complete for $(G, \leq_\theta)$, where G is the language of full clausal logic.*

Proof. Consider two clauses D_1 and D_2 such that there exist a clause E that is in $\rho^*(D_1) \cap \rho^*(D_2)$. It is easy to show that such clauses exist in the G set considered.

Since θ-subsumption forms a lattice, there exists a clause C that it is the least general generalization of D_1 and D_2. Since G contains every possible clause, it contains C.

Consider a proper, complete and optimal refinement operator ρ. Since $C \geq_\theta D_1$ and $C \geq_\theta D_2$, then there exists clauses D'_1 and D'_2 such that $D'_1 \in \rho^*(C)$, $D'_2 \in \rho^*(C)$, $D'_1 \approx_\theta D_1$ and $D'_2 \approx_\theta D_2$. For corollary 1 there is a single ρ-chain from C to D'_1 and a single ρ-chain from C to D'_2.

But $D'_1 \geq_\theta E$ and $D'_2 \geq_\theta E$, thus $\exists E_1 \in \rho^*(D'_1)$ such that $E_1 \approx_\theta E$ and $\exists E_2 \in \rho^*(D'_2)$ such that $E_2 \approx_\theta E$. Therefore there exist a single ρ-chain from D'_1 to E_1 and a single ρ-chain from D'_2 to E_2.

As a consequence, there are two distinct ρ-chains from C to clauses that are equivalent to E, one passing through D'_1 and one passing through D'_2. This leads to a contradiction for lemma 1.

3 Properties of the Aleph's `auto_refine/2` Operator

In this section we will discuss the properties of the refinement operator implemented by predicate `auto_refine/2` in Aleph's code [4].

The set G is defined by means of a number of *mode declarations* that are assertions of the form:

$$mode(PredicateMode).$$

$PredicateMode$ is of the form $p(ModeType, ModeType, \ldots)$ where $ModeType$ is either:

- $+a$: specifies that, when a literal with predicate symbol p appears in a hypothesized clause, the corresponding argument should be an "input" variable of type a,
- $-a$: specifies that the argument should be an "output" variable of type a,
- $\sharp a$: specifies that the argument should be a constant of type a

This directives ensures that, for a clause $H \leftarrow B_1, B_2, \ldots B_n$ to be in G, there must be an order of the literals in the body $B'_1, B'_2, \ldots B'_n$ such that:

- any input variable of type a in a literal B'_i appears as an output variable of type a in a literal B'_j with $j < i$, or appears as an input variable of type a in H
- any argument denoted by $\sharp a$ in the modes has only ground terms of type a

Besides the mode declarations, a specification of the G set contains also assertions of the following form

$$determination(q/n, p/m).$$

This assertion states that clauses can have predicate q/n in the head and that clauses with q/n in the head can have predicate p/m in the body. Let us call G_{modes} the set of clauses described by these declarations.

Let us now describe the `auto_refine/2` refinement operator. We will call it ρ_A. It is defined in the following way: given a clause C, obtain $C' \in \rho_A(C)$ by adding a literal L to C where

- each argument with mode $+a$ in L is substituted with an input variable in the head of type a or with an output variable in the literals already in the body of C of type a,
- each argument with mode $-a$ in L is substituted with an output variable in the head, with an input variable in the head, with an output variable in the literals already in C, with an output variable in L that precedes the argument (all of type a) or with a new variable, and
- each argument with mode $\sharp a$ in L is replaced with a constant of type a. Constants of type a can be obtained by running the query $a(X)$.

An equivalent definition of ρ_A is: given a clause C, obtain $C' \in \rho_A(C)$ by adding a literal L to C where

- each argument with mode $+a$ in L is substituted with any input variable of type a that appears in the head or with a variable that appears in literals in C excluding the head,
- each argument with mode $-a$ in L is substituted with any variable of C' of type a that comes before the argument (including the variables preceding the argument in L) or with a new variable
- each argument with mode $\sharp a$ in L is replaced with a constant of type a.

For example, consider the bias

$mode(q(+a, -a))$. $mode(p(+a, -a)$. $determination(q/2, p/2)$.

and consider the clause $C = q(X, Y) \leftarrow p(X, Z)$, the refinement operator adds the following literals[1]

$p(X, Y), p(X, X), p(X, Z), p(X, W), p(Z, Y), p(Z, X), p(Z, Z), p(Z, W)$

Let us now show that ρ_A is locally finite, not proper nor complete but weakly complete with respect to the quasi-ordered set $(G_{modes}, \geq_\theta)$. Moreover, ρ_A is also not optimal.

The local finiteness of ρ_A is evident from the fact that the number of possible assignment of variables is finite.

The fact that it is not proper can be seen from the example above: clause $C = q(X, Y) \leftarrow p(X, Z)$ is refined into $C' = q(X, Y) \leftarrow p(X, Z), p(Z, W)$ which

[1] The actual refinement operator tests whether the literal to be added is already literally present in C. If so, it does not add the literal. In this case it will not add $p(X, Z)$.

is θ-subsumption equivalent to C. This problem can be solved only if the operator can add more than one literal at a time to the clause.

The fact that ρ_A is not complete can be seen from this example: consider the two clauses

$D = q(X, Y, Z) \leftarrow p(X, Y, Z)$ $\;\; E = q(X, X, Z) \leftarrow p(X, X, Z)$

then $D \geq_\theta E$ but it is not possible to obtain a clause E' equivalent to E from D by means of the refinement operator ρ_A because the first and second arguments of $q(X, Y, Z)$ and $p(X, Y, Z)$ will never be unified.

However, ρ_A is weakly complete: given any clause $C \in G$ containing n literals, it can be obtained (up to renaming of variables) from the empty clause in n refinement steps of the ρ_A operator since at each step all the possible literals compatible with the type and mode declarations can be added.

That ρ_A is not optimal can be seen from the following example: consider the following language bias

$mode(q(+a, +a, -a))$. $mode(p(+a, -a))$. $determination(q/3, p/2)$.

and the clauses

$C_1 = q(X, Y, Z) \leftarrow p(X, Z)$

$C_2 = q(X, Y, Z) \leftarrow p(Y, Z)$

$F = q(X, Y, Z) \leftarrow p(X, Z), p(Y, Z)$

then $F \in \rho_A(C_1)$ and $F \in \rho_A(C_2)$ but $\nexists C_1' \in \rho_A^*(C_2)$ such that $C_1' \approx_\theta C_1$ and $\nexists C_2' \in \rho_A^*(C_1)$ such that $C_2' \approx_\theta C_2$. In other words, if we start refining from the empty clause, clause F will be visited at least twice, once coming from clause C_1 and once from clause C_2.

Note that ρ_A can be made complete by suitably extending the language bias and restricting the set G. In particular, if, for every type a, we add the mode declarations

$mode(+a = +a)$. $mode(+a = -a)$. $mode(-a = -a)$. $mode(-a = \sharp a)$.

$determination(target\_predicate/n, = /2)$.

then the problem seen before disappears: clause D can be refined into

$D' = q(X, Y, Z) \leftarrow p(X, Y, Z), X = Y$

that is equivalent to E once the substitution $X = Y$ is performed.

Moreover, for ρ_A to be made complete, we must also require that the clauses of G do not contain function symbols.

Lemma 2. *The refinement operator ρ_A with the extended language bias and without function symbols is complete.*

Proof. Let $C, D \in G$ such that $C >_\theta D$. Then there exist a θ such that $C\theta \subseteq D$. Let C be $\{C_1, C_2, \ldots, C_n\}$, let $C\theta$ be $\{C_1', C_2', \ldots, C_n'\}$ and let $D \setminus C\theta$ be $\{M_1, \ldots, M_m\}$.

For each C_k $k = 1, \ldots, n$, let us consider the restriction θ_k of θ to the variables of C_k. Then $C_k\theta_k = C_k'$.

By theorem 13.41 in [3] there is a finite set of *elementary substitutions* $\theta_1, \ldots, \theta_p$ such that $C_k\theta_1 \ldots \theta_p = C_k''$ is a variant of C_k'. Elementary substitutions for a literal E are substitutions of the following form

1. $\{Z/f(X_1, \ldots, X_q)\}$ where Z is a variable occurring in E, f is a functor symbol from the alphabet and $X_1, \ldots, X_q$ do not appear in E.
2. $\{Z/a\}$ where Z is a variable occurring in E and a is a constant.
3. $\{Z/X\}$ where Z and X are distinct variables occurring in E.

Since we consider only function free alphabets, only elementary substitution 2 and 3 are possible. Substitutions of type 2 can be performed by ρ_A because of the presence of $-a = \sharp a$. Substitutions of type 3 can be performed by ρ_A because of the presence of $+a = +a$, $+a = -a$ and $-a = -a$. Thus there exist a finite ρ_A-chain from C_k to C''_k and, from there, a finite ρ_A-chain to C'_k that consists just in renaming the necessary variables.

Since this is true for all the literals of C, there is a ρ_A-chain from C to $C\theta$.

Every input variable X in M_i is such that X appears in $C\theta \cup \{M_1, \ldots, M_{i-1}\}$ excluding the output variables in the head of $C\theta$. Therefore, M_i can be added by ρ_A and there is a ρ_A-chain from $C\theta$ to D. As a consequence there is a ρ_A-chain from C to D.

ρ_A is interesting for its simplicity and because it does not require the construction of a bottom clause. Thus the operator is useful when it is not possible to build a bottom clause, as in the learning from interpretation setting.

4 Acknowledgements

This work was partially funded by the IST programme of the EC, FET under the IST-2001-32530 SOCS project, within the Global Computing proactive initiative and by the Ministero dell'Istruzione, della Ricerca e dell'Università under the COFIN2003 project "La gestione e la negoziazione automatica dei diritti sulle opere dell'ingegno digitali: aspetti giuridici e informatici".

References

1. L. Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer-Verlag, 1999.
2. L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2–3):99–146, 1997.
3. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, Germany, 1997.
4. Ashwin Srinivasan. Aleph, 2004. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_toc.html.
5. P.R.J. van der Laag. *An analysis of refinement operators in inductive logic programming*. PhD thesis, Erasmus Universiteit, Rotterdam, the Netherlands, 1995.
6. P.R.J. van der Laag and S-H. Nienhuys-Cheng. Existence and nonexistence of complete refinement operators. In F. Bergadano and L. De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 307–322. Springer-Verlag, 1994.