CHAPTER IV
THE J-ALORITHM

Idestam-Almquist's technique to compute expansions can reduce most linear indirect roots to generalizations under θ-subsumption, but it only works when there are structural regularities called internal and/or external connections in the given clauses. Such structural regularities tell how appropriate linear expansions can be computed. However there are proper linear indirect roots of clauses for which it is not possible to find any appropriate linear expansions by his technique. Such linear indirect roots have a kind of structure he called cross connections.

The following definitions are derived from Idestam-Almquist [2].

Definition 1. A **position** of a term t in a simple expression E is a sequence of pairs defined as follows:

a) $<>$ is a position of t in E if $t = E$,

b) $<(f, i), p_1, p_2,..., p_m>$ is a position of t in E if E is a term $f(t_1, t_2,..., t_n)$ and $<p_1, p_2,..., p_m>$ is a position of t in t_i, where $1 \le i \le n$,

c) $<(q, i), p_1, p_2,..., p_m>$ is a position of t in E if E is a positive literal $q(t_1, t_2,..., t_n)$ and $<p_1, p_2,..., p_m>$ is a position of t in t_i, where $1 \le i \le n$, and

d) $<(\neg q, i), p_1, p_2,..., p_m>$ is a position of t in E if E is a negative literal $\neg q(t_1, t_2,..., t_n)$ and $<p_1, p_2,..., p_m>$ is a position of t in t_i, where $1 \le i \le n$.

We also say that $-p$ is a position of t in E if p is a position of E in t.

Definition 2. Let $p = <p_1, p_2,..., p_m>$ be a position. Then a position q is a **subposition** of p if and only if $q = <p_1, p_2,..., p_m, q_1, q_2,..., q_n>$.

Definition 3. Two literals are **compatible** if and only if they have the same predicate symbol and sign.

Definition 4. A pair of literals $(A, \neg B)$ is **ambivalent** if and only if A is compatible with B. A clause C is **ambivalent** if and only if there exist literals $A, \neg B \in C$ such that $(A, \neg B)$ is ambivalent.

Example. The clause $C = (\, p(a),\ q(b) \leftarrow p(\, f^{\,2}(a)))$ is ambivalent since $p(a)$ is compatible with $p(\, f^{\,2}(a))$, but C is not recursive because neither $p(a)$ nor $q(b)$ is unifiable with a variant of $p(f^{2}(a))$.

Definition 5. A pair of terms (s, t) is **ambivalent** in position p in an ambivalent pair of literals $(A, \neg B)$ if and only if s is found in a position p in A, and t is found in position p in B.

Example. Let $(A, \neg B) = (\, p(a, b), \neg\, p(\, f^{\,4}(a), d\,))$ be an ambivalent pair of literals. Then the pair of terms $(a, f^{4}(a))$ is ambivalent in position $\langle (\,p,1)\rangle$ in $(A, \neg B)$, and the pair of terms $(b, d\,)$ is ambivalent in position $\langle (\,p,2)\rangle$ in $(A, \neg B)$.

In the ambivalent pair of terms in this example, we can see that the first term is a subterm of the second term. We also have some structural regularities in the term $f^{4}(a)$.

Definition 6. Let $(A, \neg B)$ be an ambivalent pair of literals in a clause C, s a term in position p in A and t a term in position q in B. Then a sequence of terms $\boldsymbol{\kappa} = [s_0, s_1, \ldots, s_n\,]$ is a **cross connection** with structure π from s to t if and only if:

a) $s_0 = s$ and $s_n = t$,

b) p is not a subposition of q nor is q a subposition of p, and

c) $\pi = [(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)]$ is a sequence of pairs of positions such that for each $0 \leq i \leq (n-1)$ there is a literal $L_i \in C$ such that s_i is found in position p_{i+1} in L_i and s_{i+1} is found in position q_{i+1} in L_i.

Example. Consider the following clauses:

$$C = (p(x, u) \leftarrow q(x, y), r(u, v), p(v, y)),$$
$$D = (p(x, u) \leftarrow q(v, w), r(y, z), p(k, l)), \text{ and}$$
$$E = (p(x, u) \leftarrow q(x, y), r(y, z), r(u, v), q(v, w), p(z, w)).$$

In the clause C there is a cross connection $\kappa_1 = [x, y]$ with structure $\pi_1 = [(\langle(\neg q,1)\rangle,\langle(\neg q,2)\rangle)]$ from the term x in the literal $p(x, u)$ to the term y in the literal $\neg p(v, y)$, and a cross connection $\kappa_2 = [u, v]$ with structure $\pi_2 = [(\langle(\neg r,1)\rangle,\langle(\neg r,2)\rangle)]$ from the term u in the literal $p(x, u)$ to the term v in the literal $\neg p(v, y)$.The cross connections in C have disappeared in E. We have that both clauses C and D are generalizations under implication of the the clause E.

Definition 7. Let T be a set of clauses. Then, the **n^{th} linear resolution** of T, denoted $\mathcal{L}^n(T)$, is defined as:

a) $\mathcal{L}^1(T) = T$, and

b) $\mathcal{L}^n(T) = \mathcal{L}^{n-1}(T) \cup \{R \mid C \in T, D \in \mathcal{L}^{n-1}(T)$ and R is a resolvent of C and $D\}$ $(n > 1)$.

Definition 8. A clause D is an **n^{th} power** of a clause C if and only if D is a variant of a clause in $\mathcal{L}^n(\{C\})$ $(n \geq 1)$. We also say that C is an **n^{th} root** of D.

Definition 9. A clause D is an **indirect n^{th} power** of a clause C if and only if there exists a clause E such that $E \prec D$ and E is an n^{th} power of C. We also say that C is an **indirect n^{th} root** of D.

Example. Consider the following clauses:

$C = (p(x) \leftarrow p(f(x)))$,

$D = (p(x) \leftarrow p(f^2(x)))$,

$E = (p(x) \leftarrow p(f^3(x)))$,

$F = (p(a) \leftarrow p(f^2(a)), p(a))$, and

$G = (p(x) \leftarrow p(a))$.

The clause C is a second root of D, and a third root of E. The clause C is also an indirect second root of F, since C is a second root of D and D θ-subsumes F. The clause G is an indirect n^{th} root of itself for every $n \geq 1$. The clause G is also an indirect first root of F.

Example. Consider the following clauses:

$C = (p(x) \leftarrow p(f(x)), p(g(x)))$,

$D = (p(z) \leftarrow p(f^3(z)), p(gf^2(z)), p(gf(z)), p(g(z)))$, and

$E = (p(a) \leftarrow p(f^3(a)), p(gf(a)), p(gf^2(a)), p(g(a)))$.

The clause C is a third root of D. The clause C is also a indirect third root of E.

The following proposition shows the relation of predicate, function, and constant symbol between two clauses in which one clause implies the other. While it is not used directly in what follows, it does help to motivate the J-algorithm.

Proposition 10. Let A and B be Horn clauses such that B is nonvalid and $A \Rightarrow B$. Then every predicate, function, and constant symbol occurring in A must also occur in B.

Proof. Let I be an interpretation such that B is false with respect to I.

Step I. We must show that there is no predicate symbol which occurs in A but not in B. Suppose that there is a predicate symbol p which occurs in A but not in B. First, suppose that p occurs in the body of A. Let $I' = I \setminus \{ p(t) \mid t \text{ is a ground term}\}$ (i.e., I' is I modified so that p is always false in I'). Then A is true with respect to I', but B is still false with respect to I' , because p does not occur in B. This says I' is a model for A but not a model for B, contrary to $A \Rightarrow B$. Thus, p cannot occur in the body of A.

Hence we must have that p occurs in the head of A. let $I'' = I \cup \{ p(t) \mid t$ is a ground term$\}$ (i.e., I'' is I modified so that p is always true in I''). As above, we have that A is true with respect to I'' but B is false with respect to I'', contrary to $A \Rightarrow B$. Thus, p cannot occur in the head of A, either. This shows there is no predicate symbol which occurs in A but not in B.

<u>Step II</u>. We must show that there is no function or constant symbol which occurs in A but not in B. Since constant symbols are just function symbols of arity 0, we really only need to consider the case of function symbols. Thus, let f be a function symbol which occurs in A. We must show that f also occurs in B. Since $A \Rightarrow B$, $A \vdash_\theta B$. Thus, there is a sequence of Horn clauses $A_1, A_2, \ldots, A_n$ such that for each $i \in \{1, 2, \ldots, n\}$ either

1) $A_i = A$, or

2) there exist $j, k < i$ such that A_i is a resolvent of A_j and A_k,

and there is a substitution θ such that $A_n\theta \subseteq B$.

We must show that for each $i \in \{1, 2, \ldots, n\}$ the symbol f occurs in A_i. We do this by induction on I.

<u>Case I</u>. f occurs in the head of A. We will show f occurs in the head of A_i for all $i \in \{1, 2, \ldots, n\}$.

<u>Basis Step</u>. $i = 1$. Then $A_i = A_1 = A$. Then f occurs in the head of A_i, because f occurs in the head of A.

Induction Step. Assume that f occurs in the head of A_l for all $l \in \{1, 2,..., k\}$ where $k < i$. The case $A_i = A$ is the same as the Basis Step, so we may assume A_i is a resolvent of A_j and A_k where $j, k < i$. By the induction hypothesis, f occurs in the head of A_j and A_k. By resolution, f occurs in the head of A_i.

Case II. f only occurs in the body of A. We will show f occurs only in the body of A_i for all $i \in \{1, 2,..., n\}$.

Basis Step. $i = 1$. Then $A_i = A_1 = A$. Since f only occurs in the body of A, f only occurs in the body of A_i.

Induction Step. Assume that f only occurs in the body of A_l for all $l \in \{1, 2,..., k\}$ where $k < i$. The case $A_i = A$ is the same as the Basis Step, so we may assume A_i is a resolvent of A_j and A_k where $j, k < i$. By the induction hypothesis, f only occurs in the body of A_j and A_k. By resolution, f only occurs in the body of A_i.

In particular, f occurs in A_n. Now, the substitution θ only changes variables in A_n, not function symbols, so f also occurs in $A_n\theta$. Since $A_n\theta \subseteq B$, this shows f must occur in B as well. Thus, every function symbol occurring in A also occurs in B. □

There is no algorithm to compute indirect roots of some clauses that contain cross connections. Since we are interested in finding them, we create the following algorithm, called the *J*-algorithm which finds indirect roots of clauses, even if they contain cross connections.

J-Algorithm

The J-algorithm is defined in 5 steps:

1. Input the Horn clause D.

2. Consider predicates in the clause D, and create a new clause C such that C has the same positive and negative predicates as clause D, but no negative predicate is repeated.

3. Change the terms in the negative predicates in C to new variables. Let $C_1 = C$, and $C_2 = C$.

4. Resolve the clause C_1 with C_2 to get a clause C^*. When resolving, always keep the original variables in C_1, and introduce new variables into C_2 to make the variables in C_2 disjoint from those in C_1. Also, when unifying, never replace a variable in C_1 with one of the new variables introduced into C_2.

5. Choose one predicate which occurs only as a negative predicate, and let n be the number of times that predicate occurs in C^*, and m the number of times it occurs in D.

If $n = m$, then

if we can obtain D from C^* by substituting for some variables which do not occur in C, then finish, with the output C,

else find a substitution θ such that $C^*\theta = D$, replace C with $C\theta$, let $C_1 = C$, $C_2 = C$, and go back to step 4,

else let $C_1 = C$ and $C_2 = C^*$, and go back to step 4. □

Proposition 11. Let D be a Horn clause and C the Horn clause which is the output from the *J*-algorithm when the input is D. Then C is a generalization under implication of clause D and an indirect root of D.

Proof. To show that C is a generalization under implication of D, it suffices to show $C \vdash_\theta D$. Let k be the number of times Step 4 is executed, and for each $i \in \{1, 2,..., k\}$, let C_i^* be the value of C^* after executing Step 4 the i^{th} time, and let C_i be the value of C after executing Step 4 the i^{th} time.

Note that the output $C = C_k$, and C_k^* θ-subsumes D. Let p be the smallest member of $\{1, 2,..., k\}$ such that $C_p = C_k$. Then $C_p = C$ also, so it suffices to show that

$C_p \vdash_\theta D$. By the choice of p, when we execute Step 4 the p^{th} time, the clause C_1 and C_2 in the algorithm are both equal to C_p. Thus, C_p^* follows from C_p and C_p by resolution, so $C_p \vdash_\theta C_p^*$.

Let us show that $\{C_p, C_p^*\} \vdash_\theta D$. Assume for a contradiction that $\{C_p, C_p^*\} \nvdash_\theta D$, and let $q \in \{p, p+1, ..., k\}$ be the largest number such that $\{C_p, C_q^*\} \nvdash_\theta D$.

Since C_k^* θ-subsumes D, $C_k^* \vdash_\theta D$, so certainly $\{C_p, C_k^*\} \vdash_\theta D$, and thus $q \neq k$. Since $q \leq k$, we must have $q < k$. Consider C_{q+1}^*. By the choice of q, $\{C_p, C_{q+1}^*\} \vdash_\theta D$.

We need to show that $\{C_p, C_q^*\} \vdash_\theta C_{q+1}^*$. From the *J*-algorithm, let m be the number of times that a predicate only occurring as a negative predicate occurs in D and n the number of times it occurs in C_q^*.

<u>Case $m = n$</u>. Since $q < k$, it must be true that there is a substitution θ such that $C_q^*\theta = D$, and that we set $C = C_q\theta$. Then $C_{q+1} = C_q\theta \neq C_q$, contrary to our choice of p and the fact that $q \geq p$. Thus, this case cannot occur.

<u>Case $m \neq n$</u>. In this case the clause C_1 in the *J*-algorithm is set equal to C_q and C_2 is set equal to C_q^* before going back to execute Step 4 the $p+1^{st}$ time. Thus, C_{q+1}^* follows from C_q and C_q^* by resolution. But $C_q = C_p$, so C_{q+1}^* follows from C_p and C_q^* by resolution. Hence, $\{C_p, C_q^*\} \vdash_\theta C_{q+1}^*$.

In each case we have that $\{C_p, C_q^*\} \vdash_\theta C_{q+1}^*$. Since $\{C_p, C_q^*\} \vdash_\theta C_{q+1}^*$ and $\{C_p, C_q^*\} \vdash_\theta C_p$, by Corollary 11 in chapter III $\{C_p, C_q^*\} \vdash_\theta D$, a contradiction. Thus, we must have that $\{C_p, C_p^*\} \vdash_\theta D$.

Now, we note $C_p \vdash_\theta C_p$ and $C_p \vdash_\theta C_p^*$. Thus, $C_p \vdash_\theta D$.

To see that C is an indirect root of D, note that when we do resolution in Step 4 of the J-algorithm, the clause C_1 is always equal to C, so $C_k^* \in \mathrm{L}^{k-p}(\{C\})$, and $C_k^* \prec D$. □

The following example shows how the J-algorithm can find an indirect root of a clause that contains a cross connection.

Example. Let $D = p(x, y) \leftarrow q(x, f(z)), q(y, f(w)), p(f(z), f(w))$, and note that D contains a cross connection. Let us find an indirect root C of D using the J-algorithm.

We start with $C = p(x, y) \leftarrow q(m, n), p(k, l)$, and let $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$C_1 = p(x, y) \leftarrow q(m, n), p(k, l)$, and

$C_2 = p(x', y') \leftarrow q(m', n'), p(k', l')$.

♦ Unify C_2 with $\{x'/k, y'/l\}$,

$C_1 = p(x, y) \leftarrow q(m, n), p(k, l)$, and

$C_2 = p(k, l) \leftarrow q(m', n'), p(k', l')$.

♦ Resolve on $p(k, l)$, we get

$C_s = p(x, y) \leftarrow q(m, n), q(m', n'), p(k', l')$.

♦ There are two ways to substitute for m and n that will allow C_s to match D: $\{m/x, n/f(z)\}$, or $\{m/y, n/f(w)\}$.

Case I. $\{m/x, n/f(z)\}$. Then let $C = p(x, y) \leftarrow q(x, f(z)), p(k, l))$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$C_1 = p(x, y) \leftarrow q(x, f(z)), p(k, l)$, and

$C_2 = p(x', y') \leftarrow q(x', f(z')), p(k', l')$.

♦ Unify C_2 with $\{x'/k, y'/l\}$,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(k, l), \text{ and}$$

$$C_2 = p(k, l) \leftarrow q(k, f(z')), p(k', l').$$

♦ Resolve on $p(k, l)$, we get

$$C_s = p(x, y) \leftarrow q(x, f(z)), q(k, f(z')), p(k', l').$$

♦ To allow C_s to match D we must make the substitution $\{k/y\}$.

So, let $C = p(x, y) \leftarrow q(x, f(z)), p(y, l))$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(y, l), \text{ and}$$

$$C_2 = p(x', y') \leftarrow q(x', f(z')), p(y', l').$$

♦ Unify C_2 with $\{x'/y, y'/l\}$,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(y, l), \text{ and}$$

$$C_2 = p(y, l) \leftarrow q(y, f(z')), p(l, l').$$

♦ Resolve on $p(y, l)$, we get

$$C_s = p(x, y) \leftarrow q(x, f(z)), q(y, f(z')), p(l, l').$$

♦ This shows us we need to make the substitution $\{l/f(z)\}$.

Hence, let $C = p(x, y) \leftarrow q(x, f(z)), p(y, f(z)))$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(y, f(z)), \text{ and}$$

$$C_2 = p(x', y') \leftarrow q(x', f(z')), p(y', f(z')).$$

♦ Unify C_2 with $\{x'/y, y'/f(z)\}$,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(y, f(z)), \text{ and}$$

$$C_2 = p(y, f(z)) \leftarrow q(y, f(z')), p(f(z), f(z')).$$

♦ Resolve on $p(y, f(z))$, we get

$$C_s = p(x, y) \leftarrow q(x, f(z)), q(y, f(z')), p(f(z), f(z')).$$

♦ The substitution $\{z'/w\}$ makes C_s match D. But z' is not a variable in C, so we are finished, with $C = p(x, y) \leftarrow q(x, f(z)), p(y, f(z))$.

Case II. $\{m/y,\ n/f(w)\}$. Then let $C = p(x, y) \leftarrow q(y, f(w)), p(k, l)$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(k, l), \text{ and}$$

$$C_2 = p(x', y') \leftarrow q(y', f(w')), p(k', l').$$

♦ Unify C_2 with $\{x'/k, y'/l\}$,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(k, l), \text{ and}$$

$$C_2 = p(k, l) \leftarrow q(l, f(w')), p(k', l').$$

♦ Resolve on $p(k, l)$, we get

$$C_s = p(x, y) \leftarrow q(y, f(w)), q(l, f(w')), p(k', l').$$

♦ This shows we need to make the substitution $\{l/x\}$.

So, let $C = p(x, y) \leftarrow q(y, f(w)), p(k, x)$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(k, x), \text{ and}$$

$$C_2 = p(x', y') \leftarrow q(y', f(w')), p(k', x').$$

♦ Unify C_2 with $\{x'/k, y'/x\}$,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(k, x), \text{ and}$$

$$C_2 = p(k, x) \leftarrow q(x, f(z')), p(k', k).$$

♦ Resolve on $p(k, x)$, we get

$$C_s = p(x, y) \leftarrow q(y, f(w)), q(x, f(z')), p(k', k).$$

♦ Now we need the substitution $\{k/f(w)\}$.

Hence, let $C = p(x, y) \leftarrow q(y, f(w)), p(f(w), x)$, and $C_1 = C_2 = C$.

♦ Resolve C_1 with C_2,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(f(w), x), \text{ and}$$

$$C_2 = p(x', y') \leftarrow q(y', f(w')), p(f(w'), x').$$

♦ Unify C_2 with $\{x'/f(w), y'/x\}$,

$$C_1 = p(x, y) \leftarrow q(y, f(w)), p(f(w), x), \text{ and}$$

$$C_2 = p(f(w), x) \leftarrow q(x, f(w')), p(f(w'), f(w)).$$

♦ Resolve on $p(f(w), x)$, we get

$$C_s = p(x, y) \leftarrow q(y, f(w)), q(x, f(w')), p(f(w'), f(w)).$$

♦ Now the substitution $\{w'/z\}$ makes C_s equal to D. Again, w' does not appear in C, so we are finished, with $C = p(x, y) \leftarrow q(y, f(w)), p(f(w), x)$ this time.

Consequently, from $D = p(x, y) \leftarrow q(x, f(z)), q(y, f(w)), p(f(z), f(w))$, we get two indirect roots from the J-algorithm,

$$C_1 = p(x, y) \leftarrow q(x, f(z)), p(y, f(z)), \text{ and}$$

$$C_2 = p(x, y) \leftarrow q(y, f(w)), p(f(w), x).$$

Note that C_1 and C_2 themselves contain cross connections.

Concluding Remarks

With the J-algorithm we can compute indirect roots of some clauses whose roots could not be computed before, because they contained cross connections. However, it is not clear that the J-algorithm can find indirect roots of all clauses. As a result, further study is required, which may lead to an enhanced algorithm.