A Completeness Result for Linked Resolution

Kenneth Kunen *
University of Wisconsin, Madison, WI 53706, U.S.A.
kunen@math.wisc.edu

September 9, 1996

1 Introduction

The notion of linked resolution was suggested by Wos, Veroff, Smith, and McCune [5]; see [4] for a detailed discussion. It is a very general idea designed to overcome a common problem in resolution-based provers: namely, that the database gets clogged up with many "uninteresting" clauses, which, however, cannot simply be discarded, since they may be needed to derive an "interesting" clause. Instead, the linking paradigm allows us to do several proof steps until an "interesting" clause is derived and then keep that clause and discard the intermediate results.

There are many possible ways to implement this general idea, depending on the strategy used to tell the computer which clauses are "interesting". A version of linked UR (unit-resulting) resolution is implemented in OTTER [2]. In this paper, we suggest another possible implementation, which is not tied to unit clauses. A special case of this is implementation is hyper-resolution; (positive) hyper-resolution amounts to the declaration that positive clauses are "interesting"; then, one hyper-resolution step may be viewed as a sequence of binary resolution steps in which we keep only the final (positive) product and discard the intermediate results.

*Author supported by NSF Grant CCR-9503445.

In the case of predicate logic *without* equality, we prove a completeness theorem for our implementation; this may be considered to be a generalization of the completeness of hyper-resolution. When equality and paramodulation [3] are added, we no longer have a completeness result, although we provide an example to show where our suggested implementation might still be useful.

The following is a simple example:

1. $O(x) \vee E(x)$
2. $\neg O(x) \vee \neg E(x)$
3. $\neg E(x) \vee O(s(x))$
4. $\neg O(x) \vee E(s(x))$
5. $\neg E(c)$
6. $E(s(s(c)))$

Clauses (1) and (2), taken together, say that everything (in the intended domain of discourse) is either odd or even, but not both. They each get used once in the derivation of the empty clause, which takes 5 steps in ordinary binary resolution. One defect of binary resolution is that, in an automated search for a derivation, (1,2) will serve to generate, from each clause of the form $\neg O(\tau) \vee \varphi$, an un-interesting variant $E(\tau) \vee \varphi$; likewise for occurrences of literals $O(\tau), \neg E(\tau), E(\tau)$.

Instead, one could declare to the prover that (1,2) are to be used as *pure linking clauses*. Formal definitions are in Section 2, but informally, a pure linking clause is only allowed to be used as a nucleus in a step like a hyper-resolution clash which consumes all the literals in the nucleus. One then derives the empty clause in 3 steps. In the first, (1) is used as a nucleus with satellites (5,4):

$$\begin{array}{ccccc} \neg E(c) & & & & \\ | & & & & \\ E(x) & \vee & O(x) & & \\ & & | & & \\ & & \neg O(x1) & \vee & E(s(x1)) \end{array}$$

which, after unification, results in $E(s(c))$. This yields $O(s(s(c)))$ by using binary resolution with (3). One then derives the empty clause by using linked

resolution with (2) as a nucleus and $O(s(s(c)))$ and (6) as satellites:

$$\begin{array}{ccccc} O(s(s(c))) & & & & \\ | & & & & \\ \neg O(x) & \vee & \neg E(x) & & \\ & & | & & \\ & & E(s(s(c))) & & \end{array}$$

In this case, declaring that (1,2) are to be used only as pure linking clauses, as in the figures shown, is roughly the same as saying that we are looking for a refutation from (3-6) which treats E and $\neg O$ as synonyms, rather than generating all variants obtained by replacing E by $\neg O$, O by $\neg E$, $\neg E$ by O, or $\neg O$ by E. Using this idea, one may verify that in this particular case, such a declaration is complete; that is, if we replace $(3, 4 \ldots 6)$ with another set of clauses, $(3, 4 \ldots N)$, and the set $(1, 2 \ldots N)$ is inconsistent, then we may find a derivation of the empty clause treating (1,2) as pure linking clauses.

However, this example is very special, and the outlined proof of completeness is quite ad hoc. In the next section, we prove a theorem which applies to arbitrary such declarations. Some care must be taken to state the theorem correctly; for example, if all clauses were declared to be linking clauses, we could not derive anything. In the general case, completeness requires us to allow also binary resolution among the linking clauses, generating new linking clauses. Then, in the special case above, we just observe that the only possible resolutions among (1,2) generate tautologies, which may be discarded. The completeness of hyper-resolution will be another special case of our completeness theorem; that should not be surprising, since our linking rule is like a hyper-resolution clash, except that there is no restriction on the sign of the literals involved.

It is fair to ask why completeness results are of interest in this area. In practice, when using an automated reasoning system, the set of rules one is *really* using is not complete, since one usually employs some arbitrary cutoff on clause length or term complexity to avoid the combinatorial explosion. One may view the real implementation as an approximation of a complete deductive system. Completeness results are of interest because they show that when the prover fails to find a refutation after exhausting all possibilities, the fault lies with the cutoffs (or else the clause set is consistent), not with the basic rules of inference; one may then try to modify the cutoffs in the hope of obtaining a refutation.

In the specific case of linked resolution, it is hard to say exactly what kind of completeness result would be interesting. The full unrestricted definition of linked resolution [4] is trivially complete, and in fact any proof in standard resolution may be replaced by just one linked resolution step. In practice, it would not be feasible to search for such proofs; in fact, as is clear from examples in [4], for a search to be successful, there must be very stringent bounds placed on the complexity of the linked resolution steps considered. In [4], there is a discussion of various strategies for limiting the search. The actual completeness result prove below is related to what they call the Partitioning Strategy. Perhaps a further examination of [4] might suggest other completeness results as well.

2 Basic Results

An *ordinary clause* is a set of 0 or more literals. If α is a literal, we identify $\neg\neg\alpha$ with α. We use $\square$ for the empty clause. If φ, ψ are ordinary clauses, we shall usually denote their union, $\varphi \cup \psi$, as a disjunction, $\varphi \vee \psi$ (which is exactly the same as $\psi \vee \varphi$). A *linking clause* is an entity of the form $(\varphi \parallel \psi)$, where φ and ψ are ordinary clauses. Semantically, $(\varphi \parallel \psi)$ means the same as $\varphi \vee \psi$, but the $\parallel$ signifies how the clause is to be used in linked resolution. Informally, the linking is to be done on the left of the $\parallel$; the clause on the right is simply carried along. A *pure linking clause* is of the form $(\varphi \parallel \square)$. Our declaration that $O(x) \vee E(x)$ is a pure linking clause in the above example means that in our formal syntax, we write it as $(O(x) \vee E(x) \parallel \square)$. A *pure standard clause* is of the form $(\square \parallel \varphi)$. In the above example, all clauses φ other than the ones declared linking are now written as $(\square \parallel \varphi)$.

We use letters α, β for literals, $\varphi, \psi, \xi, \zeta$ for ordinary clauses, λ for linking clauses, and σ for substitutions.

As usual in resolution, we rename each clause to have new variables before attempting unification. To more easily express our rules, we use the following notation. If S is a set of linking clauses, and λ is a linking clause, we say $\lambda \tilde{\in} S$ iff λ is a renaming of a clause in S.

We now elaborate our proof rules. We phrase them all as operations which, applied to the linking clauses in a set S, construct a new linking clause to add to S. Formally, we are specifying which linking clauses are

immediate consequences of S. The first rule will be the linking rule, but others are needed for completeness. If S is a set of pure linking clauses, then the linking rule can never apply; in this case, completeness will follow from the fact that we include as rules standard binary resolution and factoring to the left of the $\|$. Likewise, if S is a set of pure standard clauses, completeness will follow from the fact that we include as rules standard binary resolution and factoring to the right of the $\|$.

1. Linking Rule: If $(\alpha_1 \vee \cdots \vee \alpha_n \parallel \varphi)\widetilde{\in}S$, and $(\square \parallel \neg\beta_i \vee \psi_i)\widetilde{\in}S$, for each $i = 1 \ldots n$, where all these $n+1$ clauses have disjoint variables, and σ is a most general substitution which unifies each α_i with β_i $(i = 1 \ldots n)$, then $(\square \parallel \psi_1 \vee \cdots \vee \psi_n \vee \varphi)\sigma$ is an immediate consequence of S. We call $(\alpha_1 \vee \cdots \vee \alpha_n \parallel \varphi)$ the *nucleus* and the $(\square \parallel \neg\beta_i \vee \psi_i)$ the *satellites*.

2. Left Resolution: Ordinary binary resolution to the left of the $\|$. That is, if $(\alpha \vee \varphi \parallel \psi)\widetilde{\in}S$ and $(\neg\beta \vee \xi \parallel \zeta)\widetilde{\in}S$, these two clauses have disjoint variables, and σ is a most general unifier of α and β, then $(\varphi \vee \xi \parallel \psi \vee \zeta)\sigma$ is an immediate consequence of S.

3. Left Factoring: Factoring to the left of the $\|$. That is, if $(\alpha \vee \beta \vee \varphi \parallel \psi) \in S$ and σ is a most general unifier of α and β, then $(\beta \vee \varphi \parallel \psi)\sigma$ is an immediate consequence of S.

4. Right Resolution: Ordinary binary resolution to the right of the $\|$.

5. Right Factoring: Factoring to the right of the $\|$.

Hyper-resolution is a special case of linked resolution. For example, positive hyper-resolution is equivalent to linked resolution in the case that all clauses are of the form $(\varphi \parallel \psi)$, with φ purely negative and ψ purely positive; in this case, left resolution and right resolution are impossible.

We say $S \vdash \lambda$ iff there is a sequence $\lambda_1 \ldots \lambda_n$, where λ_n is λ, and, for each i, either $\lambda_i \in S$ or λ_i is an immediate consequence of $\{\lambda_j : j < i\}$ by one of Rules (1-5). Soundness (if $S \vdash (\square \parallel \square)$ then S is inconsistent) is obvious.

Theorem 2.1 (Completeness) *If S is inconsistent then $S \vdash (\square \parallel \square)$.*

Before the proof, a remark on factoring. Since a clause here is a *set* of literals, factoring cannot occur in ground derivations; that is, $(\alpha \vee \alpha \vee \varphi \parallel \psi)$ and $(\alpha \vee \varphi \parallel \psi)$ are the same clause. In lifting proofs, such as ours, factoring steps in the lifted derivation correspond to removing a duplicate in the ground derivation. Note that for completeness, we never need to factor across the $\|$; that is $(p \parallel p)$ has neither $(\square \parallel p)$ nor $(p \parallel \square)$ as an immediate consequence.

Let $size(\varphi \parallel \psi)$ be the sum of the numbers of literals in φ and in ψ. If S is a finite set of linking clauses, $size(S)$ is sum of all $size(\lambda)$ for $\lambda \in S$. The proof will be by induction on $size(S)$; equivalently, one may induct on the excess literal parameter of Anderson and Bledsoe [1]. The induction step will use:

Lemma 2.2 *If $S \cup \{(\varphi \parallel \psi)\} \vdash (\square \parallel \square)$, and α is a ground literal, then either $S \cup \{(\varphi \parallel \alpha \vee \psi)\} \vdash (\square \parallel \alpha)$ or $S \cup \{(\varphi \parallel \alpha \vee \psi)\} \vdash (\square \parallel \square)$.*

Proof. Given the original derivation, just tack on an α each time that $(\varphi \parallel \psi)$ gets used, resulting in a derivation of $(\square \parallel \alpha)$ or $(\square \parallel \square)$. ∎

We remark that in the "usual" case here, we would get a derivation of $(\square \parallel \alpha)$, but a derivation of $(\square \parallel \square)$ is also possible; for example, this would happen if $(\varphi \parallel \psi)$ was not used at all, or if $\alpha \vee \psi$ is the same as ψ (if α is already a literal of ψ).

In the proof of Theorem 2.1, the basis of the induction will use the following well-known completeness fact from ordinary resolution:

Lemma 2.3 *If S is a set of ordinary propositional clauses, ξ is an ordinary propositional clause which is not a tautology, and ξ is logical consequence of S, then by ordinary binary resolution, one can derive from S some clause ξ' which subsumes ξ.*

Proof of Theorem 2.1. By the usual lifting and compactness argument, if the Theorem fails then it fails for some S which is finite and propositional, so we consider only such S. We induct on $size(S)$; that is, we assume that S is inconsistent, and that the Theorem holds for all S' of smaller $size$.

First, suppose that S contains a linking clause of the form $(\varphi \parallel \alpha \vee \psi)$, where either φ or ψ is non-empty. Let S' be the other linking clauses in S. Let $S_1 = S' \cup \{(\varphi \parallel \psi)\}$ and $S_2 = S' \cup \{(\square \parallel \alpha)\}$. Then both S_1 and S_2 have smaller $size$ than S, and are both inconsistent, so the induction hypothesis applies to them. By $S_1 \vdash (\square \parallel \square)$ and Lemma 2.2, either $S \vdash (\square \parallel \square)$ (so we are done immediately), or $S \vdash (\square \parallel \alpha)$, in which case, $S_2 \vdash (\square \parallel \square)$ yields $S \vdash (\square \parallel \square)$.

The *base case* of the proof occurs when a reduction as in the previous paragraph cannot be done. In this case, we may partition S into $S_1 \cup S_2$, where elements of S_1 are of the form $(\varphi \parallel \square)$ and elements of S_2 are of the

form $(\Box \parallel \alpha)$, where α is a literal. Let ξ be the disjunction of all $\neg\alpha$ such that $(\Box \parallel \alpha) \in S_2$. Then ξ is a logical consequence of S_1, since a truth assignment which satisfied S_1 and $\neg\xi$ would satisfy S. If ξ is a tautology, then we can derive $(\Box \parallel \Box)$ from S_2 in one binary resolution step. Otherwise, by Lemma 2.3, we may, from S_1, derive $(\xi' \parallel \Box)$ for some ξ' which subsumes ξ; we then get $(\Box \parallel \Box)$ by one use of the linking rule. ∎

We remark that there is no upper bound to $size(S)$ in the base case. We must have $size(S) \geq 2$ if S is to be inconsistent. An example of the base case would be where S_2 is $\{(\Box \parallel \neg p),\ (\Box \parallel \neg q)\}$ and S_1 is $\{(p \vee r \parallel \Box),\ (q \vee \neg r \parallel \Box)\}$. Then we use left resolution from S_1 to derive $(p \vee q \parallel \Box)$, from whence we get $(\Box \parallel \Box)$ by one use of the linking rule with S_1.

One problem with the rules as stated is that one is sometimes forced to maintain tautologies in the database. For example, from $(\Box \parallel p \vee \neg p)$ and $(p \parallel \psi)$, the linking rule gives $(\Box \parallel p \vee \psi)$; without the tautology, there might be no way of bringing the p across the $\parallel$. More concretely, say we start with just $(p \parallel p)$ and $(\neg p \parallel \neg p)$. We can get $(\Box \parallel p \vee \neg p)$ by left resolution; then two uses of linking yield $(\Box \parallel p)$ and $(\Box \parallel \neg p)$, whence we get $(\Box \parallel \Box)$. One cannot derive $(\Box \parallel \Box)$ without generating a tautology. In this case, one may eliminate the tautology by adding a rule for factoring across the $\parallel$, but in general this is not possible. For example, if one starts with $(p \parallel q)$, $(\neg p \parallel \neg q)$, $(\neg q \parallel p)$, and $(q \parallel \neg p)$, then the only first steps possible are ones which generates tautologies.

Left tautologies (that is, $(\varphi \parallel \psi)$, containing an α and $\neg\alpha$, inside φ) can safely be eliminated. Formally, one proves, by induction on the length of the derivation, that if $S \vdash (\xi \parallel \zeta)$, then there is a derivation of some $(\xi' \parallel \zeta')$ using no left tautologies, where ξ' subsumes ξ and ζ' subsumes ζ. Note that whenever a left tautology gets used as a nucleus, one can always replace this use by a right resolution between two of the satellites. Hence,

Corollary 2.4 *If S is inconsistent then $S \vdash (\Box \parallel \Box)$ using a derivation containing no left tautologies.*

Probably, an implementation should make the keeping of tautologies optional, in the same spirit in which factoring in ordinary resolution is made optional in OTTER [2], even though it is necessary for completeness.

3 Examples

1. There is a class of examples, such as the one in the Introduction, in which left factoring and left resolution cannot apply, so that only pure standard clauses get added during the derivation.

1a. As a sub-class of these, we have the situation where each predicate letter which occurs to the left of the $\|$ in a linking clause always does so with the same sign, so that this is really just a variant of hyper-resolution. For example, we could express the transitivity of an order relation by:

$$x \not< y \vee y \not< z \parallel x < z \quad .$$

Or, one could also add irreflexivity:

$$x \not< x \parallel \square \quad .$$

All other clauses would be pure standard clauses, of form $\square \parallel \varphi$. Then application of the proof rule only generates new pure standard clauses. Thus, as in hyper-resolution we prevent transitivity from resolving on itself and creating longer and longer clauses of the form $w \not< x \vee x \not< y \vee y \not< z \vee w < z$. Likewise, in the example in the Introduction, if we only used (1), this would be a variant of positive hyper-resolution, and if we only used (2), this would be a variant of negative hyper-resolution. However, if we use both, then hyper-resolution does not apply.

1b. In the example in the Introduction, left resolution does actually create two new linking clauses, $(E(x) \vee \neg E(x) \parallel \square)$ and $(O(x) \vee \neg O(x) \parallel \square)$. However, these tautologies are left tautologies, so they can be discarded.

1c. Instead of O and E, we could have 3 mutually exclusive categories:

$$\begin{aligned} M(x) \vee F(x) \vee N(x) &\parallel \square \\ \neg M(x) \vee \neg F(x) &\parallel \square \\ \neg F(x) \vee \neg N(x) &\parallel \square \\ \neg N(x) \vee \neg M(x) &\parallel \square \end{aligned}$$

Again, if the other clauses are all pure standard, then only standard clauses get generated, except for some left tautologies, which can be discarded.

2. In the general case, given a set of ordinary clauses from which we wish to derive a contradiction, we may insert a $\|$ inside each one arbitrarily, and the derivation may well produce new pure linking clauses as well as new pure

standard clauses. It may require some insight to determine a good place to put the $\|$ to speed up the search for a derivation. Thus, this may be viewed as one more tool in the theorem prover's bag of tricks (see [6] and [7] for a discussion of others), to be employed in those cases where it seems to be useful.

The following very simple example illustrates the fact that linked resolution even in the restricted sense described in this paper may result in a shorter proof than does binary resolution, positive or negative hyper-resolution, or UR resolution. It also illustrates the necessity (for completeness) of allowing resolutions among the linking clauses; of course, as with other options, one might make such resolutions an option to be set by the user.

1.	$x \not< x$	$\|$	$\square$
2.	$x \not< y \vee y \not< z \vee x < z$	$\|$	$\square$
3.	$\square$	$\|$	$p < q \vee a < b$
4.	$\square$	$\|$	$q < p \vee a < b$
5.	$\square$	$\|$	$p < q \vee b < a$
6.	$\square$	$\|$	$q < p \vee b < a$

Here, we put the general properties of order on the left side of the $\|$ and the specific facts on the right side. The proof requires 4 steps, as follows.

7.	$(1,2)$	$x \not< y \vee y \not< x$	$\|$	$\square$
8.	$(7;3,4)$	$\square$	$\|$	$a < b$
9.	$(7;5,6)$	$\square$	$\|$	$b < a$
10.	$(7;8,9)$	$\square$	$\|$	$\square$

The linking steps are (8,9,10), which use, as a nucleus, not an original linking clause, but (7), which was derived by left resolution. The proof requires 7 steps in binary resolution, 6 in positive hyper-resolution, and 5 in negative hyper-resolution. There is no proof in UR resolution. Actually, one could get a linked resolution proof in 3 steps if clause (1) were written as ($\square \parallel x \not< x$), but maybe this choice is less natural.

Of course, many other new linking clauses get generated as well; so in practice this technique would be used in conjunction with other techniques, such as bounding clause length or term complexity, to prevent the search from getting out of hand. Presumably, an implementation would also give the user the option of whether or not to allow left resolution.

3. Although we have no completeness theorem, here is an example where linked paramodulation is useful. In the above even/odd example, we wanted to tell the prover that the *predicates* E and $\neg O$ are aliases for each other, but we did not want the database to get clogged up with every possible $E/\neg O$ variant. Likewise, we may want to tell the prover that objects b and d are aliases for each other, without making it derive every b/d variant. However, from $a < b$ and $d < c$, we do want to derive $a < c$. In analogy with examples (1a) and (1c) above, we should postulate our axioms as:

$$\begin{array}{rcl} x \not< y \vee y \not< z & \| & x < z \\ \square & \| & a < b \\ \square & \| & b < d \\ b = d & \| & \square \end{array}$$

Now, the rule for linked resolution said essentially that every literal on the left of the $\|$ gets consumed. In the case of linked paramodulation, the correct requirement is that for an equation on the left side of the $\|$, *both sides* of the equation get consumed. Thus, schematically, the derivation of $a < c$ would look like:

$$\begin{array}{ccccccccccc} x & \not< & y & \vee & y & \not< & z & \vee & x & < & z \\ | & & & & & & & & & & \\ a & < & b & & & & & & & & \\ & & | & & & & & & & & \\ & & b & = & d & & & & & & \\ & & & & | & & & & & & \\ & & & & d & < & c & & & & \end{array}$$

Actually, if the aliases are only between constant symbols, the problem of needless variants could be handled by demodulation rather than linked paramodulation. In this example, one would declare $d = b$ a demodulator, which would simply replace all occurrences of 'd' by 'b'. However, if the aliases are between terms, there need not in general be a natural choice. For example, under the associative law, there are potentially many ways to express a product, and a choice of one particular association as the canonical one could lose some derivations. Using the associative law as a linking clause allows the prover to change an association and apply a proof rule, without storing all possible associations. For a specific example, given

$$\begin{array}{rcl} & \| & (ap)q > 0 \\ (xy)z = x(yz) & \| & \\ & \| & pq = c \end{array}$$

one can derive $ac > 0$ in one step, without storing $a(pq) > 0$. In this respect, linked paramodulation has a similar goal as AC unification, but applies in a potentially greater number of circumstances.

4 Acknowledgements

The author is grateful to the referee for a number of helpful comments on the exposition.

Research was supported by NSF Grant DMS-8501521 and CCR-9503445.

References

[1] R. Anderson and W. W. Bledsoe, A Linear Format for Resolution with Merging and a New Technique for Establishing Completeness, *JACM*, 17:525–534 (1970).

[2] W. W. McCune, *OTTER 3.0 Reference Manual and Guide*, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, IL, 1994.

[3] G. Robinson and L. Wos, Paramodulation and theorem proving in first-order theories with equality, in: B. Meltzer and D. Michie (eds.) *Machine Intelligence 4*, Edinburgh University Press, 1969, pp. 135–150.

[4] R. Veroff and L. Wos, The Linked Inference Principle, I: The Formal Treatment, *J. Automated Reasoning* 8:213–274 (1992).

[5] L. Wos, R. Veroff, B. Smith, and W. W. McCune, The Linked Inference Principle, II: The User's Viewpoint. In R. Shostak, editor, *Proceedings of the Seventh International Conference on Automated Deduction* (Lecture Notes in Computer Science, Vol. 170), pp. 316–332 Springer-Verlag, New York, 1984,

[6] L. Wos, *Automated Reasoning: 33 Basic Research Problems*. Prentice Hall, New Jersey, 1988.

[7] L. Wos, R. Overbeek, E. Lusk, and J. Boyle, *Automated Reasoning, Second Edition.* McGraw-Hill, 1992.