A COMPLETENESS RESULT FOR LINKED RESOLUTION

by

Kenneth Kunen

Computer Sciences Technical Report #1013

March 1991

A Completeness Result for Linked Resolution

Kenneth Kunen[1]
Computer Sciences Department
University of Wisconsin
Madison, WI 53706, U.S.A.
kunen@cs.wisc.edu
March 6, 1991

ABSTRACT

Linked resolution is a general technique for doing several binary resolution steps at once, keeping the final clause, and discarding the intermediate clauses. This technique diminishes the number of clauses stored in the database. The most unlimited version of linked resolution is trivially complete, but is also not feasible to implement because of the large number of sequences of binary resolutions which must be examined at each stage of the deduction. Various restrictions on the use of linking rule have been suggested; these restrictions do not always result in a complete deductive system. In this paper, we prove the completeness of a version of linked resolution which limits the manner in which clauses may be used as linking clauses; the completeness of hyper-resolution is a special case of this.

§0. INTRODUCTION. The notion of linked resolution was suggested by Wos, Veroff, Smith, and McCune [4]; see [3] for a detailed discussion. 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 $\neg O(\tau) \vee \phi$, an un-interesting variant $E(\tau) \vee \phi$; 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 §1, 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

[1] This research was supported by NSF Grant DMS-8501521.

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 yield $O(s(s(c)))$ 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 resolution 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 proof after exhausting all possibilities, the fault lies with the cutoffs, not with the basic rules of inference; one may then try to modify the cutoffs in the hope of obtaining a proof.

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 [3] is trivially complete, since 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 [3], for a search to be successful, there must be very stringent bounds placed on the complexity of the linked resolution steps considered. In [3], 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 [3] might suggest other completeness results as well.

§1. 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 $\Box$ for the empty clause. If ϕ, ψ are ordinary clauses, we shall usually denote their union, $\phi \cup \psi$, as a disjunction, $\phi \vee \psi = \psi \vee \phi$. A *linking clause* is an entity of the form $(\phi \parallel \psi)$, where ϕ and ψ are ordinary clauses. Semantically, $(\phi \parallel \psi)$ means the same as $\phi \vee \psi$, 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 $(\phi \parallel \Box)$. Our declaration that $O(x) \vee E(x)$ is a linking clause in the above example means that in our formal syntax, we write it as $(O(x) \vee E(x) \parallel \Box)$. A *pure standard clause* is of the form $(\Box \parallel \phi)$. In the above example, all clauses ϕ other than the ones declared linking are now written as $(\Box \parallel \phi)$.

We use letters α, β for literals, ϕ, ψ, ξ, ζ for ordinary clauses, λ for linking clauses, and σ for substitutions.

As usual in resolution, we rename 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 $\lambda \widetilde{\in} S$ 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 consequence* 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 $\parallel$. 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 $\parallel$.

1. Linking Rule: If $(\alpha_1 \vee \cdots \vee \alpha_n \parallel \phi) \widetilde{\in} S$, and for each $i = 1 \ldots n$, $(\Box \parallel \neg\beta_i \vee \psi_i) \widetilde{\in} S$, 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 $(\Box \parallel \psi_1 \vee \cdots \vee \psi_n \vee \phi)\sigma$ is an immediate consequence of S. We call $(\alpha_1 \vee \cdots \vee \alpha_n \parallel \phi)$ the *nucleus* and the $(\Box \parallel \neg\beta_i \vee \psi_i)$ the *satellites*.

2. Left Resolution: Ordinary binary resolution to the left of the $\parallel$. That is, if $(\alpha \vee \phi \parallel \psi) \widetilde{\in} S$ 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 $(\phi \vee \xi \parallel \psi \vee \zeta)\sigma$ is an immediate consequence of S.

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

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

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

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 $S \vdash (\Box \parallel \Box)$ then S is inconsistent) is obvious.

Completeness Theorem. If S is inconsistent then $S \vdash (\Box \parallel \Box)$.

Let $size(\phi \parallel \psi)$ 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 1. If $S \cup \{(\xi \parallel \phi)\} \vdash (\Box \parallel \Box)$, and ψ is ground, then either $S \cup \{(\xi \parallel \phi \vee \psi)\} \vdash (\Box \parallel \psi)$ or $S \cup \{(\xi \parallel \phi \vee \psi)\} \vdash (\Box \parallel \Box)$.

Proof. Given the original proof, just tack on a ψ each time that $(\xi \parallel \phi)$ gets used, and then factor away the extra copies of ψ if $(\xi \parallel \phi)$ gets used twice or more. In the special case that it was not used at all, one winds up with a proof of $(\Box \parallel \Box)$. ∎

The basis of the induction will use the following well-known completeness fact from ordinary resolution:

Lemma 2. 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 and factoring, one can derive from S some clause ξ' which subsumes ξ.

Proof of Completeness Theorem. 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 $(\phi \parallel \alpha \vee \psi)$, where either ϕ or ψ is non-empty. Let S' be the other linking clauses in S. Let $S_1 = S' \cup \{(\phi \parallel \psi)\}$ and $S_2 = S' \cup \{(\Box \parallel \alpha)\}$. Then both S_1 and S_2 have smaller *size* than S, so the induction hypothesis applies to them. By $S_1 \vdash (\Box \parallel \Box)$ and Lemma 1, either $S \vdash (\Box \parallel \Box)$ (so we are done immediately), or $S \vdash (\Box \parallel \alpha)$, in which case, $S_2 \vdash (\Box \parallel \Box)$ yields $S \vdash (\Box \parallel \Box)$.

If a reduction as in the previous paragraph cannot be done, then we may partition S into $S_1 \cup S_2$, where elements of S_1 are of the form $(\phi \parallel \Box)$ 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, 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. ∎

One problem with the rules as stated is that one is sometimes forced to maintain tautologies in the database. The problem is with tautologies of the form $(\phi \parallel \psi)$, where ψ contains a literal and its negation; call these *right tautologies*. 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 $\|$. 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.

Other tautologies (that is, $(\phi \parallel \psi)$, containing an α and $\neg\alpha$, not both inside ψ) can safely be eliminated. Formally, one proves, by induction on length of derivation, that if $S \vdash (\xi \parallel \zeta)$, then there is a derivation of some $(\xi' \parallel \zeta')$ using no tautologies other than right tautologies, where ξ' subsumes ξ and ζ' subsumes ζ. Hence,

Corollary. If S is inconsistent then $S \vdash (\Box \parallel \Box)$ using a derivation containing no tautologies other than right tautologies.

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

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 $(\phi \parallel \psi)$, with ϕ purely negative and ψ purely positive; in this case, left resolution and right resolution are impossible, and no right tautologies can get generated, so we have the usual completeness result for positive hyper-resolution.

§2. 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 $\parallel$ 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:

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

All other clauses would be pure standard clauses, of form $\Box \parallel \phi$. 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, $(E(x) \vee \neg E(x) \parallel \Box)$ and $(O(x) \vee \neg O(x) \parallel \Box)$. However, these tautologies are not right tautologies, so they can be discarded.

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

$$M(x) \vee F(x) \vee N(x) \parallel \square$$
$$\neg M(x) \vee \neg F(x) \parallel \square$$
$$\neg F(x) \vee \neg N(x) \parallel \square$$
$$\neg N(x) \vee \neg M(x) \parallel \square$$

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 $\parallel$ 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 $\parallel$ 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 [5] 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 (unit-resulting) resolution. It also illustrates the necessity of allowing resolution among the linking clauses.

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

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

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

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.

REFERENCES.

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

[2] McCune, W. W., OTTER 2.0 Users Guide, Technical Report ANL-90/9, Argonne National Laboratory, 1990.

[3] Veroff, R. and Wos, L., The Linked Inference Principle, I: The Formal Treatment, *preprint.*

[4] Wos, L., Veroff, R., Smith, B., and McCune, W., The Linked Inference Principle, II: The User's Viewpoint, in: R. Shostak (ed.), *Proceedings of the Seventh International Conference on Automated Deduction* (Lecture Notes in Computer Science, Vol. 230), Springer-Verlag, New York, 1984, pp. 316-332.

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