# Completeness of Input Resolution with Instance Subsumption

by

Larry M. Hines

May 1983 ATP-74

This work was supported by NSF Grant MCS 77-20701.

## ABSTRACT

Input resolution is unfortunately incomplete. This paper introduces a new restriction for resolution which has a close relation to input resolution. Resolution is allowed only if one of the parent clauses is an input clause or subsumes an instance of an input clause and is an ancestor of the other parent. This system is shown to be complete and is shown to be compatible with Loveland's s-linear resolution.

## TABLE OF CONTENTS



## LIST OF FIGURES

# Introduction

SI-input resolution will be used in this paper as an abbreviation for input resolution with instance subsumption.

Definitions. Let S be a set of clauses. A clause R is said to be an **SI-clause** if R subsumes some instance of a clause C in S. (I.e., $R\sigma \subseteq C\sigma$, for some substitution $\sigma$.[1]) An **SI-input resolution** is a resolution in which at least one of the two parent clauses either is an input clause (i.e., is in S) or is an SI-clause which is an ancestor of the other parent. An SI-input resolvent is a resolvent of an SI-input resolution. An **SI-input deduction** is a linear deduction in which every resolution is an SI-input resolution. An **SI-input refutation** is an SI-input deduction of [] from S.

Example. Consider the following set of clauses. The instance of the input clause which is subsumed is indicated at the side of each SI-clause.

```
S(a)  v  ~T(a)
~R(x)  v  ~S(x)
~Q(a)  v  R(x)
Q(x)  v  R(x)
T(x)
```

```
S(a) v ~T(a)
 |
 | ~R(x) v ~S(x)
 |/
~R(a) v ~T(a)
 |
 | T(x)
 |/
~R(a)          [subsumes ~R(a) v ~S(a)]
 |
 | ~Q(a) v R(x)
 |/
~Q(a)          [subsumes ~Q(a) v R(x)]
 |
 | Q(x) v R(x)
 |/
R(a)           [subsumes Q(a) v R(a)]
 |
 | ~R(a)
 |/
[].
```

**Figure 1:** An SI-input Resolution Example

[1] $R\theta \subseteq C\tau$ for some substitutions $\theta$ and $\tau$ is the exact translation of R subsumes some instance of C. Yet since we may rename apart the variables in R and C, $\theta$ and $\tau$ can be composed. Let $\sigma = \theta \circ \tau$. Thus we have $R\sigma \subseteq C\sigma$, for some substitution $\sigma$.

## Completeness Proof

Lemma 1. Let S be an unsatisfiable set of ground clauses. If C is a clause in S such that S - {C} is satisfiable, then there is a SI-input deduction of [] from S with top clause C.

Let k(S) be defined to the total number of appearances of literals in S minus the number of clauses in S. k(S) is called the excess literal parameter by Anderson and Bledsoe [1]. If [] is in S then C must be [] and the lemma is obvious. Assume [] is not in S. The proof is by induction.

If $k(S) = 0$, then S consists solely of unit clauses. Let L be the sole literal in C. Since S is unsatisfiable and S - {C} is satisfiable, there is a clause C' in S - {C} such that C' = ~L. Clearly [] is an SI-input resolvent of L and ~L. Thus the lemma holds for the base case $k(S) = 0$.

Assume the lemma holds when $k(S) < n$ and consider the case where $k(S) = n$ and $n > 0$.

*Case* 1. C is not a unit clause. Let C = C' v L where C' is a nonempty clause. Let $S_1 = (S - \{C\}) \cup \{C'\}$. Clearly, $S_1$ is unsatisfiable and $S_1 - \{C'\}$ is satisfiable. However, $k(S_1) < n$. Hence, by the induction hypothesis, there is an SI-input deduction $D'_1$ of [] from $S_1$ with top clause C'. Let $D_1$ be the deduction obtained from $D'_1$ by putting L back with C'. Clearly, $D_1$ is also an SI-input deduction from $S_1$ of [] or L with top clause C. If it is the former, we are done.

If it is the latter, let $S_2 = (S - \{C\}) \cup \{L\}$. Clearly, $S_2$ is unsatisfiable and $S_2 - \{L\}$ is satisfiable. However, $k(S_2) < n$. Hence, by the induction hypothesis, there is an SI-input deduction $D'_2$ of [] from $S_2$ with top clause {L}. So by combining $D_1$ and $D'_2$ we obtain an SI-input deduction of [] from S with top clause C.

*Case* 2. C is a unit clause. Let L be the sole literal in C. Since S is unsatisfiable and (S - {C}) is satisfiable, there is a clause C' = (C" v ~L) in S. If C" is empty, then let D be the deduction with the sole resolution of C against C'. Since C and C' are input clauses, D is an SI-input deduction from S of [] with top clause C.

If C" is not empty, consider $S_1 = (S - \{C'\})$.

*Case* 2.1. $S_1$ is unsatisfiable. Since $k(\{C'\}) > 0$, $k(S_1) < n$. Since S - {C} is satisfiable, $S_1 - \{C\}$ is satisfiable. Thus by the induction hypothesis, there is an SI-input deduction D of [] from $S_1$ with top clause C. Since $S_1$ is a subset of S, D is an SI-input deduction of [] from S with top clause C.

*Case* 2.2. $S_1$ is satisfiable. Note that $k(S_1 \cup \{C"\}) < n$ and that $S_1 \cup \{C"\}$ is unsatisfiable. Thus by the induction hypothesis, there is an SI-input deduction of [] from $S_1 \cup \{C"\}$ with top clause C". Let D be a deduction whose first resolution is C against C' to produce C" and whose remaining resolutions are D'. Clearly D is an SI-input deduction of [] from S with top clause C.

Lemma 2. If S is an unsatisfiable set of clauses and C is a clause in S such that S - {C} is satisfiable, then there is an SI-input deduction of [] from S with top clause C.

Since S is unsatisfiable, by Herbrand's theorem there is a finite minimally unsatisfiable set S' of ground instances of clauses in S. Let C' be the instance of C in S'. Hence, S' - {C'} is satisfiable. By Lemma 1, there is an SI-input deduction D' of [] from S' with top clause C'. From the SI-input deduction D', we now show that we can produce a SI-input deduction of [] from S. Let $r'_i$, $i > 0$, be the resolvent derived at the ith step in D' from $r'_{i-1}$ and $c'_{i-1}$. By definition of SI-input deduction either $c'_{i-1}$ is an input clause or $c'_{i-1} = r'_j$ for some j such that $j < i$ and $r'_j$ is a SI-clause.

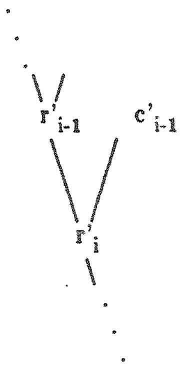



**Figure 2:** A node in D'

In the case $i = 1$, $r'_{i-1}$ and $c'_{i-1}$ are both in S'. ($r'_0 = C'$.)

Using the Lifting Theorem, let D be a lifting of D' to a deduction of [] from S. We will show that D is an SI-input deduction of [] from S.

Let $r_i$, $i > 0$, be the resolvent derived at the ith step in D from $r_{i-1}$ and $c_{i-1}$.

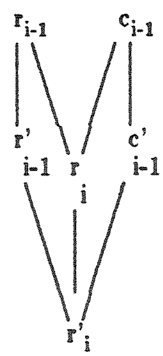



**Figure 3:** Lifting D' to D

Clearly, D is a linear deduction which means that at each step in D at least one parent is either an input clause or an ancestor of the other clause. So we need only to show that if neither parent is in S, then one of the parents is an SI-clause. We will show that if $c_{i-1}$ is not in S, then $c_{i-1}$ is an SI-clause.

Assume $c_{i-1}$ is not in S. Clearly, then, $c'_{i-1}$ is not in S' and, since D' is an SI-input deduction, for some $j < i$ $c'_{i-1} = r'_j$, where $r'_j$ is an SI-clause. Since $r_j$ subsumes $r'_j$ and $r'_j$ subsumes some clause e' in S', $r_j$ subsumes an instance of a clause e in S where e is the lifting of e'. So, $r_j$ is an SI-clause and, hence, $c_{i-1}$ is an SI-clause.

Since D is a linear deduction and in every resolution in D one of the parent clauses is either in S or is an SI-clause, D is an SI-input deduction of [] from S.

## SI-Clause Algorithm

Let Rl be a list of literals in a resolvent R. Let Cl be a list of literals in an input clause C. R subsumes an instance of C if and only if (SubsumInst Rl Cl) returns T. Thus R is an SI-clause only if for some input clause C, (SubsumInst Rl Cl) returns T.

```
(SubsumInst (LAMBDA (Rl Cl Cl2)
   (COND [(NULL Rl) T]
         [(NULL Cl) NIL]
         [((LAMBDA (Sigma)
                (AND Sigma
                     (SubsumInst (SubstSig Sigma (CDR Rl))
                                 (SubstSig Sigma (APPEND Cl2 Cl))
                                 NIL)))
           (Unify (CAR Rl)(CAR Cl)))]
         [T (SubsumInst Rl (CDR Cl) (APPEND Cl2 (LIST (CAR Cl))))]))).
```

(SubstSig Sigma Cl) instantiates the literals in the list Cl with the substitution Sigma. (Unify Lit1 Lit2) returns the mgu of the literals Lit1 and Lit2.

## Compatibility with S-linear Resolution

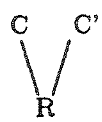
In [3] Loveland proposed a linear resolution system, called s-linear resolution, which had a subsumption restriction. An **s-linear deduction** D from the set S of clauses is a <u>linear</u> deduction, $B_1, B_2, \ldots, B_n$, such that at least one parent clause of $B_i$, $1 < i \leq n$, is either

1. in S or

2. a clause $B_j$, $j < i\text{-}1$, chosen so that the resolvent $B_i$ subsumes an instance of $B_{i-1}$. (I.e. $B_i\sigma \subseteq B_{i-1}\sigma$ for some substitution $\sigma$.)

In s-linear resolution the subsumption is between a resolvent and one of its parents; whereas, in SI-input resolution it is between one of the parent clauses and one of the input clauses. Though the restrictions are quite different they are compatible.

Let S be a set of clauses. An **SIS-input resolution** (SI-input and s-linear) is an SI-input resolution such that if neither parent clause is an input clause (thus one parent, say C, is an SI-clause) then an instance of C is subsumed by the resulting resolvent.




That is, if neither C nor C' is in S then C (or C') is an ancestor of C' (or C) and is an SI-clause (i.e., $C\sigma \subseteq C^*\sigma$ for some substitution $\sigma$ and for some $C^*$ in S) and for some substitution $\mu$, $R\mu \subseteq C'\mu$ (or $R\mu \subseteq C\mu$). An **SIS-input deduction** is a <u>linear</u> deduction in which every resolution is a SIS-input resolution.

Lemma 1. Let S be an unsatisfiable set of ground clauses. If C is a ground clause in S such that S - {C} is satisfiable, then there is a SIS-input deduction of [] from S with top clause C.

The proof is only slightly different from the proof of lemma 1 with the SI-input replaced by with SIS-input. The only change is that we must show that in case 1 under the inductive hypothesis that $D_1$ is an s-linear deduction. Consider those resolutions in $D_1$ where L was added to the resolvent in forming $D_1$ from $D'_1$.

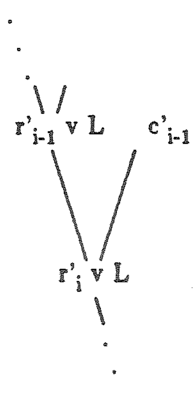



**Figure 4:** A node in $D_1$

Since $r'_i$ subsumes $r'_{i-1}$ ($D'_1$ is an s-linear deduction), $r'_i \vee L$ subsumes $r'_{i-1} \vee L$. Thus $D_1$ is an SIS-input deduction from $S_1$ of [] or L.

Lemma 2. If S is an unsatisfiable set of clauses and C is a clause in S such that S - {C} is satisfiable, then there is a SIS-input deduction of [] from S with top clause C.

The proof differs from the proof of lemma 2 with SI-input replaced by SIS-input in only one substantial way. We must additionally showed that if $c_{i-1}$ is not in S that $r_i$ subsumes an instance of $r_{i-1}$. (See Figure 3.) Since D' is an s-linear deduction, $r'_i \subseteq r'_{i-1}$. Thus $r_i$ subsumes an instance of $r_{i-1}$, namely $r'_{i-1}$.

## An Example Show to a Difference Between M.S.L. Resolution and SI-input Resolution

Resolution with merging was first proposed by Andrews in [2]. It was later strengthened by Reiter in [4]. Anderson and Bledsoe in [1] combined Loveland's s-linear resolution [3] with merging. Since there may be some confusion about the difference between these versions of resolution and SI-input resolution, an example is given to illustrate the difference. The version considered here is Anderson and Bledsoe's.

Definitions. Let R be a resolvent of clauses $C_1$ and $C_2$ with mgu $\theta$. R is a merge clause if there are literals $L_1$ and $L_2$ in $C_1$ and $C_2$, respectively, such that $L_1\theta = L_2\theta$ and if neither $L_1$ nor $L_2$ were the literals resolved on. A **m.s.l. deduction** (merging, subsumption, linear) of $R_n$ from a set of clauses S with top clause C, a clause in S, is any deduction of the form:

$$\begin{tikzcd}[column sep=small]
C \arrow[dr, dash] & & C_0 \arrow[dl, dash] & & & & & \\
& R_1 \arrow[dr, dash] & & C_1 \arrow[dl, dash] & & & & \\
& & R_2 \arrow[dr, dash] & & C_2 \arrow[dl, dash] & & & \\
& & & \cdots \arrow[dr, dash] & & & & \\
& & & & R_{n-1} \arrow[dr, dash] & & C_{n-1} \arrow[dl, dash] & \\
& & & & & R_n & &
\end{tikzcd}$$

**Figure 5:** M.S.L Deduction

$R_i$, $i = 1$ to n, is a resolvent of the two clauses immediately above it. $C_{i-1}$, $i = 1$ to n, is either in S or is a $R_j$ for some $j < i-1$. If $C_{i-1}$ is not in S then $R_i$ subsumes $R_{i-1}$, $C_{i-1}$ is a merge clause, and the literal resolved upon in $C_{i-1}$ is a merged literal of $C_i$.

Consider the same set of clauses mentioned above.

```
S(a) v ~T(a)
~R(x) v ~S(x)
~Q(a) v R(x)
Q(x) v R(x)
T(x)
```

```
S(a) v ~T(a)
  |
  | ~R(x) v ~S(x)
  | /
  |/
~R(a) v ~T(a)
  |
  | T(x)
  | /
  |/
~R(a)
  |
  | ~Q(a) v R(x)
  | /
  |/
~Q(a)
  |
  | Q(x) v R(x)
  | /
  |/
R(a)
  |
  | ~R(x) v ~S(x)
  | /
  |/
~S(a)
  |
  | S(a) v ~T(a)
  | /
  |/
~T(a)
  |
  | T(x)
  | /
  |/
[].
```

**Figure 6:** A M.S.L. Resolution Example

No merge clauses occur in this example. The deduction is longer than the SI-input deduction on the same example. No claim is made here about the efficiency of SI-input resolution versus merge resolution. However, where many of the input clauses contain common literals merging may not restrict the search.

## Subsuming an Input Clause

SI-input resolution can be further augmented. When a subsumption of an input clause (not just an instance) by an SI-clause is recognized, the SI-clause should take the input clause's place in the input set. In the example given above, the resolvent ~Q(a) subsumes not only an instance of the input clause ~Q(a) v R(x) but subsumes the clause itself.

## Appendix

```
(SI-Resolution (InputSet)
    [(λ (PreviousClauses)
        (SOME InputSet '(λ (TopClause)
                            (SI-Refute TopClause NIL NIL)))
     InputSet]).

(SI-Refute (CenterClause SI-Set SI-Boolean)
    (OR (SOME SI-Set '(λ (SideClause)
                        (Resolve&Refute SideClause NIL)))
        (SOME InputSet '(λ (SideClause)
                          (Resolve&Refute SideClause NIL))))).

(Resolve&Refute (SideClause InputClause-Boolean)
    (SOME CenterClause '(λ (Lit1)
                (SOME SideClause '(λ (Lit2)
                        (COND [(AND (OR (NEQ 'NOT (CAR Lit1))
                                        (EQ 'NOT (CAR Lit2))
                                        (NULL
                                         (SETQ Sig
                                               (Unify (CADR Lit1)
                                                      Lit2))))
                                    (OR (EQ 'NOT (CAR Lit1))
                                        (NEQ 'NOT (CAR Lit2))
                                        (NULL
                                         (SETQ Sig
                                             (Unify Lit1
                                                    (CADR Lit2))))))
                               (* Lit1 and Lit2 do not have opposite
                                  signs or do not unify.)
                               NIL]
                              [(NULL
                                (SETQ R
                                   (Crunch
                                      (SubstSig Sig
                                            (APPEND
                                               (REMOVE Lit1
                                                  CenterClause)
                                               (REMOVE Lit2
                                                  SideClause))))))
                               (* R reduces to Box.)
                               T]
                              [(Tautology R) NIL]
                              [(NULL (RemSubsumed PreviousClauses
                                                  (LIST R)))
                               (* R has been derived already.)
                               NIL]
                              [(AND (NOT InputClause-Boolean)
                                    (NULL (SETQ Sig
                                                (SubsumInst R
                                                     CenterClause))))
                               (* The side clause is not an input
                                  clause; yet, R fails to subsume an
                                  instance of its other parent.
                                  Thus s-linear restriction fails.)
                               NIL]
                              [T (push PreviousClauses R)
                                 (MAPC InputSet '(λ (Cl)
                                      (AND (Subsume R Cl)
                                           (DSUBST R Cl InputSet))))
                                 (* R replaces the input clauses
                                    that it subsumes.)
```

```
                    (SI-Refute R
                         (COND [SI-Boolean
                                (CONS CenterClause
                                      SI-Set)]
                               [T SI-Set])
                         (SOME InputSet '(λ (C1)
                               (SubsumInst R C NIL))))]
               )))))).
```

SubsumInst, SubstSig, and Unify are as described earlier. (Subsume R C) returns T only if R subsumes C. (Tautology R) returns T only if R is a tautology. (RemSubsumed L1 L2) returns those clauses in L2 that are not subsumed by the clauses in L1. (Crunch R) reduces R.

## References

[1] Anderson, R., and Bledsoe, W. W.
A linear format for resolution with merging and a new techinque for establishing completeness.
*J. ACM* 17:525-534, July, 1970.

[2] Andrews, P.
Resolution with merging.
*J. ACM* 15:367-381, October, 1968.

[3] Loveland, D. W.
A linear format for resolution.
In M. Laudet (editor), *Symposium on Automatic Demostration.* Springer-Verlag, New York, 1970.

[4] Reiter, B.
Two results on ordering for resolution with merging and linear format.
*J. ACM* 18:630-646, October, 1971.