SUBSUMPTION AND CONNECTION GRAPHS

Norbert Eisinger

Institut fur Informatik I
Universit&t Karlsruhe
D-7500 Karlsruhe 1, Postfach 6380

ABSTRACT

A subsumption test based on the principal idea of Kowalski's connection graph proof procedure is developed. In contrast to the standard test this new test is sufficiently efficient to permit the unrestricted use of the subsumption rule in practice. The test is not limited to the connection graph proof procedure, but most naturally embedded into it. In the latter case the unrestricted combination of subsumption with other deletion rules is shown to be inconsistent.

1. INTRODUCTION

R. Kowalski's connection graph proof procedure [K751 represents a set of logical formulae not as a set but as a graph like structure. Inference rules such as resolution and factoring are expressed as operations on that graph. In addition to these operations, which usually expand the graph, the procedure also provides for deletion operations that remove certain components from the graph. Other approaches based on graphs have been proposed by [SI76],CSH76], [AN79],[B79].

One of the striking properties of the connection graph proof procedure is that application of a deletion operation can result in the applicability of further deletion operations, thus potentially leading to a snowball effect which rapidly reduces the graph. The probability of this effect rises with the number of deletion rules available.

A very powerful deletion rule for resolution based systems is the subsumption rule ([L78], chapter 4). Unfortunately a test for subsumption is very expensive and is usually implemented only for restricted cases. In this paper a test for subsumption based on the principal idea of the connection graph proof procedure is developed, which in contrast to the standard test (CL78]) is sufficiently efficient to permit unrestricted subsumption in practical cases. Though not limited to it, the test is most naturally embedded into the connection graph proof procedure, but unrestricted combination of subsumption, tautology, and purity deletion is shown to render the connection graph proof procedure inconsistent.

2. DEFINITION AND NOTATIONS

The paper is based on the first order predicate calculus with the usual notational conventions for constants, variables, terms, atoms, literals, and clauses. The number of literals in a clause C is denoted by |CI. The empty clause is denoted by o. The elements in a set of clauses are always assumed to be variable disjoint.

A substitution is a mapping a from variables to terms identical almost everywhere. It is extended to mappings on terms, atoms, literals, and clauses by the usual homomorphism. A unifier for two terms (or literals) s, t is a substitution a for which o(s) »a(t). If for any other unifier 0 for s, t there is a 0' with 0 * ©•©c, we say a is a most general unifier (mgu) for s and t. Two literals are called a-complementary, if they are of opposite sign and a is an mgu for

their atoms. Two substitutions σ, Θ are strongly compatible, if for each variable v: σ(v)≠v ∧ Θ(v)≠v ⇒ σ(v)=Θ(v). Note that for a set of pairwise strongly compatible substitutions their functional composition is commutative.

A connection graph is a pair (C,L) for which

1) C is a set of clauses
2) Let $LIT = \bigcup_{C \in C} C$ be the set of all literals occuring in the clauses of C. Then $L \subseteq C \times LIT \times C \times LIT$ is a relation such that
   a) $(C,L,C',L') \in L \Rightarrow C \neq C'$, $L \in C$, $L' \in C'$, L and L' are σ-complementary
   b) $(C,L,C',L') \in L \Leftrightarrow (C',L',C,L) \in L$

The graph is said to be total, if condition 2a) also holds in the opposite direction. A literal L in a clause C is pure, if there are no C', L' such that $(C,L,C',L') \in L$. The elements of L are called links. They connect σ-complementary literals in different clauses, thus indicating possible resolutions.

The connection graph proof procedure defined in [K75] processes a set C of clauses in the following way: first all possible links within C are computed, i.e. a total graph (C,L) is constructed. This graph is then reduced by some deletion rules yielding another total graph. Next a link is selected and its resolvent created and incorporated into the graph together with all its factors. Then the link resolved upon is deleted from the graph and again the reduction rules are applied. The next link is selected and so on until $\square \in C$ or $L = \emptyset$.

Incorporating a new clause into the graph means creating all links between its literals and the rest of the graph. A literal in a resolvent or factor has to be connected to all those places in the graph to which at least one of its parent literals is connected by a link. Thus the links are simply inherited. In addition to these inherited links all links between a resolvent (or factor) and its parent clauses have to be explicitly computed, or, more efficiently, derived from "autolinks" between complementary literals in self resolving clauses (CB753).

The reduction rules contribute both to the *practical* attractivity and the *theoretical* difficulties of the connection graph proof procedure. The original rules are: delete a clause if it contains a pure literal and delete a clause which is a tautology. Further possible rules include: delete a link if its resolvent is a tautology CWA81], delete a clause if it is subsumed by another clause in the graph, delete a link if its resolvent is subsumed by another clause in the graph. Note that each deletion may cause a purity to arise, thereby causing further deletions. It is not yet known as to which combinations of these deletion rules preserve the completeness of the procedure.

3. SUBSUMPTION AND THE S-LINK TEST

Let C and D be clauses. C o-subsumes D, if ICI s iDI and o is a substitution such that o(C) c D. Throughout this paper the terms subsumption and o-subsumption will be used interchangeably, though usually subsumption is defined more generally than a-subsumption (see [L78]).

The standard test for o-subsumption works as follows: given C and D, first make sure that Id s IDI and that D is not a tautology. Then negate D and change variables in D to constants, yielding a set D of ground unit clauses. C asubsumes D iffa is derivable from {C} u D. (Details can be found in CC173] and CL78].).

The positive aspect of this subsumption test is that it uses the same mechanism which underlies the entire deduction system, i.e. resolution. But from a practical point of view this turns out to be a disavantage. Normally one has to check for

subsumption as soon as a new clause is generated, i.e. after each resolution step. This means that each "major" resolution step is followed by several "minor" resolution steps for the subsumption test, thus multiplying the overall expense. Yet even worse, given a resolvent C there is no hint as to which clauses potentially subsume or are subsumed by C. So the test, already expensive in itself, has to be performed within an iteration over all elements of the given set of clauses. In practice, of course, one would first make sure that the predicates are in common, so that the test is not performed during each iteration step.

The resulting cost is such that for practical systems only restricted versions of subsumption are implemented, e.g. only for cases where the subsuming clause is a unit. Omitting subsumption, on the other hand, can cause considerable redundancies.

The central problem for a subsumption test consists in efficiently finding out which literals in which clauses are unifiable. Disregarding the signs of the literals this corresponds to the very same problem that arises when two clauses have to be selected for the next resolution step. In both cases comparing all literals of all clauses is a possible but inefficient solution.

In the resolution case the connection graph procedure provides for a more efficient alternative. The literals of a set of clauses are compared with each other once and forever when the initial graph is constructed. Subsequently the necessary information is directly available in the form of the links. Because of the inheritance mechanism for links the new literals in resolvents and factors need not go through any search process either. Thus the problem of finding two resolvable clauses is reduced to simply picking a link.

This basic idea can be applied to the subsumption problem by introducing links of a new type that connect unifiable literals. Formally we define a subsumption graph (S-graph) as a pair (C,S) such that

1) C is a set of clauses
2) Let $LIT = \bigcup_{C \in C} C$ be the set of all literals occuring in the clauses of C. Then $S \subseteq C \times LIT \times C \times LIT$ is a relation such that
   a) $(C,L,C',L') \in S \Rightarrow C \neq C'$, $L \in C$, $L' \in C'$, L and L' are unifiable
   b) $(C,L,C',L') \in S \Leftrightarrow (C',L',C,L) \in S$

The S-graph is said to be S-total, if condition 2a) also holds in the opposite direction. A literal L in a clause C is S-pure, if there are no C', L' such that $(C,L,C',L') \in S$. The elements of S are called S-links.

Given a set of clauses to be refuted, we initially compute all possible S-links between literals in these clauses. When a new resolvent or factor is derived, the S-links are inherited from the parent clauses in the same way as are the resolution links in the connection graph proof procedure. But in contrast to resolution links (R-links) S-links are never deleted, unless one of the parent clauses is removed from C. It can be shown easily that under these circumstances the S-graph remains S-total throughout the entire computation (see [E80]).

In order to develop a subsumption test using S-graphs we need some further definitions:
Let (C,S) be an S-graph, $C \in C$ a clause and $L \in C$ a literal.
We define $con(C,L) := \{D \in C \mid \exists K \in D (C,L,D,K) \in S\}$ as the set of all clauses connected to L in C by S-links.
Further let $sub(C) := \bigcap_{L \in C} con(C,L)$ be the set of all clauses connected to every literal in L by S-links.

For $L \in C$ and another clause $D \in C$ we define $uni(C,L,D) := \{\sigma \mid \exists K \in D (C,L,D,K) \in S \wedge \sigma(L) = K\}$ as the set of all matching substitutions

mapping L onto some literal in D. Finally let $U_1,\ldots,U_n$ be sets of substitutions. Then merge $(U_1,\ldots,U_n) := \{(\sigma_1,\ldots,\sigma_n) \in U_1\times\ldots\times U_n \mid$ the σ_i are pairwise strongly compatible$\}$ is the subset of their cartesian product, for which the functional composition of the components yields a unique substitution regardless of their order.

The subsumption test is provided by the following theorems:

Theorem 1

Let (C,S) be an S-total subsumption graph and $C = \{L_1,\ldots,L_n\} \in S$ a clause, $n \geq 1$. Then for $D \in C$

C σ-subsumes D iff $|C| \leq |D| \wedge D \in sub(C) \wedge$ merge$(uni(C,L_1,D),\ldots,uni(C,L_n,D)) \neq \emptyset$.

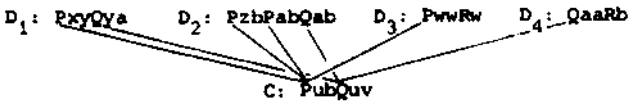
Theorem 2

Let (C,S) be an S-total subsumption graph and $D = \{K_1,\ldots,K_m\} \in C$ a clause, $m \geq 1$. Then for $C \neq \square$

C σ-subsumes D only if $C \in \bigcup_{i=1}^{n} con(D,K_i)$.

Detailed proofs can be found in [EBO].

The following example illustrates the principle of a test based on Theorem 1. Assume the set of clauses $C = \{C,D_1,D_2,D_3,D_4\}$ with $C = \{Pub,Quv\}$, $D_1 = \{Pxy,Qya\}$, $D_2 = \{Pzb,Pab,Qab\}$, $D_3 = \{Pww,Pw\}$, $D_4 = \{Qaa,Rb\}$. We want to find all clauses subsumed by C. In this case only S-links connected to C are relevant, so for reasons of clarity all other S-links are omitted in the S-graph for C:



Now $|C| \leq |D_i|$ for all i and $sub(C) = \{D_1,D_2\}$, because D_3 and D_4 are each connected to only one literal of C. We have to associate with each literal of C a set of matching substitutions for each clause in sub(C). In this case $uni(C,Pub,D_1) = \emptyset$ because there is no σ such that σ(Pub) = Pxy. Thus D_1 can be disregarded.

For D_2 we obtain $uni(C,Pub,D_2) = \{u := z\}$, {u :» a}}and uni(C,Quv,D_2> «{{u:«a, v:»bf}. The substitutions {u :« a} and {u ;» a, v :« b} are strongly compatible, thus C subsumes D_2 (but none of D,,D_3,D_4).

The main point of the test is that the expensive unification is postponed until the function sub preselected a plausible subset of candidates for subsumption. In case we are looking for subsuming rather than subsumed clauses/ this preselection process is slightly more complicated. By Theorem 2 we first determine all clauses connected by at least one S-link to the given clause D. Then from this set we ascertain those C for which Id s IDI and D€sub(C) holds and only then the unification operations are performed.

In both cases this preselection and the fact that the literals to be unified are explicitly known, can save considerable time. On the other hand some effort has to be invested for the computation of all S-links in the initial graph. As with the connection graph proof procedure this advance cost can be higher than a possible gain, if the set of clauses is only small. For more complex examples however, there is certainly a pay off. But of course any gain in time has to be paid for by additional storage needed for the S-links.

4. REFINEMENTS OF THE S-LINK TEST

The inheriting mechanism for S-links can be optimized in the same way as described in [B75] for resolution links. Here the proofs are very simple because S-graphs are always total.

Another refinement results from the observation that for a clause C containing an S-pure literal sub(C) « 0. That means that such a clause cannot subsume any other clause.

When computing the unKCL^D) we need to know which K€D are unifiable with L.. As

the definition of uni shows, these are exactly those literals we already had to consider for the computation of con and s\±>. The information obtained during the computation of con and sub should be stored in an appropriate data structure to avoid having to recompute it for uni.

Thus far the S-link test was developed without considering the underlying inference mechanism. Since they are based on the same principal idea as connection graphs, S-graphs appear to be combined most naturally with this inference method. We can modify the definition of a connection graph to be a triple *(C,R,S)* such that *(C,F)* is a connection graph (in the hitherto sense) and (£,5) is an S-graph. For such a graph we can define a new kind of subsumption: a clause C subsumes an R-link (D,,K,,D_2*K_2),if C subsumes the resolvent of (D^K^D-,!^). It is possible to extend the test to cover this kind of subsumption (see [E80]).

Deleting subsumed links has the effect as if all resolutions leading to subsumed resolvents were performed prior to other steps. This results in a stronger reduction of the graph as in the usual case were subsumptions occur only randomly. The difference is similar to the one between deleting tautology clauses and deleting tautology links in a graph [WA81].

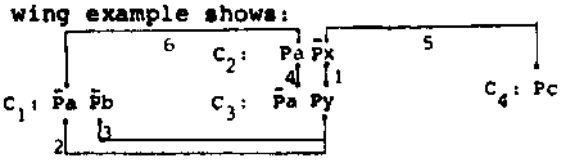
5. CONSISTENCY OF THE CONNECTION GRAPH PROCEDURE WITH SUBSUMPTION

The soundness of a proof procedure guarantees that whenever an unsatisfiable set of clauses (e.g. one containing o) is derived, then the original set of clauses was unsatisfiable. Consistency, on the other hand, assures that from an unsatisfiable set of clauses only unsatisfiable sets of clauses can be derived. Thus consistency is a necessary condition for completeness.

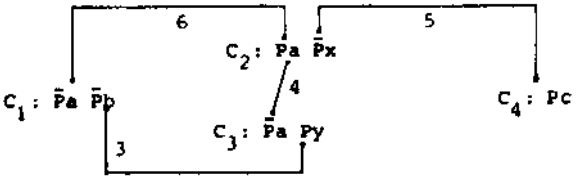
In most resolution based proof procedures consistency is trivial, because the original set of clauses is only extended by further clauses. In the case of the connection graph procedure, however, clauses can be deleted from the original set. This fact causes consistency to be an extraordinarily hard problem for connection graphs.

In [E80] the consistency of all deletion rules mentioned in the introduction is proven for o-derivations [SS76 1, using a proof technique introduced in [B79 3. Unfortunately this technique is not suitable to arbitrary derivations, but results shown in CSS76] indicate that there is a translation between ot-derivations and arbitrary ground case strategies. The respective proofs are very complex and hard to follow, but if extended to the case of subsumption they might ensure the consistency for the ground case.

In the general case, however, the connection graph proof procedure with unrestricted subsumption is inconsistent as the following example shows:



The resolvent of link 1 is a tautology, the resolvent of link 2 is subsumed by C,,thus both links are deleted and the graph becomes



Resolution on links 3, 5 and the successor of 6 (in that order) will yield the empty clause. Hence the original set of clauses was unsatisfiable, because of the soundness of the resolution principle. If instead of that we select link 4 for the next step, the resolvent is PuPv with link 3 and 5 in-

herited. This resolvent subsumes both C~ and C^, which are deleted. The resulting graph is

C_1: PaPb Pu ^{5}' C_4: *c

Now C contains a pure literal and the subsequent deletions finally erase the entire graph, i.e. it is possible to derive a satisfiable set of clauses from the original unsatisfiable set!

This result demonstrates the necessity of restrictions for the deletion rules. It has yet to be shown which restrictions preserve consistency and completeness.

6. PRACTICAL RESULTS

The S-link test has been implemented in the Markgraf Karl system at the University of Karlsruhe [ESSUW80], [SS801, fESSW81]. Subsumption was restricted such that a resolvent may not be subsumed by its own parent clauses. The necessity of this restriction appears plausible, because a similar one applies to factors. No example for the inconsistency of the procedure with this restriction could be found thus far, but neither has there been a formal proof of its consistency.

On the average a graph has about the same number of S-links as it has R-links. This may seem an inappropriate increase in storage requirement. But in the actual implementation S-links need much less storage than R-links. Moreover, it is not the physical storage that is important, but the number of active R-links in the search space, and this number can be reduced considerably.

Practical tests indicate that the reduction of the graph caused by subsumption usually more than compensates for the storage used by the S-links. An example is P. Andrews[1] "challenge" proposed at the deduction workshop in Austin 1979: (3xQx«VyQy) ■ OxVy ■ Qx ■ Qy). Here subsumption reduces the initial graph by 89 % of the clauses * and by 99,5 % of the R-links (which is, however, an extreme case).

In order to get some experience with more "natural" problems, a selection of examples from TMOW76] and [WM76] was run using the strategies basic resolution, set-of-support, and unit refutation, each with and without subsumption. The values compared are:

r r^r^war^** - tt clauses in proof
G-penetrance « « clauses generated
^ ~A~^.fc.~»~„A - # Resolvents/Factors in proof
D-penetrance - $ Resolv./Fact. totally Sauced
R value *m* ft clauses deleted
clauses generated

The results are compiled in table 1 (- means that the system did not find a proof).

The table shows that subsumption usually caused a considerable improvement of the penetrances, i.e. fewer unnecessary steps were performed. This demonstrates the reduction of the search space. Sometimes the system even found a proof where it did not without subsumption. The increase of the R-value indicates that subsumption infact has a very strong impact on the size of the graph.

7. CONCLUSION

The S-link test is a rather efficient subsumbtion criterion and the practical results are encouraging. The hard theoretical problem of consistency and completeness of the connection graph procedure with various combinations of deletion rules has yet to await a comprehensive treatment.

8. ACKNOWLEDGEMENT

The members of the Markgraf Karl Group in Karlsruhe provided many valuable discussions about the S-link test. Jttrg Siekmann and Jack Minker went over the first draft of this paper and gave many helpful hints for its improvement.

Name & strategy	G-penetrance no subs	G-penetrance subs	D-penetrance no subs	D-penetrance subs	R-value no subs	R-value subs
Burstall						
BASIC	0,37	0,39	0,29	0,31	0,33	0,50
SOS	0,49	0,50	0,43	0,44	0,51	0,52
UNIT	0,45	0,48	0,38	0,42	0,39	0,42
Burstall-S						
BASIC	0,37	0,36	0,29	0,28	0,28	0,41
SOS	0,42	0,50	0,33	0,44	0,42	0,42
UNIT	0,42	0,45	0,34	0,38	0,32	0,35
Prim						
BASIC	0,87	0,81	0,86	0,86	0,35	0,43
SOS	0,57	0,71	0,46	0,63	0,34	0,43
UNIT	0,95	0,96	1,00	1,00	0,23	0,33
MQW						
BASIC	0,01	0,37	0,01	0,29	0,83	0,58
SOS	0,78	0,78	1,00	1,00	0,11	0,22
UNIT	-	-	-	-	-	-
EWJ						
BASIC	0,77	0,89	0,77	1,00	0,27	0,95
SOS	-	0,94	-	1,00	-	0,94
UNIT	-	-	-	-	-	-
Hasparts-2						
BASIC	0,87	0,91	0,80	0,86	0,35	0,36
SOS	0,95	1,00	0,92	1,00	0,35	0,42
UNIT	-	-	-	-	-	-
LS-17						
BASIC	0,23	0,50	0,17	0,45	0,39	0,35
SOS	-	0,76	-	0,88	-	0,38
UNIT	0,15	0,41	0,09	0,33	0,22	0,41
LS-115						
BASIC	-	0,41	-	0,64	-	0,34
SOS	0,46	0,46	1,00	1,00	0,00	0,11
UNIT	-	0,41	-	0,64	-	0,12

table 1

REFERENCES

[AN79] P. Andrews: Theorem Proving via General Matings, JACM, vol 28, no 2, 1981

[B79] W. Bibel: On Matrices with Connections, Univ. Karlsruhe, 1979

[B75J M. Bruynooghe: The Inheritance of Links in a Connection Graph, Rep. CW2, Kath. Univ. Leuven, 1975

[CL73J C.L. Chang, R.C.T. Lee: Symbolic Logic and Mechanical Theorem Proving, Acad. Press, New York, 1973

[E80] N. Eisinger: Connect!onsgraphen und Subsumptionsregeln, Diplom thesis, Univ. Karlsruhe, I960 (in German)

[ESSW80] N. Eisinger, J. Siekroann, G. Smolka, E. Unvericht, C. Walther: The Markgraf Karl Refutation Procedure, Proc. of AISB Conf., Amsterdam, 1980

[ESSWK81] N. Eisinger, J. Siekmann, G. Smolka, P. Kursawe: The Markgraf Karl Refutation Procedure: USER MANUAL, Univ. Karlsruhe, 1981

[ESW78] N. Eisinger, J. Siekmann, G. Wrightson: Pararoodulated Connection Graphs, Proc. of the AISB/GI Conf., Hamburg, 1978

[H773 G. Huet: Confluent Reductions: Abstract Properties and Applications to Rewrite Systems, IRIA lab.,Rapp. de Recherche no 250, 1977

[K75] R. Kowalski: A Proof Procedure Using Connection Graphs, JACM, vol 22, no 4, 1975

[L78] D. Loveland: Automated Theorem Proving: A Logical Basis.Fundamental Studies in Computer Science, North-Holland, 1978

[LS75] M. Livesey, J. Siekmann: Termination and Decidability Results for String Unification, Essex Univ. Memo CSM-8-75, 1975

[M0W76] J.McCharen, R. Overbeck, L. Wos: Problems and Experiments for and with Automated Theorem Proving Programs, IEEE Trans, on Comp., vol C-25, no 6, 1976

(R65] J.A. Robinson: A Machine-Oriented Logic Based on the Resolution Principle, JACM,vol 12, 1965

(SH76] R.E. Shostak: Refutation Graphs, J. of Art. Intelligence, vol 7, no 1, 1976

[SI76] S. Sickel: A Search Technique for Clause Interconnectivity Graphs, IEEE Trans, on Comp., vol C-25, no 8, 1976

[SS76] J. Siekmann, W. Stephan: Completeness and Soundness of the Connection Graph Proof Procedure, Univ. Karlsruhe, 7/76, 1976

[SS80] J. Siekmann, G. Smolka: Selection Heuristics, Deletion Strategies and Terminator Configurations for the Connection Graph Proof Procedure, Univ. Karlsruhe, 1980

[SS81] J. Siekmann, W. Stephan: Completeness and Consistency of the Connection Graph Proof Procedure, to appear 1981

TWAGI] C. Walther: Elimination of Redundant Links in Extended Connection Graphs, Proc. of GWAI-81, Springer Fachberichte, 1981

[WM76] E. Wilson, J. Minker: Resolution, Refinements and Search Strategies,- A Comparative Study, IEEE Trans, on Computers, vol C-25, no 6, 1976