A Tractable Class of Abduction Problems

Kave Eshghi*
Hewlett Packard Laboratories,
1501 Page Mill Road,
Palo Alto, CA 94304
USA
Email: ke@hplb.hpLhp.com

Abstract

The problem of finding a set of assumptions which explain a given proposition is in general NP-hard, even when the background theory is an acyclic Horn theory. In this paper it is shown that when the background theory is acyclic Horn and its pseudo-completion is unit refutable, there is a polynomial time algorithm for finding minimal explanations. A test for unit-refutability of clausal theories is presented, based on the topology of the connection graph of the theory.

1 Introduction

The abduction problem in which we are interested can be defined as follows:

Definition 1 The triple <Π,A,g> is an abductive problem iff Π is a set of propositional Horn Clauses, A is a set of propositions and g is a proposition. The set of propositions Δ is a solution of the abductive problem <Π,A,g> iff

1. $\Delta \subseteq A$

2. $\Delta \cup \Pi \vdash g$

3. $\Delta \cup \Pi$ is consistent

4. $(a \in A$ and $\Delta + \Pi \vdash a) \rightarrow a \in \Delta$

Δ is a minimal solution of <Π,A,g> iff it is a solution of <Π,A,g> and no subset of Δ is a solution of <Π,A,g>.

Given the abductive problem <Π,A,g> we call Π the ***back-*** *ground theory,* A the *abducible set,* and g the *goal.* A proposition is *abducible* if it belongs to A. (Throughout the paper, we will use 8 to refer to the set of non-abducible propositions in $\Pi \cup \{g\}$)

The purpose of condition 4 is to enable us to include clauses with abducibles at their head in the theory. It ensures that the set of assumptions adopted as the solution is closed under logical implication.

Selman and Levesque [16] show that even when only one solution is required, and the background theory n is restricted to be acyclic, finding abductive solutions is NP-hard. This has led to pessimism regarding the practical utility of abduction as formulated above.

literal: Let p be a proposition. Then p and -p are literals.

clause: A clause is a disjunction of literals.

Horn Clause: A Horn Clause is a clause in which there is at most one positive literal.

definite clause: A definite clause is a Hom Clause in which there is exactly one positive literal. A definite clause such as $p_1 \vee \neg p_2 \vee \neg p_3 \ldots \vee \neg p_n$ can also be written as $p_1 \leftarrow p_2, p_3, \ldots, p_n$. We call pi the head of the clause, and the conjunction $p_2 \wedge p_3 \wedge \ldots \wedge p_n$ the body of the clause.

denial: A denial is a Horn clause in which there are no positive literals. A denial such as $\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n$ can also be written as $\leftarrow p_1, p_2 \ldots p_n$

unit clause: A unit clause is a clause which has one literal.

In this paper, we present a collection of definitions, results and algorithms, which together define a class of abduction problems for which finding a minimal solution is tractable. We also provide a polynomial time test for the membership of the abduction problem in this class. Below is a summary of these definitions, results and algorithms. Some of these are taken from the literature, the rest are new.

1. The definition of the pseudo-completion of a Horn theory with respect to a set of abducibles. (This is a variation on the standard definition of completion in logic programming)
2. The definition of the minimization of a set of proposition symbols with respect to a propositional theory (this is a variation on the minimization ideas used in circumscription and the notion of minimal diagnosis in model-based diagnosis)
3. A theorem stating that for abduction problems with acyclic background theories, minimal solutions of the abduction problem correspond to minimizations of the abducible set with respect to the pseudo-completion of the background theory. This result is new.
4. An algorithm for finding a minimization of a set of propositions with respect to a clausal theory. This algorithm is known in the literature.
5. A definition of unit-refutability for propositional clausal theories. Unit refutability ensures that computing a minimi-

*This research was conducted while the author was at Hewlett Packard Labs-Stanford Science Center.

zation of a set of proposition with respect to the theory is tractable. This definition is new.

6. A test for unit-refutability of clausal theories. This test relies on the topology of the connection graph of the theory, and is polynomial time with respect to the size of the theory. Passing this test is a sufficient, but not necessary condition for unit refutability. This test is new.

As a result, we argue that the class of abduction problems which have a background theory with a unit-refutable pseudo-completion is tractable, and provide a sufficient, but not necessary, test for membership of this class.

Notice that the background theory can have general denials (as opposed to denials restricted to only abducible propositions), and that we allow abducibles to occur at the head of clauses.

For the case that the test fails, or we do not perform it, we will not know whether or not the pseudo-completion is unit refutable. In this situation, we can still use the algorithm, but we need to augment it with a correctness test to make sure that the set of assumptions returned is a solution. Sometimes this test will fail, i.e. we will not be able to generate a solution. The correctness test itself is of linear complexity, so when we do not know that the pseudo-completion is unit-refutable, we have a correct, but incomplete, tractable technique for finding solutions to abduction problems of the above form.

2 Pseudo-Completions

In order to define the pseudo-completion of Π, the first step is to assign to each clause c in Π a unique proposition n_c which does not occur in $\Pi\cup\{g\}\cup A$. We call n_c the name of c. No two clauses have the same name.

Definition 2 Let the clauses $p\leftarrow Q_1$, $p\leftarrow Q_2,\ldots$, $p\leftarrow Q_k$, where $Q_1,Q_2,\ldots,Q_k$ are conjunctions of propositions, be all the clauses in Π which have p at their head. Let $n_{c1}, n_{c2},\ldots,n_{ck}$ be the names of these clauses. Then the only-if set of p with respect to Π is $\{\neg p\vee n_{c1}\vee n_{c2}\vee\ldots\vee n_{ck}, n_{c1}\rightarrow Q_1, n_{c2}\rightarrow Q_2\ldots n_{ck}\rightarrow Q_k\}$. For a set of propositions S and the Horn clause theory T, we use only-if(T,S) to denote the union of only-if sets of all the propositions in S with respect to T. We use only-if(T) to denote only-if(T,props(T)) where props(T) is the set of all propositions in T.

Since each Q_i is a conjunction of propositions, a sentence of the form $n_{ci}\rightarrow Q_i$ is equivalent to the set of clauses $\{n_{ci}\rightarrow q_{i1}, n_{ci}\rightarrow q_{i2},\ldots,n_{ci}\rightarrow q_{in}\}$ where $Q_i \equiv q_{i1}\wedge q_{i2}\wedge\ldots\wedge q_{in}$. Thus computing the clausal form of only-if sets is trivial.

Example: Let the clauses in T with p at their head be $p\leftarrow q,r$ and $p\leftarrow s,t$. Let the names of these clauses be n_1 and n_2. Then the only-if set of p with respect to T is $\{\neg p\vee n_1\vee n_2, n_1\rightarrow q, n_1\rightarrow r, n_2\rightarrow s, n_2\rightarrow t\}$

Given the abductive problem $<\Pi,A,g>$, we define the pseudo-completion of Π to be $\Pi\cup$only-if(Π,Θ). (Recall the definition of Θ given on the first page). From the discussion above it is clear that we can assume the pseudo-completion to be in clausal form, which is the major reason we chose pseudo-completions in preference to the standard notion of completion.

We compute solutions to the abductive problem $<\Pi,A,g>$ by minimizing A with respect to $\Pi\cup$only-if$(\Pi,\Theta)\cup\{g\}$. In the next section we make precise the notion of minimization we are using.

3 Minimization with respect to a theory

In order to define minimization, we need to define models.

Definition 3 Let C be a propositional clausal theory, S th set of propositions in C, and M a set of propositions. Then th truth-assignment induced by M is the assignment of *true* to a propositions in S which are in M, and *false* to thos propositions in S which are not in M. M is a model of C iff th truth assignment induced by M satisfies all the clauses in C.

Definition 4 Let C be a propositional clausal theory, and a set of propositions. Then M is a model of C whic minimizes A iff

1. M is a model of C

2. There is no other model M' of C such that $M'\cap A\subset M\cap A$

We say that Δ is a minimization of A with respect to C iff ther is a model M of C which minimizes A and $\Delta=M\cap A$

Our definition of minimization corresponds to the notion c minimal model used in circumscription, when the propositions in A are circumscribed relative to C, and all other propositions are allowed to vary [10] . It also corresponds to th notion of minimal diagnosis used in [14] , where the assumptions A will be the abnormality assumptions associated wit the components of the system.

4 Abductive solutions and minimization

Theorem 1 is the basis of our technique for finding minim: solutions of abductive problems. (Proof in Appendix 1)

Theorem 1 Let $F= <\Pi,A,g>$ be an abduction problem where Π is acyclic. Let Θ be the set of all propositions i $\Pi\cup\{g\}$ which do not occur in A. Then Δ is a minimal solutio of $<\Pi,A,g>$ iff it is a minimization of A with respect t $\Pi\cup$only-if$(\Pi,\Theta)\cup\{g\}$.

Corollary: When Π is a acyclic, $\Pi\cup$only-if$(\Pi,\Theta)\cup\{g\}$ is inconsistent iff $<\Pi,A,g>$ has no solution.

In order to use this result for finding a solution to the abuc tion problem $<\Pi,A,g>$ we proceed as follows: first we compute only-if(Π,Θ) (the pseudo-completion of Π). Then w check $\Pi\cup$only-if$(\Pi,\Theta)\cup\{g\}$ for consistency. If inconsistent, th abduction problem has no solution. If consistent, we use th algorithm in the next section to find a minimization of A wit: respect to $\Pi\cup$only-if$(\Pi,\Theta)\cup\{g\}$. The set of assumptions returned is a minimal solution to the abduction problem.

5 Finding a minimization of A

The following algorithm (reported, for example, in [6]) can be used for finding a minimization of A with respect to C. This algorithm assumes that C is consistent. (M holds a set of literals, N and S hold a set of propositions, a holds a proposition):

```
M:={};N:={};S:=A;
while S≠{}
{
    choose a from S;
    S:=S - a;
    if $\{\neg a\} \cup C \cup M$ is consistent $M:=M \cup \{\neg a\}$
    else $N:=N \cup \{a\}$;
}
```

Notice that the selection of a from S in the first step ol the loop is non-deterministic, but this Is a don't care type of non-determinism, i.e. no matter which proposition is chosen, the algorithm will succeed and there is no need for backtracking.

It is easy to prove that, for propositional C and A, this algorithm always terminates, and when it does, N holds a minimization of A with respect to C. Let k be the size of A. Then it is clear that the cost of computing a minimization of A with respect to C using this algorithm is k times the cost of checking the consistency of CuM, where M is a set of unit clauses whose size is bounded by the size of A.

In general, checking the consistency of CuM is an NP-complete problem. But there is a class of theories for which the cost of consistency checking, using unit resolution, is linear with respect to the size of the theory. We investigate this class, and the repercussions for finding solutions to abduction problems, next.

6 Unit Resolution

Unit resolution is a restriction of the resolution rule where one of the resolvent must be a unit clause. Unit resolution, especially for propositional clauses, can be implemented very efficiently. But unit resolution is not a complete rule of inference, i.e. for some clausal theories, there are theorems which can not be proved using unit resolution. For a clausal theory C and a clause s, we say that s is *unit-derivable* from C iff there is a derivation of s from C in which every step is a unit resolution.

Definition 5 A propositional clausal theory C is unit refutable iff, for every set of unit clauses U, if $C \cup U$ is inconsistent, then the empty clause is unit-derivable from $C \cup U$.

Notation: For a clause c, size(c) denotes the number of literals in c. For a set of clauses C, size(C) denotes the sum of the sizes of all the clauses in C.

Lemma 1 Let C be a unit-refutable propositional theory, and U a set of unit clauses. Then the cost of checking the consistency of $C \cup U$ is linear with respect to size($C \cup U$) (proof omitted)

Thus, if C is unit refutable, the cost of minimizing A with respect to C using the above algorithm is bounded by O((k+n)k), where k is the size of A, and n is the size of C.

Lemma 2 Let <Π,A,g> be an abduction problem. Then size($\Pi \cup$only-if(Π,Θ))<4(size(Π)) (proof omitted)

The following lemma states the consequences of these facts for finding solutions to abduction problems:

Lemma 3 Let <Π,A,g> be an abduction problem, where $\Pi \cup$only-if(Π,Θ) is acyclic, and $\Pi \cup$only-if(Π,Θ) is unit refutable. Then the cost of finding a minimal solution to this problem is bounded by O((k+n+1)(k+1)), where k is the size of A, and n is the size of Π.

This lemma relies on using unit resolution to check the consistency of $\Pi \cup$only-if(Π,Θ)$\cup${g}, and using the algorithm above for minimizing A with respect to $\Pi \cup$only-if(Π,Θ)$\cup${g}.

At this point, two questions need to be asked: how can we say whether $\Pi \cup$only-if(Π,Θ) is unit refutable, and what happens if it isn't?

In the next section we introduce a test for unit refutability based on the topology of the connection graph of the theory.

7 When is a set of clauses unit refutable?

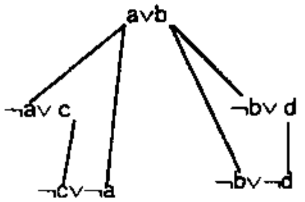
Our test for unit refutability is based on the topology of the connection graph of the set of clauses. We show that if the connection graph of a propositional clausal theory lacks certain topological features, which we have called *tied chains*, it is unit refutable. Notice that this is a sufficient, but not necessary condition. But first, we need some preliminaries.

7.1 Connection Graphs

Connection graphs were introduced in [9] to help construct more efficient resolution theorem provers.

Definition 6 Given a set of clauses C, the *connection graph* of C is the graph obtained by drawing a link between each complimentary pair of literals in the set.

Example: Let C be the set {a∨b, ¬a∨c, ¬c∨¬a, ¬b∨¬d, ¬b∨d}. Then the connection graph of C is:



We say that the clause c_1 is linked to the clause c_2 if there is a link between a literal in c_1 and a literal in c_2, i.e. c_1 has a literal such as p where c_2 has ¬p.

The next concept we need to introduce is that of a *chain*. Intuitively, a chain corresponds to a subgraph comprised of a sequence of clauses each linked to the next, with the links connecting the clause c to the previous and next clauses landing on different literals in c. For example, the following corresponds to a chain:

¬a∨c ———————— ¬c∨¬a

To define chains formally, we need the notion of a *c-triple*. A c-triple is a triple (x,e,y) where e is a clause and x and y are different literals in e.

Definition 7 Given the set of clauses C, the sequence $[(x_1,e_1,y_1),(x_2,e_2,y_2)...(x_n,e_n,y_n)]$ of c-triples is a *chain* in C iff for all k, e_k is a clause in C and $y_k=\neg x_{k+1}$.

Definition 8 $[(x_1,e_1,y_1),(x_2,e_2,y_2)...(x_n,e_n,y_n)]$ is a *tied chain* in C iff it is a chain in C and $x_1=y_n$.

Example: (¬a,¬a∨c,c)(¬c,¬c∨¬a,¬a) is a tied chain.

Theorem 2 Every prepositional clausal theory whose connection graph has no tied chains is unit refutable. (Proof in Appendix 2)

Notice that the reverse is not necessarily true, i.e. there are unit refutable sets of clauses which have tied chains.

7.2 The cost of checking for tied chains

To estimate the cost of checking for tied chains, for the set of clauses C we define the binary relations chain between literals as follows:

$\mathrm{chain}(l1,\neg l_2)$ iff l_1 and l_2 are literals in a clause in C and $l_1 \neq l_2$.

Let chain* be the transitive closure of chain. Then it is easy to show that there is a tied chain in C iff for some literal l, $\mathrm{chain}^*(l,\neg l)$ holds. Thus the cost of checking for existence of tied chains is bounded by the cost of computing the relation chain*, which is polynomial.

8 What if we don't know if $\Pi \cup \text{only-if}(\Pi,\Theta)$ is unit-refutable

We might not want to do the test for tied chains in riuonly-if(Π,Θ), or we might test and find a tied chain. In both cases, we do not know whether or not $\Pi \cup \text{only-if}(\Pi,\Theta)$ is unit-refutable. So what happens?

It is important to emphasize that while lack of tied chains in $\Pi \cup \text{only-if}(\Pi,\Theta)$ is a sufficient condition for the correctness and completeness of the algorithm, it is not a necessary condition. In other words, even if $\Pi \cup \text{only-if}(\Pi,\Theta)$ has tied chains, in many instances the algorithm returns a correct solution. Thus, when we do not know that $\Pi \cup \text{only-if}(\Pi,\Theta)$ is unit refutable, the best strategy to adopt is to use the algorithm to find a potential solution Δ anyway. But, in order to make sure that we have a solution, we can test the conditions 2 to 4 in Definition 1 in linear time (since Π is Horn). If the conditions are satisfied, then we have a solution. In other words, without tiny consideration as to the unit-refutability of $\Pi \cup \text{only-if}(\Pi,\Theta)$, the combination of the algorithm in section 5 and the further test of correctness can be considered a tractable algorithm for finding solutions to abduction problems which is correct, but incomplete.

In fact, practical experience has shown that, without any checking of $\Pi \cup \text{only-if}(\Pi,\Theta)$, the algorithm rarely fails to return a correct answer. More substantial practical evidence supporting this claim will be presented when enough data is gathered.

9 Comparison with related work and conclusion

The ATMS [31 essentially computes all the minimal solutions for the abductive problems $<\Pi,A,g_1>$, $<\Pi,A,g_2>$,.. where g_1, g_2 ,..are the propositions in Π. But the performance of ATMS can be exponential with the number of propositions in n[I2].

The use of predicate completion to characterize abduction as deduction has been proposed by a number of researchers [2] [13] [8] . In this regard, the relationship established by Theorem 1 is closest to the work of Console et. al. in [2] . They, too, consider acyclic Hom background theories, but they restrict them so that a) all definite clauses have non-abducibles at the head and b) all the denials only have abducibles atoms. Also, they do not establish a relationship between minimal solutions and the minimization of the abducible set with respect to the completion.

As far as the test for unit-refutability is concerned, to our knowledge the work reported here is new. DeKleer in [4J advocates the use of unit resolution (which he calls clausal binary constraint propagation) and gives some pragmatic techniques for making unit resolution complete for consistency checking.

A top-down algorithm for abduction using negation as failure type reasoning was reported in [5]. There are a number of published algorithms for finding genendized stable models of logic programs [7] , such as the algorithms reported in [15] . Computing general stable models is a generalised version of the abduction problem, thus all the algorithms developed for that purpose can be used for computing solutions to the abductive problems of the type considered in this paper. No tractability results have been published concerning these algorithms.

The techniques presented here (without the test for unit-refutability) have recently been implemented in the context of a general purpose abduction/reason-maintenance system. So far, the system has given promising performance, though definitive judgement has to await further experiments.

Acknowledgment: I would like to thank Bruno Bertolino, Chris Preist, Ajay Gupta and Alex Loopik for many useful discussions on the subject of this paper.

References

11] Clarke, K. *Negation as Failure.* Logic and Databases pp293-322, cd H Gallaire, J. Minker, 1978

[2] Console!., Theseider D. & Torasso, P. *On the relationship between abduction and deduction.* Journal of Logic and Computation, 2(5), Sept. 1991.

[3] deKleer, J. *An assumption based truth maintenance system,* Artificial Intelligence Journal 28, 1986, pp127-162

[4] deKleer, J., *Exploiting locality in a TMS,* in Proceedings of AAA1-90, Boston, MA (1990)264-271

[5] K. Eshghi & R. Kowalski. *Abduction compared with Negation as Failure.* Proceedings of the 6th Int. Conf. on Logic Programming, Lisbon 1989, pp234-254.

[6] G. Friedrich, G. Gotdab, W. Nejdl *Physical Impossibility Instead of Fault Models.* Proceedings of AAAI - 90, Boston, MA (1990) 331-336

[7] Kakas, A.C. & Mancarella, P., *Generalized stable models: a semantics for abduction.* Proceedings of ECAI90, 1990pp.385-391

[8] Konolige, K. *A general theory of abduction,* Proceedings AAAI symposium on Automated Abduction, Stanford 1990 pp 62-66

[9] Kowalski, R., A proof procedure using connection graphs, J ACM Vol. 22 No. 4, (1975) 572-595

[10] Lifschitz, V. Computing circumscription, Proceedings IJCAI85, 1985 pp 121-127

[11] Lloyd, J. Foundations of Logic Programming, 2nd edition, Springer Verlag (1987)

[12] McAllester, D. A widely used truth maintenance system, MIT AI Lab Memo, 1985

[13] Poole, D. Representing knowledge for logic based diagnosis. Proceedings of the International. Conference on Fifth Generation Computer Systems, Tokyo 1988 pp 1282-1290

[14] R. Reiter. A theory of diagnosis from first principles, /Artificial Intelligence Journal 32, 1987

[15] K. Satoh & N. Iwayama. Computing Abduction by using the TMS. Proceedings of the Eighth International Conference on Logic Programming, 1991.

[161 Selman, B. & Levesque, H.J., Abductive and default reasoning: a computational core, in Proceedings of AAAI-90. Boston, MA (1990) 343-348

Appendix 1: Proof of Theorem 1

Preliminaries: For a set of propositions $S=\{s_1,s_2..\}$ and a set of Horn clauses T, we use $\neg S$ to denote $\{\neg s_1,\neg s_2...\}$, and T^s to denote the set of clauses in T which have a proposition from S at their head. Let $p\leftarrow Q_1$, $p\leftarrow Q_2...p\leftarrow Q_n$ be all the clauses in T with p at their head, where Q_1, $Q_2..Q_n$ are conjunctions of propositions. Then the completion of p relative to T is $p\equiv Q_1\vee Q_2\vee...\vee Q_n$. If there is no clause in T with p at its head, then the completion of p relative to T is $\neg p$.We use Comp(T,S) to denote the completion of all propositions in S relative to T.

We state the following proposition without proof. Its proof straightforwardly follows from known results in the logic programming literature [11] .

Proposition 1 Let T be an acyclic propositional definite clause theory and S a set of propositions which includes all propositions in T. Then for all propositions p in S, $T\nvdash p$ iff $\mathrm{Comp}(T,S)\vdash\neg p$

We also need the following lemmas:

Lemma 4 Let Δ be a solution of $<\Pi,A,g>$. Let M be the minimal model of $\Pi\cup\Delta$. Then M is also the minimal model of $\Pi^{\Theta}\cup\Delta$ (Π^{Θ} is the set of all clauses in Π which have a proposition from Θ at their head).

Proof: Let the proposition p be provable from $\Pi\cup\Delta$. We show that it is also provable from $\Pi^{\Theta}\cup\Delta$. Since the minimal model of a propositional Horn clause theory is the set of propositions provable from it, this would establish the lemma.

First, we observe that if p it is abducible, then since Δ is a solution of $<\Pi,A,g>$, by definition p is in Δ.

For the case where p is non-abducible, we show that p is provable from $\Pi^{\Theta}\cup\Delta$ by induction on the length of the shortest resolution refutation proof of p from $\Pi\cup\Delta$.

Base Case: p is provable from $\Pi\cup\Delta$ in zero steps, i.e. p is a unit clause in Π. Thus it is in Π^{Θ}.

Inductive Case: Let the first step in the shortest proof of p from $\Pi\cup\Delta$ be the resolution of $\neg p$ with the clause $p\leftarrow p_1,p_2...$ Then $p_1,p_2..$ are all provable from $\Pi\cup\Delta$ with proof shorter than that of p, thus by inductive assumption they are provable from $\Pi^{\Theta}\cup\Delta$. Since $p\leftarrow p_1,p_2...$ is a clause in Π^{Θ}, it follows that p is provable from $\Pi^{\Theta}\cup\Delta$.

Lemma 5 Let T be a definite theory, with M its minimal model. Then only-if(T) has a model M' which is a superset of M, and M'\M only contains names of clauses in T.

Proof: We construct a model M' of only-if(T) as follows: Let p be a proposition in M. Then $T\vdash p$, i.e. there is a clause $p\leftarrow Q$ in T where Q is a conjunction of propositions, and all the propositions in Q are in M. Let the name of this clause be n_c. Then we add n_c to M. We repeat this procedure for all the propositions in M. It is easy to verify that M' satisfies all the clauses in only-if(T).

Proposition 2 For all sets of assumptions Δ, Δ is a solution of $<\Pi,A,g>$ iff $\Pi\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$ is consistent

Proof of ←

Let $L=\Theta\cup A$, i.e. L is the set of all propositions in the language of $<\Pi,A,g>$. Now $\Pi\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$ is consistent thus $\Pi^{\Theta}\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$ is consistent. It is easy to show that $\mathrm{Comp}(\Pi^{\Theta}\cup\Delta,L)$ is a logical consequence of $\Pi^{\Theta}\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)$. Thus $\mathrm{Comp}(\Pi^{\Theta}\cup\Delta,L)\cup\{g\}$ is consistent, which shows that $\mathrm{Comp}(\Pi^{\Theta}\cup\Delta,L)\nvdash\neg g$. Thus, since $\Pi^{\Theta}\cup\Delta$ is an acyclic propositional definite theory, by Proposition 1 $\Pi^{\Theta}\cup\Delta\vdash g$.

It remains to show that ($a\in A$ and $\Delta+\Pi\vdash a$) → $a\in\Delta$. Suppose this is not true, i.e. there is an a such that $a\in A$ and $\Delta+\Pi\vdash a$ and $a\notin\Delta$. But if $a\notin\Delta$, it will be in $A\backslash\Delta$, which means $\neg a$ is in $\neg(A\backslash\Delta)$. Thus $\Pi\cup\Delta\cup\neg(A\backslash\Delta)$ would be inconsistent, which is contrary to assumption. QED

Proof of →

We show $\Pi\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$ is consistent by showing that it has a model. Now since Δ is a solution of $<\Pi,A,g>$ by definition $\Pi\cup\Delta$ is consistent. Let M be the minimal model of $\Pi\cup\Delta$. Since Δ is a solution of $<\Pi,A,g>$ by definition $\Pi\cup\Delta\vdash g$, thus $g\in M$. By Lemma 4 M is the minimal model of $\Pi^{\Theta}\cup\Delta$, thus by Lemma 5 there is a model M' of only-if($\Pi^{\Theta}\cup\Delta$) which is a superset of M and none of the propositions in M'\M occur in Π or A. But only-if($\Pi^{\Theta}\cup\Delta$) is a superset of only-if(Π,Θ), thus M' satisfies Π,Δ, only-if(Π,Θ), and g, so we need only to show that M' satisfies $\neg(A\backslash\Delta)$. Let $\neg a$ be a member of $\neg(A\backslash\Delta)$. Since Δ is a solution of $<\Pi,A,g>$, by definition a is not provable from $\Pi\cup\Delta$, thus $a\notin M$, thus $a\notin M'$. Therefore M' satisfies $\neg(A\backslash\Delta)$. QED

Proof of Theorem 1

It easily follows from Proposition 2 that Δ is a minimal solution of $<\Pi,A,g>$ iff Δ is a minimal set of assumptions such that $\Pi\cup\text{only-if}(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$ is consistent. It is also easy to show that, for any clausal theory C, Δ is a minimization of A with respect to C iff Δ is a minimal subset of A such that $C\cup\Delta\cup\neg(A\backslash\Delta)$ is consistent. Thus Δ is a minimal solution of

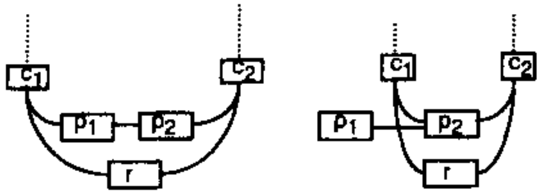
$<\Pi,A,g>$ iff Δ is a minimization of A with respect to $\Pi\cup$only-if$(\Pi,\Theta)\cup\Delta\cup\neg(A\backslash\Delta)\cup\{g\}$.

Appendix 2: Proof of Theorem 2

First, we need a few preliminary definitions and propositions.

Proposition 3 Let C_1 be derived from C by one resolution step. Then if C_1 has a tied chain, so does C.

Sketch of Proof: Let C_1 have a tied chain, and be derived from C by the resolution of the clauses p_1 and p_2 in C. Let the resolvent of p_1 and p_2 be r. If the tied chain in C_1 does not involve r, then it is a tied chain of C too. If it involves r, and r is not the first or last clause in the chain, then one of the following situations obtains (c_1 and c_2 are the clauses before and after r in the chain):



In the situation depicted on the left, r inherits the link to c_1 from p_1 and the link to c_2 from p_2. In the situation depicted on the right, it inherits both links from one parent, say p_2. In both cases from the topology of the graphs it is easy to see that there is a tied chain in C. There is a similar argument when r is the first or last clause in the chain.

Convention: The length of a (unit) resolution derivation of s from C is the number of (unit) resolution steps in that derivation. We define the length of the derivation to be 0 iff $s\in C$.

Proposition 4 Let C be a consistent propositional clausal theory without tied chains which has a unit theorem u. Then C has at least one unit clause.

Proof: Let $\sigma(C,u)$ be the length of the shortest resolution derivation of u from C. We prove the proposition by induction on $\sigma(C,u)$.

Base Case: $\sigma(C,u)=0$, so $u\in C$.

Inductive case: Let $\sigma(C,u)=n$. We assume that the proposition holds for all sets of clauses such as C_i where $\sigma(C_i,u)<n$, and then we prove that it holds for C.

Proof: Let C, C_{n-1},C_{n-2},....C_0 be one of the shortest resolution derivation of u from C. Thus C_{i-1} is derived from C_i by one resolution, and C_0 contains u. Now, $\sigma(C_{n-1},u)=n-1$. By Proposition 4, C_{n-1} does not have any tied chains. Thus, by inductive assumption, C_{n-1} contains a unit clause. Let π_1 and π_2 be the two clauses which are resolved in C, giving the resolvent ρ. Thus $C_{n-1}=C\cup\{\rho\}$. Now since $C\cup\{\rho\}$ contains a unit clause, if C does not contain a unit clause, ρ must be unit. Thus for some literal λ, $\pi_1=\rho\vee\lambda$ and $\pi_2=\neg\lambda\vee\rho$. (because this is the only way a unit clause can be derived from two non-unit clauses). But this means there is a tied chain in C, contradictory to assumption.

$$\rho\vee\lambda \text{———} \neg\lambda\vee\rho$$

Proposition 5 Let C be an inconsistent propositional clausal theory without tied chains. Then C has at least one unit clause.

Proof: Let C' be a minimally inconsistent subset of C. Let $l_1\vee l_2\vee..l_k$ be a clause in C'. Let $C"=C'\backslash\{l_1\vee l_2\vee..l_k\}$. Then C" is consistent, and the unit clauses $\neg l_1$, $\neg l_2..\neg l_k$ are theorems of C". Thus by Proposition 4 C" has a unit clause, and since $C"\subset C$, C has a unit clause.

Definition 9 Let C be a consistent propositional clausal theory, and λ a unit clause in C. Then C_λ is the set of clauses derived by deleting from C all clauses which have λ as a literal, and deleting $\neg\lambda$ from the all other clauses.

Proposition 6 Let C be without tied chains. Then for all λ for which C_λ is defined, C_λ is without tied chains. (This is a special case of Proposition 3)

Proposition 7 Let C be a consistent propositional clausal theory, and λ a unit clause in C. Let u be a theorem of C which is different from λ. Then u is a theorem of C_λ. (proof omitted)

Proposition 8 Let C be a consistent set of propositional clauses without tied chains, and let u be a theorem of C. Then there is a unit derivation of u from of C.

Proof by induction on the number of proposition symbols in C.

Base Case: The number of proposition symbols in C is one. Then $C=\{u\}$ and there is a zero step unit derivation of u from C.

Inductive Case: Let n be the number of propositional symbols in C. Assume that Proposition 8 holds for all C_i such that the number of propositional symbols in C_i is less than n. Then we prove that it holds for C.

Proof of the induction step: By Proposition 4 C has at least one unit clause. Let this clause be λ. If $\lambda=u$, we are done. Otherwise, by Proposition 7 u is provable from C_λ. C_λ has one less proposition symbol than C, and By Proposition 6 it does not have any tied chains, so by inductive assumption there is a unit derivation of u from C_λ. But C_λ can be derived from C by a series of unit resolutions (i.e. resolving the unit clause λ with all the clauses in C with which it can be resolved) thus there is a unit derivation of u from C.

Proof of Theorem 2 Let C be a set of clauses without any tied chains, and U a set of unit clauses, where $C\cup U$ is inconsistent. Let C' be a minimally inconsistent subset of C, and $l_1\vee l_2...\vee l_k$ a clause in C'. Let $C"=C'\backslash\{l_1\vee l_2...\vee l_k\}$. Since C has no tied chains, neither does C". Now, C" is consistent, and $\neg l_1$, $\neg l_2...\neg l_k$ are unit theorems of C". So by Proposition 8 there is a unit derivation of the unit clauses $\neg l_1$, $\neg l_2,...\neg l_k$ from C", thus since $l_1\vee l_2...\vee l_k$ is a clause in C, there is a unit derivation of the empty clause from C (by first deriving $\neg l_1$, $\neg l_2...\neg l_k$ from C, and then resolving these with $l_1\vee l_2...\vee l_k$ in k unit resolution steps).