Artificial Inteligence - Resolution for propositional calculus

Lila Kari

The University of Western Ontario

Artificial Intelligence

- The major interest of computer scientists in propositional and predicate calculus has been to exploit its expressive power to prove theorems:

  Theorem: $\text{Premise } 1, \text{Premise } 2, \ldots, \text{Premise } n \vdash \text{ Conclusion}$

- In the field of Artificial Intelligence, there have been many attempts to construct programs that could prove theorems automatically.
- Given a set of axioms and a technique for deriving new theorems from old theorems and axioms, would such a program be able to prove a particular theorem?

Automated theorem proving

- Early attempts faltered because there seemed to be no efficient technique for deriving new theorems.
- 1965: J.A.Robinson at Syracuse University discovered the technique called resolution.

John Allan Robinson, born 1928

Resolution

Resolution theorem proving is a method of formal derivation (deduction) that has the following features:

Resolution

Resolution theorem proving is a method of formal derivation (deduction) that has the following features:

- The only formulas allowed in resolution theorem proving are disjunctions of literals.

Resolution

Resolution theorem proving is a method of formal derivation (deduction) that has the following features:

- The only formulas allowed in resolution theorem proving are disjunctions of literals.
- A disjunction of literals is called a clause. Hence, all formulas involved in resolution theorem proving must be clauses.

Resolution

Resolution theorem proving is a method of formal derivation (deduction) that has the following features:

- The only formulas allowed in resolution theorem proving are disjunctions of literals.
- A disjunction of literals is called a clause. Hence, all formulas involved in resolution theorem proving must be clauses.
- Resolution follows the refutation principle; that is, it shows that the negation of the conclusion is inconsistent with the premises.

Resolution

Resolution theorem proving is a method of formal derivation (deduction) that has the following features:

- The only formulas allowed in resolution theorem proving are disjunctions of literals.
- A disjunction of literals is called a clause. Hence, all formulas involved in resolution theorem proving must be clauses.
- Resolution follows the refutation principle; that is, it shows that the negation of the conclusion is inconsistent with the premises.
- There is essentially only one rule of formal deduction, resolution.

- In a refutation system one proves that the argument $A_1, A_2, \ldots, A_n \vdash \neg C$ is valid by showing that $A_1, A_2, \ldots, A_n$ and $\neg C$ cannot all be true.
- In other words one shows that the formulas

$$A_1, A_2, \ldots, A_n, \neg C$$

  are inconsistent.
- This is shown by proving that for some formula P, both P and $\neg P$ can be derived.

Input for resolution algorithms - clauses

- In general, one can convert any formula into one or more clauses.

Input for resolution algorithms - clauses

- In general, one can convert any formula into one or more clauses.
- To do this, one first converts the formula into a conjunction of disjunctions; that is, one converts the formula into conjunctive normal form.

Input for resolution algorithms - clauses

- In general, one can convert any formula into one or more clauses.
- To do this, one first converts the formula into a conjunction of disjunctions; that is, one converts the formula into conjunctive normal form.
- Each term of the conjunction is then made into a clause of its own.

Clauses

Example: Convert $P \rightarrow (Q \wedge R)$ into clauses.

Clauses

Example: Convert $P \to (Q \wedge R)$ into clauses.

Solution.

Clauses

Example: Convert $P \to (Q \wedge R)$ into clauses.

Solution.
We first eliminate the $\to$ by writing $\neg P \vee (Q \wedge R)$.

Clauses

Example: Convert $P \to (Q \wedge R)$ into clauses.

Solution.
We first eliminate the $\to$ by writing $\neg P \vee (Q \wedge R)$.

We then apply the distributive law to obtain

$$P \to (Q \wedge R) \models\!\!\!\!\dashv (\neg P \vee Q) \wedge (\neg P \vee R).$$

Clauses

Example: Convert $P \to (Q \land R)$ into clauses.

Solution.
We first eliminate the $\to$ by writing $\neg P \lor (Q \land R)$.

We then apply the distributive law to obtain

$$P \to (Q \land R) \models\!\!\!\mid (\neg P \lor Q) \land (\neg P \lor R).$$

This yields the two clauses $\neg P \lor Q$ and $\neg P \lor R$.

Only one rule of formal deduction: resolution

Resolution

Only one rule of formal deduction: resolution

Resolution

- Two clauses can be resolved if and only if they contain two complementary literals.

Only one rule of formal deduction: resolution

Resolution

- Two clauses can be resolved if and only if they contain two complementary literals.
- In this case, they give rise to a new clause, called the resolvent.

Only one rule of formal deduction: resolution

Resolution

- Two clauses can be resolved if and only if they contain two complementary literals.
- In this case, they give rise to a new clause, called the resolvent.
- If the complementary literals are P and $\neg P$, one says the resolution in on P (or over P).

Only one rule of formal deduction: resolution

Resolution

- Two clauses can be resolved if and only if they contain two complementary literals.
- In this case, they give rise to a new clause, called the resolvent.
- If the complementary literals are P and $\neg P$, one says the resolution in on P (or over P).
- The clauses giving rise to the resolvent are called parent clauses.

Only one rule of formal deduction: resolution

Resolution

- Two clauses can be resolved if and only if they contain two complementary literals.
- In this case, they give rise to a new clause, called the resolvent.
- If the complementary literals are P and $\neg P$, one says the resolution in on P (or over P).
- The clauses giving rise to the resolvent are called parent clauses.
- The resolvent on P is the disjunction of all literals of the parent clauses, except that P and $\neg P$ are omitted from the resolvent.

Example

Find the resolvent of $P \vee \neg Q \vee R$ and $\neg S \vee Q$.

Example

Find the resolvent of $P \vee \neg Q \vee R$ and $\neg S \vee Q$.

Solution.
The two clauses $P \vee \neg Q \vee R$ and $\neg S \vee Q$ can be resolved over Q because Q is negative in the first clause and positive in the second.

Example

Find the resolvent of $P \vee \neg Q \vee R$ and $\neg S \vee Q$.

Solution.
The two clauses $P \vee \neg Q \vee R$ and $\neg S \vee Q$ can be resolved over Q because Q is negative in the first clause and positive in the second.

The resolvent is the disjunction of $P \vee R$ with $\neg S$, which yields $P \vee R \vee \neg S$.

Soundness of resolution-based formal deduction

The resolvent is logically implied by its parent clauses, which makes resolution a sound rule of formal deduction.

Soundness of resolution-based formal deduction

The resolvent is logically implied by its parent clauses, which makes resolution a sound rule of formal deduction.

To see this, let P be a propositional variable, and let A and B be (possibly empty) clauses.

Soundness of resolution-based formal deduction

The resolvent is logically implied by its parent clauses, which makes resolution a sound rule of formal deduction.

To see this, let P be a propositional variable, and let A and B be (possibly empty) clauses.
One has

$$P \vee A, \neg P \vee B \models A \vee B$$

This is valid for the following reasons.

Soundness of resolution

- If P is false, then A must be true, because otherwise $P \vee A$ is false.

Soundness of resolution

- If P is false, then A must be true, because otherwise $P \vee A$ is false.
- Similarly, if P is true, then B must be true, because otherwise $\neg P \vee B$ is false.

Soundness of resolution

- If P is false, then A must be true, because otherwise $P \vee A$ is false.
- Similarly, if P is true, then B must be true, because otherwise $\neg P \vee B$ is false.
- Since P must be true or false, either A or B must be true, and the result follows.

Soundness of resolution

- If P is false, then A must be true, because otherwise $P \vee A$ is false.
- Similarly, if P is true, then B must be true, because otherwise $\neg P \vee B$ is false.
- Since P must be true or false, either A or B must be true, and the result follows.
- Of course, $A \vee B$ is the resolvent of the parent clauses $P \vee A$ and $\neg P \vee B$ on P, which proves the soundness of resolution.

Unit clauses

Resolution is particularly effective when one of the parent clause is a unit clause, that is, a clause that contains only one literal.

Unit clauses

Resolution is particularly effective when one of the parent clause is a unit clause, that is, a clause that contains only one literal.

Example: The resolution of $\neg P \vee Q \vee R$ and $\neg R$. The resolvent is $\neg P \vee Q$.

Unit clauses

Resolution is particularly effective when one of the parent clause is a unit clause, that is, a clause that contains only one literal.

Example: The resolution of $\neg P \vee Q \vee R$ and $\neg R$. The resolvent is $\neg P \vee Q$.

Example: The resolution of $P \vee Q$ with $\neg P$ yields Q, which agrees with the disjunctive syllogism.

Unit clauses

Resolution is particularly effective when one of the parent clause is a unit clause, that is, a clause that contains only one literal.

Example: The resolution of $\neg P \vee Q \vee R$ and $\neg R$. The resolvent is $\neg P \vee Q$.

Example: The resolution of $P \vee Q$ with $\neg P$ yields Q, which agrees with the disjunctive syllogism.

Example: The two clauses $\neg P$ and P have the empty clause as a resolvent, which is correct, since P and $\neg P$ are contradictory and therefore false like the empty clause.

Prove Modus Ponens by resolution

$$P, P \to Q \;\vdash\; Q$$

Prove Modus Ponens by resolution

$$P, P \to Q \vdash Q$$

1.	P	Premise
2.	$\neg P \lor Q$	Premise
3.	$\neg Q$	Negation of conclusion
4.	Q	Resolvent of 1, 2
5.	0	Resolvent of 3, 4

Prove Hypothetical Syllogism by resolution

$$P \to Q, Q \to R \vdash P \to R$$

Prove Hypothetical Syllogism by resolution

$$P \to Q, Q \to R \vdash P \to R$$

1.	$\neg P \vee Q$	Premise
2.	$\neg Q \vee R$	Premise
3.	P	Derived from negation of conclusion
4.	$\neg R$	Derived from the negation of conclusion
5.	Q	Resolvent of 1, 3
6.	$\neg Q$	Resolvent of 2, 4
7.	0	Resolvent of 5, 6

Resolution strategies

When doing resolution automatically, one has to decide in which order to resolve the clauses. This order can greatly affect the time needed to find a contradiction. Strategies include:

- Unit resolution: all resolutions involve at least one unit clause. Moreover, preference is given to clauses that have not been used yet.
- Set of support strategy
- Davis Putnam procedure

Example of unit resolution

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4$, $P_3 \rightarrow P_5$, $P_6 \rightarrow \neg P_5$ and P_6.

Example of unit resolution

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4$, $P_3 \rightarrow P_5$, $P_6 \rightarrow \neg P_5$ and P_6.

1.	$\neg P_1 \vee P_2$	Premise
2.	$\neg P_2$	Premise
3.	$P_1 \vee P_3 \vee P_4$	Premise
4.	$\neg P_3 \vee P_5$	Premise
5.	$\neg P_6 \vee \neg P_5$	Premise
6.	P_6	Premise
7.	$\neg P_4$	Negation of conclusion
8.	$\neg P_1$	Resolvent of 1, 2
9.	$\neg P_5$	Resolvent of 5, 6
10.	$P_1 \vee P_3$	Resolvent of 3, 7
11.	$\neg P_3$	Resolvent of 4, 9
12.	P_3	Resolvent of 8, 10
13.	0	Resolvent of 11, 12

Unit resolution is not complete

- The unit resolution is not complete.

Unit resolution is not complete

- The unit resolution is not complete.
- This is demonstrated by the following example.

Unit resolution is not complete

- The unit resolution is not complete.
- This is demonstrated by the following example.
- The premises are $Q \vee R$, $Q \vee \neg R$, and $\neg Q \vee R$, and the conclusion is $Q \wedge R$.

Unit resolution is not complete

- The unit resolution is not complete.
- This is demonstrated by the following example.
- The premises are $Q \vee R$, $Q \vee \neg R$, and $\neg Q \vee R$, and the conclusion is $Q \wedge R$.
- In this case there is no unit clause, which makes unit resolution impossible.

Set of support strategy

- One partitions all clauses into two sets, the set of support and the auxiliary set.

Set of support strategy

- One partitions all clauses into two sets, the set of support and the auxiliary set.
- The auxiliary set is formed in such a way that the formulas in it are not contradictory.

Set of support strategy

- One partitions all clauses into two sets, the set of support and the auxiliary set.
- The auxiliary set is formed in such a way that the formulas in it are not contradictory.
- For instance, the premises are usually not inconsistent (not contradictory). The inconsistency only arises after one adds the negation of the conclusion.

Set of support strategy

- One partitions all clauses into two sets, the set of support and the auxiliary set.
- The auxiliary set is formed in such a way that the formulas in it are not contradictory.
- For instance, the premises are usually not inconsistent (not contradictory). The inconsistency only arises after one adds the negation of the conclusion.
- One often uses the premises as the initial auxiliary set and the negation of the conclusion as the initial set of support.

Set of support styrategy

- Since one cannot derive any contradiction by resolving clauses within the auxiliary set, one avoids such resolutions.

Set of support styrategy

- Since one cannot derive any contradiction by resolving clauses within the auxiliary set, one avoids such resolutions.
- Stated positively, each resolution takes at least one clause from the set of support.

Set of support styrategy

- Since one cannot derive any contradiction by resolving clauses within the auxiliary set, one avoids such resolutions.
- Stated positively, each resolution takes at least one clause from the set of support.
- The resolvent is then added to the set of support.

Set of support styrategy

- Since one cannot derive any contradiction by resolving clauses within the auxiliary set, one avoids such resolutions.
- Stated positively, each resolution takes at least one clause from the set of support.
- The resolvent is then added to the set of support.
- Resolution with the set of support strategy is complete.

Example

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4$, $P_3 \rightarrow P_5$, $P_6 \rightarrow \neg P_5$ and P_6, by using the set of support strategy.

Example

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4$, $P_3 \rightarrow P_5$, $P_6 \rightarrow \neg P_5$ and P_6, by using the set of support strategy.

Initially the set of support is given by $\neg P_4$, the negation of the conclusion.

Example

Prove P_4 from $P_1 \to P_2, \neg P_2, \neg P_1 \to P_3 \vee P_4$, $P_3 \to P_5$, $P_6 \to \neg P_5$ and P_6, by using the set of support strategy.

Initially the set of support is given by $\neg P_4$, the negation of the conclusion.

One then does all the possible resolutions involving $\neg P_4$, then all possible resolutions involving the resulting resolvents, and so on.

Example

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4$, $P_3 \rightarrow P_5$, $P_6 \rightarrow \neg P_5$ and P_6, by using the set of support strategy.

Initially the set of support is given by $\neg P_4$, the negation of the conclusion.

One then does all the possible resolutions involving $\neg P_4$, then all possible resolutions involving the resulting resolvents, and so on.

If the initial 7 clauses are omitted, this yields the following derivation:

8.	$P_1 \vee P_3$	Resolvent of 7, 3
9.	$P_2 \vee P_3$	Resolvent of 1, 8
10.	P_3	Resolvent of 2, 9
11.	P_5	Resolvent of 4, 10
12.	$\neg P_6$	Resolvent of 5, 11
13.	0	Resolvent of 6, 12

Davis Putnam Procedure - Inputs

- Any clause corresponds to a set of literals, that is, the literals contained within the clause.

Davis Putnam Procedure - Inputs

- Any clause corresponds to a set of literals, that is, the literals contained within the clause.
- For instance, the clause $P \vee \neg Q \vee R$ corresponds to the set $\{P, \neg Q, R\}$ and $\neg S \vee Q$ corresponds to the set $\{\neg S, Q\}$.

Davis Putnam Procedure - Inputs

- Any clause corresponds to a set of literals, that is, the literals contained within the clause.
- For instance, the clause $P \vee \neg Q \vee R$ corresponds to the set $\{P, \neg Q, R\}$ and $\neg S \vee Q$ corresponds to the set $\{\neg S, Q\}$.
- Since the order of the literals in a disjunction is irrelevant, and since the same is true for the multiplicity in which the terms occur, the set associated with the clause completely determines the clause.

Davis Putnam Procedure - Inputs

- Any clause corresponds to a set of literals, that is, the literals contained within the clause.
- For instance, the clause $P \vee \neg Q \vee R$ corresponds to the set $\{P, \neg Q, R\}$ and $\neg S \vee Q$ corresponds to the set $\{\neg S, Q\}$.
- Since the order of the literals in a disjunction is irrelevant, and since the same is true for the multiplicity in which the terms occur, the set associated with the clause completely determines the clause.
- For this reason, one frequently treats clauses as sets, which allows one to speak of the union of two clauses.

Resolution as operation between sets

Resolution as operation between sets

- If clauses are represented as sets, one can write the resolvent of two clauses A and B on P as follows:

$$C = (A \cup B) \setminus \{P, \neg P\}.$$

- In words, the resolvent is the union of all literals of A and B except that the two literals involving P are omitted.

The Davis-Putnam procedure

Given as input a nonempty set of clauses in the propositional variables $P_1, P_2, \ldots P_n$, the Davis Putnam Procedure (DPP) repeats the following steps until there are no variables left:

The Davis-Putnam procedure

Given as input a nonempty set of clauses in the propositional variables $P_1, P_2, \ldots P_n$, the Davis Putnam Procedure (DPP) repeats the following steps until there are no variables left:

- Discard all clauses that have both a literal L and its complement $\neg L$ in them.

The Davis-Putnam procedure

Given as input a nonempty set of clauses in the propositional variables $P_1, P_2, \ldots P_n$, the Davis Putnam Procedure (DPP) repeats the following steps until there are no variables left:

- Discard all clauses that have both a literal L and its complement $\neg L$ in them.
- Choose a variable P appearing in one of the clauses.

The Davis-Putnam procedure

Given as input a nonempty set of clauses in the propositional variables $P_1, P_2, \ldots P_n$, the Davis Putnam Procedure (DPP) repeats the following steps until there are no variables left:

- Discard all clauses that have both a literal L and its complement $\neg L$ in them.
- Choose a variable P appearing in one of the clauses.
- Add all possible resolvents using resolution on P to the set of clauses.

The Davis-Putnam procedure

Given as input a nonempty set of clauses in the propositional variables $P_1, P_2, \dots P_n$, the Davis Putnam Procedure (DPP) repeats the following steps until there are no variables left:

- Discard all clauses that have both a literal L and its complement $\neg L$ in them.
- Choose a variable P appearing in one of the clauses.
- Add all possible resolvents using resolution on P to the set of clauses.
- Discard all clauses with P or $\neg P$ in them.

DPP - Eliminating a variable

- We refer to the preceding sequence as eliminating the variable P.

DPP - Eliminating a variable

- We refer to the preceding sequence as eliminating the variable P.
- If in some step one resolves $\{P\}$ and $\{\neg P\}$ then one has the empty clause, and it will be the only clause at the end of the procedure.

DPP - Eliminating a variable

- We refer to the preceding sequence as eliminating the variable P.
- If in some step one resolves $\{P\}$ and $\{\neg P\}$ then one has the empty clause, and it will be the only clause at the end of the procedure.
- If one never has a pair $\{P\}$ and $\{\neg P\}$ to resolve, then all the clauses will be thrown out and the output will be no clauses.

DPP - Eliminating a variable

- We refer to the preceding sequence as eliminating the variable P.
- If in some step one resolves $\{P\}$ and $\{\neg P\}$ then one has the empty clause, and it will be the only clause at the end of the procedure.
- If one never has a pair $\{P\}$ and $\{\neg P\}$ to resolve, then all the clauses will be thrown out and the output will be no clauses.
- So the output of DPP either the empty clause or no clauses.

DPP - Eliminating a variable

- We refer to the preceding sequence as eliminating the variable P.
- If in some step one resolves $\{P\}$ and $\{\neg P\}$ then one has the empty clause, and it will be the only clause at the end of the procedure.
- If one never has a pair $\{P\}$ and $\{\neg P\}$ to resolve, then all the clauses will be thrown out and the output will be no clauses.
- So the output of DPP either the empty clause or no clauses.
- This may seem rather subtle but just think of the difference between arriving in the library with (1) an empty backpack and (2) no backpack.

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.
- Discard members of S_i in which a literal and its complement appear, to obtain S_i'.

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.
- Discard members of S_i in which a literal and its complement appear, to obtain S_i'.
- Let T_i be the set of parent clauses in S_i' in which P_i or $\neg P_i$ appears.

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.
- Discard members of S_i in which a literal and its complement appear, to obtain S_i'.
- Let T_i be the set of parent clauses in S_i' in which P_i or $\neg P_i$ appears.
- Let U_i be the set of resolvent clauses obtained by resolving (over P_i) every pair of clauses $C \cup \{P_i\}$ and $D \cup \{\neg P_i\}$ in T_i.

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.
- Discard members of S_i in which a literal and its complement appear, to obtain S_i'.
- Let T_i be the set of parent clauses in S_i' in which P_i or $\neg P_i$ appears.
- Let U_i be the set of resolvent clauses obtained by resolving (over P_i) every pair of clauses $C \cup \{P_i\}$ and $D \cup \{\neg P_i\}$ in T_i.
- Set S_{i+1} equal to $(S_i' \setminus T_i) \cup U_i$. (Eliminate P_i).

Davis-Putnam algorithm

- Let $S_1 = S$.
- Let $i = 1$.
- LOOP until $i = n + 1$.
- Discard members of S_i in which a literal and its complement appear, to obtain S_i'.
- Let T_i be the set of parent clauses in S_i' in which P_i or $\neg P_i$ appears.
- Let U_i be the set of resolvent clauses obtained by resolving (over P_i) every pair of clauses $C \cup \{P_i\}$ and $D \cup \{\neg P_i\}$ in T_i.
- Set S_{i+1} equal to $(S_i' \setminus T_i) \cup U_i$. (Eliminate P_i).
- Let i be increased by 1.
- ENDLOOP.
- Output S_{n+1}.

Example

Let us apply the Davis-Putnam procedure to the clauses

$$\{\neg P, Q\}, \{\neg Q, \neg R, S\}, \{P\}, \{R\}, \{\neg S\}$$

Example

Let us apply the Davis-Putnam procedure to the clauses

$$\{\neg P, Q\}, \{\neg Q, \neg R, S\}, \{P\}, \{R\}, \{\neg S\}$$

- Eliminating P gives $\{Q\}, \{\neg Q, \neg R, S\}, \{R\}, \{\neg S\}$ (This is S_2 and S_2').
- Eliminating Q gives $\{\neg R, S\}, \{R\}, \{\neg S\}$. (This is S_3 and S_3'.)
- Eliminating R gives $\{S\}, \{\neg S\}$. (This is S_4 and S_4'.)
- Eliminating S gives $\{\}$. (This is S_5.)

So the output is the empty clause.

Comments

- If the set of clauses is more complicated, before each phase of applying resolution we number the clauses (the T_i steps) and in the next phase (the U_i steps) we provide two numbers with each clause, to describe the two clauses used to provide that resolvent.

Comments

- If the set of clauses is more complicated, before each phase of applying resolution we number the clauses (the T_i steps) and in the next phase (the U_i steps) we provide two numbers with each clause, to describe the two clauses used to provide that resolvent.
- If the output of DPP is the empty clause, this indicates that both P and $\neg P$ were produced, that is, the clauses that originated from the premises and negation of the conclusion are inconsistent, that is, the original argument (theorem) is valid.

Comments

- If the set of clauses is more complicated, before each phase of applying resolution we number the clauses (the T_i steps) and in the next phase (the U_i steps) we provide two numbers with each clause, to describe the two clauses used to provide that resolvent.
- If the output of DPP is the empty clause, this indicates that both P and $\neg P$ were produced, that is, the clauses that originated from the premises and negation of the conclusion are inconsistent, that is, the original argument (theorem) is valid.
- If the output of DPP is no clause, no contradiction can be found, and the original argument (theorem) is not valid.

Soundness and Completeness of DPP

Theorem [The DPP is sound and complete].

Let S be a finite set of clauses. Then S is not satisfiable iff the output of the Davis-Putnam procedure is the empty clause.