Abstract
This paper explores the possibility of incorporating conditional proof which is an important deduction strategy in deductive logic into automated deduction to enhance the power of the deduction. It analyses the necessity and feasibility and suggests detailed procedure for the incorporation.

Keywords
automated deduction, conditional proof

1. Introduction
The aim of automated deduction is to simulate human being's logic deduction, so that a computer can be employed to conduct the deduction which normally conducted by our human beings. However automated deduction today is far from being satisfaction. There are vast quantity of problems which can be easily solved by our human beings are still outstanding questions for automated deduction [3].

In this paper, we attempt to introduce conditional proof [2] which is a widely adapted proof technique in deductive logic, into automated deduction to enhance the power of deduction strategies. We will, in particular, discuss the incorporation of conditional proof into SLD-resolution [1] which is an important derivation strategy in this area, and examine the impact of this incorporation.

Conditional proof is a widely applied technique in deductive logic. The technique is to make an assumption during a deduction to produce a conditional. The idea of making assumption has been mentioned in [11] when making assumption is defined as the trigger technique in GOPT-resolution [5, 6, 7, 8]. However the discussion was limited to GOPT-resolution which is a bottom-up deduction strategy.

Many different deduction strategies have been developed in automated deduction and basically they can be classified into three categories – top-down, bottom-up and combined top-down and bottom-up. Strategies in the first category start their deduction from the top -- the goal, and backtrack down to the bottom -- the facts. The second group starts its deduction from the bottom -- a set of facts and a set of rules. It derives new facts from the existing facts and rules, adds the new facts into the fact set, again derives new facts based on the updated fact set till the stage that no new fact can be derived or the goal is proved. The third approach is a combined of the first and second approaches. In other words, the deduction starts from both top and bottom concurrently till the sub-goals and the derived facts meet. In this paper, we will mainly concentrated on the integration of conditional proof into the strategies in the first category -- those top-down approaches.

2. Analysis
This section presents a precise definition of conditional proof and discusses the necessity of introducing conditional proof into automated deduction. It then concentrates on the detailed procedure of incorporating conditional proof into SLD-resolution.

2.1. Conditional Proof
Firstly, what is conditional proof? Conditional proof is a proof technique in deductive logic [2]. Conditional proof is to make an assumption at any point of a derivation. The assumption must be used to justify a conditional and then be discharged, before the derivation ends. The conditional established with the aid of the assumption will have the assumption as its antecedent, and the previous line of the derivation as its consequent. Figure 1 is the formula of conditional proof
Below is a simple example that illustrates the fundamental idea of conditional proof.

Example 1 [2]
If the burglar did not come through the door then he came either across the roof or up the wall. If he came up the wall then he would have used a ladder. Let us assume, for the sake of argument, that he did not use a ladder. It follows that he did not come up the wall. From that it follows that if he did not come through the door then he came across the roof. So we may conclude that if he did not use a ladder, then if did not come through the door he came across the roof.

With the dictionary
D = The burglar came through the door
R = The burglar came across the roof
W = The burglar came up the wall
L = The burglar used a ladder

The conditional proof will be

1. \( \neg D \rightarrow R \lor W \) promise
2. \( W \rightarrow L \) promise
3. \( \neg L \) assumption
4. \( \neg W \) 2, 3
5. \( \neg D \rightarrow (\neg R \rightarrow W) \) 1
6. \( (\neg D \land \neg R) \rightarrow W \) 5
7. \( \neg (\neg D \land \neg R) \) 4, 6
8. \( \neg D \lor \neg R \) 7
9. \( \neg D \rightarrow R \) 8
10. \( \neg L \rightarrow (\neg D \rightarrow R) \)

As demonstrated in [11], the problem can be easily solved by a human being with making assumptions in the derivation. However, comes to automated deduction, it illustrated that (a) a top-down approach can never get out of the infinite recursion and hence the derivation has to be aborted; (b) without making assumptions, a bottom-up approach can never deduce enough facts to answer the query; and (3) it is also impossible to solve the problem by using a combined top-down and bottom-up approach. Therefore introducing conditional proof is essential on solving the problem. In the paper, the problem was eventually solved by using GOPT-resolution with the trigger technique.

2.2. The Necessity of Incorporating Conditional Proof

Why is it necessary to introduce conditional proof into automated deduction? What is the significance of this incorporation? In this sub-section, we will use an example which was presented in [11] to answer these questions.

Example 2 [11]
There are five houses, each of a different colour and inhabited by men of different nationalities, with different pets, drinks, and cigarettes.

(1) The Englishman lives in the red house.
(2) The Spaniard owns the dog.
(3) Coffee is drunk in the green house.
(4) The Ukrainian drinks tea.
(5) The green house is immediately to the right (your right) of the ivory house.
(6) The Old Gold smoker owns snails.
(7) Kools are smoked in the yellow house.
(8) Milk is drunk in the middle house.
(9) The man who smokes Chesterfields lives in the house next to the man with the fox.
(10) The Norwegian lives in the leftmost (your left) house.
(11) Kools are smoked in the house next to the house where the horse is kept.
(12) The Lucky Strike smoker drinks orange juice.
(13) The Japanese smokes Parliaments.
(14) The Norwegian lives next to the blue house.

Now, who drinks water? And who owns the zebra?


2.3. Conditional Proof and SLD-Resolution

Logic programming has been studied from both syntactic and semantic perspectives [3], that is the proof theory and model theory of logic programming. From syntactic perspective, a logic program is a finite set of clauses and a Horn clause logic program is a finite set of Horn clauses; The goal of the program is a clause and therefore the deduction with the goal on the program is actually a process of finding the derivation of the goal.

A widely adapted derivation technique in this area is SLD-resolution [3]. SLD-resolution starts from the goal and backtracks to find the derivation for the goal or to show that the goal can not be proved, in which case the goal is not a consequence of the program. However SLD-resolution has its limitations and, as pointed previously, there are derivations which can be easily conducted by our human beings, but cannot found by SLD-resolution.

Conditional proof is to introduce an assumption into a derivation to deduce a conditional. The assumption is put as an antecedent to deduce a conditional before the end of the derivation. Conditional proof can be incorporated into both top-down and bottom-up reasoning. In this paper, however, we will mainly concentrate on incorporating conditional proof into top-down approaches. We will use SLD-resolution as an example to examine how it is accommodated into the derivation.

Conditional proof can be introduced into SLD-resolution from two different angels: let P be a logic program and let A be a program clause

1. assume that A is a consequence of P and eventually find the derivation for A;
2. assume that A is a consequence of P and this assumption leads to contradiction, therefore \( \neg A \) is a consequence of P.

The formula of incorporating conditional proof into SLD-resolution is as follows.

(1)

\[
P \quad \text{the logic program - a set of program clauses} \\
A \quad \text{making an assumption} \\
T_1 \\
\vdots \\
T_m \\
A \rightarrow T_m \\
T_m \\
\]

Figure 3 conditional proof in SLD-resolution (1)

(2)

\[
P \quad \text{the logic program - a set of program clauses} \\
A \quad \text{making an assumption} \\
T_1 \\
\vdots \\
T_m \\
A \rightarrow T_m \\
\neg T_m \quad T_m \text{ is contradictory with } P \\
\neg A \\
\]

Figure 4 conditional proof in SLD-resolution (2)
In case (1) the assumption A is a consequence of the program P, so the conditional proof becomes a derivation and anything is derived from it is a consequence of P. In Figure 3, \( T_m \) is a consequence of P.

In case (2) the conditional which is deduced from the program and the assumption is contradictory with P and this leads to the conclusion that A is not a consequence of P and hence \( \neg A \) is a consequence of P.

### 2.4. Case Study

In this section, we will use an example to illustrate the conditional proof in SLD-resolution.

**Example 3**

There are six people \{a, b, c, d, e, f\} in a party and they are three child-parent pairs. Each parent has only one child and each child has only one parent in the party. One cannot be the child of a person, at the mean time, be the parent of another person. It is known that

1. b, c, e, f are female.
2. a is a parent of f.
3. b is a parent of e.

The query is who is the mother of d.

To solve the problem with SLD-resolution in a conventional way, three predicate symbols are defined:

- \( \text{female}(X) \rightarrow X \) is female
- \( \text{parent}(X, Y) \rightarrow X \) is a parent of Y
- \( \text{mother}(X, Y) \rightarrow X \) is the mother of Y

Based on the problem statements, the following logic program P is built

\[
\begin{align*}
\text{female}(b) & \leftarrow \\
\neg \text{female}(c) & \leftarrow \\
\neg \text{female}(e) & \leftarrow \\
\neg \text{female}(f) & \leftarrow \\
\neg \text{parent}(a, f) & \leftarrow \\
\neg \text{parent}(b, e) & \leftarrow \\
\neg \text{mother}(X, Y) & \leftarrow \text{female}(X) \land \text{parent}(X, Y)
\end{align*}
\]

The query is

\( \leftarrow \text{mother}(X, d) \)

The derivation can be presented with a SLD-tree [3] as follows.

![Figure 5 SLD-tree](image)

The derivation is terminated and the answer to the query is that no one is the mother of d. The answer is obviously incorrect.

To incorporate conditional proof into the deduction, the logic program needs to be slightly modified. Although predicate symbols and program clauses remain the same, a set of extra clauses is defined to indicate the contradictory. The clauses are

- \( \text{contradictory} \leftarrow \text{parent}(X, Y) \land \text{parent}(X, Z) \land Y \neq Z \) (a parent cannot have more than one child in the party)
- \( \text{contradictory} \leftarrow \text{parent}(Y, X) \land \text{parent}(Z, X) \land Y \neq Z \) (a child cannot have more than one parent in the party)
- \( \text{contradictory} \leftarrow \text{female}(a) \) (a is not female)
- \( \text{contradictory} \leftarrow \text{female}(d) \) (d is not female)

Therefore the program \( P_1 \) is as follows
The query is, again,
\[ \leftarrow \text{mother}(X,d) \]

Based on problem statements, one and only one of the six people \( a, b, c, d, e, f \) is the mother of \( d \), the following assumptions are made

(1) \( a \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(a,d) \))
(2) \( b \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(b,d) \))
(3) \( c \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(c,d) \))
(4) \( d \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(d,d) \))
(5) \( e \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(e,d) \))
(6) \( f \) is the mother of \( d \) (ie. \( \leftarrow \text{mother}(f,d) \))

We understand that one and only one of the assumptions is true and all other assumptions will lead to contradictory. In other words, only one of the derivations with the assumptions will not lead to contradictory.

The SLD-tree with the goal \( \leftarrow \text{mother}(a,d) \) is

\[
\text{mother}(a,d) \\
\text{female}(a) \land \text{parent}(a,d) \\
\text{contradictory} \leftarrow \text{female}(a)
\]

Again the derivation is terminated with contradiction "b is a parent of d" and "b is a parent of e" which is contradictory against problem statements. Assumption (2) is dropped.

Similarly, the derivations with goals \( \leftarrow \text{mother}(d,d) \), \( \leftarrow \text{mother}(e,d) \) and \( \leftarrow \text{mother}(f,d) \) are all terminated by contradiction and so those assumptions are all dropped. The only derivation which is not terminated by contradiction is the derivation with the goal \( \leftarrow \text{mother}(c,d) \). The derivation is as follows

\[
\text{mother}(c,d) \\
\text{female}(c) \land \text{parent}(c,d) \\
\text{contradictory} \leftarrow \text{parent}(X,Y) \land \text{parent}(Z,X) \land Y \neq Z
\]
As indicated previously, one of the six assumptions must be true. Since Assumptions (1) (2) (4) (5) (6) have all led to contradiction and have been dropped, Assumption (3) c is the mother of d is accepted.

The answer to the query $\neg \text{mother}(X, d)$ is $X = c$.

3. Conclusions

In this paper, we have discussed the integration of conditional proof into automated deduction to enhance the power of deduction strategies. In particular, we have analysed the necessity and feasibility of incorporating conditional proof into SLD-resolution and presented the detailed procedure.

Conditional proof is incorporated into the SLD-derivation from two different angels: let P be a logic program and let A be a program clause

1. assume that A is a consequence of P and eventually find the derivation for A;
2. assume that A is a consequence of P and this assumption leads to contradiction, therefore $\neg A$ is a consequence of P.

This provides the deduction a means to eliminate the assumptions which lead to contradiction and so that the only assumption which does not produce contradiction remains.

References