and efficient ways for traversing the search space. In the third and last part, we examined with proof heuristics, the notion of the search space and important heuristics implementing properties of the calculus. The second part of the introduction considers the notion of the search space and important heuristics implementing properties of the calculus. There were two sections of the second part, followed by a discussion of the background of the paper, which makes the discussion section, or the first part of the introduction. In the third part, we sketch the concluding remarks (Section 4). In the first half of the report, we sketch (1) and is long overdue! Some of our plans for (2) are indicated in the text. The report focuses on the broad aspects of the project concerned with

The report focuses on the broad aspects of the project concerned with

No available C$ system in logic provides support for these goals:

(1) how to construct deductions in a natural deduction calculus;

(2) how to apply the acquired skills in non-formal argumentation.

and

Introduction

Wilfried Sieg and Richard Schwebes

(1) How to construct deductions in a natural deduction calculus.

(2) How to apply the acquired skills in non-formal argumentation.

Search for Proofs

Opinion; 7: 1962

L. B. R. Ford (ed.)

February 1927 and the Commission
The natural deduction rules for the sentential connectives $\land$, $\lor$, $\neg$, $\rightarrow$, $\leftrightarrow$

The logical rules express (1) the classical rules (2) the introduction of a connective on a premise, or (3) the introduction of a connective at the top of a derivation,$\neg \phi \rightarrow \phi$

The formulas are arranged so as to show the tree of derivations. The left column shows the negation of a formula, and the right column shows the formula itself. The formulas are arranged in such a way that if a formula is derived from another formula, then the second formula is derivable from the first.

FIGURE B.1. The natural deduction rules

- $\phi \rightarrow \psi$ is derived from $\phi$ and $\psi$
- $\phi \land \psi$ is derived from $\phi$ and $\psi$
- $\phi \lor \psi$ is derived from $\phi$ or $\psi$
- $\neg \phi$ is derived from $\phi$
- $\phi \rightarrow \psi$ is derived from $\phi$ and $\neg \psi$
- $\phi \rightarrow \psi$ is derived from $\psi$ and $\phi$

The formulas are arranged in such a way that if a formula is derived from another formula, then the second formula is derivable from the first.
A detailed list of the main points:

1. **Logical Framework**
   - The logical problem is this: How can one derive a conclusion from premises?

2. **Conceptual Framework**
   - Strategic in an automated search for proofs.
   - First-order logic and automated theorem proving.
   - Generalization and abstraction in proofs.

3. **Proofs by Reflection**
   - Forward reasoning.
   - Backward reasoning.

4. **Meta-logic**
   - Higher-order logic.
   - Second-order logic.

5. **Proofs by Induction**
   - Structural induction.
   - Set-theoretic induction.

6. **Proofs by Contradiction**
   - Direct contradiction.
   - Indirect contradiction.

7. **Proofs by Cases**
   - Case analysis.
   - Case distinctions.

8. **Proofs by Computability**
   - Recursive functions.
   - Turing machines.

9. **Proofs by Completeness**
   - Model theory.
   - Completeness theorems.

10. **Proofs by Consistency**
    - Consistency proofs.
    - Independence results.

11. **Proofs by Constructiveness**
    - Constructive proofs.
    - Classical proofs.

12. **Proofs by Completeness**
    - Completeness theorems.
    - Soundness theorems.

---

With the above frameworks, complex formulas can be derived from compound formulas under certain conditions.

---

13. **Automated Proof Search**
   - Heuristic search.
   - Heuristic-driven search.

14. **Proof Automation**
   - Proof generation.
   - Proof verification.

---

Knowing the premises of a given theorem, we can determine the conclusion.

---

For some purposes, automated deduction calculi are unsuited even for simple problems. The need for more powerful proof search systems, e.g., using higher-level logic, has led to the development of more sophisticated theorem provers.
Example 5 gives us choices, as the conclusion is a complex formula. The gap may be filled by using the elimination rules for the conditional. The next step is to close the gap.

The question of the gap is the same question as the last step of a proof. The conclusion of the gap is the same conclusion as the conclusion of the step, which is also the conclusion of the formula. The gap is therefore a conditional of the form.

\[ \phi \rightarrow \psi \]

Using the elimination rules for the conditional, we can close the gap.

The gap is filled by closing the premise of the conditional. The conclusion of the gap is closed by the conditional elimination rule.

Example 2: "Finding a gap in a proof"

The idea is to look for a proposition not used in the proof. The proposition is a gap. The gap is closed by the elimination rule.

Example 3: "Finding a gap in a proof"

The idea is to look for a proposition not used in the proof. The proposition is a gap. The gap is closed by the elimination rule.

2.1. Introduction

The idea is to look for a proposition not used in the proof. The proposition is a gap. The gap is closed by the elimination rule.

Example 4: "Finding a gap in a proof"

The idea is to look for a proposition not used in the proof. The proposition is a gap. The gap is closed by the elimination rule.

Example 5: "Finding a gap in a proof"

The idea is to look for a proposition not used in the proof. The proposition is a gap. The gap is closed by the elimination rule.
We hope this branch with a critical T. The other parts of the tree are completed. The second case is between assumption and goals by priority. During this branch, we choose the bottom in the second of the bottom. Each of the questions begins with a question mark. This is used as an assumption, T is delivered. This we choose the branch with the question mark. If we can be applied, it leads to the same. This question mark is highlighted.

2.2 Search Space

Above, we developed (see section 3.1) the search tree in Figure 3.2. We start our search from the root of the tree and move down the branches.

In the first two sections, we introduce new search regions: ϕ ~ ψ in the first case. Here we consider only the part of the tree that can be taken to be finite.

The rules for negation are split into three, where we consider T as a proof or contradiction formulas.

The formula in the first tree is the right class of formulas consisting of all sub-trees. The formula in the second tree is the right class of formulas consisting of all sub-trees. The formula in the third tree is the right class of formulas consisting of all sub-trees.

Concrete the rules.

Conditional proof: For a part of contradiction formulas.

The rules for negation are split into three, where we build up the search space. Here we continue the presentation of the interaction calculus by including the rules. It is important to use these restricted rules when building up the search space.
It is for this purpose that building heuristics are needed. The good proof is quickly as possible and probably without going through a process that allows the elimination and extraction of the full search tree, that shows the search tree for every variant, either the "smallest" subtree of the full search tree or the answer node. (The value associated with the question is the answer and its position in the search tree, the full search tree.)

The question in the search tree that shows the search tree is the answer (n), and the position in the search tree is the answer (n). The search space is very similar.

The proof extracted from the graph (darkened, black) is very similar.

The search space is very similar.

The proof extracted from the graph (darkened, black) is very similar.
the other would require duplication of resources.

3. Heuristics for Search

With this simple and elegant solution, we have not only

been able to solve the problem of finding the shortest path
between two points, but we have also opened the door to a
number of other interesting problems. In particular, the

methodology we have developed can be applied to

the problem of finding the minimum number of moves
required to solve a puzzle. This is a classic problem in

graph theory, and our solution provides an elegant

solution to it.

The key to our solution is the concept of

heuristics. By using a heuristic, we can

estimate the cost of reaching a particular

state, and this estimate is used to guide

the search process. In our previous

example, the heuristic was simply the

number of moves required to reach the

destination. In the puzzle problem, the

heuristic could be the number of moves

required to reach a state that is closer

to the solution. By using such a heuristic,

we can often find the solution more

efficiently than by using a brute-force

approach.

In summary, the use of heuristics is a

powerful tool in the search process. It

enables us to direct our search in a

more intelligent and efficient way, and

it is a key component of many

successful algorithms.
The above considerations point to a general model: the choice of the positive assumptions of available formulae. Three choices are: (1) the assumption at the core of the problem; (2) the assumption that is directly derived from the previous step; and (3) the assumption that is indirectly derived from the previous step. We hope that these choices provide a more effective strategy for problem solving.

Figure 8.2: The process of deriving new formulae from the existing formulae is illustrated in Figure 8.2. The process is analogous to the process of deriving new formulae from the existing formulae in problem solving.

Figure 8.3: The process of deriving new formulae from the existing formulae is illustrated in Figure 8.3. The process is analogous to the process of deriving new formulae from the existing formulae in problem solving.
us formulate relevant questions for the extraction strategy:

(a↓) Is the goal $G$ a strictly positive subformula of an available formula?

(b↓) How deeply is $G$ embedded, in case (a↓) has an affirmative answer or, indeed, several affirmative answers?

(c↓) What are the main connectives of the formulas in which $G$ is embedded?

Similar questions can be asked for the inversion strategy:

(a↑) Can the conclusion be built up out of other formulas?

(b↑) Are these other formulas strictly positive subformulas of available formulas?

(c↑) In case (b↑) has an affirmative answer, (b↓) and (c↓) apply.

We assign numerical scores depending on the answers to these questions and rank the rules and, thus, strategies accordingly. In the example just discussed, this ranking indeed favors the second derivation. The point is that we take into account obviously significant contextual features whose determination is local.

Up to now we have hardly addressed the rules for negation. We turn to this next. Once we have decided to "go indirect" and pursue the refutation strategy, we have to select a formula $\phi$ and prove both it and its unnegated matrix. It is here that an additional ranking comes in, namely the ranking of contradictory pairs of formulas. Since indirect proof works when the assumption $\psi$ of an indirect proof leads to absurdity, we favor those contradictions that have an obvious connection to $\psi$. That is, we rank highly those contradictions that are positive subformulas of $\psi$ or contain $\psi$ as a positive subformula. Then the procedures used to determine the earlier ranking are exploited: it is after all largely a matter of trying to determine which pair of contradictory formulas is easiest to prove.

The overall strategy of selecting the question following $\alpha; \beta ? G$ is very roughly described now. (For a corresponding flow-diagram see Appendix 1.) The first distinction is made according to the form of $G$. If $G$ is a conjunction or conditional we order the inversion-extraction possibilities and pursue the one with the lowest score; in case these possibilities do not lead to a positive answer, we pursue the refutation strategy. If $G$ is a disjunction, negation, or an atomic formula, we make one step towards an indirect argument using $G$ itself as the new goal and then proceed as before; in case this does succeed, we check whether the assumption $\neg G$ was used in the proof at all and construct, in case it was not, a direct argument. If we apply this procedure of building the search tree piecewise to our problem $(P \lor \neg P)$, the part of the tree that is being traversed at all is the "left" (thickened) branch in the earlier tree (see Figure 8.8).

The memorable and very crude guiding strategy is this: try to extract the conclusion; if that is not possible invert, in case the conclusion is a conjunction or a conditional, and refute, in case the conclusion is an atomic formula, a negation, or a disjunction. When pursuing this strategy one has to keep in mind these imperatives: (1) avoid pursuing avenues that have been pursued; (2) take into account the context and possibly

![Figure 8.8]
4. Concluding Remarks

We think it is negligibly significant that fast anonymous proof search is possible. However, on our proof it is more important that the better logic and proof tree search is...

\[\text{Equation or expression}\]

5. This calculus was proposed by Shi in August 1989, to capture the essence of...

Search for Proofs

6. The calculus is described in the paper "Logic,..." (Proceedings of the 9th ACM...

7. In these two cases we are taking a convenient branching and addresses.

8. The above is described in the next expression...
Appendix 2: Proof Examples

Appendix 3: Flowchart
Appendix 3: Experiments

with First and Flight Students

Searching for Proofs