REASONING BY CASES AND THE FORMATION OF CONDITIONAL PROGRAMS[1]

Douglas R. Smith
Kestrel Institute
1801 Page Mill Road
Palo Alto, CA 94304 USA

ABSTRACT

Reasoning by cases, a natural feature of human reasoning, has been difficult to formulate so that it can be performed naturally when needed. Several difficulties arise: (1) how to motivate the use of reasoning by cases when and only when needed, (2) how to determine an appropriate analysis of the goal into cases, and (3) how to carry out the deduction in each case and combine the results. In this paper we focus on how reasoning by cases can be naturally accomplished in the framework of derived antecedents (Smith 1982). Our main technical contributions are (1) a set of strategies that draw on the context of a deduction to provide an appropriate case analysis for a goal, and (2) inference rules for carrying out reasoning by cases and forming conditional terms for the existentially quantified variables in the initial goal.

I Introduction

Reasoning by cases, a natural feature of human reasoning, has been difficult to formulate so that it can be performed naturally when needed. Several difficulties arise: (1) how to motivate the use of reasoning by cases when and only when needed, (2) how to determine an appropriate analysis of the goal into cases, and (3) how to carry out the deduction in each case and combine the results. On motivating the use of reasoning by cases, we merely note that this type of reasoning is typically used only as a last resort since it multiplies the number of goals that need to be considered. Our main technical contributions in this paper are (1) a set of strategies that draw on the context of a deduction to provide an appropriate case analysis for a goal, and (2) inference rules for carrying out reasoning by cases and forming conditional terms for the existentially quantified variables in the initial goal.

We restrict our attention to goal formulas of the form

$$\forall x_1, \ldots, x_n \exists z [I(x_1, \ldots, x_n) \Longrightarrow O(x_1, \ldots, x_n, z)] \quad (1)$$

although the techniques presented are not necessarily limited to this type of goal. What is desired as an output of the deductive process is a way to construct an algorithm that computes a value for z as a function of the variables $x_1, \ldots, x_n$. We sometimes call (1) a problem specification and refer to $x_1, \ldots, x_n$ as input variables and z as the output variable. When a reasoning by cases argument is needed to establish (1) then the corresponding algorithm has the form of a conditional, where each case corresponds to a branch of the conditional. That is, if (1) can be established by separately considering the cases $C_1, \ldots, C_m$ then we want to be able to construct a conditional expression

$$\begin{array}{l} \textbf{if} \\ \quad P_1(x_1, \ldots, x_n) \rightarrow f_1(x_1, \ldots, x_n) \\ \quad \cdots \\ \quad P_m(x_1, \ldots, x_n) \rightarrow f_m(x_1, \ldots, x_n) \\ \textbf{fi} \end{array}$$

where if ... fl is a functional version of Dijkstra's nondeterministic conditional. The function f_i computes a value for the output variable z corresponding to case C_i. Predicate P_i, called a *guard,* characterizes the conditions under which f_i provides a solution to the initial problem specification.

Evidently, what is needed in order to form a conditional expression from a reasoning by cases argument is an analysis of the goal into cases $C_1, \ldots, C_m$ and extraction of a guard P_i and function f_i from the proof of case C_i for $i = 1, \ldots, m$. There are many ways to analyze a goal into cases. Such analysis is easy when the cases are explicitly in the initial goal. We present below several strategies for handling the more difficult situation in which the analysis depends on the context of a goal.

In the next section we review the notion of a system for deriving antecedents and present a few inference rules pertinent to reasoning by cases. In the succeeding section we analyze the notion of reasoning by cases into two general types and present several instances of these types, called strategies for reasoning by cases. The application of these strategies to a number of program synthesis tasks may be found in (Smith 1985a).

[1]This research was supported in part by the Office of Naval Research under Contract N00014-84-C-0473 and in part by the Defense Advanced Research Projects Agency Contract N00014-81-C-0582, monitered by the Office of Naval Research. The views and conclusions contained in this paper are those of the author and should not be interpreted as representing the official policies, either expressed or implied of DARPA, ONR, or the US Government.

II Derived Antecedents

Reasoning by cases can be naturally accomplished in the framework of derived antecedents (Smith 1982,Smith 1985b). This framework allows a deduction whose aim is not limited to establishing the validity of a goal formula *G,* but more generally seeks to derive a formula *A,* called a *derived antecedent,* satisfying certain constraints and such that $A \Longrightarrow G$ is valid. The constraint we are concerned with simply checks whether the free variables of a formula are a subset of some fixed set that depends on *G.* If *G* happens to be a valid formula in our current theory then the antecedent *true* should be derived - thus ordinary theorem-proving is a special case of deriving antecedents.

A *solution* to a problem specification (1) is a tuple *(A, a)* where a is a term called the *answer,* and A, called the *derived antecedent,* is a formula whose free variables are a subset of $\{x_1,\ldots,x_n\}$ and such that, if we define the algorithm *F* by $F\{x_1 \ldots,x_n) = a$, then the formula

$$\forall x_1,\ldots,x_n[A(x_1,\ldots,x_n) \wedge I(x_1,\ldots,x_n) \Longrightarrow O(x_1,\ldots,x_n,F(x_1,\ldots,x_n))]$$

is valid. In other words a solution provides a term that computes a value for the existentially quantified variable as a function of the input variables $x_1,\ldots,x_n$, but only under the additional assumption of the derived antecedent. The derived antecedent *A* is called the *derived input condition* of algorithm *F.*

In a natural deduction-like system the derivation of a solution for a goal *G* can be described by a two stage process. In the first stage reduction rules are repeatedly applied to goals reducing them to subgoals. A primitive rule is applied whenever possible. The result of this reduction process can be envisioned as a goal tree in which 1) nodes represent goals /subgoals, 2) arcs represent reduction rule applications, and 3) leaf nodes represent goals to which a primitive rule has been applied. The second stage involves the bottom-up composition of solutions. Initially each application of a primitive rule to a subgoal yields a solution. Subsequently whenever a solution has been found for each subgoal of a goal *G* then a solution is composed for *G* according to the reduction rule employed. In a working system we have developed, called RAINBOW, a single solution is selected from among the alternate derived solutions of a goal by maximizing over a heuristic measure of weakness and structural simplicity.

Solutions obtained by solving subgoals are composed to obtain a solution for the parent goal. The reduction rules presented below require two composition methods: the disjunctive and conjunctive composition of solutions. The *disjunctive composition* of solutions $\langle A_1,\alpha_1\rangle$ and $\langle A_2,\alpha_2\rangle$ is

$$\langle A_1 \vee A_2, \text{if } A_1 \rightarrow \alpha_1 \parallel A_2 \rightarrow \alpha_2 \text{ fi}\rangle.$$

Intuitively, the disjunctive composition of solutions returns an answer which behaves like α_1 when *Ai* holds and behaves like α_2 when A_2 holds. If A_1 or A_2 is *false* then the conditional collapses to just α_2 or α_1 respectively. For example, the disjunctive composition of solutions $\langle a \leq b, a\rangle$ and $\langle b \leq a, b\rangle$ is

$$\langle a \leq b \vee b \leq a, \text{if } a \leq b \rightarrow a \parallel b \leq a \rightarrow b \text{ fi}\rangle.$$

The *conjunctive composition of solutions* $\langle A_1,\alpha_1\rangle$ and $\langle A_2,\alpha_2\rangle$ is

$$\langle A_1 \wedge A_2, uc(\{\alpha_1/z\},\{\alpha_2/z\})\rangle$$

where *uc* computes the unifying composition of substitutions (Nilsson 1980). We do not elaborate on conjunctive composition since it does not play a central role in this paper.

Natural deduct ion-like systems for deriving solutions may be found in (Smith 1982,Smith 1985b). Here we merely list the inference rules necessary to support reasoning by cases. Goals are prepared by treating the universally quantified variables as constants and then dropping all quantifiers. We use the notation $\frac{G}{H}$ where $\mathbf{H} = \{h_1,h_2,\ldots,h_k\}$ as an abbreviation of the formula $h_1 \wedge h_2 \wedge \ldots \wedge h_k \Longrightarrow G$.

Rule R2. Reduction of a disjunctive goal. If the goal formula has the form $\frac{F \vee G}{H}$, then generate subgoa $\frac{F}{H}$ and ^. If these subgoals return solutions $\langle P,\alpha_1\rangle$ and $\langle Q,\alpha_2\rangle$ respectively, then return the disjunctive composition

$$\langle P \vee Q, \text{ if } P \rightarrow \alpha_1 \parallel Q \rightarrow \alpha_2 \text{ fi}\rangle$$

as a solution to goal $\frac{F \vee G}{H}$. Naturally if subgoal $\frac{F}{H}$ is explored first and a solution $\langle true,\alpha_1\rangle$ is returned, then *F* has been proved and there is no need to explc $\frac{G}{H}$. The solution $\langle true,\alpha_1\rangle$ can be returned for $\frac{F \vee G}{H}$.

Rule R2 assumes that the goal has only two disjuncts. The generalization to multiple disjuncts in this and later rules is straightforward.

Something like the following rule is commonly called the "case analysis" rule in many theorem-proving systems (see for example (Bledsoe 1977)).

Rule R3. Reduction by a disjunctive hypothesis. If $P \vee Q$ is an axiom or hypothesis then reduce goal $\frac{G}{H}$ to subgoals $\frac{G}{H_P}$ and $\frac{G}{H_Q}$ where *HR* is a set of hypotheses obtained by limited forward inference on *R* and the set *H*[2]. If solutions $\langle A_1,\alpha_1\rangle$ and $\langle A_2,\alpha_2\rangle$ are obtained for these subgoals respectively, then return their conjunctive composition as the solution to $\frac{G}{H}$.

While this rule is sound, it often fails to find a solution when one exists. The solutions to the various cases typically have

[2]I.e. *HR* contain! *R* and all hypotheses in *H* plus any interesting consequences of *H* and *R* deriveable within some bounded amount of computational resource.

incompatible answers. The unifying composition operator simply reports that such answers are inconsistent. A better rule is the following.

Rule R4. Reduction by disjunctive hypothesis. If there is axiom or hypothesis $P \vee Q$ then reduce goal GH to subgoals $\overset{G}{H_P}$ and $\overset{G}{H_Q}$. If solutions $\langle A_1, \alpha_1 \rangle$ and $\langle A_2, \alpha_2 \rangle$ are obtained for these subgoals respectively, then return their composition

$$\langle (A_1 \wedge P) \vee (A_2 \wedge Q), \text{ if } A_1 \wedge P \rightarrow \alpha_1 \parallel A_2 \wedge Q \rightarrow \alpha_2 \text{ fi} \rangle$$

as a solution to the goal $\overset{G}{H}$.

III Strategies for Reasoning By Cases

The idea behind reasoning by cases is to treat a goal in a number of alternate contexts, or cases, that are collectively exhaustive. Often the difficulty in solving a goal is due to the fact that each case requires a different argument.

There are two identifiable types of reasoning by cases. In the first type, the goal *G* is reduced to a conjunction of weaker goals. This occurs for instance when we assume a disjunction $P \vee \neg P$. The goal becomes $P \vee \neg P \Longrightarrow G$ or equivalently $(P \Longrightarrow G) \wedge (\neg P \Longrightarrow G)$. Rules R3 and R4 can then be used to handle the cases. The second type of reasoning by cases involves reducing the goal *G* to a disjunction of stronger goals. This occurs for instance when we conjoin a disjunction $P \vee \neg P$ to the goal. The goal becomes $G \wedge (P \vee \neg P)$ or equivalently $(G \wedge P) \vee (G \wedge \neg P)$. Rule R2 can then be used to handle the cases.

Sometimes the motivation for performing a case analysis is intrinsic to the goal. Rules R2 and R4 can be used to carry out the reasoning by cases. Sometimes however, motivation for case analysis must come from context. To exploit context in setting up a case analysis we use special procedures called *strategies* that have the effect of an inference rule but may involve arbitrary amounts of processing. Strategies are intended to encode in one deductive step a commonly occurring reasoning pattern. We present several strategies for reasoning by cases below. They are described in terms of (i) a heuristic for determining when to apply reasoning by cases and with what cases, and (ii) a description of how cases are generated and how the solutions for these cases are composed. Omitted from the description of these strategies is the common heuristic that they be applied only as a last resort; that is, only if other means are not applicable or have failed.

Strategy RBC1. Exploiting an unverified subgoal.

Apply-when: A subgoal *P* of goal *G* cannot be easily verified and the variables of *P* are a subset of the input variables.

Procedure: Use rule R4 with assumption $P \vee \neg P$.

The procedure part of RBC1 should be viewed as descriptive rather than prescriptive. We do not propose that RBC1 form the disjunction $P \vee \neg P$ only to have rule R4 break it down. Rather, RBCI should directly create subgoals $\overset{G}{H_P}$ and $\overset{G}{H_{\neg P}}$ and should compose the solutions to these subgoals as R4 would. Similar remarks hold for the following strategies.

A typical use of strategy RBCI occurs when term r in a conditional rewrite rule

$$r \rightarrow s \quad \text{if } C$$

matches term *t* in goal *G(t)* with unifier *6* but the condition $C\theta$ cannot be verified. Strategy RBCI would apply rule R4 with $C\theta \vee \neg C\theta$. Concrete examples of the use of a strategy like RBCI may be found in (Buchanan and Luckham 1974).

Strategy RBC2. Alternate ways to apply a conditional rewrite rule.

Apply-when: The goal has the form *G(t)* and there is a conditional rewrite rule

$$r \rightarrow s \quad \text{if } C$$

such that r matches *t* with several distinct unifiers. Suppose that there are only two such unifiers θ_1 and θ_2 and that $C\theta_1$ and $C\theta_2$ are both verifiable.

Procedure: Use rule R2 after reducing $G(t)$ to $G(s)\theta_1 \vee G(s)\theta_2$.

Strategy RBC3. Alternate forms of an input variable.

Apply-when: A set of terms can be constructed that describe the possible forms of an input. Suppose for example that an input *x* of type *D* either has the form f_1 or $f_2(s_1(x), s_2(x))$ where f_1 and f_2 are generators of type *D* and s_1 and s_2 are selectors for f_2

Procedure: Use rule R4 with assumption $x = f_1 \vee x = f_2(s_1(x), s_2(x))$.

To support RBC3 each data type known to the system should have one or more inductive definitions in terms of a set of generators. These provide standard alternative structures for the values of the type. Since they are based on inductive definitions of the type, they prepare the way for an inductive proof of the initial goal.

Sometimes there is an input condition that restricts the set of values that a variable may vary over. A technique for creating a set of alternative forms taking into account an input condition is presented in (Smith 1985a).

Strategy RBC4. Alternate forms of a term for the output variable.

Apply-when: A set of terms can be constructed that describe the possible forms of the output. Suppose for example that the output variable has type *R* and either has the form f_1 or $f_2(u, v)$.

Procedure: Use rule R2 after reducing goal $G(z)$ to $G(f_1) \vee G(f_2(u, v))$ where u and *v* are fresh variables.

The previous strategies break a goal into cases that are determined before the cases are explored. They may be called *a* prion strategies. Our last strategy creates subgoals based on the solutions to previous subgoals and so may be called an *a posteriori* strategy.

Strategy RBC5. Handling an unsolved case.

Apply-when: Solution $\langle A_1, \alpha_1 \rangle$ is produced for goal $\overset{G}{H}$ but A_1 is insufficiently weak or simple.

Procedure: Generate subgoal $\overset{G}{H_{\neg A_1}}$; if solution $\langle A_2, \alpha_2 \rangle$ is produced then generate the solution

$$\langle A_1 \vee A_2, \text{ if } A_1 \rightarrow \alpha_1 \, [] \, \neg A_1 \wedge A_2 \rightarrow \alpha_2 \text{ fi} \rangle$$

for $\overset{G}{H}$.

IV Concluding Remarks

A limited form of antecedent derivation is used by Bledsoe and Tyson (Bledsoe and Tyson 1977) to perform reasoning by cases on program variables during program verification. The difficulty in getting rules like R3 to produce a satisfactory solution has stimulated a number of potential extensions, including the use of "conditional substitutions" (Nilsson 1980) and "generalized substitutions" (Tyson and Bledsoe 1979).

This paper is based on experience with the CYPRESS program synthesis system (Smith 1985b). This system depended on an antecedent deriver for all cf its deduction and used strategies related to RBC3 and RBC4 for creating various divide-and-conquer algorithms, including the selection sort algorithm in Example 4. CYPRESS used a strategy related to RBC2 for creating composition and decomposition operators on various composite data types, including the decomposition operator derived in Example 3. We are currently constructing a new version of this system at Kestrel Institute. This new system not only includes special strategies for constructing various classes of algorithms but can fall back on general purpose deductive program synthesis when these special methods fail.

One goal of this paper has been to explore the notion of reasoning by cases and to formulate the logical mechanisms needed to implement this type of reasoning in a natural way. The framework of derived antecedents supplements the ease with which a natural deduction-like system can motivate and introduce case analysis with essentially logical mechanisms for handling case structure and forming conditional terms for existentially quantified variables. Although we have presented strategies for reasoning by cases in the context of a natural deduction-like system, we believe these strategies can also usefully augment a resolution-based system.

ACKNOWLEDGEMENTS

I would like to thank Allen Goldberg for his valuable comments on a previous draft.

REFERENCES

[1] Bledsoe, W.W., "Nonresolution Theorem Proving". *Artificial Intelligence* 9:1 (1977) 1-35.

[2] Bledsoe, W.W. and Tyson, M., "Typing and Proof by Cases in Program Verification". *Machine Intelligence 8: Machine Representations of Knowledge.* E.W.Elcock and D. Michie, Eds., Ellis Horwood Ltd., Chichester, England, 1977, 30-51.

[3] Buchanan, J.R. and Luckham, D.C., "On Automating the Construction of Programs." Technical Report STAN-CS-74-433, Stanford University, 1974.

[4] Nilsson, N.J., *Principles of Artificial Intelligence.* Tioga Press, Palo Alto, CA, 1980.

[5] Smith, D.R., "Derived Preconditions and Their Use in Program Synthesis." in *Proc. Sixth Conference on Automated Deduction,* Ed. D.W. Love land, Lecture Notes in Computer Science 138, Springer-Verlag, New York, 1982, 172-193.

[6] Smith, D.R., "Reasoning by cases and the formation of conditional programs," Technical Report Kes.U.-85.4, Kestrel Institute, Palo Alto, CA, 1985.

[7] Smith, D.R., "Top-down synthesis of divide and conquer algorithms." to appear in Artificial Intelligence.

[8] Tyson, M. and Bledsoe, W.W., "Conflicting Bindings and Generalized Substitutions" In *Proc. Fourth Workshop on Automated Deduction.* Austin, Texas, 1979, pp. 14-18.