Spreadsheets with Incremental Queries as a User Interface for Logic Programming

M. H. van EMDEN
Department of Computer Science,
University of Waterloo,
Waterloo, Ontario, Canada.
Masaru OHKI and Akikazu TAKEUCHI
ICOT Research Center,
Institute for New Generation Computer Technology,
Mita, Minato-ku, Tokyo 108, Japan.

Received 18 October 1985.
Revised manuscript received 20 May 1986.

Abstract We believe that currently marketed programs leave unexploited much of the potential of the spreadsheet interface. The purpose of our work is to obtain suggestions for wider application of this interface by showing how to obtain its main features as a subset of logic programming. Our work is based on two observations. The first is that spreadsheets would already be a useful enhancement to interactive languages such as APL and Basic. Although Prolog is also an interactive language, this interface cannot be used in the same direct way. Hence our second observation: the usual query mechanism of Prolog does not provide the kind of interaction this application requires. But it can be provided by the Incremental Query, a new query mechanism for Prolog. The two observations together yield the spreadsheet as a display of the state of the substitution of an incremental query in Prolog. Recalculation of dependent cells is achieved by automatic modification of the query in response to a new increment that would make it unsolvable without the modification.

Keywords: Spreadsheet, User Interface, Logic Programming Incremental Query, Interactive Problem Solving

§1 Introduction

We belong to a growing group of scientists working toward a situation where computers are universally used as tools to empower human minds. Not just the minds of programmers or clever non-programmers, but of any non-
retarded, toilet-trained human. Of course, natural language will play a role in achieving this situation, but we hardly have an idea of the work that needs to be done. Given our ignorance in this area, is our goal perhaps over-ambitious?

We believe not. It is clear that natural language, though indeed natural, is not the preferred vehicle of thought in all areas. Centuries ago, problems, even numerical ones, were exclusively stated and solved in natural language. Renaissance mathematicians discovered that algebra is a superior language for a large class of such problems. About one century ago Frege found that problems of non-numerical reasoning can often be solved easier in the symbolic calculus he invented.

Even older, perhaps, is the saying *A picture is worth a thousand words.* Note that these are words of natural language. Recently, the iconic user interface of the Apple Macintosh computer has proved its effectiveness not only with the recently toilet-trained, but also with their parents and grandparents. Electronic spreadsheets, perhaps not quite iconic, but certainly not natural language, have taken the world of business by storm. Together with word-processing programs, they have proved that universal accessibility of computers has been achieved already.

But of course this desirable degree of accessibility exists only in a very narrow area. We believe that logic programming has the potential of dramatically enlarging that area. A problem that remains to be addressed is to construct a universally accessible user interface for logic programs. Approaches via natural language are being pursued, and these are promising. In this paper, however, we are concerned with the unexploited potential of the spreadsheet interface.

We explain first (Section 2) in what sense spreadsheets are more widely applicable than in VISICALC, MULTIPLAN, LOTUS 1-2-3, SUPERCALC, etc. (we will refer to such programs as TYPICALC). We find that this wider applicability includes Prolog, though not via the queries typically used with Prolog. We use incremental queries, a modified type of query proposed in Ref. 1). They are Prolog queries which are incrementally entered by the user. These are briefly reviewed in Section 3, which also includes a front end for processing incremental queries, something that was missing from Ref. 1).

When the goals of an incremental query are restricted to equations, we almost have TYPICALC as a subset of Prolog. What remains to be added is the facility for changing cells and recalculating cell contents dependent on the ones changed. We discuss a prototype Prolog program for this in Section 4. It depends on automatically modifying sets of equations that have become unsolvable. In Section 5 we argue that a more general facility allowing the user to cancel arbitrary constraints is a powerful tool in interactive problem solving. A Prolog prototype for this facility is discussed as well.