sumptions in a transition, such as the failure of a variable to maintain a specific value, can be modelled by letting an expression e represent the potential new value of a variable x, then computing the predicate difference $d(T \Rightarrow S)^x_e$. The predicate difference gives the conditions under which the invariant will change truth value, that is, the conditions under which the system would not be secure.

8.3. Formalizing the database inference problem

An important topic in computer security and privacy is database inference, which concerns the question of how database systems can prevent the inference of confidential information from responses to queries. Although the system may never explicitly release the confidential fact, a user may be able to infer the secret by putting together other facts that are not protected. This problem can be modelled by letting S represent the secret information and F represent the facts that the database has revealed so far. If a new query is given, changing F to F', and $F \Rightarrow S$, then the secret has effectively been disclosed and security has failed.

The database inference problem can be modelled using predicate differences. Assume that the system has not revealed the secret S, i.e. $\neg(F \Rightarrow S)$ is true. If x is a variable in the predicate F that defines the current set of facts, and a query will give an expression e as the new value of x, then the system is secure if and only if the expression $\neg(F \Rightarrow S)$ is independent of the substitution of e for x. Let I be the invariant $F \& \neg S$, which is equivalent to $\neg(F \Rightarrow S)$. Then if $dI/dF^x_e = 0$, the substitution of e for x will not affect the invariant: i.e. secrecy will not be compromised.

9. CONCLUSIONS AND FUTURE DIRECTIONS

Predicate differences can be an effective analytical tool for evaluating the effect of changes to formal specifications. They may also be useful in re-verifying specifications after modification; determining if a change will cause a previously secure system to become non-secure; and as a metric for changes to predicates.

Examples presented in this paper were based on real applications, but additional experience is needed to explore the technique. Integrating the calculation of predicate differences into a tool for a formal specification language would make it possible to compute 'predicate slices' of the formal specification. Tools to compute predicate slices for popular specification languages such as Z and InaJo could be useful in evolving and maintaining system specifications.

Acknowledgements

I am grateful to John Cherniavsky, Jim Lyle, Bill Majurski, and Charles Nicholas for many helpful comments on this work.

REFERENCES

1. S. B. Akers, On a theory of Boolean functions, *SIAM Journal* 7 (4).
2. N. Bell, E. W. Page and M. G. Thomason, Extension of the Boolean difference concept to multi-valued logic systems. *Proceedings of the 1972 Symposium on the Theory and Applications of Multiple-Valued Logic Design.*
3. E. W. Dijkstra, *A Discipline of Programming.* Prentice Hall, Englewood Cliffs, NJ (1976).
4. S. Gibbs, D. Tsichritzis and E. Casais, Class management for software communities. *CACM* **33** (9) (1990).
5. D. Gries, *The Science of Programming.* Springer Verlag, New York (1987).
6. D. R. Kuhn, Predicate differences and the analysis of dependencies in formal specifications. *14th National Computer Security Conference.* Washington, D.C. (October 1991).
7. H. Lu and S. C. Lee, Fault detection in M-logic circuits using the M-difference. *Proceedings of the International Symposium on the Multiple Valued Logic, 1984.*
8. P. N. Marinos, Derivation of minimal complete sets of test-input sequences using Boolean differences. *IEEE Transactions on Computers* **C-20** (1) (1971).
9. D. E. Muller, Application of Boolean algebra to switching circuit design and error detection. *Transactions of the Institute of Radio Engineers* **EC-3** (1954).
10. I. S. Reed, A class of multiple-error correcting codes and the decoding scheme. *Transactions of the Institute of Radio Engineers* **IT-4** (1954).
11. I. S. Reed, Boolean difference calculus and fault finding. *SIAM Journal of Applied Mathematics* **24** (1) (1973).
12. R. P. Trueblood and A. Sengupta, Dynamic analysis of the effects access rule modifications have upon security. *IEEE Transactions on Software Engineering* **SE-12** (8) (1986).
13. M. Weiser, Program slicing. *IEEE Transactions on Software Engineering* **SE-10** (7) (1984).
14. M. Whitney and J. Muzio, Decisive differences and partial differences for stuck-at fault detection in MVL circuits. *18th Intl. Symposium on Multiple-Valued Logic, Palma de Mallorca, Spain,* 24–26 May (1988).

Notice

The book "ML for the Working Programmer", written by L C Paulson and published by Cambridge University Press, was reviewed in this Journal in October 1992, page 450. The publishers have asked us to announce that this book is now available in paper back (ISBN 0-521-42225-6, price £14.95).