I want to thank Alan Bundy and his group for hosting this conference, and the Herbrand award committee and Maria Paola for selecting me for this award. This award means a lot to me. Thanks to the Herbrand award committee for their work, and thanks also to those who supported me in this award.

Perhaps some historical reflections from early in my career would be appropriate. At that time, resolution had been developed about five years before, and the Knuth-Bendix procedure was new. Termination proofs for rewriting systems were in their infancy. Lipton and Snyder (On the Halting of Tree Replacement Systems, 1977) had an ordering that could prove termination of the distributive law, but not in combination with other common equations. Renato Iturriaga in his 1967 Ph.D. thesis at Carnegie-Mellon had a technique involving arithmetic and repeated exponentiation that could handle rules similar to those dealt with by the recursive path ordering, but only a small number of them. Each ordering had to be proved terminating by special methods.

I think it was Dave Luckham at Stanford who expressed the view that there might be better methods for dealing with equational systems. I started looking into this area, and was making progress towards something like unfailing completion, which did not come on the scene until much later. However, a departmental report by Dallas Lankford of Louisiana Tech arrived at about that time, and he had apparently already developed an unfailing completion method and proved it complete. He said the completeness proof would appear in the next technical report. It never appeared, and he never completed the proof, but this report discouraged me from continuing in this line of research. I should have contacted him to find out more information, and collaborate with him, but did not have the maturity to know how to handle the situation properly. This is the characteristic of youth, full of ambition and energy and brilliant of intellect, but lacking in maturity. And if we of greater maturity had to do it over again, we might not do things as well as we did the first time.

I got a job at the University of Illinois and became interested in termination of rewriting. I developed something called the path of subterms ordering, and published it in an Illinois departmental report. It was similar to the recursive path ordering that Dershowitz developed soon afterwards, but was more complex. It involved multisets and partial orderings on symbols, as did most of the later orderings. Nachum saw what I was doing and it motivated him to develop the well-known recursive path ordering. He also developed a general technique for proving termination of term-rewriting systems. I even

remember him trying to find this termination method, and he was asking me questions that I could not answer. Dave Liu at Illinois was able to answer some of his questions. It was a brilliant insight of Nachum that such a general termination technique might exist. He was basically trying to prove Kruskal's tree theorem. Later he found that the termination method he was seeking was already known, and he was able to apply this to show that all simplification orderings are well-founded.

I submitted my departmental reports to the J.ACM. Unfortunately, I did not reference people who were in the proper community, and this led to problems in the refereeing. My department chair, Jim Snyder, implied that one of the referees of my reports committed suicide. This delayed the refereeing process, and by the time my papers were read, Nachum's recursive path ordering had already been published, and my submissions were rejected. When I explained the situation to the editors of J.ACM, they apparently felt bad and offered to have me write a report for the J.ACM on the uses of term-rewriting systems. However, as another illustration of youthful immaturity, I did not do it. I should have asked Nachum to co-author it with me. At the time I was concentrating on NP-completeness, algorithms, and complexity research.

Claude Kirchner saw my Illinois departmental reports on the path of subterms ordering. He did not look at the proofs but just looked at the examples. This is what led him to become interested in the area of term rewriting systems. Pierre Lescanne was also motivated by this work to develop his recursive decomposition ordering. Since then Pierre has done a tremendous amount of high-quality work in various formal areas.

At the time Sam Kamin was at Illinois and Kamin and Levy wrote their amazing hand-written report on the lexicographic path ordering, which has had tremendous influence.

One of my main interests in theorem proving became the study of the general first-order inference problem, focusing on developing the knowledge needed to implement uniform proof procedures for first-order logic. I thought that a backward chaining strategy, somewhat like Loveland's Model Elimination, would work well, but then someone showed me the x^2 = e implies commutativity group problem, on which a backward chaining prover can be cumbersome. This showed me the need for a different approach. I looked at forward chaining, but then found a problem that was propositionally simple but on which a forward chaining prover started

combining a small number of literals in numerous ways and generated a large search space. Then I thought that the non-Horn property was the problem, and tried to deal with the non-Horn aspects of first-order logic by case analysis, leading to the simplified and modified problem reduction formats. However, this approach also seemed unsatisfying and I decided that the Horn property was not essential, but what was needed was propositional efficiency like the DPLL method in first-order logic. This led to a focus on instance-based methods, starting with clause linking, then hyper-linking, semantic hyper-linking, and ordered semantic hyper-linking. The clause linking work led to the disconnection method of Billon and eventually to the disconnection calculus theorem prover of Letz and Stenz, a substantial theorem prover. Others such as Peter Baumgartner, Harald Ganzinger, Konstantin Korovin (with iProver), Koen Claessen (with Equinox), Uwe Waldmann, Chris Lynch, and others later became interested in instance-based strategies, and instance-based provers are becoming more powerful. Currently, however, the Vampire prover of Voronkov, Riazanov and Hoder still seems to be the most powerful prover overall. It will be interesting to see how instance-based and resolution-based approaches compare when instance-based methods are engineered as well as Vampire and have the best methods for equality.

It may be worthwhile to speculate on the degree to which theorem provers are becoming better, and how we might analyze this theoretically. Are we obtaining more theorems because of better engineering, better methods, or more provers? What will it take before theorem provers have wide applicability and attract substantial funding from industry? Of course already we see some applications of provers, such as the inclusion of Waldmeister in Wolfram's Mathematica, and the use of provers in hardware and software verification. Can we develop a theoretical framework for comparing provers asymptotically? Historically, logicians have looked at proof length, but what is more important for us is the total search space size. Perhaps more work needs to be done giving a machine-independent, asymptotic analysis of the size of the total search space, but this depends on the search strategy. This was the object of my book with Yunshan Zhu on the efficiency of theorem proving strategies. One thing to keep in mind is that the best prover strategies for computers may not be the same as the best strategies used by humans; this is the case, for example, in computer chess.

We really are standing on the shoulders of giants in this field. Frege and others developed first-order logic, and Herbrand developed the foundations of uniform proof procedures. Zermelo-Fraenkel set theory was developed.

Hilbert proposed a universal proof procedure for mathematics, which Goedel showed was not possible. Church and Turing developed the idea of effective computability, from which the concept of undecidability arises. Gilmore had an early instance-based method, not based on clause form. Martin Davis applied clause form to computer theorem proving. Robinson developed unification and resolution. Later the DPLL method, which has had tremendous influence, was developed. We now understand NP-completeness, and that propositional satisfiability is NP-complete. Knuth (and others) founded completion methods, and since this work the field has developed and flowered in many ways. We now have AC unification and AC completion, for example. And the dependency pair method of Arts and Giesl has become a highly refined method of proving termination, which can often produce very long proofs. We take many things for granted that arose from the hard work and brilliant insights of many predecessors. However, the field of theorem proving has not yet reached its full potential, by any means, and we do not know at this point exactly how this can occur. On hard problems, automated provers still cannot compete with human mathematicians except in very rare instances like the Robbins problem. We still do not understand exactly what is going on in the mind of a mathematician during theorem proving.

As for applications of provers, perhaps provers can contribute more to computer science in the area of software engineering and logic programming. It may be desirable to have a closer integration of computation and inference. For example, when a program functions incorrectly now, one often obtains mysterious error messages. It would be better if the program could explain why it did what it did. Integrating computation and inference could also improve program reliability and programmer productivity. Of course, I am only repeating ideas that many others understand very well and promote.

Other promising areas for research are the design of languages for the implementation of theorem provers, and the development of methods for comparing theorem proving strategies independent of how well engineered they are.

Already theorem provers are finding many applications, such as the use of Waldmeister in Wolfram's Mathematica. As another example, an article "Automated Termination Proofs for Logic Programs by Term Rewriting" appeared in the October, 2009 issue of ACM Transactions on Computational Logic; this article applies term-rewriting termination techniques to

termination of logic programs. These developments are encouraging for our field.

As advice to young researchers, I want to close by encouraging in you a love of perfection and a love of excellence. We all know that our papers should be correct and of high quality, but is this knowledge only in our head, or also in our heart? It is better for the love of perfection to be in our heart. We may need to proofread our papers multiple times to make them as nearly perfect as possible. I have not always followed this practice myself, but I was able to instill this love in my student, and he has begun getting more acceptances as a result. Every symbol in our papers should be as correct, clear, and concise as possible.

Thank you for the chance to share some of my accumulated experiences in this brief talk. This is a tremendously exciting area to be in, one that I have always felt is crucial for computer science. I wish you all the best in the exciting and challenging times ahead.