Conclusion

This book presented the key ideas needed to get started with productive use of Coq. Many people have learned to use Coq through a variety of resources, yet there is a distinct lack of agreement on structuring principles and techniques for easing the evolution of Coq developments over time. Here I have emphasized two unusual techniques: programming with dependent types and proving with scripted proof automation. I have also presented other material relevant to keeping Coq code beautiful and scalable.

Part of the attraction of Coq and similar tools is that their logical foundations are small. A few pages of LaTeX code suffice to define CIC, Coq's logic, yet there do not seem to be any practical limits on which mathematical concepts may be encoded on top of this modest base. At the same time, the *pragmatic* foundation of Coq is vast, encompassing tactics, libraries, and design patterns for programs, theorem statements, and proof scripts. I hope the preceding chapters have given a sense of just how much there is to learn before it is possible to use Coq with the same ease with which many readers write informal proofs. The payoff of this learning process is that proofs, especially those with many details to check, become much easier to write than they are on paper. Further, the truth of such theorems may be established with much greater confidence, even without reading proof details.

I have not attempted to describe all the many parts of Coq. My advice to readers is to read through the Coq manual, front to back, at some level of detail. Get a sense for which bits of functionality are available. Dig more into those categories that sound relevant to the developments you want to build, and keep the rest in mind in case they come in handy later.

In a domain as rich as this one, the learning process never ends. The Coq Club mailing list (linked from the Coq home page) is a great place to get involved in discussions of the latest improvements, or to ask questions about stumbling blocks that you encounter. I believe the best

way to learn is to get started using Coq to build some development that interests you.