Part IV
More Applications of the Unknown Component Problem

The unknown component problem can be applied in several domains, such as: logic synthesis; model matching; testing; game theory; cryptography (FSM-based cryptosystems). Logic synthesis has been thoroughly discussed in Part III. In this part we consider some more applications: supervisory control, testing, game theory, synthesis with respect to ω-specifications.

Chapter 15 introduces the supervisory control problem, that deals with solving the unknown component problem for the controller's topology, and must also take into account partial controllability (and partial observability). We show how to model the problem in the framework of solving language equations for full controllability and partial controllability; for those cases we characterize controllers that are solutions. We also introduce the notion of weak controllers that model an extended class of controllers with respect to a looser notion of partial controllability that has a natural interpretation.

Chapter 16 addresses the problem of testing a component embedded within a modular system [131], that is known as the problem of testing in context or embedded testing. Solving this problem, the modular system is conveniently represented as two communicating machines, the embedded component machine, and the context machine that models the remaining part of the system which is assumed to be correctly implemented.

Chapter 17 discusses the application of language solving to games. First we summarize some of the existing results in game theory. Then we show how the theory of this book and the BALM program can be applied to games. Typically the rules of the game are described in the fixed part of the description which also keeps track of the game state. In addition, it is usually convenient to have an output which indicates if the game has been won and by whom. This reduces the specification to a simple automaton which monitors this output. The unknown component represents one players winning strategy, which is to be solved for if it exists. The other player is represented by unconstrained inputs. To restrict this player to legitimate moves, we map all illegal moves as a win for the first player. However, a winning strategy (or at least a non-losing strategy) for the first player is one where he must be able to win (or tie) no matter what (legal) move the second player makes. The game of tic-

tac-toe is used to illustrate how a game can be modeled BLIF-MV and solved using BALM. The techniques described in this chapter generally apply to finite games where after a finite number of moves, the game is over. However, if a strictly winning strategy is desired, it might be necessary to model the game with ω-automata (as discussed in Chap. 18).

Chapter 18 illustrates how, in certain situations, the theory and implementation developed for finite-word automata can be applied to infinite-word automata. Although the theory of language solving can be extended formally (as discussed in Chap. 4 in Part I), the complexity of solving becomes doubly exponential in general. In this chapter, a restricted form of ω-language solving is discussed, which almost fits into the finite-word implementation, as available in BALM, except for a small modification of the derived result at the end.