Applications of Net Synthesis
In this last part we examine a few domains where the techniques of region-based net synthesis have been successfully applied. This presentation does not purport to be an exhaustive survey. We rather describe some illustrative case studies in the first three chapters.

**Extracting Concurrency** Petri nets can provide a compact description of a concurrent system due to their explicit representation of concurrency. Thus Petri net synthesis can be used for extracting concurrency from specifications initially given by a transition system or a language. A related concern is to obtain a distributed implementation of the concurrent model. One can amend the synthesis algorithm in order to synthesize a so-called distributable Petri net (a Petri net free from distributed conflicts) which henceforth can be deployed on a distributed asynchronous architecture. Namely, one can translate a distributable Petri net into an equivalent network of communicating automata.

**Process Discovery** Synthesizing a net system from a language, seen as a set of event logs of a process one wants to discover or reconstruct, falls under the more general field of process discovery or model identification. Net synthesis can be used as a technique to reconstruct a model from its execution traces or to construct such a model from some of its expected behaviours. Of course the log does not provide an exhaustive description of system behaviour. Thus a discovery process always implicitly includes some generalization process. Interestingly enough the synthesized net provides the best over-approximation of the language given as input.

**Control Synthesis** A place of a net system can be interpreted as a (synchronous) constraint on the behaviour of the system. One can indeed progressively restrict the behaviour of a net system by adding new places. Thus one can be tempted to synthesize places in order to enforce specific constraints on system behaviour. In this spirit, region-based synthesis has found applications in the field of supervisory control of discrete event systems.

The last chapter provides a quick overview of the computer-assisted solution to the complete state coding problem (CSC) for asynchronous circuits based on elementary regions. The method is supported by a tool, called PETRIFY, which includes a net synthesis procedure that was used, since then, in many different contexts remote from the domain of asynchronous circuit design for which PETRIFY was originally conceived.