The notion of type in the $\lambda$-calculus inspires a notion of type in the $\pi$-calculus. Part III of the book examines typed $\pi$-calculi; a number of type systems are developed that can detect certain run-time errors. Part IV of the book continues this development by showing how one can use type systems for the $\pi$-calculus to reason about the behaviour of processes.

The remainder of the book describes how the $\pi$-calculus can be used as a meta-language for interpreting central ideas from programming language theory.

An alternative to the name passing ability of the $\pi$-calculus is *process passing*, i.e., transmitting an entire process along a channel. Process passing is central to describing, for instance, the World Wide Web where a WWW page may transfer a Java applet to the client machine reading the page.

Part V describes the relationship between process passing and name passing. The chapters of this part introduce and analyse HO$\pi$ (higher-order $\pi$), a calculus that allows name passing, and prove that in fact HO$\pi$ can be interpreted in the basic $\pi$-calculus.

Part VI relates the $\pi$-calculus to the $\lambda$-calculus, known to computer scientists as the theory underlying functional programming languages. In particular, this part of the book describes the interpretation of call-by-value and call-by-name reduction strategies of the $\lambda$-calculus in the $\pi$-calculus and extends the interpretations to typed settings. The equivalence on untyped $\lambda$-terms induced by the call-by-name encoding into the $\pi$-calculus corresponds to the equivalence on Lévy–Longo trees.

The $\pi$-calculus is a calculus of communicable references and is therefore well suited for the study of object-oriented programming languages, where references to resources play an important part. Part VII describes an encoding of a small class-based object-oriented programming language in the $\pi$-calculus and shows that one can use the encoding together with reasoning principles from the book to reason about properties of programs.

The book also contains an extensive bibliography, covering a large part of the existing $\pi$-calculus literature.

Sangiorgi and Walker have written an extremely valuable account of the $\pi$-calculus which should be appreciated by researchers in theoretical computer science and related areas of symbolic logic, as it covers the basic theory rigorously and in great depth. It is a valuable reference for those of us who are active researchers in the area. Readers who are primarily interested in a short introduction to the $\pi$-calculus with an emphasis on somewhat smaller examples should start with Milner's textbook ***Communicating and mobile systems: the $\pi$-calculus***, Cambridge University Press, 1999, as is also suggested by the authors of the present book.

HANS HÜTTEL

BRICS, Department of Computer Science, Aalborg University, Fredrik Bajersvej 7E, DK-9220 Aalborg Ø, Denmark. hans@cs.auc.dk.

SVEN OVE HANSSON. ***The structure of values and norms.*** Cambridge studies in probability, induction, and decision theory. Cambridge University Press, Cambridge, New York, etc., 2001, xiii + 314 pp.

"*Prolegomena*" (pp. 1–11) examines the virtues and dangers of formalization, defending the use of logic provided that a trade-off between simplicity and faithfulness be reached. As suggested by the title, the book contains two parts, the former on values (Chapters 2–8), the latter on norms (Chapters 9–13). There is also an "*Epilogue*" and then sixty pages devoted to the proofs of the statements ("theorems," "observations," or "corollaries") asserted in the first two parts.

Values are defined by means of the dyadic concepts of preference ("better," "worse," "equal in value to"), the monadic concepts ("good," etc.) being studied (Chapter 8) by virtue of the dyadic ones. A major distinction is considered regarding the objects of preferences: whether they are mutually exclusive (Chapters 2–4) or not, that is *combinative* (Chapters 5–7).