Type theories support increasingly the expressiveness of programming languages, theorem provers and logics without losing important theoretical properties. Thus, type theories are the foundation of safe (paradox-free) logics, safe theorem proving and safe programming languages. Types have existed since the times of the ancient Greeks as is witnessed by Euclid’s formulation of Geometry in his *Elements*. In his *Elements*, Euclid prevented undesirable situations, like considering whether two points (instead of two lines) are parallel, by distinguishing classes of objects. The 20th century has seen an explosion of type theories that were invented to avoid undesirable situations and to guarantee safety.

*Rewriting* is a method for applying the rules of logic, mathematics, or computation in a stepwise manner. It is the action of replacing a subexpression which is matched by an instance of one side of a rule, by the corresponding instance of the other side of the same rule. This supports reasoning about efficient computation strategies and equivalences between propositions or programs. Rewriting existed since the ancient times of the Babylonians who developed techniques for symbolic computations through their work on *Algebra* which can be viewed as rewriting. The 20th century has seen an explosion of work on rewriting systems and on establishing important properties of rewriting such as termination or strong normalisation and confluence. A rewriting system that terminates/halts is one that avoids infinite computation sequences of the form \[ P \rightarrow P_1 \rightarrow P_2 \rightarrow \cdots \]. A confluent rewriting system guarantees that the result of rewriting is independent of the order in which the rules are used. For example, \( 1 + 2 + 3 \) should rewrite to 6, no matter how we evaluate it.

In addition to the advances on type theory and term rewriting in the 20th century, attempts in the last quarter of the 20th century, were made at combining the two disciplines. This was due to the large consensus that type theory and term rewriting are both essential for computation and so, combining them will lead to more benefits. In such combinations, all the earlier desirable properties (such as safety, termination, confluence, etc.) must be preserved.

This volume contains six articles devoted to the study of type theory, term rewriting and their combinations. In particular, the articles are devoted to explicit substitution, confluence and normalisation in type-free and typed versions of the \( \lambda \)-calculus and rewrite systems. It is a selection of papers based on talks that were presented at the *First International School on Type Theory and Term Rewriting* that took place in September 1996 in Glasgow Scotland. Those papers went through a thorough reviewing process and three stages of revisions; less than half of the submitted papers appear in this volume.

The article of Glauert, Kennaway and Khasidashvili extends the notions of normalisation and needed reduction by considering ‘results’ instead of normal forms. The article introduces interesting notions of neededness and studies minimal and optimal normalisation which can be applicable to a general notion of rewrite systems.

Kamareddine and Ríos’s article establishes a connection between the two styles of explicit substitution (the \( \sigma \) style where combinator-like operators are used to represent substitutions, and the \( \lambda \sigma \) style which remains as close as possible to the \( \lambda \)-calculus) and considers both type-free and typed versions. The questions of confluence, preservation of termination and safety (subject reduction) are considered.

The article of Khasidashvili and Piperno establishes that a certain simple property (the alternation property) implies normalisation. This result is used to obtain new normalisation proofs for two typed theories: Curry’s simply typed lambda calculus and terms typable with
intersection types. The reader is encouraged to look for the interesting way terms are divided and new classes of terms are defined (e.g. the hyperbalanced terms).

The article of Kfoury embeds the standard lambda calculus in a bigger calculus satisfying linearity and provides a characterisation of weak and strong normalisation for standard lambda terms.

The article of Klop, van Oostrom and de Vrijer provides a geometric proof of confluence by decreasing diagrams. The notion of 'decreasing diagrams' was originally developed by De Bruijn, in order to formulate a confluence criterion applicable to various members of the Automath family of typed lambda calculi. The proof of the decreasing diagram theorem given here is by an analysis of the geometry of, possibly infinite, reduction diagrams.

The article of Mellies establishes a useful theory of normalisation that he applies to the $\lambda\sigma$ style of explicit substitutions. In particular he shows that every needed (reduction) strategy of the $\lambda\sigma$-calculus is normalising.

We are grateful to the following people for their thorough refereeing contribution: Mauricio Ayala-Rincon, Franco Barbanera, Gilles Barthe, Inge Bethke, Roel Bloo, Mariangola Dezani, Maribel Fernandez, Herman Geuvers, Philippe de Groote, Therese Hardin, Jean-Pierre Jouannaud, Fairouz Kamareddine, Zurab Khasidashvili, Richard Kennaway, Jan Willem Klop, Pierre Lescanne, Zhaoui Luo, Peter Moller Neergaard, Cesar Munoz, Adolfo Piperno, Fer-Jan de Vries, Roel de Vrijer, Elena Zucca, Jan Zwanenburgh.

FAIROUZ KAMAREDDINE AND JAN WILLEM KLOP