COLLEGE ED
DOCTORAL MATHEMATIQUES
PAYS DE LA LOIRE STIC

Nantes Université

THÈSE DE DOCTORAT DE

NANTES UNIVERSITÉ

ÉCOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Yann Leray

Investigations into Conversion in Type Theory, and its Extension with Rewrite Rules

Thèse présentée et soutenue à Nantes, le 19 novembre 2026
Unité de recherche : LS2N

Rapporteurs avant soutenance :

Brigitte PIENTKA	Professor, McGill University
Herman GEUVERS	Professor, Radboud Universiteit

Composition du Jury :

Attention, en cas dabsence dun des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour sassurer quelle est conforme et devra être répercutée sur la couverture de thèse

Examinateurs :	Frédéric BLANQUI	Directeur de Recherche, Inria
	Jesper COCKX	Assistant Professor, TU Delft
Dir. de thèse :	Nicolas TABAREAU	Directeur de Recherche, Inria
Co-enc. de thèse :	Matthieu SOZEAU	Chargé de Recherche, Inria
	Théo WINTERHALTER	Chargé de Recherche, Inria

COLLEGE ED
DOCTORAL MATHEMATIQUES
PAYS DE LA LOIRE STIC

Nantes Université

Titre : Étude de la conversion en théorie des types, et son extension aux règles de réécriture

Mot clés : théorie des types, conversion, assistants de preuve, règles de réécriture

Résumé : Dans les langages de programmation simplement typés, un certain nombre de propriétés et notamment la préservation du typage sont recherchées afin que les intuitions soient vérifiées. Dans les langages avec des types dépendants, ces propriétés deviennent requises pour implémenter un vérificateur de typage et rendre le langage utilisable. Les preuves de ces propriétés, simples quand les types le sont, requièrent cependant une double propriété non-triviale en théorie des types : l'injectivité et la non-confusion de la conversion pour les constructeurs de type. Si l'on se cantonne aux approches qui ne requièrent pas une logique plus forte que le langage étudié, les techniques de réécriture permettent de supporter une conversion non-typée, sans règle de conversion . Dans cette thèse, nous allons d'abord étudier comment les preuves existantes peuvent s'adapter à une conversion typée, et avec des règles suffisamment restreintes. Dans un second temps, nous étudierons les règles de réécriture en théorie des types, qui permettent d'étendre la conversion de manière libre et dirigée par l'utilisateur, les restrictions à imposer pour que les bonnes propriétés du langage complet et leurs preuves restent applicables, comment implémenter et mettre en place au mieux le mécanisme pour les règles de réécriture et comment celles-ci peuvent être mises en uvre dans des démonstrations et développements en pratique.

Title: Investigations into Conversion in Type Theory, and its Extension with Rewrite Rules

Keywords: type theory, conversion, proof assistants, rewrite rules

Abstract: In simply-typed programming languages, a number of properties–notably Subject Reduction–are sought after so that intuitions about the language are verified. In dependently-typed languages, these properties become necessary so that a typechecker can be implemented, and the language be usable. The proofs for these properties, if simple for simple types, do require a non-trivial twin property in the context of type theories: injectivity and no-confusion of conversion on type constructors. If we keep to methods that do not involve an ambient logic which is stronger than the studied language, rewriting techniques can support an untyped conversion, without laws. In this thesis, we will first study how these existing proofs adapt to a typed conversion, with sufficiently restrictive laws. Secondly, we will look into rewrite rules in type theory, which allow the extension of conversion in a free and user-directed manner. We will see the restrictions that must be imposed to preserve the good properties of the global language, how to best implement and frame the rewrite rule mechanism in the language and how they can serve in practice for demonstrations and developments.