Curry-Howard Isomorphism
Daniel Knight
March 5, 2019

@incollection {curry:34,
  author = {H.B. Curry},
  journal = {Proceedings of the National Academy of Sciences},
  title   = {Functionality in Combinatory Logic},
  year   = {1934},
  volume = {20},
  number = {11},
  pages = {584-590},
  doi = {10.1073/pnas.20.11.584},
  address = {Department of Mathematics, The Pennsylvania State College},
}

Summary: The paper begins with an introduction to combinatory logic and a rationale for Curry's desire to study computable functions in that setting. Curry goes on to define notation and axioms with a special importance placed on axiom F, the definition of what it means to be a function in combinatory logic, and axiom Π , an axiom pertaining to self application of F. Using these, he observes that every type of a function could be read as a provable proposition and vice versa. From this basis, he defines F', the property of being a function in combinatory logic. He then shows that F' precludes axiom Π and leads to contradictions such as Russell's and Epimenide's paradoxes.
Evaluation: Although not the most seminal of Curry's work, the observations that function types could be viewed as propositions was an important first step in understanding the Isomorphism. His exploration and observation of function types as propositions paved the way for his future findings in 1958 and the eventual Curry-Howard correspondence.

@incollection{howard:80,
  author = {William A. Howard},
  journal = {To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism},
  title = {The Formulae-as-types notion of construction},
  year = {1980 (originally circulated 1969)},
  pages = "479-490",
  ISBN = {978-0-12-349050-6}
}

Summary: The paper opens with a background in Intuitionistic Propositional Logic and Natural Deduction. Howard explains how the connectives $(\wedge, \rightarrow)$ in this logic directly relate to computation types (x, +) in simply typed lambda calculus. He enriches the analogy further by pointing out the similarity between the normalization of terms and cut elimination in a proof.

Howard then adds a discussion of Heyting arithmetic to introduce the idea that the Universal and Existential quantifiers could correspond to new types themselves.
Evaluation: This paper's exploration of the simply typed lambda calculus and Intuitionistic Propositional Logic was a massive academic finding. Prior to this, comparisons between the two systems had been informal at best. As such, Howard's contribution laid the groundwork for a deep equivalence between logic and programming languages that has allowed the two fields to continually inspire one another.

@incollection{barendregt:92,
author = {Barendregt, Henk},
title = {Lambda Calculi with Types},
booktitle = {Handbook of Logic in Computer Science},
publisher = {Oxford University Press},
year = {1992},
editor = {Abramsky, S. and Gabbay, D.M. and Maibaum, T.S.E.},
volume = {2},
pages = {117--309}
}

Summary: This paper begins with an introduction to the type free lambda calculus and its features. Barendgret then details typing a la Church and typing a la Curry and uses that as a basis for comparison between different type systems. He goes on to discuss the lambda cube as a geometric representation of different lambda calculi with each axis representing a particular feature of either terms depending on types, types depending on terms, or types depending on types. Using the lambda cube, he goes on to describe pure type systems and verify the viewpoint of propositions as types.
Evaluation: The paper provides a strong geometric foundation for understanding typed programming languages. It's greatest contribution can be thought of relating lambda calculi to one another in a way that is visual, intuitive and useful and relating them to logical systems. This intuition laid the framework for work in dependent types and more advanced type systems.