# Metamathematics based on nominal terms

Murdoch J. Gabbay

Thanks to the ARW organisers, and to Nick Taylor

April 11, 2011

# Metamathematics

Computer science is the mathematical study of the tools that make programming possible.

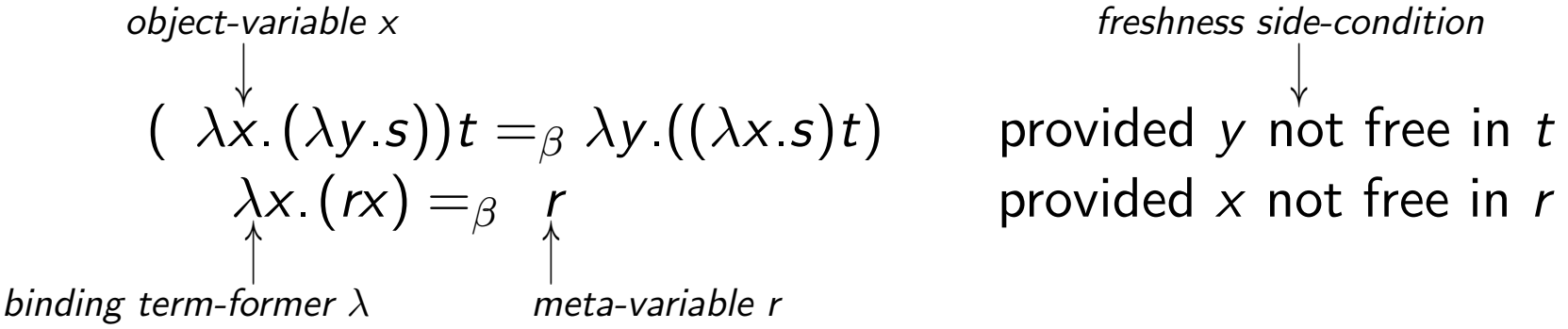
This requires formalism (of course).

A formalism is a mathematically precise model of a certain class of informal behaviour. Why make a mathematically precise model? Because that's how we implement and prove theorems.

So let's model specifications of logical and computational systems with binding:

# Informal specification

If you've studied functional programming then you'll be familiar with the following informal specification:




This is typical. Similar equalities specify substitution, logics with quantifiers, $\pi$-calculus, and more.

They all have common structure as annotated above.

My recent research has concentrated (in collaboration) on developing formalisms and semantics for that common structure.

# Formalisms and semantics

In collaboration with Pitts, Urban, Cheney, Fernández, Mathijssen, Dowek, and Mulligan, we have:

- Captured the structure above in formal languages: nominal rewriting and nominal algebra (equality) and permissive-nominal logic (first-order logic).
- Given semantics in nominal sets and simply-typed $\lambda$-calculus.
- Axiomatised significant theories, including substitution, $\lambda$-calculus, first-order logic, and arithmetic—and proved these theories sound and complete.
- Developed type systems.

# Conclusions

This can be read as proof-of-concept for a nominal theorem-prover.

The underlying mathematics is abstract, and flexible.

This can also be read as a new tool in language design. Structures with binding seem to appear in meta-programming, module systems, concurrency, proof-search, and more.

Find me in the poster session to talk about this, and applications to type systems, concurrency, and more.

This is just a beginning.