Church's lambda calculus reduces all of computation to what?

What is the bridge from lambda calculus to a language like ML or Haskell?

What are the three kinds of terms in lambda calculus?

What's the difference between an *internal* and *external* language?

Which way does function application associate?

The body of a lambda abstraction extends to where?

What's the difference between a *bound* and *free* variable.

What's a *closed term*?

Describe computation in the lambda calculus.

What is an evaluation strategy?

Do not confuse with ...

Name some evaluation strategies.

Which is the most popular?

What is *full beta-reduction*?

What is the *normal order* strategy?

What is the *call-by-name* strategy?

What is a *value*?

What is the *call-by-value* strategy?

What's the difference between *strict* and *non-strict* evaluation strategies?

What evaluation strategy does Haskell use?

How does the lambda calculus represent multiple argument functions?

What is *Church encoding*?

It is analogous to what?

Define the two Church booleans.

Church encode `if`/`else` expressions.