Programming Haskell Graham Hutton

X 63f2fb61fc and one from 63f2fb61fc Parsers using combinators have been used extensively in the prototyping of compilers and processors for domain-specific languages such as natural-language user interfaces to databases, where complex and varied semantic actions are closely integrated... 63f2fb61fc × 63f2fb61fc A distinct, but related concept outside of functional programming, which is popular in object-oriented programming, is called nullable types (often expressed as A?). The core difference between option types and nullable types is that option types support nesting (e.g. Maybe (Maybe String) ≠ Maybe String), while nullable types... 63f2fb61fc Applicative functors first appeared as a library feature in Haskell, but have since spread to other languages such as Idris, Agda, OCaml, Scala... 63f2fb61fc The curried form of this function treats the first argument as a parameter, so as to create a family of functions... 63f2fb61fc

Mutual recursion ISBN 978-0-52169269-4. Cambridge University Press. ISBN 978-0-26208281-5. MIT Press. Programming in Haskell. Computer Science. Mutual recursion is very common in functional programming and in some problem domains, such as recursive descent parsers, where the datatypes are naturally mutually recursive. In mathematics and computer science, mutual recursion is a form of recursion where two or more mathematical or computational objects, such as functions or datatypes, are defined in terms of each other. Hutton, Graham (2007) 63f2fb61fc

Folds are in a sense dual to unfolds, which take a seed value and apply a function corecursively to decide how to progressively construct a corecursive data structure, whereas a fold recursively... 63f2fb61fc and produces objects in 63f2fb61fc

Applicative functor Applicative functors allow for functorial computations to be sequenced (unlike plain functors), but don't allow using results from prior computations in the definition of subsequent ones (unlike monads). pp. Applicative functors are the programming equivalent of lax monoidal functors with tensorial strength in category theory. "Control. Description of the Applicative typeclass in the Haskell docs In functional programming, an applicative functor, or an applicative for short, is an intermediate structure between functors and monads. Applicative". 157–163.). In category theory they are called closed monoidal functors. Programming in Haskell (2 ed. Hutton, Graham (2016) 63f2fb61fc

) 63f2fb61fc Haskell's semantics are historically based on those of the Miranda programming language, which served to focus the efforts of the initial Haskell working group. The last formal specification of the language was made in July 2010, while the development of GHC continues to expand Haskell via language extensions. 63f2fb61fc

Fold (higher-order function) Journal of Functional Programming. Typically, a fold is presented with a combining function, a top node of a data structure, and possibly some default values to be used under certain conditions. 2018-04-10. 9 (4): 355–372 In functional programming, fold (also termed reduce, accumulate, aggregate, compress, or inject) refers to a family of higher-order functions that analyze a recursive data structure and through use of a given combining operation, recombine the results of recursively processing its constituent parts, building up a return value. "A tutorial on the universality and expressiveness of fold" (PDF). The fold then proceeds to combine elements of the data structure's hierarchy, using the function in a systematic way. Hutton, Graham (1999) 63f2fb61fc

. 63f2fb61fc → 63f2fb61fc Both the concept of a monad and the term originally come from category theory, where a monad is... 63f2fb61fc {\displaystyle Z.} 63f2fb61fc (63f2fb61fc

Option type g. It consists of a constructor which either is empty (often named None or Nothing), or which encapsulates the original data type A (often written Just A or Some A). , it is used as the return type of functions which may or may not return a meaningful value when they are applied. In programming languages (especially functional programming languages) and type theory, an option type or maybe type is a polymorphic type that represents In programming languages (especially functional programming languages) and type theory, an option type or maybe type is a polymorphic type that represents encapsulation of an optional value; e 63f2fb61fc

that takes two arguments, one from 63f2fb61fc , 63f2fb61fc

Currying Hutton, Graham; Jones In mathematics and computer science, currying is the technique of translating a function that takes multiple arguments into a sequence of families of functions, each taking a single argument. Vol. Curry, Haskell; Feys, Robert (1958). Combinatory logic. Amsterdam, Netherlands: North-Holland Publishing Company. I (2 ed.) 63f2fb61fc

Haskell is used in academia and industry. As of May 2021, Haskell was the 28th most popular programming... 63f2fb61fc f 63f2fb61fc Z 63f2fb61fc {\displaystyle X} 63f2fb61fc Y 63f2fb61fc {\displaystyle Y,} 63f2fb61fc Y 63f2fb61fc {\displaystyle f:(X\times Y)\to Z} 63f2fb61fc X 63f2fb61fc : 63f2fb61fc Applicative functors were introduced in 2008 by Conor McBride and Ross Paterson in their paper Applicative programming with effects. 63f2fb61fc

Parser combinator Parser combinators enable a recursive descent parsing strategy that facilitates modular piecewise construction and testing. Monadic Parser In computer programming, a parser combinator is a higher-order function that accepts several parsers as input and returns a new parser as its output. Hutton,

Graham; Meijer, Erik. combinators written in the Haskell language based on the same algorithm. This parsing technique is called combinatory parsing. Hutton 1992. In this context, a parser is a function accepting strings as input and returning some structure as output, typically a parse tree or a set of indices representing locations in the string where parsing stopped successfully. Frost & Launchbury 1989 63f2fb61fc

Monad (functional programming)). A. Option, List, etc. g. More formally, a monad is a type constructor M equipped with two operations, return : <A>(a : A) -> M(A) which lifts a value into the monadic context, and bind : <A,B>(m_a : M(A), f : A -> M(B)) -> M(B) which chains monadic computations. McCann's answer (Jul 23 '10 at 23:39) How and why does the Haskell Cont monad work. chapter 9. In simpler terms, monads can be thought of as interfaces implemented on type constructors, that allow for functions to abstract over various type constructor variants that implement monad (e. to Haskell 98. Graham Hutton (2016) Programming in In functional programming, monads are a way to structure computations as a sequence of steps, where each step not only produces a value but also some extra information about the computation, such as a potential failure, non-determinism, or side effect. C 63f2fb61fc

Where recursion allows programs to operate on arbitrarily complex data, so long as they can be reduced to simple data (base... 63f2fb61fc In the prototypical example, one begins with a function 63f2fb61fc

Exception handling (programming) , Compiler based Structured Exception Handling section Graham Hutton, Joel Wright, "Compiling Exceptions Correctly Archived 2014-09-11 at In computer programming, several language mechanisms exist for exception handling. Execution is transferred to a catch. Retrieved 2009-11-21. One mechanism to transfer control, or raise an exception, is known as a throw; the exception is said to be thrown. The term exception is typically used to denote a data structure storing information about an exceptional condition 63f2fb61fc

Z 63f2fb61fc

Corecursion Jeremy Gibbons; Graham Hutton (April 2005) In computer science, corecursion is a type of operation that is dual to recursion. 10 (7): 751–768. doi:10. Whereas recursion works analytically, starting on data further from a base case and breaking it down into smaller data and repeating until one reaches a base case, corecursion works synthetically, starting from a base case and building it up, iteratively producing data further removed from a base case. Journal of Universal Computer Science. 3217/jucs-010-07-0751. Functional Programming". A similar but distinct concept is generative recursion, which may lack a definite "direction" inherent in corecursion and recursion. Put simply, corecursive algorithms use the data that they themselves produce, bit by bit, as they become available, and needed, to produce further bits of data 63f2fb61fc

Haskell It is named after logician Haskell Curry. ISBN 978-0521643382. Haskell's main implementation is the Glasgow Haskell Compiler (GHC). Programming in Haskell. Functional Programming through Multimedia. Haskell pioneered several programming language features such as type classes, which enable type-safe operator overloading, and monadic input/output (IO). Cambridge Haskell () is a general-purpose, statically typed, purely functional programming language with type inference and lazy evaluation. New York: Cambridge University Press. Hutton, Graham (2007) 63f2fb61fc