# Curry–Howard correspondence

en.wikipedia.org

March 20, 2022

# Contents

**1 Curry–Howard correspondence** **3**

1.1 Origin, scope, and consequences . . . . 4

1.2 General formulation . . . . 6

1.3 Corresponding systems . . . . 8

1.4 Related proofs-as-programs correspondences . . . . 13

1.5 Examples . . . . 15

1.6 Other applications . . . . 18

1.7 Generalizations . . . . 18

1.8 References . . . . 19

1.9 Further reading . . . . 25

1.10 External links . . . . 26

**2 Contributors** **27**

**List of Figures** **33**

**3 Licenses** **37**

3.1 GNU GENERAL PUBLIC LICENSE . . . . 37

3.2 GNU Free Documentation License . . . . 38

3.3 GNU Lesser General Public License . . . . 39

# 1 Curry–Howard correspondence

Isomorphism between computer programs and constructive mathematical proofs

**A proof written as a functional program**

```
plus_comm =
fun n m : nat =>
nat_ind (fun n0 : nat => n0 + m = m + n0)
  (plus_n_0 m)
  (fun (y : nat) (H : y + m = m + y) =>
   eq_ind (S (m + y))
     (fun n0 : nat => S (y + m) = n0)
     (f_equal S H)
     (m + S y)
     (plus_n_Sm m y)) n
     : forall n m : nat, n + m = m + n
```

A proof of commutativity of addition on natural numbers in the proof assistant Coq[1]. `nat_ind` stands for mathematical induction[2], `eq_ind` for substitution of equals, and `f_equal` for taking the same function on both sides of the equality. Earlier theorems are referenced showing $m = m + 0$ and $S(m + y) = m + Sy$.

In programming language theory[3] and proof theory[4], the **Curry–Howard correspondence** (also known as the **Curry–Howard isomorphism** or **equivalence**, or the **proofs-as-programs** and **propositions-** or **formulae-as-types interpretation**) is the direct relationship between computer programs[5] and mathematical proofs[6].

It is a generalization of a syntactic analogy[7] between systems of formal logic and computational calculi that was first discovered by the American mathematician[8] Haskell Curry[9] and the logician[10] William Alvin Howard[11].[1] It is the link between logic and computation that is usually attributed to Curry and Howard, although the idea is related to the operational interpretation of intuitionistic logic[12] given in various formulations by L. E. J. Brouwer[13],

[1] https://en.wikipedia.org/wiki/Coq
[2] https://en.wikipedia.org/wiki/Mathematical_induction
[3] https://en.wikipedia.org/wiki/Programming_language_theory
[4] https://en.wikipedia.org/wiki/Proof_theory
[5] https://en.wikipedia.org/wiki/Computer_program
[6] https://en.wikipedia.org/wiki/Mathematical_proof
[7] https://en.wikipedia.org/wiki/Analogy
[8] https://en.wikipedia.org/wiki/Mathematician
[9] https://en.wikipedia.org/wiki/Haskell_Curry
[10] https://en.wikipedia.org/wiki/Logician
[11] https://en.wikipedia.org/wiki/William_Alvin_Howard
[12] https://en.wikipedia.org/wiki/Intuitionistic_logic
[13] https://en.wikipedia.org/wiki/L._E._J._Brouwer

Arend Heyting[14] and Andrey Kolmogorov[15] (see Brouwer–Heyting–Kolmogorov interpretation[16])[2] and Stephen Kleene[17] (see Realizability[18]). The relationship has been extended to include category theory[19] as the three-way **Curry–Howard–Lambek correspondence**.

## 1.1 Origin, scope, and consequences

The beginnings of the **Curry–Howard correspondence** lie in several observations:

1. In 1934 Curry[20] observes that the types[21] of the combinators could be seen as axiom-schemes[22] for intuitionistic[23] implicational logic.[3]
2. In 1958 he observes that a certain kind of proof system[24], referred to as Hilbert-style deduction systems[25], coincides on some fragment to the typed fragment of a standard model of computation[26] known as combinatory logic[27].[4]
3. In 1969 Howard[28] observes that another, more "high-level" proof system[29], referred to as natural deduction[30], can be directly interpreted in its intuitionistic[31] version as a typed variant of the model of computation[32] known as lambda calculus[33].[5]

In other words, the Curry–Howard correspondence is the observation that two families of seemingly unrelated formalisms—namely, the proof systems on one hand, and the models of computation on the other—are in fact the same kind of mathematical objects.

If one abstracts on the peculiarities of either formalism, the following generalization arises: *a proof is a program, and the formula it proves is the type for the program.* More informally, this can be seen as an analogy[34] that states that the return type[35] of a function (i.e., the type of values returned by a function) is analogous to a logical theorem, subject to hypotheses corresponding to the types of the argument values passed to the function; and that the program to compute that function is analogous to a proof of that theorem. This sets a form

[14] https://en.wikipedia.org/wiki/Arend_Heyting
[15] https://en.wikipedia.org/wiki/Andrey_Kolmogorov
[16] https://en.wikipedia.org/wiki/Brouwer%E2%80%93Heyting%E2%80%93Kolmogorov_interpretation
[17] https://en.wikipedia.org/wiki/Stephen_Kleene
[18] https://en.wikipedia.org/wiki/Realizability
[19] https://en.wikipedia.org/wiki/Category_theory
[20] https://en.wikipedia.org/wiki/Haskell_Curry
[21] https://en.wikipedia.org/wiki/Typed_lambda_calculus
[22] https://en.wikipedia.org/wiki/Axiom-scheme
[23] https://en.wikipedia.org/wiki/Intuitionism
[24] https://en.wikipedia.org/wiki/Proof_calculus
[25] https://en.wikipedia.org/wiki/Hilbert-style_deduction_system
[26] https://en.wikipedia.org/wiki/Model_of_computation
[27] https://en.wikipedia.org/wiki/Combinatory_logic
[28] https://en.wikipedia.org/wiki/William_Alvin_Howard
[29] https://en.wikipedia.org/wiki/Proof_calculus
[30] https://en.wikipedia.org/wiki/Natural_deduction
[31] https://en.wikipedia.org/wiki/Intuitionistic
[32] https://en.wikipedia.org/wiki/Model_of_computation
[33] https://en.wikipedia.org/wiki/Lambda_calculus
[34] https://en.wikipedia.org/wiki/Analogy
[35] https://en.wikipedia.org/wiki/Return_type

of logic programming[36] on a rigorous foundation: *proofs can be represented as programs, and especially as lambda terms*, or *proofs can be* ***run***.

The correspondence has been the starting point of a large spectrum of new research after its discovery, leading in particular to a new class of formal systems[37] designed to act both as a proof system[38] and as a typed functional programming language[39]. This includes Martin-Löf[40]'s intuitionistic type theory[41] and Coquand[42]'s Calculus of Constructions[43], two calculi in which proofs are regular objects of the discourse and in which one can state properties of proofs the same way as of any program. This field of research is usually referred to as modern type theory[44].

Such typed lambda calculi[45] derived from the Curry–Howard paradigm led to software like Coq[46] in which proofs seen as programs can be formalized, checked, and run.

A converse direction is to *use a program to extract a proof*, given its correctness[47]—an area of research closely related to proof-carrying code[48]. This is only feasible if the programming language[49] the program is written for is very richly typed: the development of such type systems has been partly motivated by the wish to make the Curry–Howard correspondence practically relevant.

The Curry–Howard correspondence also raised new questions regarding the computational content of proof concepts that were not covered by the original works of Curry and Howard. In particular, classical logic[50] has been shown to correspond to the ability to manipulate the continuation[51] of programs and the symmetry of sequent calculus[52] to express the duality between the two evaluation strategies[53] known as call-by-name and call-by-value.

Speculatively, the Curry–Howard correspondence might be expected to lead to a substantial unification[54] between mathematical logic and foundational computer science:

Hilbert-style logic and natural deduction are but two kinds of proof systems among a large family of formalisms. Alternative syntaxes include sequent calculus[55], proof nets[56], calcu-

[36] https://en.wikipedia.org/wiki/Logic_programming
[37] https://en.wikipedia.org/wiki/Formal_system
[38] https://en.wikipedia.org/wiki/Proof_calculus
[39] https://en.wikipedia.org/wiki/Functional_programming_language
[40] https://en.wikipedia.org/wiki/Martin-L%C3%B6f
[41] https://en.wikipedia.org/wiki/Intuitionistic_type_theory
[42] https://en.wikipedia.org/wiki/Thierry_Coquand
[43] https://en.wikipedia.org/wiki/Calculus_of_Constructions
[44] https://en.wikipedia.org/wiki/Type_theory
[45] https://en.wikipedia.org/wiki/Typed_lambda_calculus
[46] https://en.wikipedia.org/wiki/Coq
[47] https://en.wikipedia.org/wiki/Program_correctness
[48] https://en.wikipedia.org/wiki/Proof-carrying_code
[49] https://en.wikipedia.org/wiki/Programming_language
[50] https://en.wikipedia.org/wiki/Classical_logic
[51] https://en.wikipedia.org/wiki/Continuation
[52] https://en.wikipedia.org/wiki/Sequent_calculus
[53] https://en.wikipedia.org/wiki/Evaluation_strategy
[54] https://en.wikipedia.org/wiki/Unifying_theories_in_mathematics
[55] https://en.wikipedia.org/wiki/Sequent_calculus
[56] https://en.wikipedia.org/wiki/Proof_net

lus of structures[57], etc. If one admits the Curry–Howard correspondence as the general principle that any proof system hides a model of computation, a theory of the underlying untyped computational structure of these kinds of proof system should be possible. Then, a natural question is whether something mathematically interesting can be said about these underlying computational calculi.

Conversely, combinatory logic[58] and simply typed lambda calculus[59] are not the only models of computation[60], either. Girard's linear logic[61] was developed from the fine analysis of the use of resources in some models of lambda calculus; is there typed version of Turing's machine[62] that would behave as a proof system? Typed assembly languages[63] are such an instance of ”low-level” models of computation that carry types.

Because of the possibility of writing non-terminating programs, Turing-complete[64] models of computation (such as languages with arbitrary recursive functions[65]) must be interpreted with care, as naive application of the correspondence leads to an inconsistent logic. The best way of dealing with arbitrary computation from a logical point of view is still an actively debated research question, but one popular approach is based on using monads[66] to segregate provably terminating from potentially non-terminating code (an approach that also generalizes to much richer models of computation,[6] and is itself related to modal logic by a natural extension of the Curry–Howard isomorphism[ext 1]). A more radical approach, advocated by total functional programming[67], is to eliminate unrestricted recursion (and forgo Turing completeness[68], although still retaining high computational complexity), using more controlled corecursion[69] wherever non-terminating behavior is actually desired.

## 1.2 General formulation

In its more general formulation, the Curry–Howard correspondence is a correspondence between formal proof calculi[70] and type systems[71] for models of computation[72]. In particular, it splits into two correspondences. One at the level of formulas[73] and types[74] that is independent of which particular proof system or model of computation is considered, and one

[57] https://en.wikipedia.org/wiki/Calculus_of_structures
[58] https://en.wikipedia.org/wiki/Combinatory_logic
[59] https://en.wikipedia.org/wiki/Simply_typed_lambda_calculus
[60] https://en.wikipedia.org/wiki/Models_of_computation
[61] https://en.wikipedia.org/wiki/Linear_logic
[62] https://en.wikipedia.org/wiki/Turing_machine
[63] https://en.wikipedia.org/wiki/Typed_assembly_language
[64] https://en.wikipedia.org/wiki/Turing-complete
[65] https://en.wikipedia.org/wiki/Recursion_(computer_science)
[66] https://en.wikipedia.org/wiki/Monad_(functional_programming)
[67] https://en.wikipedia.org/wiki/Total_functional_programming
[68] https://en.wikipedia.org/wiki/Turing_completeness
[69] https://en.wikipedia.org/wiki/Corecursion
[70] https://en.wikipedia.org/wiki/Proof_calculus
[71] https://en.wikipedia.org/wiki/Type_systems
[72] https://en.wikipedia.org/wiki/Model_of_computation
[73] https://en.wikipedia.org/wiki/Formula_(mathematical_logic)
[74] https://en.wikipedia.org/wiki/Data_type

at the level of proofs[75] and programs[76] which, this time, is specific to the particular choice of proof system and model of computation considered.

At the level of formulas and types, the correspondence says that implication behaves the same as a function type, conjunction as a ”product” type (this may be called a tuple, a struct, a list, or some other term depending on the language), disjunction as a sum type (this type may be called a union), the false formula as the empty type and the true formula as the singleton type (whose sole member is the null object). Quantifiers correspond to dependent[77] function space or products (as appropriate). This is summarized in the following table:

| Logic side | Programming side |
|---|---|
| universal quantification[78] | generalised product type[79] (Π type) |
| existential quantification[80] | generalised sum type[81] (Σ type) |
| implication[82] | function type[83] |
| conjunction[84] | product type[85] |
| disjunction[86] | sum type[87] |
| true formula | unit type[88] |
| false formula | bottom type[89] |

At the level of proof systems and models of computations, the correspondence mainly shows the identity of structure, first, between some particular formulations of systems known as Hilbert-style deduction system[90] and combinatory logic[91], and, secondly, between some particular formulations of systems known as natural deduction[92] and lambda calculus[93].

| Logic side | Programming side |
|---|---|
| Hilbert-style deduction system[94] | type system for combinatory logic[95] |
| natural deduction[96] | type system for lambda calculus[97] |

Between the natural deduction system and the lambda calculus there are the following correspondences:

| Logic side | Programming side |
|---|---|

[75] https://en.wikipedia.org/wiki/Mathematical_proof
[76] https://en.wikipedia.org/wiki/Computer_program
[77] https://en.wikipedia.org/wiki/Dependent_type
[78] https://en.wikipedia.org/wiki/Universal_quantification
[79] https://en.wikipedia.org/wiki/Dependent_type#%CE%A0_type
[80] https://en.wikipedia.org/wiki/Existential_quantification
[81] https://en.wikipedia.org/wiki/Dependent_type#%CE%A3_type
[82] https://en.wikipedia.org/wiki/Logical_implication
[83] https://en.wikipedia.org/wiki/Function_type
[84] https://en.wikipedia.org/wiki/Logical_conjunction
[85] https://en.wikipedia.org/wiki/Product_type
[86] https://en.wikipedia.org/wiki/Logical_disjunction
[87] https://en.wikipedia.org/wiki/Sum_type
[88] https://en.wikipedia.org/wiki/Unit_type
[89] https://en.wikipedia.org/wiki/Bottom_type
[90] https://en.wikipedia.org/wiki/Hilbert-style_deduction_system
[91] https://en.wikipedia.org/wiki/Combinatory_logic
[92] https://en.wikipedia.org/wiki/Natural_deduction
[93] https://en.wikipedia.org/wiki/Lambda_calculus
[94] https://en.wikipedia.org/wiki/Hilbert-style_deduction_system
[95] https://en.wikipedia.org/wiki/Combinatory_logic
[96] https://en.wikipedia.org/wiki/Natural_deduction
[97] https://en.wikipedia.org/wiki/Lambda_calculus

| Logic side | Programming side |
|---|---|
| hypotheses[98] | free variables |
| implication elimination[99] (*modus ponens*) | application[100] |
| implication introduction[101] | abstraction |

## 1.3 Corresponding systems

### 1.3.1 Hilbert-style deduction systems and combinatory logic

It was at the beginning a simple remark in Curry and Feys's 1958 book on combinatory logic: the simplest types for the basic combinators K and S of combinatory logic[102] surprisingly corresponded to the respective axiom schemes[103] $\alpha \to (\beta \to \alpha)$ and $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$ used in Hilbert-style deduction systems[104]. For this reason, these schemes are now often called axioms K and S. Examples of programs seen as proofs in a Hilbert-style logic are given below[105].

If one restricts to the implicational intuitionistic fragment, a simple way to formalize logic in Hilbert's style is as follows. Let Γ be a finite collection of formulas, considered as hypotheses. Then δ is *derivable* from Γ, denoted Γ ⊢δ, in the following cases:

- δ is an hypothesis, i.e. it is a formula of Γ,
- δ is an instance of an axiom scheme; i.e., under the most common axiom system:
  - δ has the form $\alpha \to (\beta \to \alpha)$, or
  - δ has the form $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$,
- δ follows by deduction, i.e., for some $\alpha$, both $\alpha \to \delta$ and $\alpha$ are already derivable from Γ (this is the rule of modus ponens[106])

This can be formalized using inference rules[107], as in the left column of the following table.

Typed combinatory logic can be formulated using a similar syntax: let Γ be a finite collection of variables, annotated with their types. A term T (also annotated with its type) will depend on these variables [Γ ⊢T:$\delta$] when:

- T is one of the variables in Γ,
- T is a basic combinator; i.e., under the most common combinator basis:
  - T is K:$\alpha \to (\beta \to \alpha)$ [where $\alpha$ and $\beta$ denote the types of its arguments], or
  - T is S:$(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$,
- T is the composition of two subterms which depend on the variables in Γ.

The generation rules defined here are given in the right-column below. Curry's remark simply states that both columns are in one-to-one correspondence. The restriction of the

[98] https://en.wikipedia.org/wiki/Hypotheses
[99] https://en.wikipedia.org/wiki/Implication_elimination
[100] https://en.wikipedia.org/wiki/Apply
[101] https://en.wikipedia.org/wiki/Implication_introduction
[102] https://en.wikipedia.org/wiki/Combinatory_logic
[103] https://en.wikipedia.org/wiki/Axiom_scheme
[104] https://en.wikipedia.org/wiki/Hilbert-style_deduction_system
[105] #Examples
[106] https://en.wikipedia.org/wiki/Modus_ponens
[107] https://en.wikipedia.org/wiki/Inference_rules

correspondence to intuitionistic logic[108] means that some classical[109] tautologies[110], such as Peirce's law[111] $((\alpha \to \beta) \to \alpha) \to \alpha$, are excluded from the correspondence.

| **Hilbert-style intuitionistic implicational logic** | **Simply typed combinatory logic** |
|---|---|
| $\frac{\alpha \in \Gamma}{\Gamma \vdash \alpha}$ Assum | $\frac{x : \alpha \in \Gamma}{\Gamma \vdash x : \alpha}$ |
| $\frac{}{\Gamma \vdash \alpha \to (\beta \to \alpha)}\mathrm{Ax}_K$ | $\frac{}{\Gamma \vdash K : \alpha \to (\beta \to \alpha)}$ |
| $\frac{}{\Gamma \vdash (\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))}\mathrm{Ax}_S$ | $\frac{}{\Gamma \vdash S : (\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))}$ |
| $\frac{\Gamma \vdash \alpha \to \beta \qquad \Gamma \vdash \alpha}{\Gamma \vdash \beta}$ Modus Ponens | $\frac{\Gamma \vdash E_1 : \alpha \to \beta \qquad \Gamma \vdash E_2 : \alpha}{\Gamma \vdash E_1 E_2 : \beta}$ |

Seen at a more abstract level, the correspondence can be restated as shown in the following table. Especially, the deduction theorem[112] specific to Hilbert-style logic matches the process of abstraction elimination[113] of combinatory logic.

| **Logic side** | **Programming side** |
|---|---|
| assumption | variable |
| axioms | combinators |
| modus ponens | application |
| deduction theorem[114] | abstraction elimination[115] |

Thanks to the correspondence, results from combinatory logic can be transferred to Hilbert-style logic and vice versa. For instance, the notion of reduction[116] of terms in combinatory logic can be transferred to Hilbert-style logic and it provides a way to canonically transform proofs into other proofs of the same statement. One can also transfer the notion of normal terms to a notion of normal proofs, expressing that the hypotheses of the axioms never need to be all detached (since otherwise a simplification can happen).

Conversely, the non provability in intuitionistic logic of Peirce's law[117] can be transferred back to combinatory logic: there is no typed term of combinatory logic that is typable with type

$$((\alpha \to \beta) \to \alpha) \to \alpha.$$

Results on the completeness of some sets of combinators or axioms can also be transferred. For instance, the fact that the combinator **X** constitutes a one-point basis[118] of (extensional) combinatory logic implies that the single axiom scheme

$$(((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))) \to ((\delta \to (\varepsilon \to \delta)) \to \zeta)) \to \zeta,$$

[108] https://en.wikipedia.org/wiki/Intuitionistic_logic

[109] https://en.wikipedia.org/wiki/Classical_logic

[110] https://en.wikipedia.org/wiki/Tautology_(logic)

[111] https://en.wikipedia.org/wiki/Peirce%27s_law

[112] https://en.wikipedia.org/wiki/Deduction_theorem

[113] https://en.wikipedia.org/wiki/Combinatory_logic#Conversion_of_a_lambda_term_to_an_equivalent_combinatorial_term

[114] https://en.wikipedia.org/wiki/Deduction_theorem

[115] https://en.wikipedia.org/wiki/Combinatory_logic#Conversion_of_a_lambda_term_to_an_equivalent_combinatorial_term

[116] https://en.wikipedia.org/wiki/Combinatory_logic#Reduction_in_combinatory_logic

[117] https://en.wikipedia.org/wiki/Peirce%27s_law

[118] https://en.wikipedia.org/wiki/Combinatory_logic#One-point_basis

which is the principal type[119] of **X**, is an adequate replacement to the combination of the axiom schemes

$\alpha \to (\beta \to \alpha)$ and

$(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$.

### 1.3.2 Natural deduction and lambda calculus

After Curry[120] emphasized the syntactic correspondence between Hilbert-style deduction[121] and combinatory logic[122], Howard[123] made explicit in 1969 a syntactic analogy between the programs of simply typed lambda calculus[124] and the proofs of natural deduction[125]. Below, the left-hand side formalizes intuitionistic implicational natural deduction as a calculus of sequents[126] (the use of sequents is standard in discussions of the Curry–Howard isomorphism as it allows the deduction rules to be stated more cleanly) with implicit weakening and the right-hand side shows the typing rules of lambda calculus[127]. In the left-hand side, $\Gamma$, $\Gamma_1$ and $\Gamma_2$ denote ordered sequences of formulas while in the right-hand side, they denote sequences of named (i.e., typed) formulas with all names different.

| **Intuitionistic implicational natural deduction** | **Lambda calculus type assignment rules** |
|---|---|
| $\dfrac{}{\Gamma_1, \alpha, \Gamma_2 \vdash \alpha}\,\text{Ax}$ | $\dfrac{}{\Gamma_1, x : \alpha, \Gamma_2 \vdash x : \alpha}$ |
| $\dfrac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \to \beta} \to I$ | $\dfrac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash (\lambda x : \alpha.\ t) : \alpha \to \beta}$ |
| $\dfrac{\Gamma \vdash \alpha \to \beta \qquad \Gamma \vdash \alpha}{\Gamma \vdash \beta} \to E$ | $\dfrac{\Gamma \vdash t : \alpha \to \beta \qquad \Gamma \vdash u : \alpha}{\Gamma \vdash tu : \beta}$ |

To paraphrase the correspondence, proving $\Gamma \vdash \alpha$ means having a program that, given values with the types listed in $\Gamma$, manufactures an object of type $\alpha$. An axiom corresponds to the introduction of a new variable with a new, unconstrained type, the $\to I$ rule corresponds to function abstraction and the $\to E$ rule corresponds to function application. Observe that the correspondence is not exact if the context $\Gamma$ is taken to be a set of formulas as, e.g., the λ-terms $\lambda x.\lambda y.x$ and $\lambda x.\lambda y.y$ of type $\alpha \to \alpha \to \alpha$ would not be distinguished in the correspondence. Examples are given below[128].

Howard showed that the correspondence extends to other connectives of the logic and other constructions of simply typed lambda calculus. Seen at an abstract level, the correspondence can then be summarized as shown in the following table. Especially, it also shows

[119] https://en.wikipedia.org/wiki/Principal_type
[120] https://en.wikipedia.org/wiki/Haskell_Curry
[121] https://en.wikipedia.org/wiki/Hilbert-style_deduction_system
[122] https://en.wikipedia.org/wiki/Combinatory_logic
[123] https://en.wikipedia.org/wiki/William_Alvin_Howard
[124] https://en.wikipedia.org/wiki/Simply_typed_lambda_calculus
[125] https://en.wikipedia.org/wiki/Natural_deduction
[126] https://en.wikipedia.org/wiki/Sequent
[127] https://en.wikipedia.org/wiki/Lambda_calculus
[128] #Examples

that the notion of normal forms in lambda calculus[129] matches Prawitz[130]'s notion of normal deduction in natural deduction[131], from which it follows that the algorithms for the type inhabitation problem[132] can be turned into algorithms for deciding intuitionistic[133] provability.

| **Logic side** | **Programming side** |
|---|---|
| axiom | variable |
| introduction rule | constructor |
| elimination rule | destructor |
| normal deduction | normal form |
| normalisation of deductions | weak normalisation[134] |
| provability | type inhabitation problem[135] |
| intuitionistic tautology | inhabited type |

Howard's correspondence naturally extends to other extensions of natural deduction[136] and simply typed lambda calculus[137]. Here is a non-exhaustive list:

- Girard-Reynolds System F[138] as a common language for both second-order propositional logic and polymorphic lambda calculus,
- higher-order logic[139] and Girard's System $F_\omega$[140]
- inductive types as algebraic data type[141]
- necessity □ in modal logic[142] and staged computation[ext 2]
- possibility ◇ in modal logic[143] and monadic types for effects[ext 1]
- The $\lambda_I$ calculus corresponds to relevant logic[144].[7]
- The local truth (∇) modality in Grothendieck topology[145] or the equivalent ”lax” modality (◯) of Benton, Bierman, and de Paiva (1998) correspond to CL-logic describing ”computation types”.[8]

[129] https://en.wikipedia.org/wiki/Lambda_calculus
[130] https://en.wikipedia.org/wiki/Dag_Prawitz
[131] https://en.wikipedia.org/wiki/Natural_deduction
[132] https://en.wikipedia.org/wiki/Type_inhabitation_problem
[133] https://en.wikipedia.org/wiki/Intuitionistic
[134] https://en.wikipedia.org/wiki/Normalization_property_(lambda-calculus)
[135] https://en.wikipedia.org/wiki/Type_inhabitation_problem
[136] https://en.wikipedia.org/wiki/Natural_deduction
[137] https://en.wikipedia.org/wiki/Simply_typed_lambda_calculus
[138] https://en.wikipedia.org/wiki/System_F
[139] https://en.wikipedia.org/wiki/Higher-order_logic
[140] https://en.wikipedia.org/wiki/System_F
[141] https://en.wikipedia.org/wiki/Algebraic_data_type
[142] https://en.wikipedia.org/wiki/Modal_logic
[143] https://en.wikipedia.org/wiki/Modal_logic
[144] https://en.wikipedia.org/wiki/Relevant_logic
[145] https://en.wikipedia.org/wiki/Grothendieck_topology

### 1.3.3 Classical logic and control operators

At the time of Curry, and also at the time of Howard, the proofs-as-programs correspondence concerned only intuitionistic logic[146], i.e. a logic in which, in particular, Peirce's law[147] was *not* deducible. The extension of the correspondence to Peirce's law and hence to classical logic[148] became clear from the work of Griffin on typing operators that capture the evaluation context of a given program execution so that this evaluation context can be later on reinstalled. The basic Curry–Howard-style correspondence for classical logic is given below. Note the correspondence between the double-negation translation[149] used to map classical proofs to intuitionistic logic and the continuation-passing-style[150] translation used to map lambda terms involving control to pure lambda terms. More particularly, call-by-name continuation-passing-style translations relates to Kolmogorov[151]'s double negation translation and call-by-value continuation-passing-style translations relates to a kind of double-negation translation due to Kuroda.

| **Logic side** | **Programming side** |
|---|---|
| Peirce's law[152]: $((\alpha \to \beta) \to \alpha) \to \alpha$ | call-with-current-continuation[153] |
| double-negation translation[154] | continuation-passing-style translation[155] |

A finer Curry–Howard correspondence exists for classical logic if one defines classical logic not by adding an axiom such as Peirce's law[156], but by allowing several conclusions in sequents. In the case of classical natural deduction, there exists a proofs-as-programs correspondence with the typed programs of Parigot's λμ-calculus[157].

### 1.3.4 Sequent calculus

A proofs-as-programs correspondence can be settled for the formalism known as Gentzen[158]'s sequent calculus[159] but it is not a correspondence with a well-defined pre-existing model of computation as it was for Hilbert-style and natural deductions.

Sequent calculus is characterized by the presence of left introduction rules, right introduction rule and a cut rule that can be eliminated. The structure of sequent calculus relates to a calculus whose structure is close to the one of some abstract machines[160]. The informal correspondence is as follows:

[146] https://en.wikipedia.org/wiki/Intuitionistic_logic
[147] https://en.wikipedia.org/wiki/Peirce%27s_law
[148] https://en.wikipedia.org/wiki/Classical_logic
[149] https://en.wikipedia.org/wiki/Double-negation_translation
[150] https://en.wikipedia.org/wiki/Continuation-passing_style
[151] https://en.wikipedia.org/wiki/Kolmogorov
[152] https://en.wikipedia.org/wiki/Peirce%27s_law
[153] https://en.wikipedia.org/wiki/Call-with-current-continuation
[154] https://en.wikipedia.org/wiki/Double-negation_translation
[155] https://en.wikipedia.org/wiki/Continuation-passing_style
[156] https://en.wikipedia.org/wiki/Peirce%27s_law
[157] https://en.wikipedia.org/wiki/Lambda-mu_calculus
[158] https://en.wikipedia.org/wiki/Gentzen
[159] https://en.wikipedia.org/wiki/Sequent_calculus
[160] https://en.wikipedia.org/wiki/Abstract_machine

| Logic side | Programming side |
|---|---|
| cut elimination | reduction in a form of abstract machine |
| right introduction rules | constructors of code |
| left introduction rules | constructors of evaluation stacks |
| priority to right-hand side in cut-elimination | call-by-name[161] reduction |
| priority to left-hand side in cut-elimination | call-by-value[162] reduction |

## 1.4 Related proofs-as-programs correspondences

### 1.4.1 The role of de Bruijn

N. G. de Bruijn[163] used the lambda notation for representing proofs of the theorem checker Automath[164], and represented propositions as ”categories” of their proofs. It was in the late 1960s at the same period of time Howard wrote his manuscript; de Bruijn was likely unaware of Howard's work, and stated the correspondence independently (Sørensen & Urzyczyn [1998] 2006, pp 98–99). Some researchers tend to use the term Curry–Howard–de Bruijn correspondence in place of Curry–Howard correspondence.

### 1.4.2 BHK interpretation

The BHK interpretation[165] interprets intuitionistic proofs as functions but it does not specify the class of functions relevant for the interpretation. If one takes lambda calculus for this class of function, then the BHK interpretation[166] tells the same as Howard's correspondence between natural deduction and lambda calculus.

### 1.4.3 Realizability

Kleene[167]'s recursive realizability[168] splits proofs of intuitionistic arithmetic into the pair of a recursive function and of a proof of a formula expressing that the recursive function ”realizes”, i.e. correctly instantiates the disjunctions and existential quantifiers of the initial formula so that the formula gets true.

Kreisel[169]'s modified realizability applies to intuitionistic higher-order predicate logic and shows that the simply typed lambda term[170] inductively extracted from the proof realizes the initial formula. In the case of propositional logic, it coincides with Howard's statement:

[161] https://en.wikipedia.org/wiki/Call-by-name

[162] https://en.wikipedia.org/wiki/Call-by-value

[163] https://en.wikipedia.org/wiki/Nicolaas_Govert_de_Bruijn

[164] https://en.wikipedia.org/wiki/Automath

[165] https://en.wikipedia.org/wiki/BHK_interpretation

[166] https://en.wikipedia.org/wiki/BHK_interpretation

[167] https://en.wikipedia.org/wiki/Stephen_Cole_Kleene

[168] https://en.wikipedia.org/wiki/Realizability

[169] https://en.wikipedia.org/wiki/Georg_Kreisel

[170] https://en.wikipedia.org/wiki/Simply_typed_lambda_calculus

the extracted lambda term is the proof itself (seen as an untyped lambda term) and the realizability statement is a paraphrase of the fact that the extracted lambda term has the type that the formula means (seen as a type).

Gödel[171]'s dialectica interpretation[172] realizes (an extension of) intuitionistic arithmetic with computable functions. The connection with lambda calculus is unclear, even in the case of natural deduction.

### 1.4.4 Curry–Howard–Lambek correspondence

Joachim Lambek[173] showed in the early 1970s that the proofs of intuitionistic propositional logic and the combinators of typed combinatory logic[174] share a common equational theory which is the one of cartesian closed categories[175]. The expression Curry–Howard–Lambek correspondence is now used by some people to refer to the three way isomorphism between intuitionistic logic, typed lambda calculus and cartesian closed categories, with objects being interpreted as types or propositions and morphisms as terms or proofs. The correspondence works at the equational level and is not the expression of a syntactic identity of structures as it is the case for each of Curry's and Howard's correspondences: i.e. the structure of a well-defined morphism in a cartesian-closed category is not comparable to the structure of a proof of the corresponding judgment in either Hilbert-style logic or natural deduction. To clarify this distinction, the underlying syntactic structure of cartesian closed categories is rephrased below.

Objects (types) are defined by

- $\top$ is an object
- if α and β are objects then $\alpha \times \beta$ and $\alpha \rightarrow \beta$ are objects.

Morphisms (terms) are defined by

- $id$, $\star$, eval, $\pi_1$ and $\pi_2$ are morphisms
- if t is a morphism, λt is a morphism
- if t and u are morphisms, $(t,u)$ and $u \circ t$ are morphisms.

Well-defined morphisms (typed terms) are defined by the following typing rules[176] (in which the usual categorical morphism notation $f : \alpha \rightarrow \beta$ is replaced with sequent calculus[177] notation $f : - \ \ \alpha \ \vdash \ \beta$).

Identity:

$$\frac{}{id : - \ \ \alpha \ \vdash \ \alpha}$$

Composition:

$$\frac{t : - \ \ \alpha \ \vdash \ \beta \qquad u : - \ \ \beta \ \vdash \ \gamma}{u \circ t : - \ \alpha \ \vdash \ \gamma}$$

[171] https://en.wikipedia.org/wiki/Kurt_G%C3%B6del
[172] https://en.wikipedia.org/wiki/Dialectica_interpretation
[173] https://en.wikipedia.org/wiki/Joachim_Lambek
[174] https://en.wikipedia.org/wiki/Combinatory_logic
[175] https://en.wikipedia.org/wiki/Cartesian_closed_categories
[176] https://en.wikipedia.org/wiki/Type_rule
[177] https://en.wikipedia.org/wiki/Sequent_calculus

Unit type[178] (terminal object[179]):

$$\frac{}{\star :- \ \alpha \vdash \top}$$

Cartesian product:

$$\frac{t :- \ \alpha \vdash \beta \qquad u :- \ \alpha \vdash \gamma}{(t,u) :- \ \alpha \vdash \beta \times \gamma}$$

Left and right projection:

$$\frac{}{\pi_1 :- \ \alpha \times \beta \vdash \alpha} \qquad \frac{}{\pi_2 :- \ \alpha \times \beta \vdash \beta}$$

Currying[180]:

$$\frac{t :- \ \alpha \times \beta \vdash \gamma}{\lambda t :- \ \alpha \vdash \beta \to \gamma}$$

Application[181]:

$$\frac{}{eval :- \ (\alpha \to \beta) \times \alpha \vdash \beta}$$

Finally, the equations of the category are

- $id \circ t = t$
- $t \circ id = t$
- $(v \circ u) \circ t = v \circ (u \circ t)$
- $\star = id$ (if well-typed)
- $\star \circ u = \star$
- $\pi_1 \circ (t,u) = t$
- $\pi_2 \circ (t,u) = u$
- $(\pi_1, \pi_2) = id$
- $(t_1, t_2) \circ u = (t_1 \circ u, t_2 \circ u)$
- $eval \circ (\lambda t \circ \pi_1, \pi_2) = t$
- $\lambda eval = id$
- $\lambda t \circ u = \lambda(t \circ (u \circ \pi_1, \pi_2))$

These equations imply the following $\eta$-laws:

- $(\pi_1 \circ t, \pi_2 \circ t) = t$
- $\lambda(eval \circ (t \circ \pi_1, \pi_2)) = t$

Now, there exists t such that $t :- \ \alpha_1 \times \ldots \times \alpha_n \vdash \beta$ iff $\alpha_1, \ldots, \alpha_n \vdash \beta$ is provable in implicational intuitionistic logic,.

## 1.5 Examples

Thanks to the Curry–Howard correspondence, a typed expression whose type corresponds to a logical formula is analogous to a proof of that formula. Here are examples.

[178] https://en.wikipedia.org/wiki/Unit_type

[179] https://en.wikipedia.org/wiki/Terminal_object

[180] https://en.wikipedia.org/wiki/Currying

[181] https://en.wikipedia.org/wiki/Apply

### 1.5.1 The identity combinator seen as a proof of $\alpha \to \alpha$ in Hilbert-style logic

As an example, consider a proof of the theorem $\alpha \to \alpha$. In lambda calculus[182], this is the type of the identity function $\mathbf{I} = \lambda x.x$ and in combinatory logic, the identity function is obtained by applying $\mathbf{S} = \lambda fgx.fx(gx)$ twice to $\mathbf{K} = \lambda xy.x$. That is, $\mathbf{I} = ((\mathbf{S}\ \mathbf{K})\ \mathbf{K})$. As a description of a proof, this says that the following steps can be used to prove $\alpha \to \alpha$:

- instantiate the second axiom scheme with the formulas $\alpha$, $\beta \to \alpha$ and $\alpha$ to obtain a proof of $(\alpha \to ((\beta \to \alpha) \to \alpha)) \to ((\alpha \to (\beta \to \alpha)) \to (\alpha \to \alpha))$,
- instantiate the first axiom scheme once with $\alpha$ and $\beta \to \alpha$ to obtain a proof of $\alpha \to ((\beta \to \alpha) \to \alpha)$,
- instantiate the first axiom scheme a second time with α and β to obtain a proof of $\alpha \to (\beta \to \alpha)$,
- apply modus ponens twice to obtain a proof of $\alpha \to \alpha$

In general, the procedure is that whenever the program contains an application of the form $(P\ Q)$, these steps should be followed:

1. First prove theorems corresponding to the types of $P$ and $Q$.
2. Since $P$ is being applied to $Q$, the type of $P$ must have the form $\alpha \to \beta$ and the type of $Q$ must have the form $\alpha$ for some $\alpha$ and $\beta$. Therefore, it is possible to detach the conclusion, β, via the modus ponens rule.

### 1.5.2 The composition combinator seen as a proof of $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$ in Hilbert-style logic

As a more complicated example, let's look at the theorem that corresponds to the **B** function. The type of **B** is $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$. **B** is equivalent to (**S** (**K** **S**) **K**). This is our roadmap for the proof of the theorem $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$.

The first step is to construct (**K** **S**). To make the antecedent of the **K** axiom look like the **S** axiom, set $\alpha$ equal to $(\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$, and $\beta$ equal to $\delta$ (to avoid variable collisions):

$\mathbf{K} : \alpha \to \beta \to \alpha$

$\mathbf{K}[\alpha = (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma, \beta = \delta] : ((\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma) \to \delta \to (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$

Since the antecedent here is just **S**, the consequent can be detached using Modus Ponens:

$\mathbf{K}\ \mathbf{S} : \delta \to (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$

This is the theorem that corresponds to the type of (**K** **S**). Now apply **S** to this expression. Taking **S** as follows

$\mathbf{S} : (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$,

put $\alpha = \delta$, $\beta = \alpha \to \beta \to \gamma$, and $\gamma = (\alpha \to \beta) \to \alpha \to \gamma$, yielding

[182] https://en.wikipedia.org/wiki/Lambda_calculus

$\mathbf{S}[\alpha = \delta, \beta = \alpha \to \beta \to \gamma, \gamma = (\alpha \to \beta) \to \alpha \to \gamma] : (\delta \to (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma) \to (\delta \to (\alpha \to \beta \to \gamma)) \to \delta \to (\alpha \to \beta) \to \alpha \to \gamma$

and then detach the consequent:

$\mathbf{S}\ (\mathbf{K}\ \mathbf{S}) : (\delta \to \alpha \to \beta \to \gamma) \to \delta \to (\alpha \to \beta) \to \alpha \to \gamma$

This is the formula for the type of (**S** (**K** **S**)). A special case of this theorem has $\delta = (\beta \to \gamma)$:

$\mathbf{S}\ (\mathbf{K}\ \mathbf{S})[\delta = \beta \to \gamma] : ((\beta \to \gamma) \to \alpha \to \beta \to \gamma) \to (\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$

This last formula must be applied to **K**. Specialize **K** again, this time by replacing $\alpha$ with $(\beta \to \gamma)$ and $\beta$ with $\alpha$:

$\mathbf{K} : \alpha \to \beta \to \alpha$

$\mathbf{K}[\alpha = \beta \to \gamma, \beta = \alpha] : (\beta \to \gamma) \to \alpha \to (\beta \to \gamma)$

This is the same as the antecedent of the prior formula so, detaching the consequent:

$\mathbf{S}\ (\mathbf{K}\ \mathbf{S})\ \mathbf{K} : (\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$

Switching the names of the variables $\alpha$ and $\gamma$ gives us

$(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$

which was what remained to prove.

### 1.5.3 The normal proof of $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$ in natural deduction seen as a λ-term

The diagram below gives proof of $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$ in natural deduction and shows how it can be interpreted as the λ-expression $\lambda a.\lambda b.\lambda g.(a\ (b\ g))$ of type $(\beta \to \alpha) \to (\gamma \to \beta) \to \gamma \to \alpha$.

a:β → α, b:γ → β, g:γ ⊢ b : γ → β    a:β →
α, b:γ → β, g:γ ⊢ g : γ

a:β → α, b:γ → β, g:γ ⊢ a : β → α    a:β → α, b:γ → β, g:γ ⊢ b g : β

a:β → α, b:γ → β, g:γ ⊢ a (b g) : α

a:β → α, b:γ → β ⊢ λ g. a (b g) : γ → α

a:β → α ⊢ λ b. λ g. a (b g) : (γ → β) -> γ → α

⊢ λ a. λ b. λ g. a (b g) : (β → α) -> (γ → β) ->
γ → α

## 1.6 Other applications

Recently, the isomorphism has been proposed as a way to define search space partition in genetic programming[183].[9] The method indexes sets of genotypes (the program trees evolved by the GP system) by their Curry–Howard isomorphic proof (referred to as a species).

As noted by INRIA[184] research director Bernard Lang,[10] the Curry-Howard correspondence constitutes an argument against the patentability of software: since algorithms are mathematical proofs, patentability of the former would imply patentability of the latter. A theorem could be private property; a mathematician would have to pay for using it, and to trust the company that sells it but keeps its proof secret and rejects responsibility for any errors.

## 1.7 Generalizations

The correspondences listed here go much farther and deeper. For example, cartesian closed categories are generalized by closed monoidal categories[185]. The internal language[186] of these categories is the linear type system[187] (corresponding to linear logic[188]), which generalizes simply-typed lambda calculus as the internal language of cartesian closed categories. Moreover, these can be shown to correspond to cobordisms[189],[11] which play a vital role in string theory[190].

An extended set of equivalences is also explored in homotopy type theory[191], which became a very active area of research around 2013 and as of 2018[update][192] still is.[12] Here, type theory[193] is extended by the univalence axiom[194] ("equivalence is equivalent to equality") which permits homotopy type theory to be used as a foundation for all of mathematics (including set theory[195] and classical logic, providing new ways to discuss the axiom of choice[196] and many other things). That is, the Curry–Howard correspondence that proofs are elements of inhabited types is generalized to the notion of homotopic equivalence[197] of proofs (as paths in space, the identity type[198] or equality type[199] of type theory being interpreted as a path).[13]

[183] https://en.wikipedia.org/wiki/Genetic_programming
[184] https://en.wikipedia.org/wiki/INRIA
[185] https://en.wikipedia.org/wiki/Closed_monoidal_category
[186] https://en.wikipedia.org/wiki/Internal_language
[187] https://en.wikipedia.org/wiki/Linear_type_system
[188] https://en.wikipedia.org/wiki/Linear_logic
[189] https://en.wikipedia.org/wiki/Cobordism
[190] https://en.wikipedia.org/wiki/String_theory
[191] https://en.wikipedia.org/wiki/Homotopy_type_theory
[193] https://en.wikipedia.org/wiki/Type_theory
[194] https://en.wikipedia.org/wiki/Univalence_axiom
[195] https://en.wikipedia.org/wiki/Set_theory
[196] https://en.wikipedia.org/wiki/Axiom_of_choice
[197] https://en.wikipedia.org/wiki/Homotopy
[198] https://en.wikipedia.org/wiki/Identity_type
[199] https://en.wikipedia.org/wiki/Intuitionistic_type_theory#Connectives_of_type_theory

## 1.8 References

This article includes a list of general references[200], but **it lacks sufficient corresponding inline citations[201]**. Please help to improve[202] this article by introducing[203] more precise citations. *(April 2010)(Learn how and when to remove this template message[204])*

1. The correspondence was first made explicit in Howard 1980[205]. See, for example section 4.6, p.53 Gert Smolka and Jan Schwinghammer (2007-8), Lecture Notes in Semantics[206]
2. The Brouwer–Heyting–Kolmogorov interpretation is also called the 'proof interpretation': p. 161 of Juliette Kennedy, Roman Kossak, eds. 2011. *Set Theory, Arithmetic, and Foundations of Mathematics: Theorems, Philosophies*[207] ISBN[208] 978-1-107-00804-5[209]
3. Curry 1934[210].
4. Curry & Feys 1958[211].
5. Howard 1980[212].
6. Moggi, Eugenio (1991), "Notions of Computation and Monads"[213] (PDF), *Information and Computation*, **93** (1): 55–92, doi[214]:10.1016/0890-5401(91)90052-4[215]
7. Sørenson, Morten; Urzyczyn, Paweł (1998), *Lectures on the Curry-Howard Isomorphism*, CiteSeerX[216] 10.1.1.17.7385[217]
8. Goldblatt, "7.6 Grothendieck Topology as Intuitionistic Modality"[218] (PDF), *Mathematical Modal Logic: A Model of its Evolution*, pp. 76–81. The "lax" modality referred to is from
   Benton; Bierman; de Paiva (1998), "Computational types from a log-

[200] https://en.wikipedia.org/wiki/Wikipedia:Citing_sources

[201] https://en.wikipedia.org/wiki/Wikipedia:Citing_sources#Inline_citations

[202] https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Fact_and_Reference_Check

[203] https://en.wikipedia.org/wiki/Wikipedia:When_to_cite

[204] https://en.wikipedia.org/wiki/Help:Maintenance_template_removal

[205] #CITEREFHoward1980

[206] http://www.ps.uni-saarland.de/courses/sem-ws07/notes/0.pdf

[207] https://books.google.com/books?id=x1aPcJnz4iYC&pg=PA161&dq=Brouwer%E2%80%93Heyting%E2%80%93Kolmogorov+interpretation&hl=en&sa=X&ei=CbdRVPvyB6-_iQL--YG4AQ&ved=0CCcQ6AEwAQ#v=onepage&q=Brouwer%E2%80%93Heyting%E2%80%93Kolmogorov%20interpretation&f=false

[208] https://en.wikipedia.org/wiki/ISBN_(identifier)

[209] https://en.wikipedia.org/wiki/Special:BookSources/978-1-107-00804-5

[210] #CITEREFCurry1934

[211] #CITEREFCurryFeys1958

[212] #CITEREFHoward1980

[213] http://www.disi.unige.it/person/MoggiE/ftp/ic91.pdf

[214] https://en.wikipedia.org/wiki/Doi_(identifier)

[215] https://doi.org/10.1016%2F0890-5401%2891%2990052-4

[216] https://en.wikipedia.org/wiki/CiteSeerX_(identifier)

[217] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.7385

[218] http://homepages.mcs.vuw.ac.nz/~rob/papers/modalhist.pdf

ical perspective", *Journal of Functional Programming,* **8** (2): 177–193, CiteSeerX[219] 10.1.1.258.6004[220], doi[221]:10.1017/s0956796898002998[222]

9. F. Binard and A. Felty, "Genetic programming with polymorphic types and higher-order functions." In *Proceedings of the 10th annual conference on Genetic and evolutionary computation,* pages 1187 1194, 2008.[1][223]
10. "Article"[224]. *bat8.inria.fr.* Retrieved 2020-01-31.
11. John c. Baez and Mike Stay, "Physics, Topology, Logic and Computation: A Rosetta Stone[225]", (2009) ArXiv 0903.0340[226] in *New Structures for Physics*, ed. Bob Coecke, *Lecture Notes in Physics* vol. **813**, Springer, Berlin, 2011, pp. 95–174.
12. "homotopy type theory - Google Trends"[227]. *trends.google.com.* Retrieved 2018-01-26.
13. *Homotopy Type Theory: Univalent Foundations of Mathematics*[228]. (2013) The Univalent Foundations Program. Institute for Advanced Study[229].

### 1.8.1 Seminal references

- Curry, H B (1934-09-20). "Functionality in Combinatory Logic"[230]. *Proceedings of the National Academy of Sciences of the United States of America.* **20** (11): 584–90. Bibcode[231]:1934PNAS...20..584C[232]. doi[233]:10.1073/pnas.20.11.584[234]. ISSN[235] 0027-8424[236]. PMC[237] 1076489[238]. PMID[239] 16577644[240].
- Curry, Haskell B; Feys, Robert (1958). Craig, William (ed.). *Combinatory Logic.* Studies in Logic and the Foundations of Mathematics. Vol. 1. North-Holland Publishing Company. LCCN[241] a59001593[242]; with two sections by Craig, William; see paragraph 9E`{{cite book`[243]`}}`: CS1 maint: postscript (link[244])

[219] https://en.wikipedia.org/wiki/CiteSeerX_(identifier)

[220] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.258.6004

[221] https://en.wikipedia.org/wiki/Doi_(identifier)

[222] https://doi.org/10.1017%2Fs0956796898002998

[223] http://www.site.uottawa.ca/~afelty/dist/gecco08.pdf

[224] http://bat8.inria.fr/~lang/ecrits/larecherche/03280721.html

[225] http://math.ucr.edu/home/baez/rosetta/rose3.pdf

[226] https://arxiv.org/abs/0903.0340/

[227] https://trends.google.com/trends/explore?date=all&q=%22homotopy%20type%20theory%22

[228] http://homotopytypetheory.org/book/

[229] https://en.wikipedia.org/wiki/Institute_for_Advanced_Study

[230] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076489

[231] https://en.wikipedia.org/wiki/Bibcode_(identifier)

[232] https://ui.adsabs.harvard.edu/abs/1934PNAS...20..584C

[233] https://en.wikipedia.org/wiki/Doi_(identifier)

[234] https://doi.org/10.1073%2Fpnas.20.11.584

[235] https://en.wikipedia.org/wiki/ISSN_(identifier)

[236] http://www.worldcat.org/issn/0027-8424

[237] https://en.wikipedia.org/wiki/PMC_(identifier)

[238] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076489

[239] https://en.wikipedia.org/wiki/PMID_(identifier)

[240] http://pubmed.ncbi.nlm.nih.gov/16577644

[241] https://en.wikipedia.org/wiki/LCCN_(identifier)

[242] http://lccn.loc.gov/a59001593

[243] https://en.wikipedia.org/wiki/Template:Cite_book

[244] https://en.wikipedia.org/wiki/Category:CS1_maint:_postscript

- De Bruijn, Nicolaas (1968), *Automath, a language for mathematics*, Department of Mathematics, Eindhoven University of Technology, TH-report 68-WSK-05. Reprinted in revised form, with two pages commentary, in: *Automation and Reasoning, vol 2, Classical papers on computational logic 1967–1970*, Springer Verlag, 1983, pp. 159–200.
- Howard, William A. (September 1980) [original paper manuscript from 1969], "The formulae-as-types notion of construction", in Seldin, Jonathan P.[245]; Hindley, J. Roger[246] (eds.), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press[247], pp. 479–490, ISBN[248] 978-0-12-349050-6[249].

### 1.8.2 Extensions of the correspondence

1. Pfenning, Frank; Davies, Rowan (2001), "A Judgmental Reconstruction of Modal Logic"[250] (PDF), *Mathematical Structures in Computer Science*, **11** (4): 511–540, CiteSeerX[251] 10.1.1.43.1611[252], doi[253]:10.1017/S0960129501003322[254], S2CID[255] 16467268[256]
2. Davies, Rowan; Pfenning, Frank (2001), "A Modal Analysis of Staged Computation"[257] (PDF), *Journal of the ACM*, **48** (3): 555–604, CiteSeerX[258] 10.1.1.3.5442[259], doi[260]:10.1145/382780.382785[261], S2CID[262] 52148006[263]

- Griffin, Timothy G. (1990), "The Formulae-as-Types Notion of Control", *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17–19 Jan 1990*, pp. 47–57, doi[264]:10.1145/96709.96714[265], ISBN[266] 978-0-89791-343-0[267], S2CID[268] 3005134[269].
- Parigot, Michel (1992), "Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction", *International Conference on Logic Program-*

[245] https://en.wikipedia.org/w/index.php?title=Jonathan_P._Seldin&action=edit&redlink=1
[246] https://en.wikipedia.org/wiki/J._Roger_Hindley
[247] https://en.wikipedia.org/wiki/Academic_Press
[248] https://en.wikipedia.org/wiki/ISBN_(identifier)
[249] https://en.wikipedia.org/wiki/Special:BookSources/978-0-12-349050-6
[250] https://www.cs.cmu.edu/~fp/papers/mscs00.pdf
[251] https://en.wikipedia.org/wiki/CiteSeerX_(identifier)
[252] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.1611
[253] https://en.wikipedia.org/wiki/Doi_(identifier)
[254] https://doi.org/10.1017%2FS0960129501003322
[255] https://en.wikipedia.org/wiki/S2CID_(identifier)
[256] https://api.semanticscholar.org/CorpusID:16467268
[257] https://www.cs.cmu.edu/~fp/papers/jacm00.pdf
[258] https://en.wikipedia.org/wiki/CiteSeerX_(identifier)
[259] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.5442
[260] https://en.wikipedia.org/wiki/Doi_(identifier)
[261] https://doi.org/10.1145%2F382780.382785
[262] https://en.wikipedia.org/wiki/S2CID_(identifier)
[263] https://api.semanticscholar.org/CorpusID:52148006
[264] https://en.wikipedia.org/wiki/Doi_(identifier)
[265] https://doi.org/10.1145%2F96709.96714
[266] https://en.wikipedia.org/wiki/ISBN_(identifier)
[267] https://en.wikipedia.org/wiki/Special:BookSources/978-0-89791-343-0
[268] https://en.wikipedia.org/wiki/S2CID_(identifier)
[269] https://api.semanticscholar.org/CorpusID:3005134

*ming and Automated Reasoning: LPAR '92 Proceedings, St. Petersburg, Russia*[270], Lecture Notes in Computer Science, vol. 624, Springer-Verlag[271], pp. 190–201, ISBN[272] 978-3-540-55727-2[273].

- Herbelin, Hugo (1995), ”A Lambda-Calculus Structure Isomorphic to Gentzen-Style Sequent Calculus Structure”, in Pacholski, Leszek; Tiuryn, Jerzy (eds.), *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25–30, 1994, Selected Papers*, Lecture Notes in Computer Science, vol. 933, Springer-Verlag[274], pp. 61–75, ISBN[275] 978-3-540-60017-6[276].
- Gabbay, Dov; de Queiroz, Ruy[277] (1992). ”Extending the Curry–Howard interpretation to linear, relevant and other resource logics”. *Journal of Symbolic Logic. The Journal of Symbolic Logic.* Vol. 57. pp. 1319–1365. doi[278]:10.2307/2275370[279]. JSTOR[280] 2275370[281].. (Full version of the paper presented at *Logic Colloquium '90*, Helsinki. Abstract in *JSL* 56(3):1139–1140, 1991.)
- de Queiroz, Ruy; Gabbay, Dov (1994), ”Equality in Labelled Deductive Systems and the Functional Interpretation of Propositional Equality”, in Dekker, Paul; Stokhof, Martin (eds.), *Proceedings of the Ninth Amsterdam Colloquium*, ILLC/Department of Philosophy, University of Amsterdam, pp. 547–565, ISBN[282] 978-90-74795-07-4[283].
- de Queiroz, Ruy; Gabbay, Dov (1995), ”The Functional Interpretation of the Existential Quantifier”[284], *Bulletin of the Interest Group in Pure and Applied Logics*, vol. 3, pp. 243–290. (Full version of a paper presented at *Logic Colloquium '91*, Uppsala. Abstract in *JSL* 58(2):753–754, 1993.)
- de Queiroz, Ruy; Gabbay, Dov (1997), ”The Functional Interpretation of Modal Necessity”, in de Rijke, Maarten (ed.), *Advances in Intensional Logic*, Applied Logic Series, vol. 7, Springer-Verlag[285], pp. 61–91, ISBN[286] 978-0-7923-4711-8[287].
- de Queiroz, Ruy; Gabbay, Dov (1999), ”Labelled Natural Deduction”[288], in Ohlbach, Hans-Juergen; Reyle, Uwe (eds.), *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay*, Trends in Logic, vol. 7, Kluwer, pp. 173–250, ISBN[289] 978-0-7923-5687-5[290].

[270] https://en.wikipedia.org/wiki/International_Conference_on_Logic_Programming_and_Automated_Reasoning

[271] https://en.wikipedia.org/wiki/Springer-Verlag

[272] https://en.wikipedia.org/wiki/ISBN_(identifier)

[273] https://en.wikipedia.org/wiki/Special:BookSources/978-3-540-55727-2

[274] https://en.wikipedia.org/wiki/Springer-Verlag

[275] https://en.wikipedia.org/wiki/ISBN_(identifier)

[276] https://en.wikipedia.org/wiki/Special:BookSources/978-3-540-60017-6

[277] https://en.wikipedia.org/wiki/Ruy_de_Queiroz

[278] https://en.wikipedia.org/wiki/Doi_(identifier)

[279] https://doi.org/10.2307%2F2275370

[280] https://en.wikipedia.org/wiki/JSTOR_(identifier)

[281] http://www.jstor.org/stable/2275370

[282] https://en.wikipedia.org/wiki/ISBN_(identifier)

[283] https://en.wikipedia.org/wiki/Special:BookSources/978-90-74795-07-4

[284] https://academic.oup.com/jigpal/article-abstract/3/2-3/243/2897783

[285] https://en.wikipedia.org/wiki/Springer-Verlag

[286] https://en.wikipedia.org/wiki/ISBN_(identifier)

[287] https://en.wikipedia.org/wiki/Special:BookSources/978-0-7923-4711-8

[288] https://www.springer.com/philosophy/logic/book/978-0-7923-5687-5

[289] https://en.wikipedia.org/wiki/ISBN_(identifier)

[290] https://en.wikipedia.org/wiki/Special:BookSources/978-0-7923-5687-5

- de Oliveira, Anjolina; de Queiroz, Ruy (1999), ”A Normalization Procedure for the Equational Fragment of Labelled Natural Deduction”, *Logic Journal of the Interest Group in Pure and Applied Logics*, vol. 7, Oxford University Press[291], pp. 173–215. (Full version of a paper presented at *2nd WoLLIC'95*, Recife. Abstract in *Journal of the Interest Group in Pure and Applied Logics* 4(2):330–332, 1996.)
- Poernomo, Iman; Crossley, John; Wirsing; Martin (2005), *Adapting Proofs-as-Programs: The Curry–Howard Protocol*, Monographs in Computer Science, Springer[292], ISBN[293] 978-0-387-23759-6[294], concerns the adaptation of proofs-as-programs program synthesis to coarse-grain and imperative program development problems, via a method the authors call the Curry–Howard protocol. Includes a discussion of the Curry–Howard correspondence from a Computer Science perspective.
- de Queiroz, Ruy J.G.B.; de Oliveira, Anjolina (2011), ”The Functional Interpretation of Direct Computations”, *Electronic Notes in Theoretical Computer Science*, Elsevier[295], **269**: 19–40, doi[296]:10.1016/j.entcs.2011.03.003[297]. (Full version of a paper presented at *LSFA 2010*, Natal, Brazil.)

### 1.8.3 Philosophical interpretations

- de Queiroz, Ruy J.G.B. (1994), ”Normalisation and language-games”[298], *Dialectica*, vol. 48, pp. 83–123. (Early version presented at *Logic Colloquium '88*, Padova. Abstract in *JSL* 55:425, 1990.)
- de Queiroz, Ruy J.G.B. (2001), ”Meaning, function, purpose, usefulness, consequences – interconnected concepts”[299], *Logic Journal of the Interest Group in Pure and Applied Logics*, vol. 9, pp. 693–734. (Early version presented at *Fourteenth International Wittgenstein Symposium (Centenary Celebration)* held in Kirchberg/Wechsel, August 13–20, 1989.)
- de Queiroz, Ruy J.G.B. (2008), ”On Reduction Rules, Meaning-as-use, and Proof-theoretic Semantics”, *Studia Logica*, **90** (2): 211–247, doi[300]:10.1007/s11225-008-9150-5[301], S2CID[302] 11321602[303].

[291] https://en.wikipedia.org/wiki/Oxford_University_Press

[292] https://en.wikipedia.org/wiki/Springer_Science%2BBusiness_Media

[293] https://en.wikipedia.org/wiki/ISBN_(identifier)

[294] https://en.wikipedia.org/wiki/Special:BookSources/978-0-387-23759-6

[295] https://en.wikipedia.org/wiki/Elsevier

[296] https://en.wikipedia.org/wiki/Doi_(identifier)

[297] https://doi.org/10.1016%2Fj.entcs.2011.03.003

[298] http://www3.interscience.wiley.com/journal/119262585/abstract?CRETRY=1&SRETRY=0

[299] http://jigpal.oxfordjournals.org/cgi/content/abstract/9/5/693

[300] https://en.wikipedia.org/wiki/Doi_(identifier)

[301] https://doi.org/10.1007%2Fs11225-008-9150-5

[302] https://en.wikipedia.org/wiki/S2CID_(identifier)

[303] https://api.semanticscholar.org/CorpusID:11321602

### 1.8.4 Synthetic papers

- De Bruijn, Nicolaas Govert (1995), ”On the roles of types in mathematics”[304] (PDF), in Groote, Philippe de (ed.), *De Groote 1995[305], pp. 27–54*, the contribution of de Bruijn by himself.
- Geuvers, Herman (1995), ”The Calculus of Constructions and Higher Order Logic”, *De Groote 1995[306], pp. 139–191*, contains a synthetic introduction to the Curry–Howard correspondence.
- Gallier, Jean H.[307] (1995), ”On the Correspondence between Proofs and Lambda-Terms”[308] (PDF), *De Groote 1995[309], pp. 55–138*, contains a synthetic introduction to the Curry–Howard correspondence.
- Wadler, Philip[310] (2014), ”Propositions as Types”[311] (PDF), *Communications of the ACM*, **58** (12): 75–84, doi[312]:10.1145/2699407[313], S2CID[314] 11957500[315]

### 1.8.5 Books

- De Groote, Philippe, ed. (1995), *The Curry–Howard Isomorphism*, Cahiers du Centre de Logique (Université catholique de Louvain), vol. 8, Academia-Bruylant, ISBN[316] 978-2-87209-363-2[317], reproduces the seminal papers of Curry-Feys and Howard, a paper by de Bruijn and a few other papers.
- Sørensen, Morten Heine; Urzyczyn, Paweł (2006) [1998], *Lectures on the Curry–Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, vol. 149, Elsevier Science[318], CiteSeerX[319] 10.1.1.17.7385[320], ISBN[321] 978-0-444-52077-7[322], notes on proof theory and type theory, that includes a presentation of the Curry–Howard correspondence, with a focus on the formulae-as-types correspondence
- Girard, Jean-Yves (1987–1990), *Proof and Types*[323], Cambridge Tracts in Theoretical Computer Science, vol. 7, Translated by and with appendices by Lafont, Yves and Taylor, Paul, Cambridge University Press, ISBN[324] 0-521-

[304] http://alexandria.tue.nl/repository/freearticles/597627.pdf
[305] #CITEREFDe_Groote1995
[306] #CITEREFDe_Groote1995
[307] https://en.wikipedia.org/wiki/Jean_Gallier
[308] ftp://ftp.cis.upenn.edu/pub/papers/gallier/cahiers.pdf
[309] #CITEREFDe_Groote1995
[310] https://en.wikipedia.org/wiki/Philip_Wadler
[311] http://homepages.inf.ed.ac.uk/wadler/papers/propositions-as-types/propositions-as-types.pdf
[312] https://en.wikipedia.org/wiki/Doi_(identifier)
[313] https://doi.org/10.1145%2F2699407
[314] https://en.wikipedia.org/wiki/S2CID_(identifier)
[315] https://api.semanticscholar.org/CorpusID:11957500
[316] https://en.wikipedia.org/wiki/ISBN_(identifier)
[317] https://en.wikipedia.org/wiki/Special:BookSources/978-2-87209-363-2
[318] https://en.wikipedia.org/wiki/Elsevier_Science
[319] https://en.wikipedia.org/wiki/CiteSeerX_(identifier)
[320] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.7385
[321] https://en.wikipedia.org/wiki/ISBN_(identifier)
[322] https://en.wikipedia.org/wiki/Special:BookSources/978-0-444-52077-7
[323] https://web.archive.org/web/20080418044121/http://www.monad.me.uk/stable/Proofs+Types.html
[324] https://en.wikipedia.org/wiki/ISBN_(identifier)

37181-3[325], ARCHIVED FROM THE ORIGINAL[326] ON 2008-04-18, notes on proof theory with a presentation of the Curry–Howard correspondence.

- THOMPSON, SIMON (1991), *Type Theory and Functional Programming*[327], ADDISON–WESLEY, ISBN[328] 0-201-41667-0[329].
- POERNOMO, IMAN; CROSSLEY, JOHN; WIRSING; MARTIN (2005), *Adapting Proofs-as-Programs: The Curry–Howard Protocol*, Monographs in Computer Science, Springer[330], ISBN[331] 978-0-387-23759-6[332], concerns the adaptation of proofs-as-programs program synthesis to coarse-grain and imperative program development problems, via a method the authors call the Curry–Howard protocol. Includes a discussion of the Curry–Howard correspondence from a Computer Science perspective.
- BINARD, F.; FELTY, A. (2008), "GENETIC PROGRAMMING WITH POLYMORPHIC TYPES AND HIGHER-ORDER FUNCTIONS"[333] (PDF), *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, Association for Computing Machinery, pp. 1187–94, doi[334]:10.1145/1389095.1389330[335], ISBN[336] 9781605581309[337], S2CID[338] 3669630[339]
- DE QUEIROZ, RUY J.G.B.; DE OLIVEIRA, ANJOLINA G.; GABBAY, DOV M. (2011), *The Functional Interpretation of Logical Deduction*[340], ADVANCES IN LOGIC, VOL. 5, IMPERIAL COLLEGE PRESS/WORLD SCIENTIFIC, ISBN[341] 978-981-4360-95-1[342].
- MIMRAM, SAMUEL (2020), *Program = proof*[343], INDEPENDENTLY PUBLISHED, ISBN[344] 979-8615591839[345]

## 1.9 Further reading

- JOHNSTONE, P.T.[346] (2002), "D4.2 λ-CALCULUS AND CARTESIAN CLOSED CATEGORIES", *Sketches of an Elephant*[347], A TOPOS THEORY COMPENDIUM, VOL. 2, CLARENDON PRESS,

[325] https://en.wikipedia.org/wiki/Special:BookSources/0-521-37181-3
[326] http://www.monad.me.uk/stable/Proofs+Types.html
[327] http://www.cs.kent.ac.uk/people/staff/sjt/TTFP/
[328] https://en.wikipedia.org/wiki/ISBN_(identifier)
[329] https://en.wikipedia.org/wiki/Special:BookSources/0-201-41667-0
[330] https://en.wikipedia.org/wiki/Springer_Science%2BBusiness_Media
[331] https://en.wikipedia.org/wiki/ISBN_(identifier)
[332] https://en.wikipedia.org/wiki/Special:BookSources/978-0-387-23759-6
[333] http://www.site.uottawa.ca/~afelty/dist/gecco08.pdf
[334] https://en.wikipedia.org/wiki/Doi_(identifier)
[335] https://doi.org/10.1145%2F1389095.1389330
[336] https://en.wikipedia.org/wiki/ISBN_(identifier)
[337] https://en.wikipedia.org/wiki/Special:BookSources/9781605581309
[338] https://en.wikipedia.org/wiki/S2CID_(identifier)
[339] https://api.semanticscholar.org/CorpusID:3669630
[340] https://books.google.com/books?id=aFO6CgAAQBAJ
[341] https://en.wikipedia.org/wiki/ISBN_(identifier)
[342] https://en.wikipedia.org/wiki/Special:BookSources/978-981-4360-95-1
[343] https://www.amazon.com/PROGRAM-PROOF-Samuel-Mimram/dp/B08C97TD9G
[344] https://en.wikipedia.org/wiki/ISBN_(identifier)
[345] https://en.wikipedia.org/wiki/Special:BookSources/979-8615591839
[346] https://en.wikipedia.org/wiki/P.T._Johnstone
[347] https://books.google.com/books?id=TLHfQPHNs0QC

PP. 951–962, ISBN[348] 978-0-19-851598-2[349] — gives a categorical[350] view of "what happens" in the Curry–Howard correspondence.

## 1.10 External links

The Wikibook *Haskell*[351] has a page on the topic of: ***The Curry–Howard isomorphism***[352]

- Howard on Curry-Howard[353]
- The Curry–Howard Correspondence in Haskell[354]
- The Monad Reader 6: Adventures in Classical-Land[355]: Curry–Howard in Haskell, Pierce's law.

- This page was last edited on 15 February 2022, at 16:30 (UTC).
-

[348] https://en.wikipedia.org/wiki/ISBN_(identifier)
[349] https://en.wikipedia.org/wiki/Special:BookSources/978-0-19-851598-2
[350] https://en.wikipedia.org/wiki/Categorical_logic
[351] https://en.wikibooks.org/wiki/Haskell
[352] https://en.wikibooks.org/wiki/Haskell/The_Curry%E2%80%93Howard_isomorphism
[353] http://wadler.blogspot.com/2014/08/howard-on-curry-howard.html
[354] https://web.archive.org/web/20080819185521/http://www.thenewsh.com/~newsham/formal/curryhoward/
[355] http://www.haskell.org/wikiupload/1/14/TMR-Issue6.pdf
[356] http://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License
[357] http://creativecommons.org/licenses/by-sa/3.0/
[358] http://foundation.wikimedia.org/wiki/Terms_of_Use
[359] http://foundation.wikimedia.org/wiki/Privacy_policy
[360] http://www.wikimediafoundation.org/

# 2 Contributors

| Edits | User |
|---|---|
| 1 | 22merlin[1] |
| 1 | 414owen[2] |
| 1 | Addbot[3] |
| 7 | Ancheta Wis[4] |
| 1 | Andreasabel[5] |
| 1 | Andyhowlett[6] |
| 2 | AnomieBOT[7] |
| 1 | Anthony[8] |
| 1 | Anthony Appleyard[9] |
| 1 | ArnoldReinhold[10] |
| 1 | AugPi[11] |
| 1 | BattyBot[12] |
| 2 | Beland[13] |
| 1 | Bellerophon5685[14] |
| 13 | Ben Standeven[15] |
| 2 | Bender the Bot[16] |
| 1 | Burritoburritoburrito[17] |
| 1 | CRGreathouse[18] |
| 11 | Cedar101[19] |
| 5 | Chalst[20] |
| 14 | Charles Matthews[21] |

[1] https://en.wikipedia.org/wiki/User:22merlin

[2] https://en.wikipedia.org/w/index.php%3ftitle=User:414owen&action=edit&redlink=1

[3] https://en.wikipedia.org/wiki/User:Addbot

[4] https://en.wikipedia.org/wiki/User:Ancheta_Wis

[5] https://en.wikipedia.org/w/index.php%3ftitle=User:Andreasabel&action=edit&redlink=1

[6] https://en.wikipedia.org/w/index.php%3ftitle=User:Andyhowlett&action=edit&redlink=1

[7] https://en.wikipedia.org/wiki/User:AnomieBOT

[8] https://en.wikipedia.org/wiki/User:Anthony

[9] https://en.wikipedia.org/wiki/User:Anthony_Appleyard

[10] https://en.wikipedia.org/wiki/User:ArnoldReinhold

[11] https://en.wikipedia.org/wiki/User:AugPi

[12] https://en.wikipedia.org/wiki/User:BattyBot

[13] https://en.wikipedia.org/wiki/User:Beland

[14] https://en.wikipedia.org/wiki/User:Bellerophon5685

[15] https://en.wikipedia.org/wiki/User:Ben_Standeven

[16] https://en.wikipedia.org/wiki/User:Bender_the_Bot

[17] https://en.wikipedia.org/wiki/User:Burritoburritoburrito

[18] https://en.wikipedia.org/wiki/User:CRGreathouse

[19] https://en.wikipedia.org/w/index.php%3ftitle=User:Cedar101&action=edit&redlink=1

[20] https://en.wikipedia.org/wiki/User:Chalst

[21] https://en.wikipedia.org/wiki/User:Charles_Matthews

| | |
|---|---|
| 2 | Chinju[22] |
| 3 | Chris55[23] |
| 7 | Citation bot[24] |
| 1 | CitationCleanerBot[25] |
| 16 | Classicalecon[26] |
| 1 | ClueBot NG[27] |
| 1 | Cmdrjameson[28] |
| 2 | Cybercobra[29] |
| 2 | David Eppstein[30] |
| 1 | DefLog~enwiki[31] |
| 4 | Demonburrito[32] |
| 1 | Dionyziz[33] |
| 16 | Dominus[34] |
| 2 | Doradus[35] |
| 1 | Eaefremov[36] |
| 2 | Edward[37] |
| 1 | Elaz85[38] |
| 1 | EmausBot[39] |
| 2 | EricP[40] |
| 1 | Eskimbot[41] |
| 2 | Four Dog Night[42] |
| 2 | Francis Lima[43] |
| 1 | François Robere[44] |
| 4 | FrescoBot[45] |
| 2 | Frietjes[46] |

[22] https://en.wikipedia.org/wiki/User:Chinju

[23] https://en.wikipedia.org/wiki/User:Chris55

[24] https://en.wikipedia.org/wiki/User:Citation_bot

[25] https://en.wikipedia.org/wiki/User:CitationCleanerBot

[26] https://en.wikipedia.org/wiki/User:Classicalecon

[27] https://en.wikipedia.org/wiki/User:ClueBot_NG

[28] https://en.wikipedia.org/wiki/User:Cmdrjameson

[29] https://en.wikipedia.org/wiki/User:Cybercobra

[30] https://en.wikipedia.org/wiki/User:David_Eppstein

[31] https://en.wikipedia.org/wiki/User:DefLog~enwiki

[32] https://en.wikipedia.org/wiki/User:Demonburrito

[33] https://en.wikipedia.org/wiki/User:Dionyziz

[34] https://en.wikipedia.org/wiki/User:Dominus

[35] https://en.wikipedia.org/wiki/User:Doradus

[36] https://en.wikipedia.org/wiki/User:Eaefremov

[37] https://en.wikipedia.org/wiki/User:Edward

[38] https://en.wikipedia.org/wiki/User:Elaz85

[39] https://en.wikipedia.org/wiki/User:EmausBot

[40] https://en.wikipedia.org/wiki/User:EricP

[41] https://en.wikipedia.org/wiki/User:Eskimbot

[42] https://en.wikipedia.org/wiki/User:Four_Dog_Night

[43] https://en.wikipedia.org/w/index.php%3ftitle=User:Francis_Lima&action=edit&redlink=1

[44] https://en.wikipedia.org/w/index.php%3ftitle=User:Fran%25C3%25A7ois_Robere&action=edit&redlink=1

[45] https://en.wikipedia.org/wiki/User:FrescoBot

[46] https://en.wikipedia.org/wiki/User:Frietjes

| | |
|---|---|
| 1 | Genneth[47] |
| 4 | Giftlite[48] |
| 1 | Goheeca[49] |
| 1 | Goldenowl[50] |
| 1 | GreenWeasel11[51] |
| 1 | Greenbreen[52] |
| 2 | Gregbard[53] |
| 1 | Gwern[54] |
| 19 | Hairy Dude[55] |
| 1 | Headbomb[56] |
| 1 | Helpful Pixie Bot[57] |
| 42 | Hugo Herbelin[58] |
| 1 | Igrant[59] |
| 1 | InternetArchiveBot[60] |
| 3 | Iwehrman[61] |
| 3 | JJMC89 bot III[62] |
| 1 | JRSpriggs[63] |
| 1 | JaGa[64] |
| 1 | Jamiemichelle[65] |
| 1 | Jarble[66] |
| 1 | Jleedev[67] |
| 8 | Jochen Burghardt[68] |
| 1 | John of Reading[69] |
| 2 | Jon Awbrey[70] |
| 1 | Josve05a[71] |

[47] `https://en.wikipedia.org/wiki/User:Genneth`
[48] `https://en.wikipedia.org/wiki/User:Giftlite`
[49] `https://en.wikipedia.org/w/index.php%3ftitle=User:Goheeca&action=edit&redlink=1`
[50] `https://en.wikipedia.org/w/index.php%3ftitle=User:Goldenowl&action=edit&redlink=1`
[51] `https://en.wikipedia.org/wiki/User:GreenWeasel11`
[52] `https://en.wikipedia.org/wiki/User:Greenbreen`
[53] `https://en.wikipedia.org/wiki/User:Gregbard`
[54] `https://en.wikipedia.org/wiki/User:Gwern`
[55] `https://en.wikipedia.org/wiki/User:Hairy_Dude`
[56] `https://en.wikipedia.org/wiki/User:Headbomb`
[57] `https://en.wikipedia.org/wiki/User:Helpful_Pixie_Bot`
[58] `https://en.wikipedia.org/wiki/User:Hugo_Herbelin`
[59] `https://en.wikipedia.org/wiki/User:Igrant`
[60] `https://en.wikipedia.org/wiki/User:InternetArchiveBot`
[61] `https://en.wikipedia.org/wiki/User:Iwehrman`
[62] `https://en.wikipedia.org/wiki/User:JJMC89_bot_III`
[63] `https://en.wikipedia.org/wiki/User:JRSpriggs`
[64] `https://en.wikipedia.org/wiki/User:JaGa`
[65] `https://en.wikipedia.org/wiki/User:Jamiemichelle`
[66] `https://en.wikipedia.org/wiki/User:Jarble`
[67] `https://en.wikipedia.org/wiki/User:Jleedev`
[68] `https://en.wikipedia.org/wiki/User:Jochen_Burghardt`
[69] `https://en.wikipedia.org/wiki/User:John_of_Reading`
[70] `https://en.wikipedia.org/wiki/User:Jon_Awbrey`
[71] `https://en.wikipedia.org/wiki/User:Josve05a`

| | |
|---|---|
| 1 | Jpgross3[72] |
| 1 | Jrtayloriv[73] |
| 1 | Karlheg[74] |
| 2 | KolbertBot[75] |
| 1 | Kranix[76] |
| 1 | LaaknorBot[77] |
| 1 | Lambda Fairy[78] |
| 1 | Laocoön11[79] |
| 1 | Legobot[80] |
| 1 | Leibniz[81] |
| 1 | Linas[82] |
| 1 | Magic links bot[83] |
| 1 | Mathbot[84] |
| 1 | Mattghg[85] |
| 1 | McM.bot[86] |
| 1 | Mdaviscs[87] |
| 4 | Mhss[88] |
| 17 | Michael Hardy[89] |
| 1 | Michael Slone[90] |
| 1 | Michaelmalak[91] |
| 2 | MilesAgain[92] |
| 3 | Monkbot[93] |
| 1 | N4nojohn[94] |
| 1 | Natematic[95] |
| 1 | Nickj[96] |

[72] https://en.wikipedia.org/w/index.php%3ftitle=User:Jpgross3&action=edit&redlink=1

[73] https://en.wikipedia.org/wiki/User:Jrtayloriv

[74] https://en.wikipedia.org/wiki/User:Karlheg

[75] https://en.wikipedia.org/wiki/User:KolbertBot

[76] https://en.wikipedia.org/wiki/User:Kranix

[77] https://en.wikipedia.org/wiki/User:LaaknorBot

[78] https://en.wikipedia.org/w/index.php%3ftitle=User:Lambda_Fairy&action=edit&redlink=1

[79] https://en.wikipedia.org/w/index.php%3ftitle=User:Laoco%25C3%25B6n11&action=edit&redlink=1

[80] https://en.wikipedia.org/wiki/User:Legobot

[81] https://en.wikipedia.org/wiki/User:Leibniz

[82] https://en.wikipedia.org/wiki/User:Linas

[83] https://en.wikipedia.org/wiki/User:Magic_links_bot

[84] https://en.wikipedia.org/wiki/User:Mathbot

[85] https://en.wikipedia.org/wiki/User:Mattghg

[86] https://en.wikipedia.org/wiki/User:McM.bot

[87] https://en.wikipedia.org/w/index.php%3ftitle=User:Mdaviscs&action=edit&redlink=1

[88] https://en.wikipedia.org/wiki/User:Mhss

[89] https://en.wikipedia.org/wiki/User:Michael_Hardy

[90] https://en.wikipedia.org/wiki/User:Michael_Slone

[91] https://en.wikipedia.org/wiki/User:Michaelmalak

[92] https://en.wikipedia.org/wiki/User:MilesAgain

[93] https://en.wikipedia.org/wiki/User:Monkbot

[94] https://en.wikipedia.org/wiki/User:N4nojohn

[95] https://en.wikipedia.org/wiki/User:Natematic

[96] https://en.wikipedia.org/wiki/User:Nickj

| | |
|---|---|
| 1 | Nnxion[97] |
| 2 | Noamz[98] |
| 3 | OAbot[99] |
| 2 | Oerjan[100] |
| 2 | Oleg Alexandrov[101] |
| 1 | OriumX[102] |
| 1 | Paradoctor[103] |
| 3 | Pcap[104] |
| 1 | Phil Boswell[105] |
| 5 | Physis[106] |
| 1 | Pintoch[107] |
| 1 | Ptbotgourou[108] |
| 2 | Quondum[109] |
| 2 | RDBrown[110] |
| 1 | RebelRobot[111] |
| 1 | RedBot[112] |
| 9 | Rjwilmsi[113] |
| 2 | RjwilmsiBot[114] |
| 2 | Rowandavies[115] |
| 18 | Ruud Koot[116] |
| 1 | Sarming[117] |
| 1 | SethTisue[118] |
| 1 | Seunghun[119] |
| 1 | ShelfSkewed[120] |
| 1 | SieBot[121] |

[97] https://en.wikipedia.org/w/index.php%3ftitle=User:Nnxion&action=edit&redlink=1
[98] https://en.wikipedia.org/wiki/User:Noamz
[99] https://en.wikipedia.org/wiki/User:OAbot
[100] https://en.wikipedia.org/wiki/User:Oerjan
[101] https://en.wikipedia.org/wiki/User:Oleg_Alexandrov
[102] https://en.wikipedia.org/wiki/User:OriumX
[103] https://en.wikipedia.org/wiki/User:Paradoctor
[104] https://en.wikipedia.org/wiki/User:Pcap
[105] https://en.wikipedia.org/wiki/User:Phil_Boswell
[106] https://en.wikipedia.org/wiki/User:Physis
[107] https://en.wikipedia.org/wiki/User:Pintoch
[108] https://en.wikipedia.org/wiki/User:Ptbotgourou
[109] https://en.wikipedia.org/wiki/User:Quondum
[110] https://en.wikipedia.org/wiki/User:RDBrown
[111] https://en.wikipedia.org/wiki/User:RebelRobot
[112] https://en.wikipedia.org/wiki/User:RedBot
[113] https://en.wikipedia.org/wiki/User:Rjwilmsi
[114] https://en.wikipedia.org/wiki/User:RjwilmsiBot
[115] https://en.wikipedia.org/w/index.php%3ftitle=User:Rowandavies&action=edit&redlink=1
[116] https://en.wikipedia.org/wiki/User:Ruud_Koot
[117] https://en.wikipedia.org/w/index.php%3ftitle=User:Sarming&action=edit&redlink=1
[118] https://en.wikipedia.org/wiki/User:SethTisue
[119] https://en.wikipedia.org/w/index.php%3ftitle=User:Seunghun&action=edit&redlink=1
[120] https://en.wikipedia.org/wiki/User:ShelfSkewed
[121] https://en.wikipedia.org/wiki/User:SieBot

| | |
|---|---|
| 1 | SparsityProblem[122] |
| 2 | Taral[123] |
| 3 | Tea2min[124] |
| 1 | Toby Bartels[125] |
| 1 | Tre2[126] |
| 1 | Txa[127] |
| 1 | Vlad Patryshev[128] |
| 3 | Wasell[129] |
| 7 | Wavelength[130] |
| 1 | WikiCleanerBot[131] |
| 1 | Xqbot[132] |
| 4 | Yobot[133] |
| 3 | ZéroBot[134] |

[122] https://en.wikipedia.org/wiki/User:SparsityProblem

[123] https://en.wikipedia.org/wiki/User:Taral

[124] https://en.wikipedia.org/wiki/User:Tea2min

[125] https://en.wikipedia.org/wiki/User:Toby_Bartels

[126] https://en.wikipedia.org/wiki/User:Tre2

[127] https://en.wikipedia.org/wiki/User:Txa

[128] https://en.wikipedia.org/wiki/User:Vlad_Patryshev

[129] https://en.wikipedia.org/wiki/User:Wasell

[130] https://en.wikipedia.org/wiki/User:Wavelength

[131] https://en.wikipedia.org/wiki/User:WikiCleanerBot

[132] https://en.wikipedia.org/wiki/User:Xqbot

[133] https://en.wikipedia.org/wiki/User:Yobot

[134] https://en.wikipedia.org/wiki/User:Z%25C3%25A9roBot

# List of Figures

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/deed.en
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.
- LFK: Lizenz Freie Kunst. http://artlibre.org/licence/lal/de
- CFR: Copyright free use.

- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[135]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

[135] Chapter 3 on page 37

# 3 Licenses

## 3.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”. * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

## 3.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

## 3.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user’s computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.