Dirace: Practical Proof Automation of Dirac Notation Equations

No Author Given

No Institute Given

Abstract. Dirac notations is the fundamental language in quantum computation and quantum information reasoning. Recently, the term rewriting system DiracDec [3] was introduced to automate equational reasoning with Dirac notations, a critical yet time-intensive component of formalization tasks. This work bases upon and extends DiracDec, aiming to develop a solver optimized for practical applications. Enhancements include an improved typing system, a simplified language and rewriting system, more efficient algorithms, and added support for labelled Dirac notations. The solver Dirace, implemented in C++ with a Mathematica backend, demonstrates superior decision-making power and computational efficiency, successfully addressing all examples in its benchmark suite.

1 Introduction

In 1939, Dirac proposed his notation [2] for quantum mechanics, which is designed to represent linear algebra formulae in a compact and convenient way. For instance, $a\,|\psi\rangle + b\,|\phi\rangle$ indicates the addition of two vectors, i.e., the superposition of two states $|\psi\rangle$ and $|\phi\rangle$. Dirac notation is now widely accepted as the working language in quantum computation and quantum information. The reasonings of Dirac notation play a fundamental role in research and application, just like boolean and integer logic to classical computer science. For example, formalization works of quantum algorithms and quantum programming languages involves plenty of equational proofs of Dirac notation, which are critical, repeating but time-intensive jobs. Dirac notation are also used in quantum programming languages to define the program states, operations and assertions. In order to automate the verification procedure, we need to simplify and check the equivalence of pre-conditions. However, unlike the existence of SAT and SMT solver, we are still in need of a working Dirac notation solver, and this absence has become an obstacle of many areas.

Recently, Xu et al. [3] proposed a theory to decide the equivalence of Dirac notations, as well as a prototype implementation in Mathematica called DiracDec. They proved that the equivalence of basic Dirac notations are decidable. Their algorithm sticks with a pure term rewriting system, which allows to prove important properties such as confluence and termination of the system. Even though, there is still a gap between DiracDec and a practical solver for Dirac notation equivalence.

One problem is the algorithm efficiency for equivalence out of the scope of rewritings. DiracDec decides the whole equational theory by rewriting modulo E, where E is a set of equational axioms that cannot be decided by normalization in rewritings, e.g. the associativity and commutativity (AC) of some function symbols. DiracDec uses a direct by inefficient algorithm to decide E, which search through all possible permutations and has factorial complexity. This is witnessed in their evaluation: DiracDec is not good at dealing with "computational" examples, which has many AC symbols and are therefore time consuming.

Another problem is the lack of labelled Dirac notations. which makes the reasoning more convenient and scaleable.

One last problem is about usability. The typing of DiracDec only deals with Dirac notations, which is not sufficient for a working scenario, where defined symbols are desirable, and variables have their typing assumptions in the context. Moreover, to avoid type checking during term rewriting, DiracDec uses different symbols for the same casual notation to disambiguate, which is not close to the natural way of using the langauge, and makes the system bloated. It is hard to integrate the Mathematica implementation as solvers into into other tools. Although DiracDec solve most examples in the benchmark, including some very complicated equations, there are still a few examples that DiracDec failds to decide.

Aimming at building a practical solver, this work bases on and extends the DiracDec result, transform the term rewriting system into a hybrid algorithm, and overcomes the problems mentioned above. Our main contributions and techniques are:

- An efficient algorithm to decide the extra equational theories E. The basic idea is normalization by sorting, which considers the interplay of AC symbols, SUM-SWAP and α-equivalence. This algorithm reduces the complexity from the factorial level to the polynomial level.
- The support for constant register labels. The equivalence decision of labelled Dirac notations is then reduced into the unlabelled situation.
- A more usable C++ solver Dirace with usable language and typing. Defined symbols (e.g., transpose and trace) are modelled by functions.

As an evaluation, we tested Dirace on the benchmark of DiracDec and new example for labelled Dirac notations. Dirace successfully decides all the examples that are expressable in its language, including those failed by DiracDec because of complexity or insufficient decision power.

2 Preliminary and Motivation

The preliminary for the problem are two folds: Dirac notation and the equational logic in universal algebra.

Quantum states live in complex Hilbert spaces. We use vectors in the space to descirbe pure quantum states, and operations corresponds to linear transformations. Dirac notation are used to express the linear algebra formula in a convenient and compat way. It uses the ket $|i\rangle$ and the bra $\langle i|$ to indicate bases of the space and the dual space. Together with other variable symbols, they are composed with each other in sequence, and the composition will be interpreted into different operations, depending on the type of operands. For example, $\langle i|j\rangle$ represents the inner product of $\langle i|$ and $|j\rangle$, which is a scalar, while $|i\rangle \langle j|$ represents the outer product, resulting in an operator. Dirac notation also use $\otimes$ to indicate the vectors and operators in the tensor product space.

Dirac notation further enjoys the property that the interpretation is independent on the order of composition, thus parentheses can be omitted. For example, the formula $\langle i| \, |\phi\rangle \, \langle\psi| \, |j\rangle$ can be understood as

$$\langle i|\phi\rangle \langle\psi|j\rangle = \langle i| \left(|\phi\rangle \langle\psi|\right) |j\rangle ,$$

and they are equivalent for all variables.

With the concrete basis, Dirac notations can be interpreted as matrices. For example, the ket of two basis of qubit states are column vectors $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$, and the corresponding bras are row vectors in the dual space $\langle 0| = \begin{bmatrix}1 \ 0\end{bmatrix}$ and $\langle 1| = \begin{bmatrix}0 \ 1\end{bmatrix}$. The calculation between Dirac notations then corresponds to matrix operatoins. For example, inner product $\langle 0|1\rangle = \begin{bmatrix}0 \ 1\end{bmatrix} \begin{bmatrix}1\\0\end{bmatrix} = 0$, and tensor product $|0\rangle \otimes |1\rangle = |(0,1)\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$.

In practice, Dirac notations are combined with other syntax for better expressiveness. For example, the big operator sum $\sum_{i\in S} A$ are used in theoretical reasonings. We also use labelled Dirac notation, where Dirac notations are subscripted with registers to denote the quantum system in consideration, e.g. $|i\rangle_q \otimes |i\rangle_r$.

We use universal algebra and equational logic to formally represent Dirac notations and the reasoning procedure. A universal algebra defines a signature of function symbols and the axioms. Terms in the algebra are constructed by constants, variables or function application on terms. In our case of Dirac notation, the signature consists of constructors and operations like $|i\rangle$ or $A \otimes B$, and the axioms Dirac notation equivalenc rules with equational logic.

Here is an example proof of a equivalence lemma.

Example 1. Let q and r be two quantum systems on $\mathcal{H}_T$ space. Let M be a quantum operation on $\mathcal{H}_T$, and $|\Phi\rangle = \sum_{i\in T} |i\rangle \otimes |i\rangle$ be the maximally entangled state, then we have

$$M_q \, |\Phi\rangle_{q;r} = M_r^T \, |\Phi\rangle_{q;r} .$$

[YX] : I guess it's wise to avoid mentioning that we don't impose fixed arity.

3 Language, Typing and Semantics

The syntax of Dirac notations involves three layers: the index, the type and the term. Terms describe the concrete instances like ket, bra and operator, which will be checked and typed. Index represents classical datatypes, and they appear in type expression to distinguish between different Hilbert spaces and sets.

Definition 1 (index syntax). *The syntax for type indices is*

$$\sigma ::= x \mid \sigma_1 \times \sigma_2 \mid \mathsf{bool}.$$

Here x is a variable. $\sigma_1 \times \sigma_2$ is the product type, which is used to represent the type of tensor product space, or Cartesian product sets. bool is a constant index with two basis, and is used in Dirac notations of qubit systems.

Definition 2 (type syntax). *The syntax for Dirac notation types is*

$$T ::= \mathsf{Basis}(\sigma) \mid \mathcal{S} \mid \mathcal{K}(\sigma) \mid \mathcal{B}(\sigma) \mid \mathcal{O}(\sigma_1, \sigma_2) \mid T_1 \to T_2 \mid \forall x.T \mid \mathsf{Set}(\sigma).$$

$\mathsf{Basis}(\sigma)$ is the type for basis in index σ. $\mathcal{S}$ indicates scalars, $\mathcal{K}(\sigma)$ and $\mathcal{B}(\sigma)$ indicates ket and bra terms in Hilbert space σ, and $\mathcal{O}(\sigma_1, \sigma_2)$ indicates linear operators with Hilbert space σ_2 as domain and σ_1 as codomain. $\mathsf{Set}(\sigma)$ is the type of subsets in σ index, and is used to denote the values of bound variables in summation. The remaining two generators are types for functions. $T_1 \to T_2$ denotes normal functions, which accepts a T_1 type argument and results in a T_2 type term. $\forall x.T$ denotes index functions, which accepts an index argument x and results in a T type term, where T can depend on x.

Definition 3 (term syntax). *The syntax for Dirac notation terms is*

$$\begin{aligned} e ::=\ & x \mid \lambda x : T.e \mid \mu x.e \mid e_1\ e_2 \mid e_1 \circ e_2 \\ & \mid \hat{0} \mid \hat{1} \mid (e_1, e_2) \\ & \mid 0 \mid 1 \mid \mathsf{ADDS}(e_1, \cdots, e_n) \mid e_1 \times \cdots \times e_n \mid e^* \mid \delta_{e_1, e_2} \mid \mathsf{DOT}(e_1, e_2) \\ & \mid \mathbf{0}_\mathcal{K}(\sigma) \mid \mathbf{0}_\mathcal{B}(\sigma) \mid \mathbf{0}_\mathcal{O}(\sigma_1, \sigma_2) \mid \mathbf{1}_\mathcal{O}(\sigma) \\ & \mid\ |e\rangle \mid \langle t| \mid e^\dagger \mid e_1.e_2 \mid \mathsf{ADD}(e_1, \cdots, e_n) \mid e_1 \otimes e_2 \\ & \mid \mathsf{MULK}(e_1, e_2) \mid \mathsf{MULB}(e_1, e_2) \mid \mathsf{OUTER}(e_1, e_2) \mid \mathsf{MULO}(e_1, e_2) \\ & \mid \mathbf{U}(e) \mid e_1 \star e_2 \mid \sum_{e_1} e_2. \end{aligned}$$

The generators are explained in order. $\lambda x : T.e$ is the abstraction for normal functions, and $\mu x.e$ is the abstraction for index functions. $e_1\ e_2$ is function application. $e_1 \circ e_2$ is the composition operator of Dirac notations. $\hat{0}$ and $\hat{1}$ are constant basis of $\mathsf{Basis}(\mathsf{bool})$ type. (e_1, e_2) is the basis pair for product types. 0, 1, ADDS, $e_1 \times \cdots \times e_n$ and e^* are symbols for scalars. The next line includes constant symbols for ket, bra and operator. $e^\dagger$ denotes the conjugate transpose of e. $e_1.e_2$ denotes scaling the term e_2 with scalar e_1. $\mathbf{U}(e)$ is the universal set with index e. $e_1 \star e_2$ is the Cartesian product. $\sum_{e_1} e_2$ is the big operator sum, which is modelled by folding the function e_2 over value sets e_1. Usually, the sum body is specified by an abstraction. Therefore we use the notation $\sum_{x\in s} X$ to denote $\sum_s \lambda x : T.X$ as well.

Here ADDS and ADD are two different AC symbols representing the scalar addition and the linear algebra addition respectively. There are five kinds of linear algebra multiplications among ket, bra and operator. We encode the typing information by using five different symbols, namely DOT, MULK, MULB, OUTER and MULO. They are denoted as $B \cdot K$, $K_1 \cdot K_2$, $B_1 \cdot B_2$, $K \cdot B$ and $O_1 \cdot O_2$, respectively.

Compared to DiracDec, our language for Dirac notations merges the symbols with identital properties, such as the addition and scaling symbols for ket, bra and operator. One important difference is that AC symbols are treated as functions with indefinite arities.

3.1 Typing System

The type checking of our language involves maintaining a well-formed environment and context $E[\Gamma]$, which is defined as follows.

Definition 4 (environment and context).

$$E ::= [] \mid E; x : \mathsf{Index} \mid E; x : T \mid E; x := t : T.$$
$$\Gamma ::= [] \mid \Gamma; x : \mathsf{Index} \mid \Gamma; x : T.$$

The environment and the context are sequences of assumptions $x : T$ or definitions $x := t : T$. With the environment, we can declare the type of variable symbols, and encode more operations on Dirac notations as definitions, such as the trace operator. Also, the existence of lambda abstractions requires a context of bound variables.

We say an expression t has type X in context $E[\Gamma]$, if the typing judgement $E[\Gamma] \vdash t : X$ can be proved through the typing rules in Appendix A. Here we present and explain the rules selectively. Firstly, well-formed contexts $\mathcal{WF}(E)[\Gamma]$ are built in the incremental way, e.g.:

$$\frac{}{\mathcal{WF}([])[]} \qquad \frac{\mathcal{WF}(E)[] \qquad x \notin E}{\mathcal{WF}(E; x : \mathsf{Index})[]} \qquad \frac{E[] \vdash t : T \qquad x \notin E}{\mathcal{WF}(E; x := t : T)[]}.$$

Starting from an empty context, we can assume new index symbols, and assume or define symbols with checked types. Based on the well-formed context, typing judgements can be proved by information from $E[\Gamma]$, or built inductively. In the following rules, for example, the condition $x : \mathsf{Index} \in E[\Gamma]$ is true if E or Γ has the assumption in their sequences, and $\sigma \times \tau$ is a index if both σ and τ are typed as the index. $\mathcal{K}(\sigma)$ and $\mathcal{O}(\sigma, \tau)$ will be valid types for kets and operators, if their arguments are typed as the index.

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad x : \mathsf{Index} \in E[\Gamma]}{E[\Gamma] \vdash x : \mathsf{Index}} \qquad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index} \qquad E[\Gamma] \vdash \tau : \mathsf{Index}}{E[\Gamma] \vdash \sigma \times \tau : \mathsf{Index}}$$

$$\frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathcal{K}(\sigma) : \mathsf{Type}} \qquad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index} \qquad E[\Gamma] \vdash \tau : \mathsf{Index}}{E[\Gamma] \vdash \mathcal{O}(\sigma, \tau) : \mathsf{Type}}$$

The Dirac notations will then be typed accordingly. For example, the ket syntax $|t\rangle$ has type $\mathcal{K}(\sigma)$, if t is typed as a basis term of index σ. Also, the inner product of a bra and a ket with the same type index σ is typed as the scalar. This corresponds to the constraint of inner products that vectors should be in the same Hilbert space.

$$\frac{E[\Gamma] \vdash t : \mathsf{Basis}(\sigma)}{E[\Gamma] \vdash |t\rangle : \mathcal{K}(\sigma)} \qquad \frac{E[\Gamma] \vdash B : \mathcal{B}(\sigma) \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash B \cdot K : \mathcal{S}}$$

The typing for functions and applications follow the common practice. We specify the syntax and typing rule for functions of indices $\mu x.t$: here x is a bound variable of typed as Index, and the type $U\{x/u\}$ of application (tu) is obtained by replacing x with the index instance u.

$$\frac{E[\Gamma; x : T] \vdash t : U}{E[\Gamma] \vdash (\lambda x : T.t) : T \to U} \qquad \frac{E[\Gamma] \vdash t : U \to T \qquad E[\Gamma] \vdash u : U}{E[\Gamma] \vdash (t\ u) : T}$$

$$\frac{E[\Gamma; x : \mathsf{Index}] \vdash t : U}{E[\Gamma] \vdash (\mu x.t) : \forall x.U} \qquad \frac{E[\Gamma] \vdash t : \forall x.U \qquad E[\Gamma] \vdash u : \mathsf{Index}}{E[\Gamma] \vdash (t\ u) : U\{x/u\}}$$

The big operator sum is modelled by folding a function on a set, therefore the typing rule is as follows:

$$\frac{E[\Gamma] \vdash s : \mathsf{Set}(\sigma) \qquad E[\Gamma] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{K}(\tau)}{E[\Gamma] \vdash \sum_s f : \mathcal{K}(\tau)}.$$

And lastly we have the typing rules for composition $x \circ y$. Similar to casual Dirac notations, the typing of composition depends on the types of operands.

$$\frac{E[\Gamma] \vdash x : \mathcal{S} \qquad E[\Gamma] \vdash y : \mathcal{K}(\sigma)}{E[\Gamma] \vdash x \circ y : \mathcal{K}(\sigma)} \qquad \frac{E[\Gamma] \vdash x : \mathcal{O}(\sigma, \tau) \qquad E[\Gamma] \vdash y : \mathcal{K}(\tau)}{E[\Gamma] \vdash x \circ y : \mathcal{K}(\sigma)}$$

Lemma 1. *The typing of expressions are decidable and unique.*

Proof. The type can be calculated in the recursive manner. There is at most one typing rule for the same function symbol and argument types, which leads to the uniqueness of typing.

3.2 Semantics

Semantics define the meanings of expressions, and the goal of the desirable algorithm is to decide whether two expressions have the equivalent semantics. We can give the semantics by axiomatic equations and the denotational interpretations.

Denotational semantics The denotational way interprets every expression as a set theory object in linear algebra, according to a valuation mapping v from variables to their values. This explanation formalizes the common explanation of Dirac notations, and best describes the target of the algorithm. The semantics of e with valuation v is written as $[\![e]\!]_v$, and two expressions e_1 and e_2 are equivalence if they have equal semantics in the mathematical sense for all valuations, written as $[\![e_1]\!] = [\![e_2]\!]$.

Variables typed with Index are interpreted as finite sets, and index product $[\![\sigma_1 \times \sigma_2]\!] = [\![\sigma_1]\!] \times [\![\sigma_2]\!]$ are defined as Cartesian product of sets. Following on, every type is interpreted as a set. For example, the scalar type $[\![\mathcal{S}]\!] = \mathbb{C}$ is interpreted as complex number set, the the ket and bra type $[\![\mathcal{K}(\sigma)]\!] = \mathcal{H}_{[\![\sigma]\!]}$, $[\![\mathcal{K}(\sigma)]\!] = \mathcal{H}^*_{[\![\sigma]\!]}$ are interpreted as the Hilbert space and dual space depending on σ. One special case is the delta symbol $[\![\delta_{s,t}]\!] = \begin{cases} 1, \text{where } [\![s]\!] = [\![t]\!], \\ 0, \text{where } [\![s]\!] \neq [\![t]\!]. \end{cases}$.

The idea behind the interpretation of types and terms is to characterize the typing relation with set theory inclusion. Therefore we have the following lemma.

Lemma 2. *For all well-formed context $E[\Gamma]$, term t and type T, if $E[\Gamma] \vdash t : T$, then for all valuations v, $[\![t]\!]_v \in [\![T]\!]_v$.*

Axiomatic semantics An other way is to express the equivalence by a set of axioms. For example, we have the absorption law of zero symbols $X \cdot \mathbf{0} = \mathbf{0}$, and the bilinearity of dot product

$$\begin{array}{cc} (a.X) \cdot Y = a.(X \cdot Y) & X \cdot (Y_1 + Y_2) = X \cdot Y_1 + X \cdot Y_2 \\ X \cdot (a.Y) = a.(X \cdot Y) & (X_1 + X_2) \cdot Y = X_1 \cdot Y + X_2 \cdot Y \end{array}$$

Note that some equational theories cannot be decided by term rewriting, and are collected in the set E:

$$\begin{array}{llll} \text{AC-equivalence} & \text{e.g.} \quad a + b = b + a & & (a + b) + c = a + (b + c) \\ \alpha\text{-equivalence} & \lambda x.A = \lambda y.A\{x/y\} & & \\ \text{(SUM-SWAP)} & \displaystyle\sum_{i \in s_1} \sum_{j \in s_2} A = \sum_{j \in s_2} \sum_{i \in s_1} A & & \end{array}$$

The equations should also include axioms for scalars. In the implementation, we use the Mathematica kernel to decide scalar equivalences, so the axioms are not given here.

The equational axioms provide an operable theory for the proof automation algorithm, and the denotational semantics can be considered as one model for the theory. In this sense, the equivalence derived by axioms is weaker than the denotational semantics.

Lemma 3. *For all well-formed context $E[\Gamma]$ and terms e_1, e_2, if $AX \vdash e_1 = e_2$, then $[\![e_1]\!] = [\![e_2]\!]$.*

Unfortunately, the inverse does not hold: there are expressions that are equivalent in denotational interpretations but not provable through the axioms. One notorious example is $\delta_{i,j} \times \delta_{j,k} = \delta_{i,k} \times \delta_{j,k}$. Intuitively, the two terms will be evaluated to 1 if and only if $i = j = k$, but it is difficult to incorporate such properties as axioms. This incapability further limits the algorithmm, which is designed to automate reasonings using the axioms, and blocks us from proving complete results with respect to the semantics. But our work focuses on solving the practical examples, which are covered by our axioms, as is shown by experiments.

[YX] : where should we put the example? To conclude the design introduction, we demonstrate part of the formalization of the preliminary example.

Example 2 (formalizing the preliminary example). Definitions and assumptions in the environment:

$$\begin{aligned}
&\text{TPO} && := \mu T_1.\mu T_2.\lambda O : \mathcal{O}(T_1, T_2). \sum_{i \in \mathbf{U}(T_1)} \sum_{j \in \mathbf{U}(T_2)} \langle i| \, O \, |j\rangle \, . \, |j\rangle \, \langle i| \\
& && : \forall T_1.\forall T_2.\mathcal{O}(T_1, T_2) \to \mathcal{O}(T_2, T_1); \\
&\text{phi} && := \mu T. \sum_{i \in \mathbf{U}(T)} \sum_{j \in \mathbf{U}(T)} |(i, j)\rangle : \forall T.\mathcal{K}(T \times T); \\
&\text{T} && : \mathsf{Index}; \\
&\text{M} && : \mathcal{O}(T, T).
\end{aligned}$$

The symbol TPO is the encoding of operator transpose. It takes two index T_1 and T_2, and transforms the operators. phi takes the index T and defines the maximally entangled states. It sums over all basis in T, indicated by the universal set $\mathbf{U}(T)$. With the assumption of index T and operator M, we can write the equivalence as

$$(\text{M} \otimes \mathbf{1}_{\mathcal{O}}(\text{T})) \cdot (\text{phi T}) = (\mathbf{1}_{\mathcal{O}}(\text{T}) \otimes (\text{TPO T T M})) \cdot (\text{phi T}).$$

4 Algorithm for Deciding Dirac notations equations

The following two sections discuss how to decide the equivalence of Dirac notations implied by the axioms. In the previous work, the equational axioms are separated into two parts: a set E of equations that can not be decided by rewriting, and the remaining equations to be decided by term rewriting. Here, the equational theories E inlude:

- commutativity of symbols $a + b$, $a \times b$ and $\delta_{s,t}$,
- α-equivalence of bound variables, i.e., $\lambda x.t = \lambda y.t\{x/y\}$,
- swapping successive summations, i.e., $\sum_{i\in s_1}\sum_{j\in s_2} A = \sum_{j\in s_2}\sum_{i\in s_1} A$, and
- equational theories for scalars.

In this work, the general idea is to carry through the normalization procedure, so that semantical equivalence can be directly checked by the syntax of normal forms. The procedure of the normalization is displayed in Figure 1.

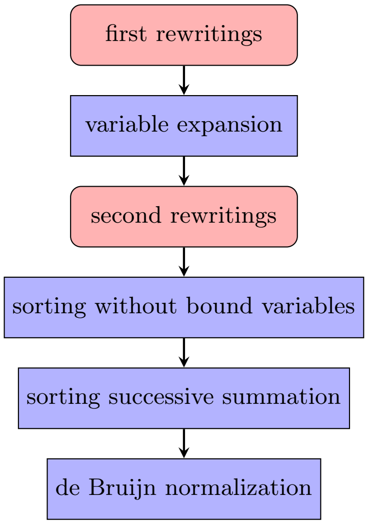

Fig. 1. A flowchart for normalization of Dirac notations.

The last three steps deals with the equational theory E, while the first three steps use term rewriting to work on the structure of Dirac notations.

4.1 Normalization modulo E by Term Rewriting

Term rewriting rules, written as $l- > r$, are used to normalize terms by recursively matching the subterms of the term with the left hand side l, and replace them with the corresponding right hand side r. The procedure terminates when no more rewritings can be made, and the order of rewritings will be irrelevant if the term rewriting system is *confluent*, which is a desirable property.

To express the term rewriting system, the previous work adheres to use the naive universal algebra, where each function symbol has a fixed arity, and the left hand side pattern only allows constant symbols and variables. This constraint enables them to check the confluence and termination of their term rewriting system with other tools.

The full list of rewriting rules are in Appendix C. Here we present some of them to illustrate the design idea. The whole rule list consists of several types.

Firstly, we use the following flattening rule to decide the associative with AC symbols.

$$a_1 + \cdots + (b_1 + \cdots + b_m) + \cdots + a_n \ \triangleright\ a_1 + \cdots + b_1 + \cdots + b_m + \cdots + a_n,$$

and commutativity is left for a sorting algorithm later on.

Some equational axioms imply obvious rewriting diraction, for example:

$$\begin{array}{ll}
\text{(R-DOT6)} & \langle s| \cdot |t\rangle \ \triangleright \delta_{s,t} \\
\text{(R-DELTA0)} & \delta_{s,s} \ \triangleright\ 1 \\
\text{(R-MULK1)} & O : \mathcal{O}(\sigma,\tau) \Rightarrow O \cdot \mathbf{0}_{\mathcal{K}(\tau)} \ \triangleright \mathbf{0}_{\mathcal{K}(\sigma)}
\end{array}$$

The (R-DOT6) says that inner product of two basis will be evaluated to a delta expression, and (R-DELTA0) further transforms delta on identical basis to scalar 1. (R-MULK1) corresponds to the axiom that multiplication on zero vectors results in zero. It has a condition on typing, which is calculated during rewriting. They share the common feature that right hand side is simpler than the left hand side.

Other part of rules assign the rewriting diraction to equations according to the property of symbols and intuitive observations of their relations. When different rules rewrites the same term to different forms, rules for confluence purpose are added.

$$\begin{array}{ll}
\text{(R-SCR1)} & a.(b.X) \ \triangleright\ (a \times b).X \\
\text{(R-DOT10)} & (B \cdot O) \cdot K \ \triangleright\ B \cdot (O \cdot K), \\
\text{(R-DOT11)} & \langle (s,t)| \cdot ((O_1 \otimes O_2) \cdot K) \ \triangleright\ (((\langle s| \cdot O_1) \otimes (\langle t| \cdot O_2)) \cdot K, \\
\text{(R-MULB10)} & \langle (s,t)| \cdot (O_1 \otimes O_2) \ \triangleright\ (\langle s| \cdot O_1) \otimes (\langle t| \cdot O_2),
\end{array}$$

The (R-SCR1) rule follows the idea to reduce linear algebra scaling into scalar multiplications, and (R-MULB10) transform inner product into tensor product. As an example of the completetion rule, here the normalization of term $(\langle (s,t)| \cdot (O_1 \otimes O_2)) \cdot K$ have two rewriting paths: (a). apply (R-MULB10) and get $(((\langle s| \cdot O_1) \otimes (\langle t| \cdot O_2)) \cdot K$, or (b). first apply (R-DOT10) and sort the term into $\langle (s,t)| \cdot ((O_1 \otimes O_2) \cdot K)$, and then apply (R-DOT11) to get the same result. Here (R-DOT11) is for completeness of cases similar to this example.

There are also rules dealing with sum expressions. For example,

$$\begin{array}{ll}
\text{(R-SUM-ELIM2)} & i \text{ free in } t \Rightarrow \sum_{i \in \mathbf{U}(\sigma)} \sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} (\delta_{i,t}.A) \\
& \triangleright \sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} A\{i/t\} \\
\text{(R-SUM-PUSH5)} & (\sum_{i \in M} B) \cdot K \ \triangleright\ \sum_{i \in M} (B \cdot K)
\end{array}$$

The (R-SUM-ELIM2) rule will try to eliminate the delta expression in summations. And the (R-SUM-PUSH5) rule will lift sum to the outside of inner product.

One important technique revealed in the DiracDec work is the expansion of variables:

$$\frac{E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash K \ \triangleright\ \sum_{i \in \mathbf{U}(\sigma)} (\langle i| \cdot K).\, |i\rangle} \qquad \frac{E[\Gamma] \vdash B : \mathcal{B}(\sigma)}{E[\Gamma] \vdash B \ \triangleright\ \sum_{i \in \mathbf{U}(\sigma)} (B \cdot |i\rangle).\, \langle i|}$$

$$\frac{E[\Gamma] \vdash O : \mathcal{O}(\sigma, \tau)}{E[\Gamma] \vdash O \ \triangleright\ \sum_{i \in \mathbf{U}(\sigma)} \sum_{j \in \mathbf{U}(\tau)} (\langle i| \cdot O \cdot |j\rangle).(|i\rangle \cdot \langle j|)}$$

The above three rules are not terminating, and this is why we have the rewriting-expansion-rewriting steps in the algorithm. On the other hand, we discovered that doing the expansion on all variables only once is already sufficient.

Lemma 4. *Let expand(e) indicate the result of expanding all variables in e once. For all well-typed term e in $E[\Gamma]$, we have expand$($expand$(e)) \downarrow=$ expand$(e) \downarrow$.*

Proof.

5 Deciding Equational Theory *E*

In the Mathematica implementation of DiracDec, the equational theory E is decided by a unification, which tries to find a substitution of summation bound variables and AC symbol arguments that makes the two expressions syntactically equivalent. The unification iterates through all permutations, and the complexity is factorial in general.

As the first improvement, the α-equivalence are checked by de Bruijn index [1], where references to bound variable names are replaced by the distance of the lambda abstraction to the variable. For example, the nominal lambda abstraction $\lambda x.x$ is transformed into $\lambda.1$, while $\lambda x.\lambda y.(x\ (y\ x))$ will is transformed into $\lambda.\lambda.(2\ (1\ 2))$. Therefore, transformation into de Bruijn index is at the last step of the normalization to check the equivalence of terms with different bound variable names.

A standard approach to decide this permutation equivalence is to normalize by sorting with a given order. For example, given the dictionary order $a < b < c$, the term $b + c + a$ (and any other AC equivalent ones) will be normalized into $a + b + c$. For our setting, there are two difficulties intertwined together: how to assign such an order to all terms in our language, and how to normalize with respect to E. As a typical example, the two terms

$$\sum_{i \in s_1} \sum_{j \in s_2} \langle i| A |j\rangle \times \langle j| B |i\rangle = \sum_{i \in s_2} \sum_{j \in s_1} \langle i| B |j\rangle \times \langle j| A |i\rangle$$

are equivalent due to the swap of summations and commutativity of scalar multiplication. However, directly sorting elemnts of scalar multiplication will not lead to the same form.

We propose an algorithm to solve the problem by sorting without bound variables. The observation is that in a successive sum expression $\sum_{i\in s_1} \cdots \sum_{j\in s_n} A$, the name and order of bound variables $i, \cdots, j$ can be permuted freely. Therefore, all bound variables should be treated uniformly during sorting, and the order of sum can then be decided according to the position of their bound variables.

In the above example, we ignore the bound variables first, and sort the sum body into $\langle\bullet| A |\bullet\rangle \times \langle\bullet| B |\bullet\rangle$. Then we swap the sum so that the bound variable at first $\bullet$ position appears at the outside, and the two terms will have the same de Bruijn normal form as $\sum_{s_1} \sum_{s_2} \langle\$1| A |\$0\rangle \times \langle\$0| B |\$1\rangle$.

To describe the algorithm, we introduce two notations: for a term $e = f(a_1, a_2, \cdots, a_n)$, $\mathrm{head}(e)$ indicates the function symbol f, and $\mathrm{arg}(e, i)$ indicates the i-th argument a_i.

Definition 5 (order without bound variables). *Assume we have a total order of all symbols. Let $\mathcal{B}$ be the set of bound variables. For simplicity, we assume all bound variables are unique. The relation $e_1 =_{\mathcal{B}} e_2$ holds when*

- *$head(e_1) = head(e_2)$ and $arg(e_1, i) =_{\mathcal{B}} arg(e_2, i)$ for all i, or*
- *$e_1 \in \mathcal{B}$ and $e_2 \in \mathcal{B}$.*

The relation $e_1 <_{\mathcal{B}} e_2$ between the two terms holds when:

- *$e_1 \notin \mathcal{B}$ and $e_2 \in \mathcal{B}$, or*
- *$head(e_1) < head(e_2)$, or*
- *$head(e_1) = head(e_2)$, and there exists n with $arg(e_1, n) <_{\mathcal{B}} arg(e_2, n)$, where $arg(e_1, i) =_{\mathcal{B}} arg(e_2, i)$ for all $i < n$.*

It can be checked that $e_1 =_{\mathcal{B}} e_2$ if and only if neither $e_1 <_{\mathcal{B}} e_2$ or $e_2 <_{\mathcal{B}} e_1$ holds. The idea is to compare the function symbols in the top down order, and omit bound variable occurance.

The idea is to assign an order to terms, which is independent on the bound variables. Because we can have terms with nested AC symbols.

At last, we will prove that equivalence by this normalization procedure is sound in the semantics.

Theorem 1 (soundness). *For any well-formed context $E[\Gamma]$ and well-typed expressions e_1 and e_2, if $e_1 \downarrow = e_2 \downarrow$, then $[\![e_1]\!] = [\![e_2]\!]$.*

Proof.

6 Labelled Dirac Notation

Labelled Dirac notation uses register names to indicate the quantum system of vectors and operators. This allows us to talk aboud the states and operations locally, without referring to the whole system. For instance, assume Q and R are two registers, we have

$$M_Q \cdot |\Phi\rangle_{(Q,R)} = ((M \otimes I)\, |\Phi\rangle)_{(Q,R)}.$$

In the left hand side, the state of two subsystems is $|\Phi\rangle$, and we apply quantum operation M on the system Q.

In this section, we introduce the language and typing of registers and labelled Dirac notations, and demonstrate how to transform the equivalence problem into the Dirac notations studied above.

We assume the set of register symbols is $\mathcal{R}$. The syntax of quantum registers is defined as follows.

Definition 6 (quantum registers).

$$R ::= r \in \mathcal{R} \mid (R, R)$$

Registers can be composed by pairs of (R, R), and this structure corresponds to the structure of tensor product spaces in Dirac notations.

To reason about the registers, we define their variable set as the enumeration of all register symbols involved.

Definition 7 (register variable set). *The variable set of a register is defined inductively by:*

- *$R \equiv r$: $\mathsf{var}(R) = \{r\}$;*
- *$R \equiv (R_1, R_2)$: $\mathsf{var}(R) = \mathsf{var}(R_1) \cup \mathsf{var}(R_2)$.*

Remark: Set operations: $\cup$ for union; $\cap$ for intersection; $\setminus$ for difference. So, $S_1 \cap S_2 \equiv S_1 \cup S_2 \setminus (S_1 \setminus S_2) \setminus (S_2 \setminus S_1)$.

Definition 8 (labelled Dirac notation). *The **labelled Dirac notation** includes all Dirac notation symbols and the generators defined below. Here, $s \subseteq \mathcal{R}$ is a register variable set.*

$$\begin{aligned} T &::= \mathcal{D}(s, s) \mid \mathsf{Reg}(\sigma) \\ e &::= R \mid |i\rangle_r \mid {}_r\langle i| \mid e_R \mid e_{R;R} \mid e \otimes e \otimes \cdots \otimes e \mid e \cdot e \end{aligned}$$

$\mathcal{D}(s_1, s_2)$ is the unified type for all labelled Dirac notations, where s_1 indicates the codomain systems and s_2 indicates the domain systems. For instance, labelled ket has type $\mathcal{D}(s_1, \emptyset)$, and labelled bra has type $\mathcal{D}(\emptyset, s_2)$. $\mathsf{Reg}(\sigma)$ are types for registers R, and the index σ indicates the type of Hilbert space represented by the register. Terms also include the labelled notation e_R for bra, ket and $e_{R;R}$ for operators. We introduce new dot and tensor product symbols for labelled Dirac notations. In labelled Dirac notation, the structure of tensor product does not matter. Therefore $\otimes$ is an AC symbol. Following the unified type $\mathcal{D}(s, s)$, all kinds of multiplications are represented by the same dot product $e \cdot e$. Finally, $|i\rangle_r$ and ${}_r\langle i|$ are labelled basis for the normal form of labelled Dirac notations, where r are registers symbols in $\mathcal{R}$.

Typing rules

$$\frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathsf{Reg}(\sigma) : \mathsf{Type}} \qquad \frac{E[\Gamma] \vdash r : \mathsf{Reg}(\sigma_r) \text{ for all } r \text{ in } s_1 \text{ and } s_2}{E[\Gamma] \vdash \mathcal{D}(s_1, s_2) : \mathsf{Type}}$$

$$\frac{E[\Gamma] \vdash R : \mathsf{Reg}(\sigma) \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash K_R : \mathcal{D}(\mathsf{var}R, \emptyset)} \qquad \frac{E[\Gamma] \vdash D : \mathcal{D}(s_1, s_2)}{E[\Gamma] \vdash D^\dagger : \mathcal{D}(s_2, s_1)}$$

The typing rule for tensor product is

$$\frac{E[\Gamma] \vdash D_i : \mathcal{D}(s_i, s_i') \qquad \bigcap_i s_i = \emptyset \qquad \bigcap_i s_i' = \emptyset}{E[\Gamma] \vdash D_1 \otimes \cdots \otimes D_i : \mathcal{D}(\bigcup_i s_i, \bigcup_i s_i')}.$$

The typing rule for dot product is

$$\frac{\begin{array}{ll} E[\Gamma] \vdash D_1 : \mathcal{D}(s_1, s_1') & s_1 \cap s_2 \backslash s_1' = \emptyset \\ E[\Gamma] \vdash D_2 : \mathcal{D}(s_2, s_2') & s_2' \cap s_1' \backslash s_2 = \emptyset \end{array}}{E[\Gamma] \vdash D_1 \cdot D_2 : \mathcal{D}(s_1 \cup (s_2 \backslash s_1'), s_2' \cup (s_1' \backslash s_2))}.$$

6.1 Normalization

The main idea of normalizing labelled Dirac notations is *label elimination*. It expands e_R and $e_{R;R}$ by tensor products of labelled basis, which are formalized as rules below.

$$\begin{aligned}
K_R &\ \triangleright \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} (\langle i_R| \cdot K).(|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes |i_{r_n}\rangle_{r_n}) \\
B_R &\ \triangleright \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} (B \cdot |i_R\rangle).({}_{r_1}\langle i_{r_1}| \otimes \cdots \otimes {}_{r_n}\langle i_{r_n}|) \\
O_{R,R'} &\ \triangleright \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} \sum_{i_{r_1'} \in \mathbf{U}(\sigma_{r_1'})} \cdots \sum_{i_{r_{n'}'} \in \mathbf{U}(\sigma_{r_{n'}'})} \\
&\quad (\langle i_R| \cdot O \cdot |i_{R'}\rangle).(|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes |i_{r_n}\rangle_{r_n} \otimes {}_{r_1'}\langle i_{r_1'}| \otimes \cdots \otimes {}_{r_{n'}'}\langle i_{r_{n'}'}|)
\end{aligned}$$

$$\begin{array}{ll}
\text{(R-SUM-PUSHD0)} & X_1 \otimes \cdots (\sum_{i \in M} D) \cdots \otimes X_2 \ \triangleright \ \sum_{i \in M} (X_1 \otimes \cdots D \cdots \otimes X_n) \\
\text{(R-SUM-PUSHD1)} & (\sum_{i \in M} D_1) \cdot D_2 \ \triangleright \ \sum_{i \in M} (D_1 \cdot D_2)
\end{array}$$

$$\begin{array}{ll}
\text{(R-L-SORT0)} & A : \mathcal{D}(s_1, s_2), B : \mathcal{D}(s_1', s_2'), s_2 \cap s_1' = \emptyset \Rightarrow A \cdot B \ \triangleright \ A \otimes B \\
\text{(R-L-SORT1)} & {}_r\langle i| \cdot |j\rangle_r \ \triangleright \ \delta_{i,j} \\
\text{(R-L-SORT2)} & {}_r\langle i| \cdot (Y_1 \otimes \cdots \otimes |j\rangle_r \otimes \cdots \otimes Y_m) \ \triangleright \ \delta_{i,j}.(Y_1 \otimes \cdots \otimes Y_m)
\end{array}$$

Eliminating the registers

1. For any register R s.t. $E[\Gamma] \vdash R : \mathsf{Reg}\sigma$, suppose $\mathsf{var}R = \{r_1, r_2, \cdots, r_n\}$, we introduce variables $i_{r_k} : \mathsf{Basis}(\sigma_{r_k})$ with $r_k : \mathsf{Reg}\sigma_{r_k} \in E$ for $k = 1, \cdots, n$. We reconstruct the basis $|i_R\rangle$ (which is of type $\mathcal{K}(\sigma)$) and $\langle i_R|$ (which is of type $\mathcal{B}(\sigma)$) of R by:
   - $R = r_k$, $|i_R\rangle \triangleq |i_{r_k}\rangle$ and $\langle i_R| \triangleq \langle i_{r_k}|$;
   - $R = (R_1, R_2)$: $|i_R\rangle \triangleq |i_{R_1}\rangle \otimes |i_{R_2}\rangle$ and $\langle i_R| \triangleq \langle i_{R_1}| \otimes \langle i_{R_2}|$.
2. Expansion of all $K_R, B_R, O_{R,R'}$
   After the rewriting of this step, the expression we have is always in the following form:
   $$\Big(\sum_{i_1} \cdots \sum_{i_n} a.\tilde{D}_f\Big) + \cdots + \Big(\sum_{i_1} \cdots \sum_{i_{n'}} a.\tilde{D}_f\Big)$$
   where $\tilde{D}_f$ is generated by :
   $$\tilde{D}_f ::= |i\rangle_r \mid {}_r\langle i| \mid \tilde{e} \otimes \tilde{e} \mid \tilde{e} \cdot \tilde{e}.$$
3. Suppose $\mathsf{CDom}(\tilde{D}_f) = \{(i_1, r_1), \cdots, (i_n, r_n)\}$ and $\mathsf{Dom}(\tilde{D}_f) = \{(i'_1, r'_1), \cdots, (i'_{n'}, r'_{n'})\}$ where the list is sorted by a default order of variables in $\mathcal{R}$. We define:
   $$D_f = \Bigg(\prod_{(i,j)\in\mathsf{Can}(\tilde{D}_f)} \delta_{i,j}\Bigg) . \Big((|i_1\rangle \otimes \cdots \otimes |i_n\rangle) \cdot (\langle i'_1| \otimes \cdots \otimes \langle i'_{n'}|)\Big).$$
   If Dom or CDom is an empty list, the D_f is just a Ket or Bra. Then we claim:
   - If $E[\Gamma] \vdash \tilde{D}_f : \mathcal{D}(s_1, s_2)$, then $\{r_1, \cdots, r_n\} = s_1$ and $\{r'_1, \cdots, r'_{n'}\} = s_2$.
   - Set $R = ((\cdots(r_1, r_2), \cdots), r_n)$ and $R' = ((\cdots(r'_1, r'_2), \cdots), r'_{n'})$, then
   $$\tilde{D}_f = (D_f)_{R,R'}$$
4. For the obtained formula in (3), we replace each $\tilde{D}_f$ by D_f which is computed by step (4). Note that D_f is a Dirac term without labels, so the equivalence of two labelled Dirac terms is reduced to decide the equivalence of two Dirac terms. This is guaranteed if two labelled Dirac terms are of the same type (by typing rules).

7 Implementation and Case Study

The main purpose of this work is to build a practical tool that works well in checking Dirac notation equations. The refinements and extensions above concludes in our implementation called Dirace. Dirace is a solver written in C++ working as a command line tool. It has a parser built by ANTLR4, and scalar reasonings are powered by a Mathematica kernel. The user can use commands to make definitions and assumptions in the maintained context, conduct the normalization and equivalence checking, and obtain the rewriting trace output. This implementation is tested on the benchmark of the DiracDec work, and succeeds in proving most of them efficiently.

Here is the code for the preliminary example.

```
Var T : INDEX. Var M : OTYPE[T, T].
Def phi := idx T => Sum nv in USET[T], |(nv, nv)>.
Var r1 : REG[T]. Var r2 : REG[T].
CheckEq M_r1;r1 (phi T)_(r1, r2) with (TPO T T M)_r2;r2 (phi T)_(r1, r2).
```

The first three lines use the `Var` and `Def` commands to set up the context for the Dirac notation. Here `T` is a type index, representing arbitrary Hilbert space type. `M` is assume to be an operator in Hilbert space with type `T`. `phi` is defined as the maximally entangled state.

Besides, the symbol `TPO` represents transpose of operators, and the definition is provided as

```
Def TPO := idx sigma => idx tau => fun O : OTYPE[sigma, tau] => Sum i in
    USET[sigma], Sum j in USET[tau], (<i| O |j>).(|j> <i|).
```

And the output result is:

```
The two terms are equal.
[Normalized Term] SUM[USET[T], FUN[BASIS[T], SUM[USET[T], FUN[BASIS[T],
    SCR[DOT[BRA[$1], MULK[M, KET[$0]]], LTSR[LKET[$1, r1], LKET[$0, r2
    ]]]]]]] : DTYPE[RSET[r1, r2], RSET]
```

The normal form can be transformed into the formatted version:

$$\sum_{\mathbf{U}(T)} \sum_{\mathbf{U}(T)} \langle \$1 | M | \$0 \rangle \, . \, |\$1\rangle_{r1} \otimes |\$0\rangle_{r2} : \mathcal{D}(\{r1, r2\}, \emptyset)$$

Here $\$0$ and $\$1$ are de Bruijn indices. The result is a ket on the {r1, r2} system as expected.

7.1 Benchmark performance

It does not support fst and snd projectors. It does not support interactive proof, as is used in the reasonings of quantum programming language theory.

8 Related Work

Other previous works exploring the language and algorithm to reasong about quantum computation differently.

9 Conclusion

Based on the first Dirac notation equational reasoning tool DiracDec, this work refines and extends the theory for practical application, and provides the solver Dirace. Experimental results shows that the tool can correctly identify all encoded equivalent expressions efficiently. We expect Dirace will play a role in areas like quantum program verification or proofs of post-quantum cryptography protocols.

10 TODO

- better output.
- better error logic
- the $\bar{U_{AC}}$, $\bar{U_{BC}}$ terms can be modelled by quantum registers.

11 Things to Note

- The context should also be maintained during rewriting matching.
- We don't allow eta reduction. It will intertwine with SUM-SWAP and break the confluence.
- In each Delta reduction, the bound variables are replaced with unique variables. This should help solve the problem of conflicting variable names during substitution.

12 `Diracoq` language

$$
\begin{aligned}
\mathit{cmd} ::= {} & \texttt{Def}(\texttt{ID}\ \mathit{term}) \\
& \mid \texttt{Def}(\texttt{ID}\ \mathit{term}\ \mathit{type}) \\
& \mid \texttt{Var}(\texttt{ID}\ \mathit{term}) \\
& \mid \texttt{Check}(\mathit{term}) \\
& \mid \texttt{Show}(\texttt{ID}) \\
& \mid \texttt{ShowAll} \\
& \mid \texttt{Normalize}(\mathit{term}) \mid \texttt{Normalize}(\mathit{term}\ \textsf{Trace}) \\
& \mid \texttt{CheckEq}(\mathit{term}\ \mathit{term}) \\
\mathit{type} ::= {} & \texttt{Type} \mid \texttt{Arrow}(\mathit{type}\ \mathit{type}) \\
& \mid \texttt{Base} \\
\mathit{term} ::= {} & \texttt{Type} \mid \texttt{fun}(\texttt{ID}\ \mathit{type}\ \mathit{term}) \mid \texttt{apply}(\mathit{term}\ \mathit{term}) \mid \texttt{ID}
\end{aligned}
$$

Comment `(* ... *)` can be inserted between commands.

These are tye parsing rules for different expressions:

- `Def ID := term.` — `Def(ID term)`
- `Def ID := term : type.` — `Def(ID term type)`
- `Var ID : type.` — `Var(ID type)`
- `Check ID.` — `Check(term)`
- `Show ID.` — `Show(ID)`
- `ShowAll.` — `ShowAll`
- `Normalize term.` — `Normalize(term)`
- `Normalize term with trace.` — `Normalize(term Trace)`
- `Check term = term.` — `CheckEq(term term)`

- `T1 -> T2` — `Arrow(T1 T2)`.
- `forall x. T` — `Forall(x T)`
- `(e1, e2)` — `PAIR(e1 e2)`
- `fun x : T => e` — `fun(x T e)`
- `idx x => e` — `idx(x e)`
- `e1 @ e2` — `COMPO(e1 e2)`
- `e1 + ... + en` — `ADDG(e1 ... en)`
- `e1 * ... * en` — `STAR(e1 ... en)`
- `e1^*` — `CONJ(e1)`
- `delta(e1, e2)` — `DELTA(e1 e2)`
- `|e>` — `KET(e)`
- `<e|` — `BRA(e)`
- `e1^D` — `ADJ(e1)`
- `e1.e2` — `SCR(e1 e2)`
- `Sum(i in s, e)` — `SSUM(i s e)`

These symbols will be transformed into internal language:

- $A \circ B$: $S \circ S$, $S \circ K$, $S \circ B$, $S \circ O$, $K \circ S$, $K \circ K$, $K \circ B$, $B \circ S$, $B \circ K$, $B \circ B$, $B \circ O$, $O \circ S$, $O \circ K$, $O \circ O$, $f \circ a$ (arrow), $f \circ a$ (index).
- `STAR(a ... b)` : $\sigma_1 \times \sigma_2$, $\mathsf{MULS}(a \cdots b)$, $O_1 \otimes O_2$, $M_1 \star M_2$
- `ADDG(e1 ... en)` : $\mathsf{ADDS}(e1 \cdots en)$, $\mathsf{ADD}(e1 \cdots en)$
- `SSUM(i S e)` : `SUM(s FUN(i T e))`

References

1. de Bruijn, N.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. Indagationes Mathematicae (Proceedings) **75**(5), 381–392 (1972). https://doi.org/https://doi.org/10.1016/1385-7258(72)90034-0, https://www.sciencedirect.com/science/article/pii/1385725872900340
2. Dirac, P.A.M.: A new notation for quantum mechanics. In: Mathematical proceedings of the Cambridge philosophical society. vol. 35, pp. 416–418. Cambridge University Press (1939). https://doi.org/10.1017/S0305004100021162
3. Xu, Y., Barthe, G., Zhou, L.: Automating equational proofs in dirac notation. Proc. ACM Program. Lang. **9**(POPL) (Jan 2025). https://doi.org/10.1145/3704878, https://doi.org/10.1145/3704878

A Full Typing Rules

This section includes the full list of typing rules.

- Rules for a well-formed environment and context.

W-Empty
$$\frac{}{\mathcal{WF}([])[]}$$

W-AssumE-Index
$$\frac{\mathcal{WF}(E)[] \qquad x \notin E}{\mathcal{WF}(E; x : \mathsf{Index})[]}$$

W-AssumE-Term
$$\frac{E[] \vdash T : \mathsf{Type} \qquad x \notin E}{\mathcal{WF}(E; x : T)[]}$$

W-Def-Term
$$\frac{E[] \vdash t : T \qquad x \notin E}{\mathcal{WF}(E; x := t : T)[]}$$

W-AssumC-Index
$$\frac{\mathcal{WF}(E)[\varGamma]}{\mathcal{WF}(E)[\varGamma; x : \mathsf{Index}]}$$

W-AssumC-Term
$$\frac{E[\varGamma] \vdash T : \mathsf{Type}}{\mathcal{WF}(E)[\varGamma; x : T]}$$

- Rules for type indices.

Index-Var
$$\frac{\mathcal{WF}(E)[\varGamma] \qquad x : \mathsf{Index} \in E[\varGamma]}{E[\varGamma] \vdash x : \mathsf{Index}}$$

Index-Prod
$$\frac{E[\varGamma] \vdash \sigma : \mathsf{Index} \qquad E[\varGamma] \vdash \tau : \mathsf{Index}}{E[\varGamma] \vdash \sigma \times \tau : \mathsf{Index}}$$

Index-Qudit
$$\frac{\mathcal{WF}(E)[\varGamma]}{E[\varGamma] \vdash \mathsf{bool} : \mathsf{Index}}$$

- Rules for types.

Type-Lam
$$\frac{E[\varGamma] \vdash T : \mathsf{Type} \qquad E[\varGamma] \vdash U : \mathsf{Type}}{E[\varGamma] \vdash T \to U : \mathsf{Type}}$$

Type-Index
$$\frac{E[\varGamma; x : \mathsf{Index}] \vdash U : \mathsf{Type}}{E[\varGamma] \vdash \forall x.U : \mathsf{Type}}$$

Type-Basis
$$\frac{E[\varGamma] \vdash \sigma : \mathsf{Index}}{E[\varGamma] \vdash \mathsf{Basis}(\sigma) : \mathsf{Type}}$$

Type-Ket
$$\frac{E[\varGamma] \vdash \sigma : \mathsf{Index}}{E[\varGamma] \vdash \mathcal{K}(\sigma) : \mathsf{Type}}$$

Type-Bra
$$\frac{E[\varGamma] \vdash \sigma : \mathsf{Index}}{E[\varGamma] \vdash \mathcal{B}(\sigma) : \mathsf{Type}}$$

Type-Opt
$$\frac{E[\varGamma] \vdash \sigma : \mathsf{Index} \qquad E[\varGamma] \vdash \tau : \mathsf{Index}}{E[\varGamma] \vdash \mathcal{O}(\sigma, \tau) : \mathsf{Type}}$$

$$\textbf{Type-Scalar}\qquad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathcal{S} : \mathsf{Type}}$$

$$\textbf{Type-Set}\qquad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathsf{Set}(\sigma) : \mathsf{Type}}$$

$$\textbf{Type-Register}\qquad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathsf{Reg}(\sigma) : \mathsf{Type}}$$

$$\textbf{Type-Labelled}\qquad \frac{E[\Gamma] \vdash r : \mathsf{Reg}(\sigma_r) \text{ for all } r \text{ in } s_1 \text{ and } s_2}{E[\Gamma] \vdash \mathcal{D}(s_1, s_2) : \mathsf{Type}}$$

- Rules for variable and function typings. Here $U\{x/u\}$ means replacing the bound variable x with u in U.

$$\textbf{Term-Var}\qquad \frac{\begin{array}{l}\mathcal{WF}(E)[\Gamma] \\ (x : T) \in E[\Gamma] \text{ or } (x := t : T) \in E \text{ for some } t\end{array}}{E[\Gamma] \vdash x : T}$$

$$\textbf{Lam}\qquad \frac{E[\Gamma; x : T] \vdash t : U}{E[\Gamma] \vdash (\lambda x : T.t) : T \to U}$$

$$\textbf{Index}\qquad \frac{E[\Gamma; x : \mathsf{Index}] \vdash t : U}{E[\Gamma] \vdash (\mu x.t) : \forall x.U}$$

$$\textbf{App-Lam}\qquad \frac{E[\Gamma] \vdash t : U \to T \qquad E[\Gamma] \vdash u : U}{E[\Gamma] \vdash (t\ u) : T}$$

$$\textbf{App-Index}\qquad \frac{E[\Gamma] \vdash t : \forall x.U \qquad E[\Gamma] \vdash u : \mathsf{Index}}{E[\Gamma] \vdash (t\ u) : U\{x/u\}}$$

- Basis term typing rules.

$$\textbf{Basis-0}\qquad \frac{\mathcal{WF}(E[\Gamma])}{E[\Gamma] \vdash 0 : \mathsf{Basis}(\mathsf{bool})}$$

$$\textbf{Basis-1}\qquad \frac{\mathcal{WF}(E[\Gamma])}{E[\Gamma] \vdash 1 : \mathsf{Basis}(\mathsf{bool})}$$

$$\textbf{Basis-Pair}\qquad \frac{E[\Gamma] \vdash s : \mathsf{Basis}(\sigma) \qquad E[\Gamma] \vdash t : \mathsf{Basis}(\tau)}{E[\Gamma] \vdash (s, t) : \mathsf{Basis}(\sigma \times \tau)}$$

- Composition typing rules.

$$\textbf{Compo-SS}\qquad \frac{E[\Gamma] \vdash x : \mathcal{S} \qquad E[\Gamma] \vdash y : \mathcal{S}}{E[\Gamma] \vdash x \circ y : \mathcal{S}}$$

$$\textbf{Compo-SK}\qquad \frac{E[\Gamma] \vdash x : \mathcal{S} \qquad E[\Gamma] \vdash y : \mathcal{K}(\sigma)}{E[\Gamma] \vdash x \circ y : \mathcal{K}(\sigma)}$$

$$\textbf{Compo-SB}\qquad \frac{E[\Gamma] \vdash x : \mathcal{S} \qquad E[\Gamma] \vdash y : \mathcal{B}(\sigma)}{E[\Gamma] \vdash x \circ y : \mathcal{B}(\sigma)}$$

Compo-SO
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{S} \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{O}(\sigma, \tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{O}(\sigma, \tau)}$$

Compo-KS
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{K}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{S}}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{K}(\sigma)}$$

Compo-KK
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{K}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{K}(\tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{K}(\sigma \times \tau)}$$

Compo-KB
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{K}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{B}(\tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{O}(\sigma, \tau)}$$

Compo-BS
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{B}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{S}}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{B}(\sigma)}$$

Compo-BK
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{B}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{K}(\sigma)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{S}}$$

Compo-BB
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{B}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{B}(\tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{B}(\sigma \times \tau)}$$

Compo-BO
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{B}(\sigma) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{O}(\sigma, \tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{B}(\tau)}$$

Compo-OS
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{O}(\sigma, \tau) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{S}}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{O}(\sigma, \tau)}$$

Compo-OK
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{O}(\sigma, \tau) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{K}(\tau)}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{K}(\sigma)}$$

Compo-OO
$$\frac{E[\mathit{\Gamma}] \vdash x : \mathcal{O}(\sigma, \tau) \qquad E[\mathit{\Gamma}] \vdash y : \mathcal{O}(\sigma', \tau')}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{O}(\sigma \times \sigma', \tau \times \tau')}$$

Compo-DD
$$\frac{\begin{array}{cc} E[\mathit{\Gamma}] \vdash x : \mathcal{D}(s_1, s_1') & s_1 \cap s_2 \backslash s_1' = \emptyset \\ E[\mathit{\Gamma}] \vdash y : \mathcal{D}(s_2, s_2') & s_2' \cap s_1' \backslash s_2 = \emptyset \end{array}}{E[\mathit{\Gamma}] \vdash x \circ y : \mathcal{D}(s_1 \cup (s_2 \backslash s_1'), s_2' \cup (s_1' \backslash s_2))}$$

– Scalar term typing rules.

Sca-0
$$\frac{\mathcal{WF}(E)[\mathit{\Gamma}]}{E[\mathit{\Gamma}] \vdash 0 : \mathcal{S}}$$

Sca-1
$$\frac{\mathcal{WF}(E)[\mathit{\Gamma}]}{E[\mathit{\Gamma}] \vdash 1 : \mathcal{S}}$$

Sca-Delta
$$\frac{E[\mathit{\Gamma}] \vdash s : \mathsf{Basis}(\sigma) \qquad E[\mathit{\Gamma}] \vdash t : \mathsf{Basis}(\sigma)}{E[\mathit{\Gamma}] \vdash \delta_{s,t} : \mathcal{S}}$$

Sca-Add
$$\frac{E[\mathit{\Gamma}] \vdash a_i : \mathcal{S} \text{ for all } i}{E[\mathit{\Gamma}] \vdash a_1 + \cdots + a_n : \mathcal{S}}$$

Sca-Mul
$$\frac{E[\mathit{\Gamma}] \vdash a_i : \mathcal{S} \text{ for all } i}{E[\mathit{\Gamma}] \vdash a_1 \times \cdots \times a_n : \mathcal{S}}$$

$$\textbf{Sca-Conj}\quad \frac{E[\Gamma] \vdash a : \mathcal{S}}{E[\Gamma] \vdash a^* : \mathcal{S}}$$

$$\textbf{Sca-Dot}\quad \frac{E[\Gamma] \vdash B : \mathcal{B}(\sigma) \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash B \cdot K : \mathcal{S}}$$

$$\textbf{Sca-Sum}\quad \frac{E[\Gamma] \vdash s : \mathsf{Set}(\sigma) \qquad E[\Gamma] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{S}}{E[\Gamma] \vdash \sum_s f : \mathcal{S}}$$

- Ket term typing rules.

$$\textbf{Ket-0}\quad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathbf{0}_{\mathcal{K}}(\sigma) : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-Basis}\quad \frac{E[\Gamma] \vdash t : \mathsf{Basis}(\sigma)}{E[\Gamma] \vdash |t\rangle : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-Adj}\quad \frac{E[\Gamma] \vdash B : \mathcal{B}(\sigma)}{E[\Gamma] \vdash B^\dagger : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-Scr}\quad \frac{E[\Gamma] \vdash a : \mathcal{S} \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash a.K : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-Add}\quad \frac{E[\Gamma] \vdash K_i : \mathcal{K}(\sigma) \text{ for all } i}{E[\Gamma] \vdash K_1 + \cdots + K_n : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-MulK}\quad \frac{E[\Gamma] \vdash O : \mathcal{O}(\sigma, \tau) \qquad E[\Gamma] \vdash K : \mathcal{K}(\tau)}{E[\Gamma] \vdash O \cdot K : \mathcal{K}(\sigma)}$$

$$\textbf{Ket-Tsr}\quad \frac{E[\Gamma] \vdash K_1 : \mathcal{K}(\sigma) \qquad E[\Gamma] \vdash K_2 : \mathcal{K}(\tau)}{E[\Gamma] \vdash K_1 \otimes K_2 : \mathcal{K}(\sigma \times \tau)}$$

$$\textbf{Ket-Sum}\quad \frac{E[\Gamma] \vdash s : \mathsf{Set}(\sigma) \qquad E[\Gamma] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{K}(\tau)}{E[\Gamma] \vdash \sum_s f : \mathcal{K}(\tau)}$$

- Bra term typing rules.

$$\textbf{Bra-0}\quad \frac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathbf{0}_{\mathcal{B}}(\sigma) : \mathcal{B}(\sigma)}$$

$$\textbf{Bra-Basis}\quad \frac{E[\Gamma] \vdash t : \mathsf{Basis}(\sigma)}{E[\Gamma] \vdash \langle t| : \mathcal{B}(\sigma)}$$

$$\textbf{Bra-Adj}\quad \frac{E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash K^\dagger : \mathcal{B}(\sigma)}$$

$$\textbf{Bra-Scr}\quad \frac{E[\Gamma] \vdash a : \mathcal{S} \qquad E[\Gamma] \vdash B : \mathcal{B}(\sigma)}{E[\Gamma] \vdash a.B : \mathcal{B}(\sigma)}$$

$$\textbf{Bra-Add}\quad \frac{E[\Gamma] \vdash B_i : \mathcal{B}(\sigma) \text{ for all } i}{E[\Gamma] \vdash B_1 + \cdots + B_n : \mathcal{B}(\sigma)}$$

$$\textbf{Bra-MulB}\quad \frac{E[\Gamma] \vdash B : \mathcal{K}(\sigma) \qquad E[\Gamma] \vdash O : \mathcal{O}(\sigma, \tau)}{E[\Gamma] \vdash B \cdot O : \mathcal{B}(\tau)}$$

$$\textbf{Bra-Tsr}\qquad \frac{E[\varGamma] \vdash B_1 : \mathcal{B}(\sigma) \qquad E[\varGamma] \vdash B_2 : \mathcal{B}(\tau)}{E[\varGamma] \vdash B_1 \otimes B_2 : \mathcal{B}(\sigma \times \tau)}$$

$$\textbf{Bra-Sum}\qquad \frac{E[\varGamma] \vdash s : \mathsf{Set}(\sigma) \qquad E[\varGamma] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{B}(\tau)}{E[\varGamma] \vdash \sum_s f : \mathcal{B}(\tau)}$$

- Operator term typing rules.

$$\textbf{Opt-0}\qquad \frac{E[\varGamma] \vdash \sigma : \mathsf{Index} \qquad E[\varGamma] \vdash \tau : \mathsf{Index}}{E[\varGamma] \vdash \mathbf{0}_{\mathcal{O}}(\sigma, \tau) : \mathcal{O}(\sigma, \tau)}$$

$$\textbf{Opt-1}\qquad \frac{E[\varGamma] \vdash \sigma : \mathsf{Index}}{E[\varGamma] \vdash \mathbf{1}_{\mathcal{O}}(\sigma) : \mathcal{O}(\sigma, \sigma)}$$

$$\textbf{Opt-Adj}\qquad \frac{E[\varGamma] \vdash O : \mathcal{O}(\sigma, \tau)}{E[\varGamma] \vdash O^\dagger : \mathcal{O}(\tau, \sigma)}$$

$$\textbf{Opt-Scr}\qquad \frac{E[\varGamma] \vdash a : \mathcal{S} \qquad E[\varGamma] \vdash O : \mathcal{O}(\sigma, \tau)}{E[\varGamma] \vdash a.O : \mathcal{O}(\sigma, \tau)}$$

$$\textbf{Opt-Add}\qquad \frac{E[\varGamma] \vdash O_i : \mathcal{O}(\sigma, \tau) \text{ for all } i}{E[\varGamma] \vdash O_1 + \cdots + O_n : \mathcal{O}(\sigma, \tau)}$$

$$\textbf{Opt-Outer}\qquad \frac{E[\varGamma] \vdash K : \mathcal{K}(\sigma) \qquad E[\varGamma] \vdash B : \mathcal{B}(\tau)}{E[\varGamma] \vdash K \cdot B : \mathcal{O}(\sigma, \tau)}$$

$$\textbf{Opt-Mulo}\qquad \frac{E[\varGamma] \vdash O_1 : \mathcal{O}(\sigma, \tau) \qquad E[\varGamma] \vdash O_2 : \mathcal{O}(\tau, \rho)}{E[\varGamma] \vdash O_1 \cdot O_2 : \mathcal{O}(\sigma, \rho)}$$

$$\textbf{Opt-Tsr}\qquad \frac{E[\varGamma] \vdash O_1 : \mathcal{O}(\sigma_1, \tau_1) \qquad E[\varGamma] \vdash O_2 : \mathcal{O}(\sigma_2, \tau_2)}{E[\varGamma] \vdash O_1 \otimes O_2 : \mathcal{O}(\sigma_1 \times \sigma_2, \tau_1 \times \tau_2)}$$

$$\textbf{Opt-Sum}\qquad \frac{E[\varGamma] \vdash s : \mathsf{Set}(\sigma) \qquad E[\varGamma] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{O}(\tau, \rho)}{E[\varGamma] \vdash \sum_s f : \mathcal{O}(\tau, \rho)}$$

- Set term typing rules.

$$\textbf{Set-U}\qquad \frac{E[\varGamma] \vdash \sigma : \mathsf{Index}}{E[\varGamma] \vdash \mathbf{U}(\sigma) : \mathsf{Set}(\sigma)}$$

$$\textbf{Set-Prod}\qquad \frac{E[\varGamma] \vdash A : \mathsf{Set}(\sigma) \qquad E[\varGamma] \vdash B : \mathsf{Set}(\tau)}{E[\varGamma] \vdash A \star B : \mathsf{Set}(\sigma \times \tau)}$$

- Register term typing rules.

$$\textbf{Reg-Var}\qquad \frac{\mathcal{WF}(E[\varGamma]) \qquad r : \mathsf{Reg}(\sigma) \in E}{E[\varGamma] \vdash r : \mathsf{Reg}(\sigma)}$$

$$\textbf{Reg-Pair}\qquad \frac{\begin{array}{l} E[\varGamma] \vdash R : \mathsf{Reg}(\sigma) \\ E[\varGamma] \vdash Q : \mathsf{Reg}(\tau) \end{array} \qquad \mathsf{var}(R) \cap \mathsf{var}(Q) = \emptyset}{E[\varGamma] \vdash (R, Q) : \mathsf{Reg}(\sigma \times \tau)}$$

– Typing rules for labelled Dirac notations.

$$\textbf{L-Basis-Ket}\qquad \frac{r : \mathsf{Reg}(\sigma) \in E \qquad E[\mathit{\Gamma}] \vdash i : \mathsf{Basis}(\sigma)}{E[\mathit{\Gamma}] \vdash |i\rangle_r : \mathcal{D}(\{r\}, \emptyset)}$$

$$\textbf{L-Basis-Bra}\qquad \frac{r : \mathsf{Reg}(\sigma) \in E \qquad E[\mathit{\Gamma}] \vdash i : \mathsf{Basis}(\sigma)}{E[\mathit{\Gamma}] \vdash {}_r\langle i| : \mathcal{D}(\emptyset, \{r\})}$$

$$\textbf{L-Ket}\qquad \frac{E[\mathit{\Gamma}] \vdash R : \mathsf{Reg}(\sigma) \qquad E[\mathit{\Gamma}] \vdash K : \mathcal{K}(\sigma)}{E[\mathit{\Gamma}] \vdash K_R : \mathcal{D}(\mathsf{var}R, \emptyset)}$$

$$\textbf{L-Bra}\qquad \frac{E[\mathit{\Gamma}] \vdash R : \mathsf{Reg}(\sigma) \qquad E[\mathit{\Gamma}] \vdash B : \mathcal{B}(\sigma)}{E[\mathit{\Gamma}] \vdash B_R : \mathcal{D}(\emptyset, \mathsf{var}R)}$$

$$\textbf{L-Opt}\qquad \frac{\begin{array}{c} E[\mathit{\Gamma}] \vdash R_1 : \mathsf{Reg}(\sigma_1) \\ E[\mathit{\Gamma}] \vdash R_2 : \mathsf{Reg}(\sigma_2) \end{array} \qquad E[\mathit{\Gamma}] \vdash O : \mathcal{O}(\sigma_1, \sigma_2)}{E[\mathit{\Gamma}] \vdash O_{R_1;R_2} : \mathcal{D}(\mathsf{var}R_1, \mathsf{var}R_2)}$$

$$\textbf{L-Conj}\qquad \frac{E[\mathit{\Gamma}] \vdash D : \mathcal{D}(s_1, s_2)}{E[\mathit{\Gamma}] \vdash D^\dagger : \mathcal{D}(s_2, s_1)}$$

$$\textbf{L-Scl}\qquad \frac{E[\mathit{\Gamma}] \vdash S : \mathcal{S} \qquad E[\mathit{\Gamma}] \vdash D : \mathcal{D}(s_1, s_2)}{E[\mathit{\Gamma}] \vdash S.D : \mathcal{D}(s_1, s_2)}$$

$$\textbf{L-Add}\qquad \frac{E[\mathit{\Gamma}] \vdash D_i : \mathcal{D}(s_1, s_2) \quad \text{forall } i}{E[\mathit{\Gamma}] \vdash D_1 + \cdots + D_n : \mathcal{D}(s_1, s_2)}$$

$$\textbf{L-Tsr}\qquad \frac{E[\mathit{\Gamma}] \vdash D_i : \mathcal{D}(s_i, s_i') \qquad \bigcap_i s_i = \emptyset \qquad \bigcap_i s_i' = \emptyset}{E[\mathit{\Gamma}] \vdash D_1 \otimes \cdots \otimes D_i : \mathcal{D}(\bigcup_i s_i, \bigcup_i s_i')}$$

$$\textbf{L-Dot}\qquad \frac{\begin{array}{cc} E[\mathit{\Gamma}] \vdash D_1 : \mathcal{D}(s_1, s_1') & s_1 \cap s_2 \backslash s_1' = \emptyset \\ E[\mathit{\Gamma}] \vdash D_2 : \mathcal{D}(s_2, s_2') & s_2' \cap s_1' \backslash s_2 = \emptyset \end{array}}{E[\mathit{\Gamma}] \vdash D_1 \cdot D_2 : \mathcal{D}(s_1 \cup (s_2 \backslash s_1'), s_2' \cup (s_1' \backslash s_2))}$$

$$\textbf{L-Sum}\qquad \frac{E[\mathit{\Gamma}] \vdash s : \mathsf{Set}(\sigma) \qquad E[\mathit{\Gamma}] \vdash f : \mathsf{Basis}(\sigma) \to \mathcal{D}(s_1, s_2)}{E[\mathit{\Gamma}] \vdash \sum_s f : \mathcal{D}(s_1, s_2)}$$

B Axiomatic Semantics

The full list of equational axioms are provided below.

(Ax-Scalar) $(B \cdot K)^* = K^\dagger \cdot B^\dagger$

(Ax-Delta) $\delta_{s,t}^* = \delta_{s,t} \qquad \langle s| \cdot |t\rangle = \delta_{s,t}$

$\delta_{s,s} = 1 \qquad s \neq t \vdash \delta_{s,t} = 0 \qquad \delta_{s,t} = \delta_{t,s}$

(Ax-Linear) $\mathbf{0} + D = D \qquad D_1 + D_2 = D_2 + D_1$

$(D_1 + D_2) + D_3 = D_1 + (D_2 + D_3)$

$0.D = \mathbf{0} \qquad a.\mathbf{0} = \mathbf{0} \qquad 1.D = D$

$a.(b.D) = (a \times b).D \qquad (a + b).D = a.D + b.D$

$a.(D_1 + D_2) = a.D_1 + a.D_2$

(Ax-Bilinear) $D \cdot \mathbf{0} = \mathbf{0} \qquad D_1 \cdot (a.D_2) = a.(D_1 \cdot D_2)$

$D_0 \cdot (D_1 + D_2) = D_0 \cdot D_1 + D_0 \cdot D_2$

$\mathbf{0} \cdot D = \mathbf{0} \qquad (a.D_1) \cdot D_2 = a.(D_1 \cdot D_2)$

$(D_1 + D_2) \cdot D_0 = D_1 \cdot D_0 + D_2 \cdot D_0$

$D \otimes \mathbf{0} = \mathbf{0} \qquad D_1 \otimes (a.D_2) = a.(D_1 \otimes D_2)$

$D_0 \otimes (D_1 + D_2) = D_0 \otimes D_1 + D_0 \otimes D_2$

$\mathbf{0} \otimes D = \mathbf{0} \qquad (a.D_1) \otimes D_2 = a.(D_1 \otimes D_2)$

$(D_1 + D_2) \otimes D_0 = D_1 \otimes D_0 + D_2 \otimes D_0$

(Ax-Adjoint) $\mathbf{0}^\dagger = \mathbf{0} \qquad (D^\dagger)^\dagger = D \qquad (a.D)^\dagger = a^*.(D^\dagger)$

$(D_1 + D_2)^\dagger = D_1^\dagger + D_2^\dagger$

$(D_1 \cdot D_2)^\dagger = D_2^\dagger \cdot D_1^\dagger \qquad (D_1 \otimes D_2)^\dagger = D_1^\dagger \otimes D_2^\dagger$

(Ax-Comp) $D_0 \cdot (D_1 \cdot D_2) = (D_0 \cdot D_1) \cdot D_2$

$(D_1 \otimes D_2) \cdot (D_3 \otimes D_4) = (D_1 \cdot D_3) \otimes (D_2 \cdot D_4)$

$(K_1 \cdot B) \cdot K_2 = (B \cdot K_2).K_1 \qquad B_1 \cdot (K \cdot B_2) = (B_1 \cdot K).B_2$

$(B_1 \otimes B_2) \cdot (K_1 \otimes K_2) = (B_1 \cdot K_1) \times (B_2 \cdot K_2)$

(Ax-Ground) $\mathbf{1}_\mathcal{O}^\dagger = \mathbf{1}_\mathcal{O} \qquad \mathbf{1}_\mathcal{O} \cdot D = D \qquad \mathbf{1}_\mathcal{O} \otimes \mathbf{1}_\mathcal{O} = \mathbf{1}_\mathcal{O}$

$|t\rangle^\dagger = \langle t| \qquad |s\rangle \otimes |t\rangle = |(s, t)\rangle$

[YX] : to be continued

C Rewriting Rules

This section includes all the rewriting rules used in the system. Related rules are collected in the same table.

Table 1: Reductions for the definitions and function applications.

Rule	Description
BETA-ARROW	$((\lambda x : T.t)\ u) \triangleright t\{x/u\}$
BETA-INDEX	$((\mu x.t)\ u) \triangleright t\{x/u\}$
DELTA	$(c := t : T) \in E \Rightarrow c \triangleright t$

Table 2: The special to flatten all AC symbols within one call.

Rule	Description
R-FLATTEN	$a_1 + \cdots + (b_1 + \cdots + b_m) + \cdots + a_n$
	$\triangleright a_1 + \cdots + b_1 + \cdots + b_m + \cdots + a_n$
	$a_1 \times \cdots \times (b_1 \times \cdots \times b_m) \times \cdots \times a_n$
	$\triangleright a_1 \times \cdots \times b_1 \times \cdots \times b_m \times \cdots \times a_n$
	$X_1 + \cdots + (X'_1 + \cdots + X'_m) + \cdots + X_n$
	$\triangleright X_1 + \cdots + X'_1 + \cdots + X'_m + \cdots + X_n$

Table 3: Rules for scalar symbols.

Rule	Description
R-CONJ5	$\delta_{s,t}^* \triangleright \delta_{s,t}$
R-CONJ6	$(B \cdot K)^* \triangleright K^\dagger \cdot B^\dagger$
R-DOT0	$\mathbf{0}_{\mathcal{B}}(\sigma) \cdot K \triangleright 0$
R-DOT1	$B \cdot \mathbf{0}_{\mathcal{K}}(\sigma) \triangleright 0$
R-DOT2	$(a.B) \cdot K \triangleright a \times (B \cdot K)$
R-DOT3	$B \cdot (a.K) \triangleright a \times (B \cdot K)$
R-DOT4	$(B_1 + \cdots + B_n) \cdot K \triangleright B_1 \cdot K + \cdots + B_n \cdot K$
R-DOT5	$B \cdot (K_1 + \cdots + K_n) \triangleright B \cdot K_1 + \cdots + B \cdot K_n$
R-DOT6	$\langle s \vert \cdot \vert t\rangle \triangleright \delta_{s,t}$
R-DOT7	$(B_1 \otimes B_2) \cdot \vert(s,t)\rangle \triangleright (B_1 \cdot \vert s\rangle) \times (B_2 \cdot \vert t\rangle)$
R-DOT8	$\langle (s,t)\vert \cdot (K_1 \otimes K_2) \triangleright (\langle s\vert \cdot K_1) \times (\langle t\vert \cdot K_2)$
R-DOT9	$(B_1 \otimes B_2) \cdot (K_1 \otimes K_2) \triangleright (B_1 \cdot K_1) \times (B_2 \cdot K_2)$
R-DOT10	$(B \cdot O) \cdot K \triangleright B \cdot (O \cdot K)$
R-DOT11	$\langle (s,t)\vert \cdot ((O_1 \otimes O_2) \cdot K) \triangleright ((\langle s\vert \cdot O_1) \otimes (\langle t\vert \cdot O_2)) \cdot K$
R-DOT12	$(B_1 \otimes B_2) \cdot ((O_1 \otimes O_2) \cdot K) \triangleright ((B_1 \cdot O_1) \otimes (B_2 \cdot O_2)) \cdot K$

Rule	Description
R-DELTA0	$\delta_{a,a} \ \triangleright\ 1$
R-DELTA1	$\delta_{(a,b),(c,d)} \ \triangleright\ \delta_{a,c} \times \delta_{b,d}$

Table 4: Rules for scaling.

Rule	Description
R-SCR0	$1.X \ \triangleright\ X$
R-SCR1	$a.(b.X) \ \triangleright\ (a \times b).X$
R-SCR2	$a.(X_1 + \cdots + X_n) \ \triangleright\ a.X_1 + \cdots + a.X_n$
R-SCRK0	$K : \mathcal{K}(\sigma) \Rightarrow 0.K \ \triangleright\ \mathbf{0}_{\mathcal{K}}(\sigma)$
R-SCRK1	$a.\mathbf{0}_{\mathcal{K}}(\sigma) \ \triangleright\ \mathbf{0}_{\mathcal{K}}(\sigma)$
R-SCRB0	$B : \mathcal{B}(\sigma) \Rightarrow 0.B \ \triangleright\ \mathbf{0}_{\mathcal{B}}(\sigma)$
R-SCRB1	$a.\mathbf{0}_{\mathcal{B}}(\sigma) \ \triangleright\ \mathbf{0}_{\mathcal{B}}(\sigma)$
R-SCRO0	$O : \mathcal{O}(\sigma, \tau) \Rightarrow 0.O \ \triangleright\ \mathbf{0}_{\mathcal{O}}(\sigma, \tau)$
R-SCRO1	$a.\mathbf{0}_{\mathcal{O}}(\sigma, \tau) \ \triangleright\ \mathbf{0}_{\mathcal{O}}(\sigma, \tau)$

Table 5: Rules for addition.

Rule	Description
R-ADDID	$+(X) \ \triangleright\ X$
R-ADD0	$Y_1 + \cdots + X + \cdots + X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + \cdots + (1+1).X$
R-ADD1	$Y_1 + \cdots + X + \cdots + a.X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (1+a).X$
R-ADD2	$Y_1 + \cdots + a.X + \cdots + X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (a+1).X$
R-ADD3	$Y_1 + \cdots + a.X + \cdots + b.X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (a+b).X$
R-ADDK0	$K_1 + \cdots + \mathbf{0}_{\mathcal{K}}(\sigma) + \cdots + K_n \ \triangleright K_1 + \cdots + K_n$
R-ADDB0	$B_1 + \cdots + \mathbf{0}_{\mathcal{B}}(\sigma) + \cdots + B_n \ \triangleright\ B_1 + \cdots + B_n$
R-ADDO0	$O_1 + \cdots + \mathbf{0}_{\mathcal{O}}(\sigma, \tau) + \cdots + O_n \ \triangleright\ O_1 + \cdots + O_n$

Table 6: Rules for adjoint.

Rule	Description
R-ADJ0	$(X^\dagger)^\dagger \ \triangleright\ X$
R-ADJ1	$(a.X)^\dagger \ \triangleright\ (a^*).(X^\dagger)$
R-ADJ2	$(X_1 + \cdots + X_n)^\dagger \ \triangleright\ X_1^\dagger + \cdots + X_n^\dagger$
R-ADJ3	$(X \otimes Y)^\dagger \ \triangleright\ X^\dagger \otimes Y^\dagger$

Rule	Description
R-ADJK0	$\mathbf{0}_{\mathcal{B}}(\sigma)^\dagger \triangleright \mathbf{0}_{\mathcal{K}}(\sigma)$
R-ADJK1	$\langle t\|^\dagger \triangleright \|t\rangle$
R-ADJK2	$(B \cdot O)^\dagger \triangleright O^\dagger \cdot B^\dagger$
R-ADJB0	$\mathbf{0}_{\mathcal{K}}(\sigma)^\dagger \triangleright \mathbf{0}_{\mathcal{B}}(\sigma)$
R-ADJB1	$\|t\rangle^\dagger \triangleright \langle t\|$
R-ADJB2	$(O \cdot K)^\dagger \triangleright K^\dagger \cdot O^\dagger$
R-ADJO0	$\mathbf{0}_{\mathcal{O}}(\sigma, \tau)^\dagger \triangleright \mathbf{0}_{\mathcal{O}}(\tau, \sigma)$
R-ADJO1	$\mathbf{1}_{\mathcal{O}}(\sigma)^\dagger \triangleright \mathbf{1}_{\mathcal{O}}(\sigma)$
R-ADJO2	$(K \cdot B)^\dagger \triangleright B^\dagger \cdot K^\dagger$
R-ADJO3	$(O_1 \cdot O_2)^\dagger \triangleright O_2^\dagger \cdot O_1^\dagger$

Table 7: Rules for tensor product.

Rule	Description
R-TSR0	$(a.X_1) \otimes X_2 \triangleright a.(X_1 \otimes X_2)$
R-TSR1	$X_1 \otimes (a.X_2) \triangleright a.(X_1 \otimes X_2)$
R-TSR2	$(X_1 + \cdots + X_n) \otimes X' \triangleright X_1 \otimes X' + \cdots + X_n \otimes X'$
R-TSR3	$X' \otimes (X_1 + \cdots + X_n) \triangleright X' \otimes X_1 + \cdots + X' \otimes X_n$
R-TSRK0	$K : \mathcal{K}(\tau) \Rightarrow \mathbf{0}_{\mathcal{K}}(\sigma) \otimes K \triangleright \mathbf{0}_{\mathcal{K}}(\sigma \times \tau)$
R-TSRK1	$K : \mathcal{K}(\tau) \Rightarrow K \otimes \mathbf{0}_{\mathcal{K}}(\sigma) \triangleright \mathbf{0}_{\mathcal{K}}(\tau \times \sigma)$
R-TSRK2	$\|s\rangle \otimes \|t\rangle \triangleright \|(s,t)\rangle$
R-TSRB0	$B : \mathcal{B}(\tau) \Rightarrow \mathbf{0}_{\mathcal{B}}(\sigma) \otimes B \triangleright \mathbf{0}_{\mathcal{B}}(\sigma \times \tau)$
R-TSRB1	$B : \mathcal{B}(\tau) \Rightarrow B \otimes \mathbf{0}_{\mathcal{B}}(\sigma) \triangleright \mathbf{0}_{\mathcal{B}}(\tau \times \sigma)$
R-TSRB2	$\langle s\| \otimes \langle t\| \triangleright \langle (s,t)\|$
R-TSRO0	$O : \mathcal{O}(\sigma, \tau) \Rightarrow O \otimes \mathbf{0}_{\mathcal{O}}(\sigma', \tau') \triangleright \mathbf{0}_{\mathcal{O}}(\sigma \times \sigma', \tau \times \tau')$
R-TSRO1	$O : \mathcal{O}(\sigma, \tau) \Rightarrow \mathbf{0}_{\mathcal{O}}(\sigma', \tau') \otimes O \triangleright \mathbf{0}_{\mathcal{O}}(\sigma' \times \sigma, \tau' \times \tau)$
R-TSRO2	$\mathbf{1}_{\mathcal{O}}(\sigma) \otimes \mathbf{1}_{\mathcal{O}}(\tau) \triangleright \mathbf{1}_{\mathcal{O}}(\sigma \times \tau)$
R-TSRO3	$(K_1 \cdot B_1) \otimes (K_2 \cdot B_2) \triangleright (K_1 \otimes K_2) \cdot (B_1 \otimes B_2)$

Table 8: Rule for $O \cdot K$.

Rule	Description
R-MULK0	$\mathbf{0}_{\mathcal{O}}(\sigma, \tau) \cdot K \triangleright \mathbf{0}_{\mathcal{K}}(\sigma)$
R-MULK1	$O : \mathcal{O}(\sigma, \tau) \Rightarrow O \cdot \mathbf{0}_{\mathcal{K}}(\tau) \triangleright \mathbf{0}_{\mathcal{K}}(\sigma)$
R-MULK2	$\mathbf{1}_{\mathcal{O}}(\sigma) \cdot K \triangleright K$
R-MULK3	$(a.O) \cdot K \triangleright a.(O \cdot K)$

Rule	**Description**
R-MULK4	$O \cdot (a.K) \ \triangleright \ a.(O \cdot K)$
R-MULK5	$(O_1 + \cdots + O_n) \cdot K \ \triangleright \ O_1 \cdot K + \cdots + O_n \cdot K$
R-MULK6	$O \cdot (K_1 + \cdots + K_n) \ \triangleright \ O \cdot K_1 + \cdots + O \cdot K_n$
R-MULK7	$(K_1 \cdot B) \cdot K_2 \ \triangleright \ (B \cdot K_2).K_1$
R-MULK8	$(O_1 \cdot O_2) \cdot K \ \triangleright \ O_1 \cdot (O_2 \cdot K)$
R-MULK9	$(O_1 \otimes O_2) \cdot ((O'_1 \otimes O'_2) \cdot K) \ \triangleright \ ((O_1 \cdot O'_1) \otimes (O_2 \cdot O'_2)) \cdot K$
R-MULK10	$(O_1 \otimes O_2) \cdot \lvert (s,t) \rangle \ \triangleright \ (O_1 \cdot \lvert s \rangle) \otimes (O_2 \cdot \lvert t \rangle)$
R-MULK11	$(O_1 \otimes O_2) \cdot (K_1 \otimes K_2) \ \triangleright \ (O_1 \cdot K_1) \otimes (O_2 \cdot K_2)$

Table 9: Rule for $B \cdot O$.

Rule	**Description**
R-MULB0	$B \cdot \mathbf{0}_{\mathcal{O}}(\sigma, \tau) \ \triangleright \ \mathbf{0}_{\mathcal{B}}(\tau)$
R-MULB1	$O : \mathcal{O}(\sigma, \tau) \Rightarrow \mathbf{0}_{\mathcal{B}}(\sigma) \cdot O \ \triangleright \ \mathbf{0}_{\mathcal{B}}(\tau)$
R-MULB2	$B \cdot \mathbf{1}_{\mathcal{O}}(\sigma) \ \triangleright B$
R-MULB3	$(a.B) \cdot O \ \triangleright \ a.(B \cdot O)$
R-MULB4	$B \cdot (a.O) \ \triangleright \ a.(B \cdot O)$
R-MULB5	$(B_1 + \cdots + B_n) \cdot O \ \triangleright \ B_1 \cdot O + \cdots + B_n \cdot O$
R-MULB6	$B \cdot (O_1 + \cdots + O_n) \ \triangleright \ B \cdot O_1 + \cdots + B \cdot O_n$
R-MULB7	$B_1 \cdot (K \cdot B_2) \ \triangleright \ (B_1 \cdot K).B_2$
R-MULB8	$B \cdot (O_1 \cdot O_2) \ \triangleright \ (B \cdot O_1) \cdot O_2$
R-MULB9	$(B \cdot (O'_1 \otimes O'_2)) \cdot (O_1 \otimes O_2) \ \triangleright \ B \cdot ((O'_1 \otimes O'_2) \cdot (O_1 \otimes O_2))$
R-MULB10	$\langle (s,t) \rvert \cdot (O_1 \otimes O_2) \ \triangleright \ (\langle s \rvert \cdot O_1) \otimes (\langle t \rvert \cdot O_2)$
R-MULB11	$(B_1 \otimes B_2) \cdot (O_1 \otimes O_2) \ \triangleright \ (B_1 \cdot O_1) \otimes (B_2 \cdot O_2)$

Table 10: Rules for $K \cdot B$.

Rule	**Description**
R-OUTER0	$B : \mathcal{B}(\tau) \Rightarrow \mathbf{0}_{\mathcal{K}}(\sigma) \cdot B \ \triangleright \ \mathbf{0}_{\mathcal{O}}(\sigma, \tau)$
R-OUTER1	$K : \mathcal{K}(\sigma) \Rightarrow K \cdot \mathbf{0}_{\mathcal{B}}(\tau) \ \triangleright \ \mathbf{0}_{\mathcal{O}}(\sigma, \tau)$
R-OUTER2	$(a.K) \cdot B \ \triangleright \ a.(K \cdot B)$
R-OUTER3	$K \cdot (a.B) \ \triangleright \ a.(K \cdot B)$
R-OUTER4	$(K_1 + \cdots + K_n) \cdot B \ \triangleright \ K_1 \cdot B + \cdots + K_n \cdot B$
R-OUTER5	$K \cdot (B_1 + \cdots + B_n) \ \triangleright \ K \cdot B_1 + \cdots + K \cdot B_n$

Table 11: Rules for $O_1 \cdot O_2$.

Rule	Description
R-MULO0	$O : \mathcal{O}(\tau, \rho) \Rightarrow \mathbf{0}_{\mathcal{O}}(\sigma, \tau) \cdot O \ \triangleright \ \mathbf{0}_{\mathcal{O}}(\sigma, \rho)$
R-MULO1	$O : \mathcal{O}(\sigma, \tau) \Rightarrow O \cdot \mathbf{0}_{\mathcal{O}}(\tau, \rho) \ \triangleright \ \mathbf{0}_{\mathcal{O}}(\sigma, \rho)$
R-MULO2	$\mathbf{1}_{\mathcal{O}}(\sigma) \cdot O \ \triangleright \ O$
R-MULO3	$O \cdot \mathbf{1}_{\mathcal{O}}(\sigma) \ \triangleright \ O$
R-MULO4	$(K \cdot B) \cdot O \ \triangleright \ K \cdot (B \cdot O)$
R-MULO5	$O \cdot (K \cdot B) \ \triangleright \ (O \cdot K) \cdot B$
R-MULO6	$(a.O_1) \cdot O_2 \ \triangleright \ a.(O_1 \cdot O_2)$
R-MULO7	$O_1 \cdot (a.O_2) \ \triangleright \ a.(O_1 \cdot O_2)$
R-MULO8	$(O_1 + \cdots + O_n) \cdot O' \ \triangleright \ O_1 \cdot O' + \cdots + O_n \cdot O'$
R-MULO9	$O' \cdot (O_1 + \cdots + O_n) \ \triangleright \ O' \cdot O_1 + \cdots + O' \cdot O_n$
R-MULO10	$(O_1 \cdot O_2) \cdot O_3 \ \triangleright \ O_1 \cdot (O_2 \cdot O_3)$
R-MULO11	$(O_1 \otimes O_2) \cdot (O'_1 \otimes O'_2) \ \triangleright \ (O_1 \cdot O'_1) \otimes (O_2 \cdot O'_2)$
R-MULO12	$(O_1 \otimes O_2) \cdot ((O'_1 \otimes O'_2) \cdot O_3) \ \triangleright \ ((O_1 \cdot O'_1) \otimes (O_2 \cdot O'_2)) \cdot O_3$

Table 12: Rules for sets.

Rule	Description
R-SET0	$\mathbf{U}(\sigma) \star \mathbf{U}(\tau) \ \triangleright \ \mathbf{U}(\sigma \times \tau)$

Table 13: Rules for sum operators.

Rule	Description
R-SUM-CONST0	$\sum_{x \in s} 0 \ \triangleright \ 0$
R-SUM-CONST1	$\sum_{x \in s} \mathbf{0}_{\mathcal{K}}(\sigma) \ \triangleright \ \mathbf{0}_{\mathcal{K}}(\sigma)$
R-SUM-CONST2	$\sum_{x \in s} \mathbf{0}_{\mathcal{B}}(\sigma) \ \triangleright \ \mathbf{0}_{\mathcal{B}}(\sigma)$
R-SUM-CONST3	$\sum_{x \in s} \mathbf{0}_{\mathcal{O}}(\sigma, \tau) \ \triangleright \ \mathbf{0}_{\mathcal{O}}(\sigma, \tau)$
R-SUM-CONST4	$\mathbf{1}_{\mathcal{O}}(\sigma) \ \triangleright \ \sum_{i \in \mathbf{U}(\sigma)} \vert i \rangle \cdot \langle i \vert$

Table 14: Rules for eliminating $\delta_{s,t}$. These rules match the δ operator modulo the commutativity of its arguments.

Rule	Description
R-SUM-ELIM0	i free in $t \Rightarrow \sum_{i \in \mathbf{U}(\sigma)} \sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} \delta_{i,t}$ $\triangleright \ \sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} 1$

Rule	Description
R-SUM-ELIM1	i free in $t \Rightarrow$ $\sum_{i\in\mathbf{U}(\sigma)}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\times\cdots\times\delta_{i,t}\times\cdots\times a_n)$ $\triangleright \sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n} a_1\{i/t\}\times\cdots\times a_n\{i/t\}$
R-SUM-ELIM2	i free in $t \Rightarrow \sum_{i\in\mathbf{U}(\sigma)}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(\delta_{i,t}.A)$ $\triangleright \sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n} A\{i/t\}$
R-SUM-ELIM3	i free in $t \Rightarrow$ $\sum_{i\in\mathbf{U}(\sigma)}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\times\cdots\times\delta_{i,t}\times\cdots\times a_n).A$ $\triangleright \sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\{i/t\}\times\cdots\times a_n\{i/t\}).A\{i/t\}$
R-SUM-ELIM4	$\sum_{i\in M}\sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}\delta_{i,j}$ $\triangleright \sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n} 1$
R-SUM-ELIM5	$\sum_{i\in M}\sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\times\cdots\times\delta_{i,j}\times\cdots\times a_n)$ $\triangleright \sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\{j/i\}\times\cdots\times a_n\{j/i\})$
R-SUM-ELIM6	$\sum_{i\in M}\sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(\delta_{i,j}.A)$ $\triangleright \sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n} A\{j/i\}$
R-SUM-ELIM7	$\sum_{i\in M}\sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\times\cdots\times\delta_{i,j}\times\cdots\times a_n).A$ $\triangleright \sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}(a_1\{j/i\}\times\cdots\times a_n\{j/i\}).A\{j/i\}$
R-SUM-ELIM8	$\sum_{i\in M}\sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}((a_1\times\cdots\times\delta_{i,j}\times\cdots\times a_n)+$ $\cdots+(b_1\times\cdots\times\delta_{i,j}\times\cdots\times b_n)).A$ $\triangleright \sum_{j\in M}\sum_{k_1\in s_1}\cdots\sum_{k_n\in s_n}((a_1\{j/i\}\times\cdots\times a_n\{j/i\})+$ $\cdots+(b_1\{j/i\}\times\cdots\times b_n\{j/i\})).A\{j/i\}$

Table 15: Rules for pushing terms into sum operators. Because we apply type checking on variables, and stick to unique bound variables, these operations are always sound.

Rule	Description
R-SUM-PUSH0	$b_1\times\cdots\times(\sum_{i\in M} a)\times\cdots\times b_n$ $\triangleright \sum_{i\in M}(b_1\times\cdots\times a\times\cdots\times b_n)$
R-SUM-PUSH1	$(\sum_{i\in M} a)^* \ \triangleright \ \sum_{i\in M} a^*$

Rule	Description
R-SUM-PUSH2	$(\sum_{i\in M} X)^\dagger \ \triangleright\ \sum_{i\in M} X^\dagger$
R-SUM-PUSH3	$a.(\sum_{i\in M} X) \ \triangleright\ \sum_{i\in M}(a.X)$
R-SUM-PUSH4	$(\sum_{i\in M} a).X \ \triangleright\ \sum_{i\in M}(a.X)$
R-SUM-PUSH5	$(\sum_{i\in M} B)\cdot K \ \triangleright\ \sum_{i\in M}(B\cdot K)$
R-SUM-PUSH6	$(\sum_{i\in M} O)\cdot K \ \triangleright\ \sum_{i\in M}(O\cdot K)$
R-SUM-PUSH7	$(\sum_{i\in M} B)\cdot O \ \triangleright\ \sum_{i\in M}(B\cdot O)$
R-SUM-PUSH8	$(\sum_{i\in M} K)\cdot B \ \triangleright\ \sum_{i\in M}(K\cdot B)$
R-SUM-PUSH9	$(\sum_{i\in M} O_1)\cdot O_2 \ \triangleright\ \sum_{i\in M}(O_1\cdot O_2)$
R-SUM-PUSH10	$B\cdot(\sum_{i\in M} K) \ \triangleright\ \sum_{i\in M}(B\cdot K)$
R-SUM-PUSH11	$O\cdot(\sum_{i\in M} K) \ \triangleright\ \sum_{i\in M}(O\cdot K)$
R-SUM-PUSH12	$B\cdot(\sum_{i\in M} O) \ \triangleright\ \sum_{i\in M}(B\cdot O)$
R-SUM-PUSH13	$K\cdot(\sum_{i\in M} B) \ \triangleright\ \sum_{i\in M}(K\cdot B)$
R-SUM-PUSH14	$O_1\cdot(\sum_{i\in M} O_2) \ \triangleright\ \sum_{i\in M}(O_1\cdot O_2)$
R-SUM-PUSH15	$(\sum_{i\in M} X_1)\otimes X_2 \ \triangleright\ \sum_{i\in M}(X_1\otimes X_2)$
R-SUM-PUSH16	$X_1\otimes(\sum_{i\in M} X_2) \ \triangleright\ \sum_{i\in M}(X_1\otimes X_2)$

Table 16: Rules for addition and index in sum.

Rule	Description
R-SUM-ADDS0	$\sum_{i\in M}(a_1+\cdots+a_n) \ \triangleright\ (\sum_{i\in M} a_1)+\cdots+(\sum_{i\in M} a_n)$
R-SUM-ADD0	$\sum_{i\in M}(X_1+\cdots+X_n) \ \triangleright\ (\sum_{i\in M} X_1)+\cdots+(\sum_{i\in M} X_n)$
R-SUM-INDEX0	$\sum_{i\in\mathbf{U}(\sigma\times\tau)} A \ \triangleright\ \sum_{j\in\mathbf{U}(\sigma)}\sum_{k\in\mathbf{U}(\tau)} A\{i/(j,k)\}$
R-SUM-INDEX1	$\sum_{i\in M_1\star M_2} A \ \triangleright\ \sum_{j\in M_1}\sum_{k\in M_2} A\{i/(j,k)\}$

Table 17: Rules for bool index.

Rule	Description
R-BIT-DELTA	$\delta_{0,1} \ \triangleright\ 0$
R-BIT-ONEO	$\mathbf{1}_{\mathcal{O}}(\mathsf{bool}) \ \triangleright\ \vert 0\rangle\langle 0\vert + \vert 1\rangle\langle 1\vert$
R-BIT-SUM	$\sum_{i\in\mathbf{U}(\mathsf{bool})} A \ \triangleright\ A\{i/0\}+A\{i/1\}$

Table 18: Rules about addition and sum.

Rule	Description
R-MULS2	$b_1\times\cdots\times(a_1+\cdots+a_n)\times\cdots\times b_m$ $\triangleright\ (b_1\times\cdots\times a_1\times\cdots\times b_m)+\cdots+(b_1\times\cdots\times a_n\times\cdots\times b_m)$

Rule	Description
R-SUM-ADD1	$Y_1 + \cdots + Y_n + \sum_{i \in M}(a+b).X$ $\triangleright\ Y_1 + \cdots + \sum_{i \in M}(a.X) + \cdots + \sum_{i \in M}(b.X) + Y_n$
R-SUM-FACTOR	$X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} A)$ $+(\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} A) + \cdots + X_n$ $X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} (1+1).A) + \cdots + X_n$
	$X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} a.A)$ $+(\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} A) + \cdots + X_n$ $X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} (a+1).A) + \cdots + X_n$
	$X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} a.A)$ $+(\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} b.A) + \cdots + X_n$ $X_1 + \cdots + (\sum_{k_1 \in s_1} \cdots \sum_{k_n \in s_n} (a+b).A) + \cdots + X_n$

Table 19: Rules to eliminate labels in Dirac notations.

Rule	Description
R-L-EXPAND	$K_R\ \triangleright\ \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} ((\langle i_R \| \cdot K).(\|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes \|i_{r_n}\rangle_{r_n})$
	$B_R\ \triangleright\ \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} (B \cdot \|i_R\rangle).({}_{r_1}\langle i_{r_1}\| \otimes \cdots \otimes {}_{r_n}\langle i_{r_n}\|)$
	$O_{R,R'}\ \triangleright\ \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} \sum_{i_{r'_1} \in \mathbf{U}(\sigma_{r'_1})} \cdots \sum_{i_{r'_{n'}} \in \mathbf{U}(\sigma_{r'_{n'}})}$ $((\langle i_R\| \cdot O \cdot \|i_{R'}\rangle).(\|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes \|i_{r_n}\rangle_{r_n} \otimes {}_{r'_1}\langle i_{r'_1}\| \otimes \cdots \otimes {}_{r'_{n'}}\langle i_{r'_{n'}}\|)$

Table 20: Rules for labelled Dirac notations.

Rule	Description
R-ADJDK	$({}_r\langle i\|)^\dagger\ \triangleright\ \|i\rangle_r$
R-ADJDB	$(\|i\rangle_r)^\dagger\ \triangleright\ {}_r\langle i\|$
R-ADJD0	$(D_1 \otimes \cdots \otimes D_n)^\dagger\ \triangleright\ D_1^\dagger \otimes \cdots \otimes D_n^\dagger$
R-ADJD1	$(D_1 \cdot D_2)^\dagger\ \triangleright\ D_2^\dagger \cdot D_1^\dagger$
R-SCRD0	$D_1 \otimes \cdots \otimes (a.D_n) \otimes \cdots \otimes D_m\ \triangleright\ a.(D_1 \otimes \cdots \otimes D_m)$
R-SCRD1	$(a.D_1) \cdot D_2\ \triangleright\ a.(D_1 \cdot D_2)$

Rule	Description
R-SCRD2	$D_1 \cdot (a.D_2) \ \triangleright\ a.(D_1 \cdot D_2)$
R-TSRD0	$X_1 \otimes \cdots \otimes (D_1 + \cdots + D_n) \otimes \cdots X_m$
	$\triangleright\ X_1 \otimes \cdots D_1 \cdots \otimes X_m + \cdots + X_1 \otimes \cdots D_n \cdots \otimes X_m$
R-DOTD0	$(D_1 + \cdots + D_n) \cdot D \ \triangleright\ D_1 \cdot D + \cdots + D_n \cdot D$
R-DOTD1	$D \cdot (D_1 + \cdots + D_n) \ \triangleright\ D \cdot D_1 + \cdots + D \cdot D_n$
R-SUM-PUSHD0	$X_1 \otimes \cdots (\sum_{i \in M} D) \cdots \otimes X_2 \ \triangleright\ \sum_{i \in M}(X_1 \otimes \cdots D \cdots \otimes X_n)$
R-SUM-PUSHD1	$(\sum_{i \in M} D_1) \cdot D_2 \ \triangleright\ \sum_{i \in M}(D_1 \cdot D_2)$
R-SUM-PUSHD2	$D_1 \cdot (\sum_{i \in M} D_2) \ \triangleright\ \sum_{i \in M}(D_1 \cdot D_2)$

Table 21: Rules to simplify dot product in labelled Dirac notations.

Rule	Description
R-L-SORT0	$A : \mathcal{D}(s_1, s_2), B : \mathcal{D}(s_1', s_2'), s_2 \cap s_1' = \emptyset \Rightarrow A \cdot B \ \triangleright\ A \otimes B$
R-L-SORT1	${}_r\langle i \vert \cdot \vert j \rangle_r \ \triangleright\ \delta_{i,j}$
R-L-SORT2	${}_r\langle i \vert \cdot (Y_1 \otimes \cdots \otimes \vert j \rangle_r \otimes \cdots \otimes Y_m) \ \triangleright\ \delta_{i,j}.(Y_1 \otimes \cdots \otimes Y_m)$
R-L-SORT3	$(X_1 \otimes \cdots \otimes {}_r\langle i \vert \otimes \cdots \otimes X_n) \cdot \vert j \rangle_r \ \triangleright\ \delta_{i,j}.(X_1 \otimes \cdots \otimes X_n)$
R-L-SORT1	$(X_1 \otimes \cdots \otimes {}_r\langle i \vert \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes \vert j \rangle_r \otimes \cdots \otimes Y_m)$
	$\triangleright\ \delta_{i,j}.(X_1 \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes Y_m)$