Dirace: Practical Proof Automation of Dirac Notation Equations

No Author Given

No Institute Given

Abstract. Dirac notations are widely used in theoretical reasonings of quantum computation and quantum information. The previous work proved that the equivalence of basic Dirac notations is decidable, and provided a corresponding term rewriting system. This work focuses on developing the theory and tool that practically solves this problem. Compared to the previous result, this work includes a typing system and a simplified term rewriting system. A much more efficient algorithm. And a language and algorithm for labelled Dirac notation is proposed as the final contribution. We implement the whole algorithm in C++ with a Mathematica backend for theories of scalars, which decides the 158 examples in seconds.

1 Introduction

In 1939, Dirac proposed his notation [1] for quantum mechanics, which is designed to represent linear algebra formulae in a compact and convenient way. For instance, $a\left|\psi\right\rangle + b\left|\phi\right\rangle$ indicates the addition of two vectors, i.e., the superposition of two states $\left|\psi\right\rangle$ and $\left|\phi\right\rangle$. Dirac notation is also widely accepted as the working language in quantum computation and quantum information. The reasonings of Dirac notations play a fundamental role, but there was few works on automating this procedure.

The absence of a working Dirac notation solver is also a main obstacle for proof automation of quantum programming languages. In these works, Dirac notation are used 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.

Recently, the work by Yingte et al.[] provides a theory to decide the equivalence of Dirac notations, as well as a prototype implementation in Mathematica. They proved that the equivalence of basic Dirac notations are decidable. Their system sticks with a pure term rewriting system, which allows to prove important properties such as confluence and termination. Even though, there is still a gap to a practical solver for Dirac notation equivalence. The Mathematica implementation is hard to be integrated into other tools. The second concern is the efficiency. To decide the theory E, DiracDec iterates all possible permutations, which is simple and direct but has factorial complexity. And lastly, the previous

work does not consider labelled Dirac notation. The original work cannot intergrate with well with the Scalar system in mathematica. And there were a few examples that failded to decide.

This work refines the pure term-rewriting based system into a hybrid decision procedure, overcomes the problems mentioned above and focuses on practical automated equational reasonings. Our main contributions are:

- We propose a rigorous typing system for Dirac notation. Defined symbols (e.g., transpose and trace) are modelled by functions. Also, the typing system removes the difficulty of disambiguation, therefore the language and the term rewriting system are much simplified.
- An efficient algorithm to decide the extra equational theories E is provided. 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 language to support variable labels is proposed.

Related Work Other previous works explore the language and decision procedure to express quantum computation differently.

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. In 1939, Dirac proposed his notation [1] for quantum mechanics, which is essentially a wrapping of the linear algebra language. Dirac notation 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.

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

and they are equivalent for all variables. Dirac notation also use $\otimes$ to indicate the vectors and operators in the tensor product space.

In this manner, we can express long formulae in succint Dirac notations. The 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

With the concrete basis, Dirac notations can be interpreted as matrices. For example ...

We use universal algebra and equational logic to formally represent Dirac notations and the reasoning procedure in computers.

[YX] : mention the dilemma of simplicity and efficiency

3 Language, Typing and Semantics

The syntax of Dirac notations involves three layers: the index, the type and the term. Index represents classical datatypes, and they appear in type expression to denote the type of Hilbert spaces and sets.

Definition 1 (language syntax). *The syntax for type indices are defined as*

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

The syntax for Dirac notation types is defined as

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

The syntax for Dirac notation terms is defined as

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

Here i is a natural number and $\$i$ represents the i-th bound variable in de Bruijn notation. Compared to [?], this syntax for Dirac notations merges the symbols with overlapped properties, such as the addition and scaling symbols for ket, bra and operator. Here ADDS and ADD are two different AC symbols representing the scalar addition and the linear algebra addition respectively. They will be denoted as $a_1 + \cdots + a_n$ and $X_1 + \cdots + X_n$. There are five kinds of linear algebra multiplications among ket, bra and operator, whose properties are similar but still diverge to some extent. For example, the rules $(O_1 \cdot O_2) \cdot K \triangleright O_1 \cdot (O_2 \cdot K)$ and $B \cdot (O_1 \cdot O_2) \triangleright (B \cdot O_1) \cdot O_2$ indicate that the sorting of multiplication sequences depends on the subterm types. To avoid frequent but unnecessary type checkings, 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.

Usually, the sum body is specified by an abstraction. Therefore we use notation $\sum_s X$ to denote $\sum_{x \in s} \lambda x : T.X$ as well.

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 2 (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 $x \circ y$. As is the case for casual Dirac notation, 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.*

3.2 Semantics

Semantics assign the meanings to the expressions, and the goal of the decision procedure is to decide whether two expressions have the equivalent semantics. We consider two ways of definition: the axiomatic equations and the denotational semantics.

The denotational way interprets every expression as a linear algebra concept, and equivalence is considered in the common mathematical sense. This explanation formalizes the original definition of Dirac notations, and best describes the target of the decision procedure.

The semantics by equations, on the other hand, is an abstraction and axiomization. From the operational view, each equation declares a valid rewriting operation, and two expressions are equiavlent if and only if they can be rewritten into the same form using the axioms.

4 Decision procedure for Dirac notations

The following two sections talk about how to decide the equivalence of Dirac notations. 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.

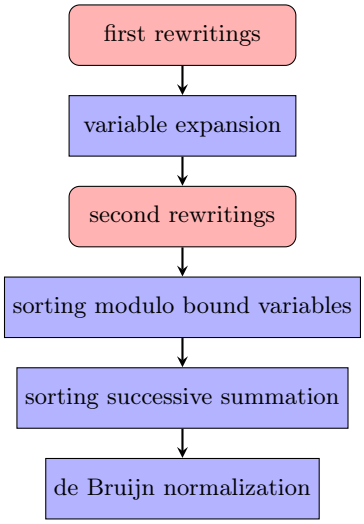

Fig. 1. A flowchart for normalization of Dirac notations.

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.

Here to enable more efficient algorithms, our langauge uses AC symbols with a variable arity.

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

When using variable arity AC symbols, the associativity can be normalized by the following flattening rule:

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

while commutativity is left for a sorting algorithm later on.

The term rewriting rule follows the system in the previous work.

The general idea is to reduce all the possible calculations, and transform multiplication into tensor product as much as possible.

And some rules exists for completeness. As an example, given the following rule snippet

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

the 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.

One important technique revealed in the previous 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 obviously not terminating, and this is why we have the rewriting-expansion-rewriting steps in the decision procedure. On the other hand, we discovered that doing the expansion on all variables only once is already sufficient.

Lemma 2. *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

The previous work did not consider the algorithm to decide equational theory E. In their Mathematica implementation, it is implemented by a unification, which tries to find a substitution of summation bound variables that makes the two expressions syntactically equivalent. To decide AC-equivalence and (SUM-SWAP), they iterate through all the permutations, and the complexity is factorial to the number of AC symbol arguments and successive summations.

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 related difficulties: how to assign such an order to all terms in our language, and how to normalize with respect to AC-equivalence and (SUM-SWAP) at the same time.

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

[LZ] : I suggest not adding first and second. Of course, we can, decide this later after discussion.

[LZ] : We further only consider constant registers.

Assume the name set of registers $\mathcal{R}$.

Definition 3 (quantum registers).

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

Definition 4 (register variable set). *The variable set of a register is defined inductively by:*

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

We define the following relations for quantum registers:

- R ***belongs to*** Q, *written as* R *in* Q, *if* $\mathsf{var}\ R \subseteq \mathsf{var}\ Q$;
- R ***is disjoint with*** Q, *written as* $R \| Q$, *if* $\mathsf{var}\ R \cap \mathsf{var}\ Q = \emptyset$.

[LZ] : Since the construction of the quantum register is known, the variable set can be directly computed. Similarly, we use the (computable) set as dependent parameter of the type of Labelled Dirac Term.

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 5 (labelled core language). *The **labelled core langauge** includes all symbols in the core language of Dirac notation, as well as symbols defined below. Here, $s \subseteq \mathcal{R}$ is the set of register's name, it is computable and thus we do not introduce syntax for it.*

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

In other words, we don't allow variables for labelled core language for now. In particular, $|i\rangle_r$ and ${}_r\langle i|$ are used as reserved syntax for eliminating registers.

Typing rules *[LZ] : context? environment? Since quantum registers are not bound variables, we might just add it to the environment.*

The extra syntax for the environment.

$$E ::= \cdots \mid E; r : \mathsf{Reg}\ (\sigma) \tag{1}$$

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:

   $$D_f = \Big(\prod_{(i,j) \in \mathsf{Can}(\tilde{D}_f)} \delta_{i,j}\Big).\Big((|i_1\rangle \otimes \cdots \otimes |i_n\rangle) \cdot (\langle i'_1| \otimes \cdots \otimes \langle i'_{n'}|)\Big).$$

   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

8 Conclusion

9 Consideration

- Sorting requires that alpha equivalent terms have the same syntax.
- de Bruijn expression satisfies this requirement, but the typing and substitution becomes very complicated in the type index scenario.
- only compute de Bruijn form in the equivalence checking and sorting phase.
- pipeline:
  1. preprocessing
  2. rename unique bound variable names
  3. 1st rewritings
  4. variable expansion
  5. 2nd rewritings
  6. sorting modulo bound variables
  7. sorting successive sum. The order depends on there occurances in the last sorting result. If no occurance, then the order will depend on the set (for sum) and the type (for lambda abstraction only).
  8. computing de Bruijn
- I found that typing of inner bound variables also requires modifying the context. Need to modify the tree visiting algorithm (Typed term rewriting with bound variables is not that easy)

10 TODO

- implemented trace output for the whole pipeline.
- better output.
- add Mathematica support for scalars
- output trace to Coq
- consider labelled Dirac notations
- better trace output
- better error logic
- better format output
- remove term bank and hash consing

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.
- I guess it may still be necessary to try all different bound variable assignments.
- the $\bar{U_{AC}}$, $\bar{U_{BC}}$ terms can be modelled by quantum registers.

12 Diracoq language

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

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. 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

A Full Typing Rules

This section includes the full list of typing rules.

– Rules for a well-formed environment and context.

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

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

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

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

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

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

– Rules for type indices.

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

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

$$\textbf{Index-Qudit} \quad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{Bit} : \mathsf{Index}}$$

– Rules for types.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

– Basis term typing rules.

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

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

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

– Composition typing rules.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

– Scalar term typing rules.

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

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

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

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

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

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

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

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

– 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.

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

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

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

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

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

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

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

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

– Operator term typing rules.

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

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

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

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

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

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

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

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

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

– Set term typing rules.

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

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

– Register term typing rules.

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

$$\textbf{Reg-Pair}\quad \frac{E[\Gamma] \vdash R : \mathsf{Reg}\ \sigma \qquad E[\Gamma] \vdash Q : \mathsf{Reg}\ \tau \qquad R \| Q}{E[\Gamma] \vdash (R, Q) : \mathsf{Reg}\ \sigma \times \tau}$$

– Typing rules for labelled Dirac notations.

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

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

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

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

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

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

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

L-Add $$\frac{E[\Gamma] \vdash \tilde{D}_i : \mathcal{D}(s_1, s_2) \quad \text{forall } i}{E[\Gamma] \vdash \tilde{D}_1 + \cdots + \tilde{D}_n : \mathcal{D}(s_1, s_2)}$$

L-Tsr $$\frac{\begin{array}{ll} E[\Gamma] \vdash \tilde{D}_1 : \mathcal{D}(s_1, s_1') & s_1 \cap s_2 = \emptyset \\ E[\Gamma] \vdash \tilde{D}_2 : \mathcal{D}(s_2, s_2') & s_1' \cap s_2' = \emptyset \end{array}}{E[\Gamma] \vdash \tilde{D}_1 \otimes \tilde{D}_2 : \mathcal{D}(s_1 \cup s_2, s_1' \cup s_2')}$$

L-Dot $$\frac{\begin{array}{ll} E[\Gamma] \vdash \tilde{D}_1 : \mathcal{D}(s_1, s_1') & s_1 \cap s_2 \backslash s_1' = \emptyset \\ E[\Gamma] \vdash \tilde{D}_2 : \mathcal{D}(s_2, s_2') & s_2' \cap s_1' \backslash s_2 = \emptyset \end{array}}{E[\Gamma] \vdash \tilde{D}_1 \cdot \tilde{D}_2 : \mathcal{D}(s_1 \cup (s_2 \backslash s_1'), s_2' \cup (s_1' \backslash s_2))}$$

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

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	$\langle s| \cdot |t\rangle\ \triangleright\ \delta_{s,t}$
R-DOT7	$(B_1 \otimes B_2) \cdot |(s,t)\rangle\ \triangleright\ (B_1 \cdot |s\rangle) \times (B_2 \cdot |t\rangle)$
R-DOT8	$\langle(s,t)| \cdot (K_1 \otimes K_2)\ \triangleright\ (\langle s| \cdot K_1) \times (\langle t| \cdot K_2)$
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	$\langle(s,t)| \cdot ((O_1 \otimes O_2) \cdot K)\ \triangleright\ ((\langle s| \cdot O_1) \otimes (\langle t| \cdot O_2)) \cdot K$
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	$\langle t\vert^\dagger \ \triangleright\ \vert t\rangle$
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	$\vert t\rangle^\dagger \ \triangleright\ \langle t\vert$
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	$\vert s\rangle \otimes \vert t\rangle \ \triangleright\ \vert(s,t)\rangle$
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	$\langle s\vert \otimes \langle t\vert \ \triangleright\ \langle(s,t)\vert$
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	$\mathbf{1}_{\mathcal{O}}(\sigma) \ \triangleright\ \sum_{i \in \mathbf{U}(\sigma)} \lvert i\rangle \cdot \langle i\rvert$

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 Bit index.

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

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-PUSHD2	$(\sum_{i \in M} D_1) \cdot D_2 \ \triangleright \ \sum_{i \in M} (D_1 \cdot D_2)$
R-SUM-PUSHD3	$D_1 \cdot (\sum_{i \in M} D_2) \ \triangleright \ \sum_{i \in M} (D_1 \cdot D_2)$

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	${}_r\langle i| \cdot |j\rangle_r \ \triangleright \ \delta_{i,j}$
R-L-SORT2	${}_r\langle i| \cdot (Y_1 \otimes \cdots \otimes |j\rangle_r \otimes \cdots \otimes Y_m) \ \triangleright \ \delta_{i,j}.(Y_1 \otimes \cdots \otimes Y_m)$
R-L-SORT3	$(X_1 \otimes \cdots \otimes {}_r\langle i| \otimes \cdots \otimes X_n) \cdot |j\rangle_r \ \triangleright \ \delta_{i,j}.(X_1 \otimes \cdots \otimes X_n)$
R-L-SORT1	$(X_1 \otimes \cdots \otimes {}_r\langle i| \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes |j\rangle_r \otimes \cdots \otimes Y_m)$ $\triangleright \delta_{i,j}.(X_1 \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes Y_m)$