DiracDec C++/Coq Implementation[1]

First Author$^{1[0000-1111-2222-3333]}$, Second Author$^{2,3[1111-2222-3333-4444]}$, and Third Author$^{3[2222--3333-4444-5555]}$

^{1} Princeton University, Princeton NJ 08544, USA
^{2} Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
`lncs@springer.com`
http://www.springer.com/gp/computer-science/lncs
^{3} ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
`{abc,lncs}@uni-heidelberg.de`

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Consideration

- We can emphasize that we are moving from pure term rewriting to a hybrid algorithm.
- AC equivalence and alpha equivalence.
- Efficient AC equivalence checking by sorting on syntax.
- 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)

2 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

3 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 U^{-}_{AC}, U^{-}_{BC} terms can be modelled by quantum registers.

4 Language Syntax

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

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

The syntax for Dirac notation types is defined as

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

The syntax for Dirac notation terms is defined as

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

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.

5 Typing System

The type checking of Dirac notations involves maintaining a well-formed environment and context $E[\Gamma]$, which specifies the definitions and typing assumtpions for variables. The environment and the conetxt are defined as follows.

Definition 2 (environment and context).

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

Note that in the following rules, $x : \mathsf{Index}$ and $x : \mathsf{Type}$ are not considered as $x : T$ judgements.

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

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

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

W-AssumC-Index $\dfrac{\mathcal{WF}(E)[\Gamma]}{\mathcal{WF}(E)[\Gamma; x : \mathsf{Index}]}$ **W-AssumC-Term** $\dfrac{E[\Gamma] \vdash T : \mathsf{Type}}{\mathcal{WF}(E)[\Gamma; x : T]}$

Fig. 1. Rules for a well-formed environment and context.

Index-Var $\dfrac{\mathcal{WF}(E)[\Gamma] \qquad x : \mathsf{Index} \in E[\Gamma]}{E[\Gamma] \vdash x : \mathsf{Index}}$ **Index-Prod** $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index} \qquad E[\Gamma] \vdash \tau : \mathsf{Index}}{E[\Gamma] \vdash \sigma \times \tau : \mathsf{Index}}$

Index-Qudit $\dfrac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{Qudit}_n : \mathsf{Index}}$

Fig. 2. Rules for type index.

Type-Arrow $\dfrac{E[\Gamma] \vdash T : \mathsf{Type} \qquad E[\Gamma] \vdash U : \mathsf{Type}}{E[\Gamma] \vdash T \to U : \mathsf{Type}}$ **Type-Index** $\dfrac{E[\Gamma; x : \mathsf{Index}] \vdash U : \mathsf{Type}}{E[\Gamma] \vdash \forall x.U : \mathsf{Type}}$

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

Type-Ket $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathcal{K}(\sigma) : \mathsf{Type}}$ **Type-Bra** $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathcal{B}(\sigma) : \mathsf{Type}}$

Type-Opt $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index} \qquad E[\Gamma] \vdash \tau : \mathsf{Index}}{E[\Gamma] \vdash \mathcal{O}(\sigma, \tau) : \mathsf{Type}}$ **Type-Scalar** $\dfrac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathcal{S} : \mathsf{Type}}$

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

Fig. 3. Rules for types.

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

Lam $$\frac{E[\Gamma; x : T] \vdash t : U}{E[\Gamma] \vdash (\lambda x : T.t) : T \to U}$$ **Index** $$\frac{E[\Gamma; x : \mathsf{Index}] \vdash t : U}{E[\Gamma] \vdash (\mu x.t) : \forall x.U}$$

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

Fig. 4. Rules for variable and function typings. Here $U[u]$ means instantiate the first bound variable with u in U.

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

Fig. 5. Typing rules for Basis.

Sca-0 $$\frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash 0 : \mathcal{S}}$$ **Sca-1** $$\frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash 1 : \mathcal{S}}$$

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

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

Sca-Conj $$\frac{E[\Gamma] \vdash a : \mathcal{S}}{E[\Gamma] \vdash a^* : \mathcal{S}}$$ **Sca-Dot** $$\frac{E[\Gamma] \vdash B : \mathcal{B}(\sigma) \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash B \cdot K : \mathcal{S}}$$

Fig. 6. Scalar typing rules.

Ket-0 $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathbf{0}_{\mathcal{K}}(\sigma) : \mathcal{K}(\sigma)}$ **Ket-Base** $\dfrac{E[\Gamma] \vdash t : \mathsf{Basis}(\sigma)}{E[\Gamma] \vdash |t\rangle : \mathcal{K}(\sigma)}$

Ket-Adj $\dfrac{E[\Gamma] \vdash B : \mathcal{B}(\sigma)}{E[\Gamma] \vdash B^\dagger : \mathcal{K}(\sigma)}$ **Ket-Scr** $\dfrac{E[\Gamma] \vdash a : \mathcal{S} \qquad E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash a.K : \mathcal{K}(\sigma)}$

Ket-Add $\dfrac{E[\Gamma] \vdash K_i : \mathcal{K}(\sigma) \text{ for all } i}{E[\Gamma] \vdash K_1 + \cdots + K_n : \mathcal{K}(\sigma)}$ **Ket-MulK** $\dfrac{E[\Gamma] \vdash O : \mathcal{O}(\sigma, \tau) \qquad E[\Gamma] \vdash K : \mathcal{K}(\tau)}{E[\Gamma] \vdash O \cdot K : \mathcal{K}(\sigma)}$

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

Fig. 7. Ket typing rules.

Bra-0 $\dfrac{E[\Gamma] \vdash \sigma : \mathsf{Index}}{E[\Gamma] \vdash \mathbf{0}_{\mathcal{B}}(\sigma) : \mathcal{B}(\sigma)}$ **Bra-Base** $\dfrac{E[\Gamma] \vdash t : \mathsf{Basis}(\sigma)}{E[\Gamma] \vdash \langle t| : \mathcal{B}(\sigma)}$

Bra-Adj $\dfrac{E[\Gamma] \vdash K : \mathcal{K}(\sigma)}{E[\Gamma] \vdash K^\dagger : \mathcal{B}(\sigma)}$ **Bra-Scr** $\dfrac{E[\Gamma] \vdash a : \mathcal{S} \qquad E[\Gamma] \vdash B : \mathcal{B}(\sigma)}{E[\Gamma] \vdash a.B : \mathcal{B}(\sigma)}$

Bra-Add $\dfrac{E[\Gamma] \vdash B_i : \mathcal{B}(\sigma) \text{ for all } i}{E[\Gamma] \vdash B_1 + \cdots + B_n : \mathcal{B}(\sigma)}$ **Bra-MulB** $\dfrac{E[\Gamma] \vdash B : \mathcal{K}(\sigma) \qquad E[\Gamma] \vdash O : \mathcal{O}(\sigma, \tau)}{E[\Gamma] \vdash B \cdot O : \mathcal{B}(\tau)}$

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

Fig. 8. Bra typing rules.

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

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

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

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

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

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

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

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

Fig. 9. Operator typing rules.

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

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

Fig. 10. Set typing rules.

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

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

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

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

Fig. 11. Sum typing rules.

6 Conversions and Reductions

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

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

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

Beta-Arrow $$\frac{}{E[\Gamma] \vdash ((\lambda x : T.t)\ u) \ \triangleright\ t\{x/u\}}$$ **Beta-Index** $$\frac{}{E[\Gamma] \vdash ((\mu x.t)\ u) \ \triangleright\ t\{x/u\}}$$

Delta $$\frac{\mathcal{WF}(E)[\Gamma] \qquad (c := t : T) \in E}{E[\Gamma] \vdash c \ \triangleright\ t}$$

Fig. 12. Conversions and reductions for the typed lambda calculus.

Table 1: The special flattening rule.

Rule	**Description**
R-FLATTEN	$a_1 + \cdots + (b_1 + \cdots + b_m) + \cdots + a_n \ \triangleright\ a_1 + \cdots + b_1 + \cdots + b_m + \cdots + a_n$
	$a_1 \times \cdots \times (b_1 \times \cdots \times b_m) \times \cdots \times a_n \ \triangleright\ a_1 \times \cdots \times b_1 \times \cdots \times b_m \times \cdots \times a_n$
	$X_1 + \cdots + (X'_1 + \cdots + X'_m) + \cdots + X_n \ \triangleright\ X_1 + \cdots + X'_1 + \cdots + X'_m + \cdots + X_n$
	A special rule to flatten all AC symbols within one call.

Table 2: Rules for scalar addition and multiplication.

Rule	**Description**
R-ADDSID	$+(a) \ \triangleright\ a$

Continued on the next page

Rule	**Description**
	Reduce the term if the AC symbol $+$ only has one argument.
R-MULSID	$\times(a) \ \triangleright \ a$
	Similar to (R-ADDSID).
R-ADDS0	$a_1 + \cdots + 0 + \cdots + a_n \ \triangleright \ a_1 + \cdots + a_n$
	This rule removes all 0 occurances and keeps the order of remaining subterms.
R-MULS0	$a_1 \times \cdots \times 0 \times \cdots \times a_n \ \triangleright \ 0$
R-MULS1	$a_1 \times \cdots \times 1 \times \cdots \times a_n \ \triangleright \ a_1 \times \cdots \times a_n$
	Similar to (R-ADDS0).
R-MULS2	$b_1 \times \cdots \times (a_1 + \cdots + a_n) \times \cdots \times b_m$
	$\triangleright\ (b_1 \times \cdots \times a_1 \times \cdots \times b_m) + \cdots + (b_1 \times \cdots \times a_n \times \cdots \times b_m)$
	This rule matches the first scalar addition subterm in the list.

Table 3: Rules for other scalar symbols.

Rule	**Description**
R-CONJ0	$0^* \ \triangleright \ 0$
R-CONJ1	$1^* \ \triangleright \ 1$
R-CONJ2	$(a_1 + \cdots + a_n)^* \ \triangleright \ a_1^* + \cdots + a_n^*$
R-CONJ3	$(a_1 \times \cdots \times a_n)^* \ \triangleright \ a_1^* \times \cdots \times a_n^*$
R-CONJ4	$(a^*)^* \ \triangleright \ a$
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	$(B_1 \otimes B_2) \cdot ((O_1 \otimes O_2) \cdot K) \ \triangleright \ ((B_1 \cdot O_1) \otimes (B_2 \cdot O_2)) \cdot K$

Continued on the next page

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 linear algebra 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 linear algebra addition.

Rule	Description
R-ADDID	$+(X) \ \triangleright\ X$ Reduce the term if the AC symbol + only has one argument.
R-ADD0	$Y_1 + \cdots + X + \cdots + X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + \cdots + (1+1).X$ This rule matches the first X in the list satisfying the pattern. The result $(1+1).X$ will be placed at the end of the list.
R-ADD1	$Y_1 + \cdots + X + \cdots + a.X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (1+a).X$ Similar to (R-ADD0).
R-ADD2	$Y_1 + \cdots + a.X + \cdots + X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (a+1).X$ Similar to (R-ADD0).
R-ADD3	$Y_1 + \cdots + a.X + \cdots + b.X + \cdots + Y_n \ \triangleright\ Y_1 + \cdots + Y_n + (a+b).X$ Similar to (R-ADD0).
R-ADDK0	$K_1 + \cdots + \mathbf{0}_{\mathcal{K}}(\sigma) + \cdots + K_n \ \triangleright K_1 + \cdots + K_n$ Similar to (R-ADDS0).
R-ADDB0	$B_1 + \cdots + \mathbf{0}_{\mathcal{B}}(\sigma) + \cdots + B_n \ \triangleright\ B_1 + \cdots + B_n$ Similar to (R-ADDS0).
R-ADDO0	$O_1 + \cdots + \mathbf{0}_{\mathcal{O}}(\sigma,\tau) + \cdots + O_n \ \triangleright\ O_1 + \cdots + O_n$

Continued on the next page

Rule	Description
	Similar to (R-ADDS0).

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	$(X \otimes Y)^\dagger \ \triangleright\ X^\dagger \otimes Y^\dagger$
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	$O : \mathcal{O}(\sigma, \tau) \Rightarrow \mathbf{0}_\mathcal{O}(\sigma', \tau') \otimes O \ \triangleright\ \mathbf{0}_\mathcal{O}(\sigma' \times \sigma, \tau' \times \tau)$

Continued on the next page

Rule	Description
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	$(a.O) \cdot K \ \triangleright\ a.(O \cdot K)$
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	$\sum_{x \in s} 0 \ \triangleright\ 0$
	Continued on the next page

Rule	**Description**
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 $\delta_{s,t}$. Note that these rules will match the δ operator modulo the commutativity of its arguments.

Rule	**Description**
R-SUM-ELIM0	i free in $t \vdash \sum_{i\in\mathbf{U}(\sigma)} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} \delta_{i,t} \ \triangleright\ \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} 1$
R-SUM-ELIM1	i free in $t \vdash \sum_{i\in\mathbf{U}(\sigma)} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} (a_1 \times \cdots \times \delta_{i,t} \times \cdots \times a_n)$
	$\triangleright\ \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} a_1\{i/t\} \times \cdots \times a_n\{i/t\}$
R-SUM-ELIM2	i free in $t \vdash \sum_{i\in\mathbf{U}(\sigma)} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} (\delta_{i,t}.A) \ \triangleright\ \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} A\{i/t\}$
R-SUM-ELIM3	i free in $t \vdash \sum_{i\in\mathbf{U}(\sigma)} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} (a_1 \times \cdots \times \delta_{i,t} \times \cdots \times a_n).A$
	$\triangleright\ \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} (a_1\{i/t\} \times \cdots \times a_n\{i/t\}).A\{i/t\}$
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	$\sum_{i\in M} \sum_{j\in M} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} ((a_1 \times \cdots \times \delta_{i,j} \times \cdots \times a_n) + \cdots + (b_1 \times \cdots \times \delta_{i,j} \times \cdots \times b_n)$
	$\triangleright\ \sum_{j\in M} \sum_{k_1\in s_1} \cdots \sum_{k_n\in s_n} ((a_1\{j/i\} \times \cdots \times a_n\{j/i\}) + \cdots + (b_1\{j/i\} \times \cdots \times b_n\{j/i\})).A\{j/$

Table 15: Rules for pushing terms into sum operators

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	$(\sum_{i\in M} a)^* \ \triangleright\ \sum_{i\in M} a^*$
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	$(\sum_{i\in M} B) \cdot O \ \triangleright\ \sum_{i\in M}(B \cdot O)$

Continued on the next page

Rule	Description
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	$X_1 \otimes (\sum_{i\in M} X_2) \ \triangleright\ \sum_{i\in M}(X_1 \otimes X_2)$

[YX] : Note: because we apply type checking on variables, and stick to unique bound variables, these pushing in operations are always sound.

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-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-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	$\sum_{i\in M_1 \star M_2} A \ \triangleright\ \sum_{j\in M_1} \sum_{k\in M_2} A\{i/(j,k)\}$

[YX] : The rules R-SUM-ADD1 to R-SUM-ADD3 needs identical sum terms, which requires the rewriting to be after the alpha normalization. We don't implement them for now.

Table 17: Rules for Qubit

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

7 Preprocessing

$$\frac{E[\varGamma] \vdash B : \mathcal{B}(\sigma) \qquad E[\varGamma] \vdash K : \mathcal{K}(\sigma)}{E[\varGamma] \vdash B \circ K : \mathcal{S}}$$

Fig. 13. The typing rules for preprocessing

8 `Diracoq` language

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

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))`

9 Rewriting Control and Intermediate Language

The associativity is already handled by the (R-FLATTEN) rule. In order to decide two terms A and B are equivalent under commutativity, we need to proof that A can be transformed into B with a structured permutation, which is described by the *permutation tree.*

Definition 3 (permutation tree). *The syntax for permutation trees are inductively defined below:*

$$P ::= \mathsf{E} \mid [(i : P)^{+}].$$

Here i represents positive numbers.

We always only consider *well-formed* permutation trees. That is, if $P \equiv [i_1 : P_1\ i_2 : P_2\ \cdots\ i_n : P_n]$, then $\{i_1, ...i_n\}$ forms the set of integers from 0 to $n-1$.

We can transform a term A with a suitable permutation tree. The transformation is defined as

```
apply($A, P$) := match $P$ with
                 | $\mathsf{E} \Rightarrow A$
                 | $[i_1 : P_1\ \cdots\ i_n : P_n] \Rightarrow$ A.head(apply(A.args[$i_1$], $P_{i_1}$) $\cdots$ apply(A.args[$i_n$], $P_{i_n}$))
                 end
```

10 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 4 (quantum registers).

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

Definition 5 (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 6 (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.*

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

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.

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

$$\textbf{W-AssumE-Reg} \qquad \frac{E[] \vdash \sigma : \mathsf{Index} \qquad r \notin E}{\mathcal{WF}[E; r : \mathsf{Reg}\ (\sigma)[]]} \tag{2}$$

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

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

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

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

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

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

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

$$\textbf{Type-L-Opt}\quad \frac{E[\Gamma] \vdash R_1 : \mathsf{Reg}\ \sigma_1 \qquad E[\Gamma] \vdash R_2 : \mathsf{Reg}\ \sigma_2 \qquad E[\Gamma] \vdash O : \mathcal{O}(\sigma_1; \sigma_2)}{E[\Gamma] \vdash O_{R_1;R_2} : \mathcal{D}(\mathsf{var}\ R_1; \mathsf{var}\ R_2)}$$

$$\textbf{Type-L-Conj}\quad \frac{E[\Gamma] \vdash \tilde{D} : \mathcal{D}(s_1; s_2)}{E[\Gamma] \vdash \tilde{D}^\dagger : \mathcal{D}(s_2; s_1)} \qquad \textbf{Type-L-SCL}\quad \frac{E[\Gamma] \vdash S : \mathcal{S} \qquad E[\Gamma] \vdash \tilde{D} : \mathcal{D}(s_1; s_2)}{E[\Gamma] \vdash S.\tilde{D} : \mathcal{D}(s_1; s_2)}$$

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

$$\textbf{Type-L-TSR}\quad \frac{E[\Gamma] \vdash \tilde{D}_1 : \mathcal{D}(s_1; s_1') \qquad E[\Gamma] \vdash \tilde{D}_2 : \mathcal{D}(s_2; s_2') \qquad s_1 \cap s_2 = \emptyset \qquad s_1' \cap s_2' = \emptyset}{E[\Gamma] \vdash \tilde{D}_1 \otimes \tilde{D}_2 : \mathcal{D}(s_1 \cup s_2; s_1' \cup s_2')}$$

$$\textbf{Type-L-DOT}\quad \frac{E[\Gamma] \vdash \tilde{D}_1 : \mathcal{D}(s_1; s_1') \qquad E[\Gamma] \vdash \tilde{D}_2 : \mathcal{D}(s_2; s_2') \qquad s_1 \cap s_2 \backslash s_1' = \emptyset \qquad s_2' \cap s_1' \backslash s_2 = \emptyset}{E[\Gamma] \vdash \tilde{D}_1 \cdot \tilde{D}_2 : \mathcal{D}(s_1 \cup (s_2 \backslash s_1'); s_2' \cup (s_1' \backslash s_2))}$$

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 $K_R, B_R, O_{R,R'}$:

$$K_R = \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} (\langle i_R| \cdot K).(|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes |i_{r_n}\rangle_{r_n}).$$

$$B_R = \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} (B \cdot |i_R\rangle).({}_{r_1}\langle i_{r_1}| \otimes \cdots \otimes {}_{r_n}\langle i_{r_n}|).$$

$$O_{R,R'} = \sum_{i_{r_1} \in \mathbf{U}(\sigma_{r_1})} \cdots \sum_{i_{r_n} \in \mathbf{U}(\sigma_{r_n})} \sum_{i_{r'_1} \in \mathbf{U}(\sigma_{r'_1})} \cdots \sum_{i_{r'_{n'}} \in \mathbf{U}(\sigma_{r'_{n'}})}$$
$$(\langle i_R| \cdot O \cdot |i_{R'}\rangle).\Big((|i_{r_1}\rangle_{r_i} \otimes \cdots \otimes |i_{r_n}\rangle_{r_n}) \otimes ({}_{r'_1}\langle i_{r'_1}| \otimes \cdots \otimes {}_{r'_{n'}}\langle i_{r'_{n'}}|)\Big).$$

3. Rewriting to push big operators and split: **Beta-Arrow**, **Beta-Index**, **Delta**

Table 18: The special flattening rule.

Rule	Description
R-ADJDK	$({}_r\langle i|)^\dagger \ \triangleright\ |i\rangle_r$
R-ADJDB	$(|i\rangle_r)^\dagger \ \triangleright\ {}_r\langle i|$
	Dealing with adjoint; now we don't have any $\tilde{D}^\dagger$ by combining push rules.
R-ADJD0	$(\tilde{D}_1 \otimes \tilde{D}_2)^\dagger \ \triangleright\ \tilde{D}_1^\dagger \otimes \tilde{D}_2^\dagger$
R-ADJD1	$(\tilde{D}_1 \cdot \tilde{D}_2)^\dagger \ \triangleright\ \tilde{D}_2^\dagger \cdot \tilde{D}_1^\dagger$
R-SCRD0	$D_1 \otimes \cdots \otimes (a.\tilde{D}_n) \otimes \cdots \otimes \tilde{D}_m \ \triangleright\ a.(\tilde{D}_1 \otimes \cdots \otimes \tilde{D}_m)$
R-SCRD1	$(a.\tilde{D}_1) \cdot \tilde{D}_2 \ \triangleright\ a.(\tilde{D}_1 \cdot \tilde{D}_2)$
R-SCRD2	$\tilde{D}_1 \cdot (a.\tilde{D}_2) \ \triangleright\ a.(\tilde{D}_1 \cdot \tilde{D}_2)$
	Dealing with scaling
R-TSRD0	$(\tilde{D}_1 + \cdots + \tilde{D}_n) \otimes \tilde{D} \ \triangleright\ \tilde{D}_1 \otimes \tilde{D} + \cdots + \tilde{D}_n \otimes \tilde{D}$
R-DOTD0	$(\tilde{D}_1 + \cdots + \tilde{D}_n) \cdot \tilde{D} \ \triangleright\ \tilde{D}_1 \cdot \tilde{D} + \cdots + \tilde{D}_n \cdot \tilde{D}$
R-DOTD1	$\tilde{D} \cdot (\tilde{D}_1 + \cdots + \tilde{D}_n) \ \triangleright\ \tilde{D} \cdot \tilde{D}_1 + \cdots + \tilde{D} \cdot \tilde{D}_n$
	Distributivity of $\cdot$ and $\otimes$.
R-SUM-PUSHD0	$(\sum_{i \in M} \tilde{D}_1) \otimes \tilde{D}_2 \ \triangleright\ \sum_{i \in M}(\tilde{D}_1 \otimes \tilde{D}_2)$
R-SUM-PUSHD2	$(\sum_{i \in M} \tilde{D}_1) \cdot \tilde{D}_2 \ \triangleright\ \sum_{i \in M}(\tilde{D}_1 \cdot \tilde{D}_2)$
R-SUM-PUSHD3	$\tilde{D}_1 \cdot (\sum_{i \in M} \tilde{D}_2) \ \triangleright\ \sum_{i \in M}(\tilde{D}_1 \cdot \tilde{D}_2)$
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	$(X_1 \otimes \cdots \otimes {}_r\langle i| \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes |j\rangle_r \otimes \cdots \otimes Y_m)$

Continued on the next page

Rule	Description
	$\triangleright\ \delta_{i,j}.(X_1 \otimes \cdots \otimes X_n) \cdot (Y_1 \otimes \cdots \otimes Y_m)$

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 :

$$\tilde{D}_f ::= |i\rangle_r \mid {}_r\langle i| \mid \tilde{e} \otimes \tilde{e} \mid \tilde{e} \cdot \tilde{e}.$$

4. Simplify $\tilde{D}_f$. Let $\mathcal{N}$ be the variable name set. For every $\tilde{D}_f$, we define three functions $\mathsf{Can}(\tilde{D}_f)$: list $(\mathcal{N} \times \mathcal{N})$, $\mathsf{CDom}(\tilde{D}_f), \mathsf{Dom}(\tilde{D}_f)$: list $(\mathcal{N} \times \mathcal{R})$ that compute the cancellation pairs of variables, codomain and domain. They are inductively defined by the structure of $\tilde{D}_f$:
   - $\tilde{D}_f \equiv |i\rangle_r$: $\mathsf{Can}(\tilde{D}_f) = []$, $\mathsf{CDom}(\tilde{D}_f) = [(i,r)]$, $\mathsf{Dom}(\tilde{D}_f) = []$.
   - $\tilde{D}_f \equiv {}_r\langle i|$: $\mathsf{Can}(\tilde{D}_f) = []$, $\mathsf{CDom}(\tilde{D}_f) = []$, $\mathsf{Dom}(\tilde{D}_f) = [(i,r)]$.
   - $\tilde{D}_f \equiv \tilde{D}_{1f} \otimes \tilde{D}_{2f}$: $\mathsf{Can}(\tilde{D}_f) = \mathsf{Can}(\tilde{D}_{1f}); \mathsf{Can}(\tilde{D}_{2f})$, $\mathsf{CDom}(\tilde{D}_f) = \mathsf{CDom}(\tilde{D}_{1f}); \mathsf{CDom}(\tilde{D}_{2f})$, $\mathsf{Dom}(\tilde{D}_f) = \mathsf{Dom}(\tilde{D}_{1f}); \mathsf{Dom}(\tilde{D}_{2f})$. (Here, ; is the concatenation of lists)
   - $\tilde{D}_f \equiv \tilde{D}_{1f} \cdot \tilde{D}_{2f}$: recursively execute:
     (a) Set $L_{\text{can}} = []$, $L_{\text{dom}} = \mathsf{Dom}(\tilde{D}_{1f})$, $L_{\text{cdom}} = \mathsf{CDom}(\tilde{D}_{2f})$.
     (b) If there exists $(i_1, r) \in L_{\text{dom}}$ and $(i_2, r) \in L_{\text{cdom}}$, then $L_{\text{can}} = L_{\text{can}}; (i_1, i_2)$, $L_{\text{dom}} = L_{\text{dom}} \backslash (i_1, r)$ and $L_{\text{cdom}} = L_{\text{cdom}} \backslash (i_2, r)$. (Here, $\backslash$ means to remove the element from the list.)
     (c) Repeat step (b) until it fails.
     Then, $\mathsf{Can}(\tilde{D}_f) = \mathsf{Can}(\tilde{D}_{1f}); \mathsf{Can}(\tilde{D}_{2f}); L_{\text{can}}$, $\mathsf{CDom}(\tilde{D}_f) = \mathsf{CDom}(\tilde{D}_{1f}); L_{\text{cdom}}$, $\mathsf{Dom}(\tilde{D}_f) = \mathsf{Dom}(\tilde{D}_f); L_{\text{dom}}$.

   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

$$\tilde{D}_f = (D_f)_{R,R'}$$

5. 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).

Acknowledgments. A bold run-in heading in small font size at the end of the paper is used for general acknowledgments, for example: This study was funded by X (grant number Y).

Disclosure of Interests. It is now necessary to declare any competing interests or to specifically state that the authors have no competing interests. Please place the statement with a bold run-in heading in small font size beneath the (optional) acknowledgments[4], for example: The authors have no competing interests to declare that are relevant to the content of this article. Or: Author A has received research grants from Company W. Author B has received a speaker honorarium from Company X and owns stock in Company Y. Author C is a member of committee Z.

References

1. Shakespeard, W.: Halmet (2666)

[4] If EquinOCS, our proceedings submission system, is used, then the disclaimer can be provided directly in the system.