Two Different Ways of Creating Quantum Symbolic Operators

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx

Introduction

This document shows different ways to create operators in the Quantum *Mathematica* add-on. The main characteristic of an operator is that its "noncommutative quantum application" is **not** transformed into a normal, commutative multiplication.

Load the Package

First load the Quantum`Notation` package. Write:
Needs["Quantum`Notation`"];
then press at the same time the keys SHIFT-ENTER to evaluate. *Mathematica* will load the package:

```
Needs["Quantum`Notation`"]
```

```
Quantum`Notation` Version 2.2.0. (July 2010)
A Mathematica package for Quantum calculations in Dirac bra-ket notation
by José Luis Gómez-Muñoz

Execute SetQuantumAliases[] in order to use
  the keyboard to enter quantum objects in Dirac's notation
SetQuantumAliases[] must be executed again in each new
  notebook that is created, only one time per notebook.
```

In order to use the keyboard to enter quantum objects write:
SetQuantumAliases[ ];
then press at the same time the keys SHIFT-ENTER to evaluate. The semicolon prevents *Mathematica* from printing the help message. Remember that SetQuantumAliases[ ] must be evaluated again in each new notebook:

```
SetQuantumAliases[];
```

Undefined Symbols are Assumed to be Scalars

First we clear (erase) values and definitions for the symbols that will be used in the examples:

```
Clear[a, b]
```

Symbols are assumed to be scalars by default. This means that their "noncommutative quantum application" is automatically transformed into an ordinary commutative multiplication.
Press the keys:
b [ESC]on[ESC] a
then press at the same time [SHIFT]-[ENTER] to evaluate. Notice that the "noncommutative quantum application" b·a is transformed into the ordinary commutative multiplication ab:

```
b · a
```

```
a b
```

Scalars are pulled out as multiplicative factors, and the hermitian conjugate of scalars becomes the ordinary complex conjugate.
Press the keys:
[ESC]her[ESC][TAB][ESC]bra[ESC][ESC]on[ESC] b [ESC]on[ESC][ESC]ket[ESC]
press [TAB] one or two times to select the first place-holder (square) and press:
x[TAB]y
finally press at the same time [SHIFT]-[ENTER] to evaluate. Notice that b is pulled out as a multiplicative factor, and that the hermitian conjugate $b^{\dagger}$becomes the complex conjugate b^{*}:

```
(⟨x | · b ·  | y⟩)†
```

```
b* ⟨y | x⟩
```

The commutator of scalars is zero $[\![a, b]\!]_{-} = a \cdot b - b \cdot a = ab - ab = 0$
Press the keys:
[ESC]comm[ESC][TAB]a[TAB]b
finally press at the same time [SHIFT]-[ENTER] to evaluate.

```
⟦a, b⟧₋
```

```
0
```

First way to define new symbolic operators: Use SetQuantumObject[]

First we clear (erase) values and definitions for the symbols that will be used in the examples:

```
Clear[c, d]
```

The command SetQuantumObject[] is used to specify that c and d are operators. The semicolon ; prevents *Mathematica* from printing a confirmation message:

```
SetQuantumObject[c, d];
```

This time the "noncommutative quantum application" is **not** transformed into a multiplication:
d [ESC]on[ESC] c
then press at the same time [SHIFT]-[ENTER] to evaluate:

```
d · c
```

```
d · c
```

This time the hermitian conjugate is **not** transformed into a complex conjugate:
[ESC]her[ESC][TAB][ESC]bra[ESC][ESC]on[ESC] c [ESC]on[ESC][ESC]ket[ESC]
press [TAB] one or two times to select the first place-holder (square) and press:
x[TAB]y
finally press at the same time [SHIFT]-[ENTER] to evaluate. Hermitian conjugation rules are applied, but $c^{\dagger}$ is **not** transformed into c^{*}

```
(⟨x | · c ·  | y⟩)†
```

```
⟨y | · c† ·  | x⟩
```

This time the commutator $[\![c, d]\!]_{-} = c \cdot d - d \cdot c$ is **not** equal to zero:
Press the keys:
[ESC]comm[ESC][TAB]c[TAB]d
finally press at the same time [SHIFT]-[ENTER] to evaluate.

```
⟦c, d⟧₋
```

```
⟦c, d⟧₋
```

The commutator can be evaluated using the command EvaluateCommutators

```
EvaluateCommutators[⟦c, d⟧₋]
```

```
-d · c + c · d
```

Second way to define new symbolic operators: Use DefineOperatorOnKets[]

First we clear (erase) values and definitions for the symbols that will be used in the examples:

```
Clear[e, f]
```

Here we define "**e**" as an operator that will have a specific effect on kets. The standard *Mathematica* symbol :→ can be entered by pressing [ESC]:>[ESC]

```
DefineOperatorOnKets[e, { | x_⟩ :→  | x + 1⟩}]
```

```
 | x_⟩ :→  | x + 1⟩
```

This is the operator "**e**" acting on a ket:

```
e · | 4〉
```

```
| 5〉
```

Here we define "**f**" as an operator that will have a specific effect on kets. The standard *Mathematica* symbol :→ can be entered by pressing [ESC]:>[ESC]

```
DefineOperatorOnKets[f, { | x_〉 :→ | x - 1〉}]
```

```
| x_〉 :→ | x - 1〉
```

This is the operator "**f**" acting on a ket:

```
f · | 4〉
```

```
| 3〉
```

This time the "noncommutative quantum application" is **not** transformed into a multiplication:
f [ESC]on[ESC] e
then press at the same time [SHIFT]-[ENTER] to evaluate:

```
f · e
```

```
f · e
```

This time the hermitian conjugate is **not** transformed into a complex conjugate.
Press [ESC]her[ESC][TAB]f

```
(f)†
```

```
f†
```

This time the commutator $[\![e, f]\!]_- = e \cdot f - f \cdot e$ is **not** equal to zero:
Press the keys:
[ESC]comm[ESC][TAB]c[TAB]d
finally press at the same time [SHIFT]-[ENTER] to evaluate.

```
⟦e, f⟧_
```

```
⟦e, f⟧_
```

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx