QuantumAlgebra Basics

This tutorial introduces some of the basic principles how to enter Quantum objects and how to perform operations on them.

Preparation of a Session

First one must load the QuantumAlgebra application by using the Needs command.

This loads the package

```
In[16]:= Needs["QuantumAlgebra`"]
```

This command normally - in a *Mathematica* front end session - loads the required context into the notebook. Additionally a few commands are performed to prepare the session. This includes the definition of notations and input aliases.

You should invoke the command SetNotebookStyles[] once per Notebook. The required display styles are assigned to the notebook.

After this is done, you are ready to input Quantum expressions and work with them.

Sets required styles in the private style sheet

```
In[9]:= SetNotebookStyles[]
```

`SetNotebookStyles[]`	Sets the required styles for proper display of objects

Function Call to set Notebook Styles.

Provide Required Input

After the session is prepared one can input Quantum objets and work with them.

The basic Quantum objects supported by the package are Bra, Ket, BraKet, Operator.

Supported input methods are the textual fullform input and the input by using the input aliases. By providing an input alias you can generate a 2-dimensional template where the required information is to be placed in placeholders.

Remark: Operators with indices can be entered in FullForm only.

`Bra[m]`	Used to enter a Bra in FullForm
`Ket[n]`	Used to enter a Ket in FullForm
`BraKet[{m},{n}]`	Used to enter a BraKet in FullForm
`Operator[H,{},{j}]`	Used to enter an Operator in FullForm,here with index
`Operator[H,"H",{},{}]`	Used to enter an Hermitian Operator

FullForm of Quantum Objects.

Input Aliases as printed out during Needs

```
ALIASES:
ESC op ESC - Operator without index
ESC hop ESC - Hermitian Operator without index
ESC ket ESC - Ket without index
ESC ketd ESC - Ket with subscript(s)
ESC ketu ESC - Ket with superscript(s)
ESC ketdu ESC - Ket with subscript(s) & superscript(s)
ESC bra ESC - Ket without index
ESC dbra ESC - Bra with subscript(s)
ESC ubra ESC - Bra with superscript(s)
ESC dubra ESC - Bra with subscript(s) & superscript(s)
ESC (du)braket(du) ESC - Bra with (left and / or right) subscript(s) and / or superscript(s).
```

Sample FullForm input - an Operator and a Ket

```
Operator[V, {i}, {}]
```

$\hat{V}_i$

```
Ket[m, {i}, {}]
```

$\left| m \right\rangle_i$

Sample input with input alias - unfilled templates: a Bra and a BraKet

$^{\square}\langle \blacksquare |$

$_{\square}\langle \blacksquare | \blacksquare \rangle_{\blacksquare}$

Sample input with input alias - with information filled in

$^{k}\langle n |$

$^{k}\langle n |$

$_{i}\langle m | n \rangle_{j}$

$_{i}\langle m | n \rangle_{j}$

A Few Simple Calculations

This section demonstrates in a few simple samples, how calculations can be done in QuantumAlgebra.

Orthonormalized BraKet Sample

The first sample demonstrates, how a BraKet can be assigned a function to calculate its value and what the result multiplying a Bra and a Ket then is.

CenterDot	Used to multiply Bra and Ket (Alias: ESC . ESC)

Multiplication Operator to concatenate Bras, Kets and (optionally) Operators.

Definition and Usage of BraKets

```
In[31]:= (* Orthonormalized BraKet *)
         i_⟨ m_ | n_ ⟩j_ := KroneckerDelta[i, j] KroneckerDelta[m, n]

In[32]:= (* Equal indices and labels yield 1 *)
         1⟨n|·| n⟩1

Out[32]= 1
```

```
In[33]:= j⟨2|·| 2⟩j
Out[33]= 1

In[34]:= (* Else yields 0 *)
         2⟨n|·| n⟩3
Out[34]= 0

In[35]:= 3⟨5|·| 4⟩3
Out[35]= 0

In[36]:= (* Different symbols - Mathematica cannot make assertions about equality *)
         j⟨k|·| l⟩j
Out[36]= KroneckerDelta[k, l]
```

The Harmonic Oscillator Sample

The Quantum Mechanical Oscillator is one of the most important yet simple systems in Quantum Theory. We try to demonstrate the application of Quantum Algebra to analyze and calculate this system.

First we define a simplification function, which can be used as appropriate.

QASet	Defines the given rule and derived rules.
QACommutatorExpand	Replaces Commutators by their definition.
QAExpandAll	Combines some expand functions of QA.

A few functions used in this example.

A simplification function making assumptions

```
In[3]:= FullSimplifyAss[exp_] := FullSimplify[exp, Assumptions → m > 0 && ℏ > 0 && ω > 0] // Expand // PowerExpand
```

Next we define the Quantum Mechanical Oscillator by introducing its Hamiltonian. Then we define the important Commutation relation for Location and Momentum Operators. Finally we define the Location and Momentum Operators as Hermitian.

Hamiltonian, Commutation Relations and Hermitian

```
In[4]:= (* Hamilton Operator defined in terms of Location and Momentum Operator *)

In[5]:= Ĥ = 1 / 2 / m p̂^2 + m ω^2 / 2 x̂^2
```

Out[5]= $\frac{\hat{p}^2}{2\,m} + \frac{1}{2}\,m\,\omega^2\,\hat{x}^2$

```
In[6]:= (* Defines the Commutation Relation between Location and Momentum *)

In[7]:= QASet[[x̂, p̂]_ = ⅈ ℏ 1̂]

In[8]:= (* Location Operator x̂ is Hermitian *)

In[9]:= x̂† = x̂
Out[9]= x̂

In[10]:= (* The same yields true for the Momentum Operator p̂ *)

In[11]:= p̂† = p̂
Out[11]= p̂
```

Next we introduce a few rules. They are all based on the definition of the so called Creation Operator and its counter part - the Annihilation Operator. Whereas Rule 1 can substitute Annihilation and Creation Operators by Location and

Momentum, Rule 2 can do the opposite. Rules 3 and 4 are rules derived from Commutator definition for the Annihilation and Creation Operators. Rules 5 and 6 introduce the so called Number Operator $\hat{N}$.

Definition of required rules

```
In[14]:= (* We define rules where a Commutator definition is
          solved after one of the both products of the contributing Operators *)

In[15]:= commutatorRules[op1_Operator, op2_Operator] :=
          Module[{commutator = [op1, op2]_, rule1, rule2},
           rule1 = Solve[commutator == (commutator // QACommutatorExpand), op1 · op2];
           rule2 = Solve[commutator == (commutator // QACommutatorExpand), op2 · op1];
           Join[rule1[[1]], rule2[[1]]]
          ]

In[16]:=   (* From our start rule we derive a second rule by Hermitian operation and unify them in a List *)

In[17]:= rule1 = {rule, Map[Hermitian[#] &, rule]} // FullSimplifyAss
```

Out[17]= $\left\{\hat{a}^{\dagger} \to -\frac{i\,\hat{p}}{\sqrt{2}\sqrt{m}\sqrt{\omega}\sqrt{\hbar}} + \frac{\sqrt{m}\sqrt{\omega}\,\hat{x}}{\sqrt{2}\sqrt{\hbar}},\ \hat{a} \to \frac{i\,\hat{p}}{\sqrt{2}\sqrt{m}\sqrt{\omega}\sqrt{\hbar}} + \frac{\sqrt{m}\sqrt{\omega}\,\hat{x}}{\sqrt{2}\sqrt{\hbar}}\right\}$

```
In[18]:= (* Rule 2 solves the rules in Rule 1 after Location and Momentum operators *)

In[19]:= rule2 = Solve[rule1 /. Rule → Equal, {x̂, p̂}][[1]] // FullSimplifyAss
```

Out[19]= $\left\{\hat{x} \to \frac{\sqrt{\hbar}\,\hat{a}}{\sqrt{2}\sqrt{m}\sqrt{\omega}} + \frac{\sqrt{\hbar}\,\hat{a}^{\dagger}}{\sqrt{2}\sqrt{m}\sqrt{\omega}},\ \hat{p} \to -\frac{i\sqrt{m}\sqrt{\omega}\sqrt{\hbar}\,\hat{a}}{\sqrt{2}} + \frac{i\sqrt{m}\sqrt{\omega}\sqrt{\hbar}\,\hat{a}^{\dagger}}{\sqrt{2}}\right\}$

```
In[20]:= (* From the commutatorRules derives rules for Creation and Annihilation Operator *)

In[21]:= rule3 = commutatorRules[â, â†][[1]]
```

Out[21]= $\hat{a}\cdot\hat{a}^{\dagger} \to [\hat{a}, \hat{a}^{\dagger}]_{-} + \hat{a}^{\dagger}\cdot\hat{a}$

```
In[22]:= rule4 = commutatorRules[â, â†][[2]]
```

Out[22]= $\hat{a}^{\dagger}\cdot\hat{a} \to -[\hat{a}, \hat{a}^{\dagger}]_{-} + \hat{a}\cdot\hat{a}^{\dagger}$

```
In[23]:= rule5 = N̂ → â† · â
```

Out[23]= $\hat{N} \to \hat{a}^{\dagger}\cdot\hat{a}$

```
In[24]:= rule6 = â† · â → N̂
```

Out[24]= $\hat{a}^{\dagger}\cdot\hat{a} \to \hat{N}$

Rule 7 derives a Commutator rule for the Annihilation and Creation Operators.

After this the Hamiltonian is simplified using both Operator definitions.

And finally a Eigenvalue - Eigenket - Relation is adopted for the Number Operator.

Commutator rule for the Annihilation and Creation Operators

```
In[25]:= rule7 = [â, â†]_ → ([â, â†]_ /. rule1 // QACommutatorExpand)
```

Out[25]= $[\hat{a}, \hat{a}^{\dagger}]_{-} \to \hat{1}$

Simplified Hamiltonian

```
In[26]:= (Ĥ /. rule2 // QAPowerExpand // FullSimplifyAss) /. rule3 /. rule7 // QAExpandAll
```

Out[26]= $\omega\,\hbar\,\hat{a}^{\dagger}\cdot\hat{a} + \frac{1}{2}\omega\,\hbar\,\hat{1}$

Eigenvalue - Eigenket - Relation for the Number Operator

```
In[28]:= QASet[N̂ · |n_〉 = n |n〉]
```

Finally we derive the meaning of the Annihilation and Creation Operators, respectively.

It becomes apparent that Annihilation / Creation Operator, applied to an Eigenket of N̂ give a new Eigenket with Eigenvalue decreased / increased by one. It also justifies the term Number Operator for N̂.

With this demonstration we conclude this little sample.

Meaning of Annihilation and Creation Operators

```
In[29]:=  ((N̂ · â† · |n⟩ /. rule5) /. rule3) /. rule6 /. rule7
Out[29]=  â† · |n⟩ + n â† · |n⟩

In[30]:=  ((N̂ · â · |n⟩ /. rule5) /. rule4) /. rule6 /. rule7
Out[30]=  −(â · |n⟩) + n â · |n⟩
```

MORE ABOUT

- SetNotebookStyles
- Bra . Ket . BraKet . Operator

RELATED TUTORIALS

- Advanced Concepts