Quantum type system

in H-hat quantum programming language

Eduardo Maschio, FOSDEM - Brussels 2025

Summary

What is H-hat?

Quantum type system

Q&A

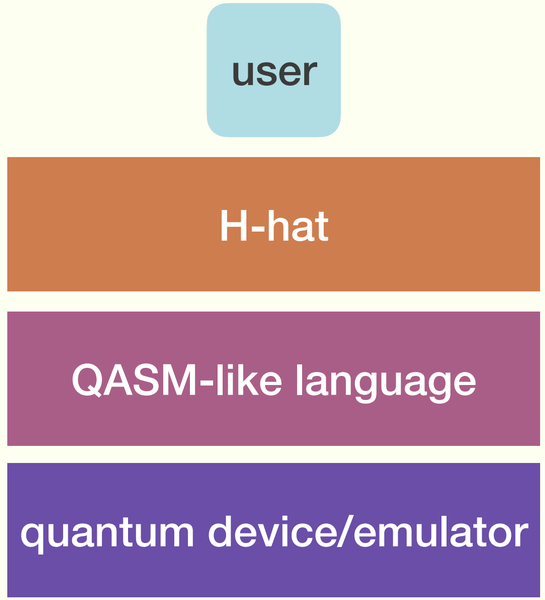
What is H-hat ($\hat{H}$) ?

What is H-hat ($\hat{H}$) ?

- A high-level abstraction quantum programming language

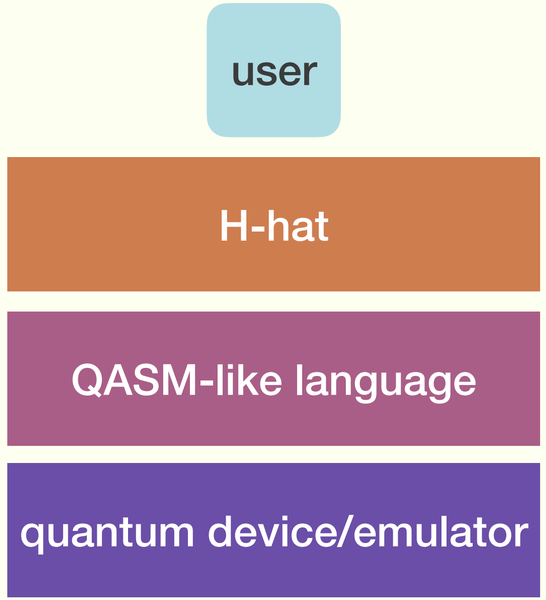
What is H-hat ($\hat{H}$) ?

- A high-level abstraction quantum programming language
- user-facing layer of the quantum stack



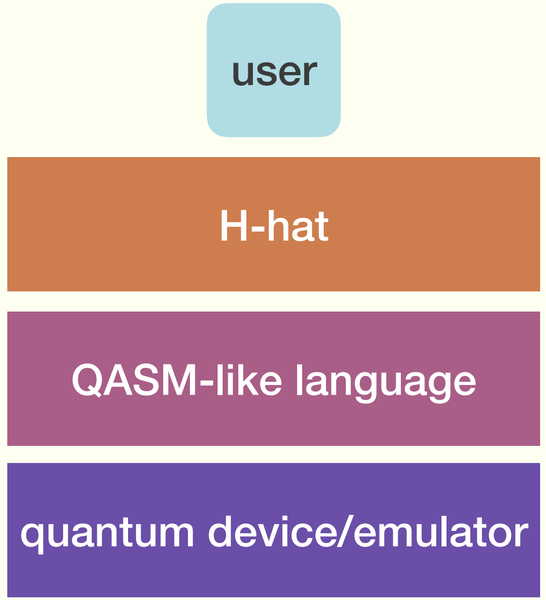
What is H-hat ($\hat{H}$) ?

- A high-level abstraction quantum programming language
- user-facing layer of the quantum stack
- data-oriented approach



What is H-hat ($\hat{H}$) ?

- A high-level abstraction quantum programming language
- user-facing layer of the quantum stack
- data-oriented approach
- closer to what programmers currently are used to



Quantum type system

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`
- Composes quantum data structure for quantum data types

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`
- Composes quantum data structure for quantum data types
  - quantum boolean (`@bool`), quantum integer (`@u2`, `@u4`), etc.

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`
- Composes quantum data structure for quantum data types
  - quantum boolean (`@bool`), quantum integer (`@u2`, `@u4`), etc.
  - custom quantum data types with structs, enums, etc. ex:

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`
- Composes quantum data structure for quantum data types
  - quantum boolean (`@bool`), quantum integer (`@u2`, `@u4`), etc.
  - custom quantum data types with structs, enums, etc. ex:

```
type @syncd_bool_t { @source:@bool @target:@bool }
```

Quantum type system

- Defines quantum data: `@0 @1 @"h" @3.14 @true`
- Composes quantum data structure for quantum data types
  - quantum boolean (`@bool`), quantum integer (`@u2`, `@u4`), etc.
  - custom quantum data types with structs, enums, etc. ex:

```
type @syncd_bool_t { @source:@bool @target:@bool }
```

```
type @teleport_bool_t {
    @data:@bool
    @remote:@conn_teleport_bool_t
}
```

Quantum type system

Quantum type system

Quantum type check

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum variable

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum variable

- appends quantum and classical instructions inside the variable stack

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum variable

- appends quantum and classical instructions inside the variable stack

```
@q:@bool = @redim(@false)
```

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum variable

- appends quantum and classical instructions inside the variable stack

```
@q:@bool = @redim(@false)
```

Quantum type system

Quantum type check

- check the number of indexes (qubits) for the quantum type recursively

Quantum variable

- appends quantum and classical instructions inside the variable stack

```
@q:@bool = @redim(@false)
```

```
@q<@bool>.data = (
  @bool::@ALLOC_INDEX,
  @bool::@REDIMENSIONALIZE(@bool::@FALSE)
  )
```

Quantum type system

Casting system

Quantum type system

Casting system

- casts quantum type into classical type, ex:

Quantum type system

Casting system

- casts quantum type into classical type, ex:

```
cast(u32 @redim(@3<@u2>))
```

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- casts quantum type into classical type, ex:

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:
    1. compiles quantum instructions into QASM-like language, ex:

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

    1. compiles quantum instructions into QASM-like language, ex:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
x q[0];
x q[1];
h q[0];
h q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

    2. executes them on emulator/QPU and retrieves measurement (dict):

```
{"00": 500, "01": 500, "10": 500, "11": 500}
```

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

    3. transforms the result according to a protocol (raw, weighted average, highest or lowest value):

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

    3. transforms the result according to a protocol (raw, weighted average, highest or lowest value):

```
raw => {"00": 450, "01": 510, "10": 550, "11": 490}

weighted_average (1.54) => "10"

highest (550) => "10"

lowest (450) => "00"
```

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- `cast`:

    4. casts it to the chosen classical type:

Quantum type system

```
cast(u32 @redim(@3<@u2>))
```

- **cast**:

    4. casts it to the chosen classical type:

```
raw, dict => not compatible with u32

weighted_average, "10" => 2<u32>

highest, "10" => 2<u32>

lowest, "00" => 0<u32>
```

Quantum type system

Quantum type system

There are many other points not covered here, such as

Quantum type system

There are many other points not covered here, such as

1. Index manager

Quantum type system

There are many other points not covered here, such as

1. Index manager
2. Quantum functions conversion to quantum instructions

Quantum type system

There are many other points not covered here, such as

1. Index manager
2. Quantum functions conversion to quantum instructions
3. Quantum instructions execution order

Quantum type system

There are many other points not covered here, such as

1. Index manager
2. Quantum functions conversion to quantum instructions
3. Quantum instructions execution order
4. ...

Quantum type system

There are many other points not covered here, such as

1. Index manager
2. Quantum functions conversion to quantum instructions
3. Quantum instructions execution order
4. ...

Quantum type system

There are many other points not covered here, such as

1. Index manager
2. Quantum functions conversion to quantum instructions
3. Quantum instructions execution order
4. ...

Reach me out to know more about the project!

github.com/hhat-lang