Bachelor Proposals

Logic and Semantics & Programming Languages

Andreas Pavlogiannis

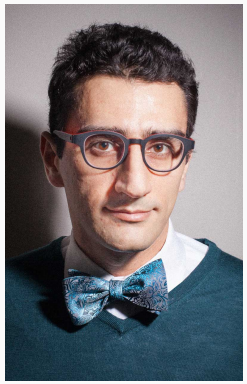
November, 2023

AARHUS UNIVERSITY

LS & PL Members

Logic and Semantics






Programming Languages



All Things Programming Languages, Analysis and Verification

- Programming languages
- Logic, compilers
- Type systems, type inference
- Security, Coq
- ...

- Automata and formal languages
- Algorithmic verification
- Symbolic model checking
- Static program analysis
- ...

Property Based Testing

How do I check that my program is correct?

- Program logics
- Static analysis
- ...
- Property Based Testing
- Random testing

Property Based Testing

How do I check that my program is correct?

- Program logics
- Static analysis
- ...
- Property Based Testing
- Random testing

- Instead of a full program spec, test a few key properties $\phi = \psi_1 \rightarrow \psi_2$
- If input satisfies ψ_1, program execution satisfies ψ_2

Property Based Testing

```
// Transitivity
def LessThan(x:a, y:b): Bool={
...
}
```

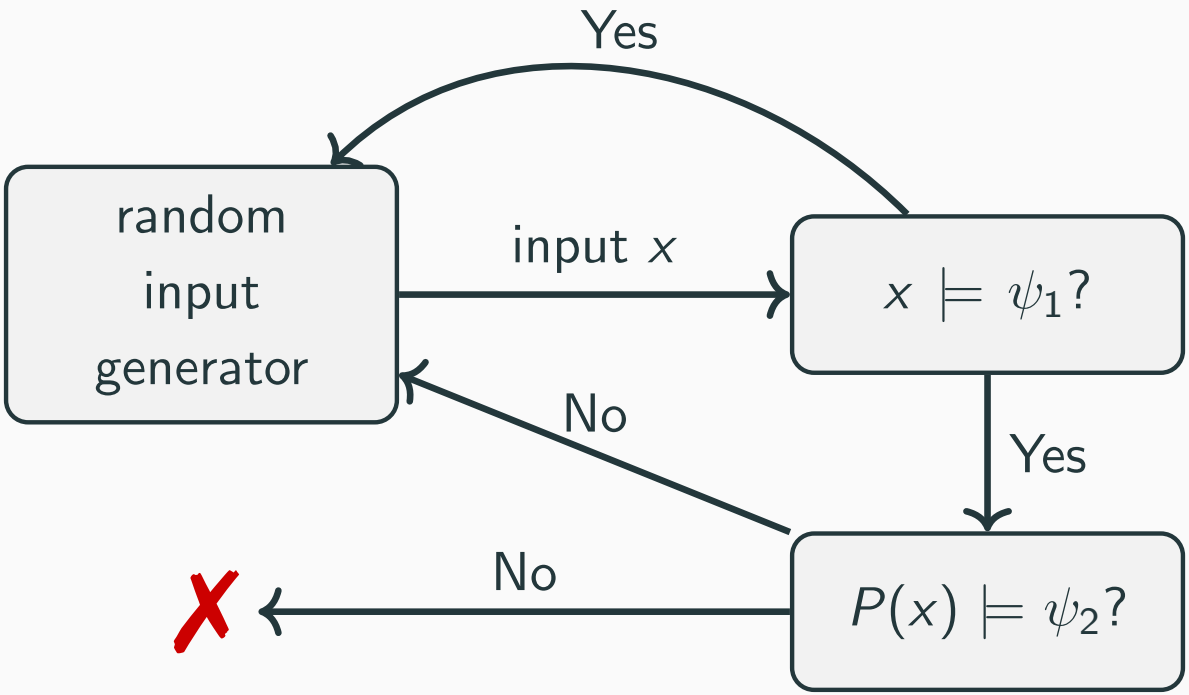
$$\texttt{LessThan}(x, y) \land \texttt{LessThan}(y, z) \implies \texttt{LessThan}(x, z)$$

```
// Idempotence
def SetUnion(x:Set[a], y:Set[a]): Set[a]={
...
}
```

$$x = y \implies \texttt{SetUnion}(x, y) = x$$

Property Based Testing

Testing program P for property $\phi = \psi_1 \rightarrow \psi_2$



Property Based Testing – in Flix

flix

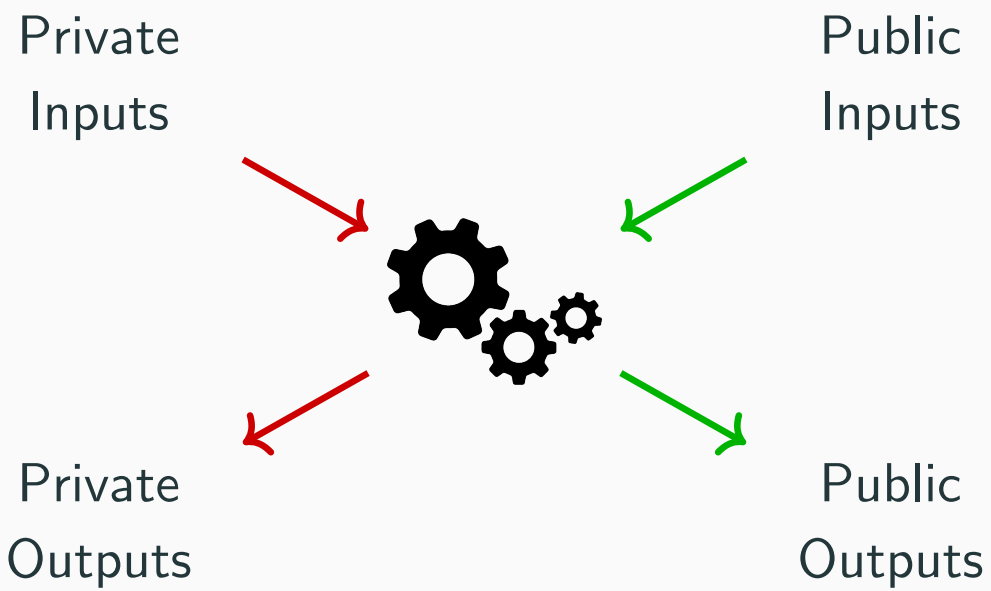
Unique combination of features:

- Algebraic data types
- Polymorphic effects
- First-class Datalog constraints
- ...

```
// The purity of `map` depends on the purity of `f`
def map(f: a->b \ ef, l: List[a]): List[b] \ ef =
    match l {
        case Nil     => Nil
        case x :: xs => f(x) :: map(f,xs)
    }
```

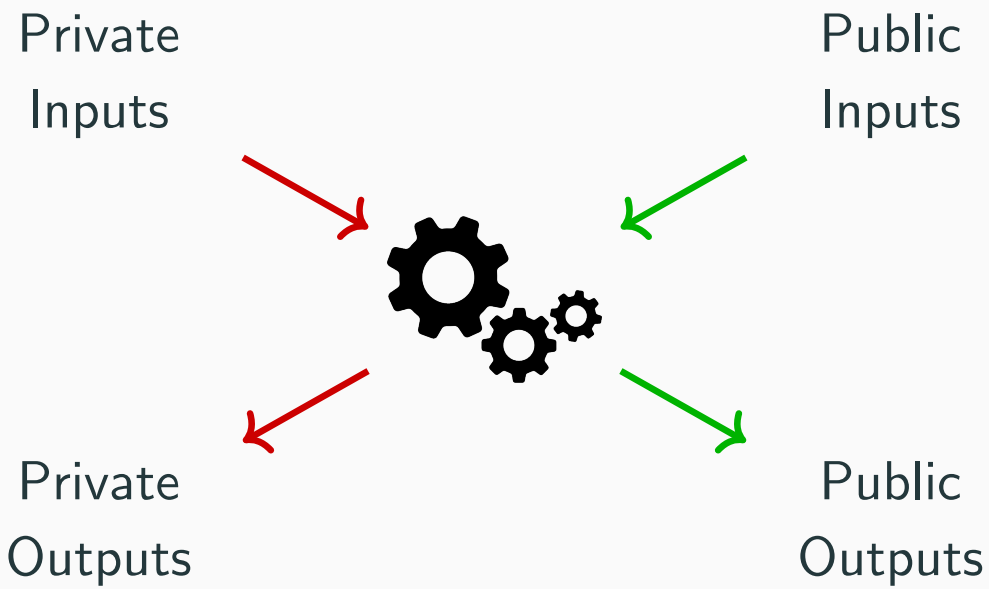
Property Based Testing – in Troupe

A language with runtime security monitoring



Property Based Testing – in Troupe

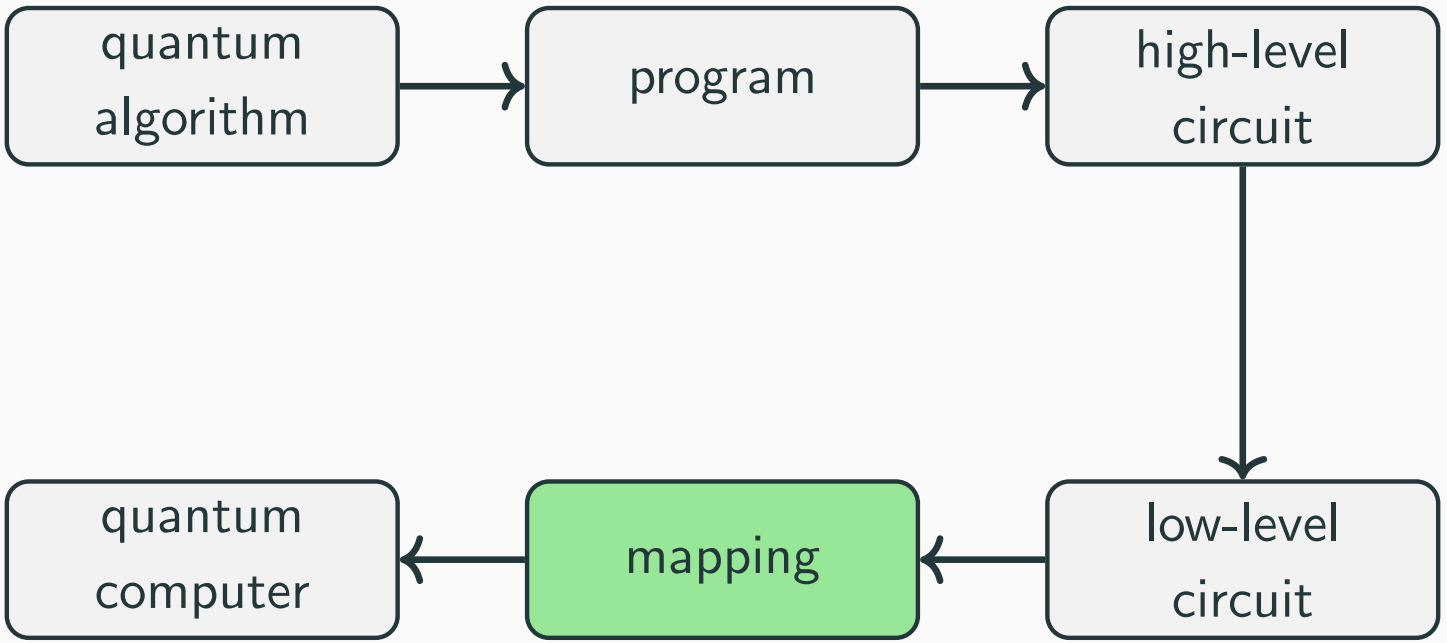
A language with runtime security monitoring



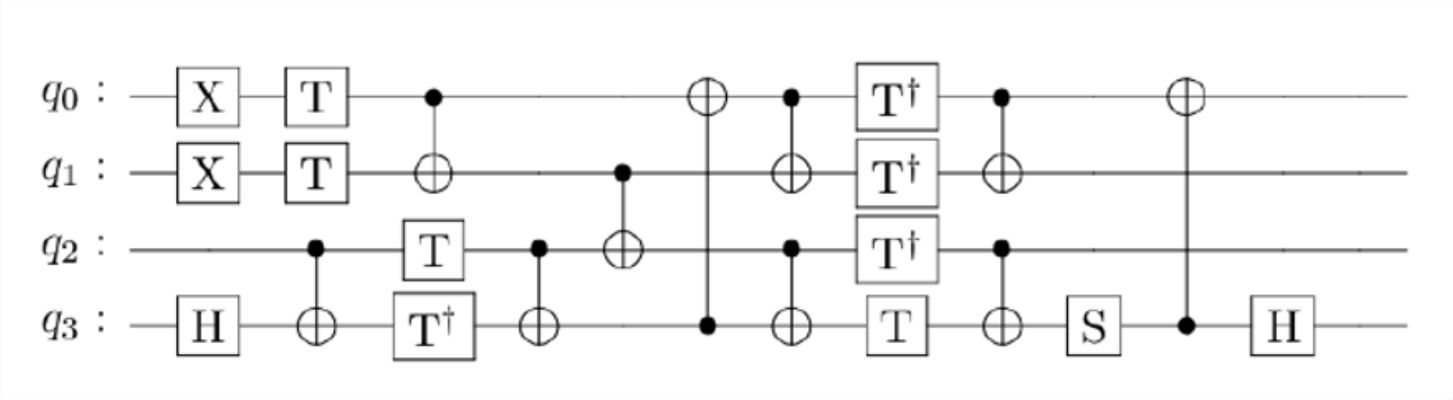
Non-interference

$$\forall x, y : (x.\text{public} = y.\text{public}) \implies (P(x).\text{public} = P(y).\text{public})$$

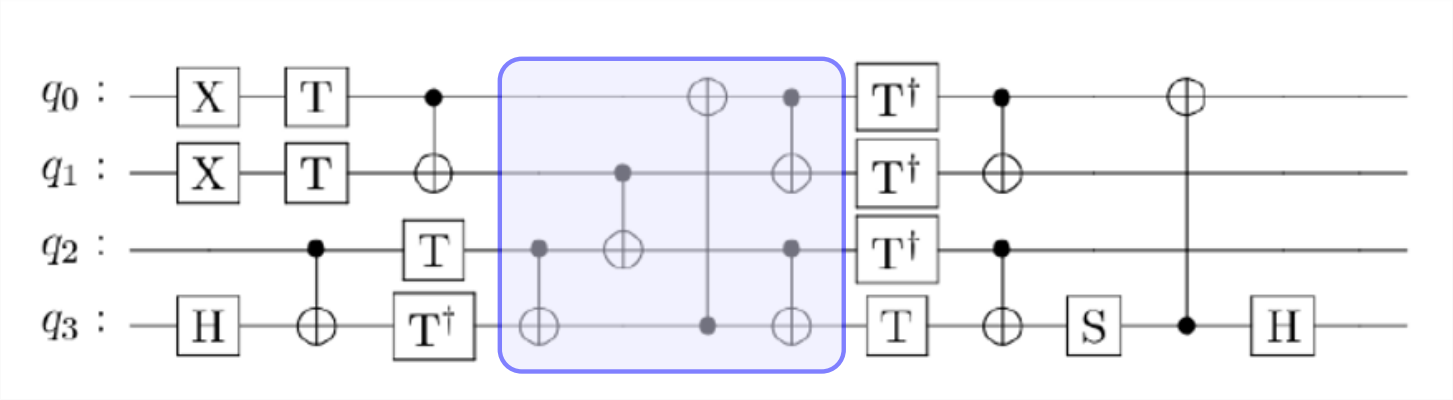
Quantum Layout Synthesis



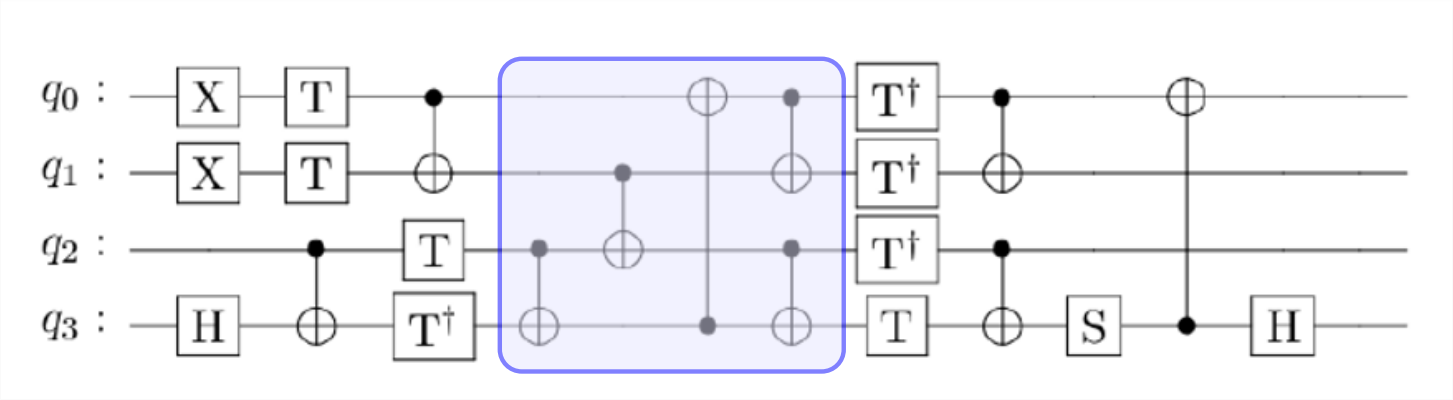
Quantum Layout Synthesis



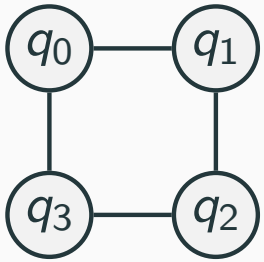
Quantum Layout Synthesis



Quantum Layout Synthesis

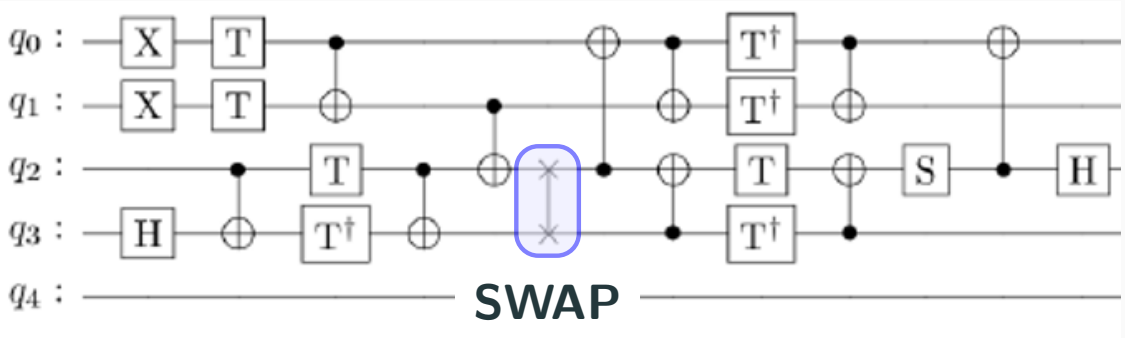


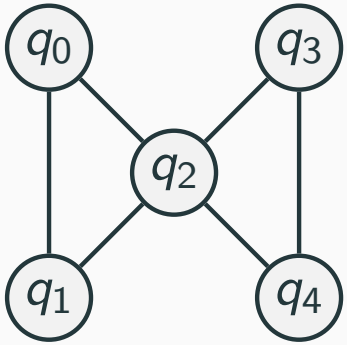
Requires



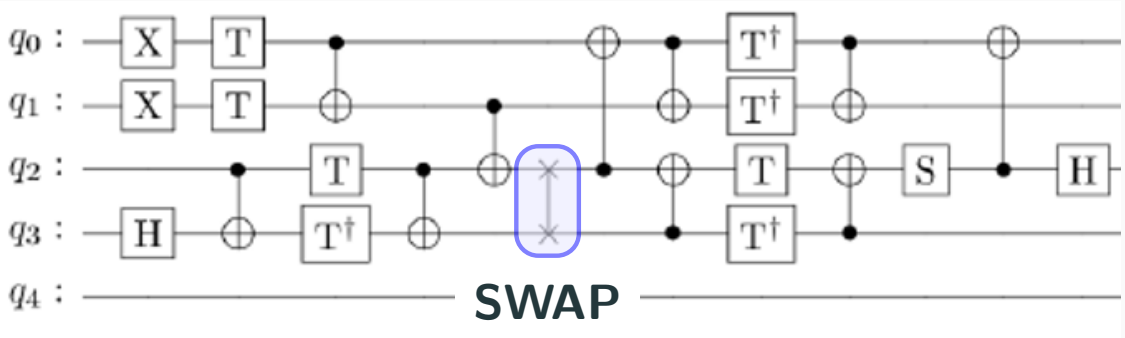
What if we only have?

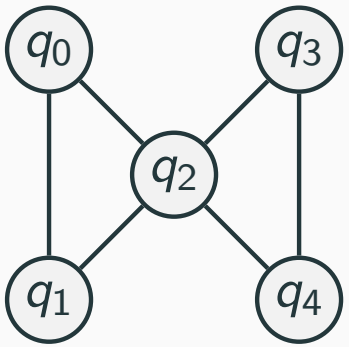
Quantum Layout Synthesis





Quantum Layout Synthesis





- Can a topology support a quantum circuit?
- What is the minimal number of **SWAP**s needed?
- What is the optimal circuit for a given topology?

Dependent Types

"Well-typed programs cannot go wrong." – Robin Milner

Well...

```
var x:Int;
...
//what if x is 0?
println(42/x);
```

```
def get(a:Array[Int], i:Int):Int={
 return a(i);
}
var a: Array[Int];
...
//What if i>=a.len?
println(get(a,i));
```

Dependent Types

"Well-typed programs cannot go wrong." – Robin Milner

Well...

```
var x:Int;
...
//what if x is 0?
println(42/x);
```

```
def get(a:Array[Int], i:Int):Int={
 return a(i);
}
var a: Array[Int];
...
//What if i>=a.len?
println(get(a,i));
```

Q: How can we prevent such errors at compile time?

Dependent Types

A: Dependent/Refinement Types!

```
var x:Int[v|v!=0];
...
//what if x is 0?
println(42/x);
```

```
def get(a:Array[Int], i:Int[v|v<a.len]):Int={
 return a(i);
}
var a: Array[Int];
...
//What if i>=a.len?
println(get(a,i));
```

Dependent Types

A: Dependent/Refinement Types!

```
var x:Int[v|v!=0];
...
//what if x is 0?
println(42/x);
```

```
def get(a:Array[Int], i:Int[v|v<a.len]):Int={
 return a(i);
}
var a: Array[Int];
...
//What if i>=a.len?
println(get(a,i));
```

- How do I design a sound type system around this concept?
- How do I typecheck programs (efficiently)?
- What kinds of type dependencies/refinements are allowed?