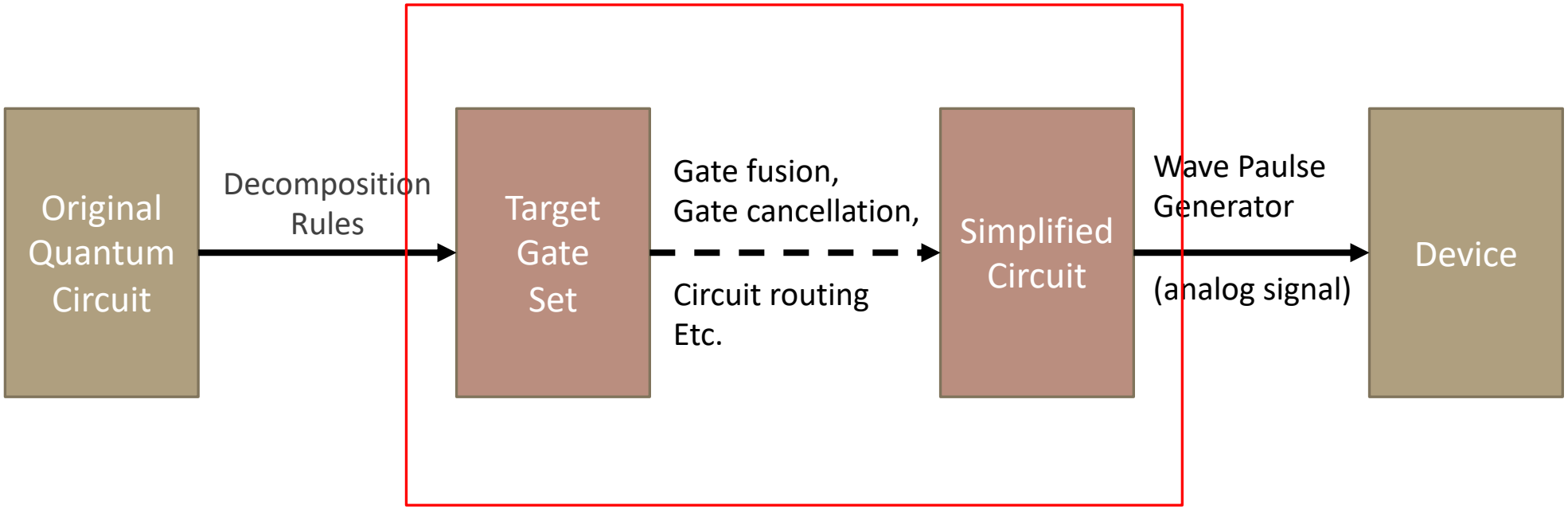
Quantum Compiler Optimization

CIRCUIT COMPILATION & SIMPLIFICATION

XIUZHE (ROGER) LUO@PIQUIL, WATERLOO

2019-12-19

Quantum Circuit Compilation



Pattern Match & Term Rewrite

MANIPULATING SYMBOLIC EXPRESSIONS

XIUZHE (ROGER) LUO@PIQUIL, WATERLOO

2019-12-19

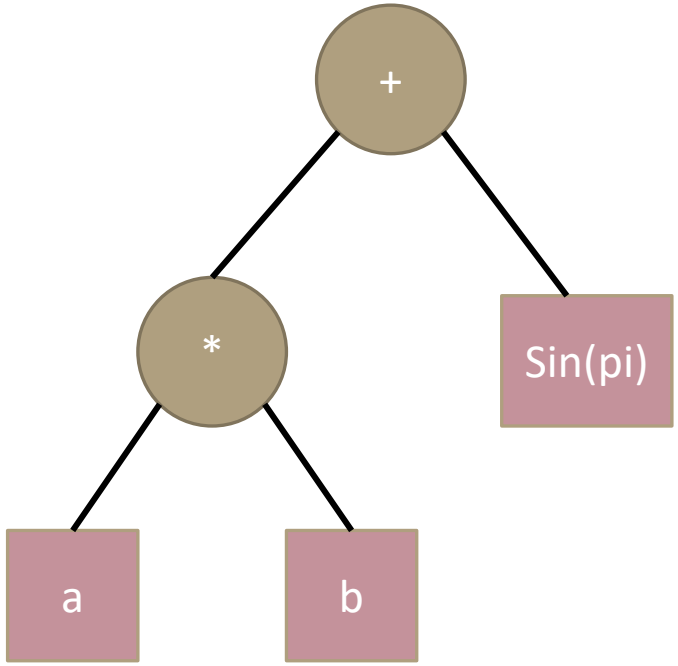
Symbolic Simplification 101

a * b + sin(pi)

Algorithm 3.1 Syntactic Matching

Input: Subject $s \in \mathcal{T}$, pattern $p \in \mathcal{T}$, and an initial substitution $\sigma \in \mathcal{S}ub$.
Output: The match σ or $\lightning$ iff there is no match.

```
1  function SyntacticMatch(s, p, σ)
2      if p ∈ 𝒳 then return {p ↦ s}
3      if p = f(p_1, ..., p_n) and s = f(s_1, ..., s_n) for some f ∈ ℱ, n ∈ ℕ then
4          for all i = 1...n do
5              σ' ← SyntacticMatch(s_i, p_i, σ)
6              if σ' = ↯ or σ' ⋬ σ then return ↯
7              σ ← σ ⊔ σ'
8          return σ
9      return ↯
```

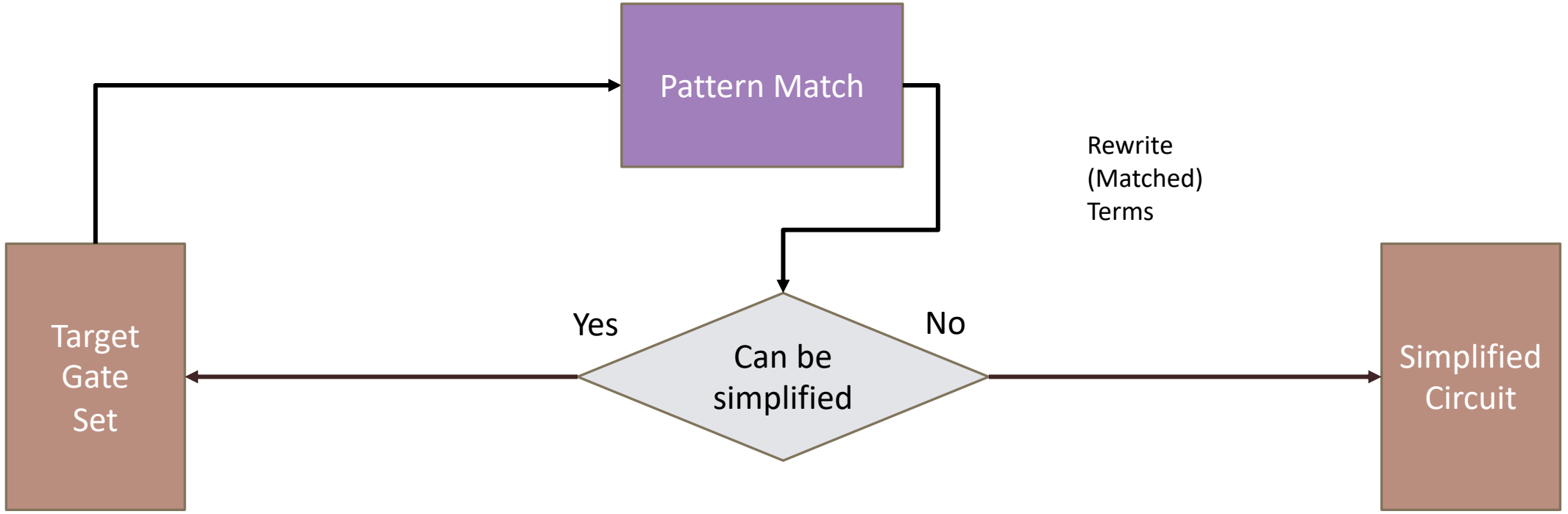


Definition

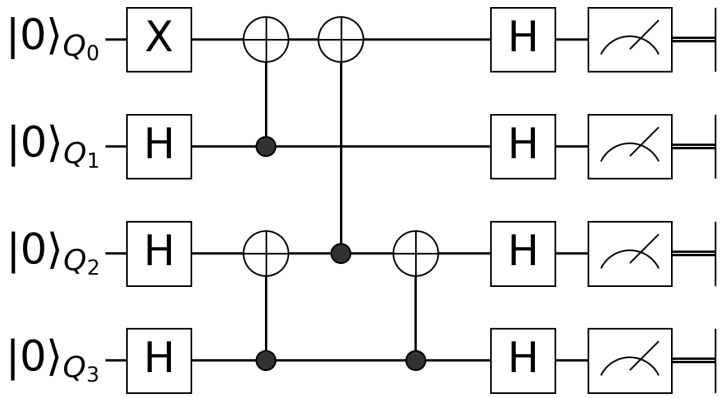
Substitution: mapping $\sigma: V \rightarrow T$ from Variable to terms, that $\sigma\big(f(t_1, \cdots, t_n)\big) = f(\sigma(t_1), \cdots, \sigma(t_n))$, and $\sigma(x) = x \; if \; x \notin Dom(\sigma)$

Match: given pattern term t, it matches a subject term s, iff there exists a substitution σ such that $\sigma(t) = s$

Circuit Optimization/Simplification

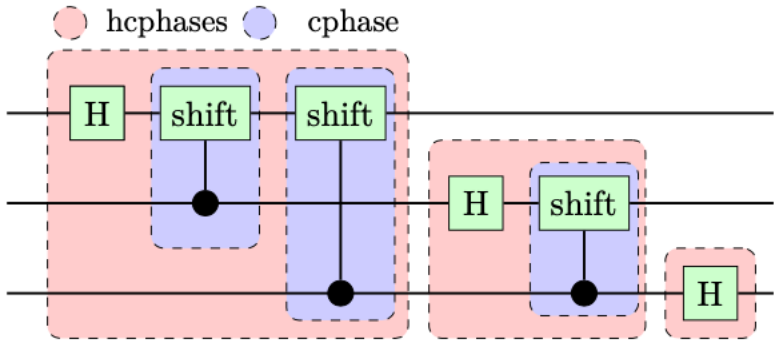


Graph Pattern Match?

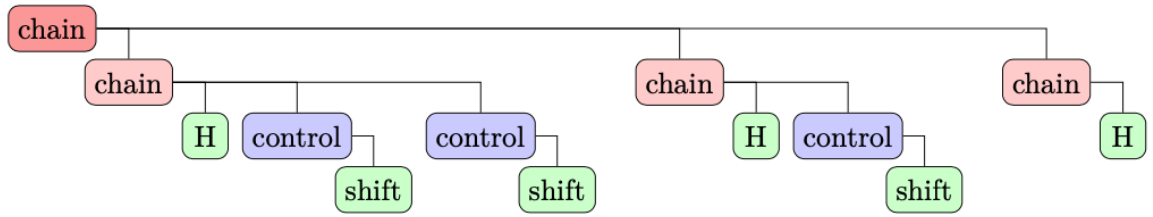


- No
- (Exact) Graph Pattern Match is subgraph isomorphism problem
- Subgraph isomorphism problem is NP-complete (Stephen A Cook. et al STOC’ 71)

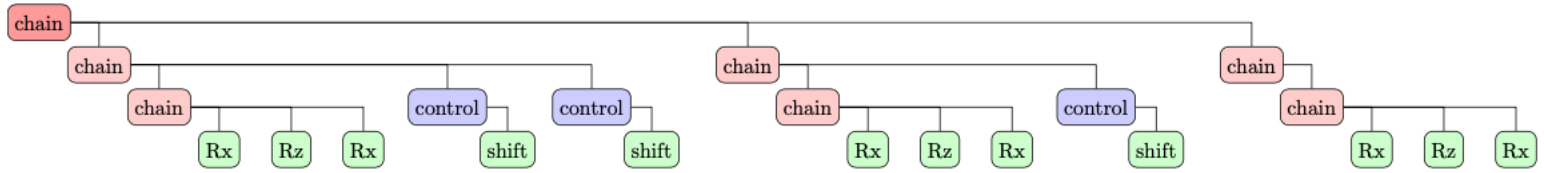
Quantum Circuit as Expression



parse



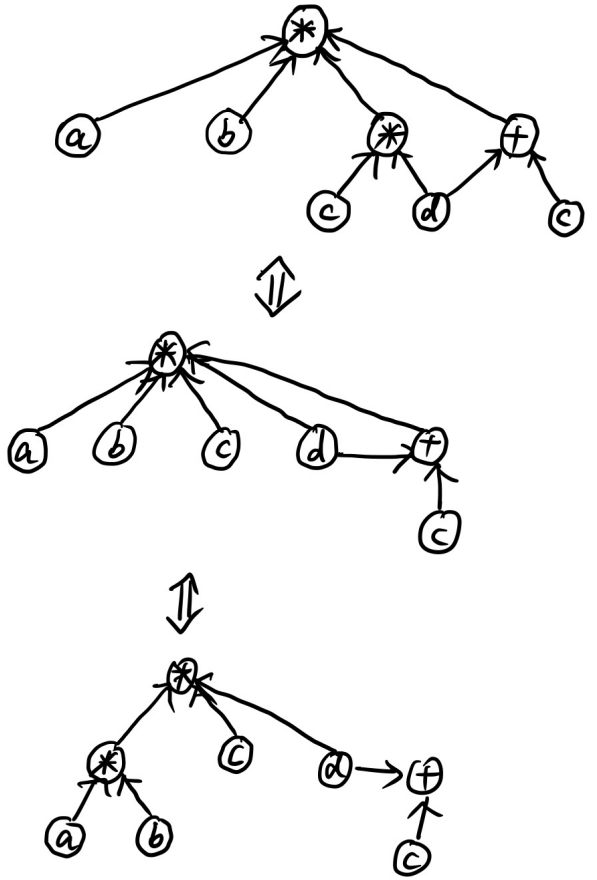
decomposition



Associative Match $O(\binom{N}{m})$

a * b * (c * d) * (d + c)

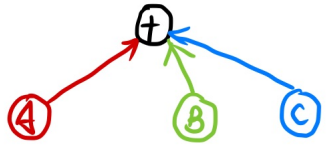
```
function match(::Type{Associative}, p::Expr, s::Expr, 0)
    @assert p.head === s.head === :call

    pname, pargs = p.args[1], p.args[2:end]
    sname, sargs = s.args[1], s.args[2:end]
    0 = match(Term, pname, sname, 0)

    m, n = length(pargs), length(sargs)
    m > n && return zero(Match)
    n_free = n - m
    n_vars = count(x -> x isa Variable, pargs)
    0ᵣ = zero(Match)

    for k ∈ Iterators.product((0:n_free for i ∈ 1:n_vars)...)
        (isempty(k) ? 0 : sum(k)) == n_free || continue  # FIXME: efficiency
        i, j = 1, 1
        0′ = 0
        for p□ ∈ pargs
            l_sub = 0
            if p□ isa Variable
                l_sub += k[j]
                j += 1
            end
            s′ = l_sub > 0 ? Expr(:call, sname, sargs[i:i+l_sub]...) : sargs[i]
            0′ = match(Term, p□, s′, 0′)
            isempty(0′) && break
            i += l_sub + 1
        end
        0ᵣ = 0ᵣ ∪ 0′
    end
    0ᵣ
end
```



Communitive Match O(N!)

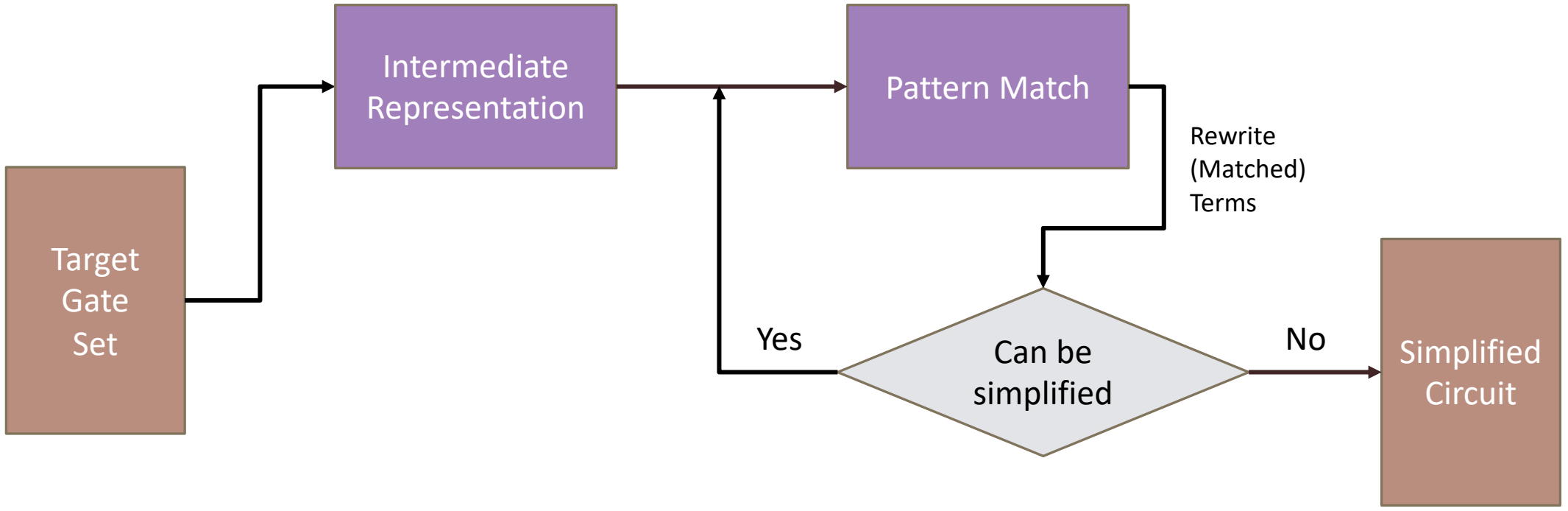
```
function match(::Type{Commutative}, p::Expr, s::Expr, 0)
    @assert p.head === s.head === :call

    matches = map(perms(s)) do s′  # FIXME: efficiency
        isvalid(Associative(s.args[1])) && return match(Associative, p, s′, 0)
        _match(p, s′, 0)
    end
    reduce(union, matches)
end
```

A + B + C + (A + D) + (A + D+ D+ B)



1. https://github.com/Roger-luo/Sym.jl
2. https://github.com/QuantumBFS/YaoBlocks.jl/pull/78

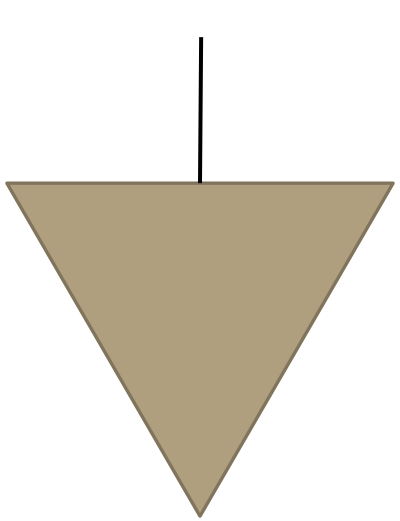
Circuit Optimization/Simplification



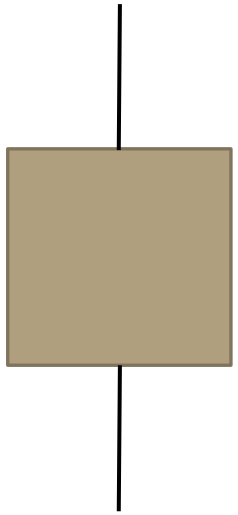
Tensor Network

A QUICK GUIDE

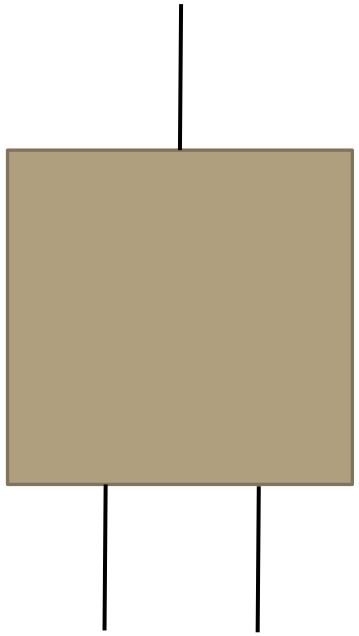
From Tensors to Networks



Vector



Matrix



Cube

ZX Diagram & Calculus

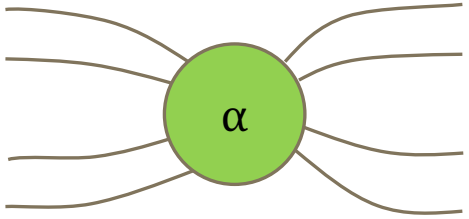
A GRAPHIC REPRESENTATION OF QUANTUM CIRCUITS

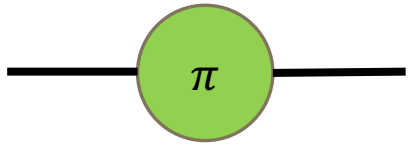
XIUZHE (ROGER) LUO@PIQUIL, WATERLOO

2019-12-19

Semantics

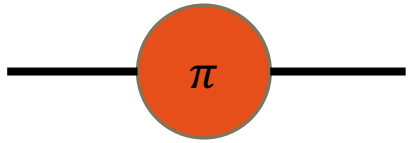
Z spider $|0 \ldots 0\rangle\langle 0 \ldots 0| + e^{i\alpha}|1 \ldots 1\rangle\langle 1 \ldots 1|$



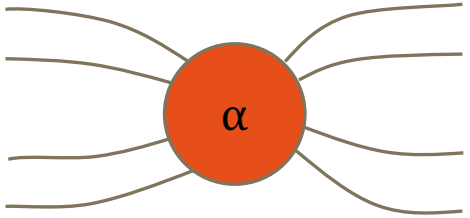


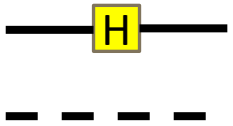
Z

X Spider $|+ \ldots +\rangle\langle + \ldots +| + e^{i\alpha}|- \ldots -\rangle\langle - \ldots -|$

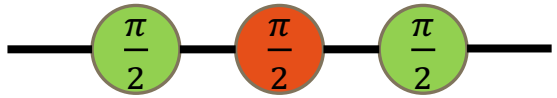


X

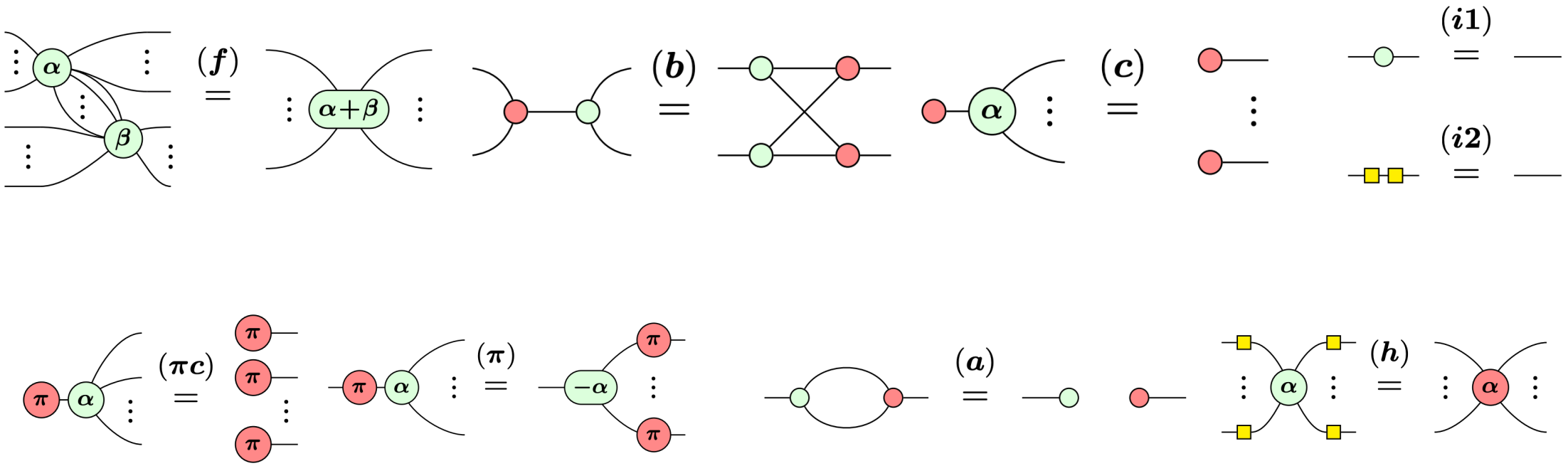




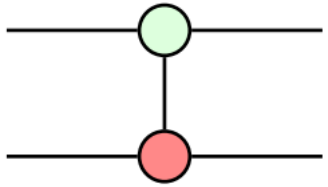
Hadamard gate



Straightforward Rules

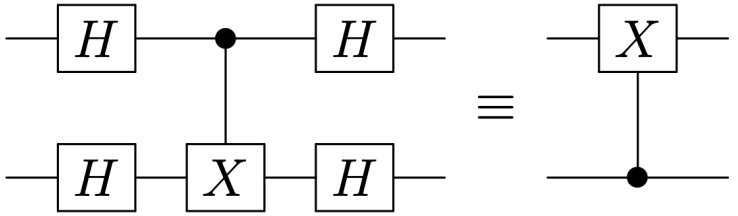


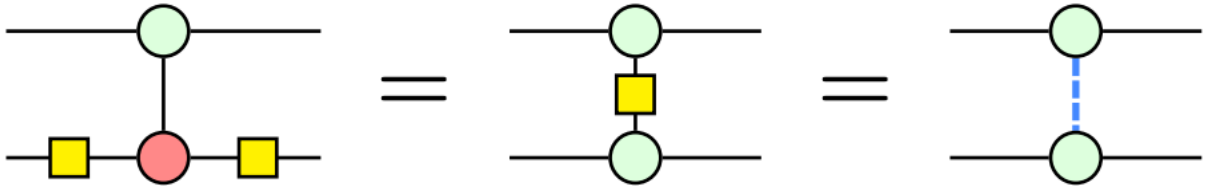
Straightforward Rules



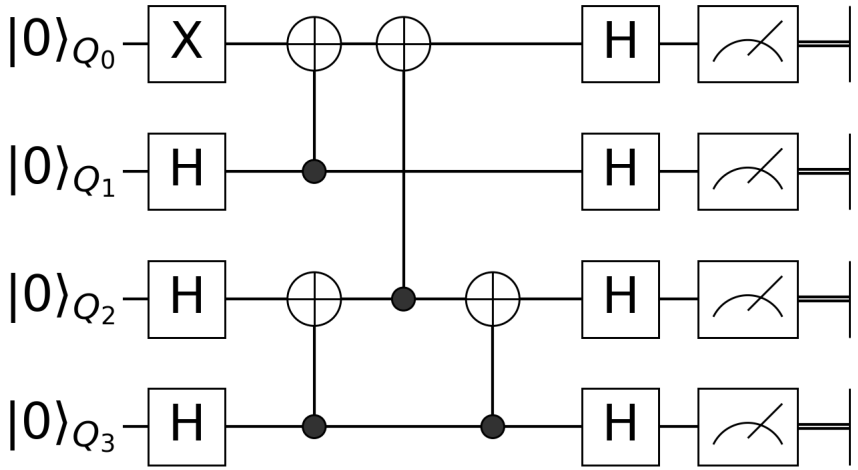
CNOT

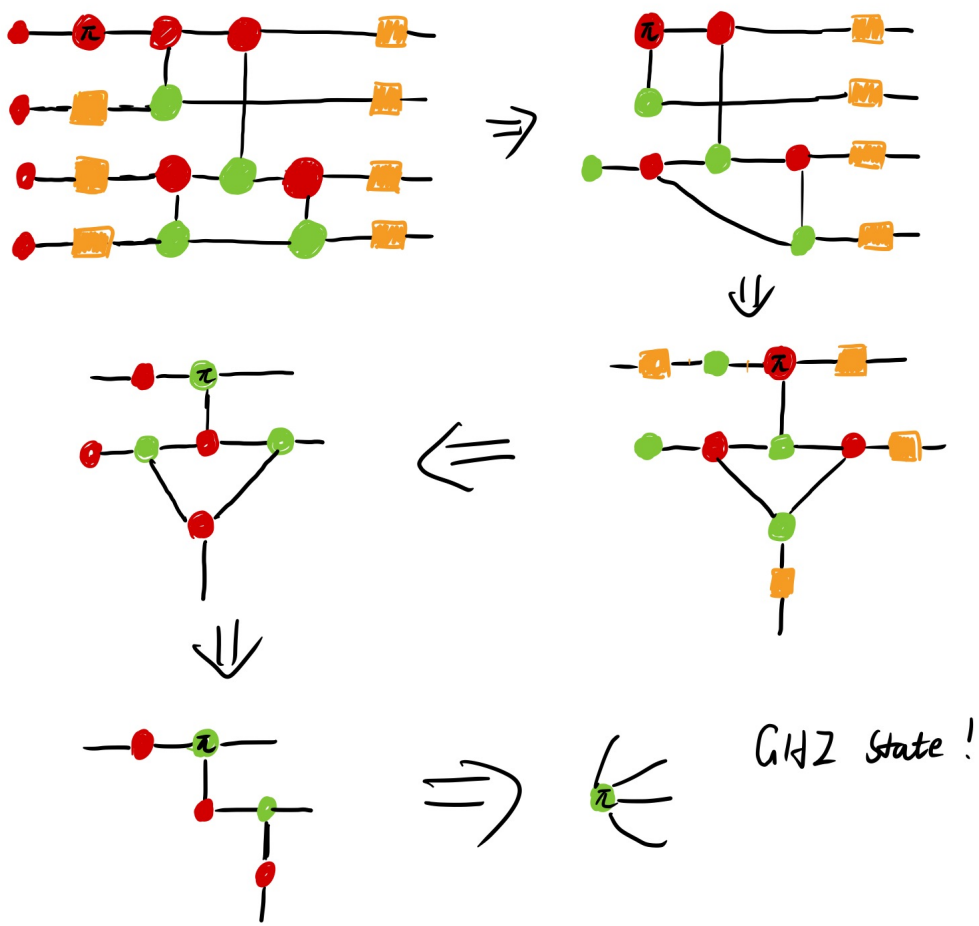
CZ





Constructing GHZ state

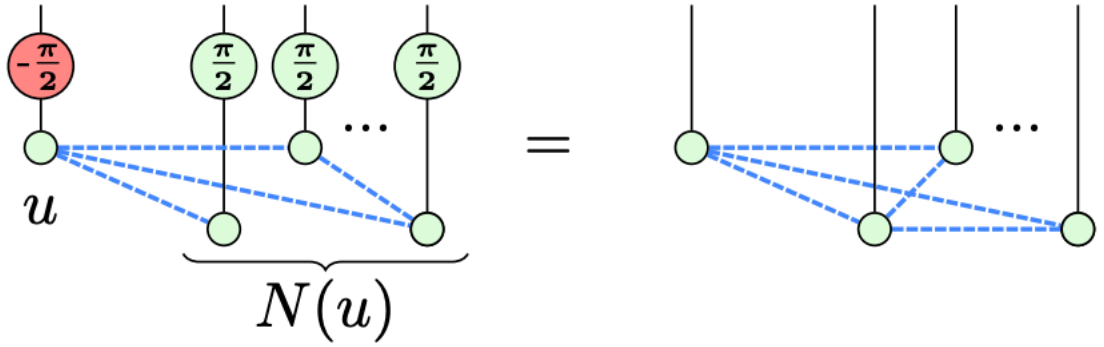




Canonicalize: graph-like ZX-diagram

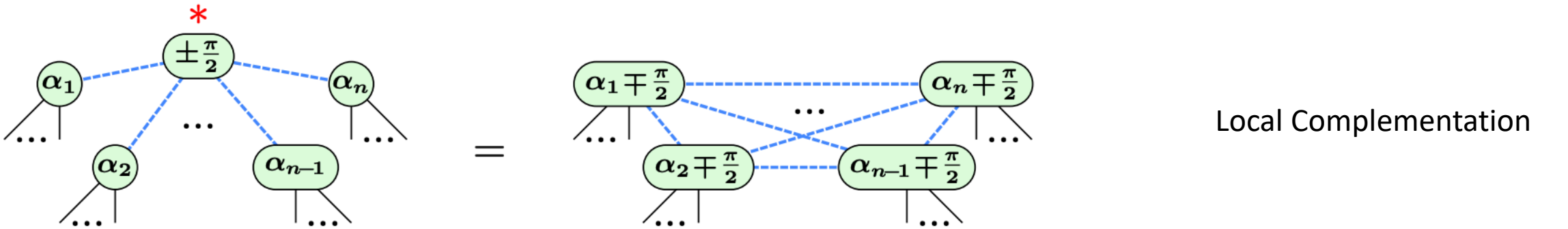
Graph State

A quantum state that constructed by CZ gate and each CZ gate represents an edge in the graph.

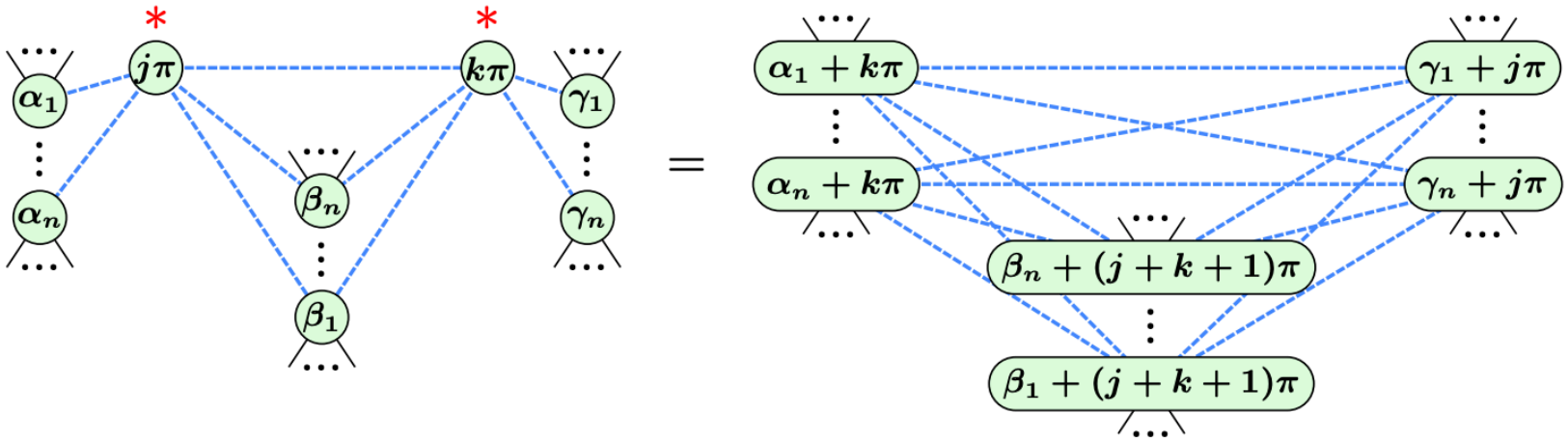


Definition: **Local Complementation** Let G be a graph and let *u* be a vertex in the graph. The local complement according to *u*, written as G ★ u is a graph which has the same vertices as G, but the sub-graph on its neighrbors is a complement graph of the original.

Two more rules



Local Complementation



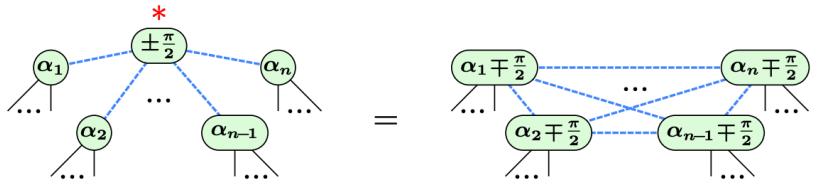
Graph Pivoting

Theorem: exists a terminating procedure to turn any graph-like ZX diagram to another ZX-diagram which does not contain

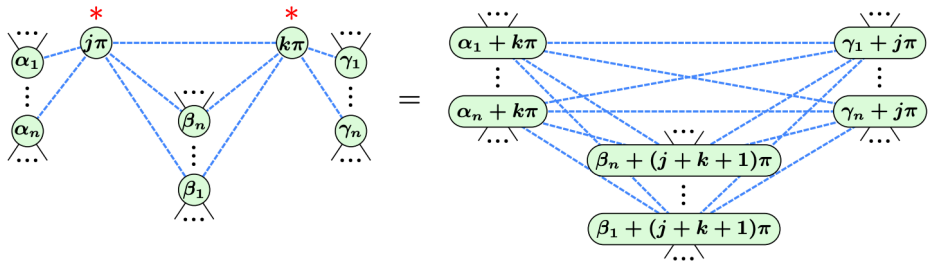
1. Interior proper Clifford spiders
2. Adjacent pairs of interior Pauli spiders
3. Interior Pauli spiders adjacent to a boundary spider

Proof

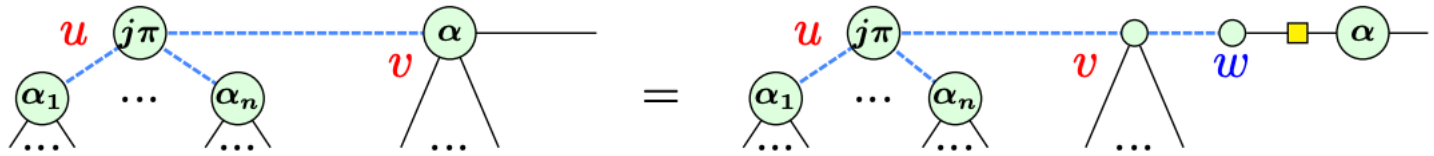
1. Remove marked Clifford



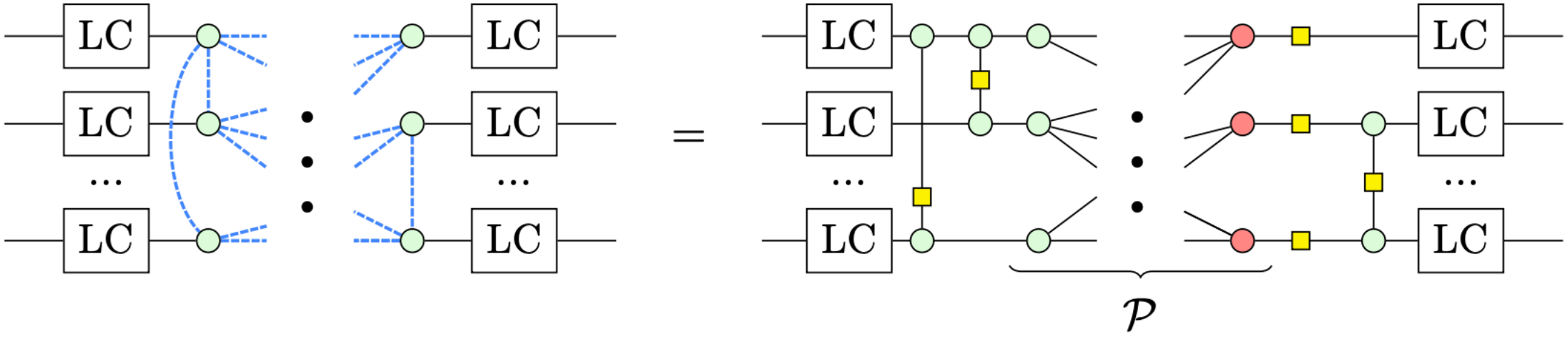
2. Remove adjacent pair of interior Pauli



3. Remove Interior Pauli adjacent to boundary



Extract The Circuit



Outlook

- How general quantum program (with classical control flows) can be compiled and modified?
- Is it possible to perform more hardware related simplification task? (e.g reducing long range interactions)
- Pattern Match on Quantum Circuits with Transformer model (arXiv: 1912.01412)

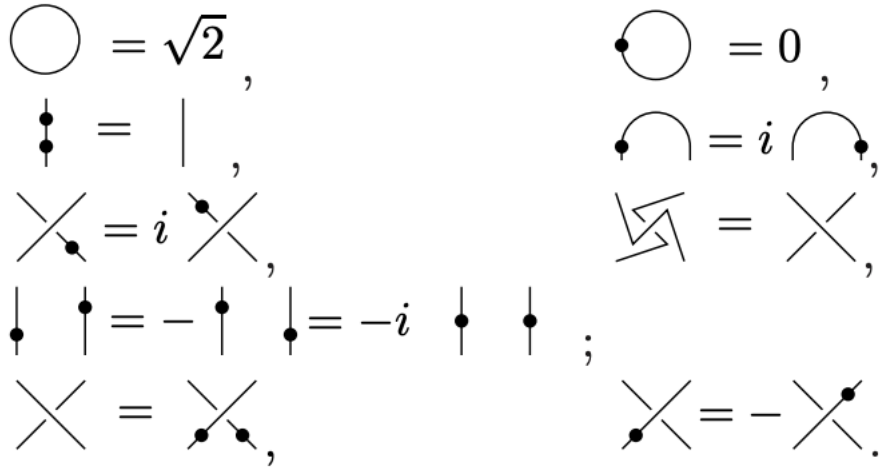
Quon

A 3D TOPOLOGICAL LANGUAGE FOR QUANTUM INFORMATION

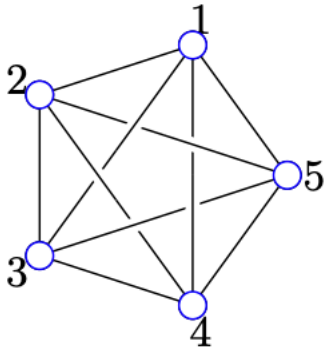
XIUZHE (ROGER) LUO@PIQUIL, WATERLOO

2019-12-19

Semantics



$v_\emptyset =$



$v_{E(\Gamma)} =$