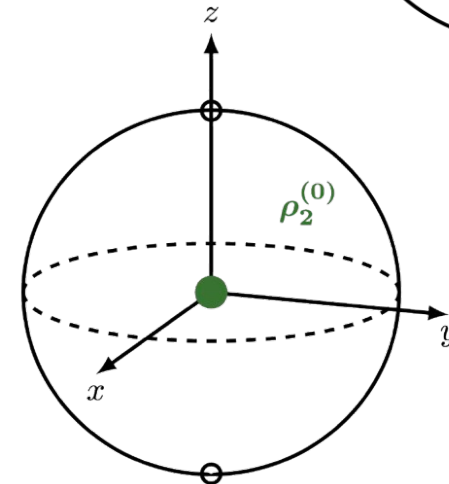
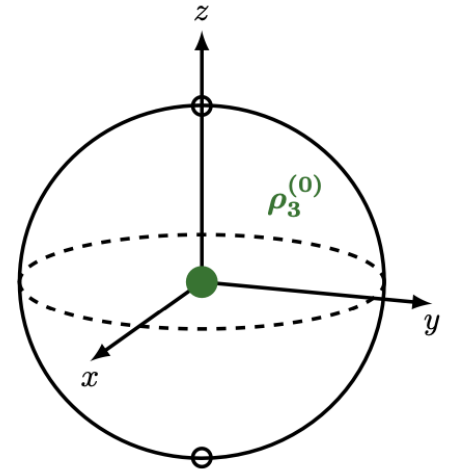
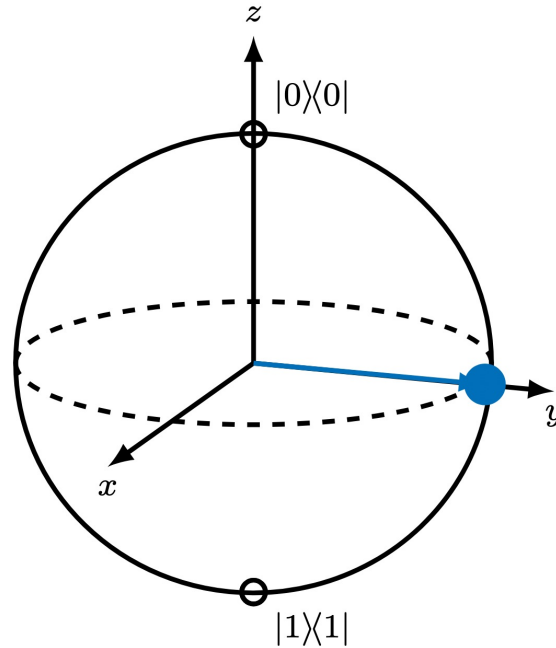


Decoherence, Predictability, and Robustness: Investigating the Preferred Division of a Quantum System into Subsystems



Aiden Karpf

Advised by Prof. Kevin Setter

Pomona College Thesis Presentation 2025

Quantum vs Classical Worlds



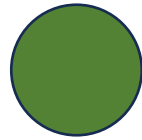
electron

$|0\rangle$



$|1\rangle$

Quantum vs Classical Worlds

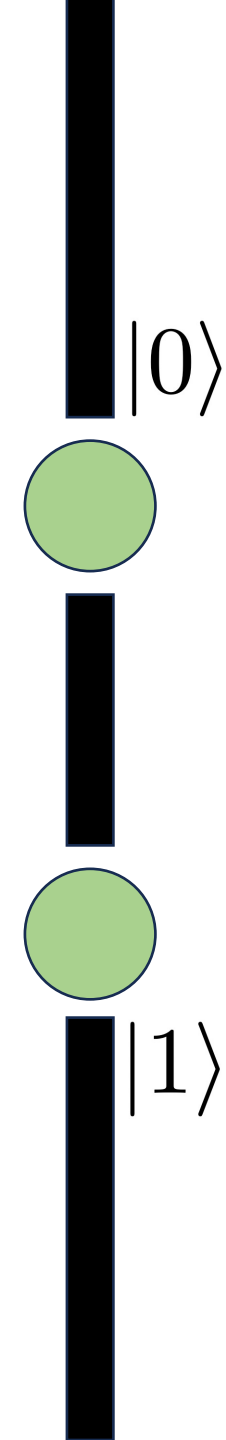


$|0\rangle$



$|1\rangle$

Quantum vs Classical Worlds



Quantum vs Classical Worlds

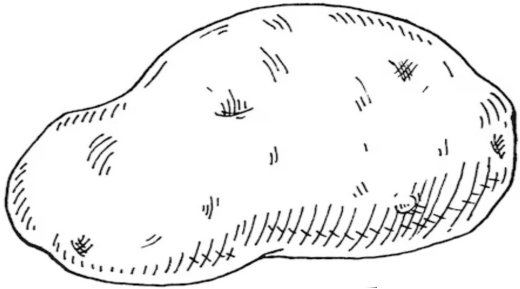


$|0\rangle$



$|1\rangle$

Quantum vs Classical Worlds



potato

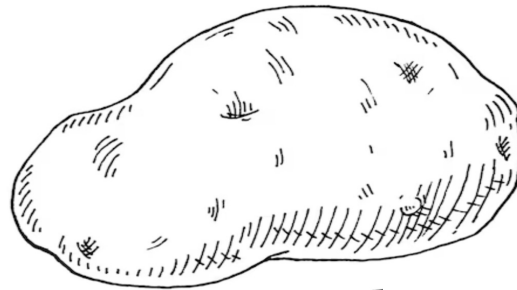
$|0\rangle$



$|1\rangle$

Quantum vs Classical Worlds

$|0\rangle$



$|1\rangle$

Quantum vs Classical Worlds



Quantum vs Classical Worlds

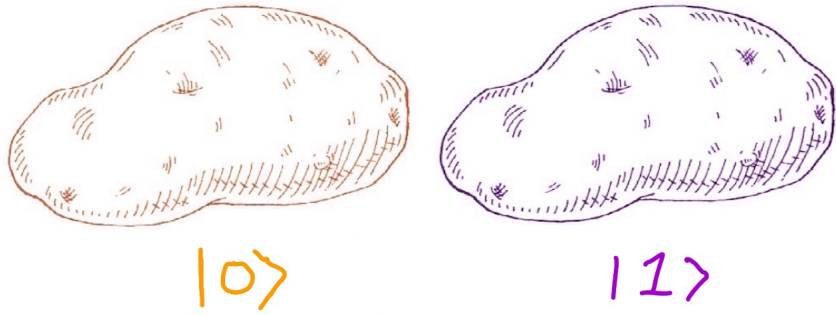


Quantum vs Classical Worlds

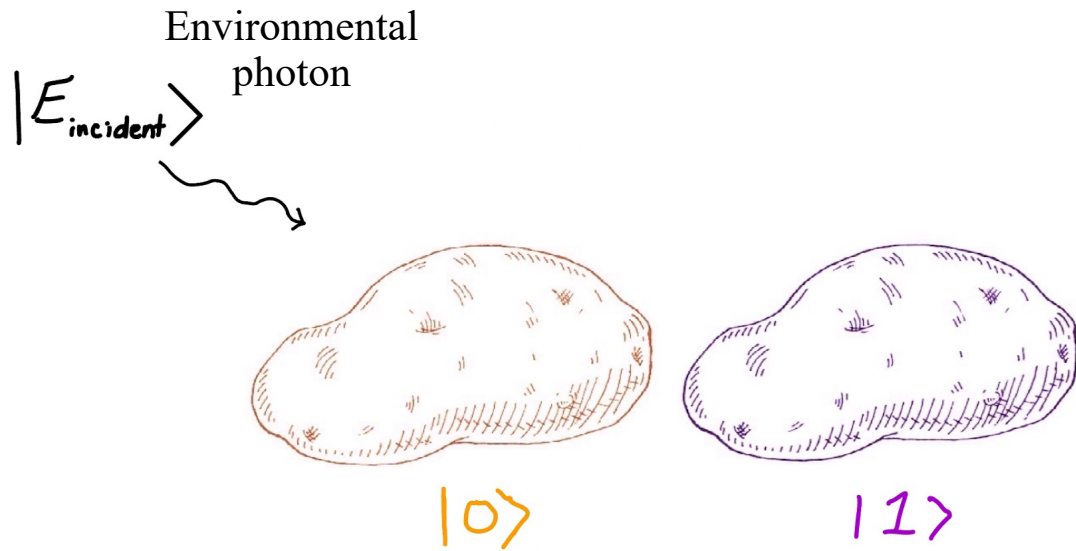
WHY NOT??



Potato in a Superposition State



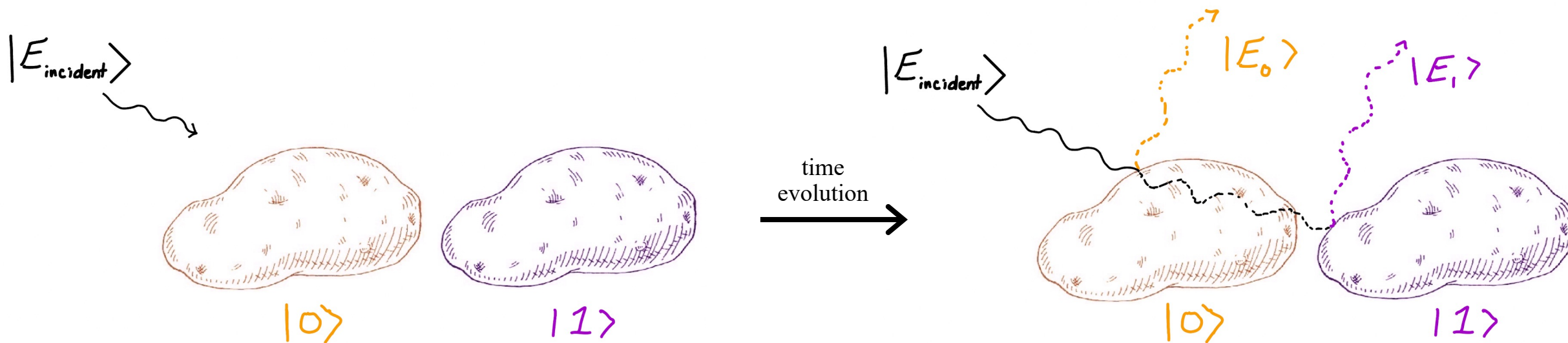
Potato in a Superposition State



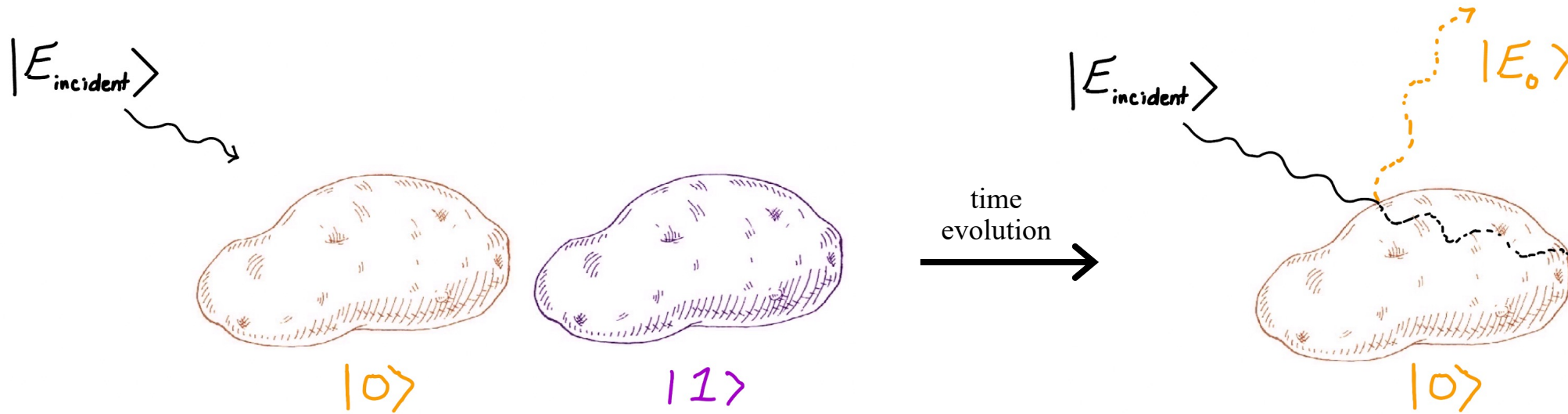
Potato in a Superposition State



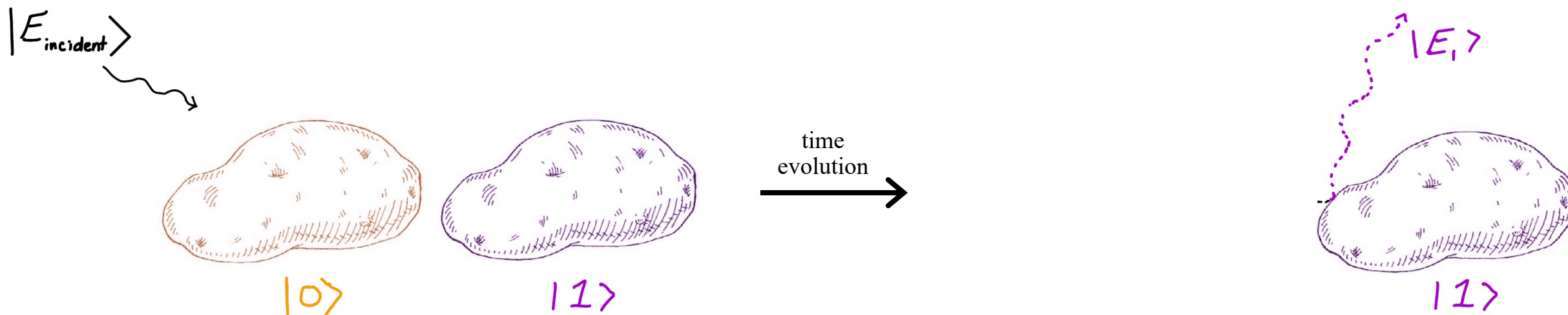
Potato in a Superposition State



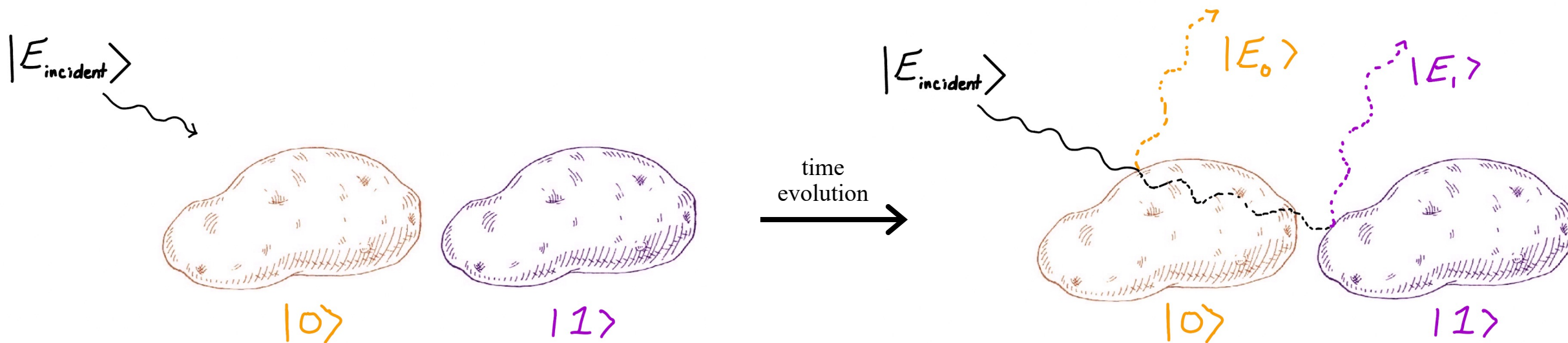
Potato in a Superposition State



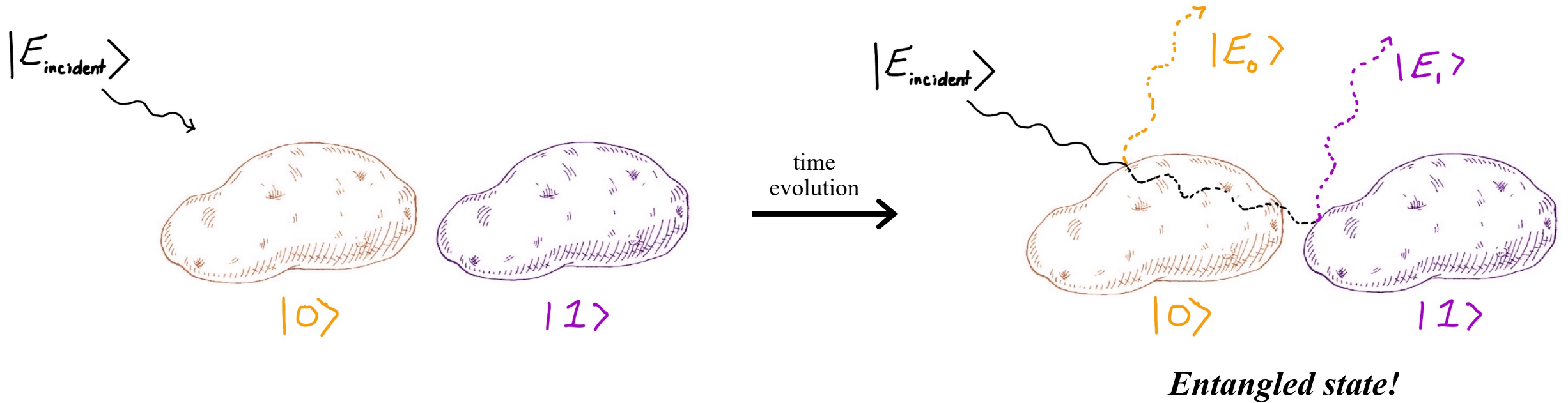
Potato in a Superposition State



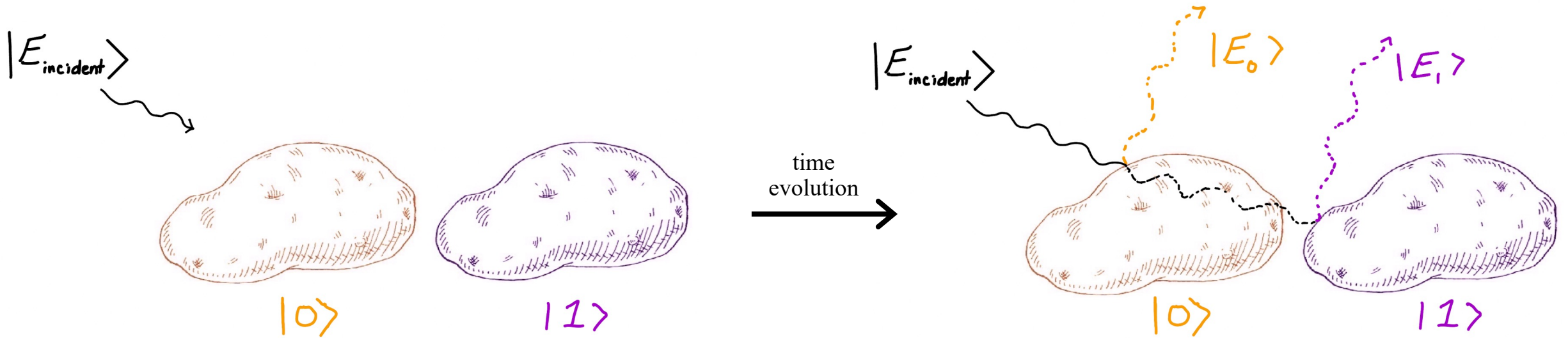
Potato in a Superposition State



Potato in a Superposition State



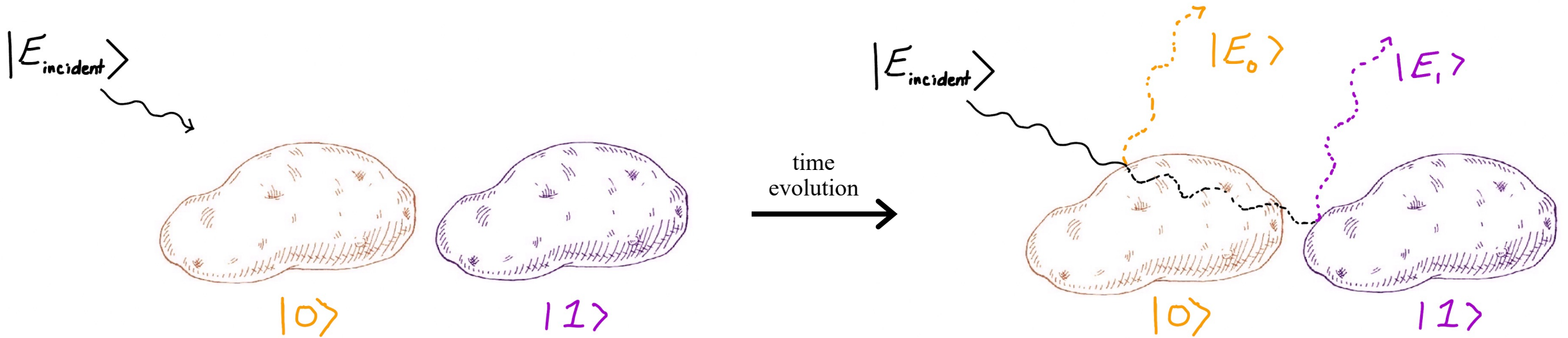
Potato in a Superposition State



Entangled state!

- The photon and potato have lost their individuality

Potato in a Superposition State

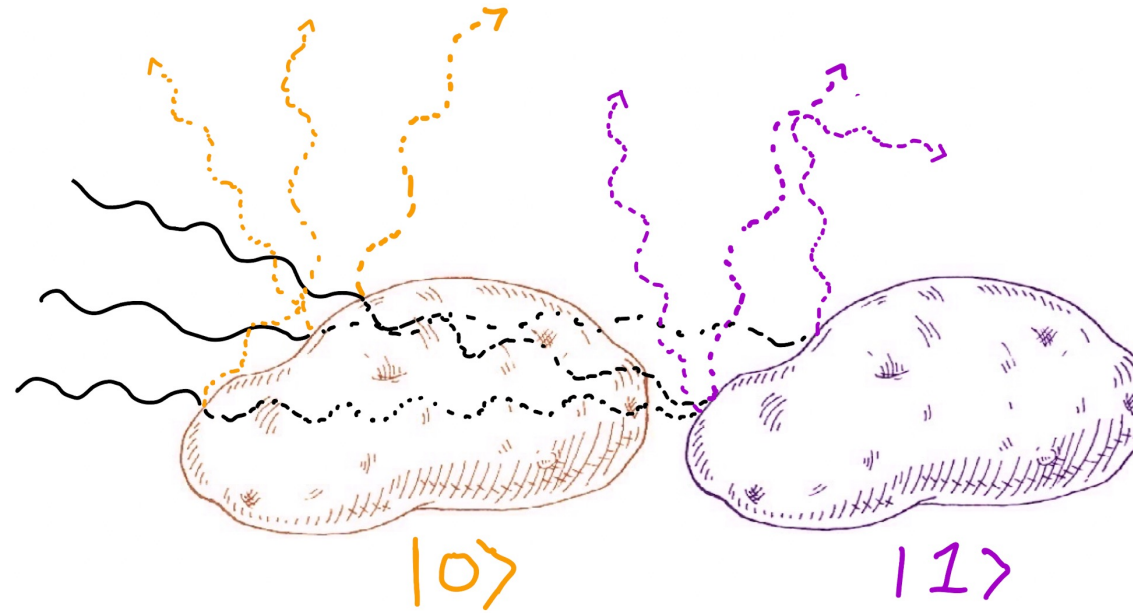


Entangled state!

- The photon and potato have lost their individuality
- The photon state is dependent on the potato state and vice versa

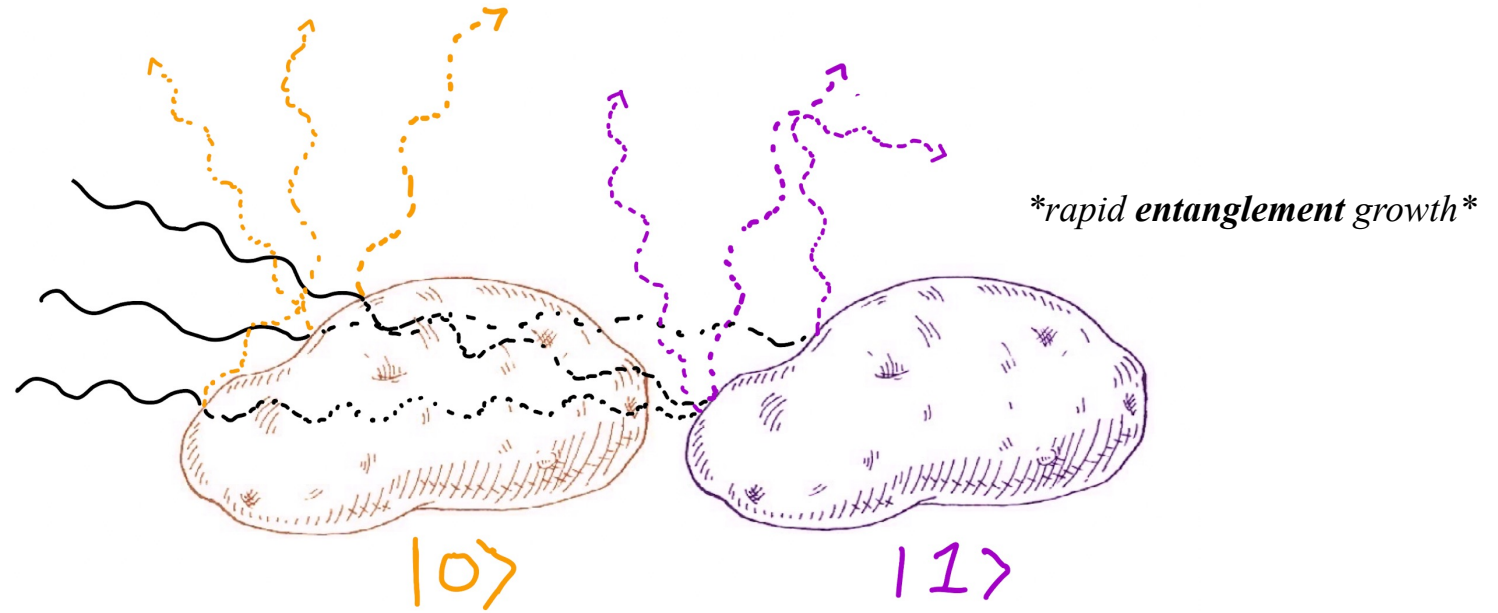
Decoherence to a Pointer State

Superposition state:



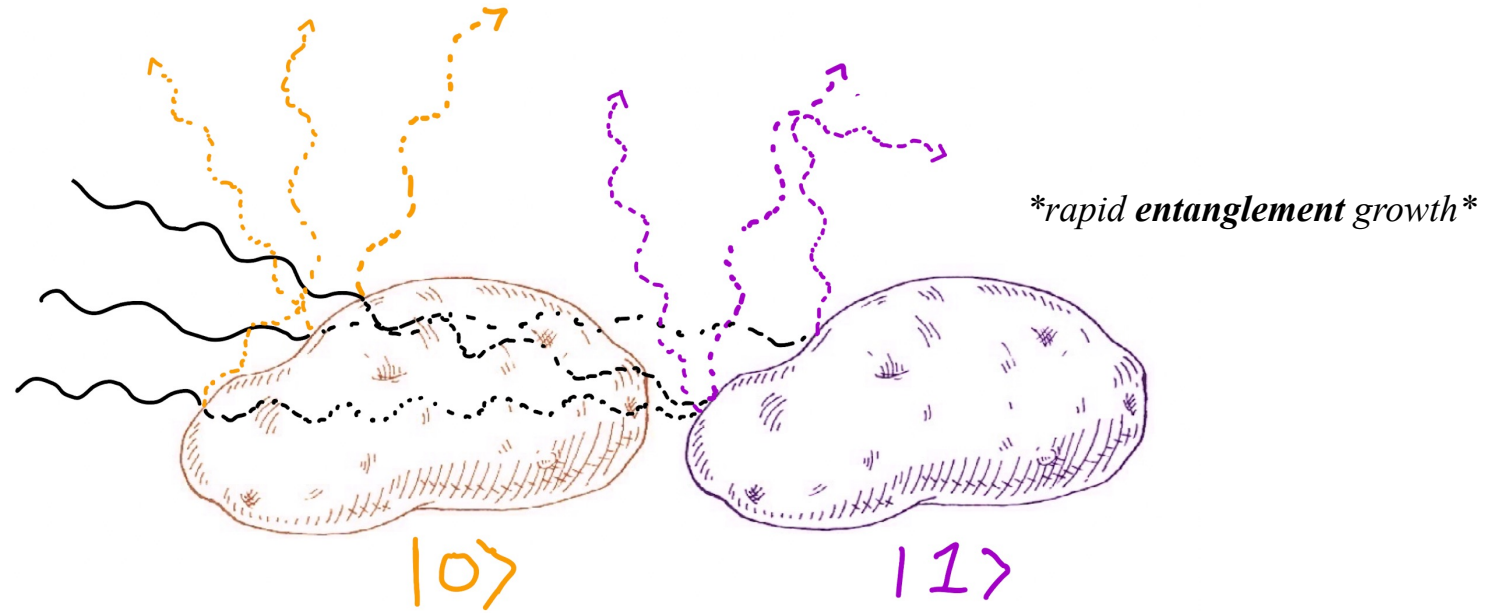
Decoherence to a Pointer State

Superposition state:



Decoherence to a Pointer State

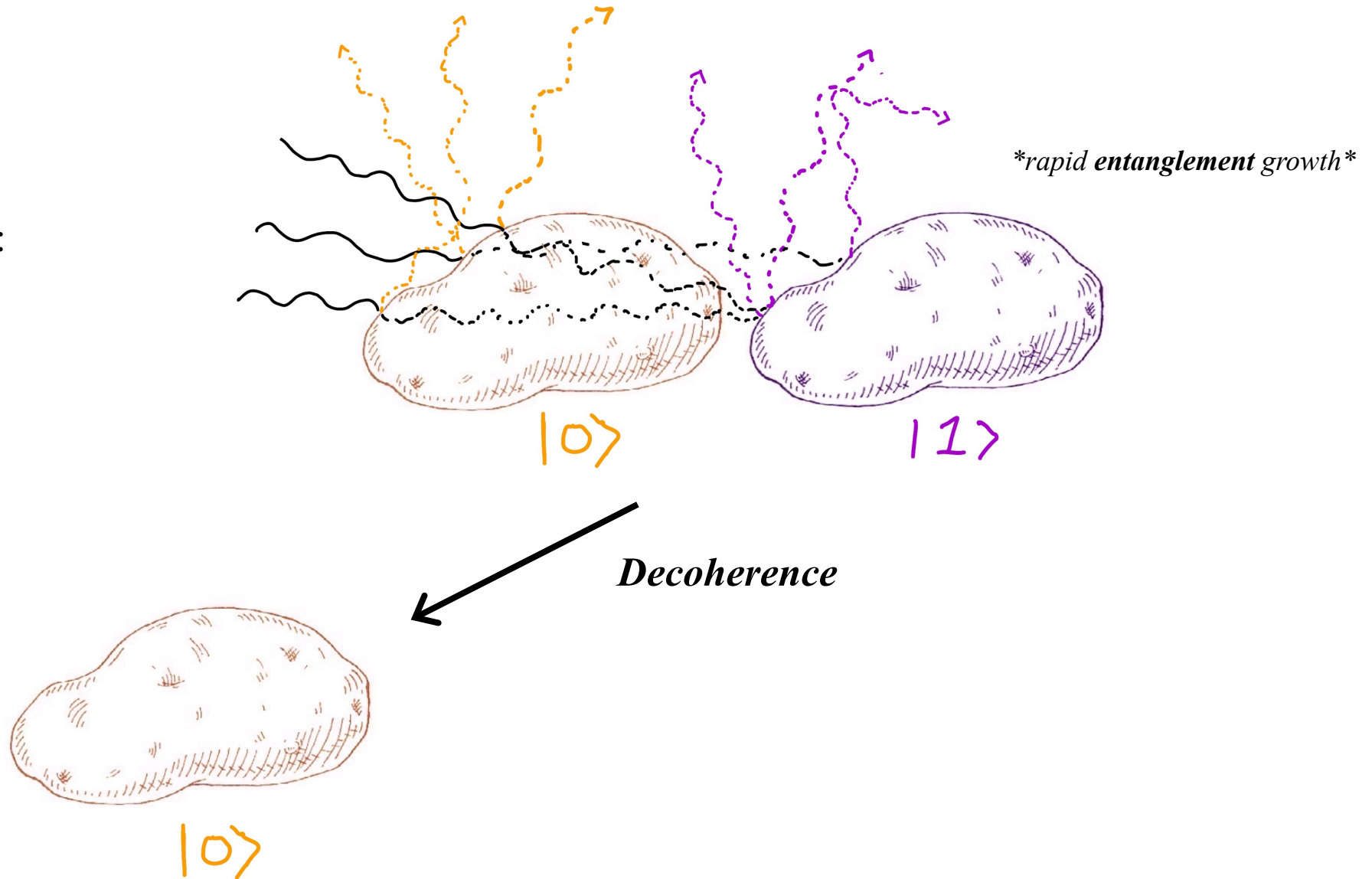
Superposition state:



Decoherence

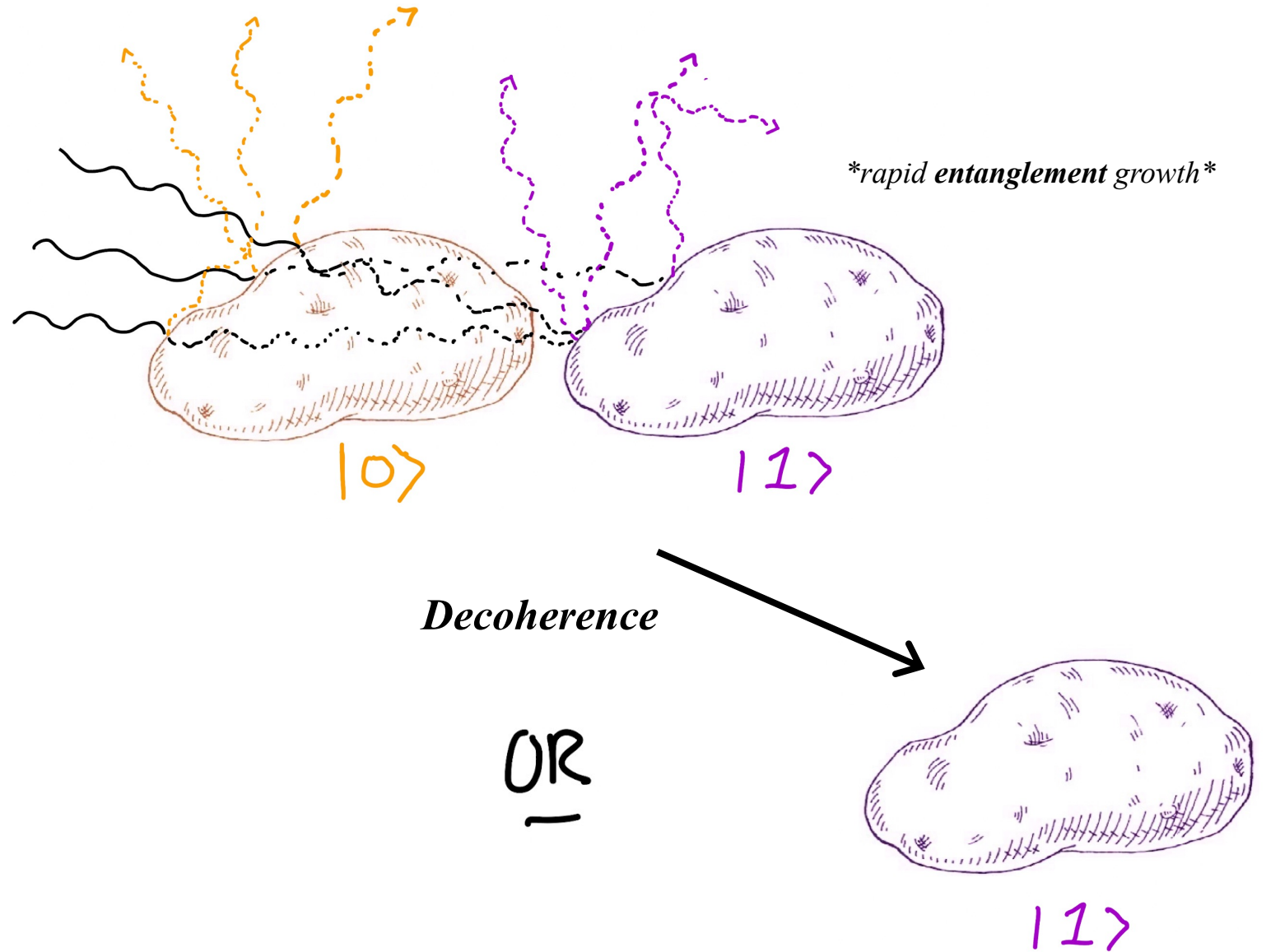
Decoherence to a Pointer State

Superposition state:



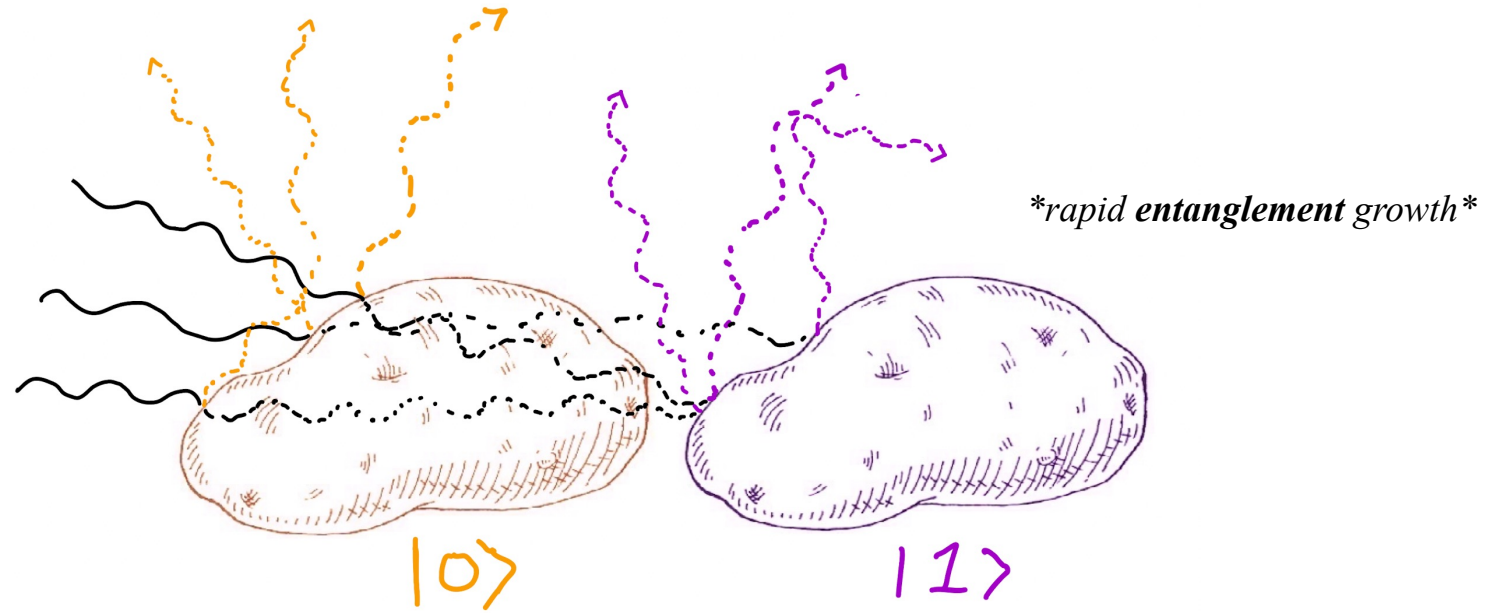
Decoherence to a Pointer State

Superposition state:

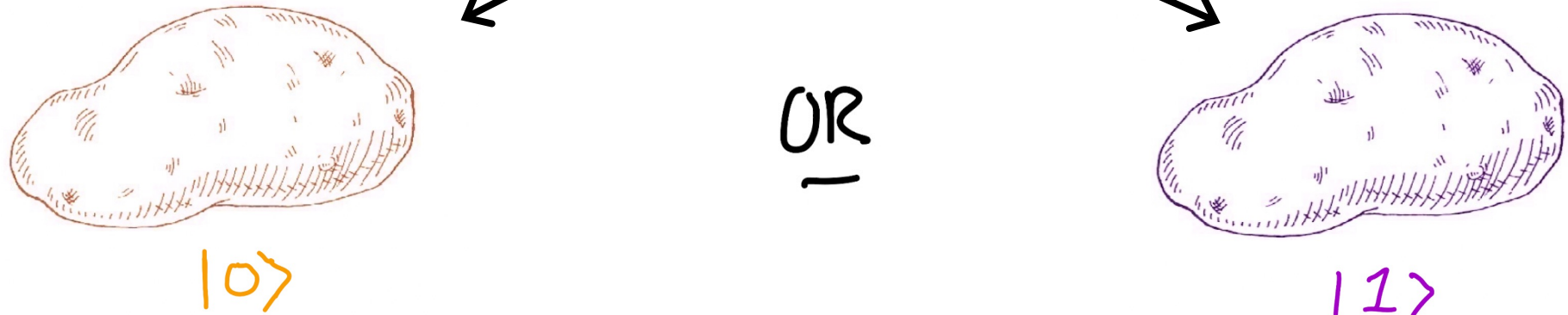


Decoherence to a Pointer State

Superposition state:



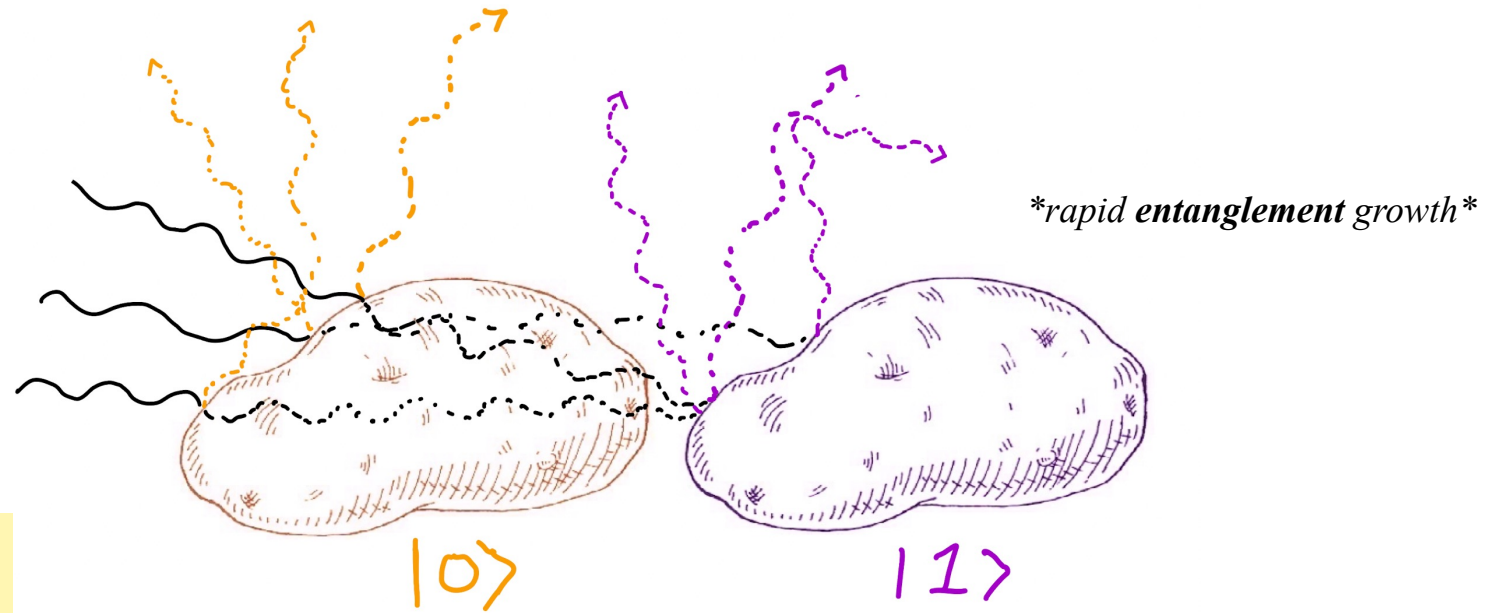
Pointer states:



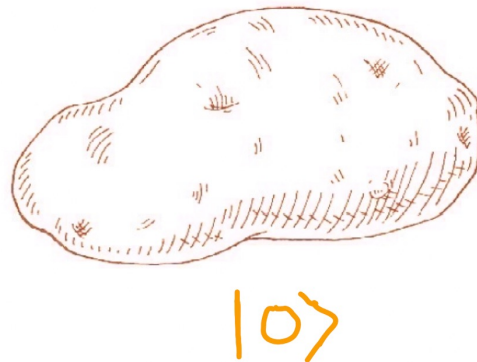
Decoherence to a Pointer State

Superposition state:

Decoherence is why potatoes
don't enter superpositions!



Pointer states:



OR

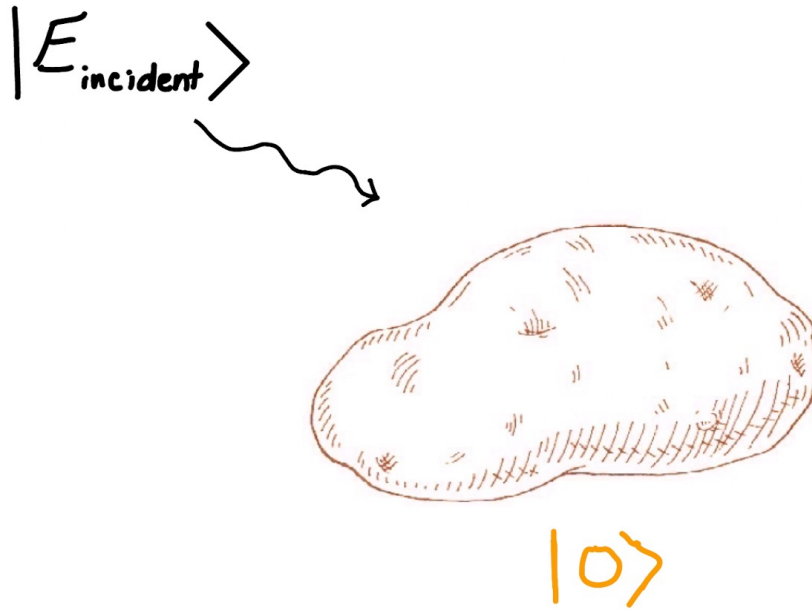


Potato Initialized in a Pointer State

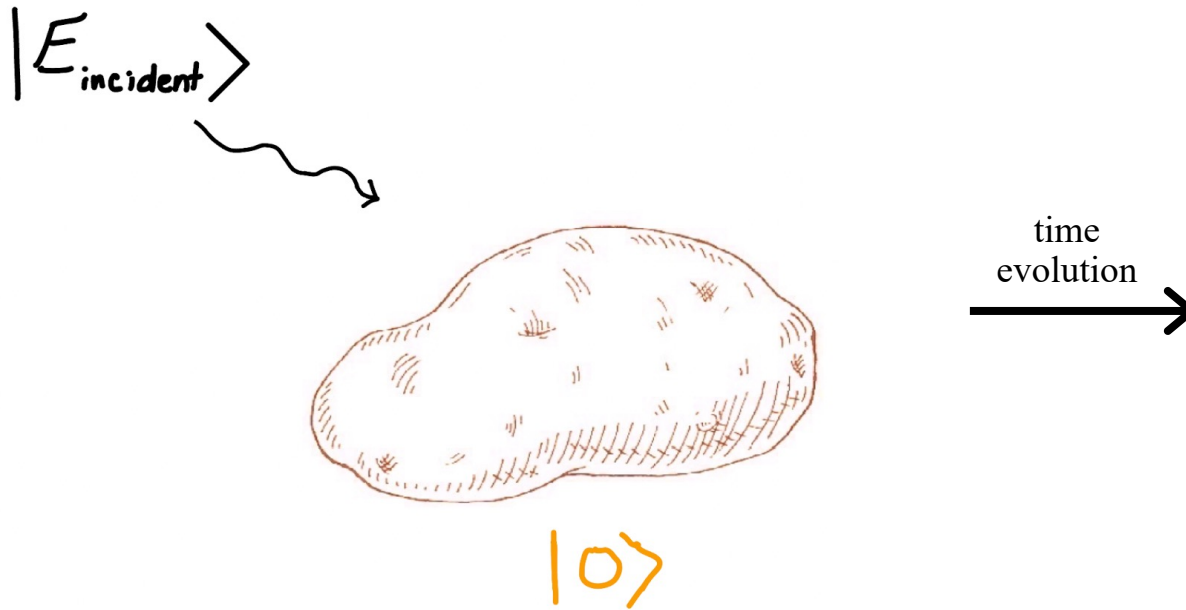


$|0\rangle$

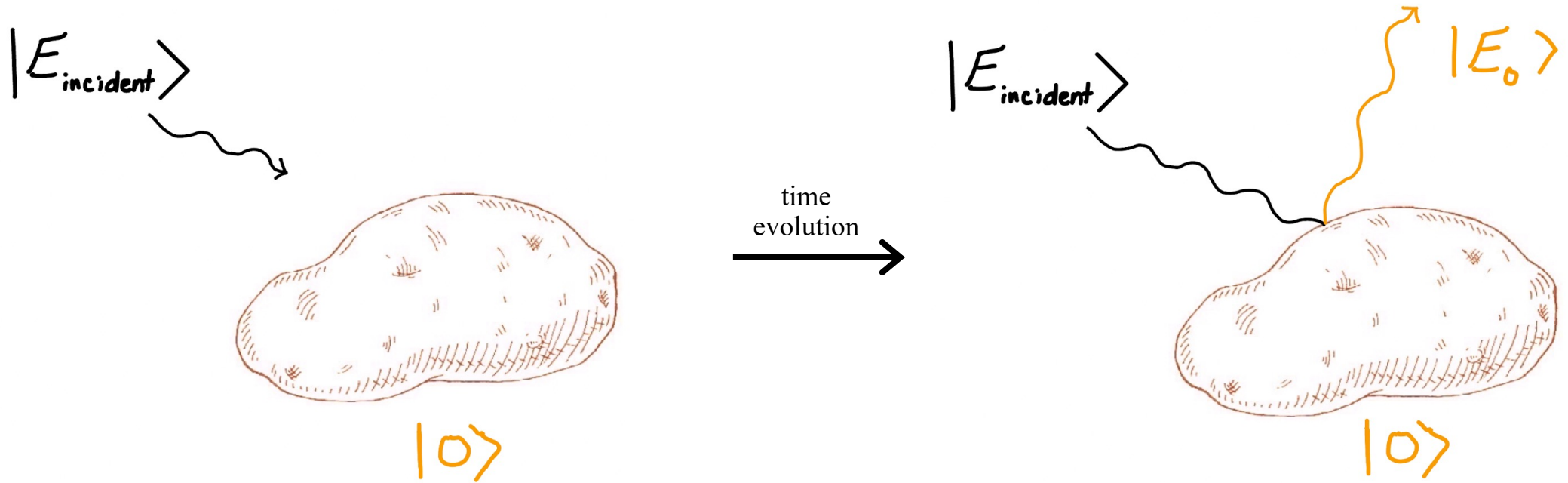
Potato Initialized in a Pointer State



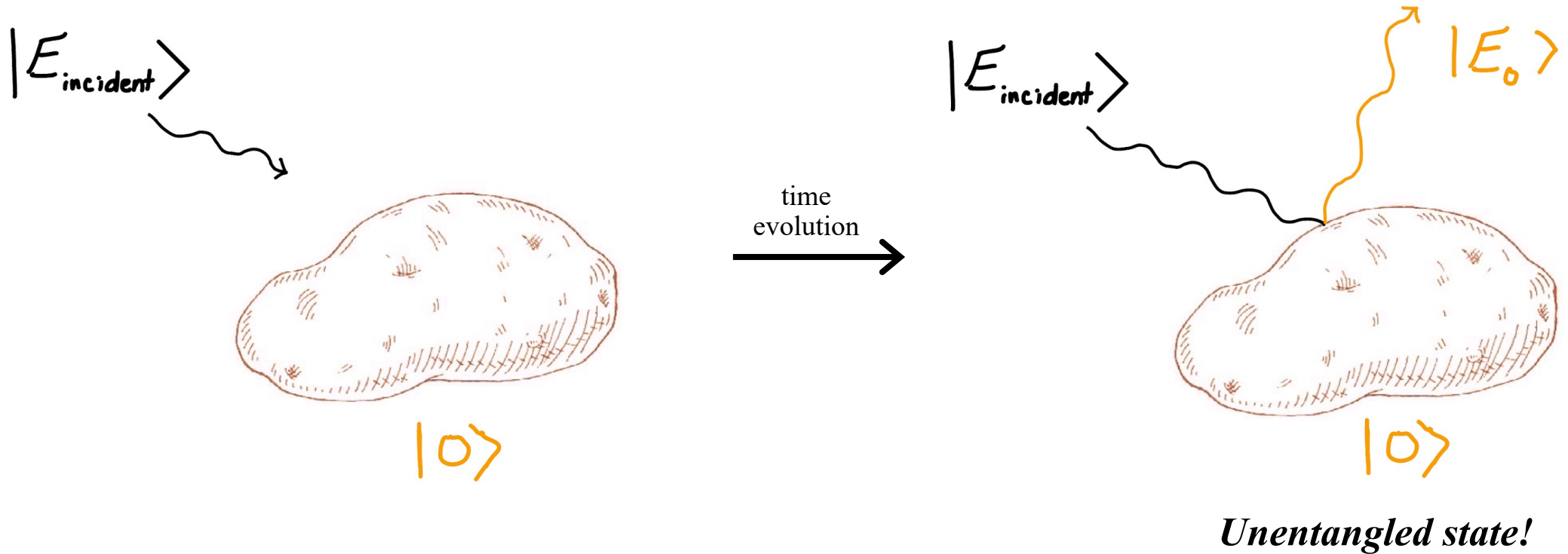
Potato Initialized in a Pointer State



Potato Initialized in a Pointer State

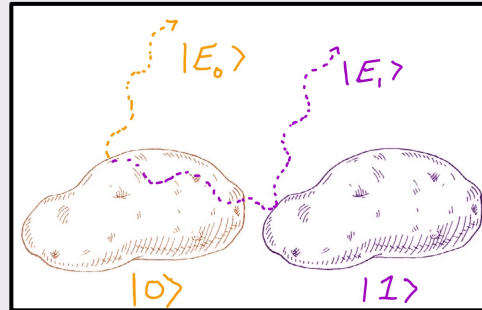


Potato Initialized in a Pointer State



Focus of Thesis:

The division of a composite quantum system into subsystems.

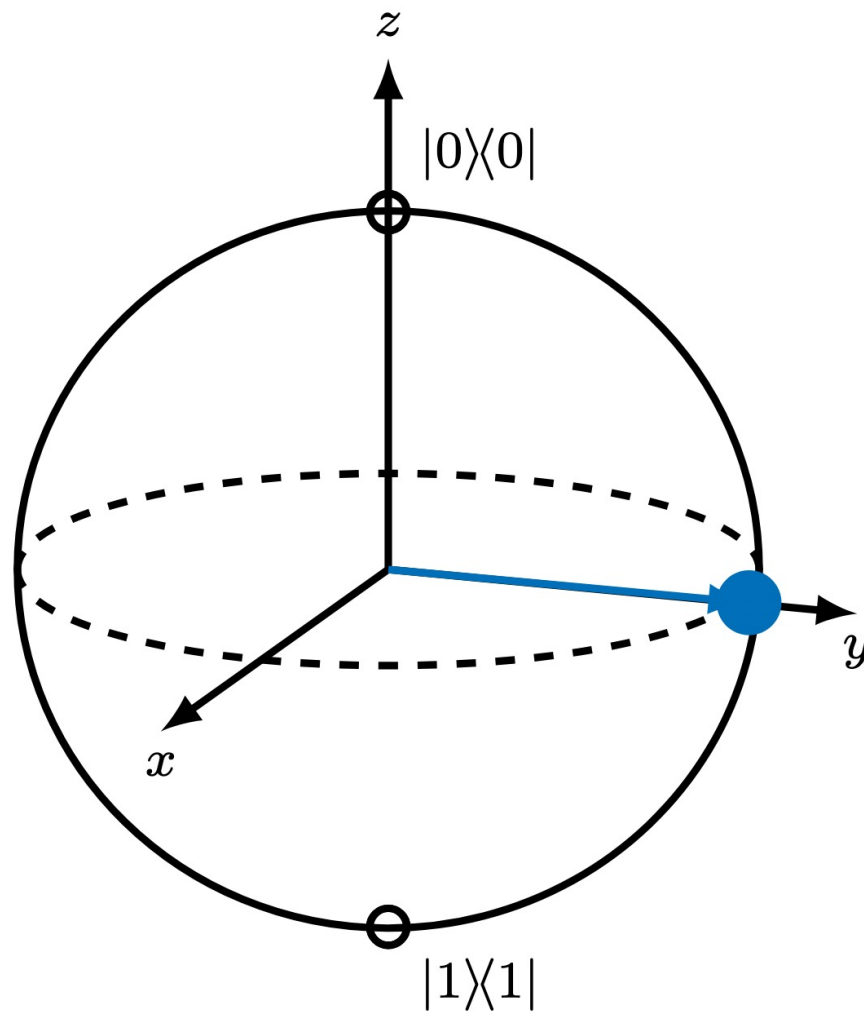


Sometimes our subsystems are not as clear as “potato” and “photon”

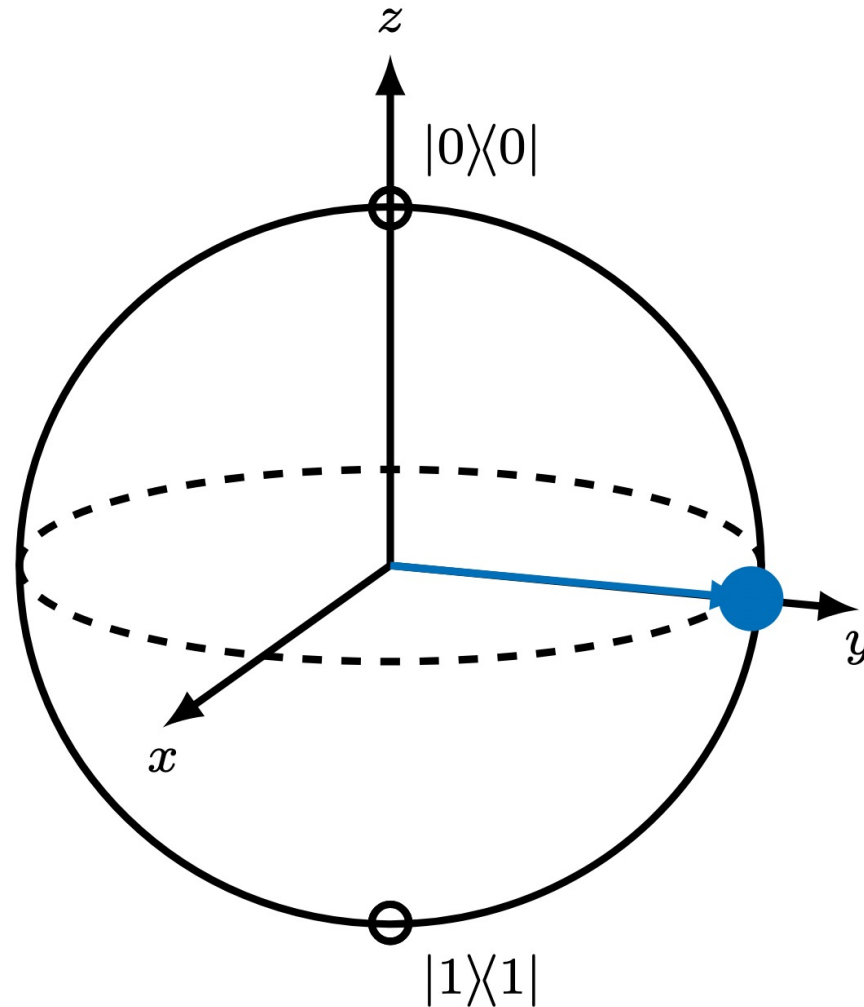
Question:

How do we determine these subsystems given an abstract quantum system?

The Bloch Ball: A Visualization Tool



The Bloch Ball: A Visualization Tool

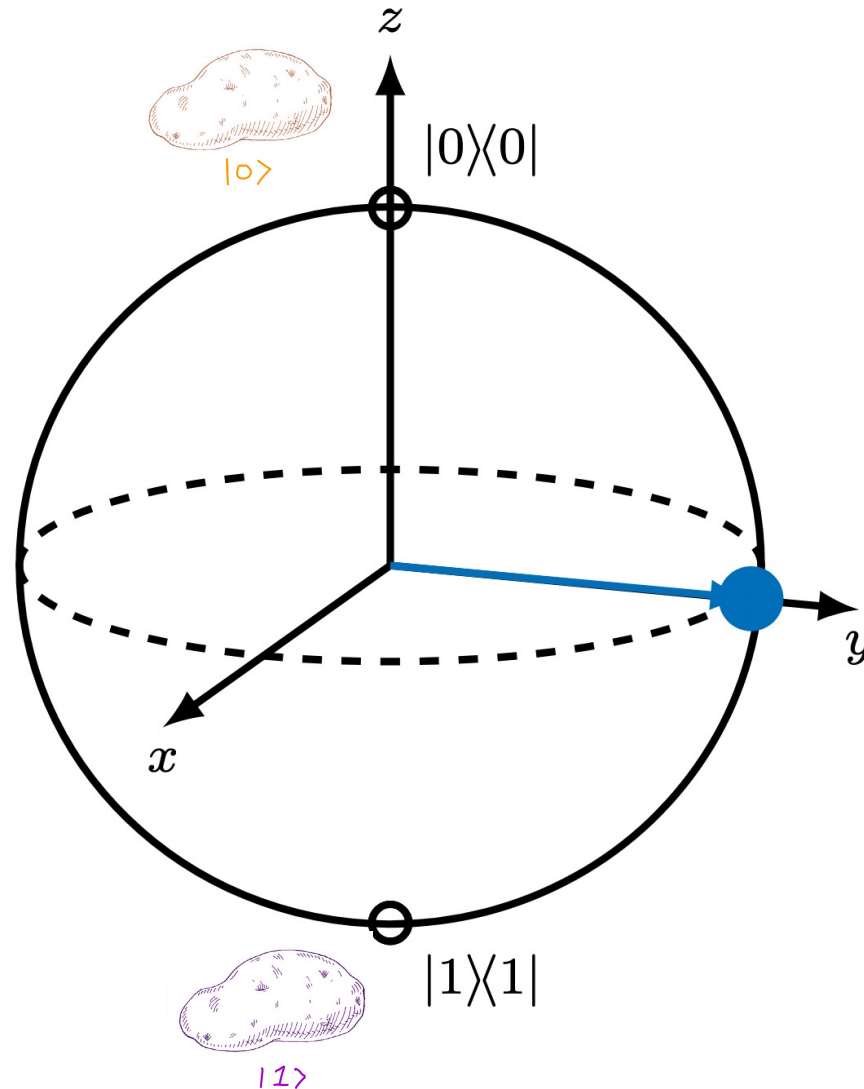


Bloch Ball with blue state
represents our “system”
(e.g., potato)

The Bloch Ball: A Visualization Tool

KEY:

Poles = basis states

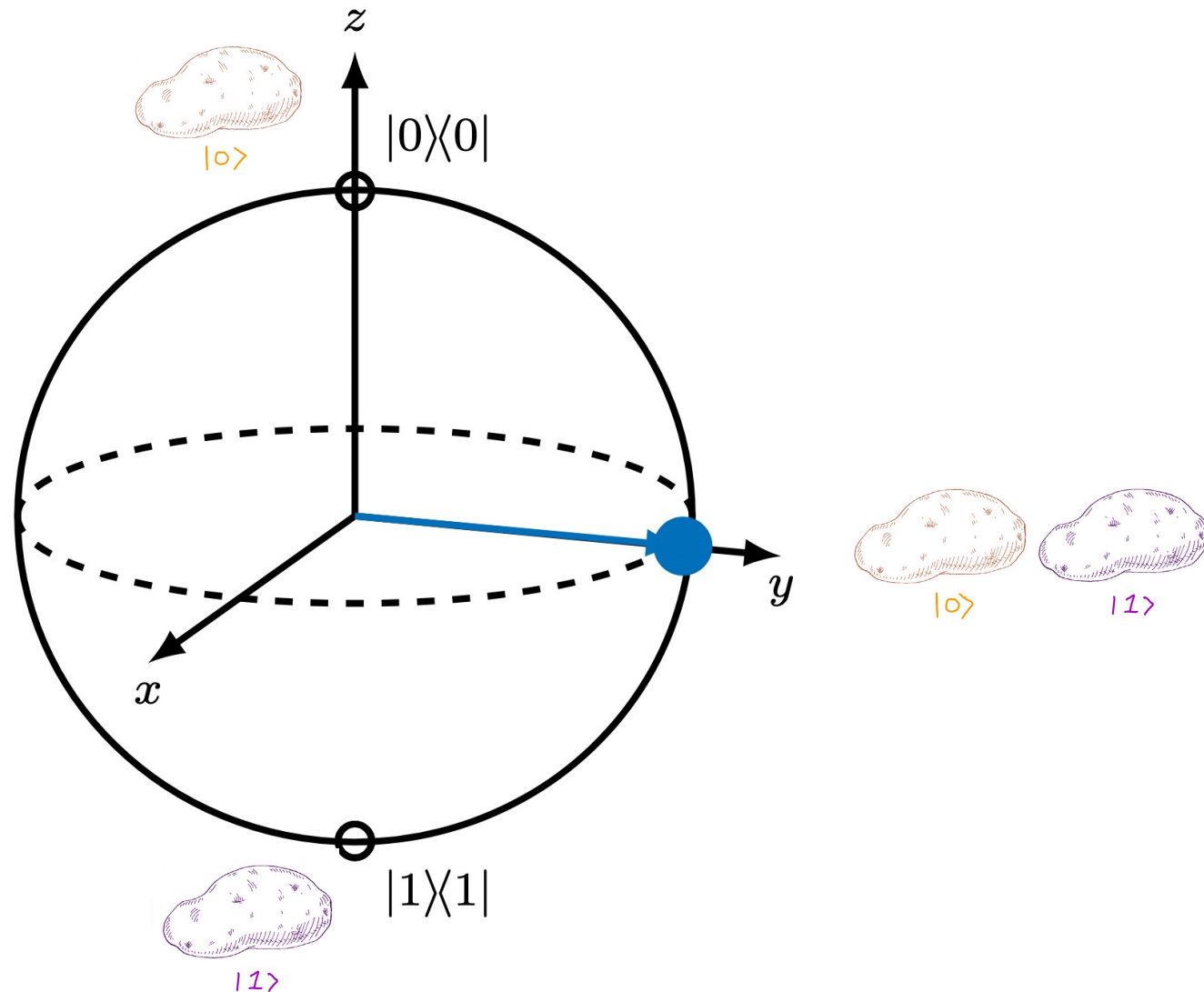


The Bloch Ball: A Visualization Tool

KEY:

Poles = basis states

Surface = unentangled state



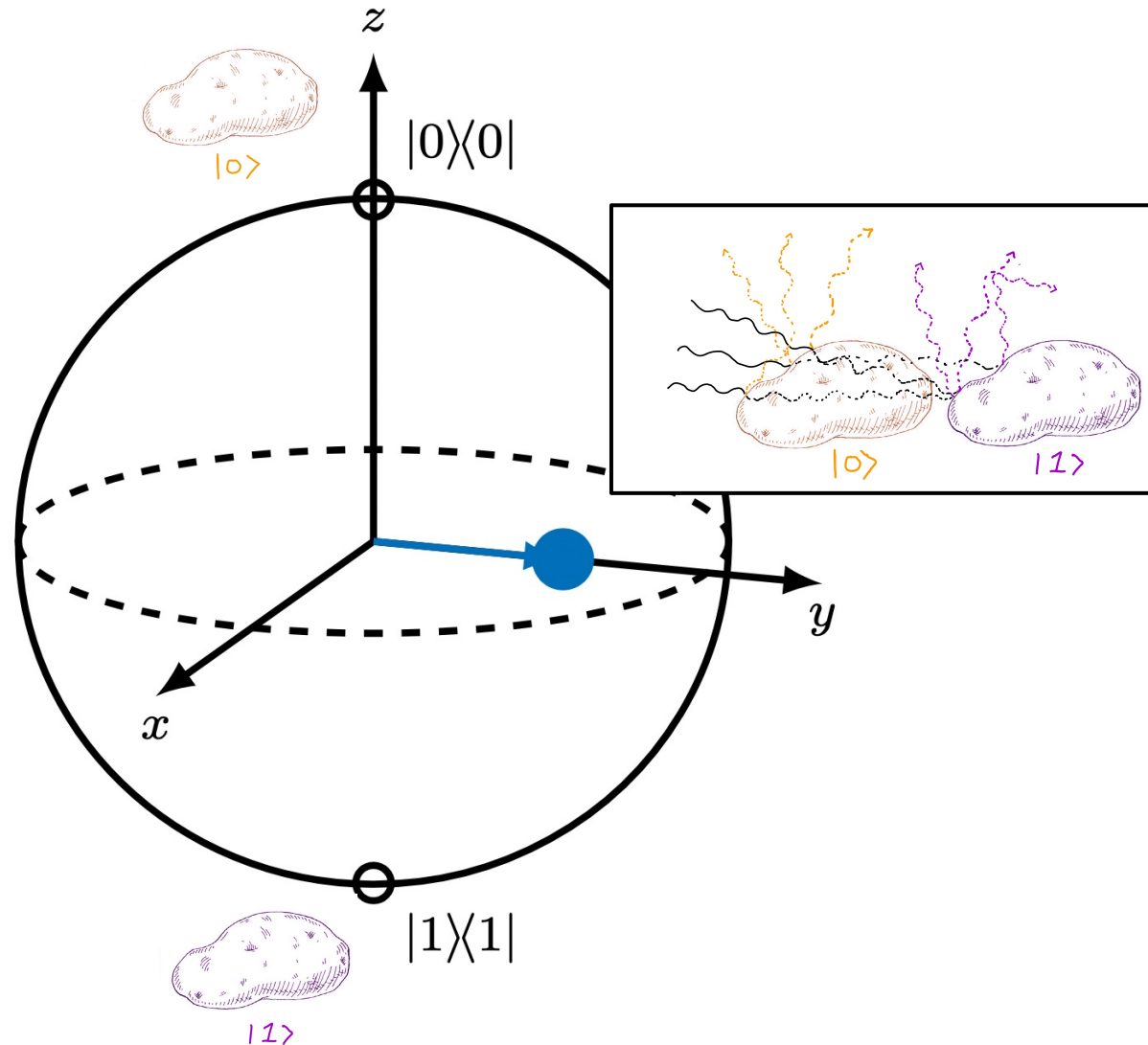
The Bloch Ball: A Visualization Tool

KEY:

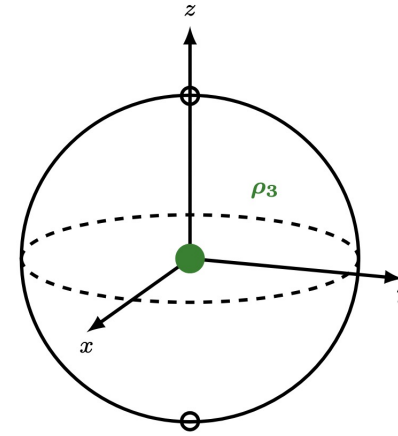
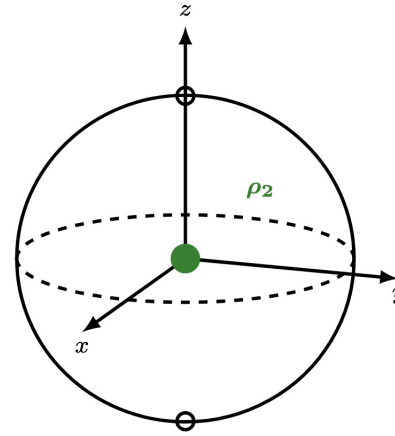
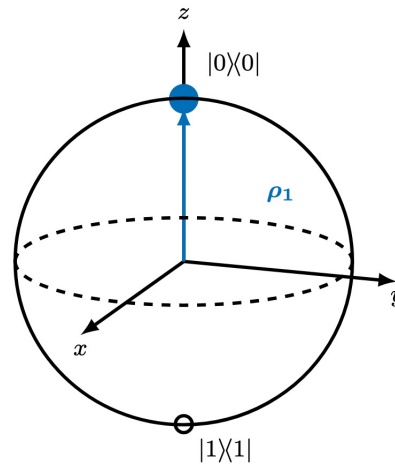
Poles = basis states

Surface = unentangled state

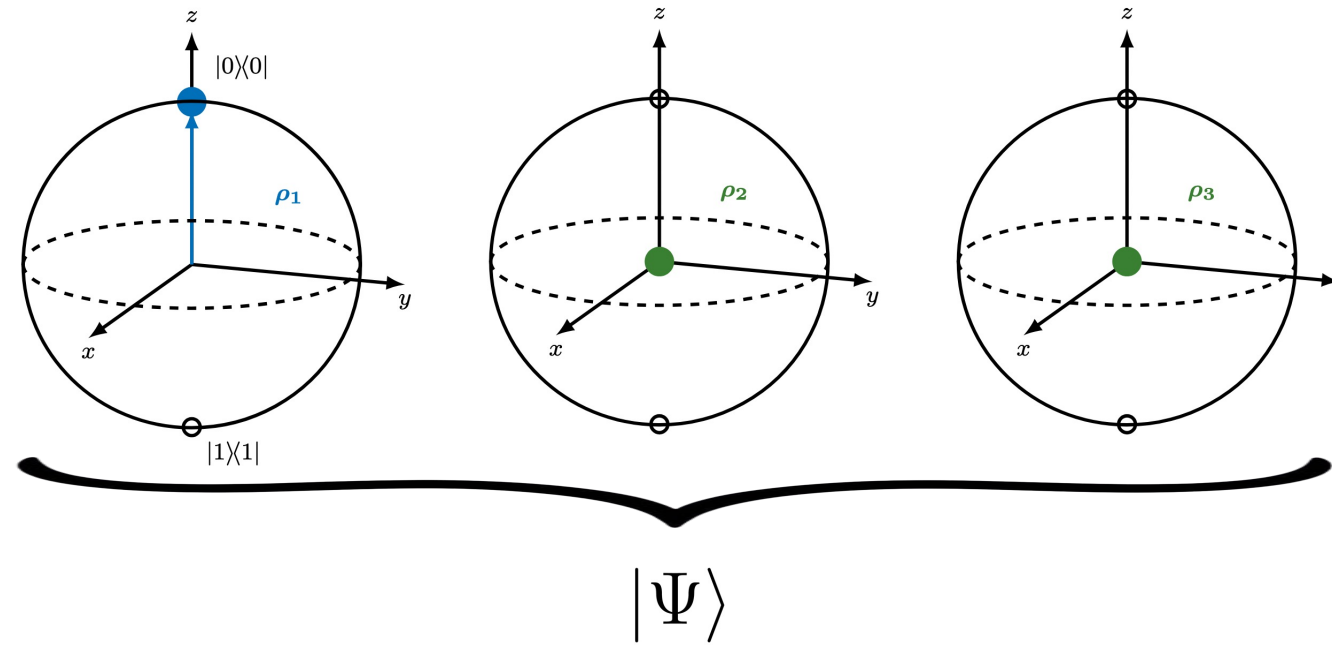
Interior = entangled state



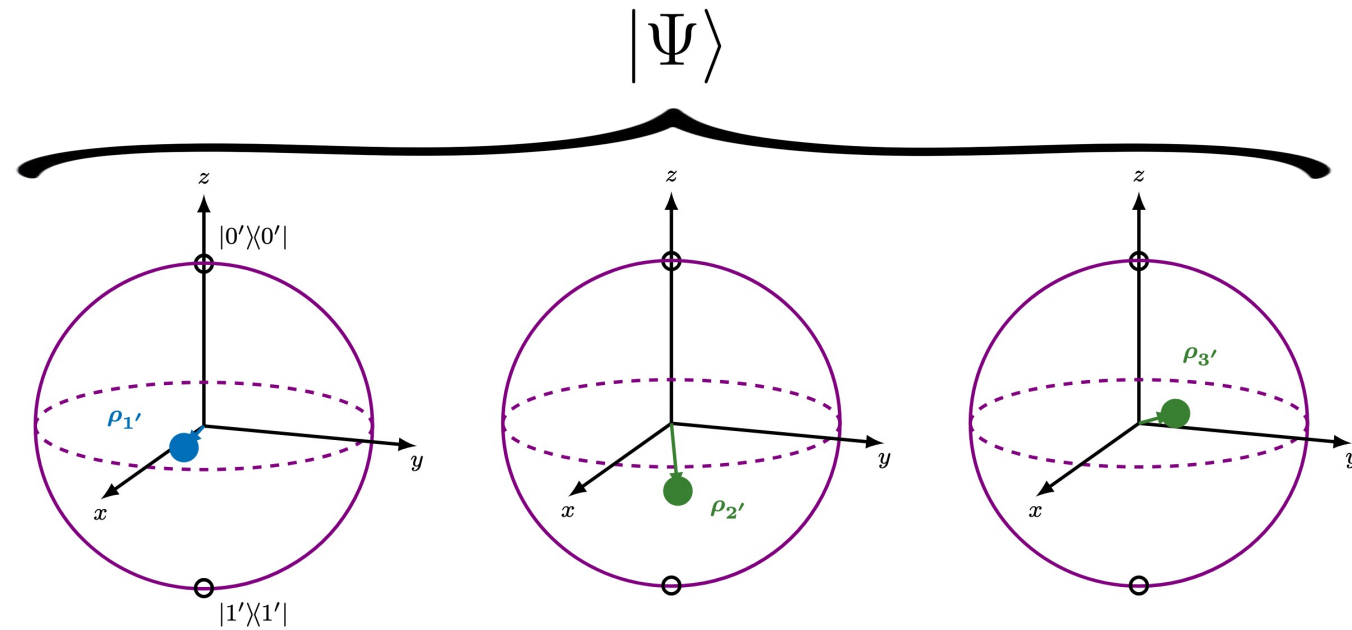
Tensor Product Structures



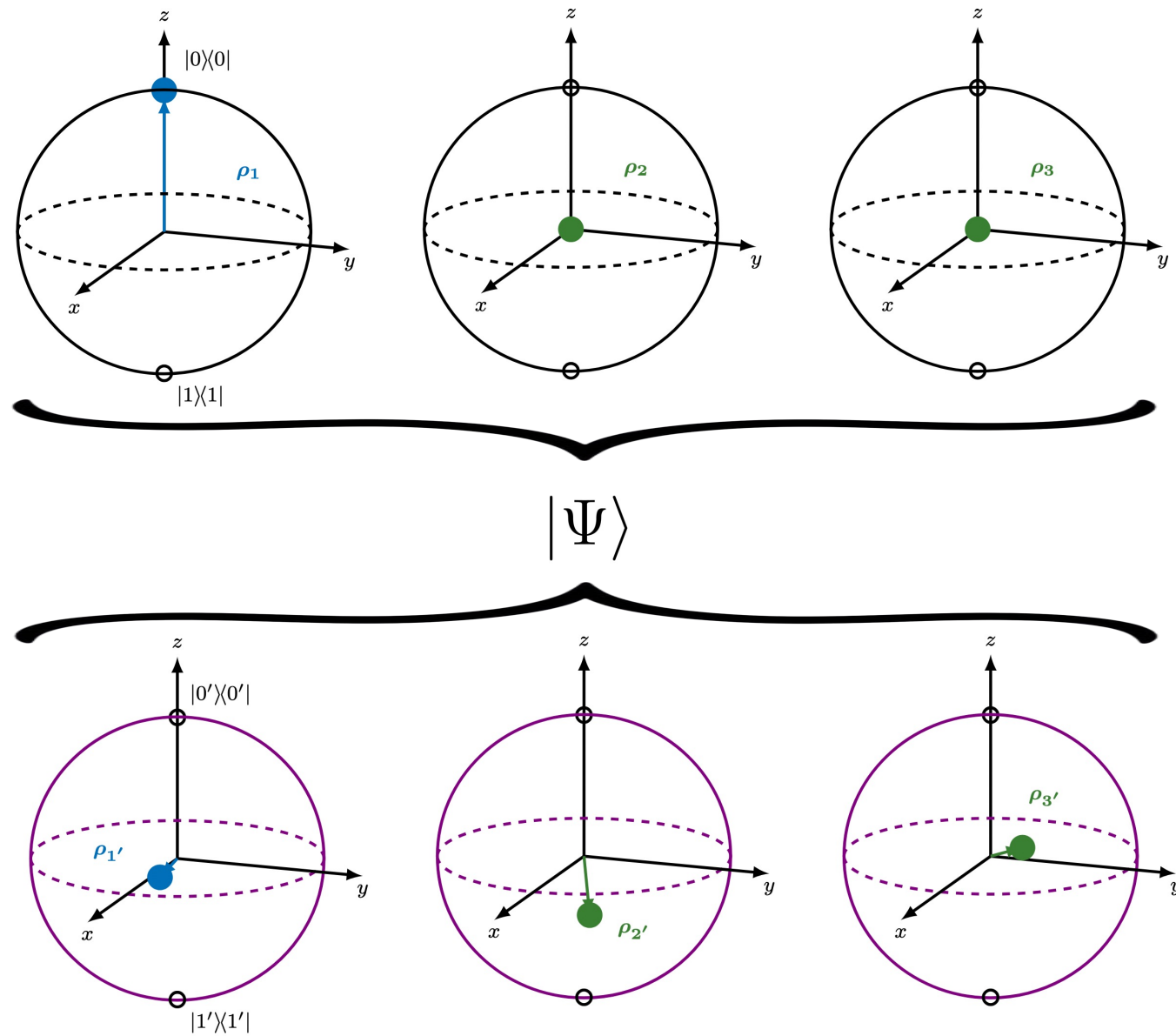
Tensor Product Structures



Tensor Product Structures

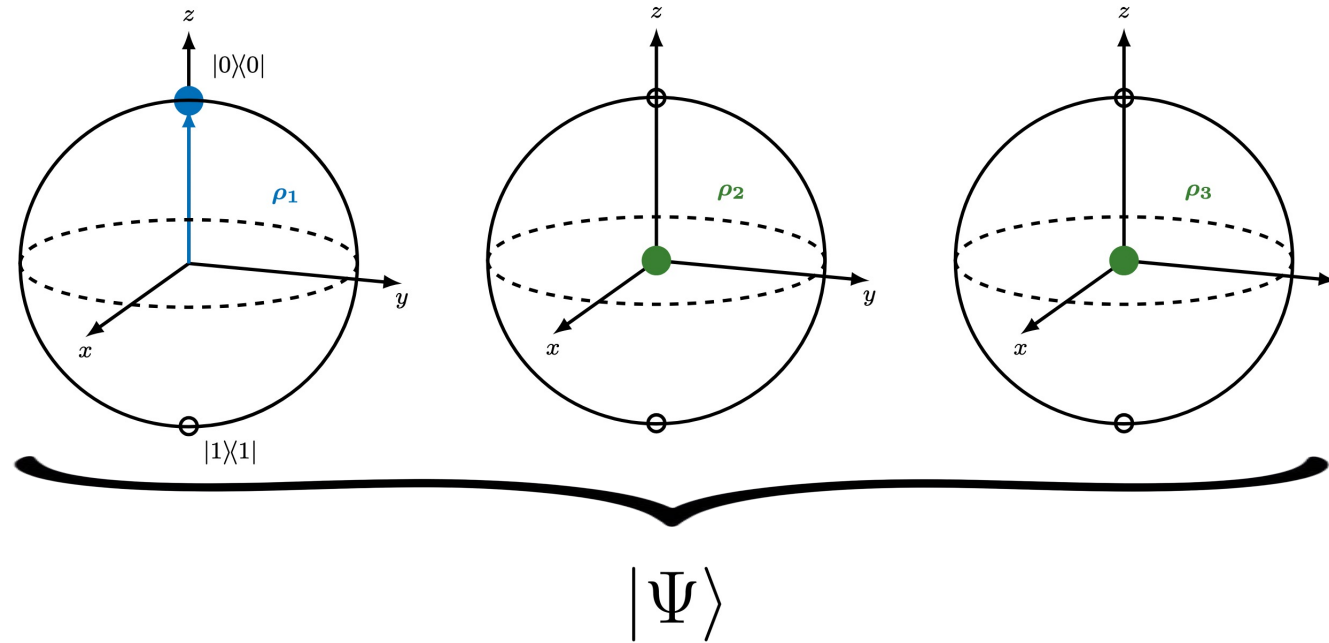


Tensor Product Structures

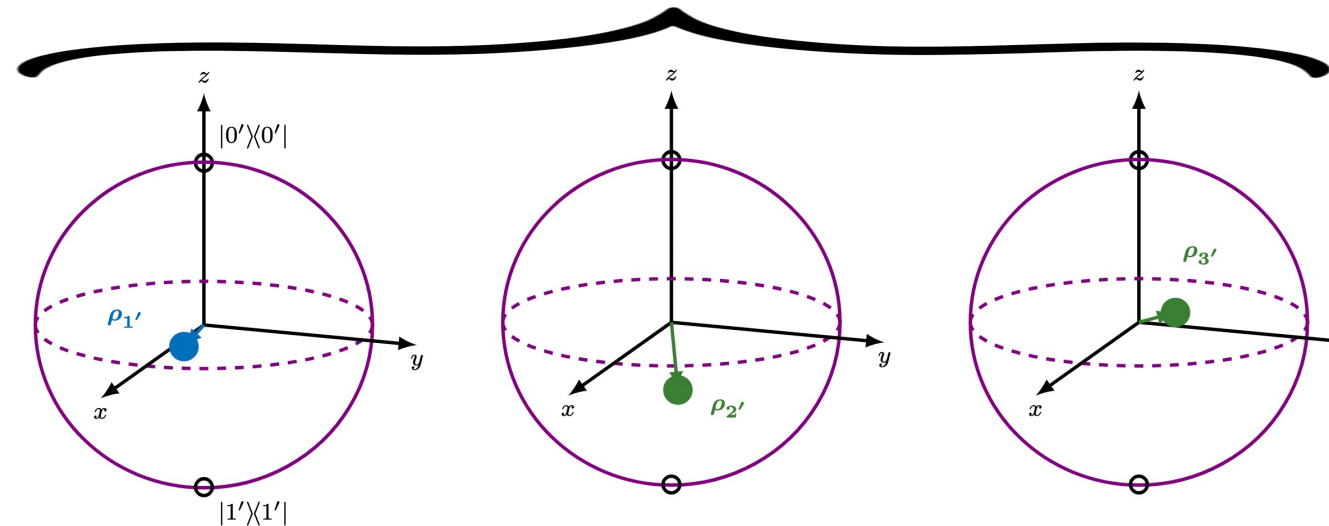


Tensor Product Structures

**Tensor Product
Structure 1
(Qubits 1, 2, 3):**

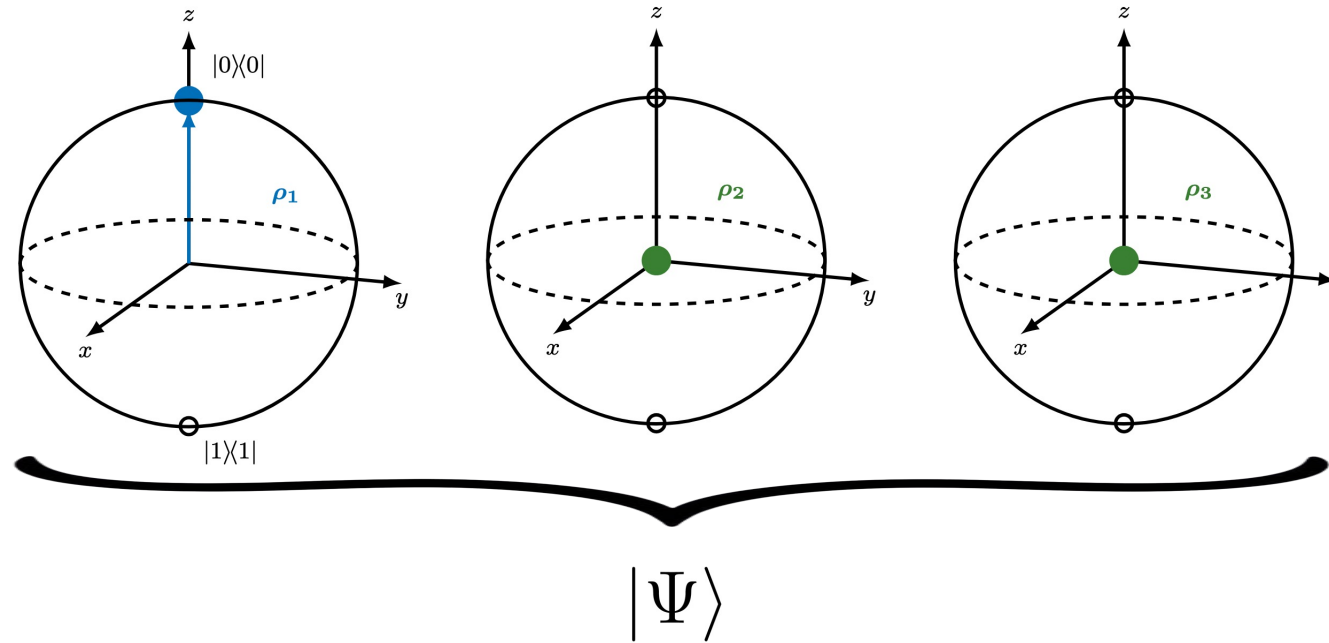


**Tensor Product
Structure 2
(Qubits 1', 2', 3'):**



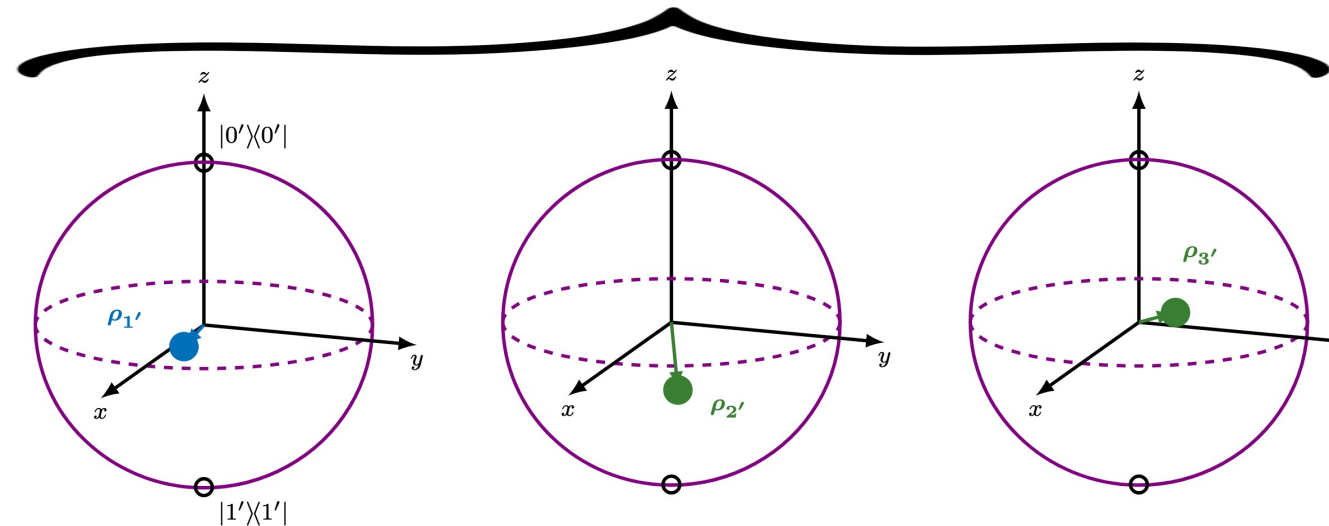
Tensor Product Structures

**Tensor Product
Structure 1
(Qubits 1, 2, 3):**



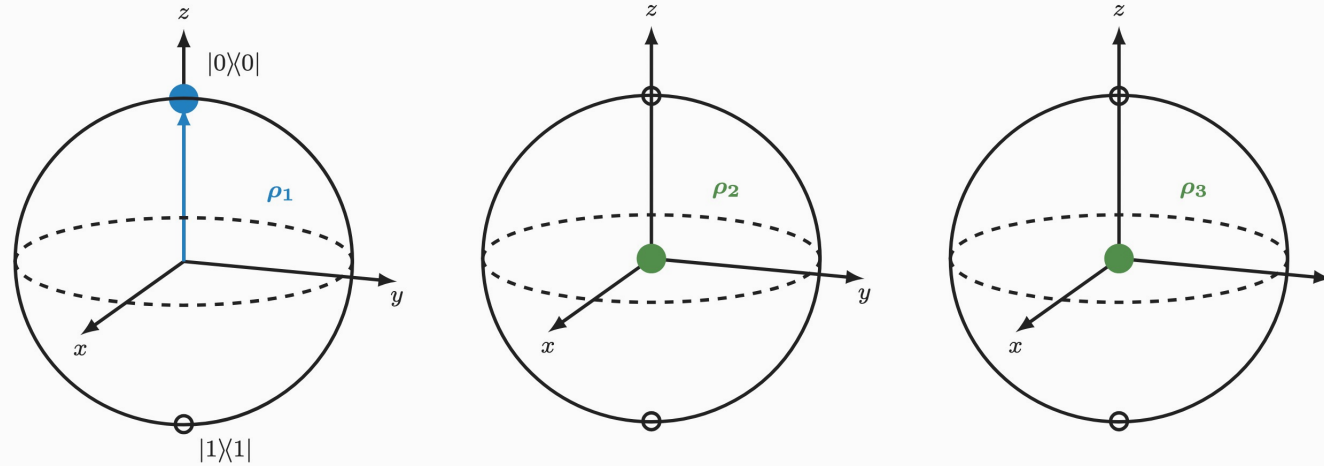
**Entanglement is a
Relative Concept!**

**Tensor Product
Structure 2
(Qubits 1', 2', 3'):**



Tensor Product Structures

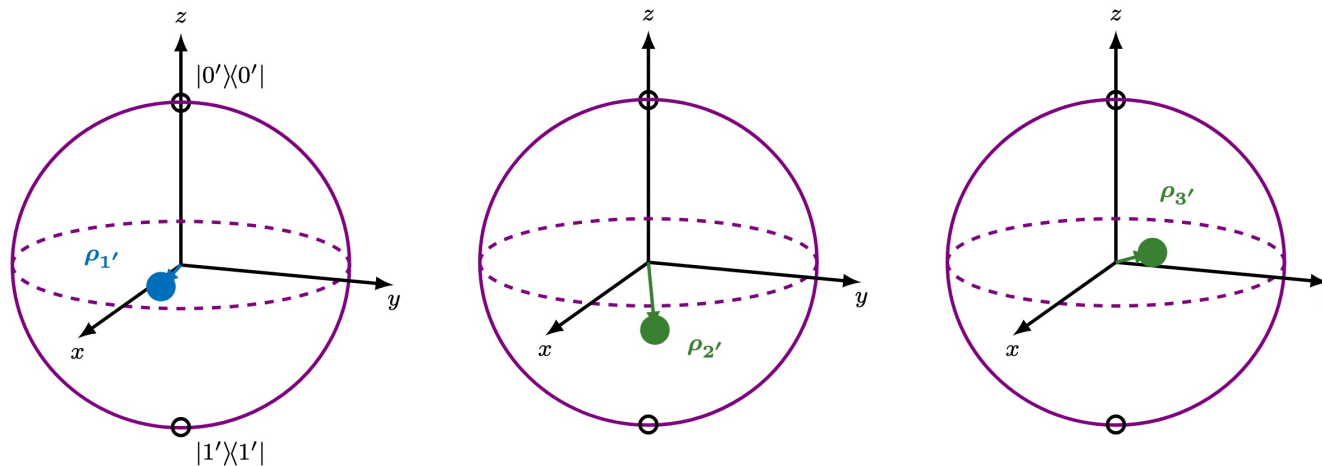
**Tensor Product
Structure 1
(Qubits 1, 2, 3):**



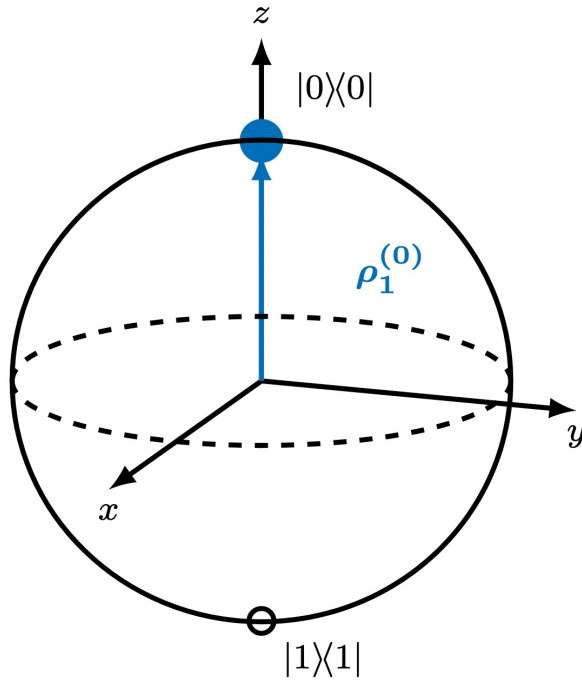
**Entanglement is a
Relative Concept!**

$|\Psi\rangle$

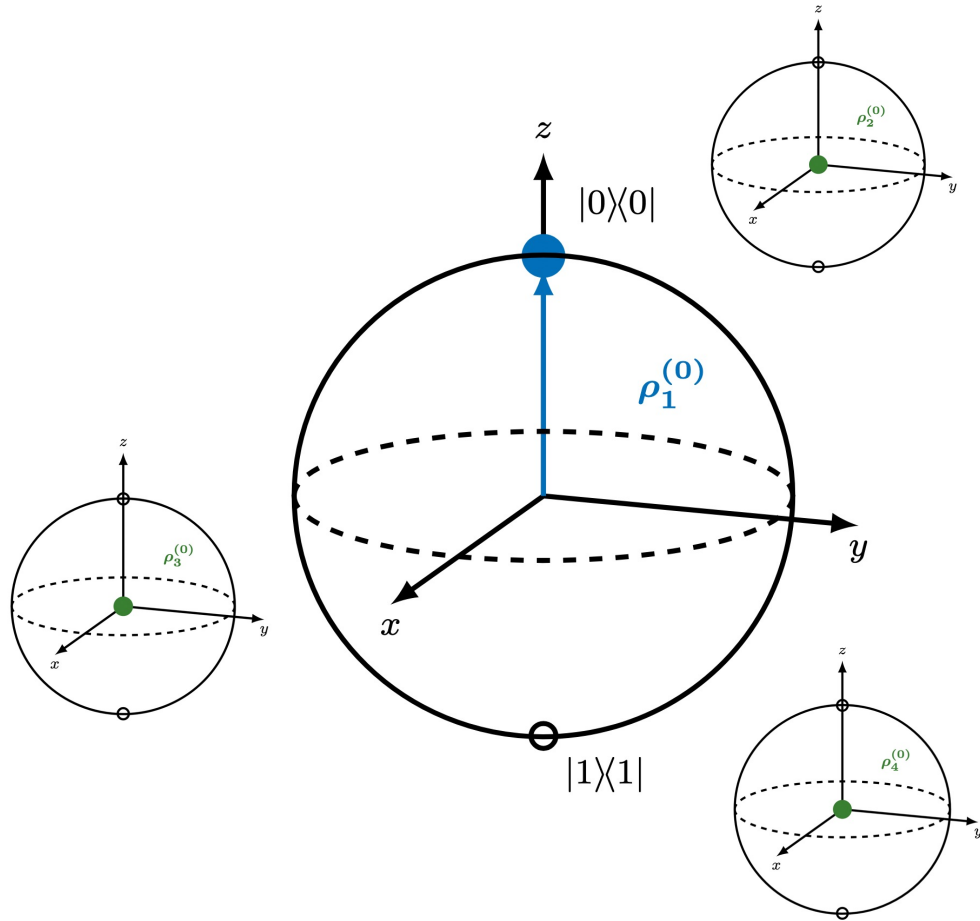
**Tensor Product
Structure 2
(Qubits 1', 2', 3'):**



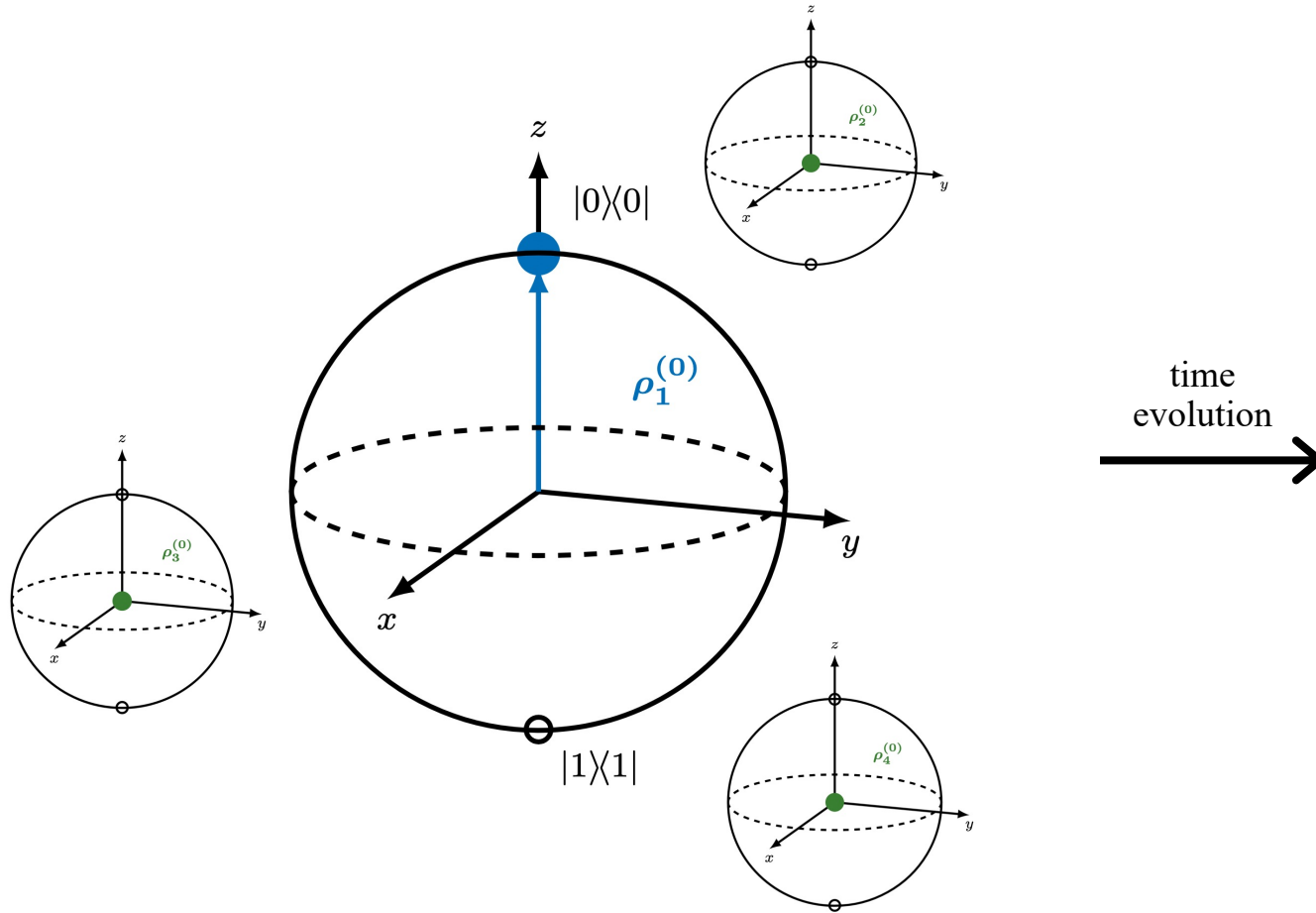
Quasi-classical Tensor Product Structure (qcTPS)



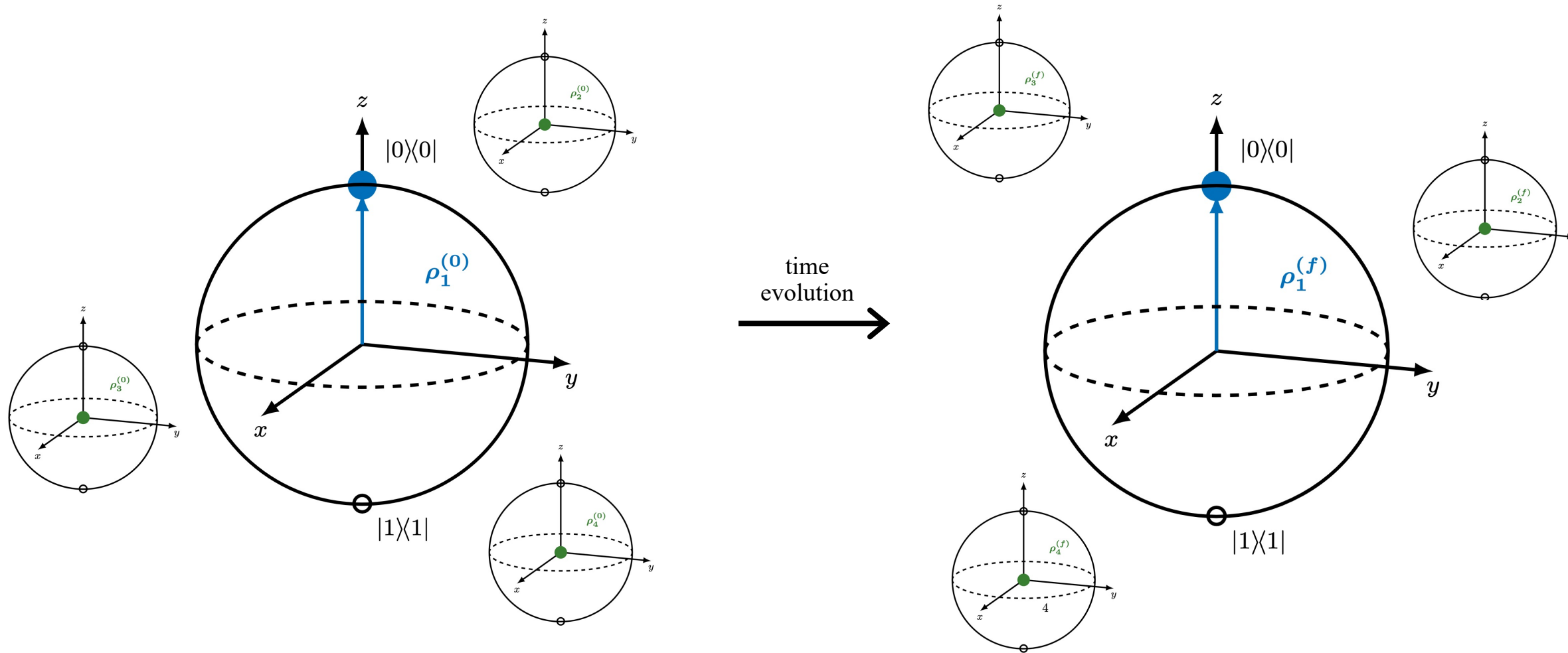
Quasi-classical Tensor Product Structure (qcTPS)



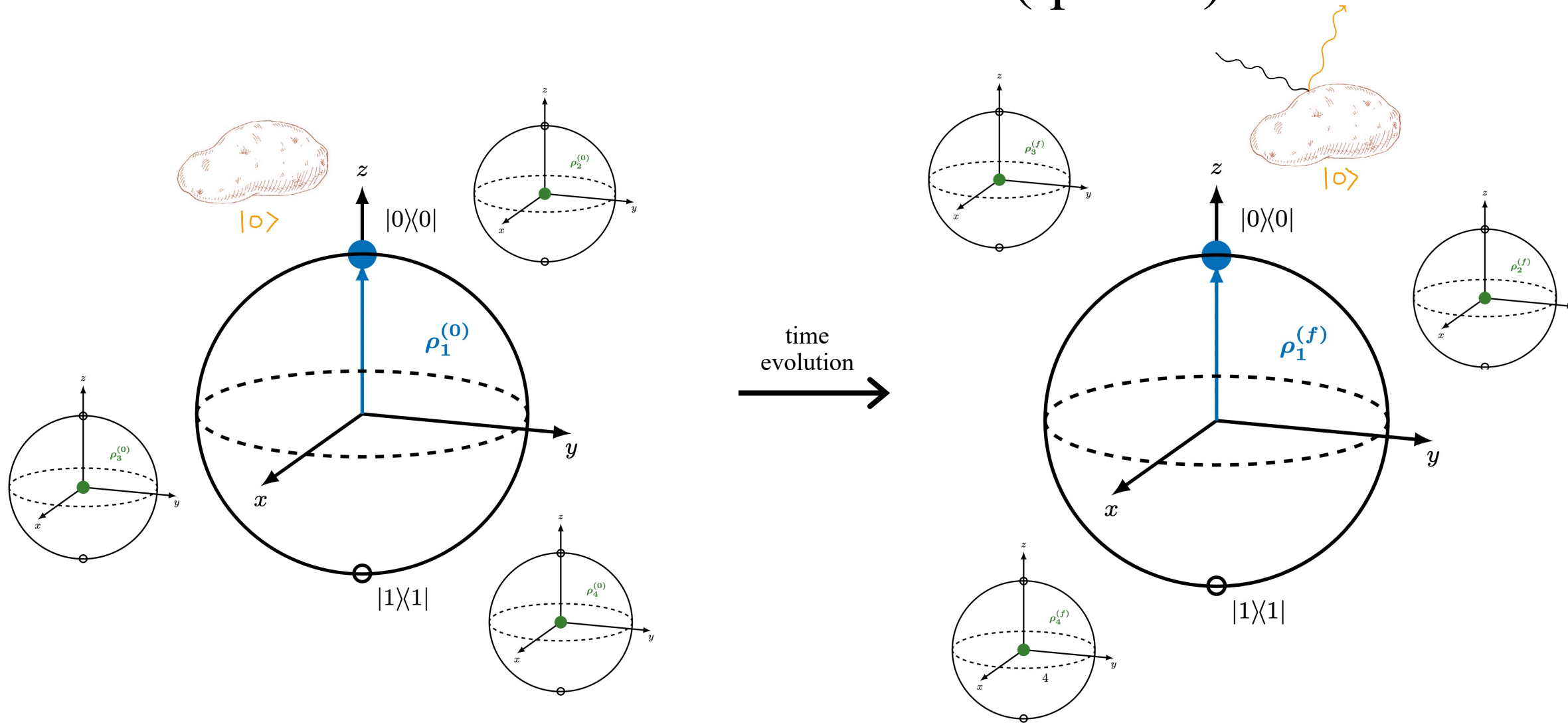
Quasi-classical Tensor Product Structure (qcTPS)



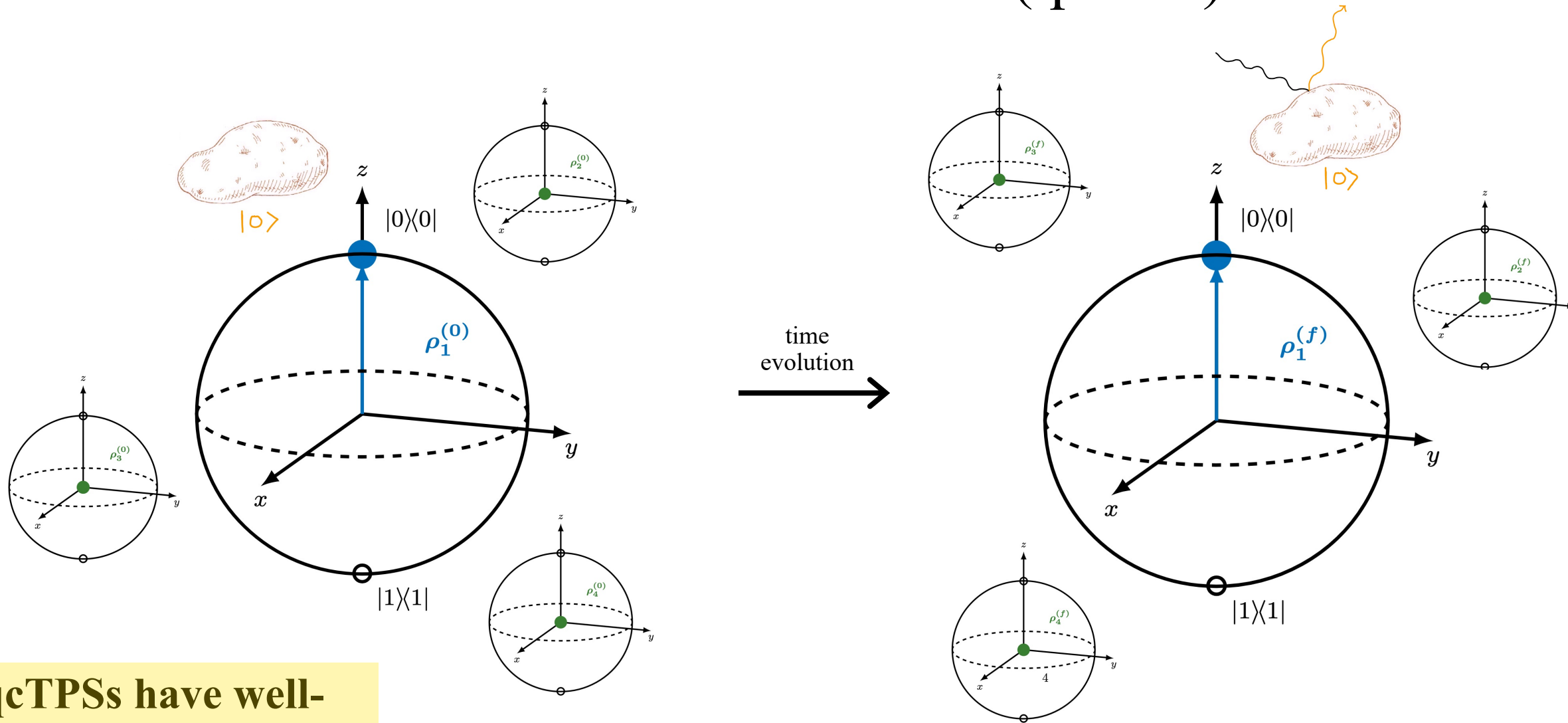
Quasi-classical Tensor Product Structure (qcTPS)



Quasi-classical Tensor Product Structure (qcTPS)



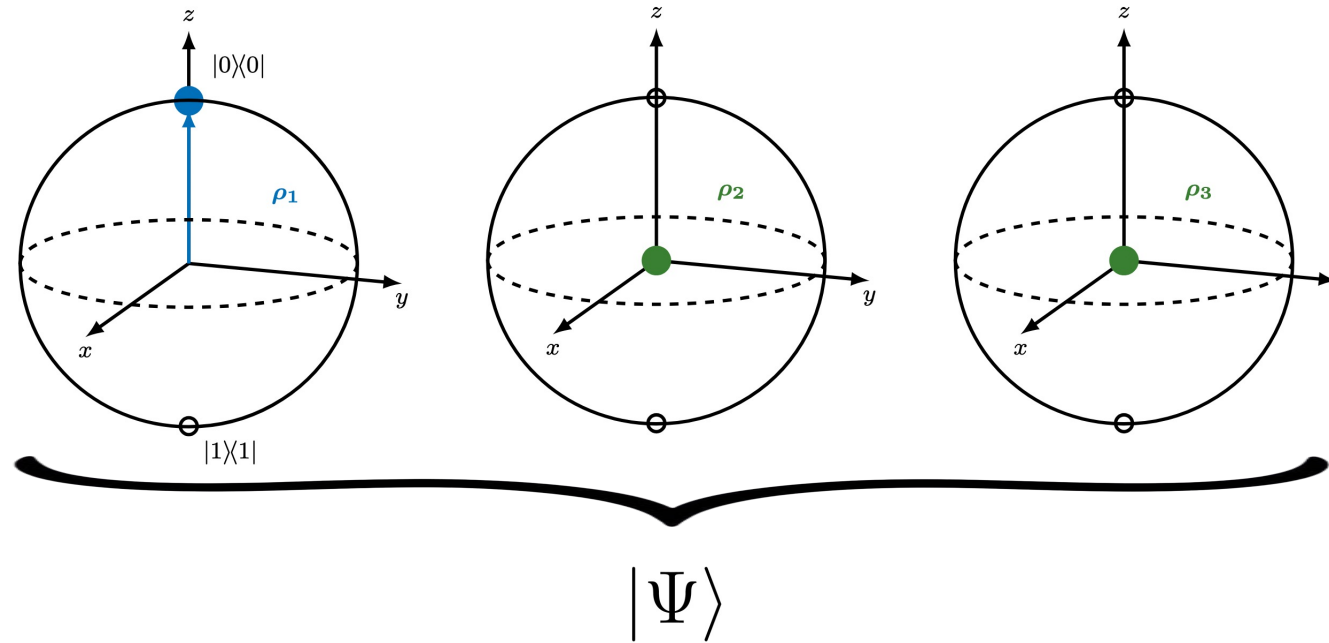
Quasi-classical Tensor Product Structure (qcTPS)



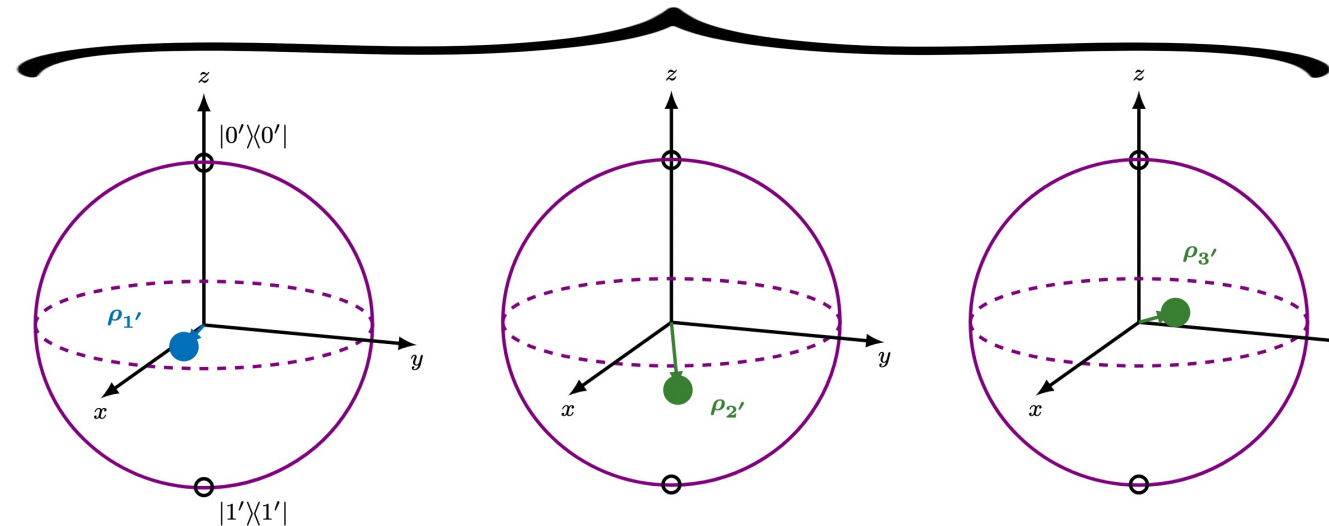
qcTPSs have well-defined pointer states!

Tensor Product Structures

**Tensor Product
Structure 1
(Qubits 1, 2, 3):**

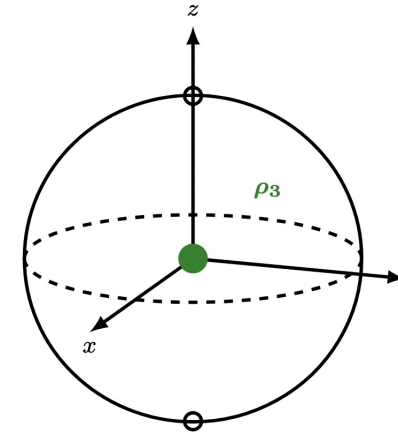
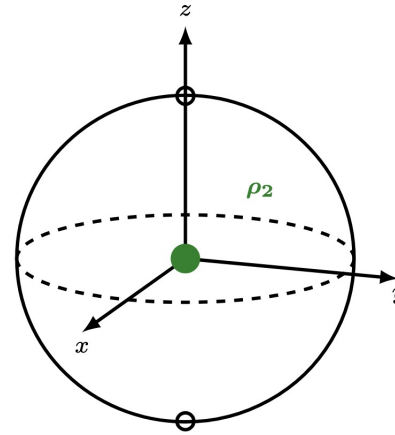
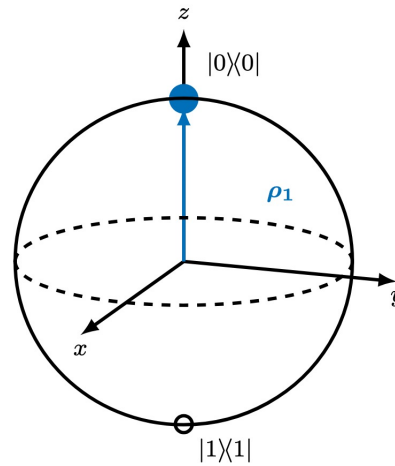


**Tensor Product
Structure 2
(Qubits 1', 2', 3'):**



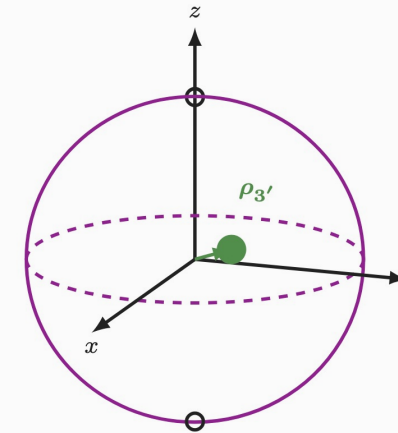
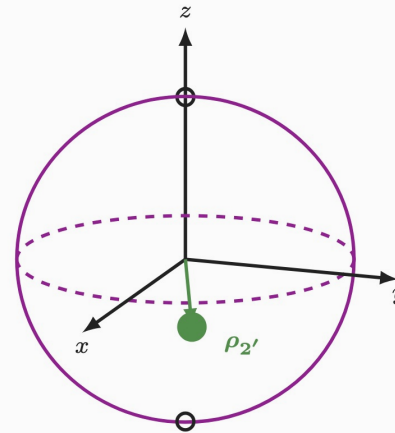
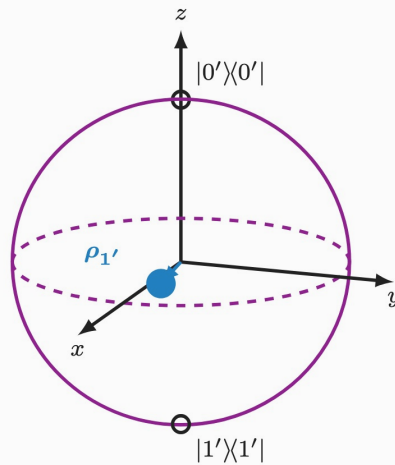
Tensor Product Structures

**Tensor Product
Structure 1
(Qubits 1, 2, 3):**

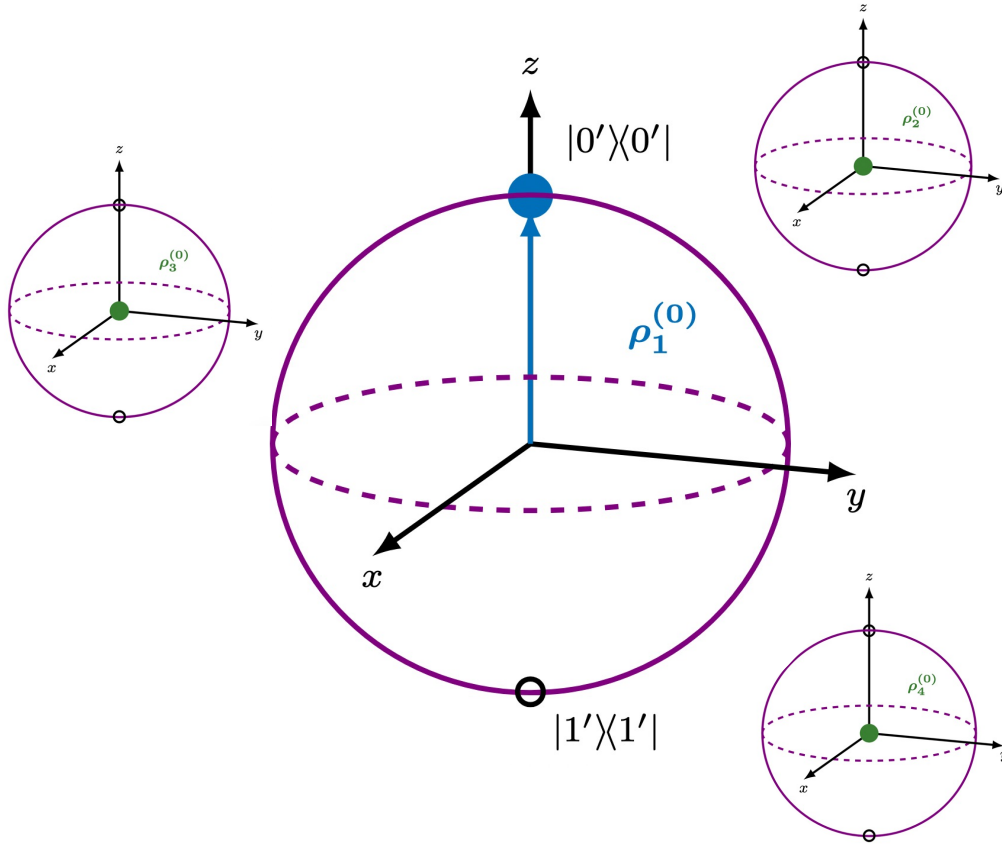


$|\Psi\rangle$

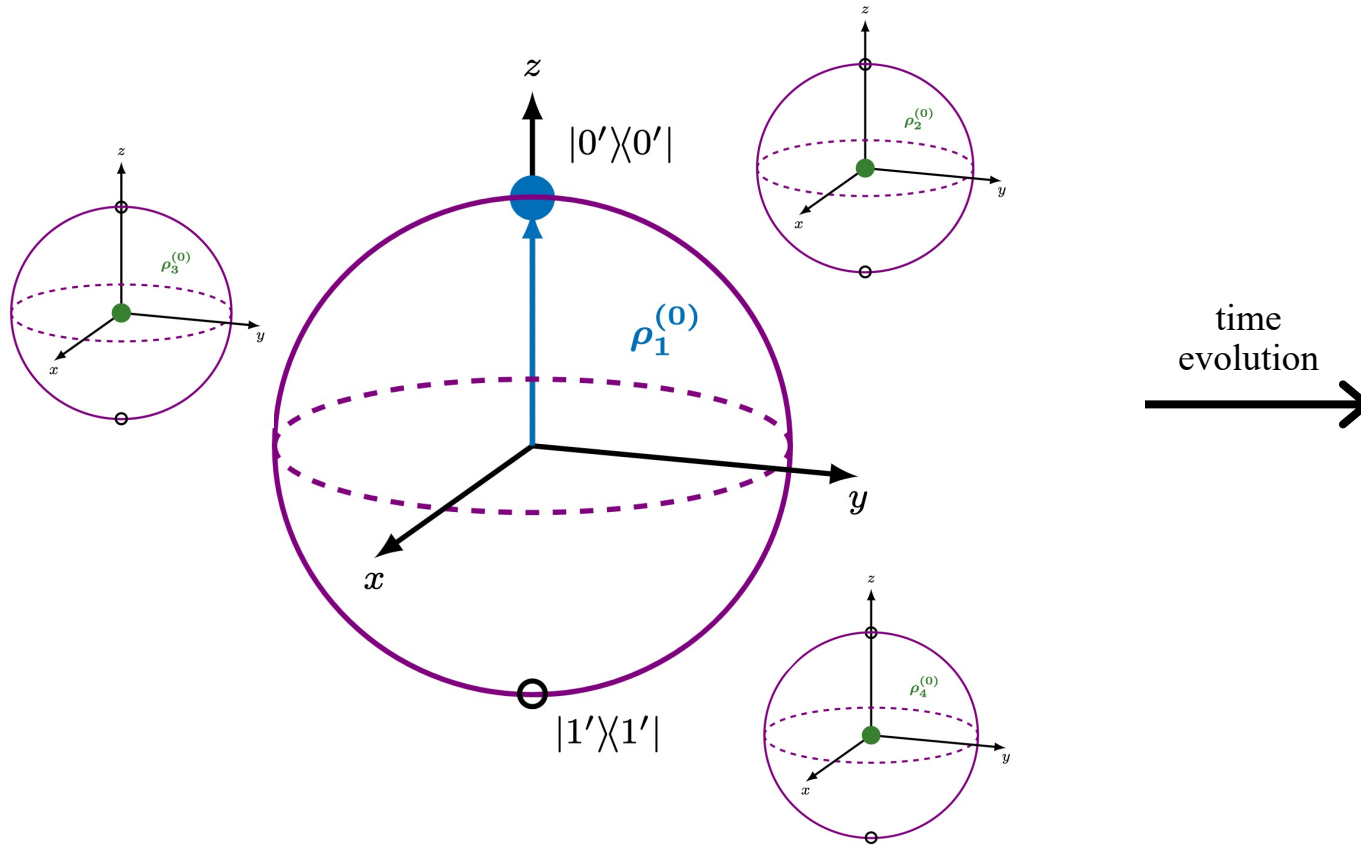
**Tensor Product
Structure 2
(Qubits 1', 2', 3'):**



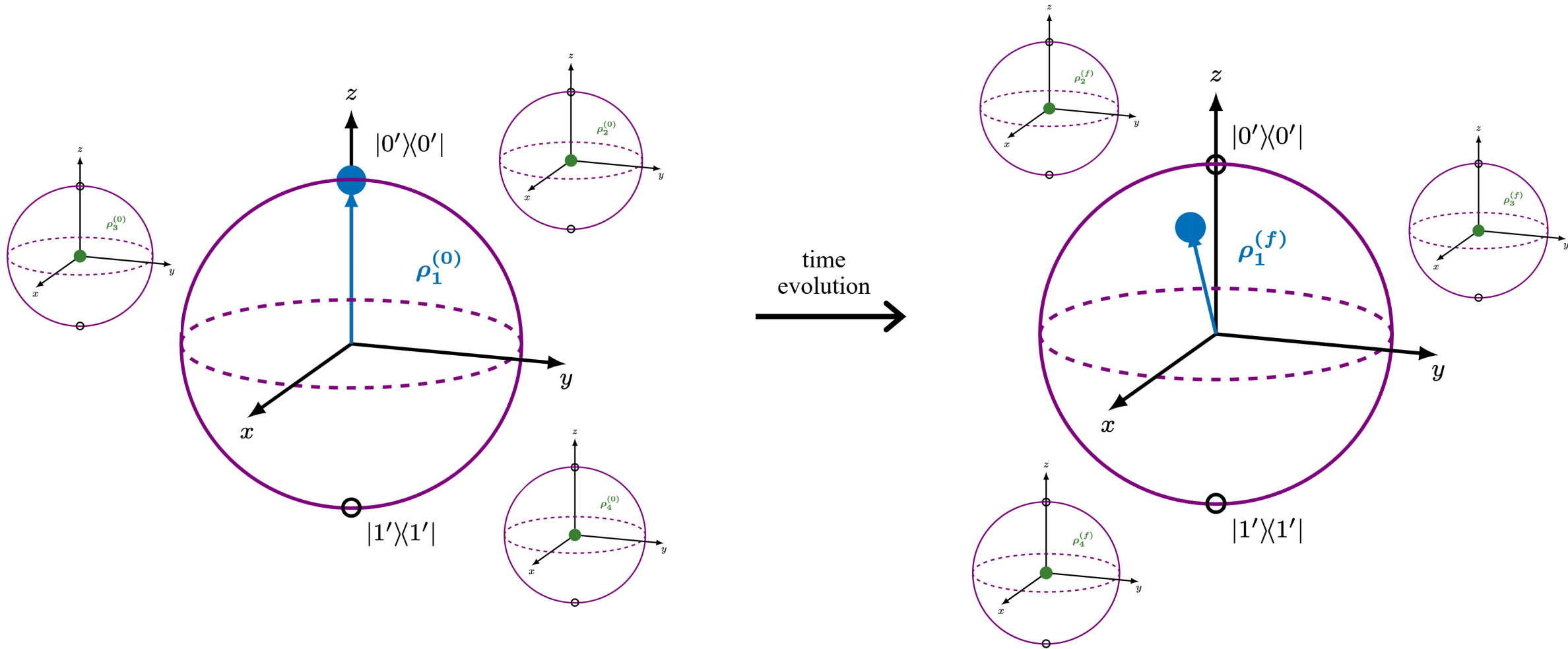
Generic Tensor Product Structure



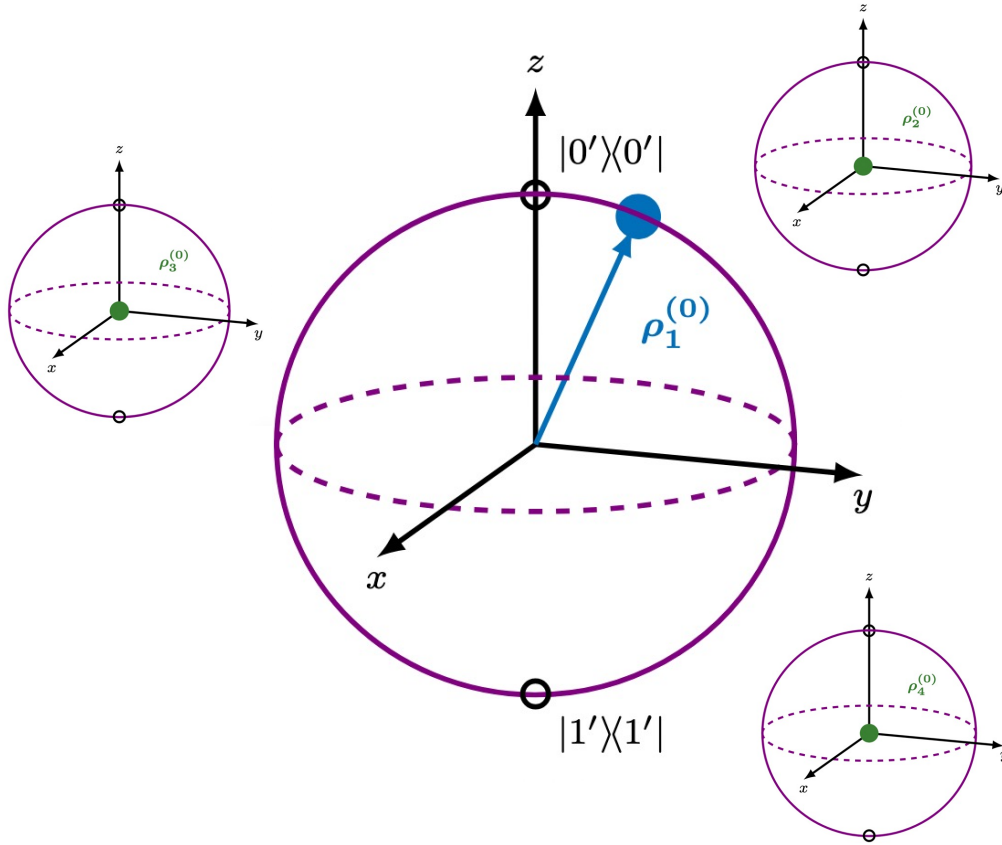
Generic Tensor Product Structure



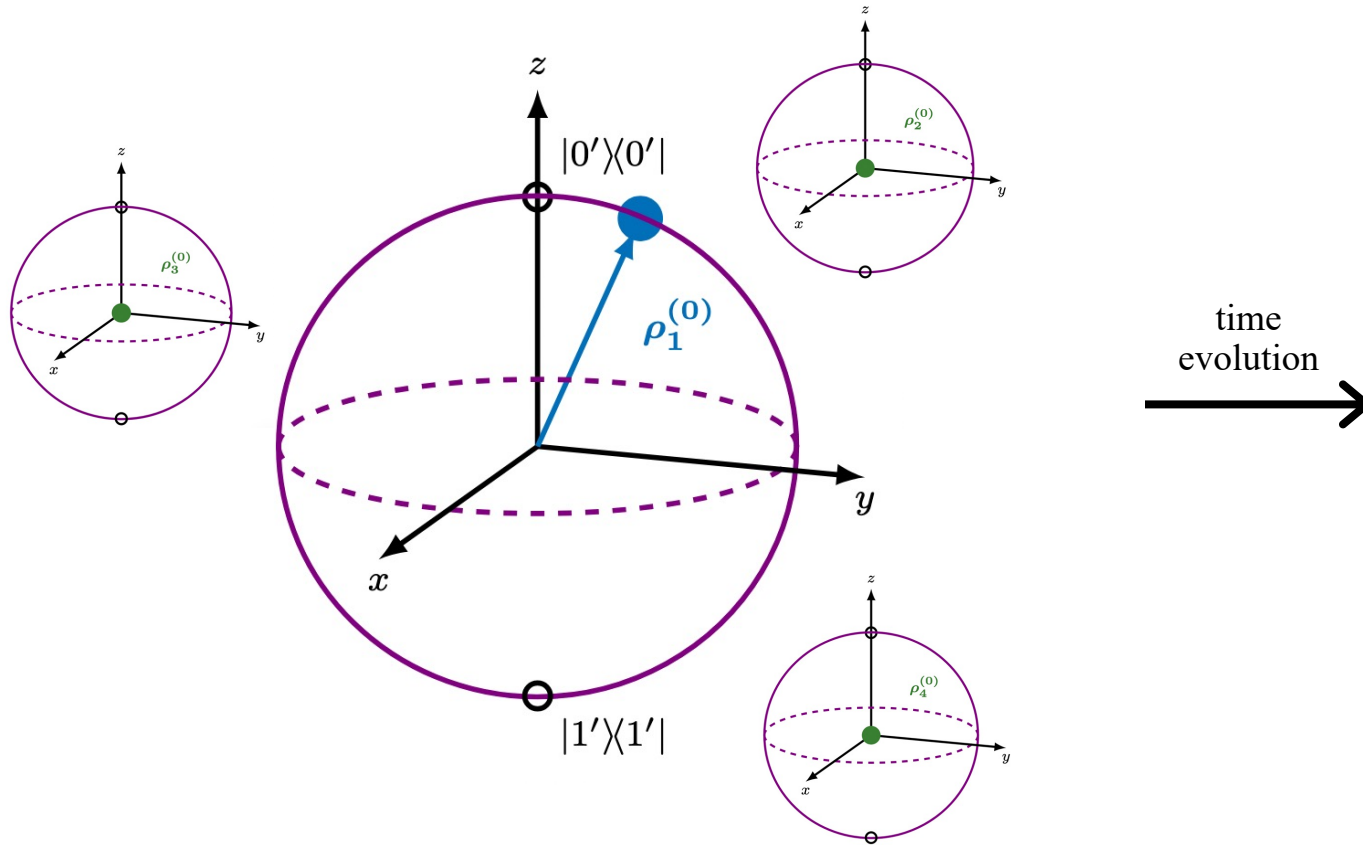
Generic Tensor Product Structure



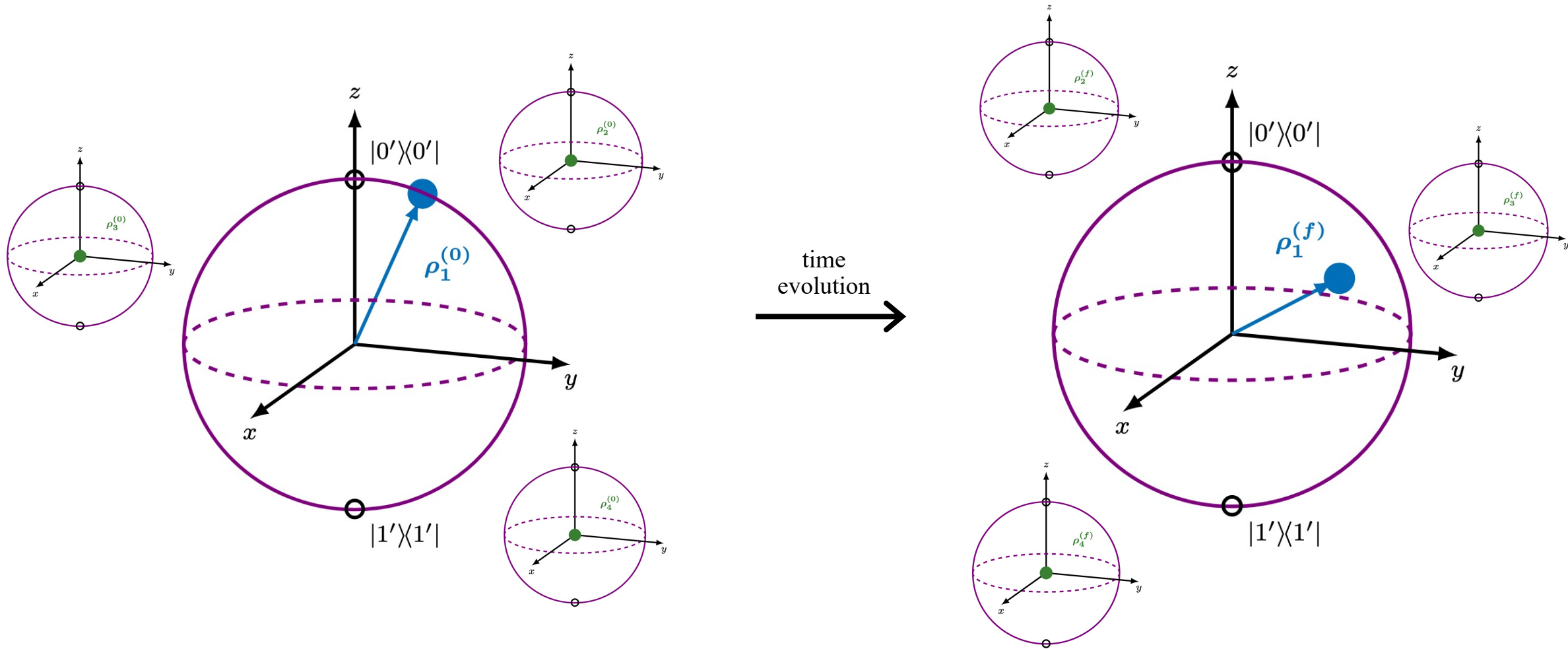
Generic Tensor Product Structure



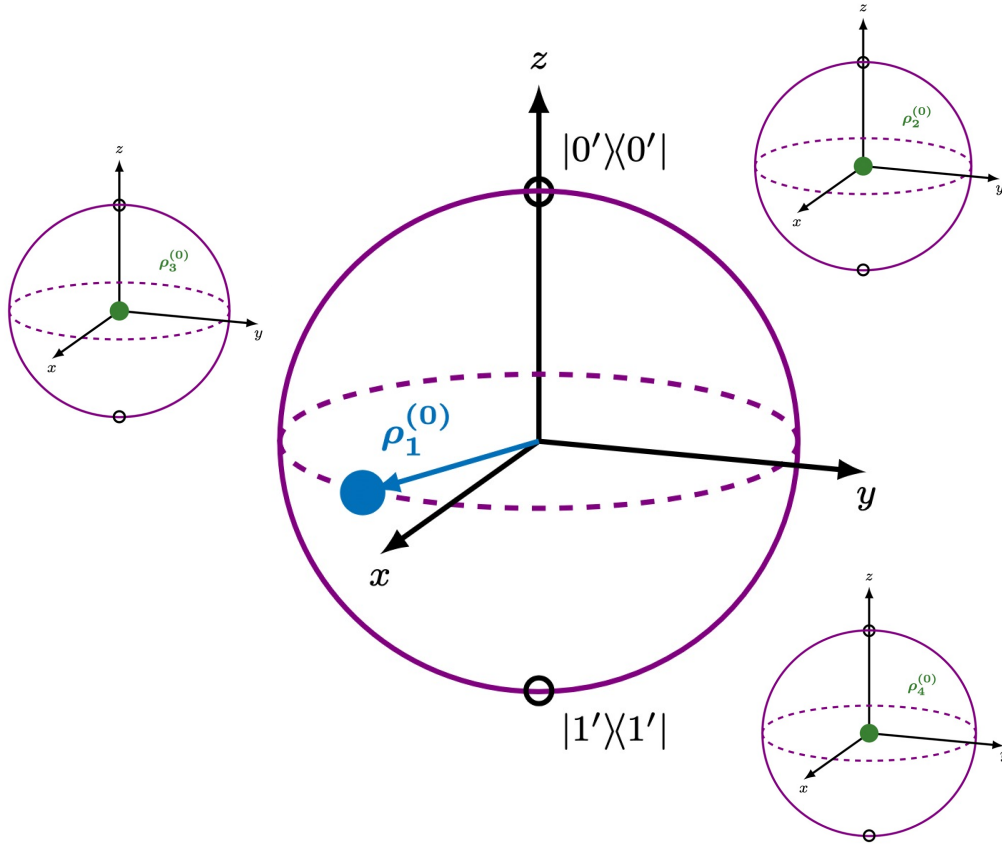
Generic Tensor Product Structure



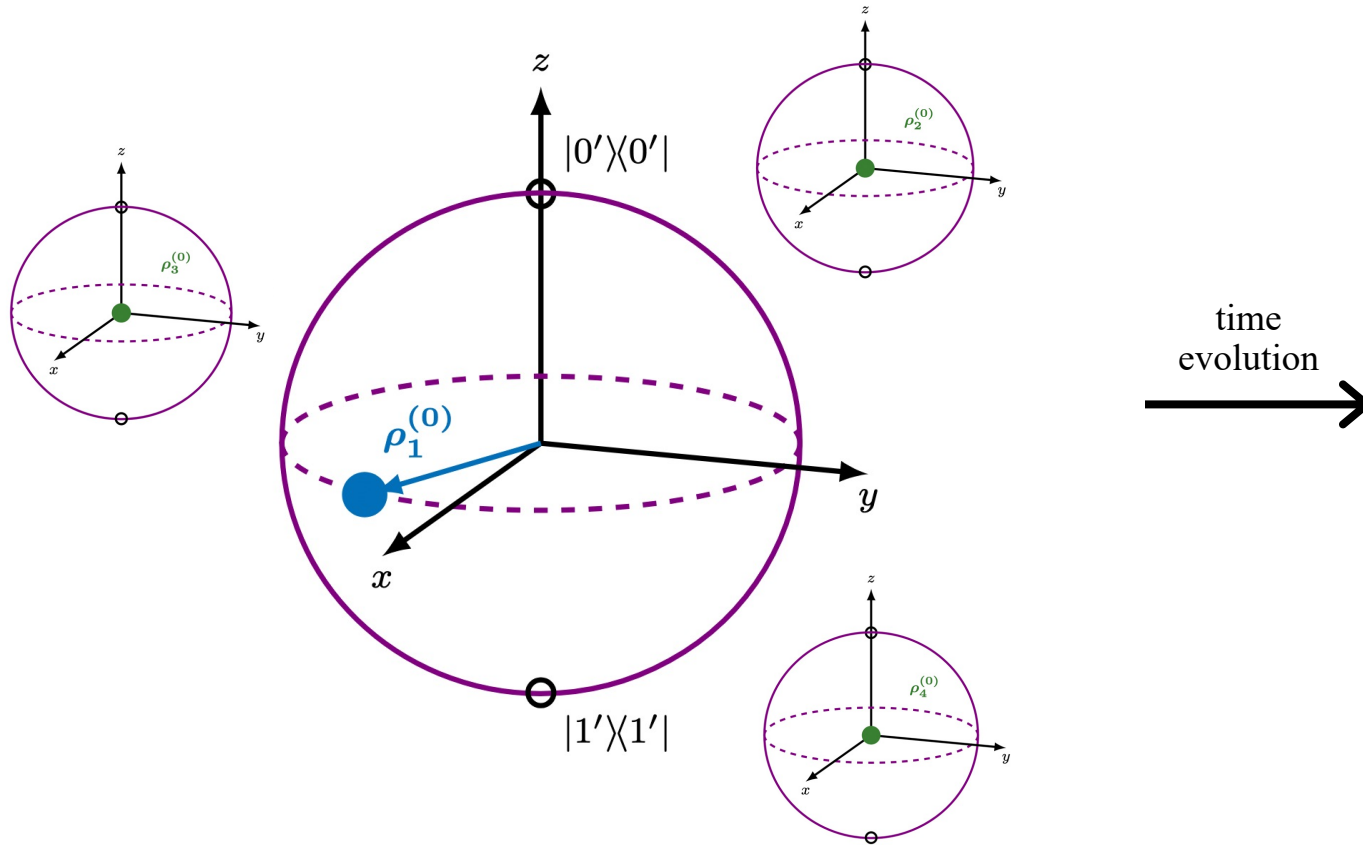
Generic Tensor Product Structure



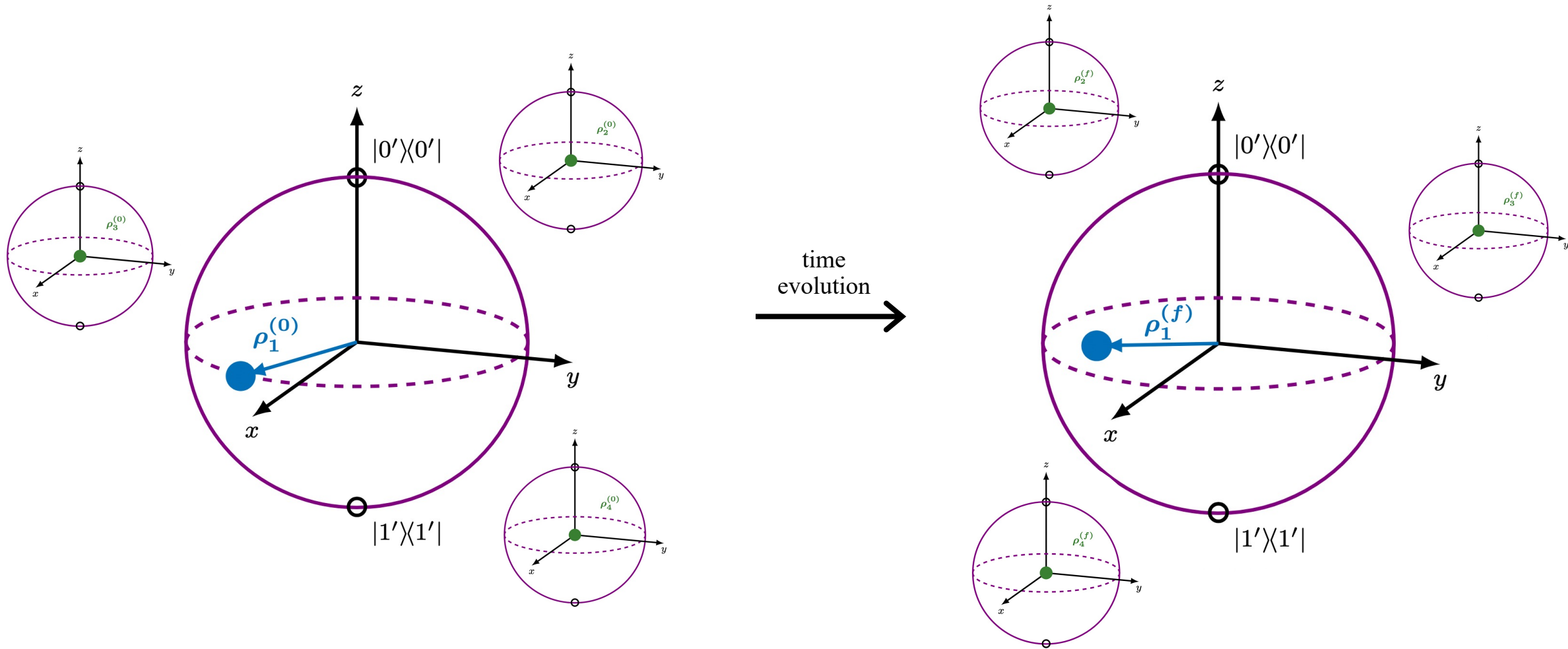
Generic Tensor Product Structure



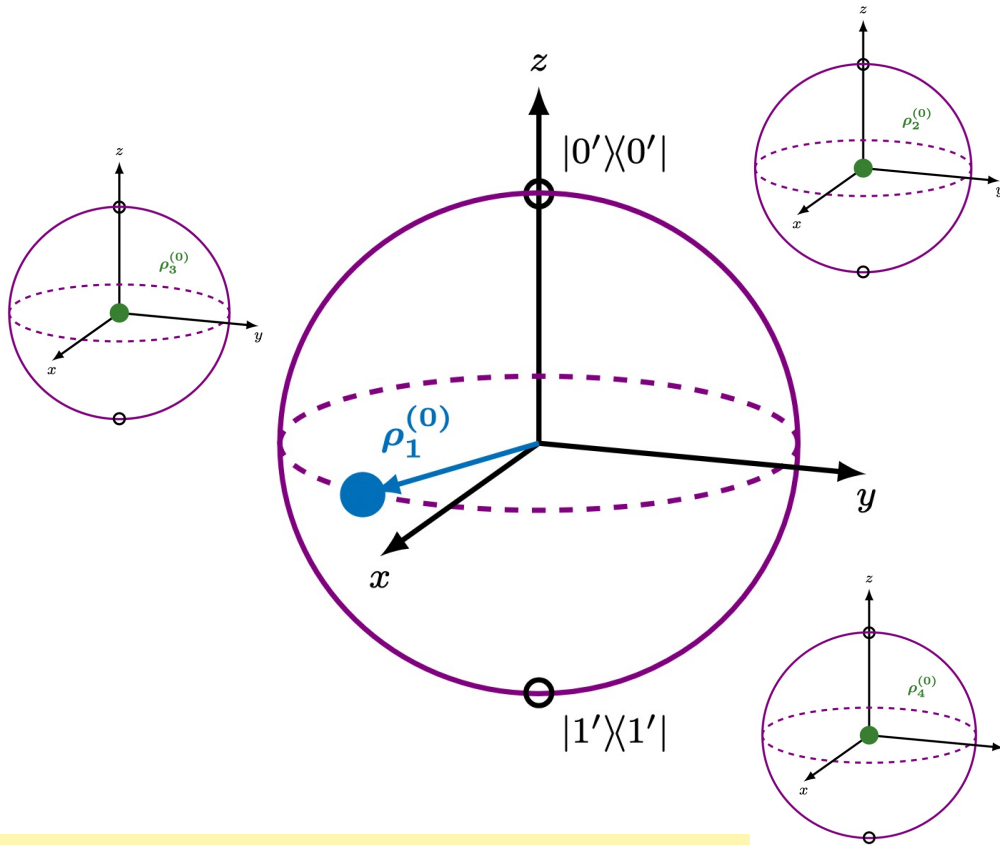
Generic Tensor Product Structure



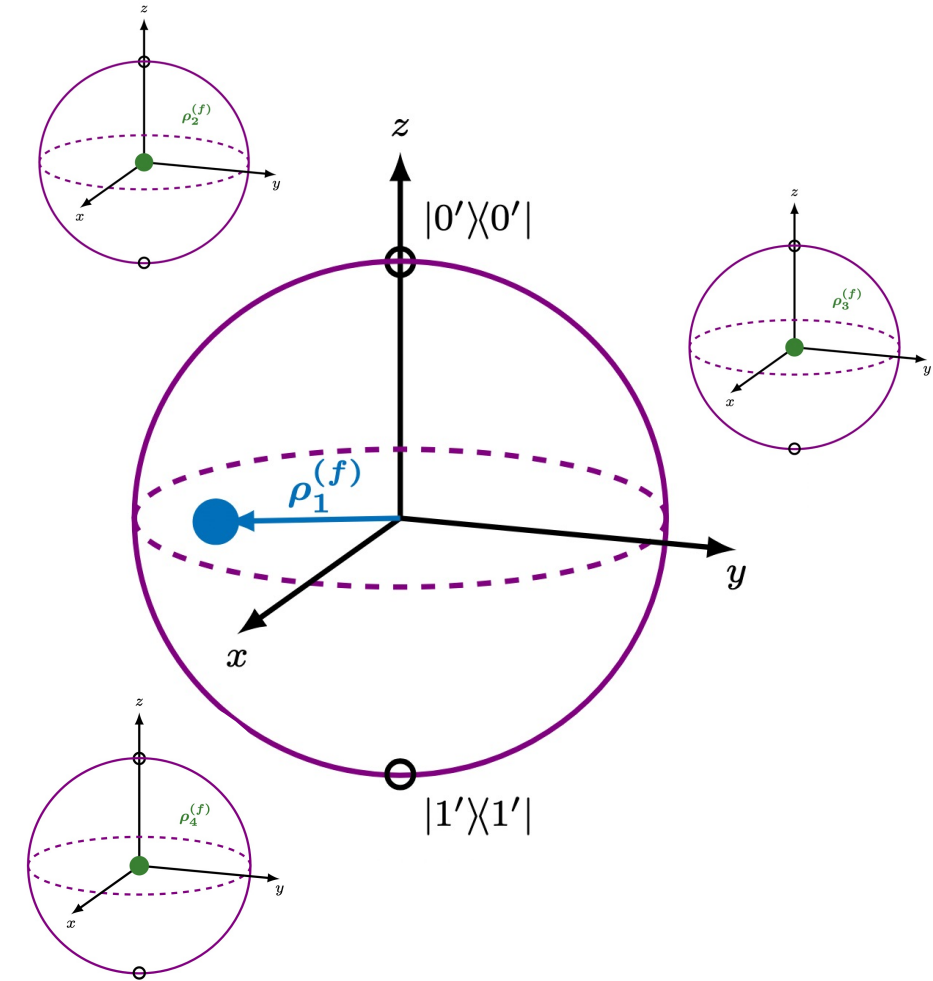
Generic Tensor Product Structure



Generic Tensor Product Structure



time
evolution
→



**In general, a TPS has
no well-defined pointer state**

How to Determine the Preferred Division into Subsystems?

- Require that a valid TPS can exhibit quasi-classical dynamics (has a well-defined pointer state)
- Metrics:
 - Predictability
 - Robustness
- Carroll and Singh [1]: Both metrics are required

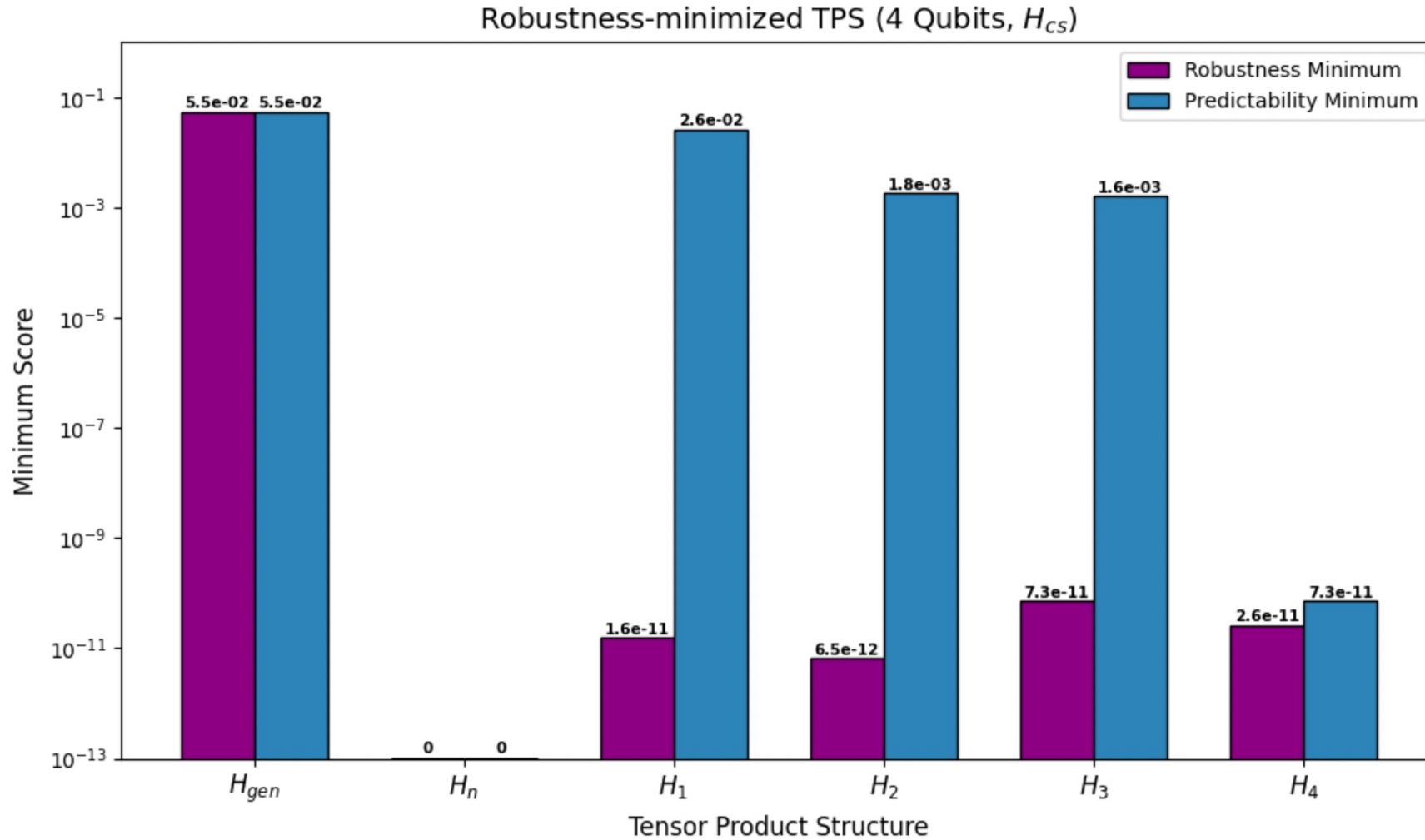
Primary Guiding Question of Thesis:

What is the relationship between the metrics of “predictability” and “robustness”?

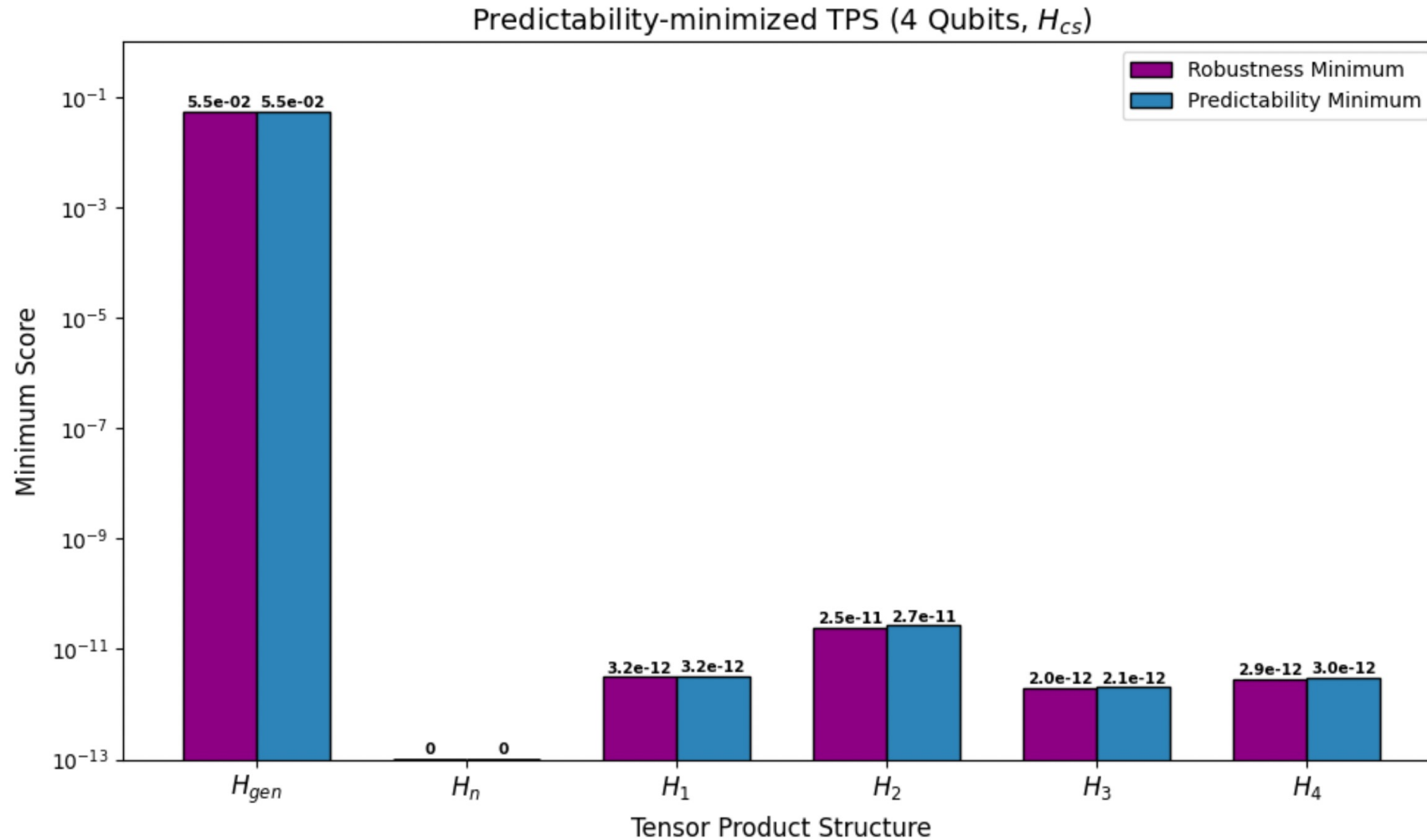
Method:

Use a machine learning algorithm to search the space of all possible TPSs to find qcTPSs according to each metric.

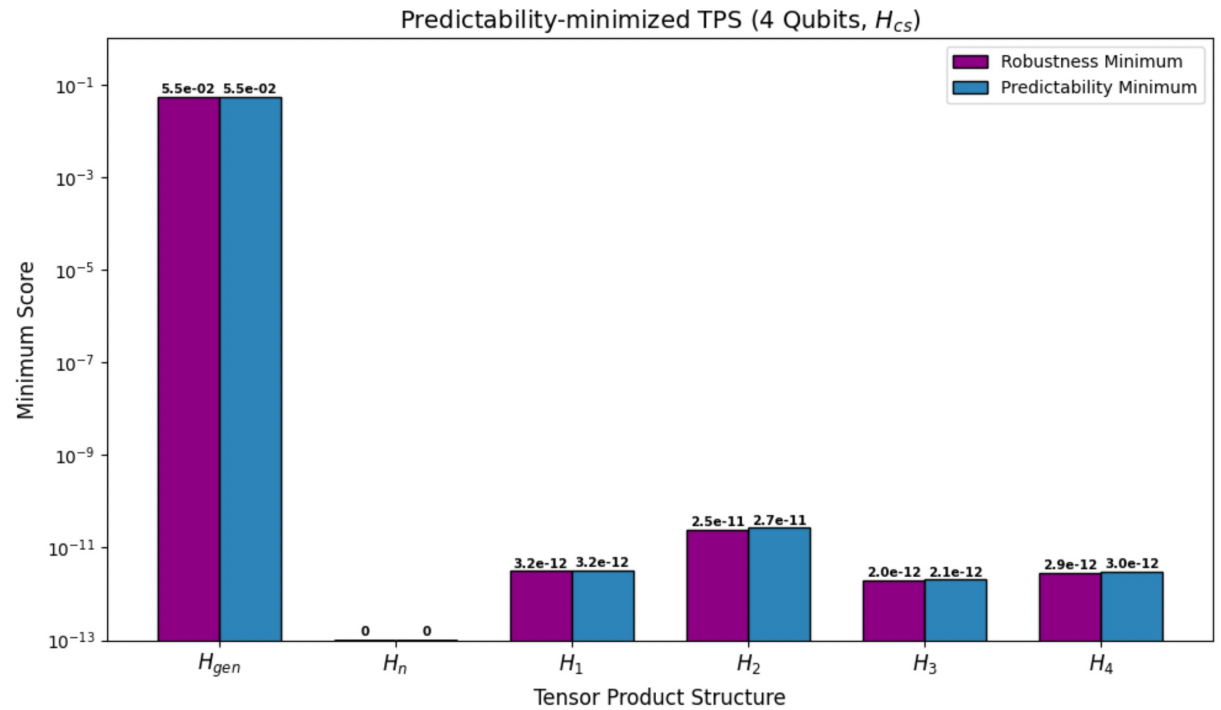
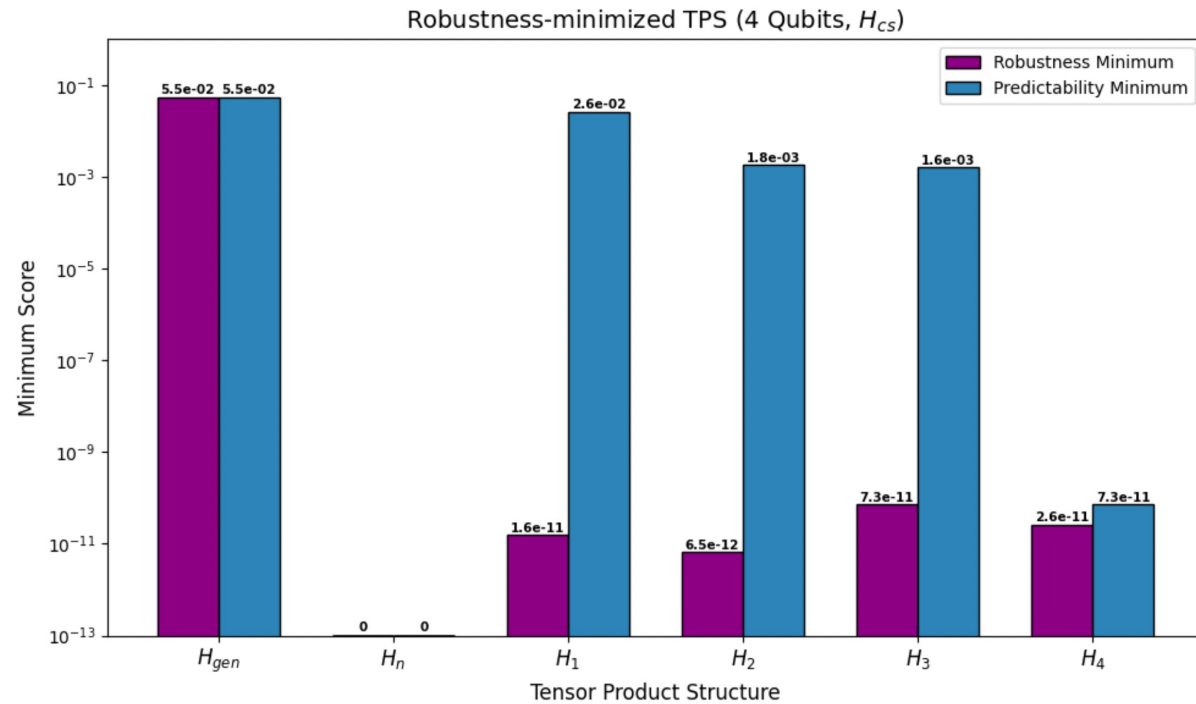
Results: Predictability Implies Robustness



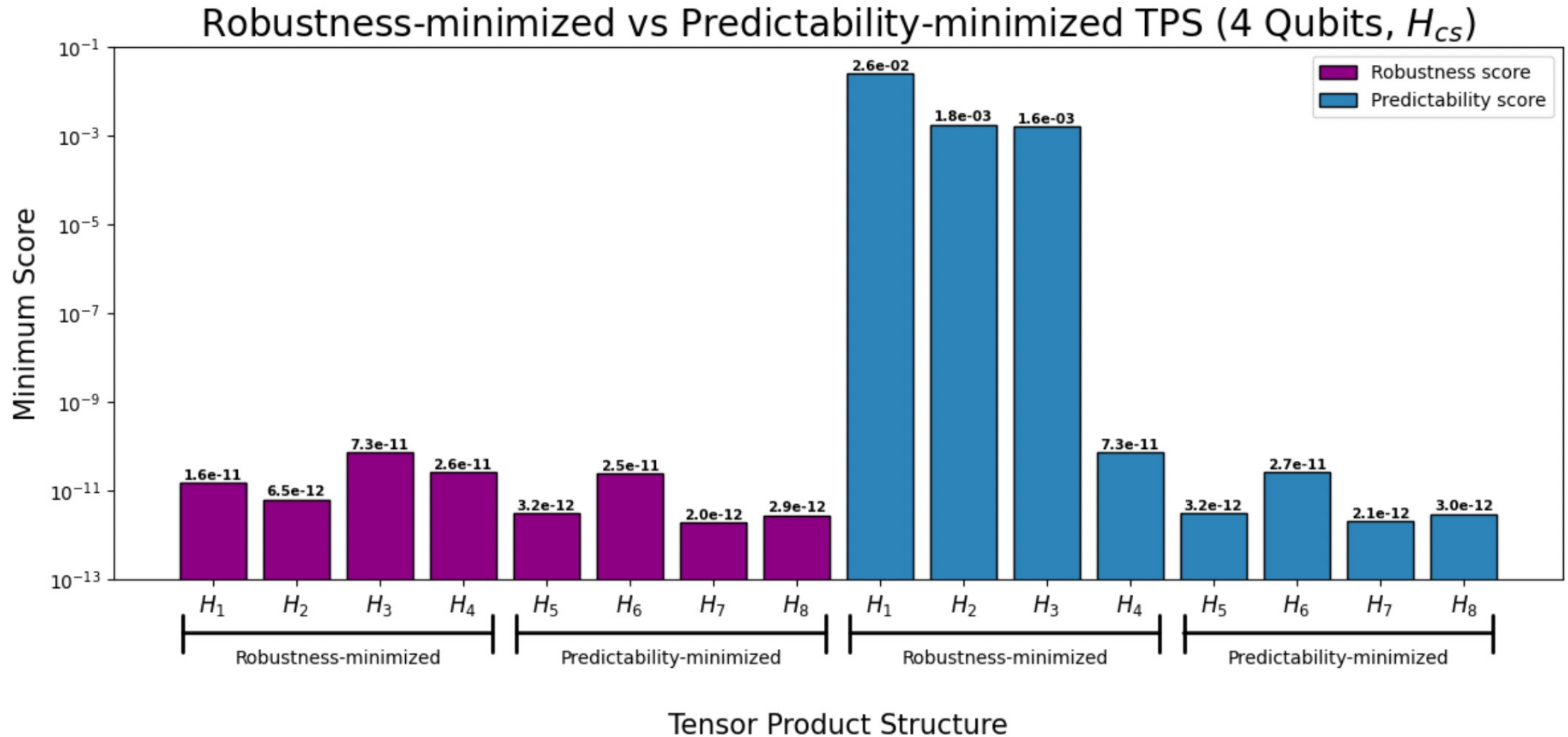
Results: Predictability Implies Robustness



Results: Predictability Implies Robustness



Results: Predictability Implies Robustness



Conclusions

1. Predictability implies robustness.

Conclusions

1. Predictability implies robustness. In particular, a TPS with a minimal predictability score must also have a minimal robustness score.

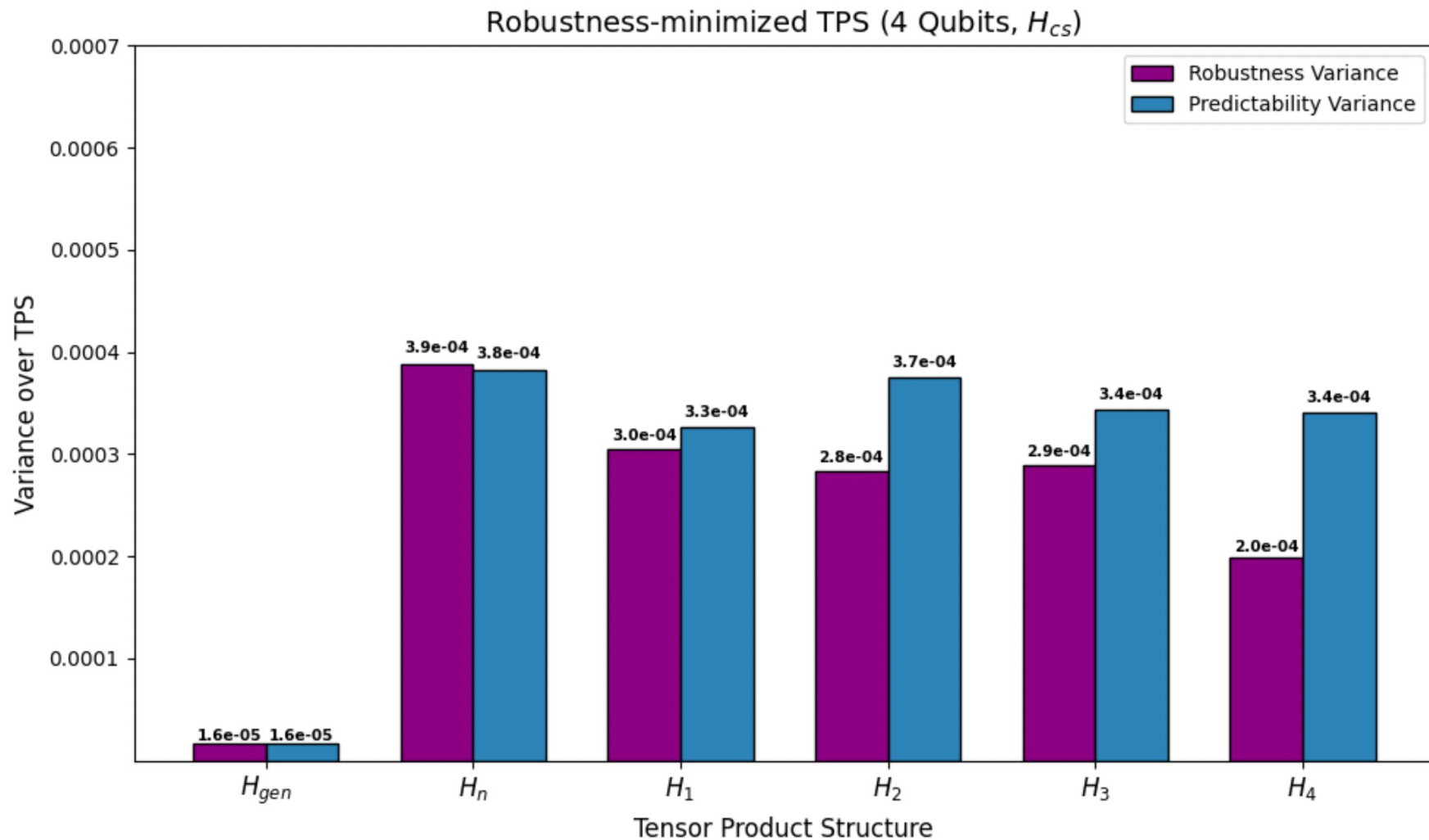
Conclusions

1. Predictability implies robustness. In particular, a TPS with a minimal predictability score must also have a minimal robustness score.
2. *Quasi-classical TPSs exhibit a significantly higher variance of scores across their initial states than a generic TPS.*

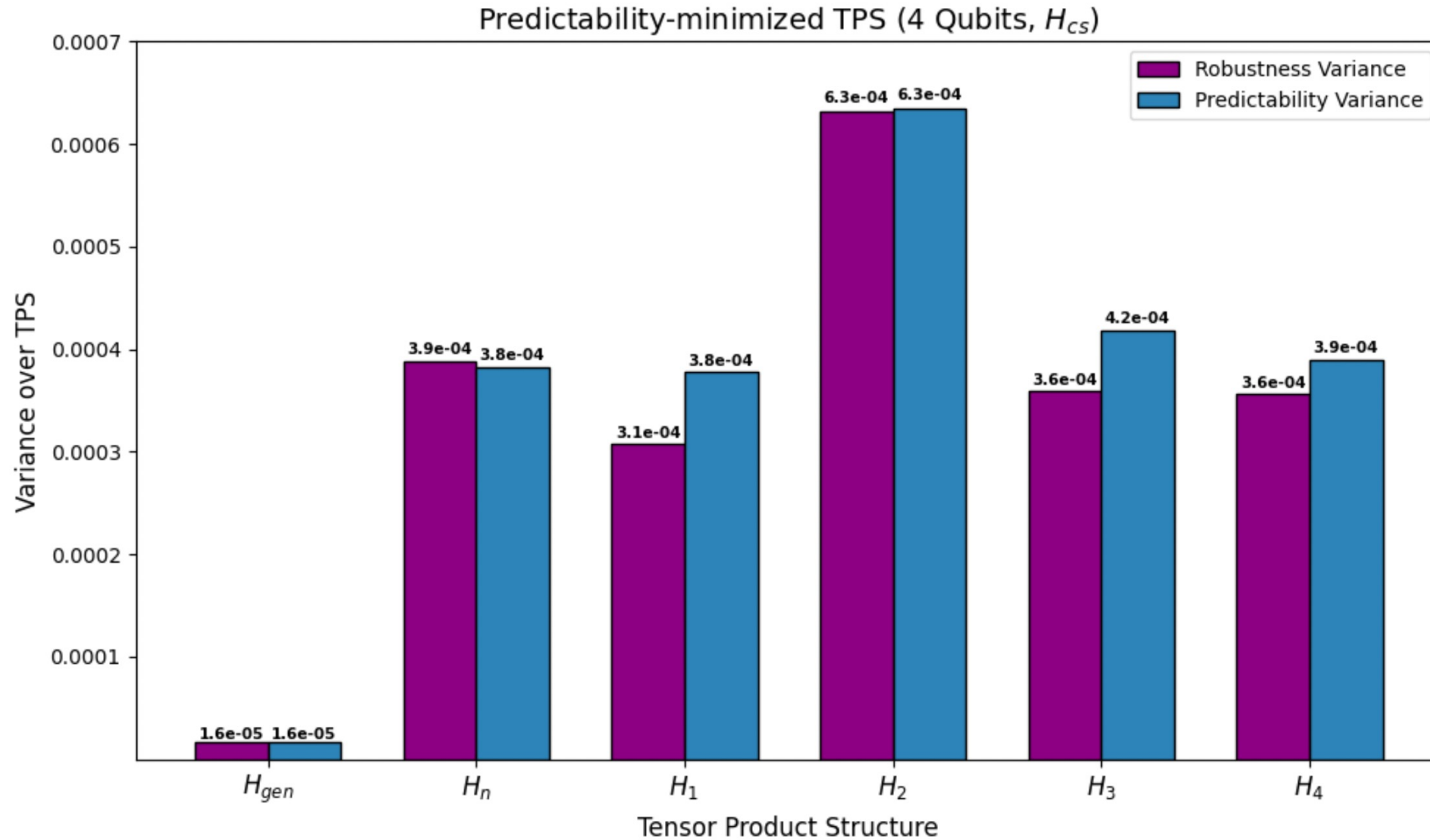
Gratitude

- Prof. Setter for all of the time he's spent talking with me about quantum and his incredible guidance
- Dwight, Prof. Moore, and the rest of the Pomona Physics Department for their unwavering support both throughout this thesis and my time at Pomona
 - Also Dwight for pushing me to attempt an explanation of my thesis without any equations
- All of the context in my life that has provided me with the privileged opportunity to study physics at Pomona College
- The indeterminacy of quantum mechanics for deeply connecting me to the universe

Results: Variance Greater for qcTPS



Results: Variance Greater for qcTPS



What Did I Actually Do?

```
def TPS_min(nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, numOrbits, numLU, minimizeVarN, fn_num, threshold):
    """Current Code Flow with Minimization:
    1. Add native H line
    2. Find values of theta (that parameterize a global unitary which transforms H to a distinct H of another orbit) that minimize CF1
    3. Determine the corresponding Hamiltonian and permutations (relabeling of tensor factors) of that Hamiltonian
    4. Check to see if Hamiltonian is LUE to another Hamiltonian already found by the procedure
    5. Apply LU transformations to generate entire TPS orbit (lightly colored dots on plot)
    6. Store min and mean values of each metric
    """

    orbitLabels = ['SH_n$']
    orbitScores, orbitMeans, orbitVariances = [], [], []
    orbitSample, orbitSamples = [], []
    minHams, minTPSThetas = [], []

    nullThetas = [0]*(dim_tot**2-1)
    native_ham = scam(nVar, initial_hamVar, nullThetas, GGMMs, 0)
    NCList = []

    # Adding native TPS first (H_0)
    rob_score = cost(nullThetas, nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, 0, 2)
    pred_score = cost(nullThetas, nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, 0, 4)
    orbitScore = [rob_score, pred_score]

    for i in range(numLU):
        # determine scores across *numLU* # of local unitary transformations for plotting
        lu_thetas = randomThetas(1)
        rob_score = cost(lu_thetas, nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, 1, 3) # scam w/ local uni
        pred_score = cost(lu_thetas, nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, 1, 4) # scam w/ local uni
        orbitSample.append([rob_score, pred_score]) # append sampled scores across LUs
        orbitScore.append([rob_score, pred_score])

    mean_rob, var_rob = mean_variance_fn(nVar, native_ham, GGMMs, tensor_initial_statesVar, 4)
    mean_pred, var_pred = mean_variance_fn(nVar, native_ham, GGMMs, tensor_initial_statesVar, 4)
    orbitMean = [mean_rob, mean_pred]
    orbitVariance = [var_rob, var_pred]

    # save native line data
    orbitScores.append(orbitScore)
    orbitMeans.append(orbitMean)
    orbitVariances.append(orbitVariance)
    orbitSamples.append(orbitSample)
    minHams.append(initial_hamVar)
```

```
def genericScore(nVar, initial_hamVar, GGMMs, tensor_initial_statesVar, numOrbits):
    """Computes the scores and statistics for a "typical" Hamiltonian using CF2/CF4"""
    rob_list, pred_list = [], []
    var_list, pred_list = [], []
    nullThetas = [0]*(dim_tot**2-1)

    k = 0
    for i in range(numOrbits):
        # clear single TPS lists for new orbit
        cf2_score = []
        cf4_score = []

        # Scramble to random_Ham with global unitary
        orbit_thetas = randomThetas(0)
        random_Ham = scam(nVar, initial_hamVar, orbit_thetas, GGMMs, 0)

        # compute scores for given orbit and append to list
        cf2_score = cost(nullThetas, nVar, random_Ham, GGMMs, tensor_initial_statesVar, 0, 2)
        cf4_score = cost(nullThetas, nVar, random_Ham, GGMMs, tensor_initial_statesVar, 0, 4)
        rob_list.append(cf2_score)
        pred_list.append(cf4_score)

        # get mean and variance for orbit and append to lists
        mean_rob, var_rob = mean_variance_fn(nVar, random_Ham, GGMMs, tensor_initial_statesVar, 2)
        mean_pred, var_pred = mean_variance_fn(nVar, random_Ham, GGMMs, tensor_initial_statesVar, 4)
        var_list.append(mean_rob)
        pred_list.append(mean_pred)
        var_list.append(var_rob)
        pred_list.append(var_pred)

    # Average over the min scores, means, and variances to obtain stats for "generic" TPS
    if i % 100 == 0:
        print(f'Computed {i} TPS out of {numOrbits}')
    gen_rob = np.mean(rob_list)
    gen_pred = np.mean(pred_list)
    gen_mean_rob = np.mean(var_list)
    gen_mean_pred = np.mean(pred_list)
```

$$S_{\text{ent}}(\vec{n}) = 1 - \frac{1}{2}$$

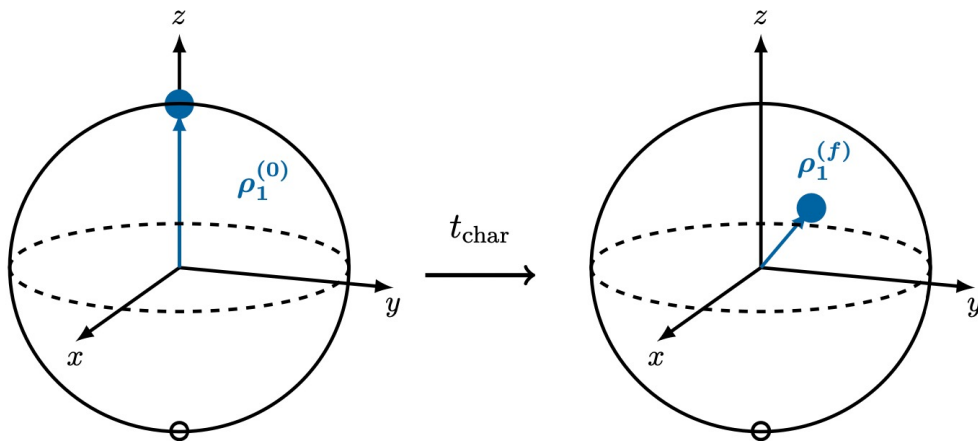
$$\langle S'_{\text{pointer}} \rangle = \left\langle \left(\sum_{i,j} n_i n_j \tilde{P}_{ij} \right)^2 \right\rangle$$

$$\int_0^{2\pi} \sin \theta d\phi d\theta$$

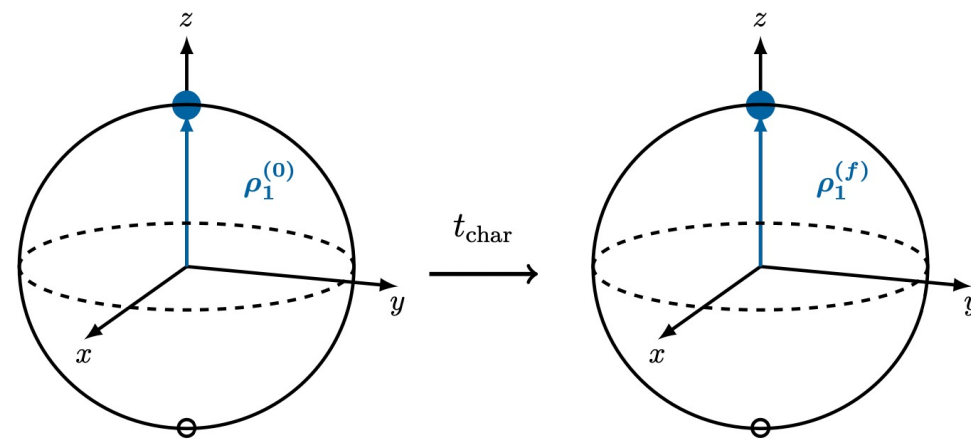
$$+ 4 \text{Tr}(\tilde{P}^2)^2$$

Predictability and Robustness

Neither robust
nor predictable:



Robust and
predictable:



Robust, not
predictable:

