Make testing magical again

Thomas Eizinger: @wheezle@hachyderm.io

When was the last time you felt like a magician?

What do good tests look like?

```rust
#[test]
fn given_an_account_when_deposit_then_increased_balance() {
    // Given / Arrange
    let mut account = Account::new(0);

    // When / Act
    account.deposit(1000);

    // Then / Assert
    assert_eq!(account.balance(), 1000);
}
```

Have we covered all the cases?

- What if the initial balance is higher?
- Does this work for all deposit amounts?
  - What about $0?
  - What about $1_000_000_000_000?

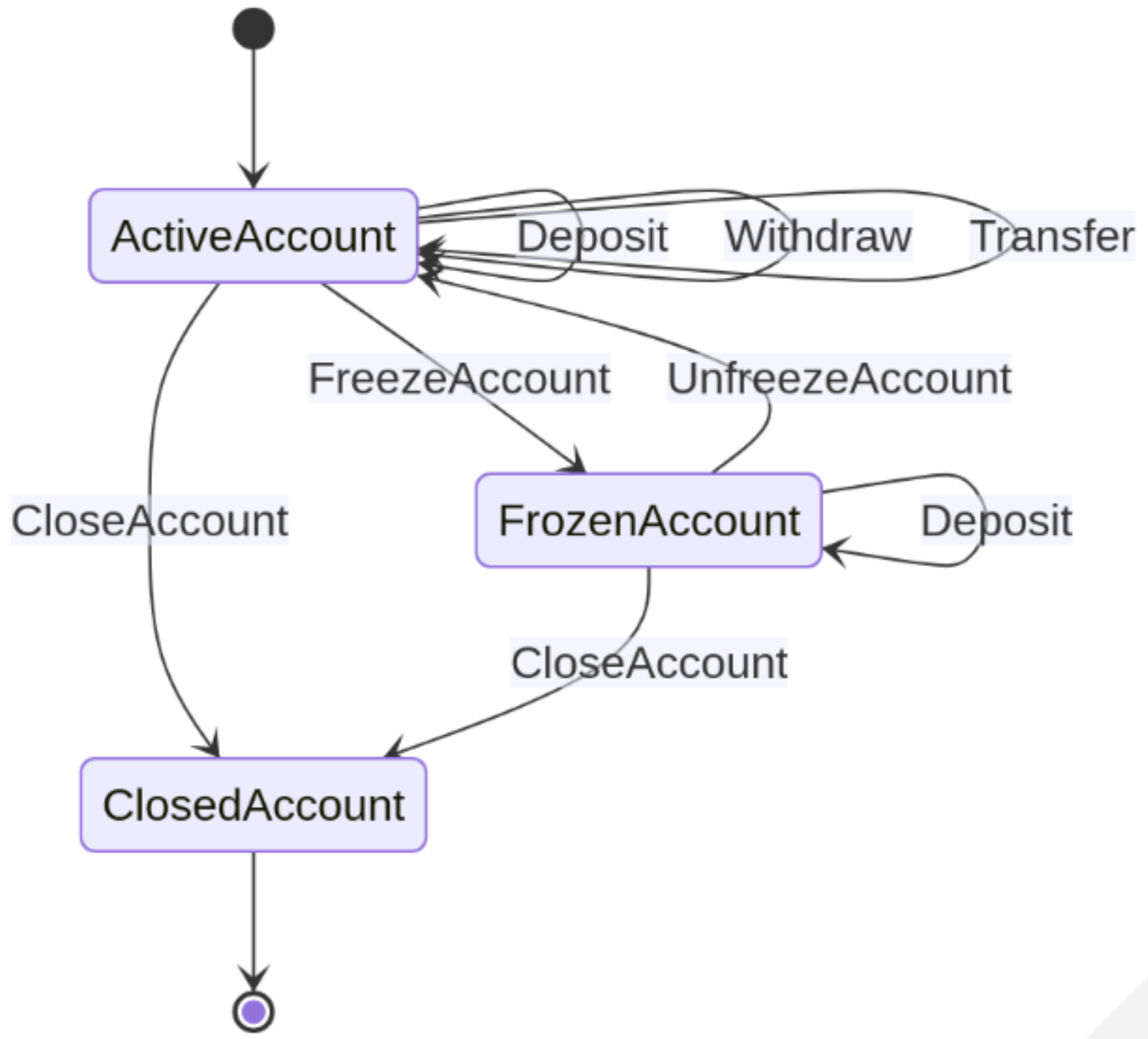
Property-based testing

```rust
#[test_strategy::proptest]
fn given_an_account_when_deposit_then_increased_balance(
    #[strategy(any::<u64>())] balance: u64,
    #[strategy(any::<u64>())] deposit: u64
) {
   let mut account = Account::new(balance);

   account.deposit(deposit);

   assert_eq!(account.balance(), balance + deposit);
}
```

State machines!



Property-based state machine testing

1. Define your source state as a "strategy"
2. Define your state transitions as a strategy (Deposit, Withdraw, etc)
3. Implement a "reference" state machine

- github.com/thomaseizinger/proptest-state-machine-banking
- github.com/firezone/firezone

Questions?