state transition

状態遷移

有限状態機械の設計と実装

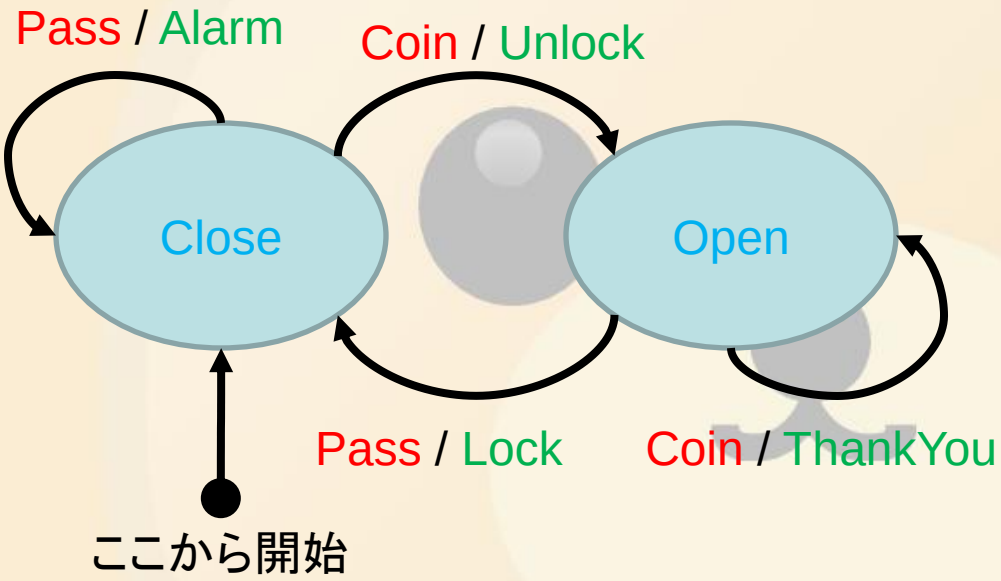
わんくま同盟

επιστημη episteme@cppll.jp

Microsoft MVP for Visual Developer Visual C++

状態遷移ってナニよ?

状態遷移図
state transition diagram



State : 状態　Event : 事象　Action : 動作

駐車場でよく見かけるコレ

状態遷移表 (state transition table / state map)

状態遷移図の表による表現

Action : 動作
→ 次の状態(遷移)

State : 状態

Event : 事象	Close	Open
Coin	Unlock → Open	ThankYou (→ Open)
Pass	Alarm (→ Close)	Lock → Close

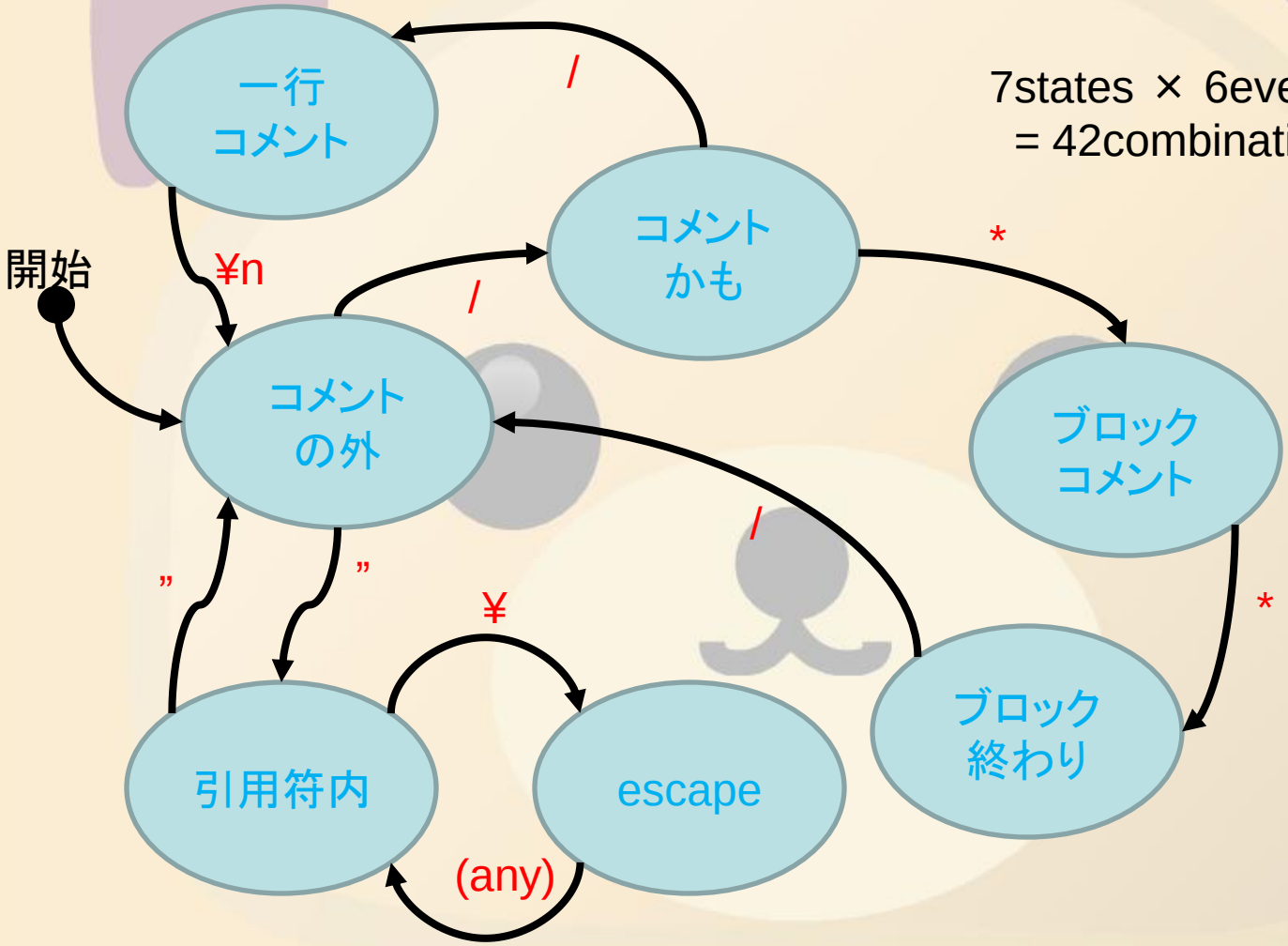
駐車場の遮断機をシミュレートしてみよう

```
private enum State { Close, Open }
private State state_ = State.Close;

private void buttonPass_Click(object sender, EventArgs e) {
    switch ( state_ ) {
    case State.Close:
        textAction.Text = "コラ!";
        break;
    case State.Open:
        textAction.Text = "ゲートを閉じます";
        state_ = State.Close;
        break;
    }
    textState.Text = state_.ToString();
}
```

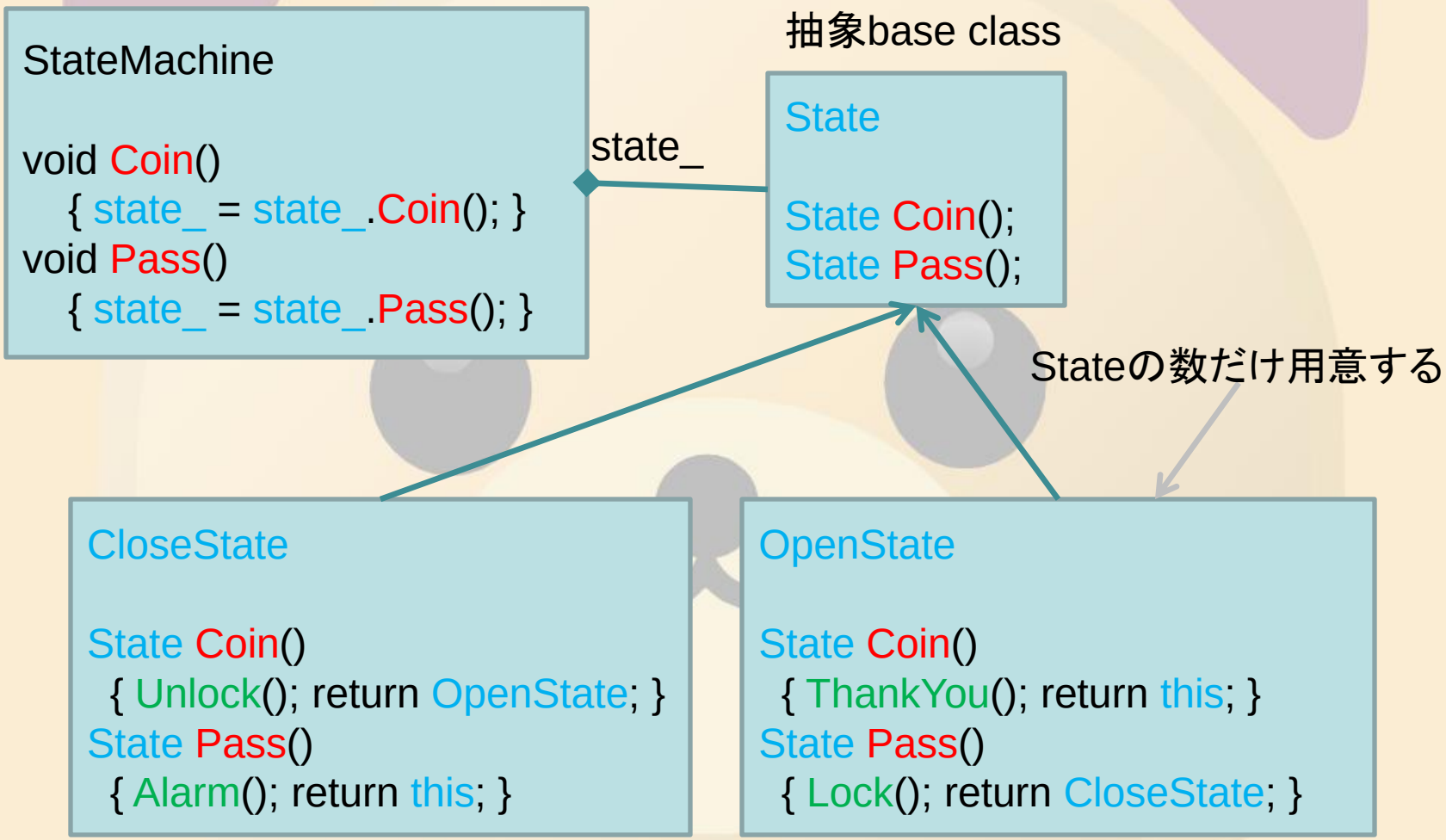
状態の数だけ並べなきゃなんないね…

C++コメント抽出を行う状態遷移



7states × 6events
= 42combination

STATE Pattern を使うぞ、と。



Next Step… StateMapCompiler

- XMLで記述した状態遷移表に基づいて Context, State,そしてStateMachineの各コードを**自動生成**する。

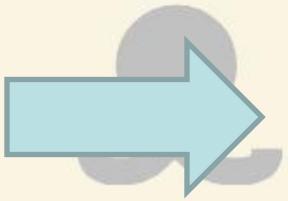
GateFSM. xml

```
<?xml version='1.0' encoding='shift_jis' ?>
<!DOCTYPE fsm SYSTEM 'smc.dtd'>
<fsm name='Gate' initial='Close'>

  <state name='Close' entry='Lock'>
    <event name='Coin'>
      <transit state='Open' if='Confirm'/>
    </event>
    <event name='Pass'>
      <execute action='Alarm'/>
    </event>
  </state>

  <state name='Open' entry='Unlock'>
    <event name='Coin' guard='Confirm'>
      <execute action='ThankYou'/>
    </event>
    <event name='Pass'>
      <transit state='Close'/>
    </event>
  </state>

</fsm>
```



GateFSM. cs

```
namespace Gate {

  public enum StateCode { Close, Open, Unknown }
  public enum EventCode { Coin, Pass, Unknown }

  public interface Context {
    void Error(StateCode s, EventCode e);
    bool Confirm();
    void Lock();
    void Unlock();
    void Alarm();
    void ThankYou();
  }

  public abstract class State {
    public abstract string Name { get; }
    public abstract StateCode Code { get; }
    internal virtual void Enter_(Context ctx) {}
    internal virtual void Exit_(Context ctx) {}
    internal virtual State Coin(Context ctx)
      { ctx.Error(Code,EventCode.Coin); return this; }
    internal virtual State Pass(Context ctx)
      { ctx.Error(Code,EventCode.Pass); return this; }
  }
  …
}
```