Учетные системы

История чисел, состояний и изменений

Содержание

- Измеримые величины их моделирование:
  От натуральных чисел до квантовой механики
- Учет изменений и вычисление состояний:
  Диаграммы состояний, регистры
- Модели учетных операций
  Роли и участники

Обо мне

- 2000-2010 ОЛМА Медиа Групп
  - Управление торговлей
  - Управление складом
- 2008-2015 ТД Амадеос
- 2016-2017 Нитэкмаш Сервис

Числа. История

- Первыми придумали натуральные числа
- Развитие цивилизации потребовало усложнений

Числа. Дробные

Как посчитать, на сколько дней осталось ресурсов?

- Взять общее количество
- Вычесть количество потребителей
- Если осталось повторить
- Если не получилось, оставить на потом

$$Дней = \frac{Количество\ ресурса}{Количество\ потребителей}$$

Числа. Анализ

- Замена сложного алгоритма и простых чисел на простой алгоритм и немного более сложные числа

Числа. Отрицательные

Из двух пунктов выехали…

В первом классе

- Если v_1 — вправо и v_2 вправо, то
- Если $v_1 > v_2$, то $t = S / (v_1 — v_2)$
- Иначе

  …….

В пятом классе

$$t = \frac{S}{v_1 - v_2}$$

Лишние(?) значения в прошлом при $t < 0$.
Такие ли они лишние? Или это «бизнес-правило»?

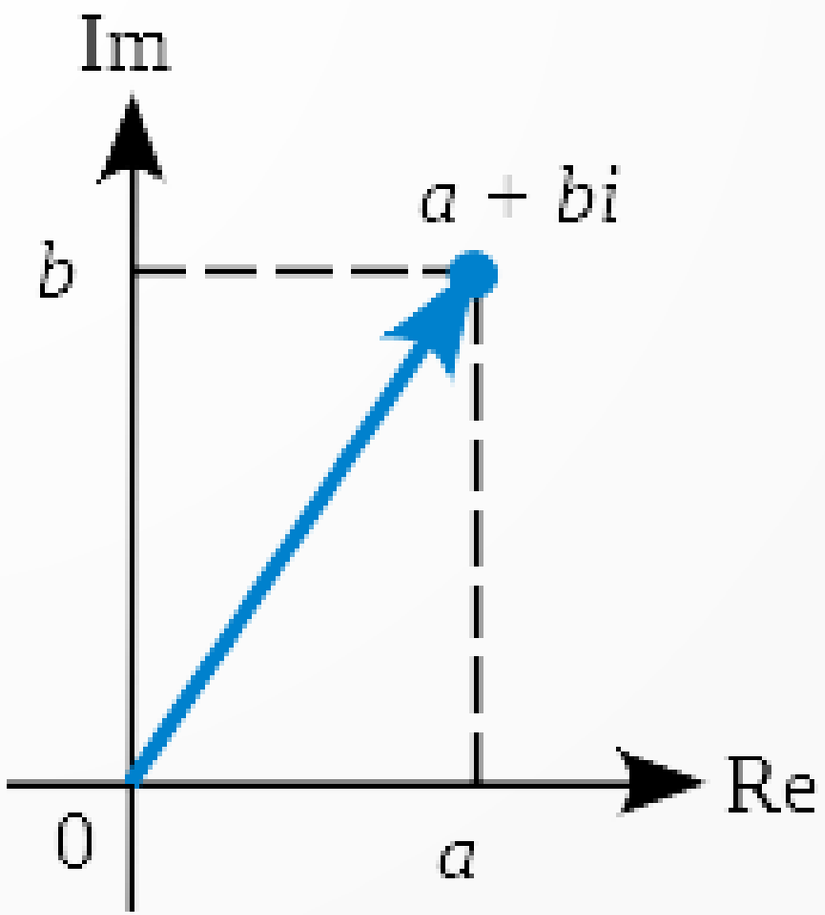
Числа. Анализ

- Замена сложного алгоритма и простых чисел на простой алгоритм и немного более сложные числа
- Разделение внутренней логики типов и «бизнес-правил»

Числа. Комплексные

- Два числа
- Упрощаем производные и дифференциальные уравнения
- Измеримый результат — все равно действительное число.

- Кстати, и отрицательных чисел в природе тоже нет...



Числа. Анализ

- Замена сложного алгоритма и простых чисел на простой алгоритм и немного более сложные числа
- Разделение внутренней логики типов и «бизнес-правил»
- Измеримое/практическое значение требует «извлечения» из результата. «Числа» - математическая абстракция.

Числа. XX век. NULL

А как представить тот факт, что мы чего-то не знаем?

y = x + 3. А что, если *x* не известен?

- Если *x* известен, то
  y = x + 3
- А если нет, то
  y — не известен

NULL

Значение + Значение =
Значение

Значение + NULL = NULL

NULL + NULL = NULL

Maybe<T>, Option<T>, Nullable<T>

Числа. Анализ

- Замена сложного алгоритма и простых чисел на простой алгоритм и немного более сложные числа
- Разделение внутренней логики типов и «бизнес-правил»
- Измеримое/практическое значение требует «извлечения» из результата. «Числа» - математическая абстракция.
- NULL — не значение, а состояние. Nullable — составное состояние.

Квантовая механика. Состояния

- Числа → Состояния
- До измерения - Суперпозиция

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Either<S_1, S_2>

Состояние:
(state vector)

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Состояния. Работа над ошибками

```
isOK = Read(param); -- out
if (!isOK)
  return "error reading...!";
isOK = Calculate(
              param, value);
if (!isOK)
  return "error calc...!";
isOK = Save(value)
```

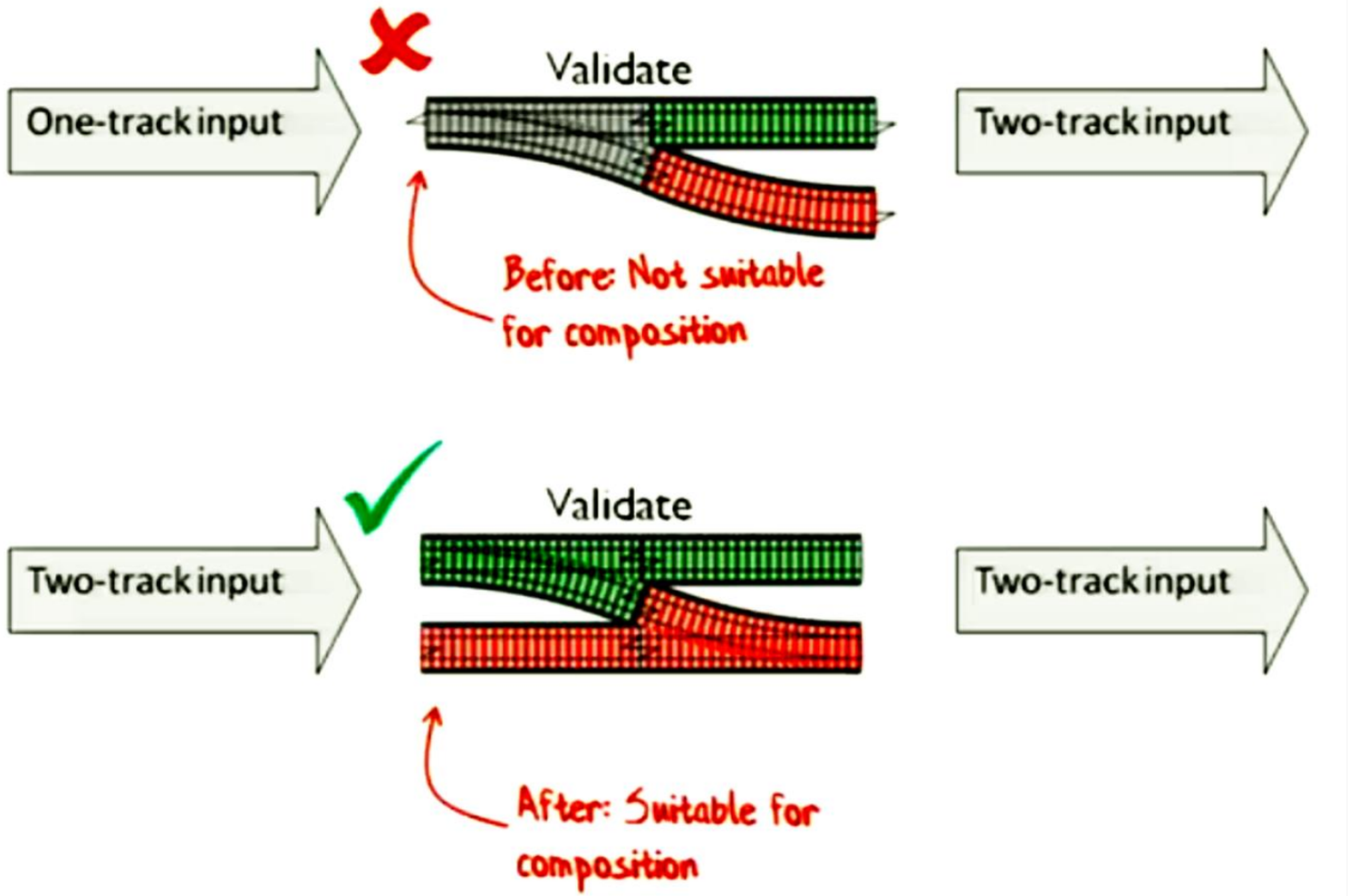
А хотели всего лишь:

- Read
- Calc
- Save

```
try
{
  param = Read();
  try
  {
    value = Calculate(param);
  }
  catch
  {
     return "error calc...!";
  }
}
catch
{
  return "error reading...!"
}
```

Either и «работа над ошибками»

- **Scott Wlaschin**: Railway-oriented programming
- Монада Either: убираем **if**'ы



fsharpforfunandprofit.com/posts/recipe-part2/

Railway C#

Marcus Denny:
Railway-Oriented
Programming in C#

```
return
   db.Read()
   .Bind(Validate)
   .Map(p => Calculate(p))
   .Bind(db.Save)
```

Result<TSuccess, TFailure>

https://youtu.be/uM906cqdFWE
https://github.com/habaneroofdoom/AltNetRop

Result. Реализация

```
public class Result<TSuccess, TFailure>
{
  public static Result<TSuccess, TFailure> Succeeded(TSuccess success)
  {
    if (success == null) throw new ArgumentNullException(nameof(success));
      return new Result<TSuccess, TFailure>
      {
        IsSuccessful = true,
        Success = success
      };
  }

  public static Result<TSuccess, TFailure> Failed(TFailure failure)
  {
    if (failure == null) throw new ArgumentNullException(nameof(failure));
    return new Result<TSuccess, TFailure>
    {
      IsSuccessful = false,
      Failure = failure
    };
  }

  public bool IsSuccess => IsSuccessful;
  public bool IsFailure => !IsSuccessful;
  public TSuccess Success { get; private set; }
  public TFailure Failure { get; private set; }
  private bool IsSuccessful { get; set; }
}
```

```
{
  public static Result<TSuccessNew, TFailure>
                      Map<TSuccess, TFailure, TSuccessNew>(
      this Result<TSuccess, TFailure> x,
      Func<TSuccess, TSuccessNew> f)
  {
    return x.IsSuccess
      ? Result<TSuccessNew, TFailure>.Succeeded(f(x.Success))
      : Result<TSuccessNew, TFailure>.Failed(x.Failure);
  }

  public static Result<TSuccessNew, TFailure>
                          Bind<TSuccess, TFailure, TSuccessNew>(
      this Result<TSuccess, TFailure> x,
      Func<TSuccess, Result<TSuccessNew, TFailure>> f)
  {
    return x.IsSuccess
      ? f(x.Success)
      : Result<TSuccessNew, TFailure>.Failed(x.Failure);
  }
}
```

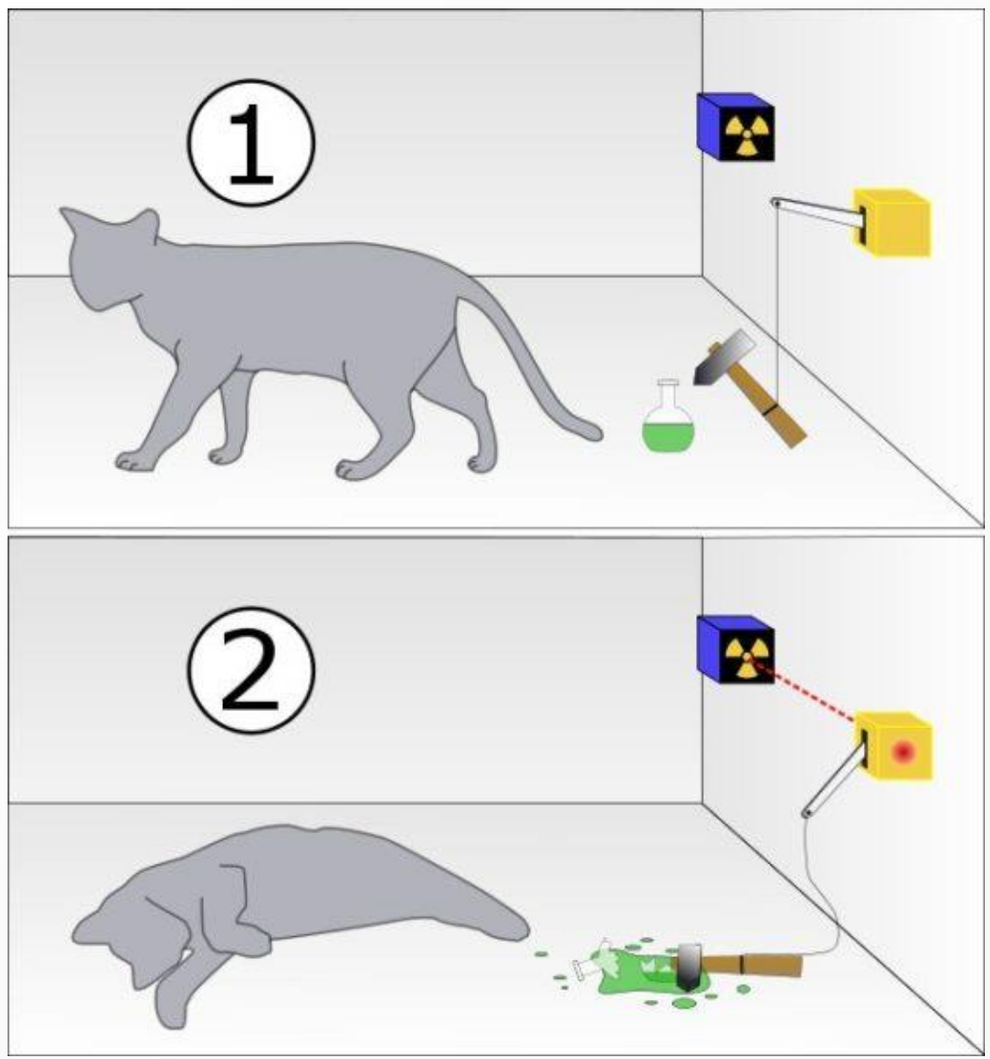
Числа (и не совсем). Анализ

- Замена сложного алгоритма и простых чисел на простой алгоритм и немного более сложные числа
- Разделение внутренней логики типов и «бизнес-правил»
- Измеримое/практическое значение требует «извлечения» из результата
- Замена «Числа → Состояния» убирает **if** и **try** с заменой линейным алгоритмом с монадами. При этом бизнес-логика отделяется от инфраструктурного кода.

Совсем не числа.

- Квантовое состояние
- Измеряемые величины — собственные значения линейных операторов
- Состояние неизвестно до момента измерения.

Динамика системы — изменение вероятности состояний



«Квантовая механика для IT'шников» ;-)

Все сложно?

$$\mathbf{H}s = \lambda s$$

Измеряемые величины — собственные значения линейных операторов, действующих на состояние.

Состояния неизвестны до момента измерения

Все просто!

$$Task<Either<S_1,S_2,...>>$$

- Значения хранятся внутри контейнеров
- Значение неизвестно до измерения
- Оператор извлечения значения - await

Как подружить бизнес-логику с async/await

Mark Seemann: Async injection

Перенос async/await на границы бизнес-домена

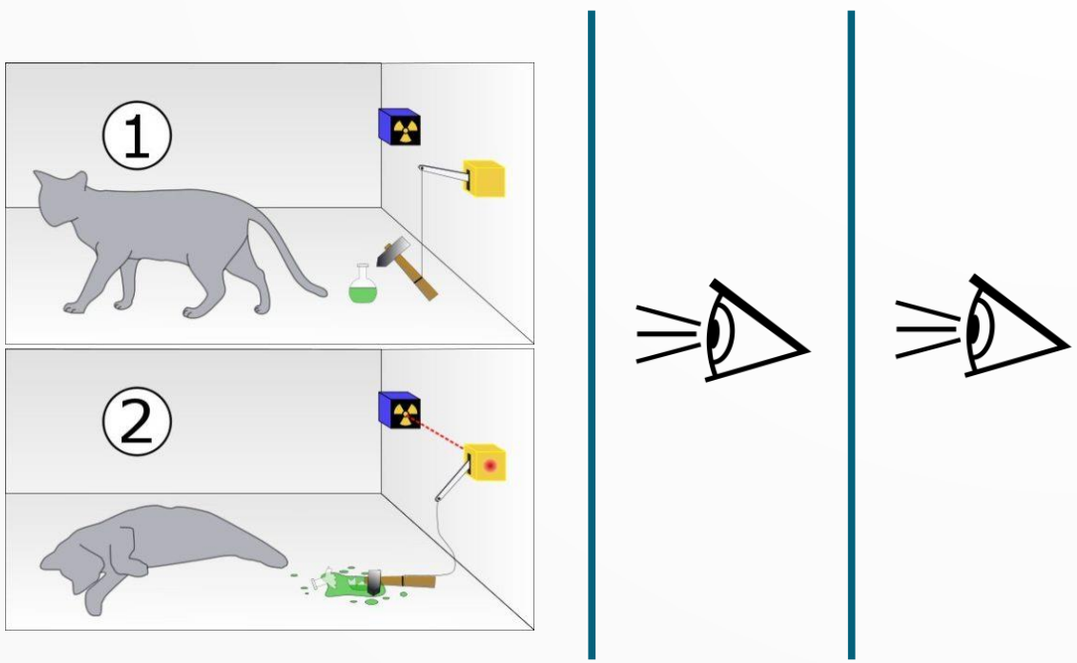
```
public async Task<IActionResult>
                    Post(Param param)
{
  int? id = await db.Find(param);
  if (id == null) return Err("..");
  string? val = await srv.Calc(id);
  return OK(val.Value);
}
```

```
public async Task<IActionResult>
                    Post(Param param)
{
  return await db.Find(param)
    .Select(x => srv.Calc(x))
    .SelectMany(x => …)
    .Match(Err(".."), Ok)
}
```

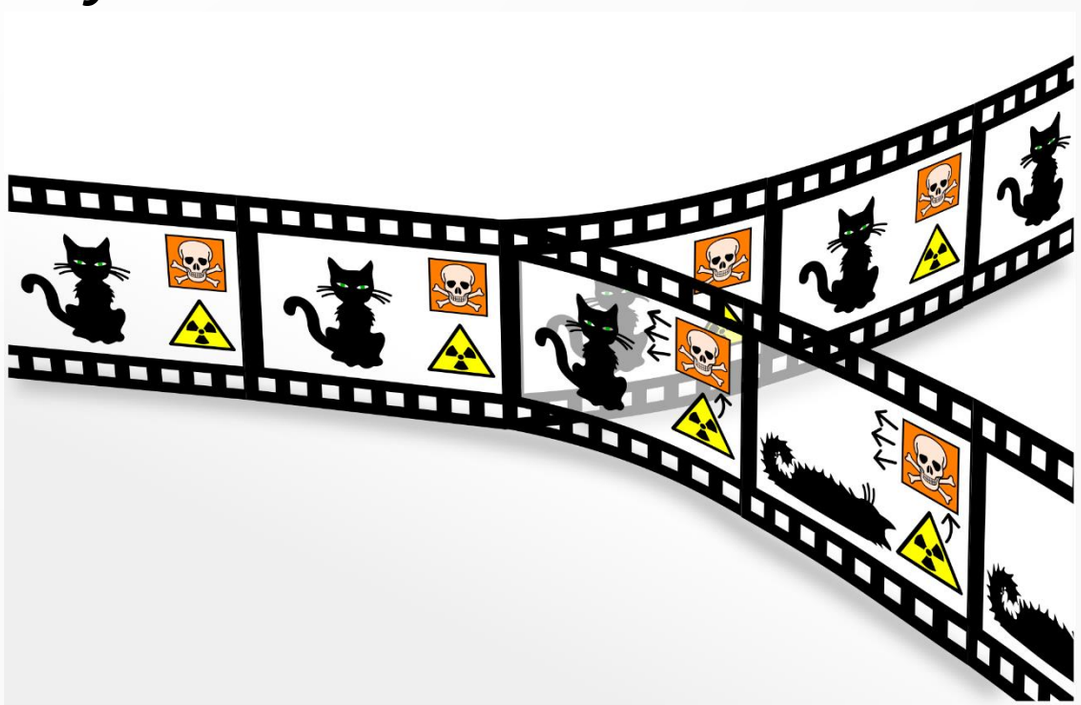
youtu.be/BsavoQWAVqM

blog.ploeh.dk

Кватновая механика. Интерпретации

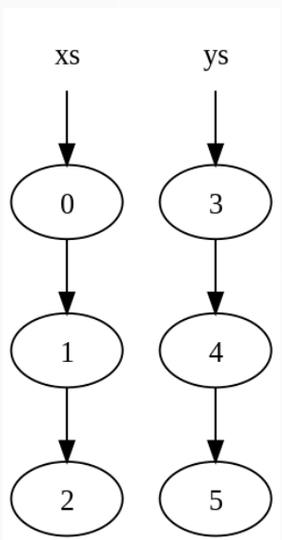


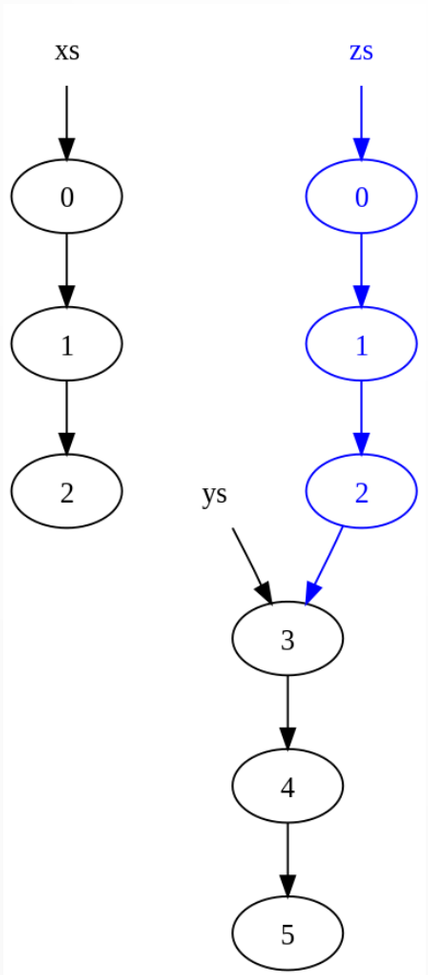
- «Коллапс» волновой функций?
- Мультивселенная.



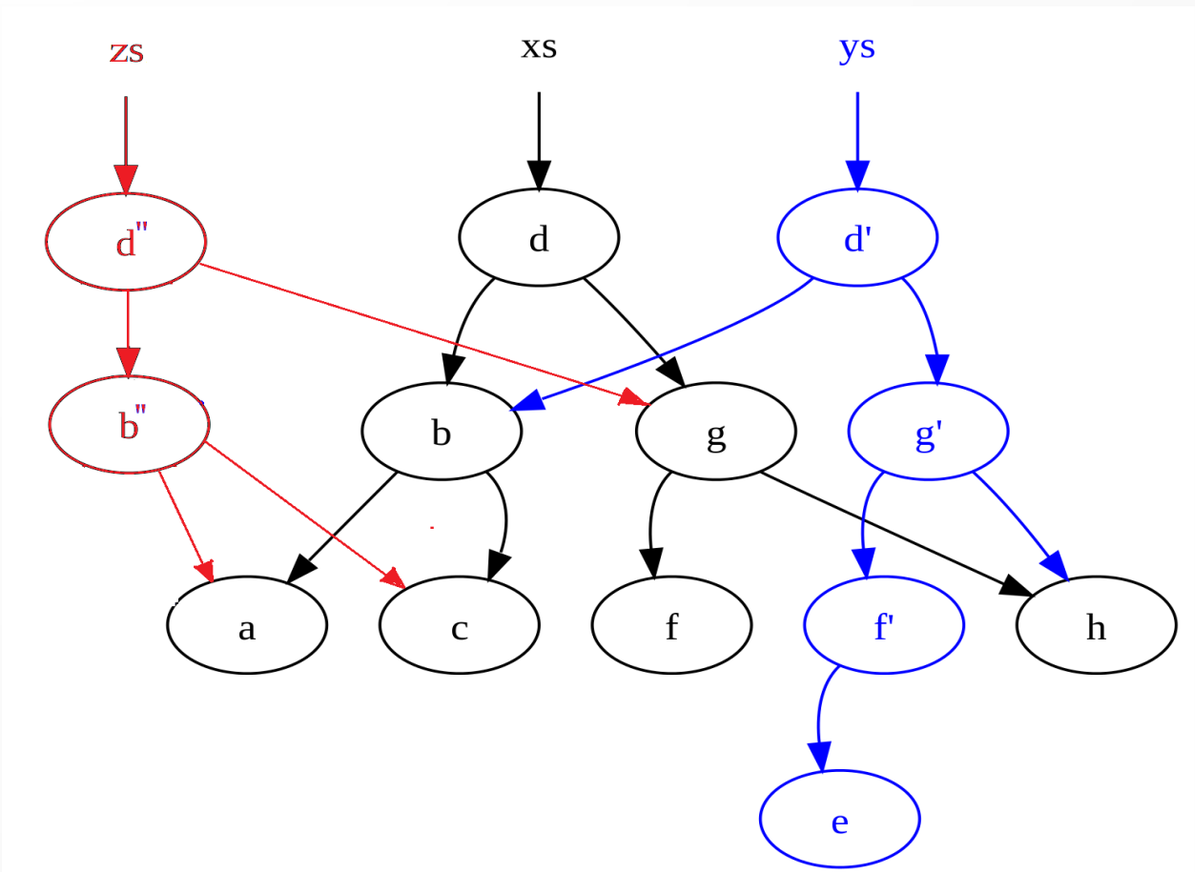
Каждое состояние вселенной неизменно

- Immutable





Много потоков — много вселенных



ACID или Saga?

- А если вселенная одна?

- На счете 100руб
- Снимаем два раза по 100
- ACID или Saga?

- Если правило относится к результатам действия — Saga — OK, если нет, то ACID

Числа

Вопросы?

Учет. Что учитываем

Что может делать человек:

- Перекладывать предметы с одного места на другое
- Разбирать предметы на части, в результате чего получаются другие предметы
- Собирать предметы вместе, создавая составные предметы

Учет. Задачи

- Знать, сколько каких предметов где находится
- Знать чьи это предметы
- Знать в каком они состоянии
- Знать как менялось количество предметов в прошлом для того, чтобы строить планы на будущее.
- Предметы → Ресурсы
- Основная модель: Процесс/Событие + Ресурсы

Учет. Регистры

Сразу пишем остатки

Предмет	Количество
Красное вино	20
Белое вино	30
Мука	90

- Точно знаем сколько чего осталось

- Путаница, ошибки
- Не знаем кто и когда поменял

Регистрируем изменения

Предмет	Приращение
Красное вино	+5
Красное вино	-1
Белое вино	+8
Мука	+45
Белое вино	-3
Мука	-10
…..	…..

- Очень долго считать, сколько осталось на сейчас

Учет. Викторина

Когда придумали CQRS и Event Sourcing?

Учет. История

- Изобрел* двойную запись
- Преподавал математику Леонардо да Винчи

Luca Pacioli, XV-XVI век

* На самом деле первым формально описал

Учет. Наши дни (почти)

- Закон Мура закончился?
- Скорость дисков не меняется 20 лет? Все БД — на Optane?
- Event Sourcing = Brute Force?
- Проблема банкоматов и ACID vs Saga.

IBM, 1956

Учет. Регистры учета

Регистр изменений

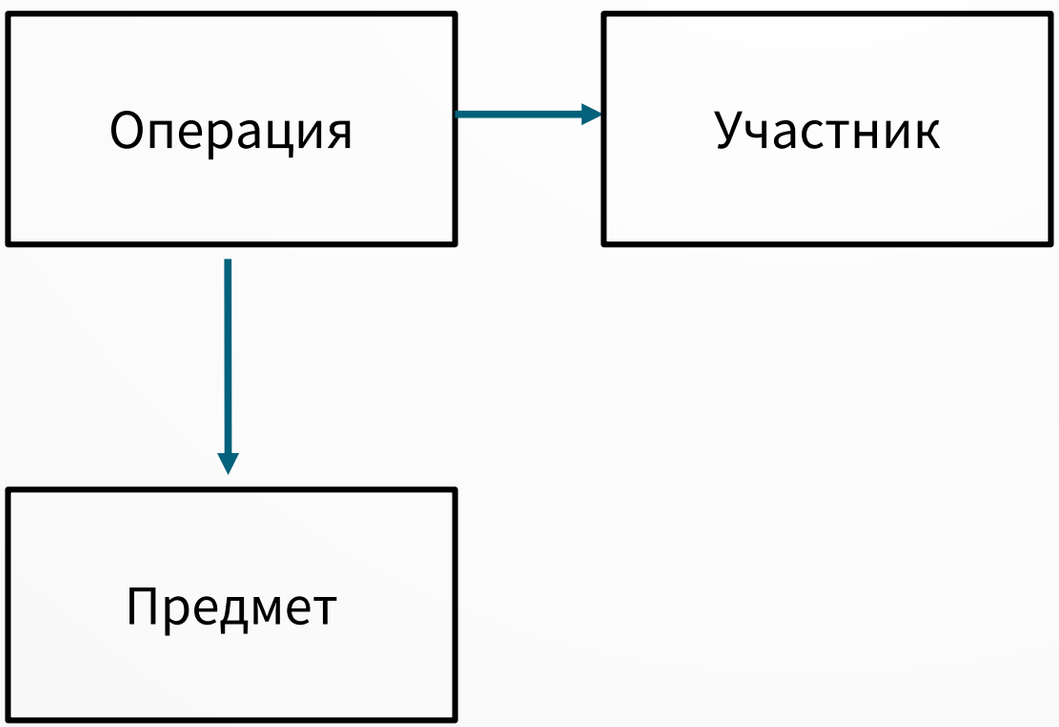
Что	Где	Состояние	Когда	На сколько

Регистр остатков

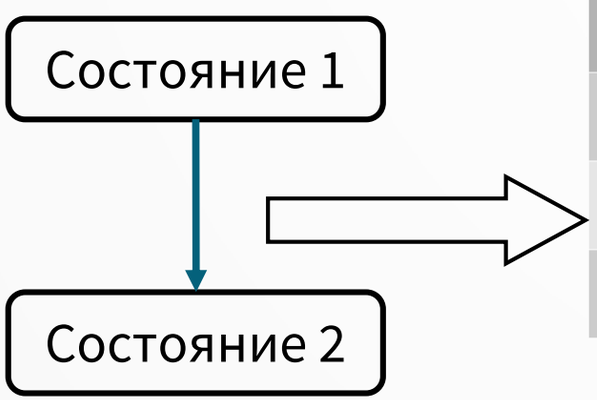
Что	Где	Состояние	Остаток

Учет. Операции

Операция — отражение этапа процесса

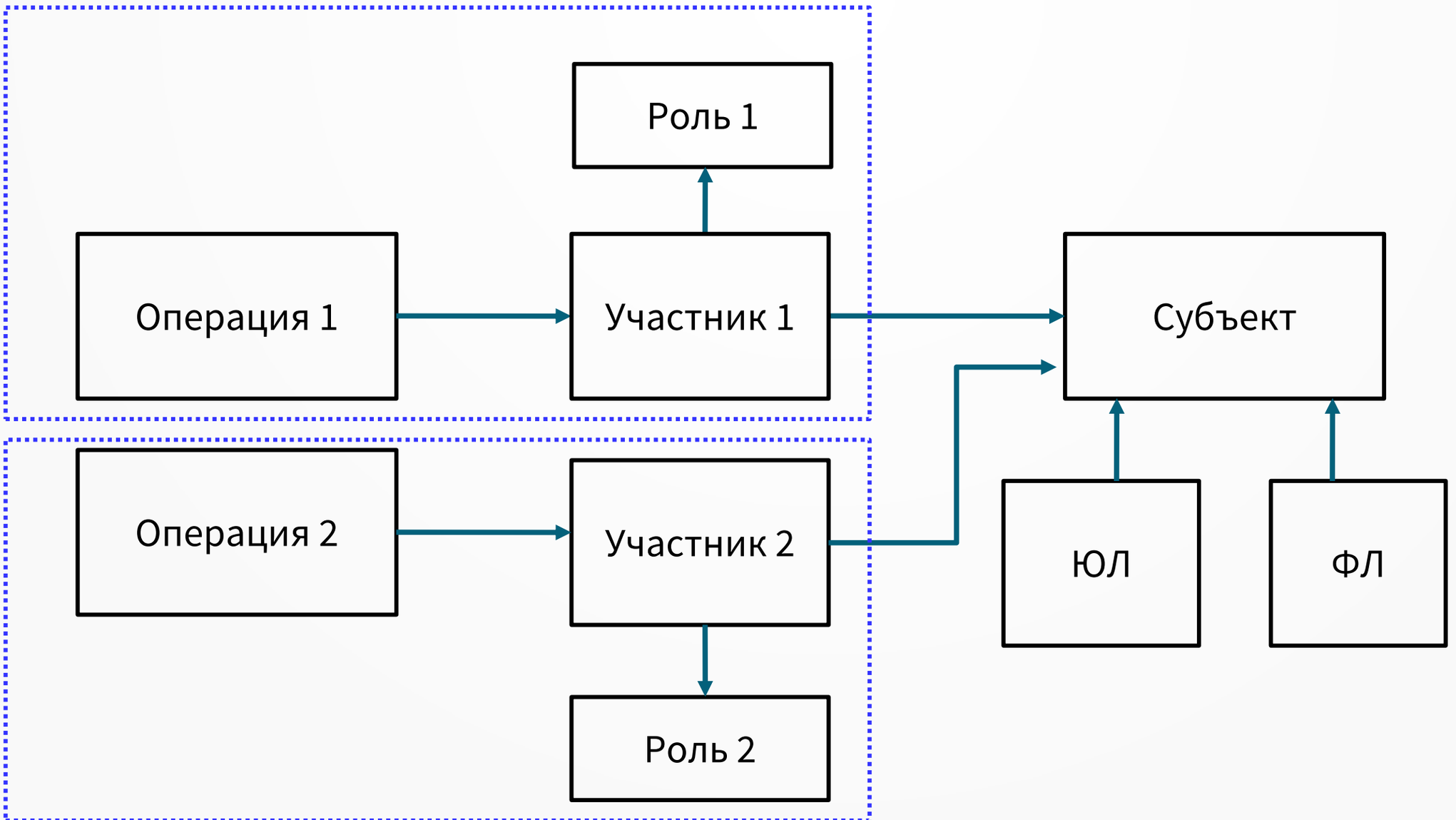


Учет. Операция → Регистр



Что	Где	Состояние	Когда	На сколько

Операция. Роли



Операция. Роли. Compliance

```
class Operation
{
  Party Sender;
  Party Receiver;
  Party ...

  Compliance compliance;

  bool CheckCompliance()
  {
    compliance.Check(Sender);
    compliance.Check(Receiver);

    ...
  }
}
```

```
class Compliance
{
  bool Check(Party party)
  {
    ...
  }
}
```

Compliance. Inversion of Control

```
class Participant
{
  Role role;
  Party party
}


class Operation
{
  Participant[] Parties;

  bool CheckCompliance()
  {
    compliance.Check(Parties);
  }

  Party Sender => Parties.
   Where(p => p.Role == Role.Sender);
}
```

```
class Compliance
{
  bool Check(Participant[] Parties)
  {
    for p in Parties
    {
       CheckParty(p);
    }
  }
}
```

Compliance. Visitor Pattern

```
class Party
{
  bool AcceptVisitor(Visitor v) =>
    v.Visit(this);
}


class Operation
{
  Party Sender;
  Party Seceiver;
  string Note;

  bool AcceptVisitor(Visitor v)
  {
    v.Visit(Note);
    Sender.AcceptVisitor(v);
    Receiver.AcceptVisitor(v);
  }
}
```

```
class Compliance: Visitor
{
  bool Visit(Party party)
  {
    return Check(p);
  }

  bool Visit(string note)
  {
    return DoSomething(note);
  }
}
```

Операции и учет. Резюме

- Изменения регистров учета происходит при движении операций по диаграмме состояний. Необходим баланс между оперативными данными для принятия решений и агрегированными для отчетов.
- В операциях участвуют роли, а не ЮЛ/ФЛ. Операции образуют агрегаты, где ЮЛ/ФЛ находятся за его пределами
- Следует избегать именованных свойств для хранения участников
- Учет не должен содержаться внутри операции, а быть внешним. Изменение правил учета не должно требовать изменения кода операции.

Вопросы?