# Подход к проектированию модульных многоагентных систем

Кудасов Николай Дмитриевич
Большакова Елена Игоревна

Кафедра алгоритмических языков

# Многоагентные системы (MAC)

- **Интеллектуальный агент** — это инкапсулированная вычислительная система, помещенная в некоторую среду и способная автономно действовать в этой среде для достижения поставленных целей.

- **Многоагентная система (MAC)** — система взаимодействующих интеллектуальных агентов.

# Приложения МАС

- Регулирование траффика
- Онлайн-торговля
- Моделирование социальных структур
- Реализация группового ИИ в играх и фильмах
- Устранение чрезвычайных ситуаций (ЧС)
- Сенсорные сети
- и т.д.

# Проектирование МАС

- **Внутренняя логика агента**
  - выбор логической модели
  - представление мира в модели
- **Механизмы координации агентов**
  - структурная организация МАС
  - динамическая координация
  - групповое принятие решений

# Средства разработки МАС: Фреймворк Jason

- Специальный язык программирования агентов
- Набор действий агента может быть расширен на Java
- Модель убеждения–желания–намерения (BDI model)
- Коммуникация агентов (сообщения)
- Встроенная поддержка организации МАС при помощи ролей, групп и миссий

**Нет средств для динамической координации и группового принятия решений.**

# Средства разработки MAC: Платформа DEFACTO

- Моделирование ЧС и систем реагирования
- 3D визуализация и участие человека
- Модель убеждения–планы, уровни автономности
- Координация через агентов-представителей
- Конфигурируемые командные стратегии для динамической координации

**Нельзя добавить новые стратегии или использовать существующие для других задач.**

**Прототип, реализация недоступна.**

# Средства разработки МАС: Библиотеки механизмов координации

- Существующие средства разработки МАС предоставляют только встроенные, ограниченные средства координации

- Механизмы координации сложны (ошибки в реализации легко допустить)

- На практике нужно проверять несколько различных механизмов координации

- Существует множество теоретических работ, предлагающих различные механизмы координации

**Нужны библиотечные реализации!**

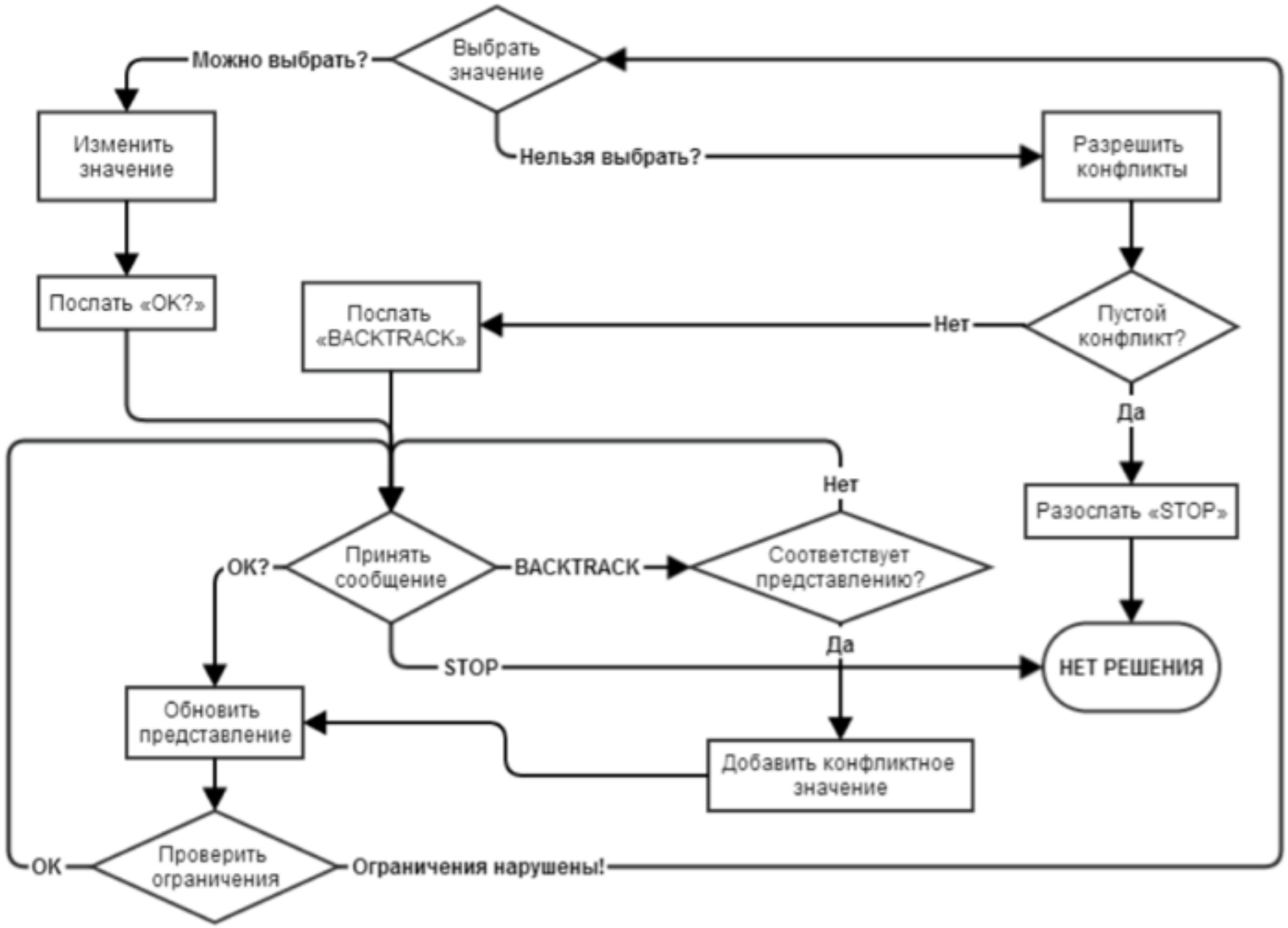
# Механизмы координации

- Для координации агентов предложено множество различных алгоритмов и паттернов проектирования
- Библиотечная реализация должна абстрагироваться от предметной области и, по возможности, от способа коммуникации агентов
- В качестве примера рассмотрим механизм координации группового принятия решений

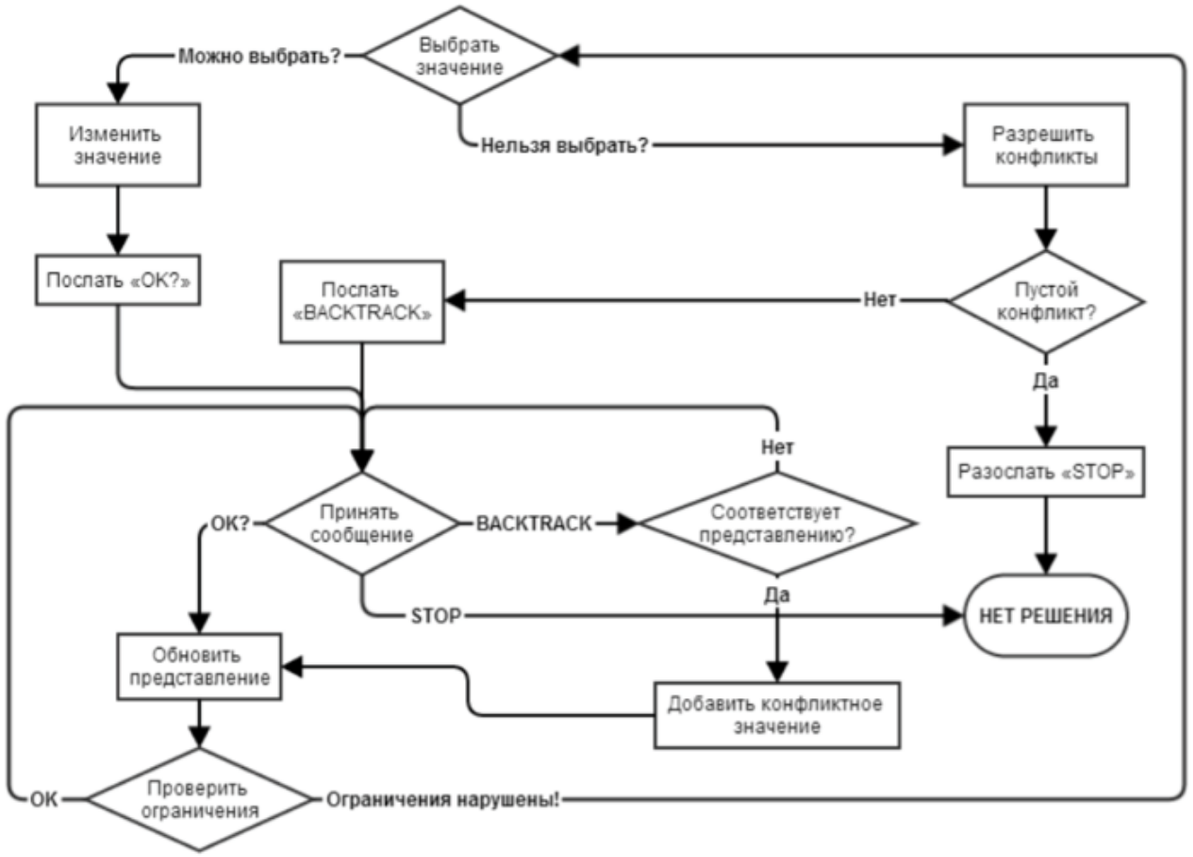
# Механизмы координации:
# Консенсус как задача с ограничениями

- Группа агентов хочет прийти к согласию (консенсусу)
- Каждый агент выбирает из **множества значений**
- Агенты могут иметь разные множества значений
- У каждого агента есть **ограничения**, которым общее решение должно удовлетворять
- Агентам необходимо координироваться для достижения согласия
- Одно из решений — асинхронный перебор (Asynchronous Backtracking)

# Механизмы координации: Асинхронный перебор



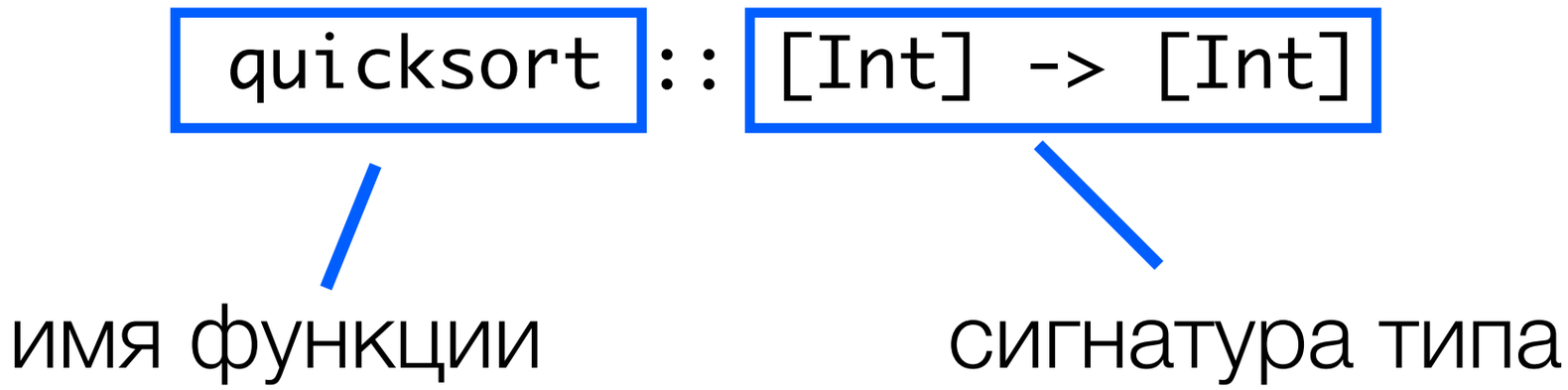
# Механизмы координации: Асинхронный перебор — реализация?




*Как реализовать асинхронный перебор в отрыве от предметной области?*

*Возможный подход — использовать полиморфные функции высшего порядка!*

# Полиморфные функции высшего порядка: Быстрая сортировка — сигнатура



# Полиморфные функции высшего порядка: Быстрая сортировка

---

```
quicksort :: [Int] -> [Int]
```

*Только для типа* `Int`*.*

*Используется стандартное сравнение чисел.*

# Полиморфные функции высшего порядка: Быстрая сортировка независимо от типа значений

```
quicksort :: [Int] -> [Int]
```

*Только для типа* `Int`*.*

*Используется стандартное сравнение чисел.*

```
quicksortBy :: (a -> a -> Ordering) -> [a] -> [a]
```

*Для любого типа* **a***.*

*Функция сравнения передаётся явно.*

# Механизмы координации: Асинхронный перебор — типы

```
abtKernel :: (Ord i, Eq v) => A i v (Maybe v)
```

**Тип идентификатора агента**

**Тип значений агентов (пространство поиска)**

Haskell

# Механизмы координации
# Асинхронный перебор — типы

```
abtKernel :: (Ord i, Eq v) => A i v (Maybe v)

type A i v a = forall m. Monad m =>
  StateT (AgentState i v) (Agent' (ABTKernelF i v) m) a
```

Состояние (для алгоритма)

Интерфейс приёма/передачи сообщений

Haskell

# Механизмы координации
# Асинхронный перебор — типы

```
abtKernel :: (Ord i, Eq v) => A i v (Maybe v)

type A i v a = forall m. Monad m =>
  StateT (AgentState i v) (Agent' (ABTKernelF i v) m) a

data AgentState i v = AgentState
  { agStop        :: Bool
  , agValue       :: Maybe v
  , agDomain      :: [v]
  , agId          :: i
  , agView        :: AgentView i v
  , agAbove       :: [i]
  , agBelow       :: [i]
  , agConstraints :: [Constraint i v]
  , agNoGoods     :: [NoGood i v]
  }
```

Ограничения агента

# Механизмы координации
# Асинхронный перебор — типы

```
abtKernel :: (Ord i, Eq v) => A i v (Maybe v)

type A i v a = forall m. Monad m =>
  StateT (AgentState i v) (Agent' (ABTKernelF i v) m) a

data AgentState i v = AgentState
  { agStop        :: Bool
  , agValue       :: Maybe v
  , agDomain      :: [v]
  , agId          :: i
  , agView        :: AgentView i v
  , agAbove       :: [i]
  , agBelow       :: [i]
  , agConstraints :: [Constraint i v]
  , agNoGoods     :: [NoGood i v]
  }

type Constraint i v = AgentView i v -> v -> Maybe (NoGood i v)

type AgentView i v = Map i v
```

Ограничения — это функции!

# Механизмы координации
# Асинхронный перебор

```
abtKernel :: (Ord i, Eq v) => A i v (Maybe v)
```

**Реализация алгоритма — это полиморфная функция высшего порядка!**

**Ограничения — это функции!**

```
type Constraint i v = AgentView i v -> v -> Maybe (NoGood i v)
```

Haskell

# Результаты

- Предложен подход к проектированию модульных MAC
- Спроектировано представление агентов и механизмов координации на языке программирования Haskell
- Реализована библиотека, демонстрирующая подход на примере алгоритмов группового принятия решения
- Реализация доступна по ссылке: https://github.com/fizruk/free-agent

# Направления дальнейшего исследования

- Библиотечная реализация механизмов динамической координации (контролируемая автономность, командные стратегии, социальные паттерны)

- Реализация концепции поведенческих паразитов

- Исследование фрактальных структур MAC и возможность их библиотечной реализации

# Спасибо за внимание!