Ещё один шаг в сторону суперкомпиляции императивных языков программирования

Сергей Кириченко

МГТУ им. Н. Э. Баумана

2 июля 2025 г.

Зачем студентам ИУ9 идти в магистратуру на ИУ7

- компиляторы — незачем
- ИИ:
  - ИИ
  - педпрактика
  - свобода в выборе темы дипломной работы

Методы оптимизации программ

- ручные или *автоматические*
- статические или *динамические*

Суперкомпиляторы

- SCP4 — Рефал
- JScp — Java
- Supero — Haskell
- SPSC, HOSC, ... — функциональные языки

Суперкомпиляция функциональных языков

- прогонка
- свёртка

Проблемы суперкомпиляции императивных языков

- состояние
- поток управления

Состояние

```haskell
newtype State s a = State
  { runState :: s -> (a, s)
  }

instance Functor (State s) where
  fmap f (State m) =
    State $ \s_1 ->
      let (x, s_2) = m s_1 in (f x, s_2)
```

Состояние

```
instance Applicative (State s) where
  pure x =
    State $ \s ->
      (x, s)

  (State m) <*> (State n) =
    State $ \s_1 ->
      let
        (f, s_2) = m s_1
        (x, s_3) = n s_2
      in
        (f x, s_3)
```

Состояние

```haskell
instance Monad (State s) where
  return =
    pure

  (State m) >>= n =
    State $ \s_1 ->
      let
        (x, s_2) = m s_1
        (y, s_3) = runState (n x) s_2
      in
        (y, s_3)
```

Состояние

```haskell
get :: State s s
get =
  State $ \s ->
    (s, s)

put :: s -> State s ()
put s =
  State $ \_ ->
    ((), s)
```

Поток управления

```haskell
newtype ContT r m a = ContT
  { runContT :: (a -> m r) -> m r
  }

instance Functor (ContT r m) where
  fmap f (ContT m) =
    ContT $ \c ->
      m (c . f)
```

Поток управления

```
instance Applicative (ContT r m) where
  pure x = ContT $ \c ->
    c x

  (ContT m) <*> (ContT n) =
    ContT $ \c ->
      m $ \f ->
        n $ \x ->
          c $ f x
```

Поток управления

```haskell
instance Monad (ContT r m) where
  return =
    pure

  (ContT m) >>= n =
    ContT $ \c ->
      m $ \x ->
        runContT (n x) c
```

Поток управления

```haskell
callCC
  :: ((a -> ContT m r b) -> ContT m r a) -> ContT m r a

callCC m =
  ContT $ \c ->
    runContT (m $ \x -> ContT $ \_ -> c x) c
```