

# 1. Конкретный синтаксис.

$\langle ident \rangle ::= \langle nondigit \rangle$   
 $\quad | \quad \langle nondigit \rangle \langle alphanum \rangle$   
 $\quad | \quad \langle nondigit \rangle \langle alphanum \rangle \langle ident \rangle$   
 $\langle binop \rangle ::= '+' | '-.' | '^' | '*' | '/' | '==' | '/=' | '<=' | '<' | '>' | '>=' | '|' | '&&'$   
 $\langle unop \rangle ::= '-' | '!'$   
 $\langle num-lit \rangle ::= \text{number}$   
 $\langle bool-lit \rangle ::= 'T' | 'F'$   
 $\langle lit \rangle ::= \langle num-lit \rangle | \langle bool-lit \rangle$   
 $\langle constr-name \rangle ::= \langle capital-alpha \rangle | \langle capital-alpha \rangle \langle ident \rangle$   
 $\langle pat \rangle ::= \langle var \rangle$   
 $\quad | \quad \langle lit \rangle$   
 $\quad | \quad \langle constr-name \rangle$   
 $\quad | \quad \langle constr-name \rangle \langle pats \rangle$   
 $\quad | \quad '(' \langle pat \rangle ')'$   
 $\langle pats \rangle ::= \langle pat \rangle | \langle pat \rangle \langle pats \rangle$   
 $\langle expr \rangle ::= \langle ident \rangle$   
 $\quad | \quad \langle lit \rangle$   
 $\quad | \quad \langle app \rangle$   
 $\quad | \quad \text{'if' } \langle expr \rangle \text{'then' } \langle expr \rangle \text{'else' } \langle expr \rangle$   
 $\quad | \quad \text{'let' var '=' } \langle expr \rangle \text{'in' } \langle expr \rangle$   
 $\quad | \quad \text{'case' '(' } \langle expr \rangle \text{' of' '{' } \langle case-body \rangle \text{'}'}$   
 $\quad | \quad \langle expr \rangle \langle binop \rangle \langle expr \rangle$   
 $\quad | \quad \langle unop \rangle \langle expr \rangle$   
 $\quad | \quad '(' \langle expr \rangle ')'$   
 $\langle app \rangle ::= \langle ident \rangle \text{' ' } \langle ident \rangle$   
 $\quad | \quad \langle ident \rangle \text{' ' } \langle lit \rangle$   
 $\quad | \quad \langle ident \rangle \text{' ' } '(' \langle expr \rangle ')'$   
 $\quad | \quad '(' \langle expr \rangle ') \langle ident \rangle$   
 $\quad | \quad '(' \langle expr \rangle ') \langle lit \rangle$   
 $\quad | \quad '(' \langle expr \rangle ') '(' \langle expr \rangle ')'$   
 $\langle case-entry \rangle ::= '|' \langle pat \rangle \text{'->' } \langle expr \rangle$   
 $\langle case-body \rangle ::= \langle case-entry \rangle | \langle case-entry \rangle \langle case-body \rangle$   
 $\langle bind \rangle ::= \langle ident \rangle (\langle arg \rangle | \varepsilon) \text{'=' } \langle expr \rangle$   
 $\langle arg \rangle ::= \langle ident \rangle | \langle ident \rangle \text{' ' } \langle arg \rangle$   
 $\langle data \rangle ::= \text{'data' } \langle constr-name \rangle \text{'=' } \langle data-body \rangle$   
 $\langle data-body \rangle ::= \langle data-entry \rangle | \langle data-entry \rangle \text{'|' } \langle data-body \rangle$   
 $\langle data-entry \rangle ::= \langle constr-name \rangle | \langle constr-name \rangle \langle pats \rangle$   
 $\langle type-def \rangle ::= \text{':' } ident \text{':' } \langle type \rangle$

$$\begin{aligned}
\langle type \rangle &::= 'Int' \\
&| 'Bool' \\
&| \langle type \rangle ' \rightarrow ' \langle type \rangle \\
\langle decl \rangle &::= \langle bind \rangle \mid \langle data \rangle \mid \langle type-def \rangle \\
\langle decls \rangle &::= \langle decl \rangle ';' \mid \langle decl \rangle ';' \langle decls \rangle
\end{aligned}$$

Введены две новые синтаксические категории: объявление типа функций (**type-def**) и объявление типов данных (**data**).

Каждой функции должно предшествовать объявление её типа.

Примеры:

1. Объявление типы данных.

```
data EitherBI = Left Bool | Right Int ;
```

2. Объявление типа функции.

```
:f: Int  $\rightarrow$  Int ;
```

Добавлена синтаксическая конструкция **case**.

```
:f: EitherBI  $\rightarrow$  Bool
f x = case (x) of {
  | Right b  $\rightarrow$  b
  | Left i   $\rightarrow$  T
};
```

И паттерн-матчинг.

```
:f: EitherBI  $\rightarrow$  Bool
f (Right b) = b;
f (Left i) = T;
```

Правила проверки типов тогда сводятся к проверке непротиворечивости объявления шаблона и его использовании.

Правила типизации шаблонов.

$$\frac{x \in \text{Int}}{\text{PatLit } x : \text{Int}}$$

$$\frac{x \in \text{Bool}}{\text{PatBool } x : \text{Bool}}$$

$$\overline{\Delta \vdash \text{PatVar } x : A}$$

$$\frac{\Delta D : \{A_i\}_{i \in 1 \dots n} \rightarrow A \quad \Delta \vdash \{a_i\}_{i \in 1 \dots n} : \{A_i\}_{i \in 1 \dots n}}{\Delta \text{PatConstr } D \{a_i\}_{i \in 1 \dots n} : A}$$

Тогда **case** будет типизироваться как

$$\frac{\Gamma, \Delta \vdash x : A \quad \Delta \vdash \{p_i\}_{i \in 1 \dots n} : A \quad \Gamma, \Delta \vdash \{e_i\}_{i \in 1 \dots n} : B}{\Gamma, \Delta \vdash \text{case } x \text{ of } \{p_i \rightarrow e_i\}_{i \in 1 \dots n} : B}$$

## 2. Абстрактный синтаксис

Представлен в виде АСД.

```

type Ast = [Decl]

data Decl = BindDecl Bind | DataDecl Data | TypeDecl Name Type

data Data = Data Name [Constr]

data Constr = Constr Name [Name]

data Type = Arrow Type Type | TInt | TBool | TData Name

data Bind = Bind Name [Pat] Expr

data Pat = PatVar Name | PatCtr Name [Pat] | PatLit Lit

data Expr = Var Name
           | Lit Lit
           | App Expr Expr
           | If Expr Expr Expr
           | Case Expr [(Pat, Expr)]
           | Let Name Expr Expr
           | UnOp UnOperator Expr
           | BinOp BinOperator Expr Expr

data Lit = ILit Integer | BLit Bool

```