

## 1 Nevek

- alfánumérikus, azaz betűk, számjegyek, perjelek és aláhúzás-jelek olyan sorozata, amely betűvel vagy perjellel kezdődik;
- szimbolikus, azaz az alábbi jelek tetszőleges, nem üres sorozata:

!	%	&	\$	#	+	-	/	:	<	=	>	?	@	\	~	‘	’	*
---	---	---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Névkent nem használhatók a következő, ún. fenntartott szavak vagy kultuszok:

```
abstype and andalso as case do datatype else end
exception fn fun handle if in infix infixr let local
nonfix of op open orelse raise rec sig signature
struct structure then type val with wthtype while
() [] {} , : >; ... - | => -> #
```

A neveket különféle osztályokba soroljuk:

var	értékváltozó	value variable	long
con	értékékonstruktör	value constructor	long
excon	kivételekonstruktör	exception constructor	long
tyvar	tipusváltozó	type variable	
tycon	tipuskonstruktör	type constructor	long
lab	nezonév	record label	
unitid	modulnev	unit identifier	

- A típusváltozó olyan alfanumerikus név, amely percjellel kezdődik, pl. ‘a.
- A mezőnév tetszőleges név lehet, vagy olyan pozitív egész, amely nem 0-val kezdődik.

- minden ‘long’ jelzésű X osztálynak van egy long-X pája. A long-X osztályba tartozo névek előre és hosszú (ún. minősített) alakban is felírhatók. A nöni! adak csak egy névből, a hosszú adak egy modulnevből, egy pontból és egy névből áll:

longx	::=	x	név	identifier
		unitid. x	minősített név	qualified identifier

## 2 Infix operátorok

- Egy név az infix vagy az infixr direktívával *infix* helyzetnek deklarállható. Ha az id név infix helyzetű, akkor az exp id exp2 kifejezés, sajátos esetén zárójelök között, minden olyan helyen használható, ahol az id(exp1, exp2) vagy

- az id(I-exp), 2-exp2 kifejezések egyébként használhatók. Infix helyzetű nevek mintában szintén használhatók.
- Egy minősített névet, vagy egy olyan nevet, amelyet az op szócska előt meg, sohasem lehet infix helyzetben alkalmazni. Az infix, infixr és nonfix direktívák szintaxisa a következő ( $n \geq 1$ ):

infix	$< \text{d} \triangleright$	$id_1 \dots id_n$	balra köt	left associative
infixr	$< \text{d} \triangleright$	$id_1 \dots id_n$	jobbra köt	right associative
nonfix		$id_1 \dots id_n$	nem köt	non-associative

A d decimalis számjegy opcionális, és a nevek precedenciáját adja meg; alapértelmezés szerinti értéle 0. Nagyobb szám magasabb precedenciát jelent.

Az infix, infixr és nonfix direktívák érvényeségi tartománya a szokásos, azaz a let kifejezésekben és a local deklarációkban deklárált nevek látáthatósgági szabályai változhatnak.

Azaz a let kifejezésekben a különféle jobbra kötő operátorok pedig jobbra könnel. Tilos viszont azonos precedenciájú, de különböző kötésű operátorokat használni ugyanabban a kifejezésben.

## 3 Általánosan használt jelölések a nyelvtanban

- minden szintaktikai osztályt változatok listájáként adunk meg, mégpedig soronként, egy változatot. Üres sor türes kifejezést jelent.
- A < és a > csúcsos zárójelpárok opcionális kifejezést fognak közre.
- The brackets and enclose optional phrases.
- Tetszőleges X szintaktikai osztályra (amelyre az x értelmezve van) az alábbiak szerint definiálunk az Xseq szintaktikai osztályt (amelyre az xseq értelmezve van):
- A kifejezésváltozatokat csökkenni precedenciájuk sorrendjében adjuk meg.
- Az L és az R betük a balra (L), ill. a jobbra (R) kötést jelzik.
- A típusok szintaxisa először kötő, mint a kifejezéseké.
- minden ismétlődő konstrukció (pl. a klasszorozat) a lehető legtöbb terjeszkedik jobbra. Ezért egy case-kifejezést, egy másik case- vagy egy fn-kifejezést, vagy egy fun-definíciót belül esetleg zároljuk közé kell rakni.

## 4 A Standard ML nyelvtana (Core language)

### 4.1 Kifejezések és klózsorozatok

$match$	::=	$match < \mid match >$	klózsorozat, vállozatsorozat	match
$match$	::=	$pat => exp$	klóz, vállozat	match rule

$exp ::= injexp$	$exp : ty$	típusmegközés (L)	type constraint (L)
	$exp1$ andalso $exp2$	szekvenciális konjunktív	sequential conjunction
	$exp1$ orelse $exp2$	szekvenciális alternáció	sequential disjunction
	$exp$ handle $match$	kivétel kezelése	handle exception
	raise $exp$	kivétel jelzése	handle exception
	$\text{if } exp1 \text{ then } exp2 \text{ else } exp3$	felteles kifejezés	raise exception
	while $exp1$ do $exp2$	iteráció	conditional expression
	case $exp$ of $match$	esetszérválasztás	iteration
	fn $match$	lambda-kifejezés	case analysis
			function expression

  

$injexp ::= appexp$	$atexp$	$injexp1 id injexp2$	infix alkalmazás	infix application

$appexp ::= atexp$	$appexp atexp$	(prefix) alkalmazás	application

$atexp ::= scon$	kilkönleges állandó	special constant	constant
	(Sec. 3)	(Sec. 3)	
$\langle op \rangle longvar$	értelekáltató	value variable	
$\langle op \rangle longcon$	adatkonsztruktor	value constructor	
$\langle op \rangle longexcon$	kivételekonsztruktor	exception constructor	
$\{ \langle exprow \rangle \}$	rekord	record	
# $lab$	rekordszelektor	record selector	
$\langle \rangle$	millas	0-tuple	
$(exp_1, \dots, exp_n)$	ennes	n-tuple, $n \geq 2$	
$[exp_1, \dots, exp_n]$	lista	list, $n \geq 0$	
# $[exp_1, \dots, exp_n]$	vektor	vector, $n \geq 0$	
$(exp_1; \dots; exp_n)$	kifejezősorozat	sequence, $n \geq 2$	
$\text{let } dec \text{ in } exp_1 \text{ ...; } exp_n \text{ end }$	lokális kifejezés	local expression, $n \geq 1$	
$(exp)$			

$exprow ::= lab = exp \langle , exprow \rangle$	kifejezősorozat	expression row

### 4.2 Declarációk és kötések

$dec ::= val ly \langle valbind \rangle$	értékdeklaráció	value declaration
fun $tynameq$ $finalbind$	függvénydeklaráció	function declaration
type $typbind$	típusdeklaráció	type declaration
datatype $datatypbind$ $< withtype$	datatype-deklaráció	datatype declaration
abstype $dathbind$ $< typbind$	abstype-deklaráció	abstype declaration
$>$	with	
$dec$ and $exception extbind$	kivételekdeklaráció	exception declaration
local $dec_1$ in $dec_2$	lokális deklaráció	local declaration
end		
open $unit_1 \dots u\_n$	open-deklaráció	open declaration
$n \geq 1$		
$dec_1 < ; > dec_2$	türes deklaráció	empty declaration
deklaráció-sorozat		
sequential declaration		
$infix < d > id_1 id_n$	infix-direktíva	infix (left) directive, $n \geq 1$
$infixr < d > id_1 id_n$	infixr-direktíva	infixr (right) directive, $n \geq 1$
nonfix $id_1 \dots id_n$	nonfix-direktíva	nonfix directive, $n \geq 1$

$tybind ::= tyvarseq \ tycon = ty < \text{and} \ tybind >$
$datbind ::= tyvarseq \ tycon = constbind < \text{and} \ datbind >$
$constbind ::= < op > con < of ty > < \text{and} constbind >$

$exbind ::= < op > econ < of ty > < \text{and} exbind >$
$< op > econ = op longcon < \text{and} exbind >$

Megjegyzés: *frallbind* fehér definíciójában, ha *var* infix helyzetének van deklarációra, akkor vagy meg kell elszínne az *op* szócskákat, vagy infix helyzetben kell használni. Ez azt jelenti, hogy a klozok elején *op var* (*atpat*, *atpat'*) helyett (*atpat var atpat'*) írható. A zárójelek elhagyhatók, ha *atpat*' után követetlenül :*ty* vagy = áll.

#### 4.3 Típuskifejezések

$ty ::= tyvar$	$\text{típusváltozó}$	$\text{típusváltozó}$	$\text{type variable}$
$\{ < tyrow > \}$	$\text{recordtipus}$	$\text{recordtipus}$	$\text{record type expression}$
$tyseq longtycon$	$\text{típuskonsztrukció}$	$\text{típuskonsztrukció}$	$\text{type construction}$
$ty_1 * \dots * ty_n$	$n\text{-es típus}$	$n\text{-es típus}$	$\text{tuple type}, n \geq 2$
$ty_1 > ty_2$	$\text{függvenytípus}$	$\text{függvenytípus}$	$\text{function type expression}$
$( ty )$			

$tyrow ::= lab : ty < , tyrow >$	$\text{típuskifejezés-sor}$	$\text{típuskifejezés-sor}$	$\text{type-expression row}$
----------------------------------	-----------------------------	-----------------------------	------------------------------

#### 4.4 Minták

$atpat ::= -$	$scon$	$\text{mindenesjel}$	$\text{wildcard}$
		$\text{különleges állandó}$	$\text{special constant}$ (Sec. 3)
	$< op > var$	$\text{változó}$	$\text{variable}$
	$< op > longcon$	$\text{konstruktör}$	$\text{constructor}$
	$< op > longecon$	$\text{kivételekonsztruktör}$	$\text{exception constructor}$
	$\{ < patrow > \}$	$\text{rekord}$	$\text{record}$
	$( pat_1, \dots, pat_n )$	$\text{millas}$	$\text{0-tuple}$
	$[pat_1, \dots, pat_n]$	$\text{enues}$	$n\text{-tuple}, n \geq 2$
	$\# [pat_1, \dots, pat_n]$	$\text{lista}$	$\text{list}, n \geq 0$
	$( pat )$	$\text{vektor}$	$\text{vector}, n \geq 0$

#### 4.5 Szintaktikai korlátozások

- Ugyanarra a névre (*var*) kétszer nem illeszthető minta. Ugyanarra a mezonévre (*lab*) kétszer nem illeszthető kifejezőssor, minthasor vagy típusssor.
- Ugyanaz a név nem köthető le két féléképpen egy *valbind*, *typbind*, *databind* vagy *ebind* deklarációban. A *databind* deklarációban ugyanez érvényes az adathalunkonstruktorra is.
- Ugyanaz a típusváltozó (*tyvar*) nem szerepelhet kétszer egy *tyvarseq* sorozatban valamely *typbind* vagy *databind* deklaráció bal oldali *tyvarseq* *tycon* részében. Minden olyan típusváltozónak (*tyvar*), amelyik előfordul a jobb oldalon, szerepelnie kell *tyvarseq*-ben.
- A *rec*-et követő minden *pat* = *exp* értékötéshben az *exp*-nek, szükség esetén zárójelk között, *fn* *match* alakúnak kell lennie, ahol egy vagy több névhez típusmegkötés is társítható.
- true, false, nil, :: és ref nem kapható tij értékkel egy *valbind*, *typbind* vagy *ebind* deklarációban. Az it név nem kaphat új értékét egy *databind* vagy *ebind* deklarációban.