Logic for Informatics (I)
(Declarative Programming)

read by Paul J. Voda
Institute of Informatics,
University of Bratislava Slovakia

The **first** of four courses:
- **Declarative programming** (Spring 2003)
- **First Order Logic** (Fall 2004)
- **Specification and Verification of Programs** (Spring 2005)
- **Computability for programmers** (Fall 2005)

We teach basics of **recursion theory** (computability theory), **first-order theories**, and of **verification of programs**.
The last means:
- definitions of programs in a 'programming language'
- formal proofs of their properties in Peano Arithmetic

Clausal Language

We use **CL** as a **tool** for teaching. It is a **programming language** and a **formal proof system**.

Designed and developed: in 1996-2002 by:
- Paul J. Voda (language processor)
- Jan Komara (theorem prover)
- Jan Kluka (web version into MathML)

Course Materials

The main **web** page:

www.ii.fmph.uniba.sk/cl/courses/lpi1/

All announcements and pointers to course materials will be posted on this page.

Text by **D. Guller**

Poznámky k prednáškam z CL

can be **downloaded** from the main web page.

Course requirements

In order to **pass** with grade **E** you have to get $\geq 50\%$ of marks.
(**A** ≥ 90, **B** ≥ 80, **C** ≥ 70, **D** ≥ 60)

There are two **tests** (midterm exams) each carrying 30 marks.
The **final exam** carries 40 marks.

All tests and exams are done in CL in the computer lab H6.

What can we do in CL?

-We can **define** functions over the **domain** of *natural numbers*:

$$\mathbb{N} = 0, 1, 2, 3, \ldots$$

which is **generated** from 0 by the **successor** function $S(x) = x + 1$. For instance:

$5 \equiv \overbrace{S \ldots S}^{5}(0) = 0{+}1{+}1{+}1{+}1{+}1 \equiv 0'''''$

Example of a definition:

$$Sum(0, y) = y \quad Sum(S(x), y) = SSum(x, y)$$

-We can **evaluate** such functions by **supplying** them with arguments in a **query**:
$Sum(4, 5) = y$

-We can **prove** properties of such functions in PA (Peano Arithmetic):

$$x > 0 \vee y > 0 \rightarrow Sum(x, y) > 0$$

Some functions built into CL

Addition $(+)$, **multiplication** $(\cdot)$.
modified subtraction:

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

integer division $(\div)$ and **remainder** (mod) operations satisfying:

$$x \div 0 = 0 \wedge x \bmod 0 = 0$$

$$\begin{aligned} y > 0 \rightarrow x \bmod y < y \wedge \\ x = (x \div y) \cdot y + (x \bmod y) \end{aligned}$$

Explicit definitions of functions
(presented in **term** form)

$f(x_1, \ldots, x_n) = s[x_1, \ldots, x_n]$ where s is a **term** (expression) referring to variables $\vec{x}$.

Terms are either:

- **simple** terms built from **variables** and **constants** by **applications** of functions
- **composed terms** built by **case** and **let** constructs:

$$\textbf{case } D_1 \to s_1 \mid \cdots \mid D_n \to s_n \textbf{ end}$$
$$\textbf{let } s_1 = v \textbf{ in } s_2$$

where D_1,... D_n are **discriminators** and s_1, s_2, ... s_n are terms.

Some discriminators (tests)

Zero discriminators:

$s = 0 \vee s > 0$

Equality discriminators

$s = t \vee s \neq t$

Dichotomy discriminators

$s \leq t \vee s > t$

Trichotomy discriminators

$s < t \vee s = t \vee s > t$

For each discriminator **exactly** one **alternative** holds.

Explicit definitions of functions

(presented in **formula** form)

$f(x_1, \ldots, x_n) = z \leftarrow A[x_1, \ldots, x_n, z]$
where the **formula** A is
$s = z$ which correspond to a **simple** term
$D_1 \wedge B_1 \vee \cdots \vee D_n \wedge B_n$ which corresponds to a **case** term $s = v \wedge B$ which corresponds to a **let** term

Explicit definitions of functions

(presented in **clausal** form)

as a collection of **clauses**
$f(x_1, \ldots, x_n) = z \leftarrow B_1 \wedge \cdots \wedge B_n$
unfolded from a **formula** form.