TEORETICKÁ INFORMATIKA VE VZTAHU K PROGRAMÁTORSKÝM TECHNIKÁM

Hashim Habiballa

Přírodovědecká fakulta Ostravské Univerzity, katedra informatiky a počítačů
30.dubna 22, 701 03 Ostrava 1, Česká Republika
tel.: +420696160241, e-mail: habiballa@volny.cz, www: http://www.volny.cz/habiballa/

Abstrakt

Příspěvek rozebírá stav výuky teoretických disciplín informatiky a jejich významu pro přípravu informatiků. Jsou podány stručné výsledky průzkumu mezi studenty různých forem studia na Ostravské Univerzitě. Dále se zabývá přínosem praktických aplikací ve výuce založených na poznatcích výše zmíněných disciplín.

1. Úvod – vymezení a význam teoretických disciplín

Informatika jako mladá vědní disciplína si v praxi již dávno získala významné postavení a má významný podíl na životě společnosti. Z tohoto důvodu je velmi důležité, aby příprava odborníků v této oblasti dostala jasnou podobu a to nejen v terciárním vzdělávání, ale i na nižších stupních studia.

Od začátku devadesátých let došlo k výraznému posunu ve výuce informatiky na SŠ. Dnes se výrazněji akcentuje aplikační pojetí výuky (ovládání textových editorů, tabulkových procesorů, informační zdroje), což má jistě svůj smysl pro všechny středoškolské studenty. Nicméně určitý základ "skutečné" informatiky je v každém případě vhodný pro studenty všeobecně vzdělávacích a technických škol. Cílem by mělo být i u studentů, kteří nejsou primárně zaměřeni na informatiku či matematiku, dosažení vyššího stupně vzdělání v informatice, tak aby se stala plnohodnotným všeobecně vzdělávacím předmětem vedle tradičních oborů, jako je matematika, chemie, fyzika atd. K tomuto vyššímu stupni nepochybně patří schopnost algoritmicky myslet, programovat na úrovni strukturovaného přístupu, optimalizovat řešení problémů a obecně vidět za hardwarem a softwarem jistý "matematický základ" – za nejlépe vystihující budiž označen pojem "informatické myšlení".

Proto by se věda o vzdělávání – didaktika – měla zabývat intenzivně také informatikou a to nejen možnostmi dosažení lepšího povědomí o informatice mezi studenty SŠ, kteří uvažují o studiu informatiky na školách přírodovědného zaměření, ale také na úrovni výuky na VŠ. V tomto příspěvku se pak bude hlouběji diskutovat způsob integrace teoretických disciplín do této výuky.

K základním znalostem informatika by měla patřit vedle výborné komplexní znalosti algoritmizace a datových struktur také matematická logika s důrazem na formální metody, znalosti z oblasti teorie jazyků a automatů a teorie algoritmů (teoretická informatika). Taktéž hraniční disciplíny jako je matematická statistika, umělá inteligence a numerická matematika jsou pro profesní nadhled a schopnost řešit nestandardní problémy velmi užitečné. Problémem

většiny z nich je však jejich vysoká náročnost oproti předmětům zabývajících se informačními technologiemi. Ta spočívá v nutnosti používaní algebry a matematické analýzy a dále v nutnosti pochopit vztahy a jejich důkazy. To studenty svádí k memorování logicky spjatých pojmů, což je ovšem zbytečné, protože bez jejich pochopení je nebudou umět v praxi aplikovat. Význam těchto disciplín je tedy nutné hledat především v jejich aplikovatelnosti v praxi (zde se nabízí především hraniční disciplíny a teorie jazyků a automatů), ale také v jejich důležitosti pro již zmíněné dobré znalosti a především dovednosti v algoritmizaci. Cílem by pak mělo být vybudování návyků a postojů, které budou absolventi používat ve své praxi. Tyto návyky založené na dobře postavených teoretických základech je povedou k systematickému přístupu, kvalitnímu a logickému řešení problémů praxe, přičemž nepůjde o řešení provizorní, ale znovupoužitelné a dlouhodobé. A tento přístup využije s výhodou nejen výzkumný pracovník, programátor nebo architekt informačního systému, ale v podstatě kdokoliv bude řešit problémovou úlohu s pomocí exaktních věd.

2. Postoj studentů Ostravské Univerzity k teoretické informatice a matematice

Na Ostravské Univerzitě se disciplíny teoretické informatiky vyučují ve třech formách studia – prezenčním, distančním a rozšiřujícím. Všechny tyto typy studia mají svá specifika. U prezenčního studia v současné době jde většinou o studenty ve věku okolo dvaceti let se zájmem spíše o informační technologie. Prezenčně se studuje magisterský neučitelský obor Informační systémy, bakalářský obor Aplikovaná informatika, Učitelství pro ZŠ v přípustných kombinacích matematika – výpočetní technika a fyzika – výpočetní technika a Učitelství pro střední školy v kombinacích matematika – informatika a fyzika – informatika. Studenti distančního bakalářského studia Aplikovaná informatika jsou pak věkově značně diferencováni, od věku srovnatelného s prezenčním studiem až po studenty věku okolo čtyřiceti let. Toto studium je realizováno především formou komunikace e-mailem mezi studentem a vyučujícím předmětu v prezenčním studiu. Z toho vyplývá značná náročnost této formy studia a to je u nesnadno pochopitelných teoretických předmětů ještě složitější. Rozšiřující studium je určeno primárně stávajícím učitelům ZŠ a SŠ, kteří si chtějí rozšířit aprobaci o informatiku. Toto studium je obsahově méně náročné než distanční a je realizováno každý pátek v odpoledních hodinách během semestru. V této formě je naopak vysoká úspěšnost studentů. Jejich věkové rozvrstvení je pestré, od začínajících učitelů až po učitele před důchodovým věkem.

Aby bylo možné se dozvědět více o přístupu studentů k teoretickým disciplínám, byl realizován průzkum mezi studenty těchto kurzů spadajících do teoretické informatiky:

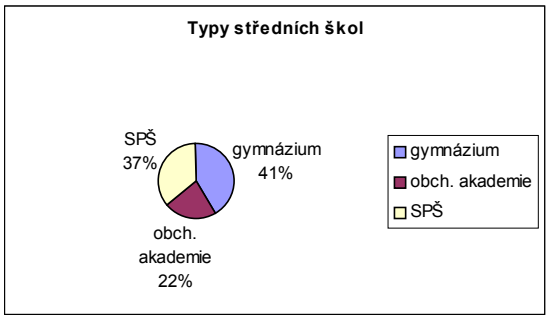
- prezenční kurz Regulární a bezkontextové jazyky I. (základy teorie formálních jazyků)
- prezenční kurz Vyčíslitelnost a složitost I. (základy teorie algoritmů)
- rozšiřující kurz Základy teoretické informatiky.

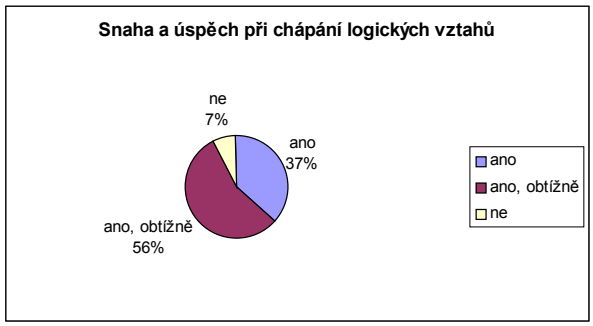
Průzkum byl realizován formou dotazníku, který sledoval kromě základních informací o osobě následující faktory ovlivňující kvalitu studijních výsledků studenta:

- typ vystudované střední školy
- motivaci pro studium informatiky
- preference jednotlivých kategorií disciplín (teoretická inf., inf. technologie, programování a matematika)
- vliv učitele a jeho vlastností na učení a preferované stránky vyučujícího
- jak se student připravuje; zda na základě logických souvislostí nebo mechanicky
- zda student preferuje spíše teoretické nebo praktické znalosti v předmětech

- speciální část pak vyhodnocuje pouze u kurzu Regulární a bezkontextové jazyky I., význam logických vztahů mezi jazyky, gramatikami a automaty a především preferovanou skladbu látky.

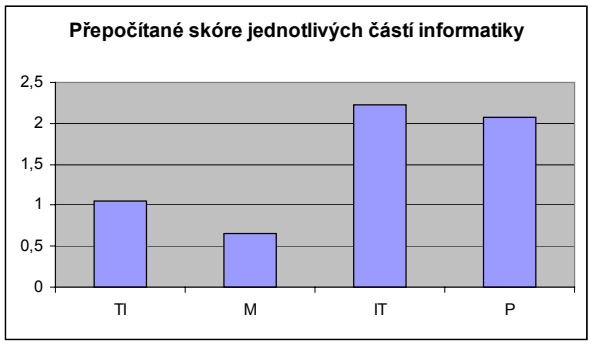
Příznivým faktorem v tomto vzorku je, že 78% studentů pochází z gymnázia nebo střední průmyslové školy (většinou elektrotechnického zaměření), kde je výuka algoritmizace, která je základem informatického vzdělání, velmi kvalitní. Následující graf ukazuje, za jak důležité považují pochopení logických vztahů uvnitř předmětu.

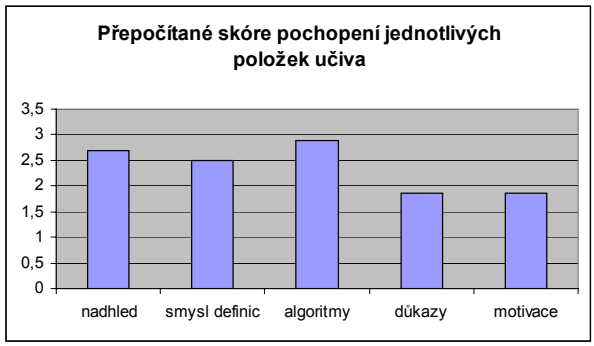




Vyplývá z něj opět pozitivní fakt, že studenti se ve většině snaží o logické chápání pojmů a vztahů, nicméně se jim to daří většinou pouze obtížně. Zároveň považují motivaci za důležitou, což by mělo podporovat snahu o zlepšení způsobu výuky. Většina studentů to ve svých obsáhlejších odpovědích velice oceňovala, pokud učitel klade důraz na souvislosti a dokáže je vysvětlit. Nejvíce jim vadí striktní formalizace typu definice-věta-důkaz, která jim bez vhledu do problematiky, navozuje situaci, kdy sklouznou k mechanickému učení. Tuto zkušenost mají především z matematiky.

Orientační průzkum také prokázal (alespoň v tomto omezeném vzorku), že studenti dávají preference především informačním technologiím (IT) a programování (P). Ze skóre je, ale patrné, že ani teoretická informatika (TI) resp. matematika (M) nedopadly v průměru tak špatně a zájem o ně je zhruba poloviční resp. čtvrtinový.





Ze speciální části dotazníku pak vyplynulo, že studenti vcelku nemají v teoretickém předmětu problém pochopit používané algoritmy (např. v teorii formálních jazyků) a poměrně dobře získávají nadhled nad učivem (tedy o jeho širších souvislostech). Na druhou stranu hůře chápou formální definice a nejhůře dopadají důkazy vět a motivace pro další studium. Zároveň pak frekventanti kurzu vyjadřovali své přání o zařazování většího množství praktických příkladů použití.

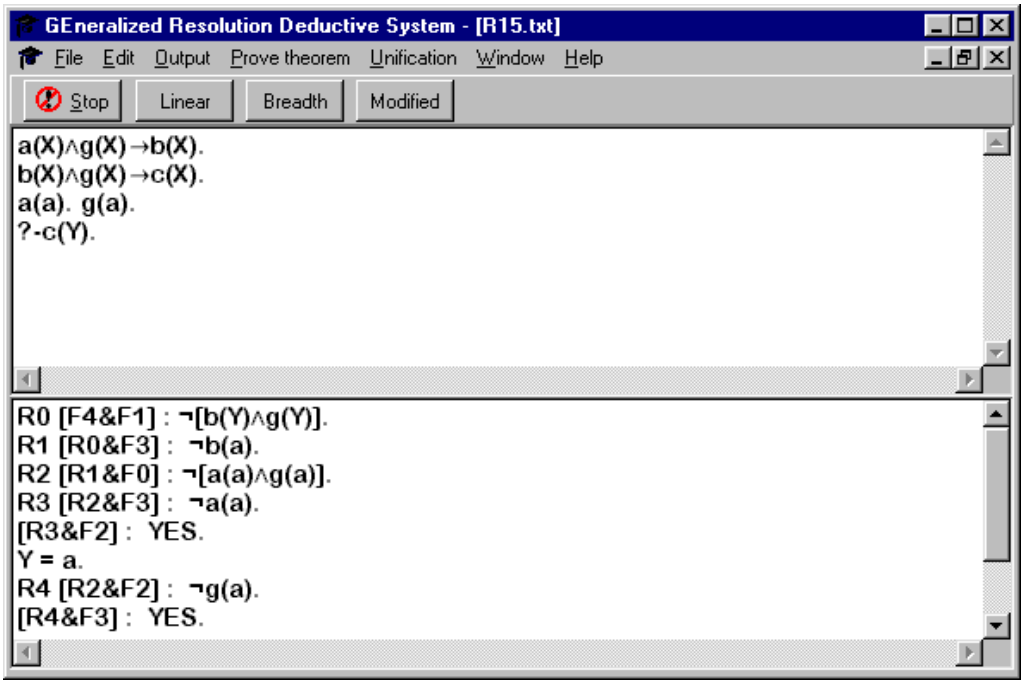
3. Způsoby zkvalitnění kurzů teoretické informatiky

Je nutné provést odklon od pouhé teoretické výuky a snažit se poskytnout také praktické aplikace. O to se bude snažit sbírka aplikací – počítačových programů, které budou studentům inspirací a zároveň výchozím bodem pro jejich vlastní tvůrčí činnost. S tím souvisí výuka způsobů implementace algoritmů např. syntaktických analyzátorů. V souvislosti s tímto cílem je třeba dosáhnout lepšího sepětí výuky algoritmizace a teoretické informatiky.

3.1 Logika

V oblasti matematické logiky je aplikační pojetí teorie zajištěno na OU v předmětu Prolog, kde studenti mají možnost prakticky vyzkoušet možnosti používání omezeného logického aparátu založeného na metodě rezolučního odvozování. Zároveň s tím je jim však také vysvětlena a předvedena možnost používání mnohem obecnějšího rezolučního aparátu – zobecněné rezoluce, s kterou pracuje autorova aplikace GERDS - (GEneralized Resolution Deductive System).

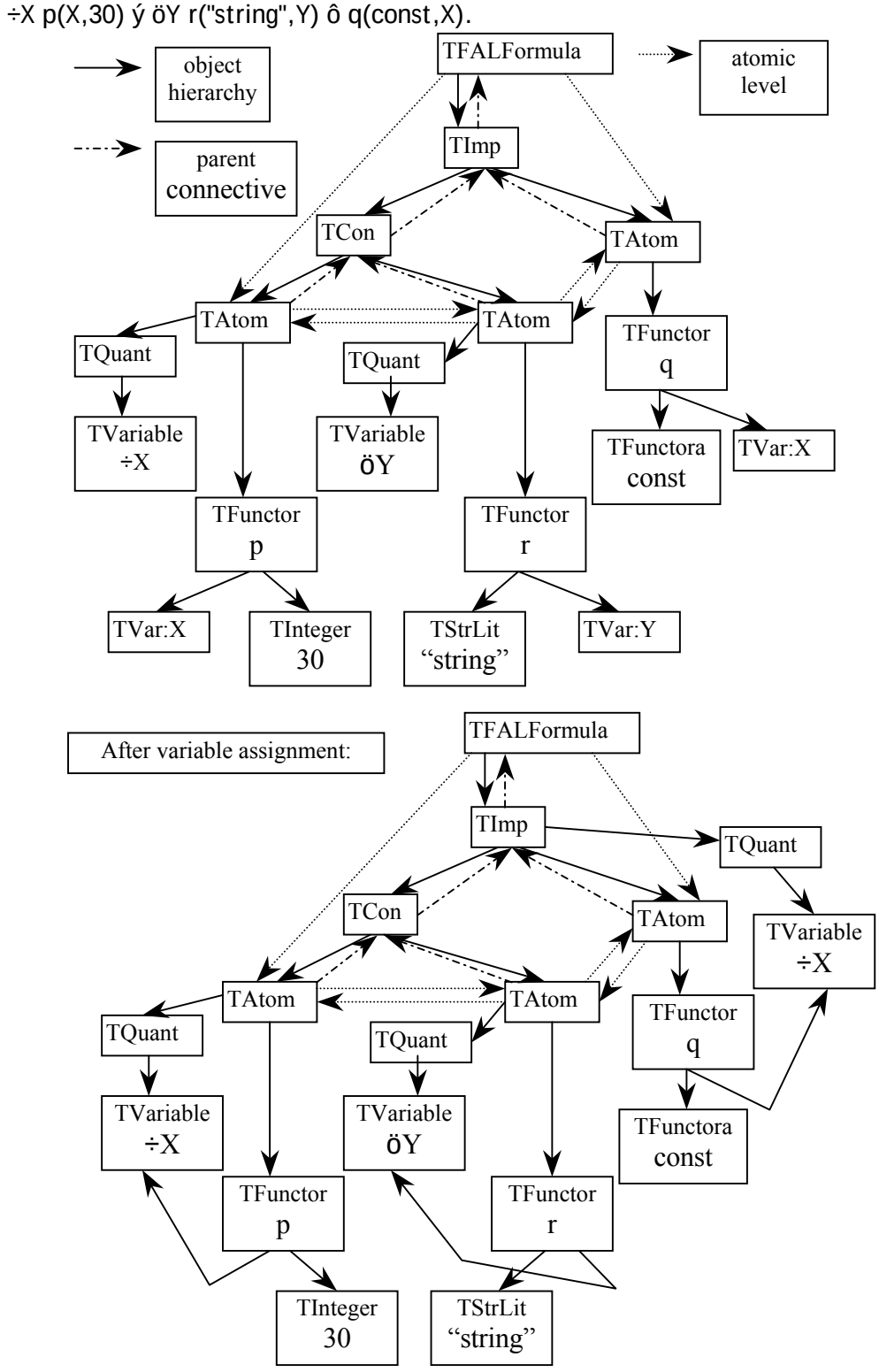
Tento aparát umožňuje provádět rezoluci na formulích predikátové logiky bez nutnosti jejich převodu do normálních forem. Dává tak nový pohled na klasické rezoluční pravidlo a pomáhá studentům pochopit principy axiomatických systémů, které si osvojili v předmětech zabývajících logickými základy umělé inteligence. Bližší informace o tomto systému i některých teoretických výsledcích lze najít v diplomové práci [1]. Následující obrázek ukazuje tuto aplikaci, která umožňuje vytvářet báze znalostí ve formě formulí predikátové logiky a následně zpracovávat dotazy na tuto bázi. Díky možnosti detailně zkoumat inferenční proces může student provádět analýzu postupu při zjišťování platnosti dotazu (při substituci možných termů). Také tato aplikace umožňuje porovnávat různé typy prohledávacích strategií a jejich výhody a nevýhody.



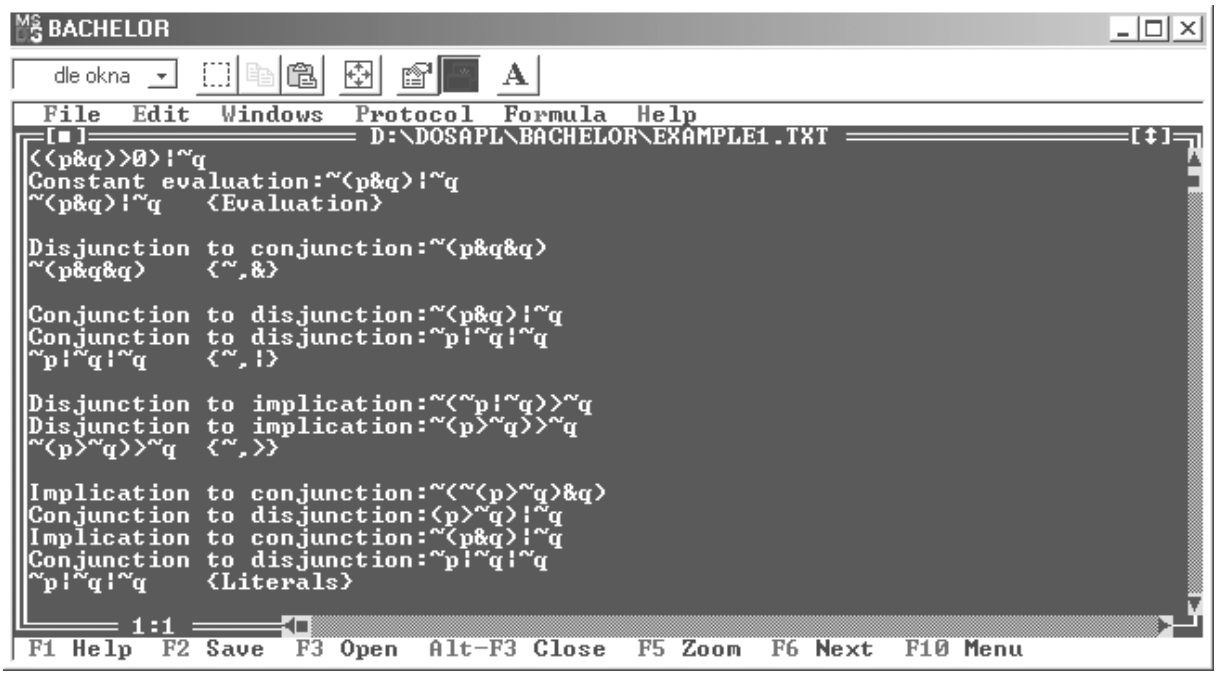
Důležité však je také to, že zároveň s tím mají studenti možnost procházet zdrojové kódy aplikace (vytvořené v prostředí Borland Delphi 2.0) a jsou jim popsány základní prvky programu:

- Uživatelské rozhraní
- Objektový model – dekompozice struktury formule do objektové hierarchie např. objekt TSub reprezentuje abstraktní koncepci podformule a TCon je jeho implementací reprezentující konjunkci dvou formulí predikátové logiky (PL)
- Kompilátor – syntaktický analyzátor jazyka PL a generátor vnitřní reprezentace ve formě prošitého stromu
- Postprocesing – úprava stromu formule navázáním proměnných na jejich kvantifikace a úprava stromu do „hyperstruktury“, která umožňuje snadnou manipulaci programátorovi při provádění inference
- Inferenční mechanismus – řízení nepřímého důkazu na zkompilovaných formulích
- Implementace pravidla zobecněné rezoluce – univerzální procedura pro vytváření rezolvent
- Konstrukce nejobecnějšího unifikátoru formulí
- Implementace původní procedury pro omezování množiny rezolvent založené na použití „selfresolution“ – detekce redundantních rezolvent

Na následujícím obrázku můžete vidět, příklad objektové hierarchie a vazeb formule.



Pro studenty základních kurzů logiky je k dispozici bakalářská práce autora – aplikace pro převody do normálních forem výrokové logiky a jejich minimalizaci. Tu mohou využít k ke kontrole řešení příkladů a zejména opět díky detailnímu rozpisu prováděných převodů. Aplikace je nazvána Bachelor a opět obsahuje zdrojové kódy, které lze využít pro demonstraci algoritmů pro formální úpravy ve výrokové logice.



Zdrojový kód ukazuje algoritmus převodu při použití pravidla distributivity formule výrokové logiky.

```
procedure TPLFormula.Distribution;
  { ( A & B ) | C = ( A | C ) & ( B | C )
    ( A | B ) & C = ( A & C ) | ( B & C ) }
    var pchar,pchar2:char;pompt1,pompt2,pompt3:PSTreeNode;
  begin
    if (p1<>nil) and (p2<>nil) and (p1^.character in ['&','|'] )
      and (p2^.character in ['&','|']) then
      begin
        RCopyTree(p2^.right,pompt1);
        pchar:=p1^.character;
        pchar2:=p2^.character;
        Changeoper(p2,pchar);Changeoper(p1,pchar2);pompt2:=p1^.right;
        p1^.right:=pompt1;new(pompt3);Changeoper(pompt3,pchar2);
        pompt3^.neg:=false;pompt3^.left:=pompt2;
        pompt3^.right:=p2^.right;p2^.right:=pompt3;
        if Protocol<>nil then
           with Protocol^ do
           if Format^.manner <= 2  then
             begin
               if Format^.manner <= 1 then
                   InsertString('Distribution:');
               XDecomp;
             end;
      end;
   end;
```

3.2 Teorie formálních jazyků a automatů

V teorii jazyků je důležité nejen zdůraznit studentům vztah mezi jazykem, gramatikou a automatem, ale zejména jim ukázat, jak jsou tyto znalosti potřebné při konstrukci znovupoužitelných a přehledných analyzátorů a překladačů. Pro tento účel se opět velice hodí aplikace GERDS. Ta implementuje již zmíněný překladač formulí do vnitřní reprezentace

metodou rekurzivního sestupu. Tento postup je na rozdíl od jiných velmi přehledný a umožňuje jednoduchou modifikaci zdrojového kódu, pokud je nutné jazyk (gramatiku) rozšířit o další syntaktické elementy. Bližší popis postupu je možné najít v knize [2]. Díky tomu je možné přiblížit účel výuky v této disciplíně. Z následujícího fragmentu definice gramatiky v BNF, lze pak jednoduše ukázat, jak sestrojit analyzátor jazyka pred. logiky:

<Formula> ::= <Imp> { ó <Imp> }
<Imp> ::= <Dis> { ô <Dis> }
<Dis> ::= <Con> { ü <Con> }
<Con> ::= <Subformula> { ý <Subformula> }
<Subformula> ::= ¬ <Subformula> | <Quantifier section> <Subformula> | '[' <Formula> ']' | <Predicate>
<Quantifier section> ::= <Quantifier character> <Variable> { , <Variable> } {<Quantifier character> <Variable> { , <Variable> } }
<Predicate> ::= <Predicate name> <List of parameters> | <Term> <InfixPred> <Term>
<Term> ::=
….
.
.

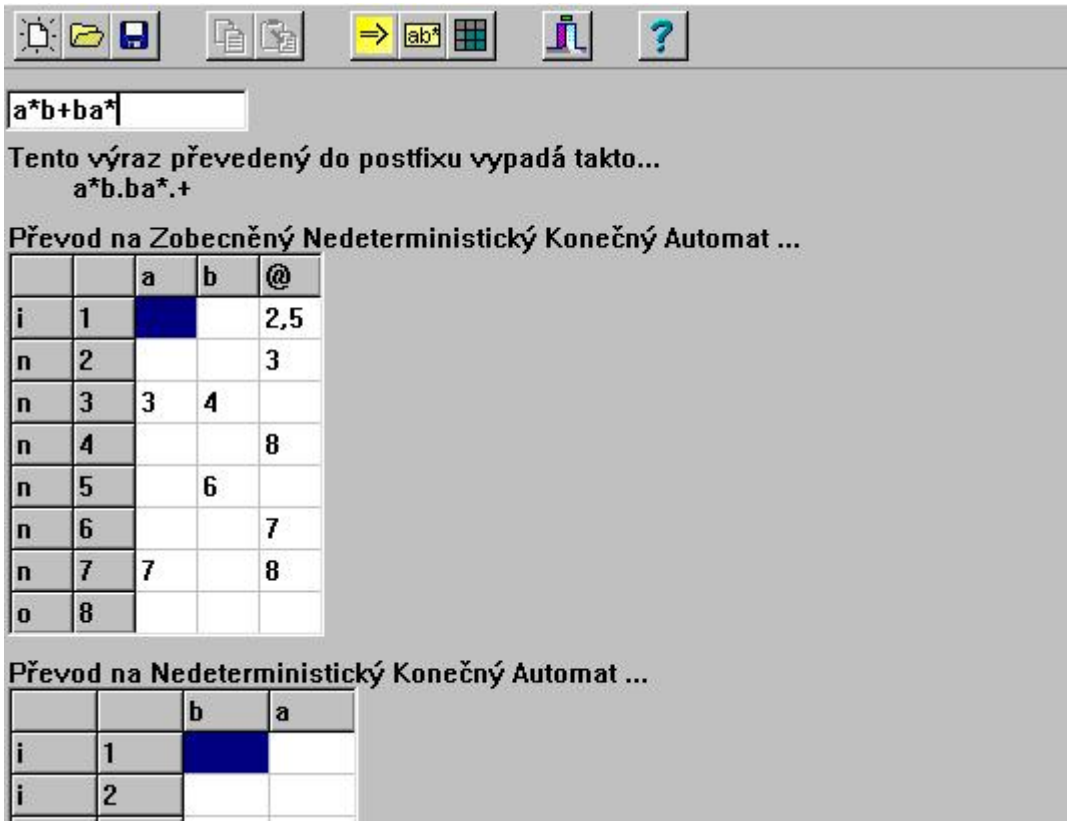
```
    procedure Eqv;
    begin
     if Error = OK then
       begin
         Imp;
         while (ch = ceqv)and(error = OK) do
           begin
            Getchar;
            Imp;
            if error = OK then AssignSub(TEqv.Create);
           end;
       end;
    end;
```

.
.

Procedura Eqv je částí analyzátoru odpovídající neterminálu <Formula> využívající jak lexikální analýzy (Getchar), tak procedury překladače (AssignSub), která konstruuje strom formule. Ukázkou simulace analýzy formule, která postupně prochází procedurami vytvořenými metodou rekurzivního sestupu se ukáže, jak probíhá vytvoření vnitřní reprezentace. Jelikož jazyk je v tomto případě poměrně jednoduchý než u programovacího jazyka, studenti dokáží na tomto analyzátoru pochopit princip jeho konstrukce.

Další užitečnou aplikací je PREGJAUT – bakalářská práce vytvořená na OU [3]. Tato aplikace umožňuje provádět a zobrazovat detaily převodu regulárních výrazů na automaty (nedeterministické, deterministické, podílové a normované). Tento program studenti využívají pro kontrolu samostatně počítaných příkladů a také pro sledování průběhu převodu jednotlivými mezikroky. K programu jsou rovněž přiloženy zdrojové kódy, které mohou studenti použít pro lepší pochopení probíraných algoritmů.

Uživatelské rozhraní PREGJAUTu vypadá následovně.



		a	b	@
i	1			2,5
n	2			3
n	3	3	4	
n	4			8
n	5		6	
n	6			7
n	7	7		8
o	8			

		b	a
i	1		
i	2		

4. Závěr

Diskutovaný aplikačněji pojatý způsob výuky teoretických předmětů je realizován v rámci studia informatiky na OU. Struktura kurzů v oblasti teorie jazyků bude rozšířena o samostatný kurz Překladače, který nabídne studentům možnost si právě teorii i samostatně vyzkoušet při konstrukci překladačů různými metodami. Při ní se bude uplatňovat koncepce, která tvoří linii tohoto článku a to je interakce s algoritmizací a programováním. Blíže se této koncepce dotýká také článek [4].

Literatura:

1. Habiballa, H. Problem Solving Through First-order Logic (theory and practice of non-clausal resolution). Graduační práce na : Ostravská Univerzita, PřF. 1999. 59 s.
2. Dvořák, S. Dekompozice a rekurzivní algoritmy. Praha:Grada, 1992, ISBN 80-85424-76-2
3. Tkačík, R. Přechod od regulárního výrazu ke konečným automatům. bakalářská práce, Ostravská Univerzita, Ostrava, 1997
4. Habiballa, H. Teoretická informatika na pomezí sekundárního a terciárního vzdělávání. In 1st International Conference. Aplimat 2002. 7 - 8. Únor, 2002. 1. vyd. Bratislava: SJF Slovenská technická univerzita, 2002. s. 193-198. ISBN 80-227-1654-5