JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PEDAGOGICKÁ FAKULTA

Katedra fyziky

Tvorba elektronického kurzu fyziky

Bakalářská práce

Ladislav Bartko

Vedoucí bakalářské práce

doc. RNDr. Josef Blažek, CSc.

České Budějovice

2010

Anotace

Předkládaná práce se zabývá problematikou publikování matematických vztahů na WWW. Hlavním tématem je znakový jazyk MathML 2.0, který umožňuje začlenit matematické vztahy a texty na web. V úvodu se věnuje jeho historii a požadavkům matematického značení v obecné rovině. Dále popisuje jeho části a pravidla fungování a na závěr uvádí způsoby implementace.

Praktickou částí práce je zpracování kurzu Teoretické mechaniky a vytvoření webových stránek.

Annotation

The presented work deals with the issue of publishing mathematical notation on the World Wide Web. It focuses on the Mathematical Markup Language MathML 2.0 that enables mathematical notation and texts to be served on the web. The introduction includes its history and requirements for mathematics Markup in general. Further it describes its parts and functioning principles. Finally, it presents implementation methods.

A practical part of the work is an elaboration of Theoretical mechanics course and making a web site.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, pouze s použitím uvedených zdrojů.

V Českých Budějovicích 30. 4. 2010 ……

Ladislav Bartko

Poděkování

Děkuji panu doc. RNDr. Josefu Blažkovi, CSc. za hodnotné rady a odborné vedení během mé práce. Dále bych chtěl poděkovat své rodině a přítelkyni za podporu při studiu.

Obsah

1 Úvod

MathML (Mathematical Markup Language) je aplikací jazyka XML, popisující matematické vztahy a texty. Cílem MathML je začlenit matematické vztahy do dokumentů na WWW (World Wide Web) a zpřístupnit tak matematiku na webu stejným způsobem, jakým HTML běžně zpřístupňuje text. Jedná se o matematickou normu vydanou W3C (World Wide Web Consortium), která vychází z jazyka XML.

Úvodní část práce pojednává o historii vzniku jazyka MathML, který je výsledkem snažení vědecké a odborné komunity o začlenění matematických vztahů na WWW. Jsou zde uvedeny základní požadavky na matematické značení a nejčastější nedostatky při zobrazení matematických vztahů na internetu. Jedna z kapitol nastiňuje princip fungování jazyka XML, který je přímým předchůdcem jazyka MathML.

Hlavní část práce je zaměřena na samotný jazyk MathML. Jsou zde vysvětleny jeho základní myšlenky a princip funkce, na kterém je založen. Dále jsou popsány hlavní rysy syntaxe MathML a typy značení s jednotlivými prvky, které se používají k popisu struktur matematických vztahů. Jednotlivé typy prvků jsou detailněji vysvětleny v samostatných kapitolách a jejich použití je demonstrováno na názorných příkladech.

Závěrečná část práce se zabývá otázkami implementace spojené s vytvářením a interpretací MathML. Jsou zde zmíněny možné způsoby vkládání vztahů MathML do jiných XML dokumentů a doporučené návrhy na jejich začlenění do jazyků, které jsou určeny k vytváření webových stránek (např. XHTML).

Praktickou částí této práce je zpracování kurzu Teoretické mechaniky, které je umístěno na přiloženém CD, a vytvoření webových stránek.

2 Úvod do MathML

MathML (Mathematical Markup Language) [1, 2, 3, 17] je aplikací jazyka XML popisující matematické výrazy a texty dodržující strukturu i obsah. Cílem MathML je začlenit matematické výrazy do dokumentů na WWW (World Wide Web) a zpřístupnit tak matematiku na webu stejným způsobem, jakým HTML běžně zpřístupňuje text. Jedná se o matematickou normu vydanou W3C (World Wide Web Consortium), jehož členové společně s veřejností vyvíjejí webové standardy pro WWW již od roku 1994.

2.1 Historie MathML

Problém kódování matematiky pro počítačové zpracování nebo elektronickou komunikaci je mnohem starší než samotný web. Běžnou praxí ve vědecké komunitě před nástupem webu bylo psát matematické výrazy v kódovací formě založené na znakové sadě ASCII (American Standard Code for Information Interchange). Před MathML existovalo několik znakových metod pro matematiku, zejména T_EX.

Od svého založení se web ukázal jako velmi efektivní způsob zpřístupňování informací vzdáleným skupinám osob. Nicméně i přesto, že WWW byl původně koncipován a realizován vědci pro vědce, možnosti zahrnout matematické výrazy v HTML byly velmi omezené. V současné době se většina matematických dokumentů na webu skládá z textu s obrázky vědeckého zápisu (v GIF nebo JPEG formátu), které jsou obtížné ke čtení, nebo celých dokumentů ve formátu PDF.

WWW konsorcium (W3C) uznalo, že nedostatek podpory pro vědeckou komunikaci je vážným problémem. Dave Raggett zahrnul návrh pro HTML Math do konceptu HTML 3.0 již v roce 1994. V květnu 1995 byl předložen návrh na začlenění matematiky do HTML týmu W3C a v březnu 1997 byla formálně ustanovena první pracovní skupina (W3C Math Working Group). MathML 1.0 byl vydán jako doporučení W3C v dubnu 1998 a verze 1.01 následovala v červenci 1999. V únoru 2001 se začalo pracovat na MathML 2.0 a finální podoba této verze se objevila v říjnu 2003. V červnu 2006 podala W3C návrh pro vývoj MathML 3.0, které je do dnešní doby pouze v podobě konceptu.

2.2 Omezení HTML

Požadavky na elektronickou komunikaci mezi vědeckou a odbornou komunitou jsou velmi vysoké. Výzkumníci, vědci, inženýři, pedagogové, studenti a technici, kteří pracují v rozdílných oblastech, zjišťují, že se stále častěji musí spoléhat na elektronickou komunikaci. V dnešní době převažuje „metoda obrázků" jako nejčastější způsob přenosu vědeckých zápisů, což není ideální. Kvalita dokumentu je špatná, vývoj obtížný a matematická informace obsažená v obrázku není zahrnuta do vyhledávání, indexování roboty nebo opakovaného použití v jiných aplikacích.

Nejviditelnější problémy s HTML pro matematickou komunikaci:

- *Problém se zobrazením*

Matematické výrazy v obrázcích se obecně hůře čtou, jsou méně kvalitní a pochopitelné než ostatní text v okně prohlížeče. Tyto problémy se navíc zhoršují při tisku dokumentu. Rozlišení obrázku s rovnicemi se pohybuje kolem 700dpi, zatímco okolní text je obvykle 300, 600dpi. Tento nezanedbatelný rozdíl v kvalitě je pro mnoho lidí neakceptovatelný.

- *Problém s kódováním*

Problém s kódováním se objevuje při prohledávání dokumentu pro části rovnice, při kopírování a vkládání do jiné aplikace, nebo dokonce při potřebě kopírování částí výrazů. Žádná z těchto běžných potřeb nemůže být správně vyřešena.

- *Načítání obrázků*

Dokumenty s matematickými výrazy, které jsou obsaženy v obrázcích publikovaných na webu, se déle načítají v prohlížečích, než by tomu bylo u prostého textu. Použitím kódování založeného na znacích se větší část zobrazovacích procesů přesunuje na PC uživatele. Barva pozadí rovnice v obrázku (většinou bílá) není zrovna ideální u pozadí, která bílá nejsou.

2.3 Požadavky na matematické značení

Při návrhu jakéhokoli znakového jazyka je nutné pečlivě zvážit potřeby potenciálních uživatelů. V případě MathML pokrývají potřeby potenciálních uživatelů široké spektrum, od vzdělávání až po výzkum a obchod.

Školní komunita je velká a významná skupina, která musí být schopna publikovat odborné materiály na webu. Studenti a pedagogové musí být schopni vytvořit matematický obsah rychle a snadno pomocí vhodných nástrojů.

Elektronické učebnice jsou dalším způsobem používání webu, který hraje ve vzdělávání velmi důležitou roli. Akademické i komerční sféry generují velké množství vědeckého materiálu.

Komerční vydavatelé jsou rovněž spojeni s matematikou na webu na všech úrovních, od elektronických verzí tištěných knih, interaktivních učebnic až po akademické časopisy. Vydavatelé vyžadují metodu vkládání matematiky na web, která je schopna vysoce kvalitního výstupu a která je navíc dostatečně silná pro širokou stupnici komerčního využití a pokud možno kompatibilní s předešlými metodami, zejména výrobními systémy založenými na SGML (Standard Generalized Markup Language), což je univerzální znakový metajazyk a přímý předchůdce jazyka XML.

2.4 Cíle MathML

Za účelem splnění různých potřeb vědecké komunity bylo MathML navrženo s následujícími hlavními cíly:

MathML by mělo:

- kódovat matematické dokumenty vhodné pro výuku a vědeckou komunikaci na všech úrovních
- kódovat jak matematické výrazy, tak matematické významy

- umožnit konverzi do jiných matematických formátů, jak prezentačních, tak významových; výstupní formáty by měli obsahovat: grafická znázornění, syntezátory řeči, vstup pro systémy počítačové algebry, další matematické typografické jazyky jako TeX, zobrazování jednoduchého textu, tiskárny a Braillovo písmo.

MathML může plnit všechny svoje cíle jako znakový jazyk a být užitečný pouze tehdy, jestliže bude dobře implementovaný. Za tímto účelem W3C Math Working Group vydala krátký seznam doplňujících cílů implementace, ve kterém se snaží přesně popsat minimální funkčnost interpretace MathML a softwaru, který to umožňuje:

- MathML výrazy na HTML (a XHTML) stránkách by se měly správně zobrazovat ve všech oblíbených webových prohlížečích v závislosti na čtenářových a autorových požadavcích a v nejvyšší možné kvalitě dané použité platformy (PC).
- Dokumenty HTML (a XHTML) obsahující MathML výrazy by se měly dát tisknout v nejlepším možném rozlišení.
- MathML výrazy na webových stránkách by měly být schopny reagovat na uživatelova gesta, například s myší, a koordinovat komunikaci s dalšími aplikacemi prostřednictvím prohlížeče.
- Vytvořené matematické editory a převodníky by měly usnadnit tvorbu webových aplikací, obsahující MathML výrazy.

Tyto cíle se řeší pomocí vestavěných prvků, jako jsou Java applety, plug-iny a ovládací prvky ActiveX. Do jaké míry jsou tyto cíle v konečném důsledku splněny, závisí na spolupráci vývojářů softwaru a podpoře prohlížečů. Matematická pracovní skupina W3C spolupracuje s vývojáři Document Object Model (DOM) a Extensible Style Language (XSL) s cílem uspokojit potřeby vědecké komunity i v budoucnu. MathML 2.0 vykazuje značný pokrok oproti situaci u MathML 1.0 (duben 1998).

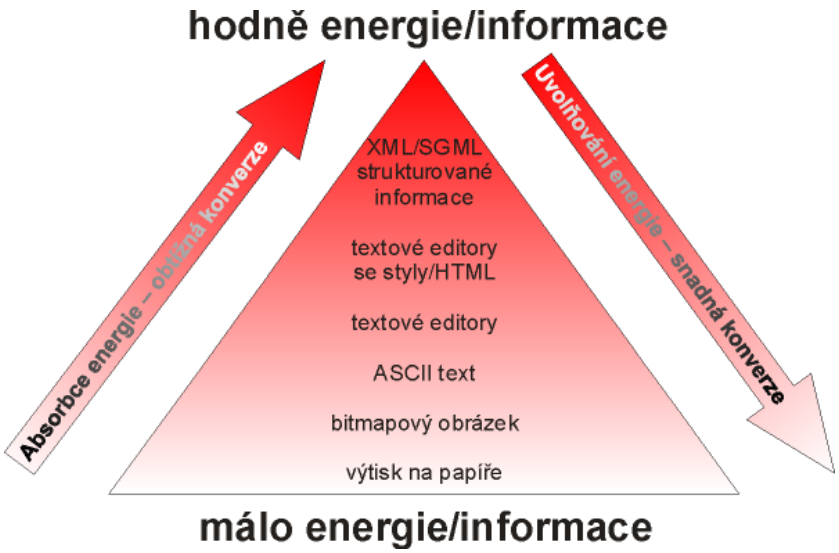
3 Základy XML

XML (eXtensible Markup Language) [4, 5, 6, 7, 8, 9] je univerzální rozšiřující znakový jazyk pro popis dat, který byl vyvinut a standardizován W3C konsorciem. XML se v dnešní době nachází všude kolem nás. Je to nejčastější nástroj pro přenos dat mezi všemi druhy aplikací a stává se čím dál populárnější v oblasti ukládání a popisu dat. XML vychází ze staršího jazyka SGML, jehož složitost bránila většímu rozšíření. V následujících odstavcích zmíním základní vlastnosti XML, které se staly jádrem matematického jazyka MathML.

3.1 XML – nástroj pro strukturování dat

Strukturovaná data zahrnují například tabulky, adresáře, konfigurační parametry, finanční transakce nebo technické výkresy. XML je souhrn pravidel pro tvorbu textových formátů, které umožňují strukturování dat. XML nepatří mezi programovací jazyky, tudíž k jeho zvládnutí není třeba znalostí programování. XML umožňuje počítači vytvářet, číst a zapisovat data, a tím zajistit jednoznačnost strukturovaných dat. Na rozdíl od běžných popisných jazyků je XML:

- rozšiřitelné
- nezávislé na platformě
- podporuje lokalizaci
- plně vyhovuje standardu UNICODE



Obr. 3.1: Význam XML

Zdroj: http://www.kosek.cz/clanky/xml/

Z obr. 3.1 je zřejmé, že soubory XML v sobě „nesou“ největší možné množství „energie“, samozřejmě ve formě strukturovaných dat. Jedná se tedy o velmi efektivní způsob zaznamenávání a přenosu informací. Této technologii je věnovaná velká pozornost, jelikož XML nenalézá uplatnění pouze ve sféře internetu, ale lze jej použít v mnohačetných aplikacích, kde je hlavní požadavek zaměřen na jednoduchost a efektivnost ve správě dat.

3.2 Historie vzniku XML

Vývoj XML začal již v roce 1996 a v lednu 1998 byl standardizován W3C. Technologie XML není příliš nová. Před XML již existovalo SGML, které bylo vyvíjeno na počátku 80. let a standardem ISO se stalo v roce 1986. SGML bylo široce používané pro rozsáhlé dokumentace. Vývoj HTML byl zahájen v roce 1990. Vývojáři XML převzali to nejlepší ze SGML a zkušenosti z HTML a vytvořili XML, které bylo stejně výkonné jako SGML, ale zároveň jednodušší na používání. A je třeba říci, že zatímco se SGML používá především pro technickou dokumentaci než na jiné druhy dat, u XML je tomu právě naopak.

3. 3 Podobnost XML v HTML

Stejně jako v HTML, i v XML se používají tvz. *tagy* (například `<p>...</p>`) a *atributy* (`název="hodnota"`). Zatímco HTML určuje, co který tag a atribut znamená, a jak bude text mezi nimi zobrazen v prohlížečích, XML používá tagy pouze k oddělení částí dat a ponechává interpretaci dat plně na dané aplikaci, která má za úkol data číst. Jinými slovy, pokud narazíte v souboru XML na `"<p>"`, nepředpokládejte, že se jedná o označení odstavce. V závislosti na kontextu to může být například parametr. Značky používané v HTML jsou předdefinované. HTML dokumenty mohou používat pouze tagy definované v normě HTML (jako `<p>`, `<h1>`, atd.). Jednotlivé tagy v XML souboru nejsou nijak definované. Uživatel si svoje tagy musí vytvořit sám.

Hlavní rozdíl mezi XML a HTML:

- XML bylo navrženo tak, aby přenášelo a vytvářelo data s důrazem na to, co obsahují
- HTML bylo vytvořeno tak, aby zobrazovalo data se zaměřením na vzhled a úpravu dokumentu

Je důležité si uvědomit, že XML není náhradou za HTML. Ve většině webových aplikací se XML používá zejména k přenášení strukturovaných dat, zatímco HTML se používá pouze k formátování a zobrazení dat.

3.4 Forma dokumentu XML

Programy, které vytvářejí tabulky, adresáře a jiné strukturované informace, ukládají tato data nejčastěji v binárním nebo textovém formátu. Textový formát má pro uživatele jednu nespornou výhodu oproti binárnímu, umožňuje uživateli prohlédnout si data obyčejným textovým editorem a je tak schopen jednotlivým datům porozumět. Textový formát ale nejvíce využívají právě vývojáři, pro které je tento formát velmi vhodný, zejména kvůli ladění programů. Stejně jako HTML i XML soubory jsou v textovém formátu, který může uživatel číst nebo podle své potřeby doplňovat.

Příklad XML dokumentu a jeho popis:

```
01. <?xml version="1.0" encoding="windows-1250"?>
02. <!DOCTYPE zprava SYSTEM "zprava.dtd">
03. <zprava>
04.   <od>Veronika</od>
05.   <pro>Klára</pro>
06.   <predmet>Připomínka</predmet>
07.   <telo>Zítra na tebe počkám po škole!</telo>
08.   <!-- Toto je komentář! -->
09. </zprava>
```

Na prvním řádku každého XML dokumentu se nachází XML deklarace, kde je uvedeno, o jakou verzi XML se jedná, a poté kódování znaků (v našem případě kódování češtiny).

Druhá řádka XML dokumentu nám určuje deklaraci typu dokumentu *(DTD - Document Type Definition)*. V našem případě, že je typ dokumentu uložen v souboru "zprava.dtd". DTD definuje tagy a atributy, které lze v daném typu dokumentu použít. V našem DTD je tedy nadefinováno, že zpráva se skládá z informace o odesílateli, adresátovi, z předmětu zprávy a těla zprávy.

Z příkladu je zřejmé, že každý XML dokument obsahuje tagový pár k definici rodičovského elementu (`<zprava> ... </zprava>`). Rodičovský element obsahuje potomky (např. `<predmet> ... </predmet>`) a ty pak mohou obsahovat další potomky atd.

Na rozdíl od HTML je XML, co se týče pravidel popisu dat, velmi striktní. Zapomenuté tagy nebo atributy bez uvozovek zneplatňují celý XML soubor, zatímco HTML je v tomto směru velmi tolerantní a text zobrazí i když se v kódu nacházejí chyby. Oficiální specifikace XML přímo zakazuje aplikacím „domýšlet si“, co tvůrce poškozeného XML souboru zamýšlel, a pokud objeví chybu, musí načítaní souboru zastavit a chybu ohlásit.

Jednotlivé příklady chyb v XML:

- Tagy v XML musí být uzavřené

Špatně

```
01. <zprava>ahoj
```

Správně

```
02. <zprava>ahoj</zprava>
```

Poznámka: Mezi párové tagy nepatří deklarace XML a deklarace typu dokumentu, tudíž nemusejí mít uzavírací tag.

- Tagy v XML rozlišují malá a velká písmena

Špatně

```
01. <Zprava>ahoj</ZPRAVA>
```

Správně

```
02. <zprava>ahoj</zprava>
```

- Tagy v XML musí být vhodně vložené

Špatně

```
01. <b><i>ahoj</b></i>
```

Správně

```
02. <b><i>ahoj</i></b>
```

- Atributy v XML musí být uzavřeny do uvozovek

Špatně

```
01. <zprava datum=10/02/2010>ahoj</zprava>
```

Správně

```
02. <zprava datum="10/02/2010">ahoj</zprava>
```

3.5 Rodina technologií XML

XML 1.0 je specifikace, která definuje, co jsou tagy a atributy. XML „rodina“ je rostoucí sada modulů, které nabízejí užitečné služby ke splnění důležitých a často požadovaných úkonů. *XLink (XML Linking Language)* popisuje standardní způsob, jak přidat hypertextové odkazy do souborů XML. *XPointer (XML Pointer Language)* slouží k odkazu na části dokumentů. Podobá se URL, jenom s tím rozdílem, že neodkazuje na dokumenty na webu, ale na části dat uvnitř XML souboru. *CSS (Cascading Style Sheets)*, známé jako kaskádové styly vzhledu dokumentu, se dají aplikovat na XML podobně jako na HTML. *XSL (eXtensible Stylesheet Language)* je pokročilý jazyk pro vytváření stylů. Je založen na *XSLT (XSL Transformations)*, což

je transformační jazyk, který plní úkoly přidávání, odebírání a úpravy tagů a atributů v XML souborech. *DOM (Document Object Model)* je standardní sada volání funkcí pro manipulaci s XML (a HTML) soubory v programovacích jazycích. *XML Schema 1* a *2* napomáhají vývojářům detailně definovat struktury u jejich vlastních formátů založených na XML. K dispozici jsou ještě jiné moduly a další se vyvíjejí.

3.6 Modularita XML

XML umožňuje definovat nový formát dokumentu tím, že kombinuje a opakovaně používá jiných formátů. Nastává tu ale problém, aby nedošlo k situaci, kdy by dva formáty, vyvíjené nezávisle na sobě, obsahovaly elementy či atributy pojmenované stejným jménem. Aby nedošlo k záměně, obsahuje XML tvz. *namespace* (mechanizmus jmenných prostorů). *XSL* a *RDF (Resource Description Framework)* jsou dobrými příklady formátů založených na XML, které používají jmenné prostory. *XML Schema* bylo navrženo tak, aby umožňovalo použití jmenných prostorů při návrhu struktury dat. Pak je snadné vytvářet nová schémata kombinací jiných a spojovat tak struktury dokumentů.

3.7 Podpora XML

XML je vhodným základem pro vytváření nejrůznějších aplikací. Podpora tohoto jazyka je rozsáhlá a obsahuje spoustu užitečných nástrojů, které uživateli usnadní jeho práci. XML má nespornou výhodu, že je k dispozici bez licence, uživatel si tak může vytvořit vlastní program bez toho, aby něco platil. XML má také velkou podporu ze strany vývojářů.

4 Pravidla MathML

Tato kapitola představuje základní myšlenky MathML [10, 17, 18]. V první části se budu věnovat rozdělení MathML prvků. Ve druhé části nastíním princip funkce MathML a v závěrečné části popíši hlavní rysy syntaxe MathML, které se vztahují na všechny tagy MathML.

4.1 Třídění MathML prvků

Všechny MathML prvky spadají do jedné ze tří kategorií (jednotlivé kategorie budou popsány v dalších kapitolách):

- *Prezentační prvky*

Tyto prvky popisují strukturu matematického zápisu. Typickým příkladem je prvek `"mrow"`, který se používá k označení horizontální řady znaků, a `"msup"` prvek, který se používá k označení základu a horního indexu. Obecným pravidlem je, že každý prezentační element odpovídá pouze jednomu druhu notačního schématu, jako je zlomek, horní a dolní index apod.

- *Významové prvky*

Významové prvky popisují přímo matematické objekty a ne zápis, který je zastupuje. Typickým příkladem může být prvek `"plus"`, což je operátor pro reálná čísla, a prvek `"vector"` pro označení vektoru z lineární algebry.

- *Prvky uživatelského rozhraní*

Používají se k začlenění a zobrazení dokumentů MathML v jiných aplikacích, zejména webových aplikací (např. HTML). Typickým příkladem může být prvek `"math"`, který uvozuje celý matematický zápis.

4.2 Princip MathML

Pokud se podíváme na jakýkoliv matematický výraz, brzy si všimneme, že ačkoli existuje mnoho matematických symbolů, existuje jen několik způsobů, jak je uspořádat – řádky, horní a dolní indexy, zlomky, matice apod. Tyto notační vzory nebo schémata se často objevují vnořené do sebe, příkladem nám může být odmocnina ze zlomku. Důležité je, že i složité matematické výrazy, které obsahují vnořené prvky, jsou sestaveny z několika jednoduchých schémat. K pochopení poslouží následující příklad:

$$(a+b)^2$$

Tento výraz se přirozeně dělí na „základ“ $(a+b)$ a na „exponent“, v tomto případě znak "2". Základ se poté ještě dělí na sekvenci dvou znaků a tří symbolů, čímž proces rozkladu končí.

Kódování této rovnice pomocí prezentační formy:

```
01.  <msup>
02.      <mfenced>
03.          <mi> a </mi>
04.          <mo> + </mo>
05.          <mi> b </mi>
06.      </mfenced>
07.      <mn> 2 </mn>
08.  </msup>
```

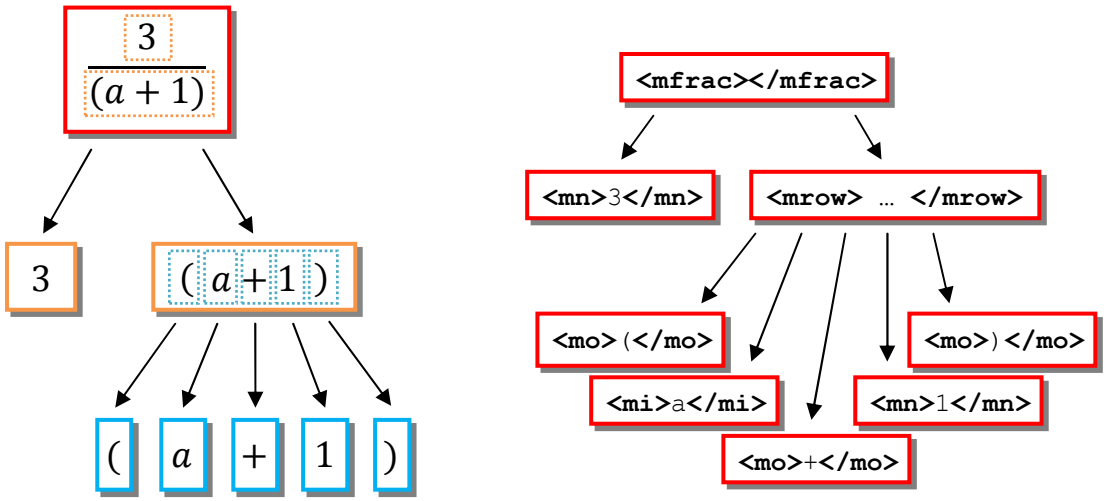
Párový tag `"msup"` má za úkol spojit „základ“ $(a+b)$ s horním indexem "2", proto celý výraz ohraničuje. Prvním podvýrazem je `"mfenced"`, který svůj obsah ohraničuje závorkami. Druhý podvýraz `"mn"` označuje, že se jedná o číslo. Podobně podvýrazy obsažené v tagu `"mfenced"` jsou opatřeny označením, kde `"mi"` určuje, že se jedná o identifikátor a `"mo"` určuje operátor.

Kódování této rovnice pomocí významové formy:

```
01.  <apply>
02.     <power/>
03.     <apply>
04.        <plus/>
05.        <ci> a </ci>
06.        <ci> b </ci>
07.     </apply>
08.     <cn> 2 </cn>
09.  </apply>
```

Nejdůležitějším tagem ve významové formě je párový tag `"apply"`, po něm musí následovat operátor, který je aplikován na ostatní prvky těla. V našem případě `<power/>` (má podobnou funkci jako párový tag `"msup"` v prezentační formě), a operátor `<plus/>`. Identifikátor je značen `"ci"` a číslo `"cn"`.

Stromová struktura prezentačního kódování:



Poznámka: Jednotlivé tagy prezentační a významové formy budou vysvětleny v následujících kapitolách.

4.3 Syntaxe MathML

Vzhledem k tomu, že MathML je aplikací jazyka XML, přebírá MathML pravidla syntaxe od XML, které jsou uvedeny v kapitole o XML. Zde uvedu jen ty nejdůležitější.

V MathML existují dvě třídy *prvků*. Většina prvků má uvozovací tag a ukončovací tag:

```
01. <nazev_prvku> … </nazev_prvku>
```

Tyto prvky mohou uvnitř obsahovat data, jako je text, čísla, symboly nebo další vnořené prvky.

Jiný typ prvku MathML je *prázdný* prvek ve tvaru:

```
02. <nazev_prvku/>
```

Tyto prvky mají pouze jeden tag, který nahrazuje uvozovací a ukončovací tag v předešlém případě.

Všechny MathML prvky připouštějí použití *atributů*, které do jisté míry ovlivňují jejich vlastnosti a poskytují další informace o prvku, ač nepovinné. Každý atribut má své jméno a hodnotu. Hodnota atributu musí být vždy uzavřena v uvozovkách. Obecný formát atributů u jednotlivých druhů prvků uvedu na příkladech:

Atribut umístěný v prvku s párovými tagy:

```
01. <nazev_prvku nazev_atributu="hodnota"> … </nazev_prvku>
```

Atribut umístěný v prázdném prvku:

```
02. <nazev_prvku nazev_atributu="hodnota"/>
```

5 Prezentační značení MathML

Tato kapitola představuje “prezentační“ prvky MathML [11, 17, 18], které lze použít k popisu struktur matematických výrazů.

5.1 Co jsou prezentační prvky

Prezentační prvky odpovídají tradičnímu matematickému zápisu. Jinými slovy jde o základní druhy symbolů a struktur výrazů, na nichž je postaven matematický zápis.

Prezentační prvky MathML jsou určeny k vyjádření syntaktické struktury matematického zápisu podobným způsobem, jako jsou nadpisy, sekce nebo odstavce u syntaktické struktury vyšší úrovně textových dokumentů. Z tohoto důvodu například jeden řádek identifikátorů a operátorů "$x + a/b$" nebude zastoupen jen jedním prvkem `<mrow>` (zobrazí vodorovný řádek s argumenty), ale více vnořenými prvky `<mrow>` odpovídající “podvýrazům“, ze kterých se skládá matematický výraz. V tomto případě:

```
01.  <mrow>
02.    <mi> x </mi>
03.    <mo> + </mo>
04.    <mrow>
05.      <mi> a </mi>
06.      <mi> / </mi>
07.      <mi> b </mi>
08.    </mrow>
09.  </mrow>
```

5.2 Třídění prezentačních prvků

Prezentační prvky jsou rozděleny do dvou skupin:

- *Symbolické prvky*

Všechny jednotlivé symboly v matematických výrazech by měly být zastoupeny symbolickými prvky MathML. Základními symbolickými prvky jsou identifikátory (například proměnné nebo názvy funkcí), čísla a operátory (včetně ohraničení, jako jsou závorky nebo oddělovače, například čárky). Existují také symbolické prvky představující text nebo mezery, které mají více estetický než matematický význam. Ačkoli symbolické prvky představují pouze jednoúčelné symboly (názvy, čísla, popisky, matematické symboly atd.), mohou se skládat z více než jednoho znaku. Například `sin` a `123` jsou zastoupeny jedním symbolickým prvkem **<mi>**`sin`**</mi>** a **<mn>**`123`**</mn>**.

- *Schémata prvků*

V tradičním matematickém zápisu jsou výrazy rekurzivně sestaveny z menších výrazů a jednotlivých symbolů a nakonec umístěny do jednoho souboru notačních struktur, který může být myšlen jako "konstrukční výraz". V MathML jsou výrazy vytvářeny stejným způsobem, kdy schémata prvků hrají roli konstrukčních výrazů. Schémata upřesňují způsob, jakým budou podvýrazy začleněny do větších matematických zápisů. Terminologie vychází ze skutečnosti, že každé schéma prvků odpovídá jinému způsobu, kterým se z podvýrazů tvoří větší zápisy v tradiční matematické sazbě. Jednotlivá schémata jsou popsána v kapitolách *5.4*, *5.5* a *5.6*.

5.3 Symbolické prvky

Symbolické prvky představují jednotlivé symboly, názvy, čísla, popisky apod. Tyto prvky mají obecně pouze obsahový charakter.

- <mi> … </mi>

  Obsah <mi> prvku je zobrazen jako **identifikátor**. Identifikátory mohou obsahovat proměnné, názvy funkcí a symbolické konstanty. Jednotlivé znaky jako například ”x” a ”d” jsou zobrazeny kurzívou, zatímco víceznakové identifikátory, jako jsou ”sin” a ”log”, písmem svislým.

  Atributy[1]: *mathvariant* a *fontstyle*

  Příklad:

```
01. <mi> x </mi>
02. <mi> sin </mi>
03. <mi mathvariant="bold"> L </mi>
```

- <mn> … </mn>

  Obsah <mn> prvku je zobrazen jako **číslo**. Čísla mohou být zobrazena s desetinnou čárkou nebo tečkou, a tím pádem reprezentovat jak celá tak desetinná čísla.

  Příklad:

```
01. <mn> 2 </mn>
02. <mn> 0,123 </mn>
03. <mn> 2.1e10 </mn>
```

- <mo> … </mo>

  Obsah <mo> prvku je zobrazen jako **operátor**. Mezi operátory patří početní úkony, závorky, oddělovače (čárky, středníky) apod.

  Atributy: *form*, *separator*, *symmetric*, *maxsize*

  Příklad:

```
01. <mo> ≤ </mo>
02. <mo> ( </mo>
03. <mo mathvariant="bold"> Σ </mo>
```

[1] http://www.w3.org/TR/MathML2/chapter3.html#presm.tokel

- `<mtext>` … `</mtext>`

  Obsah `<mtext>` prvku reprezentuje libovolný **text**. Obecně se tento prvek používá ke značení komentářů.

  Příklad:

```
01.  <mtext> věta </mtext>
02.  <mtext> / * komentář * / </mtext>
```

- `<mspace>` … `</mspace>`

  Obsah `<mspace>` prvku představuje **mezeru** jakékoliv požadované velikosti dané atributy.

  Atributy: *width*, *height*, *depth*, *linebreak*

- `<ms>` … `</ms>`

  Obsah `<ms>` prvku představuje **řetězec znaků**. Tento prvek se používá u výrazů algebry počítačových a jiných systémů, které obsahují programovací jazyky.

  Atributy: *lquote* a *rquote*

- `<mglyph>` … `</mglyph>`

  UNICODE definuje velký počet znaků používaných v matematice. Existují však znaky, které ještě nejsou v UNICODE zahrnuty a jejich začlenění je často zdlouhavý proces. MathML řeší tuto situaci právě prvkem `<mglyph>`, jehož obsah představují **znaky**, které v UNICODE nejsou definovány.

  Atributy: *alt*, *fontfamily* a *index*

5.4 Obecná schémata prvků

Kromě symbolických prvků existuje několik dalších prezentačních prvků MathML. V této kapitole uvedu základní značení, jako jsou zlomky nebo odmocniny, které patří mezi tzv. *obecná schémata prvků*.

- `<mrow>` … `</mrow>`

  Prvek `<mrow>` se používá pro skupinu libovolného počtu výrazů seskupených ve vodorovné řádce. Obvykle se skládá z jednoho nebo více prvků `<mi>`, `<mn>` nebo `<mo>`.

  Příklad: (x, y)

```
01.  <mrow>
02.    <mo> ( </mo>
03.    <mrow>
04.      <mi> x </mi>
05.      <mo> , </mo>
06.      <mi> y </mi>
07.    </mrow>
08.    <mo> ) </mo>
09.  </mrow>
```

- `<mfrac>` … `</mfrac>`

  Prvek `<mfrac>` se používá pro označení **zlomku**.

  Atributy[2]: *linethickness*, *numalign*, *denomalign* a *bevelled*

  Příklad: $\begin{bmatrix} a \\ b \end{bmatrix}$

```
01.  <mrow>
02.    <mo> [ </mo>
03.    <mfrac linethickness="0">
04.       <mi> a </mi>
05.       <mi> b </mi>
06.    </mfrac>
07.    <mo> ] </mo>
08.  </mrow>
```

[2] http://www.w3.org/TR/MathML2/chapter3.html#presm.genlayout

- `<msqrt>`, `<mroot>`

  Tyto prvky tvoří **odmocniny**. Prvek `<msqrt>` se používá pro označení druhé odmocniny, zatímco prvek `<mroot>` se používá pro vytváření odmocnin s indexem (např. třetí odmocnina). `<mroot>` vyžaduje přesně dva argumenty.

  Syntaxe pro tyto prvky:

```
01.  <msqrt> základ </msqrt>
02.  <mroot> základ index </mroot>
```

- `<mstyle>` … `</mstyle>`

  Prvek `<mstyle>` se používá na **změnu vzhledu** svého obsahu. `<mstyle>` přijímá libovolný počet argumentů. Dalo by se říct, že prvek `<mstyle>` změní výchozí hodnoty atributů prvků, které obsahuje.

  Atributy: *scriptlevel, displaystyle, scriptsizemultiplier, scriptminsize, background, veryverythinmathspace, verythinmathspace, thinmathspace, mediummathspace, thickmathspace, verythickmathspace* a *veryverythickspace*

- `<merror>` … `</merror>`

  Prvek `<merror>` se používá k zobrazení svého obsahu jako **chybové zprávy**. Toho dosahuje například tím, že zobrazí svůj obsah v červené barvě, změní barvu pozadí nebo svůj obsah rozbliká. Obsahem může být jakýkoliv výraz nebo sekvence výrazů.

- `<mpadded>` … `</mpadded>`

  Prvek `<mpadded>` se používá k umístění **mezer** kolem svého obsahu. Jak název napovídá, tento prvek určuje tzv. *padding* (velikost okrajů prvku).

  Atributy: *width, lspace, height* a *depth*

  Příklad:

```
01.  <mpadded width="100%"> … </mpadded>
02.  <mpadded width="-0.3em"> … </mpadded>
03.  <mpadded width="+0%"> … </mpadded>
```

- `<mphantom>` … `</mphantom>`

  Prvek `<mphantom>` změní svůj obsah na **transparentní** (neviditelný), ale zároveň zachová jeho velikost.

  Příklad: $\frac{x+y+z}{x \quad +z}$

```
01.  <mfrac>
02.    <mrow>
03.      <mi> x </mi>
04.      <mo> + </mo>
05.      <mi> y </mi>
06.      <mo> + </mo>
07.      <mi> z </mi>
08.    </mrow>
09.    <mrow>
10.      <mi> x </mi>
11.      <mphantom>
12.        <mo form="infix"> + </mo>
13.        <mi> y </mi>
14.      </mphantom>
15.      <mo> + </mo>
16.      <mi> z </mi>
17.    </mrow>
18.  </mfrac>
```

- `<mfenced>` … `</mfenced>`

  Prvek `<mfenced>` ohraničí svůj obsah **závorkami** nebo jinými **oddělovači** (např. čárkou).

  Atributy: *open, close* a *separators*

  Příklad: $\left\{ \frac{a}{b}; 1 \right\}$

```
01.  <mfenced open="{" close="}" separators=";">
02.    <mfrac>
03.      <mi> a </mi>
04.      <mi> b </mi>
05.    </mfrac>
06.    <mn> 1 </mn>
07.  </mfenced>
```

5.5 Schémata textu a krajních hodnot

Tato kapitola se zabývá skriptováním výrazů jako jsou například horní a dolní indexy.

- `<msup>`, `<msub>` a `<msubsup>`

  Tyto prvky se používají k vytváření **indexů**. Prvek `<msup>` tvoří horní index, prvek `<msub>` tvoří dolní index a prvek `<msubsup>` se používá pro připojení dolního a horního indexu k základnímu výrazu.

  Atributy[3]: *subscriptshift* a *supersubscriptshift*

  Příklad: $\int_0^1 e^x \mathrm{d}x$

```
01.  <mrow>
02.    <msubsup>
03.      <mo> &int; </mo>
04.      <mi> 0 </mi>
05.      <mi> 1 </mi>
06.    </msubsup>
07.    <mrow>
08.      <msup>
09.        <mi> &ExponentialE; </mi>
10.        <mi> x </mi>
11.      </msup>
12.      <mrow>
13.        <mo> &DifferentialD; </mo>
14.        <mi> x </mi>
15.      </mrow>
16.    </mrow>
17.  </mrow>
```

- `<munder>`, `<mover>` a `<munderover>`

  Tyto prvky se používají k **připojování obsahu** k základnímu výrazu. Prvek `<munder>` připojí obsah pod základ, prvek `<mover>` připojí obsah nad základ a prvek `<munderover>` připojí obsah nad i pod základní výraz.

  Atributy: *accent* a *accentunder*

[3] http://www.w3.org/TR/MathML2/chapter3.html#presm.scrlim

Syntaxe pro tyto prvky:

```
<munder> základ podvýraz </munder>
<mover> základ nadvýraz </mover>
<munderover> základ podvýraz nadvýraz </mover>
```

Příklad: $\hat{x}$

```
<mrow>
  <mover accent="true">
    <mi> x </mi> <mo> &Hat; </mo>
  </mover>
</mrow>
```

- `<mmultiscripts>` … `</mmultiscripts>`

  Prvek `<mmultiscripts>` doplňuje základní výraz o **přední index** a **tenzorové znaky** (umístěné vlevo od základny). Chybějící členy mohou být nahrazeny prázdným prvkem `<none/>`.

  Syntaxe pro tento prvek:

```
<mmultiscripts>
  základ
  dolní index
  horní index
  <mprescripts/>
  dolní tenzor
  horní tenzor
</mmultiscripts>
```

  Příklad: ${}_0X_1$

```
<mrow>
  <mmultiscripts>
    <mi> X </mi>
    <mn> 1 </mn>
    <none/>
    <mprescripts/>
    <mn> 0 </mn>
    <none/>
  </mmultiscripts>
</mrow>
```

5.6 Tabulky a matice

Matice, pole a další tabulkové matematické výrazy jsou vytvářeny pomocí prvků `<mtable>`, `<mtr>`, `<mtd>` a `<mlabeledtr>`. Tyto prvky jsou obdobné jako prvky v HTML (*table*, *td* a *tr*).

- `<mtable>` … `</mtable>`

  Prvek `<mtable>` se používá k vytváření tabulek a matic.

  Mezi atributy[4] tohoto prvku patří například *align, rowalign, columnalign, width, frame*.

- `<mtr>` … `</mtr>`

  Prvek `<mtr>` se používá k vytvoření jedné řádky v tabulce nebo matici.

  Atributy: *rowalign, columnalign, groupalign*

- `<mtd>` … `</mtd>`

  Prvek `<mtd>` představuje jeden záznam nebo buňku v tabulce nebo matici.

  Atributy: *rowspan, columnspan, rowalign, columnalign, groupalign*

- `<mlabeledtr>` … `</mlabeledtr>`

  Prvek `<mlabeledtr>` představuje označení řádku tabulky nebo matice a může být také použit pro číslování rovnic.

Příklad: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

```
01.  <mrow>
02.    <mo> ( </mo>
03.    <mtable>
04.      <mtr>
05.        <mtd> <mn> 1 </mn> </mtd>
06.        <mtd> <mn> 0 </mn> </mtd>
07.      </mtr>
08.      <mtr>
09.        <mtd> <mn> 0 </mn> </mtd>
10.        <mtd> <mn> 1 </mn> </mtd>
11.      </mtr>
12.    </mtable>
13.    <mo> ) </mo>
14.  </mrow>
```

[4] http://www.w3.org/TR/MathML2/chapter3.html#presm.tabmat

5.7 Příklady prezentačního značení MathML

$$x^3 + 6y + 5 = 0$$

```
01.  <mrow>
02.    <mrow>
03.      <msup>
04.        <mi> x </mi>
05.        <mn> 3 </mn>
06.      </msup>
07.      <mo> + </mo>
08.      <mrow>
09.        <mn> 6 </mn>
10.        <mo> &InvisibleTimes; </mo>
11.        <mi> y </mi>>
12.      </mrow>
13.      <mo> + </mo>
14.      <mn> 5 </mn>
15.    </mrow>
16.    <mo> = </mo>
17.    <mn> 0 </mn>
18.  </mrow>
```

$$A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

```
01.  <mrow>
02.    <mi> A </mi>
03.    <mo> = </mo>
04.    <mfenced open="[" close="]">
05.      <mtable>
06.        <mtr>
07.          <mtd> <mi> x </mi> </mtd>
08.          <mtd> <mi> y </mi> </mtd>
09.        </mtr>
10.        <mtr>
11.          <mtd> <mi> z </mi> </mtd>
12.          <mtd> <mi> w </mi> </mtd>
13.        </mtr>
14.      </mtable>
15.    </mfenced>
16.  </mrow>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
01.  <mrow>
02.    <mi> x </mi>
03.    <mo> = </mo>
04.    <mfrac>
05.      <mrow>
06.        <mrow>
07.          <mo> - </mo>
08.          <mi> b </mi>
09.        </mrow>
10.        <mo> &PlusMinus; </mo>
11.        <msqrt>
12.          <mrow>
13.            <msup>
14.              <mi> b </mi>
15.              <mn> 2 </mn>
16.            </msup>
17.            <mo> - </mo>
18.            <mrow>
19.              <mn> 4 </mn>
20.              <mo> &InvisibleTimes; </mo>
21.              <mi> a </mi>
22.              <mo> &InvisibleTimes; </mo>
23.              <mi> c </mi>
24.            </mrow>
25.          </mrow>
26.        </msqrt>
27.      </mrow>
28.      <mrow>
29.        <mn> 2 </mn>
30.        <mo> &InvisibleTimes; </mo>
31.        <mi> a </mi>
32.      </mrow>
33.    </mfrac>
34.  </mrow>
```

Příklady k jednotlivým prvkům prezentačního značení jsou k dispozici na adrese http://www.w3.org/Math/testsuite/mml2-testsuite/index.html.

6 Významové značení MathML

Hlavním cílem významového značení [12, 17, 18] je stanovení jednoznačných spojení mezi matematickými strukturami a jejich matematickými významy. Významové prvky přesně odpovídají částem matematického výrazového stromu.

Podstatné výhody zavedení významových prvků:

- Snížení potřeby používání prezentačních prvků. Když je matematická sémantika odvozena z prezentačních značek, vykonávající programy musí být buď velmi sofistikované, nebo musí běžet za rizika nekompletní nebo nesprávné sémantiky, pokud jsou použity špatné konstrukce.
- Z druhu použitého elementu je okamžitě jasné, jaká informace se právě kóduje.
- Kombinace sémantiky a prezentačních součástí může být mnohem efektivněji a jednodušeji využita k určení vzhledu a matematického významu.

6.1 Použití významových prvků

Významové značení MathML je založeno na konceptu výrazového stromu. Obecným pravidlem je, že uzly stromu představují základní matematické objekty, jako jsou čísla, proměnné, aritmetické operace apod. Vnitřní uzly obecně představují některé druhy funkčních aplikací nebo jiných matematických konstrukcí, ze kterých je sestaven složený objekt. Významové prvky MathML lze rozdělit do následujících kategorií na základě jejich použití:

- *konstanty a symboly*
- *buňky*
- *operátory a funkce*
- *kvalifikátory*
- *vztahy*
- *podmínky*
- *sémantické mapování*

6.2 Buňky

Stavebním prostředkem matematických objektů jsou buňky, které mají následující rozdělení:

- *Symbolické prvky*

  `ci`, `cn`, `csymbol`

- *Konstrukční prvky*

  `interval`, `list`, `matrix`, `matrixrow`, `set`, `vector`, `apply`, `reln`, `lambda`, `fn`, `piecewise`, `piece`, `otherwise`

- *Speciální prvky*

  `declare`

6.2.1 Symbolické prvky

Symbolické prvky tvoří základ výrazového stromu MathML. Používají se k označení čísel a symbolů.

- `<cn>` … `</cn>`

  Prvek `<cn>` představuje **čísla**. Podporované typy čísel: reálná, celá, racionální, komplexní-kartézská, komplexní-polární. Výchozí typ prvku `<cn>` jsou reálná čísla.

  Příklady: 21 a $^{5}/_{4}$

```
01.  <cn type="real"> 21 </cn>
```

```
02.  <cn type="rational"> 5 <sep/> 4 </cn>
```

- `<ci>` … `</ci>`

  Prvek `<ci>` neboli „významový“ identifikátor představuje **proměnné** nebo **identifikátory**. Atribut `"type"` určuje typ objektu (reprezentující symbol), jehož výchozí hodnota není nastavena. Lze použít atribut `"definitionURL"` k identifikaci zvláštních vlastností. Prvek `<ci>` obsahuje jak skaláry tak například prvky z prezentačního značení MathML.

Příklad: C_1

```
01.  <ci>
02.    <msub>
03.      <mi> C </mi>
04.      <mn> 1 </mn>
05.    </msub>
06.  </ci>
```

Obsah prvku `<ci>` je považován za jeden symbol představující reálné číslo.

- `<csymbol>` … `</csymbol>`

  Prvek `<csymbol>` neboli „významový“ symbol představuje **symboly**, které nejsou součástí významových elementů poskytovaných MathML, ale které jsou definovány mimo specifikaci jazyka MathML. Atribut `"definitionURL"` odkazuje na konkrétní význam a atribut `"encoding"` určuje syntaxi definice.

Příklad: C^2

```
01.  <csymbol definitionURL="http://…" encoding="text">
02.    <msub>
03.      <mi> C </mi>
04.      <mn> 2 </mn>
05.    </msub>
06.  </csymbol>
```

6.2.2 Konstrukční prvky

MathML obsahuje řadu prvků, které svou kombinací vytvářejí složené objekty. Mezi složené objekty patří například seznamy nebo množiny. Každý konstrukční prvek vytváří nový typ objektu.

- `<interval>` … `</interval>`

  Prvek `<interval>` představuje jednoduché matematické **intervaly** reálných čísel. Ohraničení intervalu má na starost atribut `"closure"`, který může mít některou z těchto hodnot: *open*, *closed*, *open-closed*, *closed-open*. Výchozí hodnota atributu je *closed*.

Příklad: $(a, b]$

```
01.  <interval closure="open-closed">
02.    <ci> a </ci>
03.    <ci> b </ci>
04.  </interval>
```

- `<set>` a `<list>`

  Prvek `<set>` představuje **množinu** výrazů a prvek `<list>` **seznam** výrazů.

  Příklady: $\{a, b\}$

```
01.  <set>
02.    <ci> a </ci>
03.    <ci> b </ci>
04.  </set>
```

$[A, B]$

```
01.  <list>
02.    <ci> A </ci>
03.    <ci> B </ci>
04.  </list>
```

- `<matrix>` a `<matrixrow>`

  Prvek `<matrix>` představuje matematickou **matici**, jejíž řádky jsou označeny prvkem `<matrixrow>`. Všechny `<matrixrow>` v dané matici musí obsahovat stejný počet prvků.

  Příklad: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

```
01.  <matrix>
02.    <matrixrow>
03.      <cn> 1 </cn> <cn> 0 </cn>
04.    </matrixrow>
05.    <matrixrow>
06.      <cn> 0 </cn> <cn> 1 </cn>
07.    </matrixrow>
08.  </matrix>
```

- `<vector> … </vector>`

  Prvek `<vector>` představuje **vektor** pro účely násobení s maticemi. `<vector>` je rovnocenná matice skládající se z jednoho sloupce. Chová se stejně jako matice s jedním řádkem.

  Příklad: $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

```
01.  <vector>
02.    <cn> 1 </cn>
03.    <cn> 2 </cn>
04.    <cn> 3 </cn>
05.  </vector>
```

- `<apply> … </apply>`

  Prvek `<apply>` umožňuje „aplikovat“ funkce nebo operátory na své potomky. Téměř všechny výrazy ve významovém značení MathML jsou prováděny s použitím prvku `<apply>`. Funkce nebo operátory jsou prvním argumentem tohoto prvku.

  Příklady: $a + b$

```
01.  <apply><plus/>
02.    <ci> a </ci>
03.    <ci> b </ci>
04.  </apply>
```

$\sin(a)$

```
01.  <apply><sin/>
02.    <ci> a </ci>
03.  </apply>
```

- `<reln> … </reln>`

  Prvek `<reln>` se používal v MathML 1.0 k vytváření rovnic nebo vztahů. Vztahy jsou konstruovány stejným způsobem jako u prvku `<apply>`. Prvním argumentem `<reln>` je relační operátor, který má být použit na své podvýrazy. MathML 2.0 zachovává tento prvek kvůli kompatibilitě s verzí 1.0, v ostatních případech se používá prvek `<apply>`.

Příklad: $a = b$

```
01.  <reln><eq/>
02.    <ci> a </ci>
03.    <ci> b </ci>
04.  </reln>
```

- <fn> … </fn>

  Prvek <fn> se používal v MathML 1.0, kde výslovně uváděl, že výraz je použit jako funkce nebo operátor. Tento prvek je podporován z důvodu zpětné kompatibility, ale v ostatních případech ho nahrazuje prvek <apply>.

- <piecewise>, <piece> a <otherwise>

  Tyto prvky se používají pro podporu následujícího typu deklarace:

  Příklad: "$H(x) = 0 \quad pokud\ x\ je\ menší\ než\ 0,\ jinak \quad H(x) = x$"

```
01.  <piecewise>
02.    <piece>
03.      <cn> 0 </cn>
04.      <apply><lt/> <ci> x </ci> <cn> 0 </cn> </apply>
05.    </piece>
06.      <otherwise>
07.        <ci> x </ci>
08.      </otherwise>
09.  </piecewise>
```

- <lambda> … </lambda>

  Prvek <lambda> se používá k vytváření uživatelem definovaných funkcí ve výrazech s jednou nebo více proměnnými. Prvek <lambda> s vnitřní proměnou *n* obsahuje *n+1* potomků. První (až *n*-tý) argument je prvek <bvar>, který obsahuje identifikační údaje o vnitřních proměnných. Poslední argument je výraz definující funkci, obvykle to bývá prvek <apply>.

Příklad: $\lambda\,(x, \sin x)$

```
01.  <lambda>
02.    <bvar> <ci> x </ci> </bvar>
03.    <apply>
04.      <sin/>
05.      <ci> x </ci>
06.    </apply>
07.  </lambda>
```

6.2.3 Speciální prvky

- `<declare>` … `</declare>`

  Prvek `<declare>` se používá ke změně nebo nastavení výchozích hodnot atributů pro konkrétní matematické objekty.

  Tento prvek obsahuje jeden nebo dva argumenty. První argument, který je povinný, je objekt ovlivněný deklarací. Obvykle se jedná o prvek `<ci>`, poskytující identifikátor, který je deklarován jako:

```
01.  <declare type="vector"> <ci> V </ci> </declare>
```

  Druhý argument, který je volitelný představuje inicializační proměnné.

  Příklad:

```
01.  <declare type="vector">
02.    <ci> V </ci>
03.    <vector>
04.      <cn> 1 </cn>
05.      <cn> 2 </cn>
06.      <cn> 3 </cn>
07.    </vector>
08.  </declare>
```

  Typy konstrukčního prvku a deklarace musí být shodné.

6.3 Funkce, operátory a kvalifikátory

Operátory a funkce definované v MathML jsou rozděleny do následujících skupin:

unární aritmetické	factorial, minus, abs, conjugate, arg, real, imaginary, floor, ceiling
unární logické	not
unární funkční	inverse, ident, domain, codomain, image
unární klasické elementární funkce	sin, cos, tan, sec, csc, cot, sinh, cosh, tanh, sech, csch, coth, arcsin, arccos, arctan, arccosh, arccot, arccoth, arccsc, arccsch, arcsec, arcsech, arcsinh, arctanh, exp, ln, log
unární lineární algebra	determinant, transpose
unární počet a vektorový počet	divergence, grad, curl, laplacian
unární teorie množin	card
binární aritmetické	quotient, divide, minus, power, rem
binární logické	implies, equivalent, approx
binární operátory množin	setdiff
binární lineární algebra	vectorproduct, scalarproduct, outerproduct
n-nární aritmetické	plus, times, max, min, gcd, lcm
n-nární statistické	mean, sdev, variance, median, mode
n-nární logické	and, or, xor
n-nární lineární algebra	selector
n-nární operátory množin	union, intersect, cartesianproduct
n-nární funkční	fn (zastaralé), compose
integrál, suma, součin	int, sum, product
diferenciální operátory	diff, partialdiff
kvalifikátory	forall, exists

Tab. 6.1: Přehled operátorů a funkcí
Zdroj: http://www.w3.org/TR/MathML2/

Unární operátory jsou následovány přesně jedním argumentem uvnitř prvku `<apply>`, *binární* operátory přesně dvěma argumenty a *n-nární* operátory libovolným počtem argumentů.

kvalifikátory	`lowlimit, uplimit, bvar, degree, logbase, interval, condition, domainofapplication, momentabout`
operátory	`int, sum, product, root, diff, partialdiff, limit, log, moment forall, exists`
n-nární operátory	`plus, times, max, min, gcd, lcm, mean, sdev, variance, median, mode, and, or, xor, union, intersect, cartesianproduct, compose, eq, leq, lt, geq, gt`
uživatelem definované operátory	`csymbol, ci`

Tab. 6.2: Přehled operátorů a kvalifikátorů
Zdroj: http://www.w3.org/TR/MathML2/

Operátory kvalifikace jsou prázdné funkce, které se liší od normálních prázdných funkcí pouze v tom, že podporují použití speciálních *kvalifikačních prvků,* které specifikují význam funkce ve větší míře. Kvalifikátory vždy následují operátory a předcházejí přítomným argumentům. Jestliže se nachází ve výraze více než jeden kvalifikátor, jsou uvedeny v tomto pořadí: `bvar, lowlimit, uplimit, interval, condition, domainofapplication, degree, momentabout, logbase`.

Typický příklad: $\int_0^1 x$

```
01.  <apply>
02.    <int/>
03.    <bvar> <ci> x </ci> </bvar>
04.    <lowlimit> <cn> 0 </cn> </lowlimit>
05.    <uplimit> <cn> 1 </cn> </uplimit>
06.  </apply>
```

6.4 Vztahy

Matematické vztahy se dělí do následujících skupin:

binární vztahy	neq, equivalent, approx, factorof
binární logické vztahy	implies
binární vztahy množin	in, notin, notsubset, notprsubset
binární vztahy řad	tendsto
n-nární vztahy	eq, leq, lt, geq, gt
n-nární vztahy množin	subset, prsubset

Tab. 6.3: Přehled vztahů
Zdroj: http://www.w3.org/TR/MathML2/

Významové značení MathML zahrnuje řadu prázdných elementů, které označují aritmeticko-logické vztahy. Pokud vztahy vyhodnocuje externí aplikace (MathML nespecifikuje jak vztahy hodnotit), je obvykle navrácena pravdivostní hodnota (pravda x nepravda). Naproti tomu, operátory navrátí obvykle hodnotu stejného typu jako je vstupní hodnota. Například výsledkem porovnání $a > b$ je buď pravda, nebo nepravda (na rozdíl tomu $1 + 1$ je opět číslo).

Argumenty vztahů jsou ohraničeny prvkem <apply> stejným způsobem jako u ostatních funkcí. V MathML 1.0 byly relační operátory ohraničeny prvkem <reln>. Tento prvek je stále podporován, ale v MathML 2.0 ho nahradil prvek <apply>.

Příklad: $a > b$

```
01.  <apply>
02.    <gt/>
03.    <ci> a </ci>
04.    <ci> b </ci>
05.  </apply>
```

6.5 Podmínky

podmínka	`condition`

Tab. 6.4: Značení podmínky
Zdroj: http://www.w3.org/TR/MathML2/

Prvek `<condition>` je používán v mnoha kontextech MathML. Podmínky se využívají k vytváření objektů, jako jsou pravidla v množinách nebo seznamech namísto výpočtů. Podmínky mohou být používaný společně s operátory `<exists/>` a `<forall/>` k vytváření logických výrazů nebo použity v různých způsobech v kombinaci s některými jinými operátory. Například mohou být použity s prvkem `<int>`, kde určují oblasti integrace nebo u seznamů, kde určují *min* a *max* hodnoty.

Prvek `<condition>` je téměř vždy použit spolu s jedním nebo více prvky `<bvar>` společně s podvýrazy prvků typu `<apply>` nebo `<reln>`.

Příklad: "*existuje* x *takové, že* $x^5 < 3$"

```
01.  <apply>
02.    <exists/>
03.    <bvar> <ci> x </ci> </bvar>
04.    <condition>
05.      <apply><lt/>
06.        <apply>
07.          <power/>
08.          <ci> x </ci>
09.          <cn> 5 </cn>
10.        </apply>
11.        <cn> 3 </cn>
12.      </apply>
13.    </condition>
14.    <true/>
15.  </apply>
```

6.6 Syntaxe a sémantika

mapování	`semantics, annotation, annotation-xml`

Tab. 6.5: Druhy mapování
Zdroj: http://www.w3.org/TR/MathML2/

Použití významového značení spíše než prezentačního je v matematice někdy označováno jako sémantické vyjádření. Strom platné struktury prvků za použití MathML významových nástrojů přímo odkazuje na výrazový strom a základní matematické vyjádření. Proto považujeme významové značení za kódování syntaxe matematických výrazů. To je obecně dostačující pro získání některé vykreslovací a dokonce i některé symbolické schopnosti (například faktorizaci polynomů).

Prvek `<annotation>` se používá pro libovolná data. Tato data mohou být ve formě textu, počítačové algebry, C programů nebo kterékoliv jiné aplikace. Prvek annotation má atribut `"encoding"` definující formu použití.

Prvek `<annotation-xml>` představuje sémantické informace o XML souboru.

Příklad: $\sin(x) + 5$

```
01.  <semantics>
02.    <apply>
03.      <plus/>
04.      <apply>
05.        <sin/>
06.        <ci> x </ci>
07.      </apply>
08.      <cn> 5 </cn>
09.    </apply>
10.    <annotation encoding="Maple">
11.      sin(x) + 5
12.    </annotation>
13.    <annotation-xml encoding="MathML-Presentation">
14.    …
15.    …
16.    </annotation-xml>
17.  </semantics>
```

6.7 Příklady významového značení MathML

$$x^3 + 6y + 5 = 0$$

```
01.  <reln>
02.    <eq/>
03.    <apply>
04.      <plus/>
05.      <apply>
06.        <power/>
07.        <ci> x </ci>
08.        <cn> 3 </cn>
09.      </apply>
10.      <apply>
11.        <times/>
12.        <cn> 6 </cn>
13.        <ci> y </ci>
14.      </apply>
15.      <cn> 5 </cn>
16.    </apply>
17.    <cn> 0 </cn>
18.  </reln>
```

$$A = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

```
01.  <reln>
02.    <eq/>
03.    <ci> A </ci>
04.    <matrix>
05.      <matrixrow>
06.        <ci> x </ci>
07.        <ci> y </ci>
08.      </matrixrow>
09.      <matrixrow>
10.        <ci> z </ci>
11.        <ci> w </ci>
12.      </matrixrow>
13.    </matrix>
14.  </reln>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
01.  <reln>
02.    <eq/>
03.    <ci> x </ci>
04.    <apply>
05.      <divide/>
06.      <apply>
07.        <fn> <mo> &PlusMinus; </mo> </fn>
08.        <apply>
09.          <minus/>
10.          <ci> b </ci>
11.        </apply>
12.        <apply>
13.          <root/>
14.          <apply>
15.            <minus/>
16.            <apply>
17.              <power/>
18.              <ci> b </ci>
19.              <cn> 2 </cn>
20.            </apply>
21.            <apply>
22.              <times/>
23.              <cn> 4 </cn>
24.              <ci> a </ci>
25.              <ci> c </ci>
26.            </apply>
27.          </apply>
28.        </apply>
29.      </apply>
30.      <apply>
31.        <times/>
32.        <cn> 2 </cn>
33.        <ci> a </ci>
34.      </apply>
35.    </apply>
36.  </reln>
```

Příklady k jednotlivým prvkům významového značení jsou k dispozici na adrese http://www.w3.org/Math/testsuite/mml2-testsuite/index.html.

7 Kombinace prezentačního a významového značení

Prezentační a významové značení [13, 17, 18] mohou být kombinovány dvěma způsoby. Prvním způsobem je kombinování prezentačních a významových prvků v jediném výrazovém stromě, což se nazývá *smíšené značení*. Druhým způsobem je poskytnutí explicitní prezentace a explicitního významu ve dvou stromech, což se nazývá *paralelní značení*. Tato kapitola popisuje smíšené i paralelní značení a jejich použití.

7.1 Typy značení v MathML

Prezentační značení zachycuje výrazovou strukturu, kterou popisuje dostatečně souhrnným způsobem a usnadňuje tak interpretaci pro různá média. Prezentační značení může být tedy relativně snadno vyjádřeno na obrazovkách v širokém i úzkém řádku, v ASCII, v grafice nebo v tisku. Je to tím, že poskytuje informace o seskupování jednotlivých částí výrazu, třídění symbolů atd. Prezentační značení se netýká přímo matematické struktury nebo významu výrazu, ale tyto struktury spolu v mnoha situacích úzce souvisejí. Sofistikované aplikace jsou schopny vyvodit matematický význam ze struktury zápisu za předpokladu známého kontextu. Nicméně v praxi je odvozování matematického významu přenecháno na uživateli.

Významové značení zachycuje matematickou strukturu. Kóduje matematické struktury takovým způsobem, aby byl význam matematických výrazů dobře čitelný pro aplikační programy. Ačkoli mohou být údaje o mapování matematických struktur a matematických významů velmi složité, v praxi existuje široce rozšířená dohoda o tradičních významech většiny matematických konstrukcí. Důsledkem toho je velká část významových výrazů snadno přístupná pro zpracování aplikacemi, nezávisle na tom, kde nebo jak jsou zobrazeny uživatelem. V jiných případech může být také zápis významového značení vyjmut z webového prohlížeče a zpracován vhodnými nástroji matematického softwaru, který s určitou jistotou provede potřebné výpočty.

7.2 Smíšené značení

MathML nabízí uživatelům prvky jak prezentačního, tak významového značení. Zda použít to či ono značení nebo jejich kombinaci, záleží především na aspektech zobrazení a celkové interpretaci výrazů.

7.2.1 Zakázané kombinace

Důležitým aspektem při kombinaci prezentačního a významového značení je, aby konečný výsledek dával smysl. Když jsou v prezentačním výrazu obsaženy oba druhy značení, mělo by být možné vyjádřit výsledný smíšený výraz jednoduše a účelně. Naopak, když se smíšené značení objeví ve významovém výrazu, mělo by být možné jednoduše a účelně přiřadit sémantickou interpretaci k výrazu jako celku. Při kombinování prezentačního a významového značení existují určitá omezení, která musí být dodržena, aby nedošlo k situaci, kdy je výraz nejednoznačný nebo jinak problematický.

Dva příklady, které ilustrují problémy, kterým je třeba se vyhnout při smíšeném značení:

```
01.  <mrow>
02.    <bvar> x </bvar> <mo> + </mo> <bvar> y </bvar>
03.  </mrow>
```

V tomto příkladu byl významový prvek `<bvar>` nevhodně začleněn do prezentačního výrazu. Vzhledem k tomu, že `<bvar>` prvek vyžaduje pro svůj význam uzavřený kontext, je tento výraz nejasný.

```
01.  <apply>
02.    <ci> x </ci> <mo> + </mo> <ci> y </ci>
03.  </apply>
```

Zde je problematický prvek `<mo>`. Tento způsob kombinování prezentačního a významového značení je rovněž zakázán. V tomto případě nelze prvek `<plus/>` zaměnit operátorem `<mo> + </mo>`.

7.2.2 Prezentační značení obsažené ve významovém značení

Použití prezentačního značení v rámci významového značení je omezeno na situace, které nemají vliv na schopnost významového značení jednoznačně kódovat matematické významy. Prezentační značení se může ve významovém značení objevit jen třemi možnými způsoby:

- *Prezentační značení uvnitř symbolických prvků* `<ci>` a `<cn>`

Symbolické prvky `<ci>` a `<cn>` mohou obsahovat jakoukoli sekvenci znaků MathML a prezentačních prvků.

- *Prezentační značení uvnitř prvku* `<csymbol>`

Prvek `<csymbol>` může obsahovat znaky MathML dohromady s prezentačním značením nebo s významovými prvky typu buňka. Nesmí však obsahovat prezentační a významové prvky společně.

- *Prezentační značení uvnitř prvku* `<semantics>`

Jedním z hlavních cílu prvku `<semantics>` je poskytnout mechanismus pro začlenění libovolných výrazů MathML do významového značení sémanticky smysluplným způsobem.

Jakékoliv jiné prezentační značení vyskytující se ve významovém značení je považováno za chybu.

7.2.3 Významové značení obsažené v prezentačním značení

Hlavním principem vkládání významového značení do prezentačních výrazů je jednoznačnost výsledného výrazu. Obecně to znamená, že vkládaný výraz musí být sémanticky smysluplný, protože interpretace významového značení závisí na jeho významu.

Určité významové prvky odvozují část svého významu z okolního kontextu. Ten určuje, jestli např. prvek `<bvar>` označuje integrální funkci, logické kvantifikátory nebo výrazy `<lambda>`.

Operátory, vztahy, buňky, konstanty a symbolické prvky mají smysl samy o sobě, zatímco prvky kvalifikačního a podmínkového typu nikoli.

Zvláštní komentář si zaslouží prvky `<annotation>`, `<annotation-xml>`, `<sep>`, `<declare>`, `<bvar>`, `<condition>`, `<degree>`, `<logbase>`, `<lowlimit>`, `<uplimit>`, které se objevují jen ve velmi specifických případech, a proto nejsou povoleny jako hlavní podvýrazy v prezentačním značení. Naproti tomu prvek <semantics> obsahuje dostatečné informace, aby byl v prezentačním značení povolen.

7.3 Paralelní značení

Některé aplikace jsou schopné využít informace jak prezentační, tak i významové formy. Je ale důležité, aby byly tyto formy použity pro stejný matematický výraz. To se nazývá *paralelní značení*.

Paralelní značení je dosaženo pomocí prvku `<semantics>`. Může být použito buď samostatně nebo jako součást širšího významu nebo prezentačního stromu.

Příklad kódování booleovského aritmetického výrazu $(a + b)(c + d)$*:*

```
01.  <semantics>
02.    <mrow>
03.      <mrow>
04.        <mo>(</mo><mi>a</mi><mo>+</mo><mi>b</mi><mo>)</mo>
05.      </mrow>
06.      <mo> &InvisibleTimes; </mo>
07.      <mrow>
08.        <mo>(</mo><mi>c</mi><mo>+</mo><mi>d</mi><mo>)</mo>
09.      </mrow>
10.    </mrow>
11.    <annotation-xml encoding="MathML-Content">
12.      <apply><and/>
13.        <apply><xor/><ci>a</ci> <ci>b</ci></apply>
14.        <apply><xor/><ci>c</ci> <ci>d</ci></apply>
15.      </apply>
16.    </annotation-xml>
17.  </semantics>
```

8 Znaky MathML

Znaky MathML [14, 17, 18] jsou definovány buď v UNICODE, jako je tomu u dokumentů XML, nebo pomocí prvku `<mglyph>`. Ten je používán pro znaky, které nejsou v UNICODE zahrnuty. Vzhledem k tomu, že UNICODE UCS obsahuje více než 1000 speciálních abecedních znaků pro použití v matematice s UNICODE 3.1 a přes 900 dalších speciálních symbolů s UNICODE 3.2, je použití prvku `<mglyph>` velmi výjimečné.

Libovolné znaky povolené v XML mohou být použity i v MathML a dalších XML dokumentech. Znaky jsou psány v šestnáctkové soustavě 09 (tab=U+0009), 0A (line feed= U+000A), 0D (carriage return=U+000D), 20-D7FF (U+0020…U+D7FF), E000-FFFD (U+E000…U+FFFD) a 10000-10FFFF (U+010000…U+10FFFF). Zápisy v závorkách začínající na *U+* se odkazují na znaky v UNICODE. Znaky v kódovací formě *D7FF* jsou požívány pro bloky náhradních dvojic, které nepatří do UNICODE. *U+FFFE* se používá ke stanovení pořadí bytů v některých kódováních.

V zásadě existují tři způsoby kódování znaků:

- Přímé použití znaků: například znak *A* lze zapsat jako '*A*' z klávesnice (znak U+0041). Tato možnost je dostupná pouze v případě, že kódování XML dokumentu obsahuje potřebný znak. Nejčastěji však bude znak *A* zapsán pomocí pozice v ASCII tabulce.

- Pomocí číselných odkazů na znaky v XML. Například znak *A* lze zapsat: &#65 (desítková soustava) nebo &#x41 (šestnáctková). Čísla pokaždé odkazují na kódování UNICODE (a nikoli na kódování znaků použité v souboru XML).

- Pomocí odkazů na entity: MathML DTD definuje vnitřní entity, které se rozšiřují na znaky. Každá část odkazující na entity musí používat deklaraci DOCTYPE, která upřesňuje MathML DTD. Potřeba použití DOCTYPE komplikuje zařazení MathML do některých dokumentů. Odkazy na entity jsou velmi užitečné pro malé názorné příklady.

Ve zvláštních případech může nastat situace, kdy námi potřebný znak není zahrnutý v UNICODE. V těchto případech můžeme použít prvek <mglyph> k přímému přístupu k symbolu z jiného písma a vytvořit tak v MathML náhradu za odpovídající znak. Všechny symbolické prvky MathML mohou ve svém obsahu přijímat prvek `<mglyph>`.

Některé znaky, které nemají označení <mglyph>, ale přesto jsou důležité zejména pro kvalitu tisku nebo alternativní interpretaci, se nazývají *neznačené* znaky.

Níže uvedené znaky nepředstavují jednoduché mezery v matematických výrazech, ale jedná se o obzvláště důležité nové prvky v UCS. Poskytují textové stopy, které mohou zvýšit kvalitu tisku a umožnit unikátní využití matematické sémantiky z textu, který se zdá jako nejednoznačný.

Název znaku	UNICODE	Popis
`&InvisibleTimes;`	02062	označí násobení, rozumí se bez znaku
`&InvisibleComma;`	02063	používán jako oddělovač
`&ApplyFunction;`	02061	zobrazí funkční aplikaci v prezentačním označení

Tab. 8.1: Mezery
Zdroj: http://www.w3.org/TR/MathML2/

Speciální konstanty:

Název znaku	UNICODE	Popis
`&CapitalDifferentialD;`	02145	*D* pro použití v diferenciálech, např. v rámci integrování
`&DifferentialD;`	02146	*d* pro použití v diferenciálech, např. v rámci integrování
`&ExponentialE;`	02147	*e* jako základ přirozených logaritmů
`&ImaginaryI;`	02148	*i* jako odmocnina z -1, imaginární jednotka
`&pi;`	---	Ludolfovo číslo, přibližně 3,1415926535...

Tab. 8.2: Speciální konstanty
Zdroj: http://www.w3.org/TR/MathML2/

9 Implementace MathML

Aby bylo MathML efektivní, musí dobře pracovat s širokou škálou interpretů, procesorů, překladačů a editorů. Tato kapitola se zabývá některými otázkami implementace spojené s generováním a interpretací MathML [15, 17, 18]. MathML se používá především ke kódování matematiky v dokumentech na webu jako nejdůležitější možné rozhraní. Proto jsou největší problémy spojené se zobrazováním MathML v HTML4 a XHTML.

Vkládání výrazů MathML do jiných XML dokumentů se provádí třemi různými způsoby:

- MathML musí být sémanticky integrováno. Značení MathML musí být XML aplikací bezchybně rozpoznáno jako validní. Jedná se především o správu jmenných prostorů v XML.

- Interpretace MathML musí být v případě HTML nebo XHTML začleněna do prohlížeče. Některé prohlížeče již provádějí interpretaci MathML samostatně a do budoucna lze očekávat, že se k nim přidají další prohlížeče. Zároveň mají některé prohlížeče rozvinuté infrastruktury pro usnadnění zobrazování MathML výrazů a jiné mají vkládaný obsah XML jako přídavný software. Tyto prohlížeče však ke své aktivaci vyžadují další rozhraní.

- Jednotlivé nástroje pro generování a interpretaci MathML musí být schopné mezi sebou komunikovat. Byla vyvinuta řada nástrojů MathML, včetně editorů, překladačů, systémů počítačové algebry a jiného vědeckého softwaru, který se stále vyvíjí. Výrazy v MathML jsou často zdlouhavé a z důvodu ručního psaní také náchylné k chybám. Zvláštní důraz je třeba věnovat tomu, aby bylo možné výrazy MathML jednoduše generovat za pomoci uživatelsky vhodných programů. Tyto nástroje by měly být schopné vzájemné spolupráce nezávisle na platformě.

Aktuální informace o nástrojích a aplikacích MathML jsou k dispozici na adrese http://www.w3.org/Math/.

9.1 Vkládání MathML do dokumentů XML a XHTML

MathML lze použít jako samostatný jazyk k výměně matematických výrazů mezi aplikacemi, které jsou schopné MathML zobrazit [21, 22, 23]. Hlavním cílem použití MathML je však kódování matematických vyjádření v rámci větších dokumentů. MathML se jeví jako ideální jazyk pro zakomponování matematických výrazů do jiných aplikací XML.

V dnešní době je kladen důraz především na mechanismy pro vkládání MathML do XHTML, který do jisté míry nahradil starší jazyk HTML 4.

Vkládání výrazů MathML do dokumentů založených na XML a do XHTML je záležitostí řízení jmenných prostorů, tzv. *namespace*. Namespace pro XML je kolekce jmen identifikovaných pomocí URI (Uniform Resource Identifier). URI pro namespace MathML je "http://www.w3.org/1998/Math/MathML".

Pomocí jmenných prostorů je vkládání výrazů MathML do větších XML dokumentů pouze otázkou identifikace značení MathML. Toho lze dosáhnout explicitně identifikací každého názvu prvku MathML buď připojením předpony z prostoru jmen, nebo deklarací nového prostoru jmen na vloženém prvku.

Pro deklaraci jmenných prostorů se využívá atribut `"xmlns"`, který je buď samostatný, nebo s předponou. Když se atribut `"xmlns"` používá samostatně, nastaví výchozí jmenný prostor pro prvek a všechny jeho podvýrazy.

Doporučené návrhy začlenění MathML:

```
01.  <body>
02.    …
03.    <math xmlns="http://www.w3.org/1998/Math/MathML">
04.      <mrow> … </mrow>
05.    </math>
06.    …
07.  </body>
```

nebo

```
01.  <body xmlns:m="http://www.w3.org/1998/Math/MathML">
02.    …
03.    <m:math> <m:mrow> … </m:mrow> </m:math>
04.    …
05.  </body>
```

Tyto dvě metody deklarace jmenných prostorů mohou být použity společně:

```
01.  <body xmlns:m="http://www.w3.org/1998/Math/MathML">
02.    …
03.    <m:math xmlns="http://www.w3.org/1998/Math/MathML">
04.      <mrow> … </mrow>
05.    </m:math>
06.    …
07.  </body>
```

Vložení MathML do XML dokumentu:

```
01.  <?xml version="1.0" encoding="UTF-8"?>
02.  <math xmlns="http://www.w3.org/1998/Math/MathML">
03.    <mrow> … </mrow>
04.  </math>
```

Vložení MathML vztahu do dokumentu XHTML můžeme provést například pomocí prvku `<object>`, přes který zobrazíme matematický vzorec uložený v externím XML souboru. Výsledný kód by mohl vypadat takto:

```
01.  <body>
02.    …
03.    <object data="mathml.xml" type="application/xml">
04.      Váš prohlížeč nepodporuje MathML!
05.    </object>
06.    …
07.  </body>
```

Tyto návrhy samy o sobě nemusí být dostačující pro vytváření funkčních webových stránek, obsahujících MathML značení. Ačkoli je použití MathML v jiných aplikacích XML kompletně popsáno v příslušných doporučeních W3C, stále se objevuje značný pragmatismus kolem celého problému. Podpora XML, jmenných prostorů a chování interpretace v uživatelsky oblíbených aplikacích není vždy plně v souladu s doporučeními W3C. V některých případech software nesplňuje potřebné standardy a v jiných případech nejsou ještě standardy vůbec k dispozici.

Aktuální informace o kompatibilitě a návrzích k implementaci pro současné prohlížeče a další nástroje podporující MathML jsou k dispozici na adrese http://www.w3.org/Math/.

10 Závěr

Problematika publikování matematických vztahů na WWW je velmi rozsáhlá. Předkládaná práce se snažila nastínit základní poznatky o jazyku MathML 2.0, který umožňuje vkládat matematické vztahy do dokumentů na WWW, neslouží však jako uživatelská příručka. K hlubšímu pochopení tohoto jazyka je zapotřebí nastudovat další materiály.

V úvodní části práce jsem zmínil historii jazyka MathML, uvedl jsem základní požadavky na matematické značení a nedostatky při zobrazení matematických vztahů na internetu. Také jsem nastínil princip fungování jazyka XML, jeho přímého předchůdce.

V hlavní části práce jsem se zaměřil na samotný jazyk MathML. Pokusil jsem se vysvětlit jeho základní myšlenky a princip funkce. Popsal jsem hlavní rysy syntaxe MathML a typy značení s jednotlivými prvky.

V závěrečné části práce jsem se zabýval otázkami implementace MathML a zmínil jsem možné způsoby vkládání vztahů MathML do dokumentů XML a XHTML.

Jazyk MathML má značný potenciál a mohl by tak vyřešit stávající komplikace při vytváření, zpracování, výměně a publikování matematických výrazů. Nastává však problém s nedostatečnou podporou internetových prohlížečů, zejména toho nejrozšířenějšího – Internet Explorer 8 a nižší, ve kterém lze MathML výrazy zobrazit pouze pomocí vhodných plug-inů. Tato skutečnost tak odrazuje potenciální uživatele MathML, aby vkládaly matematické výrazy na WWW pomocí tohoto jazyka. Z toho důvodu jsem i já nemohl použít MathML při tvorbě webových stránek kurzu Teoretické mechaniky a musel jsem se spokojit s dnes nejpoužívanějším způsobem vkládání matematických výrazů na internet v podobě obrázků ve formátu GIF nebo JPEG.

Lepší podpory by se v budoucnu mohlo dosáhnout s nástupem prohlížeče Internet Explorer 9. V době psaní této práce je ve vývoji také nová verze MathML 3.0, která přinese další vylepšení současné verze 2.0.

11 Seznam použité literatury

[1] W3C Math Home. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/Math/>.

[2] Mathematical Markup Language (MathML) version 2.0. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/>.

[3] MathML Introduction. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter1.html>.

[4] XML in 10 points. Text v angličtině. [cit. 2010-03-09] Dostupné z WWW: <http://www.w3.org/XML/1999/XML-in-10-points/>.

[5] XML Tutorial. Text v angličtině. [cit. 2010-03-09] Dostupné z WWW: <http://www.w3schools.com/xml/default.asp/>.

[6] Úvod do XML. Text v češtině. [cit. 2010-03-09] Dostupné z WWW: <http://www.kosek.cz/clanky/xml/xml-uvod.html/>.

[7] XML pro každého. Text v češtině. [cit. 2010-03-09] Dostupné z WWW: <http://www.kosek.cz/xml/xmlprokazdeho.pdf/>.

[8] XML pro začátečníky – 1. část. Text v češtině. [cit. 2010-03-09] Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2007030501-xml-pro-zacatecniky-1-cast/>.

[9] Extensible Markup Language. Text v češtině. [cit. 2010-03-09] Dostupné z WWW: <http://cs.wikipedia.org/wiki/Xml/>.

[10] MathML Fundamentals. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter2.html>.

[11] Presentation Markup. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter3.html>.

[12] Content Markup. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter4.html>.

[13] Combining Presentation and Content Markup. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter5.html>.

[14] Characters, Entities and Fonts. Text v angličtině. [cit. 2010-02-15] Dostupné z WWW: <http://www.w3.org/TR/MathML2/chapter6.html>.

[15] The MathML Interface. Text v angličtině. [cit. 2010-02-15]
Dostupné z WWW:
<http://www.w3.org/TR/MathML2/chapter7.html>.

[16] A Gentle Introduction to MathML. Text v angličtině. [cit. 2010-02-18]
Dostupné z WWW:
<http://www.dessci.com/en/reference/mathml/default.htm>.

[17] Český portál EVLM. Příručka učitele – Matematický jazyk MathML 2.0.
Text v češtině. [cit. 2010-02-16] Dostupné z WWW:
<http://147.228.60.216:8080/EVLM/ucitel/13.doc>.

[18] MathML (Mathematical Markup Language). Text v češtině. [cit. 2010-02-16]
Dostupné z WWW:
<http://www.cs.vsb.cz/kot/Reports/Int/index.html>.

[19] K čemu je nám MathML. Text v češtině. [cit. 2010-02-18] Dostupné z WWW:
<http://interval.cz/clanky/k-cemu-je-nam-mathml/>.

[20] MathML 1. část. Text v češtině. [cit. 2010-02-18] Dostupné z WWW:
<http://programujte.com/?akce=clanek&cl=2007102004-mathml-1-cast/>.

[21] Encyklopedie publikačních formátů: MathML. Text v češtině.
[cit. 2010-02-24] Dostupné z WWW:
<http://www.grafika.cz/art/sw/encmathml.html/>.

[22] Podpora MathML v prohlížečích a editorech. Text v češtině.
[cit. 2010-02-24] Dostupné z WWW:
<http://interval.cz/clanky/podpora-mathml-v-prohlizecich-a-editorech/>.

[23] Rozšiřujeme XHTML o MathML. Text v češtině. [cit. 2010-02-24]
Dostupné z WWW:
<http://atd.havrlant.net/rozsirujeme-xhtml-o-mathml/>.

[24] MathML. Text v angličtině. [cit. 2010-02-24] Dostupné z WWW:
<http://en.wikipedia.org/wiki/MathML/>.

[25] Mathematical Markup Language. Text v češtině. [cit. 2010-02-24]
Dostupné z WWW:
<http://cs.wikipedia.org/wiki/MathML/>.