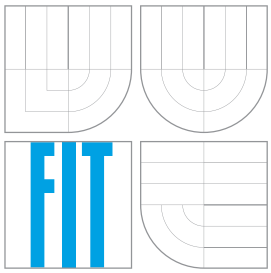
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYNTAKTICKÁ ANALÝZA A VALIDACE DATOVÝCH MODELŮ POPSANÝCH JAZYKEM YANG

PARSER AND VALIDATOR OF DATA MODELS IN YANG LANGUAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVOL VICAN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN KOŘENEK, Ph.D.

BRNO 2016

Zadání bakalářské práce/18047/2015/xvican01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Vican Pavol**
Obor: Informační technologie
Téma: **Syntaktická analýza a validace datových modelů popsaných jazykem YANG**
Parser and Validator of Data Models in YANG Language
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s principy konfigurace prostřednictvím protokolu NETCONF, knihovnami libyang, libnetconf a systémem Netopeer.
2. Nastudujte jazyk YANG a možnosti syntaktické analýzy prostřednictvím nástrojů flex a bison.
3. Navrhněte a implementujte syntaktický analyzátor a validátor pro jazyk YANG použitý pro datové modelování. Při návrhu a implementaci se zaměřte na úsporu paměti tak, aby bylo možné použít systém konfigurace na malých vestavěných zařízeních.
4. Ověřte vlastnosti vytvořené implementace na vybrané sadě datových modelů. Při vyhodnocení se zaměřte na výkonové a paměťové požadavky.
5. V závěru diskutujte dosažené výsledky a navrhněte možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kořenek Jan, Ing., Ph.D.**, UPSY FIT VUT
Konzultant: Krejčí Radek, CESNET
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Administrátori počítačových sietí potrebujú nástroje na konfiguráciu a monitorovanie sieťových zariadení. Z tohto dôvodu bol vytvorený protokol NETCONF, určený na vzdialenú konfiguráciu, a modelovací jazyk YANG, ktorý popisuje štruktúru a sémantiku konfiguračných dát. Cieľom tejto práce je rozšírenie knižnice libyang o syntaktický analyzátor, ktorý spracováva modely popísané jazykom YANG a ukladá ich do interných štruktúr knižnice libyang. Tento analyzátor je vygenerovaný pomocou nástrojov bison a flex.

Abstract

Computer network administrators need tools for configuration and monitoring of network devices. Therefore, NETCONF protocol was designed for remote devices configuration and YANG modelling language which describes the structure of the configuration data. The aim of this thesis is to extend the libyang library with syntax parser, that processes models written in YANG and stores them into internal structures. This parser is generated by bison and flex tools.

Kľúčové slová

YANG, NETCONF, libyang, syntaktická analýza, bison, flex

Keywords

YANG, NETCONF, libyang, syntax analysis, bison, flex

Citácia

VICAN, Pavol. *Syntaktická analýza a validace datových modelů popsaných jazykem YANG.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kořenek Jan.

Syntaktická analýza a validace datových modelů popsaných jazykem YANG

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doktora Jana Kořenka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Pavol Vican
18. mája 2016

Poďakovanie

Na tomto mieste by som chcel poďakovať RNDr. Radkovi Krejčimu za poskytnutú odbornú pomoc a cenné rady pri vypracovaní tejto práce.

Obsah

Kapitola 1

Úvod

V posledných rokoch si môžeme všimnúť obrovský rozvoj v oblasti sieťových technológií. Vďaka tomuto pokroku dnes môžeme internetové pripojenie používať takmer kdekoľvek (napr. v domácnostiach, v obchodoch, reštauráciách a dopravných prostriedkoch). Dostupnosť alebo nedostupnosť tohto pripojenia má veľký vplyv na používateľa, ktorý ho využíva. Preto správcovia sietí musia pružne a rýchlo reagovať pri problémoch v ich spravovaných sietiach. Z tohto dôvodu potrebujú nástroje na konfiguráciu a monitorovanie sietí.

Na prelome 80. a 90. rokov 20. storočia vznikol protokol SNMP (*Simple network management protocol*)[1], ktorý bol navrhnutý na monitorovanie a konfiguráciu sieťových zariadení. Ako sa neskôr ukázalo, na konfigurovanie nebol veľmi vhodný. SNMP nerozlišuje medzi stavovými a konfiguračnými dátami uloženými v zariadení. Pri SNMP dotaze sa môže vrátiť viac ako 500 premenných, v ktorých sa nachádzajú aj konfiguračné dáta. Avšak SNMP nedefinuje konkrétne v ktorých premenných sú uložené tieto dáta [2]. Z tohto dôvodu výrobcovia radšej dali prednosť rôznym proprietárnym nástrojom. Z tejto skutočnosti sa dá usúdiť, že je potreba vytvoriť univerzálny konfiguračný protokol, ktorý by bol jednoduchý na implementáciu a zároveň flexibilný. Tento protokol by mal motivovať výrobcov sieťových zariadení k jeho používaniu.

Organizácia IETF (Internet Engineering Task Force) začala vyvíjať protokol, ktorý by spĺňal tieto požiadavky. V roku 2006 bola publikovaná špecifikácia protokolu NETCONF[3]. Tento protokol je založený na princípe posielaní RPC (Remote Procedure Call) správ. Ide o komunikáciu klient – server, kde sa vymieňajú správy dotaz a odpoveď. Protokol NETCONF priniesol výhody pre jednotnú konfiguráciu rôznych výrobcov zariadení. Ďalšou výhodou tohto protokolu je bezpečnosť, v rámci ktorej používa transportné protokoly, ktoré zaručujú spoľahlivý prenos, autentizáciu, zabezpečenie integrity dát a ich utajenie. Najčastejšie sa používa protokol SSH (*Secure Shell*). Pri vývoji protokolu NETCONF sa začal vyvíjať aj jazyk YANG.

YANG je modelovací jazyk, ktorý bol štandardizovaný organizáciou IETF. Jeho špecifikácia vyšla v roku 2010 v RFC 6020 [4]. Tento jazyk sa používa na modelovanie konfiguračných a stavových dát sieťových prvkov. Tieto dáta sú reprezentované v stromovej štruktúre. Tento typ reprezentácie je veľmi užitočný z dôvodu vyhľadávania. Jazyk YANG umožňuje oddelenie stavových a konfiguračných dát, vytvoriť rôzne syntaktické a sémantické obmedzenia a pod. Taktiež je možné pomocou neho validovať konfiguračné dáta oproti schémam, ktoré obsahujú definície daného modelu. Modely môžu byť uložené vo formáte YIN alebo YANG.

Cieľom bakalárskej práce je doplnenie open-source knižnice libyang o podporu syntaktickej analýzy a validácie jazyka YANG. Knižnica libyang je vyvíjaná organizáciou CES-

NET. Doteraz táto knižnica podporovala len formát YIN, ktorý je založený na XML formáte.

V súčasnosti existuje málo syntaktických analyzátorov pre jazyk YANG. Ako príklad je uvedený pyang[1], ktorý je implementovaný v jazyku Python. Moja implementácia bude v jazyku C, z dôvodu rozšírenia knižnice libyang.

Štruktúra práce bude rozdelená do šiestich kapitol. Prvá kapitola sa zaoberá popisom modelovacieho jazyka YANG a protokolu NETCONF. V druhej kapitole sa popisujú princípy lexikálnej a syntaktickej analýzy. V tejto kapitole je tiež spomenuté použitie nástrojov na vytvorenie lexikálneho a syntaktického analyzátora. V tretej kapitole je rozoberaná architektúra knižnice libyang a jej rozšírenie o podporu formátu YANG. V štvrtej kapitole je popísaná implementácia rozšírenia knižnice. V predposlednej kapitole je načrtnuté testovanie implementovaného rozšírenia a v poslednej kapitole je zobrazená analýza dosiahnutých výsledkov.

[1] https://github.com/mbj4668/pyang

Kapitola 2

Konfigurácia zariadení pomocou NETCONF a YANG

2.1 Protokol NETCONF

Protokol NETCONF bol štandardizovaný v roku 2006 v RFC 4741[3]. Jeho súčasná revízia je popísaná v RFC 6241[5]. Funguje na princípe RPC. Ide o komunikáciu klient – server, kde sa vymieňajú správy dotaz a odpoveď. Klient v tomto zmysle je aplikácia alebo zariadenie, odkiaľ posielame RPC dotazy. Server je zariadenie, ktoré chceme konfigurovať. Táto komunikácia je posielaná v textovej forme, ktorá je zakódovaná v XML (*Extensible Markup Language*). Pomocou tohto serializačného formátu môžeme jednoducho zasielať hierarchické štruktúry.

Vnútri volania RPC je XML dokument obsahujúci príkaz pre server, ktorý požaduje vykonanie určitej operácie. Po vykonaní príkazu server odošle odpoveď klientovi, v ktorej môže byť potvrdenie o bezchybnom vykonaní príkazu alebo informácie, ktoré sme potrebovali získať. V prípade chyby je server povinný ohlásiť chybu z pevne danej množiny chýb. Na obrázku 2.1 môžeme vidieť operáciu get-config v RPC, ktorá získava počiatočnú konfiguráciu sieťového zariadenia.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get-config>
    <startup/>
  </get-config>
</rpc>
```

Obr. 2.1: RPC dotaz na získanie konfigurácie

Ak na serveri nastane asynchrónna udalosť, server nemôže poslať túto informáciu klientovi pomocou mechanizmu RPC. Preto bolo potrebné vytvoriť mechanizmus, ktorý by dovoľoval posielať tieto informácie. Tento mechanizmus poznáme pod pojmom asynchrónne správy a v protokole NETCONF pod pojmom notifikácie.

Hlavnou výhodou protokolu NETCONF je jeho rozšíriteľnosť. Samotný protokol obsahuje malú množinu základných operácií, ktoré väčšinou pracujú s konfiguráciami ako s celkami. Umožňujú ich vytvárať, kopírovať alebo mazať. Táto malá množina operácií

pomáha podporovať všetky zariadenia, ktoré vedia komunikovať cez protokol NETCONF. Ďalšie špeciálne operácie (napríklad čiastočne uzamknutie aktuálnych konfiguračných dát) sa pridávajú ako schopnosti (anglicky *capabilities*).

2.2 Jazyk YANG

Pre vyjadrenie štruktúry a sémantiky konfiguračných dát bolo potrebné vytvoriť modelovací jazyk, ktorý má byť jednotný pre všetkých výrobcov sieťových zariadení. Z tohto dôvodu vznikol modelovací jazyk YANG. Bol štandardizovaný v roku 2010 v RFC 6020. Cieľom tohto jazyka je vytvoriť modely (resp. schémy) pre konfiguráciu sieťových zariadení pomocou protokolu NETCONF, ktoré sú zrozumiteľné pre človeka.

Tento jazyk vznikal súčasne s protokolom NETCONF. Z tohto dôvodu obsahuje prvky, ktoré sú špecifické pre protokol NETCONF (notifikácie a RPC). Ale v súčasnosti vznikol tlak na využívanie jazyka YANG mimo protokolu NETCONF. Je to napríklad protokol RESTCONF, ktorý zatiaľ ešte nie je štandardizovaný.

Modul v jazyku YANG obsahuje štruktúru dát, ktoré popisujú konfiguračné a stavové dáta, RPC alebo notifikácie. Je identifikovaný s jeho menom a URI (Uniform Resource Identifier). Modul je možné rozdeliť na jednotlivé logické celky. Jeden takýto logický celok sa nazýva submodul. Modul aj submodul môže importovať iné moduly. Táto vlastnosť je veľmi užitočná pre opätovné použitie už definovaných nových dátových typov, groupingov, identít a pod.

YANG definuje sadu vstavaných typov. Tieto typy sú zobrazené v nasledujúcej tabuľke.

Názov	Popis
binary	Binárne dáta
bits	Sada bitov alebo príznaky
boolean	"true" alebo "false"
decimal64	64-bit desatinné číslo
empty	List, ktorý nemá žiadnu hodnotu
enumeration	Vymenované reťazce
identityref	Odkaz na abstraktnú identitu
instance-identifier	Odkaz na uzol v dátovej stromovej štruktúre
int8	8-bit znamienkové celé číslo
int16	16-bit znamienkové celé číslo
int32	32-bit znamienkové celé číslo
int64	64-bit znamienkové celé číslo
leafref	Odkaz na instanciu listu
string	Čitateľný reťazec
uint8	8-bit neznamienkové celé číslo
uint16	16-bit neznamienkové celé číslo
uint32	32-bit neznamienkové celé číslo
uint64	64-bit neznamienkové celé číslo
union	Výber z členských typov

Tabuľka 2.1: Sada vstavaných typov

Existuje aj mechanizmus na definovanie nových typov pomocou kľúčového slova *typedef*. Tieto typy sú odvodené od základného typu. Súčasne pri definovaní typu v *leaf*, *leaf-*

list alebo *typedef* umožňuje YANG obmedziť údajový typ. Napríklad pri reťazci sa môže obmedziť jeho dĺžka alebo musí odpovedať danému vzoru, ktorý je určený regulárnym výrazom.

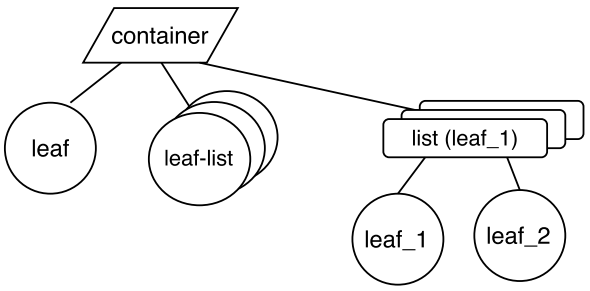
```
typedef domain-name {
  type string {
    pattern
      '((([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.)*'
    + '([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.?)'
    + '|\.';
    length "1..253";
  }
}
```

Obr. 2.2: Schéma definovania nového dátového typu a obmedzenia

Každý uzol v jazyku YANG má svoje meno a obsahuje hodnotu alebo sadu uzlov. Tieto uzly môžeme rozdeliť do dvoch skupín. Do prvej skupiny patria dátové uzly, ktoré sa vyskytujú v dátovom strome. Druhú skupinu tvoria špeciálne uzly, ktoré uľahčujú vytváranie modelov (schém) v jazyku YANG. Medzi dátové uzly patrí:

- *container* - obsahuje potomkov, ale nemá vlastnú hodnotu
- *leaf* - obsahuje práve jednu hodnotu konkrétneho dátového typu a nemá potomkov
- *leaf-list* - má rovnaké vlastnosti ako uzol leaf, pričom v dátovom strome sa môže vyskytnúť viacero jeho instancií
- *list* - v dátovom strome sa môže vyskytnúť viacero instancií tohto uzlu. Na rozlíšenie týchto instancií sa využíva kľúč, ktorý sa odkazuje len na uzol leaf, ktorý je potomkom tohto uzlu.
- *anyxml* - používa sa na reprezentáciu neznámej časti XML, ktorý nie je popísaný dátovým modelom
- *rpc* - definovanie RPC pre protokol NETCONF
- *notification* - definovanie notifikácií pre protokol NETCONF

Na obrázku 2.3 môžeme vidieť možné usporiadanie spomenutých uzlov.



Obr. 2.3: Štruktúra možného usporiadania uzlov

Do skupiny špeciálnych uzlov patrí : *choice*, *grouping*, *augment*, *deviation*.

Uzol *choice* pomáha vytvoriť usporiadanie uzlov do viacerých variánt, avšak v jednom okamihu môže byť vybraná len jedna varianta. Pre prácu s viacerými uzlami slúži ďalší uzol *grouping*. Používa sa na zoskupenie dátových uzlov do jedného virtuálneho uzlu. Tento virtuálny uzol je možné opätovne použiť pri vytváraní nových uzlov v YANG module.

Pre úpravu existujúcich uzlov nám jazyk YANG ponúka dva uzly. Uzol *augment* pridáva nové poduzly a uzol *deviation* dokáže meniť vlastnosti existujúceho uzlu.

Moduly písané vo formáte YANG je možné preložiť do ekvivalentného formátu YIN (YANG Independent Notation). Základom formátu YIN je formát XML. Konverzia z YANG do YIN je bezstratová, z toho vyplýva, že sa dá konvertovať z formátu YIN naspäť na formát YANG.

Definícia sémantiky a syntaxe jazyka YANG je popísaná v štandarde RFC 6020. Táto gramatika je zapísana v tvare ABNF (Augmented Backus–Naur Form), ktorá je definovaná v štandarde RFC 5234 [6]. Táto špecifikácia je doporučená v internetových špecifikáciách RFC a IETF.

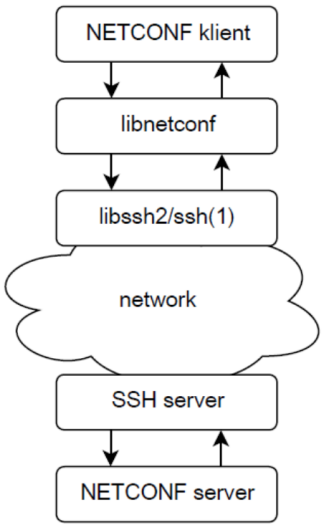
2.3 Systém Netopeer

Systém Netopeer [7] je zameraný na vzdialenú konfiguráciu zariadení. Skladá sa z dvoch častí:

a) klient (GUI aplikácia, CLI konzolová aplikácia)

b) server

Klient Netopeer komunikuje so serverom pomocou knižnice libnetconf, ktorá je napísaná v jazyku C. Táto knižnica implementuje protokol NETCONF a poskytuje základné funkcie na pripojenie, odoslanie a prijatie správ a prácu s konfiguračnými dátami. Pre spojenie používa transportný protokol SSH. Komunikácia tohto systému je znázornená na obrázku 2.4.



Obr. 2.4: Zobrazenie komunikácie pomocou knižnice libnetconf

Kapitola 3

Syntaktická a lexikálna analýza

3.1 Princíp lexikálnej analýzy

Lexikálna analýza slúži na rozdelenie vstupnej postupnosti znakov zo zdrojového súboru na lexikálne jednotky (napr. idendifikátory, čísla, kľúčové slová a pod.). Tieto lexikálne jednotky (lexémy) sú reprezentované vo forme tokenov, ktoré sú poskytnuté syntaktickej analýze.

K popisu lexikálnych jednotiek sa zvyčajne používajú regulárne výrazy. Pomocou nich sme schopný popísať množinu reťazcov, ktoré budú patriť do jednej skupiny. Napríklad identifikátor v jazyku YANG začína malým alebo veľkým ASCII písmenom alebo podčiarkovníkom. Za ním nasledujú 0 až N-krát malé alebo veľke ASCII písmená, číslice, podčiarkovník, pomlčka alebo bodka. Tento príklad môžeme popísať pomocou nasledovného regulárneho výrazu `[a-zA-Z_][a-zA-Z0-9_\-.]*`. Regulárne výrazy môžeme jednoducho previesť na konečné automaty, ktoré sa ľahko implementujú.

3.2 Princíp syntaktickej analýzy

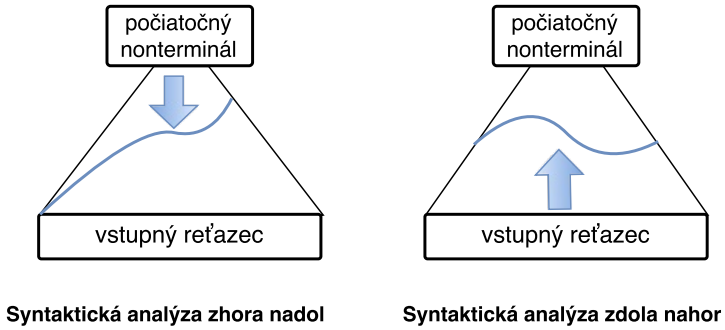
Syntaktická analýza je proces analýzy sekvencie tokenov, ktoré musia dodržiavať pravidlá predom danej gramatiky. Tokeny sú generované v lexikálnej analýze. Syntaktická analýza používa jednu z dvoch základných metód:

a) **syntaktická analýza zhora nadol**

   Táto metóda analýzy začína v počiatočnom nonterminále. Následne sa tento nonterminál začne deliť na ďalšie nonterminály podľa pravidiel, pokiaľ sa nedostane k terminálnemu symbolu. Potom sa môže porovnať so vstupom a vyhodnotiť, či sa zhoduje.

b) **syntaktická analýza zdola nahor**

   Táto metóda najskôr načíta terminál a snaží sa nájsť pravidlá, ktoré obsahujú tento terminál. Po nájdení pravidla sa nahradí celá pravá stranu pravidla za nonterminál z ľavej strany pravidla. Tento postup sa opakuje, pokiaľ sa nedostane do počiatočného nonterminálu.



Obr. 3.1: metódy syntaktickej analýzy

3.3 Použité nástroje

V definícii jazyka YANG si môžeme všimnúť, že jeho gramatika je dosť rozsiahla. Jej rozsah je približne 1000 riadkov. Pri takejto veľkej gramatike sa už oplatí využiť nástroj (aplikáciu), ktorá by vedela vytvoriť syntaktický analyzátor na danú gramatiku. Pri hľadaní vhodného nástroja sa vychádzalo, že gramatika je zadaná vo formáte ABNF.

APG - ABNF Parser Generator

Nástroj APG - ABNF Parser Generator [8] dokáže generovať syntaktický analyzátor z gramatiky vo formáte ABNF do jazyka C. Je voľne dostupný a jeho zdrojové súbory sa nachádzajú na githube. Pri inicializácii syntaktický analyzátora bola spotreba pamäte vyše 56 KB. Z tohto dôvodu bolo potrebné nájsť lepší generátor syntaktického analyzátora. Preto bol zvolený známy generátor GNU Bison, ktorý je nástupca generátoru Yacc (Yet Another Compiler Compiler).

GNU Bison

GNU Bison[1]je nástroj pre generovanie rôznych LR syntaktických analyzátorov z bezkontextových gramatík [9]. Používa metódu syntaktickej analýzy zdola nahor, ktorá je popísaná v podkapitole 3.2. Pravidlá gramatiky sú písané vo formáte so základom BNF (Backus-Naurov form). Výstupom je syntaktický analyzátor v jazyku C.

Formát vstupného súboru pre GNU Bison je veľmi špecifický. Jeho popis môžeme nájsť v jeho manuálových stránkach. Jeho základná štruktúra je nasledovná:

```
  Prológ
%%
  Pravidlá gramatiky
%%
  Epilóg
```

V prológu sa môže nachádzať kód v jazyku C obsahujúci deklarácie, ktorý je uzavretý do znakov %{ a %}. Taktiež sú tu definície tokenov, názvy a dátové typy symbolov. Tieto tokeny sa využívajú aj v lexikálnej analýze. V sekcii pravidlá gramatiky sa nachádzajú

[1]https://www.gnu.org/software/bison/

pravidlá, podľa ktorých bison vytvorí syntaktickú analýzu. Každé pravidlo začína s nonterminálnym symbolom a dvojbodkou. Za nimi nasleduje zoznam nonterminálov, tokenov a akcií. Posledná sekcia epilóg obsahuje pomocné procedúry a akcie napísane v jazyku C.

Flex

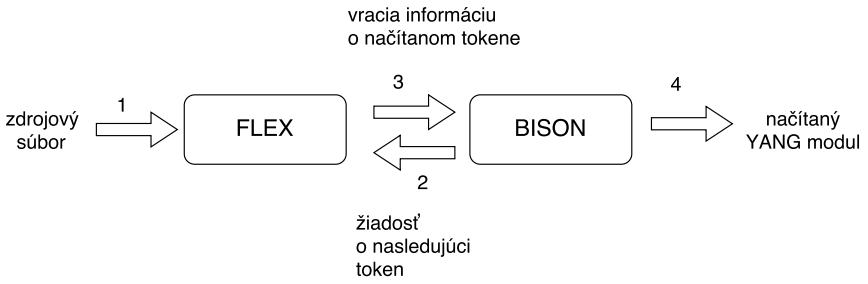
Flex (fast lexical analyzer)[2] je nástroj na generovanie lexikálneho analyzátora. Lexikálna analýza všeobecne pracuje na princípe vyhľadávaní znakových vzorov na vstupe. Jednoduchý spôsob, ako opísať tieto vzory, je využiť regulárne výrazy. Ide o rovnaký spôsob vyhľadávania ako využíva vyhľadávací program *egrep* v UNIXE.

Flexový zdrojový súbor sa skladá z regulárnych výrazov a akcií, ktoré sa majú vykonať pri zhode s regulárnym výrazom. Hľadá sa najdlhšia zhoda regulárneho výrazu. Regulárne výrazy sa musia usporiadať od špecifických až k všeobecným. Snažíme sa predísť situácii, keď sa vykoná akcia všeobecnejšieho regulárneho výrazu a nevykoná sa akcia špecifickejšieho regulárneho výrazu. Flex z týchto regulárnych výrazov si vytvorí interne svoj deterministický konečný automat. Tento automat je rýchly a jeho rýchlosť vyhľadávania nezávisí od počtu alebo komplexnosti regulárneho výrazu.

Formát vstupného súboru sa skladá z 3 častí.

```
  Prológ
%%
  Sekcia pravidiel
%%
Epilóg
```

Prológ slúži na deklaráciu premenných, možnosti nastavenia lexikálnej analýzy a deklaráciu počiatočných stavov. Za nimi nasleduje sekcia pravidiel, v ktorej sú definované lexémy pomocou regulárnych výrazov a akcie, ktoré sa majú vykonať po načítaní danej lexémy. Každá lexéma musí byť definovaná na novom riadku. V epilógu sa nachádzajú pomocné funkcie a akcie napísané v jazyku C.



Obr. 3.2: Komunikácia medzi bisonom a flexom

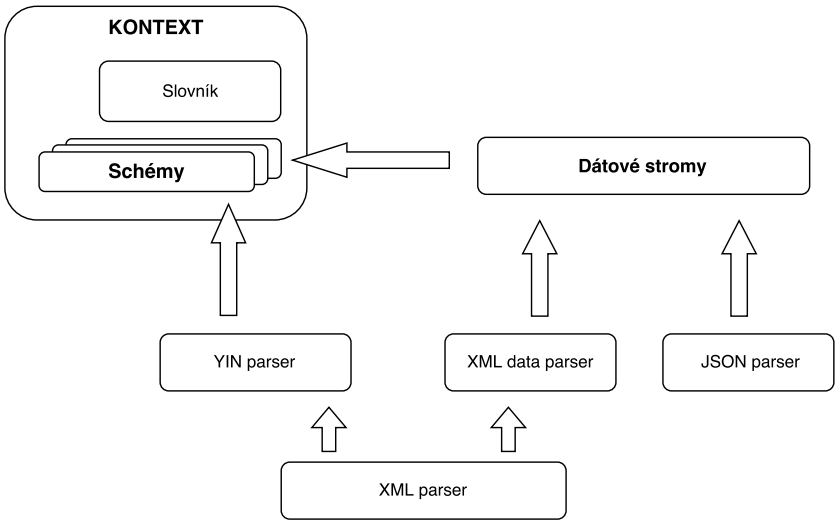
Na obrázku 3.2 je zobrazená komunikácia medzi bisonom a flexom. Na začiatku komunikácie sa načíta vstupný súbor s YANG modulom do reťazca (1). Následne sa začne vykonávať syntaktická analýza. Bison si vyžiada token od flexu (2). Flex prečíta lexému z reťazca a vráti informáciu o type načítaného tokenu (3). Tento cyklus sa opakuje, pokiaľ sa syntaktická analýza neskončí. Po úspešnej syntaktickej analýze je uložený YANG modul do kontextu knižnice libyang (4).

[2] http://flex.sourceforge.net/

Kapitola 4

Návrh knižnice libyang

Návrh rozšírenia knižnice libyang o podporu formátu YANG vychádza už z návrhu architektúry tejto knižnice. Táto knižnica pozostáva z 3 podstatných komponent: kontext, schémy a dátové stromy. Prepojenie týchto komponent zobrazuje obrázok 4.1.



Obr. 4.1: Architektúra knižnice libyang

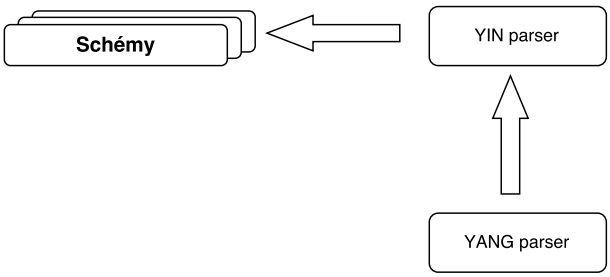
Kontext je hlavná komponenta, do ktorej sa ukladajú dáta z komponent slovníka a schém. Taktiež obsahuje ukazovatele na komponentu **dátové stromy**. V knižnici libyang je dovolené pracovať s viacerými instanciami kontextu. Pri definovaní instancie sa môže nastaviť cesta, v ktorej sa budú hľadať YANG moduly. Pri požiadavke na načítanie modulu sa hľadá najskôr v aktuálnom pracovnom priečinku. Ak sa tu nenachádza daný modul, tak sa pokúsi ešte hľadať v špecifickej ceste, ktorá bola zadaná pri instancii kontextu alebo nastavená funkciou `ly_ctx_set_searchdir()`. Kontext navyše obsahuje špecifickú komponentu slovník.

Slovník sa používa na ukladanie všetkých reťazcov. Pomocou neho sa jednoduchšie spravujú reťazce, zbytočne sa nealokuje pamäť pre rovnaký reťazec, ktorý je už vložený v slovníku. Slovník funguje na princípe hashovacej tabuľky. V tabuľke sú záznamy uložené v jednosmernom zozname.

Dátové stromy je komponenta, ktorá obsahuje funkcie na načítanie dát z XML a JSON formátu a ich validáciu. Alternatívny formát JSON je použitý pre protokol RESTCONF. Na načítanie dát je použitá komponenta XML data parser alebo JSON parser. Knižnica libyang povoľuje načítanie len takých dát, ktoré sa dajú prepojiť s niektorou schémou. Táto schéma už musí byť načítaná v danom kontexte knižnice libyang.

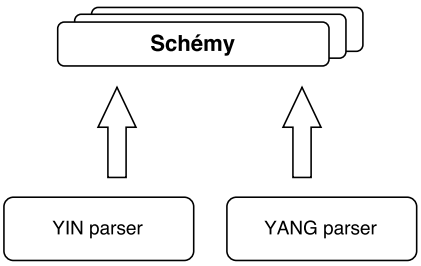
Schémy obsahujú YANG moduly, ktoré boli načítané zo schém vo formáte YIN. Na načítanie modulov využíva komponentu YIN parser, ktorá kontroluje správnosť syntaxe formátu YIN. Táto komponenta zároveň využíva ďalšiu komponentu XML parser, keďže formát YIN je založený na formáte XML. Tieto schémy sa nedajú vytvoriť ani zmeniť programovo, pretože sú trvalo prepojené s kontextom dátového stromu. Používajú sa len na prístup k definíciám dátového modelu.

Prvotný návrh riešenia obsahuje konvertovanie formátu YANG do formátu YIN a použil sa už implementovaný analyzátor formátu YIN. Výhodou tohto riešenia je jednoduché konvertovanie medzi týmito dvoma formátmi. Veľkou nevýhodou je veľká spotreba pamäte.



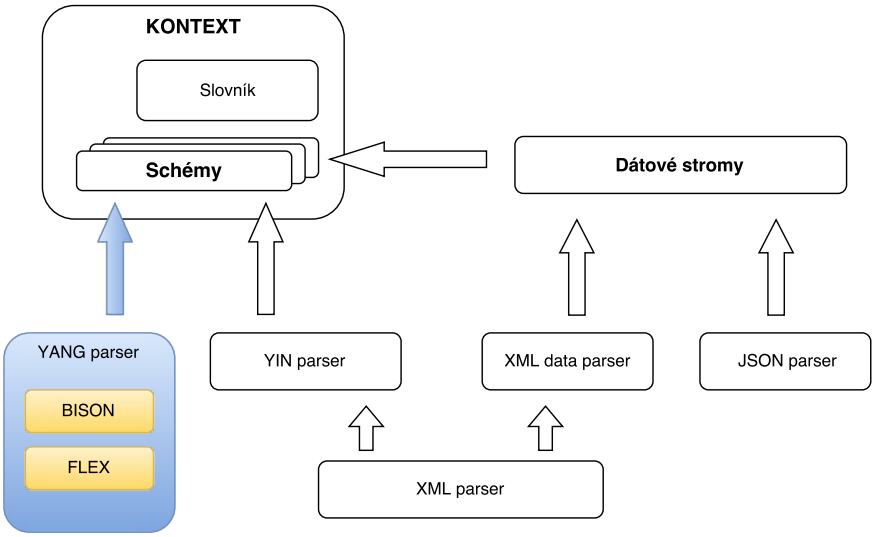
Obr. 4.2: Prvý návrh riešenia

Podľa ďalšieho návrhu by sa YANG rovno analyzoval a ukladal do interných štruktúr. Pri tomto návrhu bolo potrebné si uvedomiť zložitosť gramatiky jazyka YANG. Výhoda tohto návrhu spočíva v rýchlosti a menšej spotrebe pamäti v porovnaní s prvým návrhom.



Obr. 4.3: Druhý návrh riešenia

Zvolený bol druhý návrh, ktorý má menšie nároky na pamäť. Je veľký predpoklad, že bude táto knižnica používaná v zariadeniach, ktoré majú obmedzenú veľkosť RAM pamäte. Výsledná architektúra knižnice libyang, ktorá zahŕňa aj toto riešenie, je zobrazená na obrázku 4.4.



Obr. 4.4: Výsledná architektúra knižnice libyang

Kapitola 5

Implementácia

5.1 Zmena formátu gramatiky

Na začiatku implementácie je potrebné upraviť gramatiku jazyka YANG z ABNF formátu do formátu, ktorý vyhovuje bisonu. Pri tejto úprave som narazil na zopár problémov:

- ABNF gramatika používa pre pravidlá, ktoré sa majú opakovať nasledovnú notáciu:

  ```
  <a>*<b>pravidlo
  ```

  kde <a> a <b> sú celé neznamienkové čísla. <a> reprezentuje minimálny počet opakovaní a <b> maximálny počet opakovaní pravidla. Implicitné hodnoty sú 0 a nekonečno. Takéto pravidlo sa v bisone dá napísať jedine pomocou rekurzívneho pravidla.

- V gramatike sa nachádzajú pravidlá, ktoré je potrebné upraviť. Sú to napríklad pravidlá *refine-container-stmts*, *refine-leaf-stmts* a pod. Definície týchto pravidiel sú zobrazené na obrázku 5.1.

  ```
  refine-container-stmts =
                       *(must-stmt stmtsep)
                       [presence-stmt stmtsep]
                       [config-stmt stmtsep]
                       [description-stmt stmtsep]
                       [reference-stmt stmtsep]

  refine-leaf-stmts    =
                       *(must-stmt stmtsep)
                       [default-stmt stmtsep]
                       [config-stmt stmtsep]
                       [mandatory-stmt stmtsep]
                       [description-stmt stmtsep]
                       [reference-stmt stmtsep]
  ```

  Obr. 5.1: Definície pravidiel refine

  Tieto pravidlá môžu začínať rovnakým terminálnym symbolom, pričom sa vyskytujú v inom kontexte. Syntaktická analýza nie je schopná rozoznať, ku ktorému kontextu

nadradeného uzlu patrí dané pravidlo. Preto vzniká nejednoznačnosť výberu pravidla. Z tohto dôvodu boli upravené tieto pravidlá na jedno generické pravidlo. A súčasne pri sémantickej kontrole sa kontroluje, či pravidlo vyhovuje vybranému typu uzlu.

- Pridanie nových pravidiel do gramatiky, ktoré vyplývajú z textu RFC 6020. V pôvodnej gramatike nie je napríklad zahrnuté spájanie reťazcov pomocou operátora +.
- Niektoré pravidlá môžu mať prehodené poradie. Napríklad v pravidle *module-header-stmts*, ktoré je zobrazené na obrázku 5.1, môže byť až 6 kombinácií usporiadania pravidiel.

```
module-header-stmts =
                ;; tieto pravidlá sa môžu objaviť v ľubovoľnom poradí
                [yang-version-stmt stmtsep]
                 namespace-stmt stmtsep
                 prefix-stmt stmtsep
```

Obr. 5.2: Definícia pravidla module-header-stmts

Problém nastáva pri jednoznačnosti výberu pravidla. Tento problém sa dá vyriešiť 2 spôsobmi:

a) vytvoriť všetky kombinácie možných pravidiel. Tento spôsob je nepraktický, lebo počet pravidiel bude prudko narastať.

b) vytvoriť si rekurzívne pravidlo, ktoré nám zaručí túto funkcionalitu

Zvolil som druhý spôsob, ktorý je výhodnejší pri implementácii tohto riešenia.

5.2 Problém dopredných referencií

Jazyk YANG umožňuje použitie prvkov ako sú grouping, typedef a pod., ktoré ešte neboli definované v YANG module. Na obrázku 5.2 môžeme vidieť príklad na doprednú referenciu. Pri syntaktickej analýze sa číta zdrojový súbor s YANG modulom zhora nadol. To znamená, že uzol leaf vyžaduje referenciu na dátový typ ipv4-address. Tento typ nebol ešte analyzovaný, keďže jeho definícia sa nachádza pod uzlom leaf. Z tohto dôvodu je potrebné si uložiť informáciu o neúspešnom prepojení referencie na daný dátový typ, aby bolo možné po načítaní celého modulu vykonať kontrolu týchto referencií.

```
container address {
  leaf ip {
    type ipv4-address;
    config true;
    description "The IPv4 address on the interface.";
  }
```

```
  typedef ipv4-address {
    type string {
      pattern '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
              + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
    }
  }
}
```

Obr. 5.3: Ukážka kódu dopredných referencií

Na riešenie dopredných referencií bola vytvorená štruktúra `unres_schema`. Jej definícia je zobrazená na obrázku 5.4. V tejto štruktúre sa nachádzajú polia, do ktorých sa

```
struct unres_schema {
    void **item;
    enum UNRES_ITEM *type;
    void **str_snode;
    struct lys_module **module;
    uint32_t count;
};
```

Obr. 5.4: Schéma štruktúry unres_schema

ukladajú informácie na vyriešenie dopredných referencií. Do poľa `item` sa ukladá ukazovateľ na internú štruktúru, ktorá reprezentuje uzol, v ktorej sa našla dopredná referencia. Do poľa `type` sa ukladá typ doprednej referencie (napríklad UNRES_TYPE_DER, UNRES_TYPE_DFLT a pod.). Táto informácia je veľmi potrebná pri riešení dopredných referencií po načítaní celého modulu. Do poľa `str_snode` sa ukladá ukazovateľ na reťazec alebo ukazovateľ na rodičovský uzol, v ktorom sa našla referencia. V prípade reťazca je tu uložená informácia, v ktorej sa môže nachádzať napríklad východzia hodnota uzlov leaf, choice a pod. Inak je tu uložený ukazovateľ na rodičovský uzol, ktorý sa využíva na vynorovanie z uzlov až na globálnu úroveň pri hľadaní definícií dátových typov a pod. Predposledná položka `module` uchováva informáciu, v ktorom module sa našla dopredná referencia. Posledná položka štruktúry je počítadlo, ktoré určuje veľkosť polí. Pre manipuláciu s touto štruktúrou sú implementované funkcie, ktoré začínajú predponou `resolve`.

5.3 Analýza dátových typov

Jazyk YANG dovoľuje pri zadávaní dátového typu v uzloch leaf, leaf-list a typedef vytvoriť obmedzenia. Pri syntaktickej analýze formátu YIN sa tieto obmedzenia analyzujú až po nájdení referencie na správny dátový typ. Tento spôsob riešenia nie je vhodný pre formát YANG. Použitá lexikálna analýza pre formát YANG nedokáže preskakovať časti textu v zdrojovom súbore, preto je nutné zanalyzovať ich už v tomto okamihu.

K tomuto účelu bola vytvorená pomocná štruktúra `yang_type`, pričom jej štruktúra je zobrazená na obrázku 5.5. Táto štruktúra si uchováva názov dátového typu a ukazovateľ na internú štruktúru `lys_type`, do ktorej sa ukladajú informácie o dátovom type. Taktiež sa tu nachádza príznak na odlíšenie od pomocnej štruktúry pre formát YIN. Tento príznak

je nutný, keďže pri riešení dopredných referencií nie je známe, či sa analyzuje formát YIN alebo formát YANG. Nakoniec je tu premenná base, do ktorej sa ukladá odhadnutý dátový typ.

```
struct yang_type {
  char flags;
  LY_DATA_TYPE base;
  const char *name;
  struct lys_type *type;
};
```

Obr. 5.5: Schéma štruktúry yang_type

Pri analýze obmedzení nemusí byť jasné, ku ktorému dátovému typu patria obmedzenia. Z gramatiky môžeme zistiť, že niektoré obmedzenia sú určené práve k jednému dátovému typu. Je to napríklad obmedzenie *path* pre dátový typ *leafref*. Naopak niektoré obmedzenia môžu patriť viacerým dátovým typom. Napríklad dátový typ *string* môže obsahovať obmedzenie na dĺžku a vzor reťazca. Ale aj dátový typ *binary* môže obsahovať obmedzenie na dĺžku. Pri odhadovaní typu sa zvolí ten typ, ktorý pokryje obidva prípady. V tomto príklade sa zvolí dátový typ *string*.

Po analýze obmedzení sa syntaktická analýza pokúsi zistiť, či nájde referenciu na daný dátový typ, ktorého meno sa nachádza v položke name v štruktúre `yang_type`. Ak ho našiel, skontroluje sa odhadovaný typ od skutočného typu. Pri nezhode typov sa ešte overuje, či sa definované obmedzenia dajú aplikovať na skutočný typ. Po zlyhaní tohto overenia sa vyvolá chybová správa a ukončí sa syntaktická analýza. Inak sa nastaví odhadovaný typ na skutočný typ a skontroluje sa sémantika obmedzení.

5.4 Úprava reťazca v úvodzovkách

Z RFC 6020 vyplýva, že reťazce v úvodzovkách môžu mať odsadenie, ktoré nemá byť súčasťou textu. Na odsadenie sa využívajú znaky tabulátorov alebo medzier. Pri použití tabulátora sa musí tento znak nahradiť za 8 znakov medzier. Príklad rôzneho formátovania reťazca je zobrazený na obrázku 5.6.

```
description "Toto je ukazkovy text,
             ktory sa krasne formatuje.";

description "Toto je ukazkovy" +
               " text,
                ktory sa krasne formatuje.";

description "Toto je ukazkovy" +" text,\n" + "ktory sa krasne formatuje.";

description "Toto je ukazkovy text,\nktory sa krasne formatuje.";
```

Výsledný tvar reťazca:

```
description "Toto je ukazkovy text,
             ktory sa krasne formatuje.";
```

Obr. 5.6: Rôzne typy zápisu reťazca

Na začiatku sa nahradia znaky, ktoré majú špeciálny význam. Sú to nasledovné znaky: \t, \\, \n, \". Následne si spočítam počet výskytov tabulátora v reťazci. Tento počet je potrebný pri alokovaní miesta pre nový reťazec. Veľkosť alokácie nového reťazca je nasledovná:

```
veľkosť starého reťazca + 7 * počet výskytov tabulátorov
```

Vieme, že tabulátor sa má nahradiť za 8 medzier. Vo veľkosti starého reťazca je už započítaná jedna medzera, ktorá sa nahradí tabulátorom. Stále však chýba priestor pre sedem medzier. Preto je vo vzorci na výpočet veľkosti nového reťazca násobok 7.

Po alokácii miesta sa môže začať kopírovanie reťazca. Kopíruje sa znak po znaku, pričom sa zapamätáva, kde začína medzera. Ak sa narazí na nový riadok, skontroluje sa, či sa našli medzery na konci riadku. Ak áno, posunie sa index v reťazci o počet medzier späť, čo zaručí ich vymazanie. Následne sa zavolá funkcia `read_indent()`, ktorá prečíta požadované odsadenie. Tento postup sa opakuje dovtedy, pokiaľ sa neprejde celý reťazec. Na konci sa ešte upraví veľkosť pamäte reťazca pomocou funkcie `realloc()`.

5.5 Importovanie modulov

Jazyk YANG obsahuje mechanizmus k importovaniu modulov. Tento mechanizmus je užitočný pre využitie už definovaných rozšírení, groupingov, nových typov a pod. Používa sa na to príkaz *import*. Argument tohto príkazu je názov modulu, ktorý chceme importovať. Navyše sa môže špecifikovať, v akej revízii musí byť importovaný modul.

Pre ľahšie vytvorenie a editáciu modulov umožňuje jazyk YANG rozdeliť modul do samostatných logických celkov, ktoré sa nazývajú submoduly. Každý submodul musí patriť práve jednému modulu. Pre zjednotenie týchto submodulov sa používa príkaz *include* v hlavnom module. Argument tohto príkazu je názov submodulu, ktorý sa má vložiť.

Vďaka problému, ktorý nastal počas testovania, sa zistilo, že sa nemenil kontext pre bison a flex pri analýze importovaného modulu. Prejavovalo sa to tým, že po syntaktickej analýze importovaného modulu nebolo možné pokračovať v syntaktickej analýze predchádzajúceho modulu. V lexikálnej analýze boli definované statické dáta, ktoré obsahovali načítaný zdrojový súbor importovaného modelu. Preto bolo potrebné pridať prepínače do bisonu aj flexu, ktoré zaručia vytvorenie kontextu.

Flex ponúka prepínač *reentrant*, ktorý zmení statické premenné na dynamické. Táto zmena je potrebná, aby bolo možné meniť kontext. Taktiež musel byť pridaný prepínač *bison_bridge*. Tento prepínač sa stará o prepojenie medzi flexom a bisonom. Bez tohto prepínača, flex očakáva vo svojej funkcii `yylex()` ako prvý parameter ukazovateľ na instanciu kontextu, avšak bison vkladá ako prvý parameter premennú `yylval`, ktorá slúži na komunikáciu s flexom.

Bison taktiež ponúka prepínač *api.pure*, ktorý slúži k zmene kontextu. Pre spustenie syntaktickej analýzy slúži funkcia `yyparse()`. V prípade, keď je potrebné posunúť argument

do tejto funkcie, je nutné zadať prepínač *parse-param*. Pri zadávaní viacerých argumentov sa prepínače musia umiestniť samostatne na riadok. Taktiež, keď je nutné posunúť argument do funkcie `yylex()`, musí sa definovať ďalší prepínač *lex-param*.

V nasledujúcej tabuľke sú uvedené všetky prepínače, ktoré boli použité pre vytvorenie a použitie kontextu.

Program	Prepínač
flex	reentrant bison_bridge
bison	parser-param lex-param api.pure

Tabuľka 5.1: Použité prepínače na vytvorenie kontextu

5.6 Rozšírenia

Jazyk YANG ponúka mechanizmus na vytvorenie vlastných uzlov a vlastnosti uzlov. Definujú sa pomocou rozšírení (anglicky extension). Nové definované rozšírenie môže byť importované a používané v iných moduloch.

Tieto rozšírenia sa ťažko analyzujú. Jeho názov sa skladá s prefixu modulu, v ktorom je definovaný, a názov rozšírenia. Z tohto dôvodu sa nedá vytvoriť gramatika pre jasné definovanie rozšírení. Navyše chovanie rozšírenia sa dá definovať v popisu (anglicky description) rozšírenia. Knižnica libyang rozpoznáva rozšírenia default-deny-all, default-deny-write a get-filter-element-attributes. Ostatné rozšírenia budú ignorované. Pre pridávanie podpory iných rozšírení v budúcnosti sa budú musieť upravovať syntaktické analyzátory pre YANG aj YIN.

Rozšírenia default-deny-all a default-deny-write sa môžu objaviť v definíciách dátových uzlov, RPC a notifikáciách. Inak budú ignorované. Toto chovanie je popísané v definícii týchto rozšírení. Navyše podľa gramatiky jazyka YANG spadajú rozšírenia pod pravidlo unknown-statement. Z týchto dôvodov bolo potrebné vytvoriť premennú `data_node`. Do tejto premennej sa ukladá ukazovateľ na uzol, v ktorom sa môžu objaviť spomenuté rozšírenia. Zároveň v premennej `actual` bude ukladaný ukazovateľ na aktuálny analyzovaný uzol (napríklad kontainer, definovanie dátového typu a pod.). Na obrázku 5.7 je zobrazené použitie rozšírenia default-deny-all.

```
container nacm {
  description "Parameters for NETCONF Access Control Model.";
  typedef action-type {
    type enumeration {
      enum permit;
      enum deny;
    }
  }
  nacm:default-deny-all;
```

```
  leaf read-default {
    type action-type;
    default "permit";
  }
}
```

Obr. 5.7: Ukážka použitia rozšírenia default-deny-all

V tomto prípade bude uložený do premennej `data_node` ukazovateľ na uzol container. Pri analyzovaní rozšírenia default-deny-all v pravidle unknown-statement sa najskôr porovná ukazovateľ z premennej `data_node` s ukazovateľom v premennej `actual`. Pri zhode ukazovateľov sa do uzlu uloží príznak daného rozšírenia. V tomto prípade sa do uzlu container uloží príznak default-deny-all.

5.7 Zmeny v uzloch

Pre zmeny existujúcich uzlov slúžia uzly *deviation* a *augment*. Každý z nich má iný význam.

Uzol deviation mení vlastnosti dátového uzlu. Názov deviation uzlu určuje cieľový uzol, ktorý sa má zmeniť. Môže pridávať (deviate add), odoberať (deviate delete) a aj nahradiť (deviate replace) vlastnosti. Pri zmene vlastnosti sa najskôr skontroluje, či je možné cieľovému uzlu meniť danú vlastnosť. Knižnica libyang nepodporuje, aby cieľový uzol mohol byť z toho istého modulu.

Uzol augment pridáva nové dátové uzly do existujúcich uzlov. Názov augment uzlu určuje cieľový uzol. Rozdiel oproti deviation uzlu je, že môže pridávať uzly aj do rovnakého modulu, kde je definovaný tento uzol. Pridávanie nových uzlov musí spĺňať sémantiku jazyka YANG. Ak nám to jazyk nedovoľuje, tak to nemôžeme uskutočniť (napríklad pridať do uzlu leaf iný uzol).

Kapitola 6

Testovanie

V prvej etape testovania bol využitý iteratívny spôsob. To znamená, že po naprogramovaní syntaktickej analýzy každého uzlu jazyka YANG, bola otestovaná jej základná funkcionalita. Na odhaľovanie chýb boli využité nástroje valgrind[1] a kdbg[2]. Valgrind bol využitý k detekcii prístupu na nevalidnú adresu alebo neuvoľnenie dynamicky alokovanej pamäte.

V druhej etape bolo potrebné vytvoriť testy na celý syntaktický analyzátor formátu YANG. Z tohto dôvodu bolo nutné nájsť vhodné prostredie na testovanie. Pre jazyk C bol vytvorený framework cmocka[3], ktorý ponúka prostredie na automatické testovanie. V knižnici libyang už bol tento framework integrovaný, čo znamenalo značnú výhodu.

Na obrázku 6.1 môžeme vidieť základnú štruktúru tvorby týchto testov. Väčšinou pozostávajú z troch častí: alokácia alebo inicializácia zdrojov, samotný test a uvoľnenie zdrojov. Prvá a posledná časť môže byť vynechaná, závisí to od samotného testu. Tieto testy zoskupujeme do jednej skupiny, ktorú následne spustíme.

```
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include "libyang.h"

static int setup_ctx(void **state)
{
  *state = (void *)ly_ctx_new(NULL);
  return (*state) ? 0 : 1;
}

static int teardown_ctx(void **state)
{
  struct ly_ctx *ctx = (struct ly_ctx *)*state;
  ly_ctx_destroy(ctx, NULL);
  return 0;
}
```

[1] http://valgrind.org/
[2] http://www.kdbg.org/
[3] https://cmocka.org/

```
void test_schema_yang(void **state)
{
  struct ly_ctx *ctx = (struct ly_ctx *)*state;
  if (!lys_parse_path(ctx, "schema/yang/ietf-ip.yang", LYS_IN_YANG)) {
    fail();
  }
}

int main()
{
  const struct CMUnitTest cmut[] = {
    cmocka_unit_test_setup_teardown(test_schema_yang, setup_ctx,
                                    teardown_ctx ),
  };

  return cmocka_run_group_tests(cmut, NULL, NULL);
}
```

Obr. 6.1: Ukážka tvorby testov v Cmocka frameworku

K testovaniu boli vybraté štandardné YANG moduly[4], ako sú napríklad ietf-snmp, ietf-ip a pod., ktoré sú vytvorené skupinou IETF. Tieto moduly otestujú pomerne veľkú časť syntaktického analyzátora. Navyše boli doplnené testy súvisiace s augment uzlami, ktoré je potrebné lepšie otestovať.

V jazyku C je obtiažne porovnávať štruktúry, ktoré obsahujú ukazovatele na ďalšie štruktúry. Z tohto dôvodu musel byť navrhnutý spôsob na porovnávanie štruktúr. Knižnica libyang dokáže z interných štruktúr, ktoré reprezentujú schémy, vytlačiť YANG moduly vo formáte YIN aj YANG. Táto funkcionalita bola využitá pri kontrole rovnosti načítaných modulov pre oba formáty. Testy prebiehali v troch častiach. V prvej časti sa načítal modul vo formáte YIN, ktorý sa následne uložil do textovej formy vo formáte YIN aj YANG. V druhej časti sa tento postup zopakoval, ale vstupný súbor bol tentokrát vo formáte YANG. V tretej časti boli tieto výstupy porovnávané na rovnosť so štandardnou funkciou `strcmp()`.

6.1 Chyby v implementácii

Pri testovaní samozrejme nastali isté chyby. Sú to napríklad:

- zmena kontextu Bisonu a Flexu,
- algoritmus na odsadenie reťazcov,
- neuvoľnenie dynamickej alokovanej pamäte.

Prvá chyba a aj jej riešenie boli spomenuté v podkapitole 5.5. Druhá chyba sa objavila až pri teste porovnania výstupov. Táto chyba bola zapríčinená nedostatočnou pozornosťou

[4]dostupné na webovej stránke www.netconfcentral.com

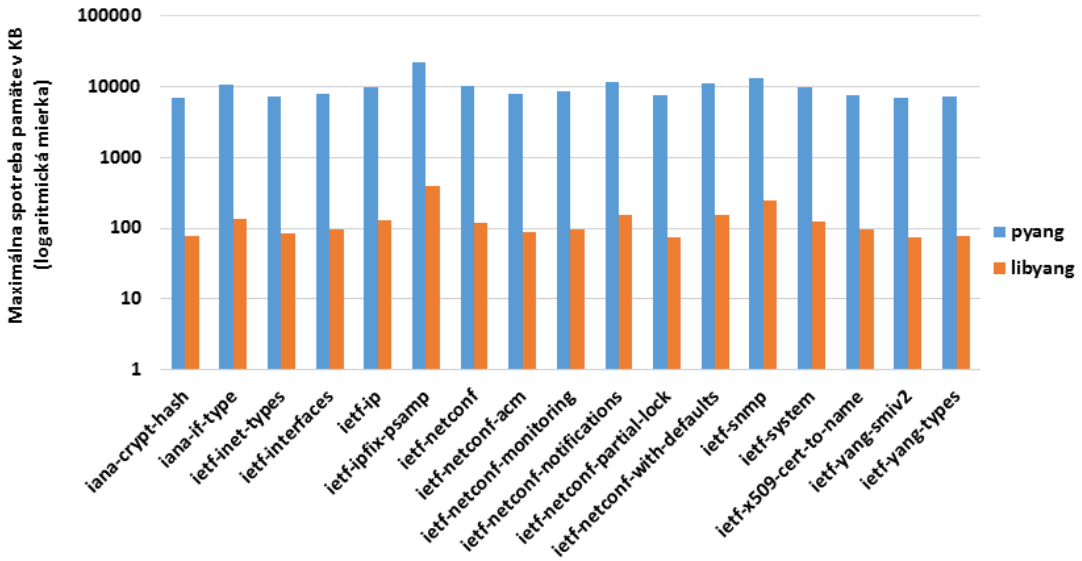
čítania RFC. Algoritmus na odstránenie odsadenia v reťazci je popísaný v podkapitole 5.4. Ďalšia chyba sa objavila pri pokuse analyzovania submodulu bez modulu, ktorému patrí. Tu stačilo len pridať volanie funkcie, ktorá uvoľní alokovanú pamäť, v ktorom bol uložený identifikátor submodulu.

Kapitola 7

Dosiahnuté výsledky

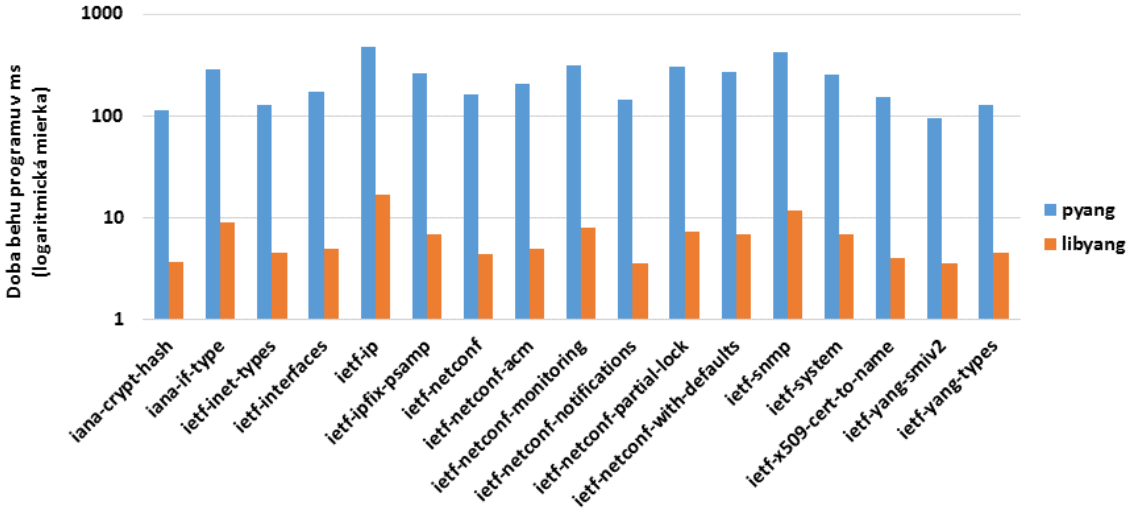
Táto kapitola obsahuje porovnania výsledkov výkonu a spotreby pamäte rozšírenia knižnice libyang. Knižnica libyang je porovnaná s programom pyang. K tomuto porovnávaniu boli zvolené rovnaké moduly ako pri otestovaní funkčnosti mojej implementácie.

Merania vychádzali z dvoch aspektov. Prvý aspekt bol zameraný na náročnosť pamäťovej zložitosti. K tomu bol použitý nástroj massif, ktorý je súčasťou nástroja valgrind. Nástroj massif slúži na nájdenie absolútneho vrcholu spotrebovanej pamäte. Druhý aspekt vychádzal z výpočtovej náročnosti. Tento aspekt bol meraný pomocou nástroja perf, ktorá zobrazila dobu behu programu a počet vykonaných inštrukcií.



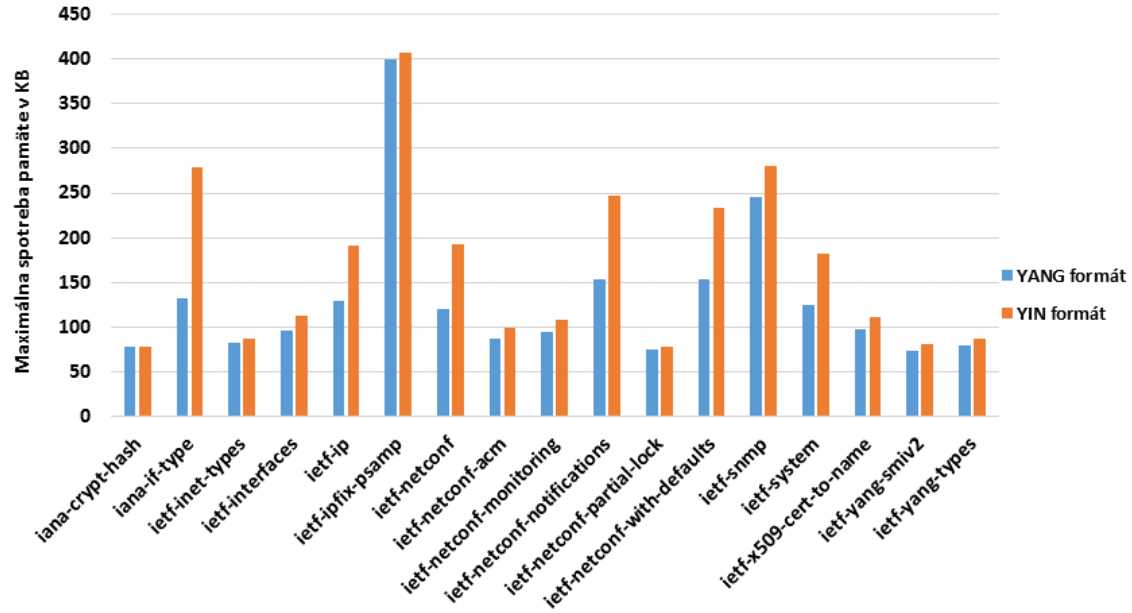
Obr. 7.1: Porovnanie libyang s pyangom - pamäť

Z obrázka 7.1 si môžeme všimnúť, že táto implementácia spotrebuje oveľa menej pamäte ako pyang. Keďže pyang je implementovaný v jazyku Python, tento výsledok bol očakávaný.



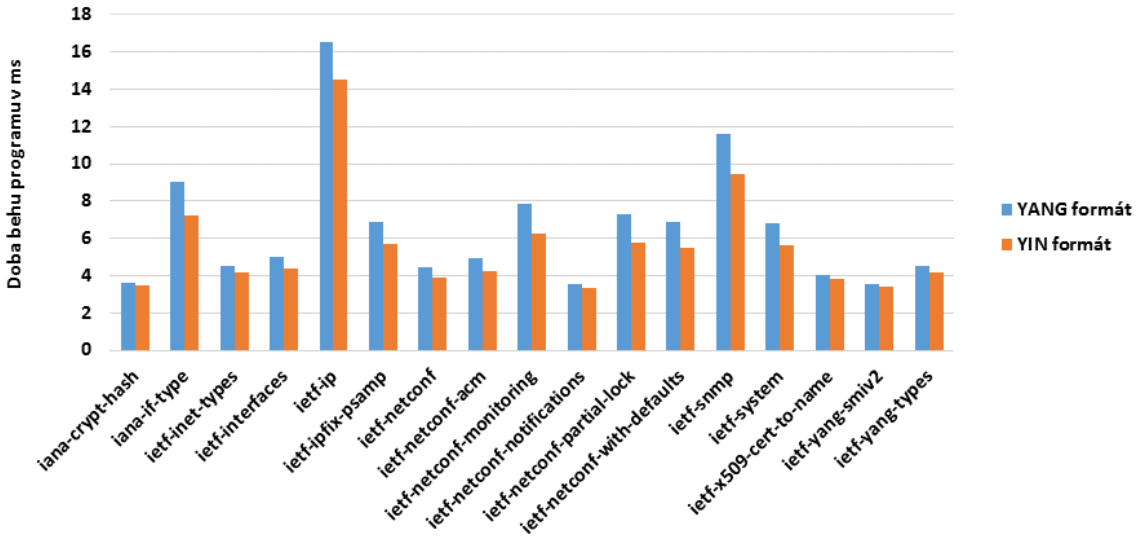
Obr. 7.2: Porovnanie libyang s pyangom - čas

Na obrázku 7.2 si môžeme všimnúť spotrebovaný čas na vykonanie analýzy YANG modulov. Takmer všetky moduly sa zanalyzovali do 10 milisekúnd pomocou knižnice libyang. Oproti pyangu je úspora vyše 96%. Táto úspora je opäť spôsobená najmä výberom implementačného jazyka.



Obr. 7.3: Porovnanie libyang YIN a YANG syntaktických analyzátorov - pamäť

Na obrázku 7.3 je zobrazené porovnanie implementácií YIN a YANG syntaktického analyzátora knižnice libyang. Môžeme si všimnúť, že syntaktický analyzátor formátu YANG je úspornejší. Táto úspora je spôsobená tým, že syntaktická analýza formátu YANG nemusí vytvárať v pamäti xml strom. Načítané dáta priamo ukladá do interných štruktúr knižnice libyang. Najviac úspory sa objavilo pri analyzovaní modulu iana-if-type, ietf-netconf-notification a ietf-netconf-with-defaults.



Obr. 7.4: Porovnanie libyang YIN a YANG syntaktických analyzátorov - čas

Posledný obrázok 7.4 zobrazuje porovnanie spotreby času na syntaktickú analýzu formátu YANG a YIN. Môžeme si všimnúť, že syntaktický analyzátor formátu YIN je rýchlejší. Táto skutočnosť môže byť zapríčinená tým, že syntaktická analýza formátu YANG musí prejsť cez celý vstupný súbor 2-krát.

V porovnaní so spotrebovanou pamäťou je výhodnejší syntaktický analyzátor formátu YANG. Napríklad v module iana-if-type formát YANG spotreboval o menej než 50% pamäte a je o 25% pomalší.

Kapitola 8

Záver

Výsledkom tejto práce je syntaktický analyzátor jazyka YANG. Tento analyzátor je navrhnutý ako rozšírenie knižnice libyang, ktorá slúži na analýzu dát a dátových modelov v jazyku YANG. Knižnicu libyang využíva ďalšia knižnica libnetconf. Táto knižnica implementuje protokol NETCONF.

V rámci práce bola naštudovaná problematika konfigurácie sieťových zariadení pomocou protokolu NETCONF a zároveň princípy knižnice libyang, libnetconf a systému Netopeer, ktoré sú vyvíjané organizáciou CESNET.

V rámci analýzy jazyka YANG bola objasnená problematika syntaktickej a lexikálnej analýzy tohto jazyka a súčasne možnosti týchto analýz pomocou nástrojov bison a flex. Následne bol navrhnutý syntaktický analyzátor jazyka YANG, ktorého gramatika je popísaná v RFC 6020.

Navrhnuté riešenie bolo implementované v podobe rozšírenia knižnice libyang. Na implementáciu rozšírenia bol zvolený jazyk C, ktorý bol už využívaný pri implementácii tejto knižnice. Implementácia pozostáva z niekoľkých častí. Prvá časť obsahuje vstupný súbor pre flex. V tomto súbore sa nachádzajú regulárne výrazy pre kľúčové slová, identifikátory, reťazce a komentáre. V druhej časti nájdeme vstupný súbor pre bison, kde sa nachádzajú pravidlá pre syntaktickú analýzu, názvy tokenov a pod. Tretia časť obsahuje súbory, v ktorých sú definované funkcie, ktoré vykonávajú sémantickú kontrolu.

Rozšírenie knižnice libyang bolo úspešne otestované na štandardných YANG moduloch. V porovnaní medzi pyangom a týmto riešením je úspora vyše 96% spotrebovaného času aj pamäti. Pri porovnaní syntaktickej analýze formátu YIN a YANG pomocou knižnice libyang bolo zistené, že táto implementácia je úspornejšia na pamäť, avšak je o trochu pomalšia oproti formátu YIN. V celkovom hodnotení vychádza syntaktický analyzátor formátu YANG o niečo lepšie ako formátu YIN.

Cieľom ďalšej práce bude zefektívnenie dedenia príznakov pri rozpoznaní rozšírenia. V súčasnej implementácii je dedenie týchto príznakov riešené až pri načítaní celého modulu, avšak toto riešenie nie je veľmi vhodné pri veľkom počte uzlov. Ďalšie rozšírenie knižnice libyang by obsahovalo automatické rozpoznávanie formátu modulu pri jeho načítavaní.

Literatúra

[1] Case, J. D.; Fedor, M.; Schoffstall, M. L.; aj.: Simple Network Management Protocol (SNMP). STD 15, RFC Editor, May 1990.
URL http://www.rfc-editor.org/rfc/rfc1157.txt

[2] Caroline Chappell: Creating the programmable network: The business case for netconf/yang in network devices. *Whitepaper on Behalf of Tail-f*. October 2013.

[3] Enns, R.: NETCONF Configuration Protocol. RFC 4741, RFC Editor, December 2006.
URL http://www.rfc-editor.org/rfc/rfc4741.txt

[4] Bjorklund, M.: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, RFC Editor, October 2010.
URL http://www.rfc-editor.org/rfc/rfc6020.txt

[5] Enns, R.; Bjorklund, M.; Schoenwaelder, J.; aj.: Network Configuration Protocol (NETCONF). RFC 6241, RFC Editor, June 2011.
URL http://www.rfc-editor.org/rfc/rfc6241.txt

[6] Crocker, D.; Overell, P.: Augmented BNF for Syntax Specifications: ABNF. STD 68, RFC Editor, January 2008.
URL http://www.rfc-editor.org/rfc/rfc5234.txt

[7] Cesnet TMC group: *Netopeer*. [cit. 2016-04-12].
URL https://www.liberouter.org/technologies/netconf/

[8] Lowell D. Thomas: *APG - ABNF Parser Generator*. [Online; navštíveno 12.4.2016].
URL http://www.coasttocoastresearch.com/

[9] Češka, M.; Rábová, Z.: *Gramatiky a jazyky*. Brno: Ediční středisko VUT, vyd. 1 vydání, 1985.

[10] Levine, J. R.: *Flex & bison*. Sebastopol, CA: O'Reilly, First edition, c2009, ISBN 0596155972.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

Priložené CD obsahuje:

- priečinok **text** obsahuje zdrojové súbory textu bakalárskej práce pre LaTeXvrátane obrázkov
- priečinok **source** obsahuje zdrojové súbory celej knižnice libyang
- súbor **README** obsahuje návod na preklad knižnice a spustenie ukážkového programu, ktorý využíva knižnicu libyang