Fordítóelmélet bevezetés

Simon Balázs

BME IIT, 2011.

forrás: http://www.info.uni-karlsruhe.de/lehre/2007WS/uebau1/

Tartalom

- Motiváció és követelmények
- Fordítás típusai
- Formális nyelvek
- Fordítás fázisai
  - Analízis
  - Leképzés
  - Kódgenerálás
- Fordító által használt adatszerkezetek

Motiváció és követelmények

Motiváció

- A gyakorlati informatika legrégebbi területe
- Stabil architektúra
- A helyesség és megbízhatóság követelmény
- Elősegíti a programnyelvek fejlődését
- Elősegíti az egyéb HW és SW interfészek fejlődését
- Egységes szövegfeldolgozást és kódgenerálást tesz lehetővé
- Sokféle alkalmazási terület:
  - szövegformázás, programtranszformáció, meta-programozás, aspektus-orientált programozás, stb.

Felhasználási területek

- Információ kinyerése szövegből
  - pl. LaTeX, Word, konfigurációs fájlok
- Szövegből másik szöveg előállítása
  - forráskód-transzformáció
  - pl. preprocesszor, kódgenerálás
- Programszövegből virtuális gép kódja
  - pl. Java byte-code, .NET CLI
- Programszövegből gépi kód
  - pl. C fordító, Pascal fordító

Mi a fordítás?

- A szöveges forrásnyelv átvitele a célnyelvre a jelentés megtartásával
- A forrás- és célnyelv lehetnek azonosak
- A célnyelv lehet számítógép által értelmezhető
- Programozási nyelveknél adott a szemantika (jelentés)
  - a program struktúrája határozza meg a jelentést
  - pl. adattípusok, vezérlőszerkezetek, stb.

Követelmények

- Helyesség
- Minimális erőforráshasználat
  - futásidő, memóriaméret, fogyasztás
- Kompatibilitás más fordítókkal
  - más nyelvekből, más számítógépeken fordított kódok
- Fordítási sebesség

Fordítás típusai

Fordítás típusai

- Interpretálás
- Előfordítás
- Futás-idejű fordítás
- Teljes fordítás
- Makrók

Interpretálás előfordítás nélkül

- Kifejezések beolvasása egyenként
- Ezek értelmezése, fordítása
- Végrehajtás egyenként
- Megéri, ha csak egyszer kell végrehajtani
  - pl. parancsértelmezők (DOS, UNIX shell)

Interpretálás előfordítással

- A forrásnyelv értelmezése
- Fordítás az interpreter által könnyen értelmezhető nyelvre
  - pl. változódefiníció és -használat, post-fix formára transzformálás
- A célnyelv nem feltétlenül gépi kód
- Példa: Java byte-code

Futási idejű fordítás

- Végrehajtás közbeni fordítás
  - JIT = Just-In-Time fordítás
- A forráskódból előállított köztes kód továbbfordítása
- Gyorsabb, mint az interpretálás
- Lassabb, mint a teljes fordítás, mert nincs meg a teljes kontextus
- Kisebb memóriaigény: a köztes kód tömörebb, mint a célkód
- Dinamikus futásidejű viselkedést is figyelembe tudja venni
- Példa: .NET (a hiányzó típusinformációk miatt nem interpretálható)

Teljes fordítás

- A gépközeli kódnak is van „absztrakt gépi kódja”
- A végrehajtást befolyásolja:
  - a célgép HW-e
  - operációs rendszer
  - futtatórendszer (pl. memóriakezelés)
- Példa: C, Pascal fordító

Makrók

- A forrás- és célnyelvek megegyeznek
- A makrók helyettesítési szabályokat definiálnak
- Példa:
  - szövegformázás
  - szövegfeldolgozás
  - preprocesszor: C, C++, C#

Formális nyelvek

Formális nyelvek

- Természetes nyelv:
  - bonyolult, strukturálatlan, félreérthető
- Számítógépek számára egyértelmű bemenet kell
- Formális nyelv:
  - strukturált, egyértelmű
  - szigorú szabályok vannak a mondatok előállítására
  - számítógép számára könnyen értelmezhető

Fogalmak

- Nemterminális (non-terminal)
  - levezetések során eltűnnek
  - jelölés: nagybetű
  - pl. A, B, X, stb.
- Terminális (terminal)
  - levezetések végén csak ezek maradnak
  - jelölés: kisbetű, egyéb szimbólum
  - pl. a, b, x, y, (,), +, stb.
- Üres szimbólum (empty)
  - azt jelzi, hogy az adott nemterminális eltűnik
  - epszilon (ε)

Fogalmak

- Mondat
  - az előállítandó terminálisokból álló szimbólumsorozat
- Mondatszimbólum
  - a kiinduló nemterminális (tipikus jelölés: S)
  - ez reprezentálja a mondatot
- Levezetési szabály (production rule)
  - megadja, hogy egy adott karaktersorozat mire cserélhető le
  - pl. $A \rightarrow aBa$, $xB \rightarrow Ay$, $X \rightarrow \varepsilon$, stb.

Fogalmak

- Levezetés:
  - a mondatszimbólumból terminálisok sorozatának előállítása a levezetési szabályok segítségével
- Elemzés:
  - terminálisok egy adott sorozatából a levezetési szabályok felhasználási sorrendjének visszafejtése egészen a mondatszimbólumig
  - (fordítóknál ez a fontos!)

Példák levezetésekre

- A->a, A->Aa
  - A=>Aa=>Aaa=>Aaaa=>aaaa
- S->AB, A->aA, A->a, B->bB, B->b
  - S=>AB=>aAB=>aaAB=>aaAbB=>aaabB=>aaabb
- S->aSb, S->ε
  - S=>aSb=>aaSbb=>aaaSbbb=>aaabbb
- S->aSBC, S->abC, CB->BC, bB->bb, bC->bc, cC->cc
  - S=>aSBC=>aabCBC=>aabBCC=>aabbCC=>aabbcC=>aabbcc

Chomsky-nyelvosztályok

- 3: Reguláris nyelvek (regular)
- 2: Kontextusfüggetlen nyelvek (context free)
- 1: Kontextusfüggő nyelvek (context sensitive)
- 0: Egyéb nyelvek

- Adott nyelvtan osztálya:
  - csak az osztálynak megfelelő feltételeket teljesítő levezetési szabályokat tartalmaz
- Adott nyelv osztálya:
  - a legegyszerűbb nyelvtan osztálya, amellyel a nyelv mondatai leírhatók

Reguláris nyelvek

- Csak az alábbi típusú szabályok:
  - $A \rightarrow Ba$ és $A \rightarrow a$ vagy $A \rightarrow aB$ és $A \rightarrow a$
  - bal oldalon egyetlen nemterminális
  - jobb oldalon csak egy terminális vagy egy nemterminális és egy terminális (de mindig ugyan abban a sorrendben!)
- Elemzés:
  - véges automatával
  - fordítók esetén: lexikai elemzés így történik

Kontextusfüggetlen nyelvek

- Csak az alábbi típusú levezetési szabályok:
  - $A \rightarrow \alpha$
  - bal oldalon egyetlen nemterminális
  - jobb oldalon (α) tetszőleges terminálisok és nemterminálisok sorozata
- Elemzés:
  - veremautomatákkal vagy LL, LR, LALR, stb.
  - fordítók esetén: szintaktikai elemzés így történik

Kontextusfüggő nyelvek

- Csak az alábbi típusú levezetési szabályok:
  - $\beta A \gamma \rightarrow \beta \alpha \gamma$
  - tehát A helyettesítése csak a β-γ kontextusban történhet
  - a szabályok nem rövidítő szabályok: alkalmazásuk során sosem lesz rövidebb a már levezetett karaktersorozat
- Nehéz őket elemezni:
  - fordítóknál nem használjuk

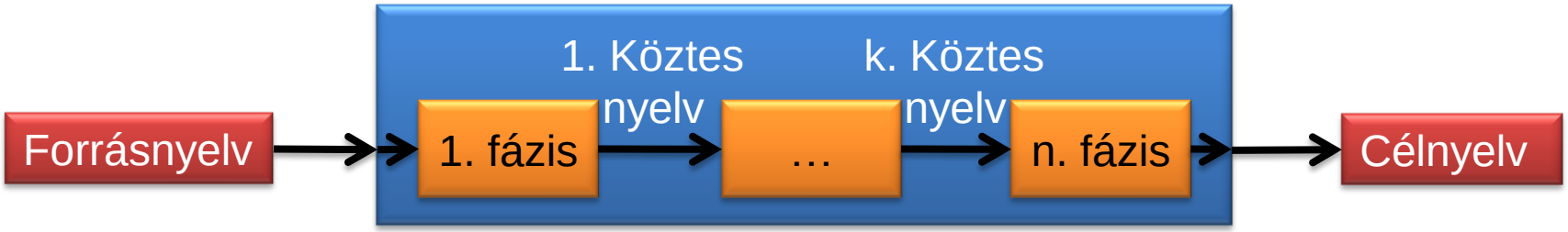
Egyéb nyelvek

- Minden szabály megengedett
- Nagyon nehéz őket elemezni:
  - Turing-gép

Fordítás fázisai

A fordítók archtektúrája

- Forrásnyelvből célnyelv előállítása
- Több fázisban
- Az egyes fázisok között: köztesnyelv
- Több köztesnyelv is lehet
- A köztesnyelvek szükségessége gyakorlati:
  - egyszerűbb a fordítás
  - kisebb a memóriaigény



A fordítás fázisai

- Elemzés:
  - 1. Lexikai elemzés (lexical analysis: lexer):
    - szimbólumok meghatározása
  - 2. Szintaktikai elemzés (syntax analysis: parser):
    - konkrét és absztrakt szintaxisfa meghatározása
  - 3. Szemantikai elemzés (semantic analysis):
    - névfeloldás, típusellenőrzés, konzisztenciaellenőrzés
- Leképzés:
  - 4. Transzformáció:
    - célnyelvhez közeli reprezentációra
  - 5. Globális optimalizálás:
    - konstansok, közös részkifejezések, kód újraszervezése
- Kódkészítés:
  - 6. Kód előállítása:
    - végrehajtás sorrendje, parancsok kiválasztása, regiszterkiosztás, utóoptimalizálás
  - 7. Assembler és linker:
    - címhivatkozások feloldása, parancsok, címek és adatok kódolása

1. Lexikai elemzés (lexer)

- A forrásnyelv feldarabolása elemi jelentést hordozó elemekre (szimbólumok)
- Felesleges karakterek elhagyása
  - kommentek, szóközök, tabulátorok, stb.
- A szintaktikai elemzés előtti fázis:
  - véges automatával megoldható (reguláris nyelvek segítségével)
  - gyorsabb

2. Szintaktikai elemzés (parser)

- Előállítja a programkód fastruktúráját
  - Concrete Syntax Tree (CST)
  - Abstract Syntax Tree (AST)
- Bemenete: szimbólumsorozat
- Kontextusfüggetlen (CF = context-free) nyelvtan alapján dolgozik
  - ezt könnyű elemezni
  - viszonylag gyorsan
  - elég nagy kifejezőerő
  - tipikus elemző automaták: LL, LR, LALR

3. Szemantikai elemzés

- AST elemzése
  - indok: a programnyelvek context-sensitive-ek
  - azonosítók (identifier) jelentésének meghatározása
- Név- és típusfeloldás
- Konzisztenciaellenőrzés
  - a programnyelv által meghatározott korlátok ellenőrzése
  - pl. statikus tömbök kiindexelése, konstansok, interfész függvényeinek implementációja, stb.
- Jelentés hozzárendelése
  - operátorok
  - definíciók és hivatkozások összerendelése (változók, függvények, stb.)

4. Transzformáció

- Tényleges fordítás
- Objektumok reprezentációjának meghatározása a célgépen
- Operációk fordítása
- Vezérlőszerkezetek fordítása
- Eredmény: célnyelvhez közeli köztesnyelv
- Még nem keletkezik célnyelvben leírt kód

5. Optimalizálás

- Optimalizáló transzformációk
- A működés helyességének megtartása
- Ez ma a fő kutatási terület
- Példák:
  - dead code elimination, expression simplification, if optimalization, register allocation, tail recursion, stb.
- Nincs optimális programkód, csak optimalizált:
  - egymásnak ellentmondó optimalizációk
  - egy optimalizációs eljárás bevezethet olyan struktúrákat, amelyeket egy korábbi eljárás már kiszűrt
  - így számít az eljárások sorrendje, akár megismétlése is

6. Kódgenerálás

- Kód előállítása
  - Parancsok kiválasztása
  - Végrehajtási sorrend kiválasztása
  - Konkrét reprezentáció kiválasztása
    - memória
    - regiszter
- Lokális utóoptimalizálás

7. Assembler és Linker

- Assembler:
  - bináris előállítása
  - szimbolikus címek feloldása, ameddig lehet
- Linker:
  - lefordított modulok összeszerkesztése
  - külső szimbolikus címhivatkozások feloldása

Fordító által használt adatszerkezetek

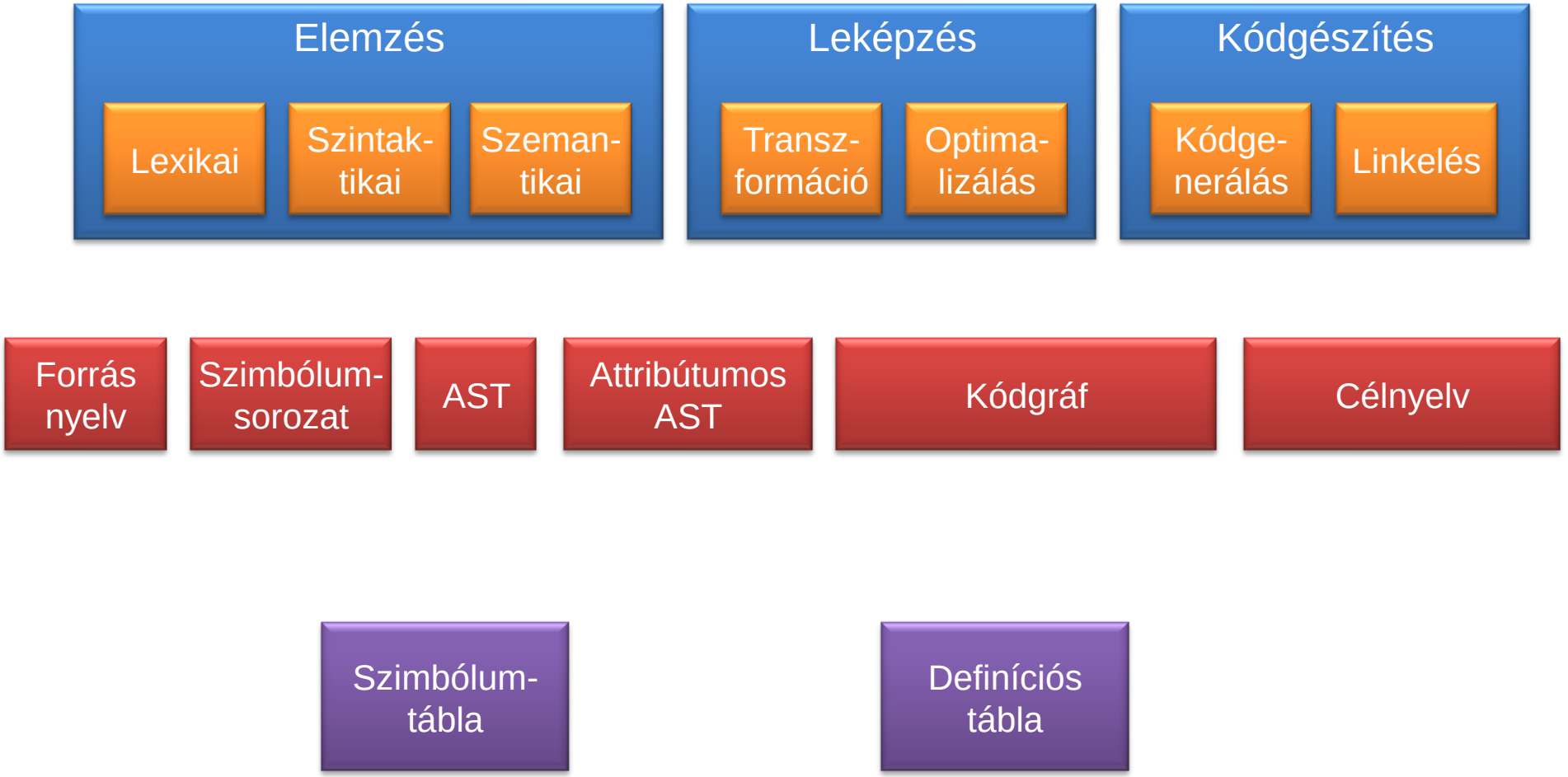
Fordító adatszerkezetei

- Szimbólumsorozat
- Concrete Syntax Tree (CST)
- Abstract Syntax Tree (AST)
- Attribútumos AST
- Kódgráf

- Szimbólumtábla
- Definíciós tábla

(csak fogalmilag fontosak, nem feltétlenül jelennek meg az implementációban)

Fordító moduláris struktúrája



Szimbólumsorozat

- Szimbólum:
  - a program elemi jelentéssel bíró egysége
  - két rész: kulcs (token) és érték (value)
- Szimbólumsorozat:
  - a programkód szimbólumok sorozataként való előállítása
- Lexer kimenete
- Parser bemenete

CST, AST

- Concrete Syntax Tree (CST)
  - a forráskód jelentéssel bíró elemeit (szimbólumokat) tartalmazó fastruktúra
- Abstract Syntax Tree (AST)
  - a szintaxis nagyban leegyszerűsítve, csak a struktúra marad
  - pl. kulcsszavak, zárójelek, pontosvesszők elhagyása
- Parser kimenete
- Szemantikai elemző bemenete

Attribútumos AST

- Attribútum:
  - név-érték pár
  - pl. típusok, nevek, konstans értékek, definíció, hivatkozás, stb.
- Attribútumos AST
  - az AST csúcsainak felruházása attribútumokkal
  - az attribútumok értékeinek számítása az AST struktúrájából, a szimbólumokból és a többi attribútum értékéből
- Szemantikai elemző kimenete
- Leképzési fázis bemenete

Kódgráf

- A program előállítása a célnyelvhez közeli adatstrukturák és műveletek segítségével
- A konkrét gépi parancsok még nem kerülnek kiválasztásra
- Nincs regiszter- és memóriakorlát
- Transzformáció kimenete
- Optimalizáció bemenete és kimenete
- Kódgenerálás bemenete

Célnyelv

- A célnyelv kiválasztott parancsai szimbolikusan kódolva
- Assembler:
  - előállítja a bináris kódot
  - külső címek csak szimbolikusak
- Linker:
  - összeszerkeszti a kódot
  - feloldja a külső hivatkozásokat
- Eredmény:
  - bináris kód szimbolikus címek nélkül

Szimbólumtábla

- Azonosítók (identifier) és konstans értékek (literal) leképzése a fordító belső reprezentációjára
- Egy táblabejegyzés:
  - szimbólum (token) és érték (value)
  - valamint sor- és oszlopindex a forráskódban (hibajelzéshez)
- Lexer építi fel
- Felépítés után végig változatlan marad a fordítás során

Definíciós tábla

- A fordító adatbázisa
- Tárolja az azonosítók definícióját
- A szimbólumokhoz jelentést társít
- Megadja, hogy azonos szimbólumok mikor jelölik ugyanazt a változót, típust, függvényt, stb.
- Tárolja a feldolgozott szimbólumok szemantikai kontextusát
- A szemantikai elemző építi fel

Fordító moduláris struktúrája