Poglavje 3

SINTAKSA

3.1 Osnovni sintaktični objekti

Uporabljali bomo dve vrsti objektov: nize in abstraktno sintaksno drevo.

Nizi so primerna podatkovna struktura za linearno predstavitev jezka, niso pa primerni za analizo.

Abstraktno sintaksno drevo prikažejo hierarhično strukturo sintakse zato je primerno za semantično analizo.

3.1.1 Simboli

Uporabjali bomo množico simbolov, ki bodo služili v različnih vlogah: znaki, spremenljivke, imena polj, ...

Simbolom bomo včasih rekli imena ali atomi ali identifikatorji v odvisnosti od navad v konkretnem delovnem okolju.

O simbolih bomo razmišljali kot o atomih, ki nimajo strukture razen identitete.

Napišemo x sym kar pomeni, da je x simbol in predpostavljamo, da imamo neskončno mnogo simbolov za to identiteto.

Sodba $x \# y$, kjer je x sym in y sym, pove, da sta x in y različna simbola.

Uporabljali bomo različne razrede simbolov. V splošnem bomo predpostavili, da sta katerakoli razreda simbolov med sabo različna, tako da ne bo zmešnjav med njimi.

3.1.2 Nizi nad abecedo

Abeceda je končna množica znakov c_1 sym,...,c_n sym. Naj Σ označuje končno množico takšnih sodb.

Forma sodbe za $\Sigma \vdash s$ str, ki definira nize nad abecedo Σ, je induktivno definirana z naslednjimi pravili.

$$\frac{}{\Sigma \vdash \varepsilon \text{ str}} \tag{3.1}$$

$$\frac{\Sigma \vdash c \text{ sym} \quad \Sigma \vdash s \text{ str}}{\Sigma \vdash c \cdot s \text{ str}} \tag{3.2}$$

Niz je torej sekvenca znakov oz. null v primeru, da ne vsebuje znakov. Pogosto ne omenjamo Σ, če je abeceda jasna iz konteksta.

V primeru nizov uporabljamo indukcijo na sledeč način. Ko hočemo pokazati lastnost P nad nizom s je zadosti, da vidimo sledeče:

- $\epsilon\ P$
- Ob predpostavki $s\ P$ in c sym pokažemo $c \cdot s\ P$

To včasih imenujejmo princip *indukcije nad nizi*. Metoda je ekvivalentna indukciji na dolžino niza le da ni potrebno definirati dolžine niza.

Naslednje pravilo induktivno definira izjavo $s_1\hat{}s_2 = s$ str, ki pravi, da je s konkatenacija s_1 in s_2.

$$\frac{}{\epsilon\hat{}s \text{ str}} \tag{3.3}$$

$$\frac{s_1\hat{}s_2 = s \text{ str}}{(c \cdot s_1)\hat{}s_2 = c \cdot s \text{ str}} \tag{3.4}$$

Enostavno je pokazati z indukcijo, da pravila definirajo totalno funkcijo ($\forall\forall\exists!$) na dveh argumentih.

Nizi so običajno zapisani kot sekvenca simbolov npr. "abcd"je predstavitev niza $a \cdot (b \cdot (c \cdot (d \cdot e)))$.

3.1.3 Abstraktna sintaksna drevesa

Abstraktno sintaksno drevo (okr. ast) je urejeno drevo v katerem so vozlišča označena s simboli, ki jih imenujemo *operatorji*.

Signatura Ω je končna množica sodb oblike $ar(o) = n$, kjer velja o sym in n nat. Te sodbe priredijo številnost n operatorjem o, tako da če $\Omega \vdash ar(o) = n$ in $\Omega \vdash ar(o) = n'$ potem $n = n'$.

Razred abstraktnih sintaksnih dreves nad signaturo Ω je induktivno definiran na sledeč način.

$$\frac{\Omega \vdash ar(o) = n \quad a_1 \text{ ast} \ldots a_n \text{ ast}}{o(a_1, \ldots, a_n) \text{ ast}} \tag{3.5}$$

Osnovni primer induktivne definicije je operator s številnostjo nič. V tem primeru je premisa prazna. Takšna vozlišča drevesa so listi.

Strukturna indukcija

Princip strukturne indukcije je specializacija principa indukcije pravil na pravila, ki definirajo abstraktna sintaksna drevesa nad signaturami.

Če želimo pokazati $P(a \text{ ast})$ je zadosti, da pokažemo da je P zaprta glede na pravilo 3.5. Z drugimi besedami, če $\Omega \vdash ar(o) = n$, potem moramo pokazati, da

$$\text{if } P(a_1 \text{ ast}), \ldots, P(a_n \text{ ast}) \text{ then } P(o(a_1, \ldots, a_n) \text{ ast}). \tag{3.6}$$

V primeru, da je $n = 0$ se dokaz reducira na to, da pokažemo $P(o())$.

Primer 3.1.1 *Poglejmo si sledečo induktivno definicijo višine ast.*

$$\frac{hgt(a_1) = h_1 \ \ldots \ hgt(a_n) = h_n \quad max(h_1, \ldots, h_n) = h}{hgt(o(a_1, \ldots, a_n)) = \mathit{succ}(h)} \tag{3.7}$$

Z uporabo strukturne indukcije lahko pokažemo $(\forall, \exists!)$*, kar pomeni, da ima vsak ast unikatno višino.*

Predpostavimo lahko, da imajo vsa ast višine $1 \leq i \leq n$ *unikatno višino* $hgt(a_i) = h_i$*. Pokažemo lako, da je maksimum vseh unikatno definiran in je torej celotna višina unikatna.* □

Spremenljivke in substitucija

V praksi pogosto potrebujemo abstraktna sintaksna drevesa s spremenljivkami kot vrzeli, ki jih je mogoče napolniti z drugimi drevesi.

Spremenljivke so instancirane z uporabo substitucije spremenljivk ast z dugim drevesom.

Poglejmo si najprej nekaj dogovorov v povezavi z notacijo. Naj bo $\mathcal{X} = x_1$ ast, $\ldots, x_n$ ast množica parametrov in hkrati seznam hipotez $\{x_1, \ldots, x_n\} \mid x_1$ ast, $\ldots, x_n$ ast, kjer velja tudi x_1 sym, $\ldots, x_n$ sym. Naprej, naj izraz $x \# \mathcal{X}$ pomeni $x \notin \{x_1, \ldots, x_n\}$.

Z uporabo te notacije lahko napišemo sodbo $\mathcal{X} \vdash a$ ast, ki je induktivno definirana z naslednjimi pravili:

$$\frac{}{\mathcal{X}, x \text{ ast} \vdash x \text{ ast}} \tag{3.8}$$

$$\frac{\Omega \vdash ar(o) = n \quad \mathcal{X} \vdash a_1 \text{ ast} \ldots \mathcal{X} \vdash a_n \text{ ast}}{\mathcal{X} \vdash o(a_1, \ldots, a_n) \text{ ast}} \tag{3.9}$$

Po principu indukcije pravil je za dokaz $P(\mathcal{X} \vdash a \text{ ast})$ zadosti, da pokažemo:

1. $P(\mathcal{X}, x \text{ ast} \vdash x \text{ ast})$.
2. Če $\Omega \vdash ar(o) = n$ in če $P(\mathcal{X} \vdash a_1 \text{ ast}), \ldots, P(\mathcal{X} \vdash a_n \text{ ast})$, potem $P(\mathcal{X} \vdash o(a_1, \ldots, a_n) \text{ ast})$.

Parametri $\mathcal{X}$ se obravnavajo kot atomični objekti, vendar ima vsak svoje abstraktno sintaksno drevo.

Definiramo sodbo $\mathcal{X} \vdash [a/x]b = c$, kar pomeni da je c rezultat substitucije a za x v b z uporabo naslednjih pravil.

$$\frac{}{\mathcal{X}, x \text{ ast} \vdash [a/x]x = a} \tag{3.10}$$

$$\frac{x \# y}{\mathcal{X}, x \text{ ast}, y \text{ ast} \vdash [a/x]y = y} \tag{3.11}$$

$$\frac{\mathcal{X} \vdash [a/x]b_1 = c_1 \ldots \mathcal{X} \vdash [a/x]b_n = c_n}{\mathcal{X} \vdash [a/x]o(b_1, \ldots, b_n) = o(c_1, \ldots, c_n)} \tag{3.12}$$

Rezultat substitucije je enolično definiran z argumenti. Kot posledico lahko napišemo $[a/x]b$ za unikaten c, tako da $[a/x]b = c$.

Izrek 3.1.1 *Če velja $\mathcal{X} \vdash a$ ast in $\mathcal{X}, x$ ast $\vdash b$ ast, kjer $x \# \mathcal{X}$, potem obstaja enoličen c, tako da $\mathcal{X} \vdash [a/x]b = c$ in $\mathcal{X} \vdash c$ ast.*

Dokaz. Trditev dokažemo s strukturno indukcijo na b relativno na kontekst $\mathcal{X}, x$ ast.

Imamo tri primere za obravnavo:

1. Ker velja $\mathcal{X}, x$ ast $\vdash x$ ast, moramo pokazati da obstaja unikaten c tako da $\mathcal{X} \vdash [a/x]x = c$. Z uporabo pravila 3.10a vidimo da je zamenjava c z a potrebna in zadostna.

2. Če $\mathcal{X}, x$ ast, y ast $\vdash y$ ast za nek $y \# x$, potem po pravilu 3.10b zamenjava c z y je potrebna in zadostna.

3. Končno, če $b = o(b_1, \ldots, b_n)$, potem velja po indukciji, da obstaja enoličen $c_1, \ldots, c_n$ tako da $\mathcal{X} \vdash [a/x]b_1 = c_1, \ldots, \mathcal{X} \vdash [a/x]b_n = c_n$. Po pravilu 3.10c je edina možna izbira za c drevo $o(c_1, \ldots, c_n)$, kar zadostuje. □

3.2 Konkretna sintaksa

Konkretna sintaksa jezika se uporablja za predstavitev gramatike z navadnim tekstom. Velikokrat uporabimo konktretno sintakso za izboljšanje berljivosti in izločanje dvoumnosti jezika.

V tej sekciji predstavimo glavne metode za specifikacijo konkretne sintakse. Uporabljali bomo jezik izrazov $\mathcal{L}(\text{nat str})$, ki predstavi elementarno aritmetiko na naravnih številih, enostavne operacije na nizih in enostavno delo s spremenljivkami.

$$
\begin{array}{llll}
\text{Tip } t & ::= & \text{num} & \text{num} \\
 & & \text{str} & \text{str} \\
\text{Izraz } e & ::= & x & x \\
 & & \text{num}[n] & n \\
 & & \text{str}[s] & s \\
 & & \text{plus}(e_1; e_2) & e_1 + e_2 \\
 & & \text{times}(e_1; e_2) & e_1 * e_2 \\
 & & \text{cat}(e_1; e_2) & e_1 \hat{} e_2 \\
 & & \text{len}(e) & |e| \\
 & & \text{let}(e_1; x.e_2) & \text{let } x \text{ be } e_1 \text{ in } e_2
\end{array}
\tag{3.13}
$$

Jezik $\mathcal{L}(\text{nat str})$ vsebuje osnovne operacije za delo z naravnimi števili in nizi. Edini sestavljen gradnik jezika je let stavek, ki izvede substitucijo spremenljivke v stavku.

3.2.1 Leksikalna struktura

Prva faza sintaktičnega procesiranja je pretvorba iz predstavitve osnovane na znakih v predstavitev osnovani na simbolih. Ta proces običajno imenujemo *leksikalna analiza*. Osnovna ideja leksikalne analize je združiti znake v simbole, ki so osnova za naslednje faze analize.

Na primer, niz znakov 467 se spemeni v celo število 467. Podobno predstavimo identifikator sestavljen iz niza znakov “temp” v spremenljivko z imenom $temp$.

Med leksikalno analio počistimo tudi vse “bele znake” (presledke, tabulatorje, itd.) in komentarje, ki se ne uporabljajo v nadalnji analizi.

Znakovna predstavitev je v največ primerih definirana z regularnimi izrazi. Leksikalna struktura $\mathcal{L}(\text{nat str})$ je določena z naslednjimi pravili.

$$
\begin{array}{llll}
Item & itm & ::= & kwd \mid id \mid num \mid lit \mid spl \\
Keyword & kwd & ::= & l\,.\,e\,.\,t\,.\,\epsilon \mid b\,.\,e\,.\,\epsilon \mid i\,.\,n\,.\,\epsilon \\
Identifier & id & ::= & ltr\ (ltr \mid dig)^* \\
Numeral & num & ::= & dig\ dig^* \\
Literal & lit & ::= & qum\ (ltr \mid dig)^*\ qum \\
Special & spl & ::= & + \mid\ *\ \mid \hat{}\ \mid (\mid) \mid \mid \\
Letter & ltr & ::= & a \mid b \mid ... \\
Digit & dig & ::= & 0 \mid 1 \mid ... \\
Quote & qum & ::= & ”
\end{array}
\tag{3.14}
$$

Leksikalni simbol je bodisi ključna beseda, identifikator, število, niz znakov ali posebni simbol. Simboli se med seboj razlikujejo po sestavi znakov, ki jih posamezna vrsta simbola vsebuje. Simboli, ki predstavljajo nize vsebujejo sekvence natisljivih znakov, ki so obdani z narekovaji.

Delo leksikalnega analizatorja je pretvorba nizov znakov v niz simbolov. Zgornja definicija leksikalne sintakse služi kot vodilo pri pretvorbi.

Vhoden niz znakov se pretvarja v niz simbolov z uporabo naslednjih pravil, ki definirajo simbolni jezik razčlenjevalnika oz. statično semantiko jezika.

$$\frac{s\ str}{ID[s]\ tok} \tag{3.15}$$

$$\frac{n\ nat}{NUM[n]\ tok} \tag{3.16}$$

$$\frac{s\ str}{LIT[s]\ tok} \tag{3.17}$$

$$\frac{}{LET\ tok} \tag{3.18}$$

$$\frac{}{BE\ tok} \tag{3.19}$$

$$\frac{}{IN\ tok} \tag{3.20}$$

$$\frac{}{ADD\ tok} \tag{3.21}$$

$$\frac{}{MUL\ tok} \tag{3.22}$$

$$\frac{}{CAT\ tok} \tag{3.23}$$

$$\frac{}{LP\ tok} \tag{3.24}$$

$$\frac{}{RP\ tok} \tag{3.25}$$

$$\frac{}{VB\ tok} \tag{3.26}$$

Lesikalna analiza je definirana s pretvorbo nizov znakov v nize simbolov. Pretvorba je induktivno definirana s pomočjo naslednjih pravil:

$$\begin{array}{ll} s\ inp \longleftrightarrow t\ tokstr & Scan\ input \\ s\ itm \longleftrightarrow t\ tok & Scan\ an\ item \\ s\ kwd \longleftrightarrow t\ tok & Scan\ a\ keyword \\ s\ id \longleftrightarrow t\ tok & Scan\ an\ identifier \\ s\ num \longleftrightarrow t\ tok & Scan\ a\ number \\ s\ spl \longleftrightarrow t\ tok & Scan\ a\ symbol \\ s\ lit \longleftrightarrow t\ tok & Scan\ a\ string\ literal \end{array} \tag{3.27}$$

Definicija form sodb, ki sledijo, uporablja številne pomožne sodbe, ki ustrezajo klasifikaciji znakov in leksikalni strukturi jezika. Na primer, $s\ whs$ pravi da se niz s sestoji samo iz "belih znakov", izjava $s\ lord$ pravi, da se s sestoji iz alfanumeričnih znakov.

$$\frac{}{\epsilon\ inp \longleftrightarrow \epsilon\ tokstr} \tag{3.28}$$

$$\frac{s = s_1\hat{}s_2\hat{}s_3\ str \quad s_1\ whs \quad s_2\ itm \longleftrightarrow\ t\ tok \quad s_3\ inp \longleftrightarrow ts\ tokstr}{s\ inp \longleftrightarrow t\ .\ ts\ tokstr} \tag{3.29}$$

$$\frac{s\ kwd \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.30}$$

$$\frac{s\ id \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.31}$$

$$\frac{s\ num \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.32}$$

$$\frac{s\ lit \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.33}$$

$$\frac{s\ spl \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.34}$$

$$\frac{s = l\,\centerdot\,e\,\centerdot\,t\,\centerdot\,\epsilon\ str}{s\ kwd \longleftrightarrow LET\ tok} \tag{3.35}$$

$$\frac{s = b\,\centerdot\,e\,\centerdot\,\epsilon\ str}{s\ kwd \longleftrightarrow BE\ tok} \tag{3.36}$$

$$\frac{s = i\,\centerdot\,n\,\centerdot\,\epsilon\ str}{s\ kwd \longleftrightarrow IN\ tok} \tag{3.37}$$

$$\frac{s = s_1\hat{}s_2\ str \quad s_1\ ltr \quad s_2\ lord}{s\ id \longleftrightarrow ID[s]\ tok} \tag{3.38}$$

$$\frac{s = s_1\hat{}s_2\ str \quad s_1\ dig \quad s_2\ dgs \quad s\ num \longleftrightarrow n\ nat}{s\ num \longleftrightarrow NUM[n]\ tok} \tag{3.39}$$

$$\frac{s = s_1\hat{}s_2\hat{}s_3\ str \quad s_1\ qum \quad s_2\ lord \quad s_3\ qum}{s\ lit \longleftrightarrow LIT[s_2]\ tok} \tag{3.40}$$

$$\frac{s = +\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow ADD\ tok} \tag{3.41}$$

$$\frac{s = *\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow MUL\ tok} \tag{3.42}$$

$$\frac{s = \hat{}\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow CAT\ tok} \tag{3.43}$$

$$\frac{s = (\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow LP\ tok} \tag{3.44}$$

$$\frac{s =)\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow RP\ tok} \tag{3.45}$$

$$\frac{s = |\,\centerdot\,\epsilon\ str}{s\ spl \longleftrightarrow VB\ tok} \tag{3.46}$$

Pravilo 3.38 velja samo, če ne velja nobeno izmed pravil od 3.35 do 3.37. Gledano tehnično ima pravilo 3.38 premise, ki izključijo ključne besede kot možene identifikatorje.

3.2.2 Kontekstno neodvisne gramatike

Standardna metoda za definicijo konkretne sintakse jezika je uporaba kontekstno neodvisne gramatike (okr. KNG). Gramatika se sestoji iz treh komponent:

1. žetonov ali *terminalnih simbolov* nad katerimi je gramatika definirana,
2. sintaktičnih razredov ali *neterminalnih simbolov*, ki se razlikujejo od terminalnih simbolov in
3. pravil ali produkcij oblike $A ::= \alpha$, kjer je A neterminalen simbol in je α niz terminalnih in neterminalnih simbolov.

Vsak sintaktični razred je kolekcija nizov žetonov. Pravila povejo kateri niz spada k katerem sintaktičnem razredu. Pri definiciji gramatike pogosto okrajšamo del produkcij tako, da združimo produkcije z istim sintaktičnim razredom.

$$\begin{aligned} A &::= \alpha_1 \\ &\;\;\vdots \\ A &::= \alpha_n \end{aligned} \tag{3.47}$$

Vse produkcije imajo enak levi del. Če sestavimo zgornje produkcije v eno samo dobimo:

$$A ::= \alpha_1 \mid ... \mid \alpha_n, \tag{3.48}$$

Takšno pravilo definira množico alternativ za razvoj sintaktičnega razreda A, ki so ločene s pokončno črto $|$.

Kontekstno neodvisna gramatika določa hkratno induktivno definicijo množice sintaktičnih razredov. Vsak neterminalni simbol ali sintaktični razred vidimo kot sodbo $s\ A$ nad nizi terminalnih simbolov.

Neterminalni simboli imajo *dvojno vlogo*: a) vlogo spremenljivk v produkciji in b) vlogo lastnosti oz. razreda objektov. Poglejmo si zdaj kako prevedemo pravila izražena v KNG v produkcije.

Za prikaz prevajanja pravil KNG v pravila izpeljave uporabimo splošno obliko pravila KNG. Objekti $s_1, \ldots, s_{n+1}$ predstavljajo terminalne simbole, medtem ko predstavljajo $A_1, \ldots, A_n$ sintaktične razrede, ki se naprej razvijejo v terminalne simbole.

$$A ::= s_1\ A_1\ s_2\ \ldots\ s_n\ A_n\ s_{n+1} \tag{3.49}$$

Pravilo prevedmo tako, da uporabljamo sintaktične razrede A_i kot sodbe s katerimi priredimo tip izpeljanim terminalnim simbolom s'_i. Vsak sintaktični razred (neterminalni simbol) je definiran z induktivno definicijo. Množica takšnih pravil sestavlja induktivno definicijo *statične semantike* jezika.

$$\frac{s'_1\ A_1\ \ldots s'_n\ A_n}{s_1\ s'_1\ s_2\ \ldots\ s_n\ s'_n\ s_{n+1}\ A.} \tag{3.50}$$

Spet se spomnimo, da je postavljanje nizov skupaj krajši zapis za konkatenacijo nizov. Pravilo lahko napišemo takole.

$$\frac{s'_1\ A_1\ \ldots\ s'_n\ A_n \quad s = s_1\hat{}s'_1\hat{}s_2\hat{}\ldots\hat{}s_n\hat{}s'_n\hat{}s_{n+1}}{s\ A} \tag{3.51}$$

Formulacija pove, da za s velja A, če lahko s razdelimo kot je opisano zgoraj: $s'_i\ \mathcal{A}_i$ za $1 \leq i \leq n$. Ker konkatenacija nizov ni enolično inverzibilna, dekompozicija niza ni unikatna in imamo lahko več različnih načinov po katerih ovrednotimo pravilo.

3.2.3 Gramatiča struktura

V okviru teorije programskih jezikov se bomo ukvarjali s statično oz. gramatično strukturo jezika—le-to bomo imenovali *statična semantika* jezika. Za specifikacijo statične strukture bomo uporabljali *abstraktno sintakso* jezika in se s tem izognili vsem problemom konkretne sintakse jezika.

S statično semantiko bomo definirali osnovno sintakso jezika ter preverjali statičen izračun tipov izrazov jezika. Z *dinamično semantiko* jezika bomo predstavili rezlične

vrste evaluacij in obdelav stavkov jezika. Več o statični in dinamični semantiki bo predstavljeno v naslednjem poglavju.

Statično semantiko $\mathcal{L}(\text{nat str})$ lahko definiramo s kontekstno neodvisno gramatiko nad simboli, ki so predstavljeni v 3.14. Gramatika ima samo en sintaktični razred, exp, ki je definiran z naslednjimi pravili:

$$\begin{array}{llll} Expression & exp & ::= & num \mid lit \mid id \mid LP\ exp\ RP \mid exp\ ADD\ exp \mid \\ & & & exp\ MUL\ exp \mid exp\ CAT\ exp \mid VB\ exp\ VB \mid \\ & & & LET\ id\ BE\ exp\ IN\ exp \\ Number & num & ::= & NUM[n] \quad (n\ nat) \\ String & lit & ::= & LIT[s] \quad (s\ str) \\ Identifier & id & ::= & ID[s] \quad (s\ str) \end{array} \tag{3.52}$$

Ta gramatika uporablja nekatere standardne konvencije za izboljšanje berljivosti: žeton identificiramo s spremenljivko in uporabljamo stik nizov za izražanje konkatenacije.

Če prikažemo interpretacijo gramatike z induktivnimi definicijami, dobimo naslednja pravila.

$$\frac{s\ num}{s\ exp} \tag{3.53}$$

$$\frac{s\ lit}{s\ exp} \tag{3.54}$$

$$\frac{s\ id}{s\ exp} \tag{3.55}$$

$$\frac{s_1\ exp \quad s_2\ exp}{s_1\ ADD\ s_2\ exp} \tag{3.56}$$

$$\frac{s_1\ exp \quad s_2\ exp}{s_1\ MUL\ s_2\ exp} \tag{3.57}$$

$$\frac{s_1\ exp \quad s_2\ exp}{s_1\ CAT\ s_2\ exp} \tag{3.58}$$

$$\frac{s\ exp}{VB\ s\ VB\ exp} \tag{3.59}$$

$$\frac{s\ exp}{LP\ s\ RP\ exp} \tag{3.60}$$

$$\frac{s1\ id \quad s2\ exp \quad s3\ exp}{LET\ s1\ BE\ s2\ IN\ s3\ exp} \tag{3.61}$$

$$\frac{n\ nat}{NUM[n]\ num} \tag{3.62}$$

$$\frac{s\ str}{LIT[s]\ lit} \tag{3.63}$$

$$\frac{s\ str}{ID[s]\ id} \tag{3.64}$$

Da bi poudarili vlogo konkatenacije nizov bi lahko napisali pravilo 3.57 na naslednji način.

$$\frac{s = s1\ MUL\ s2\ str \quad s1\ exp \quad s2\ exp}{s\ exp} \tag{3.65}$$

Izraz $s\ exp$ je izpeljiv, če je s konkatenacija niza s_1, znaka za množenje in s_2, kjer velja $s_1\ exp$ in $s_2\ exp$.

3.2.4 Kontekstno neodvisne gramatike in programski jeziki

Kontekstno neodvisne gramatike, ki so obravnavane v klasični šoli teorije avtomatov, jezikov in izračunljivosti [9, 8], predstavljajo *temeljni formalni jezik* za definicijo konkretne sintakse programskih jezikov.

V okviru teorije jezikov imamo množico specifičnih vrst gramatik, ki omogočajo enostavnejše prepoznavanje stavkav, ki so člani jezika dane gramatike. Na primer, *s-gramatike* so enostavne KNG, ki dovoljujejo za dan neterminalni simbol A eno samo produkcijo za dan začetek desne strani produkcije. Na ta način ima razčlenjevalni algoritem linearno kompleksnost.

Problemi, ki jih obravnava teorija jezikov so tudi dvoumnost jezikov in odpravljanje dvoumnosti v okviru KNG. Dvoumnosti se izognemo tako, da prepišemo gramatiko v drugo, nedvoumno obliko, ki dovoljuje eno samo drevo izpeljave za dan stavek.

Velikokrat želimo zapisati gramatiko v *normalni obliki*, preko katere lažje obravnavamo in vidimo lastnosti jezikov KNG. Najbolj znani normalni obliki gramatik sta Chomskijeva normalna oblika in Greibachova normalna oblika.

Normalne oblike ne vsebujejo marsikatere nadležne lastnosti gramatik kot so na primer produkcije enote in λ-produkcije, omogočajo pa tudi bolj enostavno teoretično in praktično analizo jezika[1]. Na primer, gramatika zapisana v Chomskijevi normalni obliki omogoča uporabo algoritma CYK za preverjanje članstva stavka v jeziku gramatike.

Kontekstno neodvisne gramatike torej služijo primarno kot orodje za specifikacijo in implementacijo programskih jezikov. Ena izmed zelo široko uporabljenih notacij za specifikacijo programskih jezikov je Backus-Naurjev zapis gramatike ali BNF je v osnovi enaka jezikom kontekstno neodisnim jezikom.

Kontekstno neodvisni jeziki so bolj podrobno obravnavani v okviru področja *prevajalnikov* in *tolmačev*. To področje seukvarja z množico zelo težkih problemov. Glavni problemi s katerimi se srečujemo pri implementaciji prevajalnikov in tolmačev so izločanje dvoumnosti gradnikov jezika, učinkovita implementacija razčlenjevanja in prevajanja jezika v bolj enostavne računalniške jezike.

Na primer, področje prevajalnikov in tolmačev se ukvarja s prevajanjem zelo kompleksnih gramatik, ki so še vedno v razredu KNG. V ta namen je bila razvita vrsta teretičnih in praktičnih orodij za razčlenjevanje kot so npr. LL in LR gramatike, ki navkljub kompleksnim gradnikom jezika omogočajo razčenjevanje v linearnem času [1]. Implementacija LL in LR gramatik je realizirana v okviru meta-programskih sistemih YACC in LEX.

Problemi, ki jih srečujemo pri razčenjevanju in prevajanju jezikov presegajo okvir zapiskov o teoriji programskih jezikov, ki se primarno ukvarja s *semantiko programskih jezikov*. Jeziki so v okviru teorije programskih jezkov predstavljeni z abstraktno sintakso. Pričakuje se, da ni problemov s katerimi se ukvarja področje prevajalnikov programskih jezikov.

[1] Bolj podrobne rezultate si lahko ogledate v [9].

3.3 Abstraktna sintaksa

Konkretna sintaksa jezika definira linearno predstavitev jezikovnih konstruktov. Program je predstavljen kot niz simbolov.

Osnovni namen konkretne sintakse je izražanje konceptov programskih jezikov na katerih sloni dan jezik. Pod koncepti programskih jezikov so mišljene abstrakcije, ki služijo kot osnovni gradniki za definicijo programa. Primeri pogosto uporabljenih abstrakcij programskih jezikov so funkcije, iteracija, polimorfizem, itn.

Konkretna sintakso lahko torej vidimo kot realizacijo abstrakcij s katerimi so predstavljeni koncepti programskih jezikov. V okviru zasnove konkretne sintakse se ukvarjamo z berljivostjo jezika in primernostjo jezika za avtomatsko obdelavo. Področje, ki se ukvarja z razčlenjevanjem, prevajanjem in interpretacijo jezikov so *prevajalniki in tolmači* [1].

Pri matematični obravnavi jezika nas zanima pomen jezika ter formalne lastnosti jezika in ne toliko površinske predstavitve. Stavke jezika želimo obravnavati čim bolj neodvisno od konkretne sintakse jezika—zanima nas predvsem statična struktura stavkov izražena z abstraktno sintakso.

Abstraktna sintaksa jezika prikaže hierarhično in povezovalno strukturo jezika ter zanemari linearno notacijo konkretne sintakse. Predstavitev programa v abstraktni sintaksi omogoča natančno predstavitev strukture stavka z uporabo dreves ali grafov.

Razčlenjevanje je proces prevajanja iz konkretne sintakse v abstraktno sintakso. Sestoji se iz analize linearne predstavitve jezika preko gramatike jezika ter prevajanja linearne predstavitve v abstraktno sintaksno drevo ali podobno obliko abstraktnega zapisa.

V naslednji sekciji si bomo ogledali abstraktno sintakso jezika $\mathcal{L}(\mathrm{natstr})$. Predstavljeno bo tudi pretvarjanje konkretne sintakse v abstraktno sintaksno drevo.

3.3.1 Abstraktna sintaksna drevesa

Abstraktno sintaksno drevo jezika $\mathcal{L}(\text{nat str})$ je definirano z naslednjo signaturo:

$$\begin{array}{l} ar(num[n]) = 0 \quad (n\ nat) \\ ar(str[s]) = 0 \quad (s\ str) \\ ar(id[s]) = 0 \quad (s\ str) \\ ar(plus) = 2 \\ ar(times) = 2 \\ ar(cat) = 2 \\ ar(len) = 1 \\ ar(let[s]) = 2 \end{array} \tag{3.66}$$

Vsak identifikator obravnavamo kot operator s številnostjo 0. Konstruktor 'let' je obravnavan kot družina operacij, ki imajo številnost 2 in so indeksirani z identifikatorjem, ki ga povezuje.

Pri specializaciji pravil za abstraktno sintaksna drevesa na predstavljeno signaturo dobimo naslednje induktivne definicije abstraktne sintakse $\mathcal{L}(\text{nat str})$:

$$\frac{n\ nat}{num[n]\ ast} \tag{3.67}$$

$$\frac{s\ str}{str[s]\ ast} \tag{3.68}$$

$$\frac{s\ str}{id[s]\ ast} \tag{3.69}$$

$$\frac{a_1\ ast \quad a_2\ ast}{plus(a_1; a_2)\ ast} \tag{3.70}$$

$$\frac{a_1 ast \quad a_2\ ast}{times(a_1; a_2)\ ast} \tag{3.71}$$

$$\frac{a_1\ ast \quad a_2\ ast}{cat(a_1; a_2)\ ast} \tag{3.72}$$

$$\frac{a\ ast}{len(a)\ ast} \tag{3.73}$$

$$\frac{s\ id \quad a_1\ ast \quad a_2\ ast}{let[s](a_1; a_2)\ ast} \tag{3.74}$$

Zadnje pravilo je specializacija pravila za 'let' stavek. Zahtevamo, da je prvi argument 'let' stavka identifikator.

3.3.2 Razčlenjevanje v abstraktno sintaksno drevo

Proces prevajanja iz konkretne sintakse v abstraktno sintakso se imenuje *razčlenjevanje* (angl. parsing).

Razčlenjevanje bomo obravnavali kot sodbo med konkretno in abstraktno sintakso jezika. Za to sodbo bo veljalo $(\forall, \exists!)$ za vse nize in ast.

Razčlenjevanje je torej parcialna funkcija med nizi in ast. Nedefinirana je za nize, ki nimajo pravilne gramatične strukture, preostale (pravilno strukturirane) nize pa enolično prevede v ast.

V nadaljevanju si bomo ogledali sodbe za razčlenjevanje jezika $\mathcal{L}(\text{nat str})$.

$s\ prg \longleftrightarrow a\ ast$	Parse as a program
$s\ exp \longleftrightarrow a\ ast$	Parse as an expression
$s\ trm \longleftrightarrow a\ ast$	Parse as a term
$s\ fct \longleftrightarrow a\ ast$	Parse as a factor
$s\ num \longleftrightarrow a\ ast$	Parse as a number
$s\ lit \longleftrightarrow a\ ast$	Parse as a literal
$s\ id \longleftrightarrow a\ ast$	Parse as an identifier

(3.75)

Sodbe za razčlenjevanje so definirane induktivno na osnovi sledečih pravil:

$$\frac{n\ nat}{NUM[n]\ num \longleftrightarrow num[n]\ ast} \tag{3.76}$$

$$\frac{s\ str}{LIT[s]\ lit \longleftrightarrow str[s]\ ast} \tag{3.77}$$

$$\frac{s\ str}{ID[s]\ id \longleftrightarrow id[s]\ ast} \tag{3.78}$$

$$\frac{s\ num \longleftrightarrow a\ ast}{s\ fct \longleftrightarrow a\ ast} \tag{3.79}$$

$$\frac{s\ lit \longleftrightarrow a\ ast}{s\ fct \longleftrightarrow a\ ast} \tag{3.80}$$

$$\frac{s\ id \longleftrightarrow a\ ast}{s\ fct \longleftrightarrow a\ ast} \tag{3.81}$$

$$\frac{s\ prg \longleftrightarrow a\ ast}{LP\ s\ RP\ fct \longleftrightarrow a\ ast} \tag{3.82}$$

$$\frac{s\ fct \longleftrightarrow a\ ast}{s\ trm \longleftrightarrow a\ ast} \tag{3.83}$$

$$\frac{s_1\ fct \longleftrightarrow a_1\ ast \quad s_2\ trm \longleftrightarrow a_2\ ast}{s_1\ MUL\ s_2\ trm \longleftrightarrow times(a_1; a_2)\ ast} \tag{3.84}$$

$$\frac{s\ fct \longleftrightarrow a\ ast}{VB\ s\ VB\ trm \longleftrightarrow len(a)\ ast} \tag{3.85}$$

$$\frac{s\ trm \longleftrightarrow a\ ast}{s\ exp \longleftrightarrow a\ ast} \tag{3.86}$$

$$\frac{s_1\ trm \longleftrightarrow a_1\ ast \quad s_2\ exp \longleftrightarrow a_2\ ast}{s_1\ ADD\ s_2\ exp \longleftrightarrow plus(a_1; a_2)\ ast} \tag{3.87}$$

$$\frac{s_1\ trm \longleftrightarrow a_1\ ast \quad s_2\ exp \longleftrightarrow a_2\ ast}{s_1\ CAT\ s_2\ exp \longleftrightarrow cat(a1; a2)\ ast} \tag{3.88}$$

$$\frac{s\ exp \longleftrightarrow a\ ast}{s\ prg \longleftrightarrow a\ ast} \tag{3.89}$$

$$\frac{s_1\ id\ \longleftrightarrow\ id[s]\ ast \quad s_2\ exp \longleftrightarrow a_2\ ast \quad s_3\ prg \longleftrightarrow a_3\ ast}{LET\ s_1\ BE\ s_2\ IN\ s_3\ prg \longleftrightarrow let[s](a_2; a_3)\ ast} \tag{3.90}$$

Uspešno razčlenjvanje pogojuje, da se niz žetonov razčleni v skladu s pravili gramatike, ki ni dvoumna, in da je rezultat razčlenjevanja dobro-definirano abstraktno sintaktično drevo.

Izrek 3.3.1 *Če $s\ prg \longleftrightarrow a\ ast$, potem $s\ prg$ in $a\ ast$. Enako velja za vse ostale sodbe.*

Velja celo več, če je niz generiran po pravilih gramatike, potem ga je mogoče razčleniti v ast.

Izrek 3.3.2 *Če $s\ prg$, potem obstaja enoličen a tako da $s\ prg \longleftrightarrow a\ ast$. Enako velja za ostale sodbe, ki opišejo razčlenjevanje: sodbe imajo lastnost ($\forall\exists!$) za vse dobro-definirane nize in ast.*

Končno, vsak zapis abstraktne sintakse je mogoče preoblikovati v niz simbolov, ki se razčleni v skladu z danim ast.

Izrek 3.3.3 *Če velja $a\ ast$, potem obstaja (ni nujno enolično) niz s tako, da velja $s\ prg$ in $s\ prg \longleftrightarrow a\ ast$. Z drugimi besedami, sodba za razčlenjevanje ima lastnost ($\exists\forall$).*

3.4 Primer enostavnega jezika

V tej sekciji bo predstavljen primer enostavnega jezika $\mathcal{L}(\text{nat bool})$, leksikalna struktura, leksikalna analiza ter gramatika $\mathcal{L}(\text{nat bool})$ v obliki pravil.

3.4.1 Jezik $\mathcal{L}(\textbf{nat bool})$

Ogledali si bomo zelo enostaven jezik, ki vsebuje samo nekatere izmed sintaktičnih oblik:

- boolove konstante $true$ in $false$,
- pogojni izraz if $-$ then $-$ else,

- numerično konstanto 0,
- aritmetične operacije $pred$ in $succ$ ter
- pogojni izraz iszero, ki preveri al ima izraz vrednost 0.

Gramatika $\mathcal{L}(\text{nat bool})$ je definirana z naslednjimi izrazi, ki predstavljajo konkretno in abstraktno sintakso. Razlika med obema je minimalna v primeru jezika $\mathcal{L}(\text{nat bool})$.

$$\begin{array}{lcll} \text{Tipi } t & ::= & \text{nat} & \text{nat} \\ & & \text{bool} & \text{bool} \\ \text{Izrazi } e & ::= & \text{bool}[t] & \text{true} \\ & & \text{bool}[f] & \text{false} \\ & & \text{if}(b, e_1, e_2) & \text{if } b \text{ then } e_1 \text{ else } e_2 \\ & & \text{nat}[z] & 0 \\ & & \text{succ}(e) & e+1 \\ & & \text{pred}(e) & e-1 \\ & & \text{iszero}(e) & \text{iszero}(e) \end{array} \tag{3.91}$$

Jezik $\mathcal{L}(\text{nat bool})$ uporablja tipe nat in bool. Konkretni primerki teh dveh tipov so cela števila izražena v Churchevi obliki in boolove vrednosti true in false. Edini strukturiran jezikovni gradnik jezika $\mathcal{L}(\text{nat bool})$ je if stavek, ki uporablja funkcijo iszero(). Rezultat evaluacije izrazov so vedno enostavne vrednosti: boolove vrednosti ali cela števila. V naslednjem poglavju si bomo ogledali tudi pravila za evaluacijo izrazov jezika $\mathcal{L}(\text{nat bool})$.

Primer 3.4.1 *Poglejmo si nekaj primerov izrazov v predstavljenem jeziku.*

$$\begin{array}{l} \textit{succ}(\textit{succ}(\textit{succ}(0))) \\ \blacktriangleright\ 3 \\ \textit{if } false \textit{ then } 0 \textit{ else } 1 \\ \blacktriangleright\ 1 \\ \textit{iszero}(\textit{pred}(\textit{succ}(0))) \\ \blacktriangleright\ true \end{array} \tag{3.92}$$

Najprej, cela števila predstavimo z večkratnim vgnezdenjem operacije succ ter pred. Drugi primer prikaže uporabo if stavka. Tretji primer uporabi operacijo iszero nad celim številom. □

3.4.2 Leksikalna analiza

Leksikalna struktura $\mathcal{L}$(**nat bool**)

Znakovna predstavitev je v največih primerih definirana z regularnimi izrazi. Leksikalna struktura $\mathcal{L}(\text{nat bool})$ je določena z naslednjimi pravili.

$$\begin{array}{lllll} Item & itm & ::= & kwd \mid boo \mid zer \mid spl & \\ Keyword & kwd & ::= & i\,.\,f\,.\,\epsilon \mid & \\ & & & t\,.\,h\,.\,e\,.\,n\,.\,\epsilon \mid & \\ & & & s\,.\,u\,.\,c\,.\,c\,.\,\epsilon \mid & \\ & & & p\,.\,r\,.\,e\,.\,d\,.\,\epsilon \mid & (3.93) \\ & & & i\,.\,s\,.\,z\,.\,e\,.\,r\,.\,o\,.\,\epsilon \mid & \\ Bool & boo & ::= & true \mid false & \\ Special & spl & ::= & (\mid) & \\ Zero & zer & ::= & 0 & \end{array}$$

Jezik žetonov $\mathcal{L}$(**nat bool**)

Vhoden niz znakov se pretvarja v niz simbolov (žetonov) z uporabo naslednjih pravil, ki definirajo simbolni jezik razčlenjevalnika oz. statično semantiko jezika.

$$\frac{}{ZERO\ tok} \tag{3.94}$$

$$\frac{l \text{ bool}}{BOOL[l]\ tok} \tag{3.95}$$

$$\frac{}{IF\ tok} \tag{3.96}$$

$$\frac{}{THEN\ tok} \tag{3.97}$$

$$\frac{}{ELSE\ tok} \tag{3.98}$$

$$\frac{}{SUCC\ tok} \tag{3.99}$$

$$\frac{}{PRED\ tok} \tag{3.100}$$

$$\frac{}{ISZERO\ tok} \tag{3.101}$$

$$\frac{}{LP\ tok} \tag{3.102}$$

$$\frac{}{RP\ tok} \tag{3.103}$$

Pretvorba $\mathcal{L}$(nat bool) v nize žetonov

Naslednja pravila predstavlja leksikalno razčlenjevanje stavkov jezika $\mathcal{L}$(nat bool) v niz žetonov.

Pravila na začetku so definirana za usmerjanje pretvorbe izrazov $\mathcal{L}$(nat bool) v žetone.

$$\frac{}{\epsilon\ inp \longleftrightarrow \epsilon\ tokstr} \tag{3.104}$$

$$\frac{s = s_1\hat{}s_2\hat{}s_3\ str \quad s_1\ whs \quad s_2\ itm \longleftrightarrow t\ tok \quad s_3\ inp \longleftrightarrow ts\ tokstr}{s\ inp \longleftrightarrow t\ .\ ts\ tokstr} \tag{3.105}$$

$$\frac{s\ kwd \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.106}$$

$$\frac{s\ boo \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.107}$$

$$\frac{s\ spl \longleftrightarrow t\ tok}{s\ itm \longleftrightarrow t\ tok} \tag{3.108}$$

$$\frac{s = i\ .\ f\ .\ \epsilon\ str}{s\ kwd \longleftrightarrow IF\ tok} \tag{3.109}$$

$$\frac{s = t\ .\ h\ .\ e\ .\ n\ .\ \epsilon\ str}{s\ kwd \longleftrightarrow THEN\ tok} \tag{3.110}$$

$$\frac{s = e \,.\, l \,.\, s \,.\, e\epsilon \; str}{s \; kwd \longleftrightarrow ELSE \; tok} \tag{3.111}$$

$$\frac{s = \; t \,.\, r \,.\, u \,.\, e \,.\, \epsilon \; str}{s \; boo \longleftrightarrow BOOL[t] \; tok} \tag{3.112}$$

$$\frac{s = \; f \,.\, a \,.\, l \,.\, s \,.\, e \,.\, \epsilon \; str}{s \; boo \longleftrightarrow BOOL[f] \; tok} \tag{3.113}$$

$$\frac{s = \; 0 \,.\, \epsilon \; str}{s \; zer \longleftrightarrow ZERO \; tok} \tag{3.114}$$

$$\frac{s = (\!.\epsilon \; str}{s \; spl \longleftrightarrow LP \; tok} \tag{3.115}$$

$$\frac{s =) \,.\, \epsilon \; str}{s \; spl \longleftrightarrow RP \; tok} \tag{3.116}$$

3.4.3 Razčlenjevanje $\mathcal{L}(\textbf{nat bool})$

V prejšnjem razdelku smo definirali kako se nizi znakov pretvarjajo v nize simbolov. Nize simbolov je naprej potrebno razčleniti v skladu z gramatično strukturo jezika $\mathcal{L}(\text{nat bool})$.

Najprej bomo definirali gramatično strukturo jezika v BNF obliki. Zapis bomo pretvorili v pravila s katerim je definirana statična struktura jezika $\mathcal{L}(\text{nat bool})$.

Gramatična struktura $\mathcal{L}(\textbf{nat bool})$

Konkretno sintakso $\mathcal{L}(\text{nat bool})$ lahko definiramo s kontekstno neodvisno gramatiko nad simboli, ki so predstavljeni v 3.93. Spet ima gramatika ima en sam sintaktični razred, exp, ki je definiran z naslednjimi pravili:

$$\begin{array}{llll} Expression & exp & ::= & zer \mid boo \mid \\ & & & \text{if } exp \text{ then } exp \text{ else } exp \mid \\ & & & \text{succ } exp \mid \text{pred } exp \mid \\ & & & \text{iszero } exp \mid LP\ exp\ RP \\ Boolean & boo & ::= & BOOL[l] \qquad (l \text{ bool}) \\ Zero & zer & ::= & ZERO \end{array} \tag{3.117}$$

Če prikažemo interpretacijo gramatike z induktivnimi definicijami, dobimo naslednja pravila.

$$\frac{s\ zer}{s\ exp} \tag{3.118}$$

$$\frac{s\ boo}{s\ exp} \tag{3.119}$$

$$\frac{s_1\ exp \quad s_2\ exp \quad s_3\ exp}{\text{if } s_1 \text{ then } s_2 \text{ else } s_3} \tag{3.120}$$

$$\frac{s\ exp}{\text{succ } s\ exp} \tag{3.121}$$

$$\frac{s\ exp}{\text{pred } s\ exp} \tag{3.122}$$

$$\frac{s\ exp}{\text{pred } s\ exp} \tag{3.123}$$

$$\frac{s\ exp}{LP\ s\ RP\ exp} \tag{3.124}$$

$$\frac{l \text{ bool}}{BOOL[l]\ boo} \tag{3.125}$$

$$\frac{}{ZERO\ zer} \tag{3.126}$$

3.5 Opombe

Poglavje vsebuje prevode izbranih sekcij iz učbenika *Practical Foundations for Programming Languages* [6] avtorja Roberta Harperja.