S00-03 Formális szemantika

Tartalom

1. Formális kontra informális definíciók, a formális szemantika alkalmazási területei, a szemantikamegadási módszerek áttekintése
2. Mesterséges nyelvek konkrét és absztrakt szintaxisa
3. Statikus és dinamikus szemantika
4. Attribútum-grammatikák és alkalmazásaik
5. Alapvető imperatív nyelvi elemek strukturális és természetes műveletei, illetve leíró szemantikája, hasonlóságok és különbségek
6. Kompozicionális és strukturális indukció
7. Rekurzív függvények és ciklusok leíró szemantikája, fixpont-elmélet
8. További források

1. Formális kontra informális definíciók, a formális szemantika alkalmazási területei, a szemantikamegadási módszerek áttekintése

Bevezetés

- Nyelvek informális specifikációjának fő gondja az, hogy félreérthető, ez a programnyelvek szemantikájának megadásakor is előjött.
  - Általában angolul adták meg a dokumentációkban, természetes nyelvű leírással
  - Például `if <cond> then <stm> fi`
  - "Itt a fordító, ezt meg azt csinálja"
- Informális leírás problémát okoz a fordító tervezőknek és a programozóknak is.
  - Ellenben matematikával és matematikai logikával
- A formális megadás komoly feladat: dokumentációt ad, a rossz koncepciók kijönnek $\Longrightarrow$ precíz!
  - Viszont nehezebb készíteni és bonyolultabb megérteni
- Alkalmazási területei:
  - Fordítóprogramok
  - Parser generátorok
  - Generátor generátorok
  - Fordító fordítók

Formális szemantika: programok jelentésének

- szigorú
- precíz
- egyértelmű
- matematikai definíciója

Formális szemantika előnyei

1. Alapvető dokumentáció
2. Félreérthető elemek feltárása
3. Precíz jelentésfogalom $\Longrightarrow$ bizonyíthatóság

Három fő komponens

1. Szintaxis
   - szimbólumok sorozatának megadása, ami jelent is valamit
2. Szemantika
   - helyesen formált programok hatása
3. Pragmatika
   - olvasható, konvencionális, hatékony kód írásának módja

Megközelítései

1. **Attribútum grammatikákkal**
   - Átírási szemantika megadása
   - Másik nyelvre fordítunk át
   - Statikus szemantika elemzéshez szükséges
   - Alkalmazás: grammatikák írása, szintaktikus-, szemantikus parser és kódgenerátor előállítása fordítóhoz
2. **Operációs (műveleti) szemantikával**
   - Azoknak, akik tudni akarják hogyan hajtódik végre a program
   - Konfigurációt készít a programhoz
   - Két fajtája
     - Strukturális (Small-step): program végrehajtása lépésről lépésre
     - Természetes (Big-step): program végrehajtása kezdő állapotból végállapotba
   - Alkalmazás: fordítók és értelmezők készítése
3. **Denotációs (leíró) szemantikával**
   - Nem érdekel hogyan hajtódik végre $\Longrightarrow$ hanem hogy milyen eredmények lesznek
   - Programok és részeinek leképzése matematikai jelölésekre
   - Nyelvi elemek jelentésének megadása kompozicionálisan (*lásd 6. fejezet*)
   - Absztraktabb az operációs szemantikánál
   - Alkalmazás: nyelvtervezés és helyességbizonyítás
4. **Axiomatikus szemantikával**
   - Program jelentésének specifikálása elő- és utófeltétellel (Hoare-hármasok)
   - Még a denotációs szemantikánál is absztraktabb

2. Mesterséges nyelvek konkrét és absztrakt szintaxisa

Konkrét szintaxis

- "Hogy írok jó mondatokat?"
- Megadja, hogy a programnyelvben, hogy lehet mondatokat, kifejezéseket leírni.
- Pontos, konkrét leírást ad a nyelvi elemekről.
- Szintaktikus elemzőt, lexikális elemzőt (*lexical analyzer*) tudunk írni ezzel a jelöléssel

```
<assignment> ::= 'LET' <variable> ':=' <expression> ';'
```

Absztrakt szintaxis

- "Milyen szintaktikus kategóriák, elemek, konstrukciók vannak?"
- Ez az absztrakt specifikáció adja a szemantikák definíciójának alapját
- Ez inkább matematikai leírás
- Absztrakt szintaxisfák felépítésének megadása

$$A :== assignment(V, E)$$

ahol

- A: értékadás nemterminálisa (assigment)
- V: változó nemterminálisa (variable)
- E: kifejezés nemterminálisa (expression)

3. Statikus és dinamikus szemantika

Szintaxisnak nem minden elemét lehet környezetfüggetlen grammatikával kifejezni

Statikus szemantika

- "Környezetfüggő szintaxis", a szintaxis és szemantika határa
- Szintaxishoz tartozik, "mi számít értelmes mondatnak?"
- Olyan tulajdonságok $\Longrightarrow$ amiket legtöbb esetben a fordító ellenőriz (fordítási időben való ellenőrzés)
- Kifejezőbb jelölésrendszer kell leírásukhoz mint környezetfüggetlen grammatika. Ezért itt környezetfüggő grammatika van

Dinamikus szemantika

- Mi a végső eredmény? Mit hajt végre a program?
- "Futásidejű szemantika"
- Például mi lesz `x + y` kiértékelése futásidőben?

Példák

- `"4" == 4` erősen típusos nyelvekben (C, C++, Haskell)
  - $< Exp >==< Exp >$
  - Nem szintaktikai, hanem statikus szemantikai hiba
- `"4" == 4` gyengén típusos nyelvekben (Python, JavaScript)
  - Van olyan nyelv, ami kasztolja 4-et és `true`-t ad
  - Van olyan nyelv, ami meg `false`-t
  - Statikus szemantika engedi
  - Dinamikus szemantika nem
- `"4" == "4"`
  - Java nyelvben: Java elfogadja
  - Ám nem sztringek értékét hasonlítja össze, hanem String objektum referenciákat
  - Statikus szemantika engedi
  - Dinamikus szemantika is engedi
  - **DE!** Nem azt csinálja amit szeretnénk! Nem egyértelmű!

4. Attribútum-grammatikák és alkalmazásaik

- A szintaxist környezetfüggetlen grammatikákkal adják meg általában.
- Általában nagyobb nyelvet generál, mint amit a fordító elfogad.
- Kifejezi, hogy hogy kell egy kifejezést megkonstruálni, egymással összekombinálni, miként kell használni.

Környezetfüggetlen grammatika

$$G = (T, N, P, S)$$

ahol

- G: maga a környezetfüggetlen grammatika
- T: terminálisok (nyelv ábécéje)
- N: nem-terminálisok (szintaktikus kategóriák)
- P: létrehozási szabályok halmaza (produkciós szabályok, *production rules*)
  - p produkciós szabályok A nemterminálisból ($A \in N$) állítanak elő α szintaktikai elemet ($\alpha \in (T \cup N)*$, α lehet nemterminális vagy terminális)

  - $\forall p \in P : p \equiv A \to \alpha$
- S: kezdőllapot, eleme N-nek
  - eleme a nem-terminálosoknak ($S \in N$)
  - innen kezdünk el levezetési fákat gyártani

Mitől környezetfüggetlen egy környezetfüggetlen grammatika?

- p produkciós szabály bal oldalán: A nemterminális áll
- p produkciós szabály jobb oldalán: α terminális vagy nemterminális áll

Attribútum grammatika

- A szintaxisfa kidekorálásra kerül és minden csomópont rekorddá válik.
- Mire jók? Minek?
  - Parser generátorok
  - Kódgenerálás (Visitor design pattern)

Attribútum grammatika formálisan

Hogy csinálunk attribútum grammatikát?

- Veszünk egy G környezetfüggetlen grammatikát és hozzádobunk egy ARC-ot!
- *Attributes, Rules, Conditions*

$$AG = (G, A, R, C)$$

ahol

- AG: maga az attribútum grammatika
- G: egy környezetfüggetlen grammatika
- A *(Attributes)*: az attribútumok halmaza
  - szimbólumokhoz tulajdonságok (attribútumok) rendelése
- R *(Rules)*: az attribútumszámítási szabályok
  - egyszerű függvények
  - paraméterük az attribútum
- C *(Conditions)*: az attribútumszámítási feltételek
  - feltétel hogy milyen attribútum megengedett
  - ha az attribútum kiszámítása után teljesül $\Longrightarrow$ boldog vagyok

Attribútumok

- Attribútumokat rendelünk midnen T és N-beli terminálishoz és nemterminálishoz

Jelölések:

- X: terminális vagy nem terminális szimbólum
- *Attr*: attribútum
- *A(X)*: X szimbólum attribútumainak halmaza
- *X.Attr* vagy *Attr(X)*: X szimbólum *Attr* nevű attribútuma
  - a két jelölés felcserélhető

Attribútumok típusai

Szintetizált

- Ha p egy olyan $p = A \to \alpha$ szabály
  - ahol $r \in R(p)$ attribútumszámítási szabály beállítja *A.Attr*-t
  - tehát az *A.Attr* egy olyan p szabályban van $\to$ ahol A a **bal oldalon** van.
- Az attribútumfában felfelé szállít infót (gyerekből szülőbe)

Örökölt

- Ha p egy olyan $p = A \to \alpha X \beta$ szabály (α: terminális, X: nemterminális, β: terminális)
  - ahol $r \in R(p)$ attribútumszámítási szabály beállítja *X.Attr*-t
  - tehát az *X.Attr* egy olyan p szabályban van $\to$ ahol X a **jobb oldalon** van.
- Az attribútumfában lefelé szállít infót (szülőből gyerekbe)
- Például kontextus megadása

Egyéb tudnivalók az attribútumok típusairól:

- Egy szimbólum minden attribútumát maximum egy szabály számíthatja!
- Egy attribútum egyszerre szintetizált és örökölt nem lehet! Egyszerre csak az egyik!
- Levezetési fa gyökerében lévő kezdő S szimbólumnak nem lehet örökölt attribútuma
  - Mert az ős kitől örökölne attribútumot?

Jól definiált attribúm grammatika (Well-defined Attributum Grammatics, WAG)

- Minden származtatási fához létezik számítható attribútum sorrend
  - minden feltétel kiértékelhető
  - egyszerűen implementálható, de nem determinisztikus algoritmussal
- Az attribútum függőségek körmentesek!
- WAG speciális osztályai
  - S attribútum grammatika
    - csak szintetizált attribútumok
    - alulról felfelé bejárás
  - L attribútum grammatika

    * szintetizált és örökölt attribútumokat is tartalmazhat
    * felülről lefelé bejárás
    * egyszeri bejárással kiszámíthatók az attribútumok

5. Alapvető imperatív nyelvi elemek strukturális és természetes műveleti, illetve leíró szemantikája, hasonlóságok és különbségek

Állapotok a szemantikában

- Az utasítások és kifejezések változókat tartalmaznak.
- $s \in State = Var \to \mathbb{Z}$, azaz
  - s egy függvény *State* függvényhalmazban
  - ami változóhoz (Var) egész értéket rendel ($\mathbb{Z}$)
- Változó olvasása: *s[x]*
  - x változó értéke az s állapotban
- Változó értékadása: $s[y \to v]$
  - s állapotban y változó értéke v lesz

Ebből: $(s[y \to v])[x]$

- v, ha $x = y$
- vagy *s[x]* minden egyéb esetben

(*Egy másik formalizáltabb felírásban ami nincs a jegyzetben, de annak jó aki így könnyebben megtanulja:*)

$$(s[y \to v])[x] = \begin{cases} v' & \text{ha } x = y' \\ s[x] & \text{egyébként} \end{cases}$$

Strukturális operációs szemantika

- Lépésről lépésre hajtódnak végre az állapotok.
- Az átmenetek az általános következtetéssel definiálhatóak, feladata egy végeredmény meghatározása.

Átmenet

- Köztes lépés a számításban:

$$\langle S, s\rangle \Longrightarrow \langle S', s'\rangle \qquad s, s' \in State$$

- S végrehajtásának végetérése s' állapotot eredményezve:

$$\langle S, s\rangle \Longrightarrow s' \qquad s, s' \in State$$

Skip strukturális operációs szemantikája

$$\frac{}{\langle \textbf{skip}, s\rangle \Longrightarrow s}$$

- s állapotban kiértékelem **skip**-et és s-be kerülök
- **NEM AZ VAN** hogy "nem csinál semmit"

Értékadás strukturális operációs szemantikája

$$\frac{}{\langle x := a, s\rangle \Longrightarrow s[x \to A[\![a]\!]s]}$$

- Ha s állapotban kiértékelem $x := a$-t: teljesen kiértékelem a jobb oldalt
- x frissítése: A kifejezés s állapotban lévő a szintaktikai elem kiértékelése

Szekvencia strukturális operációs szemantikája

Köztes állapot:

$$\frac{\langle S_1, s\rangle \Longrightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Longrightarrow \langle S_1'; S_2, s\rangle}$$

Végállapot:

$$\frac{\langle S_1, s\rangle \Longrightarrow s'}{\langle S_1; S_2, s\rangle \Longrightarrow \langle S_2, s'\rangle}$$

Elágazás strukturális operációs szemantikája

Ha igaz:

$$\frac{}{\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Longrightarrow \langle S_1, s\rangle} \quad B[\![b]\!]s = tt$$

Ha hamis:

$$\frac{}{\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Longrightarrow \langle S_2, s\rangle} \quad B[\![b]\!]s = ff$$

- Mielőtt kiértékelem az ágat, előtte a feltételt értékelem ki

Ciklus strukturális operációs szemantikája

$$\frac{}{\langle \textbf{while } b \textbf{ do } S, s\rangle \Longrightarrow \langle \textbf{if } b \textbf{ then } (S;\ \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle}$$

- Ha feltétel teljesül
  - Megcsinálja S-t
  - Majd iterál újra
- Ha feltétel nem teljesül
  - Nem csinál semmit, megy végállapotba

Szemantikus ekvivalencia (strukturális operációs szemantika)

Szemantikus ekvivalencia általában: megadja, hogy két utasítás jelentése megegyezik-e

Strukturális operációs szemantika esetén

- Nem akkor egyezik meg, ha ugyanaz a programszöveg
- Akkor egyezik meg, ha ugyanabba a konfigurációba jutok

S_1 és S_2 szemantikusan ekvivalensek ($S_1 \equiv S_2$), ha minden s állapotra

- $\langle S_1, s\rangle \Longrightarrow^* c$ akkor és csak akkor $\langle S_2, s\rangle \Longrightarrow^* c$
- $\langle S_1, s\rangle \Longrightarrow^* \infty$ akkor és csak akkor $\langle S_2, s\rangle \Longrightarrow^* \infty$

ahol c terminál vagy zsákutca konfiguráció

- Ha egy program egy adott állapotból terminál, akkor a másiknak is terminálnia kell ugyanabban a konfigurációban
- Ha ez egyik program nem terminál, akkor a másiknak sem kell

Szemantikus függvény (strukturális operációs szemantika)

$$S_{SOS} : Stm \to (State \hookrightarrow State)$$

$$S_{SOS}[\![S]\!]s = \begin{cases} s' & \text{ha } \langle S, s\rangle \Longrightarrow^* s' \\ undefined & \text{egyébként} \end{cases}$$

- Ha normál végrehajtás történik, akkor véges lépésben eljutunk s'-be
- ha végrehajtás elakad vagy divergál akkor nem definiált hogy mi lesz

Természetes operációs szemantika

- Egy reláció a kezdő és végállapot között
- Nincs köztes átmenet se beragadt konfiguráció

S utasítás végrehajtása s' állapotot eredményezi

- $\langle S, s\rangle \to s' \qquad s, s' \in Sate$

Következtetési szabályai:

Skip természetes operációs szemantikája

$$\frac{}{\langle \textbf{skip}, s\rangle \Longrightarrow s}$$

Értékadás természetes operációs szemantikája

$$\frac{}{\langle x := a, s\rangle \Longrightarrow s[x \to A[\![a]\!]s]}$$

Szekvencia természetes operációs szemantikája

$$\frac{\langle S_1, s\rangle \Longrightarrow s' \qquad \langle S_2, s'\rangle \Longrightarrow s''}{\langle S_1; S_2, s\rangle \Longrightarrow \langle s''\rangle}$$

Elágazás természetes operációs szemantikája

$$\frac{\langle S_1, s\rangle \to s'}{\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Longrightarrow s'} \quad B[\![b]\!]s = tt$$

$$\frac{\langle S_2, s\rangle \to s'}{\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Longrightarrow s'} \quad B[\![b]\!]s = ff$$

Ciklus természetes operációs szemantikája

$$\frac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle \textbf{while } b \textbf{ do } S, s\rangle \to s''} \quad B[\![b]\!]s = tt$$

$$\frac{}{\langle \textbf{while } b \textbf{ do } S, s\rangle \to s} \quad B[\![b]\!]s = ff$$

Szemantikus ekvivalencia (természetes operációs szemantika)

S_1 és S_2 szemantikusan ekvivalensek, ha minden s és s' állapotra

- $\langle S_1, s\rangle \to s'$ akkor és csak akkor, ha $\langle S_2, s\rangle \to s'$

Szemantikus függvény (természetes operációs szemantika)

$$S_{NS} : Stm \to (State \hookrightarrow State)$$

$$S_{NS}[\![S]\!]s = \begin{cases} s' & \text{ha } \langle S, s\rangle \to s' \\ undefined & \text{egyébként} \end{cases}$$

Denotációs szemantika

- Matematikai objektumok segítségével megadja a program jelentését
- A matematikai objektumok adják a nyelvi elemek jelentését.
- S meghatároz egy parciális függvényt az állapotokra $\Longrightarrow$ ez S denotációja.
- A szemantikus függvények axiómákkal definiáltak
- A szemantikus függvény kulcskérdés és nem plusz lépés!

$$S_{ds} : Stm \to (State \to State)$$

Imperatív nyelvi elemek denotációs szemantikája

$$\begin{aligned}
S_{ds}[\![\textbf{skip}]\!] &= id_{State} \\
S_{ds}[\![x := a]\!]s &= s[x \mapsto A[\![a]\!]s] \\
S_{ds}[\![S_1; S_2]\!] &= S_{ds}[\![S_2]\!] \circ S_{ds}[\![S_1]\!] \\
S_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!] &= cond(B[\![b]\!], S_{ds}[\![S_1]\!], S_{ds}[\![S_2]\!]) \\
S_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &= FIX\ F \\
where\ F\ g\ &= cond(B[\![b]\!], g \circ S_{ds}[\![S]\!], id_{State})
\end{aligned}$$

6. Kompozicionális és strukturális indukció

Kompozicionalitás

- Minden szintaktikai elemhez megadunk egy függvényklózt
- Összetett kifejezések jelentése $\Longrightarrow$ részkifejezések jelentése alapján
- Denotációs szemantikában mindig a részkifejezéseket értékeljük ki először és az alapján a teljes kifejezést
- Például aritmetikai negációt sokszor elhibázzák. Igazából

$$A[\![-a]\!]s = A[\![0 - a]\!]s$$

- Kompozicionalitás kell a strukturális indukciós bizonyításokhoz

Mi bizonyítható a strukturális indukcióval?

1. **A definíció teljes**
   - Minden lehetséges kifejezésnek megadtuk a jelentését
2. **A szemantika konzisztens**
   - Ha egy kifejezéshez két különböző jelentés is levezethető $\Longrightarrow$ akkor azok ekvivalensek
3. A kifejezésekre definiált **operációs és denotációs szemantika ekvivalens**

7. Rekurzív függvények és ciklusok leíró szemantikája, fixpont-elmélet

- Logikusnak tűnik a ciklus szemantikáját az elágazás, szekvencia és skip szemantikájára visszavezetni

$$S_{ds}[\![\textbf{while } b \textbf{ do } S]\!] = cond(B[\![b]\!], S_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ S_{ds}[\![S]\!], id_{State})$$
$$g = cond(B[\![b]\!], g \circ S_{ds}[\![S]\!], id_{State})$$

- Ez mégsem jó
  - ciklus szemantikájához felhasználjuk megint a ciklus szemantikáját
  - A szabály nem kompozicionális. Kompozicionálisan kell definiálni a jelentését.

Megoldás: $S_{ds}[\![\textbf{while } b \textbf{ do } S]\!]$ jobb oldalát alakítsuk át magasabbrendű függvénnyé, aminek paramétere g függvény!

$$F : (State \hookrightarrow State) \to (State \hookrightarrow State)$$
$$Fg = cond(B[\![b]\!], g \circ S_{ds}[\![S]\!], id_{State}) = g$$

- $F\ g$ akkor egyenlő g-vel, ha g a fixpont
- Ha F magasabbrendű függvény fixpontját megtaláljuk: az kielégíti a *cond()*-ot

Formula megoldásait tehát F fixpontjának kiszámításával kapjuk meg:

$$S_{ds}[\![\textbf{while } b \textbf{ do } S]\!] = FIX\ F$$

8. További források

- Korábbi záróvizsga tételek