Bevezetés

Horpácsi Dániel
daniel-h@elte.hu

Programozási nyelvek formális definíciói:

Motiváció, alapfogalmak, módszerek

Nyelvek

- A gondolatainkat szavakban, mondatokban fejezzük ki
- A helyesen megalkotott mondatoknak (egyértelmű) jelentése van
- Akik beszélik ugyanazt a nyelvet, kommunikálhatnak egymással

- A lingvisztika a következő kérdésekre próbál választ adni:
  - Hogyan értjük meg a nyelvet, a mondatokat?
  - Pontosan mitől lesz egy mondat “helyes”, érthető?
  - Hogyan lehet precízen meghatározni egy mondat jelentését?
  - Hogyan lehet reprezentálni a jelentésfogalmat?
  - Lehetséges a megértés folyamatát automatizálni?

Programozási nyelvek

- A program tulajdonképpen egy speciális kommunikáció ember és gép között, amelynek célja a számítógép vezérlése
- A programokat nem természetes, hanem mesterséges nyelven írjuk:
  - A természetes nyelvek rendkívül komplexek, mondataik sokszor többféleképp értelmezhetőek, így nehéz őket formalizálni, elemezni, illetve automatizálni a megértésüket
  - A géppel történő kommunikáció speciális problémák és algoritmusok közvetítése, amelyhez egyszerűbb, specifikusabb nyelvek is elegendőek
  - A közlés általában imperatív: a program tulajdonképpen utasítások sorozata, amelyeket egyenként végre kell hajtani

Programozási nyelvek definíciója

Nyelvek definícióit általában három fő komponenssel adjuk meg:

Szintaxis

Mely mondatokat tekintjük jól formáltnak?

Azaz, mely szimbólumsorozatoknak van értelme, jelentése?

Szemantika

Mit jelentenek a helyesen formált mondatok?

Azaz, mi a hatásuk, hogyan lehet őket végrehajtani?

Pragmatika

Hogyan lehet olvasható, konvencionális, hatékony kódot írni?

(Ebben a kurzusban ezzel az aspektussal keveset foglalkozunk.)

A szintaxist, a helyes mondatokat környezetfüggetlen és attribútum grammatikákkal viszonylag könnyen le tudjuk írni, de a dinamikus szemantika formális megadása már nehezebb feladat.

Példa.

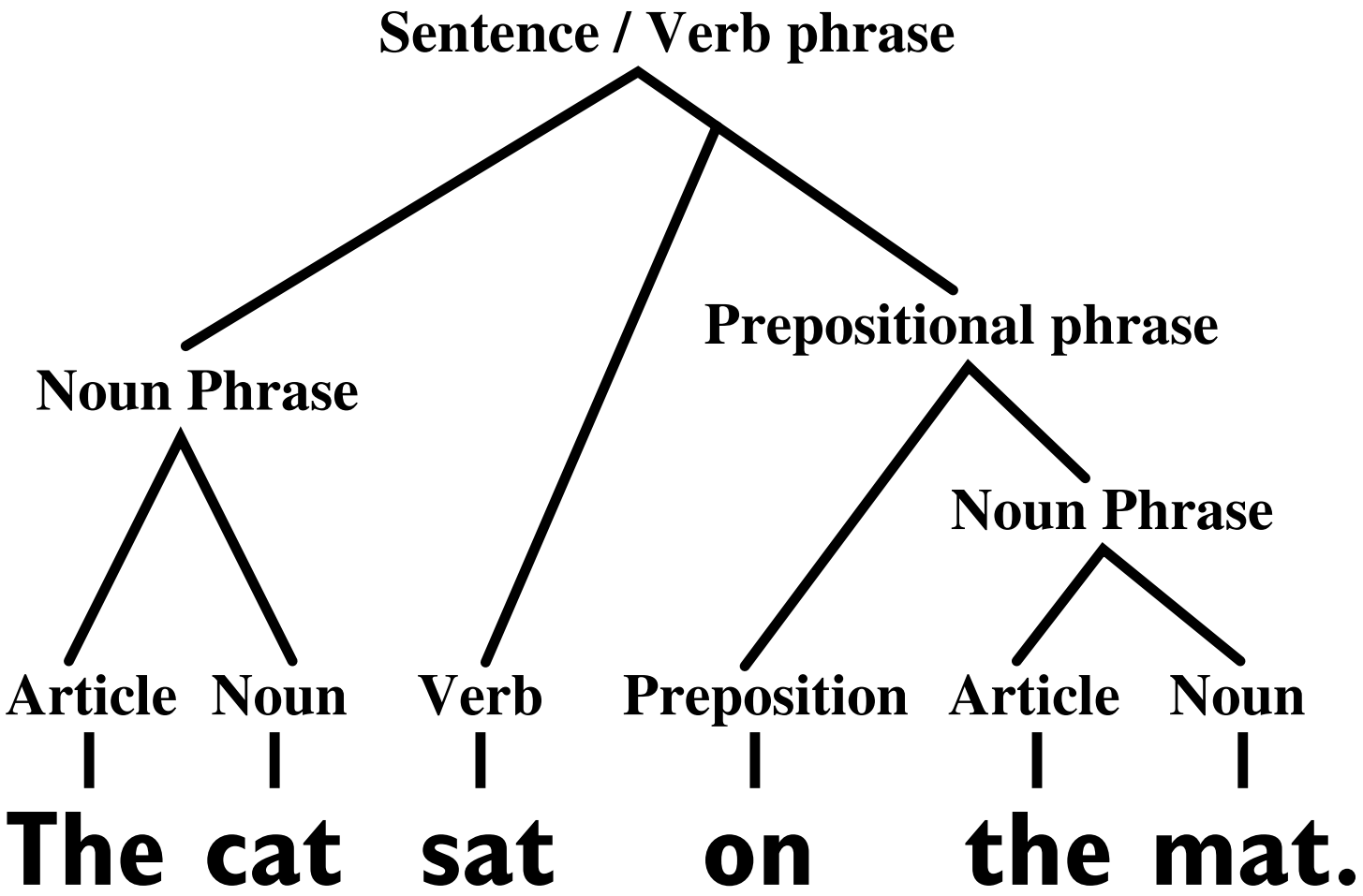
Természetes nyelvekben...

- a szavak betűk sorozatai
- a mondatok pedig szavakból és írásjelekből állnak össze

A nyelv szintaxisa leírja a mondatok megengedett szerkezeteit:

- Szavak lehetnek névelők, főnevek, igék stb.
- Ezekből különböző névszói és igei csoportokat lehet képezni
- A kifejezések megfelelő kombinációja pedig mondatot alkot

Basic constituent structure analysis of a sentence:



Szintaxis kontra szemantika

Ha lehagyunk egy névelőt (vagy egy oda nem illőt használunk), akkor a mondat helytelen, értelmetlen lesz. Ez a szintaxis következménye.

> “There is a a good shop on next corner.”

Másrészt viszont, ha lehagyunk pár vesszőt, a mondat értelmes maradhat, de megváltozhat a jelentése, a szemantikára hat.

> “My sister [,] who lives in London [,] visited me last week.”

> “A királynét megölni nem kell félnetek jó lesz ha mindenki egyetért én nem ellenzem.”

Végül pedig fontos észrevenni, hogy a jelentés nem mindig egyértelmű.

> “Time flies.”
> “Students hate annoying professors.”

Szintaxis informálisan

Definiáljunk egy egyszerű mesterséges nyelvet! Lexikális szinten vannak benne változók (kisbetűk), literálok (számok) és operátorok (*, =, ;). A lehetséges mondatokat a következőképp specifikáljuk:

- A mondatok utasítások, amelyeket pontosvessző (;) zár
- Az utasítások egy változónévből, egy egyenlőségjelből (=), és egy kifejezésből állnak
- A változónevek és a literálok helyes kifejezések
- Összetett kifejezést szorzással (*) alkothatunk

Szintaxis formálisan

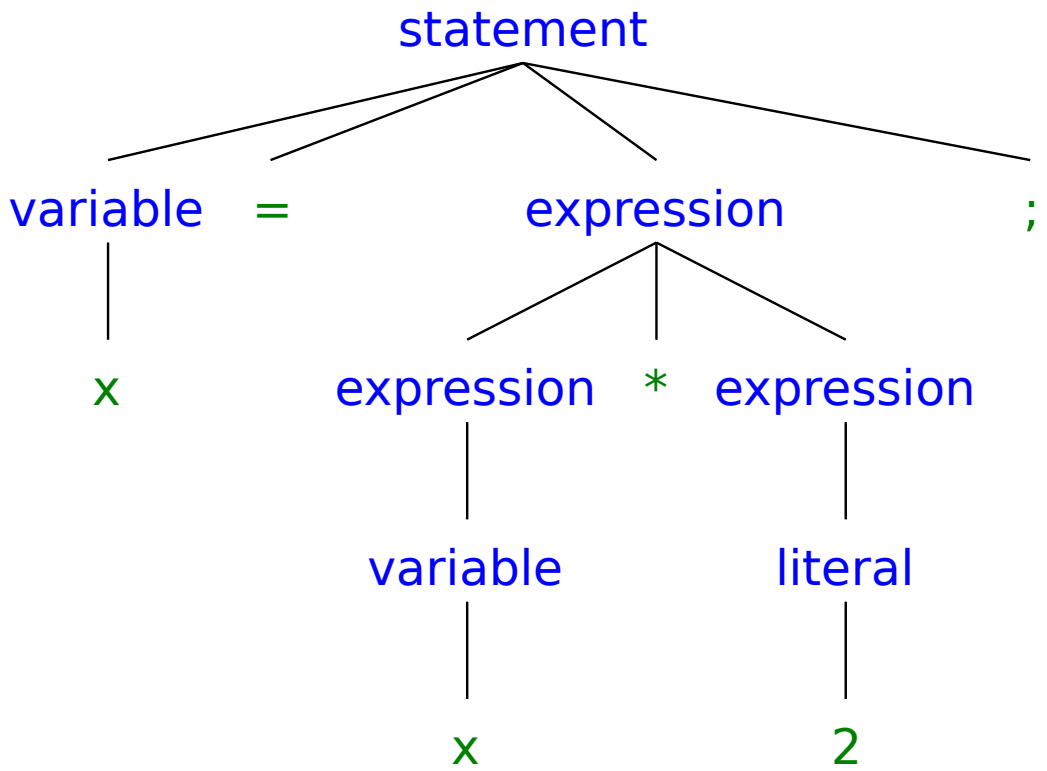
A (programozási) nyelvek szintaxisát általában formálisan definiálják (többnyire környezetfüggetlen grammatikákkal, BNF formában).

```
<statement>  ::= <variable> ’=’ <expression> ’;’
<expression> ::= <variable> | <literal>
               | <expression> ’∗’ <expression>
```

A szintaktikusan helyes mondatok nyelvtanilag helyesek, de ez nem garantálja, hogy “olvashatóak”, és hogy van értelmük, jelentésük.

```
<statement>  ::= <variable> '=' <expression> ';'
<expression> ::= <variable> | <literal>
               | <expression> '*' <expression>
```

Az “x = x * 2;” mondat helyes a fenti grammatika szerint.



Szintaxis formálisan

```
<statement>  ::= <variable> = <expression> ;
<expression> ::= <variable> | <literal>
               | <expression> * <expression>
```

Szerkezetileg helyes mondatoknak azonban nem feltétlenül van értelme, illetve többértelműek is lehetnek.

Az “`x = y * 2;`” egy szerkezetileg helyes mondat. Vajon mit jelent, ha y még nem kapott értéket? Az x változó nullázódik vagy hibát kellene jelezni? Elérkeztünk a szintaxis és a szemantika határára.

Érdekesség: különböző stílusú nyelvekben nagyon eltérően nézhetnek ki nagyon hasonló jelentésű mondatok.

```
while(x == 1)
{
  f();
}
```

Vagy pedig:

```
while x? = 1 do
  call f
done
```

Különbözőképp néznek ki, de vajon ugyanazt jelentik?

```
    if(x == 1) f();
    else       g();
```

vö.

```
    if(x == 1) {
        f();
    } else {
        g();
    }
```

vö.

```
    (x == 1) ? f() : g();
```

Informális szemantika

A programozási nyelvek megszületésével szinte csak informális leírások voltak a nyelvekről. A szemantika minden esetben angolul, természetes nyelven volt megadva.

> *Execution of the statement*
>
> `if <cond> then <stmts> fi`
>
> *checks the condition at first, and if it evaluates to* `true`*, then the statements are executed one after the other.*

- A természetes nyelvi leírások nem precízek, nem tekinthetők formális definíciónak (ellenben a matematikával és matematikai logikával), következésképp félreértésekhez vezettek.
- Másrészt tény, hogy formális szemantikadefiníciót készíteni egy nyelvhez hosszú és bonyolult feladat, továbbá a megértése is nehezebb egy olvasmányos leírásnál.

Informális kontra formális

Az informális leírásokat a fordítóprogramok és a programozók is többféleképp értelmezték...

- Ugyanahhoz a programozási nyelvhez készült fordítóprogramok különböző tárgykódokat eredményeztek egyazon program esetében, így a lefordított programok másképp működtek
- A programozók nem tudtak igazán jó és hatékony kódokat írni, mert ugyan nagyjából értették, mi mit csinál, de nem tudták, pontosan mi folyik a háttérben, milyen optimalizációkat hajt végre a rendszer

De ami nekünk, ELTE-seknek még fontosabb: ahhoz, hogy érvelhessünk programekvivalenciáról vagy programhelyességről, elengedhetetlen a formális szemantikadefiníció.

És még ma is vannak félreértések, félreértelmezések... íme egy híres-hírhedt példa C nyelven.

Mi az *f* függvény visszatérési értéke a következő C nyelvű programban?

```
int f(void) {
    int x = 0;
    return (x = 2) + (x = 1);
}
```

Informális kontra formális

És még ma is vannak félreértések, félreértelmezések... íme egy híres-hírhedt példa C nyelven.

Mi az *f* függvény visszatérési értéke a következő C nyelvű programban?

```
int f(void) {
    int x = 0;
    return (x = 2) + (x = 1);
}
```

- GCC (4.8.4): **2**
- CLANG (3.4): **3**

És még ma is vannak félreértések, félreértelmezések... íme egy híres-hírhedt példa C nyelven.

Mi az *f* függvény visszatérési értéke a következő C nyelvű programban?

```
int f(void) {
    int x = 0;
    return (x = 2) + (x = 1);
}
```

- GCC (4.8.4): **2**
- CLANG (3.4): **3**

... de legalább warningot dobnak egy ideje :)

Informális kontra formális

Mi értelme tehát a szemantikát formálisan tárgyalni?

- Alapvető dokumentációt ad a programozási nyelvnek
- Feltárja a félreértelmezhető elemeket a nyelvben, hogy végül minden fejlesztő, fordító és értelmező pontosan ugyanúgy értse a programot
- Precíz jelentésfogalmat alkot, amelynek fontos következményei:
  - Vizsgálhatjuk programok tulajdonságait
  - Vizsgálhatjuk programok ekvivalenciáját
  - Vizsgálhatjuk programok helyességét

Formális szemantika — igények

A nyelv különböző használóinak különböző formális definíciók, megfogalmazások lehetnek hasznosak. Kinek mi érdekes a nyelv szemantikájából, min van a hangsúly?

- Fordítóprogramtervezők:
  “Hogyan kell végrehajtani a programot?”
- Nyelvtervezők és helyességbizonyítók:
  “Mi a program lefutásának hatása?”
- Programozók:
  “Hogyan írjak programot, mik a főbb jellemzők?”

Formális szemantika — szintek

- Statikus szemantika
  - A szintaxis azon része, amelyet nem lehet környezetfüggetlen grammatikával megadni
  - Gyakran hívjuk környezetfüggő szintaxisnak
  - Ez is arra ad megkötéseket, hogy mi számít helyes mondatnak
  - Például: azok a változók, amelyek értékadás jobb oldalán állnak, értéket kellett, hogy kapjanak egy megelőző utasításban
- Dinamikus szemantika
  - Hogyan kell végrehajtani a programot
  - Mi lesz a végrehajtás eredménye
  - Például: az “x + y” kifejezés esetén kiértékeljük az “x”, majd az “y” kifejezést, végül összeadjuk a részeremdényeket

Formális szemantika — fajták, megközelítések

Formális szemantika: a programok jelentésének szigorú, precíz, egyértelmű, matematikai definíciója.

Sokféle megközelítés létezik. Az alapvetőek:

- Attribúrum grammatikák
- Operációs (műveleti) szemantika
- Denotációs (leíró) szemantika
- Axiomatikus szemantika

És ezeknek variációi...

- Action semantics
- Categorical semantics
- Game semantics
- Reduction semantics

Attribútum grammatikák

Általában átírási (translation) szemantikát definiálunk vele

- Mi most a statikus szemantika leírására használjuk
- Sok kutatás folyik körülötte
- Fordítóprogram generátorok szokásos bemenete
- Egyetlen grammatikából előállítható a szintaktikus és szemantikus elemző, továbbá a kódgenerátor (egy teljes fordítóprogram)

Ahogy látni fogjuk, kényelmesen megadható vele a környezetfüggetlen és környezetfüggő szintaxis is. Használják a következőkre:

- Statikus szemantikus elemzés
- Nyelvspecifikus programozási környezetek
- Teszteset-előállítás

(A denotációs szemantika speciális eseteként is kezelhető.)

Operációs (műveleti) szemantika

Fordítóprogramok és értelmezők készítőinek kiválóan alkalmas

- Definiálja, hogyan hajtódik végre a program lépésről lépésre
- Természetes dedukció és átmenetrendszerek kombinációja
- Lehet jobban vagy kevésbé részletes (strukturális, természetes)
- (“Small-step” vagy “big-step”)
- Címkézett rendszerek esetén trace szemantika

Denotációs (leíró) szemantika

Nyelvtervezés és helyességbizonyítás, továbbá végrehajtjató szemantika készítése során használatos

- A programokat és azok részeit matematikai objektumokra képezi
- Az operációs leírásnál jóval absztraktabb megfogalmazás
- Általában kompozicionálisan adjuk meg a nyelvi elemek jelentését
- Folytatásos (continuation) stílusban is használjuk

Axiomatikus szemantika

Főleg helyességbizonyításhoz, illetve programozóknak

- Szemantika = specifikáció = előfeltétel + utófeltétel
- Az állapotfogalomtól független megfogalmazási forma
- A denotációs és operációs szemantikánál is absztraktabb
- Állításokat fogalmaz meg a nyelvi elemekről

Összegzés

- Informális kontra formális
- Szintaxis kontra szemantika
- Statikus és dinamikus szemantika
- Operációs, denotációs, axiomatikus

Ajánlott irodalom

- Hanne Riis Nielson and Flemming Nielson. 1992. *Semantics with Applications: A Formal Introduction*. John Wiley & Sons, Inc.
- Kenneth Slonneger and Barry L. Kurtz. 1994. *Formal Syntax and Semantics of Programming Languages*. Addison Wesley Longman.
- Glynn Winskel. 1994. *The Formal Semantics of Programming Languages – An Introduction*, Foundations of Computing Series, MIT Press, Cambridge, MA.
- John C. Reynolds. 1998. *Theories of Programming Languages*. Cambridge University Press.

Attribútum grammatikák

Horpácsi Dániel
daniel-h@elte.hu

Attribútum grammatikák és alkalmazásaik a formális szemantika területén

Ismétlés: nyelv, szintaxis

- A programozási nyelvek olyan mesterséges nyelvek, amelyeket gépek, számítógépek vezérlésére terveztek
- A nyelv szimbólumai, szavai mondatokat (programokat) alkotnak, amelyek közül a szintaxis határozza meg, melyekhez lehet jelentést rendelni
- A szintaxist általában környezetfüggetlen grammatikával adják meg (BNF formában)

```
<expr>    ::= <literal> | <expr> '+' <expr>
<literal> ::= '0' | '1' | ... | '9'
```

Környezetfüggetlen szintaxis

A környezetfüggetlen szintaxis pontosan definiálja:

- Hogyan lehet a nyelv szimbólumaiból, szavaiból értelmes mondatokat alkotni
- A legtöbb esetben ez egy bővebb nyelvet ad meg, mint amit a fordító ténylegesen elfogad (szintaktikusan jól formált mondatoknak nem feltétlenül van egyértelmű jelentése)
- Milyen nyelvi szerkezeteket lehet használni a nyelvben, és hogyan kell ezeket pontosan jelölni, kombinálni

A grammatika és annak szerkezete általában jól mutatja, hogy

- Hogyan lehet egyszerű kifejezéseket, utasításokat konstruálni
- Hogyan lehet ezeket egymással kombinálni bonyolultabb szerkezetek, absztrakciók alkotásához

Környezetfüggetlen grammatikák

A $\mathcal{G} = (\mathcal{T}, \mathcal{N}, \mathcal{P}, \mathcal{S})$ formális grammatika környezetfüggetlen, ha

$$\forall p \in \mathcal{P}: \quad p \equiv A \to \alpha \qquad \text{ahol } A \in \mathcal{N} \text{ és } \alpha \in (\mathcal{T} \cup \mathcal{N})^*$$

- $\mathcal{T}$: terminális szimbólumok halmaza (a nyelv ábécéje)
- $\mathcal{N}$: nemterminális szimbólumok halmaza (szintaktikus kategóriák, résznyelvek)
- $\mathcal{P}$: levezetési szabályok
- $\mathcal{S} \in \mathcal{N}$: kezdőszimbólum (ebből vezetünk le teljes mondatokat)

A mondatok reprezentálhatóak a levezetési fájukkal (konkrét szintaxisfa, 'parse tree'). A szintaxisfa előállítására több ismert és hatékony algoritmus is létezik (pl. LL, LR, LALR, *GLR*).

Példa: környezetfüggetlen grammatika

Legyen $\mathcal{G} = (\mathcal{T}, \mathcal{N}, \mathcal{P}, \mathcal{S})$ környezetfüggetlen grammatika:

$$
\begin{aligned}
\mathcal{T} &= \{0, 1, \ldots, 9, +\} \\
\mathcal{N} &= \{\mathit{expr}, \mathit{literal}\} \\
\mathcal{S} &= \mathit{expr} \\
\mathcal{P} &= \{\mathit{expr} \quad \to \mathit{literal}, \\
&\qquad \mathit{expr} \quad \to \mathit{expr} + \mathit{expr}, \\
&\qquad \mathit{literal} \to 0 \\
&\qquad \mathit{literal} \to 1 \\
&\qquad \vdots \\
&\qquad \mathit{literal} \to 9\}
\end{aligned}
$$

Ugyanezt a grammatikát BNF-ben jóval olvashatóbban megadhatjuk:

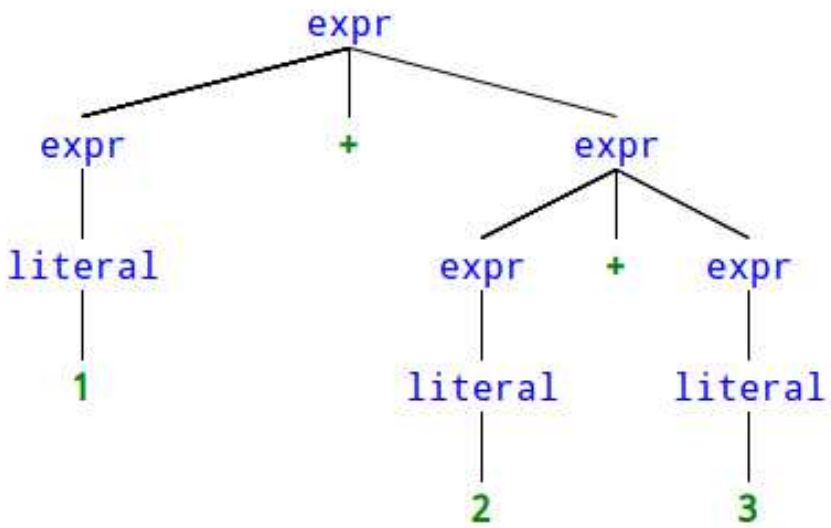
```
<expr>    ::= <literal> | <expr> '+' <expr>
<literal> ::= '0' | '1' | ... | '9'
```

Példa: mondat és levezetési fa

```
<expr>    ::= <literal> | <expr> ’+’ <expr>
<literal> ::= ’0’ | ’1’ | ... | ’9’
```

A grammatika által generált környezetfüggetlen nyelv egy mondata:

`1 + 2 + 3`



Egyértelmű a fenti grammatika? Egyszerűbben leírható EBNF-fel?

Statikus szemantika

Statikus szemantika

A szintaxisnak nem minden eleme fejezhető ki környezetfüggetlen leírással (néha igen, de rettenetesen bonyolulttá tenné a grammatikát)

- Környezetfüggő szintaxis
- “A szintaxis és a szemantika határa”
- Nem tisztán szerkezeti vagy formai jellemző, de nem is a dinamikus aspektust írja le
- Ez is a szintaxishoz tartozik olyan értelemben, hogy a “mi számít értelmes mondatnak” kérdésre ad választ
- A környezetüggetlen grammatikáknál kifejezőbb jelölésrendszer kell a leírásához

- Olyan tulajdonságok, amelyeket fordítási időben ellenőrizni lehet
- Tipikus elemei a típusellenőrzés és a névfeloldás

Statikus szemantika — felmerülő kérdések

```
1    int i = i;
2    int i = "foo";
3    int j = i;
4    for (int i = 0; i < 10; i++) {
5      int j = i;
6    }
```

- (1) Mi az értéke egy deklarálatlan/definiálatlan változónak?
- (2) Mi történik egy változó újradeklarálásakor?
- (2) Mi történik, ha egy `int` változónak szöveget adunk értékül?
- (4) A ciklus fejében lévő `i` változó elrejti/felüldefiniálja a külső változót?
- (5) A ciklus törzsében lévő `j` változó elrejti/felüldefiniálja a külső változót?

Szemantikus elemzés

Amikor feldolgozunk, értelmezünk egy mondatot,

- A szintaktikus elemzés hozza létre a szintaxisfát (vagy AST-t)
- Amelyen a szemantikus elemzés további ellenőrzéseket végez, s kiszűri azokat a mondatokat, amelyeknek biztosan nincs egyértelmű jelentése (pl. olyan programokat, amelyek biztosan futási hibát okoznának)

- A statikus elemzés környezetfüggő: a szintaxisfában egymástól távol eső csúcsokra tesz megkötéseket
- Ehhez szükség van információáramlásra a szintaxisfán belül

- Megoldás: a csúcsokra extra információt helyezünk el összetett címkézéssel (ezek lesznek az attribútumok), majd a levezetési szabályokat átalakítjuk, hogy ezeket az attribútumokat is kezeljék
- A szemantikus megkötéseket ezekre az extra információkra alapozva tudjuk megtenni

$AG = (\mathcal{G}, \mathcal{A}, \mathcal{R}, \mathcal{C})$ egy attribútum grammatika, ahol

- $\mathcal{G}$: környezetfüggetlen grammatika, a bázis
- $\mathcal{A}$: az attribútumok halmaza
- $\mathcal{R}$: az attribútumszámítási szabályok halmaza ($\mathcal{R}(p)$)
- $\mathcal{C}$: a feltételek halmaza ($\mathcal{C}(p)$)

Tehát kiterjesztjük a környezetfüggetlen grammatikát attribútumokkal, azok kiszámítási szabályaival, illetve az attribútumokra tett feltételekkel. A levezetett szintaxisfák “dekorálásra” kerülnek, a csúcsokat felcímkézzük az attribútumaikkal.

Attribútumok

- Minden terminális és nemterminális szimbólumhoz rendelhetünk attribútumokat
  - $\mathcal{A}(X)$ jelöli az X szimbólum attribútumait
  - $Attr(X)$ (vagy $X.Attr$) jelöli az X szimbólum $Attr$ nevű attribútumát

- A kiszámítási szabályokat a következő formában adjuk meg:

  $$Attr1(X) \leftarrow f(Attr2(Y), \ldots, AttrN(Z))$$

- Minden attribútumot legfeljebb egy szabály számíthat ki
- A számítás nem hivatkozhat olyan szimbólumok attribútumaira, amelyek nem szerepelnek a szabályban
- (Hasonló megkötések érvényesek a feltételekre)

Szintetizált attribútumok

Az $A.attr$ szintetizált,

- ha van olyan $p \equiv A \to \alpha$ levezetési szabály, hogy valamely $r \in \mathcal{R}(p)$ kiszámítja az $A.attr$ attribútumot
- Tehát $A.attr$ olyan szabályban kerül kiszámításra, ahol az A szimbólum a szabály bal oldalán áll

- Az információt felfelé közvetíti a szintaxisfában
- A részfákból számítja ki a fentebbi csúcsok tulajdonságait
- Pl.: névkötések, literálok értéke

- A terminális szimbólumok szintetizált attribútumai speciálisak: nincsenek részfák, az értékeket egy korábbi elemzési fázis (lexikális vagy szintaktikus) állítja be

Örökölt attribútumok

Az $X.attr$ örökölt,

- ha van olyan $p \equiv A \to \alpha X \beta$ levezetési szabály, ahol $r \in \mathcal{R}(p)$ kiszámítja az $X.attr$ attribútumot
- Azaz X a levezetési szabály jobb oldalán áll, amikor az $X.attr$ kiszámításra kerül

- A szintaxisfában lefelé és keresztben közvetíti az információt
- Pl.: deklarált változók, típusinformációk

Attribútumok nem lehetnek egyszerre szintetizáltak és örököltek is.

Speciális attribútumok

Vegyük észre, hogy bármely levezetési fa gyökerében a kezdőszimbólum áll. Következésképp a kezdőszimbólumnak nem lehet örökölt attribútuma (a gyakorlatban persze léteznek megoldások ennek kiküszöbölésére).

A terminális szimbólumok szintetizált attribútumai is speciálisak:

- A terminális szimbólumok a levezetési fa levelei
- Azaz nincsenek részfáik, amikből információt szintetizáljanak

- A gyakorlatban a leveleknek is vannak szintetizált attribútumaik, amelyekben a terminálisok szöveges reprezentációja tárolódik
- Kitüntetett szintetizált attribútum: nem használunk fel a kiszámításakor más attribútumokat
- Enélkül nem lehetne szemantikus elemzést és átírási szemantikát implementálni

Példa: az $a^n b^n c^n$ nyelv grammatikája

$$\langle \text{abcSeq} \rangle ::= \langle \text{aSeq} \rangle \ \langle \text{bSeq} \rangle \ \langle \text{cSeq} \rangle$$
$$\textit{InSize}(\langle \text{bSeq} \rangle) \leftarrow \textit{Size}(\langle \text{aSeq} \rangle)$$
$$\textit{InSize}(\langle \text{cSeq} \rangle) \leftarrow \textit{Size}(\langle \text{aSeq} \rangle)$$

$$\langle \text{aSeq} \rangle ::= \mathbf{a}$$
$$\textit{Size}(\langle \text{aSeq} \rangle) \leftarrow 1$$
$$\mid \quad \langle \text{aSeq} \rangle_2 \ \mathbf{a}$$
$$\textit{Size}(\langle \text{aSeq} \rangle) \leftarrow \textit{Size}(\langle \text{aSeq} \rangle_2) + 1$$

$$\langle \text{bSeq} \rangle ::= \mathbf{b}$$
$$\textit{Condition: InSize}(\langle \text{bSeq} \rangle) = 1$$
$$\mid \quad \langle \text{bSeq} \rangle_2 \ \mathbf{b}$$
$$\textit{InSize}(\langle \text{bSeq} \rangle_2) \leftarrow \textit{InSize}(\langle \text{bSeq} \rangle) - 1$$

$$\langle \text{cSeq} \rangle ::= \mathbf{c}$$
$$\textit{Condition: InSize}(\langle \text{cSeq} \rangle) = 1$$
$$\mid \quad \langle \text{cSeq} \rangle_2 \ \mathbf{c}$$
$$\textit{InSize}(\langle \text{cSeq} \rangle_2) \leftarrow \textit{InSize}(\langle \text{cSeq} \rangle) - 1$$

Az attribútumok kiértékelése

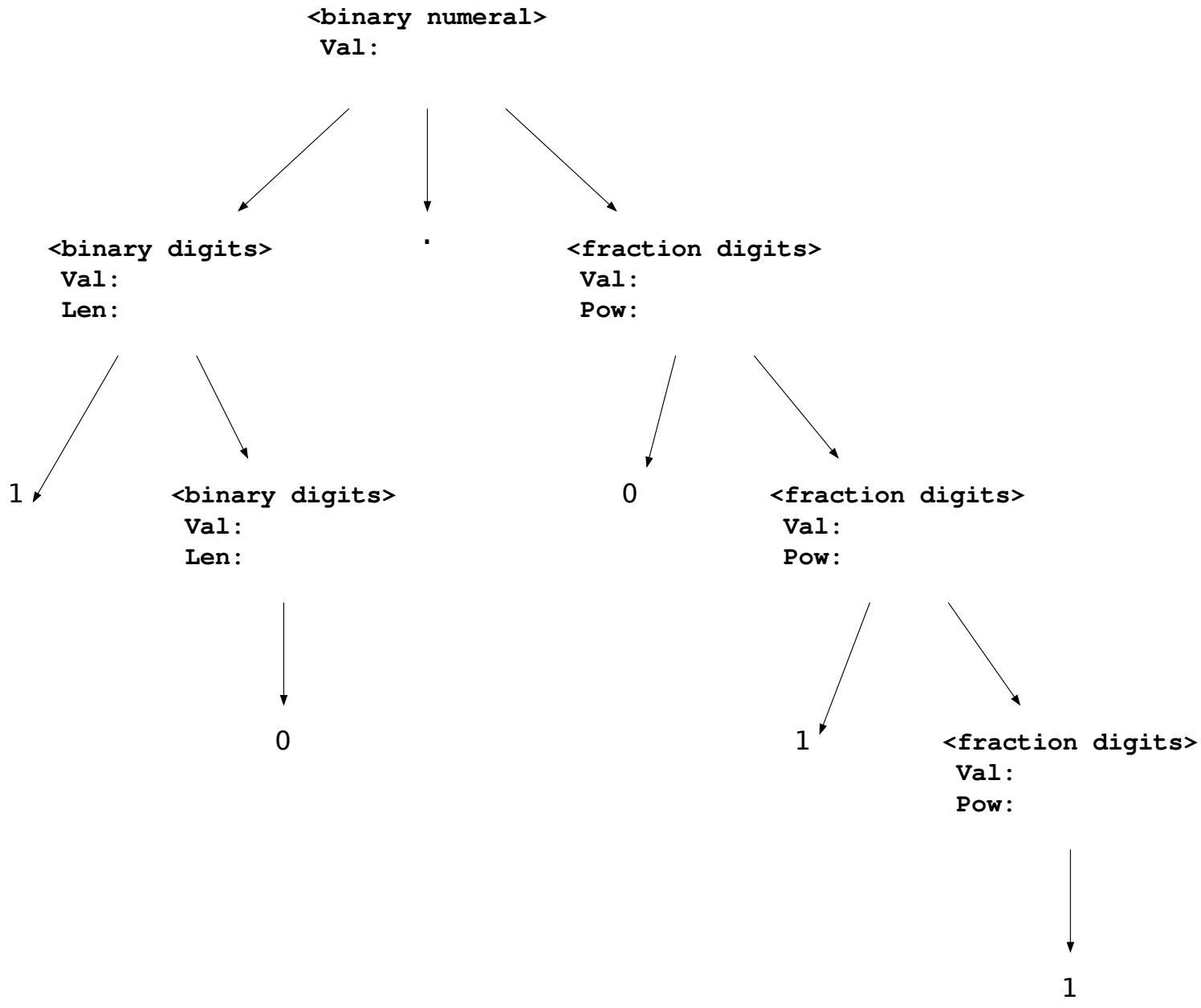
- (Úgy is mondják, hogy a szintaxisfa dekorálása.)

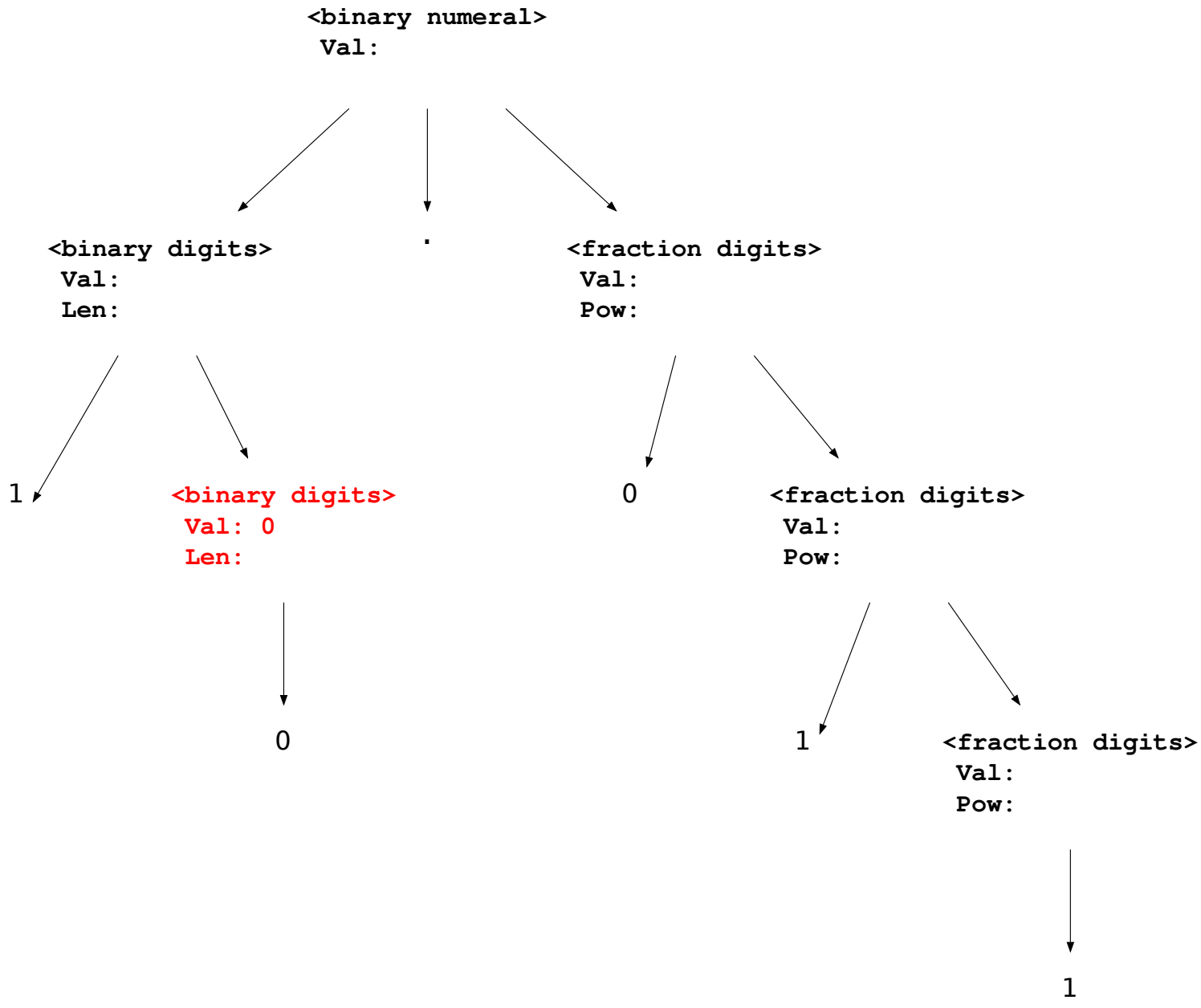
- Az attribútumok, a kiszámítási szabályaik és a feltételek együttesen alkalmasak arra, hogy a nyelv környezetfüggő tulajdonságait leírják
- Amikor egy feltétel kiértékelése hamis értéket eredményez, az szemantikus hibát jelez

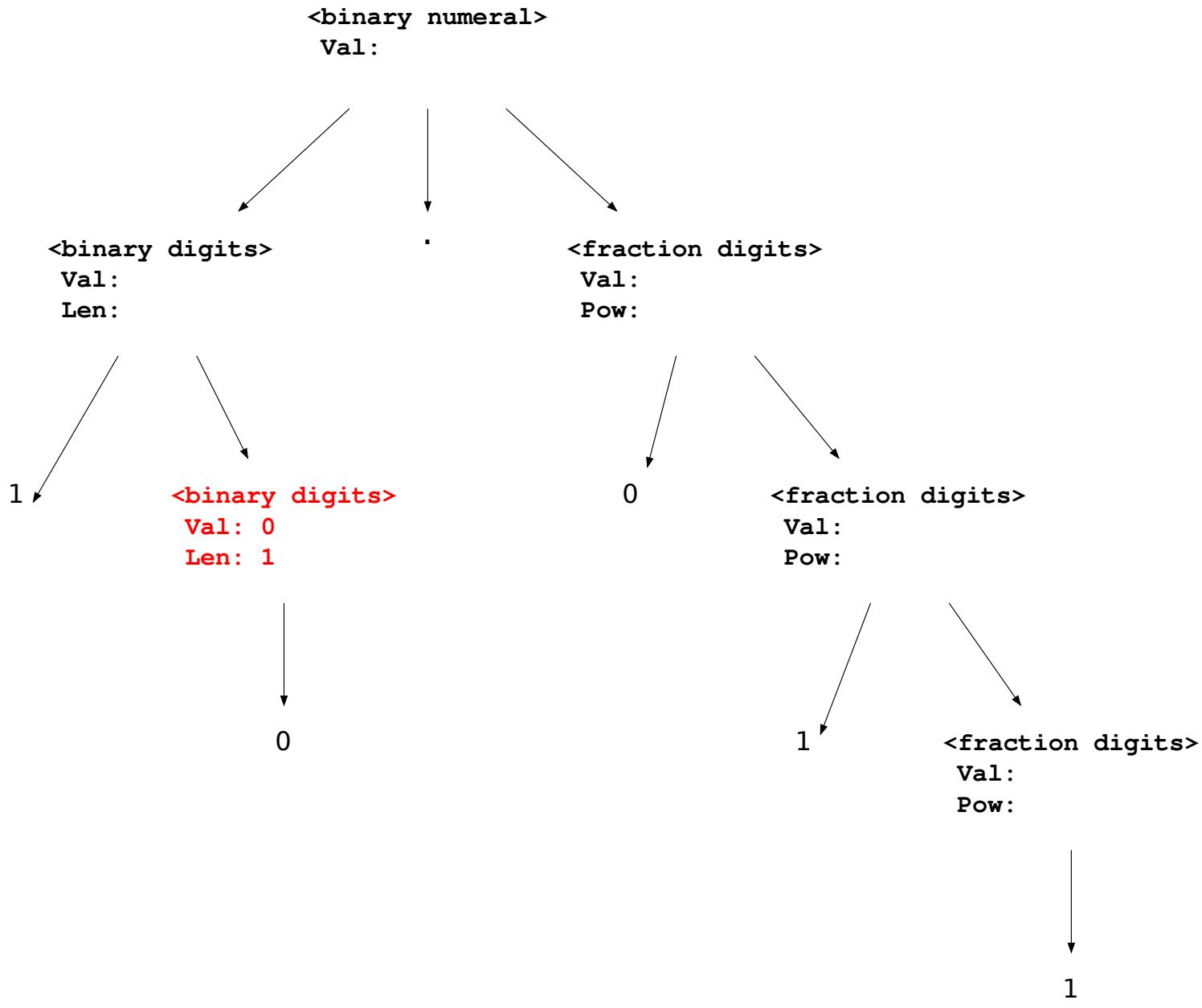
- Azonban a kiértékelés nem mindig egyszerű; az attribútumok között függőségek állnak fent, amelyeket figyelembe kell venni
- Annak eldöntése, hogy egy AG szintaxisfája kiértékelhető-e, NP-teljes probléma
- A megfelelő kiszámítási sorrend megtalálására vannak heurisztikák

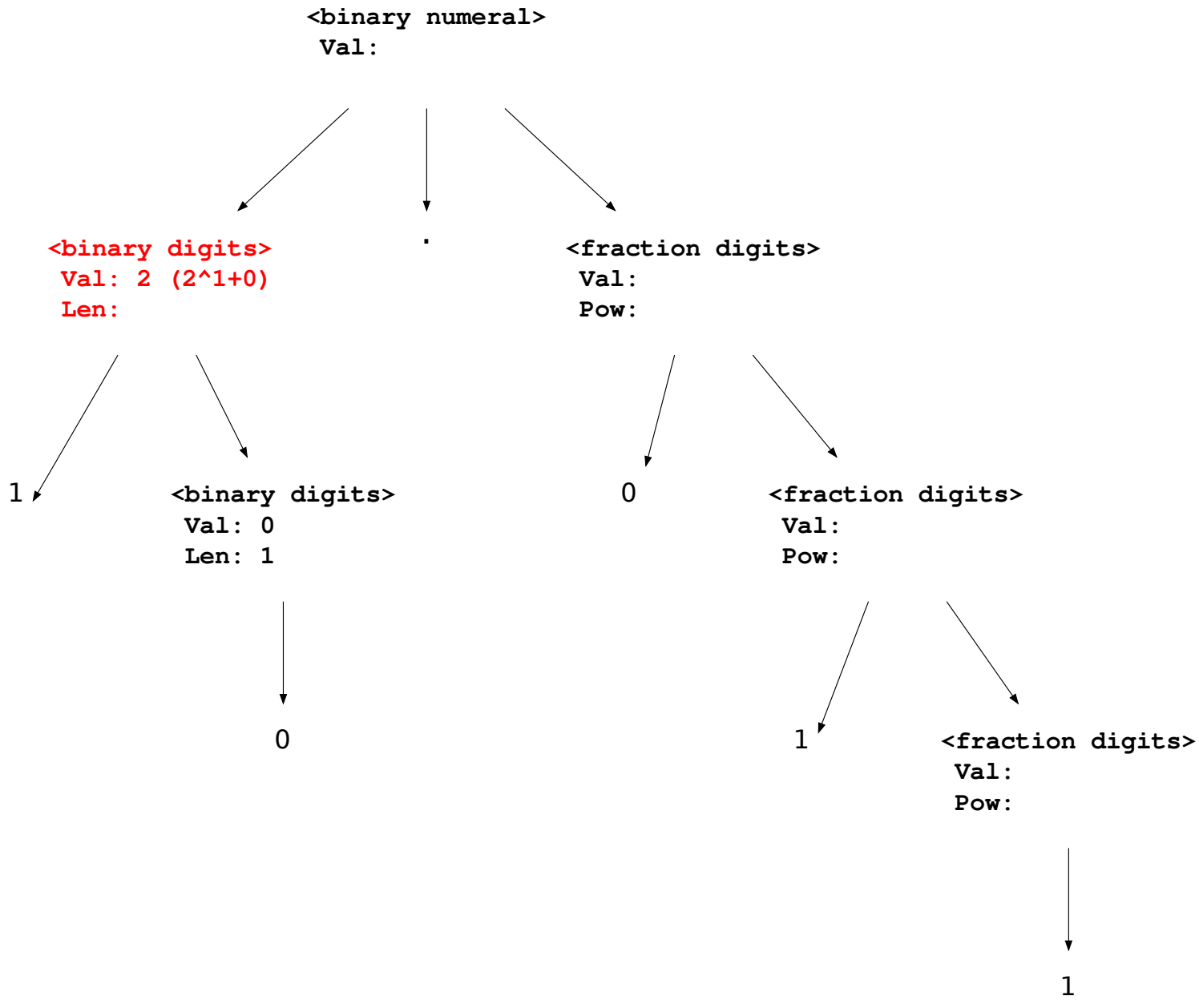
Példa — levezetési fa dekorálása

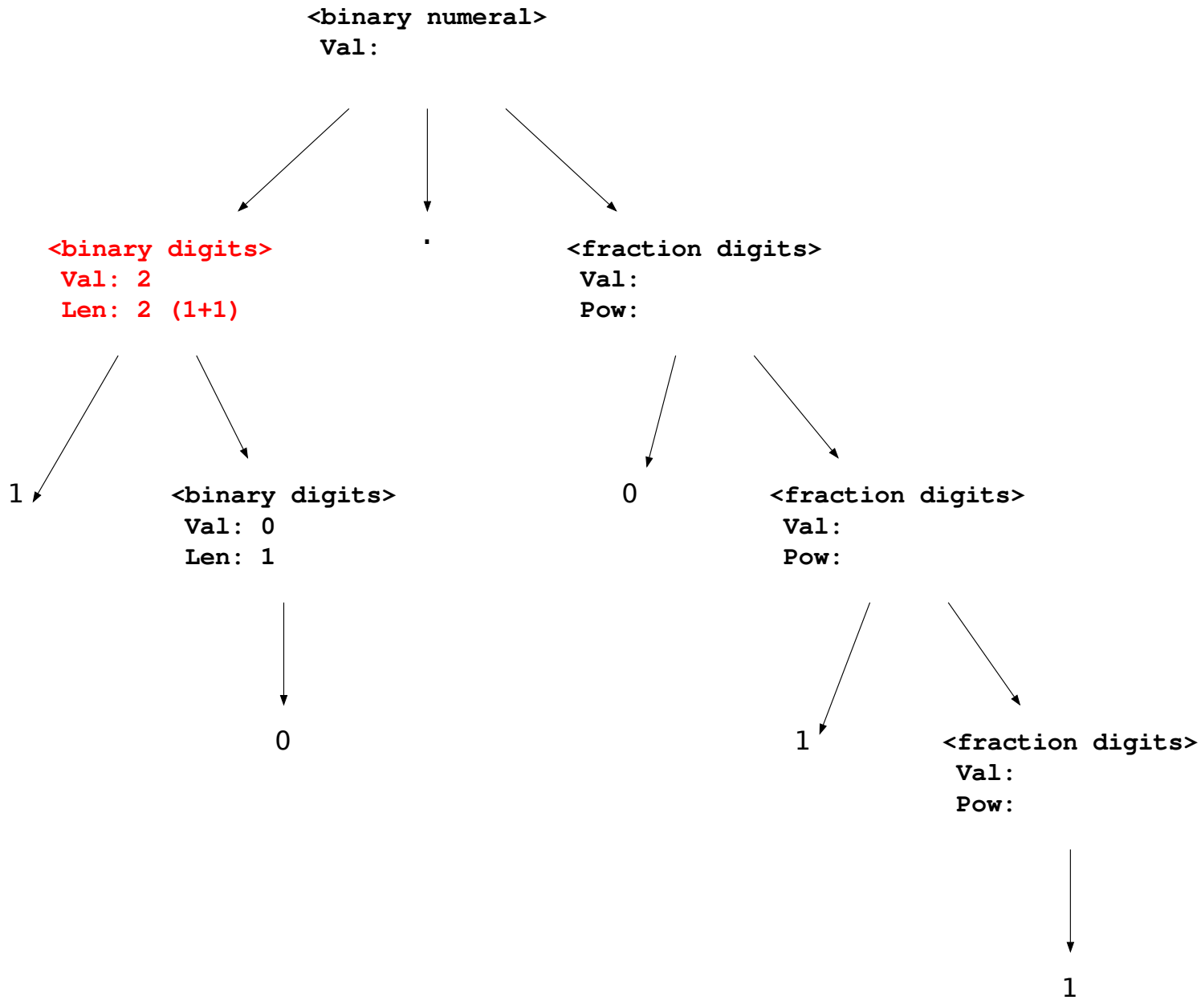
- Tekintsük a *exercises_02.pdf* dokumentumban felírt attribútum grammatikát
- A következő ábrákon látható a “10.011” mondathoz tartozó szintaxisfa attribútumainak egy lehetséges kiértékelése

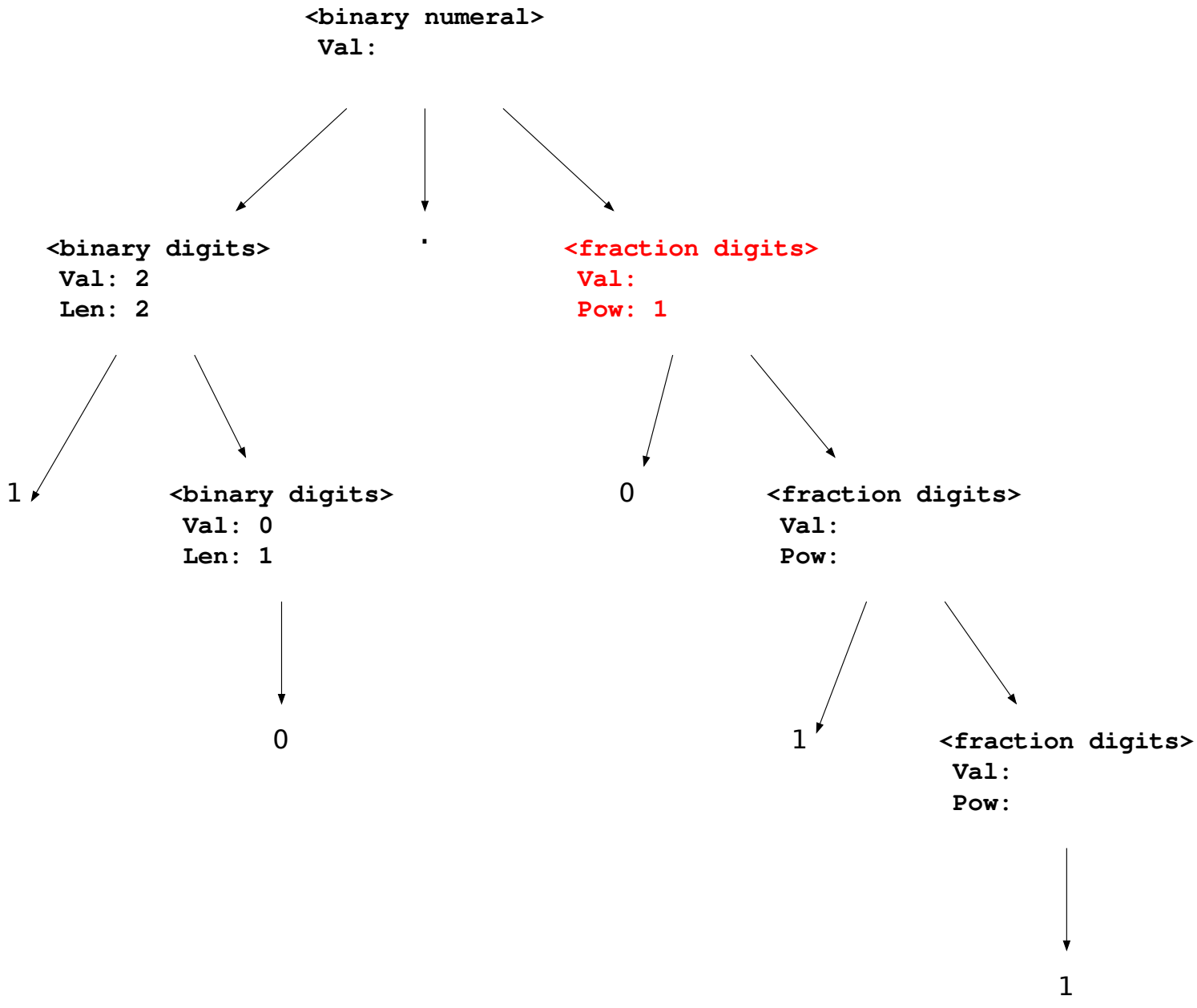

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val:
Len:
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

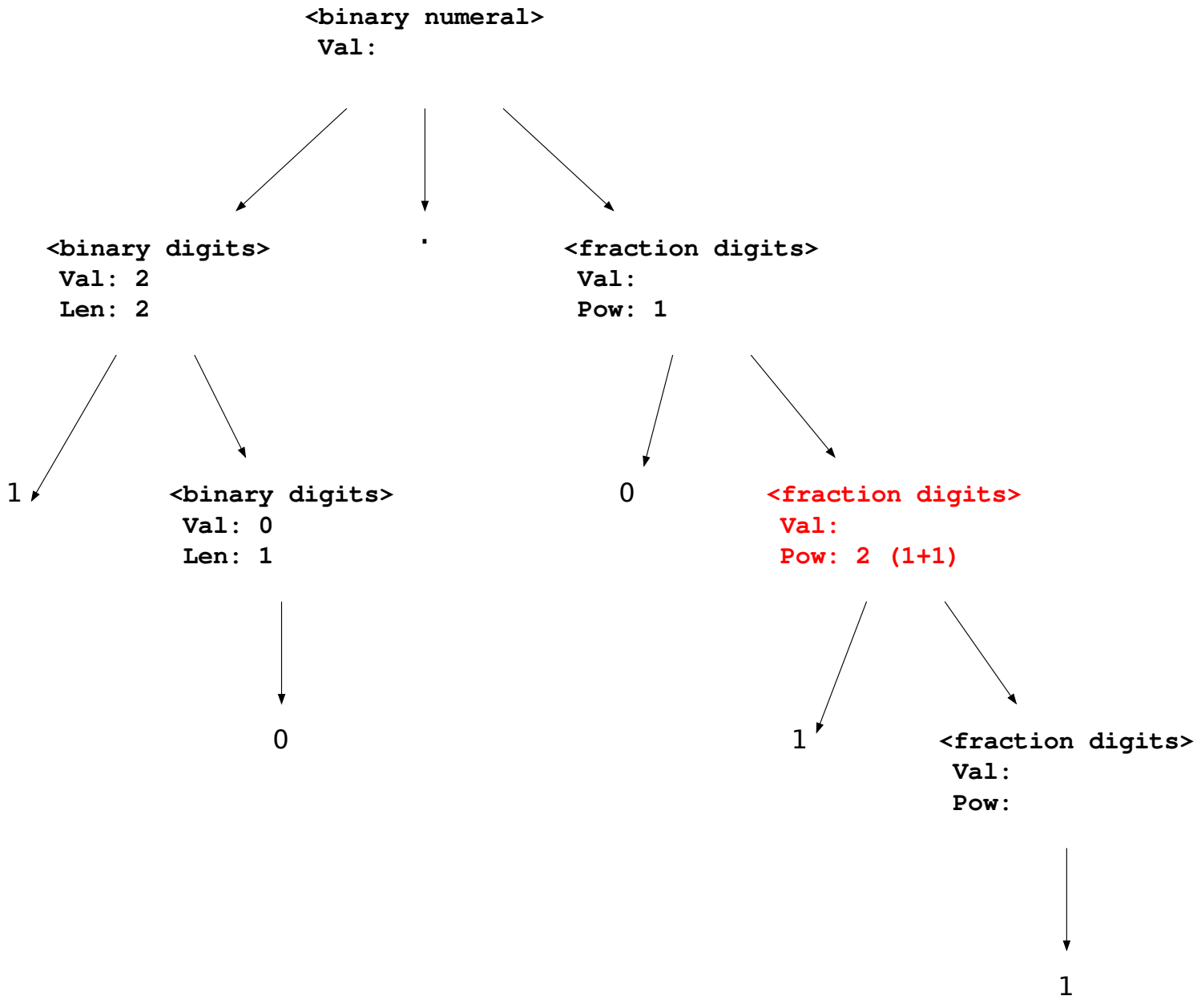

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len:
0
0
<fraction digits>
Val:
Pow:
1
<fraction digits>
Val:
Pow:
1

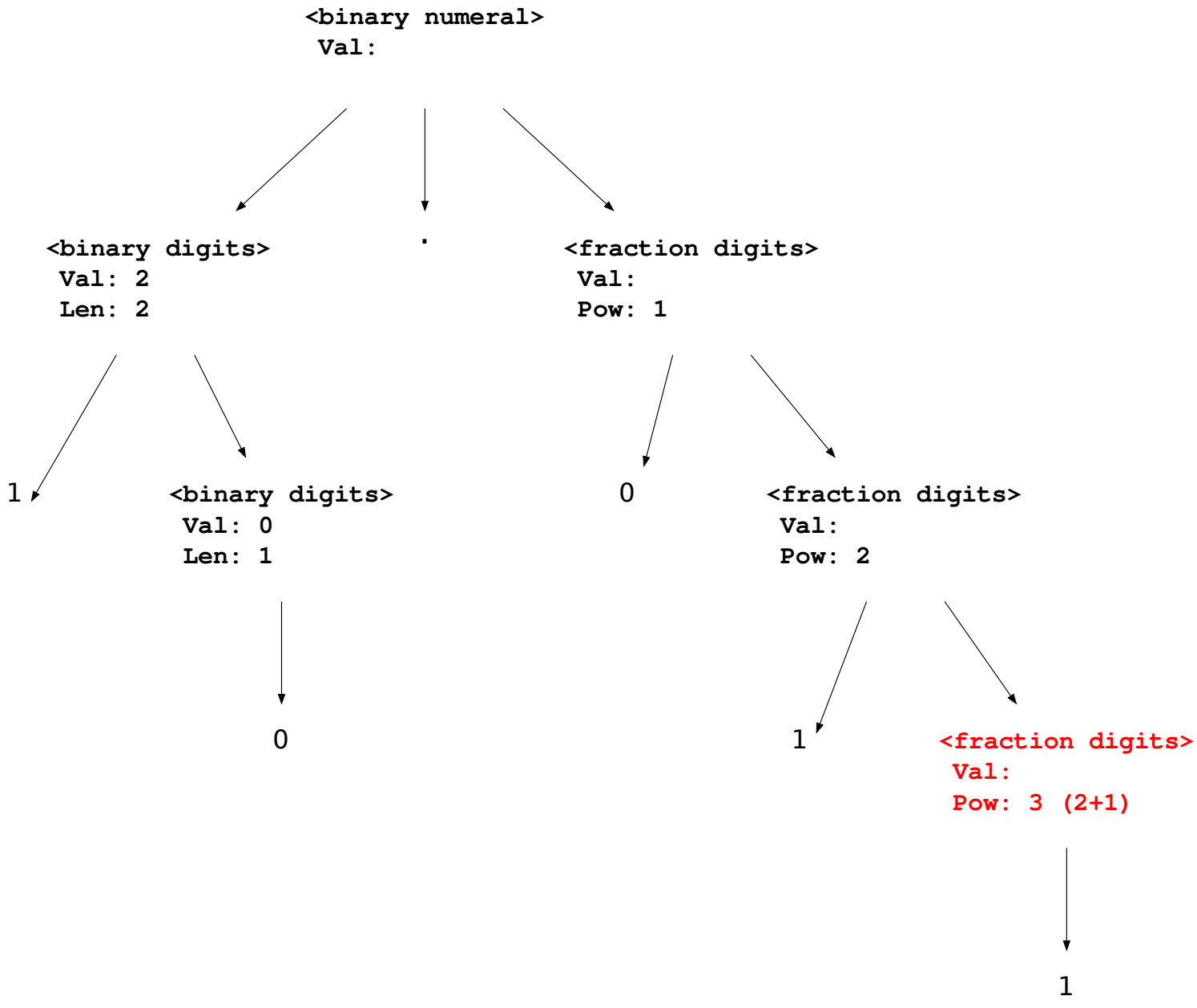

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

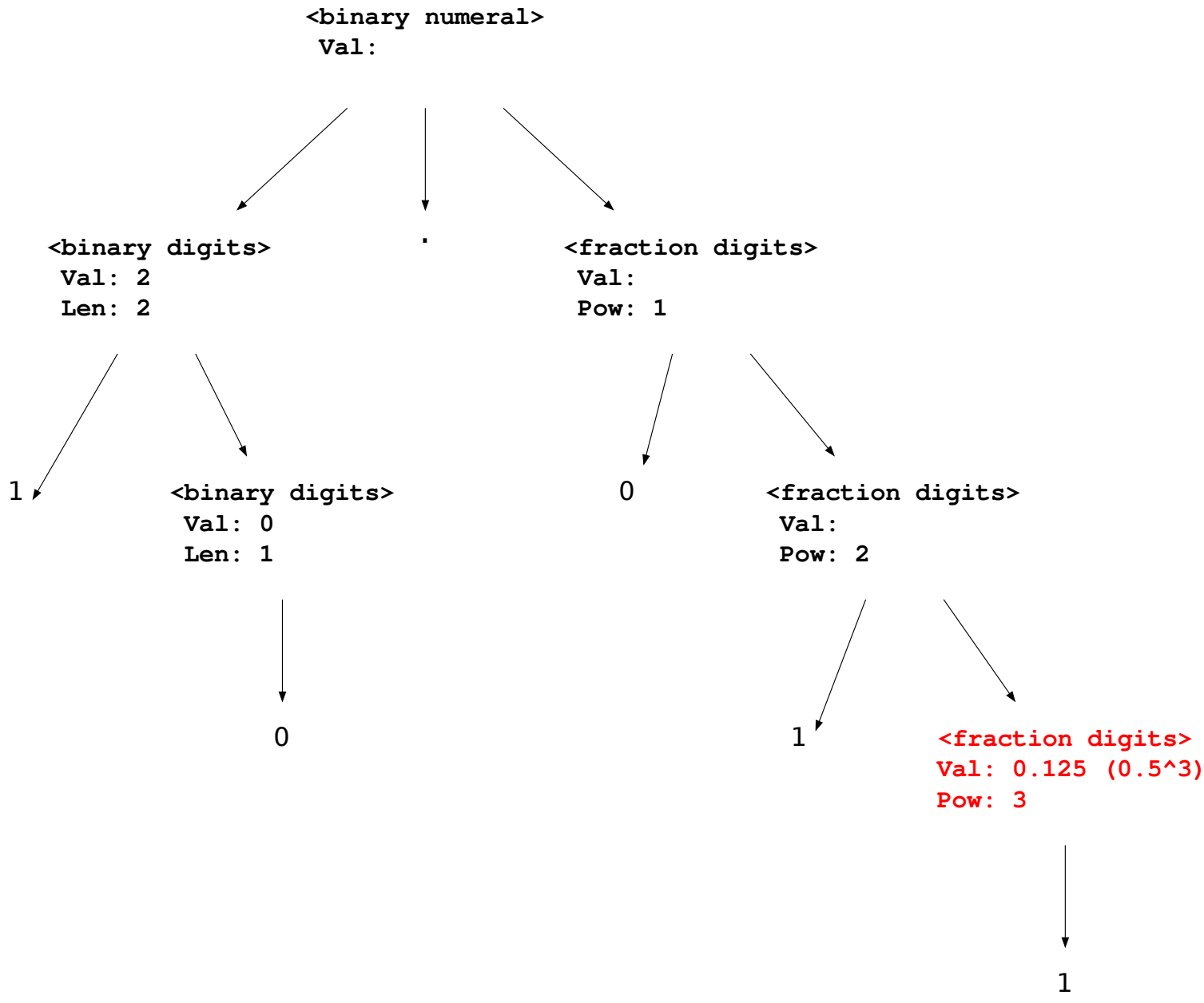

<binary numeral>
Val:
<binary digits>
Val: 2 (2^1+0)
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

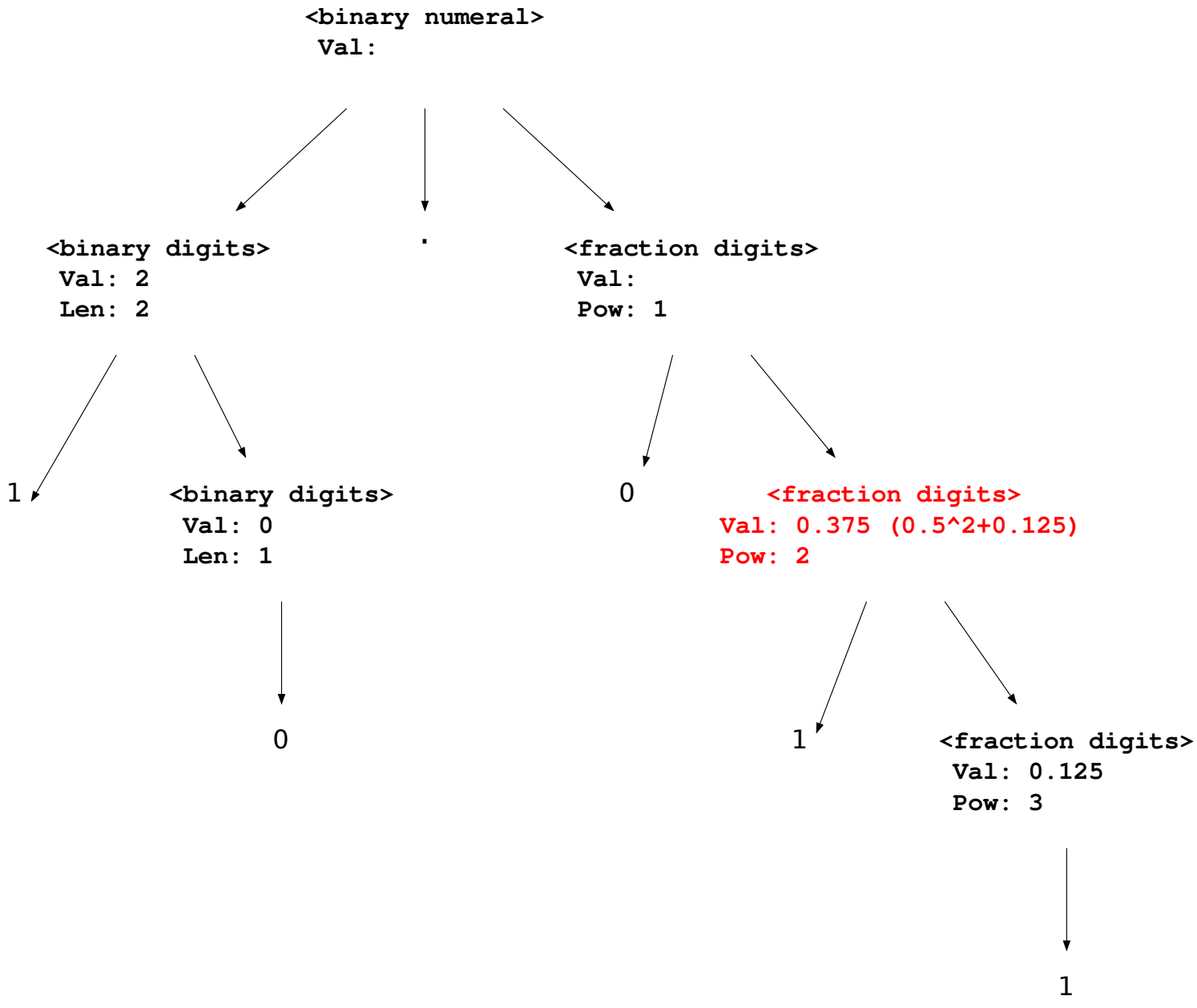

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2 (1+1)
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

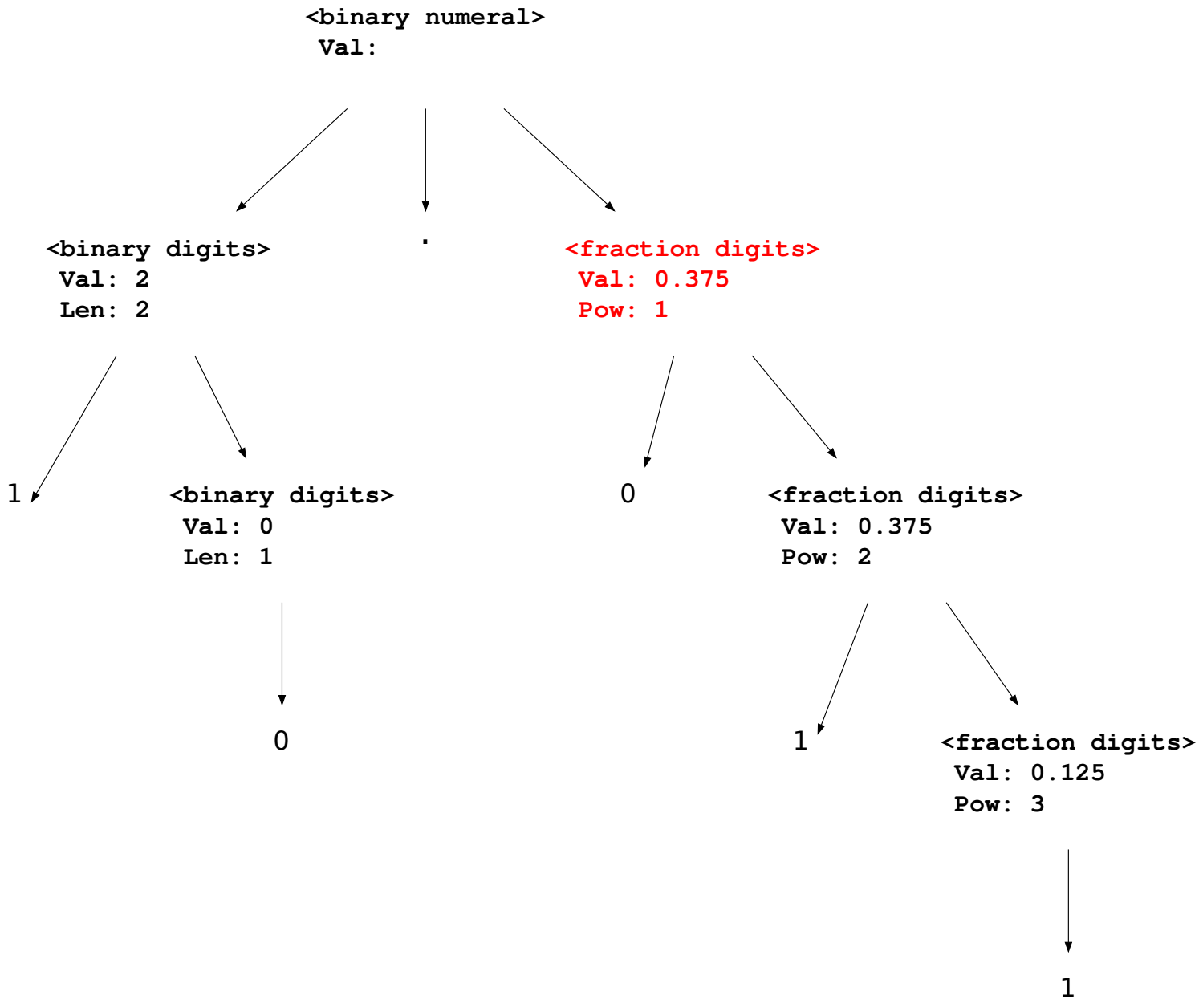

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

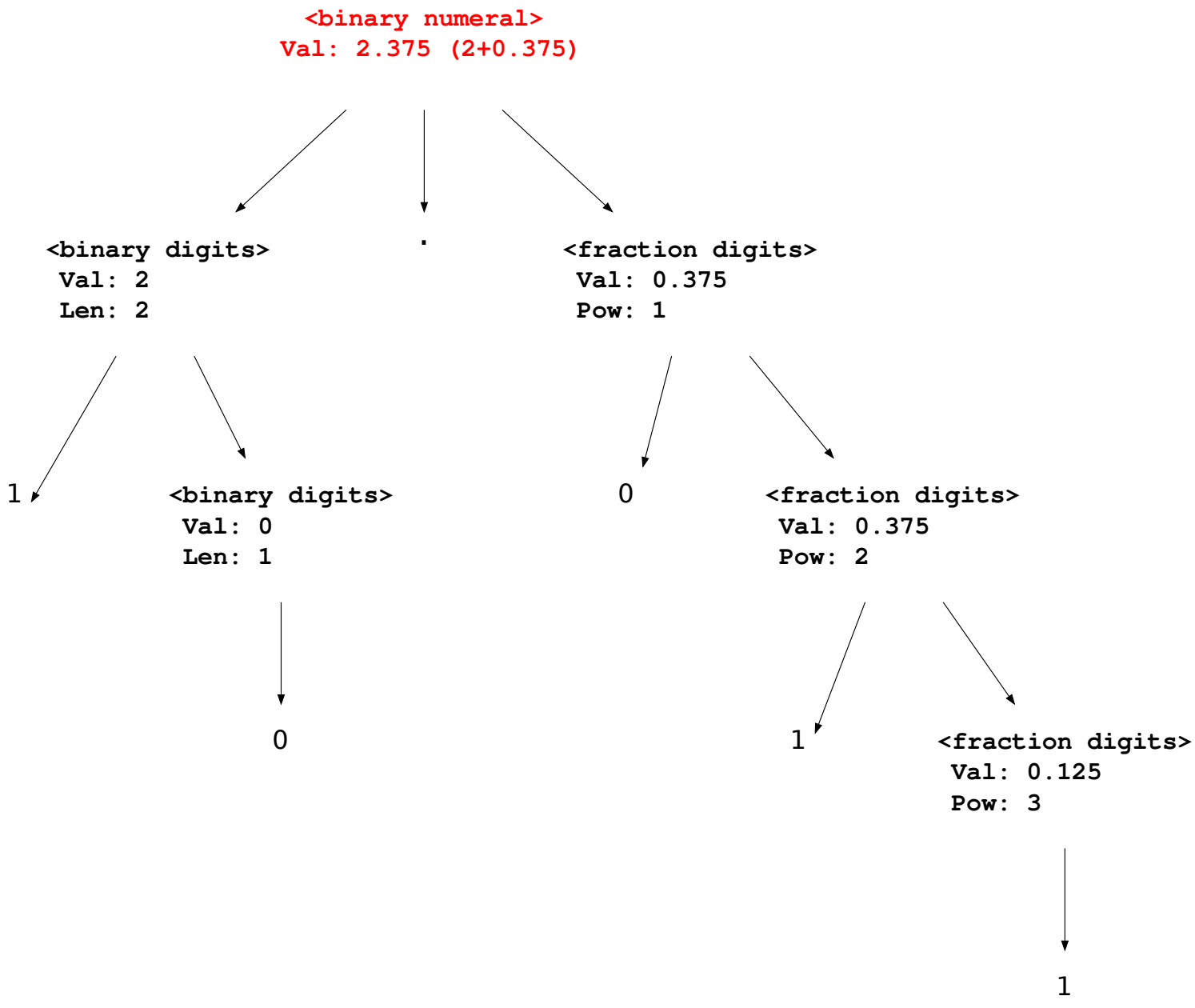

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow: 2 (1+1)
1
<fraction digits>
Val:
Pow:
1

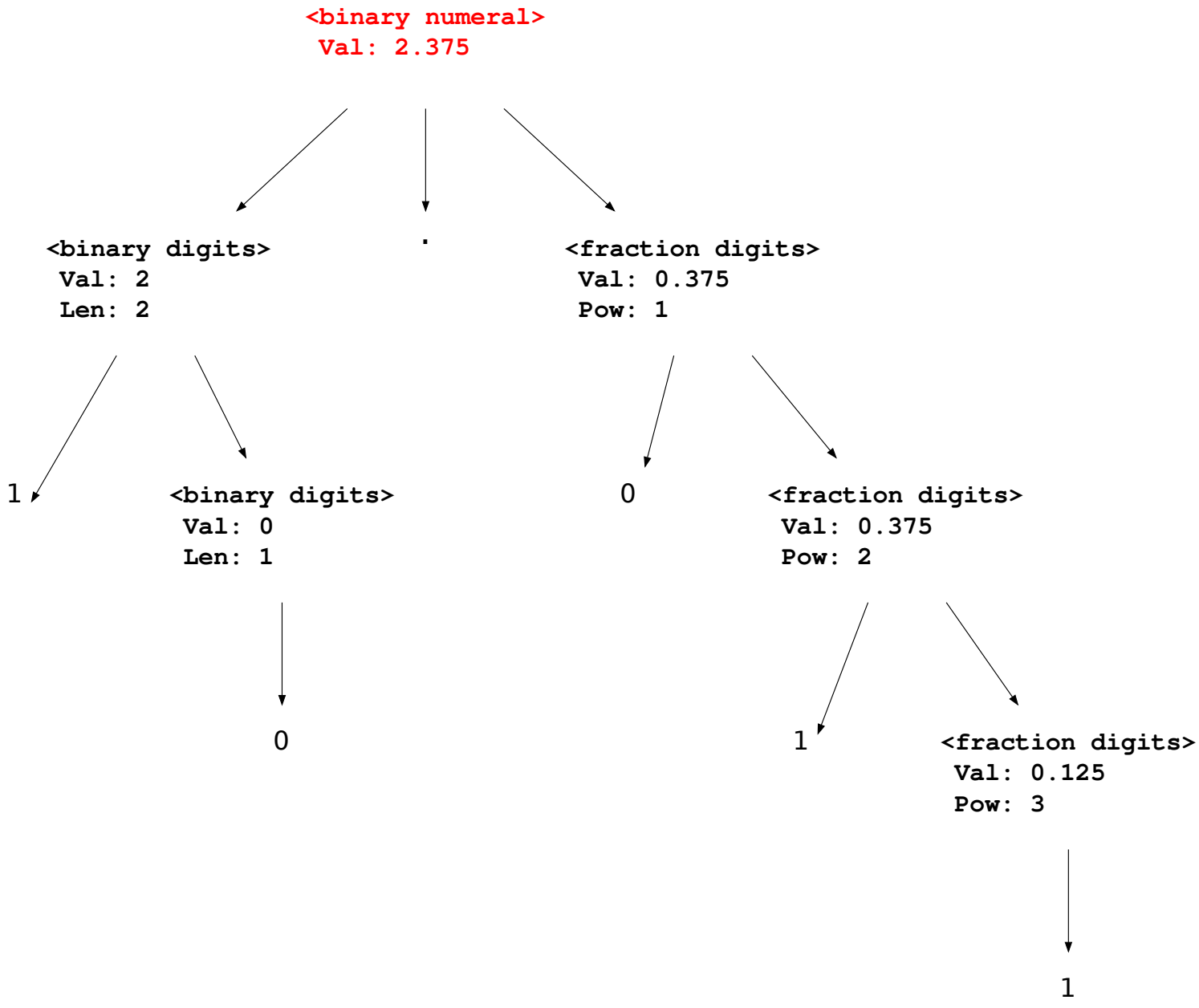

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow: 2
0
1
<fraction digits>
Val:
Pow: 3 (2+1)
1


<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow: 2
0
1
<fraction digits>
Val: 0.125 (0.5^3)
Pow: 3
1


<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val: 0.375 (0.5^2+0.125)
Pow: 2
1
<fraction digits>
Val: 0.125
Pow: 3
1


<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val: 0.375
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val: 0.375
Pow: 2
1
<fraction digits>
Val: 0.125
Pow: 3
1


<binary numeral>
Val: 2.375 (2+0.375)
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val: 0.375
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val: 0.375
Pow: 2
1
<fraction digits>
Val: 0.125
Pow: 3
1


<binary numeral>
Val: 2.375
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val: 0.375
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val: 0.375
Pow: 2
1
<fraction digits>
Val: 0.125
Pow: 3
1

Jól definiált AG

Egy jól definiált attribútum grammatikában (WAG):

- Minden lehetséges mondat minden szintaxisfájának attribútumai egyértelműen kiszámíthatóak
- Az ilyen grammatikákban a direkt attribútumfüggőségek gráfja körmentes

- Azaz biztosan létezik egy sorrend, amelyben ki tudjuk értékelni az összes attribútumot és a feltételek ellenőrizhetőek
- Általánosan egy egyszerű nemdeterminisztikus algoritmussal elvégezhető (naiv megoldás)

A WAG speciális osztályai

Speciális megkötésekkel elérhető, hogy a kiszámítási sorrend könnyen meghatározható legyen (particionált, rendezett AG).

- S-Attribútum grammatikák
  - Kizárólag szintetizált attribútumokat tartalmaznak
  - Parser-generátorok gyakran alkalmazzák
  - Az alulról-felfelé szintaktikus elemzéshez illeszkedik
- L-Attribútum grammatikák
  - Szintetizált és örökölt attribútumokat is tartalmaz, de megkötések vannak a lehetséges függőségekre
  - Az attribútumok biztosan kiszámíthatóak egyetlen bejárással
  - A felülről-lefelé szintaktikus elemzésnél használják

Parser generators (Compiler compilers)

A szintaktikus elemző generátorok olyan eszközök, amelyekkel elemzőket tudunk készíteni a nyelvhez a környezetfüggetlen grammatikája alapján.

- Tulajdonképp egy teljes lexikális és szintaktikus elemzést kapunk programozás nélkül

- Továbbá gyakran lehetővé teszik attribútumok használatát is
- Ezzel leírhatóvá válik a szemantikus elemzés és a kódgenerálás is
- Tehát fordítóprogramot készít automatikusan

- Különböző nyelveken különböző implementációk
- yacc, yecc, bisonc++, antlr, happy, ...

Compiler compilers — kódgenerálás

Az attribútum grammatikával lényegében fordítási szemantikát definiálunk.

- Mivel legtöbbször alulról-felfelé elemzőt implementálnak, S-AG-vel adják meg a szemantikát
- A kódgeneráláshoz felvesznek egy “kód” attribútumot, amely a célnyelvi szimbólumokat tartalmazza
- A kód alulról felfelé folyik a fában és végül összeépül a lefordított program

- A kezdőszimbólum kód attribútuma tartalmazza a lefordított programot

További alkalmazási területek

- Nyelvkiterjesztés
- Nyelvspecifikus keretrendszerek
- DSL implementációk
- Protokollok leírása
- Típusok leírása, adatgenerálás

Összegzés

- Környezetfüggetlen grammatikák
- Környezetfüggetlen kontra környezetfüggő
- Környezetfüggő tulajdonságok
- Attribútum grammatikák, alosztályaik, kiértékelés
- Alkalmazási területek

C++

Az $a^n b^n c^n$ környezetfüggő nyelv végrehajtható szemantikája elérhető a kurzus anyagai között. A statikus és dinamikus szemantikát is bemutató S-attribútum grammatika implementációt *flex* és *bisonc++* segítségével definiáltuk.

Kifejezések szemantikája

Horpácsi Dániel
daniel-h@elte.hu

Az előadás témája

Egyszerű kifejezések formális szemantikája

— Az első lépés a programozási nyelvek szemantikájának megadása felé

Konkrét szintaxis kontra absztrakt szintaxis

- A **konkrét szintaxis** definiálja, hogyan lehet a tárgyalt programozási nyelven jól formázott mondatokat leírni
- Megadja a nyelvi elemek precíz, konkrét leírását; milyen más elemekből, hogyan konstruálhatunk összetett szerkezeteket
- Szintaktikus elemzőt tudunk ezen definíció alapján készíteni a nyelvhez, automatikusan

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

- Az **absztrakt szintaxis** csak azt írja le, milyen alapvető szintaktikus kategóriák vannak a nyelvben és azok milyen minták mentén épülnek fel
- Megadja, hogyan épülnek majd fel az absztrakt szintaktikus termek, szintaxisfák

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

Konkrét szintaxis kontra absztrakt szintaxis

Az absztrakt szitaxisból kimaradnak például

- Konkrét kulcsszavak
- Precedenciák, asszociativitási szabályok
- Zárójelek
- A szintaktikus elemzést segítő elválasztó és termináló szimbólumok
- Az olvasást (megértést) segítő szimbólumok, jelek

Vajon hogyan reprezentálhatjuk az absztrakt szintaxist?
Visszaalakítható az absztrakt konkréttá?

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- **Operációs (műveleti)** — deduktív levezetésekkel

  Megadja, hogyan kell végrehajtani a programot

- **Denotációs (leíró)** — matematikai objektumokkal

  A jelentést denotációk hozzárendelésével adja meg

Kifejezések mint egyszerűbb nyelvi elemek

A formális szemantikadefiníciók alapvető fogalmainak tisztázására először olyan egyszerű aritmetikai és logikai kifejezések jelentését adjuk meg, amelyek a legtöbb programozási nyelvben megtalálhatóak.

- Operációs és denotációs szematikát is definiálunk, hogy felfedjük a két megközelítés közötti alapvető különbségeket
- A formális szemantika definiálásakor feltesszük, hogy a kifejezések szintaktikusan és statikus szemantikusan is helyesek, tehát biztosan van jelentésük

Kifejezések: absztrakt szintaxis

A szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)	(token)
x	$\in$	Var	(változó szimbólumok)	(token)
a	$\in$	Aexp	(aritmetikai kifejezések)	
b	$\in$	Bexp	(logikai kifejezések)	

Absztrakt produkciós szabályok

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a$$

$$b ::= \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2$$

- Feltesszük, hogy n és x már definiáltak
- Vegyük észre a bázis- és rekurzív esetek közötti különbséget
- “$a_1 + a_2$” termként pl. $\mathit{addition}(a_1, a_2)$ formában lenne írható

Állapotok

- A kifejezésekben szerepelhetnek változószimbólumok, tehát függhetnek a környezetüktől, az állapottól, amelyben kiértékeljük őket
- Az egyszerűsítés végett csak egész változókat engedünk meg, de a módszer általánosítható

- A lehetséges állapotokat függvényekkel reprezentáljuk

$$s \in \mathrm{State} = \mathrm{Var} \to \mathbb{Z}$$

- Minden ilyen függvény egy lehetséges állapotot ad meg: a változókhoz rendelt értékeket
- Jelölni listaként szoktuk: $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$

Állapotok: lekérdezés és módosítás

Vegyük a következő állapotot: $s \in \mathrm{State} = \mathrm{Var} \to \mathbb{Z}$

- Ha hivatkozunk egy változó értékére, ki kell olvasnunk a "memóriából"

  $s[x]$ megadja az x változó értékét az s állapotban

- Amikor értéket adunk egy változónak, frissíteni kell az állapotot

  $s[y \mapsto v]$ megad egy olyan állapotot, amely megegyezik az s állapottal, de y helyen az értéke v

Következésképp teljesül, hogy $$(s[y \mapsto v])[x] = \begin{cases} v & \text{if } x = y \\ s[x] & \text{if } x \neq y \end{cases}$$

Operációs szemantika

- “Hogyan értékelnénk ki egy ideális, absztrakt gépen?”
- Valójában átmeneteket definiál megfelelő konfigurációk között
- Az átmeneteket általános következtetési szabályokkal adjuk meg
- Ezek tulajdonképp sémák, amelyeket tetszőleges szintaktikus elemmel példányosíthatunk
- A levezetések szintaxis-vezéreltek
- A dedukcióval definiált átmenetrendszert használjuk a kifejezések eredményének (vagy normálformájának) meghatározására

A következő pár dián strukturális operációs szematikát definiálunk kifejezéseknek, amellyel megadjuk a részletes kiértékelési lépéseket.

Konfigurációk, átmenetek

- Ahogy már említettük, a kiértékelés függ a környezettől — az állapotot egy függvénnyel adjuk meg
- A konfigurációknak két megengedett formája lesz: egy kifejezés és egy állapot párja, vagy pedig a kifejezés eredménye (szemantikája)
- A konfigurációk közötti átmeneteket a következőképp jelöljük:

  Egy köztes lépés a számításban:

  $$\langle a, s \rangle \Rightarrow \langle a', s \rangle$$

  Egy (rész)kifejezés kiértékelése végén a végkonfiguráció az érték:

  $$\langle a, s \rangle \Rightarrow v \quad \text{ahol } v \in \mathbb{Z} \cup \{tt, ff\}$$

(Vegyük észre, hogy ezekben a számításokban az állapotokat még nem változtatjuk, tehát csak lemásolódnak az egyes átmenetekkel.)

Következtetési szabályok

Definiáltuk, mit értünk konfiguráció alatt, most azt kellene megadnunk, hogyan viselkedik az átmenet-reláció: mely konfigurációkból mely más konfigurációkba léphet az absztrakt gép.

- Következtetési szabályokkal (azok sémáival) adjuk meg
- A szabályok három részből állnak: premissza (előfeltétel), konklúzió és további feltételek

$$\frac{\mathit{Premises}}{\mathit{Conclusion}}\ \mathit{Conditions}$$

A szabály szerint a konklúzió akkor érvényes, ha a premissza kikövetkeztethető és a feltételek teljesülnek.

- A feltételek nélküli szabályok az axiómák

$$\frac{}{\mathit{Axiom}}\ \mathit{Conditions}$$

Az axiómák példányai lesznek a levezetések gyökerében.

Kifejezések operációs szemantikája (1)

Számliterálok

$$\frac{}{\langle n, s\rangle \Rightarrow \mathcal{N}[\![n]\!]}$$

Változók

$$\frac{}{\langle x, s\rangle \Rightarrow s[x]}$$

Bináris műveletek (LHS)

$$\frac{\langle a_1, s\rangle \Rightarrow \langle a_1', s\rangle}{\langle a_1 \circ a_2, s\rangle \Rightarrow \langle a_1' \circ a_2, s\rangle} \quad \circ \in \{+, -, <, =\}$$

$$\frac{\langle a_1, s\rangle \Rightarrow i}{\langle a_1 \circ a_2, s\rangle \Rightarrow \langle n \circ a_2, s\rangle} \quad \circ \in \{+, -, <, =\}, i \in \mathbb{Z}, \mathcal{N}[\![n]\!] = i$$

Kifejezések operációs szemantikája (2)

Bináris műveletek (RHS)

$$\frac{\langle a_2, s\rangle \Rightarrow \langle a_2', s\rangle}{\langle n \circ a_2, s\rangle \Rightarrow \langle n \circ a_2', s\rangle} \quad \circ \in \{+, -, <, =\}$$

Bináris műveletek (utolsó lépés)

$$\frac{\langle a_2, s\rangle \Rightarrow i}{\langle n \circ a_2, s\rangle \Rightarrow \mathcal{N}[\![n]\!] \circ i} \quad \circ \in \{+, -, <, =\}, i \in \mathbb{Z}$$

- Az utolsó szabályban az $\mathcal{N}[\![n]\!] \circ i$ a művelet tényleges végrehajtása az egész számok körében ($<$ és $=$ a *tt* vagy *ff* logikai értékek valamelyikét eredményezik)

Aritmetikai negáció

$$\frac{\langle a, s\rangle \Rightarrow \langle a', s\rangle}{\langle \text{-}a, s\rangle \Rightarrow \langle \text{-}a', s\rangle} \qquad \frac{\langle a, s\rangle \Rightarrow i}{\langle \text{-}a, s\rangle \Rightarrow \mathbf{0} - i} \quad i \in \mathbb{Z}$$

Kifejezések operációs szemantikája (3)

Logikai literálok

$$\frac{}{\langle true, s\rangle \Rightarrow tt} \qquad \frac{}{\langle false, s\rangle \Rightarrow ff}$$

Negáció

$$\frac{\langle b, s\rangle \Rightarrow \langle b', s\rangle}{\langle \neg b, s\rangle \Rightarrow \langle \neg b', s\rangle} \qquad \frac{\langle b, s\rangle \Rightarrow l}{\langle \neg b, s\rangle \Rightarrow \neg l} \quad l \in \{tt, ff\}$$

Megjegyzés: vegyük észre, hogy az utolsó szabály konklúziójában szereplő $\neg$ szimbólum különböző jelentéssel bír az átmenet jobb és bal oldalán: a bal oldalon az absztrakt szintaxis egy szimbóluma, a jobb oldalon a logikai tagadás művelete. Hasonló megjegyzés igaz a legtöbb művelet leírására.

Az utolsó szabályt helyettesíthetnénk ezzel a kettővel:

$$\frac{\langle b, s\rangle \Rightarrow tt}{\langle \neg b, s\rangle \Rightarrow ff} \qquad \frac{\langle b, s\rangle \Rightarrow ff}{\langle \neg b, s\rangle \Rightarrow tt}$$

Kifejezések operációs szemantikája (4)

Konjunkció (LHS)

$$\frac{\langle b_1, s\rangle \Rightarrow \langle b_1', s\rangle}{\langle b_1 \wedge b_2, s\rangle \Rightarrow \langle b_1' \wedge b_2, s\rangle}$$

$$\frac{\langle b_1, s\rangle \Rightarrow tt}{\langle b_1 \wedge b_2, s\rangle \Rightarrow \langle true \wedge b_2, s\rangle} \qquad \frac{\langle b_1, s\rangle \Rightarrow ff}{\langle b_1 \wedge b_2, s\rangle \Rightarrow \langle false \wedge b_2, s\rangle}$$

Konjunkció (RHS)

$$\frac{\langle b_2, s\rangle \Rightarrow \langle b_2', s\rangle}{\langle b \wedge b_2, s\rangle \Rightarrow \langle b \wedge b_2', s\rangle} \quad b \in \{true, false\}$$

Konjunkció (utolsó átmenet)

$$\frac{\langle b_2, s\rangle \Rightarrow l}{\langle true \wedge b_2, s\rangle \Rightarrow l} \qquad \frac{\langle b_2, s\rangle \Rightarrow l}{\langle false \wedge b_2, s\rangle \Rightarrow ff} \quad l \in \{tt, ff\}$$

A számliterálokról

- A formális szemantikában éles különbséget teszünk a számliterálok és azok jelentése között.
- A számokat sokféle formában le lehet írni (pl. különböző számrendszerekben), de a jelentésük minden esetben egy egész szám. Ezt a jelentést az $\mathcal{N}$ függvény adja meg, amelyről feltesszük, hogy definiált.
- Például, ha a számliterálokat decimális számrendszerben írjuk, akkor a leképezés triviális: $\mathcal{N}[\![n]\!] = \mathbf{n}$ minden $n \in \mathbb{Z}$ esetén.
- A szemantikus számokat félkövéren szedjük, a nem félkövér számok szintaktikus elemek.
- Az $\mathcal{N}[\![n]\!]$ formulában használt speciális zárójeleket gyakran használják szemantika definíciókban; ami a zárójelek között van, azt szintaktikus egységként (vagy annak mintájaként) kell kezelni.

Bizonyos irodalmakban és a gyakorlatban kiterjesztik a szintaxist a szemantikus értékekkel, ezzel egyszerűsítve a leírást.

Levezetés példa

Számoljuk ki az “$(y + 5) + z$” kifejezés értékét abban az állapotban, ahol y értéke 12 és z értéke 34!

A lineáris levezetés (konfigurációsorozat) a következőképp néz ki:

$$
\begin{aligned}
&\langle\ (y + 5) + z, && [y \mapsto 12, z \mapsto 34]\ \rangle \Rightarrow \\
&\langle\ (12 + 5) + z, && [y \mapsto 12, z \mapsto 34]\ \rangle \Rightarrow \\
&\langle\ 17 + z, && [y \mapsto 12, z \mapsto 34]\ \rangle \Rightarrow \\
&\ \mathbf{51} && \checkmark
\end{aligned}
$$

$$
\dfrac{\dfrac{\dfrac{}{\langle y, s\rangle \Rightarrow \mathbf{12}}\ \textit{Változó (axióma)}}{\langle y + 5, s\rangle \Rightarrow \langle 12 + 5, s\rangle}\ \textit{Bináris kifejezés LHS (utolsó átmenet)}}{\langle (y + 5) + z, s\rangle \Rightarrow \langle (12 + 5) + z, s\rangle}\ \textit{Bináris kifejezés LHS}
$$

Ha tranzitívvá tesszük az átmenetrelációt,

$$\frac{c_1 \Rightarrow c_2 \qquad c_2 \Rightarrow c_3}{c_1 \Rightarrow c_3}$$

a levezetést faként tudjuk ábrázolni:

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle y, s\rangle \Rightarrow \mathbf{12}}{\langle y+5, s\rangle \Rightarrow \langle 12+5, s\rangle} \quad \dfrac{\langle 5, s\rangle \Rightarrow \mathbf{5}}{\langle 12+5, s\rangle \Rightarrow \mathbf{12{+}5}}}{\langle y+5, s\rangle \Rightarrow \mathbf{17}}}{\langle (y+5)+z, s\rangle \Rightarrow \langle 17+z, s\rangle} \quad \dfrac{\langle z, s\rangle \Rightarrow \mathbf{34}}{\langle 17+z, s\rangle \Rightarrow \mathbf{17{+}34}}}{\langle (y+5)+z, s\rangle \Rightarrow \mathbf{51}}$$

Denotációs szemantika

- Csakúgy, mint eddig, az absztrakt szintaxison definiáljuk
- Matematikai objektumokat (pl. számokat, n-eseket, függvényeket) rendelünk a nyelvi szerkezetekhez

- Szemantikus függvények klózait (különböző mintákra vett eseteit) fogjuk megadni
- Általában totálisak és kompozicionálisak a definíciók

Szemantikus tartományok (semantic domains)

Az absztrakt szintaktikus szerkezeteket szemantikus objektumokra képezzük le, amelyeket a szemantikus tartományok definiálnak.

Semantic domains

$$\text{Integer} = \{\ldots, \text{-2}, \text{-1}, 0, 1, 2, \ldots\} = \mathbb{Z}$$
$$\text{Boolean} = \{tt, ff\} \qquad \textit{(igazságértékek halmaza)}$$

- A jelentést úgy adjuk meg, hogy definiáljuk a kifejezések értékét az állapot függvényében
- Az aritmetikai kifejezések értéke egy egész szám, a logikai kifejezések értéke egy igazságérték
- Vegyük észre, hogy a szemantikus értékek (tt, ff) itt is megkülönböztetésre kerültek a szintaktikus elemektől (**true** és **false**)

Szemantikus függvények

$$\mathcal{A} : \mathrm{Aexp} \to (\mathrm{State} \to \mathrm{Integer})$$
$$\mathcal{B} : \mathrm{Bexp} \to (\mathrm{State} \to \mathrm{Boolean})$$

- Minden szintaktikus kategóriához (nemterminálishoz) készítünk egy szemantikus függvényt
- A szintaktikus kategória szabályaihoz függvényklózokat írunk fel
- A függvényeket Curry-formában adjuk meg, tehát a szemantikus függvény egy szintaktikus elemre egy újabb függvényt határoz meg, amely az állapotokhoz szemantikus értékeket rendel

Kifejezések denotációs szemantikája (1)

A szemantikus függvényt definiáljuk az adott szintaktikus kategória minden bázis elemére és összetett elemére (ügyelve arra, hogy a definíciók kompozicionálisak legyenek).

Szemantikus megfeleltetések (aritmetikai kifejezések)

$$\mathcal{A}[\![n]\!]s = \mathcal{N}[\![n]\!]$$
$$\mathcal{A}[\![x]\!]s = s[x]$$
$$\mathcal{A}[\![a_1 + a_2]\!]s = \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s$$
$$\mathcal{A}[\![a_1 - a_2]\!]s = \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s$$
$$\mathcal{A}[\![\text{-}a]\!]s = \mathbf{0} - \mathcal{A}[\![a]\!]s$$

Az “$a_1 + a_2$” jelentése csak a részkifejezéseitől (a_1 és a_2) függ, ahogy a többi klóz esetében is, tehát a definíció kompozicionális.

Szemantikus megfeleltetések (logikai kifejezések)

$$\mathcal{B}[\![\mathit{true}]\!]s = \mathit{tt}$$

$$\mathcal{B}[\![\mathit{false}]\!]s = \mathit{ff}$$

$$\mathcal{B}[\![a_1 = a_2]\!]s = \begin{cases} \mathit{tt} & \text{ha } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathit{ff} & \text{ha } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![a_1 < a_2]\!]s = \begin{cases} \mathit{tt} & \text{ha } \mathcal{A}[\![a_1]\!]s < \mathcal{A}[\![a_2]\!]s \\ \mathit{ff} & \text{ha } \mathcal{A}[\![a_1]\!]s \geq \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![\neg b]\!]s = \begin{cases} \mathit{tt} & \text{ha } \mathcal{B}[\![b]\!]s = \mathit{ff} \\ \mathit{ff} & \text{ha } \mathcal{B}[\![b]\!]s = \mathit{tt} \end{cases}$$

$$\mathcal{B}[\![b_1 \wedge b_2]\!]s = \begin{cases} \mathit{tt} & \text{ha } \mathcal{B}[\![b_1]\!]s = \mathit{tt} \quad \text{és} \quad \mathcal{B}[\![b_2]\!]s = \mathit{tt} \\ \mathit{ff} & \text{ha } \mathcal{B}[\![b_1]\!]s = \mathit{ff} \quad \text{vagy} \quad \mathcal{B}[\![b_2]\!]s = \mathit{ff} \end{cases}$$

Számítsuk ki a denotációs szemantikában ugyanannak a kifejezésnek az értékét, amelynek az operációsban már kiszámoltuk!

Legyen az s állapot $[y \mapsto 12, z \mapsto 34]$, ekkor

$$\begin{aligned}
\mathcal{A}[\![(y+5)+z]\!]s &= \mathcal{A}[\![y+5]\!]s + \mathcal{A}[\![z]\!]s \\
&= \mathcal{A}[\![y+5]\!]s + \mathbf{34} \\
&= (\mathcal{A}[\![y]\!]s + \mathcal{A}[\![5]\!]s) + \mathbf{34} \\
&= (\mathbf{12} + \mathcal{N}[\![5]\!]) + \mathbf{34} \\
&= (\mathbf{12} + \mathbf{5}) + \mathbf{34} \\
&= \mathbf{51}
\end{aligned}$$

Kompozicionalitás

A megadott denotációs szemantika kompozicionálisan definiált:

- Minden szintaktikai elemhez megadtunk egy függvényklózt
- Az összetett kifejezések jelentése a részkifejezések jelentésének függvényében került megadásra

- Érdemes megfigyelni az aritmetikai negációt, amelyet tipikusan rosszul szoktak definiálni (pl. $\mathcal{A}[\![\text{-}a]\!]s = \mathcal{A}[\![0 - a]\!]s$)

- A kompozicionalitásnak nagy jelentősége lesz a későbbiekben, hiszen szükséges ahhoz, hogy strukturális indukciós bizonyításokat végezhessünk a szemantikára vonatkozóan

Megjegyzések

Indukcióval bizonyítható, hogy

- A megadott szemantikadefiníció teljes: minden lehetséges kifejezésnek megadtuk a jelentését
- A szemantikadefiníció konzisztens: ha két különböző jelentés is levezethető egy kifejezéshez, akkor azok ekvivalensek (ebben az esetben megegyeznek)

- A kifejezésekre definiált operációs és a denotációs szemantika ekvivalens

Összegzés

- Konkrét szintaxis kontra absztrakt szintaxis
- Kifejezések absztrakt szintaxisa, metaváltozók
- Strukturális operációs szemantika
- Denotációs szemantika
- A szemantikadefiníciók tulajdonságai

$\mathbb{K}$

Egy egyszerű kifejezéseket leíró nyelv szemantikája elérhető a kurzus anyagai között, amely segít megérteni a konfiguráció és az állapot fogalmát. (A példában a szintaktikus és a szemantikus számok nincsenek különválasztva.) A small-step stílusú operációs szemantikát a $\mathbb{K}$ keretrendszerben definiáltuk, kipróbálható a 3.5 és a 3.6 verziókkal.

Strukturális operációs szemantika

Horpácsi Dániel
daniel-h@elte.hu

“Small-step” műveleti szemantika
imperatív programkonstrukciókhoz

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- **Operációs (műveleti)** — deduktív levezetésekkel

  Megadja, hogyan kell végrehajtani a programot

- **Denotációs (leíró)** — matematikai objektumokkal

  A jelentést denotációk hozzárendelésével adja meg

Utasítások és állapotok

- Az előző előadáson definiáltuk egyszerű kifejezések szemantikáját, most továbblépünk: formálisan adjuk meg az alapvető imperatív nyelvi elemek jelentését (a példanyelv a “While”)
- Mostantól az állapotot (a számítás kontextusát) nem csak olvasni, módosítani is fogják majd a nyelvi elemek

- Míg a kifejezések adott értékekre értékelődnek ki és nem változtatták meg az állapotot, addig az utasításoknak nincs értéke, csak mellékhatása: megváltoztathatják a változók értékét
- Az utasítás eredménye nem egy érték, hanem az új állapot
- Valójában az utasítás szemantikája nem egy állapot, hanem állapotok közötti reláció: milyen állapotból melyik másik állapotba jut a program az utasítás végrehajtásával

Absztrakt szintaxis

- A nyelvi elemek szerkezetét az absztrakt szintaxis definiálja
- Feltesszük, hogy a vizsgált kifejezések és utasítások statikus szemantikusan helyesek

- Az absztrakt szintaxis a szemantikadefiníció alapja
- A résznyelvek szintaktikus kategóriákat határoznak meg, amelyeket absztrakt produkciós szabályokkal induktívan definiálunk
- Tulajdonképp megadjuk az összes lehetséges absztrakt szintaxisfát, amelyek a nyelv mondataihoz tartoznak

While: absztrakt szintaxis

Szintaktikus kategóriák és metaváltozóik

$S \in \mathrm{Stm}$ (utasítások halmaza)

n	$\in$	Num	(számliterálok)	(token)
x	$\in$	Var	(változók)	(token)
a	$\in$	Aexp	(aritmetikai kifejezések)	
b	$\in$	Bexp	(logikai kifejezések)	

Produkciós szabályok

$$S ::= \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S$$

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a$$
$$b ::= \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2$$

Állapotok

- A kifejezések és utasítások változókat is tartalmazhatnak, tehát a jelentésük függhet a változókörnyezettől (az állapottól, amelyben kiértékeljük őket), és meg is változtathatják az állapotot
- Az egyszerűség kedvéért most csak egész számokat vehetnek fel a változók, de a modell általánosítható

- Az állapotokat függvényekkel reprezentáljuk:

$$s \in \mathrm{State} = \mathrm{Var} \to \mathbb{Z}$$

- Minden ilyen függvény értékeket rendel a változószimbólumokhoz
- Általában listaként formázzuk, pl.: $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$

Állapotok: lekérdezés és módosítás

Vegyük a következő állapotot: $s \in \mathrm{State} = \mathrm{Var} \to \mathbb{Z}$

- Ha hivatkozunk egy változó értékére, ki kell olvasnunk a "memóriából"

  $s[x]$ megadja az x változó értékét az s állapotban

- Amikor értéket adunk egy változónak, frissíteni kell az állapotot

  $s[y \mapsto v]$ megad egy olyan állapotot, amely megegyezik az s állapottal, de y helyen az értéke v

Következésképp teljesül, hogy $$(s[y \mapsto v])[x] = \begin{cases} v & \text{if } x = y \\ s[x] & \text{if } x \neq y \end{cases}$$

Strukturális operációs szemantika

A strukturális operációs szemantika megadja, hogyan kellene lépésről lépésre végrehajtani a programot egy absztrakt gépen.

- Átmenetrendszert definiálunk konfigurációk között
- Az átmeneteket következtetési szabályokkal határozzuk meg
- A szabályokban megjelenő konfiguráció-sémák tetszőleges szintaktikus elemmel példányosíthatóak
- Természetes levezetést használunk az átmenetek érvényességének bizonyítására
- A szabályok által leírt átmenetrendszer segítségével kiszámítható a szintaktikus elem szemantikája (esetleg normálformája)
- A levezetések általában szintaxis-vezéreltek

A következőkben definiálni fogjuk az alapvető imperatív utasítások strukturális operációs szemantikáját, amelyhez azonban felhasználjuk a kifejezések denotációs szemantikáját.

Az átmenetrendszer

- A végrehajtás állapotai (konfigurációk) két típusból kerülnek ki: vagy egy utasítás és egy állapot párja, vagy pedig egy állapot (a végállapot)
- (vö. a szintézis és verifikációval: “memóriaállapot” kontra “programállapot”)
- A konfigurációátmeneteket a következőképp jelöljük:

  A számítás egy köztes lépése:

  $$\langle S, s\rangle \Rightarrow \langle S', s'\rangle \qquad s, s' \in \mathrm{State}$$

  Az S végrehajtása véget ér az s' állapotot eredményezve:

  $$\langle S, s\rangle \Rightarrow s' \qquad s, s' \in \mathrm{State}$$

- Egy $\langle S, s\rangle$ konfigurációt *beragadt vagy zsákutca* konfigurációnak hívunk, ha nincsen olyan c konfiguráció, amelyre $\langle S, s\rangle \Rightarrow c$

Következtetési szabályok

Definiáltuk a konfiguráció fogalmát, most formálisan definiáljuk a megengedett átmeneteket a konfigurációk között.

- Következtetési szabályok (sémáinak) segítségével definiáljuk, mely konfigurációk között lehetséges átmenet
- A szabályok premisszából, konklúzióból és feltételekből állnak

$$\frac{Premises}{Conclusion}\ Conditions$$

A konklúzióban szereplő átmenet érvényes, ha a premisszában lévő átmenet levezethető és a feltétel teljesül.

- A premissza nélküli szabályok az axiómák.

$$\frac{}{Axiom}\ Conditions$$

A While nyelv természetes operációs szemantikája

Skip

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

Értékadás

$$\frac{}{\langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]}$$

Szekvencia

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

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

A While nyelv természetes operációs szemantikája

Skip

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

Értékadás

$$\frac{}{\langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]}$$

Szekvencia

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

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

A While nyelv természetes operációs szemantikája

Skip

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

Értékadás

$$\frac{}{\langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]}$$

Szekvencia

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

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

A While nyelv természetes operációs szemantikája

Elágazás

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

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

Mivel a kifejezésekhez definiált operációs és denotációs szemantika ekvivalens, így a $\langle b, s\rangle \Rightarrow^* tt$ is írható a $\mathcal{B}[\![b]\!]s = tt$ feltétel helyett.

Ciklus

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

Értelmes ez a rekurzív ciklusszemantika? Lehetne másképp?

A While nyelv természetes operációs szemantikája

Elágazás

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

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

Mivel a kifejezésekhez definiált operációs és denotációs szemantika ekvivalens, így a $\langle b, s\rangle \Rightarrow^* tt$ is írható a $\mathcal{B}[\![b]\!]s = tt$ feltétel helyett.

Ciklus

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

Értelmes ez a rekurzív ciklusszemantika? Lehetne másképp?

Levezetési láncok

Az S programhoz és s állapothoz tartozó levezetési lánc vagy

- egy véges konfigurációsorozat

$$c_0, c_1, c_2 \ \ldots \ c_k \qquad (k \geq 0)$$

ahol

$$c_0 = \langle S, s \rangle$$

$$c_i \Rightarrow c_{i+1} \quad (0 \leq i < k)$$

és c_k terminális vagy zsákutca konfiguráció

- vagy pedig egy végtelen konfigurációsorozat

$$c_0, c_1, c_2 \ \ldots$$

ahol

$$c_0 = \langle S, s \rangle$$

$$c_i \Rightarrow c_{i+1} \quad (0 \leq i)$$

Megjegyzések

- A $c_0 \Rightarrow^i c_i$ azt jelöli, hogy létezik i lépésből álló levezetési lánc c_0 konfigurációból c_i konfigurációba
- $c_0 \Rightarrow^* c_i$ akkor áll fenn, ha létezik véges sok lépésből álló lánc a konfigurációk között
- Fontos: $c_0 \Rightarrow^i c_i$ és $c_0 \Rightarrow^* c_i$ csak akkor állhat fenn, ha c_i terminális vagy zsákutca konfiguráció, hiszen csak ekkor tekinthető levezetési láncnak a konfigurációsorozat

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor
létezik egy s' állapot és k_1, k_2 természetes számok,
hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$, továbbá $k = k_1 + k_2$.

Ha $\langle S_1, s\rangle \Rightarrow^k s'$ akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

$\langle S_1; S_2, s\rangle \Rightarrow^* \langle S_2, s'\rangle$ *nem* implikálja, hogy $\langle S_1, s\rangle \Rightarrow^* s'$

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor létezik s' állapot és k_1, k_2 természetes számok, hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ és $k = k_1 + k_2$.

Minden k-ra bizonyítjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$$\langle S_1; S_2, s\rangle \Rightarrow^{k+1} s'' \quad \text{azaz} \quad \langle S_1; S_2, s\rangle \Rightarrow \gamma \Rightarrow^k s''$$

1. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \Rightarrow^k s'' \qquad$ mert $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$

   $\langle S_1', s'\rangle \Rightarrow^{l_1} s^*$ és $\langle S_2, s^*\rangle \Rightarrow^{l_2} s''$ ahol $l_1 + l_2 = k$ (hipotézis)

   Ekkor $k_1 = l_1 + 1$ és $k_2 = l_2$, tehát

   $$k_1 + k_2 = (l_1 + 1) + l_2 = (l_1 + l_2) + 1 = k + 1$$

2. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \Rightarrow^k s'' \qquad$ azaz $\langle S_1, s\rangle \Rightarrow s'$

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor létezik s' állapot és k_1, k_2 természetes számok, hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ és $k = k_1 + k_2$.

Minden k-ra bizonyítjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$$\langle S_1; S_2, s\rangle \Rightarrow^{k+1} s'' \quad \text{azaz} \quad \langle S_1; S_2, s\rangle \Rightarrow \gamma \Rightarrow^k s''$$

1. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \Rightarrow^k s''$ mert $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$

   $\langle S_1', s'\rangle \Rightarrow^{l_1} s^*$ és $\langle S_2, s^*\rangle \Rightarrow^{l_2} s''$ ahol $l_1 + l_2 = k$ (hipotézis)

   Ekkor $k_1 = l_1 + 1$ és $k_2 = l_2$, tehát

   $$k_1 + k_2 = (l_1 + 1) + l_2 = (l_1 + l_2) + 1 = k + 1$$

2. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \Rightarrow^k s''$ azaz $\langle S_1, s\rangle \Rightarrow s'$

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor létezik s' állapot és k_1, k_2 természetes számok, hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ és $k = k_1 + k_2$.

Minden k-ra bizonyítjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$\langle S_1; S_2, s\rangle \Rightarrow^{k+1} s''$ azaz $\langle S_1; S_2, s\rangle \Rightarrow \gamma \Rightarrow^k s''$

1. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \Rightarrow^k s''$ mert $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$

   $\langle S_1', s'\rangle \Rightarrow^{l_1} s^*$ és $\langle S_2, s^*\rangle \Rightarrow^{l_2} s''$ ahol $l_1 + l_2 = k$ (hipotézis)

   Ekkor $k_1 = l_1 + 1$ és $k_2 = l_2$, tehát

   $k_1 + k_2 = (l_1 + 1) + l_2 = (l_1 + l_2) + 1 = k + 1$

2. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \Rightarrow^k s''$ azaz $\langle S_1, s\rangle \Rightarrow s'$

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor létezik s' állapot és k_1, k_2 természetes számok, hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ és $k = k_1 + k_2$.

Minden k-ra bizonyítjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$\langle S_1; S_2, s\rangle \Rightarrow^{k+1} s''$ azaz $\langle S_1; S_2, s\rangle \Rightarrow \gamma \Rightarrow^k s''$

1. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \Rightarrow^k s''$ mert $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$

   $\langle S_1', s'\rangle \Rightarrow^{l_1} s^*$ és $\langle S_2, s^*\rangle \Rightarrow^{l_2} s''$ ahol $l_1 + l_2 = k$ (hipotézis)

   Ekkor $k_1 = l_1 + 1$ és $k_2 = l_2$, tehát

   $k_1 + k_2 = (l_1 + 1) + l_2 = (l_1 + l_2) + 1 = k + 1$

2. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \Rightarrow^k s''$ azaz $\langle S_1, s\rangle \Rightarrow s'$

Ha $\langle S_1; S_2, s\rangle \Rightarrow^k s''$ akkor létezik s' állapot és k_1, k_2 természetes számok, hogy $\langle S_1, s\rangle \Rightarrow^{k_1} s'$ és $\langle S_2, s'\rangle \Rightarrow^{k_2} s''$ és $k = k_1 + k_2$.

Minden k-ra bizonyítjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$\langle S_1; S_2, s\rangle \Rightarrow^{k+1} s''$ azaz $\langle S_1; S_2, s\rangle \Rightarrow \gamma \Rightarrow^k s''$

1. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \Rightarrow^k s''$ mert $\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$

   $\langle S_1', s'\rangle \Rightarrow^{l_1} s^*$ és $\langle S_2, s^*\rangle \Rightarrow^{l_2} s''$ ahol $l_1 + l_2 = k$ (hipotézis)

   Ekkor $k_1 = l_1 + 1$ és $k_2 = l_2$, tehát

   $k_1 + k_2 = (l_1 + 1) + l_2 = (l_1 + l_2) + 1 = k + 1$

2. $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \Rightarrow^k s''$ azaz $\langle S_1, s\rangle \Rightarrow s'$

Ha $\langle S_1, s\rangle \Rightarrow^k s'$, akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

Minden k-ra belátjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
   - Vegyük észre, hogy ez a szekvencia szemantikájának egyik szabálya
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$$\langle S_1, s\rangle \Rightarrow^{k+1} s' \quad \text{azaz} \quad \langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \Rightarrow^k s'$$

   - $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$ (szekvencia szabálya miatt)
   - $\langle S_1'; S_2, s''\rangle \Rightarrow^k \langle S_2, s'\rangle$ (hipotézis)

Ha $\langle S_1, s\rangle \Rightarrow^k s'$, akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

Minden k-ra belátjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
   - Vegyük észre, hogy ez a szekvencia szemantikájának egyik szabálya
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

$$\langle S_1, s\rangle \Rightarrow^{k+1} s' \quad \text{azaz} \quad \langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \Rightarrow^k s'$$

   - $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$ (szekvencia szabálya miatt)
   - $\langle S_1'; S_2, s''\rangle \Rightarrow^k \langle S_2, s'\rangle$ (hipotézis)

Ha $\langle S_1, s\rangle \Rightarrow^k s'$, akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

Minden k-ra belátjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
   - Vegyük észre, hogy ez a szekvencia szemantikájának egyik szabálya
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

   $\langle S_1, s\rangle \Rightarrow^{k+1} s' \quad$ azaz $\quad \langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \Rightarrow^k s'$

   - $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$ (szekvencia szabálya miatt)
   - $\langle S_1'; S_2, s''\rangle \Rightarrow^k \langle S_2, s'\rangle$ (hipotézis)

Ha $\langle S_1, s\rangle \Rightarrow^k s'$, akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

Minden k-ra belátjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
   - Vegyük észre, hogy ez a szekvencia szemantikájának egyik szabálya
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

   $\langle S_1, s\rangle \Rightarrow^{k+1} s' \quad$ azaz $\quad \langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \Rightarrow^k s'$

   - $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$ (szekvencia szabálya miatt)
   - $\langle S_1'; S_2, s''\rangle \Rightarrow^k \langle S_2, s'\rangle$ (hipotézis)

Ha $\langle S_1, s\rangle \Rightarrow^k s'$, akkor $\langle S_1; S_2, s\rangle \Rightarrow^k \langle S_2, s'\rangle$

Minden k-ra belátjuk, a levezetési lánc hossza szerinti indukcióval.

1. Belátható, hogy az összefüggés fennáll az 1 hosszú levezetési láncokra
   - Vegyük észre, hogy ez a szekvencia szemantikájának egyik szabálya
2. Tegyük fel, hogy az összefüggés fennáll a maximum k hosszú levezetési láncokra, majd belátjuk, hogy a $k+1$ hosszú láncokra is teljesül

   $\langle S_1, s\rangle \Rightarrow^{k+1} s' \quad$ azaz $\quad \langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \Rightarrow^k s'$

   - $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$ (szekvencia szabálya miatt)
   - $\langle S_1'; S_2, s''\rangle \Rightarrow^k \langle S_2, s'\rangle$ (hipotézis)

Levezetési láncokhoz kapcsolódó fogalmak

Azt mondjuk, hogy az S program az s állapotból

- terminál

  ha létezik az $\langle S, s \rangle$ konfigurációból véges levezetési lánc

  - sikeresen terminál

    ha létezik olyan s' állapot, amelyre $\langle S, s \rangle \Rightarrow^* s'$
  - elakad (abortál)

    ha van olyan S' maradékprogram és s' állapot, amelyre $\langle S, s \rangle \Rightarrow^* \langle S', s' \rangle$ és $\langle S', s' \rangle$ egy beragadt konfiguráció
- divergál (nem terminál)

  ha létezik $\langle S, s \rangle$ konfigurációból induló végtelen levezetési lánc

  - a következőképp jelöljük: $\langle S, s \rangle \Rightarrow^* \infty$

Azt mondjuk, hogy az S program végrehajtása

- (mindig) terminál, ha minden s állapotból terminál
- (mindig) divergál, ha minden s állapotból divergál

Levezetési lánc: példa

Tegyük fel, hogy $s = [x \mapsto 41]$.

$$
\begin{aligned}
&\langle\, \textbf{while}\ x<42\ \textbf{do}\ x := x+1,\ && s\,\rangle \Rightarrow \\
&\langle\, \textbf{if}\ x<42\ \textbf{then}\ (x := x+1; \textbf{while}\ x<42\ \textbf{do}\ x := x+1)\ \textbf{else skip},\ && s\,\rangle \Rightarrow \\
&\langle\, x := x+1; \textbf{while}\ x<42\ \textbf{do}\ x := x+1,\ && s\,\rangle \Rightarrow \\
&\langle\, \textbf{while}\ x<42\ \textbf{do}\ x := x+1,\ && s[x \mapsto 42]\rangle \Rightarrow \\
&\langle\, \textbf{if}\ x<42\ \textbf{then}\ (x := x+1; \textbf{while}\ \ldots\ \textbf{do}\ \ldots)\ \textbf{else skip},\ && s[x \mapsto 42]\rangle \Rightarrow \\
&\langle\, \textbf{skip},\ && s[x \mapsto 42]\rangle \Rightarrow \\
&\quad s[x \mapsto 42] && = \\
&\quad [x \mapsto 42]
\end{aligned}
$$

Az egyes átmenetek helyessége a következtetési szabályokkal bizonyítható.

Megjegyzések

- Minden nemterminál konfigurációhoz létezik legalább egy levezetési lánc
- A fent definiált operációs szemantika determinisztikus, tehát minden konfigurációhoz pontosan egy levezetési lánc tartozik
- A While nyelv ezen változata determinisztikus és sosem kerül zsákutcába a programok végrehajtása (tehát ha egy program terminál, akkor sikeresen terminál)

- A következő előadásokon bővítjük majd a nyelvet olyan elemekkel, amelyek eredményezhetnek beragadt vagy nemdeterminisztikus számításokat

Szemantikus ekvivalencia

A szemantikus ekvivalencia megadja, hogy két utasítás jelentése megegyezik-e.

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

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

ahol c terminális vagy zsákutca konfiguráció.

Következésképp, ha egy program egy adott állapotból terminál, akkor a másiknak is terminálnia kell ugyanabban a konfigurációban (akár beragadt, akár sikeres), illetve, ha az egyik nem terminál, akkor a másik sem.

A szemantikus függvény

Az utasítások jelentését egy parciális függvénnyel karakterizáljuk (v.ö. korábbi tárgyakban programfüggvény és viselkedési reláció):

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

Ha a végrehajtás sikeresen terminál az s állapotból, akkor a terminálóó konfiguráció s' állapota adja a szemantikus függvény értékét:

$$\mathcal{S}_{sos}[\![S]\!]s = \begin{cases} s' & \text{ha } \langle S, s\rangle \Rightarrow^* s' \\ \mathit{undefined} & \text{egyébként} \end{cases}$$

Ha a számítás egy s állapotban elakad vagy divergál, akkor ott a szemantikus függvény nem definiált.

Kérdés: mi az $S_1 \equiv S_2$ és $\mathcal{S}_{sos}[\![S_1]\!] = \mathcal{S}_{sos}[\![S_2]\!]$ közötti összefüggés?

Összefoglaló

- Imperatív nyelvi elemek absztrakt szintaxisa
- Konfigurációk és átmenetek
- Strukturális operációs szemantika
- A szemantika tulajdonságai
- Levezetési láncok, terminálás
- Szemantikus ekvivalencia, szemantikus függvény

$\mathbb{K}$

A While nyelv (továbbá a foreach-jellegű ciklus) végrehajtható szemantikája elérhető a kurzus anyagai között. A small-step stílusú operációs szemantikát a $\mathbb{K}$ keretrendszerben definiáltuk, kipróbálható a 3.5 és a 3.6 verziókkal.

Természetes szemantika

Horpácsi Dániel
daniel-h@elte.hu

Természetes (big-step) műveleti szemantika

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- Operációs (műveleti)
  - Strukturális

    Minden lépés modellezése
  - **Természetes**

    A kezdő- és végállapotok közötti reláció felállítása
- Denotációs (leíró) — matematikai objektumokkal

  A jelentést denotációk hozzárendelésével adja meg

Small-step után big-step

- Az előző előadáson definiáltuk az alapvető imperatív konstrukciók strukturális operációs szemantikáját, amelyben minden végrehajtási lépést modelleztünk

- Most definiálunk egy másik jellegű operációs szemantikát, amely a kezdő- és végkonfigurációk közötti átmenetrelációt definiálja
- A módszer hasonló, az absztrakciós szint különbözik
- Az absztrakt szintaxis és a konfigurációk nem változnak, de az átmenetrelációt másképp adjuk meg

Emlékeztető: az absztrakt szintaxis

Szintaktikus kategóriák és azok definíciói, amelyek a nyelv absztrakt szintaxisfáit reprezentálják

Szintaktikus kategóriák

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változószimbólumok)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S
\end{array}
$$

Emlékeztető: állapotok

- A kifejezések és utasítások szemantikáját az állapot függvényében definiáljuk

$$s \in \mathrm{State} = \mathrm{Var} \to \mathbb{Z}$$

- Listaként formázzuk, pl.: $[x \mapsto 1, y \mapsto 2, z \mapsto 3]$
- Ha hivatkozunk egy változó értékére, ki kell olvasnunk a "memóriából"

$s[x]$ megadja az x változó értékét az s állapotban

- Amikor értéket adunk egy változónak, frissíteni kell az állapotot

$s[y \mapsto v]$ megad egy olyan állapotot, amely megegyezik az s állapottal, de y helyen az értéke v

Természetes operációs szemantika

A természetes szemantika műveleti úton definiálja a kezdő- és végkonfigurációk közötti relációt

- Operációs szemantika lévén itt is egy átmenetrelációt definiálunk
- Az átmeneteket következtetési szabályokkal határozzuk meg
- A szabályokban megjelenő konfiguráció-sémák tetszőleges szintaktikus elemmel példányosíthatóak
- Természetes levezetést használunk az átmenetek érvényességének bizonyítására
- A small-step szemantikával ellentétben itt a kikövetkeztetett átmenet mindig megadja a végállapotot, nincs további levezetésre szükség

Az átmenetrendszer

- A végrehajtás állapotai (konfigurációk) két típusból kerülnek ki: vagy egy utasítás és egy állapot párja, vagy pedig egy állapot (a végállapot)
- A konfigurációátmeneteket a következőképp jelöljük:

  Az S utasítás végrehajtása az s' állapotot eredményezi:

$$\langle S, s \rangle \to s' \qquad\qquad s, s' \in \mathrm{State}$$

Vegyük észre, hogy a stukturális operációs szemantikával ellentétben itt nincsenek köztes átmenetek, továbbá beragadt konfigurációba sem juthat a rendszer.

A While nyelv természetes szemantikája

Skip és értékadás hasonlóan a small-step definícióhoz:

Skip

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

Értékadás

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

A szekvencia big-step esetben egyetlen szabállyal megadható:

Szekvencia

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

A While nyelv természetes szemantikája

Elágazás

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

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

Ciklus

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

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

Az S program az s állapotból indítva

- terminál

  ha létezik s' állapot, amelyre $\langle S, s\rangle \to s'$

- divergál

  ha nem létezik s' állapot, amelyre $\langle S, s\rangle \to s'$

Azt mondjuk, hogy az S utasítás

- mindig terminál, ha minden s állapotból indítva terminál
- mindig divergál, ha minden s állapotból indítva divergál

$$\textbf{while } \neg(x = 1) \quad \textbf{do } (y := y * x\ ;\ x := x - 1)$$
$$\textbf{while } \neg(x < 1) \quad \textbf{do } (y := y * x\ ;\ x := x - 1)$$
$$\textbf{while } \neg(x < 1 \wedge 1 < x) \textbf{ do } (y := y * x\ ;\ x := x - 1)$$

Levezetési fák

- Ahogy korábban is, a fenti következtetési szabályokkal definiáltuk a lehetséges konfigurációátmeneteket
- Az egyes átmenetek levezetését levezetési fákkal reprezentáljuk

- A levezetési fák **gyökerében** a levezetni kívánt átmenet szerepel, míg a **levelek**et mindig axiómák adják
- A **köztes csúcsok** levezetési szabályok konklúziói úgy, hogy közvetlen gyerekeik a premisszáik
- A levezetési fa helyességéhez a szabályok feltételeinek teljesülése is szükséges

Megjegyzés: a levezetési fába nem az átmenetsémák, hanem azok példányai íródnak.

Levezetési fák konstruálása

Hogyan építsük fel az S programhoz és s állapothoz tartozó levezetési fát?

A fát a gyökerétől építjük, alulról felfelé; olyan szabályt (vagy axiómát) keresünk, amelyben a konklúzió baloldala illeszkedik az $\langle S, s \rangle$ konfigurációra:

- Ha van ilyen axióma és a feltételei teljesülnek, akkor a végállapotot meghatározza a konklúzió jobb oldala, a fa (részfa) elkészült.
- Ha van ilyen következtetési szabály, akkor a premisszáihoz is készítünk levezetési fákat; ha sikerül, továbbá a feltételek is teljesülnek, akkor a végállapotot meghatározza a konklúzió jobb oldala és a fa (részfa) építése kész.

(Vegyük észre, hogy a fenti algoritmus rekurzív.)

Példa levezetés

Vegyük az előző előadáson látott példát:

$$\textbf{while}\ x < 42\ \textbf{do}\ x := x+1$$

A végrehajtás az $s = [x \mapsto 41]$ állapotból indul.

$$\frac{\langle x := x+1, s\rangle \to s' \qquad \langle\ \textbf{while}\ x<42\ \textbf{do}\ x := x+1, s'\ \rangle \to s'}{\langle\ \textbf{while}\ x<42\ \textbf{do}\ x := x+1, s\ \rangle \to s'}$$

Nyilvánvalóan, $s' = s[x \mapsto 42]$.

Látható, hogy a small-step szemantikához képest sokkal tömörebb, absztraktabb a jelentés levezetése. Átgondolandó, hogy mely pontokon absztraktabb a leírás.

A természetes szemantika tulajdonságai

- Belátható, hogy a fent definiált természetes szemantika determinisztikus
- Látható, hogy a definíció még mindig nem kompozicionális, tehát strukturális indukció nem használható bizonyításhoz
- A szemantikára vonatkozó tulajdonságok a levezetési fák alakja szerinti indukcióval végezhetőek:
  - Bizonyítsuk, hogy a tulajdonság minden egyszerű fára (axiómára) tejesül
  - Lássuk be, hogy minden összetett fára teljesül: minden szabály esetében tegyük fel, hogy a részfa premisszáira teljesül (indukciós hipotézis), majd lássuk be a konklúzióra (feltéve, hogy a feltételek is teljesülnek)

A szemantikus ekvivalencia megadja, hogy két utasításnak megegyezik-e a hatása.

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

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

Belátható, hogy a ciklus ekvivalens a kicsomagoltjával:

$$\textbf{while } b \textbf{ do } S \quad \equiv \quad \textbf{if } b \textbf{ then } \big(S; \textbf{while } b \textbf{ do } S\big) \textbf{ else skip}$$

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

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

$$\frac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

$$\frac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff$$

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

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

$$\frac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''} \mathcal{B}[\![b]\!]s = tt$$

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

$$\frac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''} \mathcal{B}[\![b]\!]s = ff$$

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

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

$$\frac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

$$\frac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff$$

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

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

$$\frac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''} \mathcal{B}[\![b]\!]s = tt$$

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

$$\frac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''} \mathcal{B}[\![b]\!]s = ff$$

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

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

$$\frac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

$$\frac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff$$

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

- $$\dfrac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

- $$\dfrac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff, s = s''$$

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

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

- $$\dfrac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

- $$\dfrac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff, s = s''$$

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

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

- $$\dfrac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

- $$\dfrac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff, s = s''$$

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

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

- $$\dfrac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

- $$\dfrac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff, s = s''$$

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

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

- $$\dfrac{\dfrac{\langle S, s\rangle \to s' \qquad \langle \textbf{while } b \textbf{ do } S, s'\rangle \to s''}{\langle S; \textbf{while } b \textbf{ do } S, s\rangle \to s''}\ \text{(szekvencia)}}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = tt$$

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

- $$\dfrac{\dfrac{}{\langle \textbf{skip}, s\rangle \to s''}\ s = s''}{\langle \textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}, s\rangle \to s''}\ \mathcal{B}[\![b]\!]s = ff, s = s''$$

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

A szemantikus függvény

Az utasítások jelentését egy parciális függvény karakterizálja:

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

Tehát minden egyes utasításhoz van egy ilyen parciális függvény:

$$\mathcal{S}_{ns}[\![S]\!] \in \mathrm{State} \hookrightarrow \mathrm{State}$$

Ha a végrehajtás terminál az s állapotból, akkor a levezetett s' állapot adja meg az utasítás “eredményét”.

$$\mathcal{S}_{ns}[\![S]\!]s = \begin{cases} s' & \text{ha } \langle S, s\rangle \to s' \\ \mathit{undefined} & \text{egyébként} \end{cases}$$

Ha a kiértékelés végtelen ciklust/rekurziót eredményez, akkor a szemantika “nem definiált”.

$\mathcal{S}_{ns}$ kontra $\mathcal{S}_{sos}$

- Az elmúlt két előadáson definiáltunk strukturális és természetes szemantikát is a While nyelvhez
- A szekvencia, elágazás és ciklus leírásakor alapvetően különböző megközelítést alkalmaz a small-step és a big-step leírás
- Mindazonáltal belátható, hogy a két szemantika ekvivalens, azaz

Minden S utasításra: $\mathcal{S}_{sos}[\![S]\!] = \mathcal{S}_{ns}[\![S]\!]$.

Emlékeztetőül: $\mathcal{S}_{sos}[\![S]\!], \mathcal{S}_{ns}[\![S]\!] \in \mathrm{State} \hookrightarrow \mathrm{State}$.

Ekvivalencia: $\mathcal{S}_{ns}$ és $\mathcal{S}_{sos}$

Az ekvivalencia állítása, azaz $\forall S : \mathcal{S}_{sos}[\![S]\!] = \mathcal{S}_{ns}[\![S]\!]$, átfogalmazható a következőképp:

- Ha az S utasítás végrehajtása terminál az egyik szemantikában, terminálnia kell a másikban is, ugyanazzal a végállapottal
- Ha az S utasítás végrehajtása az egyik szemantikában divergál, akkor a másikban is divergálnia kell

Megjegyzés: az eddig tárgyalt szemantikák determinisztikusak voltak; továbbá zsákutca konfiguráció nem jelent meg a small-step szemantikában sem, így a terminálás alatt most sikeres terminálást értünk.

Ekvivalencia: $\mathcal{S}_{ns}$ és $\mathcal{S}_{sos}$

Az ekvivalencia a következőkkel belátható:

- Minden S utasításra, továbbá s és s' állapotra:

  $\langle S, s\rangle \Rightarrow^* s'$ implikálja, hogy $\langle S, s\rangle \to s'$

  Ha a végrehajtás terminál a stukturális szemantikában, akkor a természetes szemantikában is, ugyanazzal a végállapottal.

- Minden S utasításra, továbbá s és s' állapotra:

  $\langle S, s\rangle \to s'$ implikálja, hogy $\langle S, s\rangle \Rightarrow^* s'$

  Tehát ha a végrehajtás terminál a természetes szemantikában, akkor a strukturális szemantikában is, ugyanazzal a végállapottal.

Következésképp ugyanazon pontokon, ugyanúgy definiált a szemantika. Azaz, $\mathcal{S}_{sos}$ és $\mathcal{S}_{ns}$ mint szemantikus függvények minden pontban megegyeznek, ekvivalensek.

Áttekintés

- Természetes szemantika átmenetrendszerekkel
- Levezetési fák
- A természetes szemantika tulajdonságai
- A természetes szemantikus függvény
- A strukturális és természetes szemantika ekvivalenciája

Denotációs szemantika

Horpácsi Dániel
daniel-h@elte.hu

Az előadás témája

Leíró szemantika,

avagy a matematikai szemantika

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- Operációs (műveleti)
  - Strukturális

    Minden lépés modellezése
  - Természetes

    A kezdő- és végállapotok közötti reláció felállítása

- **Denotációs (leíró)**

  A jelentést matematikai objektumok hozzárendelésével adja meg

Előszó

- Az alapvető imperatív programkonstrukciók (a While nyelv) műveleti szemantikáját már áttekintettük
- Az operációs szemantika megadta, hogyan értékelődnek ki a kifejezések, illetve hogyan hajtódnak végre programok lépésről-lépésre

- Most egy még absztraktabb megközelítést tekintünk át, amikor a szemantikadefiníciót denotációs függvények adják meg
- Nem azt adjuk meg, hogyan működik, hanem hogy mi lesz a hatása

- Matematikai objektumokkal jellemezzük egy-egy konstrukció jelentését (pl. a számliterálokat az értékük jellemzi, a programok jelentését pedig parciális függvényekkel írjuk le)

Emlékeztető: az absztrakt szintaxis

Szintaktikus kategóriák és azok definíciói, amelyek a nyelv absztrakt szintaxisfáit definiálják

Szintaktikus kategóriák

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változószimbólumok)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \mathbf{true} \mid \mathbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \mathbf{skip} \mid x := a \mid S_1; S_2 \mid \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2 \mid \mathbf{while}\ b\ \mathbf{do}\ S
\end{array}
$$

Denotációs szemantika: az alapötlet

- Az operációs szemantikában induktívan definiálunk relációkat ($\Rightarrow$ és $\rightarrow$), amelyek az úgynevezett konfigurációk (programállapotok) között adják meg a lehetséges átmeneteket

- Amikor egy S utasítás hatásáról beszélünk, akkor a környezetére gyakorolt hatása az érdekes, tehát a mi esetünkben az, hogy hogyan változtatja meg az állapotot: $\{(s, s') \mid \langle S, s\rangle \rightarrow s'\}$

  $\{(s, s') \mid \langle S, s\rangle \Rightarrow^* s'\}$

S mindig meghatároz egy parciális függvényt az állapotok között: ez az S **denotációja**.

Denotációs szemantika: az alapötlet

- A denotációs szemantikában minden szintaktikus kategóriához (nyelvi konstrukcióhoz, résznyelvhez) megadunk egy szemantikus domaint
- Majd a szintaktikus domain elemeihez a szemantikus domain elemeit rendeljük

- A szemantikus függvényeket a szintaktikus kategóriák minden mintájára megadjuk
  - Kompozicionálisan, melynek köszönhetően strukturális indukciót használhatunk programtulajdonságok bizonyítására

Hogyan definiáljuk az imperatív konstrukciókra?

Az utasítások jelentését egy parciális függvénnyel karakterizáljuk, amely állapotokat képez állapotokra. Az operációs szemantikával ellentétben a szemantikus függvény definíciója nem egy extra lépés, hanem maga a szemantikadefiníció.

$$\mathcal{S}_{ds} : \mathrm{Stm} \to (\mathrm{State} \hookrightarrow \mathrm{State})$$

Minden S utasításhoz tartozni fog egy parciális függvény:

$$\mathcal{S}_{ds}[\![S]\!] \in \mathrm{State} \hookrightarrow \mathrm{State}$$

- Hogyan definiálnánk a szekvenciát?
- Hogyan definiálnánk az elágazást?
- Hogyan definiálnánk a ciklust?

$$\mathcal{S}_{ds} : \mathrm{Stm} \to (\mathrm{State} \hookrightarrow \mathrm{State})$$

Az operációs szemantikával ellentétben a szemantikus függvény megadása nem kiegészítő lépés, hanem maga a szemantikadefiníció.

A While leíró szemantikája

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

A speciális zárójelek: [[és]]

- Az $\mathcal{A} : \mathrm{Aexp} \to (\mathrm{State} \to \mathrm{Integer})$ szemantikus függvényt alkalmazva az *a* kifejezésre ($\mathcal{A}[\![a]\!]$) megkapjuk az *a* szintaktikus elem szemantikáját.

- Miért kell az *a*-t, a függvény argumentumát speciális zárójelek közé tenni? Azért, mert ezzel jelezzük, hogy ez a paraméter nem matematikai formula, hanem egy szintaktikus elem, így nem szabad hagyományos módon kiértékelni.
- Az $\mathcal{A}[\![3+2]\!]$ formulát akár $\mathcal{A}(\text{"}3+2\text{"})$ formában is jelölhetnénk (a lényeg a megkülönböztetés), de az előbbi jelölés az elterjedt
- Amikor a szemantikus függvényt adjuk meg, akkor szintaktikus mintákat írunk a zárójelek közé, tehát a szintaktikus elem metaváltozókat tartalmazhat (pl. $\mathcal{A}[\![a_1 + a_2]\!]$)

A szekvencia denotációs szemantikája

A függvénykompozíciót úgy értelmezzük, hogy csak akkor definiált, ha mindkét utasítás definiált az adott állapotban (az első az eredetiben, a második pedig a rákövetkezőben).
Ha valamelyik komponens szemantikája nem definiált, akkor a szekvencia sem definiált (pl. akár az első, akár a második program divergál, a szekvencia is divergál).

$$\mathcal{S}_{ds}[\![S_1; S_2]\!]s$$
$$= (\mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!])s \;=\; \mathcal{S}_{ds}[\![S_2]\!](\mathcal{S}_{ds}[\![S_1]\!]s) =$$
$$= \begin{cases} s'' & \text{ha } \exists s' \in \mathrm{State} : \mathcal{S}_{ds}[\![S_1]\!]s = s' \text{ és } \mathcal{S}_{ds}[\![S_2]\!]s' = s'' \\ \mathit{undef} & \text{ha } \mathcal{S}_{ds}[\![S_1]\!]s = \mathit{undef} \\ & \text{vagy } \exists s' \in \mathrm{State} : \mathcal{S}_{ds}[\![S_1]\!]s = s' \text{ és } \mathcal{S}_{ds}[\![S_2]\!]s' = \mathit{undef} \end{cases}$$

Segédfüggvény: *cond*

Az esetfüggő szemantika megadására bevezetünk egy segédfüggvényt. A *cond* függvény esetszétválasztást definiál:

$$cond : (\mathrm{State} \to \mathrm{Boolean}) \times (\mathrm{State} \hookrightarrow \mathrm{State}) \times (\mathrm{State} \hookrightarrow \mathrm{State}) \to (\mathrm{State} \hookrightarrow \mathrm{State})$$

Az értelmezési tartománya egy hármas (azt mondjuk, három paramétere van): az első a feltétel, a másik kettő az ágak. Az értékkészlete a szokásos állapotok közötti parciális függvény.

$$cond(p, g_1, g_2)s = \begin{cases} g_1\ s & \text{ha} \quad p\ s = tt \\ g_2\ s & \text{ha} \quad p\ s = ff \end{cases}$$

A *cond* az állapot függvényében használja az egyik vagy másik szemantikadefiníciót: ha a feltétel az állapotban igaz, akkor az első ágat használja, ha hamis, akkor a másodikat.

Az elágazás leíró szemantikája

$$\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]s =$$

$$= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])s =$$

$$= \begin{cases} \mathcal{S}_{ds}[\![S_1]\!]s & \text{ha} \quad \mathcal{B}[\![b]\!]s = tt \\ \mathcal{S}_{ds}[\![S_2]\!]s & \text{ha} \quad \mathcal{B}[\![b]\!]s = ff \end{cases}$$

$$= \begin{cases} s' & \text{ha} \quad \mathcal{B}[\![b]\!]s = tt \ \text{ és} \ \mathcal{S}_{ds}[\![S_1]\!]s = s' \\ & \text{vagy ha} \ \mathcal{B}[\![b]\!]s = ff \ \text{ és} \ \mathcal{S}_{ds}[\![S_2]\!]s = s' \\ undef & \text{ha} \quad \mathcal{B}[\![b]\!]s = tt \ \text{ és} \ \mathcal{S}_{ds}[\![S_1]\!]s = undef \\ & \text{vagy ha} \ \mathcal{B}[\![b]\!]s = ff \ \text{ és} \ \mathcal{S}_{ds}[\![S_2]\!]s = undef \end{cases}$$

A ciklus leíró szemantikája

A korábbi szemantikadefiníciók alapján elvárjuk, hogy a ciklus szemantikája ekvivalens a kifejtésének szemantikájával:

$$\begin{aligned}\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &\equiv \\ &\equiv \mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}]\!] \\ &\equiv cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S; \textbf{while } b \textbf{ do } S]\!], \mathcal{S}_{ds}[\![\textbf{skip}]\!]) \\ &\equiv cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})\end{aligned}$$

Továbbá gondolhatnánk, hogy definiálhatjuk is ezt az összefüggést kihasználva:

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

Azonban ez a definíció nem lenne kompozicionális, tehát nem ezt fogjuk használni.

A ciklus leíró szemantikája

Legyen $g = \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!]$.

Alakítsuk át az előző formulát, helyettesítsük g-t:

$$g = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$$

Tisztán látható, hogy ez egy rekurzív formula, mivel a g (a ciklus szemantikája) definíciójában a g újra megjelenik a formula jobb oldalán is.

Ahhoz, hogy g-t meghatározzuk, bevezetünk egy F funkcionált:

$$F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$$

Így F fixpontja megadja g-t. Azok lesznek “megfelelő” szemantikadefiníciók, amelyekre $F(g) = g$, de a ciklus szemantikájaként mi F legkisebb fixpontját fogjuk használni.

A ciklus leíró szemantikája

Tehát a ciklus denotációs szemantikáját a következő funkcionál fixpontjának kiszámításával kapjuk:

$$F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$$

Azaz

$$F(g)s = \begin{cases} (g \circ \mathcal{S}_{ds}[\![S]\!])s & \text{ha} \quad \mathcal{B}[\![b]\!]s = tt \\ s & \text{ha} \quad \mathcal{B}[\![b]\!]s = ff \end{cases}$$

A fixpont kiszámításához egy fixpont-kombinátort (FIX) használunk, amely a tetszőlegesen sok fixpont közül a legkisebbet fogja megadni.

Szemantikus ekvivalencia

A szemantikus ekvivalencia megadja, mikor tekintünk két utasítást ekvivalensnek, mikor megegyező a két utasítás hatása.

S_1 és S_2 szemantikusan ekvivalensek ($S_1 \equiv S_2$) a leíró szemantikában akkor és csak akkor, ha $\mathcal{S}_{ds}[\![S_1]\!] = \mathcal{S}_{ds}[\![S_2]\!]$.

Vagyis minden s és s' állapotra

- $\mathcal{S}_{ds}[\![S_1]\!]s = s'$ akkor és csak akkor $\mathcal{S}_{ds}[\![S_2]\!]s = s'$
- $\mathcal{S}_{ds}[\![S_1]\!]s = \mathit{undef}$ akkor és csak akkor $\mathcal{S}_{ds}[\![S_2]\!]s = \mathit{undef}$

Egy egyszerű utasítás szemantikája

Legyen $s = [x \mapsto 10]$ és $s' = [x \mapsto 11]$.

$$
\begin{aligned}
&\mathcal{S}_{ds}[\![x := x + 1 \ ; \ \textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!]s = \\
&\quad = (\mathcal{S}_{ds}[\![\textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!] \circ \mathcal{S}_{ds}[\![x := x + 1]\!])s = \\
&\quad = \mathcal{S}_{ds}[\![\textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!](\mathcal{S}_{ds}[\![x := x + 1]\!]s) = \\
&\quad = \mathcal{S}_{ds}[\![\textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!](s[x \mapsto \mathcal{A}[\![x + 1]\!]s]) = \\
&\quad = \mathcal{S}_{ds}[\![\textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!](s[x \mapsto 11]) =
\end{aligned}
$$

$$
\begin{aligned}
&\quad = \mathcal{S}_{ds}[\![\textbf{if } x > 10 \textbf{ then } x := 10 \textbf{ else skip}]\!]s' = \\
&\quad = cond(\mathcal{B}[\![x > 10]\!], \mathcal{S}_{ds}[\![x := 10]\!], \mathcal{S}_{ds}[\![\textbf{skip}]\!])s' = \\
&\quad = \begin{cases} \mathcal{S}_{ds}[\![x := 10]\!]s' & \text{ha} \quad \mathcal{B}[\![x > 10]\!]s' = tt \\ \mathcal{S}_{ds}[\![\textbf{skip}]\!]s' & \text{ha} \quad \mathcal{B}[\![x > 10]\!]s' = ff \end{cases} = \\
&\quad = \mathcal{S}_{ds}[\![x := 10]\!]s' = \\
&\quad = s[x \mapsto \mathcal{A}[\![10]\!]]s' = \\
&\quad = s'[x \mapsto 10] = s
\end{aligned}
$$

Megjegyzések

- Bizonyítható, hogy az eddigi szemantikadefinícióink ($\mathcal{S}_{ds}$, $\mathcal{S}_{sos}$ és $\mathcal{S}_{ns}$) ekvivalensek, tehát ugyanazokat a függvényeket rendelik az utasításhoz
- Ezek a módszerek nem egymás konkurenciái, hiszen különböző absztrakciós szinten definiálják ugyanazt a jelentésfogalmat

- A fentiek közül a denotációs szemantika a legabsztraktabb
- Az operációsakkal ellentétben ez teljesen kompozicionális
- A denotációs szemantikával kinyert jelentésfogalmak tovább transzformálhatóak

Áttekintés

- A denotációs szemantika alapötlete
- Leképezhető matematikai objektumok, domainek
- Kompozicionalitás és a strukturális indukció
- A különböző szemantikadefiníciók viszonya

λ

A While nyelv végrehajtható denotációs szemantikája elérhető a kurzus anyagai között. A leíró jellegű szemantikát Haskellben definiáltuk.

Fixpont elmélet

Horpácsi Dániel
daniel-h@elte.hu

Domainek és fixpontok

A denotációs szemantika hátterében

Szemantikus függvények

$$\mathcal{N} : \text{Num} \to \text{Integer}$$
$$\mathcal{A} : \text{Aexp} \to (\text{State} \to \text{Integer})$$
$$\mathcal{B} : \text{Bexp} \to (\text{State} \to \text{Boolean})$$
$$\mathcal{S}_{ds} : \text{Stm} \to (\text{State} \hookrightarrow \text{State})$$

- Minden szintaktikus kategóriához tartozik egy szemantikus tartomány és egy szemantikus függvény, amely leképezi a szintaxist a szemantikára
  - A jelentést a denotáció, a hozzárendelt szemantikus érték adja
  - Általában halmazok, relációk (Integer, $\text{State} \to \text{Boolean}$)
  - Nekünk most érdekes: $\text{State} \hookrightarrow \text{State}$

Kifejezések

$$\mathcal{A}[\![n]\!]s = \mathcal{N}[\![n]\!] \quad \in \text{Integer}$$
$$\mathcal{A}[\![x]\!]s = s[x] \quad \in \text{Integer} \quad (x \in \text{Var}, s \in \text{State} = \text{Var} \to \mathbb{Z})$$
$$\mathcal{A}[\![a_1 + a_2]\!]s = \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \ldots$$
$$\vdots$$

Utasítások

$$\mathcal{S}_{ds}[\![\textbf{skip}]\!] = id_{\text{State}}$$
$$\mathcal{S}_{ds}[\![x := a]\!]s = s[x \mapsto \mathcal{A}[\![a]\!]s]$$
$$\mathcal{S}_{ds}[\![S_1; S_2]\!] = \mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!]$$
$$\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])$$
$$\vdots$$

A ciklus leíró szemantikája

Ahogyan az előző előadáson említettük, logikusnak tűnhet a ciklus szemantikáját az elágazás, szekvencia és skip szemantikájára visszavezetni (ciklus kifejtés, unfolding), mint ahogyan a small-step szemantikában tettük.

$$\begin{aligned}\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) \\ g &= cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})\end{aligned}$$

Ez azonban egy olyan definíció, amelyben a ciklus szemantikájának definiáláshoz felhasználjuk a ciklus szemantikáját, tehát nem kompozicionális a szabály.

(Vegyük észre, hogy a fenti rekurzív formulának általában több megoldása is lehet.)

A ciklus leíró szemantikája

$$\begin{aligned}\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) \\ g &= cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})\end{aligned}$$

Vizsgáljuk meg a következő utasítást:

$$\textbf{while } \neg x = 0 \textbf{ do } x := x - 1$$

Látható, hogy több megoldás létezik, g több értéket is felvehet. Pl.:

$$g\ s = \begin{cases} s[x \mapsto 0] & \text{ha} \quad x \geq 0 \\ s' & \text{egyébként} \end{cases} \qquad (\text{minden } s' \text{ állapotra})$$

$$g\ s = \begin{cases} s[x \mapsto 0] & \text{ha} \quad x \geq 0 \\ undef & \text{egyébként} \end{cases}$$

A ciklus leíró szemantikája

$$\begin{aligned} \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) \\ g &= cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) \end{aligned}$$

A formula jobb oldalát (a *kontextust*) egy magasabb rendű függvénnyé (funkcionállá) alakíthatjuk, amely a g-től függ:
$F : (\mathrm{State} \hookrightarrow \mathrm{State}) \to (\mathrm{State} \hookrightarrow \mathrm{State})$

$$F\ g = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) = g$$

Így már jól látszik, hogy valójában a formula megoldásait megkapjuk F fixpontjainak kiszámításával.

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

(Minden lehetséges ciklushoz megalkotható a fent taglalt funkcionál.)

A funkcionál fixpontjai

- Minden ciklusnak megadható a leíró szemantikája?
- A *FIX* minden lehetséges *F* funkcionálra definiált?
- Minden ciklus esetén lesz legalább egy fixpontja a funkcionálnak?
- Mi történik, ha több fixpontja van? Melyik lesz a ciklus szemantikája?
- Hogyan lehet kiszámítani a *FIX* függvényt?

Megmutatjuk, hogy milyen ciklusnak definiált a leíró szemantikája és mindig ki is tudjuk számítani (a funkcionál legkisebb fixpontjaként).

Az utasítások szemantikus tartománya

Az utasítás kategóriájához rendelt szemantikus domain egy részbenrendezett matematikai struktúra:

$$(\mathrm{State} \hookrightarrow \mathrm{State}, \sqsubseteq)$$

ahol a $\sqsubseteq$ rendezés a parciális függvényeket ($\mathrm{State} \hookrightarrow \mathrm{State}$) a következőképp rendezi (g_1 kevésbé definiált, mint g_2):

$$g_1 \sqsubseteq g_2 \qquad \Longleftrightarrow \qquad g_1 s = s' \Rightarrow g_2 s = s'$$

Továbbá a domainnek definiáljuk a fenti rendezés szerinti legkisebb (minimális) elemét ($\bot$), amely sehol sem definiált (nem tartalmaz információt):

$$\bot \sqsubseteq g \quad \text{minden} \quad g \in \mathrm{State} \hookrightarrow \mathrm{State} \text{ függvényre}$$

Az utasítások szemantikus tartománya

A $(\mathrm{State} \hookrightarrow \mathrm{State}, \sqsubseteq)$ struktúra egy részbenrendezett halmaz:

$g_1 \sqsubseteq g_1$ (reflexív)

$g_1 \sqsubseteq g_2$ és $g_2 \sqsubseteq g_3$ implikálja $g_1 \sqsubseteq g_3$ (tranzitív)

$g_1 \sqsubseteq g_2$ és $g_2 \sqsubseteq g_1$ implikálja $g_1 = g_2$ (antiszimmetrikus)

Továbbá $\bot$ formálisan a következőképp definiálható:

$\mathit{graph}(\bot) = \{\}$ azaz

$\bot s = \mathit{undef}$ minden $s \in \mathrm{State}$ állapotra

A fixpont kiszámítása

A ciklus informális szemantikája szerint a ciklusmag hatását addig ismételjük, amíg a feltétel hamissá nem válik.

A leíró szemantikában a ciklus szemantikáját az F funkcionál ismételt alkalmazásával közelítjük, majd az F legkisebb fixpontjában megkapjuk a szemantika legpontosabb közelítését.
Mivel $\bot$ a legkisebb elem, ezért $\bot \sqsubseteq F(\bot)$, továbbá mivel F monoton, F ismételt alkalmazásaival egy láncot kapunk:

$$\bot \sqsubseteq F(\bot) \sqsubseteq F(F(\bot)) \sqsubseteq F(F(F(\bot))) \sqsubseteq \ldots$$

Ezen sorozat ($\mathrm{State} \hookrightarrow \mathrm{State}$ függvények növekvő lánca) egyre pontosabb közelítését adja a ciklus szemantikának.

A lánc legkisebb felső korlátja megadja a ciklus szemantikáját, amely egyenlő az F funkcionál legkisebb fixpontjával (Kleene és Tarski fixponttétele).

A fixpont kiszámítása

- Kleene fixponttétele szerint ha F folytonos és létezik az előző lánc, valamint felülről korlátos, akkor a legkisebb felső korlát megadja az F legkisebb fixpontját:

$$FIX\ F = \sqcup\{F^n(\bot) \mid n \geq 0\}$$

- Informálisan: $FIX\ F$ a "legkisebb" $\mathrm{State} \hookrightarrow \mathrm{State}$ típusú függvény, amely minden információt tartalmaz, amely az $F^n(\bot)$ láncban előáll. Azaz,

$$\bot \sqsubseteq F(\bot) \sqsubseteq F(F(\bot)) \sqsubseteq \ \ldots\ \sqsubseteq FIX\ F$$

- Megjegyzés: a legkisebb fixpont egyértelmű.

- Tehát a legkisebb fixpont előállítható a következő formulával:

$$FIX\ F = \sqcup\{F^n(\bot) \mid n \geq 0\}$$

- Ehhez azonban tudnunk kell, hogy a láncnak létezik-e legkisebb felső korlátja.

- A $(\mathrm{State} \hookrightarrow \mathrm{State}, \sqsubseteq)$ struktúra egy láncteljes részbenrendezett halmaz (chain-complete partially ordered set, ccpo), azaz minden láncnak van egy egyértelmű legkisebb felső korlátja:

  Legyen Y egy $\mathrm{State} \hookrightarrow \mathrm{State}$ lánc.

$$graph(\sqcup Y) = \cup\{graph(g) \mid g \in Y\}$$

Az F folytonossága

- Tehát a legkisebb fixpont előállítható a következő formulával:

$$FIX\ F = \sqcup\{F^n(\bot) \mid n \geq 0\}$$

- Azt már tudjuk, hogy ez a felső korlát létezik, de ahhoz, hogy megegyezzen egy fixponttal ($F\ \left(FIX\ F\right) = FIX\ F$), az F funkcionálnak folytonosnak kell lennie.

- Ha F folytonos, akkor megtartja a láncok legkisebb felső korlátját, tehát:

$$\sqcup\left\{F(g) \mid g \in Y\right\} = F(\sqcup Y)$$

$$\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] = \text{FIX } F$$
$$\text{ahol } F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\text{State}})$$

- Az F felbontható két másik függvény kompozíciójára:

$$\begin{aligned} F\ (g) &= cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\text{State}}) \\ &= (F_2 \circ F_1)(g) \quad \text{ahol} \\ F_1(g) &= g \circ \mathcal{S}_{ds}[\![S]\!] \\ F_2(g) &= cond(\mathcal{B}[\![b]\!], g, id_{\text{State}}) \end{aligned}$$

- Ha be kell látnunk, hogy F folytonos, elég belátni, hogy F_1 és F_2 folytonosak, mert két folytonos függvény kompozíciója is folytonos lesz.
- (monotonitás + legkisebb felső korlát megtartása => folytonosság)

A While nyelv leíró szemantikája

- Ezzel teljesen áttekintettük a magnyelv (While) leíró szemantikáját
- Formálisan definiáltuk a $\mathcal{S}_{ds} : \mathrm{Stm} \to (\mathrm{State} \to \mathrm{State})$ függvényben
- A ciklusnak kompozicionálisan definiáltuk a jelentését

- A denotációs szemantika ($\mathcal{S}_{ds}$) totális, tehát minden lehetséges utasításnak megadja a jelentését (strukturális indukcióval belátható)

Rekurzív definíciók

- A magnyelv csak ciklusokkal fejez ki ismétlődést, de a rekurzív függvények esetében is szükség lesz hasonló számításokra
- Azok a módszerek, amiket a ciklus szemantikájának definiálásakor felhasználtunk, alkalmasak a rekurzív függvények szemantikájának leírására is
- Valójában most is egy rekurzív függvénynek adtuk meg a szemantikáját, hiszen a ciklus kifejtésével egy olyan rekurzív formulát kaptunk, amelyet rekurzív függvénydefiníciók jelentésének meghatározásakor kapunk

Show me the code!

λ

A While nyelv végrehajtható denotációs szemantikája elérhető a kurzus anyagai között. A leíró jellegű szemantikát Haskellben definiáltuk.

További nyelvi elemek szemantikája

Horpácsi Dániel
daniel-h@elte.hu

Programozási nyelvi elemek szemantikája

Abort, nemdeterminisztikusság, konkurencia

Előszó

- Az előző előadásokban definiáltuk az alapvető imperatív konstrukciók operációs szemantikáját

- A valódi programozási nyelvek ennél sokkal komplexebbek
- Rengeteg nyelvi elemet tartalmaznak, amelyek a programozó által használt absztrakciókat implementálják
- Ezeknek köszönhető, hogy tömören, kényelmesen tudunk kifejező, megbízható programokat írni

- Gondoljuk át, melyik nyelvben hogyan írhatunk
  - Elágazásokat és ciklusokat
  - Deklarációkat, blokkokat, alprogramokat
  - Terminálást, kivételeket, párhuzamosítást
  - Rekurzív és névtelen függvényeket
  - Lusta vagy mohó módon kiszámítható kifejezéseket
  - Adattípusokat és típuskonverziókat

Abort

- Egy program végrehajtása általában akkor ér véget, ha az összes utasítását végrehajtottuk
- Néha szükség van azonban arra, hogy a programot “abnormálisan” leállítsuk
- Bevezetünk erre egy nyelvi elemet, az abortálást

- Érdemes megjegyezni, hogy az abortálás nem ugyanaz, mint a **skip** vagy a végtelen ciklus, nem tudjuk velük kifejezni

- A C++ például az *exit(int)* és *abort()* függvényekkel implementálja
  - Persze ezek bonyolultabb szemantikával bírnak
  - Kezelik a tárral és memóriával kapcsolatos kérdéseket is
- Java-ban *System.exit(int)*
  - Ez minden végrehajtási szálat terminál
  - És a teljes virtuális gép leállítást levezényli

A kiterjesztett nyelv absztrakt szintaxisa

Szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változók)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S \mid \\
 & & \textbf{abort}
\end{array}
$$

Az abortálás operációs szemantikája

- Trükk: nem változtatunk semmit a szemantikán, nem írunk fel következtetési szabályt az abortra
- Ennek köszönhetően mindkét operációs szemantika azt fogja mutatni, hogy a végrehajtás ott megakad
- Azzal, hogy nem csinálunk semmit, jó szemantikát kapunk!

Vajon mindig jól működik?

-

$\langle \textbf{if } x < 0 \textbf{ then abort else } x = x - 1, \quad [x \mapsto 10] \rangle$

-

$\langle \textbf{if } x < 0 \textbf{ then abort else } x = x - 1, \quad [x \mapsto -10] \rangle$

-

$\langle \textbf{if } x < 0 \textbf{ then skip else } x = x - 1 \textbf{ ; abort}, \quad [x \mapsto 10] \rangle$

-

$\langle \textbf{if } x < 0 \textbf{ then skip else } x = x - 1 \textbf{ ; abort}, \quad [x \mapsto -10] \rangle$

-

$\langle \textbf{abort ; if } x < 0 \textbf{ then skip else } x = x - 1, \quad [x \mapsto 10] \rangle$

-

$\langle \textbf{abort ; if } x < 0 \textbf{ then skip else } x = x - 1, \quad [x \mapsto -10] \rangle$

Az abort strukturális szemantikája

- A small-step szemantikában beragadt konfigurációk jelzik a program elakadását, hibás megállását
  - A konfiguráció beragadt, vagy zsákutca, ha nem tudunk átmenetet levezetni hozzá
  - Formálisan, $\langle S, s \rangle$ beragadt, ha nincs olyan c konfiguráció, amelyre $\langle S, s \rangle \Rightarrow c$

- Nem vettünk fel olyan szabályt, amelynek a konklúziója megadná valamely **abort** utasítást tartalmazó konfiguráció átmenetét
- Tehát azok a konfigurációk, amelyek az **abort** utasítás végrehajtását írják elő a következő lépésben, beragadnak
- Továbbá a valahol **abort**ot tartalmazó, és azt végrehajtó programok véges levezetési láncokat eredményeznek, amelyek zsákutca konfigurációban érnek véget

Természetes szemantika

- A big-step szemantikában nincsenek beragadó konfigurációk
- A programok vagy sikeresen terminálnak, vagy nem terminálnak, sikertelen terminációt nem értelmezünk
- Az **abort** utasítást tartalmazó programoknak nem számítható ki a szemantikája, “nem futtathatóak”

- Érdekesség: a fentiek következménye, hogy a természetes szemantikában az **abort** és a **while true do skip** utasítások szemantikusan ekvivalensek
- Mivel nem tud különbséget tenni az abnormális terminálás és a divergálás között (hacsak nem vezetünk be egy extremális *hiba* konfigurációt)

Nemdeterminisztikusság

- “A nemdeterminisztikus algoritmus egy olyan algoritmus, amely különböző futtatások esetén különböző viselkedést mutathat”.
- Mi most egy egyszerű konstrukcióval foglalkozunk: nemdeterminisztikus választás két utasítás között
- Tehát bármelyik utasítás végrehajtódhat, de csakis az egyik

- Prolog, Promela; Curry, Church, Amb, java2k (ezoterikus nyelvek)
- Az elterjedt programozási nyelvek általában nem implementálnak nemdeterminisztikus választást...
- ...viszont nemdeterminisztikus viselkedést eredményez a konkurencia, mert ekkor az ütemezésen múlik, milyen sorrendben hajtódnak végre az utasítások

A kiterjesztett nyelv szintaxisa

Szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változók)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S \mid \\
 & & S_1 \textbf{ or } S_2
\end{array}
$$

Nemdeterminisztikusság: példa

Az informális szemantikánk szerint ennek az utasításnak:

$$x := 1 \textbf{ or } \big(x := 3 \ ; \ x := x - 1\big)$$

két különböző eredménye lehet: olyan állapotban is véget érhet, amelyben x értéke 1, illetve olyanban is, ahol x értéke 2.

Még érdekesebb a következő utasítás szemantikája:

$$x := 1 \textbf{ or } \big(\textbf{while true do skip}\big)$$

mivel azt várjuk tőle, hogy vagy az $x = 1$ állapotban terminál, vagy pedig egyáltalán nem terminál.

Hogyan tudjuk ezt elérni a formális szemantika definícióban?

Nemdeterminisztikus választás

$$\frac{\langle S_1, s\rangle \to s'}{\langle S_1 \textbf{ or } S_2, s\rangle \to s'}$$

$$\frac{\langle S_2, s\rangle \to s'}{\langle S_1 \textbf{ or } S_2, s\rangle \to s'}$$

Ezzel levezethető mindkettő:

$$\langle x := 1 \textbf{ or } (x := 3 \ ; \ x := x - 1), s\rangle \to s[x \mapsto 1]$$

és

$$\langle x := 1 \textbf{ or } (x := 3 \ ; \ x := x - 1), s\rangle \to s[x \mapsto 2]$$

(minden s állapot esetén).

Mit mondhatunk az $x := 1$ **or** $\big($**while true do skip**$\big)$ utasítás big-step szemantikájáról?

Az biztosan igaz, hogy

$$\big\langle x := 1 \textbf{ or } \big(\textbf{while true do skip}\big), s\big\rangle \to s\big[x \mapsto 1\big]$$

De a második alternatívával nem tudunk átmenetet levezetni.

Ez azért van, mert a végtelen ciklus esetében nem tudunk levezetési fát építeni és így a nemdet. választás második szabályát nem tudjuk alkalmazni. A big-step szemantika nem fogja a második alternatívát választani: elkerüli a végtelen ciklust és az abnormális terminálást.

Mit mondhatunk az $x := 1$ **or** $\big($**while true do skip**$\big)$ utasítás big-step szemantikájáról?

Az biztosan igaz, hogy

$$\big\langle x := 1 \textbf{ or } \big(\textbf{while true do skip}\big), s\big\rangle \to s\big[x \mapsto 1\big]$$

De a második alternatívával nem tudunk átmenetet levezetni.

Ez azért van, mert a végtelen ciklus esetében nem tudunk levezetési fát építeni és így a nemdet. választás második szabályát nem tudjuk alkalmazni. A big-step szemantika nem fogja a második alternatívát választani: elkerüli a végtelen ciklust és az abnormális terminálást.

Nemdeterminisztikus választás

$$\frac{}{\langle S_1 \textbf{ or } S_2, s\rangle \Rightarrow \langle S_1, s\rangle}$$

$$\frac{}{\langle S_1 \textbf{ or } S_2, s\rangle \Rightarrow \langle S_2, s\rangle}$$

Ahogy az informális szemantika alapján várjuk, levezethető mindkét eredmény:

$$\langle x := 1 \textbf{ or } (x := 3 \,;\; x := x - 1), s\rangle \Rightarrow^* s[x \mapsto 1]$$

és

$$\langle x := 1 \textbf{ or } (x := 3 \,;\; x := x - 1), s\rangle \Rightarrow^* s[x \mapsto 2]$$

(bármely s állapotra).

Ha az $x := 1$ **or** $(\textbf{while true do skip})$ utasítás small-step szemantikát vizsgáljuk, a következőt kapjuk.

Előáll egy véges levezetési lánc:

$$\langle x := 1 \textbf{ or } (\textbf{while true do skip}), s\rangle \Rightarrow^* s[x \mapsto 1]$$

és egy végtelen levezetési lánc is:

$$\langle x := 1 \textbf{ or } (\textbf{while true do skip}), s\rangle \Rightarrow^* \infty$$

Tehát a small-step szemantika a “rossz” úton is el tudja végezni a végrehajtást. Ez a formális szemantika jobban egybevág az informális szemantikával.

- Bevezetünk parbegin-parend jellegű nyelvi elemet, amely párhuzamos végrehajtást tesz lehetővé
- Mi két utasítás konkurens végrehajtását fogjuk lehetővé tenni
- A szemantika összefésüléses

$$x := 1 \textbf{ par } \left(x := 3 \ ; \ x := x - 1\right)$$

A fenti utasítás x-hez három különböző értéket rendelhet: 0, 1 vagy 2.

A kiterjesztett nyelv szintaxisa

Szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változók)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S \mid \\
 & & S_1 \textbf{ par } S_2
\end{array}
$$

$$\frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1 \textbf{ par } S_2, s\rangle \Rightarrow \langle S_1' \textbf{ par } S_2, s'\rangle}$$

$$\frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1 \textbf{ par } S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

$$\frac{\langle S_2, s\rangle \Rightarrow \langle S_2', s'\rangle}{\langle S_1 \textbf{ par } S_2, s\rangle \Rightarrow \langle S_1 \textbf{ par } S_2', s'\rangle}$$

$$\frac{\langle S_2, s\rangle \Rightarrow s'}{\langle S_1 \textbf{ par } S_2, s\rangle \Rightarrow \langle S_1, s'\rangle}$$

Példa levezetések

$$\begin{aligned}
&\langle\, x := 1 \;\mathbf{par}\; \big(x := 3 \;;\; x := x - 1\big), && s \,\rangle \Rightarrow \\
&\langle\, x := 3 \;;\; x := x - 1, && s[x \mapsto 1] \,\rangle \Rightarrow \\
&\langle\, x := x - 1, && s[x \mapsto 3] \,\rangle \Rightarrow \\
&\quad s[x \mapsto 2]
\end{aligned}$$

$$\begin{aligned}
&\langle\, x := 1 \;\mathbf{par}\; \big(x := 3 \;;\; x := x - 1\big), && s \,\rangle \Rightarrow \\
&\langle\, x := 1 \;\mathbf{par}\; x := x - 1, && s[x \mapsto 3] \,\rangle \Rightarrow \\
&\langle\, x := x - 1, && s[x \mapsto 1] \,\rangle \Rightarrow \\
&\quad s[x \mapsto 0]
\end{aligned}$$

$$\begin{aligned}
&\langle\, x := 1 \;\mathbf{par}\; \big(x := 3 \;;\; x := x - 1\big), && s \,\rangle \Rightarrow \\
&\langle\, x := 1 \;\mathbf{par}\; x := x - 1, && s[x \mapsto 3] \,\rangle \Rightarrow \\
&\langle\, x := 1, && s[x \mapsto 2] \,\rangle \Rightarrow \\
&\quad s[x \mapsto 1]
\end{aligned}$$

Természetes szemantika

- A természetes, big-step szemantikában a részkifejezések mindig teljesen kiértékelésre kerülnek egy átmenettel, így összefésülést nem tudunk leírni
- Következésképp a párhuzamos konstrukciónknak nem tudunk big-step szemantikát definiálni, vagy legalábbis összefésüléssel nem

- Továbbá a big-step szemantikában a nemdeterminisztikus választás elkerüli a végtelen ciklusokat és az abort szemantikusan ekvivalens a végtelen ciklussal
- Ezek a példák jól mutatják, hogy a természetes szemantika jóval absztraktabb, mint a strukturális

$\mathbb{K}$

A foreach-jellegű ciklus végrehajtható szemantikája elérhető a kurzus anyagai között. A small-step stílusú operációs szemantikát a $\mathbb{K}$ keretrendszerben definiáltuk, kipróbálható a 3.5 és a 3.6 verziókkal.

λ

A switch-case jellegű elágazás végrehajtható szemantikája elérhető a kurzus anyagai között. A leíró szemantikát Haskellben adtuk meg.

A kivételkezelés szemantikája

Horpácsi Dániel
daniel-h@elte.hu

Folytatásos denotációs szemantika:

A kivételkezelés modellezése

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- Operációs (műveleti)
  - Strukturális

    Minden lépés modellezése
  - Természetes

    A kezdő- és végállapotok közötti reláció felállítása
- **Denotációs (leíró)**

  A jelentést matematikai objektumok hozzárendelésével adja meg

A haladás iránya

- Az előző előadásokon definiáltuk a While nyelv (direkt) denotációs szemantikáját
- Megkülönböztetett figyelemmel a rekurzív szerkezetekre, a fixpont-elméletre

- A direkt szemantika minden utasításhoz hozzárendelt egy függvényt, amely karakterizálja az utasítás jelentését, végrehajtásának hatását
- A magnyelvhez megfelelő, de bizonyos programozási nyelvi elemek jelentésének kifejezésére nem alkalmas, nem elég kifejező

- **Milyen matematikai eszközzel definiálhatjuk a kivételkezelés és ugró utasítások leíró szemantikáját?**

A kivételkezeléssel kiegészített While

Szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változók)
e	$\in$	Exception	(kivételek)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)

Produkciós szabályok

$$
\begin{array}{lll}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \mid \textbf{while } b \textbf{ do } S \mid \\
 & & \textbf{try } S_1 \textbf{ catch } e : S_2 \mid \textbf{throw } e
\end{array}
$$

Informális szemantika, motiváció

(Direkt kontra folytatásos szemantika)

- $\left(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2\right) \; ; \; S_3$
- $\left(\textbf{try } S_1 \textbf{ catch } e : S_2\right) \; ; \; S_3$
- $\left(\textbf{try throw } e \textbf{ catch } e : S_2\right) \; ; \; S_3$
- $\left(\textbf{try throw } e \; ; \; S_1 \textbf{ catch } e : S_2\right) \; ; \; S_3$
- $\left(\textbf{try } S_1 \; ; \; \textbf{throw } e \textbf{ catch } e : S_2\right) \; ; \; S_3$
- $\left(\textbf{try throw } e_1 \; ; \; S_1 \textbf{ catch } e_2 : S_2\right) \; ; \; S_3$
- $\left(\textbf{try } \left(\textbf{if } b \textbf{ then } S_1 \textbf{ else throw } e\right) \textbf{ catch } e : S_2\right) \; ; \; S_3$

(Az S_n elemek utasítások a *magnyelvben*)

(Vezérlésfolyam)

```
try
  while true do
    if x > 0 then
      throw exit
    else
      x := x + 1
catch exit : y := 1
```

Tehát egy végtelen ciklus (*while true do*...) is lehet véges?

Informális szemantika

- A *try...catch* szerkezet kivételkezelő blokkot vezet be
- Az el nem kapott kivételek továbbterjednek külsőbb blokkok felé

- A *throw* utasítás használható kivétel kiváltására
- Megszakítja a blokk futását, a hátralévő utasítások nem kerülnek végrehajtásra
- A vezérlés átadódik egy megfelelő kivételkezelő kódra
- A kivételkezelő blokk végrehajtása után “normálisan” folytatódik a kiértékelés, a kivételkezelt blokkot követő programrészlettel

A folytatásos szemantika alapötlete

- A kivételek (és általában az ugró utasítások) hatásának leírásához más denotációs szemantikus formalizmusra lesz szükségünk
- Amikor kiváltódik egy kivétel, tudnunk kell, hogyan folytatódik a programvégrehajtás a kivételkezelő blokk után (hogyan folytatódna a kivételkezelt blokk után)

$$\mathrm{Cont} = \mathrm{State} \hookrightarrow \mathrm{State}$$

- A folytatás megadja a program hátralévő része végrehajtásának hatását

$$S'_{cs} : \mathrm{Stm} \to (\mathrm{Cont} \to \mathrm{Cont})$$

- Először a magnyelvhez definiálunk folytatásos leíró szemantikát

$$\mathcal{S}'_{cs} : \mathrm{Stm} \to (\mathrm{Cont} \to \mathrm{Cont})$$

Folytatásos szemantika

$$\begin{aligned}
\mathcal{S}'_{cs}[\![\textbf{skip}]\!] &= id_{\mathrm{Cont}} \\
\mathcal{S}'_{cs}[\![x := a]\!]c\ s &= c(s[x \mapsto \mathcal{A}[\![a]\!]s]) \\
\mathcal{S}'_{cs}[\![S_1; S_2]\!] &= \mathcal{S}'_{cs}[\![S_1]\!] \circ \mathcal{S}'_{cs}[\![S_2]\!] \\
\mathcal{S}'_{cs}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]c &= cond(\mathcal{B}[\![b]\!], \mathcal{S}'_{cs}[\![S_1]\!]c, \mathcal{S}'_{cs}[\![S_2]\!]c) \\
\mathcal{S}'_{cs}[\![\textbf{while } b \textbf{ do } S]\!] &= \mathrm{FIX}\ G \\
\mathrm{ahol}\ (G\ g)\ c &= cond(\mathcal{B}[\![b]\!], \mathcal{S}'_{cs}[\![S]\!](g\ c), c)
\end{aligned}$$

A While folytatásos szemantikája

Szignatúra:

$\mathrm{Stm} \to (\mathrm{Cont} \to (\mathrm{State} \hookrightarrow \mathrm{State}))$ vs $\mathrm{Stm} \to (\mathrm{State} \hookrightarrow \mathrm{State})$

Skip:

id_{Cont} vs id_{State}

Szekvencia:

$\mathcal{S}'_{cs}[\![S_1]\!] \circ \mathcal{S}'_{cs}[\![S_2]\!]$ vs $\mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!]$

Ciklus:

$G\ g\ c = cond(\mathcal{B}[\![b]\!], \mathcal{S}'_{cs}[\![S]\!](g\ c), c)$ vs $F\ g = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$

A denotációs szemantikák kapcsolata

- A direkt szemantikában azt adjuk meg, mi az adott konstrukció végrehajtásának hatása
- A folytatásos szemantikában “visszafelé” haladva állítjuk elő a konstrukciók jelentését: a folytatás függvényében adjuk meg, mi az utasítás jelentése, ha a megadott folytatás követi a végrehajtásban

- Mégis, a két megközelítés között tiszta összefüggés írható fel
- Bizonyítható, hogy

$$\mathcal{S}'_{cs}[\![S]\!]c = c \circ \mathcal{S}_{ds}[\![S]\!] \quad \text{minden } S \text{ utasításra és } c \text{ folytatásra}$$

- Következésképp

$$\mathcal{S}'_{cs}[\![S]\!]\ id_{\mathrm{State}} = \mathcal{S}_{ds}[\![S]\!]$$

Folytatásos szemantika

$$\mathcal{S}'_{cs}[\![\textbf{try } S_1 \textbf{ catch } e : S_2]\!] = ?$$

$$\mathcal{S}'_{cs}[\![\textbf{throw } e]\!] = ?$$

- A kivételkezelt blokk tetszőlegesen komplex lehet
- A *throw* utasítás működése/hatása függ attól, hogy a környezetében (tetszőlegesen távol) milyen kivételkezelő blokkokat definiáltunk
- A külső környezetből minket most a definiált kivételkezelők (*catch* szekciók) érdekelnek

A ’throw’ viselkedése csak a környezet jelentésének ismeretében adható meg

A kivételkörnyezet

- A kivételek “jelentését” kivételkörnyezetben tartjuk nyilván:

$$\mathrm{Env}_E = \mathrm{Exception} \to \mathrm{Cont}$$

- Definiálja, mi történik a kivétel kiváltása esetén (mi a hatása a hátralévő programrésznek)
- Ez természetesen függ a környezettől, a kivételkezelők definiálják ezt a környezetet

A kivételkörnyezettől függ a jelentés, így paraméterezzük vele a leíró szemantikus függvényt:

$$S_{cs} : \mathrm{Stm} \to \mathrm{Env}_E \to (\mathrm{Cont} \to \mathrm{Cont})$$

$$\mathcal{S}_{cs} : \mathrm{Stm} \to \mathrm{Env}_E \to (\mathrm{Cont} \to \mathrm{Cont})$$

A folytatásos szemantikai szabályai

$$\begin{aligned}
\mathcal{S}_{cs}[\![\textbf{skip}]\!]\ env_E &= id_{\mathrm{Cont}} \\
\mathcal{S}_{cs}[\![x := a]\!]\ env_E\ c\ s &= c(s[x \mapsto \mathcal{A}[\![a]\!]s]) \\
\mathcal{S}_{cs}[\![S_1; S_2]\!]\ env_E &= (\mathcal{S}_{cs}[\![S_1]\!]\ env_E\ ) \circ (\mathcal{S}_{cs}[\![S_2]\!]\ env_E\ ) \\
\mathcal{S}_{cs}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]\ env_E\ c &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{cs}[\![S_1]\!]\ env_E\ c, \mathcal{S}_{cs}[\![S_2]\!]\ env_E\ c) \\
\mathcal{S}_{cs}[\![\textbf{while } b \textbf{ do } S]\!]\ env_E &= \mathrm{FIX}\ G \\
\text{ahol } (G\ g)\ c &= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{cs}[\![S]\!]\ env_E\ (g\ c), c) \\
\mathcal{S}_{cs}[\![\textbf{try } S_1 \textbf{ catch } e : S_2]\!]\ env_E\ c &= \mathcal{S}_{cs}[\![S_1]\!](\ env_E\ [e \mapsto \mathcal{S}_{cs}[\![S_2]\!]\ env_E\ c])\ c \\
\mathcal{S}_{cs}[\![\textbf{throw } e]\!]\ env_E\ c &= \ \ env_E\ e
\end{aligned}$$

Tegyük fel, hogy van egy kezdeti env_E kivételkörnyezetünk.

$$\mathcal{S}_{cs}[\![\textbf{try while true do}\ \textbf{if}\ x > 0\ \textbf{then throw}\ exit\ \textbf{else}\ x := x + 1\ \textbf{catch}\ exit : y := 1]\!]\ env_E\ id_{\mathrm{State}} =$$
$$\mathcal{S}_{cs}[\![\textbf{while true do} \ldots]\!]\ env_E[exit \mapsto c_{exit}]\ id_{\mathrm{State}} = (FIX\ G)\ id_{\mathrm{State}}$$

$$\begin{aligned} G\ g\ c\ s &= cond(\mathcal{B}[\![\textbf{true}]\!], \\ &\qquad cond(\mathcal{B}[\![x > 0]\!], c_{exit}, \mathcal{S}_{cs}[\![x := x + 1]\!]\ env_E[exit \mapsto c_{exit}](g\ c)), \\ &\qquad c)\ s = \\ &= cond(\mathcal{B}[\![x > 0]\!], c_{exit}, \mathcal{S}_{cs}[\![x := x + 1]\!]\ env_E[exit \mapsto c_{exit}](g\ c))\ s = \\ &= \begin{cases} c_{exit}\ s & \text{if} \quad s[x] > 0 \\ (g\ c)(s[x \mapsto s[x] + 1]) & \text{if} \quad s[x] \leq 0 \end{cases} \end{aligned}$$

$$c_{exit}\ s = id_{\mathrm{State}} s[y \mapsto 1] = s[y \mapsto 1]$$

$$(FIX\ G)\ id_{\mathrm{State}}\ s = \begin{cases} s[y \mapsto 1] & \text{if} \quad s[x] > 0 \\ s[x \mapsto 1][y \mapsto 1] & \text{if} \quad s[x] \leq 0 \end{cases}$$

- Az kivételkezelés operációs szemantikáját csak vázlatosan adjuk meg
- Hogyan lehetne big-step szemantikát definiálni a kivételkezeléshez?

Throw

$$\frac{?}{\langle \textbf{throw}\ e, s\rangle \to ?}\ ?$$

Try-catch

$$\frac{?}{\langle \textbf{try}\ S_1\ \textbf{catch}\ e : S_2, s\rangle \to ?}\ ?$$

- Kiegészítve a lehetséges konfigurációkat kivétel-állapot párosokkal:

Throw

$$\frac{}{\langle \textbf{throw}\ e, s\rangle \to \langle e, s\rangle}$$

Try-catch

$$\frac{\langle S_1, s\rangle \to s'}{\langle \textbf{try}\ S_1\ \textbf{catch}\ e : S_2, s\rangle \to s'}$$

$$\frac{\langle S_1, s\rangle \to \langle e, s'\rangle \quad \langle S_2, s'\rangle \to c}{\langle \textbf{try}\ S_1\ \textbf{catch}\ e : S_2, s\rangle \to c}$$

Szekvencia szabály (módosított)

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

Szekvencia szabály (hozzáadott)

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

A szemantikadefiníció befejezéséhez az összes utasítás esetében delegálni kell a kivételeket (az elágazás és ciklus szabályát is igazítani kell).

λ

A While nyelv és a fent bemutatott kivételkezelés végrehajtható denotációs szemantikája elérhető a kurzus anyagai között. A leíró jellegű szemantikát Haskellben definiáltuk.

Blokkok és alprogramok

Horpácsi Dániel
daniel-h@elte.hu

Az előadás témája

Blokkok, hatókör, láthatóság, alprogramok

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- Operációs (műveleti)
  - Strukturális

    Minden lépés modellezése
  - Természetes

    A kezdő- és végállapotok közötti reláció felállítása
- **Denotációs (leíró)**

  A jelentést matematikai objektumok hozzárendelésével adja meg

A következő lépés

- A magnyelvnek megadtuk az operációs és denotációs szemantikáját is
- És további nyelvi elemeknek is megmutattuk a formális szemantikáját:
  - Abort, nemdeterminisztikus választás, összefésülés, kivételkezelés

- Habár az **if** és **while** utasítások is blokkokat képeznek (magukba foglalnak újabb utasításokat), blokkra lokális változókat nem lehetett definiálni
- Most megnézzük, hogyan lehet leírni a blokkszerkezetet, illetve deklarációkat, névtereket
- Hatókör, láthatóság, élettartam? Alprogramok és azok hívása?

Kapcsolódó fogalmak

Gondoljuk át a következő fogalmakat:

- Utasítás
- Utasításblokk
- Változó
- Alprogram
- Deklaráció, definíció
- Névkötés
- Névtér
- Statikus/dinamikus scope (hatókör, láthatóság)

Szintaktikus kategóriák és metaváltozóik

n	$\in$	Num	(számliterálok)
x	$\in$	Var	(változók)
p	$\in$	Proc	(alprogramnevek)
a	$\in$	Aexp	(aritmetikai kifejezések)
b	$\in$	Bexp	(logikai kifejezések)
S	$\in$	Stm	(utasítások)
D_V	$\in$	$\mathrm{Dec_V}$	(változódefiníciók)
D_P	$\in$	$\mathrm{Dec_P}$	(alprogramdefiníciók)

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid \text{-}a \\
b & ::= & \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & \textbf{skip} \mid x := a \mid S_1; S_2 \mid \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2 \mid \textbf{while}\ b\ \textbf{do}\ S \\
 & & \mid\ \textbf{begin}\ D_V\ D_P\ S\ \textbf{end} \mid \textbf{call}\ p \\
D_V & ::= & \textbf{var}\ x := a\ ;\ D_V \mid \varepsilon \\
D_P & ::= & \textbf{proc}\ p\ \textbf{is}\ S\ ;\ D_P \mid \varepsilon
\end{array}
$$

(Névkötés)

$x := 1 \; ; \; y := x + 1$

begin var $x := 1 \; ;$ **var** $y := 1 \; ; \; y := x + 1$ **end**

```
begin var x := 1 ; var y := 1 ; var z := 1 ;
    begin var x := 2 ;
        y := x + 1 ;
        x := x + 4
    end ;
    z := x + 1
end
```

(Statikus/dinamikus hatókör)

```
begin
    var x := 1 ; var y := 1 ; var z := 1 ; proc p is x := 0 ;
    begin
        var x := 2 ;
        call p ; y := x + 1
    end ;
    z := x + 1
end
```

Finomítsuk a memóriafogalmat: location (cím)

Az eddigi memóriállapotokban a változókhoz rendeltük az értékeket:

$$\text{State} = \text{Var} \to \mathbb{Z}$$

Mostantól a blokkok (lokális változók) miatt egy név több különböző változót jelölhet. A névhez rendelt érték függ a scope-tól.

- Az értékek helyett társítsunk memóriacímeket a nevekhez:

$$\text{Env}_\text{V} = \text{Var} \to \text{Loc}$$

- És vezessük be a tár fogalmát, amely a címhez értéket rendel:

$$\text{Store} = \text{Loc} \to \mathbb{Z}$$

- Az egyszerűség kedvéért a címeket most egész számokkal reprezentáljuk:

$$\text{Loc} = \mathbb{Z}$$

Változókörnyezet

- A változónevekhez címeket rendelünk:

$$\mathrm{Env_V} = \mathrm{Var} \to \mathrm{Loc}$$

- És a tárral adjuk meg, milyen érték van az adott címen:

$$\mathrm{Store} = \mathrm{Loc} \cup \{\mathit{next}\} \to \mathbb{Z}$$

(Trükk: a Loc halmazhoz itt hozzáveszünk egy extremális elemet, a *next* szimbólumot, amelyhez nem változóértéket rendelünk majd, hanem memóriacímet.)

- A következő szabad címet a *new* függvény számítja ki:

$$\mathit{new} : \mathrm{Loc} \to \mathrm{Loc}$$

$$\mathit{new}\ n = n + 1$$

Változó értékének lekérdezése

A változó értékét mostantól a változókörnyezet (env_V) és a tár (sto) kombinációból nyert állapot (s) alapján határozzuk meg.

$$s = sto \circ env_V$$

Ezt az összefüggést fogalommá emeljük, ez lesz a változólekérdezés (lookup) függvény:

$$lookup : \mathrm{Env_V} \to \mathrm{Store} \to \mathrm{State}$$

$$lookup\ env_V\ sto = sto \circ env_V$$

ahol

$$env_V \in \mathrm{Var} \to \mathrm{Loc} \quad \text{és} \quad sto \in \mathrm{Loc} \cup \{next\} \to \mathbb{Z}$$

$$\mathcal{S}'_{ds} : \mathrm{Stm} \to \mathrm{Env_V} \to (\mathrm{Store} \hookrightarrow \mathrm{Store})$$

Címekkel bővített denotációs szemantika

$$\mathcal{S}'_{ds}[\![\textbf{skip}]\!]env_V = id_{\mathrm{Store}}$$

$$\mathcal{S}'_{ds}[\![x := a]\!]env_V\ sto = sto[l \mapsto \mathcal{A}[\![a]\!](lookup\ env_V\ sto)]$$

$$\text{ahol } l = env_V(x)$$

$$\mathcal{S}'_{ds}[\![S_1; S_2]\!]env_V = (\mathcal{S}'_{ds}[\![S_2]\!]env_V) \circ (\mathcal{S}'_{ds}[\![S_1]\!]env_V)$$

$$\mathcal{S}'_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]env_V =$$

$$cond(\mathcal{B}[\![b]\!] \circ (lookup\ env_V), \mathcal{S}'_{ds}[\![S_1]\!]env_V, \mathcal{S}'_{ds}[\![S_2]\!]env_V)$$

$$\mathcal{S}'_{ds}[\![\textbf{while } b \textbf{ do } S]\!]env_V = \mathrm{FIX}\ F$$

$$\text{ahol } F\ g = cond(\mathcal{B}[\![b]\!] \circ (lookup\ env_V), g \circ (\mathcal{S}'_{ds}[\![S]\!]env_V), id_{\mathrm{Store}})$$

$$cond : (\mathrm{Store} \to \mathrm{Boolean}) \times (\mathrm{Store} \hookrightarrow \mathrm{Store}) \times (\mathrm{Store} \hookrightarrow \mathrm{Store}) \to (\mathrm{Store} \hookrightarrow \mathrm{Store})$$

A While kibővített szemantikája

- A memóriaállapot valamivel kevésbé absztrakt modelljével lehetővé tettük scope-ok bevezetését, blokkra lokális változószimbólumok deklarációját
- Bizonyítható, hogy a fent leírt szemantika ekvivalens a korábbi denotációs szemantikával:

$$\mathcal{S}_{ds}[\![S]\!] \circ (lookup\ env_V) = (lookup\ env_V) \circ (\mathcal{S}'_{ds}[\![S]\!] env_V)$$

(minden env_V környezetre)

- Hogyan írjuk le a szemantikáját a blokkoknak és az alprogramhívásoknak?

$\mathcal{S}_{ds}[\![\textbf{begin}\ D_V\ D_P\ S\ \textbf{end}]\!] = ?$
$\mathcal{S}_{ds}[\![\textbf{call}\ p]\!] = ?$

Változódeklarációk szemantikája

- A szintaxisban definiáltunk két új kategóriát: a változódeklarációkat és az eljárásdeklarációkat
- A denotációs szemantikában megszokott módon ezekhez egy-egy szemantikus domain és szemantikus függvény fog tartozni

- Definiáljuk a változódeklaráció szemantikáját:

$$\mathcal{D}^V_{ds} : \mathrm{Dec_V} \to \mathrm{Env_V} \times \mathrm{Store} \to \mathrm{Env_V} \times \mathrm{Store}$$

$$\mathcal{D}^V_{ds}[\![\textbf{var } x := a \ ; \ D_V]\!](env_V, sto) =$$
$$\quad \mathcal{D}^V_{ds}[\![D_V]\!](env_V[x \mapsto l], sto[l \mapsto v][next \mapsto new\ l])$$
$$\qquad \text{ahol } l = sto\ next \quad \text{és} \quad v = \mathcal{A}[\![a]\!](lookup\ env_V\ sto)$$

$$\mathcal{D}^V_{ds}[\![\varepsilon]\!] = id_{\mathrm{Env_V} \times \mathrm{Store}}$$

Alprogramkörnyezet

Hasonlóan a kivételekhez, az eljárások meghívásakor (odaugráskor) tudnunk kell, mi az alprogram jelentése. Ehhez bevezetjük az alprogramkörnyezet fogalmát:

$$\mathrm{Env_P} = \mathrm{Proc} \to (\mathrm{Store} \hookrightarrow \mathrm{Store})$$

És a változódeklarációkhoz hasonlóan definiáljuk az eljárások deklarációinak szemantikáját is:

$$\mathcal{D}^P_{ds} : \mathrm{Dec_P} \to \mathrm{Env_V} \to \mathrm{Env_P} \to \mathrm{Env_P}$$

$$\mathcal{D}^P_{ds}[\![\textbf{proc } p \textbf{ is } S \ ; \ D_P]\!] env_V \ env_P = \mathcal{D}^P_{ds}[\![D_P]\!] env_V \ (env_P[p \mapsto g])$$
ahol $g = \mathcal{S}_{ds}[\![S]\!] env_V \ env_P$

$$\mathcal{D}^P_{ds}[\![\varepsilon]\!] env_V = id_{\mathrm{Env_P}}$$

A kiterjesztett nyelv leíró szemantikája

A kiegészített nyelven írt programok jelentése függ a változó- és alprogramkörnyezettől. Emiatt az előző előadáson látottakhoz hasonlóan kibővítjük a szemantikus függvényünket újabb paraméterekkel:

$$S_{ds} : \mathrm{Stm} \to \mathrm{Env_V} \to \mathrm{Env_P} \to (\mathrm{Store} \hookrightarrow \mathrm{Store})$$

A környezeteket a deklarációs részek állítják be:

Címekkel bővített denotációs szemantika

$$\mathcal{S}_{ds}[\![\textbf{begin } D_V\ D_P\ S\ \textbf{end}]\!] env_v\ env_P\ s = \mathcal{S}_{ds}[\![S]\!] env'_V\ env'_P\ s'$$
$$\text{ahol } \mathcal{D}^V_{ds}[\![D_V]\!](env_V, s) = (env'_V, s')$$
$$\text{és } \mathcal{D}^P_{ds}[\![D_P]\!] env'_V\ env_P = env'_P$$

$$S_{ds} : \mathrm{Stm} \to \mathrm{Env_V} \to \mathrm{Env_P} \to (\mathrm{Store} \hookrightarrow \mathrm{Store})$$

Címekkel bővített denotációs szemantika

$$\mathcal{S}_{ds}[\![\textbf{skip}]\!]e_v\ e_p = id_{\mathrm{Store}}$$

$$\mathcal{S}_{ds}[\![x := a]\!]e_v\ e_p\ sto = sto[l \mapsto \mathcal{A}[\![a]\!](lookup\ e_v\ sto)]$$

$$\text{ahol } l = e_v(x)$$

$$\mathcal{S}_{ds}[\![S_1; S_2]\!]e_v\ e_p = (\mathcal{S}_{ds}[\![S_2]\!]e_v\ e_p) \circ (\mathcal{S}_{ds}[\![S_1]\!]e_v\ e_p)$$

$$\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]e_v\ e_p =$$

$$cond(\mathcal{B}[\![b]\!] \circ (lookup\ env_V), \mathcal{S}_{ds}[\![S_1]\!]e_v\ e_p, \mathcal{S}_{ds}[\![S_2]\!]e_v\ e_p)$$

$$\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!]e_v\ e_p = \mathrm{FIX}\ F$$

$$\text{ahol } F\ g = cond(\mathcal{B}[\![b]\!] \circ (lookup\ e_v), g \circ (\mathcal{S}_{ds}[\![S]\!]e_v\ e_p), id_{\mathrm{Store}})$$

$$\mathcal{S}_{ds}[\![\textbf{begin } D_V\ D_P\ S \textbf{ end}]\!]e_v\ e_p\ s = \mathcal{S}_{ds}[\![S]\!]e'_v\ e'_p\ s'$$

$$\text{ahol } \mathcal{D}^V_{ds}[\![D_V]\!](e_v, s) = (e'_v, s')$$

$$\text{és } \mathcal{D}^P_{ds}[\![D_P]\!]e'_v\ e_p = e'_p$$

$$\mathcal{S}_{ds}[\![\textbf{call } p]\!]e_v\ e_p = e_p(p)$$

- Meghívhatja egy eljárás saját magát? Ha igen, akkor az eljáráskörnyezetnek már tartalmaznia kell az eljárást, amikor még nem is dolgoztuk fel a deklarációját...

$$\mathcal{D}^P_{ds}[\![\textbf{proc } p \textbf{ is } S\ ;\ D_P]\!]env_V\ env_P = \mathcal{D}^P_{ds}[\![D_P]\!]env_V\ (env_P[p \mapsto g])$$
$$\text{ahol } g = \mathcal{S}_{ds}[\![S]\!]env_V\ (env_P[p \mapsto g])$$

A ciklus szemantikájának definíciójában is egy rekurzív formulából indultunk ki!

- Megoldás: fixpont kombinátor

$$\mathcal{D}^P_{ds}[\![\textbf{proc } p \textbf{ is } S\ ;\ D_P]\!]env_V\ env_P = \mathcal{D}^P_{ds}[\![D_P]\!]env_V\ (env_P[p \mapsto FIX\ F])$$
$$\text{ahol } F\ g = \mathcal{S}_{ds}[\![S]\!]env_V\ (env_P[p \mapsto g])$$

$$\mathcal{D}^P_{ds}[\![\varepsilon]\!]env_V = id_{\mathrm{Env_P}}$$

Az eredeti példánk

Legyen S a következő program:

```
begin
    var x := 1 ; proc p is x := 0 ;
    begin
        var x := 2 ;
        call p
    end
end
```

Tegyük fel, hogy létezik env_V és env_P kezdeti környezetek és egy tár, sto, amelyre $sto\ next = 12$.

$$\mathcal{S}_{ds}[\![S]\!] env_V\ env_P\ sto = ?$$

begin var $x := 1$; **proc** p **is** $x := 0$; ... **end**

- A változódeklaráció szemantikája:

$$\mathcal{D}^V_{ds}[\![\textbf{var } x := 1 \ ; \ \varepsilon]\!](env_V, sto) =$$
$$\mathcal{D}^V_{ds}[\![\varepsilon]\!](env_V[x \mapsto 12], sto[12 \mapsto 1][next \mapsto 13]) =$$
$$(env_V[x \mapsto 12], sto[12 \mapsto 1][next \mapsto 13])$$

- Az eljárás szemantikája:

$$\mathcal{D}^P_{ds}[\![\textbf{proc } p \textbf{ is } x := 0 \ ; \ \varepsilon]\!] env_V[x \mapsto 12] \ env_P =$$
$$\mathcal{D}^P_{ds}[\![\varepsilon]\!] env_V[x \mapsto 12] \ env_P[p \mapsto g] =$$
$$env_P[p \mapsto g]$$

ahol $g \ sto = \mathcal{S}_{ds}[\![x := 0]\!] env_V[x \mapsto 12] \ env_P \ sto = sto[12 \mapsto 0]$

Most már meghatároztuk, hogy a külső blokk szemantikája a következő:

$$\mathcal{S}_{ds}[\![\textbf{begin var } x := 1 \ ; \ \textbf{proc } p \textbf{ is } x := 0 \ ; \quad \ldots \quad \textbf{end}]\!] env_V \ env_P \ sto = \mathcal{S}_{ds}[\![\ldots]\!] env_V[x \mapsto 12] \ env_P[p \mapsto g] \ sto[12 \mapsto 1][next \mapsto 13]$$

ahol ... a **begin var** $x := 2$; **call** p **end** helyett áll

Határozzuk meg a belső blokk jelentését is, hogy megkapjuk a teljes szemantikát:

$$\mathcal{S}_{ds}[\![\textbf{begin var } x := 2 \ ; \ \textbf{call } p \textbf{ end}]\!] \ env_V[x \mapsto 12] \ env_P[p \mapsto g] \ sto[12 \mapsto 1][next \mapsto 13]$$

begin var $x := 2$; **call** p **end**

- A változódeklaráció:

$$\mathcal{D}^V_{ds}[\![\textbf{var } x := 2 \ ; \ \varepsilon]\!](env_V[x \mapsto 12], sto[12 \mapsto 1][next \mapsto 13]) =$$
$$\mathcal{D}^V_{ds}[\![\varepsilon]\!](env_V[x \mapsto 13], sto[12 \mapsto 1][13 \mapsto 2][next \mapsto 14]) =$$
$$(env_V[x \mapsto 13], sto[12 \mapsto 1][13 \mapsto 2][next \mapsto 14])$$

- Az eljárásdeklaráció:

$$\mathcal{D}^P_{ds}[\![\varepsilon]\!] env_V[x \mapsto 13] \ env_P[p \mapsto g] = env_P[p \mapsto g]$$

$$\mathcal{S}_{ds}[\![\textbf{begin var } x := 2 \ ; \ \textbf{call } p \ \textbf{end}]\!]$$
$$(env_V[x \mapsto 12]) \ (env_P[p \mapsto g]) \ (sto[12 \mapsto 1][next \mapsto 13]) =$$
$$\mathcal{S}_{ds}[\![\textbf{call } p]\!]$$
$$(env_V[x \mapsto 13]) \ (env_P[p \mapsto g]) \ (sto[12 \mapsto 1][13 \mapsto 2][next \mapsto 14])$$

Az eredeti példánk

$$\mathcal{S}_{ds}[\![\textbf{begin var } x := 2 \ ; \ \textbf{call } p \ \textbf{end}]\!]$$
$$\quad (env_V[x \mapsto 12]) \ (env_P[p \mapsto g]) \ (sto[12 \mapsto 1][next \mapsto 13]) =$$
$$\mathcal{S}_{ds}[\![\textbf{call } p]\!]$$
$$\quad (env_V[x \mapsto 13]) \ (env_P[p \mapsto g]) \ (sto[12 \mapsto 1][13 \mapsto 2][next \mapsto 14]) =$$
$$g \ (sto[12 \mapsto 1][13 \mapsto 2][next \mapsto 14]) =$$
$$sto[12 \mapsto 0][13 \mapsto 2][next \mapsto 14]$$

Mivel 12 a külső x címe és 13 a belső x címe, látható, hogy az alprogramhívás a külső változót módosítja, tehát statikus hatóköri szabályokat alkalmaz a szemantikadefiníciónk.

λ

A While nyelv, a fent bemutatott eljárások és blokkszerkezet végrehajtható denotációs szemantikája elérhető a kurzus anyagai között. A leíró jellegű szemantikát Haskellben definiáltuk.

Szemantikadefiníciók ekvivalenciája

Horpácsi Dániel
daniel-h@elte.hu

A denotációs és az operációs szemantika ekvivalenciája

Emlékeztető: formális szemantikadefiníciók

- A programozási nyelvek szemantikája általában informálisan kerül leírásra, továbbá praktikusan a nyelv fordítóprogramja definiálja
- Ezek közül egyiket sem tekintjük formális definíciónak, nem használhatóak bizonyításhoz

Megoldás: használjuk a jó öreg matematikát és logikát!

Két alapvető megközelítés:

- Operációs (műveleti)
  - **Strukturális**

    Minden lépés modellezése
  - Természetes

    A kezdő- és végállapotok közötti reláció felállítása
- **Denotációs (leíró)**

  A jelentést matematikai objektumok hozzárendelésével adja meg

A szemantikadefiníciók kapcsolata

- A *While* magnyelvnek definiáltunk egy operációs és egy leíró szemantikát is
- Az alapvető imperatív programkonstrukcióknak (szekvencia, elágazás, ciklus) különbözőképp adtuk meg a jelentését a különböző módszerekkel

- A **leíró szemantika** absztraktabb, magasabb szintű leírás
- A **műveleti szemantika** közelebb áll a tényleges végrehajtáshoz, leírja, hogyan hajtanánk végre a programot lépésről lépésre

- *Összehasonlíthatóak ezek a különböző szemantikadefiníciók?*
- *Belátható, hogy ugyanazt a jelentést definiálják, csak másképp?*

A szemantikák ekvivalenciája

- Bizonyítani fogjuk, hogy

  minden S utasításra $\mathcal{S}_{sos}[\![S]\!] = \mathcal{S}_{ds}[\![S]\!]$

- Az utasításokhoz rendelt szemantikus domain a $(\mathrm{State} \hookrightarrow \mathrm{State}, \sqsubseteq)$ részbenrendezett halmaz
  - Ahhoz, hogy belássuk két szemantikus függvény, g_1 and g_2, ekvivalenciáját (egyenlőségét, $g_1 = g_2$), elég belátni, hogy $g_1 \sqsubseteq g_2$ és $g_2 \sqsubseteq g_1$.

- Tehát a két szemantika ekvivalenciájának belátásához belátjuk, hogy

  $\mathcal{S}_{sos}[\![S]\!] \sqsubseteq \mathcal{S}_{ds}[\![S]\!]$ és $\mathcal{S}_{ds}[\![S]\!] \sqsubseteq \mathcal{S}_{sos}[\![S]\!]$

$$\mathcal{S}_{sos}[\![S]\!]s = \begin{cases} s' & \text{ha } \langle S, s\rangle \Rightarrow^* s' \\ \mathit{undef} & \text{egyébként} \end{cases}$$

Az $\mathcal{S}_{sos}$ szemantikus függvény definíciója szerint be kell látni, hogy

$$\langle S, s\rangle \Rightarrow^* s' \quad \Longrightarrow \quad \mathcal{S}_{ds}[\![S]\!]s = s'$$

Ez belátható a levezetési láncok hossza szerinti indukcióval. Kell:

$$\langle S, s\rangle \Rightarrow s' \quad \Longrightarrow \quad \mathcal{S}_{ds}[\![S]\!]s = s'$$

$$\langle S, s\rangle \Rightarrow \langle S', s'\rangle \quad \Longrightarrow \quad \mathcal{S}_{ds}[\![S]\!]s = \mathcal{S}_{ds}[\![S']\!]s'$$

Ezeket a levezetési fa alakja szerinti indukcióval bizonyítjuk.

- $\langle \textbf{skip}, s \rangle \Rightarrow s$

  $\mathcal{S}_{ds}[\![\textbf{skip}]\!]s = s$

- $\langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$

  $\mathcal{S}_{ds}[\![x := a]\!]s = s[x \mapsto \mathcal{A}[\![a]\!]s]$

- $\langle S_1; S_2, s \rangle \Rightarrow \langle S_1'; S_2, s' \rangle$ mivel $\langle S_1, s \rangle \Rightarrow \langle S_1', s' \rangle$

  $\mathcal{S}_{ds}[\![S_1]\!]s = \mathcal{S}_{ds}[\![S_1']\!]s'$ (indukciós hipotézis)

$\mathcal{S}_{ds}[\![S_1; S_2]\!] = \mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!]$ (a szekvencia szemantikája)

$$\begin{aligned}\mathcal{S}_{ds}[\![S_1; S_2]\!]s &= \mathcal{S}_{ds}[\![S_2]\!](\mathcal{S}_{ds}[\![S_1]\!]s) = \mathcal{S}_{ds}[\![S_2]\!](\mathcal{S}_{ds}[\![S_1']\!]s') \\ &= \mathcal{S}_{ds}[\![S_1'; S_2]\!]s'\end{aligned}$$

- $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle$ mivel $\langle S_1, s\rangle \Rightarrow s'$

  $\mathcal{S}_{ds}[\![S_1]\!]s = s'$ (indukciós hipotézis)

  $\mathcal{S}_{ds}[\![S_1; S_2]\!]s = \mathcal{S}_{ds}[\![S_2]\!](\mathcal{S}_{ds}[\![S_1]\!]s) = \mathcal{S}_{ds}[\![S_2]\!]s'$

- $\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Rightarrow \langle S_1, s\rangle$ mivel $\mathcal{B}[\![b]\!]s = tt$

  $\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]s = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])s$
  $= \mathcal{S}_{ds}[\![S_1]\!]s$

- $\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$ mivel $\mathcal{B}[\![b]\!]s = ff$

  $\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]s = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])s$
  $= \mathcal{S}_{ds}[\![S_2]\!]s$

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

  A leíró szemantika szerint:

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

  $\quad$ ahol $F\ g = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$

$$
\begin{aligned}
\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] &= \mathrm{FIX}\ F \\
&= F\ (\mathrm{FIX}\ F) \\
&= F\ (\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!]) \\
&= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}}) \\
&= cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S\ ;\ \textbf{while } b \textbf{ do } S]\!], \mathcal{S}_{ds}[\![\textbf{skip}]\!]) \\
&= \mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}]\!]
\end{aligned}
$$

A bizonyítás fele kész. Viszont a leíró szemantika még lehet olyan állapotokban definiálva, ahol a műveleti nincs. A bizonyítást a másik irányba is elvégezzük.

Belátjuk, hogy

$$\mathcal{S}_{ds}[\![S]\!]s = s' \quad \Longrightarrow \quad \mathcal{S}_{sos}[\![S]\!]s = s'$$

Azaz,

$$\mathcal{S}_{ds}[\![S]\!]s = s' \quad \Longrightarrow \quad \langle S, s\rangle \Rightarrow^* s'$$

Mivel a denotációs szemantikát kompozicionálisan definiáltuk, bizonyíthatunk strukturális indukció segítségével.

- Nyilvánvalóan $\mathcal{S}_{ds}[\![\textbf{skip}]\!]s = \mathcal{S}_{sos}[\![\textbf{skip}]\!]s$
- És $\mathcal{S}_{ds}[\![x := a]\!]s = s[x \mapsto \mathcal{A}[\![a]\!]s] = \mathcal{S}_{sos}[\![x := a]\!]s$
- $\mathcal{S}_{ds}[\![S_1; S_2]\!] = \mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!]$

$\mathcal{S}_{ds}[\![S_2]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_2]\!]$ (indukciós hipotézis)
$\mathcal{S}_{ds}[\![S_1]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_1]\!]$ (indukciós hipotézis)

$\mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_2]\!] \circ \mathcal{S}_{sos}[\![S_1]\!]$
($\circ$ mindkét argumentumában monoton)

$(\mathcal{S}_{sos}[\![S_2]\!] \circ \mathcal{S}_{sos}[\![S_1]\!])s = s''$ ha $\mathcal{S}_{sos}[\![S_1]\!]s = s'$ és $\mathcal{S}_{sos}[\![S_2]\!]s' = s''$

- $\mathcal{S}_{sos}[\![S_1]\!]s = s' \implies \langle S_1, s\rangle \Rightarrow^* s'$
- $\langle S_1, s\rangle \Rightarrow^* s' \implies \langle S_1; S_2, s\rangle \Rightarrow^* \langle S_2, s'\rangle$ (korábban beláttuk)
- $\mathcal{S}_{sos}[\![S_2]\!]s' = s'' \implies \langle S_2, s'\rangle \Rightarrow^* s''$

Kapjuk tehát, hogy

$\langle S_1; S_2, s\rangle \Rightarrow^* s''$

Azaz,

$(\mathcal{S}_{sos}[\![S_1 \ ; \ S_2]\!])s = s''$

Következésképp,

$\mathcal{S}_{sos}[\![S_2]\!] \circ \mathcal{S}_{sos}[\![S_1]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_1 \ ; \ S_2]\!]$

- $\mathcal{S}_{ds}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])$

$\mathcal{S}_{ds}[\![S_2]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_2]\!]$ (indukciós hipotézis)
$\mathcal{S}_{ds}[\![S_1]\!] \sqsubseteq \mathcal{S}_{sos}[\![S_1]\!]$ (indukciós hipotézis)

$cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!]) \sqsubseteq cond(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![S_1]\!], \mathcal{S}_{sos}[\![S_2]\!])$
(*cond* monoton a második és harmadik argumentumában)

$\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Rightarrow \langle S_1, s\rangle$ ha $\mathcal{B}[\![b]\!]s = tt$
$\langle \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$ ha $\mathcal{B}[\![b]\!]s = ff$

$$cond(p, g_1, g_2)s = \begin{cases} g_1\ s & \text{ha} \quad p\ s = tt \\ g_2\ s & \text{ha} \quad p\ s = ff \end{cases}$$

$cond(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![S_1]\!], \mathcal{S}_{sos}[\![S_2]\!]) = \mathcal{S}_{sos}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]$

- $\mathcal{S}_{ds}[\![\textbf{while } b \textbf{ do } S]\!] = \mathrm{FIX}\ F$
  ahol $F\ g = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id_{\mathrm{State}})$ és F folytonos

Elég belátni, hogy

$$F(\mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!]) \sqsubseteq \mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!]$$

mert akkor

$$FIX\ F \sqsubseteq \mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!]$$

Trükk:

$$F\ g \sqsubseteq g \quad \Longrightarrow \quad FIX\ F \sqsubseteq g$$

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

$$\begin{aligned}
&\mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!] = \mathcal{S}_{sos}[\![\textbf{if } b \textbf{ then } (S; \textbf{while } b \textbf{ do } S) \textbf{ else skip}]\!] \\
&\quad = \mathit{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![S \ ; \ \textbf{while } b \textbf{ do } S]\!], \mathit{id}_{\mathrm{State}}) \\
&\quad \sqsupseteq \mathit{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{sos}[\![S]\!], \mathit{id}_{\mathrm{State}})
\end{aligned}$$

És az indukciós hipotézis használatával:

$$\mathcal{S}_{ds}[\![S]\!] \sqsubseteq \mathcal{S}_{sos}[\![S]\!]$$

A $\circ$ és *cond* monotonitása miatt:

$$\begin{aligned}
&\mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!] \\
&\quad \sqsupseteq \mathit{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{sos}[\![S]\!], \mathit{id}_{\mathrm{State}}) \\
&\quad \sqsupseteq \mathit{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!] \circ \mathcal{S}_{ds}[\![S]\!], \mathit{id}_{\mathrm{State}}) \\
&\quad = F(\mathcal{S}_{sos}[\![\textbf{while } b \textbf{ do } S]\!])
\end{aligned}$$

Az ekvivalencia

- Beláttuk, hogy

$$\mathcal{S}_{sos}[\![S]\!] \sqsubseteq \mathcal{S}_{ds}[\![S]\!] \qquad \text{és} \qquad \mathcal{S}_{ds}[\![S]\!] \sqsubseteq \mathcal{S}_{sos}[\![S]\!]$$

- Következésképp

$$\mathcal{S}_{ds}[\![S]\!] = \mathcal{S}_{sos}[\![S]\!]$$

A leíró és a small-step műveleti szemantika ekvivalens

- A small-step és a big-step ekvivalenciája is belátható lenne, tehát hogy mindhárom definíciónk ekvivalens
- Viszont vegyük észre, hogy itt csak a magnyelvet vizsgáltuk

Specifikus nyelvek implementációja

Horpácsi Dániel
daniel-h@elte.hu

Formális szemantika mint eszköz

alkalmazás-specifikus nyelvek megvalósításában

DSL: alkalmazás-specifikus nyelvek

- Kisebb, specifikus (programozási) nyelvek, limitált kifejezőerővel
- Egy konkrét terület, probléma megoldásának leírására fókuszál
- Az adott terület/probléma fogalmait használja nyelvi elemként
- (4GL programozási nyelvek)

Előnyei:

- A megoldás egy problémára specifikus nyelven könnyebben és gyorsabban megadható
- A programozási tudással nem rendelkező, de a problématerületet jól ismerő emberek megértik és akár “programozni” is tudják a nyelvet

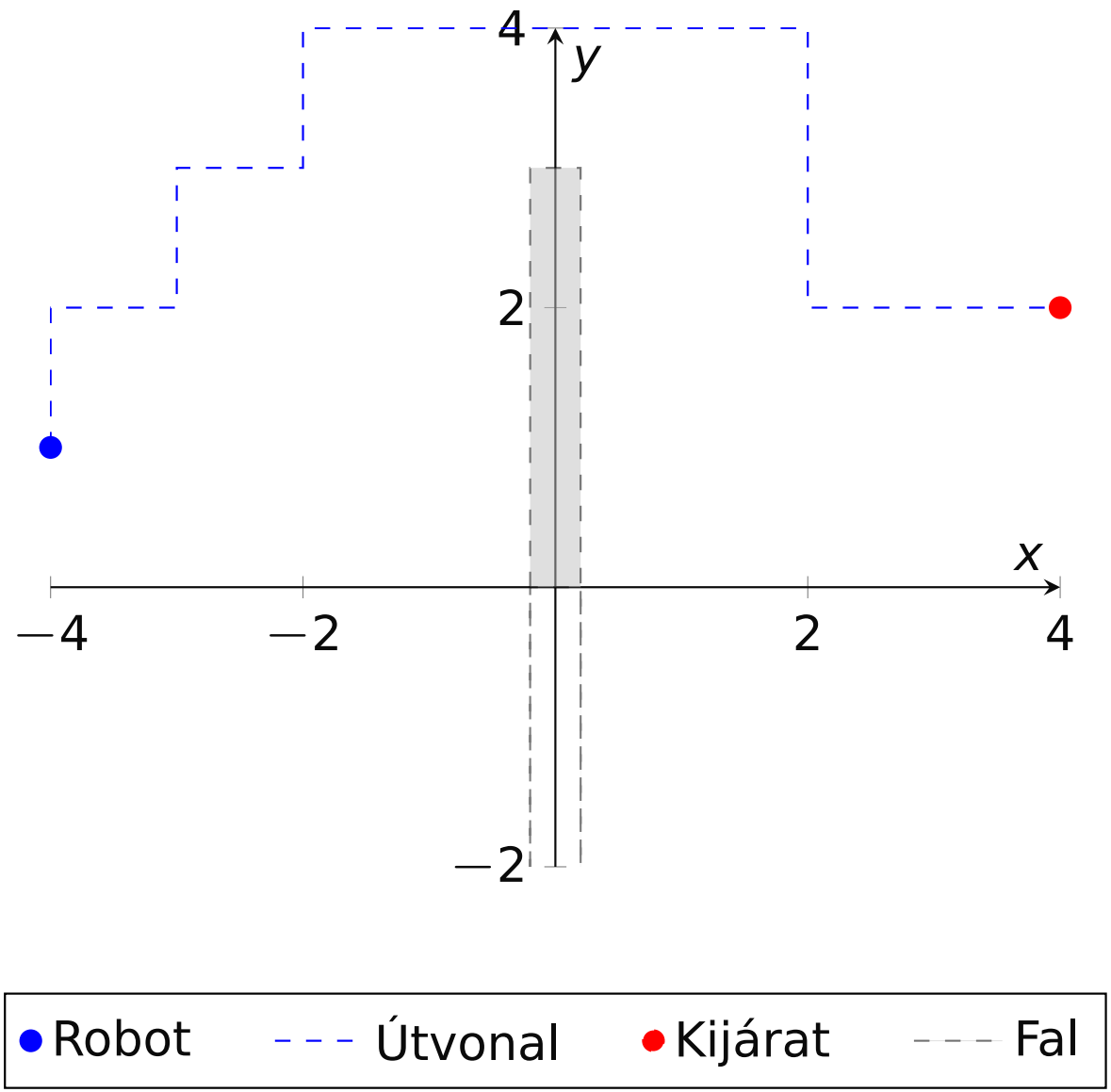
A specifikus nyelvek szemantikája

- Természetesen más nyelvi elemeket tartalmaznak, mint amiket eddig tárgyaltunk, más absztrakciós szinttel — viszont sokszor egy általános célú programozási nyelvbe ágyazva implementálják
- Következésképp a DSL szemantikáját a gazdanyelven fejezik ki

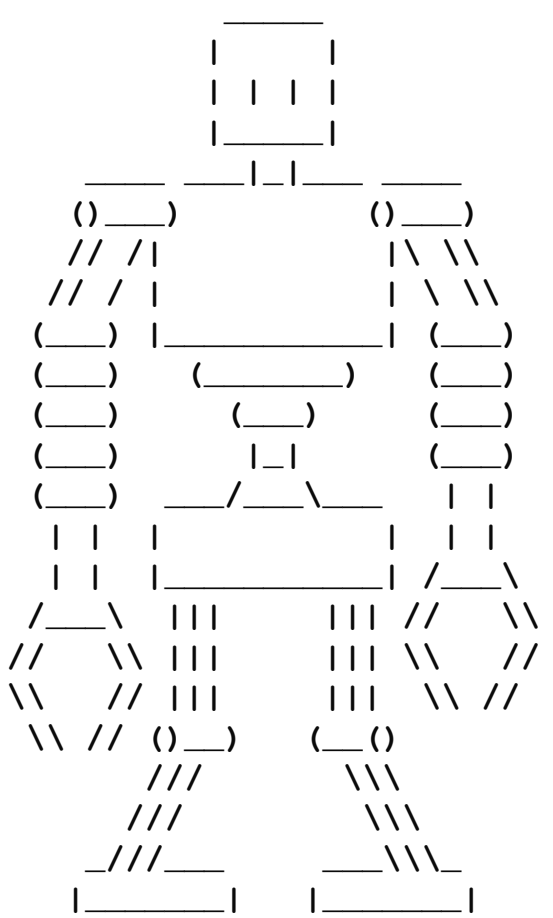
- Vajon mennyire precíz megadása ez a szemantikának? Jól definiált a gazdanyelv szemantikája?
- Vajon tudjuk követni a DSL program végrehajtását?
- Tudunk formálisan érvelni a tárgyspecifikus program működéséről?

Miért ne használnánk a szemantikadefiníciós módszereket specifikus nyelvek definiálására?

Példanyelv: ROBOT



This example was inspired by
Prof. Marjan Mernik
University of Maribor



A nyelv fogalmai, absztrakt szitaxis

Egy képzeletbeli problémához készítünk nyelvet: egy robotnak adunk utasításokat, amelyekkel egy falat kikerülve eljuthat a kijárathoz.

Két szintaktikus kategóriát veszünk fel: utasítás (command) és számliterál. Utóbbit az előbbi paraméterezésére használjuk majd.

Szintaktikus kategóriák

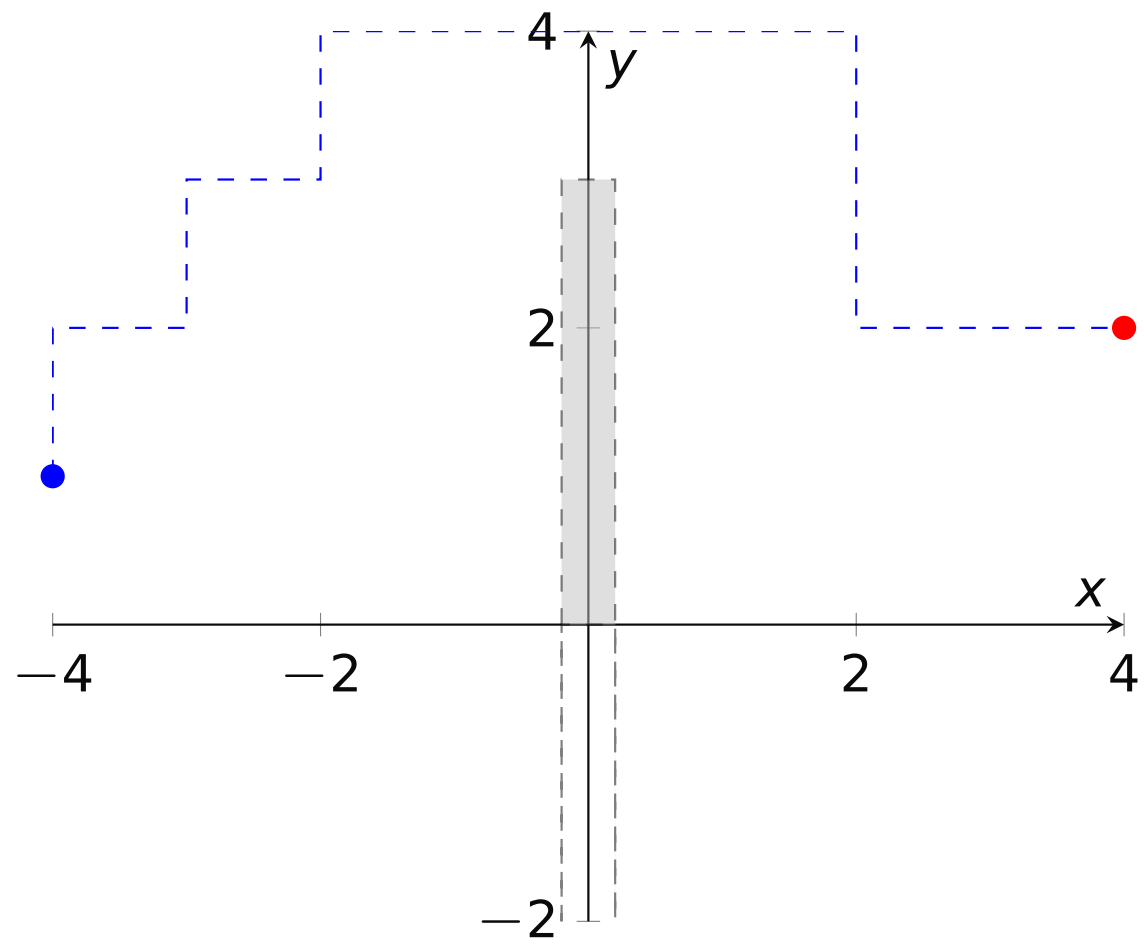
$$n \in \mathrm{Num} \quad \text{(számliterál)}$$
$$C \in \mathrm{Comm} \quad \text{(utasítás)}$$

A lehetséges robotparancsokat rekurzívan definiáljuk:

Produkciós szabályok

$$\begin{aligned} C ::= \; & \textbf{left} \mid \textbf{right} \mid \textbf{up} \mid \textbf{down} \mid \\ & \textbf{left}\ n \mid \textbf{right}\ n \mid \textbf{up}\ n \mid \textbf{down}\ n \mid \\ & \textbf{reset} \mid C_1\ ;\ C_2 \end{aligned}$$

Példaprogram a specifikus nyelvben



reset ; **up** ; **right** ;
up ; **right** ; **up** ;
right ; **right** ;
right ; **right** ;
down ; **down** ;
right ; **right**

Mi egy ilyen program jelentése? Mi a denotációja egy parancssorozatnak?

Képezzük le a robot mozgását a koordinátarendszerre. A robot pozíciójának módosítása lesz az utasítások szemantikája.

Szemantikus domain

$\mathrm{Point} = \mathbb{Z} \times \mathbb{Z}$ (x és y koordináták)

A denotációs szemantika nem a robot útját, hanem annak végpontját, a robot utolsó pozícióját adja meg.

Az egyszerűség kedvéért most nem foglalkozunk a fallal. Statikus szemantikus szabályokkal vagy kivételkezeléssel azt is lehetne modellezni, hogy a robot megakad a falnál (és persze operációs szemantikában gondoljunk a zsákutca konfigurációkra is).

$$\mathcal{C} : \mathrm{Comm} \to (\mathrm{Point} \to \mathrm{Point})$$

A szemantikus függvény

$$\begin{aligned}
\mathcal{C}[\![\textbf{left}]\!](x,y) &= (x-1,y)\\
\mathcal{C}[\![\textbf{right}]\!](x,y) &= (x+1,y)\\
\mathcal{C}[\![\textbf{up}]\!](x,y) &= (x,y+1)\\
\mathcal{C}[\![\textbf{down}]\!](x,y) &= (x,y-1)\\
\mathcal{C}[\![\textbf{left}\ n]\!](x,y) &= (x-\mathcal{N}[\![n]\!],y)\\
\mathcal{C}[\![\textbf{right}\ n]\!](x,y) &= (x+\mathcal{N}[\![n]\!],y)\\
\mathcal{C}[\![\textbf{up}\ n]\!](x,y) &= (x,y+\mathcal{N}[\![n]\!])\\
\mathcal{C}[\![\textbf{down}\ n]\!](x,y) &= (x,y-\mathcal{N}[\![n]\!])\\
\mathcal{C}[\![\textbf{reset}]\!]p &= (-4,1)\\
\mathcal{C}[\![C_1\ ;\ C_2]\!] &= \mathcal{C}[\![C_2]\!] \circ \mathcal{C}[\![C_1]\!]
\end{aligned}$$

- A konfigurációk most vagy egy parancs és egy pozíció által alkotott párt adnak meg, vagy egy végső pozíciót.

- Az átmeneteket a következő formában adjuk meg:

  Egy közbülső lépés, ahol C' a hátralévő lépéseket tartalmazza:

  $$\langle C, p \rangle \Rightarrow \langle C', p' \rangle \qquad p, p' \in \mathrm{Point}$$

  A C parancs végrehajtása után a végső pozíció p':

  $$\langle C, p \rangle \Rightarrow p' \qquad p, p' \in \mathrm{Point}$$

Mozgató utasítások

$$\frac{}{\langle \textbf{left}, (x, y)\rangle \Rightarrow (x-1, y)} \qquad \frac{}{\langle \textbf{left}\ 1, p\rangle \Rightarrow \langle \textbf{left}, p\rangle}$$

$$\frac{}{\langle \textbf{left}\ n, p\rangle \Rightarrow \langle \textbf{left}\ ;\ \textbf{left}\ m, p\rangle}\ \mathcal{N}[\![n]\!] = \mathcal{N}[\![m]\!] + 1 > 1$$

(A **right**, **up** és **down** hasonlóan megadhatók.)

Reset

$$\frac{}{\langle \textbf{reset}, p\rangle \Rightarrow (-4, 1)}$$

Parancsok szekvenciája

$$\frac{\langle C_1, (x, y)\rangle \Rightarrow \langle C_1', (x', y')\rangle}{\langle C_1; C_2, (x, y)\rangle \Rightarrow \langle C_1'; C_2, (x', y')\rangle} \qquad \frac{\langle C_1, (x, y)\rangle \Rightarrow (x', y')}{\langle C_1; C_2, (x, y)\rangle \Rightarrow \langle C_2, (x', y')\rangle}$$

A Robot nyelv kiterjesztésa

Tegyük a robotot forgathatóvá, hogy tudjuk mozgatni abba az irányba, amerre áll (gondoljunk a LOGO nyelvre). A nyelv fogalomrendszerét kibővítjük a haladási iránnyal, forgással és előrehaladással.

A kiterjesztett nyelv szintaxisa a következőképp alakul:

Produkciós szabályok

$$
\begin{aligned}
C \quad ::= \quad & \textbf{left} \mid \textbf{right} \mid \textbf{up} \mid \textbf{down} \mid \\
& \textbf{left}\ n \mid \textbf{right}\ n \mid \textbf{up}\ n \mid \textbf{down}\ n \mid \\
& \textbf{reset} \mid C_1\ ;\ C_2 \mid \\
& \textbf{forward} \mid \textbf{forward}\ n \mid \textbf{turn left} \mid \textbf{turn right}
\end{aligned}
$$

Operációs szematika kiterjesztése

- A konfigurációt kibővítjük a haladási irányt reprezentáló cellával. Az irányt egy szögként ábrázoljuk (0 jelenti a felfelé haladást, 90 a jobbra, 180 a lefelé és 270 a balra forduló robotot).
- A szabályokat úgy adjuk meg, hogy bármilyen szögben tud haladni a robot, de a jelenlegi nyelv csak a négy alapvető irányba mozgást támogatja.

$$\text{Angle} = \mathbb{Z}$$

- Köztes lépés:

$$\langle C, p, \alpha \rangle \Rightarrow \langle C', p', \alpha' \rangle \qquad p, p' \in \text{Point} \qquad \alpha, \alpha' \in \text{Angle}$$

Utolsó lépés:

$$\langle C, p, \alpha \rangle \Rightarrow \langle p', \alpha' \rangle \qquad p, p' \in \text{Point} \qquad \alpha, \alpha' \in \text{Angle}$$

Fordulás

$$\frac{}{\langle \textbf{turn left}, p, \alpha \rangle \Rightarrow \langle p, (\alpha + 270) \mod 360 \rangle}$$

$$\frac{}{\langle \textbf{turn right}, p, \alpha \rangle \Rightarrow \langle p, (\alpha + 90) \mod 360 \rangle}$$

Az érdekesség kedvéért a többlépéses **forward** jelentését az eddigiektől eltérően definiáljuk, a végrehajtása egyetlen szemantikai lépésben megtörténik (“kevésbé small-step”).

Előre haladás

$$\frac{}{\langle \textbf{forward}, (x, y), \alpha \rangle \Rightarrow \langle (x + \sin\alpha, y + \cos\alpha), \alpha \rangle}$$

$$\frac{}{\langle \textbf{forward}\ n, (x, y), \alpha \rangle \Rightarrow \langle (x + i * \sin\alpha, y + i * \cos\alpha), \alpha \rangle}\ \mathcal{N}[\![n]\!] = i$$

($\sin\alpha$ és $\cos\alpha$ értéke -1, 0 vagy 1, hiszen α most osztható 90-nel.)

A jobbra/balra/fel/le mozgás szemantikáját újradefiniáljuk a forgás és előrehaladás segítségével:

Mozgás jobbra

$$\frac{}{\langle \textbf{right}, p, \alpha \rangle \Rightarrow \langle \textbf{forward}, p, \alpha \rangle} \; \alpha = 90$$

$$\frac{}{\langle \textbf{right}, p, \alpha \rangle \Rightarrow \langle \textbf{turn left}; \textbf{right}, p, \alpha \rangle} \; \alpha \neq 90$$

$$\frac{}{\langle \textbf{right}\ n, p, \alpha \rangle \Rightarrow \langle \textbf{right}\ ;\ \textbf{right}\ m, p, \alpha \rangle} \; \mathcal{N}[\![n]\!] = \mathcal{N}[\![m]\!] + 1 > 1$$

$$\frac{}{\langle \textbf{right}\ 1, p, \alpha \rangle \Rightarrow \langle \textbf{right}, p, \alpha \rangle}$$

Mozgás balra

$$\frac{}{\langle \textbf{left}, p, \alpha \rangle \Rightarrow \langle \textbf{forward}, p, \alpha \rangle} \; \alpha = 270$$

$$\frac{}{\langle \textbf{left}, p, \alpha \rangle \Rightarrow \langle \textbf{turn right}; \textbf{left}, p, \alpha \rangle} \; \alpha \neq 270$$

$$\frac{}{\langle \textbf{left}\ n, p, \alpha \rangle \Rightarrow \langle \textbf{left}\ ;\ \textbf{left}\ m, p, \alpha \rangle} \; \mathcal{N}[\![n]\!] = \mathcal{N}[\![m]\!] + 1 > 1$$

$$\frac{}{\langle \textbf{left}\ 1, p, \alpha \rangle \Rightarrow \langle \textbf{left}, p, \alpha \rangle}$$

Ezek után az **up** és **down** jelentése hasonlóan megadható.

A **reset** parancs jelentése is megváltozik, hiszen a haladási irányt is be kell állítani:

Reset

$$\frac{}{\langle \textbf{reset}, p, \alpha \rangle \Rightarrow \langle ((-4, 1), 0) \rangle}$$

A parancsok kombinációja hasonlóan definiálható, mind eddig, csak hozzá kell igazítani a szabályokat a konfigurációk új formájához:

Parancsok kombinációja

$$\frac{\langle C_1, (x, y), \alpha \rangle \Rightarrow \langle C_1', (x', y'), \alpha' \rangle}{\langle C_1; C_2, (x, y), \alpha \rangle \Rightarrow \langle C_1'; C_2, (x', y')\alpha' \rangle}$$

$$\frac{\langle C_1, (x, y) \rangle \Rightarrow \langle (x', y'), \alpha' \rangle}{\langle C_1; C_2, (x, y) \rangle \Rightarrow \langle C_2, (x', y'), \alpha' \rangle}$$

$\mathbb{K}$

A fenti, kiterjesztett Robot nyelv végrehajtható szemantikája elérhető a kurzus anyagai között. A small-step stílusú operációs szemantikát a $\mathbb{K}$ keretrendszerben definiáltuk, kipróbálható a 3.5 és a 3.6 verziókkal.