A szintaxistól a szemantikáig

Dr. Horpácsi Dániel
ELTE Informatikai Kar
2023-2024-2

A szintaxis és statikus szemantika leírása

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:

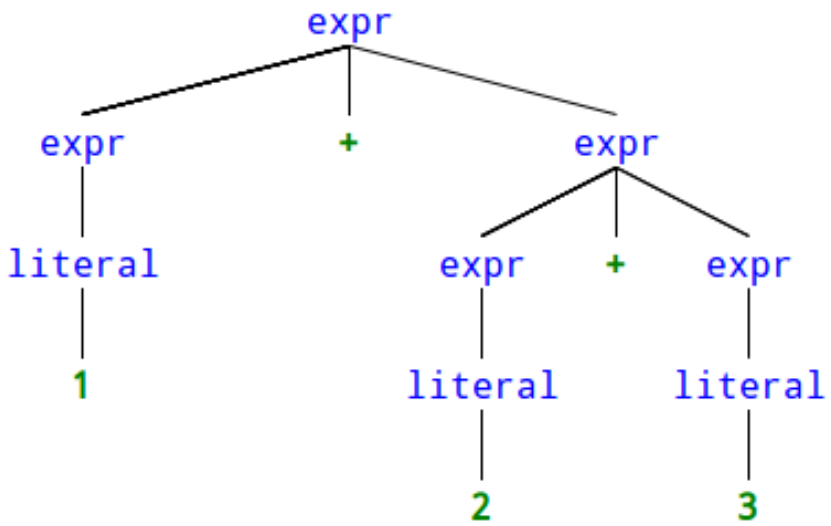
```
<expr>    ::= <literal> | <expr> '+' <expr>
<literal> ::= '0' | '1' | ... | '9'
```

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?

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

```
<expr> ::= <expr> '+' <expr>
```

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

```
expr ::= {add} expr expr
add : expr -> expr -> expr
```

Konkrét szintaxis kontra absztrakt szintaxis

Az absztrakt szitaxisból kimaradnak például

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

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 (majd az 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

- Típusrendszer: levezetési szabályok környezetfüggő típusítéletekre

$$\frac{}{n : \text{Nat}} \qquad \frac{e_1 : \text{Nat} \qquad e_2 \text{Nat}}{e_1 + e_2 : \text{Nat}}$$

- Attribútum grammatika: a fa csúcsaira extra információt helyezünk el összetett címkézéssel, az attribútumok értékeire megkötéseket teszünk

```
expr -> num
        { $$ = nat }
      | expr '+' expr
        { if($1 == nat && $3 == nat) $$ = nat;
          else error(); }
```

Attribútum grammatikák

$AG = (\mathcal{G}, \mathcal{A}, \mathcal{R}, \mathcal{C})$ 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

$$
\begin{array}{lll}
\langle \text{abcSeq} \rangle ::= & \langle \text{aSeq} \rangle \ \langle \text{bSeq} \rangle \ \langle \text{cSeq} \rangle & \\
& \quad \mathit{InSize}(\langle \text{bSeq} \rangle) \leftarrow \mathit{Size}(\langle \text{aSeq} \rangle) & \\
& \quad \mathit{InSize}(\langle \text{cSeq} \rangle) \leftarrow \mathit{Size}(\langle \text{aSeq} \rangle) & \\
\langle \text{aSeq} \rangle ::= & \mathbf{a} & \\
& \mathit{Size}(\langle \text{aSeq} \rangle) \leftarrow 1 & \\
\quad | & \langle \text{aSeq} \rangle_2 \ \mathbf{a} & \\
& \mathit{Size}(\langle \text{aSeq} \rangle) \leftarrow \mathit{Size}(\langle \text{aSeq} \rangle_2) + 1 & \\
\langle \text{bSeq} \rangle ::= & \mathbf{b} & \\
& \mathit{Condition:}\ \mathit{InSize}(\langle \text{bSeq} \rangle) = 1 & \\
\quad | & \langle \text{bSeq} \rangle_2 \ \mathbf{b} & \\
& \mathit{InSize}(\langle \text{bSeq} \rangle_2) \leftarrow \mathit{InSize}(\langle \text{bSeq} \rangle) - 1 & \\
\langle \text{cSeq} \rangle ::= & \mathbf{c} & \\
& \mathit{Condition:}\ \mathit{InSize}(\langle \text{cSeq} \rangle) = 1 & \\
\quad | & \langle \text{cSeq} \rangle_2 \ \mathbf{c} & \\
& \mathit{InSize}(\langle \text{cSeq} \rangle_2) \leftarrow \mathit{InSize}(\langle \text{cSeq} \rangle) - 1 &
\end{array}
$$

Az attribútumok kiértékelése

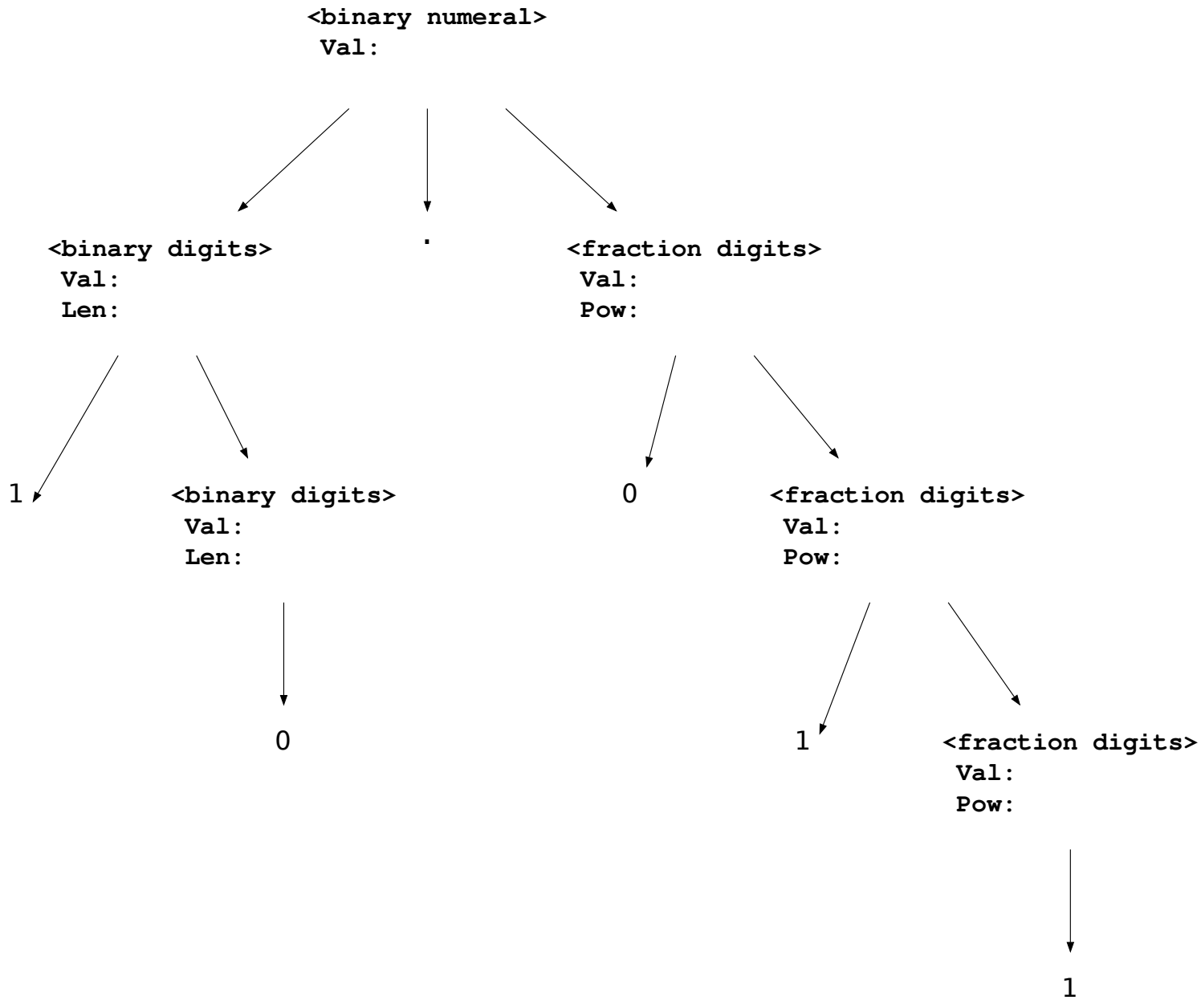
- (Úgy is mondják, hogy a szintaxisfa dekorálása.)

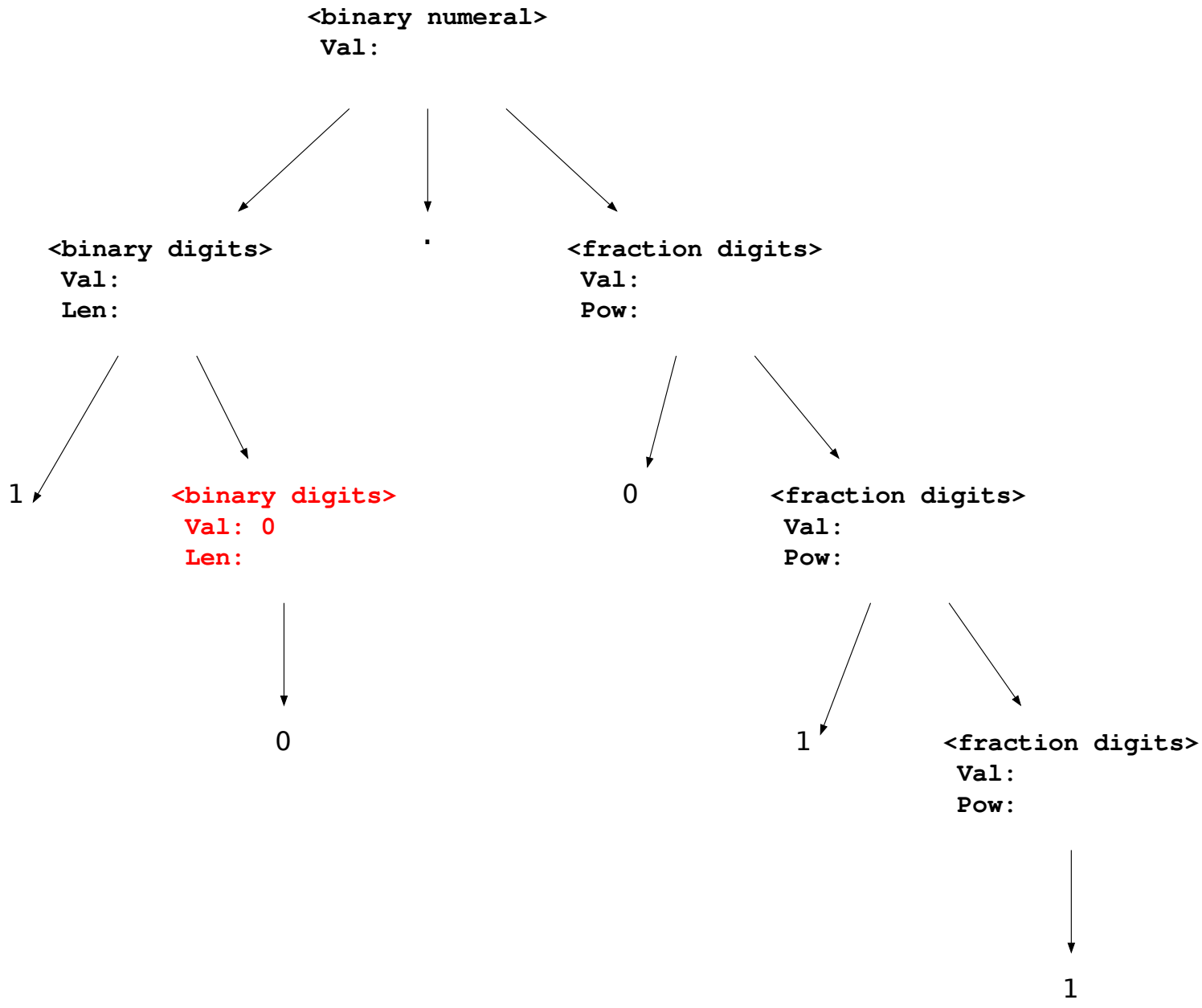
- Az attribútumok, a kiszámítási szabályaik és a feltételek együttesen alkalmasak arra, hogy a nyelv környezetfüggő tulajdonságait leírják
- Amikor egy feltétel kiértékelése hamis értéket eredményez, az szemantikus hibát jelez

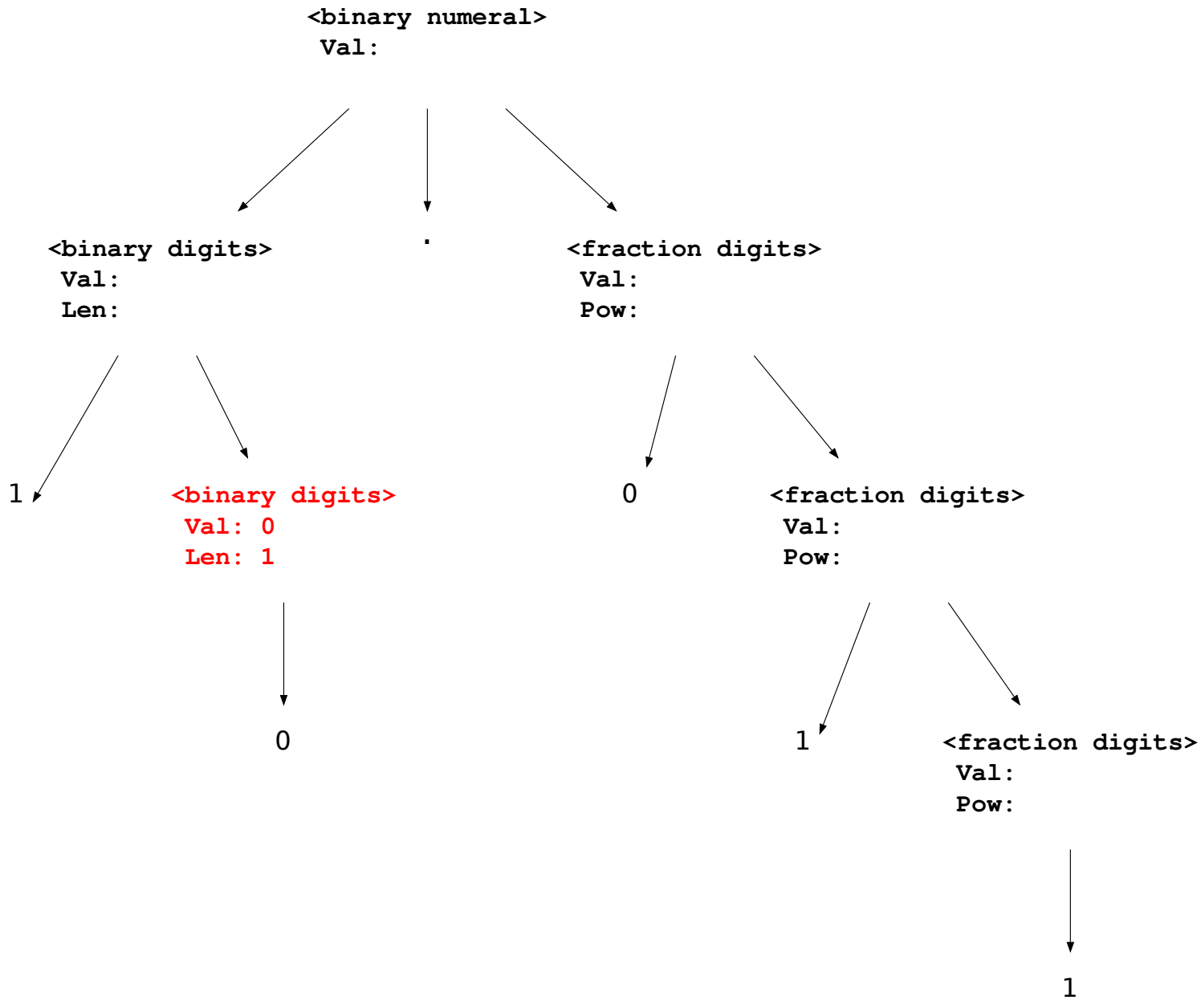
- Azonban a kiértékelés nem mindig egyszerű; az attribútumok között függőségek állnak fent, amelyeket figyelembe kell venni
- Annak eldöntése, hogy egy AG szintaxisfája kiértékelhető-e, NP-teljes probléma
- A megfelelő kiszámítási sorrend megtalálására vannak heurisztikák

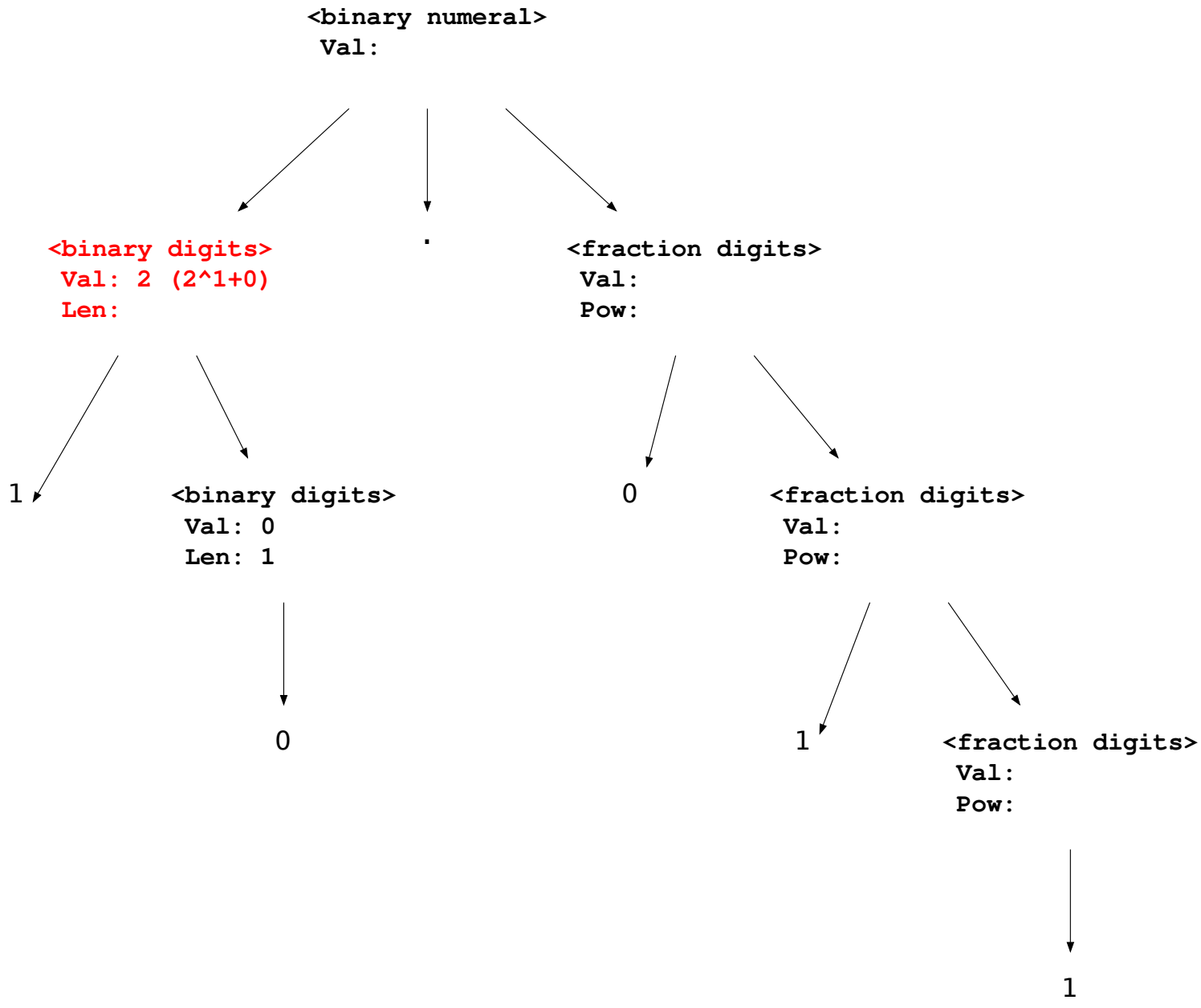
Példa — levezetési fa dekorálása

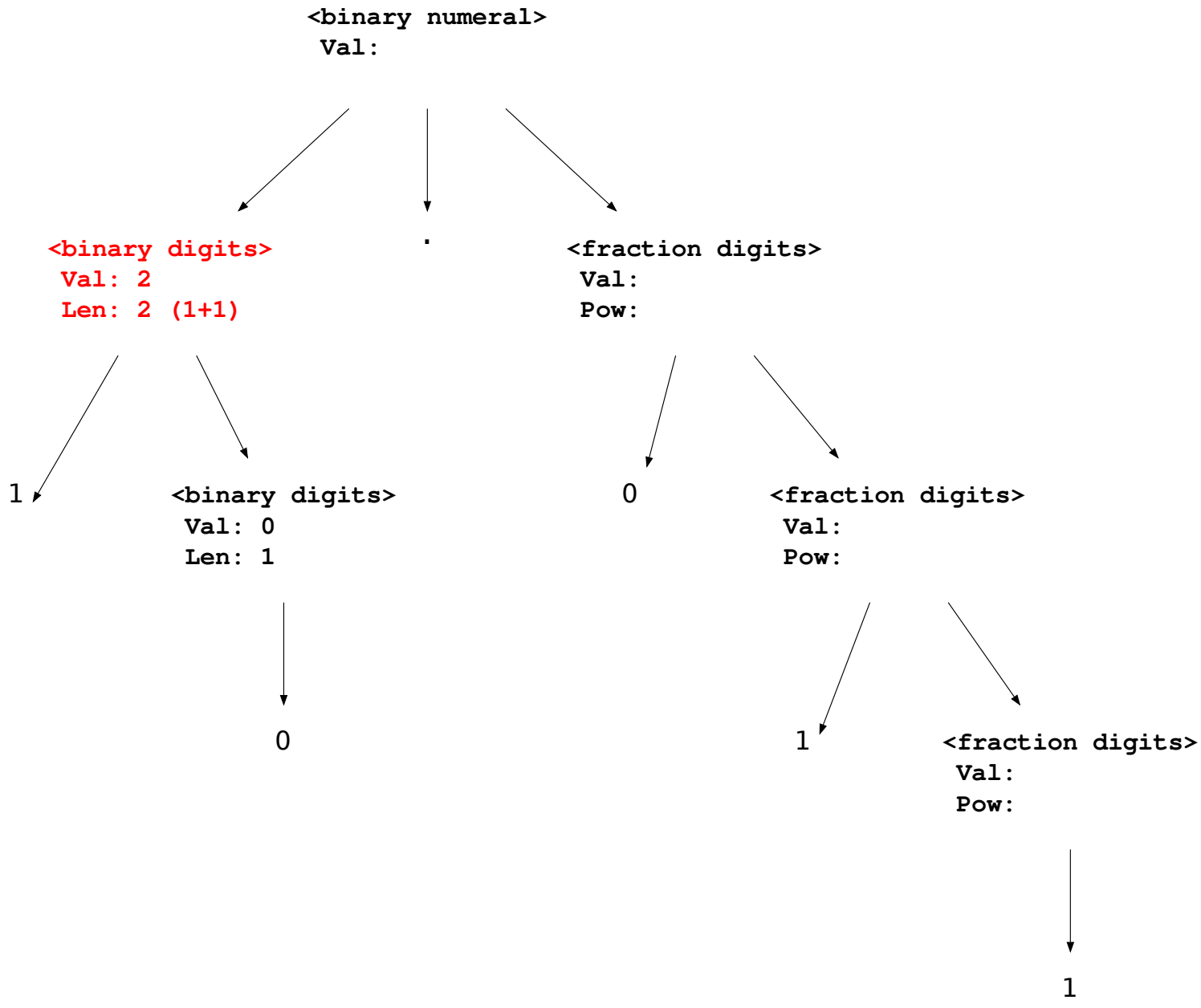
- Tekintsük a *exercises_02.pdf* dokumentumban felírt attribútum grammatikát
- A következő ábrákon látható a “10.011” mondathoz tartozó szintaxisfa attribútumainak egy lehetséges kiértékelése

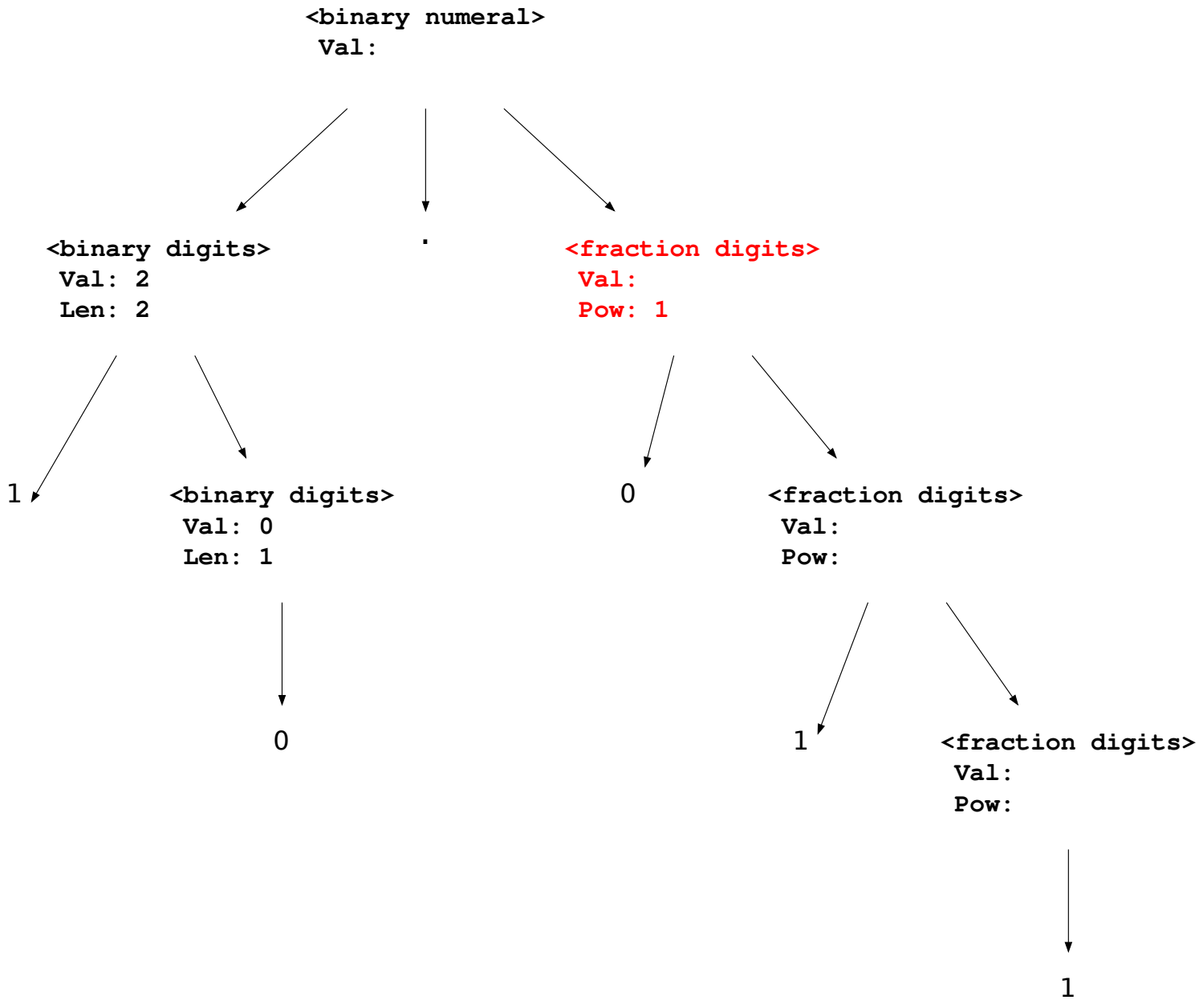

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val:
Len:
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

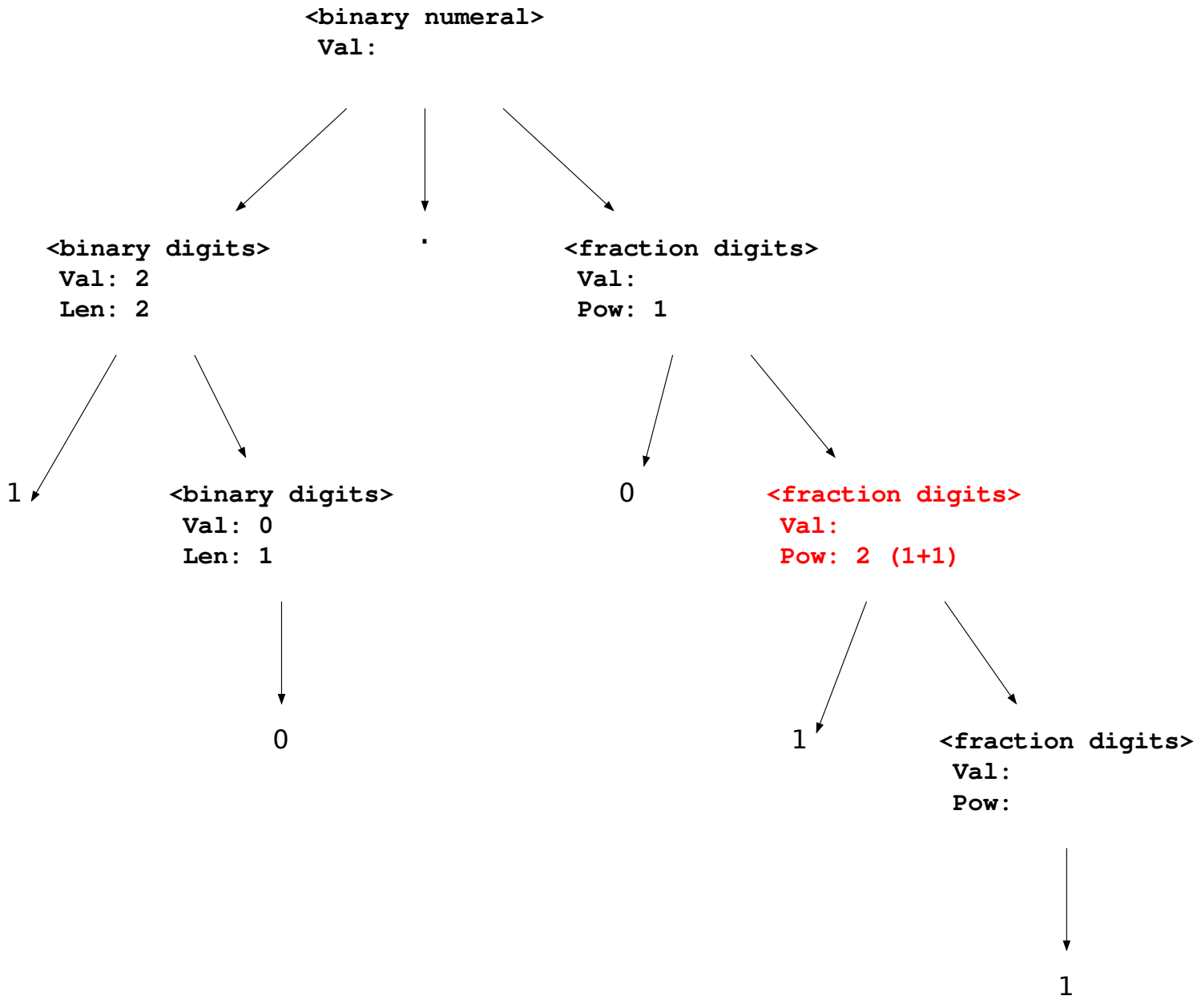

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len:
0
0
<fraction digits>
Val:
Pow:
1
<fraction digits>
Val:
Pow:
1

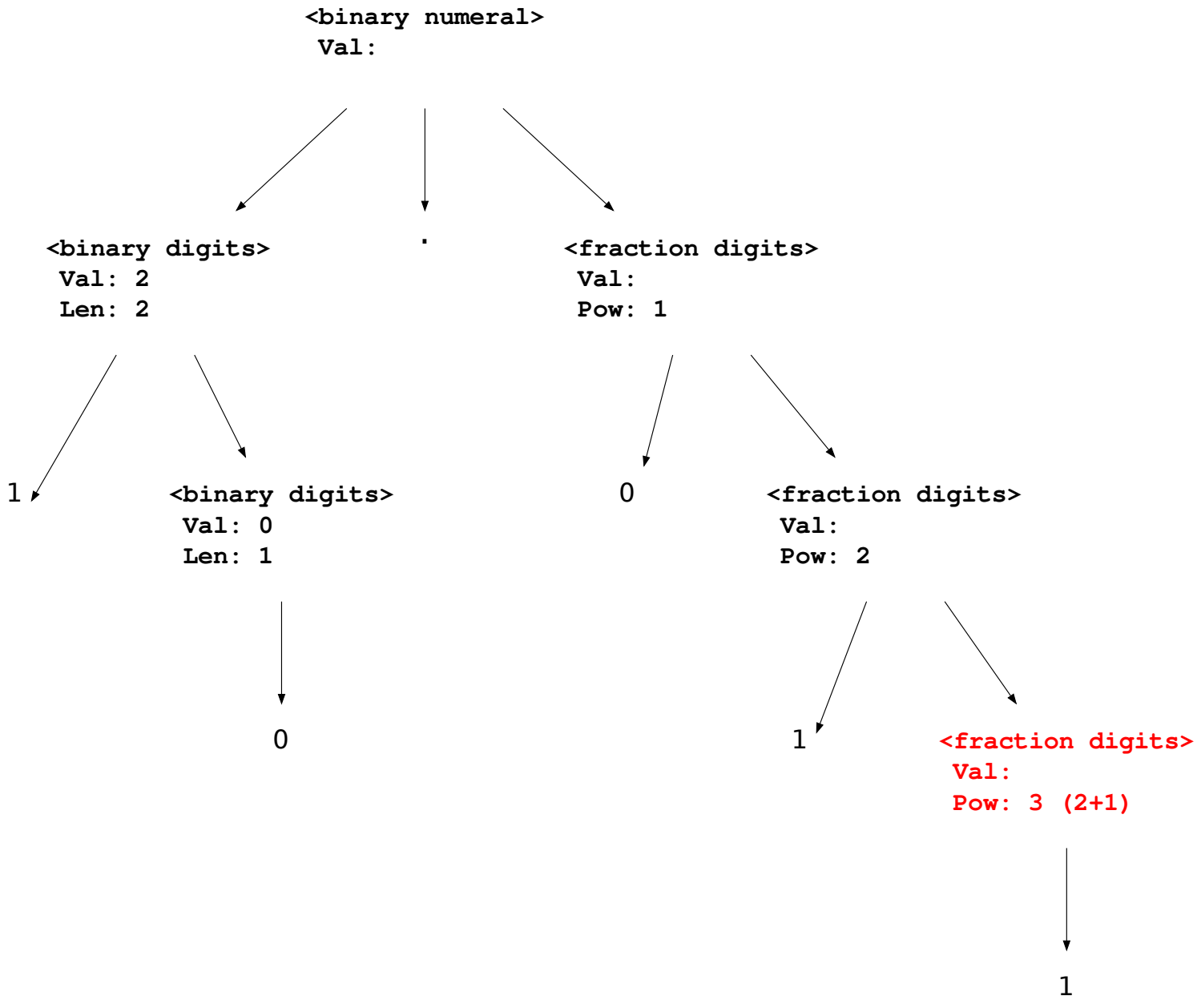

<binary numeral>
Val:
<binary digits>
Val:
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

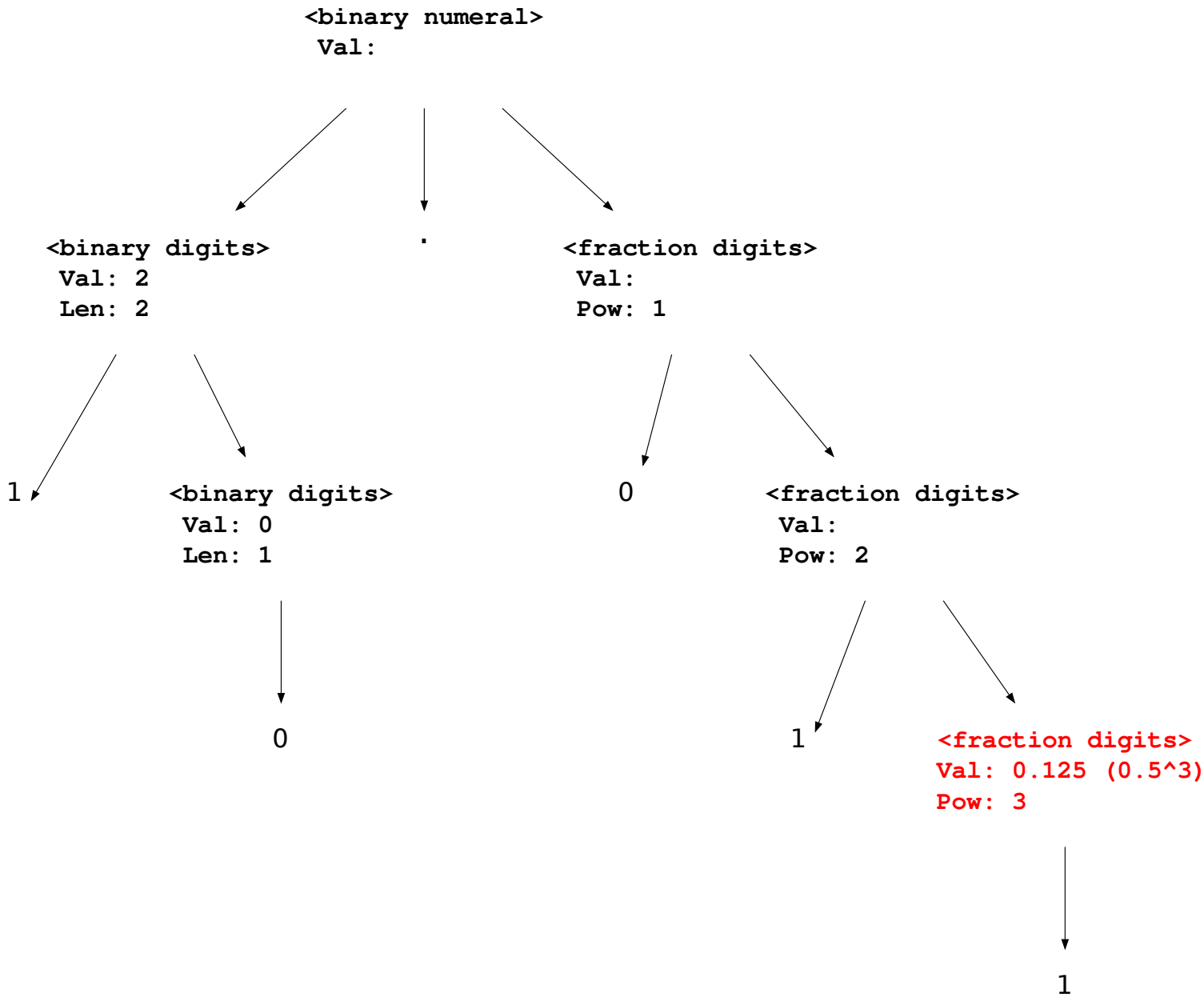

<binary numeral>
Val:
<binary digits>
Val: 2 (2^1+0)
Len:
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

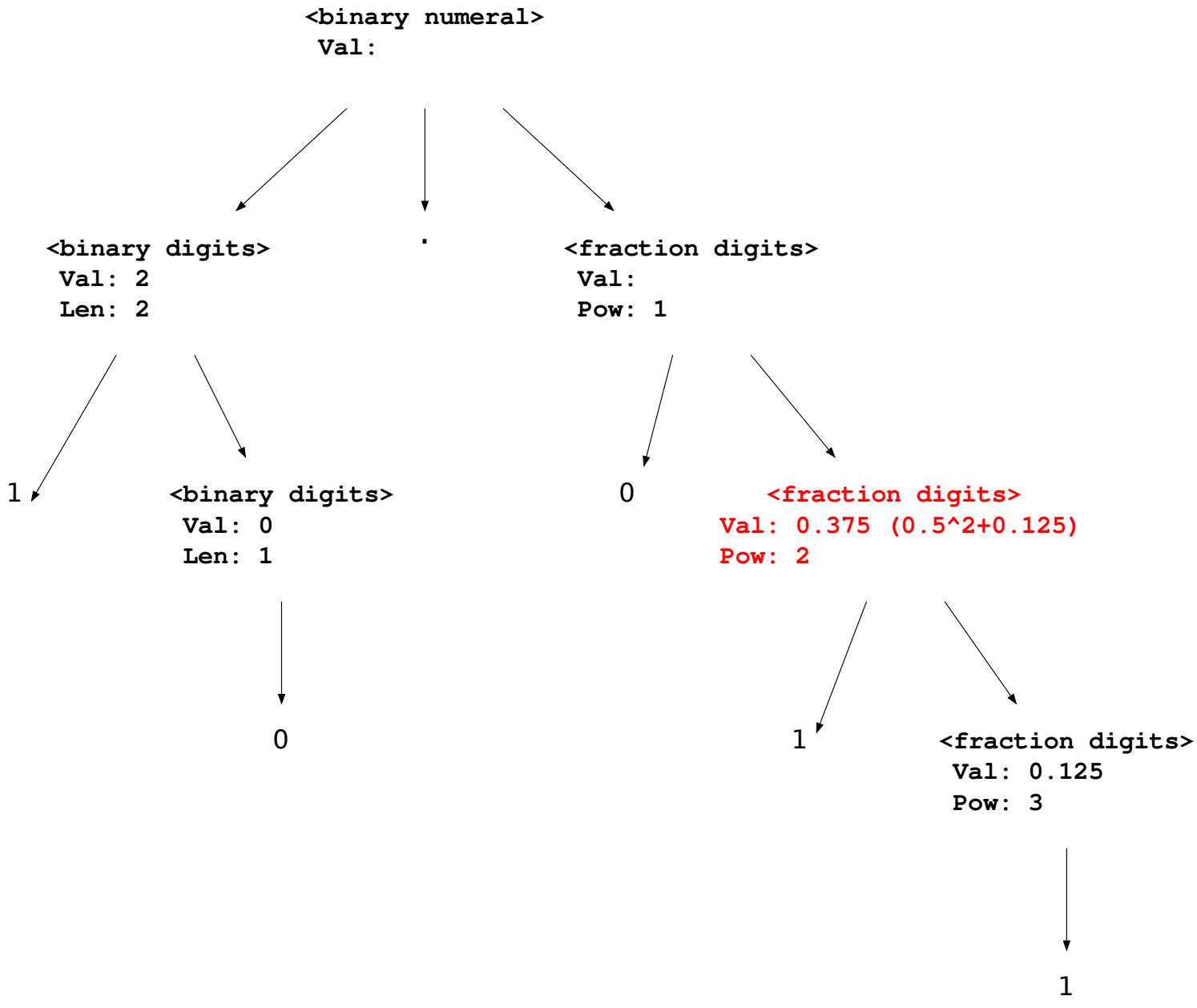

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2 (1+1)
.
<fraction digits>
Val:
Pow:
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
0
1
<fraction digits>
Val:
Pow:
1

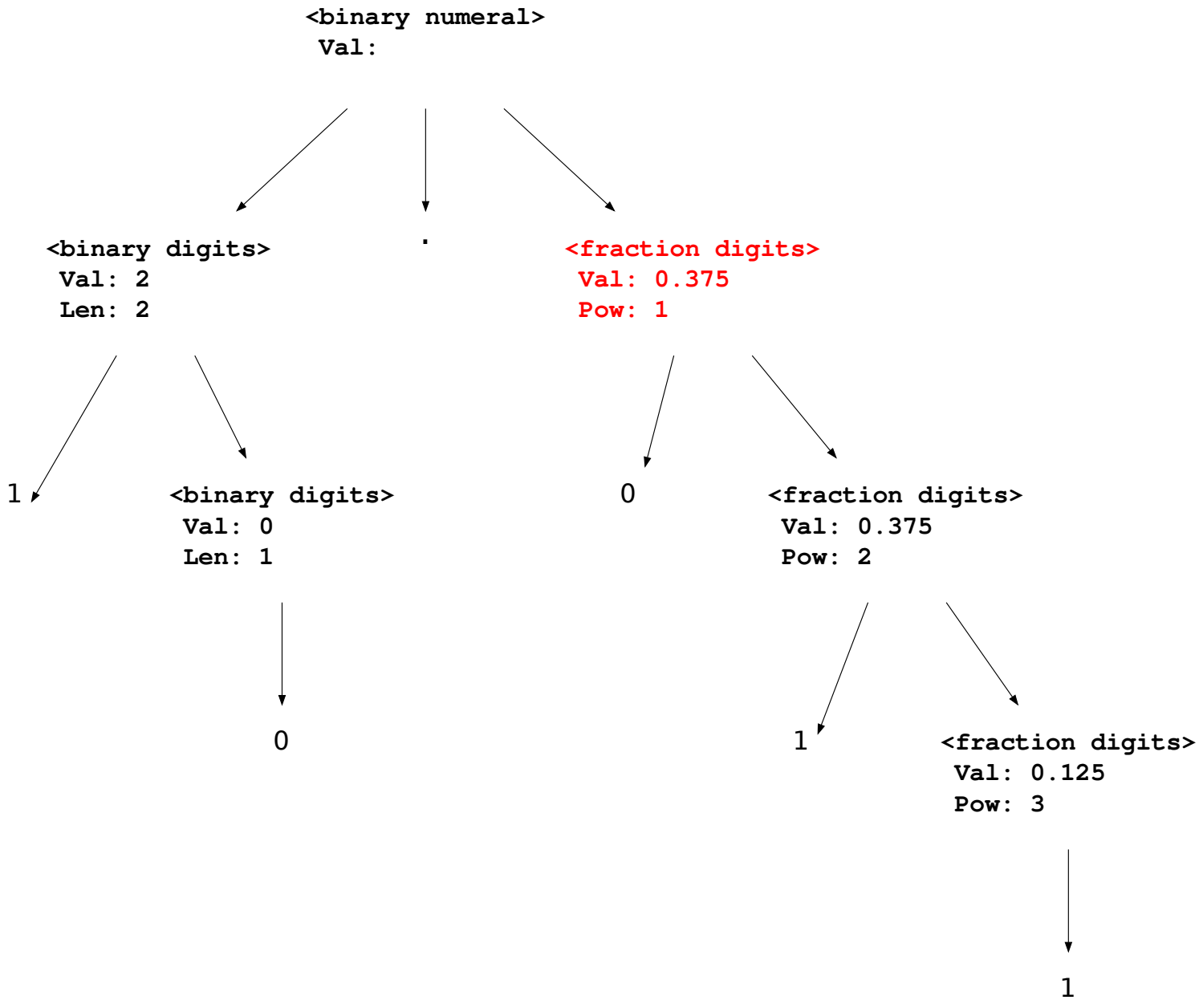

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow:
1
<fraction digits>
Val:
Pow:
1

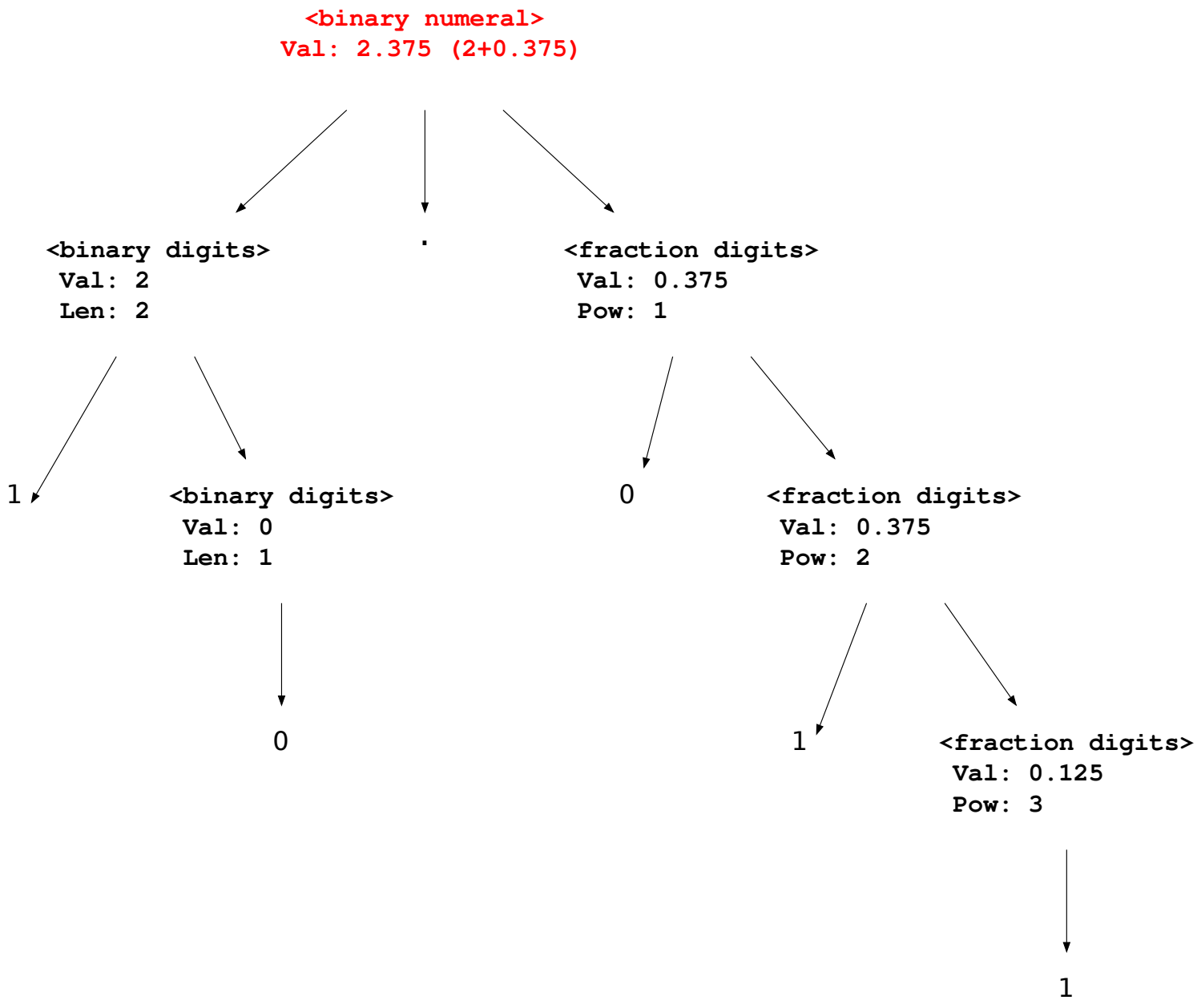

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow: 2 (1+1)
0
1
<fraction digits>
Val:
Pow:
1

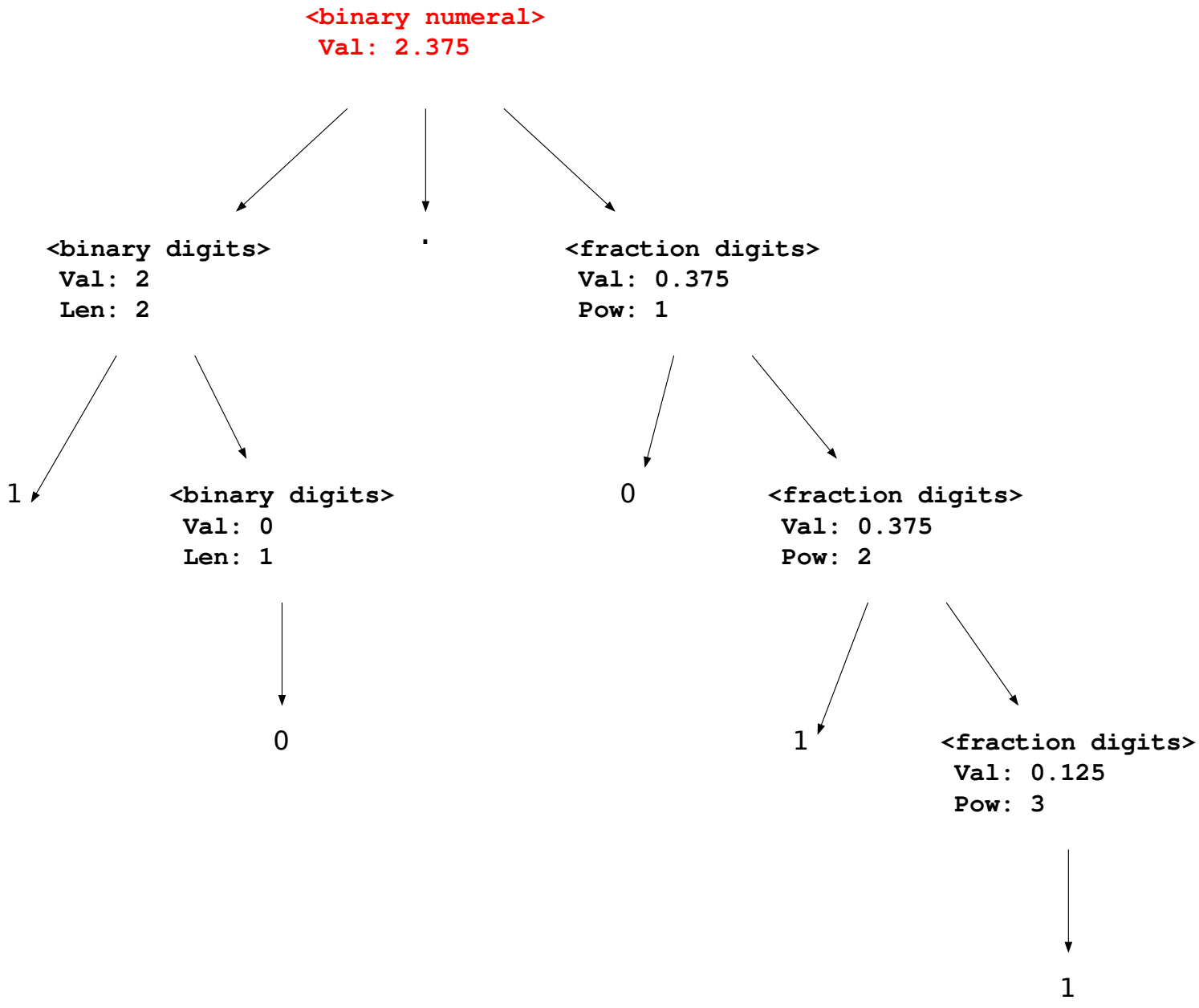

<binary numeral>
Val:
<binary digits>
Val: 2
Len: 2
.
<fraction digits>
Val:
Pow: 1
1
<binary digits>
Val: 0
Len: 1
0
0
<fraction digits>
Val:
Pow: 2
0
1
<fraction digits>
Val:
Pow: 3 (2+1)
1


<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.