GONDOLATOK A TÍPUS-SPECIFIKÁCIÓK KOMPATIBILITÁSÁNAK VIZSGÁLATÁRÓL

Eszmefuttatásom célja egy módszert vázolni, amely perspektivikus lehet annak vizsgálatában, hogy a típusok, típusosztályok algebrai, valamint a „hagyományosabb" külső felület szerinti specifikációjának[1] egymáshoz illeszkedése nem sérül-e. A módszer egyik sarkalatos pontja a Prolog nyelv felhasználása, amely –elképzeléseim szerint– képes támogatni eme vizsgálatot. Elvárásom, hogy egy alkalmasan felépített Prolog nyelvű programmal igazolható, a típus két leírásának kompatibilitása: logikai egymáshoz illeszkedősége.

Egy ilyen módszer kidolgozásának elsősorban gyakorlati jelentősége van. Ismert [KV2001] *ugyanis, hogy* ***algebrai specifikációból formálisan is előállíthatók az érintett operációk elő- és utófeltételes specifikációi****, tehát a típusosztály P/P-specifikációja. A programfejlesztés során a programozó nem az említett tétel alkalmazásával (még ha tud is róla), hanem inkább „józanésszel" jut el a típusosztály-megvalósításhoz közelebb álló (azaz egyéb operációktól függetlenül megfogalmazott) másik specifikációjához. Ekkor nagyobb „űr" tátong a kétféle specifikáció között, ami fölveti a kompatibilitás meglétének kérdését.*

Egy ilyen módszer más szempontból is hasznos lehet. Ha ugyanis valóban „mechanikussá" sikerül tenni a kompatibilitás vizsgálatát, jó eséllyel akár programmal is elvégezhetővé válik az ellenőrzés.

A harmadik részben további arra keressük a választ, mit ígérhet és mit nem a módszer, mit várhatunk el tőle, és miben ne reménykedjünk. Igyekszem összefoglalni azokat a tudnivalókat, amelyek a gyakorlatból szűrhetők le, de foglalkozok az elméleti alapokkal.

---

[1] Szokás az irodalomban ezt P/P-specifikációnak is nevezni, mivel az ***p***re- és ***p***ostcondition, vagyis az elő- és az utófeltétel megadását jelenti.

III. RÉSZ
A MÓDSZER PERSPEKTÍVÁJA

VÁZLAT:

0. A MÓDSZER MATEMATIKAI MODELLJE

Ebben a részben a módszer teljesítőképességét vesszük górcső alá. Megvizsgáljuk a működésének elméleti alapjait, és alkalmazásának gyakorlatát.
Az elméleti taglaláshoz néhány alapfogalomra lesz szükségünk. Nagyban építünk a [KV2001] irodalomra, így jónéhány fogalmat definiálás nélkül átveszünk. (Pl. típusosztályhoz kapcsolódó szignatúra, specifikáció absztrakt fogalmait.)

Definíció – P/P-specifikáció

Egy TIP *tipusosztály* ***P/P-specifikációja*** *alatt a* $\text{P/P-spec}_{TIP}=\{\Sigma,E_{PP}\}$*, ahol* $\Sigma=(S,OP)$ *szignatúra és* $E_{PP}=\{(Ef_i,Uf_i)\ i=1..\|OP\|\}$ (Ef_i,Uf_i) *elő- és utófeltételek véges halmazát értjük,* $\|OP\|$ *a* TIP *operációinak a számát jelenti.*

Értelemszerűen az (Ef_i,Uf_i) elő- és utófeltétel-pár a TIP i. operációjához tartozik. (Arról azonban tudjunk, hogy minden operáció előfeltétele az implicit error-axiómával együtt ki kell, adja a bemeneti értékek összességét. Azaz valójában nincs az operáció használata korlátozva, de lényegi transzformációra csak a fenti előfeltételek teljesülése mellett számíthatunk. Tegyük ehhez hozzá azt, hogy ha az operátor `NemDef` értéket állít elő, akkor a bemeneti értékeket változatlan értékben meghagyja.)
Például:

L. sor exportmodulját!

Definíció – algebrai specifikáció

Egy TIP *tipusosztály* ***algebrai specifikációja*** *alatt a* $\text{Alg-spec}_{TIP}=\{\Sigma,E_{ax}\}$*, ahol* $\Sigma=(S,OP)$ *szignatúra és* $E_{ax}=\{Ax_i\ i=1..N\}$ Ax_i *axiómák véges halmazát értjük.*
Az axióma egy leképezés a típus bázishalmazáról (T_0*) a logikai értékek (***L***) halmazára:*
$Ax: T_0 \rightarrow \mathbf{L}$

Az axiómák tartalmazzák a TIP-hoz rendelt operációk közül számosat. Ezek kapcsolatait írják le. Vizsgálatunk tárgya, hogy az operációk működését leíró P/P-specifikációk jól illenek-e bele az axiómák rendszerébe. Értelmes felvetés tehát egy axióma igazságtartalmára rákérdezni. Vagyis egy adott bázishalmazbeli értékből kiindulva az egyes axiómák igazak-e, amint „megszólaltatják” és kapcsolatba hozzák egymással a bennük foglalt műveleteket.
Például:

L. sor algebrai specifikációját!

Definíció – Operátor P/P-kiértékelése

Adott egy TIP *tipusosztály* $\text{P/P-spec}_{TIP}=\{\Sigma,E_{PP}\}$ *P/P-specifikációja. Az* $(Ef_i,Uf_i) \in E_{pp}$ *elő- utófeltétellel adott operációjának* $t \in T_0$ *melletti* **P/P-kiért(op,t)** *kiértékelése alatt azon értékek halmazát értjük (*$\subseteq \boldsymbol{R}_{op}$*), amelyet úgy kapunk, hogy bármely, az előfeltételt teljesítő paraméterezése és az adott t mellett az operátor kimeneti*

értékei az utófeltételt kielégítik. Azaz
ha $op:T_0\mathbf{x}T_{i_1}\mathbf{x}\ldots\mathbf{x}T_{i_j}\rightarrow T_0\mathbf{x}T_{i_k}\mathbf{x}\ldots\mathbf{x}T_{i_m}$,
akkor P/P-kiért(op,t):=$(tt, tt_1,\ldots,tt_m)\in \boldsymbol{R}_{op}$ $(=T_0\mathbf{x}T_{i_k}\mathbf{x}\ldots\mathbf{x}T_{i_m})$:
$\forall(t_1,\ldots,t_j)\in T_{i_1}\mathbf{x}\ldots\mathbf{x}T_{i_j}$: $Ef_{op}(t, t_1,\ldots,t_j) \Rightarrow Uf_{op}((tt, tt_1,\ldots,tt_m), (t, t_1,\ldots,t_j))$
A $\{(t_1,\ldots,t_j)\in T_{i_1}\mathbf{x}\ldots\mathbf{x}T_{i_j}: Ef_{op}(t, t_1,\ldots,t_j)\}$ *halmazt nevezzük az operátor t melletti tartójának. S jelöljük így*: **Tartó(op,t)**,

$(tt, tt_1,\ldots,tt_m)=op(t, t_1,\ldots,t_j)$, azaz $(tt, tt_1,\ldots,tt_m)$ jelöli az operátor által transzformált értéket a $(t, t_1,\ldots,t_j)$ helyen; továbbá a $T_{i_1}\mathbf{x}\ldots\mathbf{x}T_{i_j}$ halmaz a paramétereinek a halmaza. A tartó tehát azon halmaz, amelybe tartozó paraméterek estén elvárható az operátor utófeltétel szerinti működése.

Például:

A sor bázishalmazát jelöljük `S`-sel, ami `Elem`-iterált – amint ez az exportmodul nyitó részéből derül ki. A `Sorból` és az `Üres?` operációk szignatúrája az algebrai specifikációban szereplőtől némileg eltér. Pontosítottuk az exportmodulban olvashatók szerint: hozzávettük a definiáltság-állapotát jelző {`Def`, `NemDef`} komponenst „hiba” néven. Ezt mint *latens* paramétert tekintjük; azaz a sor belső komponenseként használjuk, az operátorok paraméterei között explicite nem, csak a sor részeként fog megjelenni. Emiatt az axiómákat újrafogalmazni kényszerülünk.

```
   Sorból: Sor x hiba → Sor x hiba x Elem [1]
   Üres?: Sor x hiba → Sor x hiba x L
ahol
   hiba={Def,NemDef}
   L={Igaz,Hamis}
```

Exportmodulbeli P/P-specifikációik:

```
   Eljárás Sorból(Változó s:Sor, e:ElemTip)
                                          [a sor első elemének kivétele]
      Ef:  ∃s∈Sor ∧ ¬s.hiba
      Uf:  s.sorozat=(s1,...,sn) ⇒ s'=((s2,...,sn),Hamis) ∧ e=s1
           s=() ⇒ s'.hiba=Igaz
   Függvény Üres?(Konstans s:Sor): Logikai
                                                    [üres-e a Sor]
      Ef:  ∃s∈Sor ∧ ¬s.hiba
      Uf:  Üres?(s)=(s.sorozat=())
```

Lássunk néhány P/P-kiértékelést:

1. mintastruktúra a `Sorból` operációhoz: ( ) – üres sorozat

a) Tartó(Sorból, ( )):={Def}

[1] A definiáltságot valójában *logikai*ként kezeljük, ugyanis értelmezés nélkül alkalmazzuk rá pl. a *negáció* műveletét. A könnyebb követhetőség miatt neveztük át konstansait `Def`-re , illetve `NemDef`-re `Hamis`, illetve `Igaz` értelemben.

ui.: $Ef_{Sorból}((\ ), Def)=Igaz$, $Ef_{Sorból}((\ ), NemDef)=Hamis \Rightarrow$ Tartó(Sorból, ( )):={Def}

(azaz csak a Def paraméterre teljesül az előfeltétel, 1-elemű a tartó az üres sorozatra)

b) P/P-kiért(Sorból, ( )):={(( ),NemDef,e) : ∀e∈Elem }

ui.: ∀h∈Tartó(Sorból, ( )), a)⇒ ∃!h=Def

(szavakkal: az üres sorozat mellett, elegendő csak a Def-re meghatározni azt a kimeneti halmazt, amelyre az utófeltétel igaz)

$Uf_{Sorból}((ss,hh,ee), (( ),Def))=Igaz \Leftrightarrow ss=(\ ) \wedge hh=NemDef \wedge ee=e\ \forall e\in Elem$

(szavakkal: az utófeltétel igaz lesz NemDef állapot és üres sorozat mellett tetszőleges elemre).

2. mintastruktúra a `Sorból` operációhoz: (e) – egy elemű sorozat

a) Tartó(Sorból, (e)):={Def}

ui.: $Ef_{Sorból}(((e), Def))=Igaz$, $Ef_{Sorból}((e), NemDef)=Hamis \Rightarrow$ Tartó(Sorból, (e)):={Def}

b) P/P-kiért(Sorból, (e)):={(( ),Def,e)}

ui.: ∀h∈Tartó(Sorból, (e)), a)⇒ ∃!h=Def

$Uf_{Sorból}((ss,hh,ee), ((e),Def))=Igaz \Leftrightarrow ss=(\ ) \wedge hh=Def \wedge ee=e;$

3. mintastruktúra az `Üres?` operációhoz: ( ) – üres sorozat

a) Tartó(Üres?, ( )):={Def}

ui.: $Ef_{Üres?}((( ), Def))=Igaz$, $Ef_{Üres?}((( ), NemDef))=Hamis \Rightarrow$ Tartó(Üres?, ( )):={Def}

b) P/P-kiért(Üres?, ( )):={(( ),Def,Igaz)}

ui.: ∀h∈Tartó(Üres?,( )), a)⇒ ∃!h=Def

$Uf_{Üres?}((ss,hh,ll), (( ),Def))=Igaz \Leftrightarrow ss=(\ ) \wedge hh=Def \wedge ll=Igaz;$

4. mintastruktúra az `Üres?` operációhoz: (e) – egy elemű sorozat

a) Tartó(Üres?, (e)):={Def}

ui.: $Ef_{Üres?}(((e),Def))=Igaz$, $Ef_{Üres?}(((e), NemDef))=Hamis \Rightarrow$ Tartó(Üres?, (e)):={Def}

b) P/P-kiért(Üres?, (e)):={((e),Def,Hamis)}

ui.: ∀h∈Tartó(Üres?, (e)), a)⇒ ∃!h=Def

$Uf_{Üres?}((ss,hh,ll), ((e),Def))=Igaz \Leftrightarrow ss=(e) \wedge hh=Def \wedge ll=Hamis;$

5. … egy nem ennyire triviális példa

…

Definíció – Axióma kiértékelése

Adott egy TIP *tipusosztály* Alg-spec$_{TIP}$={Σ,E$_{ax}$} *algebrai specifikációja és* P/P-spec$_{TIP}$={Σ,E$_{PP}$} *P/P-specifikációja. Az* Ax∈E$_{ax}$ *axiómájának* t∈T$_0$ *melletti* Ax-kiért(Ax,t) *kiértékelése alatt azon logikai érték~~(ek)~~et értjük, amelyet úgy kapunk, hogy*

a.) a logikai műveletek kiértékelési sorrendjének megfelelően sorbavesszük az operátorokat, abból a célból, hogy

b.) külön-külön kiértékeljük; természetesen az elsőt az adott t-re, a továbbiakat olyan paraméterekre, amelyeket az addigi kiértékelések meghatároznak;

c.) majd figyelembe vesszük az előírt logikai műveleteket, megkapjuk a végeredményt.

Például:

Az eredeti 4'° axióma:

4'° Sorból – hiba-axióma *Üres?(s) ⇒ Sorból(s)=NemDef*

… és az újrafogalmazott változata:

4'° Sorból – hiba-axióma *Üres?(s).L ⇒ Sorból(s).hiba=NemDef*

1. mintastruktúra a 4'° axiómához: ( ) – üres sorozat

Ax-kiért(4'°, ( )) ≡ …

a) A kiértékelés sorrendje és a művelet paramétere:

1. Üres? – ( )
2. Sorból – P/P-kiért(Üres?, ( )) meghatározta Sor-komponens

b) A korábbi példákat alapul véve:

P/P-kiért(Üres?, ( )):={(( ),Def,Igaz)},

Ennek Sor-komponense: ( ), ezért

P/P-kiért(Sorból, ( )):={(( ),NemDef,e) : ∀e∈Elem }.

Így

Ax-kiért(4'°, ( )) ≡
≡ Üres?((( ),Def)).L ⇒ Sorból((( ),Def)).hiba=NemDef ≡
≡ Igaz ⇒ NemDef=NemDef ≡
≡ Igaz

2. mintastruktúra a 4'° axiómához: (e) – egy elemű sorozat

Ax-kiért(4'°,((e),Def)) ≡ …

a) A kiértékelés sorrendje és a művelet paramétere:

1. Üres? – (e)
2. Sorból – P/P-kiért(Üres?, (e)) meghatározta Sor-komponens

b) A korábbi példákat alapul véve:

P/P-kiért(Üres?, (e)):={((e),Def,Hamis)},

Ennek Sor-komponense: (e), ezért

P/P-kiért(Sorból, (e)):={(( ),Def,e)}.

Így

Ax-kiért(4'°,((e),Def)) ≡

≡ Üres?(((e),Def)).L ⇒ Sorból(((e),Def)).hiba=NemDef ≡

≡ Hamis ⇒ ((),Def)=NemDef ≡

≡ Igaz

Definíció – mintastruktúrák halmazának Ax-szerinti partícionálása

Adott egy TIP *tipusosztály* P/P-spec$_{TIP}$={Σ, E$_{PP}$} *P/P-specifikációja és az* Alg-spec$_{TIP}$={Σ, E$_{ax}$} *algebrai specifikációja. A* $t_1,t_2 \in T_0$ *ugyanabba* T_0*–partícióba tartozik egy adott* $Ax \in E_{ax}$ *szerint,* ~~*ha az* **Ax-kiért(Ax,t_1)≡Ax-kiért(Ax,t_2)**~~*.*

Jelöljük egy $t \in T_0$*-t tartalmazó* ***partíció****t:* Partíció(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax,t)*;*

és a ***reprezentánsok halmazá****t:* Reprezentáns(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax)*, amelyre igaz, hogy* $\forall t_1,t_2 \in T_0$: $t_1 \neq t_2 \wedge$

Partíció(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax,t_1)≠Partíció(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax,t_2).

Így elegendő mindig csak a partíció valamely reprezentánsát vizsgálni. Értelmes általánosítása ezeknek, a minden axiómát figyelembe vevő partícionálás és reprezentánsok halmaza.

Például:

t_1=(( ),Def), t_2=((e),Def), $t_1,t_2 \in T_0$: $t_1 \neq t_2$

(előző példák) ⇒

Ax-kiért(4'°,t_1)=Ax-kiért(4'°,(( ),Def))=Igaz ∧

Ax-kiért(4'°,t_2)=Ax-kiért(4'°,((e),Def))=Igaz

⇒

Partíció(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax,t_1)=Partíció(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax,t_2).

Definíció – P/P-specifikáció kompatibilitása az Ax axiómával

Egy TIP *tipusosztály* P/P-spec$_{TIP}$={Σ,E$_{PP}$} ***P/P-specifikációja kompatíbilis*** *az* Alg-spec$_{TIP}$={Σ, E$_{ax}$} *algebrai specifikáció egy* **Ax∈E$_{ax}$** ***axiómájával****, ha* ∀*t*∈*Reprezentáns(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$,Ax) mintastruktúra esetén Ax kiértékelése igaz értékre vezet.*

Például:

Definíció – P/P-specifikáció kompatibilitása az algebrai specifikációval

Egy TIP *tipusosztály* P/P-spec$_{TIP}$={Σ,E$_{PP}$} ***P/P-specifikációja kompatíbilis*** *az* Alg-spec$_{TIP}$={Σ, E$_{ax}$} ***algebrai specifikációjával****, ha* ∀*t*∈*Reprezentáns(P/P-spec$_{TIP}$,Ax-spec$_{TIP}$) mintastruktúra esetén minden axióma kiértékelése igaz értékre vezet.*

Például:

1. A MEGBÍZHATÓSÁG

1.1. A megbízhatóság fogalma

A megbízhatóságon két dolgot értünk: egyrészt, ha a program elfogad egy axiómát jónak, akkor mennyire hihetünk eben; másrészt ha hibásnak jelez egy axiómát, akkor az valóban inkompatibilitást jelez-e?
Az első részéhez keserű tapasztalatok fűződnek. Vannak „szerencsétlen interferencia” okozta kimutatlanul maradt hibák. Lássunk ezek közül néhányat!

1.2. Rejtett hibák

A verem példánál maradva, könnyű olyan hibás P/P-szabályt találni az `Üres` és az `ÜresE` operátorhoz, amelyek menthetetlenül elfedik egymás hibáját. [Verst3H1]

a hibás operációk szabályai[1]	*axióma*	*Prolog „lényegi” szabály*[2]
`ures(verem([],`***`nemDef`***`)).` `uresE(verem(_,nemDef),_).` `uresE(verem([],def),`***`hamis`***`).` `uresE(verem([_],def),hamis).` `uresE(verem([_|_],def),hamis).`	1º Üres – axióma *V=Üres ⇔ Üres?(V)*	`axioma1(V,igaz) if` `  ures(V) and uresE(V,igaz).` `axioma1(V,igaz) if` `  not(ures(V)) and` `  not(uresE(V,igaz)).` `axioma1(verem(_,nemDef),igaz).` `axioma1(_,hamis).`

A `verem([],def)` paraméterrel hívva az `axioma1` szabályt, az `ures(V)` illeszthetetlen volta miatt rögvest `fail`-lel visszalép, és a 2. alternatívára tér át. Itt a `not(ures(V))` `true` lesz, de nem várt ok miatt: nem az ürességet, hanem a definiáltságot kifogásolja a Prolog. Továbblépve a `not(uresE(V,igaz))` a hibás `uresE` axióma miatt lesz `true`. A végeredmény: az `axioma1` teljesül!
A `verem([],nemDef)` paraméterrel aktivizálva az `axioma1`-et azon csúszik el a hibaérzékelés, hogy a hibás `Üres` szabályon áthalad a vezérlés az `ÜresE` helyesen megfogalmazott implicit-error axióma miatt tényleges vizsgálódás nélkül jut túl.

Egy másik példában a `Tető` és a `Verembe` operációk interferálnak rosszul. [VerSt3H2]

a hibás operációk szabályai	*axióma*	*Prolog „lényegi” szabály*
`teto(verem(V,nemDef),_,` `  verem(V,nemDef)).` `teto(verem([E|V],def),E,` `  verem([E|V],def)).` `teto(verem([],def),_,` `  verem([],`***`def`***`)).` `verembe(verem(V,nemDef),_,`	3º Tető-Verembe – axióma *Tető(Verembe (v,e))=e*	`axioma3(V,igaz)if` `  verembe(V,E,W) and ! and` `  teto(W,E,W).` `axioma3(verem(_,nemDef),igaz).` `axioma3(_,hamis).`

[1] ***Kiemeltük*** a hibás paramétereket.
[2] Csak a „logikát” kódoltuk. Nem bonyolítottuk a vágással és a miatt szükséges al-szabályokra bontással; továbbá a paramétergeneráláshoz szükséges `sorozatok` részformula beillesztésével.

```
        verem(V,def)).
Verembe(verem(V,def),E,
        verem([E|V],def)).
```

A `verem([],def)` paraméterrel hívva az `axioma3` szabályt, minden joggal rendben fut le. A `verem([],nemDef)`-re viszont a válasz hibásan lesz igenlő. Ui.: a `verembe(V,E,W)` úgy tud illeszkedni a hibás az implicit-error axiómához, hogy közben a `Tető` hasonlóan hibás axiómája együttesen is elfogadhatóvá teszi az axiómát.

Az „abszolút” megbízhatóság elvileg sem biztosítható. Ezt mondja ki az alábbi állítás.

Állítás:

Ha a típus axiómái között található alábbi (*) *szerkezetű, akkor a benne szereplő operációknak van olyan átértelmezése –szemantikus módosítása–, amelyre az axióma változatlanul igaz marad.*

(*) op_k, $op_s \in OP$, $op_s: T_0 \rightarrow T_i \cup \{NemDef\}$, $op_k: \rightarrow T_0 \cup \{NemDef\}$, $A \in T_i$,

$Ax \in L$ – *az axióma „maradék” része, amely nem függ az* op_s*-től és az* op_k*-tól*

$op_s(op_k) = A \wedge Ax$

Bizonyítás:

Könnyen definiálható olyan

op_s': $T_0 \rightarrow T_i \cup \{NemDef\}$, $op_k': \rightarrow T_0 \cup \{NemDef\}$,

amelyre

$op_s' \neq op_s$, $op_k' \neq op_k$, *de* $op_s'(op_k') = A$.

Konstrukció:

Az operációkat P/P-specifikációval adjuk meg. Tehát:

P/P-spec(op_s):

$\forall t \in T_0$: $Ef_{ops}(t) \Rightarrow Uf_{ops}(op_s(t),t)$ – *ha az* Ef_{ops} *igaz, akkor az* Uf_{ops} *is az*

$\forall t \in T_0$: $\neg Ef_{ops}(t) \Rightarrow op_s = NemDef$ – *ha az* Ef_{ops} *hamis, akkor* NemDef

P/P-spec(op_k):

$Uf_{ops}(op_s)$

Legyenek a módosított operátorok az alábbi módon definiálva:

$t' \in T_0$: $Ef_{ops}(t')$

P/P-spec(op_s'):

$op_s'(t') = A$ – *átdefiniálva egy* t'*-re*

$\forall t \in T_0$: $t \neq t' \wedge Ef_{ops}(t) \Rightarrow Uf_{ops'}(op_s'(t),t) = Uf_{ops}(op_s(t),t)$

$\forall t \in T_0$: $\neg Ef_{ops}(t) \Rightarrow op_s' = NemDef$

P/P-spec(op_k'):

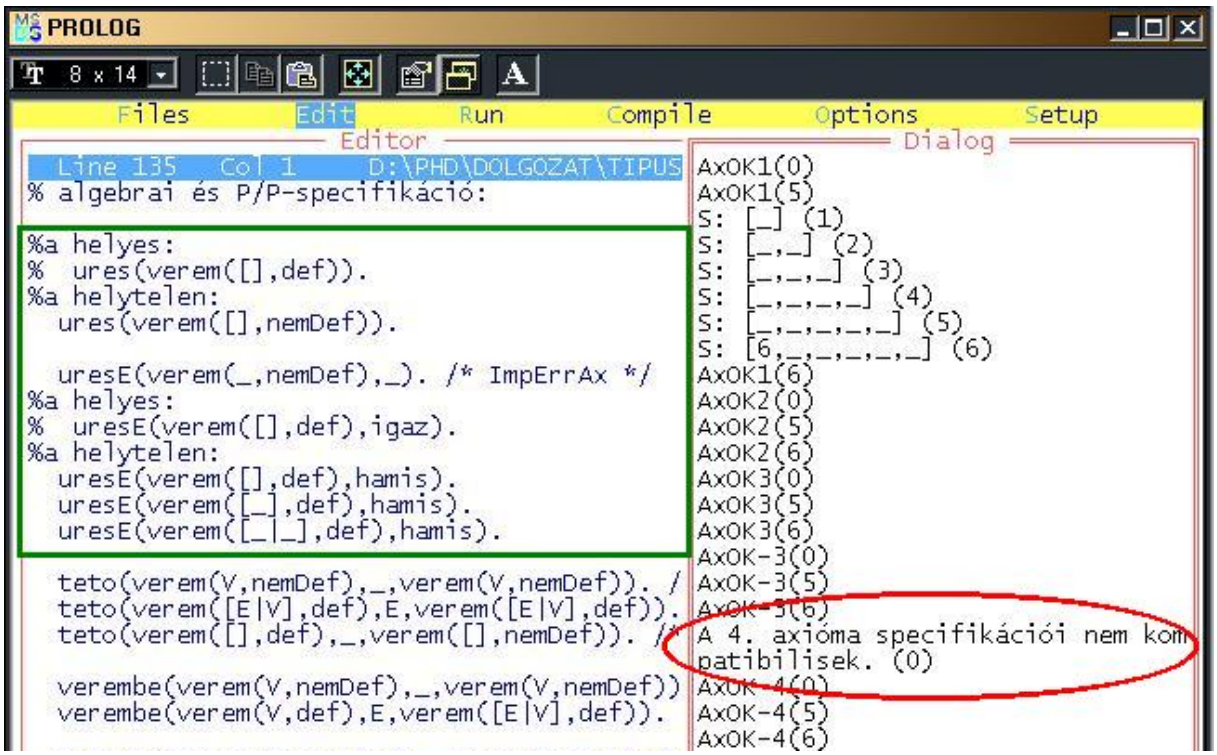
$op_k' = t'$ – *ez maga a módosított* $Uf_{opk'}$

Megjegyzem: a bizonyítás nyilvánvalóságot formalizál. Nem állítom, hogy a hibák képződésének „gyakorlatias” módját mintázza. Cél csak annyi volt, hogy beláttassa: rejtve maradó hibák létezhetnek.

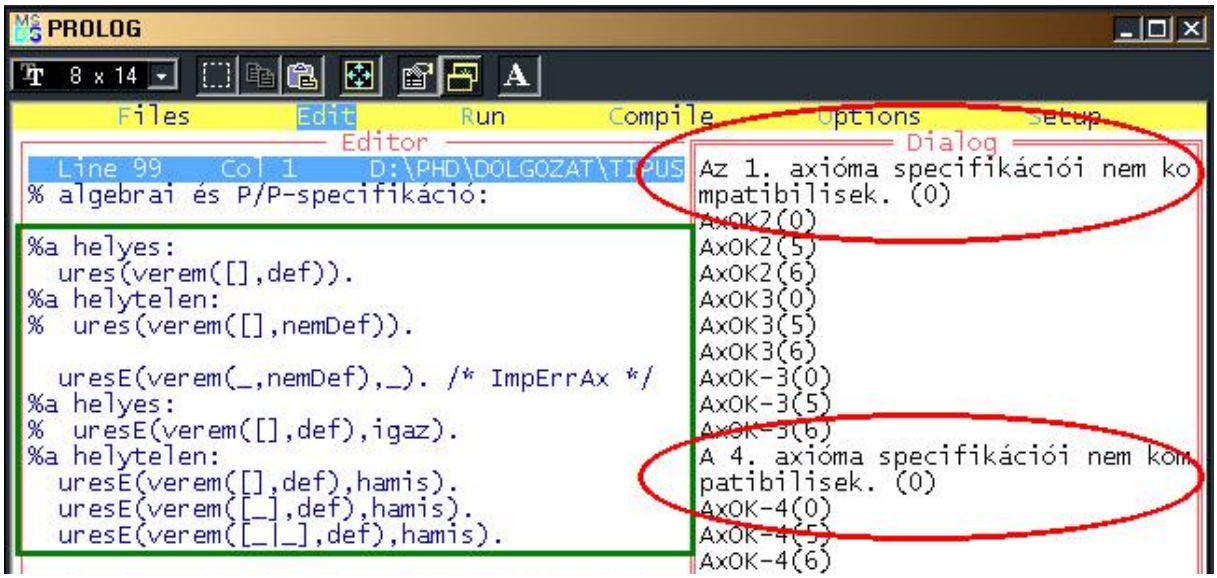
Az állítás után mire számíthatunk a módszer használhatóságát illetően?

Szerencsére ritka az olyan abszolút szerencsétlen interferencia, amely teljesen rejtve tud maradni. Ezt példázza a [Verst3H1] program is, amelyben ugyan az 1° axiómában szereplő két hibás operáció el tudott rejtőzni, de a 4° axiómában egy hiba mégis felszínre került. Ez alapján már gyanú ébredhet az `ÜresE` operációval szemben. Azt javítva, már az 1° axióma is jelzi a „maradék” problémát. L. az 1.1. és 1.2. ábrát!

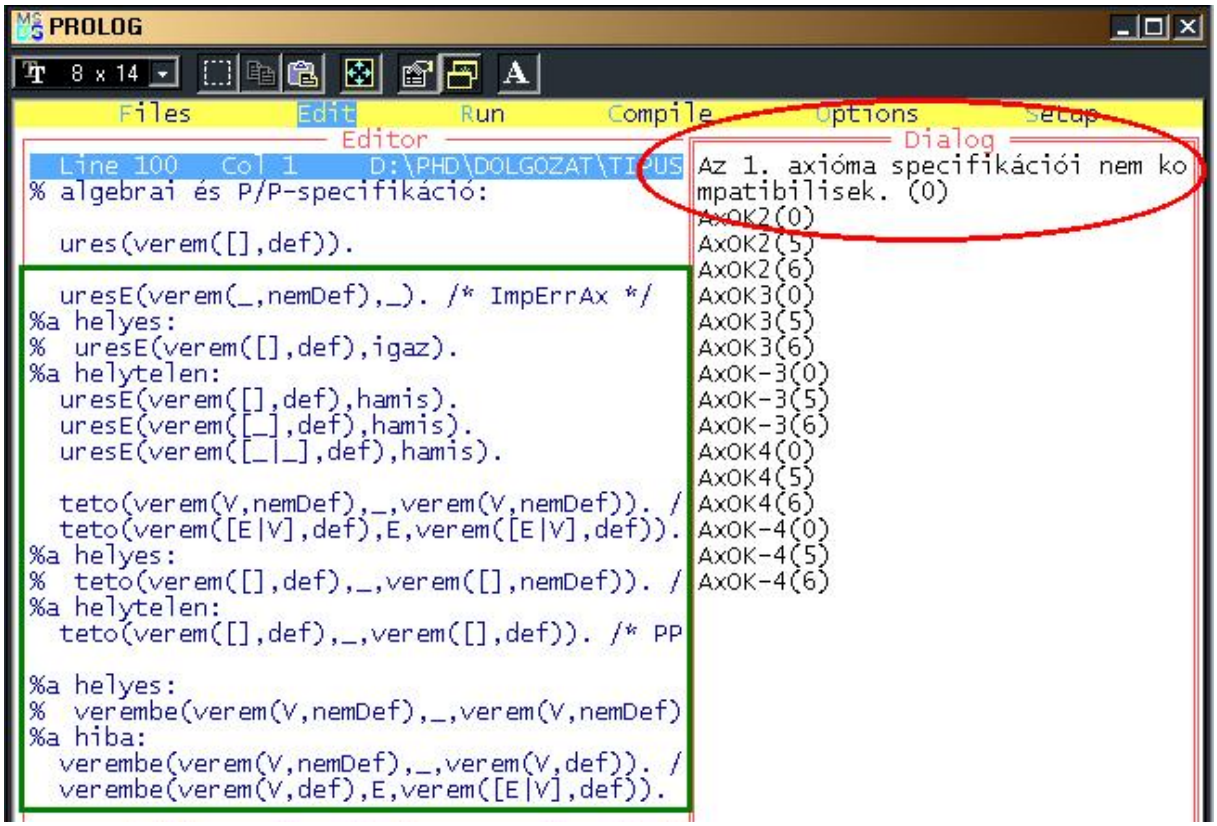
Hasonlóan tárulkozik fel az elfedődött hiba [VerSt3H2] program példájában. Kezdetben a három hiba 2 axiómában jelződik, de a 3° és a 3'° axiómában rejtve maradnak a hibás operátorok. L. az 1.3. és 1.4. ábrát! A hibajelzés után a gyanús `ÜresE` operációról bebizonyosodott a helytelen specifikáció, amit javítunk. A javítás után újabb hibajelzés, újabb javítás… A dolog érdekessége az, hogy a hibák javítása közben soha sem jutottunk olyan időleges állapotba, amely során az elemzett 3° axióma hibásnak mutatkozott volna. Ez nem is probléma, hisz a lényeg, hogy amíg hiba volt, addig volt hibásnak talált axióma is.



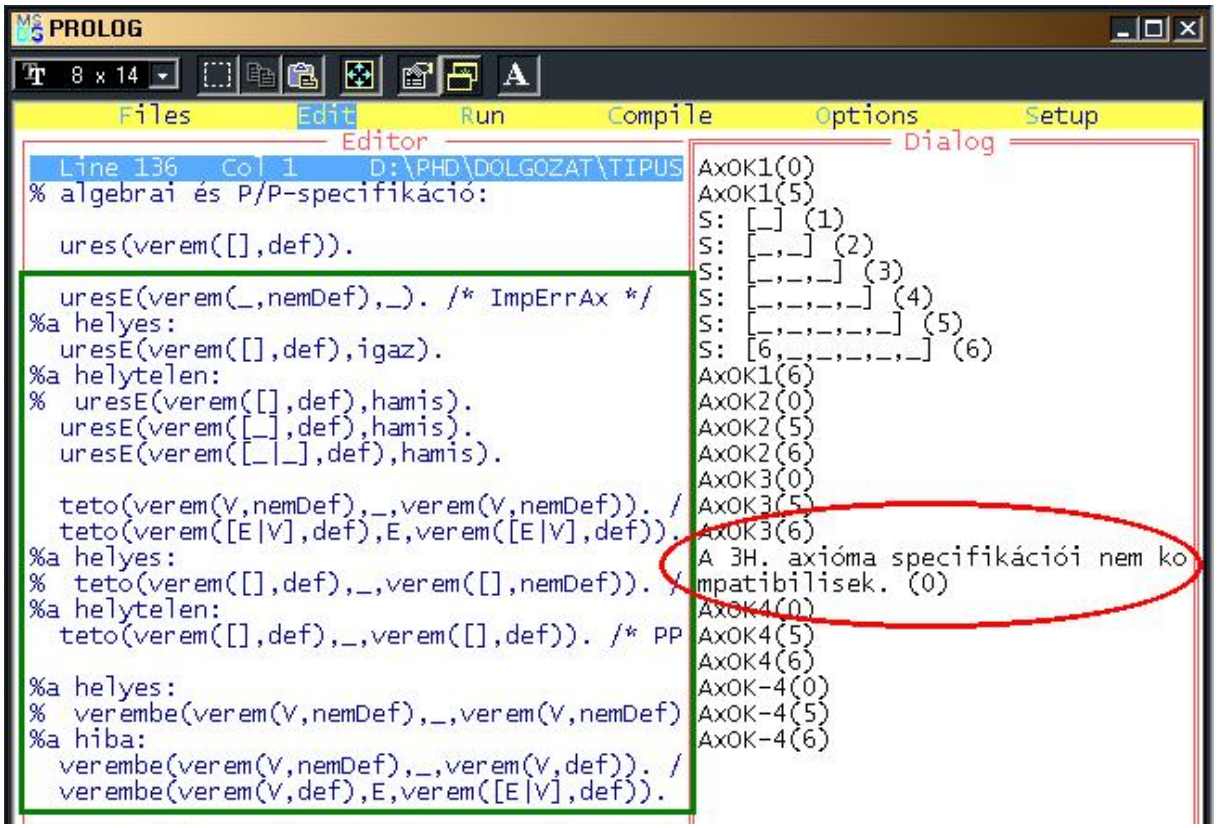
1.1. ábra – futási eredmény
a verem-specifikációk kompatibilitás ellen őrzése strukturális indukcióval –
az Üres és az ÜresE operációk kimutatat lan hibával



1.2. ábra – futási eredmény
a verem-specifikációk kompatibilitás ellen őrzése strukturális indukcióval –
az Üres hibájának megszüntetése után



1.3. ábra – futási eredmény
a verem-specifikációk kompatibilitás ellenőrzése strukturális indukcióval –
az ÜresE, a Tető és a Verembe operációk kimutatatlan hibával



1.4. ábra – futási eredmény
a verem-specifikációk kompatibilitás ellen őrzése strukturális indukcióval –
az ÜresE javítása után

Megállapíthatjuk, hogy a hibák rejtve maradása ellen úgy védekezhetünk, hogy minél inkább „redundánssá” tesszük az algebrai rendszert. Ennek egy kitűnő eszköze lehet a típusosztályt jellemző állítások kimondása, formalizálása, és a Prolog programba való beillesztése.

Amint már korábban hangsúlyoztuk, a hibák rejtve maradásának másik oka az lehet, ha nem megfelelően választjuk meg a típusosztályt tesztelő mintastruktúrákat. Itt vezérelvként jelentettük ki: *egy sorozattípus esetén minden közvetlenül hivatkozott elemnek legalább egy mintastruktúrában fel kell bukkannia*. Más szóval, ha van op_i operátor, amely a sorozat `i.` elemével tesz valamit (módosítja, beszúr elé/mögé, törli, vagy egyszerűen az értékét teszteli), kell legalább egy legalább i db elemet tartalmazó mintasorozat.

1.3. Hamis riasztás

A módszer lényegéből fakadóan, feltéve a P/P- és az algebrai specifikáció precíz kódolását, csak olyan axiómák esetében jelez „fölöslegesen” inkompatibilitást, ha az axióma nem fedi le a „teljes esemény rendszert”. Vagyis hiányzik olyan alternatíva, amely nem képes „elfogadólag nyilatkozni” valamely mintastruktúrára, s ez okozhat téves hibajelzést

2. A TELJESSÉG

Teljesség: az „intuitíve” elvárt minden elem előállítható-e az axióma rendszer leírt műveletek által? E kérdés tehát nem a módszerre vonatkozik, hanem magára az axióma renszerre. Érdekes meggodolni azért, hogy mi módon lehetne „algoritmikusan” ellenőrizni ezt a fajta teljességet.

3. PRAKTIKUS KÉRDÉSEK

Néhány olyan kérdést vetek föl az alábbiakban, amelyek a módszer kidolgozása közben vetődtek föl. Alighanem hasznosak lehetnek a mindennapi használat során. Kérdések:

1. Hogyan készítsük el a típusosztály *algebrai* specifikációját?
2. Hogyan készítsük el a típusosztály *P/P*-specifikációját?
3. *Többletállítások* a típusosztályhoz, invariáns állítások
4. Egy bizonyítás *nyomon követése* a Prolog adatbázis-kezelő szolgáltatására építve

4. EKVIVALENS FORMULÁK KERESÉSE

Az eddigi gondolatmenet egyik lényegi tulajdonsága az volt, hogy a kétféle rendszerben felállított állítások valamifajta kompatibilitását tudtuk kimutatni, feltéve, hogy megvolt. Most többre vágyunk, és az ekvivalencia reményével igyekszünk az algebraiból a P/P-rendszerbe áttérni. A [KV2001] irodalomban szereplő állítás erre jogalapot teremt.

Állítás:

Az algebrai specifikáció elő- és utófeltételes formára hozható.

Bizonyítás:

A bizonyítás konstruktív, amelyre építhetünk a példánk esetében..

Konstrukció:

…… a bizonyítás lépései …

Az eddigi kifogásunk a fenti átírással szemben az volt, hogy a P/P-specifikációban olyan hivatkozásokat tartalmaz (más operációkra), amelyek nem kívánatosak ilyen rendszerben. Ezen úgy segíthetünk, hogy a *sorozat-nyelv* [SzPG2002, PSzZs1998, SzP1999] elemeit fölhasználjuk, s mintegy a kapott formulák még nem kívánatos részleteinek megfeleltetjük a sorozat-nyelvbeli analogonját.

Figyeljük meg a verem esetén az áttérést! Időnként az axiómát kettébontjuk, s majd egyesítjük a specifikációdarabokat.

f_0 = Üres. I(v)= 0≤Hossz(v)[1]

Algebrai axióma	***P/P-specifikáció$_1$***	***P/P-specifikáció$_2$***
–	Üres: Ef: – Uf: 0≤Hossz(v) ∧ v=Üres	 Ef: – Uf: 0≤Hossz(v) ∧ v=()
Üres=v ⇔ Üres?(v)		
Üres=v ⇒ Üres?(v)	Üres?(v): Ef: Üres=v Uf: Üres?(v)	 Ef: v=() Uf: Üres?(v)

[1] A felülről korlátosságot csak bizonyos (folytonos) ábrázolás esetén kell elvárni.

Üres≠v ⇒ ¬Üres?(v)	Üres?(v): Ef: Üres≠v Uf: ¬Üres?(v)	 Ef: v≠() Uf: ¬Üres?(v)
	Üres?(v): Ef: Üres≠v Uf: Üres=v ⇒ Üres?(v) Üres≠v ⇒ ¬Üres?(v)	 Ef: – Uf: ()=v ⇒ Üres?(v) ()≠v ⇒ ¬Üres?(v)

Az axiómát két részre vágtuk, majd a egyesítettük az átírás eredményeit.

¬ Üres?(Verembe(v,e))	Verembe(v,e): Ef: 0≤Hossz(Verembe(v,e)) Uf: 0≤Hossz(w) ∧ w=Verembe(v,e)	Verembe(v,e): Ef: 0≤Hossz(v&(e)) Uf: 0≤Hossz(w) ∧ w=Verembe(v,e)

Látható nehézség: a Verembe konstrukciós művelet szempontjából nem veszi számításba az axióma tényleges mondanivalóját. Ha az Üres? szelekciós művelet szempontjából tekintenénk, akkor semmivel több információhoz nem jutnánk, mint amennyihez a korábbi axióma átírásával jutottunk.

Tető(Verembe(v,e))=e	Tető(v): Ef: w=Verembe(v,e) Uf: Tető(w)=e	 Ef: – Uf: Tető(v&(e))=e

A sorozatnyelvre áttérés valójában két lépésben történt. Az első:

Ef: w=v&(e)

Uf: Tető(w)=e

Majd a w kiküszöbölésével kapjuk a végső átírtat.

IRODALOM

[KV2001]

Kozma L. – Varga L.: „Adattípusok osztálya.”, ELTE TTK Informatikai TanszékCsoport, 2001

[SzPG2002]

Szlávi P. – Pap Gné: „Adatszerkezet – Típusok”, Informatika a felsőoktatásban’99 konferencia, Debrecen, 2002

[PSzZs1998]

Pap Gné – Szlávi P. – Zsakó L.: „Módszeres programozás – Adattípusok”, ELTE TTK Informatikai TanszékCsoport, 1998

[SzP1999]

Szlávi P.: „Programok, programspecifikációk.”, Informatika a felsőoktatásban’99 konferencia, Debrecen, 1999

PROGRAMHIVATKOZÁSOK

[VerSt3H1]

Szlávi P.: „A verem specifikációinak kompatibilitását ellenőrző Prolog program – strukturális indukció alapján; hibás Tető és Verembe operáció esetén.”,.

http://izzo.inf.elte.hu/szlavi/Doktori/TipSpecKomp/VERST3H2.PRO

[VerSt3H2]

Szlávi P.: „A verem specifikációinak kompatibilitását ellenőrző Prolog program – strukturális indukció alapján; hibás Tető és Verembe operáció esetén.”,.

http://izzo.inf.elte.hu/szlavi/Doktori/TipSpecKomp/VERST3H2.PRO

FÜGGELÉK