

# **TDK dolgozat**

Adrián Patrik



**DEBRECENI  
EGYETEM**

Informatikai Kar  
Számítógéptudományi Tanszék

# Szintaktikai elemzők erős negációs szemantikájú logikai nyelvtanokhoz

Témavezető:

dr. Vaszil György

egyetemi tanár

Szerző:

Adrián Patrik

programtervező informatikus MSc, 2. évf.

Debrecen

2023



## Kivonat

A formális nyelvek elmélete a számítástudomány egyik alapvető kutatási területe, melynek gyakorlati alkalmazásai jelentős szerepet töltenek be az informatika számos területén, többek között a fordítóprogramok és értelmezők alapját adó szintaktikai elemzésben is.

A Chomsky nevéhez kötődő környezetfüggetlen nyelvtanok ezen a területen vitathatatlan népszerűségnek örvendenek, köszönhetően többek között könnyen érthető szemantikájuknak. Noha időközben a formális nyelvek tudománya nyelvek és nyelvtanok számos fajtájával gazdagodott, ezek mind inkább megmaradtak kutatási érdekességnek, mintsem hogy széles körben alkalmazott eszközökké váljanak.

A környezetfüggetlen nyelvtanok meglévő műveleteinek (unió, konkatenáció) a konjunkció (metszet), majd a negáció (komplementum) műveletével való kibővítésével A. Okhotin foglalkozott részletesen ([15], [17]). Ezek a logikainak (Boolean) nevezett nyelvtanok szigorúan erősebbek, mint környezetfüggetlen elődeik, ugyanakkor új problémákat is felvetnek a modellezhetőség és egyértelműség vonatkozásában.

A logikai nyelvtanok szemantikai modellezése nehéz feladat, mivel a negáció bevezetésével lehetőség nyílik önellentmondást leíró szabályok megfogalmazására. Kountouriotis és mások ([20]) ezen hiányosság feloldására a A. V. Gelder és mások ([13]) által kidolgozott, háromértékű logikára épülő megalapozott (well-founded) szemantika alkalmazását javasolta logikai nyelvtanok kiértékeléséhez.

Jelen munkában ugyancsak a logikai nyelvtanok ugyancsak háromértékű szemantikájával foglalkozunk a klauzális logikai programok Fitting-féle szemantikájára ([11]) építkezve. Ehhez kapcsolódóan megmutatjuk, hogyan alakíthatók át a szintaktikai elemzők „nagy családjai” a logikai nyelvtanok ezen modelljeinek kiszámítására.



# Tartalomjegyzék

|                                                             |           |
|-------------------------------------------------------------|-----------|
| <b>1. Logikai alapok</b>                                    | <b>3</b>  |
| 1.1. Logikai programok . . . . .                            | 5         |
| 1.2. Parciális interpretációk . . . . .                     | 11        |
| <b>2. Formális nyelvek a háromértékű logikában</b>          | <b>17</b> |
| 2.1. Háromértékű nyelvek . . . . .                          | 18        |
| 2.2. Logikai nyelvtanok . . . . .                           | 22        |
| <b>3. Szintaktikai elemzés</b>                              | <b>31</b> |
| 3.1. Felülről lefelé (top-down) tabuláris elemző . . . . .  | 34        |
| 3.2. Alulról felfelé (bottom-up) tabuláris elemző . . . . . | 36        |
| 3.3. Olvasásirányú online elemző . . . . .                  | 38        |
| 3.4. Visszirányú online elemző . . . . .                    | 43        |
| <b>4. Összefoglalás</b>                                     | <b>51</b> |
| 4.1. Saját eredmények . . . . .                             | 52        |



# Bevezetés

A formális nyelvek leírásának legelterjedtebb megközelítése az átírási rendszerek irányából való, ezen formalizmusok közül is kiemelkednek a Chomsky-féle környezetfüggetlen nyelvtanok (CFG, [1]), melyek széleskörű alkalmazhatóságuknak és intuitív szemantikájuknak köszönhetően a formális nyelvek specifikációjának szinte egyeduralgó eszközeivé lettek. A környezetfüggetlen nyelvtanok általánosítására rengeteg módszer született: Chomsky környezetfüggő nyelvtanaitól kezdve a Joshi-féle TAG-okon (tree adjoining grammar, [7]) és az Aho-féle indexált nyelvtanokon át ([4]) számos módszerrel lettünk gazdagabbak az elmúlt évtizedek során.

Az Okhotin-féle logikai nyelvtanok (Boolean grammars) a CFG-kben megengedett konkatenációt és uniót a metszet és a komplement műveleteivel egészítik ki ([15] és [17]), melyek természetesen, a formalizmus szintjén teszik lehetővé többértelműségi vagy kizárási feltételek megfogalmazását.

A nyelvtani formalizmus kibővítésével nyert többleterő alkalmassá teszi a logikai nyelvtanokat arra, hogy azokban önmagukat tagadó állításokat fogalmazzunk meg, amely problémára a klasszikus logikának nincs kielégítő válasza. Éppen ezért Okhotin eredeti modelljei, az egyértelmű és a természetesen elérhető megoldások szemantikái nem tekinthetők a logikai nyelvtanok által felvetett problémára adott kielégítő válasznak, mivel azok nem minden nyelvtan esetén definiáltak és még csak létezésük sem eldönthető.

Kountouriotis, Nomikos és Rondogiannis ([20]) a klauzális logikára épülő logikai programozás számára kidolgozott megalapozott (well-founded) háromértékű modellre ([13]) alapozva a logikai nyelvtanok egy olyan háromértékű szemantikai modelljét adta, mely minden esetben értelmezett.

Ezen modellnek a formális nyelvekre való alkalmazása azonban néhány rekurzívan definiált nyelvtan esetében olyan, általunk nemkívánatosnak tartott következménnyel jár, miszerint például a „ $S \leftarrow S$ ” nyelvtan kanonikus modelljében minden szó tartalmazása *hamis*, az „ $S \leftarrow \neg S$ ” nyelvtanában ugyanakkor *meghatározatlan*. Véleményünk szerint, és ez vezetett jelen munka kidolgozásához, a *helyes* következtetés mindkét

---

esetben az, hogy a tartalmazás meghatározatlan, hiszen mindkét nyelvtan egy végtelenül rekurzív következtetési folyamatot ír le.

Jelen dolgozat a logikai nyelvtanok háromértékű elemzésével foglalkozik Fitting ([11]) szemantikai modelljére alapján, amely mentes a fent említett furcsaságoktól.

Az első fejezetben áttekintő jelleggel a logikai programokat, mint a klauzális logika gyakorlati implementációit és azok szemantikai modelljeit mutatjuk be. Ezután bevezetjük a logikai nyelvtanokat és azok háromértékű interpretációit, megalkotjuk a logikai nyelvtanok modelljeit, igazoljuk, hogy ezen logikai modellek ténylegesen alkalmasak logikai grammatikák modellezésére, majd tárgyaljuk a szintaktikai elemzésüket, aminek keretében néhány klasszikus algoritmus generalizációját mutatjuk be logikai nyelvtanokra.

## 1. fejezet

# Logikai alapok

Az elsőrendű logikai nyelv *szignatúrája* a  $\sigma_1 = \langle \text{Func}, \text{Preds}, \text{Var} \rangle$  diszjunkt halmazhármass, ahol

$$\text{Func} = \{a/0, b/0, c/0, \dots, f/1, g/1, h/1, \dots, f/2, g/2, h/2, \dots\}$$

a nyelv függvénszimbólumainak,

$$\text{Preds} = \{P/0, Q/0, R/0, \dots, P/1, Q/1, R/1, \dots\}$$

predikátum- vagy relációszimbólumainak,

$$\text{Var} = \{x, y, z, \dots\}$$

változóinak megszámlálhatóan végtelen halmazai. A nulladrendű logikai nyelv  $\sigma_0$  szignatúráját – a szokástól eltérően – hasonlóan definiáljuk, de abban *Var* üres. A formulákban előforduló egyéb logikai és strukturális szimbólumok (pl. zárójelek) halmazát nem tekintjük a szignatúra részének.

A függvény- és predikátumszimbólumok után álló,  $/n$  módon jelölt szám<sup>(1)</sup> az adott szimbólum *aritása*; ezt a nyelv szintaktikai konstrukcióiban nem jelöljük. A logikai nyelvben  $t$  *term*, ha változó vagy  $f(t_1, \dots, t_n)$  alakú, ahol  $f$  egy  $n \geq 0$  aritású függvénszimbólum és  $t_1, \dots, t_n$  termek;  $A$  *atom*, ha  $P(t_1, \dots, t_n)$  alakú, ahol  $t_1, \dots, t_n$  termek és  $P$  egy  $n$  aritású predikátumszimbólum. A nyelv összes termjeinek és formuláinak halmazaira *Func* és *Forms* néven hivatkozunk.

Az elsőrendű logikai nyelv termjeinek és atomjainak legnagyobb nulladrendű részhalmazát rendre a nyelv *Herbrand-univerzumának* és *-bázisának* nevezzük és  $\mathcal{U}$ -val valamint  $\mathcal{B}$ -vel jelöljük. Ezen halmazok elemeit rendre *alaptermeknek* és *-formuláknak* is nevezzük.

---

<sup>1</sup> Jelen dolgozatban minden előforduló szám  $\mathbb{N}$ -ből való, így ezt külön említés nélkül is adottnak tekintjük.

Formulákat a szokásos módon, a  $\neg$  (negáció) unáris, és  $\wedge, \vee, \rightarrow, \leftrightarrow$  (rendre: konjunkció, diszjunkció, materiális implikáció, materiális ekvivalencia) infix bináris operátorok, valamint a  $\forall$  és  $\exists$  kvantorok segítségével képezzük.

Az értékkorlátozott kvantorokat úgy értjük, hogy  $\exists\phi(x) : \psi(x) \leftrightarrow \exists x : \phi(x) \wedge \psi(x)$  valamint  $\forall\phi(x) : \psi(x) \leftrightarrow \forall x : \phi(x) \rightarrow \psi(x)$ . Ha  $\phi(x)$  csak véges sok  $x$ -re teljesül, akkor a kvantoros formulák ekvivalensen diszjunkciókká és konjunkciókká alakíthatók:

$$\exists\phi(x) : \psi(x) \leftrightarrow \bigvee_{\phi(t)} \psi(t) \quad \text{és} \quad \forall\phi(x) : \psi(x) \leftrightarrow \bigwedge_{\phi(t)} \psi(t).$$

Valamely változóelőfordulás *szabad*, ha nem áll az adott változót megnevező kvantor hatáskörében. A szabadon előforduló változók a formula *paraméterei*. Egy formula *zárt*, vagy más néven *állítás*, ha nincs paramétere.

A  $\nu : \text{Var} \rightarrow \text{Term}$  függvényt *termhelyettesítésnek* nevezzük. A  $\nu$  termhelyettesítés alkalmazása a  $\phi$  formulára az a  $\nu(\phi)$  formula, amelyben  $\phi$  minden  $\nu$  paraméterét a  $\nu(\nu)$  termre cseréljük. Egy termhelyettesítést megadhatunk  $\phi[x \mapsto t]$  formában, ilyenkor a fel nem sorolt változókat önmagukra képezzük le. A termhelyettesítés  $\phi$ -re vonatkozóan *változóiértékelés*, ha  $\phi$  minden paraméterét  $\mathcal{U}$ -ba képezi.

Az  $I = \langle I^+, I^- \rangle$  pár (*totális*) *Herbrand-interpretáció*, ha  $I^+ \cap I^- = \emptyset$  és  $I^+ \cup I^- = \mathcal{B}$ . A pár első tagját az interpretáció tartalmazási, míg utóbbit a kizárási halmazának nevezzük, azaz előbbieket az interpretációban igaznak, utóbbiakat hamisnak tekintjük.

Az Herbrand-interpretációk halmazát  $\mathcal{I}$ -vel jelöljük.

Az interpretáció *karakterisztikus*- vagy *indikátorfüggvénye* az a  $I : \mathcal{B} \rightarrow \mathbb{B}$  ( $\mathbb{B} = \{\top, \perp\}$ , az igazságértékek halmaza) függvény, amire minden  $A \in \mathcal{B}$  atom esetén

$$I(A) = \begin{cases} \top & \text{ha } A \in I^+, \\ \perp & \text{ha } A \in I^-. \end{cases}$$

Formulák igazságértékét a szokásos módon értelmezzük. A  $\phi$  formula  $I$  interpretációban felvett értékét  $I(\phi)$ , abban adott igazságértékkel való tartalmazását esetleg  $+\phi \in I$  vagy  $-\phi \in I$  módon jelöljük, ahol előbbi alatt a  $I(\phi) = \top$ , utóbbi alatt  $I(\phi) = \perp$  állítást értjük.<sup>(2)</sup>

---

<sup>2</sup> A  $\pm$  jelölést vagylagos értelemben használjuk, azaz  $\pm\phi \in I$  jelentése, hogy  $+\phi \in I$  vagy  $-\phi \in I$ , nem pedig, hogy ezek egyszerre teljesülnének.

Egy interpretációt megadhatunk például igazsághalmazával ( $I^+$ ), *előjeles atomokként* (pl.  $\{+P(a, f(b)), -Q(f(a)), \dots\}$ ) vagy karakterisztikus függvényének segítségével is. Az interpretációk közül kettőt külön jelöléssel is ellátunk: legyen  $I_\top = \langle \mathcal{B}, \emptyset \rangle$  és  $I_\perp = \langle \emptyset, \mathcal{B} \rangle$ .

Az  $I \in \mathcal{I}$  interpretáció zárt formulák egy  $\Delta$  halmaznak *modellje* ( $I \models \Delta$ ), ha minden  $\phi \in \Delta$  formulára  $\vdash \phi \in I$ .

$\Delta$  *kielégíthető*, ha van modellje és *értvényes*, ha minden interpretáció egyben modellje is.

$\Delta$ -nak *következménye* a  $\phi$  formula ( $\Delta \models \phi$ ), ha  $\Delta$  minden modellje  $\phi$ -t is kielégíti; ekkor  $\Delta \cup \{\neg\phi\}$  kielégíthetetlen.

## 1.1 Logikai programok

Logikai program alatt  $A \leftarrow B_1, \dots, B_m$  alakú szabályok egy véges  $\Pi$  halmazát értjük, ahol  $A$  atom és  $B_1, \dots, B_m$  *literálok* (atomok vagy azok negáltjai). Tények azok a szabályok, ahol  $m = 0$ .  $A$ -t a szabály fejének,  $B_1, \dots, B_m$ -et a törzsének nevezzük. Egy szabály Horn-szabály, ha benne minden  $B_i$  pozitív (negatív); ha egy  $\Pi$  logikai program minden szabálya Horn-szabály,  $\Pi$  Horn-program. A fejükben azonos predikátumot megnevező szabályok alkotják a program egy *partícióját*.

$\Pi$ -nek az  $\mathcal{U}$  univerzum feletti  $\Pi^*$  *példányosítása* az a legszűkebb szabályhalmaz, amire ha  $\pi \in \Pi$  és  $\nu : \text{Var} \rightarrow \mathcal{U}$ , akkor  $\nu(\pi) \in \Pi^*$ , azaz a példányosítás során a szabályokban a változók helyére az univerzum elemeit reprezentáló termeket minden lehetséges módon behelyettesítjük. A példányosított program nulladrendű, de általában végtelen.

### 1.1.1 Szemantika és inferencia Horn-programokban

A logikai programok jelentését általában három fő megközelítésből keressük:

1. A *deklaratív szemantika* meghatározásakor a programból valamilyen elsőrendű logikai elméletet képezünk, és ezen elmélet szemantikai tulajdonságait vizsgáljuk.
2. A *procedurális szemantika* szerint a program jelentését egy konkrét algoritmus kimeneteként definiáljuk.

3. A *kanonikus modell szemantikája* az előző kettő keverékének is tekinthető, eszerint a programnak valamely meghatározott, nevezetes modelljét keressük, általában egy iterált operátor fixpontjaként.

Legyen  $\Pi$  Horn-program. A legegyszerűbb deklaratív szemantika szerint  $\Pi$  szabályai a

$$\forall x_1, \dots, x_n (B_1 \wedge \dots \wedge B_m \rightarrow A)$$

formulákat jelölik, ahol  $x_1, \dots, x_n$  a formula paraméterei, a program egésze ezen szabályok halmazával, mint formulahalmazzal írható le. Ezen egyszerű elmélet ugyan alkalmas arra, hogy a program *pozitív* következményeiről beszélhessünk, *negatív* következményeket azonban nem definiál:  $I_\top$  a programtól függetlenül az elmélet modellje.

A procedurális megközelítéshez egy formális rendszert definiálunk, amiben a levezethetőséget és a levezetést, mint eljárást értelmezzük.

**1. DEFINÍCIÓ** (levezethetőség).  $\Pi$ -ből egy  $A$  alapatom *levezethető* ( $\Pi \vdash A$ ), ha létezik alapatomok olyan  $\Lambda_{(n)}$  véges sorozata, amelynek utolsó eleme  $A$  és minden  $i$ -edik tagjához létezik egy  $(\Lambda_i \leftarrow B_1, \dots, B_m) \in \Pi^*$  szabály úgy, hogy ezen szabályban minden  $B_k$ -hoz van olyan  $j < i$ , hogy  $\Lambda_j = B_k$ .  $A$

$$\frac{B_1, \dots, B_m, (\Lambda_i \leftarrow B_1, \dots, B_m)}{\Lambda_i}$$

következtetésformát *leválasztási szabálynak* (modus ponens) nevezzük.

Horn-programok esetén a leválasztási szabály önmagában helyes és teljes, azaz  $\Pi \vdash A$  pontosan akkor, ha  $\Pi \models A$ . Ennek ellenére (ha egy alapatom a programnak logikai következménye, akkor előáll valamilyen levezetési sorozat véges sokadik tagjaként) az, hogy egy atom levezethető-e a programból, eldönthetetlen (a levezethető atomok halmaza felsorolható, de komplemente nem, így nem is eldönthető).

Egy rendezett, irányított, véges *fa levezetési fa*, ha minden csúcsa egy-egy  $A \in \mathcal{U}$  alapatommal címkézett és ha gyermekeinek címkéi rendre  $B_1, \dots, B_m$ , akkor  $(A \leftarrow B_1, \dots, B_m) \in \Pi^*$ . A levezetési fa csúcsainak egy posztfix felsorolása a fa gyökerében álló atom egy levezetését adja. Egy  $A$  alapatom *levezetési mélysége* annak a legkisebb magasságú levezetési fának a magassága, aminek az adott atom a gyökerét címkézi.

Sem a deklaratív, sem a procedurális szemantika kínál módszert hamis állítások kikövetkeztetésére. A *zárt világ feltételezés* (closed-world assumption, CWA) szerint csak a programból levezethető állításokat tekintjük igaznak, minden máshoz hamis értéket rendelünk. Ennek Reiter-féle formalizációja szerint ([9]) a programnak pontosan akkor CWA-következménye  $\neg A$ , ha  $\Pi \not\models A$ . Ezen következményreláció Horn-programok esetén konzisztens, de általános logikai elméletek esetén nem.<sup>(3)</sup>

<sup>3</sup> Például az  $A \vee B$  formulának CWA-következménye  $\neg A$  és  $\neg B$  is, ugyanakkor  $\{A \vee B, \neg A, \neg B\}$  inkonzisztens.

Mivel egy Horn-programból CWA-következtetéssel előálló interpretáció a programnak mindig modellje, ezért általánosan ezt szokás a program szándékolt jelentésének tulajdonítani. Ugyanezen modellt állítja elő Apt és van Emden következtetési operátora ([10]) is:

**2. DEFINÍCIÓ** (következtetési operátor). Legyen  $I \in \mathcal{I}$  a példányosított logikai program egy interpretációja és  $\Phi : \mathcal{I} \rightarrow \mathcal{I}$  a következőképpen:

$$\Phi(I)(A) = \begin{cases} \top & \text{ha legalább egy } A \leftarrow B_1, \dots, B_m \text{ szabályban } I(B_i) = \top \text{ minden } i\text{-re,} \\ \perp & \text{ha minden } A \leftarrow B_1, \dots, B_m \text{ szabályban } I(B_i) = \perp \text{ legalább egy } i\text{-re.} \end{cases}$$

Ha egy  $C$  atom egyetlen szabály fejében sem szerepel, akkor az azt definiáló szabályhalmaz üres, vagyis  $\Phi(I)$ -ben a  $C$  atom értéke  $\perp$ .

Legyenek  $I_1, I_2 \in \mathcal{I}$  és  $\sqsubseteq$  egy  $\mathcal{I}$  feletti részbenrendezés úgy, hogy  $I_1 \sqsubseteq I_2$  pontosan akkor, ha  $I_1^+ \subseteq I_2^+$  és  $I_1^- \supseteq I_2^-$ . Ekkor  $\langle \mathcal{I}, \sqsubseteq \rangle$  egy teljes rács,  $\Phi$  egy  $\sqsubseteq$  szerint monoton függvény, így a Knaster-Tarski fixponttétel szerint  $\Phi$  fixpontjai<sup>(4)</sup> maguk is egy teljes rácsot alkotnak, aminek van minimuma, így  $\Phi$ -nek van legkisebb fixpontja.

**3. DEFINÍCIÓ** (iterált modell). Legyen

$$\begin{aligned} \Phi^0(I) &= I \\ \Phi^\alpha(I) &= \Phi(\Phi^{\alpha-1}(I)) && \text{ahol } \alpha \text{ rákövetkező sorszám} \\ \Phi^\lambda(I) &= \sup_{\alpha < \lambda} \Phi^\alpha(I) && \text{ahol } \lambda \text{ limeszrendszám} \end{aligned}$$

és  $\Omega_\perp = \Phi^{\omega_0}(I_\perp)$ .<sup>(5)</sup>

Belátható, hogy  $\Phi^\alpha(I_\perp)^+$  atomjai legfeljebb  $\alpha$  mélységgel levezethetők. Mivel minden levezethető pozitív atomnak van véges levezetése, ezért  $\Phi^{\omega_0}$  az összes levezethető atomot tartalmazza.

$\Omega_\perp$  a  $\Phi$  operátor  $\sqsubseteq$  szerinti legkisebb fixpontja,  $\Pi$ -nek  $\sqsubseteq$ -minimális modellje, valamint minden  $A \in \Omega_\perp^+$  atomra  $\Pi \models A$ .

<sup>4</sup>  $x$  az  $f$  függvény fixpontja, ha  $f(x) = x$ .

<sup>5</sup>  $\omega_0$  a legkisebb megszámlálható limeszrendszám.

### 1.1.2 Negáció a logikai programokban

Jelentősen bonyolultabb a helyzet, ha a szabályok törzsében is megengedjük a negatív literálokat. Ekkor a leválasztási elv kiegészítésre szorul, valamint a  $\Phi$  operátor nem monoton és nem is szükségszerűen konvergens. Az ilyen programokban történő inferencia a logikai programozás egyik fontos kutatási területe.

A *negáció, mint kudarc* (negation as failure, NAF) inferenciaszabálya:

$$\frac{\Pi \not\models A}{\neg A},$$

azaz  $\Lambda$ -hoz  $\neg A$  hozzáadható, ha anélkül  $A$ -nak nincs levezetése. A NAF-inferencia inkonzisztens, ha ez  $A$  levezethetőségét okozza, hiszen ekkor  $A$  pontosan akkor levezethető, ha nem.

Mivel  $\Pi \not\models A$  eldönthetetlen, ezért a NAF-inferencia nem teljes és ezáltal nem azonos a CWA-következményfogalommal, mely maga sem azonos a negatív következményfogalommal.

A következőkben néhány példát mutatunk az előbbieket illusztrációjára. Az állítások ellenőrzését az érdeklődő Olvasóra bízunk.

1. *példa.*

$$P \leftarrow P$$

A programnak léteznek  $\{+P\}$ -t és  $\{-P\}$ -t tartalmazó modelljei is, így sem  $+P$ , sem  $-P$  nem logikai következménye, emiatt viszont CWA-következménye  $-P$ .  $\Omega_{\perp}$ -ban  $P$  hamis.

2. *példa.*

$$P \leftarrow \neg P$$

A  $\neg P \rightarrow P$  formula minden modelljében  $\{+P\}$ , így logikai következménye  $+P$ , vagyis nem CWA-következménye  $-P$ . A NAF-inferencia ugyanakkor inkonzisztens és  $\Phi$  is divergens.

3. *példa.*

$$P \leftarrow \neg Q, R$$

$$R \leftarrow$$

A program a  $\{\neg Q \wedge R \rightarrow P, R\}$  formulahalmazt írja le, aminek modelljei a  $\{+P, +Q, +R\}$ ,  $\{+P, -Q, +R\}$  és  $\{-P, +Q, +R\}$  interpretációk; ennek következménye  $+R$ , de sem  $\pm P$ , sem  $\pm Q$  nem az. A NAF alkalmazásával  $-Q$  levezethető, így eszerint csak a  $\{+P, -Q, +R\}$  modell fogadható el, valamint  $\Phi$  is ugyaneke konvergál. A CWA-modell viszont inkonzisztens: a  $\{-P, -Q, +R\}$  interpretáció nem modellje a programnak.

## Clark programkiegészítési szemantikája

Clark ([8]) a NAF-inferencia egy deklaratív modelljét írta le, amire a program Clark-kiegészítéseként hivatkozunk. A korábbi deklaratív modellel szemben, amely a szabályokat egyszerű implikációknak tekinti, a Clark-kiegészítés abból a feltételezésből indul ki, hogy a program szabályai teljes leírását adják annak, mit tekintünk igaznak, így a szabályokból nem következő atomokat hamisnak tekinthetjük

A  $Clark(\Pi)$  formulahalmazt a program szabályhalmazának mechanikus transzformációjával állítjuk elő.  $Clark(\Pi)$  egy elsőrendű elmélet, a  $\Pi$  programot a Clark-kiegészítés előtt *nem* példányosítjuk. Feltesszük, hogy legalább egy partíción belül a különböző paraméterek egyediek: ezt szükség esetén átnevezésekkel biztosítjuk.

1. Az

$$A(t_1, \dots, t_n) \leftarrow B_1, \dots, B_m$$

szabály helyett a

$$A(x_1, \dots, x_n) \leftarrow \exists y_1, \dots, y_k (x_1 = t_1 \wedge \dots \wedge x_n = t_n \wedge B_1 \wedge \dots \wedge B_m)$$

formulát írjuk, ahol  $x_1, \dots, x_n$  új, egy partíción belül azonos változók és  $y_1, \dots, y_k$  a szabály eredeti jobb oldalának a fejből nem előforduló paraméterei.

2. Az egy partíción belüli

$$\begin{aligned} A(x_1, \dots, x_n) &\leftarrow \phi_1 \\ &\vdots \\ A(x_1, \dots, x_n) &\leftarrow \phi_l \end{aligned}$$

formulákat a

$$\forall x_1, \dots, x_n (A(x_1, \dots, x_n) \leftarrow \phi_1 \vee \dots \vee \phi_l)$$

formulára cseréljük.

3. Ha egy  $A$  reláció egyetlen partíció fejében sem szerepel, hozzáadjuk a

$$\forall x_1, \dots, x_n (A(x_1, \dots, x_n) \leftarrow \perp)$$

formulát.

4. Az implikációk helyett ekvivalenciát írunk.

A fenti módon előálló formulahalmazt tekintjük a kiindulási logikai program Clark-kiegészítésének.  $Clark(\Pi)$  formulái nem Horn-formulák.

Formálisan a Clark-kiegészítésnek része még az egyenlőségre vonatkozó axiómarendszer (Clark equality theory, CET) is, amely röviden azt mondja ki, hogy a) csak a szintaktikailag azonos termek egyenlők; b) az egyenlőség ekvivalenciareláció. (Például a  $+(1, 4)$  és  $+(3, 2)$  termek a CET szerint nem egyenlők.)

A programkiegészítésből nyert elméletnek egyes negált literálok immáron tételei. Például a  $\forall x (P(x) \leftrightarrow (x = a))$  formula magjában az egyenlőség csak az  $[x \mapsto a]$  helyettesítéssel elégíthető ki, így egy interpretáció csak akkor lehet modellje a kiegészített halmaznak, ha abban  $+P(a)$  és minden  $b \neq a$  alaptermre  $-P(b)$ . Mivel  $-P(b)$  minden modellben, ezért  $Clark(\Pi) \models \neg P(b)$ .

A kibővített program szemantikája bizonyos esetekben eltérhet attól, amire naivan számíthatunk. Az egyetlen,  $P \leftarrow \neg P$  szabályt tartalmazó programnak nincs Clark-modellje, hiszen  $P \leftrightarrow \neg P$  kielégíthetetlen. Ugyanakkor ha a programot a  $P \leftarrow P$  tautológiával bővítjük, a kapott Clark-formula  $P \leftrightarrow \neg P \vee P$  lesz, aminek a  $+P$ -t tartalmazó interpretációk már modelljei.

A Clark-modell másik gyengeségét illusztrálja a következő példa:

4. példa. Tekinsük az alábbi programot:

$$\begin{aligned} reachable(x, y) &\leftarrow edge(x, y) \\ reachable(x, y) &\leftarrow reachable(x, z), edge(z, y) \\ edge(A, B) &\leftarrow \\ edge(B, A) &\leftarrow \\ edge(C, D) &\leftarrow \\ edge(D, C) &\leftarrow \end{aligned}$$

A program egy olyan gráfot modellez, amelynek négy csúcsából kettő-kettő között páronként kölcsönösen vezet él.  $reachable$  az  $edge$  szomszédsági reláció tranzitív lezártja. Ezen gráfban például  $reachable(A, D)$  nyilvánvalóan hamis, ahogyan ez a szabályoknak nem is logikai következménye:  $+reachable(A, B)$ ,  $+reachable(B, A)$ ,  $+reachable(C, D)$ ,  $+reachable(D, C)$  az első szabályból következnek a megfelelő helyettesítésekkel, a második szabályból következnek még  $+reachable(A, A)$ ,  $+reachable(B, B)$ ,  $+reachable(C, C)$  és  $+reachable(D, D)$ .  $-reachable(A, D)$  a programnak

CWA-következménye és  $\Omega_\perp$ -ban is így szerepel. Tekintsük a program Clark-kiegészítését:

$$\begin{aligned} \forall x, y \left( \text{reachable}(x, y) \leftrightarrow \text{edge}(x, y) \vee \exists z \left( \text{reachable}(x, z) \wedge \text{edge}(z, y) \right) \right) \\ \forall x, y \left( \text{edge}(x, y) \leftrightarrow (x = A \wedge y = B) \vee (x = B \wedge y = A) \vee \right. \\ \left. (x = C \wedge y = D) \vee (x = D \wedge y = C) \right) \end{aligned}$$

A kiegészített programot  $\Omega_\perp$  valóban kielégíti, azonban van olyan modellje is, amelyben  $\text{reachable}(x, y)$ , ahol  $x, y \in \{A, B, C, D\}$ .

## 1.2 Parciális interpretációk

A negációt tartalmazó logikai programok klasszikus szemantikai modelljei mind deficitesek a szó azon értelmében, hogy egyes esetekben vagy nem értelmezettek, vagy ellentmondásosak. Ez a deficitesség általánosan szükségszerűnek látszik, és indokolható azzal, hogy a következtetési szabályok néha inkonzisztensek vagy egyszerűen értelmetlenek. A lehetséges megoldások egyik iránya a többértékű logikákra támaszkodik.<sup>(6)</sup>

Tekintsük az igazságértékek  $\mathbb{B}^+ = \{\top, \sim, \perp\}$  hármelemű halmazát, amelynek elemei a felsorolás sorrendjében az igaz, eldönt(het)etlen és hamis ítéleteket jelölik.

$\mathbb{B}^+$  elemeinek *igazság szerinti rendezése* az a totális  $\sqsubseteq$  rendezési reláció, melyben

$$\perp \sqsubseteq \sim \sqsubseteq \top.$$

Ezzel szemben a *bizonyosság szerinti rendezése* az a  $<$  reláció, melyben

$$\sim < \perp \quad \text{és} \quad \sim < \top.$$

$<$  csak szigorú részbenrendezés, mivel  $\top$  és  $\perp$   $<$  szerint nem hasonlíthatók össze.

**4. DEFINÍCIÓ** (parciális interpretáció). Legyen  $I^+, I^- \subseteq \mathcal{B}$  és  $I^+ \cap I^- = \emptyset$ . Az  $I = \langle I^+, I^- \rangle$  halmazpárt parciális vagy háromértékű (Herbrand-)interpretációnak nevezzük.

<sup>6</sup> Használatos még az *értékréses logika* kifejezés is.

Egy parciális interpretáció karakterisztikus függvénye az  $I : \mathcal{B} \rightarrow \mathbb{B}^+$  függvény, ahol minden  $A \in \mathcal{B}$  alapatomra

$$I(A) = \begin{cases} \top & \text{ha } A \in I^+, \\ \perp & \text{ha } A \in I^-, \\ \sim & \text{egyébként.} \end{cases}$$

$\mathbb{B}^+$  rendezéseit az interpretációkra is kiterjeszthetjük: ha  $I_1, I_2 \in \mathcal{I}$  interpretációk és  $\circ \in \{\leq, \sqsubseteq\}$ , akkor  $I_1 \circ I_2$  pontosan akkor, ha minden  $A \in \mathcal{B}$  atom esetén  $I_1(A) \circ I_2(A)$ .

A pozitivitási és negativitási halmazokkal megfogalmazva  $I_1 \sqsubseteq I_2$  pontosan akkor, ha

$$I_1^- \supseteq I_2^- \quad \text{és} \quad I_1^+ \subseteq I_2^+.$$

és  $I_1 \leq I_2$  pontosan akkor, ha

$$I_1^- \subseteq I_2^- \quad \text{és} \quad I_1^+ \subseteq I_2^+.$$

$A \leq$  szerint minimális  $I_{\sim} = \langle \emptyset, \emptyset \rangle$  interpretációt *nullinterpretációnak* nevezzük.

A  $\{\neg, \wedge, \vee\}$  műveletek definícióival a következőképpen okoskodunk: amellett, hogy elvárjuk a normalitást (klasszikus igazságértékekre a klasszikus műveletekkel azonos eredményt szolgáltatson), amikor legalább az egyik operandus  $\sim$ , azt vizsgáljuk, hogy miként viselkedne az adott művelet, ha  $\sim$  érték helyett  $\top$  vagy  $\perp$  állna. Így tehát ha egy diszjunkcióban az egyik operandus  $\top$ , a másik helyére bármit is írva az eredmény  $\top$  lesz. Ezzel pont ellentétesen, ha a konjunkció egyik operandusa  $\perp$ , akkor a másik semmilyen értékére nem vesz fel  $\top$  értéket. Ezeket Kleene háromértékű erős műveleteinek is nevezik, igazságtábláikat a **1.1. ábra** tartalmazza.

**1. LEMMA.** Legyenek  $I_1, I_2 \in \mathcal{I}$  interpretációk úgy, hogy  $I_1 \leq I_2$ . Ekkor tetszőleges, a  $\vee, \wedge$  és  $\neg$  konnektívumokból épített  $\phi$  formulára  $I_1(\phi) \leq I_2(\phi)$ .

*Bizonyítás.* Esetelemzéssel. □

Eszerint egy interpretáció  $\leq$  szerinti bővítésével a formulák értéke legfeljebb határozatlanról a klasszikus igazságértékek valamelyikére, de utóbbiak között soha nem változhat, azaz a formulák értéke  $\leq$ -re nézve monoton, így a tudásbázis  $\leq$ -kel kompatibilis bővítése soha nem vezethet ellentmondáshoz.

Egy  $I$  interpretáció kielégíti a  $\Delta$  formulahalmazt, ha minden  $\Delta$ -beli  $\phi$  formulára  $I(\phi) = \top$  és cáfolja, ha valamelyikre  $I(\phi) = \perp$ . Az is lehetséges, hogy interpretáció sem nem elégíti ki, sem nem cáfolja  $\Delta$ -t. Ha

| $\vee$  | $\top$ | $\sim$ | $\perp$ |
|---------|--------|--------|---------|
| $\top$  | $\top$ | $\top$ | $\top$  |
| $\sim$  | $\top$ | $\sim$ | $\sim$  |
| $\perp$ | $\top$ | $\sim$ | $\perp$ |

(a) Diszjunkció

| $\wedge$ | $\top$  | $\sim$  | $\perp$ |
|----------|---------|---------|---------|
| $\top$   | $\top$  | $\sim$  | $\perp$ |
| $\sim$   | $\sim$  | $\sim$  | $\perp$ |
| $\perp$  | $\perp$ | $\perp$ | $\perp$ |

(b) Konjunkció

| $\neg$  |         |
|---------|---------|
| $\top$  | $\perp$ |
| $\sim$  | $\sim$  |
| $\perp$ | $\top$  |

(c) Negáció

 1.1. ábra.  $\vee, \wedge, \neg$  háromértékű kiterjesztései.

| $\rightarrow$ | $\top$ | $\sim$  | $\perp$ |
|---------------|--------|---------|---------|
| $\top$        | $\top$ | $\perp$ | $\perp$ |
| $\sim$        | $\top$ | $\top$  | $\perp$ |
| $\perp$       | $\top$ | $\top$  | $\top$  |

(a) Implikáció

| $\leftrightarrow$ | $\top$  | $\sim$  | $\perp$ |
|-------------------|---------|---------|---------|
| $\top$            | $\top$  | $\perp$ | $\perp$ |
| $\sim$            | $\perp$ | $\top$  | $\perp$ |
| $\perp$           | $\perp$ | $\perp$ | $\top$  |

(b) Ekvivalencia

1.2. ábra. Implikáció és ekvivalencia.

$\Delta^\wedge$  a  $\Delta$  véges formulahalmaz elemeiből konjunkcióval képzett formula, akkor  $I$  pontosan akkor elégíti ki  $\Delta$ -t, ha  $I(\Delta^\wedge) = \top$  és pontosan akkor cáfolat, ha  $I(\Delta^\wedge) = \perp$ .

A  $\Delta$  formulahalmaz logikai következménye a  $\phi$  formula ha minden,  $\Delta$ -t kielégítő interpretációban  $\phi$  igaz és minden  $\Delta$ -t nem cáfolóban  $\phi$  nem hamis. Ezzel párhuzamosan a  $\phi \rightarrow \psi$  implikációs formula pontosan akkor igaz, ha  $\phi \sqsubseteq \psi$  és egyébként hamis;  $\phi \leftrightarrow \psi$  pedig pontosan akkor, ha  $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$ . Ezek igazságtábláit a 1.2. ábra tartalmazza.

### 1.2.1 A Fitting-modell

A pozitív logikai programokra alkalmazott  $\Phi$  operátor negáció jelenlétében – kétértékű logikában – nem szükségszerűen konvergens, és ha mégis, akkor sem feltétlenül a  $\sqsubseteq$ -minimális modellhez teszi. A határozatlanságot önálló igazságértéknek tekintve megtehetjük, hogy az ilyen állításokhoz egy harmadik, a negációra nézve stabil igazságértéket rendelünk. Ennek következtében a  $\Phi$  operátor lényegi változtatás nélkül visszanyeri konvergenciáját. A logikai programok ezen modelljét Fitting ([11]) vizsgálta.

**5. DEFINÍCIÓ** (következtetési operátor). Legyen  $I$  a  $\Pi^*$  program háromértékű interpretációja.

$$\Phi(I)(A) = \begin{cases} \top & \text{ha legalább egy } A \leftarrow B_1, \dots, B_m \text{ szabályban } I(B_i) = \top \text{ minden } i\text{-re,} \\ \perp & \text{ha minden } A \leftarrow B_1, \dots, B_m \text{ szabályban } I(B_i) = \perp \text{ legalább egy } i\text{-re,} \\ \sim & \text{egyébként.} \end{cases}$$

**6. DEFINÍCIÓ** (Fitting-modell). Legyen

$$\Phi^0(I) = I$$

$$\Phi^\alpha(I) = \Phi(\Phi^{\alpha-1}(I)) \quad \text{ahol } \alpha \text{ r  kovetke   sorsz  m}$$

$$\Phi^\lambda(I) = \sup_{\alpha < \lambda} \Phi^\alpha(I) \quad \text{ahol } \lambda \text{ limeszrendsz  m}$$

Legyen  $\Omega = \Phi^{\omega_1}(I_-)^{(7)}$ . Ekkor  $\Omega$  a  $\Phi$  oper  tor  $\leq$  szerinti legkisebb fixpontja   s  $\Pi$ -nek minim  lis modellje. Az  $\Omega$  interpret  ci   a logikai program Fitting-modellje.

A  $\Pi$  logikai programnak h  rom  rt  k   Clark-modellje egy  $I$  interpret  ci  , ha  $I \models \text{Clark}(\Pi)$ , ahol  $\text{Clark}(\Pi)$ -t h  rom  rt  k  en   rtelmezz  k. Egy interpret  ci   pontosan akkor a  $\Phi$  oper  tor fixponja, ha  $\Pi$ -nek Clark-modellje ([11] 7.2).

Indokl  st ig  nyel, hogy  $\Phi^\lambda$  defin  ci  j  ban szerepl    $\sup$  val  ban l  tezik. M  g a  $\sqsubseteq$  rel  ci  nak b  rmely k  t interpret  ci  ra van infimuma   s szupr  muma is, addig  $\leq$ -re ez csak az infimum eset  n igaz: ha k  t interpret  ci   egy atomot m  s el  jellel tartalmaz, nincs szupr  mumuk. Ez azonban nem probl  ma, amit a k  vetkezik lemm  k igazolnak:

**2. LEMMA.** Legyen  $I$  interpret  ci  .  $\Phi(I)$  konzisztens.

*Bizony  t  s.* Mivel  $I$  interpret  ci  , ez  rt konzisztens. Tegy  k fel, hogy  $\Phi(I)$  inkonzisztens, azaz valamely  $A$  atomra  $+A \in \Phi(I)$    s  $-A \in \Phi(I)$ . Tekints  k a  $+A$ -t megalapoz    $A \leftarrow B_1, \dots, B_m$  szab  lyt. Ha  $\{+B_1, \dots, +B_m\} \subseteq I$ , akkor nem igaz, hogy minden szab  lyban van olyan  $B_i$ , hogy  $-B_i \in I$  (hiszen ezen szab  lyban pont nincs), te  t  $-A \notin \Phi(I)$ .  $\square$

**3. LEMMA.** Legyen  $I_\alpha = \Phi^\alpha(I_-)$ .  $I_{(\alpha)} \leq$  szerint monoton.

*Bizony  t  s.* A bizony  t  st transzfinit indukci   segítségével v  gezz  k.

$I_0 \leq I_1$  trivi  lis, hiszen a nullinterpret  ci    $\leq$  minimuma.

Legyen  $\alpha$  r  kovetkezik sorsz  m.

Ha valamely  $A$  atomra  $+A \in \Phi(I_{\alpha-1})$  ( $-A \in \Phi(I_{\alpha-1})$ ), akkor l  tezik atomok olyan  $B \subseteq I_{\alpha-1}^+$  ( $B \subseteq I_{\alpha-1}^-$ ) halmaz, hogy valamely (minden)  $A \leftarrow B_1, \dots, B_m$  szab  lyban minden (valamely)  $i$ -re  $B_i \in B$ . Ha  $I_{\alpha-1} \leq I_\alpha$ , akkor  $I_{\alpha-1}^+ \subseteq I_\alpha^+$  ( $I_{\alpha-1}^- \subseteq I_\alpha^-$ ), azaz  $B \subseteq I_\alpha^+$  ( $B \subseteq I_\alpha^-$ ), ekkor viszont  $+A \in \Phi(I_\alpha)$  ( $-A \in \Phi(I_\alpha)$ ) is, azaz r  kovetkezik sorsz  mokra  $I_{\alpha-1} \leq I_\alpha \Rightarrow \Phi(I_{\alpha-1}) \leq \Phi(I_\alpha)$ .

---

<sup>7</sup>  $\omega_1$  a legkisebb megs  ml  lhatatlan limeszrendsz  m.

Ha  $\lambda$  limeszrendszám, tekintsük a  $\langle \bigcup_{\alpha < \lambda} I_\alpha^+, \bigcup_{\alpha < \lambda} I_\alpha^- \rangle$  halmazpárt. Az világos, hogy a szuprémum ennél nem lehet kisebb. Megmutatjuk, hogy ez valóban egy interpretáció, azaz konzisztens.

Feltételezzük, hogy a két halmaz nem diszjunkt, azaz létezik olyan  $A$  atom, hogy  $+A \in I_\lambda$  és  $-A \in I_\lambda$ . Ekkor léteznek olyan  $\alpha, \beta < \lambda$  legkisebb rákövetkező rendszámok, hogy  $+A \in I_\alpha$  és  $-A \in I_\beta$ . Legyen  $\gamma = \max(\alpha, \beta)$ . Ekkor mivel  $I_\alpha \leq I_\gamma$  és  $I_\beta \leq I_\gamma$ , ezért  $+A \in I_\gamma$  és  $-A \in I_\gamma$ , azaz  $I_\gamma$  is ellentmondásos. Ez viszont nem lehetséges, mert  $\Phi$  konzisztens.

Adósak vagyunk még a kezdőlépés limeszrendszámokra történő bizonyításával, azaz hogy  $I_\lambda \leq I_{\lambda+1}$  minden  $\lambda$  limeszrendszámra. Mivel a szuprémum itt csak uniót képez, ezért  $\Phi^\lambda$  új atomot nem vezet be, a már meglévőekre pedig alkalmazható a rákövetkező sorszámmakra adott érvelés: ha  $+A \in I_\lambda$  ( $-A \in I_\lambda$ ), akkor van olyan legkisebb  $\alpha$  rákövetkező sorszám, hogy  $+A \in I_\alpha$  ( $-A \in I_\alpha$ ), ekkor viszont a fenti értelemben használt  $B$  halmaz ugyancsak részhalmaza  $I_\lambda^+$ -nak ( $I_\lambda^-$ -nak), azaz  $+A \in \Phi(I_\lambda)$  ( $-A \in \Phi(I_\lambda)$ ).  $\square$

## A Fitting-modell generalizált logikai programokban

A logikai programok szabályai mindig  $A \leftarrow B_1, \dots, B_n$  alakúak (standard alak). Generalizált esetben megengedjük a  $A \leftarrow \phi$  formát is, ahol  $\phi$  a  $\neg, \wedge, \vee$  műveletekkel összekapcsolt, tetszőleges formula.

Standard programok esetén az egyes partíciók tekinthetők egyfajta diszjunktív normálformának is. Az már a normálformák sajnálatos tulajdonsága, hogy általában nehezebben érthetők és előállíthatók mint egy, a felhasználó szándékát közvetlenebbül kifejező, általános alak.

A normálforma követelményének elvetése nem csak a végfelhasználói specifikáció szempontjából lehet praktikus: legyen  $\Pi^*$  egy olyan példányosított, generalizált logikai program, amelynek minden partíciója triviális, azaz minden előforduló  $A$  atom pontosan egy szabálynak áll a fejében (a továbbiakban: generalizált alak). Ilyenkor  $\Phi$  nagymértékben egyszerűsödik:

**7. DEFINÍCIÓ** (következtetési operátor generalizált programban). Legyen  $I$  a  $\Pi^*$  példányosított, generalizált program interpretációja.

$$\Phi(I)(A) = I(\phi) \text{ ahol } (A \leftarrow \phi) \in \Pi^*$$

Létezik-e minden standard logikai programhoz generalizált alak? Egyetlen atomot akár végtelenül sok példányosított szabály is definiálhat (pl. a  $P(x) \leftarrow Q(y)$  szabály esetében), így a kérdésünkre adott válasz attól függ, megengedünk-e végtelen formulákat. Ha azonban a szabályok jobb oldalai mind olyanok, hogy azoknak nincs a fejben nem szereplő paramétere, a generalizált alak mindig létezik.

**4. LEMMA.** Legyen  $\Pi^*$  példányosított logikai program. Az alábbiak a Fitting-modellre vonatkozóan ekvivalens átalakítások ( $A$  atom,  $\Delta$  formulahalmaz,  $\phi, \psi$  formulák):

$$\frac{A \leftarrow \Delta, \phi, \psi}{A \leftarrow \Delta, \phi \wedge \psi} \text{ (conj.)}$$

$$\frac{A \leftarrow \Delta, \phi \quad A \leftarrow \Delta, \psi}{A \leftarrow \Delta, \phi \vee \psi} \text{ (disj.)}$$

$$\frac{A \leftarrow \Delta, \phi}{A \leftarrow \Delta, \psi} \text{ (repl.)}$$

A helyettesítési szabály (repl.) akkor alkalmazható, ha  $\phi$  és  $\psi$  ekvivalens formulák, azaz  $\models \phi \leftrightarrow \psi$ .

A fenti három szabály segítségével minden standard logikai program generalizálható (ha végesen generalizálható). Ezen átalakításokra a Clark-formulák konstrukciójában is ráismerhetünk.

## 2. fejezet

# Formális nyelvek a háromértékű logikában

Egy  $\Sigma$  *ábécé* olyan véges halmaz, melynek elemei *szimbólumok*. Sztringeknek az ezekből képzett véges  $n$ -eseket nevezzük. Egy  $w$  sztring hosszát  $|w|$ -vel, az  $n$  hosszú sztringek halmazát  $\Sigma^n$ -nel jelöljük.  $\Sigma$  Kleene-lezártja az összes,  $\Sigma$  feletti sztring halmaza,

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i.$$

Sztringek tetszőleges halmazát *nyelvnek* nevezzük.

A  $w_1, \dots, w_n$  sztringek  $w_1 \cdots w_n$  *konkatenációja* az a  $w$  sztring, amelyet  $w_1, \dots, w_n$  egymás után írásával kapunk. Formálisan, a konkatenáció az a sztring, amelyre

$$|w| = |w_1| + \dots + |w_n| \wedge \forall i \leq n \forall j \leq |w_i| : w[|w_1| + \dots + |w_{i-1}| + j] = w_i[j],$$

ahol  $u[i]$  az  $u$  sztring  $i$  indexű betűje.

Az  $L_1, \dots, L_n$  nyelvek konkatenációja az a legszűkebb  $L_1 \cdots L_n$  nyelv, amelyre ha  $\langle w_1, \dots, w_n \rangle \in L_1 \times \dots \times L_n$ , akkor  $w_1 \cdots w_n \in L_1 \cdots L_n$ , azaz

$$L_1 \cdots L_n = \{w_1 \cdots w_n \mid \langle w_1, \dots, w_n \rangle \in L_1 \times \dots \times L_n\}.$$

$\mathbf{w} = \langle w_1, \dots, w_n \rangle$ -et a  $w$  szó egy  $n$ -felosztásának nevezzük, ha  $w = w_1 \cdots w_n$ .  $w$  összes  $n$ -felosztásainak halmazára  $\Xi_n(w)$ -vel hivatkozunk. Hasonlóan, a  $\mathbf{p} = \langle p_1, \dots, p_n \rangle$  számok az  $r$  szám egy  $n$ -felosztását alkotják, ha  $p_1 + \dots + p_n = r$ ; ezek halmazát, miként félreértést általában nem okoz, ugyancsak  $\Xi_n(r)$ -rel jelöljük. A  $w$  sztring és a  $|w|$  szám felosztásai egymásnak kölcsönösen egyértelműen megfeleltethetők olyan

módon, hogy  $\langle w_1, \dots, w_n \rangle$ -höz  $\langle |w_1|, \dots, |w_n| \rangle$ -t rendeljük és viszont. Adott  $w$ -re ez a megfeleltetés mindig egyértelmű.

Legyenek  $U_I$  és  $U_S$  halmazok (indexhalmaz és individuumhalmaz),  $(S_i)_{i \in U_I} \subseteq U_S$  egy  $U_I$  által indexált halmazcsalád, valamint  $\phi(x)$  egy egyparaméteres formula. Tetszőleges  $t \in U_S$  individuumra

$$t \in \bigcup_{\phi(i)} S_i \leftrightarrow \exists \phi(i) : t \in S_i \quad \text{és} \quad t \in \bigcap_{\phi(i)} S_i \leftrightarrow \forall \phi(i) : t \in S_i$$

Ezen definíciókat kapcsolja össze a következő lemma:

**5. LEMMA.** Legyenek  $L_1, \dots, L_n \subseteq \Sigma^*$  nyelvek és legyen  $L^l = L \cap \Sigma^l$ . Ekkor adott  $w$ -re

$$\begin{aligned} w \in L_1 \cdots L_n &\leftrightarrow w \in \bigcup_{\ell \geq 0} \bigcup_{\mathbf{p} \in \Xi_n(\ell)} \bigcap_{i \leq n} \Sigma^{p_1 + \dots + p_{i-1}} L_i \Sigma^{p_i + \dots + p_n} \\ &\leftrightarrow \exists \mathbf{w} \in \Xi(w) \forall i \leq n : w_i \in L_i. \end{aligned}$$

## 2.1 Háromértékű nyelvek

Egy  $\Sigma$  feletti  $\langle L^+, L^- \rangle$  nyelvpár ( $L^+ \subseteq \Sigma^*, L^- \subseteq \Sigma^*$ ) *konzisztens*, ha  $L^+ \cap L^- = \emptyset$ . A konzisztens nyelvpárokat  $\Sigma$  feletti *háromértékű nyelveknek* nevezzük.

Az  $L$  háromértékű nyelv *karakterisztikus* vagy *indikátorfüggvénye* egy olyan  $\Sigma^* \rightarrow \mathbb{B}^3$  függvény, amelyre

$$L(w) = \begin{cases} \top & w \in L^+ \\ \perp & w \in L^- \\ \sim & \text{egyébként.} \end{cases}$$

$L$  tartalmazza a  $w$  szót ( $w \in L$ ), ha  $L(w) = \top$ , nem tartalmazza ( $w \notin L$ ), ha  $L(w) = \perp$  és a tartalmazás eldöntetlen, ha  $L(w) = \sim$ . Egy adott  $\Sigma$  ábécé feletti összes háromértékű nyelv halmazát  $\mathcal{L}$ -vel jelöljük.

A klasszikus műveleteket az alábbi módon definiáljuk háromértékű nyelveken ([20] nyomán):

$$\begin{aligned}
 L_1 \cup \dots \cup L_n &= \langle L_1^+ \cup \dots \cup L_n^+, L_1^- \cap \dots \cap L_n^- \rangle \\
 L_1 \cap \dots \cap L_n &= \langle L_1^+ \cap \dots \cap L_n^+, L_1^- \cup \dots \cup L_n^- \rangle \\
 \bar{L} &= \langle L^-, L^+ \rangle \\
 L_1 \cdots L_n &= \left\langle \bigcup_{\ell \geq 0} \bigcup_{\mathbf{p} \in \Xi_n(\ell)} \bigcap_{i \leq n} \Sigma^{p_1 + \dots + p_{i-1}} L_i^+ | p_i \Sigma^{p_{i+1} + \dots + p_n}, \right. \\
 &\quad \left. \bigcup_{\ell \geq 0} \bigcap_{\mathbf{p} \in \Xi_n(\ell)} \bigcup_{i \leq n} \Sigma^{p_1 + \dots + p_{i-1}} L_i^- | p_i \Sigma^{p_{i+1} + \dots + p_n} \right\rangle
 \end{aligned}$$

Ugyanezen nyelvi műveleteknek adhatjuk logikai leírását is:

$$\begin{aligned}
 (L_1 \cup \dots \cup L_n)(w) &= \begin{cases} \top & \exists i \leq n : w \in L_i \\ \perp & \forall i \leq n : w \notin L_i \\ \sim & \text{egyébként} \end{cases} & (L_1 \cap \dots \cap L_n)(w) &= \begin{cases} \top & \forall i \leq n : w \in L_i \\ \perp & \exists i \leq n : w \notin L_i \\ \sim & \text{egyébként} \end{cases} \\
 \bar{L}(w) &= \begin{cases} \top & w \notin L \\ \perp & w \in L \\ \sim & \text{egyébként} \end{cases} & (L_1 \cdots L_n)(w) &= \begin{cases} \top & \exists \mathbf{w} \in \Xi(w), \forall i \leq n : w_i \in L_i \\ \perp & \forall \mathbf{w} \in \Xi(w), \exists i \leq n : w_i \notin L_i \\ \sim & \text{egyébként} \end{cases}
 \end{aligned}$$

Az első három definíciót evidensnek gondoljuk, ezért magyarázattal csak a konkatenációt illetjük. Egy  $w$  szót pontosan akkor tartalmaz az  $L_1, \dots, L_n$  nyelvek konkatenációja, ha előállítható az ezekben a nyelvekben tartalmazott szavak konkatenációiként, azaz létezik  $w$ -nek olyan  $n$ -partíciója, amelyben az egyes szavak rendre az egyes nyelvekből származnak. A konkatenáció kizártjának definíciója pontosan ennek logikai negáltja, azaz egy  $w$  szó pontosan akkor kerül kizárásra nyelvek konkatenációjából, ha annak minden  $n$ -partíciójában található olyan  $w_i$  szó, amit az  $i$ -edik nyelv kizár. A belső unió előállítja az összes olyan  $\ell$  hosszú szót, amire egy adott  $p$  partícióban igaz, hogy valamelyik szelete a megfelelő nyelvből kizárásra került. Mivel egy szó csak akkor szerepelhet a kizártban, ha ezt a tulajdonságot minden felosztásban teljesíti, ezért ezen halmazoknak az összes felosztásra vett metszetét tekintjük; ezek adják a kizárt  $\ell$  hosszú szavakat, amiből az összes szót úgy kapjuk, hogy  $\ell$ -et az összes lehetséges szóhosszon végigiteráljuk.

Azt, hogy a logikai definíciók a halmazokkal való definíciókkal ekvivalensek, a kétértékű esetekkel analóg módon bizonyíthatjuk. (Elegendő mindig csak az  $\ell = |w|$  esetet tekinteni.)

A következőkben megmutatjuk, hogy az ilyen módon definiált nyelvi műveletek konzisztensek, azaz alkalmazásuk során nem állhat elő olyan helyzet, hogy valamely művelet eredményéül kapott  $L$  nyelvhez létezik olyan  $w$  szó, hogy egyszerre  $w \in L$  és  $w \notin L$ .

**6. LEMMA.** Legyenek  $L, L_1, \dots, L_n \in \mathcal{L}$  háromértékű nyelvek.  $L_1 \cup \dots \cup L_n, L_1 \cap \dots \cap L_n, \bar{L}$  és  $L_1 \cdots L_n$  konzisztensek.

*Bizonyítás.*  $L_1 \cup \dots \cup L_n$  konzisztens.

$$\begin{aligned} (L_1 \cup \dots \cup L_n)^+ \cap (L_1 \cup \dots \cup L_n)^- &= (L_1^+ \cup \dots \cup L_n^+) \cap (L_1^- \cap \dots \cap L_n^-) \\ &= (L_1^+ \cap L_1^- \cap \dots \cap L_n^-) \cup \dots \cup (L_n^+ \cap L_1^- \cap \dots \cap L_n^-) \\ &= ((L_1^+ \cap L_1^-) \cap \dots) \cup \dots \cup ((L_n^+ \cap L_n^-) \cap \dots) \\ &= \emptyset \cup \dots \cup \emptyset \\ &= \emptyset \end{aligned}$$

□

*Bizonyítás.*  $L_1 \cap \dots \cap L_n$  konzisztens. Az előzővel analóg módon.

□

*Bizonyítás.*  $\bar{L}$  konzisztens.

$$L^+ \cap L^- = L^- \cap L^+ = \emptyset$$

□

*Bizonyítás.*  $L_1 \cdots L_n$  konzisztens. Megmutatjuk, hogy a

$$\{\exists \mathbf{p} \in \Xi(|w|) \forall i \leq n : w \in \Sigma^{p_1} \cdots L_i^+ |^{p_i} \cdots \Sigma^{p_n}, \forall \mathbf{p} \in \Xi(|w|) \exists i \leq n : w \in \Sigma^{p_1} \cdots L_i^- |^{p_i} \cdots \Sigma^{p_n}\}$$

formulahalmaz minden  $w \in \Sigma^*$ -re kielégíthetetlen.

Tegyük fel, hogy  $\exists \mathbf{p} \in \Xi(|w|) \forall i \leq n : w \in \Sigma^{p_1} \cdots L_i^+ |^{p_i} \cdots \Sigma^{p_n}$  kielégíthető és legyen  $\mathbf{q}$  egy ilyen felosztás. Ha  $w \in \Sigma^{q_1} \cdots L_i^+ |^{q_i} \cdots \Sigma^{q_n}$ , akkor egyértelműen léteznek olyan  $w_1, \dots, w_n$  szavak, hogy  $\mathbf{w} \in \Sigma^{q_1} \times \dots \times \Sigma^{q_n} \wedge w_i \in L_i^+$ . Ezen irányban mindkét kvantor disztributív a konjunkcióra, azaz  $\mathbf{w} \in \Sigma^{q_1} \times \dots \times \Sigma^{q_n} \wedge \forall i \leq n : w_i \in L_i^+$  is.

Megmutatjuk, hogy a második állítás  $\mathbf{q}$ -ra nem lehet igaz. Ha  $\exists i \leq n : w \in \Sigma^{q_1} \cdots L_i^- |^{q_i} \cdots \Sigma^{q_n}$ , akkor  $\mathbf{w} \in \Sigma^{q_1} \times \dots \times \Sigma^{q_n} \wedge \exists i \leq n : w_i \in L_i^-$  is.  $\forall i \leq n : w_i \in L_i^+$  és  $\exists i \leq n : w_i \in L_i^-$  csak úgy lehet igaz, ha létezik olyan  $i$ , hogy  $w_i \in L_i^+$  és  $w_i \in L_i^-$  is, ekkor viszont  $L_i$  inkonzisztens.

□

A konkatenációt igazolni fogjuk, hogy asszociatív. Ehhez előbb belátunk két rövid lemmát:

**7. LEMMA.** Ha  $L_1, L_2, L \subseteq \Sigma^*$  kétértékű nyelvek úgy, hogy  $\forall u_1, u_2 \in L : |u_1| = |u_2|$ , akkor  $(L_1 \cap L_2)L = L_1L \cap L_2L$  és  $L(L_1 \cap L_2) = LL_1 \cap LL_2$ .

*Bizonyítás.* Csak a jobb oldali disztributivitást bizonyítjuk, a fordítottja teljesen analóg módon igazolható. Ismeretes, hogy  $(L_1 \cap L_2)L \subseteq L_1L \cap L_2L$ . Megmutatjuk, hogy az egyenlőtlenség a megadott feltételek mellett nem teljesülhet.

Tegyük fel, hogy  $w$  egy diszkriminátor, azaz  $w \in L_1L \cap L_2L$ , de  $w \notin (L_1 \cap L_2)L$ . Mivel  $w \in L_1L \cap L_2L$ , ezért  $w = w_1u_1 = w_2u_2$ , ahol  $w_1 \in L_1$ ,  $w_2 \in L_2$  és  $u_1, u_2 \in L$ . Az előfeltételek miatt  $|u_1| = |u_2|$ , viszont ekkor  $u_1 = u_2$  is, mert ugyanazon szónak szuffixei, és ugyanezért  $w_1 = w_2$  is. Ekkor viszont a prefix mindkét nyelvnek eleme, vagyis  $w \in (L_1 \cap L_2)L$ , ez viszont ellentmond a hipotézisünknek, miszerint  $w$  egy diszkriminátor. □

**8. LEMMA.** Legyenek  $\mathbf{p} \in \Xi_{k_1}(x)$  és  $\mathbf{q} \in \Xi_{k_2}(x)$   $x \in \mathbb{N}$ -nek felosztásai úgy, hogy  $k_1 \leq k_2$ .  $\mathbf{q}$ -t  $\mathbf{p}$  finomításának nevezzük, ha léteznek  $\mathbf{p}$  elemeinek olyan felosztásai, hogy azok elemeit sorrendben konkatenálva  $\mathbf{q}$ -t kapjuk. Tetszőleges  $k_1 \leq k_2$  számpárookra teljesül, hogy minden  $\mathbf{q} \in \Xi_{k_2}(x)$  felosztáshoz létezik olyan  $\mathbf{p} \in \Xi_{k_1}(x)$  felosztás, hogy  $\mathbf{q}$   $\mathbf{p}$ -nek finomítása, azaz minden  $k_2$ -felosztás felírható egy  $k_1$ -felosztás finomításaként.

*Bizonyítás.* Legyen kezdetben adott egy  $\mathbf{q} \in \Xi_{k_2}(x)$  felosztás.  $\mathbf{p}$  létezését egy egyszerű algoritmussal bizonyíthatjuk: amíg  $\mathbf{q}$  több mint  $k_1$  elemet tartalmaz, válasszuk ki  $\mathbf{q}$  két szomszédos elemét, és ezek helyére egyetlen értéket, az összegüket írjuk. A felosztás elemszáma ezáltal eggyel csökken, ugyanakkor összege nem változik, így  $k_2 - k_1$  lépésben egy  $\Xi_{k_1}(x)$ -beli felosztást kapunk. □

**9. TÉTEL.** Legyenek  $L_1, \dots, L_n$  háromértékű nyelvek. A konkatenáció asszociatív, azaz bármely  $m$ -re ha  $k$   $n$ -nek egy  $m$ -felbontása, akkor  $L_1 \cdots L_n = (L_1 \cdots L_{k_1}) \cdots (L_{k_1+\dots+k_{m-1}+1} \cdots L_{k_1+\dots+k_m})$ .

*Bizonyítás.* A tartalmazási halmazra a konkatenáció hagyományos definícióját alkalmazzuk, amelynek asszociativitása közzismert, így ismét csak a kizárási halmazzal foglalkozunk. Azt kell megmutatnunk, hogy ez megegyezik a

$$\bigcup_{\ell \geq 0} \bigcap_{\mathbf{p} \in \Xi_m(\ell)} \bigcup_{i=1}^m \Sigma^{p_1+\dots+p_{i-1}} \left( \bigcap_{\mathbf{q} \in \Xi_{p_i}(k_i)} \bigcup_{j=1}^{k_i} \Sigma^{q_1+\dots+q_{j-1}} L_{k_1+\dots+k_{i-1}+j}^{-} \Sigma^{q_j} \Sigma^{q_{j+1}+\dots+q_{k_i}} \right) \Sigma^{p_{i+1}+\dots+p_m}$$

halmazzal, ahol  $L_i^-|^{q_j}$  helyére behelyettesítettük az  $L_{k_1+\dots+k_{i-1}+j}$  nyelv  $q_j$  hosszú kizártjainak halmazát, amit a definícióból az első unió elhagyásával kapunk.

Alkalmazva a konkatenáció disztributivitását a

$$\bigcup_{\ell \geq 0} \bigcap_{\mathbf{p} \in \Xi_m(\ell)} \bigcup_{i=1}^m \bigcap_{\mathbf{q} \in \Xi_{k_i}(p_i)} \bigcup_{j=1}^{k_i} \Sigma^{p_1+\dots+p_{i-1}+q_1+\dots+q_{j-1}} L_{k_1+\dots+k_{i-1}+j}^- |^{q_j} \Sigma^{q_{j+1}+\dots+q_{k_i}+p_{i+1}+\dots+p_m}$$

formulát kapjuk. Most a halmazműveletek disztributivitásával élve a harmadik és negyedik nagyoperátor felcserélhető:

$$\bigcup_{\ell \geq 0} \bigcap_{\mathbf{p} \in \Xi_m(\ell)} \bigcap_{\mathbf{q} \in \bigtimes_{v=1}^m \Xi_{k_v}(p_v)} \bigcup_{i=1}^m \bigcup_{j=1}^{k_i} \Sigma^{p_1+\dots+p_{i-1}+q_{i,1}+\dots+q_{i,j-1}} L_{k_1+\dots+k_{i-1}+j}^- |^{q_{i,j}} \Sigma^{q_{i,j+1}+\dots+q_{i,k_i}+p_{i+1}+\dots+p_m}$$

Mivel  $k_1 + \dots + k_m = n$ , ezért a belső metszet a külső felosztás minden finomításán, együttesen  $\ell$  minden  $n$ -felbontásán, azaz  $\Xi_n(\ell)$ -en fut végig. Hasonlóan, az ezeket követő két unió két indexe összesen 1-től  $n$ -ig számol. A megfelelő nagyoperátorokat ezek alapján összevonva pontosan a bizonyítani kívánt állítást kapjuk eredményül.  $\square$

## 2.2 Logikai nyelvtanok

*Logikai nyelvtan* alatt olyan  $G = \langle \mathbf{V}, \Sigma, \mathbf{P} \rangle$  hármast értünk, amelyben:

- $\mathbf{V}$  a nyelvi változók halmaza (generatív értelemben: nemterminális szimbólumok);
- $\Sigma$  a konstansok halmaza (generatív értelemben: terminális szimbólumok);
- $\mathbf{P}$  a nyelvtani szabályok halmaza.

Mivel általában nem okoz félreértést, a nyelvtan szimbólumaiból levezethető nyelvek karakterisztikus függvényeinek nevéként ugyancsak a nyelvtan szimbólumait használjuk. Terminálisok esetén a tartalmazás kérdése mindig eldönthető, azaz minden  $a \in \Sigma$  és  $w \in \Sigma^*$  esetén  $a(w) \stackrel{\text{def}}{=} (w = a)$ . A nyelv szimbólumainak halmazát  $\Gamma = \Sigma \cup \mathbf{V}$  módon definiáljuk.

A logikai nyelvtanok struktúráját az alábbiként írhatjuk le, ahol  $\epsilon \notin \Sigma$  egy újonnan bevezetett üresszó-szimbólum:

- $\Gamma \cup \{\epsilon\}$  elemei *atomok* és egyben formulák;
- $\neg\phi$  *negált formula*, ha  $\phi$  formula;
- $\phi_1 \wedge \dots \wedge \phi_n$  *konjunktív formula*, ha  $\phi_1, \dots, \phi_n$  ( $n \geq 2$ ) formulák;
- $\phi_1 \vee \dots \vee \phi_n$  *diszjunktív formula*, ha  $\phi_1, \dots, \phi_n$  ( $n \geq 2$ ) formulák;

- $\phi_1 \cdots \phi_n$  *konkatenációs formula*, ha  $\phi_1, \dots, \phi_n$  ( $n \geq 2$ ) formulák;
- $S \rightarrow \phi$  *nyelvtani szabály*, ha  $S \in \mathbf{V}$  és  $\phi$  formula.

A  $\mathbf{P}$  halmazra vonatkozóan megköjtjük, hogy abban minden nemterminálishoz *pontosan egy* olyan szabály tartozik, amelynek bal oldalán az illető nemterminális áll. Ismételten megengedve a nevek túlterhelését, a  $\mathbf{P}(A)$  jelölés alatt azt az  $\phi$  formulát értjük, amelyre  $(A \rightarrow \phi) \in \mathbf{P}$ .

Ha a nyelvtan nem tartalmaz negációt, *konjunktív*nak, ha ezen felül konjunkciót sem, *diszjunktív*nek nevezzük.

### 2.2.1 A logikai nyelvtanok szemantikája

Egy  $G$  grammatika *interpretációján* egy, a nyelvtan minden változóján értelmezett  $I : \mathbf{V} \rightarrow \mathcal{L}$  függvényt értünk, amely azokhoz egy háromértékű nyelvet rendel. Ez természetesen terjeszthető ki nyelvtani formulákra a következőképpen:

- $I(\epsilon) = \langle \{\epsilon\}, \overline{\{\epsilon\}} \rangle$ ;
- $I(a) = \langle \{a\}, \overline{\{a\}} \rangle$ , ha  $a \in \Sigma$ ;
- $I(\neg\phi) = \overline{I(\phi)}$ ;
- $I(\phi_1 \wedge \cdots \wedge \phi_n) = I(\phi_1) \cap \cdots \cap I(\phi_n)$ ;
- $I(\phi_1 \vee \cdots \vee \phi_n) = I(\phi_1) \cup \cdots \cup I(\phi_n)$ ;
- $I(\phi_1 \cdots \phi_n) = I(\phi_1) \cdots I(\phi_n)$ .

Az  $\epsilon$  üresszó-szimbólum és a terminális szimbólumok esetén a formulákhoz tartozó nyelv független az interpretációtól. (A fenti képletekben a komplement  $\Sigma^*$ -ra vonatkozóan értelmezzük, azaz például  $\overline{\{a\}} = \Sigma^* \setminus \{a\}$ .) A nemterminálisokhoz rendelt nyelvet közvetlenül az interpretáció határozza meg; az összetett formulákat a megfelelő nyelvi műveletekre vezetjük vissza.

Képezzünk a szabályokból egyenleteket úgy, hogy a  $\leftarrow$  helyére  $=$  jelet, a logikai műveletek helyett a fenti módon nyelvi műveleteket írunk. Ezen nyelvi egyenletrendszer megoldásai (modelljei) azon interpretációk, melyekben az egyenlőségek teljesülnek.

Tekintsük a  $w \in \Sigma^*$  szót és a  $G$  grammatika valamely  $I$  interpretációját; ekkor  $w$  tartalmazását  $G$  valamely  $\phi$  formulájához rendelt nyelvben felírhatjuk  $I(\phi)$  indikátorfüggvényének segítségével. Ha  $I$  a  $G$  nyelvtannak modellje, akkor  $I(A) = I(\mathbf{P}(A))$  minden  $A \in \mathbf{V}$  esetén, tehát  $I(A)(w) = I(\mathbf{P}(A))(w)$ .

Az a nyelvi műveletek definícióiból világos, hogy egy  $w$  szó tartalmazásának eldöntését a háromértékű nyelvek esetén is mindig visszavezethetjük  $w$  (nem feltétlenül valódi) szeleteinek tartalmazására. Ez a visszavezetés azonban nem szigorúan monoton, azaz a visszavezetési láncban kör keletkezhet. (Ilyen például az  $S \rightarrow S \vee a$  nyelvtan, ahol egy  $w \neq a$  szó tartalmazását kívánjuk eldönteni.) Ennek feloldásához a logikai programokat és az azokhoz kidolgozott modellezési stratégiákat hívjuk segítségül.

A formalizációhoz szükségünk lesz egy  $\Upsilon/n + 1$  predikátumcsaládra ( $\Upsilon \notin \Gamma, n \geq 2$ ), amely pontosan akkor igaz, ha az  $n + 1$ -edik argumentuma megegyezik az első  $n$  argumentumának konkatenációjával. Az  $n + 1$  paraméteres  $\Upsilon$  definíciós sémája:

$$\begin{aligned} \Upsilon(\epsilon, \dots, \epsilon, \epsilon) &\leftarrow \\ \Upsilon(a(w_1), w_2, \dots, w_n, a(w)) &\leftarrow \Upsilon(w_1, w_2, \dots, w_n, w) \\ \Upsilon(\epsilon, a(w_2), \dots, w_n, a(w)) &\leftarrow \Upsilon(\epsilon, w_2, \dots, w_n, w) \\ &\vdots \\ \Upsilon(\epsilon, \epsilon, \dots, a(w_n), a(w)) &\leftarrow \Upsilon(\epsilon, \epsilon, \dots, w_n, w) \end{aligned}$$

A séma szabályait minden  $a \in \Sigma$  terminálisra megalkotjuk, ahol a terminálisokat mint függvényszimbólumokat (sztringkonstruktorokat) használjuk. Az  $abab \in \Sigma^*$  sztring így például megfelel a logikai nyelv  $a(b(a(b(\epsilon)))) \in \mathcal{U}$  termjének. Mivel a konkatenációt minimum kétargumentumúként definiáltuk, ezért csak az  $\Upsilon/3, \dots$  predikátumokat definiáljuk.

A konkatenációs predikátummal minden rendelkezésünkre áll, hogy megadjuk a logikai nyelvtanok programmodelljét. Ennek első lépéseként egy formulagenerátor függvényt definiálunk:

**8. DEFINÍCIÓ.** Legyen  $\rho$  a nyelvtan formuláihoz logikai formulákat rendelő függvény az alábbi módon:

- $\rho(\phi) = \phi(w)$  ha  $\phi$  atomi;
- $\rho(\neg\phi) = \neg\rho(\phi)$ ;
- $\rho(\phi_1 \wedge \dots \wedge \phi_n) = \rho(\phi_1) \wedge \dots \wedge \rho(\phi_n)$ ;
- $\rho(\phi_1 \vee \dots \vee \phi_n) = \rho(\phi_1) \vee \dots \vee \rho(\phi_n)$ ;
- $\rho(\phi_1 \dots \phi_n) = \Upsilon(w_1, \dots, w_n, w) \wedge \rho(\phi_1)(w_1) \wedge \dots \wedge \rho(\phi_n)(w_n)$ .

A  $G$  logikai nyelvtant leíró elsőrendű  $\Pi$  logikai program szabályait három séma szerint generáljuk:

- A  $\epsilon(\epsilon) \leftarrow$  séma alapján pontosan egy szabályt generálunk, önmagát.

- $A(a) \leftarrow$  séma alapján  $|\Sigma|$  szabályt generálunk  $a$  helyére  $\Sigma$  elemeinek helyettesítésével.
- $A(w) \leftarrow \rho(\mathbf{P}(A))$  séma alapján  $|\mathbf{V}|$  szabályt generálunk  $A$  helyére  $\mathbf{V}$  elemeinek helyettesítésével.

A fenti sémák alkalmazásával előálló,  $|\Sigma| + |\mathbf{V}| + 1$  szabályt tartalmazó elsőrendű logikai programot a logikai nyelvtan kanonikus programmodelljének nevezzük.

A kanonikus programmodell ugyan maradéktalanul kifejezi a logikai nyelvtanok szándékolt szemantikáját, nehézséget okoz, hogy a konkatenációban bevezetett  $w_1, \dots, w_n$  szabad változók miatt a program példányosítása partícióként is végtelen, így végesen nem generalizálható.

Vegyük észre, hogy bármely  $w$  szónak csak véges sok felosztása létezik, ezek száma csak  $|w|$  függvénye, így  $w$  bármely értékére a  $\Upsilon(w_1, \dots, w_n, w)$  predikátum is csak véges sok helyen lehet igaz. Ezt figyelembe véve a  $\rho$  függvény egy lehetséges alternatív definíciója:

**9. DEFINÍCIÓ.** Legyen  $\rho$  a nyelvtan formuláihoz logikai formulákat rendelő függvény az alábbi módon:

- $\rho(\phi) = \phi(w)$  ha  $\phi$  atomi;
- $\rho(\neg\phi) = \neg\rho(\phi)$ ;
- $\rho(\phi_1 \wedge \dots \wedge \phi_n) = \rho(\phi_1) \wedge \dots \wedge \rho(\phi_n)$ ;
- $\rho(\phi_1 \vee \dots \vee \phi_n) = \rho(\phi_1) \vee \dots \vee \rho(\phi_n)$ ;
- $\rho(\phi_1 \dots \phi_n) = \exists \langle w_1, \dots, w_n \rangle \in \Xi(w) \forall i \leq n : \rho(\phi_i)(w_i)$ .

Ezen definíció már végesen generalizálható, viszont a klasszikus értelemben véve nem példányosítható, hiszen a példányosítás csak termhelyettesítést foglal magába, itt viszont –  $w$  függvényében – formulahelyettesítés történik, mivel az  $\exists$ -et helyettesítő diszjunkció tagjainak száma nem állandó. (A példányosítás szót ilyen értelemben a következőkben informálisan használjuk.)

5. *példa.* Tekintsük a következő logikai nyelvtant ([20]):

$$S \rightarrow \neg(A \vee B \vee AB \vee BA)$$

$$A \rightarrow CAC \vee a$$

$$B \rightarrow CBC \vee b$$

$$C \rightarrow a \vee b$$

Az ezen nyelvtanhoz tartozó elsőrendű program:

$$\epsilon(\epsilon) \leftarrow$$

$$a(a) \leftarrow$$

$$b(b) \leftarrow$$

$$S(w) \leftarrow \neg(A(w) \vee B(w) \vee \exists \langle w_1, w_2 \rangle \in \Xi(w) [A(w_1) \wedge B(w_2)] \vee \exists \langle w_1, w_2 \rangle \in \Xi(w) [B(w_1) \wedge A(w_2)])$$

$$A(w) \leftarrow \exists \langle w_1, w_2, w_3 \rangle \in \Xi(w) [C(w_1) \wedge A(w_2) \wedge C(w_3)] \vee a(w)$$

$$B(w) \leftarrow \exists \langle w_1, w_2, w_3 \rangle \in \Xi(w) [C(w_1) \wedge B(w_2) \wedge C(w_3)] \vee b(w)$$

$$C(w) \leftarrow a(w) \vee b(w)$$

6. *példa.* Az előbbi példa példányosítása még egy olyan rövid sztringre és részeire, mint *abab*, is igen hosszú, ezért csak néhány kiválasztott szabályát közöljük példa jelleggel:

$$S(abab) \leftarrow \neg(A(abab) \vee B(abab) \vee$$

$$(A(\epsilon) \wedge B(abab) \vee A(a) \wedge B(bab) \vee A(ab) \wedge B(ab) \vee$$

$$A(aba) \wedge B(b) \vee A(abab) \wedge B(\epsilon)) \vee$$

$$(B(\epsilon) \wedge A(abab) \vee B(a) \wedge A(bab) \vee B(ab) \wedge A(ab) \vee$$

$$B(aba) \wedge A(b) \vee B(abab) \wedge A(\epsilon)))$$

$$A(\epsilon) \leftarrow (C(\epsilon) \wedge A(\epsilon) \wedge C(\epsilon)) \vee a(\epsilon)$$

$$A(a) \leftarrow (C(a) \wedge A(\epsilon) \wedge C(\epsilon) \vee C(\epsilon) \wedge A(a) \wedge C(\epsilon) \vee C(\epsilon) \wedge A(\epsilon) \wedge C(a)) \vee a(a)$$

$\vdots$

$$A(aba) \leftarrow (C(\epsilon) \wedge A(\epsilon) \wedge C(aba) \vee C(\epsilon) \wedge A(a) \wedge C(ba) \vee C(\epsilon) \wedge A(ab) \wedge C(a) \vee$$

$$C(\epsilon) \wedge A(aba) \wedge C(\epsilon) \vee C(a) \wedge A(\epsilon) \wedge C(ba) \vee C(a) \wedge A(b) \wedge C(a) \vee$$

$$C(a) \wedge A(ba) \wedge C(\epsilon) \vee C(ab) \wedge A(\epsilon) \wedge C(a) \vee C(ab) \wedge A(a) \wedge C(\epsilon) \vee$$

$$C(aba) \wedge A(\epsilon) \wedge C(\epsilon)) \vee a(aba)$$

$\vdots$

$$C(a) \leftarrow a(a) \vee b(a)$$

$$C(ab) \leftarrow a(ab) \vee b(ab)$$

$\vdots$

$$a(a) \leftarrow$$

$$b(b) \leftarrow$$

$\vdots$

A fenti logikai program teljes,  $\Sigma^*$  feletti példányosítának Fitting-modelljében az  $S$  nyelv pontosan a  $ww$  :  $w \in \{a, b\}^*$  alakú szavakat tartalmazza.

A logikai nyelvtanok és a belőlük képzett logikai programok interpretációit értelemszerűen feleltetjük meg egymásnak: a nyelvi interpretáció által az  $A \in \mathbf{V}$  változóhoz rendelt nyelv pontosan azokat a  $w$  szavakat tartalmazza, amelyekre a logikai program modelljében  $+A(w)$  és pontosan azokat zárja ki, ahol  $-A(w)$ .

A következő, a dolgozat elméleti részének lezárásául szolgáló tétel teremti meg az összefüggést a logikai nyelvtanok formális nyelvi és logikai interpretációi között:

**10. TÉTEL.** Legyen  $M \models \text{Clark}(\Pi)$ . Ekkor  $M$  a  $G$  nyelvtannak is modellje.

*Bizonyítás.* A  $\rho$  függvény által az egyes nemterminálisokhoz rendelt formulák pontosan a háromértékű nyelvek logikai definícióiban szereplő formulák; tetszőleges  $w \in \Sigma^*$  szóra  $\rho(\phi)(w)$  azt adja meg, hogy a  $w$  szó eleme-e a  $\phi$  nyelvtani formulával leírt nyelvnek.

Rögzítsük  $\Sigma^*$  egy  $w$  szavát. Vegyük észre, hogy egy  $w$  szónak a nyelvtan tetszőleges modelljében vett nyelvben való tartalmazásának eldöntéséhez elegendő  $\Pi^*$  egy véges, a  $w$ -t tartalmazó, szeletelésre zárt termékkel példányosított részalmazát tekinteni, aminek megkonstruálható a Clark-kiegészítése. (A teljes nulladrendű példányosítás véges formulákkal nem írható le.)

Standardizáljuk a nulladrendű programot a szabályok jobb oldalainak diszjunktív normálformára alakításával, majd ebben az egyes elemi konjunkciókból képezzünk külön szabályokat, végül a literálok konjunkcióját cseréljük azok listájára.

A Clark-transzformáció első lépéseként a fejatomok unifikációjával párhuzamosan a szabályok jobb oldalain bevezetjük az ezt korrigáló egyenlőségeket, majd az unifikált formulák jobb oldalait előbb konjunkciókká, majd azokat egyetlen diszjunktíóvá egyesítjük.

$\epsilon$  és bármely  $a \in \Sigma$  predikátum esetén a Clark-kiegészítés helyessége triviális, hiszen az ezekből keletkező Clark-formulák  $\forall x(\epsilon(x) \leftrightarrow x = \epsilon)$  és  $\forall x(a(x) \leftrightarrow x = a)$  alakúak, melyek helyesen írják le a megfelelő nyelvben való tartalmazást.

A továbbiakban feltesszük, hogy  $A \in \mathbf{V}$  nyelvi változó (nemterminális).

Mivel a nulladrendű program minden fejatomja alapatom volt, így minden elemi konjunkció tartalmaz egy  $x = u$  korrekciós egyenlőséget, ahol  $x$  az újonnan bevezetett változó,  $u \in \Sigma^*$  az eredetileg az atomban szereplő term. Az azonos korrekciós egyenlőséget tartalmazó konjunkciókat csoportosítva, majd abból

ezen egyenlőséget kiemelve egy olyan diszjunktív formulát kapunk eredményül, amely ezen egyenlőségekkel „címkézett” részformulákat tartalmaz:

$$\forall x(A(x) \leftrightarrow x = u_1 \wedge \text{DNF}(\rho'(A, u_1)) \vee x = u_2 \wedge \text{DNF}(\rho'(A, u_2)) \vee \dots \vee x = w \wedge \text{DNF}(\rho'(A, w)))$$

Vegyük észre, hogy ezen egyenlőségben az alapterm  $\Sigma^*$  egy szava, míg a velük konjunkcióban álló formula pontosan a kiindulási program megfelelő szabálya jobb oldalának diszjunktív normálformáját tartalmazza, ahol  $\rho'(A, u)$  alatt  $\rho(\mathbf{P}(A))$ -nak az  $u$  szóra vett nulladrendű példányosítását értjük.

Mivel kiindulásul  $\Pi^*$ -nek egy, a szelelésre zárt,  $w$ -t tartalmazó részhalmazát tekintettük, ezért tetszőleges  $A \in \mathbf{V}$  esetén pontosan egy olyan korrigáló egyenlőség van, aminek alaptermje  $w$ . (Az összes többi egyenlőség értéke, mivel  $w$ -tól különböző termet neveznek meg,  $\perp$ .) Az ezzel konjunkcióban álló részformula –  $\text{Clark}(\Pi^*)$  konstrukciójából adódóan – pontosan  $\text{DNF}(\rho'(A, w))$ , ami definíció szerint ekvivalens  $\rho(A)(w)$ -vel. Mivel  $\rho(A)$  a nyelvek logikai definícióját adja, ezért ezen formula szemantikai értéke pontosan a  $w$  szónak az  $A$  nemterminális nyelvében való tartalmazást írja le.

Mivel a  $w$  szó választása tetszőleges, ezért az állítás  $\Sigma^*$  minden elemére igaz. □

Megjegyezzük, hogy egy logikai program háromértékű modellje nem feltétlenül egyértelmű, így egy logikai nyelvtannak is többféle modellje létezhet. Annak eldöntését, hogy ezek közül melyiket (ha egyáltalán bármelyiket) tekinti helyesnek, az Olvasóra bízunk.

### 2.2.2 Mi szükség még egy szemantikára?

A szintaktikai elemzés nem egy általános logikai probléma. Nem pusztán az a célunk, hogy sztringek tartalmazásának kérdéseire ellentmondásmentes válaszokat rendeljünk, hanem legalább ennyire fontos, ha nem fontosabb, hogy ezen konklúziók levezetése egyben valamiféle feltárását jelentse az adott sztring szerkezetének. Egy fordítóprogramnak nem pusztán igen-nem válaszokra van szüksége azt illetően, hogy egy sztring egy szintaktikailag helyes programot ábrázol-e, hanem a levezetési fára, amely alapján a program-szöveg feldolgozható.

[20] a  $S \rightarrow \neg S$  nyelvtant értelmetlennek nevezte (amivel egyetértünk), ugyanakkor az  $S \rightarrow S$  nyelvtant már „érdekesnek” tartotta. Véleményünk szerint mindkét nyelvtan értelmetlen, mivel mindkettő egy totálisan rekurzív következtetési folyamatot ír le, aminek nincs triviális esete. Hogy előbbinek nincs is modellje, míg utóbbinak minden interpretáció egyben modellje is, de ezek metszete üres (azaz nincs logikai értelemben

vett következménye), immateriális, hiszen szintaktikai elemzésre, ugyan más-más okból, de mindkettő alkalmatlan.

Amellett érvelünk, hogy olyan szemantikát válasszunk, amelyben szavak egy nyelvben való tartalmazását kizárólag a terminálisokra levezetve, azokkal ütköztetve döntsük el. Ha tudunk ilyen levezetést konstruálni, az adott szónak a nyelvben való tartalmazását elfogadottnak tekintjük; ha explicite meg tudjuk mutatni, hogy egy adott szó minden lehetséges levezetése ellentmondásba torkollik, akkor azt a szót a nyelvből kizárjuk; minden egyéb esetben a tartalmazás kérdését eldöntetlennek minősítjük. A Fitting-szemantika ilyen tulajdonságú, míg a [20] által használt megalapozott szemantika nem. Ennek következménye, hogy ha egy definíciónak nincs (tranzitívan sem) triviális esete, mint például a korábban említett  $S \rightarrow S$ , akkor abban a nyelvben *minden* szó tartalmazása eldöntetlen.

Ez a konstruktivista megközelítés ugyancsak segíti a levezetési fák építését, amivel [20] nem foglalkozott.

Jegyezzük ugyanakkor meg, hogy a tartalmazás tekintetében ezen szemantika is kompatibilis a környezet-független nyelvtanok hagyományos modelljével.



### 3. fejezet

## Szintaktikai elemzés

A szintaktikai elemzési probléma annak eldöntése, hogy egy adott  $G$  grammatika és annak egy  $S$  változója (illetve valamely modelljében: nyelve), valamint egy  $w \in \Sigma^*$  szó esetén  $S$  tartalmazza-e a  $w$  szót, bővebb értelemben egy ezt bizonyító adatszerkezet (levezetési gráf) előállítás. A továbbiakban külön említés nélkül feltételezzük, hogy a  $G$  nyelvtan és a  $w$  szó adottak, valamint rögzítjük, hogy a tartalmazást a nyelvtan Fitting-modelljére vonatkozóan szeretnénk eldönteni. Ahhoz, hogy egy  $w$  szó tartalmazását eldöntsük, elegendő a nyelvtan modelljének legkisebb,  $w$ -t tartalmazó, szeletelésre zárt halmazával vett metszetét vizsgálni, hiszen minden szabály olyan, hogy abban a függőségek ezen halmazon belüliek.

Tekintsük a nyelvtan szabályait absztrakt szintaxisgráfjukkal adotttnak, amely reprezentáció már rendelkezésünkre áll; az egyes formulákra és azok gyermekeire/szülőire így közvetlenül hivatkozhatunk. Egy  $\psi$  formula szülője a  $\phi$  formula, ha  $\psi$   $\phi$ -nek közvetlen részformulája (gyereke) vagy  $\phi$  és  $\psi$  rendre egy szabály bal és jobb oldalán álló formulák. Egy  $A$  változónak *közvetlen függősége* a  $B$ , ha  $\mathbf{P}(A)$ -ban szerepel  $B$ . A nyelvtan összes formulájának halmazát jelölje  $\mathcal{F}$ .

Praktikus megfontolásból a  $w$  sztring szeleteit egy  $0 \leq i \leq j \leq |w|$  indexpárral azonosítjuk úgy, hogy a pozíciókat a sztring karakterei között értelmezzük, azaz a 0 indexű pozíció az első karakter előtt, a  $|w|$  indexű az utolsó után van. Például az *abbab* szó  $[2, 5]$  szelete a *bab* szó.

Egy  $n$  szám  $k$ -felosztásainak száma  $O(n^{k-1})$  nagyságrendű, azaz  $k$ -ban exponenciális. Megjegyezzük, hogy ezen kitevő kizárólag a nyelvtan függvénye, a vizsgált  $w$  sztring e szempontból indifferens. A konkatenáció asszociativitását felhasználva a számítási igény csökkenthető, ha a konkatenációt rekurzívan definiáljuk:  $f_1 \cdots f_k = f_1(\cdots f_k) = (f_1 \cdots)f_k$ . (Ez a klasszikusan alkalmazott binarizáció, amit a konkatenáció

asszociativitása miatt logikai nyelvtanok esetén is alkalmazhatunk.) A binarizáció eredményeként kapott formulákra a HEAD és TAIL nevekkal fogunk hivatkozni. Azt hogy melyik fajta (jobbról vagy balról törté-  
nő) binarizációt érdemes választanunk, az algoritmusok döntenek el; a tabuláris elemzők erre érzéketlenek,  
míg az LL-jellegű elemzés balról, az LR pedig jobbról asszociatív.

A logikai formulák értéke indexpárokkal kifejezve:

$$\begin{aligned}
\epsilon(i, j) &\leftarrow (i = j) \\
a(i, j) &\leftarrow (w[i, j] = a) \\
A(i, j) &\leftarrow \mathbf{P}(A)(i, j) \\
(\neg\phi)(i, j) &\leftarrow \neg\phi(i, j) \\
(\phi_1 \wedge \dots \wedge \phi_n)(i, j) &\leftarrow \phi_1(i, j) \wedge \dots \wedge \phi_n(i, j) \\
(\phi_1 \vee \dots \vee \phi_n)(i, j) &\leftarrow \phi_1(i, j) \vee \dots \vee \phi_n(i, j) \\
(\phi_1 \dots \phi_n)(i, j) &\leftarrow \bigvee_{i \leq k \leq j} \text{HEAD}[\phi_1 \dots \phi_n](i, k) \wedge \text{TAIL}[\phi_1 \dots \phi_n](k, j)
\end{aligned}$$

A kiüresedőség (nullability), azaz hogy valamely változóból levezethető-e az üres sztring, az algoritmu-  
sok végrehajtásához nélkülözhetetlen ismeret. Mivel az kiüresedőség a nyelvtan és nem a bemenő sztring  
tulajdonsága, ezért egy előfeldolgozási lépésben kiszámítható a fenti szabályokba történő azonos pozíció-  
értékek helyettesítésével. Szemben a környezetfüggetlen elemzéssel, a kiüresedő nemterminálisok elimini-  
ációja itt nem lehetséges, tekintettel, hogy a komplementképzés egy nem kiüresedő nemterminálisból is elő-  
tudja azokat állítani. Az  $A$  nemterminális kiüresedőségére  $\text{NULLABLE}[A]$ -val hivatkozunk, kiszámítását az

**1. algoritmus** tartalmazza.

---

DETERMINE-NULLABLES()

```

1: let  $Q \leftarrow V$ 
2: let  $N \leftarrow \emptyset$ 
3: while  $Q$  has an unprocessed element  $A$  do
4:   if  $A \in N$  then
5:     continue
6:   if  $\pm \mathbf{P}(A) \in N$  then
7:     add  $\pm A$  to  $N$ 
8:   for each  $B$  where  $\mathbf{P}(B)$  contains  $A$  and  $B \notin N$  do
9:     add  $B$  to  $Q$ 
10: return  $N$ 

```

---

**1. algoritmus:** A kiüresedőség eldöntése.

A kiüresedő szavakat kiszámító algoritmus, ahogy végeredményben eljárásaink mindegyike, szaturációs elvű: addig ismétli az algoritmus magját, amíg az további új eredményeket már nem keletkeztet. A  $Q$  sor kezdetben az összes nemterminálist tartalmazza, míg az  $N$  interpretáció üres. Amint egy nemterminális kiüresedőségét megállapítjuk, azt  $N$ -ben rögzítjük és minden olyan változót, aminek a definíciója erre hivatkozik, visszateszünk a sorba. Belátható, hogy mivel az összes eldöntetlen értékelésű változók száma monoton (legfeljebb  $|V|$  lépésenként) csökken, ezért  $Q$  előbb-utóbb kiürül és az eljárás véget ér.

A kiüresedőség kiszámító eljárás egyben rávilágít a Fitting-modell kiszámításának gyakorlati mikéntjére is: egy nemterminális a bemenet valamely szeletére vonatkozó értékét mindig úgy számítjuk, hogy az eddig megszilárdult tudásunkat (az aktuális interpretációt) felhasználva kiszámítjuk a szószelet adott változóra vonatkozó értékét, és ha immáron konkrétan meghatározható, akkor ezt visszahelyettesítjük az őt tartalmazó formulákba is, sít.

A  $w$  szóra vonatkozó Fitting-modell  $\Phi$  operátor közvetlen alkalmazásával történő kiszámítását a **2. algoritmus** írja le.

---

```

CANONICAL-FITTING()
1: let  $M \leftarrow [\cdot, \cdot]$ 
2: loop
3:   let  $M' \leftarrow [\cdot, \cdot]$ 
4:   for each  $i, j$  where  $0 \leq i \leq j \leq |w|$  do
5:     for each  $A$  in  $V$  do
6:       if  $\pm P(A) \in M[i, j]$  then
7:         add  $\pm A$  to  $M'[i, j]$ 
8:   if  $M < M'$  then
9:      $M \leftarrow M'$ 
10:  else
11:    return  $M$ 

```

---

**2. algoritmus:** A Fitting-modell kanonikus kiszámítása.

Belátható, hogy az algoritmus által előállított interpretációkra mindig  $M \leq M'$ , és mivel egy véges szószelethez tartozó interpretáció legfeljebb  $|w|^2 \cdot |V|$  méretű lehet, ezért legfeljebb ennyi iterációt követően az algoritmus megáll.

Fitting szemantikája konstruktív: egy formulát csak akkor fogad el határozott értékelésűnek, ha arra valamelyik részformula már meghatározott értékelése bizonyítékul szolgál. A leglényegesebb különbség a klasszikus, kétértékű elemzéshez képest, hogy abban a kizárt harmadik elvének megfelelően a hamis konk-

lúziókat az igazság hiányaként definiáljuk (gyenge negáció). Ezzel szemben a Fitting-szemantika mindig (a formula fajtájától függő módon) részformulák igazságából származtatja a formulák igazságértékét.<sup>(1)</sup>

A következőkben négy algoritmust mutatunk be, melyek mindegyike egy klasszikus, a környezetfüggetlen nyelvtanokra kitalált algoritmus adaptációja logikai nyelvtanokra.

Az első két módszer a tabuláris (offline) algoritmusok közül való, melyek a bemenet egészének egyszerre történő feldolgozásával dolgoznak, általában jól és könnyen párhuzamosíthatók, valamint jóval egyszerűbbek is.

Ezzel szemben a két későbbi eljárás online elemzőket valósít meg, melyek a bemeneten szekvenciálisan haladva dolgozzák azt fel. Ezek – az offline algoritmusokkal szemben – olyan bemenetek elemzésére is használhatók, melyek egésze az algoritmus futásának kezdetén nem elérhető, pl. a felhasználó által gépelt szövegek esetén.

Minden algoritmusunk csak felismerést végez, a levezetési fák konstrukciójával nem foglalkozik. Úgy gondoljuk azonban, hogy az eljárások ilyen irányú kibővítése a szakirodalomból már ismert technikák alkalmazásával ([21], [19]) nem jelent különösebb problémát.

## 3.1 Felülről lefelé (top-down) tabuláris elemző

Az első eljárás a klasszikus rekurzív ereszkedési (recursive descent) technika dinamikus programozással bővített módszerét alkalmazza az eldöntési probléma megoldására. A módszer Unger ([5]) környezetfüggetlen elemzőjének általánosítása (**3. algoritmus**).

A rekurzív ereszkedő eljárások klasszikus problémája, hogy körök esetén könnyen végtelen rekurzióba kerülnek. A már bejárt csúcsok megjelölésének módszere, amelyet ez a megoldás is alkalmaz, pontosan ezt a problémát hivatott megelőzni, garantálja, hogy a rekurzió a hívási gráfnak valamely feszítőfája mentén halad. Magyarázatot igényel, hogy ez a keresési stratégia helyes eredményt ad.

**11. LEMMA.** Ha  $EVALUATE(S, 0, |w|)$ -nek van véges levezetése, akkor van körmentes levezetése is.

*Bizonyítás.* Tekintsük az  $EVALUATE$  függvény  $C$  hívási fáját valamely véges levezetés esetén, ahol a csúcsokat a függvényhívás argumentumaival címkézzük. A hívási fa mindig körmentes, de a csúcsok címkéi

---

<sup>1</sup> Más háromértékű szemantikák, például a megalapozott szemantika, bizonyos körülmények között elégségesnek tekinti azt egy atom hamisságának igazolásához, hogy az adott atom nem lehet igaz.

PARSE( $S$ )

```
1: let  $M \leftarrow [\cdot, \cdot]$ 
2: for each  $i$  in  $[0 .. |w|]$  do
3:   for each  $\phi$  in  $\mathcal{F}$  do
4:     add  $\phi$  with sign  $\text{NULLABLE}[\phi]$  to  $M[i, i]$ 
5:   for each  $j$  in  $[i .. |w|]$  do
6:     for each  $c$  in  $\Sigma$  do
7:       add  $c$  with sign  $w[i, j] = c$  to  $M[i, j]$ 
8: return EVALUATE( $S, 0, |w|$ )
```

EVALUATE( $\phi, i, j$ )

```
1: if  $\pm\phi \in M[i, j]$  then
2:   return the sign of  $\phi$  in  $M[i, j]$ 
3: if  $\langle \phi, i, j \rangle$  is marked then
4:   return  $\sim$ 
5: mark  $\langle \phi, i, j \rangle$ 
6: if  $\phi \in \mathbf{V}$  then
7:   let  $s \leftarrow \text{EVALUATE}(\mathbf{P}(\phi), i, j)$ 
8: if  $\phi$  is  $\neg\phi_1$  then
9:   let  $s \leftarrow \neg\text{EVALUATE}(\phi_1, i, j)$ 
10: if  $\phi$  is  $\phi_1 \vee \dots \vee \phi_n$  then
11:   let  $s \leftarrow \text{EVALUATE}(\phi_1, i, j) \vee \dots \vee \text{EVALUATE}(\phi_n, i, j)$ 
12: if  $\phi$  is  $\phi_1 \wedge \dots \wedge \phi_n$  then
13:   let  $s \leftarrow \text{EVALUATE}(\phi_1, i, j) \wedge \dots \wedge \text{EVALUATE}(\phi_n, i, j)$ 
14: if  $\phi$  is  $\phi_1 \dots \phi_n$  then
15:   let  $s \leftarrow \bigvee_{i \leq k \leq j} \text{EVALUATE}(\text{HEAD}[\phi], i, k) \wedge \text{EVALUATE}(\text{TAIL}[\phi], k, j)$ 
16: if  $s \neq \sim$  then
17:   add  $\phi$  with sign  $s$  to  $M[i, j]$ 
18: unmark  $\langle \phi, i, j \rangle$ 
19: return  $s$ 
```

---

**3. algoritmus:** Rekurzív ereszkedő elemző logikai nyelvtanokhoz.

ismétlődhetnek. Ha egy csúcs egy vele azonos címkéjű csúcsnak leszarmazottja, akkor a hívási láncban kör keletkezett.

Ha  $C$  által reprezentált hívási lánc körmentes, akkor nyilván van körmentes levezetés. Tegyük fel, hogy a hívási lánc nem körmentes, ekkor van olyan  $v$  csúcs a hívási fában, aminek vele azonos címkéjű leszarmazottja. Mivel a hívási fa véges, ezért ilyen leszarmazottból van legalacsonyabban fekvő is ( $u$ ), amelynek már nincs vele azonos címkéjű leszarmazottja.

Ha  $u$  értéke  $\sim$ , akkor önkényes levágása és  $\sim$ -tel való helyettesítése helyes. Ha  $u$  értéke nem  $\sim$ , akkor a modell konzisztenciája miatt  $v$  és  $u$  értéke megegyezik, azaz  $v$ -t  $u$ -val helyettesítve ismételtlen egy levezetést kapunk. Mivel  $C$ -ből így minden kör eliminálható, ezért  $EVALUATE(S, 0, |w|)$ -nek van körmentes levezetése is.  $\square$

Az algoritmus futásidejét az  $M$  táblázat kitöltése korlátozza. Az  $EVALUATE$  függvényt összesen  $|\mathcal{F}| \times |w|^2$ -féle argumentummal hívhatjuk, minden lehetséges híváshoz legfeljebb egy  $M$ -beli érték tartozik. Ha minden formula  $\top$  vagy  $\perp$  értékű, akkor  $EVALUATE$  nem triviális hívásainak száma ugyanezen érték által korlátozott. Mivel  $EVALUATE$  futásideje a konkatenáció esetében  $j - i$  függvényében lineáris, ezért a teljes eljárás  $O(|\mathcal{F}| \times |w|^3)$  komplexitású.  $|\mathcal{F}|$  kizárólag a nyelvtan függvénye, ezért a komplexitás a bemenet hosszának függvényében  $O(|w|^3)$ , ami megegyezik a környezetfüggetlen elemzőkével.

Ha az algoritmust úgy módosítanánk, hogy  $M$ -ben nem az összes formulát, hanem csak a nemterminálisokat tárolnánk, az algoritmus futásideje a következőképpen alakulna: negáció esetében a futásidő növekménye konstans (egy függvényhívás), konjunkció és diszjunkció esetén konstansszoros (a közvetlen részformulák számának függvényében), konkatenáció esetén  $O((j-i)^{k-2})$ -szeres, ahol  $k$  a konkatenálandó elemek száma; a teljes algoritmus tehát  $O(|w|^{k+1})$  komplexitású volna, ahol  $k$  a leghosszabb konkatenáció. (Ez ugyancsak megegyezik a hasonló környezetfüggetlen eredményekkel.)

### 3.2 Alulról felfelé (bottom-up) tabuláris elemző

A tabuláris elemzők családjából származó második algoritmus a szintaktikai elemzés irodalmából jól ismert CYK elemző ([3]) általánosítása (**4. algoritmus**). Unger megoldásához hasonlóan a CYK elemző is egy  $O(|w|^2)$  méretű táblázatot tölt ki, azonban míg előbbi ezt a teljes sztringet reprezentáló cellánál (a tetején) kezdi, a CYK algoritmus az egykarakteres sztringekkel.

---

PARSE ()

```

1: for each  $i$  in  $0 \dots |w|$  do
2:   for each  $\phi$  where  $\text{NULLABLE}[\phi] \neq \sim$  do
3:     add  $\langle \text{NULLABLE}[\phi], \phi \rangle$  to  $M[i, i]$ 
4:   for each  $\ell$  in  $1 \dots |w|$  do
5:     for each  $i$  in  $0 \dots |w| - \ell$  do
6:       let  $j \leftarrow i + \ell$ 
7:       for each  $t$  in  $\Sigma$  do
8:         add  $\langle (w[i, j] = t), t \rangle$  to  $M[i, j]$ 
9:       for each  $\phi$  where  $\phi$  is a concatenation formula in  $\mathcal{F}$  do
10:        if  $\text{VERIFY}(\phi, i, j)$  then
11:          add  $\phi$  to  $M[i, j]$ 
12:        let  $Q \leftarrow \emptyset$ 
13:        for each  $\phi$  in  $M[i, j]$  do
14:          add all parents of  $\phi$  to  $Q$ 
15:        while  $Q$  has an unprocessed element  $\phi$  do
16:          if  $\phi$  is in  $M[i, j]$  then
17:            continue
18:          if  $\text{VERIFY}(\phi, i, j)$  then
19:            add  $\phi$  to  $M[i, j]$ 
20:          add all parents of  $\phi$  to  $Q$ 

```

VERIFY ( $\phi, i, j$ )

```

1: return  $\phi$  is a variable, negation, positive disjunction or negative conjunction or
    $\phi$  is a negative disjunction or a positive conjunction and all children of  $\phi$  are in  $M[i, j]$  or
    $\phi$  is a positive concatenation and  $\bigvee_{i \leq k \leq j} \text{HEAD}[\phi] \in M[i, k] \wedge \text{TAIL}[\phi] \in M[k, j]$  or
    $\phi$  is a negative concatenation and  $\bigwedge_{i \leq k \leq j} \text{HEAD}[\phi] \in M[i, k] \vee \text{TAIL}[\phi] \in M[k, j]$ 

```

---

#### 4. algoritmus: Bottom-up tabuláris elemző

Az eljárás a CYK elemzőhöz hasonlóan arra épül, hogy hossz szerint felfelé haladva minden szószeletre megállapítja az összes szintaktikai kategóriát, amibe az adott szó tartozhat. Azt, hogy milyen nyelvi formulákra lehet szükségünk, az  $\mathcal{F}$  halmaz meghatározza, így az algoritmus futásideje véges keretek közé szorítható.

Mivel a negációs, konkatenációs és diszjunktív formulák olyanok, hogy egy adott szóra való teljesülésük csak más formulák ugyanazon szóra való teljesülésétől függ (informálisan: ugyanazon cellában maradnak), ezért a gyermekcellákra csak a konkatenáció kiértékelése során van szükségünk, illetve egy cella kezdeti tartalmát is pontosan ezek a formulák adják majd. Sajnos nem tehetjük meg, hogy a konkatenációval a cella kezdeti feltöltése után már nem foglalkozunk: ha a konkatenáció kielégítéséhez szükséges egy

olyan felosztás teljesülése, amelyben az egyik szelet a teljes szó, akkor megeshet, hogy ezt csak a részben lefolytatott szaturáció után tudjuk levezetni.

A köbös futásidő eléréséhez itt is megkerülhetetlen a konkatenáció binarizálása, amit bármilyen felosztással (pl. a fejelem leválasztásával,  $\phi_1$  és  $\phi_2 \cdots \phi_n$ ) megtehetünk. Az algoritmus  $O(|w|^2)$  cella kitöltését végzi el, tehát azt kell belátnunk, hogy cellánként legfeljebb  $O(|w|)$  időt tölt.

A VERIFY eljárás azt ellenőrzi, hogy egy formula érvényességi feltételei valóban teljesülnek-e és hozzáadható-e az aktuális cellához. Ezt két esetben hívhatjuk: konkatenáció esetében minden konkatenációs formulára egyszer, a cella feldolgozásának megkezdésekor más formulára akkor, ha valamely gyermeke teljesül. Éppen ezért ha egyetlen gyermek teljesülése elegendő és a formula nem konkatenáció, a formulát az eljárás azonnal el is fogadja. Negatív diszjunkció és pozitív konjunkció esetén ellenőriznie kell az összes részformula teljesülését, ez azonban  $|w|$ -től független.

Mivel konkatenáció esetén az osztópont az  $[i..j]$  tartományon fut végig, ezért a futásidő  $O(|w|)$  nagyságrendű.

Hányszor hívhatjuk meg a VERIFY() eljárást egy adott formulára? Legfeljebb annyiszor, ahány részformula van, konkatenációkra a kezdőkitöltés miatt eggyel többször. Hasonló elgondolás alapján a Q sorba is legfeljebb a részkifejezések számától függő esetben kerülhet be egy formula. Ezek mind a nyelvtan bemenettől független tulajdonságai.

Mindezt összesítve az algoritmus  $O(|w|^3)$  futásidejű.

## 3.3 Olvasásirányú online elemző

A következőkben Earley generalizált környezetfüggetlen elemzőjének ([6]) Aycok és Horspool által módosított ([16], az 5. szakasszal bezárólag) változatát adaptáljuk logikai nyelvtanokra.

Az levezetési gráfban, ha egy adott formulához valamely  $(i, j)$  pozíciópár között nem tartozik él, azt jelenti, hogy az adott nyelvi formula levezethetőségét az  $(i, j)$  pozíciók között nem ismerjük. Az eljárás itt is szaturációs elven működik (addig generálunk újabb és újabb elemeket, míg többet már nem tudunk), a már megtalált levezetéseket (akár pozitív, akár negatív) konkrét élekkel reprezentáljuk. Ezek az élek létrejööttüket követően nem változnak, a kikövetkeztetett tudást biztosnak tekinthetjük. A nyelvi formulák

definícióiból adódóan egy adott formulához egy adott pozíciópáron nem létezhet egyszerre pozitív és negatív él is.

Leginkább abban térünk el Earley klasszikus algoritmusától, hogy a cáfoló levezetések is explicit reprezentációt kapnak, tehát ha valamely  $\phi$  formulára  $-\phi(i, j)$ , akkor ezt kifejezendő egy új élt is felvesszünk a levezetési gráfba, nem csak „üresen hagyjuk” a helyét.

A legtöbb esetben ez a módosítás „természetesen” alkalmazható. Részletesebb tárgyalást a negatív konkatenáció esete igényel, vagyis amikor  $-\phi_1 \dots \phi_n(i, j)$  levezetését kívánjuk megkeresni. Az erős negációs szemantika miatt a probléma nem redukálható arra, hogy  $+\phi_1 \dots \phi_n(i, j)$  bizonyítását megpróbáljuk elvégezni, majd ennek sikertelensége esetén  $-\phi_1 \dots \phi_n(i, j)$ -t bizonyítotttnak tekintjük.

Ha  $\phi = \phi_1 \dots \phi_n$  konkatenáció ( $n \geq 2$ ), akkor  $+\phi(w)$  bizonyításához elegendő taláni a  $w$  szónak *legalább* egy olyan  $\langle w_1, \dots, w_n \rangle$  felosztását, amelyre  $+\phi_1(w_1) \wedge \dots \wedge +\phi_n(w_n)$ . A komplementum közvetlen bizonyításához azonban azt kell belátni, hogy *minden* lehetséges partícióban  $-\phi_1(w_1) \vee \dots \vee -\phi_n(w_n)$ .

Mivel a felosztások lehetséges száma a konkatenáció hosszának exponenciális függvénye, ezért a binarizáció alkalmazása valamilyen formában elkerülhetetlen. A negatív konkatenáció arra redukálható, hogy egy  $(i, j)$  szakasz minden  $k$  osztópontú 2-felosztására  $-\phi_1 \dots \phi_{n-1}(i, k)$  és  $-\phi_n(k, j)$  közül legalább az egyiknek levezethetőnek kell lennie. Előbbi esetben prefix-, utóbbi esetben szuffixlevezetésről beszélünk. Az már „csak” implementáció kérdése, hogy ezek közül mindkét lehetőséget végigjárjuk-e, vagy csak abban az esetben követeljük meg a szabály maradékának a negatív levezetését, ha a másik nem járt sikerrel.

Levezetési erdők konstruáljuk híján az első módszer a kívánatos, ezzel ugyanis potenciálisan jelentős mennyiségű számítás megspórolható. Ha azonban a teljes levezetési gráfra szükségünk van, eféle optimalizációra nincs lehetőség. Megjegyezzük, hogy ugyanezen dilemma az összes diszjunktív szerkezetet érinti, azaz a pozitív diszjunktcióra és a negatív konkatenációra egyaránt alkalmazható.

A negált konkatenáció esete azért is különleges, mert az egyetlen olyan nyelvi struktúra, amelyben a bemenet tetszőleges (de egybefüggő) részét „átugorhatjuk” mindenféle megkötés nélkül. Ez mindenképpen újdonság a környezetfüggetlen elemzéshez képest, ahol ilyesmire nincs lehetőség, az egyes elemek szigorúan egymást követve kell szerepeljenek a bemenetben.

Earley algoritmusában egyetlen itemfajta van, az  $A \rightarrow \alpha \bullet \beta$  pontozott szabály, ami egyben a szabályban haladó automata állapotleírása is. Mivel esetünkben a szabályok jobb oldala teljesen általános szerkezetű formula, ezért a leírókat is némileg eltérően használjuk.

A leíróknak (itemeknek) alapvetően két fajtáját különböztetjük meg, a kiindulási és a teljesülési itemeket. A kiindulási itemek egy formula illesztésének bal, a teljesülési itemek jobb végét reprezentálják. Minden item tartalmaz egy előjeles formulát, amelynek az illesztését nyilvántartja és kiindulási itemek esetén hivatkozások egy halmazát a szüleikre. A konkatenációs itemek ezen felül tartalmaznak még egy természetes számot, a „pont” helyzetét, valamint a negatív konkatenációs leírók ezzen felül egy többértékű szótárat is, melynek funkcióját később részletezzük. A bemenet egy adott pozíciójához tartozó itemek halmazát egy generációnak nevezzük.

Kiindulási itemek esetén a halmaz azon itemekre hivatkozik, amelyekből az adott leíró származik, és egy indexet, ami megadja, hogy teljesülése esetén a szülőitemben szereplő formula melyik részformuláját elégíti ki. Teljesülési itemek ilyen hivatkozásokat nem tartalmaznak, viszont nyilvántartják, hogy mely kiindulási leírókból származnak. A teljesülési itemeknek két fajtáját különböztetjük meg, ezeket (a megfelelő kvantorok nyomán) egzisztenciális és univerzális itemeknek nevezzük.

A kétféle teljesülés között csak olyan formulák esetén van értelme különbséget tenni, amelyek részeik teljesülésének valamilyen aggregációjára épülnek. A terminális, nemterminális és negációs formulákat tehát mindig egzisztenciálisan teljesülőnek tekintjük, mert ennek kezelése egyszerűbb. Ugyancsak egzisztenciálisan teljesül a pozitív diszjunkció, a negatív konjunkció és a pozitív konkatenáció; univerzálisan teljesül a negatív diszjunkció, a pozitív konjunkció és a negatív konkatenáció.

Egy teljesülési item megjelenése csak egzisztenciális esetben jelenti azt, hogy egy item a bemenetben maradéktalanul felismerésre került; univerzális teljesülés esetén az item csak akkor tekinthető késznek, ha annak minden része teljesült.

Az algoritmus egy  $S$  kezdőszimbólum levezetéséhez két kezdőitemet rendel a kezdőgenerációban: a  $+S$  és  $-S$  itemeket, amelyek annak pozitív és negatív levezetését reprezenálják.

Hasonlóan az eredeti algoritmushoz, ez a változat is nagyrészt három részre osztható: a szkennel feladata az itemmek kiterjesztése, a prediktor új itemeket hoz létre míg a lezáró szakasz a teljesült itemek feldolgozásáért felelős.

Az algoritmust az Aycock és Horspool-féle  $\epsilon$ -eliminációval ([16]) tervezzük, vagyis minden művelethez hozzávesszük annak  $\epsilon$ -lezártját is. Ennek következményeképp a levezetési gráfban soha nem hozunk létre azonos generációkba tartozó itemek között élet.

A szkennel fázis a legegyszerűbb: az algoritmusban nyilvántartjuk azokat az itemeket, melyek terminálisal vagy  $\epsilon$ -nal címkézettek (akár pozitívan, akár negatívan), és ezek mindegyike esetén vizsgáljuk, hogy az

adott formula teljesül-e az item pozíciója és az aktuális pozíció között. Ha igen, egy, a formula egzisztenciális teljesülését igazoló teljesülési itemet hozunk létre az aktuális generációban.

A prediktor feladata az újabb leírók előállítását az alábbiak szerint:

- $A \pm \epsilon$  alakú itemek nem generálnak továbbiakat;
- $A \pm a$  alakú itemek, ahol  $a \in \Sigma$ , nem generálnak további itemeket;
- $A \pm A$  alakú itemek, ahol  $A \in \mathbf{V}$ , a  $\pm \mathbf{P}(A)$  itemet generálják;
- $A \pm \phi_1 \vee \dots \vee \phi_n$  alakú itemek a  $\pm \phi_1, \dots, \pm \phi_n$  itemeket generálják;
- $A \pm \phi_1 \wedge \dots \wedge \phi_n$  alakú itemek a  $\pm \phi_1, \dots, \pm \phi_n$  itemeket generálják;
- $A \pm \neg \phi$  alakú itemek a  $\mp \phi$  itemet generálják;
- $A + \phi_1 \dots \phi_n$  alakú itemek a  $+\phi_1$  itemet és annak  $\epsilon$ -rákövetkezőit generálják;
- $A - \phi_1 \dots \phi_n$  alakú itemek a  $-\phi_1, -*\phi_2, \dots, -*\phi_n$  itemeket generálják;

Vegyük észre, hogy itemek előjelét csak a negáció változtatja meg. Előfordul, hogy egy gyermeket több szülő generál, ilyenkor az adott itemet csak egyszer hozzuk létre, viszont a gyermekben minden szülőjére való hivatkozást tároljuk.

Az utolsó két szabályról ejtsünk még néhány szót. A pozitív konkatenáció előállítási szabálya megfelel az Aycok-Horspool-féle  $\epsilon$ -eliminációnak: a  $\phi_1$  itemet mindenképpen generáljuk, viszont ha ez kiüresedő, akkor a levezetést kezdetjük a  $\phi_2$  itemmel is; ha ez is kiüresedő,  $\phi_3$ -mal is, stb.

A negatív konkatenációban a  $*$  azt jelöli, hogy az item *bármilyen* prefix mellett teljesülhet, ami természetesen lehet az üres sztring is, ezért ezeket az itemeket a szülő generációjától kezdődően minden további generációban meg kell ismételni.

Tekintsük röviden át a lezáró szakasz működését.

A lezáró (completer) fázis teljesülési itemeket dolgoz fel. Teljesülési itemek keletkezhetnek a szkennelés fázisban, amikor a terminálisokat a bemenetre illesztjük, és a lezáró fázisban, amikor részformulák lezárásával azok szülei is lezárásra kerülnek. Egy  $j$ -edik generációbeli teljesülési itemhez pontosan egy  $i \leq j$ -beli nyitóitem tartozik. Ha a teljesülési item készen van, akkor az általa reprezentált részformula a nyitóitem minden szülőjében való előfordulását egy-egy újabb teljesülési item létrehozásával rögzítjük.

Ha a kész teljesülési item szülője (pontosabban: nyitóitemének szülője) pl. negáció, és a teljesülési item a  $\mp \phi$  formulát reprezentálja az  $(i, j)$  szakaszon, akkor az  $i$  generációbeli,  $\pm \phi$  formulával címkézett nyitóitem is teljesül  $j$ -ben.

Univerzális esetben, pl. ha a szülő pozitív konjunkció, a helyzet nem ilyen egyszerű, hiszen ennek teljesüléséhez az összes részformula teljesülésére szükség van. Pozitív konjunkció és negatív diszjunkció esetén tehát egy univerzális teljesülési item azt tartja számon, hogy az eredet-item részformulái közül melyek kerültek levezetésre. Ezen esetekben az univerzális teljesülési item akkor válik késszé, ha minden részformula teljesülése megtörtént.

Ha a szülő pozitív  $n$ -áris konkatenáció, és annak valamely  $r$ -edik részformulája teljesült, akkor, ha  $r < n$ , tovább kell haladnunk az olvasásban, így a  $+\phi_{r+1}$  itemet prediktáljuk a  $j$  generációban, melynek szülője az  $i$ -edik generációbeli konkatenációs eredet-item. Természetesen, ha  $\phi_{r+1}$  kiüresedő, akkor a rákövetkezőjével is folytathatjuk az olvasást, stb. Ha  $r = n$ , vagy ha  $r < n$  de az összes további részformula kiüresedő, akkor a teljes konkatenáció egzisztenciálisan teljesül a  $j$  pozíción.

Nézzük, miként oldható meg a negatív konkatenáció esete! Tfh. a  $-\phi_1 \cdots -\phi_n$  címkéjű eredet-item valamely  $r$ -edik gyermeke teljesül. Ahhoz, hogy a negatív konkatenáció  $-\phi_1 \cdots -\phi_r$  prefixét teljesültnek tekinthessük az  $(i, j)$  szakaszon, minden  $i \leq k \leq j$  osztópontra be kell bizonyítani a teljesülést. Ha  $r = 1$ , akkor a  $-\phi_1$  item csak az  $i$ -edik pozíción kezdődhetett, tehát  $-\phi_1$  levezetésével a  $j$  osztóponthoz bizonyítottuk a prefix-levezethetőséget. Ha  $r > 1$ , akkor  $\phi_r$  levezetése valamilyen  $k \geq i$  ponton kezdődött, és ezen  $k$  osztóponthoz tartozó szuffix-levezetést találtunk.

A negatív konkatenáció teljesülési itemei tehát mindig a konkatenáció valamely  $-\phi_1 \cdots \phi_r$  prefixének az  $(i, j)$  szakaszon való levezethetőségét követik, amely akkor teljesül, ha a felosztópontok mindegyikére teljesül. Ellentétben tehát a logikai műveletekkel, a konkatenáció teljesülési itemeiben számontartott részteljesülések száma nem a nyelvtan jellemzője, hanem az illesztett szakasz hosszánál eggyel nagyobb.

Ha valamely osztópontra sikerül szuffix-levezetést adni, akkor világos, hogy a levezetés végét adó pozíción megjelenő teljesülési itemen ezt a pozíciót bejegyezzük. Nem szükséges azonban, hogy minden pozícióhoz tartozzon szuffix-levezetés, sőt,  $-\phi_1 \cdots \phi_n$  úgy is levehethető lehet valamely  $(i, j)$  szakaszon, hogy közben egyetlen pozíción sem szuffix-levezethető. A  $-\phi_1 \cdots \phi_r$  prefixhez tartozó teljesülési itemben tehát tudnunk kell azt is, hogy a  $-\phi_1 \cdots \phi_{r-1}$  prefix milyen osztópontokra vezethető le, hiszen ezek bármely sztringgel (\*) kiegészítve  $-\phi_1 \cdots \phi_r$  egy adott osztóponthoz tartozó levezetését adják.

Mivel ezen pontok halmaza a konkatenációs formulától, a kezdőpozíciótól és a prefix elemszámától (azaz  $r$  értékétől) függnek, ezért érdemes lehet őket a nyitóitemekben tárolni. A negatív konkatenációk kezdőitemeiben tehát egy multiszótarat tárolunk, aminek kulcsa az  $r$  értéke, értékei pedig azon  $k$  pozíciók, amelyekhez  $(i, k)$  szakaszon  $-\phi_1 \cdots \phi_r$  levezethető.

Ha valamely  $-\phi_1 \cdots \phi_r$  prefixet az  $(i, j)$  szakaszon levezettünk és  $r < n$ , azzal egyben megtaláltuk  $-\phi_1 \cdots \phi_{r+1}$  ugyanezen szakaszhoz tartozó,  $j$  osztópontú levezetését is. Ez induktívan további prefixek, vagy akár a teljes item teljesüléséhez vezethet.

Az algoritmus pszeudokódját a **5. algoritmus** tartalmazza.

### 3.4 Visszirányú online elemző

Utolsó algoritmusunkban az LR stratégiát fogjuk alkalmazni logikai nyelvtanok elemzésére. Knuth eredeti, determinisztikus algoritmusát ([2]) helyett annak Tomita-féle generalizációját ([12]) alkalmazzuk, annak is a Nederhof és Sarbo által módosított  $\epsilon$ -GLR változatát ([14]).

A determinisztikus LR automata verme modellezhető egy olyan irányított, lineáris gráffal (láncolt listával), amelynek csúcsai automataállapotok, az éleket az állapotok között vezető átmenet során olvasott szimbólummal címkézzük. Kiterjesztés (shifting) során a láncot egy új csúccsal és éllel bővítjük; redukció során a lánc végéről eltávolítunk valamely szabály jobb oldalának megfelelő számú elemet, és a láncot redukáló szabály bal oldalán álló nemterminálissal kiterjesztjük.

Az LR automata valamely állapota konfliktusos, ha egyszerre több redukció alkalmazható (reduce/reduce), vagy a verem egyszerre redukálható és kiterjeszthető (shift/reduce). Konfliktusos esetben a standard LR algoritmus nem alkalmazható.

A GLR (generalized LR) algoritmus ezt a veremreprezentációt bővíti egy aciklikus gráffá (graph structured stack, GSS), amelynek minden csúcsa egy elérhető állapotot, a kezdőállapotból oda vezető élsorozat pedig az állapot eléréséhez olvasott bemenetet reprezentálja. Az egy adott bemeneti pozícióhoz tartozó csúcsokat egy generációnak nevezzük. Az algoritmus a bemenetben balról jobbra haladva az egyes generációkban létrehozza a terminálisokkal elérhető átmeneteket (shifter fázis), majd az összes lehetséges módon visszaolvassa a verem tartalmát és az így felismert szimbólumok számára ismételt új éleket hoz létre, amíg az aktuális generáció nem telítődik.

Nederhof és Sarbo  $\epsilon$ -eliminációja ugyanazon a megfontoláson alapul, mint ahogyan Aycock és Horspool eliminálta ([16]) az  $\epsilon$ -átmeneteket Earley algoritmusából. Egy nemterminálisnak mindig csak a nem- $\epsilon$  levezetései vagyunk kíváncsiak, hiszen az  $\epsilon$ -levezetések a bemenettől függetlenek és előzetesen, pusztán a nyelvtan alapján megkonstruálhatók.

PARSE( $S$ )

- 1: **if** NULLABLE[ $S$ ] **then**
- 2:     **yield** NULLABLE[ $S$ ] at position 0
- 3: **add**  $+S$  to  $U_0$
- 4: **add**  $-S$  to  $U_0$
- 5: **for each**  $j$  in  $[0..|w|]$  **do**
- 6:     SCANNER()
- 7:     COMPLETER()
- 8:     PREDICTOR()

CREATE-ITEM( $\phi, p, r = 0$ )

- 1: **add** a new item  $o$  with formula  $\phi$  to  $U_j$
- 2: **add**  $\langle p, r \rangle$  to PARENTS[ $o$ ]

SCANNER()

- 1: **for each** item  $\iota$  labeled  $\pm c$  in  $U_i$  **where**  $i < j$  **and**  $c \in \Sigma \cup \{\epsilon\}$
- 2:     **if**  $w[i, j] = \pm c$  **then**
- 3:         **create** an  $\exists$ -completion for  $\iota$
- 4: **for each** negative concatenation  $\iota$  labeled  $-\phi_1 \cdots \phi_n$  in  $U_i$  **where**  $i < j$
- 5:     **for each**  $r$  in  $[2..n]$  **do**
- 6:         CREATE-ITEM( $-\gamma_r, \iota, r$ )
- 7:         **if** NULLABLE[ $\phi_r$ ] =  $\perp$  **then**
- 8:             DO-NEGATIVE-CONCATENATION-COMPLETION( $\iota, r, j$ )

PREDICTOR()

- 1: **while** there is an unprocessed item  $\iota$  labeled  $\phi$  in  $U_j$  **do**
- 2:     **if**  $\phi$  is a grammar variable **then**
- 3:         CREATE-ITEM( $\pm P(\phi), \iota$ )
- 4:     **else if**  $\phi$  is  $\neg\phi_1$  **then**
- 5:         CREATE-ITEM( $\mp\phi_1, \iota$ )
- 6:     **else if**  $\phi$  is  $\pm\phi_1 \vee \cdots \vee \phi_n$  **or**  $\pm\phi_1 \wedge \cdots \wedge \phi_n$  **then**
- 7:         **for each**  $r$  in  $[1..n]$  **do**
- 8:             CREATE-ITEM( $\pm\phi_r, \iota, r$ )
- 9:     **else if**  $\phi$  is  $+\phi_1 \cdots \phi_n$  **then**
- 10:         **for each**  $r$  in  $[1..n]$  **do**
- 11:             **if** NULLABLE[ $\phi_1 \cdots \phi_{r-1}$ ] =  $\top$  **then**
- 12:                 CREATE-ITEM( $+\phi_r, \iota, r$ )
- 13:     **else if**  $\phi$  is  $-\phi_1 \cdots \phi_n$  **then**
- 14:         **for each**  $r$  in  $[1..n]$  **do**
- 15:             CREATE-ITEM( $-\phi_r, \iota, r$ )
- 16:             **if** NULLABLE[ $\phi_1 \cdots \phi_r$ ] =  $\perp$  **then**
- 17:                 **mark** position  $j$  for the  $r$ -prefix of  $\iota$  as complete

---

5. algoritmus: Earley-stílusú elemző logikai nyelvtanokhoz.

COMPLETER()

- 1: **while** there is an uprocessed completion  $\tau$  that is not a prefix-completion **do**
- 2:   **for each**  $\langle \iota, r \rangle$  in parents of  $\text{ORIGIN}[\tau]$
- 3:     **let**  $\phi$  be the formula labeling  $\iota$
- 4:     **if**  $\phi$  is a variable **then**
- 5:       **create** an  $\exists$ -completion for  $\iota$
- 6:       **if**  $\phi = \pm S$  **and**  $\iota$  is in  $U_0$  **then**
- 7:         **yield** the sign of  $\phi$  at position  $j$
- 8:       **else if**  $\phi$  is  $\pm \neg \phi_1$  **or**  $+\phi_1 \vee \dots \vee \phi_n$  **or**  $-\phi_1 \wedge \dots \wedge \phi_n$  **then**
- 9:         **create** an  $\exists$ -completion for  $\iota$
- 10:       **else if**  $\phi$  is  $-\phi_1 \vee \dots \vee \phi_n$  **or**  $+\phi_1 \wedge \dots \wedge \phi_n$  **then**
- 11:         **let**  $\gamma$  be the  $\forall$ -completion for  $\iota$
- 12:         **mark**  $\phi_r$  as complete in  $\gamma$
- 13:       **else if**  $\phi$  is  $+\phi_1 \dots \phi_n$  **then**
- 14:         DO-POSITIVE-CONCATENATION-COMPLETION( $\iota, r$ )
- 15:       **else if**  $\phi$  is  $-\phi_1 \dots \phi_n$  **then**
- 16:         DO-NEGATIVE-CONCATENATION-COMPLETION( $\iota, r$ , the position of  $\text{ORIGIN}[\tau]$ )

DO-POSITIVE-CONCATENATION-COMPLETION( $\iota, r$ )

- 1: **if**  $r = n$  **or**  $\text{NULLABLE}[\phi_{r+1} \dots \phi_n] = \top$  **then**
- 2:   **create** an  $\exists$ -completion for  $\iota$
- 3:   **for each**  $s$  in  $[r + 1 .. n]$  **do**
- 4:     **if**  $\text{NULLABLE}[\phi_{r+1} \dots \phi_s] = \top$  **then**
- 5:       CREATE-ITEM( $+\phi_s, \iota, s$ )

DO-NEGATIVE-CONCATENATION-COMPLETION( $\iota, r, k$ )

- 1: **if**  $r > 1$  **then**
- 2:   **let**  $\gamma$  be the  $\forall$ -completion for the  $r$ -prefix of  $\iota$
- 3:   **mark** position  $k$  as complete in  $\gamma$
- 4:   **if**  $\gamma$  is not complete **then**
- 5:     **return**
- 6:   **while**  $r < n$  **do**
- 7:     **mark** position  $j$  for the  $r$ -prefix of  $\iota$  as complete
- 8:     **let**  $\gamma'$  be the  $\forall$ -completion for the  $r + 1$ -prefix of  $\iota$
- 9:     **mark** position  $j$  as complete in  $\gamma'$
- 10:    **if**  $\gamma'$  is not complete **then**
- 11:     **return**
- 12:    **else**
- 13:      $r \leftarrow r + 1$

---

5. algoritmus: Earley-stílusú elemző logikai nyelvtanokhoz. (folyt.)

Az  $\epsilon$ -eliminációnak a felesleges munka elkerülésén túlmutató előnyei is vannak. Az egyik a helyesség: Tomita algoritmus nem minden esetben találja meg a bemenet levezetését, ami belátható, hogy az kiüresedő jobbvégű szabályoknak köszönhető, melyeket az  $\epsilon$ -elimináció helyesen megkerül. A másik a körmentesség: az így kapott GSS mindig körmentes, élei a bemenet nem nulla hosszú szakaszait fedik le.

A klasszikus GLR elemzőhöz hasonlóan az általunk módosított algoritmus is három fázisból áll, ezek az automataépítési, a shifter és reducer fázisok (melyek nagyvonalakban megfelelnek az Earley-elemző prediktor, szekner és lezáró fázisainak).

Az automataépítés során az átmenetek címkézésére jelölt formulákat alkalmazunk, kivéve a speciális  $\neg*$  jelölést, ami a tetszőleges szakasz fölötti átmenetet reprezentálja. A negált, konjunktív és diszjunktív formulák esetén ez az átmenet mindig egyetlen lépésből áll, az ezeket leíró itemeknek két állapotát különböztetjük meg, a kiindulási és a végállapotot, melyet a szokásos pontozásos jelöléssel  $\pm \bullet \phi$  és  $\pm \phi \bullet$  módon is írhatunk. A konkatenáció esetében az átmenetek számát a részformulák száma korlátozza ( $\epsilon$ -eliminált átmenetek esetén azoknál kevesebb is lehet). A negált konkatenációban szereplő részformulákat opcionálisnak jelöljük, ami azt jelenti, hogy az ilyen átmenettel rendelkező állapotpárok között  $*$ -átmenet is megengedhető.

Az automataépítési eljárás megenged bizonyos fokú rugalmasságot. Legyen  $\neg\phi$  egy opcionális negatív átmenet, azaz  $u \xrightarrow{\neg\phi} v$ , ahol  $u$  és  $v$  automataállapotok. Helyes az is, ha a  $\neg\phi$  és  $\neg*$  átmenetek számára külön állapotokat definiálunk, azaz  $u \xrightarrow{\neg*} v'$ , ahol  $v \neq v'$ . Ekkor az átmenetek a célállapotokat egyértelműen meghatározzák, viszont – attól függően, hány és milyen hosszú negatív konkatenációs itemet tartalmaz egy állapot – a  $*$  átmenet célállapota egy „szuperállapot” lesz, ami mindenféle opcionális átmenet eredményét tartalmazza. Ezzel szemben, ha a  $*$ -átmeneteket a  $\phi$ -átmenetekkel azonos állapotba irányítjuk, akkor egyrészt az állapotátmeneteket (a  $*$  esetén legalábbis) innestől egy többértékű függvény definiálja, másrészt a célállapotban szükségessé válik a bemenő élek címke szerinti megkülönböztetése.

Hasonló dilemmával szembesülünk a logikai műveletek konstrukciójánál is. Tegyük fel, hogy az  $u$  állapot tartalmaz egy  $+\neg \bullet \phi$  itemet. Célszerűbb-e, ha a  $\neg\phi$  és  $+\neg\phi$  átmenetek két különböző állapotba visznek, vagy rendeljük mindkettőhöz ugyanazt a célt? Az állapotok ilyen összevonásának itt is ugyanaz a hátránya, hogy ekkor a redukció során az éleket címkék szerint szűrni szükséges.

Az olvasó fázis a GLR esetén is triviális: amikor az algoritmus a következő pozícióra lép, megvizsgálja, hogy mely állapotokból lehetséges terminálisokon keresztül (ideértve az  $\epsilon$  üresszó-szimbólumot és a  $*$ -ot is) mozogni, és azokat a bemenettel összevetve hoz létre új állapotokat és éleket az aktuális generációban.

A reduktor szakasz feladata a komplexebb formulák építése. Minden redukció egy  $\langle u, e \rangle$  párral indul, ahol  $u$  az aktuális generációban lévő állapot,  $e$  pedig egy  $u$ -ból kiinduló él. A redukciónak alapvetően három esete van:

1. Az  $e$  élet feliratozó formulának szülője valamely változó, diszjunkció, konjunkció vagy negáció  $v$ -ben. Ha ezen formula egzisztenciálisan redukálható,  $v$ -ből azonnal létrehozuk az átmenetet; univerzális redukció esetén csak feljegyezzük a részformula teljesülését és az utolsó részformulánál végezzük el az átmenetet.
2. Az  $e$  élet címkéző  $+\phi_r$  formula valamely pozitív  $+\phi_1 \cdots \phi_r \cdots \phi_n$  konkatenáció része. Egy pozitív konkatenáció pontosan akkor redukálható, ha a formula után fennmaradó jobbvége kiüresedő, azaz  $\text{NULLABLE}[\phi_{r+1} \cdots \phi_n] = \top$ . Ekkor a jobbvéggel nem foglalkozunk és a redukciót  $e$  végpontján folytatjuk a  $+\phi_1 \cdots \bullet \phi_r \cdots \phi_n$  itemmel. Mivel az  $e$  élen való áthaladással biztosított, hogy a redukált szakasz nem nulla hosszú, ezért a további szimbólumokat – ha lehetséges – megpróbáljuk kiüresített és nem kiüresített módon is a gráfra illeszteni.
3. Az  $e$  élet címkéző formula valamely negatív  $-\phi_1 \cdots \phi_r \cdots \phi_n$  konkatenáció része. Ha  $e$  címkéje  $-\phi_r$ , akkor a  $\phi_{r+1} \cdots \phi_n$  szakaszt  $\epsilon$ -redukáljuk és  $u$ -t az  $e$  végpontjában végződő  $-\phi_1 \cdots \bullet \phi_r \cdots \phi_n$  parciális redukció egy felosztásához tartozó osztópontként regisztráljuk. Ha  $e$  címkéje  $-*$ ,  $u$  pontosan akkor osztópont, ha  $\text{NULLABLE}[\phi_{r+1} \cdots \phi_n] = \perp$ . A redukció folytatásakor hasonlóan járunk el, ha a konkatenáció valamely utószelete teljesül az aktuális csúcsnál, akkor abból bármilyen (akár  $\epsilon$ -eliminált) élen továbbhaladva az eggyel hosszabb konkatenáció egy adott osztópontra vonatkozó felosztását találtuk meg (suffix teljesülés).

Az eljárás során épített GSS méretére a következő felső korlátok vonatkoznak:

- a generációk száma  $O(|w|)$ ;
- egy generáción belül a létrejöheto csúcsok számát az automata állapottere korlátozza, ami csak a nyelvtannak függvénye, így a bemenet hosszában  $O(1)$  méretű;
- egy állapotból minden más, korábbi csúcsba legfeljebb két él vezet; ezek száma ismét a bemenet hosszának függvényében  $O(|w|^2)$ .

A teljes GSS mérete tehát  $O(|w|^2)$  nagyságrendű. Megjegyezzük, hogy az ehhez járuló konstans szorzók a nyelvtanból adódóan nagyok is lehetnek, de mivel ezek  $|w|$ -től függetlenek, ezért formálisan elhanyagolhatók.

Az algoritmus pszeudokódját a **6. algoritmus** tartalmazza.

PARSE( $S$ )

- 1: **let**  $s_0$  be a state with an empty kernel and items  $+\bullet S$  and  $-\bullet S$
- 2: **while** there is an unprocessed state  $s$  **do**
- 3:   CLOSURE( $s$ )
- 4: **create** a node labeled  $s_0$  in  $U_0$
- 5: **if** NULLABLE[ $S$ ] **then**
- 6:   **yield** NULLABLE[ $S$ ] at position 0
- 7: **for each**  $j$  in  $[1 \dots |w|]$  **do**
- 8:   SHIFTER()
- 9:   REDUCER()

CLOSURE( $s$ )

- 1: **while** there is an unprocessed item  $\iota$  in  $s$  that is not a completion item **do**
- 2:   **if**  $\iota$  is labeled  $-\epsilon$  or  $\pm c$  where  $c \in \Sigma$  **then**
- 3:     **add** the completion of  $\iota$  to the transition on the label of  $\iota$
- 4:   **else if**  $\iota$  is labeled  $\pm A$  where  $A \in \mathbf{V}$  **then**
- 5:     **create** a new item  $\iota'$  in  $s$  labeled  $\mathbf{P}(A)$  with the same sign as  $\iota$
- 6:     **add**  $\iota$  to the parents of  $\iota'$
- 7:     **add** the completion of  $\iota$  to the transition on  $\pm A$
- 8:   **else if**  $\iota$  is labeled  $\pm \neg \phi$  **then**
- 9:     **add** a new item  $\iota'$  labeled  $\mp \phi$  in  $s$
- 10:    **add**  $\iota$  to the parents of  $\iota'$
- 11:    **add** the completion of  $\iota$  to the transition on  $\pm \neg \phi$
- 12:   **else if**  $\iota$  is labeled  $\pm \phi_1 \vee \dots \vee \phi_n$  or  $\pm \phi_1 \wedge \dots \wedge \phi_n$  **then**
- 13:     **for each**  $\phi$  in  $\phi_1, \dots, \phi_n$  **do**
- 14:       **add** a new item  $\iota'$  labeled  $\pm \phi$  in  $s$
- 15:       **add**  $\iota$  to the parents of  $\iota'$
- 16:       **add** the completion of  $\iota$  to the transition on the label of  $\iota$
- 17:   **else if**  $\iota$  is labeled  $+\phi_1 \dots \bullet \phi_r \dots \phi_n$  **then**
- 18:     **add** a new item labeled  $+\phi_r$  to  $s$
- 19:     **add** the successor of  $\iota$  to the transition on  $+\phi_r$
- 20:     **if**  $\phi_r$  is nullable **and** ( $r < n$  **or** there is a right-nullable kernel item  $+\phi_1 \dots \phi_n$  in  $s$ ) **then**
- 21:       **add** the successor of  $\iota$  to  $s$
- 22:   **else if**  $\iota$  is labeled  $-\phi_1 \dots \bullet \phi_r \dots \phi_n$  **then**
- 23:     **add** a new item labeled  $-\phi_r$  to  $s$
- 24:     **add** the successor of  $\iota$  to the transition on  $-\phi_r$
- 25:     **mark** the transition on  $-\phi_r$  as optional
- 26:     **if**  $r < n$  **or** there is a kernel item  $-\phi_1 \dots \phi_n$  in  $s$  **then**
- 27:       **add** the successor of  $\iota$  to  $s$

---

6. algoritmus: GLR-stílusú elemző logikai nyelvtanokhoz.

SHIFTER()

- 1: **for each** node  $u$  **in** generations  $U_i$  **where**  $i < j$  **do**
- 2:   **for each** terminal transition  $t$  from  $u$  **do**
- 3:     **if**  $w[i, j]$  matches  $t$  **then**
- 4:       CREATE-EDGE( $u, t$ )
- 5:   **for each** optional transition  $-\phi$  from  $u$  **do**
- 6:     CREATE-EDGE( $u, -\phi, \text{false}$ )

REDUCER()

- 1: **while** there is a pending reduction  $\langle u, e \rangle$  in  $\Delta$  **do**
- 2:   FINISH-REDUCTION(TARGET[ $e$ ], LABEL[ $e$ ])
- 3:   **if**  $e$  is positive **then**
- 4:     **for each**  $\iota$  **in** KERNEL[ $u$ ] labeled  $+\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  **and** NULLABLE[ $\phi_{r+1} \cdots \phi_n$ ] **do**
- 5:       EXTEND-POSITIVE-REDUCTION(TARGET[ $e$ ],  $+\phi_1 \cdots \bullet \phi_r \cdots \phi_n$ )
- 6:     **while** there is an unprocessed continuation  $\langle v, +\phi_1 \cdots \phi_r \bullet \cdots \phi_n \rangle$  in  $\Delta$  **do**
- 7:       CONTINUE-POSITIVE-REDUCTION( $v, +\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )
- 8:   **else**
- 9:     **for each**  $\iota$  **in** KERNEL[ $u$ ] labeled  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  **do**
- 10:      **if**  $e$  is labeled  $-\phi_r$  **or**  $e$  is labeled  $-*$  **and not** NULLABLE[ $\phi_{r+1} \cdots \phi_n$ ] **then**
- 11:       **if**  $e$  is labeled  $-\phi_r$  **then**
- 12:          **permanently mark** position  $j$  as complete for  $-\phi_1 \cdots \bullet \phi_r \cdots \phi_n$  at  $v$
- 13:       EXTEND-NEGATIVE-REDUCTION(TARGET[ $e$ ],  $-\phi_1 \cdots \bullet \phi_r \cdots \phi_n, j$ )
- 14:      **while** there is an unprocessed continuation  $\langle v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n \rangle$  in  $\Delta$  **do**
- 15:       CONTINUE-NEGATIVE-REDUCTION( $v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )

CREATE-EDGE( $u, \phi, l = \text{true}$ )

- 1: **let**  $v$  be the node in  $U_j$  reached by the transition on  $\phi$  from  $u$
- 2: **add** an edge  $e$  labeled  $\phi$  **if**  $l$  **else**  $-*$  from  $v$  to  $u$
- 3: **add**  $\langle v, e \rangle$  to  $\Delta$

FINISH-REDUCTION( $u, \phi$ )

- 1: **let**  $\iota$  be the item in  $u$  that is labeled  $\bullet \phi$
- 2: **for each**  $\iota'$  **in** PARENTS[ $\iota$ ] **do**
- 3:   **if**  $\iota'$  is a variable, negation, positive disjunction or negative conjunction **then**
- 4:     CREATE-EDGE( $u, \phi$ )
- 5:   **if**  $\phi = \pm S$  **then**
- 6:     **yield** the sign of  $\phi$  at position  $j$
- 7:   **else**
- 8:     **mark** the subformula  $\phi$  as complete for  $\iota'$  in  $u$
- 9:     **if** all subformulas of  $\iota'$  are complete in  $u$  **then**
- 10:      CREATE-EDGE( $u, \phi$ )

6. algoritmus: GLR-stílusú elemző logikai nyelvtanokhoz. (folyt.)

CONTINUE-POSITIVE-REDUCTION( $v, +\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )

- 1: **if** there is an item  $+\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  in  $v$  **and** NULLABLE[ $\phi_r$ ] **then**
- 2:     EXTEND-POSITIVE-REDUCTION( $v, +\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )
- 3: **if**  $+\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  is a kernel item in  $v$  **then**
- 4:     **for each**  $e$  **in** EDGES[ $v$ ] **do**
- 5:         EXTEND-POSITIVE-REDUCTION(TARGET[ $e$ ],  $+\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )

CONTINUE-NEGATIVE-REDUCTION( $v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )

- 1: **let**  $i \leftarrow$  the position of  $v$
- 2: **if**  $v$  contains the item  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  **then**
- 3:     EXTEND-NEGATIVE-REDUCTION( $v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n, i$ )
- 4: **if**  $v$  contains the kernel item  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  **then**
- 5:     **for each**  $e$  **in** EDGES[ $v$ ] **where**  $e$  is labeled  $-\phi_r$  **or**  $e$  is labeled  $-*$  **do**
- 6:         EXTEND-NEGATIVE-REDUCTION(TARGET[ $e$ ],  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n, i$ )

EXTEND-POSITIVE-REDUCTION( $v, +\phi_1 \cdots \phi_r \bullet \cdots \phi_n$ )

- 1: **if**  $r = 1$  **then**
- 2:     FINISH-REDUCTION( $v, +\phi_1 \cdots \phi_n$ )
- 3: **else**
- 4:     **add**  $\langle v, \pm\phi_1 \cdots \phi_r \bullet \cdots \phi_n \rangle$  **to**  $\Delta$

EXTEND-NEGATIVE-REDUCTION( $v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n, p$ )

- 1: **mark** position  $p$  as complete for  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  at  $v$
- 2: **if**  $r = n$  **or**  $-\phi_1 \cdots \phi_r \bullet \cdots \phi_n$  just got completed at  $v$  **then**
- 3:     **if**  $r = 1$  **then**
- 4:         FINISH-REDUCTION( $v, -\phi_1 \cdots \phi_n$ )
- 5:     **else**
- 6:         **add**  $\langle v, -\phi_1 \cdots \phi_r \bullet \cdots \phi_n \rangle$  **to**  $\Delta$

---

**6. algoritmus:** GLR-stílusú elemző logikai nyelvtanokhoz. (folyt.)

## 4. fejezet

# Összefoglalás

A metszet és különösen a komplement a Chomsky-féle környezetfüggetlen nyelvtanokba történő bevezetése a klasszikus logika korlátait feszegeti: a rekurzív nyelvi egyenletrendszer olyan, ellentmondó összefüggéseket is leírhat, melyek szerint egy nyelvben pontosan akkor tartalmazott egy szó, ha nem.

Ezen ellentmondások feloldására a klasszikus logikának nincs válasza. Az értékréses logikák ezzel szemben megengedik, hogy bizonyos állítások igazságát meghatározatlanként hagyjuk, ezáltal lehetővé téve ellentmondásos állítások modellezését. Ezen modellek közül a legismertebbek Fitting ([11]) és Gelder és mások ([13]) háromértékű modelljei, melyek közül mindkettő egy-egy következtetési operátor fixpontjaként definiálja a következtetések kimenetét.

Kountouriotis és mások ([20]) már foglalkozott a logikai nyelvtanok háromértékű elemzésével a megalapozott szemantika alapján, azonban úgy gondoljuk, a szintaktikai elemzés céljából Fitting modelljének alkalmazása célravezetőbb: az  $S \leftarrow S$  nyelvtannak azon interpretációt tekintjük kívánatos modelljének, amelyben minden szóhoz tartozó értékelés meghatározatlan.

A háromértékű logika alkalmazása a szintaktikai elemzés gyakorlatában azt jelenti, hogy nem csak a pozitív tudás reprezentációja válik szükségessé, hanem a negatívé is. Ehhez a klasszikus elemzőalgoritmusokat úgy szükséges módosítani, hogy azok a negatív levezetéseket is előállítsák.

Kutatásunk fő eredményének a klasszikus környezetfüggetlen elemzők nagyobb családjából egy-egy algoritmus alakítását tekintjük, melyek immáron helyesen számítják ki a szavak logikai nyelvtanok Fitting modelljében való tartalmazását:

- Unger elemzője (offline, top-down);
- a CYK algoritmus (offline, bottom-up);
- Earley generalizált LL-stílusú elemzője (online, top-down);
- a GLR elemző (online, bottom-up).

Az algoritmusok mindegyike  $O(n^3)$  legrosszabb eseti tárbonyolultság mellett és az offline elemzők esetében ugyanilyen legrosszabb eseti időbonyolultság mellett képes logikai nyelvtanok Fitting-modelljeinek kiszámítására. Ezen legrosszabb eseti korlátok megegyeznek a környezetfüggetlen elemzésből ismert legrosszabb eseti korlátokkal, noha az átlagos eseti komplexitás esetünkben jóval magasabb.

Noha a logikai nyelvtanok segítségével leírható nyelvek osztálya szigorúan bővebb a környezetfüggetlen nyelveknél, nyitott a kérdés, hogy ezen elméleti többleterő megéri-e az átlagos eseti idő- és tárbonyolultság növekedését, amit a negatív tudás bizonyításának szükséglete okoz, különösen annak fényében, hogy a környezetfüggetlen nyelvtanok esetén is számos stratégia áll rendelkezésünkre, amivel azok kifejezőereje a hatékonyság megőrzése mellett növelhető. A logikai nyelvtanok elméleti hátterét a praktikusság kérdésétől függetlenül érdekesnek és kutatásra érdemesnek tartjuk.

Jelen munka keretében csak felismerő (recognizer) algoritmusokat készítettünk, melyek nem tartalmazzák a levezetések helyességét igazoló adatszerkezetek (levezetési erdők) konstrukcióját. A levezetési erdők kérdése nem csupán elméleti szempontból érdekes, hanem a szintaktikai elemzők ezen keresztül fedik fel az elemzett sztring struktúráját.

### 4.1 Saját eredmények

Saját eredménynek tekintjük a Fitting-szemantika logikai nyelvtanokra való alkalmazásának ötletét, melyet – legjobb tudomásunk szerint – korábban még senki nem vizsgált. Ezzel összhangban a négy hagyományos elemzőalgoritmus ezen szemantikai modell kiszámítására való adaptációja is teljes egészében saját munka.

A dolgozatban szereplő bizonyítások mindegyike saját kidolgozás eredménye, bár a bizonyított állítások egy része korábbról már ismert a szakirodalomban. A két, tételként megjelölt, fontosabb állítást és azok bizonyításait eredetinek vélelmezzük.

# Irodalom

- [1] N. Chomsky, „Three models for the description of language”, *IEEE Transactions on Information Theory*, 2. évf., 3. sz., 113–124. old., 1956. szept. DOI: 10.1109/tit.1956.1056813.
- [2] D. E. Knuth, „On the translation of languages from left to right”, *Information and Control*, 8. évf., 6. sz., 607–639. old., 1965. dec. 1., ISSN: 0019-9958. DOI: 10.1016/S0019-9958(65)90426-2.
- [3] D. H. Younger, „Recognition and parsing of context-free languages in time  $n^3$ ”, *Information and Control*, 10. évf., 2. sz., 189–208. old., 1967. febr. 1., ISSN: 0019-9958. DOI: 10.1016/S0019-9958(67)80007-X.
- [4] A. V. Aho, „Indexed Grammars—An Extension of Context-Free Grammars”, *J. ACM*, 15. évf., 4. sz., 647–671. old., 1968. okt., ISSN: 0004-5411. DOI: 10.1145/321479.321488.
- [5] S. H. Unger, „A Global Parser for Context-Free Phrase Structure Grammars”, *Commun. ACM*, 11. évf., 4. sz., 240–247. old., 1968. ápr., ISSN: 0001-0782. DOI: 10.1145/362991.363001.
- [6] J. Earley, „An Efficient Context-Free Parsing Algorithm”, *Communications of the ACM*, 13. évf., 2. sz., 94–102. old., 1970. febr., Place: New York, NY, USA, ISSN: 0001-0782. DOI: 10.1145/362007.362035.
- [7] A. K. Joshi, L. S. Levy és M. Takahashi, „Tree adjunct grammars”, *Journal of Computer and System Sciences*, 10. évf., 1. sz., 136–163. old., 1975, ISSN: 0022-0000. DOI: 10.1016/S0022-0000(75)80019-5.
- [8] K. L. Clark, „Negation as Failure”, *Logic and Data Bases*, Springer US, 1978, 293–322. old. DOI: 10.1007/978-1-4684-3384-5\_11.
- [9] R. Reiter, „On Closed World Data Bases”, *Logic and Data Bases*, H. Gallaire és J. Minker, szerk., Boston, MA: Springer US, 1978, 55–76. old. DOI: 10.1007/978-1-4684-3384-5\_3.
- [10] K. R. Apt és M. H. van Emden, „Contributions to the Theory of Logic Programming”, *J. ACM*, 29. évf., 3. sz., 841–862. old., 1982. júl., ISSN: 0004-5411. DOI: 10.1145/322326.322339.

- [11] M. Fitting, „A Kripke-Kleene semantics for logic programs”, *The Journal of Logic Programming*, 2. évf., 4. sz., 295–312. old., 1985. dec. DOI: 10.1016/s0743-1066(85)80005-4.
- [12] M. Tomita, „An Efficient Context-Free Parsing Algorithm for Natural Languages”, *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI’85* sor., Los Angeles, California: Morgan Kaufmann Publishers Inc., 1985, 756–764. old., ISBN: 0934613028.
- [13] A. V. Gelder, K. A. Ross és J. S. Schlipf, „The well-founded semantics for general logic programs”, *Journal of the ACM*, 38. évf., 3. sz., 619–649. old., 1991. júl. DOI: 10.1145/116825.116838.
- [14] M.-J. Nederhof és J. J. Sarbo, „Increasing the Applicability of LR Parsing”, *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg, Netherlands és Durbuy, Belgium: Association for Computational Linguistics, 1993. aug., 187–202. old. cím: <https://aclanthology.org/1993.iwpt-1.16>.
- [15] A. Okhotin, „Conjunctive Grammars”, *Journal of Automata, Languages and Combinatorics*, 6. évf., 4. sz., 519–535. old., 2001. DOI: 10.25596/jalc-2001-519.
- [16] J. Aycock és R. N. Horspool, „Practical Earley parsing”, *The Computer Journal*, 45. évf., 6. sz., 620–630. old., 2002, Publisher: OUP.
- [17] A. Okhotin, „Boolean grammars”, *Information and Computation*, 194. évf., 1. sz., 19–48. old., 2004. okt. DOI: 10.1016/j.ic.2004.03.006.
- [18] A. Okhotin, „Generalized LR parsing algorithm for Boolean grammars”, *International Journal of Foundations of Computer Science*, 17. évf., 3. sz., 629–664. old., 2006. jún. DOI: 10.1142/s0129054106004029.
- [19] E. Scott, A. Johnstone és R. Economopoulos, „BRNGLR: a cubic Tomita-style GLR parsing algorithm”, *Acta Informatica*, 44. évf., 6. sz., 427–461. old., 2007. okt. 1., ISSN: 1432-0525. DOI: 10.1007/s00236-007-0054-z.
- [20] V. Kountouriotis, C. Nomikos és P. Rondogiannis, „Well-founded semantics for Boolean grammars”, *Information and Computation*, 207. évf., 9. sz., 945–967. old., 2009. szept. DOI: 10.1016/j.ic.2009.05.002.
- [21] E. Scott és A. Johnstone, „Recognition is not parsing — SPPF-style parsing from cubic recognisers”, *Science of Computer Programming*, 75. évf., 1. sz., 55–70. old., 2010, Special Issue on ETAPS 2006 and 2007 Workshops on Language Descriptions, Tools, and Applications (LDTA ’06 and ’07), ISSN: 0167-6423. DOI: 10.1016/j.scico.2009.07.001.

# Tárgymutató

- alapformula, 4
- atom, 3, 26
- diszjunktív formula, 26
- diszjunktív nyelvtan, 26
- előjeles atom, 4
- felosztás, 21
- Herbrand-bázis, 3
- Herbrand-példányosítás, 5
- Herbrand-univerzum, 3
- Horn-formula, 5
- háromértékű nyelv, 21
- indefinit program, 7
- indikátorfüggvény, 4, 22
- interpretáció, 26
- interpretáció bizonytalansága, 36
- karakterisztikus függvény, 4, 22
- kielégíthető formulahalmaz, 4
- konjunktív formula, 26
- konjunktív nyelvtan, 26
- konkatenációs formula, 26
- konzisztens nyelvpár, 21
- konzisztens állításhalmaz, 34
- következtetési operátor, 5
- logikai következmény, 4
- logikai nyelv szignatúrája, 3
- logikai nyelvtan, 25
- logikai törvény, 4
- megalapozatlansági halmaz, 16
- modell, 4
- modus ponens, 5
- negált formula, 26
- nulladrendű logikai nyelv, 3
- nullinterpretáció, 15, 36
- nyelvtani szabály, 26
- nyílt világ, 6
- paraméter, 3
- partíció, 9
- pozitív definit program, 5
- prenex formula, 4
- példányosítás, 5
- szabad változóelőfordulás, 3
- szimbólum, 21
- sztring, 21
- tartalmazás, 22
- term, 3
- termhelyettesítés, 4
- totális Herbrand-interpretáció, 4
- zárt formula, 3
- zárt világ, 6
- ábécé, 21
- állítás, 4