Multi-hipertabló az automatikus tételbizonyításban

Doktori (Ph.D.) értekezés

Kovásznai Gergely

Debreceni Egyetem
Informatikai Kar
Debrecen, 2007.

Multi-hipertabló az automatikus tételbizonyításban

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében
számítástudomány tudományágban

Írta: Kovásznai Gergely okleveles programtervező matematikus

Készült a Debreceni Egyetem Matematika- és Számítástudományok Doktori Iskola
Informatika programja keretében

Témavezető: Dr. Várterész Magda

A doktori szigorlati bizottság:

elnök:	Dr. .....
tagok:	Dr. .....
	Dr. .....

A doktori szigorlat időpontja: 2006. .....

Az értekezés bírálói:

	Dr. .....
	Dr. .....

A bírálóbizottság:

elnök:	Dr. .....
tagok:	Dr. .....
	Dr. .....
	Dr. .....
	Dr. .....

Az értekezés védésének időpontja: 2007. .....

Ezen értekezést a Debreceni Egyetem Matematika- és Számítástudományok Doktori Iskola Informatika programja keretében készítettem a Debreceni Egyetem doktori (Ph.D.) fokozatának elnyerése céljából.

Debrecen, 2007. .....

.....
Kovásznai Gergely
jelölt

Tanúsítom, hogy Kovásznai Gergely doktorjelölt 2001–2006 között a fent megnevezett doktori iskola Informatika programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.

Debrecen, 2007. .....

.....
Dr. Várterész Magda
témavezető

Tartalomjegyzék

1. fejezet

Bevezetés

Jelen disszertációban a klasszikus elsőrendű logika tételbizonyítással kapcsolatos kérdéseivel kívánunk foglalkozni. Church [9] nyomán tudjuk, hogy annak igazolása, hogy egy formula érvényes-e – vagy annak, hogy egy formulahalmaznak egy formula logikai következménye-e – eldönthetetlen probléma. Más szóval nem létezik általános tételbizonyító algoritmus. Mégis, a tételbizonyításra (az eldöntésprobléma megoldására) különböző speciális algoritmusokat dolgoztak ki, amelyeket széles körben használnak is. Ezeket általában *automatikus tételbizonyító algoritmusoknak* nevezik, melyek alapját képezik a tételbizonyító szoftvereknek. Ezek fontos alkotóelemei a széles körben használt szakértői rendszereknek és azokon alapuló olyan komplex alkalmazásoknak, mint például a párbeszédes rendszerek[1] és a párbeszédes ágensek[2] [14, 15, 16, 17]. Hasonlóképp a hátterét jelentik a logikai programozási nyelveknek, melyek közül legismertebb a Prolog, ahol a lineáris inputrezolúció ez a háttér. Church eredményét követően a logikával foglalkozó szakemberek két irányban fejtettek ki erős kutatást. Egyrészt megkísérelték az elsőrendű logikának olyan részlogikáit[3] megtalálni, melyekben az érvényesség kérdése a teljes elsőrendű logikával szemben eldönthető [1]. Másrészt kutatások folytak teljesen automatikus, emberi közbeavatkozást nem igénylő tételbizonyító módszerek irányában. Ezen két kutatási ág eredményeinek gyümölcseiként konkrét tételbizonyító alkalmazások jöttek létre, melyek közül a legismertebb és számítógépen is implementált a fent említett Prolog, mely a Horn-logikának nevezett részlogikában működik (mely egyébként nem eldönthető) és teljesen automatikus.

A disszertációban megkíséreljük túllépni a Horn-logika képezte határvonalat, és a teljes elsőrendű logikában használható automatikus tételbizonyító eljárást leírni. A 2. fejezetben azon kalkulusokat vesszük sorra és írjuk le nagy részletességgel, melyekből mi is meríteni kívánunk saját kalkulusunk megalkotásakor. Az analitikus tabló és a rezolúció különböző változatait írjuk le, majd eljutunk a hipertablóhoz a 2.3. fejezetben, melyre leginkább támaszkodtunk a munkánk során. A hipertabló – amellett,

[1] angol: *dialogue system*
[2] angol: *conversational agent*
[3] „fregmenseit", angol: *fragment*

hogy elsőrendű logikában helyes és teljes kalkulus – komoly automatizálási gondokkal küzd. Ezen problémák lehetséges orvoslására a 2.3.1. fejezetben leírt kalkulust hozzuk fel példaként, mely ígéretes kiindulópont saját kalkulusunk felé, hiszen többé-kevésbé utat mutat a fentebb említett automatizálási problémák leküzdéséhez, ám eközben a kalkulus elveszti a teljesség tulajdonságát.

A legfontosabb, saját eredményeket tartalmazó fejezet a 3. fejezet, melyben saját kalkulusunkat írjuk le. Ezen kalkulust *multi-hipertablónak* neveztük el, hiszen a hipertabló egy továbbfejlesztése. A kalkulusnak a 3.1. fejezetben részletes leírását adjuk, majd bizonyítjuk helyességét. A kalkulus teljességét a 3.2. fejezetben látjuk be – hozzátéve: a teljes elsőrendű logikában. Mint látható lesz, a kalkulus minden eleme teljesen automatizált. A további fejezetekben más, a multi-hipertablóhoz is kapcsolódó saját eredményeket írunk le. A 3.3. fejezetben egy tablókon alapuló *klózgeneráló algoritmust* ismertetünk, mely – szemben az irodalomban ismert klózgeneráló algoritmusokkal – lineáris. A 3.4. fejezetben megpróbáljuk körüljárni, hogyan lehetne a multi-hipertablóhoz *redundanciafogalmat* megadni; hiszen egy kalkulus csak akkor kecsegtethet bármiféle gyakorlati felhasználás lehetőségével, ha jól meghatározott redundanciafogalom is társul hozzá. A 3.5. fejezetben megmutatjuk, hogy a multi-hipertablóban benne rejlik és ki is használható a *heurisztikus tételbizonyítás* lehetősége, majd ehhez kapcsolódóan a 3.5.1. fejezetben azzal is foglalkozunk, hogyan szüntethető meg a klózok zártságára kimondott megszorítás, vagyis hogyan tehető képessé a kalkulus arra, hogy „igen”/„nem” válasz helyett összetett válasszal is tudjon szolgálni, mely arra vonatkozik, hogy a bizonyítandó formula a szabad változók mely helyettesítéseire válik tétellé.

1.1. Alapfogalmak

A jelen értekezésben kizárólag az *elsőrendű klasszikus logika* keretei között dolgozunk. A kapcsolódó fogalmakat az irodalomban megszokott módon értelmezzük és használjuk, ha attól el kívánunk térni, akkor azt mindig jelezni fogjuk.

1.1.1. Szintaktikai fogalmak

Logikai nyelv alatt egy $\langle \mathcal{V}ar, \mathcal{P}r, \mathcal{F}n \rangle$ hármast értünk, ahol $\mathcal{V}ar$ az (individuum)változók halmaza, $\mathcal{P}r$ a predikátumszimbólumok halmaza és $\mathcal{F}n$ a függvényszimbólumok halmaza. A nyelv minden $P \in \mathcal{P}r \cup \mathcal{F}n$ elemhez rendel egy nemnegatív egész számot, amelyet a P fokának nevezünk. Az irodalomban helyenként használt *konstansszimbólumokat* mint 0-fokú függvényszimbólumokat értelmezzük. Minden $P()$ termet és atomi formulát (azaz ha P predikátum- vagy függvényszimbólum foka 0) mint P-t fogjuk kiírni, a rövidség érdekében. Az általánosan használt *logikai jelek* mellett ($\neg$ – negáció, $\wedge$ – konjunkció, $\vee$ – diszjunkció, $\forall$ – univerzális kvantor, $\exists$ – egzisztenciális kvantor) a $\supset$ jelet használjuk az implikáció jelölésére. Ezen logikai jeleken kívül használni fogjuk a $\top$ és $\bot$ jeleket, melyek atomi formulák, és melyeket azonosan igazként, illetve azonosan hamisként interpretálunk.

1.1.1. Definíció. (Szabad változók halmaza)

Egy F logikai kifejezés (formula vagy term) *szabad változóinak halmazát* $\mathcal{FV}(F)$-fel fogjuk jelölni. Egy $\mathcal{H}$ formulahalmaz szabad változóinak halmaza:

$$\mathcal{FV}(\mathcal{H}) \quad = \quad \bigcup_{F \in \mathcal{H}} \mathcal{FV}(F) \quad . \qquad \square$$

1.1.2. Definíció. (Zárt/nyitott formula(halmaz))

Egy F formula(halmaz) *zárt*, ha $\mathcal{FV}(F) = \emptyset$; egyébként *nyitott*. $\square$

1.1.3. Definíció. (Formula lezártja[4])

Egy F formula *lezártjának* a $\forall x_1 \ldots \forall x_k F$ formulát nevezzük, ahol $\mathcal{FV}(F) = \{x_1, \ldots, x_k\}$; és $\forall F$-fel *jelöljük*. $\square$

1.1.4. Definíció. (Változóidegenség)

Az A és B formulákat akkor nevezzük *változóidegennek*, ha $\mathcal{FV}(A) \cap \mathcal{FV}(B) = \emptyset$. Egy formulahalmaz változóidegen, ha annak formulái páronként változóidegenek. $\square$

1.1.5. Definíció. (Helyettesítés[5])

Helyettesítés alatt olyan σ függvényt értünk, melynek értelmezési tartománya $\big(\mathcal{D}om(\sigma)\big)$ változóknak egy halmaza, értékkészlete $\big(\mathcal{R}ange(\sigma)\big)$ pedig termhalmaz, és $\mathcal{D}om(\sigma) \cap \mathcal{FV}\big(\mathcal{R}ange(\sigma)\big) = \emptyset$. $\square$

Sokszor, ha az kényelmesebb lesz, egy σ helyettesítésre használni fogjuk az

$$\{x_1/t_1, \ldots, x_k/t_k\}$$

jelölést, ahol $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_k\}$, és minden i-re $\sigma(x_i) = t_i$. Azaz σ-t mint x_i/t_i kifejezések halmazát adjuk meg.

Az üres halmazt, mint helyettesítést, *üres helyettesítésnek* nevezzük, és ϵ-nal jelöljük.

Ha σ egyelemű, azaz $\sigma = \{x/t\}$, akkor σ megadásakor a halmazoknál megkövetelt külső kapcsos zárójeleket elhagyhatjuk, azaz használni fogjuk a $\sigma = x/t$ megadási módot.

A későbbiekben a következő szóhasználattal fogunk élni: ha $\sigma(x) = t$, akkor azt mondjuk, hogy „x-be t-t helyettesítjük", illetve hogy „x-et behelyettesítjük t-vel".

Gyakran fogjuk használni két σ és θ helyettesítés *kompozícióját*, melyet szintén az irodalomban megszokott módon definiálunk és $\sigma\theta$-val jelölünk.

[4] másképp: univerzális lezártja, általánosítása
[5] angol: *(term) substitution*

1.1.6. Definíció. (Általánosabb helyettesítés)

A σ helyettesítés akkor *általánosabb* a θ helyettesítésnél, ha létezik olyan π helyettesítés, hogy $\sigma\pi = \theta$. □

1.1.7. Definíció. (Megengedett helyettesítés)

A σ helyettesítés *megengedett* az F formulán, ha semelyik $x \in \mathcal{FV}(F) \cap \mathcal{D}om(\sigma)$ változónak sincs olyan szabad előfordulása F-ben, mely benne lenne valamely $y \in \mathcal{FV}\big(\sigma(x)\big)$-et kötő kvantor hatáskörében. A σ megengedett egy $\mathcal{H}$ formulahalmazon, ha minden $F \in \mathcal{H}$ formulán megengedett. □

Ha egy σ helyettesítés megengedett egy F formulán, akkor σ-t *elvégezhetjük* F-en, azaz szimultán az összes $x \in \mathcal{D}om(\sigma)$ változó összes F-beli szabad előfordulását felülírjuk $\sigma(x)$-szel. A σ elvégzésével F-ből kapott formulát $F\sigma$-val jelöljük. Egy $\mathcal{H}$ formulahalmaz esetén

$$\mathcal{H}\sigma \quad = \quad \bigcup_{F \in \mathcal{H}} F\sigma \quad .$$

1.1.8. Definíció. (Példány)

Egy A formula akkor *példánya* egy B formulának, ha létezik olyan σ helyettesítés, hogy $A = B\sigma$. □

A következő két definíció a helyettesítések egy speciális osztályára, az ún. átnevezésekre[6] vonatkozik.

1.1.9. Definíció. (Átnevezés)

Az *átnevezés* egy olyan σ helyettesítést, ahol

(1) minden $x \in \mathcal{D}om(\sigma)$ esetén $\sigma(x)$ változó, és

(2) bármely $x, y \in \mathcal{D}om(\sigma)$ esetén ha $x \neq y$, akkor $\sigma(x) \neq \sigma(y)$. □

1.1.10. Definíció. (Átnevezett)

Az A formula akkor *átnevezettje* a B formulának, ha létezik olyan σ átnevezés, hogy $A = B\sigma$. □

Megjegyezzük, hogy egy formula átnevezettjei az adott formula speciális példányai, ahol a fent jelölt σ átnevezés.

A formulák osztályán belül kitüntetett szerepet szánunk az ún. literáloknak és klózoknak[7].

[6] angol: *(variable) renaming*
[7] angol: *clause*

1.1.11. Definíció. (Literál)

Literálnak nevezünk egy A vagy $\neg A$ formulát, ahol A atomi formula. Megkülönböztetünk *pozitív és negatív literálokat* a következő definíció szerint:

(1) A akkor és csak akkor pozitív, ha $A \neq \bot$;

(2) $\neg A$ akkor és csak akkor pozitív, ha A negatív.

Egy L literál esetén az L *alapját* $\underline{L}$-lel jelöljük, és a következőképpen definiáljuk:

$$\underline{A} = \underline{\neg A} \quad = \quad \begin{cases} \top & \text{, ha } A = \bot \\ A & \text{, egyébként} \end{cases} \qquad \square$$

1.1.12. Definíció. (Klóz)

Klóznak nevezünk egy $C = \forall x_1 \ldots \forall x_k (L_1 \vee L_2 \vee \ldots \vee L_n)$ *zárt* formulát, ahol minden L_i literál, $k \geq 0$, $n \geq 0$. A C *magja* alatt az $L_1 \vee L_2 \vee \ldots \vee L_n$ formulát értjük, melyet $\langle C \rangle$-vel jelölünk. $\square$

Mivel a klóz zárt formula, nem fog félreértéshez vezetni, ha kvantoros előtagjait nem írjuk le; azaz a fenti C klóz helyett magját, $L_1 \vee L_2 \vee \ldots \vee L_n$-t is használhatjuk. Ugyanezen klózt szokás csupán *literáljai multihalmazaként*[8], azaz $\{L_1, L_2, \ldots, L_n\}$-ként is jelölni és használni.

Az *üres klózt* (azaz mikor $n = 0$) $\bot$-val jelöljük.

Egy C klóz akkor *pozitív* (*negatív*), ha minden $L \in C$ literál pozitív (negatív).

1.1.13. Definíció. (Klóz új példánya)

Egy C *klóz új példánya* – adott $\mathcal{V}$ változóhalmaz mellett – C-nek egy olyan $\langle C \rangle \sigma$ példánya, ahol

(1) σ átnevezés,

(2) $\mathcal{D}om(\sigma) = \mathcal{FV}\big(\langle C \rangle\big)$ és

(3) $\mathcal{R}ange(\sigma) \cap \mathcal{V} = \emptyset$. $\square$

Felhívjuk a figyelmet arra, hogy az 1.1.12. definícióban kikötjük, hogy klózaink zártak, azaz nem tartalmazhatnak szabad változókat. Egészen a 3.5.1. fejezetig nem is lazítunk ezen a szigorú kikötésen. Csupán a 3.5.1. fejezetben fogjuk megvizsgálni, hogy ezen kikötés eltörlése milyen következményekkel jár.

[8] Multihalmaz alatt olyan halmazt értünk, melynek elemei ismétlődhetnek.

1.1.2. Szemantikai fogalmak

Annak jelölésére, hogy egy F formula egy $\mathcal{M}$ *modellben igaz*, az $\mathcal{M} \models F$ jelölést fogjuk alkalmazni. *Nyitott* F esetén $\mathcal{M} \models F$ akkor és csak akkor, ha $\mathcal{M} \models \forall F$.

Két A és B formulát *ekvivalensnek* nevezünk, ha bármely $\mathcal{M}$ modell esetén $\mathcal{M} \models A$ akkor és csak akkor, ha $\mathcal{M} \models B$; ennek jelölése: $A \sim B$.

A $P_1, \ldots, P_n$ $(n \geq 0)$ formuláknak akkor és csak akkor *logikai következménye* a K formula, ha minden olyan $\mathcal{M}$ modellben, ahol minden P_i esetén $\mathcal{M} \models P_i$, az $\mathcal{M} \models K$ is teljesül. Jelölése: $P_1, \ldots, P_n \models K$.

2. fejezet

Tételbizonyító módszerek

Jelen fejezetben azon kalkulusokat vesszük sorra, melyekre munkánk során támaszkodtunk. Ezen kalkulusok mind ún. *cáfoló* módszerek, vagyis egy formulahalmaz kielégíthetetlenségének bizonyítását végzik el pusztán szintaktikai eszközökkel. Mi most csak véges formulahalmazok kielégíthetetlenségét vizsgáljuk. A vizsgálatokra két – lényegében különböző – módszercsoportot ismertetünk:

(1) A 2.1. fejezetben ismertetjük az *analitikus tablót*, a 2.1.1. fejezetben a *szabadváltozós tablót*, majd a 2.1.2. fejezetben a *klóztablót*.

(2) A 2.2. fejezetben a bináris *rezolúciót* írjuk le, majd a *lineáris inputrezolúciót* a 2.2.1. fejezetben. Ezt követően bemutatjuk a *hiperrezolúciót* a 2.2.2. fejezetben.

A hiperrezolúció és a klóztabló egyfajta kombinációjaként alakult ki a *hipertabló* kalkulus (2.3. fejezet), mely alapvető ötleteket adott saját kalkulusunk megalkotásához. Mint részletezni fogjuk, a hipertabló komoly automatizálási gondokkal küzd, melyek kezelése messze nem triviális, így egy lehetséges utat mutatunk be a 2.3.1. fejezetben a *rigid hipertabló* kalkulus keretében. Ez lesz az az irány, amerre mi is el fogunk indulni a későbbiekben.

2.1. Analitikus tabló

Az analitikus tablók módszere – kihasználva a logikai formulák analitikus tulajdonságát – egy *fát* konstruál levezetésként. A fa csúcsai formulákkal címkézettek, s a fa ágai egy-egy modellt képviselnek, melyben az adott ág összes formulája igaz. Az analitikus tablók módszerét Smullyan publikálta az 1960-as évek végén [28].

A tabló ágait sok esetben – ahol ez tömörebb megfogalmazást tesz lehetővé – mint csúcsai *formuláinak multihalmazát* ragadjuk meg. Hasonlóképpen, a tablót sokszor mint *ágainak multihalmazát* értelmezzük.

Az analitikus tablókalkulus minden egyes „formulatípushoz” egy-egy levezetési szabályt ad meg, melyet a fa egy csúcsára alkalmazva a fa egyes ágait továbbépíthetjük,

illetve kettéágaztathatjuk. Smullyan 4 *formulatípust* különböztetett meg az elsőrendű logikában: α-, β-, γ- és δ-típust. Az elsőrendű formulák α, β, γ és δ osztályozása a 2.1. ábrán látható.

α	α_1	α_2
$A \wedge B$	A	B
$\neg\neg A$	A	–
$\neg\beta$	$\neg\beta_1$	$\neg\beta_2$

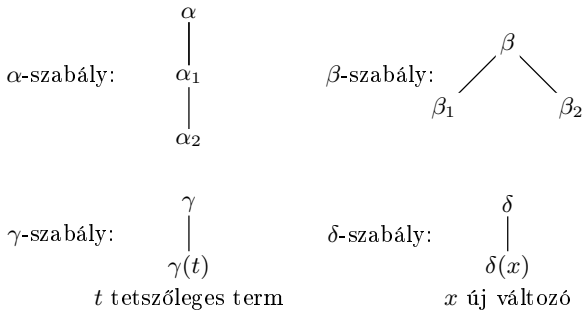
β	β_1	β_2
$A \vee B$	A	B
$A \supset B$	$\neg A$	B
$\neg\alpha$	$\neg\alpha_1$	$\neg\alpha_2$

γ	$\gamma(a)$
$\forall x A(x)$	$A(a)$
$\neg\delta$	$\neg\delta(a)$

δ	$\delta(a)$
$\exists x A(x)$	$A(a)$
$\neg\gamma$	$\neg\gamma(a)$

2.1. ábra. Formulatípusok

A 2.2. ábrán mutatjuk meg az analitikus tabló ezen formulatípusokhoz tartozó levezetési szabályait. Mint az ábrán látható, egy szabályt a tabló egy ágának egy A



2.2. ábra. Analitikus tabló – levezetési szabályok

formulájára alkalmazva

(1) α-, γ- és δ-szabály esetén: az ághoz egy vagy két formulát fűzünk;

(2) β-szabály esetén: az ágat elágaztatjuk, és az így kapott ágakhoz egy-egy formulát fűzünk.

2.1.1. Definíció. (Analitikus tabló)

Adott egy $\mathcal{F} = \{F_1, \ldots, F_n\}$ formulahalmaz. $\mathcal{F}$-hez a következő induktív definícióval adjuk meg az *analitikus tabló* fogalmát:

(1) $\big\{\{F_1, \ldots, F_n\}\big\}$ az $\mathcal{F}$ analitikus tablója. Azaz azon egyetlen ágból álló tabló, mely minden $\mathcal{F}$-beli formulához egy-egy csúcsot tartalmaz.

(2) Ha $\mathcal{T}$ az $\mathcal{F}$ analitikus tablója, akkor $\mathcal{T}'$ is az, amennyiben $\mathcal{T}'$-t a 2.2. ábrán látható szabályok valamelyikének alkalmazásával nyertük $\mathcal{T}$-ből. □

Egy tabló ágának *zártságát* (nyíltságát) az irodalomban megszokott módon értelmezzük: egy tabló egy $\mathcal{B}$ ága zárt, ha van olyan A formula, hogy $A \in \mathcal{B}$ és $\neg A \in \mathcal{B}$. Egy $\mathcal{T}$ tabló zárt, ha minden $\mathcal{B} \in \mathcal{T}$ ága zárt.

2.1.2. Tétel. (Analitikus tabló – Helyesség és teljesség)

Az analitikus tablók módszere helyes és teljes, azaz $\mathcal{F}$ akkor és csak akkor kielégíthetetlen, ha létezik hozzá zárt analitikus tabló.

A tételt Smullyan bizonyította könyvében [28].

A későbbiekben – főként a tárgyalandó tablókalkulusok helyességi bizonyításakor – szükségünk lesz a következő fogalomra:

2.1.3. Definíció. (Tabló által reprezentált formula)

Legyen

$$\mathcal{T} \quad = \quad \begin{array}{ccccc} & & A & & \\ & \swarrow & \downarrow & \searrow & \\ \mathcal{T}_1 & & \mathcal{T}_2 & \cdots & \mathcal{T}_k \end{array}$$

tabló, ahol A formula, valamint $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ tablók, $k \geq 0$. Definiáljuk induktíve a tablókon értelmezett F függvényt:

$$F(\mathcal{T}) \quad = \quad A \wedge \Big(\bigvee_{i=1}^{k} F(\mathcal{T}_i) \Big)$$

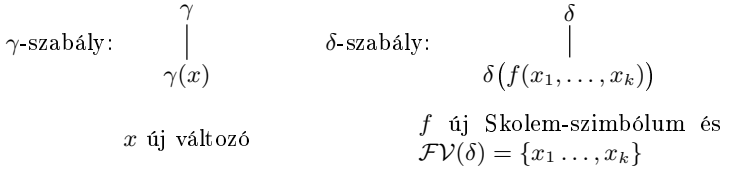
A $\mathcal{T}$ *által reprezentált formula* alatt az $F(\mathcal{T})$ formulát értjük. □

2.1.1. Szabadváltozós tabló, unifikáció

Amikor egy kalkulus automatizálhatósága merül fel kérdésként, azt kell megvizsgálni, hogy a levezetési szabályok szisztematikus alkalmazása nem veszélyezteti-e a levezetések végességét, azaz esetenként emberi közbeavatkozás szükséges-e. Az analitikus tablók módszerének α-, β- és δ-szabályai szisztematikusan alkalmazhatók, azzal a kikötéssel, hogy a tabló egy ágának egy α-, β- vagy δ-formulájára legfeljebb csak egyszer alkalmazunk szabályt. Itt jegyzendő meg, hogy a bizonyítandó formulahalmazon

végzett előkészítő lépéssel, az ún. skolemizálással [27] kiküszöbölhetők a tablóból a δ-formulák; ilyen előkészítés mellett a δ-szabály el is hagyható a kalkulusból. Visszatérve az automatizálhatóság kérdéséhez, a γ-szabály nem alkalmazható szisztematikusan, hiszen a szabály egy term tetszőleges kiválasztását írja elő. Márpedig a lehetséges termek száma (megszámlálhatóan) végtelen. Ráadásul egy ágon egy γ-formulára akár többször is meg kell engednünk szabály alkalmazását. Úgy tűnik tehát, hogy ezeken a pontokon valamiféle közbeavatkozás szükséges a tabló megkonstruálása során.

Ezen a problémán próbál enyhíteni a szabadváltozós tablók[1] módszere, melyet Fitting publikált [11]. A módszer lényege, hogy a kritikus elemet, a γ-szabálybeli t term kiválasztását, megpróbáljuk minél későbbre halasztani, mégpedig akkorra, amikor már látszik, hogy milyen termet érdemes t helyére választanunk. A későbbi időpillanatra halasztás úgy történik, hogy a γ-szabály alkalmazásakor $\gamma(a)$-ban a egy *új változó*. Ám a γ-szabálynak ez a módosítása problémát szül a δ-szabályok vonatkozásában: ha elhalasztottuk annak az eldöntését, hogy milyen termeket használjunk a γ-szabály alkalmazásokban, hogyan lehetünk biztosak abban, hogy a δ-szabály alkalmazásokban használt változók újak? Ezt a problémát *Skolem-szimbólumok* (azaz függvényszimbólumok) bevezetésével hidaljuk át, mégpedig nem előkészítő lépésként skolemizálunk, hanem „on the fly", tételbizonyítás közben. Tehát a γ- és δ-szabályok a 2.3. ábrán láthatóan módosulnak.



2.3. ábra. Szabadváltozós tabló – levezetési szabályok

Ha a γ-szabály által használt t term megválasztását későbbre halasztottuk, felmerül a kérdés, hogy mikor jön el a pillanat, amikor t-t megválaszthatjuk, és milyen t-t válasszunk. A kérdés megválaszolásához pár fogalomra szükségünk lesz.

2.1.4. Definíció. (Megengedett helyettesítés tablón)

Egy σ *helyettesítés* akkor *megengedett* egy $\mathcal{T}$ *tablón*, ha σ a $\mathcal{T}$ minden formuláján megengedett helyettesítés (lásd: 1.1.7. definíció). □

2.1.5. Definíció. (Helyettesítés tablón)

Legyen $\mathcal{T}$ egy *tabló* és σ egy $\mathcal{T}$-n megengedett *helyettesítés*. A σ $\mathcal{T}$-n való *elvégzésén* értjük azt a tablót, melyet $\mathcal{T}$-ből kapunk oly módon, hogy minden $\mathcal{T}$-beli A formulát $A\sigma$-val írunk felül. Jelölése: $\mathcal{T}\sigma$. □

[1] angol: *free-variable tableaux*

Ezek után a tablóépítés α- és β- (2.2. ábra), illetve γ- és δ-szabályaihoz (2.3. ábra) hozzáveszünk egy ötödiket, az ún. tablóhelyettesítési szabályt.

TABLÓHELYETTESÍTÉSI SZABÁLY[2]:

Ha $\mathcal{T}$ az $\mathcal{F}$ formulahalmaz tablója és σ megengedett helyettesítés $\mathcal{T}$-n, akkor $\mathcal{T}\sigma$ is $\mathcal{F}$ tablója lesz. □

Ezekkel a fogalmakkal még csak egy nagyon általános, nemdeterminisztikus szabadváltozós tablómódszert adtunk meg. A fentebb feltett kérdések, azaz hogy mikor válasszuk meg t-t és milyen t-t válasszunk, még mindig megválaszolatlanok. Az előző kérdések most már átfogalmazhatók a következő kérdéssé: melyik megengedett helyettesítést alkalmazzuk az adott tablón? Az evidens válasz: azt a helyettesítést, mely zárttá teszi valamelyik ágát a tablónak. Erre fogjuk használni az unifikációt (illesztő helyettesítést), és az ágak atomi lezárását.

2.1.6. DEFINÍCIÓ. (UNIFIKÁTOR)

A σ helyettesítés az $(E_1, F_1), \ldots, (E_n, F_n)$ *kifejezéspárok unifikátora* ($n \geq 0$), amennyiben minden (E_i, F_i) esetén $E_i\sigma = F_i\sigma$. □

2.1.7. DEFINÍCIÓ. (LEGÁLTALÁNOSABB UNIFIKÁTOR[3])

Kifejezéspároknak egy σ unifikátora azok *legáltalánosabb unifikátora*, amennyiben általánosabb minden unifikátoruknál. □

Két atomi formula legáltalánosabb unifikátorának (MGU) – amennyiben az létezik – előállítását végző algoritmust Robinson publikált először [25]. Ez az algoritmus a formulákat mint karaktersorozatokat kezeli, és kevéssé igazodik a logikai kifejezések (formulák és termek) induktív definíciójához. Az A.1. függelékben egy *rekurzív algoritmust* adunk meg kifejezéspárok MGU-jának kiszámítására. Az algoritmus az $\mathcal{U}()$ függvényt definiálja, melynek

- értelmezési tartománya: $U \times U$ hatványhalmaza, ahol U az összes termek és atomi formulák halmaza;
- értékkészlete: helyettesítések halmaza.

Az algoritmus által kiszámított $\mathcal{U}()$ tehát az $(E_1, F_1), \ldots, (E_n, F_n)$ kifejezéspárokhoz ($n \geq 0$) rendeli azok MGU-ját az $\mathcal{U}\big((E_1, F_1), \ldots, (E_n, F_n)\big)$ függvényhívás eredményeként, amennyiben az létezik. Mint az algoritmus leírásában is már használjuk, egyetlen (E, F) kifejezéspár unifikálása esetén rövidítésképpen használhatjuk az $\mathcal{U}(E, F)$ jelölést is.

Arra a kérdésre, hogy melyik helyettesítést végezzük el a tablón, most megadjuk a pontos választ, mégpedig oly módon, hogy a helyettesítések alkalmazását korlátozzuk az MGU-kra. A tablóhelyettesítési szabály helyett bevezetjük a következő tablóépítési szabályt, melyet MGU lezárási szabálynak [11] nevezünk el.

[2] angol: *tableau substitution rule*
[3] angol: *most general unifier (MGU)*

MGU LEZÁRÁSI SZABÁLY[4]:

(1) Legyen $\mathcal{T}$ az $\mathcal{F}$ formulahalmaz tablója, és legyen $\mathcal{B} \in \mathcal{T}$ ennek egy ága.

(2) Legyen továbbá $L_1, L_2 \in \mathcal{B}$ két ellentétes előjelű literál.
Ha $\sigma = \mathcal{U}(\underline{L}_1, \underline{L}_2)$ – azaz L_1 és L_2 alapjainak MGU-ja – létezik és σ megengedett helyettesítés $\mathcal{T}$-n, akkor $\mathcal{T}\sigma$ is $\mathcal{F}$ tablója lesz. □

Eljutottunk oda, hogy szabatosan tudjuk definiálni a szabadváltozós tabló kalkulust:

2.1.8. DEFINÍCIÓ. (SZABADVÁLTOZÓS TABLÓ)

Adott egy $\mathcal{F} = \{F_1, \ldots, F_n\}$ formulahalmaz. $\mathcal{F}$-hez a következő induktív definícióval adjuk meg a *szabadváltozós tabló* fogalmát:

(1) $\big\{\{F_1, \ldots, F_n\}\big\}$ az $\mathcal{F}$ szabadváltozós tablója.

(2) Ha $\mathcal{T}$ az $\mathcal{F}$ szabadváltozós tablója, akkor $\mathcal{T}'$ is az, amennyiben $\mathcal{T}'$-t a következő szabályok valamelyikének alkalmazásával nyertük $\mathcal{T}$-ből:

- a 2.2. ábrán látható α- és β-szabályok,
- a 2.3. ábrán látható γ- és δ-szabályok, és
- az MGU lezárási szabály. □

A szabadváltozós tabló konstrukciójából következik, hogy az MGU lezárási szabályban a $\mathcal{T}$ szabadváltozós tablón alkalmazott σ mindig megengedett $\mathcal{T}$-n.

2.1.9. TÉTEL. (SZABADVÁLTOZÓS TABLÓ – HELYESSÉG ÉS TELJESSÉG)

A szabadváltozós tablók módszere helyes és teljes.

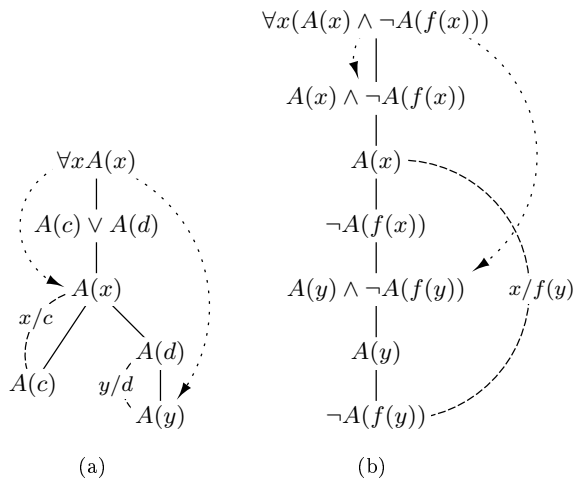
A bizonyítást Fitting adta meg könyvében [11].

Annak ellenére, hogy a szabadváltozós tabló automatikus módszert ad a γ-szabály alkalmazásaival kapcsolatos problémák kezelésére, van még mindig két nehezen automatizálható része a módszernek:

- Mikor van szükség γ-szabály alkalmazására? – Előfordulhat, hogy hiába dolgoztunk fel már ezelőtt egy γ-formulát, újra szükséges alkalmaznunk rá a γ-szabályt. Lásd a 2.4(a) ábrát, ahol az x/c helyettesítés elvégzése után újra alkalmaznunk kell a γ-szabályt a $\forall x A(x)$ formulára, csak így zárható le a jobb oldali ág.

- Hányszor van szükség γ-szabály alkalmazására? – Előfordulhat, hogy egy γ-formulára rögtön egymás után kétszer is kell a γ-szabályt alkalmazni. Lásd a 2.4(b) ábrát, ahol kétszer egymás után kell alkalmaznunk a γ-szabályt a $\forall x(A(x) \wedge \neg A(f(x)))$ formulára, csak így zárható le az ág.

[4] angol: *MGU atomic closure rule*



2.4. ábra. Problémás γ-szabály alkalmazások

2.1.2. Klóztabló

Gyakori, hogy a feldolgozandó formulákat valamilyen inicializáló lépés keretében speciális, kényelmesebb formájúra hozzuk. Egyik ilyen speciális forma a *klóz normálforma*, mely nem más, mint klózok halmaza. A klózok definíció szerint egyszerű esetszétválasztást írnak le, így egy klózhalmaz nem más, mint ilyen esetszétválasztások együttese. Bármely $\mathcal{F}$ formulahalmazt egy $\mathcal{C}$ klózhalmazzá alakíthatunk, ahol $\mathcal{F}$ akkor és csak akkor kielégíthető, ha $\mathcal{C}$ is kielégíthető. Ezen konverzióra, melyet klózgenerálásnak[5] hívnak, számtalan módszer létezik [11, 21, 22]; a 3.3. fejezetben egy saját, lineáris bonyolultságú algoritmust mutatunk be. A disszertáció hátralévő részében minden $\mathcal{C}$ klózhalmazról feltesszük, hogy nem tartalmazza az üres klózt, azaz $\bot \notin \mathcal{C}$.

Mivel a klóz normálforma igen egyszerű és áttekinthető, az ezen működő tételbizonyító módszerek is hasonló előnyökkel bírnak:

- könnyen áttekinthető levezetések,
- könnyű implementálhatóság,
- (viszonylag) egyszerű teljességi bizonyítás,
- a különböző módszerek könnyű összehasonlíthatósága.

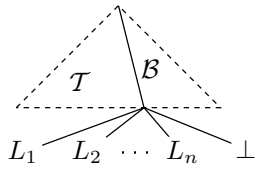
[5] angol: *clause generation*

A következőkben leírjuk a klóztablók módszerét [13], egy klózokon működő tablókalkulust. Az értekezés hátralévő részében csak klózokkal operáló tablókalkulusokkal fogunk dolgozni, ezért célszerű a következő definíció keretében egy tömör jelölést bevezetnünk.

2.1.10. Definíció. (Klóz csatolása a tabló egy ágához)

Egy $C = \{L_1, L_2, \ldots, L_n\}$ *klóznak* a $\mathcal{T}$ tabló egy $\mathcal{B} \in \mathcal{T}$ ágához való *csatolása* alatt értjük a

$$\mathcal{T}+^{\mathcal{B}}C \quad = \quad$$



tablót. □

A fenti definícióban C minden egyes literáljával és a $\bot$-val egyenként kiegészítjük a $\mathcal{B}$ ágat, egy-egy új ágat hozva létre ezzel; a tabló egyéb ágai változatlanok maradnak.

2.1.11. Definíció. (Klóztabló[6])

Adott egy $\mathcal{C}$ klózhalmaz. $\mathcal{C}$-hez a következő induktív definícióval adjuk meg a *klóztabló* fogalmát:

(1) Inicializációs szabály[7]:
$\{\{\top\}\}$ a $\mathcal{C}$-nek klóztablója.

(2) Kiterjesztési szabály[8]:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek klóztablója, és legyen $\mathcal{B} \in \mathcal{T}$.

(b) Legyen továbbá C egy $\mathcal{C}$-beli klóznak egy új példánya.

Ekkor $\mathcal{T}+^{\mathcal{B}}C$ is $\mathcal{C}$ klóztablója.

(3) Lezárási szabály[9]:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek klóztablója, és legyen $\mathcal{B} \in \mathcal{T}$.

(b) Legyen továbbá $L_1, L_2 \in \mathcal{B}$ két ellentétes előjelű literál, úgy hogy $\sigma = \mathcal{U}(\underline{L}_1, \underline{L}_2)$ létezik.

Ekkor $\mathcal{T}\sigma$ is $\mathcal{C}$ klóztablója. □

[6] angol: *clause tableaux*
[7] angol: *initialization rule*
[8] angol: *extension rule*
[9] angol: *closure rule*

Tehát a kezdeti klóztabló egyetlen csúcsot tartalmaz, melyet $\top$-vel címkézünk (inicializációs szabály).

Később a tabló valamely ágához csatolni tudjuk valamely input klóz, azaz $\mathcal{C}$-beli klóz *új példányát* (kiterjesztési szabály), melynek literáljaival rögtön egy-egy új ágat hozunk létre. Kérdés, hogy ezen példányok mely $\mathcal{V}$ *változóhalmaz* mellett újak (lásd: 1.1.13. definíció)? A később tárgyalandó klózokon működő tablókalkulusok esetén is, és most is a következőképpen határozzuk meg $\mathcal{V}$-t:

$$\mathcal{V} \quad = \quad \mathcal{FV}(\mathcal{T}) \quad .$$

A lezárási szabály nem más, mint az előző fejezetben ismertetett MGU lezárási szabály.

2.1.12. Tétel. (Klóztabló – Helyesség és teljesség)

A klóztablók módszere helyes és teljes.

A helyesség bizonyítása nem ütközik különösebb nehézségbe, hiszen a klóztabló levezetési szabályait tekintve nem más, mint egy klózhalmazon működő szabadváltozós tabló (MGU lezárási szabállyal). A 2.1.11.(2) szabály helyettesíthető lenne a szabadváltozós tabló γ- és β-szabályainak valahányszori alkalmazásával. A γ-szabály alkalmazásoknak természetesen az új példány előállítása felel meg. A teljesség bizonyítása összetettebb, a bizonyítás megtalálható Hähnle könyvfejezetében [13] a 128. oldalon.

A 2.4(b) ábra által reprezentált probléma már magával a klóz normálformával megoldódik, hiszen $\forall x(A(x) \wedge \neg A(f(x)))$ formula két klózra bomlik szét, melyek különkülön példányosíthatók. A 2.4(a) ábra által mutatott problémára viszont még mindig nem nyújt orvosságot a klóztabló, ezért a kalkulusnak további finomítása szükséges, mellyel a 2.3. fejezetben fogunk megpróbálkozni.

2.2. Rezolúció

A rezolúciós módszer elsőrendű logikában Robinson [25] nevéhez fűződik, aki azt az 1960-as évek végén publikálta. A rezolúció, akárcsak a klóztabló, *klózhalmazok* kielégíthetetlenségének vizsgálatára használható. A rezolúciós kalkulusokról a [2] könyvfejezetben találunk részletes összefoglalót.

2.2.1. Definíció. (Bináris rezolvens)

Legyenek C_1 és C_2 klózok. Legyenek $C_i = C_i' \vee L_i$ alakúak ($i \in \{1,2\}$), ahol L_1 és L_2 ellentétes előjelű literálok, úgy hogy $\sigma = \mathcal{U}\big(\underline{L}_1, \underline{L}_2\big)$ létezik. Ekkor C_1 és C_2 *bináris rezolvense* a $(C_1' \vee C_2')\sigma$ klóz. □

Könnyen bizonyítható a következő állítás:

2.2.2. Lemma.

Legyen a C klóz a C_1 és C_2 klózok bináris rezolvense. Ekkor $C_1, C_2 \models C$.

Szükségünk lesz a következő fogalomra [8]:

2.2.3. Definíció. (Klóz faktora)

Legyen C egy klóz, melyben $L_1, L_2, \ldots, L_k \in C$ azonos előjelű literálok és $\sigma = \mathcal{U}\big(\underline{L}_1, \underline{L}_2, \ldots, \underline{L}_k\big)$ létezik. Ekkor a $\langle C\rangle\sigma$-t a C *faktorának* nevezzük.

Ha egy C klóznak a C' faktora, és C'-nek a C'' faktora, akkor C-nek a C'' is faktora. □

2.2.4. Megjegyzés.

Triviális,hogy ha C' a C klóz faktora, akkor $C \models C'$. □

Bevezetjük az elsőrendű rezolvens fogalmát [8]:

2.2.5. Definíció. (Elsőrendű rezolvens)

A C_1 és C_2 klózok *elsőrendű rezolvense* egy a következő bináris rezolvensek közül:

(1) C_1 és C_2 bináris rezolvense;

(2) C_1-nek és C_2 faktorának a bináris rezolvense;

(3) C_1 faktorának és C_2-nek a bináris rezolvense;

(4) C_1 faktorának és C_2 faktorának a bináris rezolvense. □

2.2.6. Megjegyzés.

A 2.2.2. lemma és a 2.2.4. megjegyzés alapján: ha a C klóz a C_1 és C_2 klózok elsőrendű rezolvense, akkor $C_1, C_2 \models C$. □

2.2.7. Definíció. (Rezolúciós levezetés)

Egy $\mathcal{C}$ klózhalmazból való *rezolúciós levezetés* klózoknak egy olyan véges $C_1, C_2, \ldots, C_n$ $(n \geq 1)$ sorozata, ahol minden C_i

(1) vagy egy $\mathcal{C}$-beli klóz új példánya;

(2) vagy a C' és C'' klózok elsőrendű rezolvense, ahol $C', C'' \in \{C_1, \ldots, C_{i-1}\}$.

Ha $C_n = \bot$, akkor $C_1, \ldots, C_n$ a $\mathcal{C}$ *rezolúciós cáfolata.* □

Kérdés, hogy ha a rezolúciós levezetés C_i eleme egy input klóz egy *új példánya*, akkor C_i mely $\mathcal{V}$ *változóhalmaz* mellett új (lásd: 1.1.13. definíció)? A később tárgyalandó rezolúciós kalkulusok esetén is, és most is a következőképpen határozzuk meg $\mathcal{V}$-t:

$$\mathcal{V} \quad = \quad \mathcal{FV}\Big(\big\{\langle C_1\rangle, \ldots, \langle C_{i-1}\rangle\big\}\Big) \quad .$$

A 2.2.6. megjegyzés alapján egy $\mathcal{C}$-hez adott rezolúciós levezetés összes klóza $\mathcal{C}$ *logikai következménye*. Ez vezet ahhoz a felismeréshez, hogy ha sikerül rezolúciós cáfolatot előállítanunk $\mathcal{C}$-hez, akkor $\mathcal{C} \models \bot$, azaz $\mathcal{C}$ kielégíthetetlen.

2.2.8. Tétel. (Rezolúció – Helyesség és teljesség)

A rezolúciós kalkulus helyes és teljes.

A helyesség bizonyítását már a fentiekben megadtuk. A kalkulus teljességének bizonyítását Robinson [25] adta meg. Ezzel kapcsolatban felhívjuk a figyelmet a 2.2.5. definícióban a faktorizáció fontosságára, ugyanis anélkül a rezolúciós kalkulus nem teljes.

A rezolúció automatizálhatósága három ponton ütközik nehézségekbe:

- Hogy döntsük el, hogy a levezetés következő eleme input klóz legyen vagy pedig rezolvens?
- Hogy döntsük el, hogy egy klózt vagy annak faktorát használjuk?
- Rezolváláskor melyik két klózt rezolváljuk?

Erre próbálnak szisztematikus megoldást adni a különböző rezolúciós levezetési stratégiák.

2.2.1. Lineáris inputrezolúció, Prolog

A legismertebb rezolúciós levezetési stratégia az ún. lineáris rezolúciós stratégia.

2.2.9. Definíció. (Lineáris rezolúciós levezetés)

Egy *lineáris rezolúciós levezetés* olyan $C_1, M_1, C_2, M_2, \ldots, C_n$ rezolúciós levezetés, ahol minden $i > 1$-re C_i: a C_{i-1} és az M_{i-1} elsőrendű rezolvense. □

Látható, hogy a lineáris rezolúciós stratégia a fentebb feltett három kérdés közül az elsőnek és a harmadiknak egy-egy részét válaszolja meg: minden *centrális klóz* (C_i, $i > 1$) rezolvens. A mellékklózok (M_i) viszont egyaránt lehetnek input- és rezolvens klózok.

Mivel a levezetés szabály ezúttal is a rezolvensképzés, a lineáris rezolúció helyes. A lineáris rezolúciós stratégia teljességére ad választ a következő tétel:

2.2.10. Tétel. (Lineáris rezolúció – Helyesség és teljesség)

A lineáris rezolúciós kalkulus helyes és teljes [8].

Implementálhatóság és hatékonyság szempontjából előnyös, ha a *mellékklózok* kiválasztására is megszorításokat teszünk. Az egyik erre irányuló rezolúciós stratégia a lineáris inputrezolúciós stratégia.

2.2.11. Definíció. (Lineáris inputrezolúciós levezetés)

Egy $\mathcal{C}$ klózhalmazból való *lineáris inputrezolúciós levezetés* olyan $C_1, M_1, C_2, M_2, \ldots, C_n$ lineáris rezolúciós levezetés, ahol minden M_i egy $\mathcal{C}$-beli klóz új példánya. □

2.2.12. Tétel. (Lineáris inputrezolúció – Helyesség)

A lineáris inputrezolúció helyes, de nem teljes.

A kalkulus helyessége ezúttal is nyilvánvaló. A teljesség cáfolásához elegendő egy ellenpéldát mutatni. Például az $\{ A \vee B, \neg A \vee B, \neg A \vee \neg B, A \vee \neg B \}$ klózhalmaz kielégíthetetlen és lineáris inputrezolúcióval nem vezethető le belőle a $\bot$.

Megmutatható azonban, hogy a lineáris inputrezolúció teljes a formulák egy speciális osztályára, a Horn-klózokra nézve.

2.2.13. Definíció. (Horn-klóz)

A *Horn-klóz*[10] olyan klóz, mely legfeljebb egy pozitív literált tartalmaz. □

2.2.14. Tétel. (Lineáris inputrezolúció – Teljesség)

A lineáris inputrezolúció teljes Horn-klózok halmazain (vagy másképp: Horn-logikában).

A tétel bizonyítása megtalálható Gallier könyvében [12][11].

2.2.2. Hiperrezolúció

A hiperrezolúció a rezolúciós kalkulusok egy olyan fajtája, mely – a lineáris inputrezolúcióval ellentétben – nem követel meg semmiféle korlátozást az input klózok alakjával kapcsolatosan. Továbbá a hiperrezolúció meglehetősen effektív is, mivel a rezolválhatóságra komoly megszorítást tesz, s ezáltal elég célirányossá válnak a rezolúciós levezetések.

A hiperrezolúciós stratégiát Robinson publikálta 1965-ben [26], ugyanabban az évben, mikor az elsőrendű rezolvens és a rezolúciós levezetés ötletét publikálta [25]. Kétfajta hiperrezolúciós stratégiát ismerünk. *Pozitív* hiperrezolúció esetén a hiperrezolvensek csupa pozitív literálból állnak. *Negatív* hiperrezolúció esetén a hiperrezolvensek literáljai mind negatívak.

2.2.15. Definíció. (Hiperrezolvens)

(1) Legyen $C^{\ominus}$ egy klóz, melyben a negatív literálokat $L_1^{\ominus}, \ldots, L_k^{\ominus}$-val jelöljük, ahol $k \geq 1$.

(2) Legyen $\{C_1^{\oplus}, \ldots, C_k^{\oplus}\}$ pozitív klózoknak egy halmaza.

(3) Legyenek továbbá $L_i^{\oplus} \in C_i^{\oplus}$, úgy hogy $\sigma = \mathcal{U}\big((\underline{L}_1^{\ominus}, \underline{L}_1^{\oplus}), \ldots, (\underline{L}_k^{\ominus}, \underline{L}_k^{\oplus})\big)$ létezik. Ekkor a

$$\Big(C^{\ominus} \backslash \{L_1^{\ominus}, \ldots, L_k^{\ominus}\} \cup \bigcup_{i=1}^{k} C_i^{\oplus} \backslash \{L_i^{\oplus}\} \Big)\sigma$$

klóz a $C^{\ominus}$ és $\{C_1^{\oplus}, \ldots, C_k^{\oplus}\}$ *pozitív hiperrezolvense*. □

[10] másnéven: *definit klóz*

[11] A könyvben az ún. SLD (*semi-linear definit*) rezolúció – mely a lineáris inputrezolúció egy speciális fajtája – teljességének bizonyítása található.

2.2.16. Definíció. (Hiperrezolúciós levezetés)

Egy $\mathcal{C}$ klózhalmazból való *pozitív hiperrezolúciós levezetés* klózoknak egy olyan véges $C_1, \ldots, C_n$ $(n \geq 1)$ sorozata, ahol minden C_i

(1) vagy egy $\mathcal{C}$-beli klóz új példánya;

(2) vagy a $C^{\ominus}$ és $\{C_1^{\oplus}, \ldots, C_j^{\oplus}\}$ pozitív hiperrezolvense, ahol: minden $C^{\ominus}, C_1^{\oplus}, C_2^{\oplus}, \ldots, C_j^{\oplus}$ egy-egy $\{C_1, \ldots, C_{i-1}\}$-beli klóz vagy annak faktora.

Ha $C_n = \bot$, akkor $C_1, \ldots, C_n$ a $\mathcal{C}$ *hiperrezolúciós cáfolata.* □

2.2.17. Megjegyzés.

A *negatív hiperrezolúció* is könnyen leírható a 2.2.15. és a 2.2.16. definíciók megfelelő átírásával: minden helyen a „pozitív" jelzőt írjuk át „negatív"-ra, és fordítva. □

2.2.18. Lemma.

Legyen a C klóz a $C^{\ominus}$ és $\{C_1^{\oplus}, \ldots, C_k^{\oplus}\}$ (pozitív vagy negatív) hiperrezolvense. Ekkor $C^{\ominus}, C_1^{\oplus}, \ldots, C_k^{\oplus} \models C$.

Bizonyítás.

Az állítás közvetlen következménye a 2.2.2. lemmának, hiszen a C k-szori (bináris) rezolválással előállítható $C^{\ominus}, C_1^{\oplus}, \ldots, C_k^{\oplus}$-ból. □

Mint már fentebb utaltunk rá:

2.2.19. Tétel. (Hiperrezolúció – Helyesség és teljesség)

A (pozitív vagy negatív) hiperrezolúció helyes és teljes [26].

Bár a hiperrezolúció elvben effektív, nehezen automatizálható, hiszen itt is felvethetők a következő kérdések:

- Hogy döntsük el, hogy a levezetés következő eleme input klóz legyen vagy pedig hiperrezolvens?
- Hogy döntsük el, hogy egy klózt vagy annak faktorát használjuk?
- Rezolváláskor mely klózokat rezolváljuk?

A hiperrezolúciós stratégia teljes, de nehezen automatizálható. A stratégia újabb kiegészítése szerint a $C^{\ominus}, C_1^{\oplus}, \ldots, C_k^{\oplus}$ hiperrezolvense akkor képezhető, ha előállítható belőlük a $C^{\ominus}$ klózzal kezdődő lineáris inputrezolúciós levezetéssel. Így a hiperrezolvensképzés hatékonnyá tehető e lehetőség felhasználásával.

2.3. Hipertabló

A hipertabló[12] kalkulus napjaink tételbizonyítással kapcsolatos kutatásainak egyik friss eredménye [6, 3, 4]. Bár a kalkulus lehetősége már a 70-es évek végén felvetődött [5], csak 1996-ban jelent meg kellő kidolgozottsággal a szakirodalomban (Baumgartner [3]). A hipertabló módszere a hiperrezolúció automatizálhatóságával kapcsolatos kérdésekre keresi a választ. Ennek érdekében egy „gazdagabb" adatstruktúrát vezet be a levezetés egészének irányítására – hiszen a hiperrezolúcióban használt *sor adatszerkezet* meglehetősen szegényes eszköz a levezetés során felmerülő információk tárolására és későbbi felhasználására. A tételbizonyításban jól ismert *tabló* eljárás fa adatszerkezete jól felhasználható a hiperrezolvens leírásában. Így lehetővé válik, hogy a sorhoz képest sokkal strukturáltabb formában tároljuk a levezetés „history"-ját, abból esetlegesen adatokat keressünk vissza, így segítve a levezetés következő lépésére vonatkozó döntési folyamatot.

A következő két definíció a Baumgartner által használt ún. tisztító helyettesítéseket adja meg, melyek használata a leginkább vitatott és kritizált eleme Baumgartner konstrukciójának.

2.3.1. Definíció. (Tiszta klóz[13])

Egy klóz *tiszta*, ha literáljainak halmaza változóidegen. □

2.3.2. Definíció. (Tisztító helyettesítés[14])

Egy π helyettesítés egy C klóznak *tisztító helyettesítése*, ha $\langle C \rangle \pi$ tiszta. □

A tisztító helyettesítések használatával tulajdonképpen klózoknak egyfajta „részben alappéldányait"[15] állítjuk elő: azon változókat, melyek a tisztaságot veszélyeztetik, alaptermekkel helyettesítjük, ám az egyéb változók (egyelőre) behelyettesítetlenek maradnak.

Az alábbiakban Baumgartner kalkulusát egy kissé egyszerűsített formában fogjuk leírni. Egy lényeges különbség, hogy Baumgartner-rel ellentétben a tabló ágait nem címkézzük explicit módon „zárt" és „nyitott" címkékkel, hanem a zártság szokásos definícióját használjuk. Egy másik különbség, hogy a kirezolvált literálokat – melyek triviálisan zárt ágakon helyezkednek el – nem vesszük fel a tablóba.

A kalkulus leírásában használni fogjuk az $\widehat{L}$ jelölést, mely az L literál egy új példányát jelenti (jobban mondva: $\widehat{L}$ a $\forall L$ klóz új példányának egyetlen literálja).

2.3.3. Definíció. (Hipertabló – Baumgartner konstrukciója)

Adott egy $\mathcal{C}$ klózhalmaz. $\mathcal{C}$-hez a következő induktív definícióval adjuk meg a *pozitív hipertabló* fogalmát:

(1) Inicializációs szabály:
$\{\{\top\}\}$ a $\mathcal{C}$-nek pozitív hipertablója.

[12] angol: *hyper tableaux*
[13] angol: *pure clause*
[14] angol: *purifying substitution*
[15] angol: *partial ground instantiations* [19]

(2) KITERJESZTÉSI SZABÁLY:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek pozitív hipertablója, és $\mathcal{B} \in \mathcal{T}$ annak egy ága.

(b) Legyen $C = L_1^{\ominus} \vee \ldots \vee L_k^{\ominus} \vee D \in \mathcal{C}$, $k \geq 0$, ahol $L_1^{\ominus}, \ldots, L_k^{\ominus}$ a C összes negatív literálja.

(c) Legyenek továbbá $L_1^{\oplus}, \ldots, L_k^{\oplus} \in \mathcal{B}$, úgy hogy $\sigma = \mathcal{U}\big((\underline{L}_1^{\ominus}, \widehat{\underline{L}}_1^{\oplus}), \ldots, (\underline{L}_k^{\ominus}, \widehat{\underline{L}}_k^{\oplus})\big)$ létezik.

Ekkor

$$\mathcal{T} +^{\mathcal{B}} D\sigma\pi \tag{d}$$

is $\mathcal{C}$ pozitív hipertablója, ahol π a $D\sigma$-nak tisztító helyettesítése. □

Tehát egy $\mathcal{C}$ klózhalmazhoz a következőképpen konstruálunk pozitív hipertablót. Az (1)-ben megkonstruáljuk a kezdeti tablót, mely egyetlen csúcsot tartalmaz, $\top$-vel címkézve. Ezek után a levezetés minden egyes lépésében kiválasztunk egy input klózt a (2)(b)-ben, és ennek minden negatív literálját kirezolváljuk a $\mathcal{B}$ ág egy-egy (pozitív) literáljával (illetve azok új példányaival), melyeket a (2)(c)-ben választunk ki. A hiperrezolvens, azaz $D\sigma$ minden literálját a (2)(d)-ben azághoz csatoljuk, egy-egy új ágat kapva ezzel, miután a rezolvenst tiszta formára hoztuk a π tisztító helyettesítésen keresztül.

2.3.4. MEGJEGYZÉS.

A Baumgartner-féle *negatív hipertabló* definícióját megkapjuk, ha a 2.3.3. definícióban minden „pozitív" jelzőt átírunk „negatív"-ra, és vice versa. □

Baumgartner konstrukciójának a *tisztító helyettesítések* használatában rejlik a gyengéje. Látható, hogy a 2.3.5. definícióban π előállítása *nincs automatizálva*. Tehát felmerül a nagyon is jogos kérdés: melyik tisztító helyettesítést válasszuk π-nek?

Baumgartner célja a tisztító helyettesítések alkalmazásával a következő összefüggésben rejlik:

$$\forall x(A \vee B) \sim \forall xA \vee \forall xB \text{ akkor és csak akkor, ha } x \notin \mathcal{FV}(A) \cap \mathcal{FV}(B) \quad .$$

Ennek következménye, hogy ha egy klóz tiszta, akkor a kvantoros előtagokat bevihetjük a klóz literáljai elé. Azaz a hipertablóban a literálokban előforduló változókra lokálisan végezhetünk helyettesítéseket; vagyis a σ MGU-t nem a teljes tablón végezzük el, mint a klóztabló esetén (a 2.1.11.(3)-ban), hanem csupán a csatolandó klózon (a 2.3.3.(2)(d)-ben). Ugyanez a magyarázata annak, hogy míg a klóztabló esetén klózok új példányait csatoljuk a tablóhoz (a 2.1.11.(2)-ben), addig hipertabló esetén magát az input klózt használjuk rezolválásra (a 2.3.3.(2)(b)-ben).

Vajon megéri-e a kalkulusba egy nem automatizált lépést bevezetni csupán a lokális helyettesíthetőség előnyéért? A válasz egyértelműen: nem. Implementációkban helyettesítések tablón való globális elvégzése semmivel sem bonyolultabb, mint lokális

elvégzésük[16]. Ennek fényében adjuk meg a következő definícióban a hipertabló egy másik, általunk kidolgozott változatát, mely jól illeszkedik a disszertációban taglalt egyéb tablókalkulusok közé.

2.3.5. Definíció. (Hipertabló)

Adott egy $\mathcal{C}$ klózhalmaz. $\mathcal{C}$-hez a következő induktív definícióval adjuk meg a *pozitív hipertabló* fogalmát:

(1) Inicializációs szabály:
$\{\{\top\}\}$ a $\mathcal{C}$-nek pozitív hipertablója.

(2) Kiterjesztési szabály:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek pozitív hipertablója, és $\mathcal{B} \in \mathcal{T}$ annak egy ága.

(b) Legyen $C = L_1^{\ominus} \vee \ldots \vee L_k^{\ominus} \vee D$ a $\mathcal{C}$ egy klózának *új példánya*, $k \geq 0$, ahol $L_1^{\ominus}, \ldots, L_k^{\ominus}$ a C összes negatív literálja.

(c) Legyenek továbbá $L_1^{\oplus}, \ldots, L_k^{\oplus} \in \mathcal{B}$, úgy hogy
$\sigma = \mathcal{U}\big((\underline{L}_1^{\ominus}, \widehat{L}_1^{\oplus}), \ldots, (\underline{L}_k^{\ominus}, \widehat{L}_k^{\oplus})\big)$ létezik.

Ekkor

$$(\mathcal{T} +^{\mathcal{B}} D)\sigma'\pi \tag{d}$$

is $\mathcal{C}$ pozitív hipertablója, ahol

- $\sigma' = \big\{x/t \in \sigma \;\big|\; x \in \mathcal{FV}(D)\big\}$, és
- π a $D\sigma'$-nek tisztító helyettesítése, ahol $\mathcal{D}om(\pi) \subseteq \mathcal{FV}(D\sigma')$. □

Ezen változat az input klózoknak új példányait állítja elő (a (2)(b)-ben), illetve a helyettesítéseket a tablón globálisan végzi el (a (2)(d)-ben). Mivel a Baumgartner-féle hipertabló esetén a tablóban már szereplő formulákon (azaz $\mathcal{T}$ elemein) nem végzünk helyettesítéseket, csak az aktuálisan csatolt klózon, így most – a helyettesítések globális elvégzése miatt – a σ MGU által megadott behelyettesítések közül eliminálunk azokat, melyek nem a D szabad változóira vonatkoznak. Ugyanezen okból a π tisztító helyettesítésre is egy (nem lényegbeli) megszorítást kell tennünk, annak érdekében, hogy a π csak $D\sigma'$-beli szabad változókat helyettesíthessen be.

2.3.6. Megjegyzés.

A *negatív hipertabló* definícióját megkapjuk, ha a 2.3.5. definícióban minden „pozitív" jelzőt átírunk „negatív"-ra, és fordítva. □

2.3.7. Tétel. (Hipertabló – Helyesség és teljesség)

A hipertabló kalkulus helyes és teljes.

[16] Lásd: Paterson és Wegman lineáris unifikációról szóló cikke [23]. Ezen algoritmus egyazon változó előfordulásait mint az adott változóra mutató referenciákat tárolja, és ily módon a változó behelyettesítése csak egy helyen történik meg.

A Baumgartner-féle hipertablóra vonatkozó bizonyítást maga Baumgartner adta meg [3]. A mi hipertabló kalkulusunk helyessége és teljessége ennek triviális következménye.

Vessük össze Baumgartner hipertablóját a hiperrezolúcióval. Az első dolog, ami szemünkbe ötlik, hogy a tabló adatszerkezet alkalmazásával sikerült a *faktorizációt eliminálnunk* a hiperrezolúcióból; pontosabban fogalmazva, míg a faktorizáció szükséges volt a hiperrezolúció teljességéhez, addig a hipertablóéhoz nem az. Baumgartner [3] cikke a Letz által leírt [20] tablókon végezhető, aciklikus faktorizációt mint opcionális, egyes esetekben hatékonyságot növelő eszközt említi.

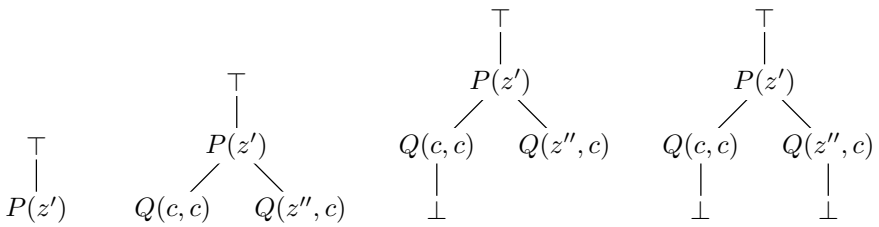
2.3.8. Példa.

A „borbély paradoxon" néven ismert klózhalmaz a következő:

$$\left\{ \begin{array}{c} \neg P(x) \vee Q(y,y) \vee Q(x,y) \ , \\ \neg P(u) \vee \neg Q(v,v) \vee \neg Q(u,v) \ , \\ P(z) \end{array} \right\}$$

Ismert tény, hogy ezen klózhalmaz kielégíthetetlen, ám mégsem lehet hozzá rezolúciós cáfolatot (és ennek következtében hiperrezolúciós cáfolatot sem) megadni anélkül, hogy faktorizálnánk a levezetés során.

Demonstráljuk, hogy a hipertabló teljességéhez nem szükséges a faktorizáció. Ennek érdekében megmutatjuk, hogy a klózhalmazhoz létezik zárt pozitív hipertabló, ennek előállítását alább, balról jobbra haladva szemléltetjük.



A kezdeti tablót a $P(z)$ input klóz új példányával bővítjük, azaz a 2.3.5.(2) jelöléseit használva:

- $C = P(z')$,
- $k = 0$,
- $\sigma = \epsilon$,
- $\pi = \epsilon$.

A következő lépésben a $\neg P(x) \vee Q(y,y) \vee Q(x,y)$ input klóz egy új példányának egyetlen negatív literálját unifikáljuk a tabló $P(z')$ literáljának új példányával. Az így kapott hiperrezolvens azonban nem tiszta klóz, így valamely π tisztító helyettesítés alkalmazásával tisztává kell azt tennünk. Tehát legyen:

- $C = \neg P(x') \vee Q(y', y') \vee Q(x', y')$,
- $k = 1$,
- $L_1^{\oplus} = P(z')$,
- $\sigma = x'/z''$,
- $\pi = y'/c$.

A harmadik lépésben a $\neg P(u) \vee \neg Q(v, v) \vee \neg Q(u, v)$ input klóz egy új példányát állítjuk elő, melynek (mivel negatív klózról van szó) minden literálját kirezolváljuk a bal oldali ág $P(z')$ és $Q(c, c)$ literáljainak felhasználásával:

- $C = \neg P(u') \vee \neg Q(v', v') \vee \neg Q(u', v')$,
- $k = 3$,
- $L_1^{\oplus} = P(z')$, $L_2^{\oplus} = Q(c, c)$, $L_3^{\oplus} = Q(c, c)$,
- $\sigma = \{u'/z''', u'/c, v'/c\}$,
- $\pi = \epsilon$.

Ebben a lépésben a hiperrezolvens $\bot$.

Hasonlóan végezhető el a negyedik lépés, ami után zárt tablót kapunk. □

Baumgartner célja a tisztító helyettesítésekkel azonban nem csak a helyettesítések lokális elvégzésének lehetővé tétele volt, hanem – bár erről ő nem ír cikkében – egy sor olyan (szintén nehezen automatizálható) problémát akart elkerülni, amelyek felbukkannak azon szerzők munkáiban, akik a hipertabló kalkulusból próbálják a tisztító helyettesítéseket száműzni. Maga Baumgartner is egy későbbi cikkében [4] megpróbálkozik a tisztító helyettesítések (azaz a „részben alappéldányok" találgatásának) eliminálásával, ám ez egy igen túlbonyolított és a hagyományos formalizmustól teljesen elrugaszkodott kalkulushoz vezetett. A következő fejezetben egy másik ilyen próbálkozással fogunk megismerkedni.

2.3.1. Rigid hipertabló

Kühn 1997-es cikkében [19] fejleszti tovább Baumgartner hipertablóját, vagy ahogy ő nevezi: a tisztított hipertablót[17]. A kalkulusból száműzi a tisztító helyettesítések használatát, ám rögtön szembesül a következő ténnyel: tisztító helyettesítések használata nélkül a (pozitív) hipertabló csak Horn-klózok esetén teljes[18]. Kühn (más szerzőkhöz is hasonlóan, például [10]) ún. rigid változók használatával próbálja megoldani a problémát. *Rigid változó* alatt a szakirodalom ([13] 114. oldal) olyan változót ért, mely a tablóban globálisan helyettesítendő, szemben az ún. *univerzális változóval*, mely pedig

[17] angol: *purified hyper tableaux*

[18] Mivel egy Horn-klóz legfeljebb egy pozitív literált tartalmaz, a 2.3.5.(2)-beli D alapvetően tiszta.

lokálisan a tabló egy adott formuláján belül. A 2.1.11. definícióban megadott klóztabló minden változót rigidként kezel, a 2.3.3. definícióban leírt Baumgartner-féle hipertabló minden változót univerzálisként, míg a 2.3.5. definíció általunk megadott hipertablója a klóztablóhoz hasonlóan rigidként kezeli a tablóban előforduló változókat.

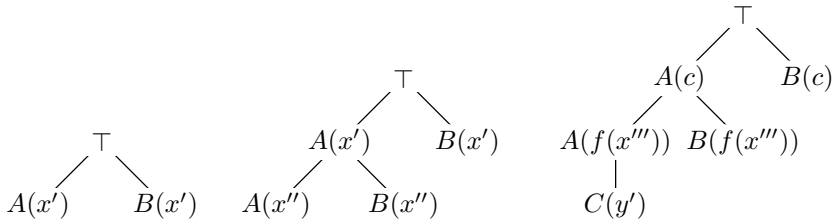
Mint fentebb írtuk, Kühn a hipertablóból kiiktatja a tisztító helyettesítések jelentette nehezen automatizálható tényezőt, ám így újabb automatizálhatósági problémákba fut bele. Ezen problémák a klózokban előforduló „ismétlődésekhez" kapcsolódnak: olyan azonos előjelű literálokhoz, melyek azonos predikátumszimbólumot tartalmaznak. Az ilyen literálok okozta probléma már a rezolúció kapcsán is felbukkant, ahol a faktorizálás elegendő gyógymód volt. Ám az „ismétlődéseket" a faktorizáció nem mindig képes megszüntetni: ha két azonos előjelű, azonos predikátumszimbólumot tartalmazó literál nem unifikálható, akkor a klózon belül nem lehet őket összevonni egy literállá.

2.3.9. Példa.

Tekintsük az

$$\left\{ \begin{array}{c} A(x) \vee B(x)\ , \\ \neg A(c) \vee \neg A(f(x)) \vee C(y) \end{array} \right\}$$

klózhalmazt. Alább, balról jobbra haladva demonstráljuk, hogy egy ágon az $A(x) \vee B(x)$ klózt duplán kell példányosítani, ha a $\neg A(c) \vee \neg A(f(x)) \vee C(y)$ klózt is példányosítani akarjuk.



□

Ezért Kühn a tablóhoz csatolt klózpéldányoknak – ún. ágklózoknak – egyfajta igény szerinti ismétlését írja le. A következő két definíció ennek leírását készíti elő.

2.3.10. Definíció. (Ágklóz[19])

(1) Legyen $\mathcal{T}$ egy literálos tabló[20], és legyen $\mathcal{B} \in \mathcal{T}$.

(2) Legyenek az $L_1, \ldots, L_k$ a $\mathcal{B}$ valamely csúcsának *összes* gyermekei a $\mathcal{T}$-ben, $k \geq 1$. Ekkor az $\{L_1, \ldots, L_k\}$ klóz a $\mathcal{B}$-nek *ágklóza*. □

2.3.11. Definíció. (Multi-kiterjesztés)

Legyen $\mathcal{T}$ egy tabló, és legyen $\mathcal{B} \in \mathcal{T}$. Legyen $\{C_1, \ldots, C_k\}$ klózoknak egy multihalmaza. Definiáljuk a $(\mathcal{T}_0, \mathcal{B}_0), \ldots, (\mathcal{T}_k, \mathcal{B}_k)$ sorozatot a következő módon:

(1) $(\mathcal{T}_0, \mathcal{B}_0) = (\mathcal{T}, \mathcal{B})$

[19] angol: *branch clause*

[20] csak literálokkal címkézett tabló; angol: *clausal tableau* ([19] 4. oldal)

(2) Minden $0 \leq i < k$-ra:

(a) $\mathcal{T}_{i+1} = \mathcal{T}_i +^{\mathcal{B}_i} \widehat{C}_{i+1}$, ahol $\widehat{C}_{i+1}$ a C_{i+1} új példánya, és

(b) $\mathcal{B}_{i+1} \in \mathcal{T}_{i+1} \backslash \mathcal{T}_i$.

A $(\mathcal{T}_k, \mathcal{B}_k)$ a $\mathcal{T}$ $\{C_1, \ldots, C_k\}$-val $\mathcal{B}$-n való *multi-kiterjesztése.* □

A definícióban egy klóz-multihalmaz (azaz esetlegesen ismétlődő klózokat tartalmazó halmaz) elemeinek új példányait csatoljuk egyenként a $\mathcal{T}$ tablónak egy $\mathcal{B}$ ágához, és minden lépés kimeneteként a kibővített tablót (a (2)(a)-ban) és annak egy új ágát (a (2)(b)-ben) adjuk meg.

A rigid hipertabló alább leírásra kerülő definíciója (2.3.13. definíció) nagyon hasonló a (tisztított) hipertabló 2.3.5. definíciójához, ám – a később taglalandó különbségek mellett – egy ág egy input klózzal való hagyományos kiterjesztése helyett az ún. hiperkiterjesztést használja.

2.3.12. Definíció. (Hiperkiterjesztés)

(1) Legyen $\mathcal{T}$ egy literálos tabló, és legyen $\mathcal{B} \in \mathcal{T}$.

(2) Legyen $C = L_1^{\ominus} \vee \ldots L_k^{\ominus} \vee D$ egy klóz, $k \geq 0$, ahol $L_1^{\ominus}, \ldots, L_k^{\ominus}$ a C összes negatív literálja.

(3) Legyen $(\mathcal{T}', \mathcal{B}')$ a $\mathcal{T}$-nek $\mathcal{B}$ valahány ágklózával (akár egyetlen eggyel sem) $\mathcal{B}$-n való multi-kiterjesztése.

(4) Legyen továbbá $L_1^{\oplus}, \ldots, L_k^{\oplus} \in \mathcal{B}'$, úgy hogy
$\sigma = \mathcal{U}\big((\underline{L}_1^{\ominus}, \underline{L}_1^{\oplus}), \ldots, (\underline{L}_k^{\ominus}, \underline{L}_k^{\oplus})\big)$ létezik.

Ekkor

$$(\mathcal{T}' +^{\mathcal{B}'} D)\sigma$$

$\mathcal{T}$-nek C-vel $\mathcal{B}$-n való *pozitív hiperkiterjesztése.* □

A fenti definíció értelmezéséhez vonjunk párhuzamot a hipertabló 2.3.5.(2) kiterjesztési szabályával. Látható, hogy most is egy $\mathcal{T}$ tablónak egy $\mathcal{B}$ ágához csatolunk egy C klózt, miközben C összes negatív literálját kirezolváljuk a $\mathcal{B}$ (pozitív) literáljaival – ezesetben ténylegesen a $\mathcal{B}$ literáljaival (és nem azok új példányaival, mint a hipertabló esetén). Ezen kívül van még egy különbség: a rigid hipertablónál a 2.3.12.(3)-ban végezzük el azt a hipertablónál még ismeretlen plusz műveletet, mely során lehetőségünk van klózok másolatait beszúrni az ágra, mielőtt magát C-t az ágon példányosítanánk. A másolandó klózok a $\mathcal{B}$-nek ágklózai.

A 2.3.12. definíciót használva már megadható a rigid hipertabló kalkulus:

2.3.13. Definíció. (Rigid hipertabló)

Adott egy $\mathcal{C}$ klózhalmaz. $\mathcal{C}$-hez a következő induktív definícióval adjuk meg a *pozitív rigid hipertabló* fogalmát:

(1) Inicializációs szabály:
$\{\{\top\}\}$ a $\mathcal{C}$-nek pozitív rigid hipertablója.

(2) Kiterjesztési szabály:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek pozitív rigid hipertablója, és $\mathcal{B} \in \mathcal{T}$ annak egy ága.

(b) Legyen C a $\mathcal{C}$ valamely klózának egy új példánya[21].

Ekkor $\mathcal{T}$-nek C-vel $\mathcal{B}$-n való pozitív hiperkiterjesztése is $\mathcal{C}$ pozitív rigid hipertablója.

(3) Faktorizálási szabály:
Ha $\mathcal{T}$ a $\mathcal{C}$-nek pozitív rigid hipertablója, akkor $\mathcal{T}$ faktorizáltja is pozitív rigid hipertablója $\mathcal{C}$-nek. □

A rigid hipertabló definíciója formailag szinte semmi újdonságot nem tartalmaz a hipertabló 2.3.5. definíciójához képest; minden lényeges változást a hiperkiterjesztés 2.3.12. definíciója tartalmaz. Vegyük észre azonban, hogy a 2.3.13. definíció a 2.3.5. definícióhoz képest kiegészül a (3) ponttal, melyben a tablót igény szerint faktorizálhatjuk. A tabló faktorizálása alatt a fentebb már említett aciklikus faktorizációt [20] vagy a Kühn által leírt ág-faktorizációt [19] értjük.

2.3.14. Megjegyzés.

A *negatív rigid hipertabló* definícióját a 2.3.12. és a 2.3.13. definíciókból kapjuk a „pozitív" és „negatív" jelzők felcserélésével. □

Kühn cikke a rigid hipertabló helyességének és teljességének nem adja bizonyítását, annak csupán egy-egy vázlatát írja le. Be lehet viszont látni, hogy a rigid hipertabló nem helyes.

2.3.15. Tétel.

A rigid hipertabló nem helyes.

Bizonyítás.

Mint a szerző is írja, a helyesség bizonyításának „érdekes pontja annak megmutatása, hogy az ágklózokkal való multi-kiterjesztés nem roncsolja a tabló kielégíthetőségét az input klózhalmaz modelljeiben" ([19] 8. oldal).

Vagyis azt kéne megmutatni, hogy bármely $\mathcal{C}$ input klózhalmaz és annak bármely $\mathcal{T}$ rigid hipertablója esetén

$$\text{ha } \mathcal{C} \models \forall F(\mathcal{T}), \text{ akkor } \mathcal{C} \models \forall F(\mathcal{T}'),$$

ahol $\mathcal{T}'$-t a $\mathcal{T}$-ből kapjuk úgy, hogy valamely $\mathcal{B} \in \mathcal{T}$ ág ágklózának új példányát a $\mathcal{B}$-hez csatoljuk.

[21] Kühn cikke ebben a tekintetben hibás: egy $\mathcal{C}$-beli klózt magát, nem pedig annak egy új példányát csatolja a tablóhoz ([19] 8. oldal).

Ezen állítást egy ellenpéldán keresztül cáfoljuk meg. Legyen az input klózhalmazunk

$$\mathcal{C} \quad = \quad \{\ A(x) \vee B(x)\ ,\ \neg A(y) \vee C(y)\ \} \quad .$$

A $\mathcal{C}$ következő rigid hipertablóját a kiterjesztési szabály kétszeri alkalmazásával állíthatjuk elő:

$$\mathcal{T} \quad = \quad \begin{array}{ccc} & \top & \\ \swarrow & & \searrow \\ A(y') & & B(y') \\ | & & \\ C(y') & & \end{array} \quad .$$

Könnyen belátható, hogy $\mathcal{C} \models \forall F(\mathcal{T})$:

- Mivel $\forall\big(A(x) \vee B(x)\big) \in \mathcal{C}$, így
$$\mathcal{C} \quad \models \quad \forall\big(A(y') \vee B(y')\big) \quad .$$
- Mivel a $\forall\big(A(x) \vee B(x)\big)$ és $\forall\big(\neg A(y') \vee C(y')\big)$ klózoknak rezolvense a $\forall\big(B(y') \vee C(y')\big)$ klóz, így
$$\mathcal{C} \quad \models \quad \forall\big(B(y') \vee C(y')\big) \quad .$$
- Ezek miatt
$$\begin{array}{c} \mathcal{C} \ \models\ \forall\Big(\ \big(A(y') \vee B(y')\big)\ \wedge\ \big(B(y') \vee C(y')\big)\ \Big) \\ \wr \\ \mathcal{C} \ \models\ \forall\Big(\ \big(A(y') \wedge C(y')\big) \vee B(y')\ \Big) \\ \wr \\ \mathcal{C} \ \models\ \forall F(\mathcal{T}) \quad . \end{array}$$

Most csatoljuk a bal oldali ág $C(y')$ ágklózának új $C(y'')$ példányát az ághoz:

$$\mathcal{T}' \quad = \quad \begin{array}{ccc} & \top & \\ \swarrow & & \searrow \\ A(y') & & B(y') \\ | & & \\ C(y') & & \\ | & & \\ C(y'') & & \end{array}$$

Belátható, hogy előfordulhat a $\mathcal{C}$-nek olyan $\mathcal{M}$ modellje és abban olyan θ változókiértékelés, hogy

$$\begin{array}{lll} \mathcal{M} & \not\models & B(y')\theta \quad \text{és} \\ \mathcal{M} & \models & A(y')\theta \quad \text{és} \\ \mathcal{M} & \models & C(y')\theta \quad , \end{array}$$

de

$$\mathcal{M} \not\models C(y'')\theta \quad .$$

Ekkor

$$\mathcal{M} \not\models F(\mathcal{T}')\theta \quad ,$$

vagyis

$$\mathcal{C} \not\models \forall F(\mathcal{T}') \quad . \qquad \square$$

A teljesség bizonyítása még kényesebb kérdés. Ezt maga Kühn is belátja cikkében, ezért meg sem próbálkozik vele. Tehát mai napig *nyitott kérdés, hogy a rigid hipertabló kalkulus teljes-e.*

A következő összegzést tehetjük tehát a rigid hipertablóval kapcsolatosan: Kühn sikeresen eliminálta a tisztító helyettesítések használatát, ezzel azonban az automatizálási problémákat csak egy másik szintre vitte át. A rigid hipertabló kalkulus két ponton is nehezen automatizálható:

(1) Mikor alkalmazzuk a 2.3.13.(3) szabályt, azaz hogyan tudjuk eldönteni, mikor kell a tabló valamely ágát faktorizálni?

(2) A 2.3.13.(2) szabályban alkalmazott hiperkiterjesztés 2.3.12.(3) pontjában mely ágklózokkal és azok közül is egyenként mennyivel (hiszen a multi-kiterjesztés 2.3.11. definíciójában klózoknak egy multihalmazát kezeljük) terjesszük ki adott ágat?

Ezeken túl, Kühn konstrukciója „túlbiztosított":

(1) A *faktorizálási szabály felesleges.* A faktorizálás nem szükséges eleme a kalkulusnak – csakúgy mint a tisztított hipertabló esetében –, a cikk [19] 8. oldalán leírt példa is hibás a következő értelemben. A példa szövege szerint a $\mathcal{C} = \{P(x) \vee P(y),\ \neg P(z)\}$ klózhalmazhoz „végtelen levezetést konstruálhatunk". A végtelen levezetések elkerülésének problematikája nem a kalkulus teljességéhez tartozik, hanem a redundancia vizsgálatának problémakörébe (lásd: 3.4. fejezet). Egy tablókalkulus teljességének bizonyításával egy zárt tabló *létezéséről* értekezünk. Be is látható, hogy a fenti $\mathcal{C}$ klózhalmazhoz konstruálható zárt rigid hipertabló a faktorizálási szabály alkalmazása nélkül is.

(2) Az *ágklózok példányosítása nem szükséges* eleme a kalkulusnak. Ahogy erről a szerző cikkének [19] 6. oldalán is említést tesz, egy ágklóz új példánya mindig előállítható ugyanazon input klózokból (vagyis azok új példányaiból), mint az eredeti ágklóz. Tehát az ágklóz-másolatok eliminálhatók a kalkulusból. Mint korábban bebizonyítottuk, az ágklózok-másolatok használata roncsolja a kalkulus helyességét. Nyitott kérdés: vajon az ágklóz-másolatok használata nélkül helyes lenne-e a kalkulus?

Kühn rigid hipertabló kalkulusa egy előremutató kísérletnek tekinthető a hipertabló (és közvetve a hiperrezolúció) teljes automatizálására. Ezért a továbbiakban érdemes belőle ötleteket merítenünk, ahogy fogjuk ezt tenni a következő fejezetben, ahol egy saját hipertabló kalkulust, a multi-hipertablót írjuk le, mely sokkal szigorúbban kontrollálja klózok tablóhoz való hozzáfűzését. Kühn kalkulusa azért is tekinthető csak kísérletnek, mert teljessége nem nyert bizonyítást. Ezzel szemben a multi-hipertabló bizonyítottan teljes lesz.

3. fejezet

Multi-hipertabló

A 2.3. fejezetben definiáltuk a (tisztított) hipertablót [3], a hiperrezolúció és a klóztabló technikáinak felhasználásával kapott kalkulust. Szó volt arról, hogy a hipertabló komoly automatizálási problémákkal küzd, melyekre lehetséges megoldásként a 2.3.1. fejezetben ismertettük Kühn rigid hipertablóját [19]. Ám a rigid hipertabló sem teljesen automatikus tételbizonyító módszer, hiszen az általa használt ágklóz másolatok és faktorizáció kezelése nem automatikus. Ezek mellett a rigid hipertabló teljességének bizonyítása sem megoldott.

Ebben a fejezetben ismertetjük az általunk kidolgozott multi-hipertabló kalkulust, mely a [18]-ben került publikálásra. A multi-hipertabló a rigid hipertabló továbbfejlesztett változata, mely

(1) kiküszöböli a hipertabló által használt tisztító helyettesítéseket,

(2) eliminálja a rigid hipertablóban használt ágklóz másolatok és faktorizáció problémáját,

(3) a hipertabló és a rigid hipertabló effektivitási problémáira is megpróbál választ találni,

(4) bizonyítottan helyes és teljes kalkulus (a teljes elsőrendű logikában).

Az (1) rigid változók használatával valósul meg, a rigid hipertablóhoz hasonlóan.

A (2)-ben említett ágklóz másolatok már Kühn cikkében is mint opcionális elemek szerepeltek. A faktorizáció mind a hipertablóban, mind a rigid hipertablóban opcionális, bár ez utóbbiban külön levezetési szabályként szerepel (igaz, hogy feleslegesen).

A (3)-ban a következő problémára gondolunk: a hipertabló a 2.3.5.(2)(b)-ben, a rigid hipertabló a 2.3.12.(2)-ben C-t mint tetszőleges (azaz akár pozitív) input klózt határozza meg, aminek az az eredménye, hogy a pozitív input klózok kontrollálatlanul, tömegesen kerülhetnek be a tablóba. Gondoljunk itt az adatbázisokra, ahol pontosan ilyen input klózból – mint egy darab pozitív literálként felírható táblabejegyzésből –

van rengeteg. Ebben a tekintetben a hipertabló és a rigid hipertabló inkább a klóztablóhoz nyúl vissza, ahol a 2.1.11.(2)(b)-ben tetszőleges input klózok csatolhatók a tablóhoz; ez az ára annak, hogy a 2.3.5.(2)(c)-ben és a 2.3.12.(4)-ben az $L_i^{\oplus}$-k csak az ágról származnak, nem az input klózokból. A multi-hipertabló inkább a hiperrezolúció oldaláról közelít a problémához, ahol a 2.2.15. definícióban kiköttetett, hogy $C^{\ominus}$ csak nem pozitív klóz lehet, aminek az lesz a vonzata, hogy az $L_i^{\oplus}$-ket nem elég csak az ágon keresnünk, hanem ki kell lépnünk az input klózhalmazba is. Ez viszont a hipertablóhoz és a rigid hipertablóhoz képest azt az egyértelműen előnyös vonást fogja a multi-hipertablónak kölcsönözni, hogy csak azon pozitív klózok példányosulnak a tablóban, melyeket ténylegesen felhasználunk egy-egy hiperrezolvens előállításához.

Pontosan ezen utóbbi megoldás, azaz hogy egyetlen dedukciós lépés során akár több input klózt is példányosíthatunk – hasonlóan a rigid hipertabló multi-kiterjesztéséhez –, ihlette a kalkulus elnevezését, annak „multi" előtagját.

3.1. A multi-hipertabló kalkulus

A továbbiakban megadjuk a multi-hipertabló kalkulus pontos leírását. Először két definíciót adunk meg, melyek leírják, mit értünk egy $\mathcal{T}$ tabló valamely $\mathcal{B}$ ágának kiterjesztése alatt.

3.1.1. Definíció. (Kiterjesztés)

Kiterjesztésnek egy $[E, \sigma]$ párt nevezünk, ahol E egy klóz és σ egy helyettesítés. □

3.1.2. Definíció. (Kiterjesztés alkalmazása)

Az $[E, \sigma]$ *kiterjesztést* oly módon *alkalmazzuk* a $\mathcal{T}$ tabló egy $\mathcal{B} \in \mathcal{T}$ ágán, hogy előállítjuk a következő tablót:

$$(\mathcal{T} +^{\mathcal{B}} E)\sigma \quad .$$ □

Látható, hogy a 3.1.2. definícióban teljesen visszanyúlunk a klóztablóhoz, és annak 2.1.11.(2) és 2.1.11.(3) szabályát egyszerre végezzük el. Ám míg a klóztabló esetén E csak input klóz (új példánya) lehetett és σ csak két (azonos ágon szereplő) literál MGU-ja, addig a multi-hipertabló esetén az E-re és a σ-ra sokkal szigorúbb megszorítást teszünk. Egy kiterjesztést egyszerre három dologhoz rendelhetünk hozzá: egy klózhoz, egy klóz-multihalmazhoz és egy literál-multihalmazhoz.

3.1.3. Definíció. (Kiterjesztés meghatározása)

(1) Legyen $C^{\ominus}$ egy klóz és legyen $\{L_1^{\ominus}, \ldots, L_k^{\ominus}\}$ a $C^{\ominus}$ összes negatív literáljának a halmaza, ahol $k \geq 1$.

(2) Legyen $\{C_1^{\oplus}, \ldots, C_k^{\oplus}\}$ pozitív klózoknak egy multihalmaza.

(3) Legyen $\{L_1^{\oplus}, \ldots, L_k^{\oplus} \mid L_i^{\oplus} \in C_i^{\oplus}\}$ (pozitív) literáloknak olyan multihalmaza, hogy $\sigma = \mathcal{U}\big((\underline{L}_1^{\ominus}, \underline{L}_1^{\oplus}), \ldots, (\underline{L}_k^{\ominus}, \underline{L}_k^{\oplus})\big)$ létezik.

Jelöljük E-vel a következő klózt:

$$C^{\ominus}\backslash\{L_1^{\ominus},\ldots,L_k^{\ominus}\} \cup \bigcup_{i=1}^{k} C_i^{\oplus}\backslash\{L_i^{\oplus}\} \quad . \tag{4}$$

Ekkor a $C^{\ominus}$-hoz, $\{C_1^{\oplus},\ldots,C_k^{\oplus}\}$-hoz és $\{L_1^{\oplus},\ldots,L_k^{\oplus}\}$-hoz tartozó,

$$e\Big(C^{\ominus},\{C_1^{\oplus},\ldots,C_k^{\oplus}\},\{L_1^{\oplus},\ldots,L_k^{\oplus}\}\Big) \quad \text{-val}$$

hivatkozott kiterjesztés legyen $[E,\sigma]$. □

A fenti definícióban az (1)-ben leírt $C^{\ominus}$ jelöli azt a nem pozitív klózt (hiszen $k \geq 1$), melynek az összes negatív literálját kirezolváljuk. Ezen literálokat a (2)-ben meghatározott $C_i^{\oplus}$ pozitív klózok egy-egy (3)-ban leírt $L_i^{\oplus}$ (pozitív) literáljával fogjuk unifikálni. A (4)-ben meghatározott E tulajdonképpen a $C^{\ominus}$ és az összes $C_i^{\oplus}$ valamennyi nem unifikált literáljából összeállított klóz, σ pedig a fent kiszámolt MGU. Lehet látni – a fenti definíciót összehasonlítva a 2.2.15. definícióval –, hogy tulajdonképpen az $\langle E\rangle\sigma$ hiperrezolvenst számítjuk ki.

A következő definíció leírja, hogy egy ág és egy input klózhalmaz esetén mely klózokhoz, klóz-multihalmazokhoz és literál-multihalmazokhoz állíthatunk elő kiterjesztéseket.

3.1.4. Definíció. (Ág és klózhalmaz kiterjesztései)

Egy $\mathcal{B}$ *ághoz* és egy $\mathcal{C}$ *klózhalmazhoz* tartozó *kiterjesztések* a következő halmaz elemei:

$$\mathcal{E}(\mathcal{B},\mathcal{C}) = \left\{ e\Big(\widehat{C}^{\ominus},\{\widehat{C}_1^{\oplus},\ldots,\widehat{C}_k^{\oplus}\},\{L_1^{\oplus},\ldots,L_k^{\oplus}\}\Big) \;\middle|\; \begin{array}{ll} C^{\ominus}\in\mathcal{C}, & (1)\\ C_i^{\oplus}\in\mathcal{B}\cup\mathcal{C}, & (2)\\ L_i^{\oplus}\in\widehat{C}_i^{\oplus} & (3)\end{array} \right\} \quad .$$

A fentiekben bármely C klóz esetén

$$\widehat{C} \;=\; \begin{cases} C \text{ új példánya} & \text{, ha } C\in\mathcal{C};\\ C & \text{, egyébként.}\end{cases}$$ □

A fenti definícióban észrevehetjük a következőket:

- Az (1)-ben kiválasztott $C^{\ominus}$ klóz – ugyanúgy mint a C a klóztabló (2.1.11.(2)), a hipertabló (2.3.5.(2)(b)) vagy a rigid hipertabló (2.3.12.(2)) esetén – az input klózhalmazból való, de ezesetben mindenképpen csak *nem pozitív* klóz lehet.
- A (2)-ben kiválasztott $C_1^{\oplus},\ldots,C_k^{\oplus}$ klózok nem csak az ágról származhatnak, mint a hipertabló (2.3.5.(2)(c)) és a rigid hipertabló (2.3.12.(4)) esetén, hanem $\mathcal{C}$-ből is, azaz (pozitív) *input klózok* is lehetnek. Ha $C_i^{\oplus}$ az ágról származik, akkor természetesen csak egyetlen pozitív literált tartalmaz.

- A 3.1.3.(2,3)-ban klóz-, illetve literál-multihalmazokat használunk. Tehát a fenti definícióban a (2)-ben kiválasztott $C_1^{\oplus}, \ldots, C_k^{\oplus}$ klózok között lehetnek azonosak is, ahogy a (3)-ban kiválasztott $L_1^{\oplus}, \ldots, L_k^{\oplus}$ literálok között is.
- Amelyik klózt a $\mathcal{C}$-ből választjuk, annak *új példányát* állítjuk elő, amelyiket az ágról, azt eredeti formájában használjuk fel.

Az előzőekben leírt definíciókat használjuk fel most a multi-hipertabló kalkulus definiálásához:

3.1.5. Definíció. (Multi-hipertabló)

Adott egy $\mathcal{C}$ klózhalmaz. $\mathcal{C}$-hez a következő induktív definícióval adjuk meg a *pozitív multi-hipertabló* fogalmát:

(1) Inicializációs szabály:
$\{\{\top\}\}$ a $\mathcal{C}$-nek pozitív multi-hipertablója.

(2) Kiterjesztési szabály:

(a) Legyen $\mathcal{T}$ a $\mathcal{C}$-nek pozitív multi-hipertablója, és legyen $\mathcal{B} \in \mathcal{T}$.

(b) Legyen $e \in \mathcal{E}(\mathcal{B}, \mathcal{C})$ kiterjesztés.

Ekkor az e-nek a $\mathcal{T}$ tabló $\mathcal{B}$ ágán való alkalmazásával kapott tabló is $\mathcal{C}$-nek pozitív multi-hipertablója. □

3.1.6. Megjegyzés.

A *negatív multi-hipertabló* definíciója megkapható a 3.1.3., a 3.1.4. és a 3.1.5. definíciókból a „pozitív" és „negatív" jelzők felcserélésével. □

Egy kiterjesztés elvégzésekor a triviálisan zárttá váló ágakat nem vesszük fel a tablóba – ahogy azt már a hipertabló és a rigid hipertabló esetében sem tettük.

Egy $\mathcal{B}$ ág akkor válik zárttá, ha létezik olyan $L_1 \in \mathcal{C}$ negatív és olyan $L_2 \in \mathcal{B}$ (pozitív) literál, hogy $\sigma = \mathcal{U}(\widehat{\underline{L}}_1, \underline{L}_2)$ létezik; azaz $\mathcal{B}$ az

$$e(\widehat{L}_1,\ \{L_2\},\ \{L_2\}) \quad = \quad [\bot, \sigma]$$

kiterjesztés elvégzése után válik zárttá. Ebből következik, hogy akkor sikerül egy ágat lezárnunk, ha üres klózt ($\bot$) tartalmazó kiterjesztést végzünk el rajta. Éppen ezért innentől kezdve a levezetés során egy-egy ágon elég azt figyelni, hogy azághoz vajon sikerült-e előállítanunk $\bot$-t tartalmazó kiterjesztést. Ez az eljárás nagymértékben hasonlít a rezolúció során alkalmazott eljáráshoz, ahol az *üres klóz előállítása a cél.*

A 3.1.3.(2,3)-ban – mint már említettük – klóz- és literál-multihalmazokat használunk[1]. Ezen tényező hozzájárul a faktorizáció eliminálhatóságához. Például legyen

[1] Hasonlóképpen, mint Baumgartner továbbfejlesztett hipertablójának `Ext` és `Link` szabályaiban ([4] 7. oldal).

$P(x) \in \mathcal{B}$ az ágon egy literál, és legyen $\neg P(f(c)) \vee \neg P(f(y))$ egy input klóz. A multi-hipertabló egyetlen kiterjesztéssel oldja meg $\mathcal{B}$ lezárását: az

$$e\Big(\neg P(f(c)) \vee \neg P(f(y')),\ \big\{P(x), P(x)\big\},\ \big\{P(x), P(x)\big\}\Big)$$

kiterjesztéssel, ami nem más, mint $\big[\bot, \{x/f(c), y'/c\}\big]$. Vegyük észre, hogy az y'/c helyettesítés pontosan az, melyet $\neg P(f(c)) \vee \neg P(f(y'))$ faktorizálása során végeznénk el.

3.1.7. MEGJEGYZÉS.

A multi-hipertabló abbeli „engedékenységét", hogy a $C_i^{\oplus}$ klózok nem csak az ágról származhatnak, hanem input klózok is lehetnek, tekinthetjük valamiféle „*előretekintésnek*", azaz a hipertabló és a rigid hipertabló „vak", szisztematikus következtetési módszerével szemben a multi-hipertabló „megpróbálja kitalálni", hogy mely input klózokat lenne érdemes a tablóban példányosítani. Magyarán szólva, a multi-hipertablóban benne rejlik a *heurisztikus következtetésnek* a lehetősége, amivel a 3.5. fejezetben részletesen foglalkozni is fogunk. □

3.1.8. TÉTEL. (MULTI-HIPERTABLÓ – HELYESSÉG)

A multi-hipertabló kalkulus helyes.

BIZONYÍTÁS.

Meg kell mutatni, hogy ha egy $\mathcal{C}$ klózhalmazhoz létezik zárt multi-hipertabló, akkor a $\mathcal{C}$ kielégíthetetlen. Ehhez induktíve bebizonyítjuk, hogy $\mathcal{C}$-nek bármely $\mathcal{T}$ multi-hipertablójára $\mathcal{C} \models F(\mathcal{T})$.

(1) A kezdeti $\big\{\{\top\}\big\}$ tablóra ez triviális.

(2) Be kell látni, hogy ha egy $\mathcal{T}$ tablóra $\mathcal{C} \models F(\mathcal{T})$, akkor $\mathcal{T}$-ből a kiterjesztési szabály alkalmazásával kapott $\mathcal{T}'$-re is $\mathcal{C} \models F(\mathcal{T}')$. A szabályban elvégzett kiterjesztést jelöljük $[E, \sigma]$-val. Mivel $\langle E\rangle\sigma$ a $\mathcal{T}$ egy $\mathcal{B}$ ága valahány literáljának és $\mathcal{C}$ valahány klózának hiperrezolvense, és a 2.2.18. lemma alapján tudjuk, hogy $\langle E\rangle\sigma$ ezeknek logikai következménye, így

$$\mathcal{C} \qumodels\quad F\big(\mathcal{T} +^{\mathcal{B}} \langle E\rangle\sigma\big) \quad .$$

Továbbá tetszőleges helyettesítésre, így σ-ra is

$$\mathcal{C} \qumodels\quad F\Big(\big(\mathcal{T} +^{\mathcal{B}} \langle E\rangle\sigma\big)\sigma\Big) \quad .$$

Mivel σ MGU, és ezért $\mathcal{FV}\big(\mathcal{D}om(\sigma)\big) \cap \mathcal{FV}\big(\mathcal{R}ange(\sigma)\big) = \emptyset$, így a $\big(\mathcal{T} +^{\mathcal{B}} \langle E\rangle\sigma\big)\sigma$ tabló azonos a $\big(\mathcal{T} +^{\mathcal{B}} \langle E\rangle\big)\sigma$ tablóval, mely viszont pont $\mathcal{T}'$. Azaz:

$$\mathcal{C} \quad\models\quad F(\mathcal{T}') \quad .$$

Mivel $\mathcal{C}$-nek van zárt multi-hipertablója, melyet jelöljünk $\mathcal{T}_\perp$-tal, így a következőket tudjuk:

$$\mathcal{C} \models F(\mathcal{T}_\perp) \quad \text{és} \quad F(\mathcal{T}_\perp) \sim \perp \quad .$$

Azaz $\mathcal{C} \models \perp$, vagyis $\mathcal{C}$ kielégíthetetlen. □

A helyesség bizonyításával szemben a teljesség bizonyítása már messze nem olyan egyszerű. Kühn rigid hipertablója inkább a klóztabló irányából közelít a hiperrezolúció felé, olyan értelemben, hogy magukat az input klózokat (illetve azok új példányait) csatolja a tablóhoz – akár a klóztabló. Ezen input klózok csatolását próbálja a kalkulus oly módon irányítani, hogy az egymással rezolválható klózok kerüljenek a tablóban egymás alá a megfelelő ágakra, ily módon lezárva egy-egy ágat. Ezzel szemben a multi-hipertabló a *hiperrezolúció irányából* közelít a klóztabló felé. Azaz – akár a hiperrezolúció – hiperrezolvenst számol, és azt csatolja a tablóhoz. Ez a megközelítési mód áttekinthetőbb kalkulust eredményez, hiszen a tabló bővítése egyidejűleg *mindig csak egy klózzal* történik, szemben a rigid hipertablóval, ahol az ágklóz másolatokkal egyszerre több klózt kell a tablóhoz csatolnunk. De a legnagyobb előny mégis a teljesség bizonyításában nyilvánul meg: a multi-hipertabló teljességének bizonyításához fel tudjuk használni a hiperrezolúció bizonyított teljességét.

3.2. A multi-hipertabló teljessége

A multi-hipertabló kalkulus teljessége a következőt jelenti: tetszőleges kielégíthetetlen $\mathcal{C}$ klózhalmaz esetén van a $\mathcal{C}$-nek zárt multi-hipertablója. Ennek bizonyításához felhasználjuk a hiperrezolúciónak a 2.2.19. tételben kimondott teljességét, azaz a $\mathcal{C}$-hez bizonyítottan létező $C_1, \ldots, C_n$ pozitív hiperrezolúciós cáfolatot. Az általánosság megszorítása nélkül $C_1, \ldots, C_n$-ről a továbbiakban feltesszük a következőt:

(H) Minden $1 \leq i < n$-re: C_i-t *pontosan egy* hiperrezolúciós lépésben használtuk fel $\perp$ előállítására.

Nyilvánvaló, hogy ha $\mathcal{C}$-hez létezik hiperrezolúciós cáfolat, akkor létezik a megadott feltételnek eleget tevő hiperrezolúciós cáfolat is.

A teljesség bizonyítása két lépcsőben fog történni:

(1) A $C_1, \ldots, C_n$-ből generálunk egy ún. *hiperrezolúciós gráfot*, melyet a következő fejezetben fogunk definiálni. Ezen gráfhoz megadjuk a gráf csúcsain értelmezett h_1 és h_2 függvényeket, melyek feladata informálisan a következő lesz:

   - A h_1 a gráf egyes csúcscsoportjaihoz a hiperrezolúciós levezetés egy-egy klózát fogja rendelni. Itt nem csupán a $C_1, \ldots, C_n$ klózokra gondolunk, hanem a hiperrezolúciós levezetés során felhasznált összes klózra (faktorokra és hiperrezolvensekre is).
   - A h_2 a gráf egyes csúcscsoportjaihoz input (azaz $\mathcal{C}$-beli) klózokat fog rendelni.

(2) A $\mathcal{C}$-hez a megkonstruált hiperrezolúciós gráfból generálunk *multi-hipertablót*. Ennek során megadunk egy a tabló csúcsain értelmezett t függvényt, mely a tabló csúcsaihoz hozzárendeli a hiperrezolúciós gráf egy-egy csúcsát.

A továbbiakban mindenképpen szükséges lesz, hogy a felhasznált klózoknak egy-egy literáljára egyértelműen tudjunk hivatkozni (hiszen ugyanazon literál akár több klózban – és egy klózon belül akár többször – is előfordulhat). Ennek érdekében minden klózon belül adottnak tételezzük fel a klóz literáljainak egy sorrendjét; azaz egy klózt mint literáljainak rendezett multihalmazát kezeljük. Ilyen értelemben beszélni fogunk egy C klóz l. literáljáról ($1 \leq l \leq |C|$), melyet $C[l]$-lel fogunk jelölni.

3.2.1. Hiperrezolúciós gráf

A hiperrezolúciós gráf fogalmának bevezetéséhez szükség lesz a következő fogalmakra:

3.2.1. Definíció. (Csoportosított gráf)

Csoportosított gráf alatt olyan véges gráfot értünk, melynek minden csúcsát diszjunkt rendezett halmazokba soroljuk, s mely halmazokat *csúcscsoportoknak* nevezzük. Ha C csúcscsoport, akkor annak i. csúcsára $C[i]$-vel fogunk hivatkozni ($1 \leq i \leq |C|$). □

3.2.2. Definíció. (Levezető csúcscsoport)

Egy csoportosított gráf valamely C csúcscsoportja akkor *levezető csúcscsoport*, ha teljesülnek rá a következő feltételek:

- Semelyik $N \in C$ csúcs sem levélcsúcs.
- Minden $N \in C$ csúcsra és minden (N, X) élre az X levélcsúcs. □

3.2.3. Definíció. ($\mathcal{N}$-él)

Egy csoportosított gráf valamely $\mathcal{N}$ csúcscsoport-halmaza esetén *$\mathcal{N}$-élnek* azon élt nevezzük, mely $\mathcal{N}$ két elemének két csúcsát köti össze. □

A $\mathcal{C}$ klózhalmaz $C_1, \ldots, C_n$ hiperrezolúciós cáfolatához generált *hiperrezolúciós gráf* egy csoportosított gráf lesz, melyhez megadjuk az előző fejezetben említett

$$h_1, h_2 \quad : \quad \mathcal{S}_\mathcal{H} \mapsto \mathcal{C}^+ \times \mathbb{N}^+$$

függvényeket, ahol

- $\mathcal{S}_\mathcal{H}$ a gráf csúcsainak halmaza,
- $\mathcal{C}^+$ a hiperrezolúciós cáfolat előállítása során felhasznált összes klózok halmaza[2], és
- $\mathbb{N}^+$ pedig a pozitív egész számok halmaza.

[2] Tehát a felhasznált új példányok, faktorok és hiperrezolvensek.

Ha a hiperrezolúciós gráf egy N csúcsára

$$h_i(N) \quad = \quad (C, l) \ , \text{ ahol } i \in \{1, 2\},$$

akkor számunkra az azt fogja jelenti, hogy a h_i függvényen keresztül N-hez hozzárendeljük a C klóz l. literálját (azaz $C[l]$-t).

A hiperrezolúciós gráf olyan speciális csoportosított gráf lesz, melynek minden $C = \{N_1, \ldots, N_k\}$ csúcscsoportjára és minden $i \in \{1, 2\}$-re teljesül a következő feltétel:

van olyan C' klóz, hogy minden $1 \leq j \leq k$ esetén $h_i(N_j) = (C', l_j)$.

Azaz a h_i függvény egy csúcscsoport összes csúcsához ugyanazt a C' klózt rendeli. Ily módon a h_i-n keresztül a C csúcscsoporthoz egyértelműen hozzárendelhető a C' klóz, melynek kifejezésére a kényelmes

$$h_i(C) \quad = \quad C'$$

jelölést fogjuk használni. Vegyük észre, hogy itt csak egy jelölést fixáltunk, természetesen a h_i függvény továbbra sem értelmezett csúcscsoportokon.

3.2.4. Definíció. (Hiperrezolúciós gráf)

A következőképpen *konstruáljuk meg* a $\mathcal{C}$ input klózhalmaz egy $C_1, \ldots, C_n$ pozitív *hiperrezolúciós cáfolatához tartozó hiperrezolúciós gráfot.*

Kezdetben a gráf legyen üres, majd minden $1 \leq i \leq n$-re végezzük el a következő lépéseket, feltéve hogy minden $1 \leq j < i$-re a lépéseket már elvégeztük:

(1) Ha C_i az $I_i \in \mathcal{C}$ *input klóz* új példánya, akkor minden $1 \leq l \leq |C_i|$ esetén felveszünk a gráfba egy új N_l csúcsot úgy, hogy

$$\begin{aligned} h_1(N_l) \quad &:= \quad (C_i, l) \quad , \\ h_2(N_l) \quad &:= \quad (I_i, l) \quad . \end{aligned}$$

Az összes N_l-t egy új csúcscsoportba foglaljuk.

(2) Ha C_i a $C^{\ominus}$ és $\{C_1^{\oplus}, \ldots, C_k^{\oplus}\}$ pozitív *hiperrezolvense*, melynek megkonstruálása során a $C^{\ominus}[l_j^{\ominus}]$ literált rezolváltuk ki a $C_j^{\oplus}[l_j^{\oplus}]$ literállal $(1 \leq j \leq k)$, akkor:

(a) Ha bármely $C \in \{C^{\ominus}, C_1^{\oplus}, \ldots, C_k^{\oplus}\}$-t valamely $C' \in \{C_1, \ldots, C_{i-1}\}$-ből képeztük *faktorizációval*, akkor minden N csúcsra végezzük el a következőt: ha $h_1(N) = (C', l')$ és a faktorizáció során a $C[l]$-t a $C'[l']$-ből állítottuk elő[3], akkor

$$h_1(N) \quad := \quad (C, l) \quad .$$

[3] ahol $1 \leq l \leq |C|$ és $1 \leq l' \leq |C'|$

(b) Minden $1 \leq j \leq k$ esetén:
minden olyan $N^{\ominus}$ és $N^{\oplus}$ csúcsra, hogy $h_1(N^{\ominus}) = (C^{\ominus}, l_j^{\ominus})$ és $h_1(N^{\oplus}) = (C_j^{\oplus}, l_j^{\oplus})$, vegyük fel a gráfba az

$$(N^{\ominus}, N^{\oplus})$$

élt.

(c) Minden N csúcsra végezzük el a következőt:
ha $h_1(N) = (C', l')$ és a rezolválás során a $C_i[l]$-t a $C'[l']$-ből állítottuk elő[4], akkor

$$h_1(N) \quad := \quad (C_i, l) \quad . \qquad \square$$

3.2.5. Példa.

Legyen adott a következő input klózhalmaz:

$$\mathcal{C} \quad = \quad \left\{ \begin{array}{c} B(x) \vee \neg E(x) \ , \\ E(f(y)) \vee B(f(y)) \ , \\ \neg A(f(z)) \vee \neg B(f(c)) \ , \\ A(u) \vee D(u) \ , \\ \neg D(v) \vee \neg D(f(w)) \vee B(g(v)) \ , \\ \neg B(g(f(d))) \end{array} \right\} \quad ,$$

és legyen adott ennek egy $C_1, \ldots, C_{10}$ hiperrezolúciós cáfolata, ahol

$$\begin{array}{rcl} C_1 & = & B(x') \vee \neg E(x') \\ C_2 & = & E(f(y')) \vee B(f(y')) \\ C_3 & = & B(f(y')) \vee B(f(y')) \\ C_4 & = & \neg A(f(z')) \vee \neg B(f(c)) \\ C_5 & = & A(u') \vee D(u') \\ C_6 & = & D(f(z')) \\ C_7 & = & \neg D(v') \vee \neg D(f(w')) \vee B(g(v')) \\ C_8 & = & B(g(f(z'))) \\ C_9 & = & \neg B(g(f(d))) \\ C_{10} & = & \bot \end{array}$$

A C_1, C_2, C_4, C_5, C_7 és C_9 input klózok új példányai, a többiek:

- A C_3 a C_1 és $\{C_2\}$ hiperrezolvense.
- A C_6 a C_4 és $\{C_5, C_3'\}$ hiperrezolvense, ahol C_3' a C_3 faktora.

[4] ahol $1 \leq l \leq |C_i|$ és $1 \leq l' \leq |C'|$

- A C_8 a C_7' és $\{C_6\}$ hiperrezolvense, ahol C_7' a C_7 faktora.
- A C_{10} a C_9 és $\{C_8\}$ hiperrezolvense.

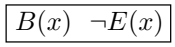
A fentiekből kikövetkeztethető, hogy c és d konstansszimbólumok (0-fokú függvényszimbólumok).

A következő lépésekben konstruáljuk meg a hiperrezolúciós gráfot:

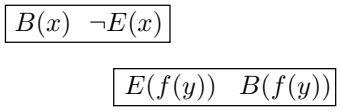
(1) Mivel C_1 input klóz új példánya, így a 3.2.4.(1) alapján felveszünk egy új csúcscsoportot a gráfba, melynek elemeihez hozzárendeljük a h_1-en keresztül a $B(x')$ és a $\neg E(x')$, h_2-n keresztül pedig a $B(x)$ és a $\neg E(x)$ literálokat.

A gráf *egyszerű szemléltetésére* a következő megoldást választjuk: a csúcscsoportokat bekeretezzük, egy csúcsot pedig a h_2 által a csúcshoz rendelt literál képvisel[5].

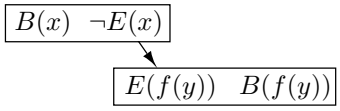
Tehát az első lépés után a gráf a következőképpen néz ki:



(2) Hasonlóan járunk el C_2-vel kapcsolatban is. Tehát a gráf az első két lépés után:



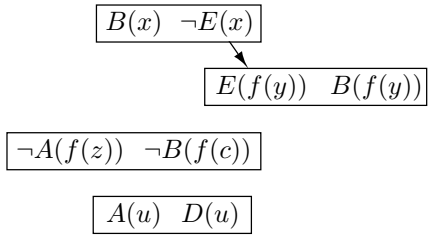
(3) A C_3 hiperrezolvens, melynek előállításakor a C_1 $\neg E(x')$ literálját és a C_2 $E(f(y'))$ literálját rezolváltuk ki. Vagyis a 3.2.4.(2)(b) alapján élt húzunk azon csúcsok között, melyekhez h_1-en keresztül ezen literálokat rendeltük:



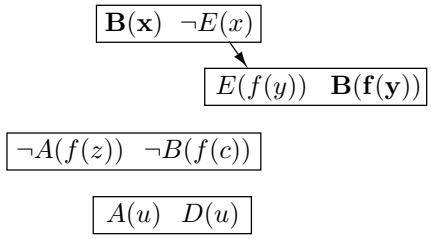
Az élek felvétele után pedig a 3.2.4.(2)(c) alapján módosítjuk a hiperrezolvensbe kerülő literáloknak megfeleltetett csúcsokra (a fent ábrázolt gráfban a $B(x)$-szel és a $B(f(y))$-nal jelölt csúcsokra) a h_1 szerinti hozzárendelést: ezen csúcsokhoz a h_1 mostantól a C_3 megfelelő literáljait fogja rendelni.

[5] Persze mindemellett soha nem szabad elfelejtenünk, hogy a gráf csúcsainak és a klózok literáljainak egyértelmű megfeleltetése érdekében a h_1 és a h_2 minden literálra az őt tartalmazó klózzal és a literál klózon belüli sorszámával hivatkozik. Például a $B(f(y'))$ literál a hiperrezolúciós cáfolat több klózában is szerepel, sőt a C_3-ban duplán is, tehát ezen literál esetén csak a fent leírt módon szüntethetők meg a kétértelműségek.

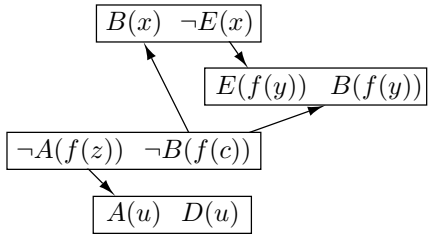
(4) A C_4 és a C_5 input klózok új példányai, ezért a gráf a következőképpen bővül:



(5) A C_6 hiperrezolvens, melynek előállításakor a C_3 faktorát használtuk. Az ábrán vastagon szedett literálok azok, melyekhez a fenti (3) pontban hozzárendeltük C_3 literáljait (a h_1-en keresztül):



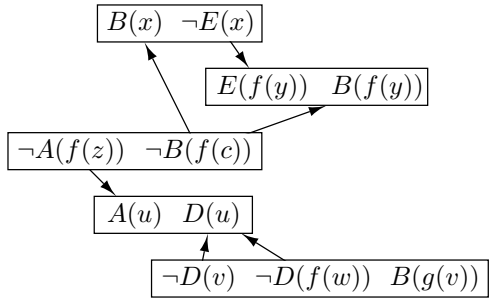
Most a 3.2.4.(2)(a) alapján a h_1-et módosítjuk oly módon, hogy a vastagon szedett csúcsokhoz a h_1 a faktornak ugyanazt a literálját rendelje (a faktornak csak egy literálja van). Ezek után a 3.2.4.(2)(b) elvégzésével a következő gráfot kapjuk:



Majd természetesen a 3.2.4.(2)(c) alapján a $D(u)$-val jelölt csúcsnak a h_1 szerinti hozzárendeltjét a C_6 egyetlen literáljára módosítjuk.

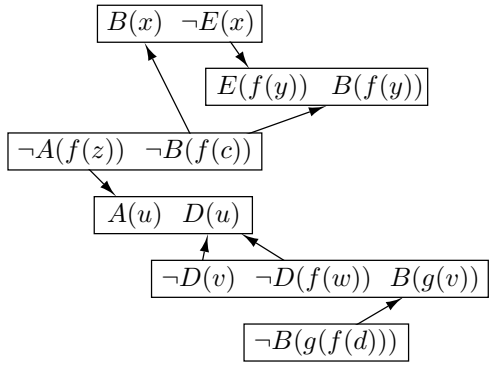
(6) A C_7 input klóz új példánya, ennek a gráfba (új csúcscsoportként) való felvétele után következzenek a C_8 hiperrezolvensre vonatkozó lépések. Mivel a C_7-nek faktorát használtuk, a h_1-nek a 3.2.4.(2)(a) által meghatározott módosítását

elvégezzük, azaz a $\neg D(v)$-vel és a $\neg D(f(w))$-vel jelölt csúcsok mostantól ezen faktor első literáljára hivatkoznak. Aztán a 3.2.4.(2)(b) alapján behúzzuk az új éleket a gráfba:



Majd a legalsó csúcscsoport $B(g(v))$-vel jelölt csúcsához h_1 szerint hozzárendeljük a C_8 hiperrezolvens egyetlen literálját, a 3.2.4.(2)(c) alapján.

(7) Utolsó két lépésként felveszünk egy C_9-nek megfelelő csúcscsoportot, majd a C_{10} hiperrezolvensre csupán egy új élt kell a gráfhoz hozzávennünk. Tehát a $\mathcal{C}$ megadott hiperrezolúciós cáfolatához tartozó hiperrezolúciós gráf a következő:



□

A hiperrezolúciós gráfhoz tartozó h_1 függvényt csupán a gráf konstruálása során használjuk. Ezzel szemben a h_2 kizárólag a multi-hipertabló hiperrezolúciós gráfból való felépítéséhez lesz felhasználva. Megjegyezzük, hogy a hiperrezolúciós gráfnak az (előző példában alkalmazott) ábrázolásából a h_2 közvetlenül kiolvasható. A multi-hipertabló megkonstruálásakor szükségünk lesz a következő fogalomra:

3.2.6. Definíció. (Csúcscsoport sokszorozása)

Adott egy hiperrezolúciós gráf valamely C_1 csúcscsoportja, és a C_1-be vezető/C_1-ből kivezető éleknek egy $\mathcal{E} = \{e_1, \ldots, e_k\}$ nem üres halmaza. A következő lépések elvégzésével *sokszorozzuk C_1-et $\mathcal{E}$ mentén a gráfban*:

(1) Hozzuk létre a C_1 csúcscsoportnak $k-1$ db. másolatát a gráfban, ezeket jelöljük $C_2, \ldots, C_k$-val. Azaz minden $1 < i \leq k$ esetén C_i $|C_1|$ db. új csúcsot tartalmazzon, és minden $1 \leq j \leq |C_i|$ esetén

$$h_2(C_i[j]) \quad := \quad h_2(C_1[j]) \quad .$$

(2) A gráf minden $e = (X, C_1[j])$, illetve $e = (C_1[j], X)$ éle[6] esetén:

(a) Töröljük e-t a gráfból.

(b) Ha $e = e_i$ valamely $1 \leq i \leq k$-ra (azaz $e \in \mathcal{E}$), akkor vegyük fel az $(X, C_i[j])$, illetve $(C_i[j], X)$ élt a gráfba[7].

(c) Ha $e \notin \mathcal{E}$, akkor minden $1 \leq i \leq k$-ra: vegyük fel az $(X, C_i[j])$, illetve $(C_i[j], X)$ élt a gráfba[7]. □

3.2.2. A teljesség bizonyítása

3.2.7. Tétel. (Multi-hipertabló – Teljesség)

A multi-hipertabló kalkulus teljes.

Bizonyítás.

Adott egy klózhalmaz, ehhez adott egy hiperrezolúciós cáfolat, melyhez az előző fejezetben leírt módon *hiperrezolúciós gráfot* konstruáltunk. Ezen gráfból a következő lépések elvégzésével építjük fel a $\mathcal{T}$ multi-hipertablót, melyhez megadjuk a 3.2. fejezetben említett

$$t : \mathcal{S}_{\mathcal{T}} \mapsto \mathcal{S}_{\mathcal{H}}$$

függvényt, ahol

- $\mathcal{S}_{\mathcal{T}}$ a $\mathcal{T}$ csúcsainak halmaza, és
- $\mathcal{S}_{\mathcal{H}}$ a hiperrezolúciós gráf csúcsainak halmaza.

A tömörebb szóbeli kifejezhetőség érdekében a következő szóhasználattal fogunk élni:

- A hiperrezolúciós gráfra egyszerűen mint *gráf* fogunk hivatkozni, míg a multi-hipertablóra mint *tabló*[8].
- A gráf valamely csúcsára mint *csúcs*, a tabló valamely csúcsára pedig mint *tablócsúcs* hivatkozunk.

[6] azaz a C_1 csúcscsoport valamely $C_1[j]$ csúcsába, illetve csúcsából vezető éle

[7] Ha $e = (X, C_1[j])$, akkor az $(X, C_i[j])$-t, ha $e = (C_1[j], X)$, akkor a $(C_i[j], X)$-et.

[8] Jóllehet, a tabló – mint fa – szintén gráf.

A tabló konstruálása során a gráf csúcscsoportjait „új”-nak, illetve „régi”-nek fogjuk minősíteni; azaz egy

$$\lambda : \mathcal{S}^{\mathcal{G}}_{\mathcal{H}} \mapsto \{\text{„új”}, \text{„régi”}\}$$

függvényt is definiálni fogunk, ahol $\mathcal{S}^{\mathcal{G}}_{\mathcal{H}}$ a gráf csúcscsoportjainak halmaza.

$\mathcal{T}$ megkonstruálása a következő lépések elvégzésével történik:

(1) (a) A gráf minden csúcscsoportját minősítsük „új”-nak.

(b) $\mathcal{T} := \{\{\top\}\}$.

(2) (a) Válasszunk a gráfban egy *levezető* $C^{\ominus}$ csúcscsoportot! Ha *nincs* ilyen csúcscsoport, akkor *vége*.

(b) Jelölje $\mathcal{N}^{\oplus}$ minden olyan $C^{\oplus}$ csúcscsoport halmazát, hogy létezik $(N^{\ominus}, N^{\oplus})$ él a gráfban úgy, hogy $N^{\ominus} \in C^{\ominus}$ és $N^{\oplus} \in C^{\oplus}$. Legyen

$$\mathcal{N} \quad := \quad \mathcal{N}^{\oplus} \cup \{C^{\ominus}\} \quad .$$

Bontsuk $\mathcal{N}$-t két diszjunkt csúcscsoport-halmazra:

$$\begin{aligned} \mathcal{N}_{\text{új}} &:= \{C \in \mathcal{N} \mid \lambda(C) = \text{„új”}\} \quad , \\ \mathcal{N}_{\text{régi}} &:= \{C \in \mathcal{N} \mid \lambda(C) = \text{„régi”}\} \quad . \end{aligned}$$

(c) Ha van olyan $C \in \mathcal{N}_{\text{új}}$ csúcscsoport, hogy valamely $N \in C$ csúcsába vagy csúcsából *több mint egy* $\mathcal{N}$-él vezet, akkor:

(i) *Sokszorozzuk meg* C-t az $\mathcal{E}_{\mathcal{N},N}$ élhalmaz mentén a gráfban, ahol $\mathcal{E}_{\mathcal{N},N}$ az összes N-be vagy N-ből vezető $\mathcal{N}$-élek halmaza.

(ii) Az összes új csúcscsoportot minősítsük „új”-nak.

(iii) Vissza a (2)-re.

(3) Vezessük be a következő jelöléseket.

(a) Jelölje $N^{\ominus}_1, \ldots, N^{\ominus}_k$ a $C^{\ominus}$ összes olyan csúcsát, hogy $N^{\ominus}_i$-ből vezet ki él $(1 \leq i \leq k)$.

(b) Jelölje $N^{\oplus}_1, \ldots, N^{\oplus}_k$ az összes olyan csúcsot, hogy létezik $(N^{\ominus}_i, N^{\oplus}_i)$ él $(1 \leq i \leq k)$.

(c) Jelölje $C^{\oplus}_1, \ldots, C^{\oplus}_k$ az összes olyan csúcscsoportot, hogy $N^{\oplus}_i \in C^{\oplus}_i$ $(1 \leq i \leq k)$.

(4) Minden $\mathcal{B} \in \mathcal{T}$ *nyílt ágra*:

(a) Minden olyan $1 \leq i \leq k$ esetén, hogy $C^{\oplus}_i \in \mathcal{N}_{\text{régi}}$:
$L^{\oplus}_i$ jelölje azon $\mathcal{B}$-beli tablócsúcsot, melyre $t(L^{\oplus}_i) = N^{\oplus}_i$, *ha létezik* ilyen tablócsúcs.

(b) Ha minden olyan $1 \leq i \leq k$ esetén, hogy $C^{\oplus}_i \in \mathcal{N}_{\text{régi}}$, *tudtunk* (az előző pontban) valamely $L^{\oplus}_i$-t megadni, akkor:

(i) Legyen

$$\widehat{C}^{\ominus} \quad := \quad \text{a } h_2(C^{\ominus}) \text{ új példánya.}$$

(ii) Minden $1 \leq i \leq k$-ra legyen

$$\widehat{C}_i^{\oplus} \quad := \quad \begin{cases} \text{a } h_2(C_i^{\oplus}) \text{ új példánya}, & \text{ha } C_i^{\oplus} \in \mathcal{N}_{\text{új}}; \\ L_i^{\oplus}, & \text{egyébként.} \end{cases}$$

(iii) Minden $1 \leq i \leq k$-ra legyen

$$\widehat{L}_i^{\oplus} \quad := \quad \begin{cases} \widehat{C}_i^{\oplus}[l], & \text{ha } C_i^{\oplus} \in \mathcal{N}_{\text{új}}; \\ L_i^{\oplus}, & \text{egyébként,} \end{cases}$$

ahol $h_2(N_i^{\oplus}) = \big(h_2(C_i^{\oplus}), l\big)$.

(iv) A következő *kiterjesztést* alkalmazzuk a $\mathcal{T}$ tabló $\mathcal{B}$ ágán:

$$e\Big(\widehat{C}^{\ominus},\ \{\widehat{C}_1^{\oplus}, \ldots, \widehat{C}_k^{\oplus}\},\ \{\widehat{L}_1^{\oplus}, \ldots, \widehat{L}_k^{\oplus}\}\Big) \quad .$$

(v) A $\mathcal{T}$ minden így kapott új L tablócsúcsára

$$t(L) \quad := \quad N \quad ,$$

ahol L-et az (i-iv) pontokban az N csúcsból hoztuk létre.

(5) Minden $C \in \mathcal{N}_{\text{új}}$ csúcscsoportra

$$\lambda(C) \quad := \quad \text{„régi”} \quad .$$

(6) Töröljük az $\mathcal{N}$-éleket a gráfból.

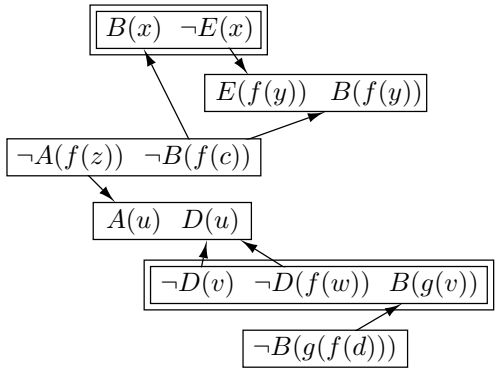
(7) Vissza a (2)-re.

A 3.2. fejezet elején rögzített (H) feltétel miatt a kezdeti hiperrezolúciós gráfnak minden csúcsából/csúcsába vezetett él. Ráadásul egy csúcscsoportból kiinduló összes élt mindig egyetlen iteráció alatt eliminátuk, és ezen iterációnak a $\mathcal{T}$ véges sok ágának a kiterjesztése felelt meg. A gráf éleit véges sok lépésben eliminátuk, és mivel a fentiek miatt a $\mathcal{T}$-be bekerülő összes literált kirezolváltuk, $\mathcal{T}$ összes ágára bekerül a $\perp$, és ezért $\mathcal{T}$ zárt. □

3.2.8. Példa.

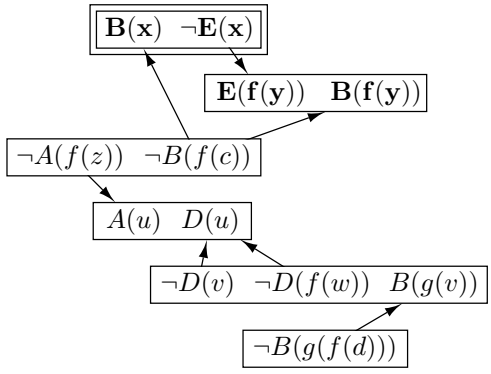
A kezdeti, csupán egyetlen, $\top$-vel címkézett csúcsból álló tabló megkonstruálása után a következő lépésekben építjük fel $\mathcal{T}$-t a 3.2.5. példában konstruált hiperrezolúciós gráfhoz.

(1) A kezdeti gráf a következő:



Ennek két levezető csúcscsoportja van, ezeket dupla keretbe tettük. Tehát közülük kell valamelyiket a 3.2.7.(2)(a) szerint $C^{\ominus}$-nak választanunk. Válasszuk önkényesen a felsőt[9].

A 3.2.7.(2)(b)-ben bevezetett $\mathcal{N}$ most a következő, vastagon szedett csúcscsoportokat tartalmazza:



$\mathcal{N}$ elemei most mindannyian „új”-ak, azaz $\mathcal{N}_{\text{új}} = \mathcal{N}$. Az $\mathcal{N}$ csúcsainak mindegyikéből/mindegyikébe most legfeljebb egy $\mathcal{N}$-él vezet[10], ezért a 3.2.7.(2)(c)-ben megfogalmazott feltétel most nem teljesül, azaz most nem kell csúcscsoportot sokszorozni.

A 3.2.7.(3)-ban bevezetett jelölések most a következőkre hivatkoznak:

[9] Azaz azt a csúcscsoportot, melynek a $B(x) \vee \neg E(x)$ input klóz van megfeleltetve (a h_2-n keresztül).

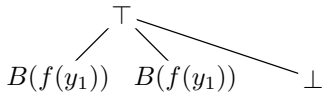
[10] Most egyetlen egy $\mathcal{N}$-él van, mégpedig a $\neg E(x)$-ből $E(f(y))$-ba húzott él.

- $k = 1$, hiszen a $C^{\ominus}$-ból egyetlen $\mathcal{N}$-él vezet ki.
- $N_1^{\ominus}$ a $\neg E(x)$-szel jelölt csúcs.
- $N_1^{\oplus}$ az $E(f(y))$-nal jelölt csúcs.
- $C_1^{\oplus}$ az $\boxed{E(f(y))\ \ B(f(y))}$-nal jelölt csúcscsoport.

A tablónak egyetlen ága van, ez nyílt, így a 3.2.7.(4)-ben bevezetett $\mathcal{B}$ most erre fog hivatkozni. Az $\mathcal{N}_{\text{régi}}$ csúcscsoport-halmaz most üres, ezért a 3.2.7.(4)(a)-ban nem vezetünk be új jelölést. Az $\mathcal{N}_{\text{régi}}$ üres volta miatt a 3.2.7.(4)(b)-ben megadott feltétel is teljesül, ezért végrehajtjuk az (i)-(v) alpontokban megadott lépéseket. Mind $\widehat{C}^{\ominus}$, mind $\widehat{C}_1^{\oplus}$ input klózok új példányai lesznek, ezeket jelölje $B(x_1)\vee\neg E(x_1)$, illetve $E(f(y_1))\vee B(f(y_1))$[11]. Az $\widehat{L}_1^{\oplus}$ nem más, mint $E(f(y_1))$[12], és így a (iv)-ben előállított kiterjesztés nem más, mint

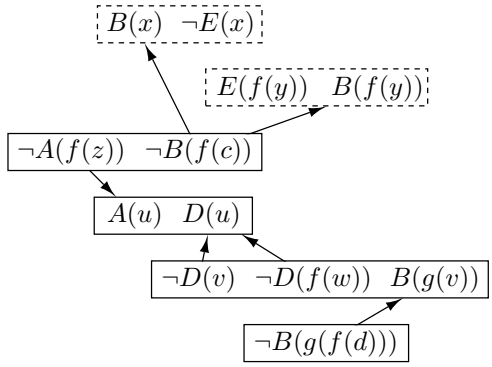
$$\left[B(x_1) \vee B(f(y_1))\ ,\ x_1/f(y_1)\right] \quad .$$

Ezt alkalmazva a $\mathcal{T}$ tabló $\mathcal{B}$ ágán a következő tablót kapjuk:



Az (v)-ben elvégzett lépés azt eredményezi, hogy a *bal* oldali $B(f(y_1))$-gyel címkézett tablócsúcshoz hozzárendeljük (t által) a gráf $B(x)$-szel jelölt csúcsát, illetve a *jobb* oldali $B(f(y_1))$ tablócsúcshoz a gráf $B(f(y))$-nal jelölt csúcsát.

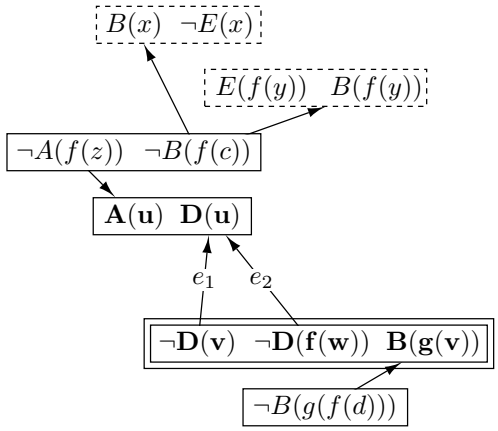
A 3.2.7.(5)-ben a gráf két felső csúcscsoportját „régi"-vé minősítjük; a gráf „régi" csúcscsoportjait szaggatott keretben fogjuk ábrázolni. Végül a 3.2.7.(6) alapján az $\mathcal{N}$-éleket töröljük a gráfból, tehát az első iteráció eredményeként a következő gráfot kapjuk:



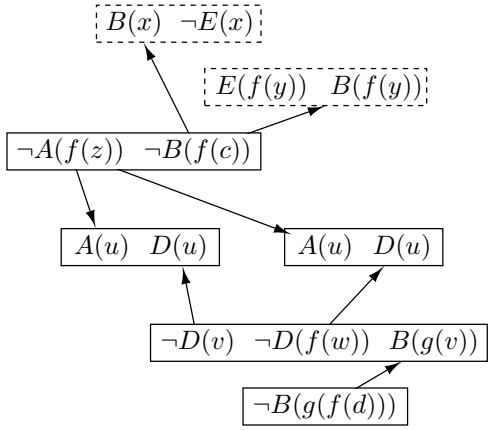
[11] Egy klóz új példányainak létrehozásakor az adott klózban szereplő változókat számokkal indexelt változataikkal fogjuk behelyettesíteni.

[12] Mivel az $N_1^{\oplus}$ saját csúcscsoportjának az 1. literálja, és a $\widehat{C}_1^{\oplus}$-nek az 1. literálja $E(f(y_1))$.

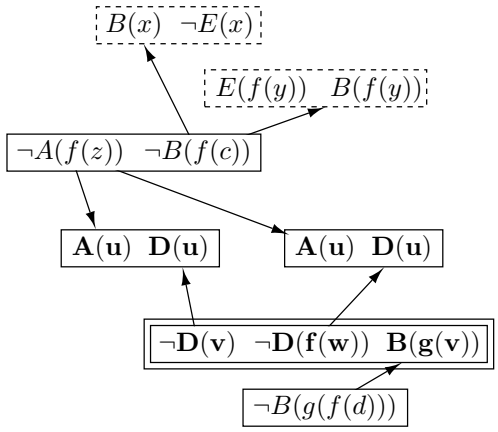
(2) Ezen iteráció elején is két levezető csúcsunk van. Válasszuk $C^{\ominus}$-nak a $\boxed{\neg D(f(v))\ \ \neg D(f(w))\ \ B(g(v))}$-vel jelölt csúcscsoportot. A vastagon szedett csúcscsoportok lesznek az $\mathcal{N}$ elemei:



Az ábráról leolvasható, hogy $\mathcal{N}_{\text{új}} = \mathcal{N}$ és $\mathcal{N}_{\text{régi}} = \emptyset$. Látható, hogy a 3.2.7.(2)(c) feltétele teljesül, hiszen az $\boxed{A(u)\ \ D(u)}$ csúcscsoport 2. csúcsába két $\mathcal{N}$-él vezet; ezeket az ábrán e_1-gyel és e_2-vel jelöltük. *Sokszoroznunk kell* a csúcscsoportot az $\{e_1, e_2\}$ mentén, melynek eredménye a következő gráf lesz:



(3) Válasszuk $C^{\ominus}$-nak ismét a $\boxed{\neg D(f(v))\ \ \neg D(f(w))\ \ B(g(v))}$ csúcscsoportot.



$\mathcal{N}$ elemei ismét mind elemei $\mathcal{N}_{\text{új}}$-nak, melynek minden csúcsából/csúcsába most legfeljebb egy-egy $\mathcal{N}$-él vezet, azaz egy csúcscsoportot sem kell sokszorozni.

A 3.2.7.(3)-ban bevezetett jelölések most a következőképpen alakulnak:

- $k = 2$.
- $N_1^{\ominus}$ a $C^{\ominus}$ $\neg D(v)$-vel, $N_2^{\ominus}$ a $\neg D(f(w))$-vel jelölt csúcsa.
- $N_1^{\oplus}$ a gráf bal oldali $D(u)$-val, $N_2^{\oplus}$ a jobb oldali $D(u)$-val jelölt csúcsa.
- $C_1^{\oplus}$ a bal oldali, $C_2^{\oplus}$ a jobb oldali $\boxed{A(u) \quad D(u)}$-val jelölt csúcscsoport.

A tabló mindkét ága nyílt, a 3.2.7.(4)(b)(i-v) lépéseket mindkettőre elvégezzük, hiszen az $\mathcal{N}_{\text{régi}}$ üres volta miatt a 3.2.7.(4)(b) feltétel mindkét ágra teljesül. A *bal oldali ág* esetén a 3.2.7.(4)(b)(i-iii)-ban bevezetett jelölések most a következőképpen alakulnak:

$$\begin{aligned}
\widehat{C}^{\ominus} &= \neg D(v_1) \vee \neg D(f(w_1)) \vee B(g(v_1)) \\
\widehat{C}_1^{\oplus} &= A(u_1) \vee D(u_1) \\
\widehat{C}_2^{\oplus} &= A(u_2) \vee D(u_2) \\
\widehat{L}_1^{\oplus} &= D(u_1) \\
\widehat{L}_2^{\oplus} &= D(u_2)
\end{aligned}$$

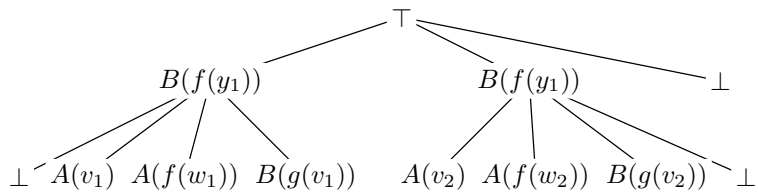
Az előállított kiterjesztés:

$$\left[A(u_1) \vee A(u_2) \vee B(g(v_1)) \ , \ \{u_1/v_1, u_2/f(w_1)\}\right] \quad .$$

Ezzel analóg módon állíthatjuk elő a *jobb oldali ághoz* az

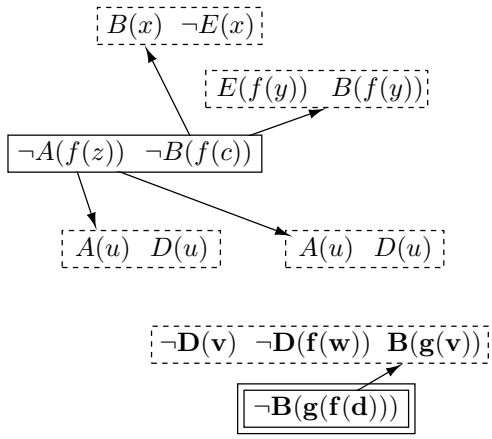
$$\left[A(u_3) \vee A(u_4) \vee B(g(v_2)) \ , \ \{u_3/v_2, u_4/f(w_2)\}\right]$$

kiterjesztést. Ezek alkalmazása után $\mathcal{T}$ a következő lesz:



A 3.2.7.(4)(b)(v)-ben az új tablócsúcsok és a gráf csúcsai közti összerendeléseket elvégezzük. Ezek alapján az $A(v_1)$ és $A(v_2)$ tablócsúcsokhoz a gráf bal oldali $A(u)$-val jelölt csúcsa, az $A(f(w_1))$ és $A(f(w_2))$ tablócsúcsokhoz a gráf jobb oldali $A(u)$-val jelölt csúcsa, illetve a $B(g(v_1))$ és $B(g(v_2))$ tablócsúcsokhoz a gráf $B(g(v))$-vel jelölt csúcsa lesz rendelve.

(4) A gráf jelenleg:



Mivel a $\boxed{\neg D(f(v))\ \ \neg D(f(w))\ \ B(g(v))}$ csúcscsoport most az $\mathcal{N}_{\text{régi}}$ eleme, és $N_1^{\oplus}$ most a csúcscsoportnak a $B(g(v))$-vel jelölt csúcsa, így a tabló minden nyílt ága esetén *megpróbálunk az ágon olyan tablócsúcsot találni*, melyhez ez a csúcs van hozzárendelve. Két ilyen ág van a tablóban: a $B(g(v_1))$-et, illetve a $B(g(v_2))$-t tartalmazó ágak. Példaként nézzük az előbbit. Mivel a 3.2.7.(4)(a)-ban bevezetjük az $L_1^{\oplus} = B(g(v_1))$ jelölést, a 3.2.7.(4)(b)(i-iii) jelölései most a következők lesznek:

$$\begin{aligned} \widehat{C}^{\ominus} &= \neg B(g(f(d))) \\ \widehat{C}_1^{\oplus} &= B(g(v_1)) \\ \widehat{L}_1^{\oplus} &= B(g(v_1)) \end{aligned}$$

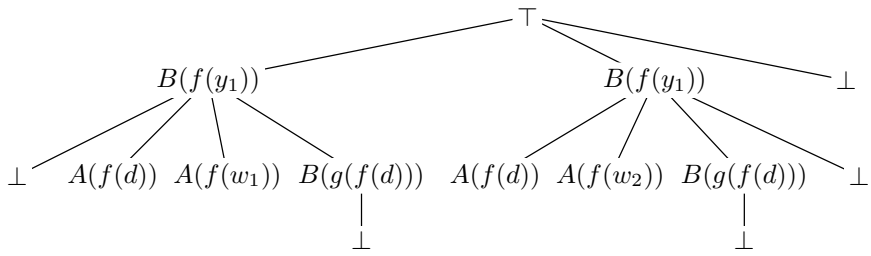
Tehát ezen ágra a

$$\left[\bot\ ,\ v_1/f(d)\right]$$

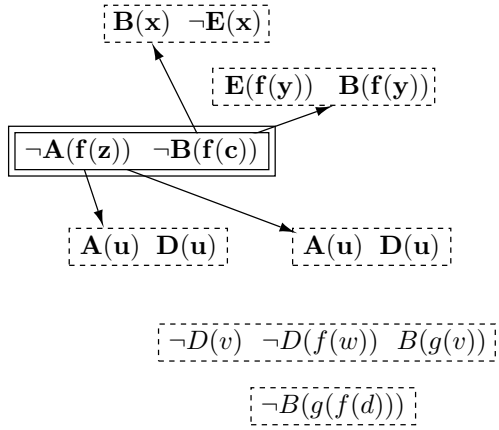
kiterjesztést fogjuk végrehajtani. Hasonlóképp, a $B(g(v_2))$-t tartalmazó ágra a

$$\left[\bot\ ,\ v_2/f(d)\right]$$

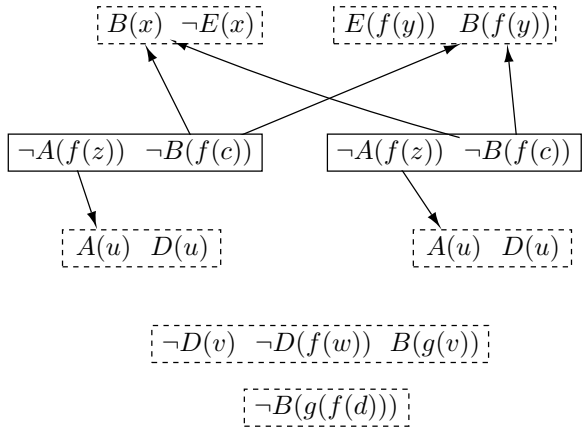
kiterjesztést fogjuk alkalmazni. A $\mathcal{T}$ tehát most a következő:



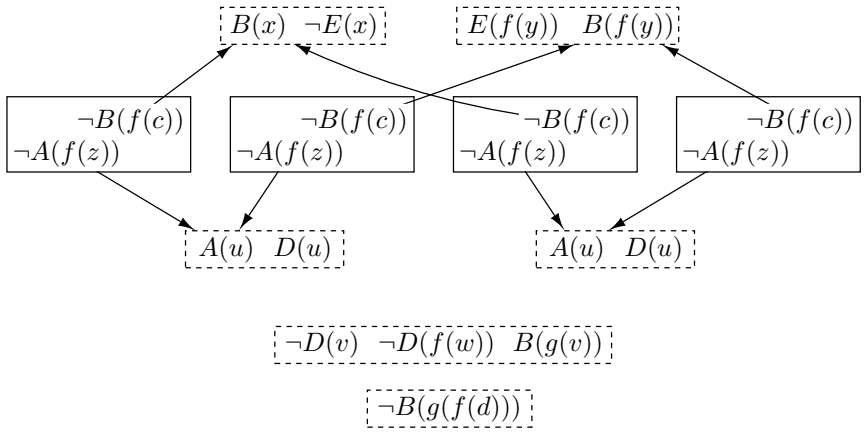
(5) A gráfban most már csak a $\boxed{\neg A(f(z))\ \ \neg B(f(c))}$ levezető csúcscsoportunk van:



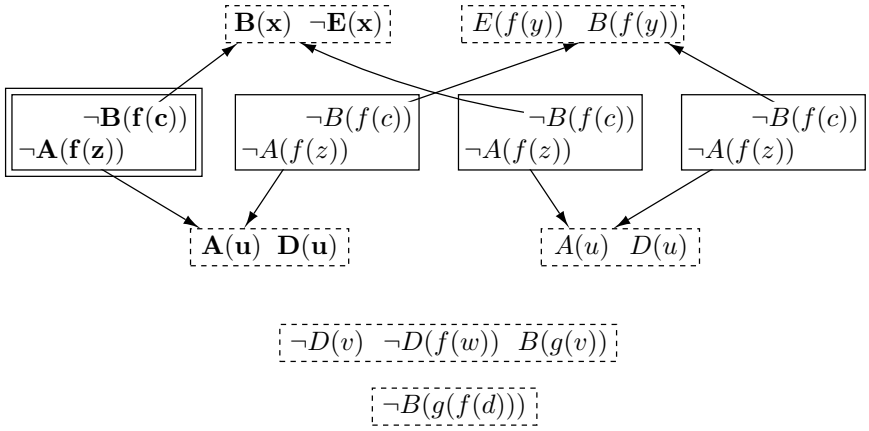
Látható, hogy a $\boxed{\neg A(f(z))\ \ \neg B(f(c))}$-nak két csúcsából két-két $\mathcal{N}$-él indul, ezért ennek a csúcscsoportnak *három másolatát* fogjuk a gráfba beszúrni, összesen három iterációban. Az első iterációban sokszorozzuk meg a csúcscsoportot az $\neg A(f(z))$-vel jelölt csúcsából kiinduló élei mentén:



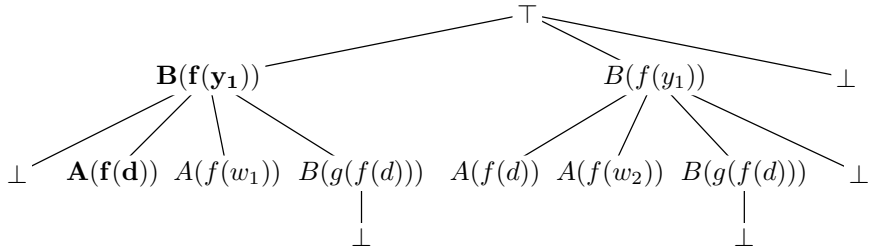
Majd mindkét $\boxed{\neg A(f(z))\ \ \neg B(f(c))}$ csúcscsoportot sokszorozzuk meg a $\neg B(f(c))$-vel jelölt csúcsaikból induló élek mentén, két iterációban. Ennek eredménye:



(6) Válasszuk a négy levezető csúcscsoport közül a bal oldali szélsőt:



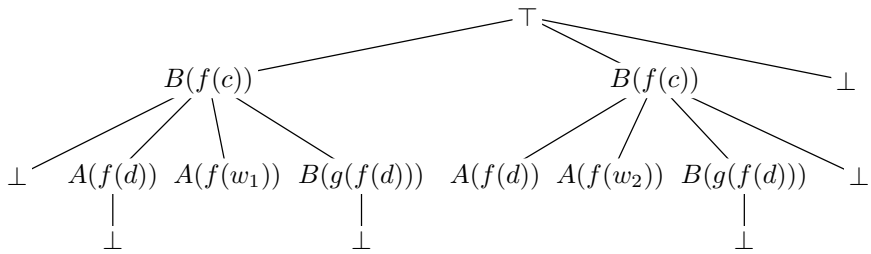
Ezt a csúcscsoportot már nem kell sokszorozni. Az $\mathcal{N}_{\text{régi}}$ most kételemű, a 3.2.7.(4)(a)-ban olyan ágát keressük a tablónak, melyen van mind a $B(x)$-szel jelölt csúcsnak, mind a bal oldali $A(u)$-val jelölt csúcsnak megfelelő tablócsúcs. A $\mathcal{T}$-ben egy ilyen ág van, az ábrában a keresett tablócsúcsokat vastagon szedjük:



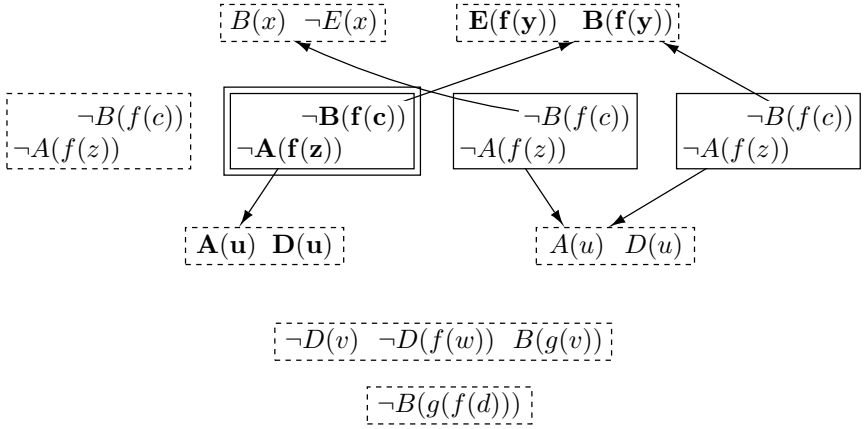
Ezen ágra fogjuk alkalmazni a

$$\big[\bot\ ,\ \{z_1/d, y_1/c\}\big]$$

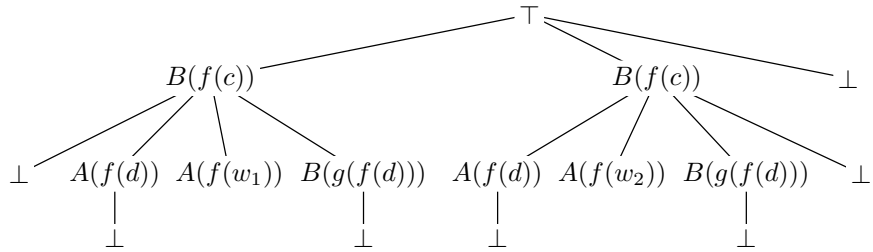
kiterjesztést, melynek eredménye:



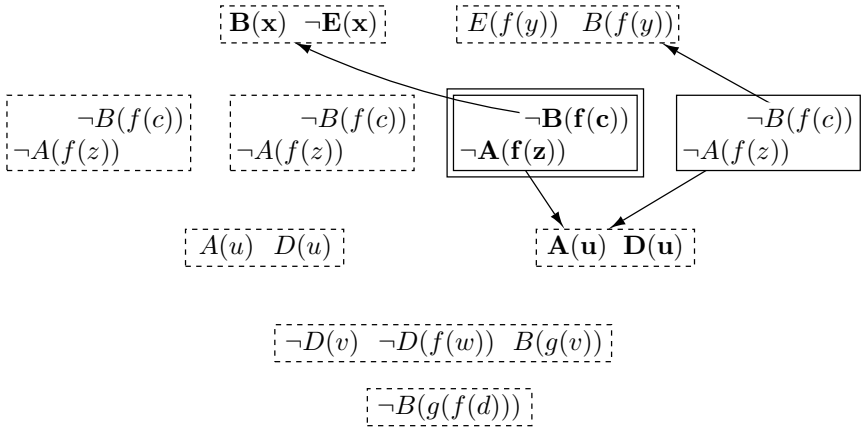
(7) Az $\mathcal{N}$ elemeit most a következőképpen választjuk meg:



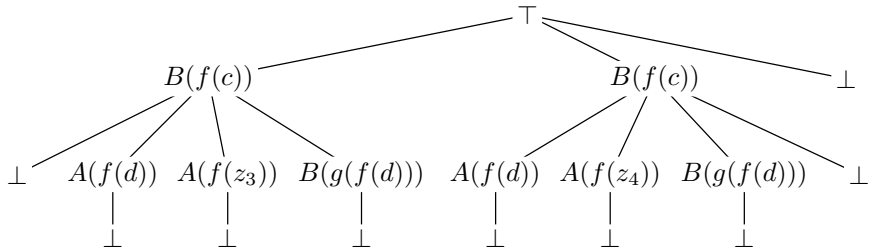
A tabló kiterjesztendő ága most az ábrán balról ötödik ág lesz, a $[\bot, z_2/d]$ kiterjesztést alkalmazzuk rá:



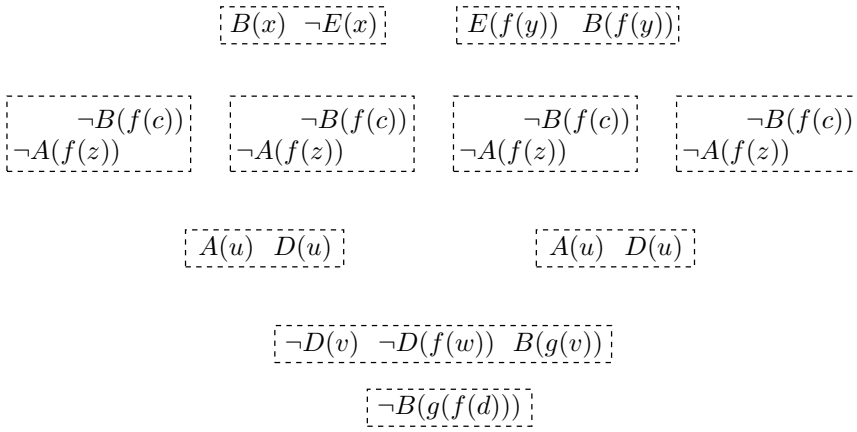
(8) Jelenlegi gráfunk:



Ebben az iterációban a tabló balról harmadik ágára végezzük el a $[\bot, w_1/z_3]$ kiterjesztést. Ezen iteráció után már csak egy levezető csúcscsoport marad a gráfban és csak egy nyílt ág a tablóban. Ezen utóbbira az utolsó iterációban a $[\bot, w_2/z_4]$ kiterjesztést alkalmazzuk. Tehát $\mathcal{T}$ végső formája:



A hiperrezolúciós gráfnak pedig ezennel minden élét eliminálták:



Ezért a gráfban nincs levezető csúcscsoport, vagyis a 3.2.7.(2)(a) miatt az algoritmus végrehajtása véget ér. □

3.3. Klózgenerálás tablóval

A multi-hipertabló klózokon működő tételbizonyító, csakúgy mint a klóztabló, vagy mint a rezolúció különböző válfajai. Bármely $\mathcal{F}$ véges formulahalmazhoz megadható olyan $\mathcal{C}$ klózhalmaz, hogy $\mathcal{F}$ akkor és csak akkor kielégíthető, ha $\mathcal{C}$ is az. Kérdés, hogy hogyan állítsuk elő $\mathcal{F}$-hez $\mathcal{C}$-t?

A legtöbb klózgeneráló algoritmus közvetlenül a klózgenerálásra vonatkozó ekvivalens átalakításokat alkalmazza a formulán – ilyenek például a de Morgan-azonosságok, az implikáció átírása diszjunkcióra vagy a disztributivitás szabályai. Az ilyen algoritmusokról találunk részletes áttekintést a [21] könyvfejezetben. Elenyésző számban, de

léteznek más, alternatív módszerek is. Ilyenek például a BDD-ken (Binary Decision Diagram) vagy más néven Shannon-gráfokon alapuló módszerek [21, 24], melyeknél a BDD mint köztes forma használt; azaz $\mathcal{F}$-ből előbb BDD-t generálunk, majd ezen utóbbiból generáljuk $\mathcal{C}$-t. A BDD-k alapvetően propozícionális logikában alkalmazhatóak, ám napjainkban egyre inkább feléjük fordul a figyelem elsőrendű logika kapcsán is. Sokszor párhuzamot vonnak a BDD-k és a tablók között [24], összehasonlítják klózgenerálásban és tételbizonyításban is a képességeiket.

Hasonlóan a tablókkal párhuzamosíthatóak a pp-kifejezéseken, a formulák duál formáján és egymásba ágyazott listákon alapuló algoritmusok [11, 22], ám ezek is csak nulladrendű logikában alkalmazhatóak. Az ilyen algoritmus formulák diszjunktív normálformára (DNF), illetve konjunktív normálformára (KNF) – más néven *klóz normálformára* – hozását végzi el. Például $\mathcal{F}$-hez DNF-re hozás esetén egy $[Q_1, \ldots, Q_m]$ listát konstruál, ahol minden Q_i $\langle L_1, \ldots, L_k \rangle$ alakú, ahol minden L_j literál. Azaz egy $\langle \ldots \rangle$ lista a saját elemeinek konjunkcióját, egy $[\ldots]$ lista az elemeinek diszjunkcióját reprezentálja; vagyis a DNF-re hozáskor előállított $[\langle \ldots \rangle, \ldots, \langle \ldots \rangle]$ lista felfogható egy *analitikus tablónak* is. KNF-re hozás esetén egy $\langle [\ldots], \ldots, [\ldots] \rangle$ listát állít elő az algoritmus, mely pedig egy *analitikus duál tablónak*[13] felel meg. A tablók és a normálformák kapcsolata régóta ismert a szakirodalomban, természetesen itt is csak propozícionális esetről beszélünk. Egy nulladrendű formulához generált befejezett analitikus tablóból a következőképpen nyerhető a formula DNF-je: minden ág esetén az adott ágon szereplő literáloknak vegyük a konjunkcióját, és az így kapott formuláknak a diszjunkcióját. Egy nulladrendű formulához generált befejezett analitikus duál tablóból pedig hasonlóképpen nyerhető a formula KNF-je: a literáloknak a diszjunkcióját kell vennünk, majd ezeknek a konjunkcióját. Az üres konjunkciónak a $\top$, az üres diszjunkciónak a $\bot$ felel meg.

Elsőrendű esetben a duál tabló egy az egyben nem használható klózgenerálásra. Ez azon múlik, hogy egy elsőrendű formula prímkomponensei a literálokon kívül kvantált formulák (γ- és δ-formulák) is lehetnek. A 2.1.1. fejezetben ismertetett *szabadváltozós tabló* γ- és δ-szabályai (2.3. ábra) tulajdonképpen a formula kifejtését imitálják az adott kvantor hatáskörébe tartozó formula egy példányának bevezetésével, úgy hogy nem sértik a kielégíthetőségi feltételeket. Ezek a γ- és δ-szabályok kiváltják a formula prenexizálását és skolemizálását, és végül a klózokban csak literálok szerepelnek.

Elsőrendű logikában klózgenerálásra tehát a következő tablómódszert használhatjuk:

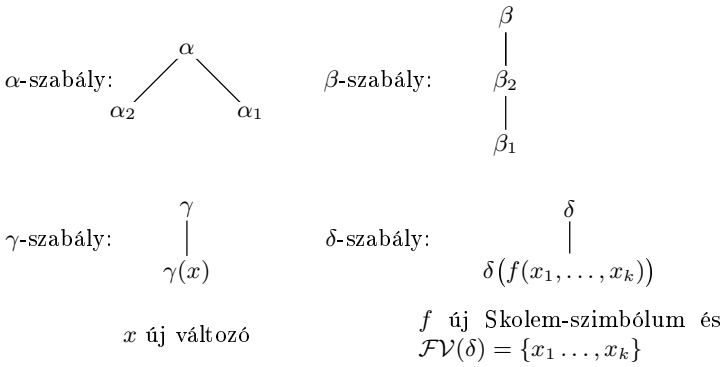
3.3.1. Definíció. (Klózgeneráló tabló)

Adott egy $\mathcal{F} = \{F_1, \ldots, F_n\}$ formulahalmaz. $\mathcal{F}$-hez a következő induktív definícióval adjuk meg a *klózgeneráló tabló* [18] fogalmát:

(1) $\big\{\{F_1\}, \ldots, \{F_n\}\big\}$ az $\mathcal{F}$ klózgeneráló tablója.

(2) Ha $\mathcal{T}$ az $\mathcal{F}$ klózgeneráló tablója, akkor $\mathcal{T}'$ is az, amennyiben $\mathcal{T}'$-t a 3.1. ábrán látható szabályok valamelyikének alkalmazásával nyertük $\mathcal{T}$-ből. □

[13] Az analitikus duál tabló szabályait az analitikus tabló α- és β-szabályainak felcserélésével kapjuk.



3.1. ábra. Klózgeneráló tabló – levezetési szabályok

Fontos momentum, hogy a tablóba kerülő minden formulára egy ágon legfeljebb *egyszer alkalmazhatunk* szabályt, szemben a tételbizonyításban használt tablómódszerekkel[14]. Ezt szemléletesen úgy fogjuk megoldani, hogy minden formulát, amire már alkalmaztunk szabályt, töröljük az ágról mint címkét. Egy klózgeneráló tablót *befejezettnek* tekintünk, ha már nem tudunk szabályt alkalmazni egy ágának egy csúcsára sem; azaz ha a tabló már *csak literálokat* tartalmaz. Azt kell bebizonyítanunk, hogy egy $\mathcal{F}$ formulahalmazhoz generált befejezett klózgeneráló tabló valóban olyan klózhalmazt állít elő, mely pontosan akkor kielégíthető, ha $\mathcal{F}$ is az. Ehhez előbb definiálunk egy olyan függvényt, mely a tablóhoz egy formulát rendel, majd pedig bebizonyítunk egy lemmát.

3.3.2. Definíció.

Az ágakon és tablókon értelmezett $cnf()$ függvényt a következőképpen definiáljuk:

(1) Egy $\mathcal{B} = \{A_1, \ldots, A_k\}$ ág esetén $cnf(\mathcal{B}) = (A_1 \vee \ldots \vee A_k)$.

(2) Egy $\mathcal{T} = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$ tabló esetén $cnf(\mathcal{T}) = cnf(\mathcal{B}_1) \wedge \ldots \wedge cnf(\mathcal{B}_k)$. □

3.3.3. Lemma.

Egy $\mathcal{T}$ klózgeneráló tabló esetén $cnf(\mathcal{T})$ akkor és csak akkor kielégíthető, ha valamely tablószabály alkalmazásával belőle nyert $\mathcal{T}'$ tabló esetén is kielégíthető $cnf(\mathcal{T}')$.

Bizonyítás.

Két esetre osztjuk a bizonyítást aszerint, hogy milyen szabályt alkalmazunk $\mathcal{T}$-ben.

(1) Ha α-, β- vagy γ-szabályt alkalmazunk:
Tegyük fel, hogy $cnf(\mathcal{T})$ kielégíthető. Mivel létezik olyan $\mathcal{M}$ modell, hogy

[14] Analitikus tabló esetén a γ-formulákra kell többszörös alkalmazhatóságot engedélyezni, ezen kalkulus csak így lesz teljes.

$\mathcal{M} \models cnf(\mathcal{T})$, így minden $\mathcal{B} \in \mathcal{T}$ ágra $\mathcal{M} \models cnf(\mathcal{B})$. Egy szabály alkalmazásával $\mathcal{T}$-nek csak egy ága módosul, így elég most erre az egy ágra összpontosítani. Ezt az ágat $\mathcal{B}$-vel fogjuk jelölni az alábbiakban.

(a) Ha $\mathcal{B} = \mathcal{X} \cup \{\beta\}$, azaz $\mathcal{M} \models cnf(\mathcal{X}) \vee (\beta_1 \vee \beta_2)$:

β-ra alkalmazunk szabályt, aminek eredményeként a

$$\mathcal{B}' \quad = \quad \mathcal{X} \cup \{\beta_1, \beta_2\}$$

ágat kapjuk. Mivel

$$cnf(\mathcal{B}') \quad = \quad cnf(\mathcal{X}) \vee \beta_1 \vee \beta_2 \quad ,$$

így

$\mathcal{M} \models cnf(\mathcal{B})$ akkor és csak akkor, ha $\mathcal{M} \models cnf(\mathcal{B}')$.

(b) Ha $\mathcal{B} = \mathcal{X} \cup \{\alpha\}$, azaz $\mathcal{M} \models cnf(\mathcal{X}) \vee (\alpha_1 \wedge \alpha_2)$:

α-ra alkalmazunk szabályt, aminek eredményeként a

$$\begin{array}{lcl} \mathcal{B}'_1 & = & \mathcal{X} \cup \{\alpha_1\} \quad \text{és a} \\ \mathcal{B}'_2 & = & \mathcal{X} \cup \{\alpha_2\} \end{array}$$

ágakat kapjuk. Mivel

$$\begin{array}{lcl} cnf(\mathcal{B}'_1) & = & cnf(\mathcal{X}) \vee \alpha_1 \quad \text{és} \\ cnf(\mathcal{B}'_2) & = & cnf(\mathcal{X}) \vee \alpha_2 \quad , \end{array}$$

így

$\mathcal{M} \models cnf(\mathcal{B})$ akkor és csak akkor, ha
$\mathcal{M} \models cnf(\mathcal{B}'_1) \wedge cnf(\mathcal{B}'_2)$.

(c) Ha $\mathcal{B} = \mathcal{X} \cup \{\gamma\}$, azaz $\mathcal{M} \models cnf(\mathcal{X}) \vee \forall x\ \gamma(x)$:

γ-ra alkalmazunk szabályt, aminek eredményeként a

$$\mathcal{B}' \quad = \quad \mathcal{X} \cup \{\gamma(x)\}$$

ágat kapjuk. Az x új változó, azaz $x \notin \mathcal{FV}(T)$. Vagyis $x \notin \mathcal{X}$. Mivel

$$cnf(\mathcal{B}') \quad = \quad cnf(\mathcal{X}) \vee \gamma(x) \quad ,$$

így

$\mathcal{M} \models cnf(\mathcal{B})$ akkor és csak akkor, ha $\mathcal{M} \models cnf(\mathcal{B}')$.

(2) Ha δ-szabályt alkalmazunk, azaz ha valamely $\mathcal{B} \in \mathcal{T}$-re $\mathcal{B} = \mathcal{X} \cup \{\delta\}$:

δ-ra alkalmazunk szabályt, aminek eredményeként a

$$\mathcal{B}' \quad = \quad \mathcal{X} \cup \{\delta(f(x_1, \ldots, x_k))\}$$

ágat kapjuk, ahol f új Skolem-szimbólum és $x_1, \ldots, x_k$ a δ összes szabad változója.

Mivel – Skolem nyomán [27] – a $cnf(\mathcal{T})$ formula akkor és csak kielégíthető, ha annak skolemizáltja is kielégíthető, így $cnf(\mathcal{T}')$ is kielégíthető. □

3.3.4. Tétel.

Egy $\mathcal{F}$ formulahalmaz akkor és csak akkor kielégíthetetlen, ha a

$$\mathcal{C} \quad = \quad \{cnf(\mathcal{B}) \mid \mathcal{B} \in \mathcal{T}\}$$

kielégíthetetlen, ahol $\mathcal{T}$ az $\mathcal{F}$ klózgeneráló tablója.

Bizonyítás.

Jelöljük $\mathcal{T}_0$-val az $\mathcal{F}$ kezdeti klózgeneráló tablóját. Nyilvánvalóan $cnf(\mathcal{T}_0)$ akkor és csak akkor kielégíthetetlen, ha $\mathcal{F}$ kielégíthetetlen.

A 3.3.3. lemma alapján $cnf(\mathcal{T}_0)$ akkor és csak akkor kielégíthetetlen, ha $cnf(\mathcal{T})$ is kielégíthetetlen.

Továbbá, $\mathcal{C}$ definíciója alapján $cnf(\mathcal{T})$ akkor és csak akkor kielégíthetetlen, ha $\mathcal{C}$ is kielégíthetetlen. □

Vegyük észre, hogy a fenti tételben ha $\mathcal{T}$ *befejezett*, akkor $\mathcal{C}$ *klózhalmaz*. Azaz a $\mathcal{C}$ a keresett klózhalmaz: pontosan akkor kielégíthetetlen, ha $\mathcal{F}$ kielégíthetetlen.

A *multi-hipertabló* és a klózgeneráló tabló módszereket a következőképpen kapcsolhatjuk tehát össze egy tételbizonyítón belül:

(1) Állítsuk elő $\mathcal{F}$ befejezett klózgeneráló tablóját.

(2) Alkalmazzuk a multi-hipertabló kalkulust ezen klózgeneráló tabló (csak literálokat tartalmazó) *ágain mint klózokon*.

Vegyük észre, hogy $\mathcal{F}$ formuláihoz akár külön-külön is előállíthatjuk a saját befejezett klózgeneráló tablójukat.

3.3.1. Lineáris algoritmus DAG-gal

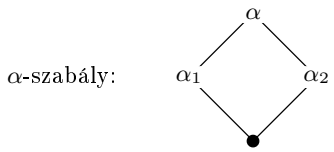
A klózgeneráló algoritmusok szinte mindegyike exponenciális bonyolultságú – ez a disztributivitásra vonatkozó ekvivalens átalakítások alkalmazása miatt van így. A legrosszabb eset az, mikor az eredeti formula DNF-ben van. Ekkor az algoritmus végén kapott klózok száma legfeljebb l^k, ahol k a DNF-ben szereplő elemi konjunkciók száma, l pedig a literálok maximális száma egy-egy elemi konjunkcióban. Ezt a korlátot

természetesen nem lehet csökkenteni. Lehet azonban a klózgeneráló algoritmusok *bonyolultságát* (azaz lépésszámát), ahogy azt a klózgeneráló tabló esetén a továbbiakban megmutatjuk.

A fentebb vázolt klózgeneráló tabló is *exponenciális algoritmus* klózgenerálásra. A probléma az, hogy egyes formulák a tabló több ágán is előfordulhatnak, így ezeket – bár azonos formulákról van szó – külön-külön kell feldolgozni. Tulajdonképpen itt is a disztributivitásból adódó probléma bukkan fel, mégpedig az α-szabály alkalmazásokon keresztül. Könnyű észrevenni, hogy a problémát maga a tabló – mint *fa adatszerkezet* – okozza. Az α-szabály alkalmazásakor egy adott ág kettéágazik, így az eredeti ágon található (még fel nem dolgozott) összetett formulákat a két kapott ágon a jövőben külön-külön kell majd feldolgozni. Ezért a fának olyan módosítása lenne előnyös, amely tartalmazhat hurkokat – ez pedig az irodalomban a DAG-ként ismert adatszerkezet, mely *irányított körmentes gráfot*[15] takar.

A DAG-ok egyébként nem ismeretlenek a logikai alkalmazások számára. Robinson exponenciális unifikációs algoritmusát a 70-es években Paterson és Wegman [23] próbálta linearizálni, DAG-ok alkalmazásával. Ezen unifikációs algoritmus egyazon változó különböző előfordulásait csupán mint az eredeti változóra mutató referenciákat tárolja, ily módon a változó helyettesítése csak egy helyen történik meg, és nem annak minden előfordulásában. A logikai alkalmazásokban széles körben alkalmazandóak a DAG-ok, bárhol, ahol a többszörözések kiküszöbölésével hatékony algoritmusok készíthetők. A DAG-ok alkalmazását nem csak hatékonysági szempontok vezérelhetik: például a DAG-okkal sokkal természetesebb módon ábrázolhatók a Kripke-féle modális logikai szemantika világai közötti relációk, mint azt Castilho és társai [7] is felismerték, akik ennek fényében alkották meg DAG-okon alapuló modális tablómódszerüket.

Esetünkben csupán a hatékonyságnövelés motiválja a DAG-ok bevezetését, illetve az, hogy az így nyert klózgeneráló tabló sokkal szemléletesebb képet mutat a klózgenerálás folyamatáról. Ahogy Paterson és Wegman nagyfokú bonyolultságbeli csökkenést produkált a reprezentációnak egy kis módosításával az unifikációs algoritmusban (exponenciálisról lineárisra csökkent a bonyolultsága), úgy mi is ezt fogjuk tenni a klózgeneráló tabló esetében. Mint fentebb említettük, a klózgeneráló tablóban csupán az α-szabály kerül módosításra, mint az a 3.2. ábrán látható. Látszik, hogy minden elágazást nyomban összezárunk, mégpedig egy címke nélküli csúcsban.



3.2. ábra. Klózgeneráló DAG – módosult α-szabály

Másik nagyon fontos dolog a szabályok alkalmazásának módja. Míg tablókalkulu-

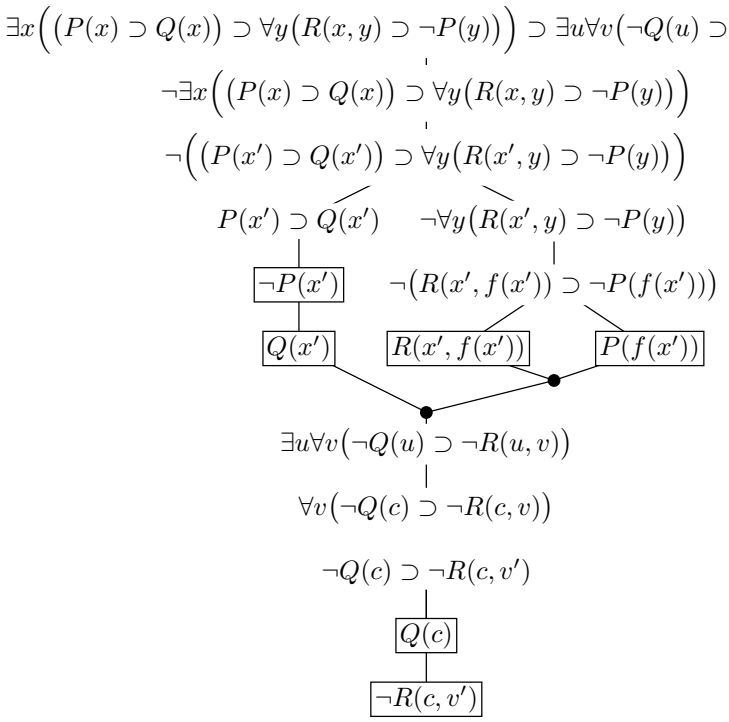
[15] angol: *directed acyclic graph*

sok esetén a hagyományos eljárás az, hogy valamely szabályalkalmazás által az adott ágra kerülő új formulá(ka)t az adott ág aljára fűzik fel, addig jelen esetben az új csúcso(ka)t mindig közvetlenül a *feldolgozandó csúcs alá* szúrjuk be. A módosult szabályokkal felépített tabló tehát egy DAG lesz, melyben az ág fogalma a következőképpen definiált: egy a gyökérből a levélcsúcsba vezető út. Vegyük észre, hogy a tablóban *pontosan egy levélcsúcs* van.

Vitathatatlan, hogy a vázolt algoritmus *lineáris bonyolultságú*, hiszen a befejezett klózgeneráló DAG megkonstruálásához szükséges szabályalkalmazások száma lineáris függvénye a kiindulási formulahalmazban szereplő logikai összekötőjelek számának.

3.3.5. Példa.

Az alábbiakban a gyökérben elhelyezett formula befejezett klózgeneráló DAG-ja szerepel:



Jól látható, hogy a tablóban minden A formula alá az(ok) a formula (formulák) kerül(nek), mely(ek)et szabályalkalmazással A-ból nyertünk. A levezetés során két Skolem-szimbólumot vezetünk be: f-et és c-t. Az ábrán a tablóból nem töröltük a szabályalkalmazások által eltávolított összetett formulákat, inkább bekeretezéssel megjelöltük azokat a csúcsokat, melyek ténylegesen címkék a befejezett tablóban. Ezekből állnak végül össze a következő klózok, azaz a gyökérben szereplő formulához keresett

$\mathcal{C}$ klózhalmaz:

$$\begin{aligned} \neg P(x') \vee Q(x') \vee Q(c) \vee \neg R(c, v') \\ R(x', f(x')) \vee Q(c) \vee \neg R(c, v') \\ P(f(x')) \vee Q(c) \vee \neg R(c, v') \end{aligned}$$ □

3.3.2. Ágak zártságán alapuló klózhalmaz-egyszerűsítés

$\mathcal{C}$-ben eliminálhatjuk azokat a klózokat, melyek *érvényesek*, hiszen $\mathcal{C}$ kielégíthetőségét ezek nem befolyásolják. Az érvényes klózok detektálása nagyon egyszerűen, a klózgeneráló tablóra épülve megtehető. A klózgeneráló tablóban egy-egy ág lezárásával láthatjuk be, hogy az adott ág (mint formulahalmaz) érvényes. A befejezett klózgeneráló tabló *minden ágát megpróbáljuk lezárni*, és ha ez egy adott ág esetén sikerül, akkor az adott ágat (mint klózt) nem vesszük fel $\mathcal{C}$-be.

Az ágak lezárhatóságát a 2.1.1. fejezetben leírt MGU lezárási szabályon alapulva fogjuk megtenni, a következő egyszerű módon a befejezett klózgeneráló tabló minden $\mathcal{B}$ ága esetén:

> Ha van két olyan $L_1, L_2 \in \mathcal{B}$ különböző előjelű literál, hogy $\mathcal{U}(\underline{L_1}, \underline{L_2})$ létezik, akkor $\mathcal{B}$-t (mint klózt) nem vesszük fel $\mathcal{C}$-be.

A 3.3.5. példában szereplő befejezett klózgeneráló tabló ágai közül az $\{R(x', f(x')), Q(c), \neg R(c, v')\}$-ből generálható

$$R(x', f(x')) \vee Q(c) \vee \neg R(c, v')$$

klózt nem vesszük fel $\mathcal{C}$-be, hiszen az $R(x', f(x'))$ és a $\neg R(c, v')$ ellentétes előjelű literálok unifikálhatóak az $\{x'/c, v'/f(c)\}$ MGU-val.

3.4. Redundancia

A redundancia fogalma általános eszköz egy rendszeren belüli ismétlődések kezelésére: definiáljuk, hogy egy rendszer mikor redundáns, majd leírjuk annak menetét, hogyan szüntethető meg a rendszer redundanciája. Ily módon ejtjük ki azokat a tényezőket a rendszerből, melyek nem relevánsak annak viselkedésére nézve. Továbbá ez lesz az eszköze, hogy elejét vegyük annak, hogy a rendszer egy vég nélküli ismétlődésben ragadjon meg; így a redundancia vizsgálata elengedhetetlen egy gyakorlatban is használni kívánt rendszer esetében. Jelen fejezetben formulák redundanciáját fogjuk definiálni, vizsgálni, különböző célokra, egyszerűsítésekre felhasználni.

3.4.1. Definíció. (Redundancia)

Egy A *formula redundáns* egy B formulára és a $\circ$ szimmetrikus, bináris logikai összekötőjelre nézve, ha $A \circ B \sim B$. □

Azaz a B formulához teljesen felesleges $\circ$-vel A-t hozzáfűznünk, azzal a formula értéke egyik modellben sem változik. A redundancia tehát egy eszköz lehet a kezünkben egy adott *formula összetettségének esetleges csökkentésére*. Pontosan ilyesfajta egyszerűsítést fogunk leírni a 3.4.2. fejezetben a klózokkal kapcsolatban.

Kalkulusok kapcsán a redundancia fogalmát szokás felhasználni a *kalkulusok levezetési szabályai alkalmazhatóságának megszorítására*; egyrészt annak érdekében, hogy tárat és időt takarítsunk meg, másrészt azért, hogy (el)kerüljük a végtelen levezetéseket. Mivel a kielégíthetetlenség kérdése elsőrendű logikában eldönthetetlen probléma, nem kerülhetjük el minden esetben a végtelen levezetéseket – ám a lehetőségüket nagymértékben csökkenthetjük. Általános igazságként elmondható, hogy egy kalkulus gyakorlati alkalmazhatóságának előfeltétele, hogy társuljon hozzá redundanciafogalom. A használt redundanciafogalom kalkulusonként más és más lehet, sőt egy adott kalkulus esetén akár többféle redundanciafogalom is megadható – attól függően, hogy pontosan milyen célra is szánjuk. A 3.4.1. fejezetben a hipertablóhoz definiáljuk a redundancia fogalmát, illetve a 3.4.3. fejezetben arról értekezünk, hogy a multi-hipertabló esetén hogyan tehetnénk meg ugyanezt. Mindehhez a fenti, 3.4.1. definíció által leírt redundanciafogalmat fogjuk alapul venni, mely formulákra mondatott ki; természetesen tablók esetén az adott tabló által reprezentált formulára fogjuk alkalmazni.

3.4.1. Redundancia hipertablóban

A (tisztított) hipertabló esetén használható redundanciafogalom meglehetősen triviális módon adódik – Baumgartner cikkében [3] is megtaláljuk „elégséges redundancia kritérium"[16] néven. Megjegyezzük, hogy Baumgartner előbb egy általánosabb redundanciafogalmat definiál, majd később valamelyest gyengít rajta, hiszen a hipertabló kiterjesztési szabályának alkalmazása a fentebb említett elégséges kritériummal is kívánt mértékben megszorítható.

A hipertabló 2.3.5. definíciójából (és azon belül is a tisztító helyettesítések használatából) következik, hogy egy hipertabló *csúcsai változóidegenek*. Ennek következménye, hogy egy $\mathcal{T}$ hipertabló által reprezentált formulában (lásd: 2.1.3. definíció) a *kvantoros előtagok bevihetők* a $\mathcal{T}$ literáljai elé, azaz a $\mathcal{T}$ által reprezentált formula a következő alakban írható fel:

$$\bigvee_{\mathcal{B}\in\mathcal{T}} \bigwedge_{L\in\mathcal{B}} \forall L$$

Vagyis $\mathcal{T}$ minden ága mint saját lezárt literáljainak konjunkciója írható fel. Ez az oka, hogy a következőkben konjunkcióbeli redundanciával fogunk foglalkozni.

3.4.2. Tétel.

Adottak az A és a $B_1 \wedge \ldots \wedge B_k$ formulák. Ha valamely B_i esetén létezik olyan σ helyettesítés, hogy $A = B_i\sigma$, akkor A redundáns $B_1 \wedge \ldots \wedge B_k$-ra és a $\wedge$-ra nézve.

[16] angol: *sufficient redundancy criterion*

BIZONYÍTÁS.

A 3.4.1. definíció azt mondja ki, hogy A akkor redundáns $B_1 \wedge \ldots \wedge B_k$-ra és a $\wedge$-ra nézve, ha $B_1 \wedge \ldots \wedge B_k$ és $B_1 \wedge \ldots \wedge B_k \wedge A$ logikailag ekvivalensek. Ezen ekvivalenciát kell bebizonyítanunk. Azaz azt, hogy bármely $\mathcal{M}$ modellben

$$\mathcal{M} \models B_1 \wedge \ldots \wedge B_k \text{ akkor és csak akkor, ha } \mathcal{M} \models B_1 \wedge \ldots \wedge B_k \wedge A.$$

Triviálisan teljesül, hogy

$$\text{ha } \mathcal{M} \models B_1 \wedge \ldots \wedge B_k \wedge A, \text{ akkor } \mathcal{M} \models B_1 \wedge \ldots \wedge B_k.$$

Az viszont nem annyira magától értetődő, hogy

$$\text{ha } \mathcal{M} \models B_1 \wedge \ldots \wedge B_k, \text{ akkor } \mathcal{M} \models B_1 \wedge \ldots \wedge B_k \wedge A.$$

Ennek bizonyításához felhasználjuk, hogy

$$\mathcal{M} \models B_1 \wedge \ldots \wedge B_k \text{ akkor és csak akkor, ha } \mathcal{M} \models B_j \text{ minden } B_j\text{-re.}$$

Azaz $\mathcal{M} \models B_i$ is teljesül. Ebből pedig – mivel $A = B_i\sigma$ – következik, hogy $\mathcal{M} \models A$. Tehát

$$\mathcal{M} \models B_1 \wedge \ldots \wedge B_k \wedge A \quad . \qquad \square$$

A fenti tétel tehát valami olyasmit mond ki, hogy ha egy konjunkció egy A tagja egy másik B_i tag példánya, akkor A elhagyható a konjunkcióból. A 3.4.2. tétel gyakorlati haszna a következőben fog megnyilvánulni: mivel a hipertabló egy $\mathcal{B}$ ága saját lezárt literáljai konjunkciójának tekinthető, így egy olyan L literált, mely példánya egy a $\mathcal{B}$-n már szereplő literálnak, felesleges $\mathcal{B}$-hez csatolnunk.

3.4.3. DEFINÍCIÓ. (REDUNDANCIA HIPERTABLÓ ÁGÁRA NÉZVE)

Egy D *klózt redundánsnak* mondunk egy hipertabló $\mathcal{B}$ *ágára* nézve, ha valamely $L_1 \in D$ és $L_2 \in \mathcal{B}$ esetén van olyan σ helyettesítés, hogy $L_1 = \widehat{L_2}\sigma$ [17]. $\square$

Ezen definíciót felhasználva tudjuk végül kimondani a hipertabló redundancia kritériumát, vagyis a hipertabló kiterjesztési szabálya alkalmazásának redundancián alapuló megszorítását.

HIPERTABLÓ REDUNDANCIA KRITÉRIUMA:

A hipertabló 2.3.5.(2) *kiterjesztési szabálya* csak akkor *alkalmazható*, ha a kiszámolt $D\sigma'\pi$ klóz nem redundáns a kiterjeszteni kívánt $\mathcal{B}$ ágra nézve. $\square$

[17] Megjegyezzük, hogy míg Baumgartner eredeti konstrukciója esetén az L_2 *új példányának* előállítása feltétlenül szükséges a redundanciavizsgálat folyamán, a hipertabló 2.3.5. definíciója esetén nem feltétlenül az, hiszen a rezolválandó klózoknak a 2.3.5.(2)(b,c)-ben új példányait állítjuk elő, vagyis L_1 és L_2 mindig változóidegenek lesznek.

3.4.4. Megjegyzés.

Egy D klóz egy hipertabló $\mathcal{B}$ ágán való redundánságának vizsgálata könnyen elvégezhető MGU képzésével (az A.1.1. unifikációs algoritmussal). Azt kell megvizsgálni, hogy léteznek-e olyan azonosan negált $L_1 \in D$ és $L_2 \in \mathcal{B}$, hogy $\sigma = \mathcal{U}(\widehat{\underline{L}}_2, \underline{L}_1)$ létezik és $L_1 = \widehat{L}_2\sigma$.

Megjegyezzük, hogy σ kiszámításánál az $\mathcal{U}()$ függvény argumentumainak, azaz $\widehat{\underline{L}}_2$-nek és $\underline{L}_1$-nek a sorrendjét rögzítjük. Ennek oka, hogy az A.1.1. algoritmus úgy lett megkonstruálva, hogy annak (7-8) soraiban elsősorban az első argumentum szabad változóit helyettesítsük be. Most – mivel azt akarjuk eldönteni, hogy $\widehat{\underline{L}}_2$-nek $\underline{L}_1$ példánya-e – elsősorban $\widehat{\underline{L}}_2$ szabad változóit kell behelyettesítenünk. □

3.4.2. Redundancián alapuló klózegyszerűsítés

A 3.3.2. fejezetben megadtunk egy módszert klózhalmaz egy bizonyos fokú egyszerűsítésére. Ott a klózhalmazból eliminálták az érvényes klózokat. A következőkben egy másféle egyszerűsítést fogunk leírni, mely a redundancián alapul [18]. Egy klóz literáljai közül eliminálni fogjuk azokat, melyek feleslegesek az adott klózban, azaz elhagyásukkal az adott klóz igazságértéke az egyes modellekben nem változik. Ehhez egy a 3.4.2. tételhez hasonló tételt írunk le, mely – mivel a klóz egy diszjunkciós formula – a diszjunkcióra vonatkozó redundanciáról fogalmaz meg állítást.

3.4.5. Tétel.

Adottak az A és a $B_1 \vee \ldots \vee B_k$ formulák. Ha

$$\mathcal{FV}(A) \cap \mathcal{FV}(B_1 \vee \ldots \vee B_k) \quad = \quad \emptyset$$

és valamely B_i esetén létezik olyan σ helyettesítés, hogy $A\sigma = B_i$, akkor A redundáns $B_1 \vee \ldots \vee B_k$-ra és a $\vee$-ra nézve.

Bizonyítás.

A 3.4.1. definíció azt mondja ki, hogy A akkor redundáns $B_1 \vee \ldots \vee B_k$-ra és a $\vee$-ra nézve, ha $B_1 \vee \ldots \vee B_k$ és $B_1 \vee \ldots \vee B_k \vee A$ logikailag ekvivalensek. Azaz azt, hogy bármely $\mathcal{M}$ modellben

$$\mathcal{M} \models B_1 \vee \ldots \vee B_k \text{ akkor és csak akkor, ha } \mathcal{M} \models B_1 \vee \ldots \vee B_k \vee A.$$

Triviálisan teljesül, hogy

$$\text{ha } \mathcal{M} \models B_1 \vee \ldots \vee B_k, \text{ akkor } \mathcal{M} \models B_1 \vee \ldots \vee B_k \vee A.$$

Az viszont nem annyira magától értetődő, hogy

$$\text{ha } \mathcal{M} \models B_1 \vee \ldots \vee B_k \vee A, \text{ akkor } \mathcal{M} \models B_1 \vee \ldots \vee B_k.$$

Ennek bizonyításához felhasználjuk azt a kikötést, hogy

$$\mathcal{FV}(A) \cap \mathcal{FV}(B_1 \vee \ldots \vee B_k) \quad = \quad \emptyset \quad , \tag{3.1}$$

melynek alapján teljesül, hogy

$$\forall(B_1 \vee \ldots \vee B_k \vee A) \quad \sim \quad \forall(B_1 \vee \ldots \vee B_k) \vee \forall A \quad . \tag{3.2}$$

Azt kell megmutatni, hogy $\mathcal{M} \models A$ esetén $\mathcal{M} \models B_1 \vee \ldots \vee B_k$ is teljesül. Ha $\mathcal{M} \models A$, akkor – mivel $A\sigma = B_i$ – teljesül, hogy $\mathcal{M} \models B_i$. Vagyis

$$\mathcal{M} \models B_1 \vee \ldots \vee B_k \quad . \qquad \square$$

Mint látható – ha a 3.4.2. tételt összehasonlítjuk a fenti tétellel – a redundancia fogalma diszjunkció esetén kiegészül egy plusz feltétellel: A és $B_1 \vee \ldots \vee B_k$ *változóidegenek*. A tétel értelmében egy diszjunkció egy A tagja elhagyható a diszjunkcióból, ha A-nak egy másik B_i tag példánya, feltéve hogy A nem tartalmaz közös szabad változót a diszjunkció A-n kívüli egyik tagjával sem.

3.4.6. Megjegyzés.

Konjunkcióra vonatkozó redundancia esetén nem volt szükség kikötni egy (3.1)-hez hasonló feltételt, hiszen

$$\forall(B_1 \wedge \ldots \wedge B_k \wedge A) \quad \sim \quad \forall(B_1 \wedge \ldots \wedge B_k) \wedge \forall A$$

tetszőleges A és $B_1 \wedge \ldots \wedge B_k$ esetén. A 3.4.2. tétel bizonyításában használtuk ezt az összefüggést, bár explicite nem említettük.

Diszjunkcióra vonatkozó redundancia esetén viszont ki kell kötni a (3.1)-et, hiszen csak ebben az esetben igaz a (3.2). $\square$

A következő definícióban leírjuk, hogy egy D klózt mikor tekintünk redundánsnak egy C klózra nézve: ha C és D változóidegenek, valamint C-nek van olyan részklóza, mely példánya a D-nek.

3.4.7. Definíció. (Redundancia klózra nézve)

Egy D *klóz redundáns* a C *klózra* nézve, ha $\mathcal{FV}\big(\langle C \rangle\big) \cap \mathcal{FV}\big(\langle D \rangle\big) = \emptyset$ és létezik olyan σ helyettesítés, hogy $\langle D \rangle \sigma \subseteq C$. $\square$

Egy klózt önmagában pedig akkor fogunk redundánsnak mondani, ha *van olyan részklóza*, mely redundáns a klóz maradék részére nézve.

3.4.8. Definíció. (Redundáns klóz)

Redundáns klóznak egy olyan C klózt nevezünk, melyre létezik olyan $D \subset C$, hogy D redundáns a $C \backslash D$ klózra nézve. $\square$

Egyszerűsítésünk lényege az lesz, hogy egy adott klóznak megpróbáljuk a redundanciáját megszüntetni – persze csak akkor, ha az eredendően fennáll. Egy klóz redundanciájának megszüntetése a *redundanciát okozó részklózainak eliminálásával* történhet, miközben biztosítjuk, hogy a klóz jelentése ne változhasson meg, vagyis az eliminálás eredményeként kapott klóz az eredeti klózzal logikailag ekvivalens legyen.

3.4.9. Definíció. (Klóz irredundáns variánsa)

Egy C *klóz irredundáns variánsa* egy olyan $C' \subseteq C$ klóz, ahol C' nem redundáns és $C' \sim C$. □

3.4.10. Tétel.

(1) *Bármely C klóznak létezik irredundáns variánsa.*

(2) *Ha C-hez több ilyen variáns létezik, akkor azok egymás átnevezettjei.*

Bizonyítás.

Jelölje $r(A)$ az A klózban azon *részklózok számát*, melyek redundánsak a klóz maradék részére nézve. Azaz

$$r(A) = \Big|\{B \mid B \subset A \text{ és } B \text{ redundáns } A\backslash B\text{-re nézve}\}\Big| \quad .$$

Megjegyezzük, hogy a 3.4.7. definíció értelmében ezen részklózoknak nem lehet közös változójuk, vagyis ezen részklózok nem fedik át egymást, diszjunktak (páronként vett metszetük üres). A tétel (1)-es és (2)-es pontját bizonyítjuk az alábbiakban:

(1) A bizonyítás $r(C)$-re nézve induktív.

   (a) Ha $r(C) = 0$, akkor C önmaga irredundáns variánsa, mivel $C \subseteq C$, $C \sim C$ és C nem redundáns.

   (b) Az induktív feltevésünk az, hogy ha $r(C) \geq 1$, akkor minden olyan A klózhoz létezik irredundáns variáns, mely esetén $r(A) < r(C)$.

   (c) Legyen $r(C) \geq 1$. Bizonyítsuk be, hogy C-nek is van irredundáns variánsa. Mivel $r(C) \geq 1$, van olyan $D \subset C$, hogy D redundáns a $C' = C\backslash D$-re nézve. Mivel $r(C') < r(C)$, létezik C'-nek valamely C'' irredundáns variánsa, azaz $C'' \subseteq C'$, $C'' \sim C'$ és C'' nem redundáns.

   A 3.4.5. tétel és a 3.4.1. definíció alapján $C' \sim C$. Mivel $C'' \subseteq C' \subseteq C$, $C'' \sim C' \sim C$ és C'' nem redundáns, így C'' irredundáns variánsa C-nek.

(2) Az (1)-es pont bizonyítása egyúttal induktív módszert ad az irredundáns variánsok előállítására, mely során C redundanciát okozó részklózait egyenként eliminálju k, míg végül nem redundáns klózt kapunk. Most tulajdonképpen annak bebizonyítása a cél, hogy a részklózok eliminálásának sorrendje lényegtelen, vagyis az eredményül kapott klózok egymás átnevezettjei.

   Elegendő azt bizonyítanunk, hogy *két részklóz különböző sorrendben* való eliminálása vagy ugyanazon klózt eredményezi, vagy pedig két olyan klózt, melyek

egymás átnevezettjei. Azaz az $r(C) \geq 2$ eset áll most bizonyításunk fókuszában. Jelölje $D_1 \subset C$ és $D_2 \subset C\backslash D_1$ a két eliminálandó részklózt, azaz feltesszük, hogy D_i redundáns a $C\backslash D_i$-re nézve (ahol $i \in \{1,2\}$). A 3.4.7. definíció alapján $\mathcal{FV}\big(\langle D_i\rangle\big) \cap \mathcal{FV}\big(\langle C\backslash D_i\rangle\big) = \emptyset$, valamint létezik olyan σ_i helyettesítés és olyan $\widetilde{D}_i \subseteq C\backslash D_i$, hogy $\langle D_i\rangle\sigma_i = \langle \widetilde{D}_i\rangle$. Az általánosság megszorítása nélkül három különböző eset lehetséges:

(a) ha $\widetilde{D}_1 \neq D_2$ és $\widetilde{D}_2 \neq D_1$: mind D_1, mind D_2 eliminálásra kerül, azaz az egyetlen klóz, amit eredményül kaphatunk: $C' = C\backslash(D_1 \cup D_2)$.

(b) ha $\widetilde{D}_1 = D_2$ és $\widetilde{D}_2 \neq D_1$: $\langle D_1\rangle\sigma_1 = \langle D_2\rangle$. Mivel $\langle D_2\rangle\sigma_2 = \langle \widetilde{D}_2\rangle$, így

$$\langle D_1\rangle\sigma_1\sigma_2 \quad = \quad \langle \widetilde{D}_2\rangle \quad .$$

Így eljutunk az előző esethez, azaz itt is C'-t kapjuk eredményül.

(c) ha $\widetilde{D}_1 = D_2$ és $\widetilde{D}_2 = D_1$: $\langle D_1\rangle\sigma_1 = \langle D_2\rangle$ és $\langle D_2\rangle\sigma_2 = \langle D_1\rangle$. Azaz

$$\langle D_1\rangle\sigma_1\sigma_2 \quad = \quad \langle D_1\rangle \quad .$$

Ez pontosan akkor teljesül, ha $\sigma_1\sigma_2 = \epsilon$, ami pedig csak akkor lehetséges, ha mind σ_1, mind σ_2 átnevezések. Azaz a D_1, illetve a D_2 eliminálásának eredményéül kapható két klóz: $C\backslash D_1 = C' \cup D_2$ és $C\backslash D_2 = C' \cup D_1$ egymás átnevezettjei. □

E tétel gyakorlati haszna nem olyan jelentős, mint a 3.4.2. tételé, hiszen a 3.4.7. definícióban a redundanciára adott *feltételek meglehetősen szigorúak*. Ezen szigorításnak pedig az az eredménye, hogy nem feltétlenül különálló literálokat, hanem változóik alapján összetartozó részklózokat kell egy-egy lépésben a klózból eliminálnunk. Ez algoritmuselméleti szempontból nagyobb bonyolultságot, gyakorlati szempontból pedig *költségesebb eljárást* jelent. Az A.2. függelék A.2.2. algoritmusában klóz irredundáns variánsának előállítását realizáljuk.

A multi-hipertablóban két ponton látunk alkalmazhatónak irredundáns variánsokon alapuló egyszerűsítést:

(1) A $\mathcal{C}$ *input klózhalmaz egyszerűsítésére*: állítsuk elő a

$$\mathcal{C}' = \{C' \mid C' \text{ a } C \in \mathcal{C} \text{ irredundáns variánsa}\}$$

klózhalmazt, majd nem $\mathcal{C}$, hanem $\mathcal{C}'$ kielégíthetetlenségét kell bizonyítani.

(2) Valamely $[E,\sigma]$ *kiterjesztés alkalmazásakor*: állítsuk elő $\langle E\rangle\sigma$ irredundáns variánsát, majd azt csatoljuk a tabló ágához.

A (2) egyszerűsítés kapcsán érdemes előretekinteni. A 3.5. fejezetben, mikor majd a *heurisztikus* tételbizonyítás lehetőségeivel foglalkozunk, egy ághoz előállított kiterjesztések közül próbáljuk majd bizonyos szempontok alapján kiválasztani és alkalmazni a legelőnyösebbet. Ezen vizsgálat egy $[E,\sigma]$ kiterjesztés esetén nem az $\langle E\rangle\sigma$ klózon,

hanem az E-n és a σ-n külön-külön végez vizsgálatokat[18]. Mielőtt egy kiterjesztést a heurisztikus függvény segítségével más kiterjesztésekkel összevetnénk, mindenképpen érdemes lenne a kiterjesztést irredundáns formára hozni; és ezúttal nem az $\langle E\rangle\sigma$ klóz irredundáns variánsának előállításáról beszélünk, hanem egy olyan $[E', \sigma']$ kiterjesztésről, hogy $\langle E'\rangle\sigma'$ irredundáns variánsa $\langle E\rangle\sigma$-nak.

3.4.11. Definíció. (Kiterjesztés irredundáns variánsa)

Egy $\mathcal{T}$ multi-hipertabló $[E, \sigma]$ *kiterjesztésének irredundáns variánsa* alatt értünk egy olyan $[E', \sigma']$ kiterjesztést, hogy

(1) $E' \subseteq E$,

(2) $\sigma' = \{x/t \in \sigma \mid x \in \mathcal{FV}(\langle E'\rangle) \cup \mathcal{FV}(\mathcal{T})\}$ és

(3) $\langle E'\rangle\sigma'$ az $\langle E\rangle\sigma$-nak irredundáns variánsa. □

A definícióban láthatjuk, hogy a fentebb már említett (3) feltételen kívül az E'-re és a σ'-re egy-egy további feltételt mondunk ki. Az (1) arról szól, hogy az E literáljai közül fogunk egyeseket eliminálni. A (2) pedig arról, hogy a σ értelmezési tartományából eliminálunk minden olyan x-et, mely csak az E-ből eliminált literálokban fordul elő (azaz se E'-ben, se $\mathcal{T}$-ben). Az A.2.3. algoritmus kiterjesztés irredundáns variánsának előállítását végzi el.

3.4.3. Redundancia multi-hipertablóban

Jelen fejezetben azzal foglalkozunk, hogy a multi-hipertablóhoz miképpen lehet redundanciafogalmat konstruálni. Ez a redundanciafogalom is a redundancia 3.4.1. definíciójára épül, hasonlóan a hipertabló 3.4.1. fejezetben definiált redundanciafogalmához. Azaz egy tabló kiterjesztésekor pusztán a *aktuális tabló és a tablóhoz csatolandó klóz alapján* szeretnénk azt eldönteni, hogy vajon a csatolás felesleges-e – vagyis a csatolandó klóz redundáns-e a kiterjesztendő ágra nézve. Meg fogjuk mutatni, hogy ilyen jellegű redundanciafogalmat *rigid literálos* (azaz rigid változókat tartalmazó és csak literálokkal címkézett) tablók esetén – amilyen többek között a 2.1.11. definícióban ismertetett klóztabló, a 2.3.5. definícióban megadott hipertabló, a 2.3.13. definícióban leírt rigid hipertabló, és persze a multi-hipertabló is – *általánosságban* nem lehet megadni. Mindenesetre kivételek akadnak, mint azt a 3.4.1. fejezet a hipertabló kapcsán igazolja is.

A fentieknek eleget tevő általános redundanciafogalom létezését a következő példán keresztül cáfoljuk.

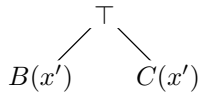
[18] Tulajdonképpen ez adta az egyik motivációt a kiterjesztések fogalmának (3.1.1. definíció) megalkotásához.

3.4.12. Példa.

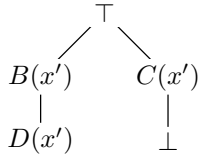
Legyen az input klózhalmaz a következő:

$$\mathcal{C} \;=\; \left\{ \begin{array}{c} A(x)\ , \\ \neg A(y) \vee B(y) \vee C(y)\ , \\ \neg B(z) \vee D(z)\ , \\ \neg C(w)\ , \\ \neg D(c) \vee \neg D(d) \end{array} \right\}\ ,$$

ahol c és d konstansszimbólumok. Ezen klózhalmaz kielégíthetetlen, amit be is láthatunk azzal, hogy $\mathcal{C}$-hez zárt multi-hipertablót konstruálunk. Az első kiterjesztési lépés után (az 1. és 2. input klózok új példányait rezolválva) a következő tablót kaphatjuk:



A $B(x')$-vel a 3. input klóz, illetve a $C(x')$-vel a 4. input klóz egy-egy új példányát rezolválva (két újabb kiterjesztési lépés eredményeként) kapjuk a következő tablót:



A bal oldali ág lezárását célul kitűzve most a következő irányokban próbálhatunk meg továbblépni. Látni fogjuk, hogy minden járható irány korábbi lépések vég nélküli ismétléséhez vezethet.

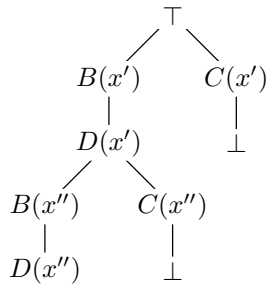
(1) Az ág $B(x')$ literálját és a $\neg B(z') \vee D(z')$ input klózpéldányt rezolváljuk. Ennek eredménye kétféle lehet, attól függően, hogy a rezolválásnál előállított MGU vajon a z'/x' vagy az x'/z' helyettesítés. Az első esetben azághoz egy újabb $D(x')$ literált csatolnánk, ami nyilvánvalóan felesleges (redundáns). A második esetben két dolgot tennénk: előbb az ághoz csatolnánk egy $D(z')$ literált, majd elvégeznénk az x'/z' helyettesítést a tablón – aminek az eredménye teljesen az első esetben kapott tablóval ekvivalens tabló lenne.

(2) Az ág másik literálja, a $D(x')$ semelyik input klózzal sem (hiper)rezolválható. Tehát a $\neg D(c) \vee \neg D(d)$-vel sem, hiszen az

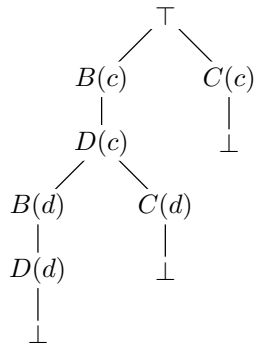
$$\mathcal{U}\Big(\big(D(x'), D(c)\big), \big(D(x'), D(d)\big)\Big)$$

MGU nem létezik.

(3) Egyetlen lehetőségünk marad hát a továbblépésre: az 1. és 2. input klózok újabb példányainak rezolválása, aminek eredményeképp a $B(x'') \vee C(x'')$ klózt kell a tablóhoz csatolni. Ezen lépést nyilvánvalóan csak a fentebbről már ismerős két lépés követheti: $B(x'')$-t kell a 3. input klóz, illetve a $C(x'')$-t a 4. input klóz egy-egy új példányával rezolválni. Az eredményül kapott tabló:



A levezetés ezen a ponton fejezhető be: a $D(x')$ és $D(x'')$ literálokat rezolváljuk a $\neg D(c) \vee \neg D(d)$ input klózzal, aminek eredményeként a következő zárt tablót kapjuk:



□

Vegyük észre, hogy az utolsónak elvégzett lépés nem az egyedül lehetséges lépés – ugyanis folytathattuk volna a bal oldali ágnak a $B(x') \vee C(x')$ és a $D(x')$ újabb és újabb példányaival való kiterjesztését, akár a végtelenségig. A fenti példa tanulsága tehát az, hogy *nem tudjuk eldönteni* csupán az aktuális tabló és a csatolandó klóz alapján, mikor lehet a kiterjesztéseknek ezt a végtelen láncolatát megszakítani, azaz mikor lehet az aktuálisan csatolandó klózt redundánssá nyilvánítani a tablóra (illetve annak adott ágára) nézve.

Vonjunk le még néhány következtetést a fenti példából – amely egyébként nagyban hasonlít az ágklózok felhasználására adott 2.3.9. példához. Ám míg ott egy *ágklóz* másolatát csatoltuk a lezárandó ághoz, a mostani példában a (3)-ban a tablónak egy komplexebb részét másoljuk le: egy olyan *résztablót* (gyökere nélkül), mely a tabló egyéb részével változóidegen[19]. A 2.3.9. példában olyan speciális tabló szerepelt, melyben a másolandó ágklóz *változóidegen* volt a tabló egyéb részével – ezért ott ezen ágklóz új példányát helyesen csatolhattuk az adott ághoz. Ám mint a 2.3.15. tétel bizonyításában rámutattunk, az ágklóz-másolatok használata általánosságban nem helyes – ezt jól demonstrálja a fenti példa is.

Általánosságban és informálisan, rigid literálos tablókkal kapcsolatosan a következőket mondhatjuk:

[19] Míg a 2.3.9. példában az ágklózok másolását a rigid hipertabló külön levezetési szabálya végzi el, addig a 3.4.12. példában a (gyökér nélküli) résztabló másolására nincs direkt eszközünk. A másolást a multi-hipertabló kiterjesztési szabályának többszöri alkalmazásával lehet kivitelezni.

(1) A lemásolandó résztablók körülhatárolása rendkívül bonyolult lehet.

(2) Kérdéses, hogy a résztabló másolatát hogyan használjuk fel annak eldöntésére, hogy a csatolandó klóz redundáns-e a teljes tablóra (illetve annak adott ágra) nézve. Ráadásul a résztablóban való redundánsság eldöntése a teljes tablóban való redundánsság eldöntésével (ami az eredeti probléma) *ekvivalens probléma.*

Ha létezik is olyan algoritmus, mellyel mindez kivitelezhető és eldönthető, valószínűleg annyira költséges, hogy a gyakorlatban használhatatlan. Megmutatjuk viszont, hogy a kilátások nem annyira borúsak, mint az első pillantásra tűnik. A 3.4.13. tételben meg fogjuk mutatni, hogy

(1) résztablók helyett elegendő csupán egy-egy (a tablóból kinyerhető) *klózt* lemásolnunk (azaz új példányát előállítanunk), és

(2) ezen klózmásolatot összevetve a csatolandó klózzal nagyon egyszerűen eldönthető, hogy ez utóbbi redundáns-e a kiterjesztendő ágra nézve.

Ehhez tekintsünk bármely $\mathcal{T}$ rigid literálos tablót a 3.3. ábrán látható formában. A jelölések magyarázata:

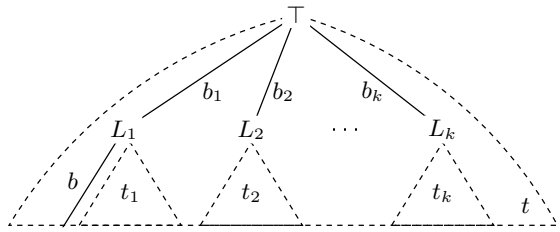
- Az $L_1, \ldots, L_k$ ($k \geq 1$) a $\mathcal{T}$ olyan tetszőleges literáljai, melyek szigorúan *különböző ágakon* helyezkednek el (vagyis $\mathcal{T}$ bármely ága az L_i-k közül legfeljebb egyet tartalmaz).
- Minden b_i ($1 \leq i \leq k$) a $\mathcal{T}$ gyökeréből az L_i-be vezető részág.
- Minden t_i – ahol $2 \leq i \leq k$ – a $\mathcal{T}$ azon maximális résztablója, melynek a gyökere L_i.
- A $\mathcal{T}$ L_1-gyökerű maximális résztablójában egy ágat megkülönböztetünk, ezt jelöljük b-vel. A résztabló egyéb ágainak összességét t_1-gyel jelöljük.
- A t pedig a $\mathcal{T}$ azon ágainak összességét jelöli, mely ágak közül egy sem halad át semelyik L_i-n ($1 \leq i \leq k$).

A $b_1, \ldots, b_k$ és a b által reprezentált formulák általánosságban konjunkciók, melyekre szintén $b_1, \ldots, b_k$-val és b-vel fogunk hivatkozni. Hasonlóan, a $t_1, \ldots, t_k$ és a t által reprezentált formulák általánosságban diszjunkciók, melyekre $t_1, \ldots, t_k$-val és t-vel fogunk hivatkozni.

3.4.13. Tétel.

Legyen $\mathcal{T}$ a 3.3. ábrán látható alakú rigid literálos tabló, ahol

- *jelölje $\mathcal{B}$ a $b_1 \cup \{L_1\} \cup b$ ágat,*
- *jelölje C az $L_1 \vee L_2 \vee \ldots \vee L_k$ klózt, és legyen $\mathcal{FV}\big(\langle C \rangle\big) \cap \mathcal{FV}(t) = \emptyset$.*



3.3. ábra. Rigid literálos tabló

Ekkor:

(1) *Ha* $\widehat{C}$ *a* C *új példánya, akkor*

$$F(\mathcal{T}) \quad \sim \quad F\big(\mathcal{T} +^{\mathcal{B}} \widehat{C}\big) \quad .$$

(2) *Ha* D *olyan klóz, hogy* $\langle \widehat{C} \rangle \sigma \subseteq D$ *valamely* σ *helyettesítésre, akkor*

$$F(\mathcal{T}) \quad \sim \quad F\big(\mathcal{T} +^{\mathcal{B}} D\big) \quad .$$

Bizonyítás.

Az $F(\mathcal{T})$ felírható a következő formában:

$$t \ \vee \ \big(b_1 \wedge L_1 \wedge (b \vee t_1)\big) \ \vee \ \bigvee_{i=2}^{k} (b_i \wedge L_i \wedge t_i)$$

$$\wr$$

$$t \ \vee \ (b_1 \wedge L_1 \wedge b) \ \vee \ (b_1 \wedge L_1 \wedge t_1) \ \vee \ \bigvee_{i=2}^{k} (b_i \wedge L_i \wedge t_i)$$

$$\wr$$

$$t \ \vee \ (b_1 \wedge L_1 \wedge b) \ \vee \ \bigvee_{i=1}^{k} (b_i \wedge L_i \wedge t_i) \tag{3.3}$$

A tétel (1) és (2) pontjának bizonyítása következik az alábbiakban:

(1) Az $F\big(\mathcal{T} +^{\mathcal{B}} \widehat{C}\big)$ a következő formában írható fel:

$$t \ \vee \ \big(b_1 \wedge L_1 \wedge b \wedge \langle \widehat{C} \rangle\big) \ \vee \ \bigvee_{i=1}^{k} (b_i \wedge L_i \wedge t_i) \quad ,$$

melyet – mivel $\widehat{C}$ *új* példány (azaz változóidegen a $\mathcal{T}$ összes formulájával) – a következő ekvivalens formában fogunk használni:

$$t \ \vee \ (b_1 \wedge L_1 \wedge b \wedge \forall\widehat{C}) \ \vee \ \bigvee_{i=1}^{k} (b_i \wedge L_i \wedge t_i) \quad . \tag{3.4}$$

Be kell látni, hogy bármely $\mathcal{M}$ modellben:

(a) ha $\mathcal{M} \models (3.4)$, akkor $\mathcal{M} \models (3.3)$: ez triviálisan teljesül.

(b) ha $\mathcal{M} \models (3.3)$, akkor $\mathcal{M} \models (3.4)$:
Alakítsuk tovább a (3.3) formulát. Először annak

$$(b_1 \wedge L_1 \wedge b) \vee \bigvee_{i=1}^{k} (b_i \wedge L_i \wedge t_i)$$

részformuláján végezzük el iteratíve a disztributivitásra vonatkozó következő ekvivalens átalakítást:

$$X \vee (Y \wedge Z) \quad \sim \quad (X \vee Y) \wedge (X \vee Z) \quad .$$

Így egy sok elemből álló konjunkciót kapunk, melynek leglényegesebb eleme számunkra a lentebb is feltüntetett $\bigvee_{i=1}^{k} L_i$, mely nem más, mint maga C.
Mivel kikötöttük, hogy $\mathcal{M} \models (3.3)$, azaz hogy $\mathcal{M} \models \forall(3.3)$, írjuk fel a $\forall(3.3)$ formulát a következő formában:

$$\forall \left[t \vee \left(\Big(\bigvee_{i=1}^{k} b_i \Big) \ \wedge \ldots \wedge \ \Big(\bigvee_{i=1}^{k} L_i \Big) \ \wedge \ldots \wedge \ \Big(b \vee \bigvee_{i=1}^{k} t_i \Big) \right) \right]$$

$$\wr$$

$$\forall \left[\Big(t \vee \bigvee_{i=1}^{k} b_i \Big) \ \wedge \ldots \wedge \ \Big(t \vee \bigvee_{i=1}^{k} L_i \Big) \ \wedge \ldots \wedge \ \Big(t \vee b \vee \bigvee_{i=1}^{k} t_i \Big) \right]$$

$$\wr$$

$$\forall \left[t \vee \bigvee_{i=1}^{k} b_i \right] \ \wedge \ldots \wedge \ \forall \left[t \vee \bigvee_{i=1}^{k} L_i \right] \ \wedge \ldots \wedge \ \forall \left[t \vee b \vee \bigvee_{i=1}^{k} t_i \right]$$

$$\wr$$

$$\ldots \wedge \ \forall \big(t \vee \langle C \rangle \big) \ \wedge \ldots$$

$$\wr \quad \longleftarrow \text{mivel } \mathcal{FV}\big(\langle C \rangle\big) \cap \mathcal{FV}(t) = \emptyset$$

$$\ldots \wedge \ (\forall t \vee \forall C) \ \wedge \ldots$$

Ezért bármely olyan $\mathcal{M}$ modellben, ahol $\mathcal{M} \models (3.3)$:

$$\mathcal{M} \models \forall t \vee \forall C \quad .$$

Ez alapján két eset lehetséges:

(i) $\mathcal{M} \models \forall t$: ekkor az $\mathcal{M} \models (3.4)$ triviálisan teljesül.

(ii) $\mathcal{M} \models \forall C$:
Be kell látnunk, hogy bármely ϱ változókiértékelés esetén ha $\mathcal{M} \models (3.3)\varrho$, akkor $\mathcal{M} \models (3.4)\varrho$.
Elegendő azon ϱ változókiértékelésekre vizsgálódnunk, melyekre

$$\mathcal{M} \quad \models \quad (b_1 \wedge L_1 \wedge b)\varrho$$

(egyéb változókiértékelések esetén a bizonyítás triviális). Ezen összefüggés miatt – mivel $\mathcal{M} \models \forall C$, és ezért $\mathcal{M} \models \forall \widehat{C}$ – a következő is teljesül:

$$\mathcal{M} \quad \models \quad (b_1 \wedge L_1 \wedge b)\varrho \ \wedge \ \forall \widehat{C} \quad .$$

Ennek triviális következménye, hogy $\mathcal{M} \models (3.4)\varrho$.

(2) Az $F\big(\mathcal{T} +^{\mathcal{B}} D\big)$ formula a következő formában írható fel:

$$t \ \vee \ \big(b_1 \wedge L_1 \wedge b \wedge \langle D \rangle\big) \ \vee \ \bigvee_{i=1}^{k} (b_i \wedge L_i \wedge t_i) \quad . \tag{3.5}$$

Be kell látni, hogy bármely $\mathcal{M}$ modellben:

(a) ha $\mathcal{M} \models (3.5)$, akkor $\mathcal{M} \models (3.3)$: ez triviálisan teljesül.

(b) ha $\mathcal{M} \models (3.3)$, akkor $\mathcal{M} \models (3.5)$:
Az (1) pont alapján tudjuk, hogy $(3.3) \sim (3.4)$. Meg fogjuk mutatni, hogy bármely $\mathcal{M}$ modell és bármely ϱ változókiértékelés esetén ha $\mathcal{M} \models (3.4)\varrho$, akkor $\mathcal{M} \models (3.5)\varrho$.
Elegendő azon $\mathcal{M}$ modellekre és azon ϱ változókiértékelésekre vizsgálódnunk, melyekre

$$\mathcal{M} \quad \models \quad (b_1 \wedge L_1 \wedge b \wedge \forall \widehat{C})\varrho$$

(egyéb modellek és változókiértékelések esetén a bizonyítás triviális). Ez az összefüggés átírható:

$$\mathcal{M} \quad \models \quad (b_1 \wedge L_1 \wedge b)\varrho \wedge \forall \widehat{C} \quad ,$$

ami alapján

$$\mathcal{M} \quad \models \quad \forall \widehat{C} \quad ,$$

és ezért

$$\mathcal{M} \quad \models \quad \forall\big(\langle \widehat{C} \rangle \sigma\big) \quad ,$$

ami pedig – mivel $\langle\widehat{C}\rangle\sigma \subseteq D$ – maga után vonja a következőt:

$$\mathcal{M} \models \forall D \quad .$$

Ezért az

$$\mathcal{M} \models (b_1 \wedge L_1 \wedge b)\varrho \wedge \forall D$$

összefüggés is teljesül, amiből adódik, hogy

$$\mathcal{M} \models \big(b_1 \wedge L_1 \wedge b \wedge \langle D\rangle\big)\varrho \quad .$$

Ennek triviális következménye, hogy $\mathcal{M} \models (3.5)\varrho$. □

Mint a 3.3. ábrán is látszik, az $C = \{L_1 \vee \ldots \vee L_k\}$ klóz hasonlít az eddig általunk *ágklóznak* (2.3.10. definíció) ismert, tablóból kinyerhető klózokra: az L_i-k különböző ágakon helyezkednek el, csak most nem kötöttük ki, hogy azonos szülővel is kell rendelkezniük. A C a felhasználási célját tekintve is hasonló az ágklózokhoz:

- A 3.4.13.(1) alapján C-nek új példányai csatolhatók bármely olyan ághoz, mely valamely L_i-t tartalmazza.
- A 3.4.13.(2) alapján, C-t felhasználva megalkotható a kívánt redundanciafogalom.

A C karakterizálásának érdekében logikusnak látszik tehát egy az ágklózhoz hasonló fogalmat definiálni:

3.4.14. Definíció. (Független ágklóz)

Legyen $\mathcal{T}$ egy tabló. Legyenek $L_1, \ldots, L_k$ $(k \geq 1)$ olyan literálok a $\mathcal{T}$-ben, hogy

(1) $\mathcal{T}$ minden ága legfeljebb egy L_i-t tartalmaz, és

(2) $\mathcal{T}$ minden olyan $\mathcal{B}$ ága esetén, mely semelyik L_i-t sem tartalmazza:

$$\mathcal{FV}\big(\{L_1, \ldots, L_k\}\big) \cap \mathcal{FV}(\mathcal{B}) \quad = \quad \emptyset \quad .$$

Ekkor az $\{L_1, \ldots, L_k\}$ *független ágklóza* bármely valamely L_i-t tartalmazó ágnak. □

A 3.4.13. tétel alapján tehát bármely olyan rigid literálos tablót építő tablókalkulus esetén, melynek valamely *levezetési szabálya biztosítja a független ágklózok új példányainak igény szerinti előállítását*, a következő redundanciafogalmat lehet kimondani:

Redundancia rigid literálos tabló ágára nézve:

A D klóz redundáns a rigid literálos $\mathcal{T}$ tabló $\mathcal{B}$ ágára nézve, ha van olyan $L \in \mathcal{B}$ és van olyan L-et tartalmazó C független ágklóza a $\mathcal{B}$-nek, hogy a D valamely részklóza példánya a C új példányának. □

Látható, hogy az itt kimondott redundanciafogalom a hipertablóhoz megadott redundanciafogalomnak (3.4.3. definíció) egy kibővítése, hiszen most is azt vizsgáljuk, hogy az adott águhoz csatolandó D klóz egy része az ág valamely elemének nem a példánya-e. Ám míg hipertabló esetén a D-nek ez a része egy-egy literál lehetett, addig most egy-egy tetszőleges *részklóz*. Ennek megfelelően az ág vizsgált elemei sem csupán literálok lehetnek most, hanem *független ágklózok* is. Vegyük észre, hogy a klózokra kimondott redundaciafogalomból (3.4.7. definíció) is átörökítettünk egy bizonyos fajta vizsgálatot: ez a *változóidegenségre* vonatkozó megszorítás a 3.4.14.(2)-ben.

Máshogy fogalmazva: a hipertabló redundanciafogalma egy speciális esete a most leírt redundanciafogalomnak. Mivel hipertabló esetén a tabló minden egyes literálja változóidegen a tabló egyéb formuláival, így a *hipertabló független ágklózai egyetlen literálból* állnak. A hipertabló kiterjesztési szabálya a 2.3.5.(2)(c)-ben a tabló bármely literáljának (azaz független ágklózának) *képezheti új példányait*. Vagyis hipertabló esetén alkalmazható a mostani redundanciafogalom; a 3.4.3. definícióban leírt redundanciafogalom pontosan ennek egy speciális változata – a hipertablóban alkalmazott *tisztító helyettesítéseknek* köszönhetően:

- A 3.4.13. tétel C független ágklóza hipertabló esetén egyetlen literálból áll.
- A 3.4.13.(1) $\widehat{C}$ új klózpéldánya szintén egyetlen literálból áll.
- A 3.4.13.(2)-ben vizsgált $\langle\widehat{C}\rangle\sigma \subseteq D$ reláció könnyebben vizsgálható, hiszen $\langle\widehat{C}\rangle$-ről mint egyetlen literálról kell megmutatni, hogy a D valamely literálja neki példánya.

Kimondhatjuk, hogy Baumgartnernél a tisztító helyettesítések alkalmazása egy egyszerűen vizsgálható redundanciafogalom kialakítását is lehetővé tette.

Multi-hipertabló esetén (mivel nincs független ágklózok új példányait előállító levezetési szabálya) a fenti redundanciafogalom nem alkalmazható. A multi-hipertabló kalkulusnak olyan továbbfejlesztése lenne a kívánatos, mely független ágklózok új példányait tudja generálni anélkül, hogy a kalkulus elvesztené a teljességét. Ennek felderítése lehet a következő feladat.

3.5. Heurisztika

Felmerül a kérdés, hogy multi-hipertabló esetén a kiterjesztések $\mathcal{E}(\mathcal{B},\mathcal{C})$ halmazának (lásd: 3.1.4. definíció) – mivel ez akár több kiterjesztést is magában foglalhat – melyik elemét alkalmazzuk a $\mathcal{B}$ ágon? Érdemes lenne azt a kiterjesztést alkalmazni, mellyel a lehető leghamarabb fejezhetjük be az aktuális levezetést. Azaz a levezetés további részéről valamiféle becslést, *heurisztikát* kell szolgáltatnunk. Már korábban, a 3.1.7. megjegyzésben is felmerült ez az igény. A kiterjesztések halmazán egy *teljes rendezést* definiálunk, hogy ezen rendezés alapján alkalmazásra kiválaszthassuk az $\mathcal{E}(\mathcal{B},\mathcal{C})$ egyik minimális elemét. A rendezés definíciója alkalmazásonként akár más és más

lehet, vagyis ez a definíció az egyik olyan komponens a multi-hipertablón alapuló tételbizonyítókban, mely igény szerint testreszabható. Az alábbiakban egy lehetséges, általánosan jól használható rendezési definíciót adunk meg [18].

3.5.1. Definíció. (Heurisztika)

Adott egy $\mathcal{T}$ multi-hipertabló. Ennek egy ágának valamely $[E_1, \sigma_1]$ és $[E_2, \sigma_2]$ kiterjesztései esetén $[E_1, \sigma_1] \leq [E_2, \sigma_2]$, ha

(1) $|E_1| < |E_2|$, vagy

(2) $|E_1| = |E_2|$ és $|R_1| \leq |R_2|$, ahol $R_i = \{x \mid x \in \mathcal{D}om(\sigma_i) \cap \mathcal{FV}(\mathcal{T})\}$. □

Tehát az $[E_1, \sigma_1] \leq [E_2, \sigma_2]$ reláció fenti vizsgálatát két szempont szerint végezzük el:

(1) Azt a kiterjesztést választjuk kisebbnek, melynek elvégzésével *kevesebb új ágat* állítunk elő. Azaz az E_i *literáljainak száma* az elsődleges szempont: ez minél kisebb, az adott kiterjesztés annál előnyösebb.

(2) Ha pedig E_1 és E_2 literáljainak száma megegyezik, akkor egy másik szempont alapján döntünk. Ezen másodlagos szempont alapja a $\mathcal{T}$ azon szabad változóinak, azaz azon *rigid változóknak a száma*, melyek az egyes kiterjesztések elvégzésekor behelyettesítődnek. Azaz ha σ_1 nem helyettesít több rigid változóba, mint σ_2, akkor a reláció fennáll.

Míg az (1) szempont használata teljesen magától értetődő, addig a (2)-é talán nem annyira. A (2)-ben tulajdonképpen a következőt próbáljuk megfogalmazni: minél kevesebb rigid változó kerül behelyettesítésre, annál inkább lehetőséget teremtünk a későbbi helyettesítések számára, azaz annál kisebbre csökkentjük annak a valószínűségét, hogy *új input klózpéldányt* kelljen a későbbi kiterjesztések előállítása során generálni.

3.5.1. Paraméteres tételek

Jelen fejezetben megvizsgáljuk az 1.1.12. definíció által az input klózokra adott megszorítás eltörlésének következményeit. Azaz engedjünk meg *nyitott input klózokat* is, vagyis egy input klóz tartalmazhasson szabad változókat.

Már az elején rögtön összeütközésbe kerülünk a klózok egyszerűsített jelölésével, mikor is egy $\forall x_1 \ldots \forall x_k(L_1 \vee \ldots \vee L_n)$ klóz kvantoros előtagjait elhagyhatjuk, vagyis a klózt $L_1 \vee \ldots \vee L_n$ alakban használhatjuk. Ez a rövidítés zárt klózok esetén nyilván helyénvaló, hisz olyankor tudjuk, hogy $L_1 \vee \ldots \vee L_n$ minden változóelőfordulása kötött. Nyitott klózok esetén viszont nem marad más választásunk, mint külön regisztrálni, hogy az input klózok $\mathcal{C}$ halmazában mely változók univerzálisak és melyek szabadok – utóbbiakat nevezzük mostantól *paraméterváltozóknak*. Egy C klóz esetén a C-ben előforduló paraméterváltozók halmazát $\mathcal{P}ar(C)$-vel jelöljük; egy $\mathcal{C}$ klózhalmaz

esetén pedig $\mathcal{P}ar(\mathcal{C}) = \bigcup_{C\in\mathcal{C}} \mathcal{P}ar(C)$. Ezen plusz adminisztrációra a klózok *új példányainak* előállítása során lesz szükségünk: az új klózpéldányokat leíró 1.1.13. definíció módosítandó oly módon, hogy a benne szereplő

$$\mathcal{D}om(\sigma) = \mathcal{FV}\big(\langle C\rangle\big)$$

feltételt átírjuk a

$$\mathcal{D}om(\sigma) = \mathcal{FV}\big(\langle C\rangle\big)\backslash\mathcal{P}ar(C)$$

feltételre.

A paraméterváltozók engedélyezése érdekes lehetőségeket nyit a tételbizonyításban. Míg zárt $\mathcal{C}$ esetén a tételbizonyító egy „igen"/„nem" válasszal tud csak szolgálni, addig nyitott $\mathcal{C}$ esetén a *levezetés kimenete* egy olyan θ *helyettesítés*, melyre $\mathcal{C}\theta$ kielégíthetetlen. Természetesen $\mathcal{D}om(\theta)$ paraméterváltozók halmaza, azaz $\mathcal{D}om(\theta) \subseteq \mathcal{P}ar(\mathcal{C})$. Vegyük észre, hogy ha $\theta = \epsilon$, akkor $\mathcal{C}\theta$ kielégíthetetlensége a $\mathcal{C}$ kielégíthetetlenségét jelenti.

A lineáris inputrezolúción alapuló *Prolog* megengedi paraméterváltozók alkalmazását. Sőt, a Prolog nem is tesz különbséget univerzális változók(ba behelyettesített rigid változók) és paraméterváltozók között, minden változót eleve paraméterváltozóként kezel. E helyettesítések kezeléséhez szükséges a *backtracking* a Prologban.

A hipertabló [3] és a rigid hipertabló [19] kalkulusok kizárólag zárt input klózokkal dolgoznak. Ennek megfelelően ezen kalkulusok – és természetesen a zárt klózokon működő multi-hipertabló is – rendelkeznek az ún. *levezetésfolytonosság*[20] tulajdonságával, azaz nincsenek „zsákutcák" a levezetésben: ha egy x univerzális változóba (behelyettesített rigid változóba) helyettesítünk a levezetés során, ezzel nem zárjuk ki más helyettesítések alkalmazását ugyanezen x-en (pontosabban: egy az x-be helyettesített másik rigid változón). Ez a tény abból fakad, hogy egy input klóz akárhányszor példányosítható, mely során az univerzális változók újabb és újabb rigid változókkal helyettesítődnek be.

Ezzel szemben egy *paraméterváltozóba helyettesíteni visszafordíthatatlan* döntést jelent. Nem véletlen tehát, hogy a fentebb említett cikkek szerzői kifejezetten megtiltották kalkulusaikban a paraméterváltozók használatát, így akarván megőrizni kalkulusaik levezetésfolytonosságát. A multi-hipertablóban viszont nem kívánunk lemondani a fentebb említett θ helyettesítés automatikus meghatározásáról, így engedélyezzük a paraméterváltozók használatát. Vizsgáljuk meg, hogy ez milyen plusz feladatokat ró a multi-hipertablón alapuló tételbizonyítókra. Első megközelítésben a következőt szögezhetjük le: paraméterváltozóba való helyettesítés esetén nem kerülhetjük el a backtracking alkalmazását, egyéb esetekben viszont célszerű az előző fejezetben leírt módon, levezetésfolytonosan menedzselnünk a levezetést.

Tehát nagy vonalakban a következőt fogjuk tenni: egy $[E, \sigma]$ kiterjesztés alkalmazásakor ha $\mathcal{D}om(\sigma)$ tartalmaz paraméterváltozót, akkor létrehozunk egy ún. *visszaállítási pontot*[21]. A visszaállítási pont elegendő információt kell tartalmazzon ahhoz,

[20] angol: *proof confluence* (lásd: [13])
[21] angol: *rollback point* (lásd: [18])

hogy a későbbiek során ha a levezetés „zsákutcába" került, akkor vissza tudjuk állítani a levezetés aktuális állapotát, és egy másik irányba tereljük a levezetést. Ezen letárolandó információk jellegét itt és most nem kíséreljük meg pontosan leírni, mivel ezen információk milyensége nagyban implementációfüggő; csak implementációs szinten határozható meg, hogy mit értünk egy levezetés státusza alatt, és mi minden szükséges ezen státusz későbbi visszaállításához. Az A.3. fejezetben leírjuk a multi-hipertabló egy lehetséges megvalósítását, melyben egész pontosan meghatározzuk a visszaállítási pontokban letárolandó információk halmazát, és az alkalmazandó backtracking jellegét. Mint ott látni fogjuk, paraméterváltozók tiltása esetén sincs igazán jelentősége levezetésfolytonosságról beszélni, mivel „zsákutcák" a levezetésben így is előfordulhatnak, csak ezen „zsákutcákban" szereplő információk törlése a levezetésből nem feltétlenül szükséges. Egy effektív, tár- és időtakarékos implementáció esetén ezen zsákutcák törlésre kerülnek, tehát a levezetés korábbi státuszai kerülnek léptennyomon visszaállításra – vagyis a backtracking egy effektív implementáció általánosan használt eszköze lesz.

Célszerű arra törekedni, hogy a tételbizonyító a *lehető legáltalánosabb θ helyettesítést* adja kimenetként. A 3.5.1. definícióban leírt teljes rendezés alábbi kiegészítése ezirányba mutat.

3.5.2. Definíció. (Heurisztika - Paraméterváltozók használata)

Adott egy $\mathcal{C}$ klózhalmaz $\mathcal{T}$ multi-hipertablója. Ennek egy ágának valamely $[E_1, \sigma_1]$ és $[E_2, \sigma_2]$ kiterjesztései esetén $[E_1, \sigma_1] \leq [E_2, \sigma_2]$, ha

(1) $|P_1| < |P_2|$, ahol $P_i = \{x \mid x \in \mathcal{D}om(\sigma_i) \cap \mathcal{P}ar(\mathcal{C})\}$, vagy

(2) $|P_1| = |P_2|$ és $|E_1| < |E_2]$, vagy

(3) $|P_1| = |P_2|$ és $|E_1| = |E_2|$ és $|R_1| \leq |R_2|$, ahol

$$R_i \quad = \quad \{x \mid x \in \mathcal{D}om(\sigma_i) \cap \mathcal{FV}(\mathcal{T})\} \quad . \qquad \square$$

Mint látszik, a reláció teljesülésének vizsgálatát kiegészítettük egy további szempontal, mely szempont az eddigi kettőnél is rangosabb, így a másik kettő elé, az (1) helyre került: az egyes kiterjesztések által behelyettesítendő *paraméterváltozók száma*. Ez minél kisebb, az adott kiterjesztés annál előnyösebb. Az (1) és a (3) együttesen azt célozza meg, hogy a kimenetként előálló θ a lehető legáltalánosabb legyen.

A Prolog implementációk alapvető szolgáltatása az is – melyről a multi-hipertabló kapcsán sem kívánunk lemondani –, hogy több lehetséges θ-t is képesek kimenetként megadni; azaz egy kimenet megtekintése után kérhető egy másik lehetséges kimenet megtekintése is, egészen addig, míg azok el nem fogynak. Megjegyezzük, hogy ebben az esetben is a backtracking ezen szolgáltatás háttere.

4. fejezet

Összefoglalás

A disszertációban a *hipertabló* (és közvetve a hiperrezolúció) teljes automatizálhatóságának kérdésével foglalkoztunk. Megvizsgáltuk Baumgartner hipertabló kalkulusát [3], melynek van egy nem automatizálható eleme: a tisztító helyettesítések előállítása. Ezen probléma megoldásának egy ígéretes iránya a rigid változók hipertablóban való alkalmazása; ezzel több cikk is foglalkozik a szakirodalomban [4, 10, 19]. Ezek közül kiemeltük és részletesen taglaltuk Kühn *rigid hipertablóját* [19], melyről bebizonyítottuk, hogy nem helyes kalkulus, illetve rámutattunk, hogy a kalkulus teljessége nem nyert bizonyítást.

A 3. fejezetben összegeztük saját eredményeinket. A 3.1. fejezetben megadtuk a *multi-hipertabló* kalkulust [18]. Ebben kiküszöböltük a tisztító helyettesítések használatát. Ezt rigid változók alkalmazásával értük el. Még ugyanabban a fejezetben bebizonyítottuk, hogy a multi-hipertabló helyes kalkulus. A multi-hipertabló *teljességének* bizonyítását a 3.2. fejezetben végeztük el. Ennek során a hiperrezolúció teljességéből indultunk ki, azaz abból, hogy bármely kielégíthetetlen klózhalmazhoz megadható hiperrezolúciós cáfolat. Első lépcsőben definiáltuk a *hiperrezolúciós gráfot*, melyet egy adott hiperrezolúciós cáfolat alapján tudunk felépíteni. Második lépcsőben algoritmust adtunk arra, hogy egy hiperrezolúciós gráfból zárt multi-hipertablót konstruáljunk.

A 3.3. fejezetben a klózgenerálás témakörével foglalkoztunk, és részletesen leírtuk a *klózgeneráló tabló* módszerét [18]. Bebizonyítottuk, hogy a klózgeneráló tabló által megadott klózhalmaz akkor és csak akkor kielégíthető, ha az eredeti formulahalmaz is kielégíthető. Az ismert klózgeneráló algoritmusok exponenciális bonyolultságúak, ezért foglalkoztunk a klózgeneráló tabló *linearizálásával*. Ezt a tabló fa adatszerkezetének módosításával, az ún. *DAG adatszerkezet* alkalmazásával értük el. Továbbá azt is megvizsgáltuk, hogy a generált klózhalmazon milyen egyszerűsítéseket végezhetünk a felépített tabló alapján: az ágak lezárásával felismerhetjük és eliminálhatjuk az *érvényes klózokat*.

A 3.4. fejezetben a *redundancia* témakörével foglalkoztunk. Három, logikailag hasonló célra használtunk fel redundanciával kapcsolatos eredményeket:

- A *hipertabló* Baumgartner által megadott *redundancia kritériumát* (azaz a hipertabló kiterjesztési szabályának megszorítását) mutattuk be és vezettük le.
- *Klózok egyszerűsítését* végeztük el redundanciavizsgálat segítségével: egy klóznak eliminálju azon részklózát, mely redundáns a klóz egyéb részére nézve. Bevezettük egy klóz *irredundáns variánsának* fogalmát, és megmutattuk, hogy bármely klóznak létezik ilyen variánsa. Azt is megmutattuk, hogy ha egy klóznak több ilyen variánsa létezik, akkor ezen variánsok egymás átnevezettjei.
- Megvizsgáltuk, hogy tetszőleges *rigid literálos tablóhoz* (és így a multi-hipertablóhoz is) hogyan adható meg redundanciafogalom. Bevezettük a *független ágklóz* fogalmát, és bebizonyítottuk, hogy egy ág független ágklózának új példánya mindig redundáns az ágra nézve. Ezen észrevételre támaszkodva mutattuk meg azt is, hogy egy olyan klóz, melynek valamely részklóza példánya a független ágklóz új példányának, szintén redundáns az ágra nézve. Rámutattunk, hogy egy rigid literálos tablót építő kalkulusnak speciális tulajdonságúnak kell lennie ahhoz, hogy a felírt redundancia kritérium alkalmazható legyen: a kalkulus valamely szabályának biztosítania kell a *független ágklózok új példányainak* előállítását.

A 3.5. fejezetben a *heurisztikus tételbizonyítás* lehetőségeivel foglalkoztunk a multi-hipertabló kapcsán, és meg is adtunk egy lehetséges heurisztikát. Azt is megvizsgáltuk, hogy a *klózok zártságára* vonatkozó kikötésnek az eltörlése vajon milyen következményekkel jár a multi-hipertablóra nézve, valamint továbbfejlesztettük az előzőekben megadott heurisztikafogalmat.

Az A. függelékben a disszertációban előforduló néhány módszernek adjuk meg algoritmikus megvalósítását. Először egy saját, egyszerűbben használható unifikációs algoritmust írunk le. Utána egy klóz, illetve egy kiterjesztés irredundáns variánsának előállítását végző algoritmust mutatunk be. Végül pedig a multi-hipertabló kalkulusnak adjuk meg az algoritmikus megvalósítását.

A disszertáció fő eredményeivel kapcsolatosan a következő megállapításokat tehetjük: a multi-hipertabló kalkulus egy teljesen automatizált, helyes és teljes hipertabló kalkulus, melyhez felhasználható a lineáris bonyolultságú klózgeneráló tabló. A multi-hipertabló egyetlen hiányossága és tulajdonképpen gyakorlatban való alkalmazhatóságának korlátozója a megfelelő redundanciafogalom hiánya.

A jövőben a multi-hipertablónak olyan továbbfejlesztéseit lesz érdemes kutatni, melyek biztosítják a független ágklózok új példányainak előállítását.

5. fejezet

Summary

In the dissertation, we investigated the problems of the automation of *hyper tableaux* (and consequently, hyper-resolution). We introduced Baumgartner's hyper tableau calculus [3], which had a component that could not be automated, namely the generation of purifying substitutions. Trying to overcome this problem, several authors apply rigid variables in hyper tableaux [4, 10, 19]. We presented one of these calculi in detail, namely Kühn's *rigid hyper tableau calculus* [19], which we proved not to be sound. Furthermore, we pointed out that this calculus had not been proved to be complete yet.

Section 3 contains our main results. In Section 3.1, we introduced the *multi-hyper tableau calculus* [18], which eliminated the use of purifying substitutions. This was achieved by applying rigid variables. In the same section, we proved the multi-hyper tableau calculus to be sound. The proof of the completeness of multi-hyper tableaux can be found in Section 3.2. In this proof, we started from the fact that hyper-resolution was complete, i.e., there was a hyper-resolution refutation for any unsatisfiable clause set. First, we defined the *hyper-resolution graph*, which could be constructed on the bases of a given hyper-resolution refutation. Next, we introduced an algorithm to generate a closed multi-hyper tableau from a hyper-resolution graph.

In Section 3.3, we dealt with the topic of clause generation, and introduced the method of *clause generating tableaux* [18] in detail. We proved that the clause set generated by a clause generating tableau was satisfiable if and only if so was the original formula set. Since the known clause generating algorithms are exponential, we discussed how to make the method of clause generating tableaux *linear*. This was achieved by turning the tree data structure applied by tableaux into the so-called *DAG data structure*. Furthermore, we discussed what kind of reduction based on the constructed tableau could be performed on the generated clause set: by closing branches, *valid clauses* could be identified and eliminated.

In Section 3.4, we dealt with the problem of *redundancy*. We performed tests based on redundancy for three similar purposes:

- The *redundancy criterion* (i.e., the restriction of the extension rule) of Baumgartner's *hyper tableaux* was introduced and obtained.

- *Clauses were reduced* by the use of a redundancy check: a subclause of a clause could be eliminated if it was redundant w.r.t. the rest of the clause. We introduced the concept of an *irredundant variant* of a clause, and showed that any clause had such a variant. We also proved that if a clause had more than one irredundant variants then these variant were renamed variants of each other.

- We discussed whether a redundancy concept could be defined for arbitrary *rigid clausal tableaux* (and consequently, for multi-hyper tableaux). We introduced the concept of a *separate branch clause*, and proved that a new instance of a separate branch clause of a branch was always redundant w.r.t. the branch. Using this fact, we showed that a clause which had a subclause being an instance of a new instance of a separate branch clause was also redundant w.r.t. the branch. We pointed out that a rigid clausal tableau calculus had to fulfil a special condition in order to make the above redundancy criterion applicable: the calculus had to support the generation of *new instances of separate branch clauses.*

In Section 3.5, we dealt with the potential of *heuristic theorem proving* in connection with multi-hyper tableaux, and proposed possible heuristics. We discussed what were the consequences of letting the *clauses be open* and improved these heuristics.

In Appendix A, we give algorithmic implementation of some methods introduced in the dissertation. First, an own easy-to-use unifying algorithm is introduced. Furthermore, we describe an algorithm which generates the irredundant variant of a clause and an extension, respectively. Finally, the multi-hyper tableau calculus is implemented algorithmically.

We are summarizing the main results of the dissertation in the following way: the multi-hyper tableau calculus is entirely automated, sound, and complete, and the linear clause generating tableau method can be employed with it. The lack of a suitable redundancy concept is the only deficiency of multi-hyper tableaux, since it limits their usability in practice.

It would be expedient to do research on such improvements of the multi-hyper tableau calculus which provides the generation of new instances of separate branch clauses.

A. függelék

Algoritmikus megoldások

A.1. Unifikáció

Az alábbi algoritmus által kiszámított $\mathcal{U}()$ függvény $n \geq 0$ db. kifejezéspárhoz rendeli azok legáltalánosabb unifikátorát (MGU), amennyiben az létezik.

A.1.1. Algoritmus. (Unifikáció)

1: **function** $\mathcal{U}((E_1, F_1), \ldots, (E_n, F_n))$
2: **if** $n = 1$ **then**
3: **if** $E_n = P(s_1, \ldots, s_k)$ és $F_n = P(t_1, \ldots, t_k)$ **then**
4: **return** $\mathcal{U}\big((s_1, t_1), \ldots, (s_k, t_k)\big)$
5: **else if** E_n vagy F_n változó **then**
6: **if** E_n változó **then** $(x, t) := (E_n, F_n)$
7: **else** $(x, t) := (F_n, E_n)$

8: **if** $x = t$ **then return** ϵ
9: **if** $x \notin \mathcal{FV}(t)$ **then return** x/t
10: **end if**

11: **else**
12: $\sigma := \epsilon$
13: **for all** $i \in \{1, \ldots, n\}$ **do**
14: **if** $\mathcal{U}\big(E_i\sigma, F_i\sigma\big)$ létezik **then** $\sigma := \sigma\, \mathcal{U}\big(E_i\sigma, F_i\sigma\big)$
15: **else** goto 19
16: **end for**
17: **return** σ
18: **end if**

19: **return** „nem létezik”
20: **end function**

Az algoritmus két fő részből tevődik össze:

- A (2-10) közötti *kalkulációs egységből*, mely 1 db. kifejezéspárnak állítja elő a MGU-ját.
- A (11-18) közötti *disztribúciós egységből*, mely az n db. kifejezéspárnak egyenként állítja elő a MGU-ját a (14)-ben (rekurzívan hívva a kalkulációs egységet), és ugyanitt veszi azok kompozícióját.

A kalkulációs egység a következő szakaszokból áll:

- (3-4): Ha mindkét kifejezés ugyanazon predikátumszimbólumból képzett atomi formula vagy ugyanazon függvényszimbólumból képzett term, akkor a disztribúciós egységgel páronként kiszámítjuk az argumentumaik MGU-ját.
- (5-10): Ha a két kifejezés közül legalább az egyik változó, akkor azt eljelöljük x-szel, míg a másik kifejezést t-vel. A (9)-ben elvégezzük az *occur check* nevű tesztet – ha t-ben előfordul x, akkor nincs MGU-juk –, majd ha ezen sikeresen túljutottunk, akkor x behelyettesíthető t-vel.

Ha (2-18) között nem sikerült a kifejezéspárok MGU-ját megállapítani, akkor a vezérlés a (19)-re kerül, ahol az algoritmus negatív válasszal tér vissza.

A.2. Klóz irredundáns variánsa

Jelen fejezetben klóz irredundáns variánsának előállítását végző algoritmust írunk le. A fejezet két függvényt tartalmaz. Az IRREDUNDANT függvény végzi az irredundáns variáns előállítását, míg az ISINSTACEOF ennek egy segédfüggvénye.

Az ISINSTANCEOF függvény két klózt, D-t és C-t kapja paraméterül, és „igaz”-at ad vissza, ha a C valamely részklóza a D-nek példánya, egyébként pedig „hamis”-at.

A.2.1. Algoritmus.

1: **function** ISINSTANCEOF(D, C)
2: Legyen $L_1 \in D$.
3: **if** L_1 nem létezik **then return** „igaz”
4: **for all** $L_2 \in C$ **do**
5: **if** L_1 és L_2 azonos előjelűek, $\sigma = \mathcal{U}\big(\underline{L}_1, \underline{L}_2\big)$ létezik és $L_1\sigma = L_2$ **then**
6: **if** ISINSTANCEOF$\Big(\langle D \backslash \{L_1\}\rangle\sigma, C \backslash \{L_2\}\Big)$ **then return** „igaz”
7: **end if**
8: **end for**
9: **return** „hamis”
10: **end function**

A függvény rekurzív; a rekurzív hívás a (6)-ban történik. A (2)-ben kiválasztunk egy L_1 literált a D-ből; ha D üres, akkor a függvény végrehajtása „igaz” válasszal leáll. Ha

azonban sikerül L_1-et kiválasztanunk, akkor a (4-8) ciklus próbálja L_1-et a C valamely L_2 literáljával unifikálni az (5)-ben. Sőt, itt ennél több is történik: megvizsgáljuk, hogy $L_1\sigma$ azonos-e L_2-vel, ahol σ az ő MGU-juk. Minden sikeres L_1 és L_2 párosítás után az unifikált literálokat töröljük a D-ből és a C-ből, és rekurzíve meghívjuk a függvényt. Mint látható, a függvény akkor és csak akkor ad „hamis" választ, ha valamely L_1-hez semelyik L_2 sem párosítható.

A következő, IRREDUNDANT függvény egy C klózt kap paraméterül, melynek irredundáns variánsát állítja elő.

A.2.2. ALGORITMUS. (KLÓZ IRREDUNDÁNS VARIÁNSA)

```
 1: function Irredundant(C)
 2:     C' := ⊥
 3:     while C ≠ ⊥ do
 4:         D := ⊥
 5:         for all L ∈ C\D do          ▷ D változása esetén a ciklust újra kell indítani!
 6:             if D = ⊥ vagy FV(L) ∩ FV(⟨D⟩) ≠ ∅ then D := D ∪ {L}
 7:         end for
 8:         C := C\D
 9:         if nem IsInstanceOf(D, C ∪ C') then C' := C' ∪ D
10:     end while
11:     return C'
12: end function
```

A függvény C-ből a C' klózt állítja elő, mely C részklóza, és nem tartalmaz olyan D részklózt, mely redundáns lenne a $C'\backslash D$-re nézve. Tehát a függvény a (2)-ben inicializált C'-t fogja mindig bővítgetni ilyen D részklózokkal, míg ezzel párhuzamosan a C-ből törli ezeket a D-ket a (8)-ban, egészen addig, míg C üressé nem válik a (3)-ban. Tehát miután D-t a (4)-ben inicializáljuk, az (5-7) ciklusban összegyűjtjük benne a C literáljainak egy olyan csoportját, melyek tartalmaznak közös változót, de a C egyéb literáljaival változóidegenek. Fontos momentum, hogy D bármely bővítése esetén a ciklust újra kell indítani. Miután C-ből kiemeljük a D-t a (8)-ban, megvizsgáljuk, hogy D-nek példánya-e valamely C-beli részklóz; ezt az IsInstanceOf függvény hívásával tesszük a (9)-ben. Mint itt látszik, a függvény második paramétere a $C \cup C'$; ennek oka az, hogy C egyes literáljait már fizikailag átmozgattuk a C'-be. Ha D-nek nem példánya a $C \cup C'$ egy részklóza sem, akkor D-t hozzávesszük a C'-höz még mindig a (9)-ben, vagyis bekerül a végleges eredménybe, melyet a függvény a (11)-ben ad vissza.

A.2.1. Kiterjesztés irredundáns variánsa

Megadjuk az IRREDUNDANT függvény egy olyan változatát, mely egy $\mathcal{T}$ multi-hipertabló $[E, \sigma]$ kiterjesztését kapja paraméterként, és ennek irredundáns variánsát állítja elő.

A.2.3. Algoritmus. (Kiterjesztés irredundáns variánsa)

1: **function** Irredundant($[E, \sigma]$, $\mathcal{T}$)
2: $E' := \bot$
3: **while** $E \neq \bot$ **do**
4: $D := \bot$
5: **for all** $L \in E \backslash D$ **do** ▷ *D változása esetén a ciklust újra kell indítani!*
6: **if** $D = \bot$ vagy $\mathcal{FV}(L\sigma) \cap \mathcal{FV}\big(\langle D \rangle \sigma\big) \neq \emptyset$ **then** $D := D \cup \{L\}$
7: **end for**
8: $E := E \backslash D$
9: **if** nem IsInstanceOf$\Big(\langle D \rangle \sigma, \langle E \cup E' \rangle \sigma\Big)$ **then** $E' := E' \cup D$
10: **end while**
11: $\sigma' := \Big\{ x/t \in \sigma \;\Big|\; x \in \mathcal{FV}\big(\langle E' \rangle\big) \cup \mathcal{FV}(\mathcal{T}) \Big\}$
12: **return** $[E', \sigma']$
13: **end function**

A fenti algoritmus nagyban hasonlít az A.2.2. algoritmushoz. A C' helyett az E' klózt generálja, mely az E-nek részklóza; tehát benne E literáljai eredeti formájukban, a σ helyettesítés elvégzése nélkül őrződnek meg. A σ-t csak időlegesen végezzük el egy-egy vizsgálat erejéig: a változóidegenség vizsgálatakor a (6)-ban, és a (9)-ben annak vizsgálatakor, hogy D-nek példánya-e az E valamely részklóza (pontosabban fogalmazva: $\langle D \rangle \sigma$-nak példánya-e az $\langle E \cup E' \rangle \sigma$ valamely részklóza). Ezzel E' generálásának a végére értünk. Ami még hátravan, az a 3.4.11.(2) által definiált σ' megkonstruálása a (11)-ben.

A.3. Multi-hipertabló

Jelen fejezetben a multi-hipertabló egy algoritmikus megvalósítását írjuk le. Magát a multi-hipertabló kalkulust a később definiálásra kerülő Mht függvény képviseli; a többi rutin ennek segédrutinja. A rutinok *három globális objektumot* használnak:

- A $\mathcal{C}$ *input klózhalmazt.*
- A $\mathcal{B}$ literálhalmazt, mely az *aktuálisan lezárandó ágat* képviseli. A tablónak tehát egyszerre csak egy ágát tároljuk; ez megfelel annak, amit a 3.5.1. fejezetben a backtrackingről írtunk, vagyis a kalkulus pontosan backtracking segítségével kerül most megvalósításra.
- A Θ helyettesítést, mely a *tablón eddig elvégzett helyettesítések* kompozíciója. A teljes tablót csupán a Θ képviseli.

Előbb az Ext$^{\ominus}$ eljárást írjuk le, mely a $\mathcal{B}$-hez és a $\mathcal{C}$-hez tartozó kiterjesztéseket állítja elő, tehát a 3.1.4. definícióban leírt $\mathcal{E}(\mathcal{B}, \mathcal{C})$ halmazt. Az előállított kiterjesztéseket az $\mathcal{E}$ nevű változóban (halmazban) gyűjti, melyet cím szerinti átadással kap meg paraméterként (a cím szerint átadott paramétereket mindig be fogjuk keretezni a paraméterlistákban).

A.3.1. Algoritmus.

1: **procedure** $\textsc{Ext}^{\ominus}(\boxed{\mathcal{E}})$
2: **for all** $C^{\ominus} \in \mathcal{C}$ nem pozitív klóz **do**
3: $\textsc{Ext}\big(\widehat{C}^{\ominus}, [\bot, \epsilon], \mathcal{E}\big)$ [1]
4: **end for**
5: **end procedure**

Mint látható, az eljárás nem csinál mást, mint a $\mathcal{C}$ minden egyes nem pozitív $C^{\ominus}$ klózára (lásd: 3.1.4.(1)) hívja az Ext segédeljárást.

Az Ext eljárás leírása következik az alábbiakban. Az eljárás 3 paramétert vár: az $\textsc{Ext}^{\ominus}$ által kiválasztott $C^{\ominus}$ klózt, az éppen konstruálás alatt álló $[E, \sigma]$ kiterjesztést[2], valamint az $\mathcal{E}$ halmazt (cím szerint), melyben a $\mathcal{B}$-hez és a $\mathcal{C}$-hez eddig összegyűjtött kiterjesztések vannak. Az eljárás rekurzív, önmagát a (9)-ben hívja.

A.3.2. Algoritmus.

1: **procedure** $\textsc{Ext}(C^{\ominus},\ [E, \sigma],\ \boxed{\mathcal{E}})$
2: Legyen $L^{\ominus} \in C^{\ominus}$ negatív literál.
3: **if** $L^{\ominus}$ nem létezik **then**
4: $\mathcal{E} \leftarrow [E \cup C^{\ominus}, \sigma]$ [3]
5: **else**
6: **for all** $C^{\oplus} \in \mathcal{B} \cup \mathcal{C}$ pozitív klóz **do**
7: **for all** $L^{\oplus} \in \widehat{C}^{\oplus}$ **do**
8: **if** $\theta := \mathcal{U}\big(\underline{L}^{\ominus}\Theta\sigma,\ \underline{L}^{\oplus}\Theta\sigma\big)$ létezik **then**
9: $\textsc{Ext}\Big(C^{\ominus}\backslash\{L^{\ominus}\},\ \Big[E \cup \big(\widehat{C}^{\oplus}\backslash\{L^{\oplus}\}\big), \sigma\theta\Big],\ \mathcal{E}\Big)$
10: **end if**
11: **end for**
12: **end for**
13: **end if**
14: **end procedure**

Az eljárás minden egyes végrehajtásakor kiválaszt a paraméterül megkapott $C^{\ominus}$-ból egy $L^{\ominus}$ negatív literált a (2)-ben; miután az ezzel a literállal kapcsolatos műveleteket elvégezte, a (9)-ben található rekurzív híváskor a $C^{\ominus}$-ból törli az $L^{\ominus}$-t. Ez egészen addig megy, vagyis addig hívja az eljárás önmagát, míg $C^{\ominus}$-ból elfogynak a negatív literálok, melynek vizsgálata a (3)-ban foglal helyet; ha így történt, vagyis az aktuális $[E, \sigma]$ kiterjesztés konstruálásával eddig a végső pontig eljutottunk, akkor a konstruált kiterjesztést felvesszük a kiterjesztések halmazába, vagyis az $\mathcal{E}$-be, a (4)-ben. Ezen kiterjesztés az E-ben található literálok és a $\mathcal{C}^{\ominus}$-ban található (már csak pozitív) literálok egyesítésével áll elő, vagyis nem más, mint $[E \cup \mathcal{C}^{\ominus}, \sigma]$.

[1] A $\widehat{C}$ jelölés értelmezéséhez szintén lásd a 3.1.4. definíciót.
[2] Most már érthetjük, hogy az $\textsc{Ext}^{\ominus}$ eljárás (3) sorában miért $[\bot, \epsilon]$ paraméterrel hívjuk az Ext eljárást: „üres" kiterjesztésként inicializáljuk a konstruálandó kiterjesztéseket.
[3] Egy H halmaz esetén $H \leftarrow x$ alatt a $H := H \cup \{x\}$ műveletet értjük.

Ha sikerült $L^{\ominus}$-t találnunk, akkor az eljárásban a (6-12) dupla ciklus kerül végrehajtásra. A külső ciklus a $\mathcal{B} \cup \mathcal{C}$-ben található pozitív $C^{\oplus}$ klózokat veszi sorra (a 3.1.4.(2)-nek megfelelően), míg a belső ciklus a $\widehat{C}^{\oplus}$-ban szereplő $L^{\oplus}$ literálokat (a 3.1.4.(3)-nak megfelelően). A (8)-ban azt vizsgáljuk meg, hogy vajon az $L^{\ominus}$ és az $L^{\oplus}$ alapjai unifikálhatóak-e. Egészen pontosan: $L^{\ominus}\Theta\sigma$ és $L^{\oplus}\Theta\sigma$ alapjait próbáljuk unifikálni; ennek oka az, hogy $C^{\ominus}$ és $C^{\oplus}$ literáljait külön-külön, páronként próbáljuk egymással unifikálni, vagyis a konstruálás alatt álló kiterjesztésben szereplő σ helyettesítést kumulatív módon állítjuk elő[4].

Ha találunk két unifikálható literált, akkor a kiterjesztés konstruálása folytatódhat: meghívja az eljárás önmagát a (9)-ben. Mint már említettük, ezelőtt $C^{\ominus}$-ból töröljük $L^{\ominus}$-t. Továbbá a konstruálás alatt álló kiterjesztésünket továbbépítjük: az E-t kiterjesztjük a $\widehat{C}^{\oplus}$ literáljaival, kivéve az $L^{\oplus}$-t; valamint σ-hoz kompozícióval hozzávesszük a fentebb számolt MGU-t, melyet θ-val jelöltünk el.

Miután az EXT$^{\ominus}$ eljárás hívásával tetszőleges $\mathcal{B}$-re és $\mathcal{C}$-re elő tudjuk állítani $\mathcal{E}(\mathcal{B},\mathcal{C})$-t, minden készen áll, hogy leírjuk a multi-hipertabló kalkulust megvalósító MHT függvényt.

A.3.3. Algoritmus.

```
1: function MHT
2:     Θ := ϵ, B := {⊤}
3:     if CLOSE then return {x/t ∈ Θ | x ∈ Par(C)}
4:     else return „hamis"
5: end function
```

A függvény $\mathcal{C}$-hez konstruál multi-hipertablót, melyet két már említett globális objektum segítségével reprezentál: a Θ-val és a $\mathcal{B}$-vel; kezdetben mindkettő alaphelyzetben áll. Az MHT függvény a lent leírásra kerülő CLOSE segédfüggvényt hívja, mely az aktuális $\mathcal{B}$ ágat próbálja lezárni. Ha ez nem sikerül, akkor az MHT negatív válasszal tér vissza a (4)-ben; ha viszont sikerül, akkor a (3)-ban a Θ-ból kiválogatja a $\mathcal{C}$ paraméterváltozóira vonatkozó helyettesítéseket, és ezeket adja vissza a függvény.

A.3.4. Algoritmus.

```
1: function CLOSE
2:     E := ∅
3:     EXT^⊖(E)
4:     while E ≠ ∅ do
5:         E --min--> e   [5]
6:         Θ' := Θ
7:         if APPLY(e) then return „igaz"
8:         Θ := Θ'
9:     end while
```

[4] Tulajdonképpen az unifikációs algoritmus A.1.1.(11-18) disztribúciós egységének (14) sora búvik itt meg rejtetten.

[5] A $\mathcal{E} \xrightarrow{min} e$ alatt értsd: e az $\mathcal{E}$ egyik minimális (heurisztikájú) eleme, valamint $\mathcal{E} := \mathcal{E}\backslash\{e\}$.

10: **return** „hamis”
11: **end function**

A CLOSE függvény először a (2)-ben létrehoz egy $\mathcal{E}$ nevű változót (halmazt), melybe betölti az $\mathcal{E}(\mathcal{B},\mathcal{C})$ tartalmát a (3)-ban, az EXT$^{\ominus}$ segédeljárás hívásával. A (4-9) ciklusban ebből emelünk ki mindig egy *minimális* (legkisebb heurisztikájú) e kiterjesztést az (5)-ben, melyet alkalmazunk a $\mathcal{B}$ ágon. A kiterjesztés alkalmazását a lentebb leírásra kerülő APPLY függvény hívásával kivitelezzük a (7)-ben, mely előtt egy *visszaállítási pontot* hozunk létre (lásd: 3.5.1. fejezet) a (6)-ban. A visszaállítási pont nem más, mint az aktuális Θ lementése egy lokális változóba, a Θ'-be[6]. Az APPLY függvény – mint alább látni fogjuk – pontosan akkor tér vissza „igaz”-zal, ha a kiterjesztés elvégzése eredményeként $\mathcal{B}$-ből előálló ágak mindegyikét sikerült lezárni; ha mégsem, akkor a visszaállítási pontot felhasználva visszatöltjük a Θ-nak a kiterjesztés előtti tartalmát a (8)-ban, és továbblépünk a következő minimális heurisztikájú kiterjesztés alkalmazása felé. Ha mindegyik $\mathcal{E}$-beli kiterjesztés alkalmazása kudarccal végződött, azaz egyikkel sem sikerült $\mathcal{B}$-t lezárni, a függvény „hamis”-sal tér vissza.

Az utolsó segédfüggvény, az APPLY leírása következik. A függvény egy paramétert vár, egy $[E,\sigma]$ kiterjesztést, melyet az aktuális $\mathcal{B}$ ágon alkalmaz.

A.3.5. ALGORITMUS.

1: **function** APPLY($[E,\sigma]$)
2: $\Theta := \Theta\sigma$
3: **for all** $L \in E$ **do**
4: $\mathcal{B} \leftarrow L$
5: *continue* := CLOSE
6: $\mathcal{B} := \mathcal{B}\backslash\{L\}$
7: **if** nem *continue* **then return** „hamis”
8: **end for**
9: **return** „igaz”
10: **end function**

Először a függvény a σ helyettesítést veszi hozzá a Θ-hoz kompozícióval a (2)-ben. Ezután következik a $\mathcal{B}$-ből származtatott új ágaknak az előállítása a (3-8) ciklusban. Ennek keretében az E klóz összes literálját vesszük sorra, egyenként hozzáfűzve őket $\mathcal{B}$-hez a (4)-ben. Ezután a már ismert CLOSE függvényt hívjuk az időközben egy elemmel kibővített $\mathcal{B}$-re az (5)-ben, melynek kimenetét („igaz” vagy „hamis”) elraktározzuk a *continue* lokális változóban. Ha a *continue* értéke bármikor hamissá válik, azaz ha a CLOSE akár egy leszármazott ágra is „hamis”-sal tér vissza, akkor az APPLY azonnal negatív választ szolgáltat a (7)-ben; ha viszont E semelyik literáljára sem lépünk ki a függvényből „hamis”-sal, az azt jelenti, hogy a kiterjesztés sikeres volt, azaz $\mathcal{B}$

[6] Látható, hogy visszaállítási pontot nem csak a legszükségesebb esetekben, azaz paraméterváltozók behelyettesítésekor hozunk létre, hanem minden egyes alkalommal, mikor a tablón helyettesítést végzünk el. Mint a 3.5.1. fejezetben említettük, implementációs szinten nincs igazán értelme levezetésfolytonosan alkalmazni kalkulusunkat, a backtracking lesz az effektív implementáció általánosan használt eszköze.

összes leszármazott ágát lezártuk, tehát pozitív válasszal térünk vissza a (9)-ben. A *continue* értéke akár igaz vagy hamis, azaz akár lehet folytatni a leszármazott ágak lezárását, akár negatív válasszal kell visszatérnünk, a $\mathcal{B}$-hez legutoljára csatolt L literált eltávolítjuk a $\mathcal{B}$-ből a (6)-ban.

A jelen fejezetben tehát a multi-hipertabló kalkulus egy *rekurzív*, *backtrackingre épülő* megvalósítását írtuk le. A fő függvény, azaz az MHT módosítható lenne olyan irányban, hogy a függvény ne álljon meg az első keresett helyettesítés megtalálásakor, hanem ciklikus megvalósítás keretében folytassa a keresést az egyéb megoldások felé.

Irodalomjegyzék

[1] H. Andréka, I. Németi, J. van Benthem, „*Modal languages and bounded fragments of predicate logic*". Journal of Philosophical Logic, Vol. 27, p. 217-274, 1998.

[2] L. Bachmair, H. Ganzinger, „*Resolution Theorem Proving*". Handbook of Automated Reasoning, by J. A. Robinson and A. Voronkov, Vol. 1, Chapter 2, p. 19-99, Elsevier and MIT Press, 2001.

[3] P. Baumgartner, U. Furbach, I. Niemelä, „*Hyper Tableaux*". Lecture Notes in Computer Science, Vol. 1126, p. 1-17, 1996.

[4] P. Baumgartner, „*Hyper Tableaux – The Next Generation*". Lecture Notes in Artificial Intelligence, Vol. 1397, p. 60-76, 1998.

[5] F. M. Brown, „*Towards the Automation of Set Theory and its Logic*". Artificial Intelligence, Vol. 10, p. 281-316, 1978.

[6] F. Bry, A. Yahya, „*Positive Unit Hyperresolution Tableaux and Their Application to Minimal Model Generation*". Journal of Automated Reasoning, Vol. 25, p. 35-82, 2000.

[7] M. A. Castilho, L. F. del Cerro, O. Gasquet, A. Herzig, „*Modal Tableaux with Propagation Rules and Structural Rules*". Fundamenta Informaticae, Vol. 32, p. 281-297, 1997.

[8] C. L. Chang, R. C. T. Lee, „*Symbolic Logic and Mechanical Theorem Proving*". Academic Press, 1973.

[9] A. Church, „*A Note on the Entsheidungsproblem*". Journal of Symbolic Logic, Vol. 1, p. 40-41, 101-102, 1936.

[10] J. van Eijck, „*Constrained Hyper Tableaux*". Lecture Notes in Computer Science, Vol. 2142, p. 232-246, 2001.

[11] M. Fitting, „*First-Order Logic and Automated Theorem Proving*". Springer-Verlag, 1996.

[12] J. H. Gallier, „*Logic for Computer Science – Foundations of Automatic Theorem Proving*". Wiley & Sons, 2003.

[13] R. Hähnle, „*Tableaux and Related Methods*". Handbook of Automated Reasoning, by J. A. Robinson and A. Voronkov, Vol. 1, Chapter 3, p. 100-178, Elsevier and MIT Press, 2001.

[14] G. Kovásznai, C. Kotropoulos, I. Pitas, „*CAML – A Universal Configuration Language for Dialogue Systems*". Lecture Notes in Computer Science, Vol. 2736, p. 896 - 906, Springer, 2003.

[15] G. Kovásznai, C. Kotropoulos, I. Pitas, „*Partly-Specified Priority Patterns in Natural Language Parsing within Dialogue Systems*". 1st International Workshop on Interactive Rich Media Content Production: Architectures, Technologies, Applications, Tools (Richmedia 2003), p. 161-168, Lausanne, Switzerland, 2003.

[16] G. Kovásznai, C. Kotropoulos, I. Pitas, „*A Novel Universal Language for Configurating Dialogue Systems*". 9th Panhellenic Conference on Informatics (PCI '03), p. 36-45, Thessaloniki, Greece, 2003.

[17] G. Kovásznai, „*Algorithmic Improvements in Natural Language Parsing within Dialogue Systems: Priority Patterns and Wildcards*". 6th International Conference on Applied Informatics (ICAI 2004), Vol. 2, p. 129-138, Eger, Hungary, 2004.

[18] G. Kovásznai, „*HyperS Tableaux – Heuristic Hyper Tableaux*". Acta Cybernetica, Vol. 17, p. 325-338, 2005, (ZBL#1099.68096, MR#2183822).

[19] M. Kühn, „*Rigid Hypertableaux*". Lecture Notes in Artificial Intelligence, Vol. 1303, p. 87-98, 1997.

[20] R. Letz, K. Mayr, C. Goller, „*Controlled Intergration of the Cut Rule into Connection Tableau Calculi*". Journal of Automated Reasoning, Vol. 13, p. 297-328, 1994.

[21] A. Nonnengart, C. Weidenbach, „*Computing Small Clause Normal Forms*". Handbook of Automated Reasoning, by J. A. Robinson and A. Voronkov, Vol. 1, Chapter 6, p. 335-367, Elsevier and MIT Press, 2001.

[22] K. Pásztorné Varga, M. Várterész, „*A Generalized Approach to the Theorem Proving Methods*". 5th International Conference on Applied Informatics (ICAI 2001), p. 191-200, Hungary, 2001.

[23] M. S. Paterson, M. N. Wegman, „*Linear Unification*". Annual ACM Symposium on Theory of Computing, p. 181-186, 1976.

[24] J. Posegga, P. H. Schmitt, „*Automated Deduction with Shannon Graphs*". Journal of Logic and Computation, Vol. 5, p. 697-729, 1995.

[25] J. A. Robinson, „*A Machine-Oriented Logic Based on the Resolution Principle*". Journal of the ACM, Vol. 12, p. 23-41, 1965.

[26] J. A. Robinson, „*Automated Deduction with Hyper-Resolution*". International Journal of Computer Mathematics, Vol. 1, p. 227-234, 1965.

[27] T. Skolem, „*Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit matematischer Sätze nebst einem Theoreme über dichte Mengen*". Videnskabsakademiet i Kristiania, Skrifter I, No. 4, p. 1-36, 1919.

[28] R. M. Smullyan, „*First-Order Logic*". Springer-Verlag, 1968.