

A SAT probléma megoldása membránrendszerekkel

Kolonits Gábor, 5. évfolyam

Témavezető:
Dr. Gazdag Zsolt
egyetemi adjunktus



Budapest, Eötvös Loránd Tudományegyetem
Informatikai Kar, Algoritmusok és Alkalmazásaik Tanszék
2012

Tartalomjegyzék

1. Bevezetés	2
2. Előzmények	5
2.1. A membránrendszerek	5
2.2. A SAT probléma	7
2.3. Aktív membrános P rendszerek	10
2.4. Eldöntő P rendszerek	12
2.5. Egy ismert példa a SAT megoldására	13
2.5.1. A bemenet elkódolása	13
2.5.2. A membránrendszer	13
3. Eredmények	16
3.1. A fordított rezolúciós módszer	16
3.1.1. Az uniform megoldás	17
A bemenet elkódolása	17
A membránrendszer	17
3.1.2. A semi-uniform megoldás	20
A bemenet elkódolása	21
A membránrendszer	21
3.2. A három módszer összehasonlítása	24
4. Összefoglalás	28
Irodalomjegyzék	29

1. fejezet

Bevezetés

Az algoritmusok idő- és tárfelhasználás alapján történő hatékonyságának vizsgálata a bonyolultságelmélet központi témája. Általában egy problémát megoldó algoritmust hatékonynak nevezünk (az időigény szempontjából), ha az a bemenet méretével polinomiálisan arányos lépésszám után megáll. Az egyik legismertebb algoritmusmodell a Turing-gép, melyről kiderült, hogy minden olyan probléma hatékonyan megoldható vele, ami algoritmikusan hatékonyan megoldható. Létezik a problémák egy osztálya, az úgynevezett NP-teljes problémák, melyek megoldására nem ismert hatékony algoritmus. Ezen problémák viszont hatékonyan megoldhatók a Turing-gépek egy kiterjesztett változatával, az úgynevezett nemdeterminisztikus Turing-gépekkel. A hatékonyság kulcsa ezen gépek esetében tulajdonképpen az, hogy az ilyen gépek képesek párhuzamos számításokat végezni: a nemdeterminisztikus működésből fakadó összes számítási sorozaton egy időegység alatt végrehajtódik a gép következő lépése. Jelenlegi tudásunk szerint nem lehet az ilyen Turing-gépeket klasszikus algoritmusokkal hatékonyan modellezni. Az elmúlt néhány évtizedben azonban számos próbálkozás történt arra, hogy a természetben végbemenő folyamatokat modellezve új elvű, nagy számítási kapacitású modelleket alkossanak meg. Ilyenek például a DNS-számítógép, a kvantumszámítógép és ilyenek a dolgozatunk témáját szolgáltató membránrendszerek is.

A membránrendszerek olyan biológiai indíttatású számítási modellek, melyek az élő sejtekben végbemenő egyes folyamatokat formális nyelvi eszközökkel modellezik. A membránrendszereket a téma úttörője, Gheorghe Păun tiszteletére szokás P rendszereknek is nevezni. Ezek a rendszerek jellemzően hierarchikusan szervezett, úgynevezett membránok által határolt régiókból állnak, amelyek objektumok multihalmazát tartalmazzák. Gyakran nevezik magukat a régiókat membránoknak. Az objektumok a sejtekben előforduló egyes kémiai alkotóelemeknek, az objektumokon értelmezett átírási szabályok pedig a sejtekben végbemenő folyamatoknak felelnek meg. Rendelkezik egy globális időzítővel, melynek hatására a membránok képesek egyszerre, lépésről-lépésre dolgozni. A membránrendszerek működé-

sét nagyfokú párhuzamosság jellemzi.

A modell 1998-as megalkotása óta a P rendszerek számos változatát definiálták és vizsgálták. A modell egy sokat kutatott változata az úgynevezett aktív membrános P rendszer. Itt a membránrendszer rendelkezik azzal a tulajdonsággal, hogy képes az úgynevezett elemi membránok kettéosztására. Ez a tulajdonság a fent említett párhuzamossággal együtt alkalmassá teszi ezeket a rendszereket arra, hogy lineáris időben exponenciális számú membránt hozzanak létre és ezeken műveleteket végezzenek. Az ilyen rendszerek egyik sokat vizsgált alkalmazása az NP-teljes problémák, jellemzően a SAT probléma megoldása (lásd pl. [1, 2, 4, 6, 10, 13, 16]). A SAT probléma annak eldöntése, hogy egy ítéletkalkulusbeli formula kielégíthető-e vagy sem. A fent említett munkákban szereplő membránrendszerek képesek a bemenő formulák méretében lineáris időben eldönteni a SAT-ot, ahol a formula méretét a formulában szereplő változók és klózikok száma együttesen határozza meg (sőt, a [6] munkában a bemenet mérete csak a változók számától függ, de ott megengedett a nem elemi membránok szétosztása is). Fontos megjegyeznünk, hogy amikor azt mondjuk, hogy membránrendszerekkel hatékonyan eldönthető a SAT, akkor ez alatt csupán elméleti hatékonyságot értünk. A membránrendszer implementációja során ez a hatékonyság természetesen romlik. A témakörben fellelhető ötletes megoldások adták az ihletet, hogy én is ezzel a témával foglalkozzam. Szemben az eddigi megoldásokkal, témavezetőm javaslatára, az ítéletkalkulusbeli rezolúciót próbáltuk megvalósítani. Sikerült a SAT megoldására olyan újszerű módszert (a fordított rezolúciót) megvalósító membránrendszereket találni, amelyek a bemenő formula kielégíthetőségét képesek a formula változószámának függvényében lineáris időben eldönteni, függetlenül a formulában lévő klózikok számától. Ezen megoldások mindegyike egy-egy membránrendszer-családot definiál. Az első megoldás a [8] munkában jelent meg. Ez egy úgynevezett uniform megoldás, ahol minden n természetes számhoz megadunk egy $\Pi(n)$ membránrendszert, ami ezután képes minden legfeljebb n ítéletváltozót tartalmazó formula kielégíthetőségét n -ben lineáris időben eldönteni. A második megoldás az első megoldáson alapszik [9]. Ez egy semi-uniform megoldás, ahol minden formulához konstruálunk egy olyan membránrendszert, ami lineáris időben eldönti a formula kielégíthetőségét. A semi-uniform megoldás esetében természetesen követelmény, hogy a rendszer megkonstruálása a formula méretében hatékonyan történjen. Az uniform megoldás kidolgozásában nekem, a semi-uniform megoldás esetében pedig a témavezetőmnek volt meghatározó része.

A dolgozatban először ismertetjük a P rendszerek klasszikus típusát, majd felelevenítjük a szükséges ítéletkalkulusbeli ismereteket, definiáljuk a SAT problémát és megadjuk az általunk használt aktív membrános P rendszerek definícióját. Ezután ismertetjük a probléma egy már létező aktív membrános P rendszerrel történő megoldását. Végül bemutatjuk a téma általunk adott újszerű megközelítését és ezen új módszernek a fent említett, már létező

megoldással való összehasonlítását.

2. fejezet

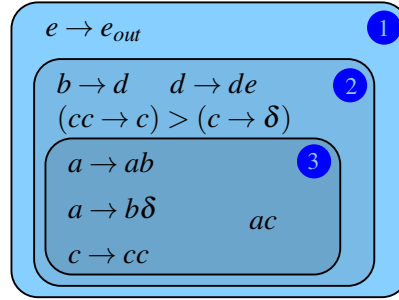
Előzmények

A következőkben először röviden bemutatjuk a membránrendszert, mint számítási modellt.

2.1. A membránrendszerek

A membránrendszer egy nemdeterminisztikus, erősen párhuzamos működésű, biológia ihletésű számítási modell [12]. Az alábbiakban röviden ismertetjük a klasszikus modellt (részletes leírás tekintetében ajánljuk a [14, 15] munkákat). Egy membránrendszer úgynevezett *membránok által határolt régiók* hierarchikusan felépített rendszere, amit a *környezet* vesz körül. Gyakran magukat a régiókat is membránoknak nevezzük. A legkülső membrán a *skin-membrán*; ez a membrán kommunikálhat közvetlenül a környezettel. Maguk a membránok, valamint a környezet is tartalmazhatják (többszörösen is) egy előre definiált, véges objektumhalmaz elemeit. A membránok része a bennük lévő objektumokon értelmezett *szabályok* véges halmaza. Minden membrán egy önálló számítási egység, ezen egységek közötti kommunikáció a szabályok segítségével biztosított. Ahogy említettük, egy objektum többször is előfordulhat egy membránban, így a membránok objektumok multihalmazait tartalmazzák. A membránokban szereplő multihalmazokat jellemzően az objektumok ábécéje feletti szavakkal adjuk meg. Természetesen, ha egy w szó egy membrán objektumainak multihalmazát jelöli, akkor a w betűinek tetszőleges permutálásával kapott szó is ugyanezt a multihalmazt jelöli. A rendszer egyik membránja kijelölhető input membránnak. Ekkor a rendszer futtatása előtt megadható az objektumok egy multihalmaza, amelyet ez a membrán - az előre definiált objektumokon kívül - tartalmazni fog.

A rendszer rendelkezik egy időzítővel, melynek egy lépése során a szabályok a membránban található egy vagy több megfelelő objektumból véges sok új objektumot hoznak létre (speciálisan törlő szabályok is megengedettek). Az, hogy melyik szabály kerül alkalmazásra, nemdeterminisztikusan dől el. Egy szabály egy lépésben többször is végrehajtható,



2.1. ábra. Egy membránrendszer grafikus ábrázolása (a membránokba beírtuk a hozzájuk tartozó szabályokat is)

amennyiben a baloldalán található objektumokból megfelelő számú a rendelkezésre áll.

A rendszer maximálisan párhuzamos, azaz egy membránban egy lépés során csak annyi és olyan objektum maradhat felhasználatlanul, amelyekre nem illeszthető szabály; tehát a fennmaradó objektumok az eredeti multihalmaz egy olyan részhalmazát alkotják, amelynek egyik (az adott membránhoz tartozó) szabály baloldala sem része. A fentiek demonstrálására tekintsünk egy egyszerű példát. Ha egy membrán három a -val, és két b -vel jelölt objektumot tartalmaz, a szabályai pedig olyanok, hogy az egyik két a -t ír át, a másik pedig egy a -t és egy b -t ír át, akkor kétféleképpen használódhatnak fel az objektumok: egyszer alkalmazzuk az első szabályt és egyszer a másodikat, ekkor marad egy b feldolgozatlanul; vagy kétszer alkalmazzuk a második szabályt, és akkor egy a marad.

A klasszikus membránrendszerekben a rendszer szerkezete, azaz a membránok hierarchiáját leíró gráf nem változik a rendszer működése során. Léteznek azonban olyan rendszerek, amelyekben az egyes membránok egy adott objektum feldolgozása közben megszűnhetnek vagy akár megduplázódhatnak. Az első esetben a fel nem használt objektumok a szülő membránba kerülnek, a másodikban pedig megduplázódnak és rendre a két új membránba kerülnek. Ezeken felül a szabályok jobb oldala meg lehet címkézve egy *in* vagy egy *out* címkével, ami azt jelzi, hogy az előállított objektumok nem maradhatnak a membránban, hanem távoznak belőle, vagy a szülő membránba, vagy valamelyik gyerek membránba (modelltől függően vagy nemdeterminisztikusan dől el, hogy melyik gyerekbe kerülnek az objektumok vagy ezt további címkézés határozza meg). Végül lehetőség van a szabályok közt sorrendet felállítani, ami azt jelenti, hogy a „gyengébb” szabály csak akkor alkalmazható, ha az „erősebb” már nem.

A rendszer addig folytatja működését, amíg van alkalmazható szabály. Ellenkező esetben a rendszer leáll és ekkor a környezetben található objektumok alkotják a működés eredményét, azaz a kimenetet.

A fentiek szemléltetésére tekintsük a 2.1. ábrán grafikusán ábrázolt membránrendszert

[17]. Ennek három membránja van, ezek közül csak a 3-as címkéjűben vannak kezdetben elemek, így eleinte csak ez fog dolgozni. A rendszer egy lehetséges működése a következő:

- Első lépésben a hármas membrán első ($a \rightarrow ab$) és harmadik ($c \rightarrow cc$) szabályát alkalmazzuk. A lépés után a membrán tartalma $abcc$ lesz.
- Ezután alkalmazzuk a membrán második ($a \rightarrow b\delta$) és harmadik szabályát, az utóbbit kétszer. A második szabály „feloldó”-szabály (ezt jelöli a δ szimbólum). Ennek alkalmazása megszünteti a membránt, ezért annak tartalma ($b\delta ccc$) az öt tartalmazó kettes membránba fog átkerülni.
- A harmadik lépésben a kettes membránban előáll két d ($b \rightarrow d$ szabály alkalmazása kétszer) valamint két c ($cc \rightarrow c$ szabály kétszer). (A $c \rightarrow \delta$ szabályt nem alkalmazhatjuk, mert mind a négy c -t „elfogyasztja” az „erősebb” $cc \rightarrow c$.) Tehát a kettes címkéjű membrán tartalma a következő lesz: $ccdd$.
- Ezután a két d -ből lesz két d és két e ($d \rightarrow de$ szabály kétszer), a két c -ből pedig egy c lesz (vagyis a kettes membrán tartalma $cddee$ lesz).
- A következő lépésben a két d -ből ismét két d és két e lesz, a c pedig törlődik és a membrán megszűnik ($c \rightarrow \delta$ szabály). Így az egyes membrán tartalma $ddeeee$ lesz.
- Az utolsó lépésben az egyes membrán a négy darab e -t kiküldi a környezetbe. Ezután már csak két d marad, melyekre nincs alkalmazható szabály, ezért a rendszer leáll.

Könnyű belátni, hogy ez a membránrendszer n^2 darab e -t állít elő a környezetben, ahol n az a szám, ahányszor a hármas membránban az $a \rightarrow ab$ szabály végrehajtódik az $a \rightarrow b\delta$ szabály alkalmazása előtt.

Nem meglepő, hogy a membránrendszerek számítási ereje a Turing-gépekével ekvivalens. A dolgozatban viszont mi arra fogunk koncentrálni, hogyan lehet ezeket a rendszereket a gyakorlatban nehezen megoldható SAT probléma (elméletileg) hatékony megoldására használni.

2.2. A SAT probléma

A SAT probléma (az elnevezés az angol satisfiability - kielégíthetőség - szóból származik) zérusrendű logikai formulák kielégíthetőségének az eldöntése. Jelen dolgozatban a SAT probléma azon változatát vizsgáljuk, amelyben a bemenetként kapott formula konjunktív normálformában van. A továbbiakhoz szükségünk lesz az alábbi definíciókra (a téma részletes

leírásának tekintetében ajánljuk a [11] munkát). Legyen $X := \{x_1, x_2, x_3, \dots\}$ egy megszámlálhatóan végtelen halmaz, melynek elemeit *ítéletváltozóknak* (vagy röviden *változónak*) nevezzük, és minden $n \geq 0$ -ra legyen $X_n := \{x_1, \dots, x_n\}$. *Literálnak* nevezzük egy $x \in X$ változót vagy annak $\bar{x}$ tagadását. Ekkor az x -et a *literál alapjának* nevezzük. *Klóznak* nevezzük véges sok különböző alapú literál diszjunkcióját. Speciálisan, ha egy klóz nem tartalmaz egyetlen literált sem, akkor ő az *üres klóz*, amit a $\square$ jellel jelölünk. Azt mondjuk, hogy egy φ formula *konjunktív normálformában* (röviden KNF-ben) van, ha a φ m darab klóz konjunkciója valamely $m \geq 1$ számra. A továbbiakban formula alatt mindig olyan formulát értünk ami KNF-ben van. Az X_n -beli változók feletti összes lehetséges formula illetve klóz halmazát rendre $Form_n$ -nel és $\mathbb{C}_n$ -nel jelöljük. Egy $\mathbb{C}_n$ -beli C klózt *teljesnek* nevezzük, ha minden X_n -beli x változóra, x vagy $\bar{x}$ szerepel a C -ben. Egy φ X_n -feletti formulára azt mondjuk, hogy *kitüntetett konjunktív normálformában* (KKNF-ben) van, ha φ -ben minden klóz teljes. A továbbiakban sokszor úgy fogunk tekinteni egy KNF-ben lévő formulára, mint klózek halmazára, a klózekra pedig úgy, mint literálok halmazára. Ekkor egy φ formula változóinak a halmaza a következőképpen definiálható: $var(\varphi) := \{x \in X \mid \exists C \in \varphi : x \in C \vee \bar{x} \in C\}$.

Azt mondjuk, hogy egy X_n -feletti φ formula *kielégíthető*, ha van az X_n -beli változóknak egy olyan $I : X_n \rightarrow \{igaz, hamis\}$ *interpretációja*, melyre $I \models \varphi$, azaz I -ben kiértékelve φ -t *igaz*-at kapunk. A fent definiált üres klóz definíció szerint kielégíthetetlen. Világos, hogy egy φ formula pontosan akkor kielégíthető, ha van olyan interpretáció, ami az összes φ -beli klózt kielégíti. Egy I interpretáció kielégít egy F formulahalmazt ($I \models F$), ha I kielégíti az F összes elemét. Azt mondjuk, hogy egy φ formula *tautologikus következménye* egy F formulahalmaznak (jele: $F \models \varphi$), ha minden olyan interpretáció, ami kielégíti az F -et, kielégíti a φ -t is. Ezek után a SAT probléma a következőképpen definiálható: *Adott egy φ formula. Döntsük el, hogy φ kielégíthető-e.*

Legyen φ_1 és φ_2 két $Form_n$ -beli formula. Azt mondjuk, hogy φ_1 és φ_2 ekvivalensnek (jele: $\varphi_1 \sim \varphi_2$), ha minden $I : X_n \rightarrow \{igaz, hamis\}$ interpretációra, φ_1 értéke akkor és csak akkor *igaz* az I -ben, ha φ_2 értéke is *igaz*.

Egy X_n -beli változók feletti φ formula kielégíthetőséget többféleképpen el lehet dönteni. Az alábbiakban vázoljuk azt a két módszert, amit a dolgozatban ismertett membránrendszerekben is használunk. Az egyik módszer az, amikor sorra megvizsgáljuk az X_n -beli változók összes lehetséges interpretációját. Ha találunk olyat ami kielégíti φ -t, akkor φ kielégíthető, egyébként pedig nem az. Nevezzük ezt a módszert *interpretációs módszernek*.

A második módszer leírásához szükségünk van némi előkészületre. Egy $C \in \mathbb{C}_n$ klóz és egy $Y \subseteq X_n$ esetén, amennyiben $var(C) \cap Y = \emptyset$, definiáljuk a C_Y klózhalmazt a következő-

képpen. Tegyük fel, hogy $Y = \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}$ ($k \leq n, 1 \leq i_1 < i_2 < \dots < i_k \leq n$). Akkor

$$C_Y := \{C \cup \{l_1, l_2, \dots, l_k\} \mid 1 \leq j \leq k : l_j \in \{x_{i_j}, \bar{x}_{i_j}\}\}.$$

Továbbá, egy $\varphi \in \text{Form}_n$ formulára legyen $\varphi' := \bigcup_{C \in \varphi} C_Y$, ahol $Y = X_n - \text{var}(C)$. A formulák kielégíthetőségének eldöntésére használt módszerünk a következő állításon alapul:

1. Állítás. Legyen $\varphi \in \text{Form}_n$ ($n \geq 0$). A φ' akkor és csak akkor tartalmazza az összes $\mathbb{C}_n$ -beli teljes klózt, ha φ kielégíthetetlen.

Bizonyítás. A fenti állítást két lépésben bizonyítjuk be. Először megmutatjuk, hogy $\varphi \sim \varphi'$, utána pedig azt, hogy φ' akkor és csak akkor kielégíthetetlen, ha tartalmazza az összes $\mathbb{C}_n$ -beli teljes klózt.

$\varphi \sim \varphi'$ bizonyításához megmutatjuk, hogy minden I interpretációra, $I \models \varphi$ akkor és csak akkor teljesül, ha $I \models \varphi'$. Legyen I egy interpretáció és tegyük fel, hogy $I \models \varphi$. Tekintsünk egy $C \in \varphi$ klózt. Mivel $I \models C$ és minden $C' \in C_Y$ -re ($Y = X_n - \text{var}(C)$), $\text{var}(C) \subseteq \text{var}(C')$, kapjuk, hogy $I \models C_Y$. Ebből viszont következik, hogy $I \models \varphi'$ is teljesül.

Most tegyük fel, hogy $I \models \varphi'$. Elég megmutatnunk, hogy minden $C \in \varphi$ -re, $I \models C$. Legyen $C \in \varphi$ és $Y := X_n - \text{var}(C) = \{x_{i_1}, \dots, x_{i_k}\}$ ($k \leq n, 1 \leq i_1 < \dots < i_k \leq n$). Defináljuk a C' klózt a következőképpen: $C' := C \cup \{l_{i_1}, \dots, l_{i_k}\}$, ahol minden $1 \leq j \leq k$ esetén $l_{i_j} = \bar{x}_{i_j}$ ha $I(x_{i_j}) = \text{igaz}$ és $l_{i_j} = x_{i_j}$ különben. Világos, hogy $C' \in C_Y$ és $I \not\models \{l_{i_1}, \dots, l_{i_k}\}$. Ekkor viszont, mivel $I \models C'$, azt kapjuk, hogy $I \models C$ -nek teljesülnie kell.

Most megmutatjuk, hogy φ' akkor és csak akkor kielégíthetetlen, ha tartalmazza az összes $\mathbb{C}_n$ -beli teljes klózt. Először tegyük fel, hogy φ' tartalmaz minden $\mathbb{C}_n$ -beli teljes klózt. Legyen I egy tetszőleges interpretáció X_n változói felett. Legyen továbbá $C' = \{l_1, \dots, l_n\}$ a φ' azon klóza, amelyre teljesül, hogy minden $1 \leq i \leq n$ esetén $l_i = \bar{x}_i$ ha $I(x_i) = \text{igaz}$ és $l_i = x_i$ különben. Világos, hogy $I \not\models C'$, ami, mivel $C' \in \varphi'$, azt jelenti, hogy $I \not\models \varphi'$. Kapjuk tehát, hogy φ' kielégíthetetlen.

Most tegyük fel, hogy φ' nem tartalmaz minden teljes klózt, és legyen $C' := \{l_1, \dots, l_n\}$ egy teljes klóz, ami nem fordul elő φ' -ben. Tekintsük a következőképpen definiált I interpretációt: minden $1 \leq i \leq n$ esetén $I(x_i) := \text{igaz}$ ha $l_i = \bar{x}_i$ és $I(x_i) := \text{hamis}$ különben. Könnyen belátható, hogy minden $C \in \varphi'$ klózban van olyan l literál, amire $I(l) = \text{igaz}$. Ebből következik, hogy I kielégít minden klózt φ' -ben. Kapjuk tehát, hogy φ' kielégíthető és ezzel az állításunk bizonyítását befejeztük. $\square$

Nyilvánvaló, hogy tetszőleges $\varphi \in \text{Form}_n$ ($n \geq 0$) formulára, φ' egy KKNF-ben lévő formula, továbbá, ahogy a fenti állítás bizonyításában láttuk, $\varphi \sim \varphi'$. A fenti állítás tehát azt mondja ki, hogy a φ kielégíthetősége attól függ, hogy a vele ekvivalens KKNF-ben lévő

formula tartalmazza-e az X_n -beli változókból felépíthető összes teljes klózt. Azt a módszert, ami a φ kielégíthetőségét úgy dönti el, hogy először meghatározza φ' -t, majd eldönti, hogy φ' tartalmazza-e az összes $\mathbb{C}_n$ -beli összes teljes klózt *fordított rezolúciós módszernek* fogjuk nevezni az alábbiak miatt. A φ' megkonstruálható a φ -ből az ítétekalkulusbeli rezolúcióban használt ún. rezolúciós lépés bizonyos értelemben vett megfordításával, azaz egy „fordított rezolúciós” lépéssel. Maga a rezolúciós lépés a következőképpen írható le. Legyen C_1 és C_2 két klóz melyekben szerepel pontosan egy komplement literálpár, azaz $C_1 = C'_1 \cup \{l\}$ és $C_2 = C'_2 \cup \{\bar{l}\}$ valamely C'_1 és C'_2 klózokra és $l, \bar{l}$ literálokra. Ekkor a C_1 és C_2 klózok *rezolvense* az $R := C'_1 \cup C'_2$ klóz. A rezolúciós tétel szerint $\{C_1, C_2\} \models R$. Másrészt, az is könnyen belátható, hogy ha R kielégíthető, akkor $\{C_1, C_2\}$ is kielégíthető. A fordított rezolúciós lépés tehát definiálható úgy, mint a következő művelet. Legyen R egy klóz és x egy R -ben nem szereplő ítéltváltozó. Legyen továbbá $C_1 := R \cup \{x\}$ és $C_2 := R \cup \{\bar{x}\}$. Ekkor azt mondjuk, hogy C_1 és C_2 az R -ből kapható fordított rezolúcióval.

Világos, hogy egy tetszőleges $\varphi \in \text{Form}_n$ ($n \geq 0$) esetén, a φ' formula definíciójában a fordított rezolúciós lépést alkalmaztuk véges sokszor. Az általunk megadott, SAT-ot megoldó membránrendszerekben az fordított rezolúciós módszert fogjuk implementálni. Ahogy említettük, a felhasznált P rendszer típus az aktív membrános P rendszer lesz.

2.3. Aktív membrános P rendszerek

Az aktív membrános P rendszer egy faszerkezetű membránrendszer [13]. Erőssége, hogy a membránok képesek osztódni, így lineáris időben képesek létrehozni exponenciális munkaterületet, amelyen utána párhuzamosan tudnak dolgozni - ebből kifolyólag jellemzően NP teljes problémák (elméletileg) hatékony megoldása kapcsán vizsgálják az ilyen rendszereket. Minden membránnak van egy véges címkehalmazból vett címkéje. A membránoknak lehet továbbá egy úgynevezett polaritása, amely a $+, -, 0$ értékek valamelyike lehet és változhat a számítás során. Ezen kívül a rendszer minden szabályára teljesül, hogy a baloldalán csak egyetlen objektum szerepel. Ezen rendszerek formális definíciójának megértéséhez szükség lehet némi formális nyelvi alapismeretre, melynek tekintetében ajánljuk az [5, 7] munkákat. Formálisan, egy *aktív membrános P rendszer* egy olyan $\Pi = (O, H, \mu, w_1, \dots, w_m, R)$ rendszer, ahol

- $m \geq 1$ a rendszer *foka* (kezdetben a rendszerben szereplő membránok száma);
- O az *objektumok ábécéje*;
- H a membránok *címkéinek* halmaza;

- μ egy m darab, H elemeivel címkézett membránból álló *membrán struktúra* (a konvenció szerint μ -t H elemeivel címkézett, egymásba ágyazott $[]$ zárójel párokkal szokás megadni);
- $w_1, \dots, w_m$ O -feletti szavak, melyek rendre az $1, \dots, m$ címkékkel jelölt membránokban kezdetben szereplő O -beli objektumok multihalmazait jelölik (ezen szavak minden betűje az adott objektum egy előfordulását jelöli a megfelelő membránban);
- R a következő típusú szabályok halmaza:
 1. Átíró szabályok: $[a]_h^e \rightarrow [v]_h^e$, ahol $a \in O$, $v \in O^*$, h a membrán címkéje, e pedig a membrán polaritása. Az ilyen szabályok egy membránon belül hajtódnak végre, és csak annak tartalmát változtatják.
 2. Beviteli szabályok: $a[]_h^{e_1} \rightarrow [b]_h^{e_1}$, ahol $a, b \in O$, h a membrán címkéje, e_1 és e_2 pedig polaritások. Az ilyen szabályok alkalmazása esetén az a objektum kívülről bekerül a h címkéjű membránba, közben megváltozhat, és megváltoztathatja a membrán polaritását.
 3. Kiviteli szabályok: $[a]_h^{e_1} \rightarrow []_h^{e_2} b$, ahol $a, b \in O$, h a membrán címkéje, e_1 és e_2 pedig polaritások. Az ilyen szabályok alkalmazása esetén az a objektum kikerül a h címkéjű membránból annak szülőjébe, közben megváltozhat, és megváltoztathatja az adott membrán polaritását.
 4. Törlő szabályok: $[a]_h^e \rightarrow b$, ahol $a, b \in O$, h a membrán címkéje, e pedig a membrán polaritása. A szabály végrehajtása közben a a -t tartalmazó h címkéjű membrán megszűnik és a tartalma a szülő membránba kerül; közben az a megváltozhat.
 5. Osztó szabályok: $[a]_h^{e_1} \rightarrow [b]_h^{e_2} [c]_h^{e_3}$, ahol $a, b, c \in O$, h a membrán címkéje, e_1 , e_2 és e_3 pedig polaritások. A szabály alkalmazása során az eredeti h címkéjű membránból két új jön létre, melyekben ott lesz az eredeti membrán tartalma, kivéve az a -t, amely helyett az egyikben b lesz, a másikban pedig c . Osztó szabály csak elemi membránokra alkalmazható, azaz olyan membrán, amely nem tartalmaz másik membránt.

Megjegyezzük, hogy a fenti szabályok mindegyike csak akkor alkalmazható egy adott membránra, ha a membrán polaritása megegyezik a szabály baloldalán szereplő polaritással.

A rendszer a következő alapelvek szerint működik:

- Először az objektumokhoz hozzárendeljük a rájuk alkalmazható szabályok egy maximális multihalmazát, figyelembe véve, hogy egy membrán legfeljebb egy 2-5. típusú szabály végrehajtásában vehet részt.

- Ezután először párhuzamosan végrehajtnak a kiválasztott 1. típusú szabályok, majd pedig a többi szabály.

A rendszer működésére a későbbiekben látunk konkrét példát is (1. Példa). Munkánk során az aktív membrános P rendszerek egy olyan változatát fogjuk használni, mely alkalmas eldöntési problémák megoldására.

2.4. Eldöntő P rendszerek

Ebben a részben leírjuk, hogy a membránrendszerek elméletében, és így a dolgozatunkban is, hogyan használjuk az aktív membrános P rendszereket problémák eldöntésére. Defináljuk továbbá, hogy mit értünk az alatt, hogy ilyen membránrendszerekkel lineáris időben megoldható a SAT probléma. Ehhez először az alábbi előkészületeket kell tennünk. Egy aktív membrános $\Pi = (O, H, \mu, w_1, \dots, w_m, R)$ P rendszert *eldöntő P rendszernek* nevezünk, ha az alábbiak teljesülnek:

- O -ban van két kitüntetett elem, a *yes* és a *no*;
- Π minden számítása megáll, és a *yes* vagy *no* szimbólumok egyikét küldi a környezetbe.

Azt mondjuk, hogy Π egy *bemeneti membrános* eldöntő P rendszer, ha

- rendelkezik egy i_0 bemeneti membránnal és
- Π kezdőkonfigurációja az i_0 membránban tartalmazza a bemenetet kódoló szót.

A leglényegesebb különbség a bemeneti membrános és az anélküli eldöntő P rendszerek között a következőképpen írható le. A bemenet nélküli P rendszerekben a probléma egy adott példányához megkonstruált rendszer tartalmazza elkódolva magát a bemenetet is, míg a bemeneti membrános P rendszerekben a rendszert egy adott $n \in \mathbb{N}$ számhoz konstruáljuk meg és a legfeljebb n méretű bemenetet kódoló szót a rendszer bemeneti membránjában helyezzük el.

Azt mondjuk, hogy egy Π (bemeneti membrános) eldöntő P rendszer *determinisztikus*, ha Π -nek a kezdőkonfigurációból (adott bemenettel az i_0 membránban) indulva csak egyetlen számítási sorozata van a rendszer megállási konfigurációjába. A továbbiakban csak determinisztikus membránrendszerekkel foglalkozunk.

Eldöntési problémák megoldására membránrendszer családokat használunk. Legyen $\Pi := (\Pi(i))_{i \in \mathbb{N}}$ bemeneti membrános eldöntő P rendszerek egy családja. Azt mondjuk, hogy Π

(*polinomiálisan*) *uniform*, ha minden $n \in \mathbb{N}$ esetén, $\Pi(n)$ megkonstruálható egy determinisztikus Turing-géppel (n -ben polinom időben). Hasonlóan, legyen $\Pi := (\Pi(\varphi))_{\varphi \in \text{Form}}$ eldöntő P rendszerek egy családja. A Π -t (*polinomiálisan*) *semi-uniformnak* nevezzük, ha minden $\varphi \in \text{Form}$ esetén, $\Pi(\varphi)$ megkonstruálható determinisztikus Turing-géppel (φ méretében polinom időben).

2.5. Egy ismert példa a SAT megoldására

Ebben a részben ismertetünk egy olyan aktív membrános P rendszert, ami a témakörben szokásos módon, az interpretációs módszeren alapulva dönti el a SAT problémát. A P rendszer megadásakor az [3] munkában leírtakat követjük.

2.5.1. A bemenet elkódolása

Világos, hogy amikor egy formulát bemenetként adunk egy membránrendszernek, akkor a formulát a rendszer számára „olvashatóvá” kell tenni. Ez azt jelenti, hogy a formulát el kell kódolni egy megfelelő ábécé feletti szóban. Az is nyilvánvaló, hogy a formula elkódolása hatékony kell hogy legyen, azaz nem fordulhat elő az, hogy az elkódolás során a formula kielégíthetőségét is kiszámítjuk. A továbbiakban ilyen, a formula méretében polinom idejű elkódolásokat fogunk használni.

Legyen $\varphi = \{C_1, C_2, \dots, C_m\} \in \text{Form}_n$ ($m \geq 1, n \geq 0$) egy KNF-ben lévő formula. Legyen továbbá minden $1 \leq i \leq m$ -re $C_i = \{y_{i,1}, y_{i,2}, \dots, y_{i,k_i}\}$, ahol $k_i \geq 0$. Az (n, m) paraméterrel rendelkező φ formulánkat az alábbi módon kódoljuk el:

$$\text{cod}(\varphi) := \bigcup_{i=1}^m (\{x_{i,j} : x_j \in C_i\} \cup \{\bar{x}_{i,j} : \bar{x}_j \in C_i\}).$$

2.5.2. A membránrendszer

Tetszőleges $m \geq 1, n \geq 0$ esetén legyen $\Pi(\langle n, m \rangle) = (O, H, \mu, w_1, w_2, R)$, ahol

- $O = \{x_{i,j}, \bar{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{c_k : 1 \leq k \leq m+2\} \cup \{d_k : 1 \leq k \leq 3n+2m+3\} \cup \{r_{i,k} : 0 \leq i \leq m, 1 \leq k \leq 2n\} \cup \{e, t\} \cup \{\text{yes}, \text{no}\};$
- $H = \{1, 2\};$
- $\mu = [[]_2]_1$ (a kezdeti membránstruktúra);
- $w_1 = \lambda, w_2 = d_1$ (a membránok tartalma kezdetben, ahol a λ , ami az üres szót jelöli, az üres multihalmaznak felel meg);

- Az input membrán a kettes címkéjű membrán, benne a bemenet (azaz $cod(\varphi)$ valamely $\varphi(m, n)$ paraméterekkel rendelkező formulára);
- R a következő átírási szabályok halmaza:

- (a) $\{[d_k]_2^0 \rightarrow [d_k]_2^+[d_k]_2^- : 1 \leq k \leq n\},$
- (b) $\{[x_{i,1} \rightarrow r_{i,1}]_2^+, [\bar{x}_{i,1} \rightarrow r_{i,1}]_2^- : 1 \leq i \leq m\},$
 $\{[\bar{x}_{i,1} \rightarrow \lambda]_2^+, [x_{i,1} \rightarrow \lambda]_2^- : 1 \leq i \leq m\},$
- (c) $\{[x_{i,j} \rightarrow x_{i,j-1}]_2^+, [x_{i,j} \rightarrow x_{i,j-1}]_2^- : 1 \leq i \leq m, 2 \leq j \leq n\},$
 $\{[\bar{x}_{i,j} \rightarrow \bar{x}_{i,j-1}]_2^+, [\bar{x}_{i,j} \rightarrow \bar{x}_{i,j-1}]_2^- : 1 \leq i \leq m, 2 \leq j \leq n\},$
- (d) $\{[d_k]_2^+ \rightarrow []_2^0 d_k, [d_k]_2^- \rightarrow []_2^0 d_k : 1 \leq k \leq n\},$
 $\{d_k []_2^0 \rightarrow [d_{k+1}]_2^0 : 1 \leq k \leq n-1\},$
- (e) $\{[r_{i,k} \rightarrow r_{i,k+1}]_2^0 : 1 \leq i \leq m, 1 \leq k \leq 2n-1\},$
- (f) $\{[d_k \rightarrow d_{k+1}]_1^0 : n \leq k \leq 3n-3\}, [d_{3n-2} \rightarrow d_{3n-1}e]_1^0,$
- (g) $e []_2^0 \rightarrow [c_1]_2^+, [d_{3n-1} \rightarrow d_{3n}]_1^0,$
- (h) $\{[d_k \rightarrow d_{k+1}]_1^0 : 3n \leq k \leq 3n+2m+2\},$
- (i) $[r_{1,2n}]_2^+ \rightarrow []_2^- r_{1,2n},$
- (j) $\{[r_{i,2n} \rightarrow r_{i-1,2n}]_2^- : 1 \leq i \leq m\},$
- (k) $r_{1,2n} []_2^- \rightarrow [r_{0,2n}]_2^+,$
- (l) $\{[c_k \rightarrow c_{k+1}]_2^- : 1 \leq k \leq m\},$
- (m) $[c_{m+1}]_2^+ \rightarrow []_2^+ c_{m+1},$
- (n) $[c_{m+1} \rightarrow c_{m+2}t]_1^0,$
- (o) $[t]_1^0 \rightarrow []_1^+ t,$
- (p) $[c_{m+2}]_1^+ \rightarrow []_1^- yes,$
- (q) $[d_{3n+2m+3}]_1^0 \rightarrow []_1^+ no.$

A rendszer működése az alábbi négy, egymást követő szakasszal írható le:

- A generálási szakaszban a belső (kettes címkéjű) membránokban előáll az összes lehetséges interpretáció az osztódási szabályok segítségével. Ezzel egy időben az is eldől, hogy az egyes klózekat kielégíti-e az adott interpretáció. Ebben a szakaszban csak (a)-(e) szabályok vesznek részt; a szakasz $3n - 1$ lépésig tart.

- A szinkronizációs szakasz célja, hogy az $r_{i,k}$ objektumok második indexei mind $2n$ -nel legyenek egyenlők. Ebben a szakaszban (e)-(g) szabályok vesznek részt; a szakasz $2n$ lépésig tart.
- Az ellenőrzési szakaszban annak megállapítása a cél, hogy hány és mely klózik értékelődtek igazra a belső membránokban. Ez klózonként két lépést igényel, összesen tehát $2m$ lépést, és a (h)-(l) szabályok játszanak szerepet benne.
- Az output szakaszban a rendszer „megállapítja”, hogy volt-e olyan interpretáció, amely minden klózt igazra értékelt ki (azaz a neki megfelelő membránban előállt-e a c_{m+1} -es objektum az előző szakaszban). Ha van ilyen, akkor egy *yes* objektum gyártódik le és kerül ki a környezetbe. Ellenkező esetben egy *no* objektum jön létre és kerül ki a környezetbe. Ez a szakasz négy lépésből áll, és az (m)-(q) szabályok vesznek részt benne, valamint a (h)-ba tartozó $[d_{3n+2m+2} \rightarrow d_{3n+2m+3}]_1^0$ szabály.

A fentiek alapján belátható, hogy $\Pi(\langle n, m \rangle)$ pontosan akkor küldi a környezetbe a *yes* szimbólumot, amikor a bemenetként kapott φ formula kielégíthető. Mivel $\Pi(\langle n, m \rangle)$ minden $m \geq 1, n \geq 0$ esetén polinom időben megkonstruálható, kapjuk a következő tételt:

1. Tétel. [3] *A SAT eldönthető bemeneti membrános eldöntő rendszerek egy polinomiálisan uniform családjával a bemenő formula méretében polinom időben, ahol a bemenet mérete a benne szereplő klózik és a felhasznált ítéleváltozók számának összege.*

3. fejezet

Eredmények

Az előző fejezetben ismertettünk egyet az irodalomban létező SAT problémát megoldó membránrendszerek közül. Ebben a részben ismertetünk két általunk definiált membránrendszert. Az első megoldás egy uniform megoldás, de nem polinomiálisan uniform. A másik egy polinomiálisan semi-uniform megoldás. Mindkét módszer a fordított rezolúciós módszert valósítja meg, és mindkét módszernek van létjogsáltusága. Az uniform megoldás esetében, egy adott $n \in \mathbb{N}$ -hez megkonstruált membránrendszer n -ben exponenciális méretű objektum-ábécével rendelkezik, mivel egy n változós formulával ekvivalens KKNF-ben lévő formula n -ben exponenciálisan sok klózt tartalmazhat. Másrészt viszont, a megkonstruált membránrendszer bármely n változós formula esetén n -ben lineáris lépésben el tudja dönteni a formula kielégíthetőségét. A semi-uniform megoldás a uniform megoldáson alapul. Itt egy adott formulából konstruáljuk meg az eldöntő membránrendszert. A rendszer működése során nem gyártjuk le az ekvivalens KKNF-ben lévő formula klózeit, hanem az eredeti formula klózeit használjuk fel megfelelően.

3.1. A fordított rezolúciós módszer

Mint korábban említettük, a fordított rezolúciós módszer a fordított rezolúciós lépésen alapszik: ha egy $\varphi \in Form_n$ ($n \geq 0$) formulában van egy klóz, ami nem tartalmazza az összes X_n -beli változót, akkor ez a klóz lecserélhető két másikra, melyeket úgy kapunk meg, hogy egy, a formulában nem szereplő X_n -beli változót hozzáveszünk a klózhoz, egyszer negálva, egyszer negálatlanul. Ezzel egy, az eredetivel formulával ekvivalens KNF-ben lévő formulát kapunk. Ezt addig ismételjük, amíg minden klózunk teljes nem lesz. Ekkor, az 1. állítás alapján a formula pontosan akkor kielégíthető, ha a klózainak száma 2^n .

Az ezen módszeren alapuló membránrendszerek szabályai két dologban térnek el az előző fejezetben látott rendszerek szabályaitól: polarizáció mentesek, azaz a membránoknak nincs

polaritásuk, és megjelenik egy új fajta osztó szabály:

- $[]_{h_1} \rightarrow [K]_{h_2}[O - K]_{h_3}$ ahol h_1 , h_2 és h_3 címkék, O az objektumok halmaza, K pedig ennek egy valódi részhalmaza (lásd pl. [14]).

Ezek a szabályok az eredeti membrán elemeit kétfelé osztják: az egyik membránba kerülnek azok az objektumok, amelyek elemei a K halmaznak, a másikba pedig a többi.

3.1.1. Az uniform megoldás

A fordított rezolúciós módszert először egy uniform membránrendszer-családdal implementáljuk.

A bemenet elkódolása

Tetszőleges $n \geq 0$ esetén legyen O_n egy $\mathbb{C}_n$ -nel azonos elemszámú ábécé, és legyen $f : O_n \rightarrow \mathbb{C}_n$ egy kölcsönösen egyértelmű leképezés. Egy $c \in O_n$ elemhez tartozó $f(c) \in \mathbb{C}_n$ klózt a könnyebb olvashatóság kedvéért a $\hat{c}$ szimbólummal fogjuk jelölni. Ekkor egy $\varphi = \{C_1, C_2, \dots, C_m\}$ formulát az a $\text{cod}(\varphi) := \{c_1, c_2, \dots, c_m\}$ halmaz kódolja, melyre igaz, hogy minden $1 \leq i \leq m$ -re $\hat{c}_i = C_i$.

A membránrendszer

Tetszőleges $n \geq 0$ esetén legyen $\Pi(n) := (O, H, \mu, w_1, w_2, w_3, R)$ egy bemeneti membrános eldöntő P rendszer, ahol

- $O := O_n \cup \{d_1, d_2, \dots, d_{n+2}, \text{yes}, \text{no}\};$
- $H := \{1, 2, \dots, n+3\};$
- $\mu := [[[]_3]_2]_1;$
- A membránok kezdeti tartalmai: $w_1 = \varepsilon$, $w_2 = d_1$, $w_3 = \varepsilon$;
- Az input membrán a hármas címkéjű membrán, benne a bemenő formulát kódoló O_n feletti szó (azaz $\text{cod}(\varphi)$ valamely $\varphi \in \text{Form}_n - \text{re}$);
- R a következő szabályok halmaza (némi magyarázattal):
 - (a) $[c \rightarrow c_1 c_2]_{i+2}$, $1 \leq i \leq n$, $c, c_1, c_2 \in O_n$, $x_i, \bar{x}_i \notin \hat{c}$, $\hat{c}_1 = \hat{c} \cup \{x_i\}$, $\hat{c}_2 = \hat{c} \cup \{\bar{x}_i\}$
(ezek a szabályok azokat a „klózokat” fogják átírni, amelyekben nem szerepel az i -dik változó, létrehozva a két kiegészített klózt);

- (b) $[]_{i+2} \rightarrow [K_i]_{i+3}[O - K_i]_{i+3}, 1 \leq i \leq n, K_i = \{c \in O_n | x_i \in \hat{c}\}$
 (ezek a szabályok a klózokat két csoportra osztják, aszerint, hogy az i -dik változó negálva van-e bennük, vagy sem; továbbá a membránok címkéit „léptetik” eggyel);
- (c) $[d_i \rightarrow d_{i+1}]_2, 1 \leq i \leq n + 1$
 (ezeket a szabályokat használjuk a rendszer lépéseinek számlálására);
- (d) $[c]_{n+3} \rightarrow \lambda$, ahol $c \in O_n$ és $\hat{c}$ teljes klóz;
- (e) $d_{n+2}[]_{n+3} \rightarrow [yes]_{n+3},$
 $[yes]_{n+3} \rightarrow []_{n+3}yes,$
 $[yes]_2 \rightarrow []_2yes,$
 $[yes]_1 \rightarrow []_1yes$
 (az első szabály létrehozza a *yes* szimbólumot, a többi szabály pedig kijuttatja azt a környezetbe);
- (f) $[d_{n+2}]_2 \rightarrow []_2[no]_2,$
 $[no]_2 \rightarrow []_2no,$
 $[no]_1 \rightarrow []_1no$
 (az első szabály létrehozza a *no* szimbólumot, a többi pedig kijuttatja azt a környezetbe).

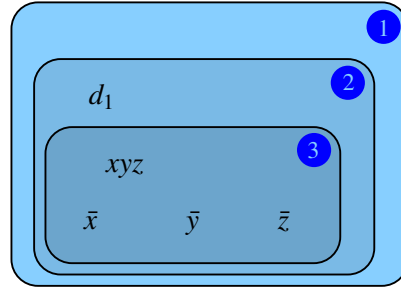
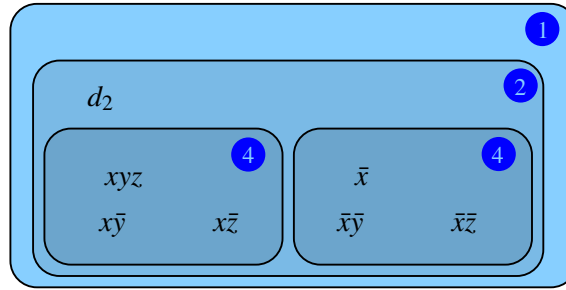
Az alábbiakban egy példán keresztül bemutatjuk, hogyan valósítja meg a membránrendszerünk egy adott formulával ekvivalens KKNF-ben lévő formula előállítását.

1. Példa. Megmutatjuk, hogyan működik $\Pi(3)$ egy három változót tartalmazó formulán. A könnyebb olvashatóság kedvéért az x_1, x_2 és x_3 változók helyett az x, y és z jelöléseket használjuk. Továbbá, az O_3 -beli szimbólumokat az általuk reprezentált klózokban szereplő literálok sorozatával jelöljük, azaz most az O_3 -beli szimbólumok literálokból álló szavak lesznek.

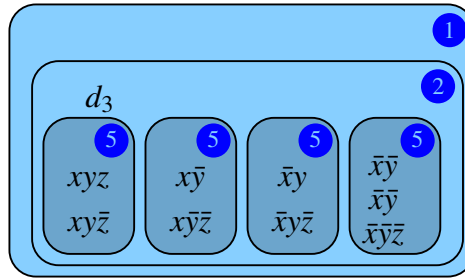
Legyen az input formula $\varphi := \{\{x, y, z\}, \{\bar{x}\}, \{\bar{y}\}, \{\bar{z}\}\}$. Ekkor $\Pi(3)$ indulásakor az $xyz, \bar{x}, \bar{y}, \bar{z}$ szimbólumok vannak az input membránban, ezért a kezdeti konfiguráció a 3.1. ábrán látható módon néz ki.

Az első lépésben a rendszer létrehozza $x\bar{y}$ -t és $\bar{x}\bar{y}$ -t $\bar{y}$ -ből, valamint $x\bar{z}$ -t és $\bar{x}\bar{z}$ -t $\bar{z}$ -ből. Továbbá két új membrán jön létre 4-es címkével, és a rendszer xyz -t, $x\bar{y}$ -t és $x\bar{z}$ -t az első új membránba rakja, $\bar{x}$ -t, $\bar{x}\bar{y}$ -t és $\bar{x}\bar{z}$ -t pedig a másodikba. Az első lépés után a rendszer konfigurációja a 3.2. ábrán látható.

A következő lépésben a rendszer az $xy\bar{z}, x\bar{y}\bar{z}$ szimbólumokat állítja elő $x\bar{z}$ -ből, az $\bar{x}y, \bar{x}\bar{y}$ szimbólumokat $\bar{x}$ -ből, valamint az $\bar{x}y\bar{z}, \bar{x}\bar{y}\bar{z}$ szimbólumokat $\bar{x}\bar{z}$ -ből. Továbbá, két-két új,

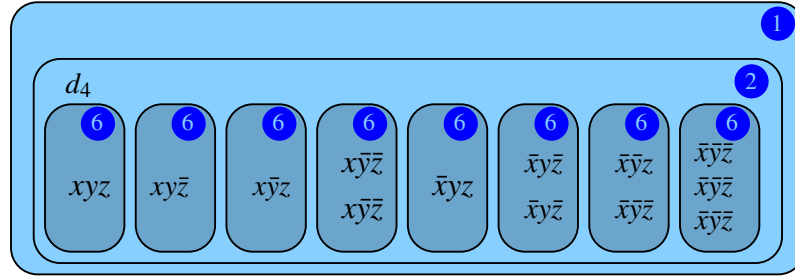
3.1. ábra. $\Pi(3)$ kezdőkonfigurációja3.2. ábra. $\Pi(3)$ konfigurációja az első lépés után

5-ös címkéjű membrán jön létre a 4-es címkéjűekből, a szimbólumok pedig ezekbe az új membránokba szeparálódnak. A második lépés után a rendszer konfigurációja a 3.3. ábrán látható.

3.3. ábra. $\Pi(3)$ konfigurációja a második lépés után

Végül, a harmadik lépés után a rendszer konfigurációja a 3.4. ábrán látható módon néz ki.

Ezek után, mivel minden 6-os címkéjű membrán tartalmaz legalább egy objektumot, a rendszer megszünteti az összes ilyen membránt a (d) -beli szabályok alkalmazásával. A rendszer következő lépésekor tehát csak az első (e) -beli szabály alkalmazható, ami létrehozza a no szimbólumot. A további (e) -beli szabályok pedig kijuttatják a no -t a környezetbe. A rendszer válasza tehát arra a kérdésre, hogy kielégíthető-e $\varnothing$ helyesen az, hogy nem .

3.4. ábra. $\Pi(3)$ konfigurációja a harmadik lépés után

Általánosan a rendszer működése a következőképp zajlik. Kezdetben az egyetlen hármas címkéjű membránban vannak benne a klózek, a kettes címkéjű membránban pedig egy lépe számláló (d_1). Az első n lépésben rendre a következők történnek: a megfelelő (a)-típusú szabályok átírják a klózekat az $i + 2$ -es címkéjű membránokban, ezzel egy időben a számláló eggyel nő a kettes membránban. Ezután az i -dik (b)-típusú szabály kettéosztja az $i + 2$ -es címkéjű membránokat. Az n -dik lépés végére tehát lesz 2^n db $n + 3$ -as címkéjű membránunk, amelyek mindegyike üres, vagy egyféle objektumot tartalmaz (de abból akár többet is), a kettes membránban pedig ott lesz a d_{n+1} -es objektum. A következő lépésben a d_{n+1} átíródik d_{n+2} -re, az $n + 3$ -as címkéjű membránokból pedig azok, amelyekben volt objektum, törlődnek. A következő lépésben, ha nem maradt $n + 3$ -as címkéjű membrán, akkor végre hajtható az első (f) szabály, a kettes membrán kettéosztódik, és az egyikben a d_{n+2} -ből *no* lesz. Ha volt olyan $n + 3$ -as címkéjű membrán, amelyik nem törlődött, az az első (e) szabálynak köszönhetően „beszippantja” d_{n+2} -t, és átalakítja *yes* objektummá, ami a további lépések során kikerül a környezetbe (ha több ilyen membrán is maradt, akkor értelemszerűen csak az egyik végzi el ezt). Ezután a keletkezett *yes* vagy *no* objektum két vagy három lépéssel később kikerül a környezetbe. Ekkor a rendszer működése leáll, a környezetben pedig vagy egy *yes*, vagy egy *no* objektum lesz, annak megfelelően, hogy a formula kielégíthető-e. A fenti gondolatmenet és az 1. állítás alapján, adódik a következő tétel:

2. Tétel. *A SAT eldönthető bemeneti membrános eldöntő P rendszerek egy uniform családjával a bemenő formula méretében polinom időben, ahol a bemenet méretét a benne szereplő ítéletváltozók száma határozza meg.*

3.1.2. A semi-uniform megoldás

Ahogy azt már taglaltuk, az előző módszernek van néhány hátrányos tulajdonsága: a bemenetben szereplő változók számában exponenciálisan sok szabálya van és ugyanez fennáll a munkaábécé méretére. Továbbá, a számítások során könnyen duplikátumok képződhetnek az egyes klózekat reprezentáló objektumokból. Ennek kiküszöbölésére készítettük el a

fordított rezolúciós módszer membránrendszerekkel való implementációjának egy második változatát.

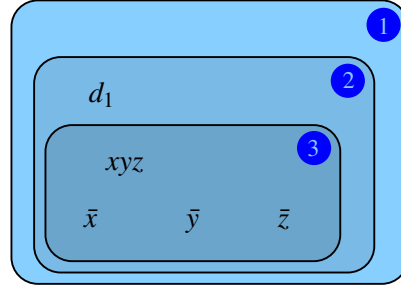
A bemenet elkódolása

Tetszőleges $\varphi \in Form_n$ -re ($n \geq 0$), kódoljuk el φ -t a $cod(\varphi)$ halmazzal a 3.1.1 részben látott módon. Mivel szükségünk lesz a $cod(\varphi)$ -beli szimbólumok egy másolatára, legyen $cod'(\varphi) := \{c' \mid c \in cod(\varphi)\}$.

A membránrendszer

Tetszőleges $\varphi \in Form_n$ -re ($n \geq 0$) esetén legyen $\Pi(\varphi) := (O, H, \mu, w_1, w_2, w_3, R)$ egy eldöntő P rendszer, ahol

- $O := cod(\varphi) \cup cod'(\varphi) \cup \{d_1, d_2, \dots, d_{n+2}, yes, no\}$;
- $H := \{1, 2, \dots, n+3\}$;
- $\mu := [[[]_3]_2]_1$;
- A membránok kezdeti tartalmai: $w_1 = \varepsilon$, $w_2 = d_1$, $w_3 = cod(\varphi)$;
- Nincs kitüntetett input membrán;
- R a következő szabályok halmaza (némi magyarázattal):
 - (a1) $[c \rightarrow c']_{i+2}, 1 \leq i \leq n, c \in cod(\varphi), \bar{x}_i \in \hat{c}$
(ezek a szabályok „megjelölik” azokat a „klózokat”, amelyekben az i -dik változó negálva szerepel);
 - (a2) $[c' \rightarrow c]_{i+2}, 1 \leq i \leq n, c' \in cod'(\varphi), x_i \in \hat{c}$
(ezek a szabályok törlik a jelölést azokról a „klózokról”, amelyekben az i -dik változó negálatlanul szerepel);
 - (a3) $[c \rightarrow cc']_{i+2}, [c' \rightarrow cc']_{i+2}, 1 \leq i \leq n, c \in cod(\varphi), x_i, \bar{x}_i \notin \hat{c}$
(ezek a szabályok duplikálják azokat az objektumokat, amelyek olyan klózokat jelölnek, amelyekben nem szerepel az i -dik változó; a két új szimbólum közül az egyik jelöletlen lesz, a másik megjelölt);
 - (b) $[]_{i+2} \rightarrow [K]_{i+3} [O - K]_{i+3}, 1 \leq i \leq n, K = cod(\varphi)$
(ezek a szabályok a klózokat jelölő objektumokat két csoportra osztják, aszerint, hogy az i -dik változó negálva szerepel a jelölt klózokban vagy sem; továbbá a membránok címkéit „léptetik” egyet);

3.5. ábra. $\Pi(\varphi)$ kezdőkonfigurációja

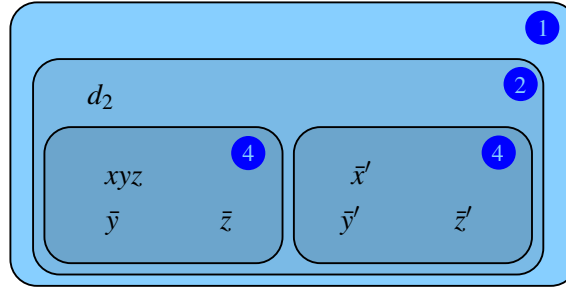
- (c) $[d_i \rightarrow d_{i+1}]_2, 1 \leq i \leq n+1;$
- (d) $[c]_{n+3} \rightarrow \lambda, c \in \text{cod}(\varphi) \cup \text{cod}'(\varphi);$
- (e) $d_{n+2}[\]_{n+3} \rightarrow [\text{yes}]_{n+3},$
 $[\text{yes}]_{n+3} \rightarrow [\]_{n+3}\text{yes},$
 $[\text{yes}]_2 \rightarrow [\]_2\text{yes},$
 $[\text{yes}]_1 \rightarrow [\]_1\text{yes};$
- (f) $[d_{n+2}]_2 \rightarrow [\]_2[\text{no}]_2,$
 $[\text{no}]_2 \rightarrow [\]_2\text{no},$
 $[\text{no}]_1 \rightarrow [\]_1\text{no}.$

A (c)-(f) szabályoknak hasonló a szerepük, mint a megfelelő szabályoknak a 3.1.1 részben. Ahogyan a korábbi fejezetben is tettük, most is egy példán keresztül szeretnénk könnyebbé tenni a membránrendszerünk működésének megértését.

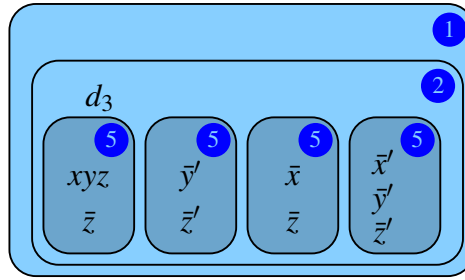
2. Példa. Tekintsük újra az 1. példát és az abban szereplő $\varphi = \{\{x, y, z\}, \{\bar{x}\}, \{\bar{y}\}, \{\bar{z}\}\}$ formulát.

Legyen $\Pi(\varphi)$ a 3.1.1. részben leírt módon φ -ből megkonstruált P rendszer. $\Pi(\varphi)$ kezdeti konfigurációja a 3.5. ábrán látható.

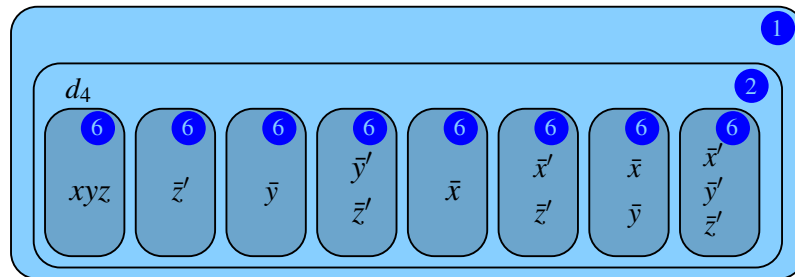
Az első lépés során xyz változatlan marad, $\bar{x}$ -t egy aposztróffal látjuk el, valamint $\bar{y}$ -ból és $\bar{z}$ -ból az $\bar{y}$, $\bar{y}'$ illetve $\bar{z}$, $\bar{z}'$ szimbólumok jönnek létre. Ezután, ugyanebben a lépésben, $\Pi(\varphi)$ a 3-as címkéjű membránban lévő szimbólumokat szeparálja aszerint, hogy meg vannak-e jelölve, vagy sem. Az első lépés után a rendszer konfigurációja a 3.6. ábrán látható.

3.6. ábra. $\Pi(\varphi)$ konfigurációja az első lépés után

A második lépés során xyz és y' változatlanok maradnak, $\bar{y}$ megjelölt lesz, valamint $\bar{x}'$, $\bar{z}$ és $\bar{z}'$ átíródnak a megfelelő két-két szimbólumra az (a3)-beli szabályoknak megfelelően. Ezután a szimbólumok szétosztódnak az új membránokba (3.7. ábra).

3.7. ábra. $\Pi(\varphi)$ konfigurációja a második lépés után

Végül, $\Pi(\varphi)$ harmadik lépése után a rendszer konfigurációja a 3.8. ábrán látható módon néz ki.

3.8. ábra. $\Pi(\varphi)$ konfigurációja a harmadik lépés után

Ezután, mivel minden 6-os címkéjű membrán tartalmaz legalább egy szimbólumot, $\Pi(\varphi)$ ugyanúgy folytatja a működését és küldi a no szimbólumot a környezetbe, mint ahogy azt $\Pi(3)$ is teszi.

Általánosan, egy $\varphi \in Form_n$ ($n \geq 0$) formulára $\Pi(\varphi)$ a következőképpen működik: kezdetben az egyetlen hármas címkéjű membránban vannak a φ klózzai elkódolva, a kettes cím-

kéjű membránban pedig egy lépésszámláló (d_1). Az i -ik lépésben ($1 \leq i \leq n$) rendre a következők történnek: a megfelelő (a)-típusú szabályok az olyan klózatok reprezentáló objektumokat melyekben szerepel x_i alapú literál megjelölik egy vesszővel illetve jelöletlenül hagyják, attól függően, hogy az x_i változó negálva vagy negálatlanul szerepel az objektum által reprezentált klózban. Továbbá, minden olyan objektumra ami olyan klózt reprezentál melyben nem szerepel x_i alapú literál, a rendszer létrehoz az objektumból egy megjelölt és egy jelöletlen változatot is. Ez felel meg az implementált módszerben a fordított rezolúciós lépésnek. Ezzel egy időben a számláló eggyel nő a kettes membránban. Ezután az i -dik (b)-típusú szabály kettéosztja az $i + 2$ -es címkéjű membránokat, és szétosztja a bennük szereplő objektumokat aszerint, hogy azok megjelöltek vagy jelöletlenek-e. Az n -dik lépés végére tehát lesz 2^n db $n + 3$ -as címkéjű membránunk, a kettes membránban pedig ott lesz a d_{n+1} -es objektum. A következő lépésben a d_{n+1} -ből d_{n+2} lesz, az $n + 3$ -as címkéjű membránok közül pedig azok, amelyekben volt objektum, törölődnek. Ezután a rendszer ugyanúgy működik, mint ahogy azt a 2.5.2. részben látott $\Pi(n)$ is teszi, ha a bemeneti membránjában kezdetben $cod(\varphi)$ szerepel. Könnyen belátható, hogy minden $0 \leq i \leq n$ esetén, a $\Pi(\varphi)$ és $\Pi(n)$ rendszerek i -ik lépés utáni membránstruktúrája megegyezik. Az is belátható, hogy az n -ik lépés után a $\Pi(\varphi)$ egy $n + 3$ -as címkéjű m membránja pontosan akkor üres, ha $\Pi(n)$ -ben az m -nek megfelelő $n + 3$ -as címkéjű membrán is üres. Ez azt jelenti, hogy $\Pi(\varphi)$ pontosan akkor küldi a környezetbe a *yes* szimbólumot, amikor $\Pi(n)$ a környezetbe küldi azt a φ bemeneten. Kapjuk tehát a következő tételt.

3. Tétel. *A SAT eldönthető eldöntő P rendszerek egy polinomiálisan semi-uniform családjával a bemenő formula méretében polinom időben, ahol a bemenet méretét a benne szereplő ítéletváltozók száma határozza meg.*

3.2. A három módszer összehasonlítása

Világos, hogy a dolgozatban látott három aktív membrános P rendszer különböző erőforrásigényekkel rendelkezik. Így általánosságban egyik rendszert sem lehet a másik fölé emelni. Bizonyos erőforrásokat tekintve az egyik a jobb, míg mások esetén a másik. Az általunk definiált rendszerek az időigény tekintetében jobbak, viszont az uniform rendszerünk objektumhalmaza exponenciális méretű. Ez hátránynak tűnhet az általunk a 2.5.2. részben bemutatott, már ismert rendszerhez képest. Viszont az is könnyen belátható, hogy a 2.5.2. részben látott rendszer, bár az objektumhalmaza polinomiális méretű, a működése során exponenciális számú membránt használ, ami az implementáció során, véleményünk szerint, másodlagossá teszi azt a tényt, hogy az objektumhalmaz polinom méretű. Azért, hogy az Olvasó könnyebben össze tudja hasonlítani az ismertetett rendszerek erőforrásigényét, ebben a feje-

zetben összefoglaljuk az ezzel kapcsolatos ismereteinket. Ehhez legyen $\varphi = \{C_1, C_2, \dots, C_m\}$ egy $Form_n$ -beli formula ($n \geq 0, m \geq 1$). A három módszer hatékonyságát az alábbi mértékek jellemzik (rendre $\Pi(\langle n, m \rangle)$ -ként, $\Pi(n)$ -ként és $\Pi(\varphi)$ -ként hivatkozunk az 2.5.2, 3.1.1 és 3.1.2. részekben definiált membránrendszerekre):

- Munka ábécé mérete:

– $\Pi(\langle n, m \rangle)$:

$$\begin{aligned}
 |O| &= |\{x_{i,j}, \bar{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\}| + |\{c_k : 1 \leq k \leq m+2\}| + \\
 &+ |\{d_k : 1 \leq k \leq 3n+2m+3\}| + |\{r_{i,k} : 0 \leq i \leq m, 1 \leq k \leq 2n\}| + \\
 &+ |\{e, t\}| + |\{yes, no\}| = \\
 &= 2mn + (m+2) + (3n+2m+3) + (m+1)2n + 2 + 2 \\
 &= 2mn + 2mn + 3m + 5n + 9 = 4mn + 3m + 5n + 9 = \\
 &= \Theta(mn)
 \end{aligned}$$

– $\Pi(n)$:

$$\begin{aligned}
 |O_n| &+ |\{d_1, d_2, \dots, d_{n+2}, yes, no\}| = \\
 &= 2^n + (n+4) = \\
 &= \Theta(2^n)
 \end{aligned}$$

– $\Pi(\varphi)$:

$$\begin{aligned}
 |cod(\varphi) \cup cod'(\varphi)| &+ |\{d_1, d_2, \dots, d_{n+2}, yes, no\}| = \\
 &= 2m + (n+4) = \\
 &= \Theta(m+n)
 \end{aligned}$$

- A szabályok száma:

– $\Pi(\langle n, m \rangle)$:

$$\begin{aligned}
 |R| &= |(a)| + |(b)| + \dots + |(q)| = \\
 &= n + 4m + 4m(n-1) + (3n-1) + \\
 &+ m(2n-1) + (2n-1) + 2 + (2m+3) + \\
 &+ 1 + m + 1 + m + 1 + 1 + 1 + 1 + 1 = \\
 &= 6mn + 3m + 6n + 7 = \\
 &= \Theta(mn).
 \end{aligned}$$

– $\Pi(n)$:

$$\begin{aligned}
 |R| &= |(a)| + |(b)| + \dots + |(f)| \\
 &= \left(\sum_{i=1}^n 2^{i-1} 3^{n-i} - 1\right) + 3^{n-1} + (n+2) + 2^n + 4 + 3 \\
 &= (3^n - 2^n - 1) + 3^{n-1} + 2^n + n + 9 = \\
 &= 4 \cdot 3^{n-1} + n + 8 = \\
 &= \Theta(3^n).
 \end{aligned}$$

$$\begin{aligned}
- \Pi(\varphi): \\
|R| &= |(a)| + |(b)| + \dots + |(f)| \\
&= 2mn + n + (n + 2) + 2m + 4 + 3 \\
&= 2mn + 2m + 2n + 9 \\
&= \Theta(mn).
\end{aligned}$$

- Membránok száma a rendszer működése során:

- $\Pi(\langle n, m \rangle)$: kezdetben $2[1 + 1]$, majd a következő $2n$ lépésben $2^{\lceil i/2 \rceil} + 1$ ($1 \leq i \leq 2n$), azután végig $2^n + 1$ marad.
- $\Pi(n)$: kezdetben 3, majd a következő n lépésben $2^i + 2$ ($1 \leq i \leq n$), utána a 2^n darab $n + 3$ -as címkéjű membránok egy része (akár az összes is) törlődik.
- $\Pi(\varphi)$: kezdetben 3, majd a következő n lépésben $2^i + 2$ ($1 \leq i \leq n$), utána a 2^n darab $n + 3$ -as címkéjű membránok egy része (akár az összes is) törlődik.

- Objektumok száma a futás során:

- $\Pi(\langle n, m \rangle)$: kezdetben $\sigma = \sum_{i=1}^m |C_i|$ számú objektum van a kettes membránban, továbbá egy darab d_1 az egyes membránban. Az $2n$ -dik lépés után az egyes membránban lesz 2^n darab d_n , a kettes címkéjű membránokban pedig összesen $\sigma 2^{n-1}$ (minden kezdeti objektum pontosan a membránok felében jelenik meg r -es objektum formájában). A későbbiek során még létrejön az egyes membránban 2^n darab e objektum, melyek átíródnak a kettes címkéjű membránokban c objektumokra. A c -k egy része létrehoz t -ket valamint yes -eket (vagy akár egyet sem); továbbá a $d_{3n+2m+3}$ objektumok létrehozhatnak no objektumokat. A működés során tehát jellemzően $\Theta(2^n)$ objektum van jelen a rendszerben.
- $\Pi(n)$: kezdetben m darab c objektum a hármas membránban, valamint egy d objektum a kettes membránban. A következő n lépésben az $i + 2$ -es címkéjű membránokban ($1 \leq i \leq n$) lévő elemek száma legfeljebb megduplázódik, és a lépések végére legfeljebb $m2^n$ darab lesz. Az ezt követő lépésben az $n + 3$ -as címkéjű membránokban lévő objektumok törlődnek, a d_{n+2} -es objektumból pedig létrejön egy yes vagy egy no . Tehát a rendszerben az objektumok maximális száma egy időben $m2^n$.
- $\Pi(\varphi)$: kezdetben m darab c objektum a hármas membránban, valamint egy d objektum a kettes membránban. A következő n lépésben az $i + 2$ -es címkéjű membránokban ($1 \leq i \leq n$) lévő elemek száma legfeljebb megduplázódik, és a

	<i>Munkaábécé mérete</i>	<i>Szabályok száma</i>	<i>Membránok száma a működés során</i>	<i>Objektumok száma a működés során</i>	<i>Futásidő (lépésszám)</i>
$\Pi(\langle n, m \rangle)$	$\Theta(mn)$	$\Theta(mn)$	$\Theta(2^n)$	$\Theta(2^n)$	$5n + 2m + 3$
$\Pi(n)$	$\Theta(2^n)$	$\Theta(3^n)$	$\Theta(2^n)$	$\Theta(m2^n)$	$n + 5$
$\Pi(\varphi)$	$\Theta(m + n)$	$\Theta(mn)$	$\Theta(2^n)$	$\Theta(2^n)$	$n + 5$

3.1. táblázat. A dolgozatban szereplő SAT-ot megoldó rendszerek erőforrásigénye

lépések végére legfeljebb 2^n darab lesz. Az ezt követő lépésben az $n + 3$ -as címkéjű membránokban lévő objektumok törlődnek, a d_{n+2} -as objektumból pedig létre jön egy *yes* vagy egy *no*. Tehát a rendszerben az objektumok maximális száma egy időben 2^n .

• Futásidő (lépések száma):

- $\Pi(\langle n, m \rangle)$: $5n + 2m + 3$.
- $\Pi(n)$: $n + 5$.
- $\Pi(\varphi)$: $n + 5$.

A fent leírtakat a 3.1. táblázatban foglaltuk össze.

Fontos megjegyezni, hogy a bemenő formulában szereplő klózik száma (m) nagy is lehet: általános esetben n -ben exponenciális. A $\Pi(n)$ nagy előnye a $\Pi(\langle n, m \rangle)$ és $\Pi(\varphi)$ rendszerekkel szemben az, hogy $\Pi(n)$ a klózoktól (a formulától) függetlenül megkonstruálható, míg a másik két rendszer esetén a formula mérete nem csak az inputot befolyásolja, hanem a munka és input ábécét is (mindkettő függ m -től, a klózik számától, sőt $\Pi(\varphi)$ esetében maguktól a klózoktól is). $\Pi(n)$ munka és input ábécéje ezzel szemben csak a változók számától függ. Megjegyezzük továbbá, hogy $\Pi(n)$ esetében az (a) illetve (c) – (f) szabályok mérete konstans. Továbbá, nem nehéz belátni, hogy $\Pi(n)$ működése során az i címkéjű membránok ($3 \leq i \leq n + 3$) legfeljebb a klózik számával, azaz m -mel megegyező számú objektumot tartalmazhatnak. Ebből következik, hogy a (b) osztó szabályok mindig olyan membránokra kerülnek alkalmazásra, amelyek legfeljebb m elemet tartalmaznak. Hasonló megállapítások érvényesek $\Pi(\varphi)$ esetében is.

4. fejezet

Összefoglalás

Dolgozatunkban ismertettünk egy módszert, amellyel egy KNF-ben lévő formula kielégíthetőségét úgy döntjük el, hogy először megkonstruáljuk a formulával ekvivalens kitüntetett KNF-ben lévő formulát, majd pedig megvizsgáljuk, hogy ezen új formula klózái között szerepel-e az összes teljes klóz. Ezután definiáltuk aktív membrános P rendszerek egy uniform és egy semi-uniform családját, amelyekben a fenti módszert implementáltuk. Mindkét rendszer a változók számában lineáris időben képes eldönteni egy KNF-ben lévő formula kielégíthetőségét. Ezt a hatékonyságot eddig csak olyan membránrendszerekkel sikerült elérni, amelyek - a mi rendszereinktől eltérően - nem elemi membránokat szétosztó szabályokat is használnak. A mi rendszereink nem használják a membránok polarizálhatóságának lehetőségét sem, helyette inkább olyan osztó szabályokat használnak, amelyek képesek megváltoztatni a membránok címkéit. Az eddig ismert, csak elemi membránokat szétosztani képes rendszerek esetében a számítások időigénye (számítási lépések száma) a formulában szereplő klózek számától is függ. Ettől függetlenül nem mondhatjuk azt, hogy az eredményeink megjavítják a már ismert eredményeket a következők miatt: az uniform megoldásunk nem polinomiálisan uniform, míg a második megoldásunk nem uniform. Pontosan ezért tervezzük egy olyan polinomiálisan uniform megoldás létrehozását, amely az általunk kitalált módszeren alapul de olyan formula elkódolási technikát használ, amelynek segítségével a kitűzött cél elérhető.

Azt gondoljuk, hogy a megoldásaink tovább fejleszthetők az alábbi észrevételek kidolgozásával és implementálásával: először tekintsük az első példában szereplő $\Pi(3)$ rendszert és $\varnothing$ bemenetet. Észrevehető, hogy mivel egy 4-es címkéjű membrán tartalmazza $\bar{x}$ -et, minden ebből származó 6-os címkéjű membrán tartalmazni fog egy teljes klózt. A rendszer tehát nyugodtan törölhetné ezt a 4-es címkéjű membránt anélkül, hogy létrehozná a szóban forgó 6-os címkéjű membránokat, a bemenő formula kielégíthetőségének eldöntését ez nem befolyásolná. Általánosságban ez annyit tesz, hogy ha egy 3.1.1. részben megadott $\Pi(n)$ P rendszert

elindítunk egy $Form_n$ -beli formulával és a rendszer i -ik lépése után van olyan $i + 3$ -as címkéjű membrán ($1 \leq i \leq n$), amely tartalmaz egy olyan klózt, amely nem tartalmaz egyetlen változót sem az $\{x_{i+1}, \dots, x_n\}$ halmazból, akkor ezt a membránt törölni lehet, ez nem fog változtatni a végeredményen, és nem kell további $\Theta(2^{n-i})$ számú membránt létrehozni.

Az is nyilvánvaló, hogy ha $\Pi(n)$ az i -ik lépés után tartalmaz egy üres $i + 3$ -as címkéjű membránt, akkor az n -ik lépés után lesz a rendszerben egy üres $n + 3$ -as címkéjű membrán is. Tehát a bemenő formula kielégíthetősége már az n -dik lépés előtt is kiderülhet és ezt felhasználva tovább lehetne csökkenteni a működés során létrehozott membránok számát. Meg kell jegyezni azonban, hogy a P rendszereink tárigénye általánosan még ezen javításokkal együtt is exponenciális lenne a bemenő formula változói számában függvényében.

Mivel a 3.1.1. részben definiált $\Pi(n)$ rendszer mérete exponenciális n -ben, ésszerű a kérdés, hogy létezhet-e olyan $\Pi(n)$ -en alapuló membránrendszer ami a SAT problémát képes konstans időben eldönteni. Az könnyen látható, hogy $\Pi(n)$ -ből kis módosítással meg lehetne adni egy olyan $\Pi'(n)$ rendszert, amely egy $\varphi \in Form_n$ formula klózaiból képes lenne egy lépésben létrehozni a φ -vel ekvivalens, kitüntetett KNF-ben lévő φ' formula klózeit. Másrészt viszont nem világos, hogy $\Pi'(n)$ hogyan tudná konstans lépésben eldönteni azt, hogy az így létrehozott klózek között szerepel-e az összes teljes klóz vagy sem.

Végül megjegyezzük, hogy terveink közt szerepel a fenti rendszerek implementálása egy megfelelő párhuzamos környezetben, mert szeretnénk látni, hogy a gyakorlatban milyen szinten lehet kiaknázni az általunk megadott membránrendszerek elméleti hatékonyságát.

Irodalomjegyzék

- [1] Artiom Alhazov. Minimal parallelism and number of membrane polarizations. *The Computer Science Journal of Moldova*, 18(2):149–170, 2010.
- [2] Artiom Alhazov, Linqiang Pan, and Gheorghe Paun. Trading polarizations for labels in p systems with active membranes. *Acta Inf.*, 41(2-3):111–144, 2004.
- [3] José M. Cecilia, José M. García, Ginés D. Guerrero, Miguel A. Martínez del Amor, Ignacio Pérez-Hurtado, and Mario J. Pérez-Jiménez. Simulating a P system based efficient solution to SAT by using GPUs. *J. Log. Algebr. Program.*, 79(6):317–325, 2010.
- [4] Gabriel Ciobanu, Linqiang Pan, Gheorghe Paun, and Mario J. Pérez-Jiménez. P systems with minimal parallelism. *Theor. Comput. Sci.*, 378(1):117–130, 2007.
- [5] János Demetrovics, Jordan Denev, and Radiszlav Pavlov. *A számítástudomány matematikai alapjai*. Nemzeti Tankönyvkiadó Rt., Budapest, 1999.
- [6] Rudolf Freund, Gheorghe Paun, and Mario J. Pérez-Jiménez. Polarizationless p systems with active membranes working in the minimally parallel mode. In *UC*, pages 62–76, 2007.
- [7] Zoltán Fülöp. *Formális nyelvek és szintaktikus elemzésük*. SzTE Bolyai Intézet, Szeged, 2004.
- [8] Zsolt Gazdag and Gábor Kolonits. A new approach for solving SAT by P systems with active membranes. In Erzsébet Csuhaj-Varjú, Marian Gheorghe, and György Vaszil, editors, *Proceedings of the 13th International Conference on Membrane Computing (CMC13)*, pages 211–220, 2012.
- [9] Zsolt Gazdag and Gábor Kolonits. A new approach for solving SAT by P systems with active membranes. In Erzsébet Csuhaj-Varjú, Marian Gheorghe, and György Vaszil, editors, *CMC 2012, LNCS 7762*, pages 195–207, 2013.

- [10] Linqiang Pan and Artiom Alhazov. Solving hpp and sat by p systems with active membranes and separation rules. *Acta Inf.*, 43(2):131–145, 2006.
- [11] Katalin Páztorné Varga and Magda Várterész. *A matematikai logika alkalmazás-szemléletű tárgyalása*. Panem Könyvkiadó, Budapest, 2003.
- [12] Gheorghe Paun. Computing with membranes. *J. Comput. Syst. Sci.*, 61(1):108–143, 2000.
- [13] Gheorghe Paun. P systems with active membranes: Attacking np-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.
- [14] Gheorghe Paun. Introduction to membrane computing. In *Applications of Membrane Computing*, pages 1–42. 2006.
- [15] Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.
- [16] Mario J. Pérez-Jiménez, Álvaro Romero Jiménez, and Fernando Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3):265–285, 2003.
- [17] Wikipedia. P system, May 2012.