Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik Tanszék

Membránrendszerek vizualizációja és szimulálása

Témavezető:
Kolonits Gábor
egyetemi tanársegéd

Szerző:
Buzga Levente
programtervező informatikus BSc

Budapest, 2022

A membránrendszer vagy más néven P-rendszer olyan biológiailag inspirált számítási modell, amely képes a biokémiai folyamatok törvényszerűségeinek modellezésére. A rendszer több belső térből, úgynevezett régióból áll, amelyek hierarchikus membránstruktúrát alkotnak. Ez lehetővé teszi a sejten belüli és a sejtek közötti információáramlás formális modellezését.

Az egyes régiókban különböző nyersanyagok fordulhatnak elő, amelyek egymással reakcióba léphetnek. A reakció következtében ezek a nyersanyagok átalakulhatnak, régiók között átvándorolhatnak, illetve membránok is feloldódhatnak. Ezeket a reakciókat evolúciós szabályoknak hívjuk. A felépítésükből adódóan a membránrendszerek nagy mértékű párhuzamosításra is alkalmasak. A membránrendszer számítása közben kialakuló állapotát egy konfigurációja írja le, amelyet minden membrán esetén objektumokból álló multihalmaz reprezentál. A számítás eredményét többféleképpen is definiálhatjuk, alapértelmezetten egy kitüntetett membránban a megálláskor jelen lévő objektumok mennyiségét jelenti.

Szakdolgozatom célja egy olyan szoftver elkészítése, amely képes különböző típusú membránrendszerek grafikus megjelenítésére, betöltésére és exportálására, illetve egy megadott szabályrendszer mentén történő számításaik szimulálására. A felhasználónak lehetősége van a membránstruktúra vizuális megkonstruálására, illetve egy megadott formátum segítségével szabályok rögzítésére. A számítás végrehajtásánál támogatott a lépésenkénti futtatás.

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. A számításelmélet rövid története

A számításelmélet az 1930-as évek közepén Alan Turing és Alonzo Church munkássága révén vált külön tudományággá, amelynek központjában a számítások és a számítógépek formális modellezése és elemzése áll. A terület azóta is kulcsfontosságú szerepet kap az informatikában. Az előbb említett matematikai logikával foglalkozó tudósok és társaik arra a kérdésre szerettek volna választ adni, hogy mit jelent maga a számítás? Ezzel az algoritmus matematikai modelljének megalkotását tűzték ki célul. Természetesen a számításelmélet létrejötte előtt is születtek algoritmusok, de azok nem törekedtek formalitásra, hanem egyszerűen csak rögzítették a végrehajtandó utasítások sorrendjét. A kérdés megválaszolása érdekében Turing bevezette a Turing gépet, Church pedig a lambda kalkulust. Ezen eszközök segítségével elkezdték vizsgálni a kiszámíthatóságot. Azt fedezték fel, hogy vannak olyan számítási feladatok, amelyekre nem adható algoritmikusan válasz, illetve, hogy vannak olyan univerzális számítógépek, amelyek minden másik számításra alkalmas gépet szimulálni tudnak. Ez vezetett a Church-Turing tézis megfogalmazásához, amely szerint minden olyan matematikailag formalizálható probléma, ami megoldható valamilyen algoritmussal, az Turing géppel is megoldható. Később kiderült, hogy léteznek a Turing géppel azonos számítási erejű modellek, viszont azóta sem találtak ennél nagyobb számítási erővel rendelkező absztrakt modellt. Ezek szerint ha elfogadjuk a tézist, akkor a Turing gép tekinthető az algoritmus formális modelljének.

A tudományág fejlődését ösztönözte és elősegítette a digitális számítógépek megjelenése. A tézis után pár évvel Konrad Zuse elkészítette az első általános célú progra-

mozható elektromechanikus számítógépet, amelyet elektromágneses relékkel működtetett. 1945-ben készült el az első teljesen elektromos számítógép, az ENIAC, majd Neumann János 1947-ben megfogalmazta a mai számítógép működésének alapjául szolgáló Neumann-elveket. Azóta a számítógépek számítási kapacitása rohamos ütemben növekedik, ám az akkor fontosnak vélt számítási elemzések és korlátok a mai napig fontos szempontot adnak algoritmusok megalkotásánál.

A mai számításelmélet nagyon változazos témák tárházával rendelkezik, magába foglalja az algoritmusok, adatszerkezetek, elosztott és párhuzomísott számítások, kvantumszámítások, biológiai számítások vizsgálatát, illetve nem utolsó sorban a bonyolultságelméletet és az információelméletet. Mindezekből látszik, hogy milyen szerteágazó tematikája és problémaköre van a számításelméletnek és hogy ezek az irányvonalak folyamatosan képesek fejlődni és gazdagodni.

1.2. Membránrendszerek

A 20. század második felében egyre nagyobb figyelmet kaptak az olyan számítások, amely a természetben vagy valamilyen természeti jelenséghez kapcsolódóan mennek végbe. Ezen megfigyelések eredményeképpen megszületett a biológiai számítások területe, amelynek számos új módszert és algoritmust köszönhetünk. Ide tartoznak például a genetikus algoritmusok (általánosabban véve az evolúciós algoritmusok), illetve a neurális hálók. Ezen irányvonal kiterjesztésének tekinthetőek az 21.század legelején kibontakozó területek mint a membránszámítások és a DNS számítások, amely közül az utóbbit a molekuláris biológia alapjaira modelleztek.

A membránszámítások legnagyobb inspirációját a biológiai sejtek jelentik, amelyek az élet legelemibb építőelemei. A sejtek működésében és struktúrájában pedig elengedhetetlenek a membránok, amelyek a sejten belüli rétegződést határozzák meg. Egy sejtet a környezetétől a legkülső membránja (úgynevezett skin) határol el, a rétegződésért pedig a belső membránok felelősek. Egy-egy ilyen membrán pedig egy saját belső teret, ún. régiót határol el. Ezen régiókban pedig kémiai reakciók mehetnek végbe, amely hatására különböző molekulák átvándorolhatnak a membránokon keresztül új régiókba. Ezzel a térbeli struktúráltsággal egészítik ki a membránszámítások a DNS számításokat.A membránszámítás motivációja, hogy egy olyan modellt tudjuk konstruálni, amely képes a sejtek közötti és sejten belüli molekulák által közvetített információáramlás mechanizmusának leírására, szemléltetésére.

Ezen célkitűzéssel vezette be Gheorge Paun a membránrendszereket 2000-ben. Később az ilyen modelleket P-rendszereknek is elnevezték az ő tiszteletére. Ahhoz, hogy a biokémiai folyamatokat modellezni tudja, figyelembe vette ezen folyamatok törvényszerűségeit. Nevezetesen, hogy az ilyen reakciók végbemeneteléhez kellő számban szükség van a reakcióhoz szükséges kezdeti molekulákra és ha ezek rendelkezésre állnak, akkor a reakció biztosan meg is fog történni. Emellett egyszerre több reakció (vagy akár egy reakció többször) is végbemehet, ha van hozzá(juk) elegendő nyersanyag. Speciális esetekben egy reakció hatására membránok feloldódhatnak, ilyenkor a benne rejlő molekulák és belső régiók az őt körülvevő régióba kerülnek, illetve az is előfordulhat, hogy a reakciók között valamilyen prioritási sorrend is fennállhat; azaz az egyik reakció nem hajtódhat végre, amíg egy a másik reakciónak lehetősége van rá. Ezen reakciók úgynevezett evolúciós lépésekbe szerveződnek, amelyek során a régiók molekulái és a membránrendszer struktúrája is átalakul.

A számítás központi szereplői az előbbi tulajdonságokkal rendelkező reakciók, amelyek a membránrendszerekben szabályoknak nevezünk, a résztvevő molekulákat pedig objektumoknak. Ezek a szabályok régiókhoz vannak rendelve. A szabályok alkalmazására a maximális párhuzamosság elve érvényes, azaz ha a jelenleg alkalmazott szabályok után még alkalmazható lenne egy újabb vagy egy eddig már alkalmazott szabály, akkor az a szabály végre is lesz hajtva az adott evolúciós lépésben. Ennek megfelelően a membránrendszer evolúciós lépések sorozatán megy keresztül, egészen addig, amíg már nincsen egyetlen alkalmazható szabály sem. Ilyenkor a számítás eredményét alapértelmezetten a környezetbe kijutó objektumok számának tekinjük, de ez egyes rendszerekben másképpen is megadható (például egy kijelölt kimeneti régió objektumainak számával).

Érdeklődésemet a téma iránt az SAT eldöntési probléma aktív P-rendszerekkel (amely típust a dolgozatban nem részletezem) való lineáris időben történő megoldása keltette fel. Ez elsőre egy hihetetlen eredmény, hiszen ez volt a legelső probléma, amiről bebizonyították, hogy NP-teljes. Az ellentmondást a membránrendszerek masszív párhuzamosítási képességével lehet megmagyarázni, amelynek segítségével a változók és a klózok számának függvényében igaz, hogy lineáris időben megoldható, viszont ezt a membránrendszert szimuláló Turing gép már nem tudna minden bemeneti példányra lineáris időben leállni. Ezután kezdtem el foglalkozni membránrendszerekkel és úgy éreztem, hogy egy szimulációs szoftver hasznosnak bizonyulna kutatási és oktatási célokra egyaránt.

1.3. Megvalósítandó feladat

A dolgozatom célja egy olyan szoftver elkészítése, amely lehetővé teszi a felhasználó számára különböző felépítésű membránrendszerek megkonstruálását, grafikus módon való megjelenítését, illetve számításainak végrehajtását. A támogatott számítási modellek magukba foglalják a membránrendszerek alapmodelljét (amelyben egyes régiók feloldódhatnak, illetve a szabályok között lehet prioritási sorrend) és a szimport-antiport rendszereket. A membránrendszer létrehozását a felhasználó egy (megfelelő formátumú) karakterlánc megadásával kezdeményezheti, ami egyszerűen és szemléletesen írja le a membránrendszer struktúráját és opcionálisan tartalmazhatja régiokra bontva a benne található objektumokat. A megalkotott membránrendszer ilyenkor még nem rendelkezik evolúciós szabályokkal, azokat egy külön menü segítségével tudja beállítani. Ilyenkor a kiválasztott régió objektumai is szerkeszthetőek.

A szoftver különösen hasznos tud lenni oktatási célokra, hiszen biztosítja a szabályok elemzésének lehetőségét a lépésenkénti futtatás funkcióval, illetve a teljes szimuláció során megkapott eredmények segítségével tanulmányozható, hogy a létrehozott modell a kívánt nyelvet (ebben az esetben a természetes számok valamely részhalmazát) generálja-e, ezzel segítve a helyes és helytelen működés (nem teljeskörű) kiszűrését. A program biztosít a felhasználó számára egy használati útmutatót, melyben megtalálja a legfontosabb információkat ahhoz, hogy korábbi ismeretek nélkül is képes legyen egy működő membránrendszert megalkotni és számítását szimulálni.

Lehetőség van a membránrendszer aktuális állapotának lementésére, illetve egy korábbi mentés betöltésére. A membránrendszer *JSON* formátumban kerül mentésre, így azt a felhasználónak lehetősége nyílik arra, hogy a fájlban is módosítson, majd a módosított állapotot töltse be. Ezáltal a szoftver biztosítani tudja azt, hogy egy bonyolult vagy gyakran vizsgált membránrendszert nem szükségszerű a kezdetektől megalkotni, elegendő egyszer a kívánt konfigurációját előállítani, majd később betölteni a már elmentett verziót.

A feladatot *Python* programozási nyelven, a *Qt* keretrendszer felhasználásával valósítottam meg.

2. fejezet

Felhasználói dokumentáció

2.1. Az alkalmazás célja

Az alkalmazás célja, hogy a felhasználó segítségével membránrendszereket hozzon létre majd szimulálja számításaikat. A membránrendszer egy olyan biológiailag inspirált számítási modell, amely az eukarióta sejtek működését és felépítését követve evolúciós lépéseken keresztül történő információáramlást ír le membránok között. Minden membrán által körbezárt *ún.* régió tartalmaz evolúciós szabályokat, amelyek nem változnak a membránrendszer működése közben. Az információt a rendszerben a régiókban található molekulák, *ún.* objektumok hordozzák. Egy szabály csak akkor tud végbemenni, ha rendelkezésre állnak a szükséges objektumok kellő számban. Ilyen helyzetekben a szabályoknak végre is kell hajtódnia, tehát nem fordulhat elő, hogy minden objektum hozzáférhető, de nem kerül a szabály alkalmazásra. Egy evolúciós lépésben a maximális párhuzamosság elve érvényesül, azaz a szabályok véletszerűen kerülnek kiválasztása, egészen addig, amíg van alkalmazható szabály. Megadhatóak olyan speciális szamályok, amelyek alkalmazásának hatására egy membrán feloldódhat, ilyenkor a tartalma az őt körbevevő régióba kerül. A szabályok között prioritási sorrend is felállítható. A számítás legfontosabb tulajdonsága annak kimenete, amely általában a legkülső régión kívülre (azaz a környezetbe) kijutó objektumok számát jelenti.

2.2. Hardver és szoftver követelmények

A szoftver futtatásához Linux környezetre van szükség, amely támogatja az `ELF` formátumú bináris állományok értelmezését. Ezen felül a futtatási környezetnek rendelkeznie kell *Python interpreterrel*, illetve a *Qt* keretrendszerhez való hozzáférés érdekében `PySide6` modullal. Az utóbbi könnyen megtehető shell környezetben a `pip install PySide6` paranccsal . A program teljes funkcionalitásának kihasználásához a felhasználó számítógépének a bemeneti perifériák közül egérrel és billentyűzettel kell rendelkeznie. A szoftver hardverigénye nem igényel részletesebb specifikációt.

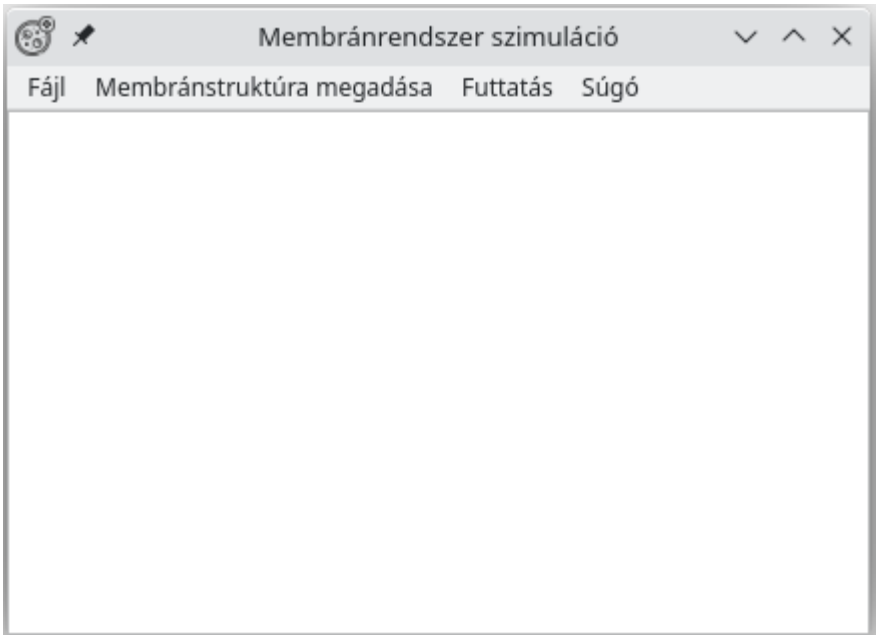
2.3. Futtatás

Mivel a program futtatható állományban kerül a felhasználóhoz, ezért annak az indításhoz elegendő megnyitni a fájlt tartalmazó mappát, majd duplán kattintani a fájlt reprezentáló ikonra. Ugyanez parancssori környezetben is elvégezhető, ilyenkor a terminálban a megfelelő mappába való elnavigálás után a `./MembraneSimulator` parancs megadásával futtatható a program.

2.4. Grafikus felhasználói felület

A felhasználó az alkalmazással a grafikus felhasználói felületen keresztül tud kommunikálni, amely a főablakot és az igény szerint megjelenő dialógusablakokat foglalja magába.

2.4.1. Főablak



2.1. ábra. A főablak az alkalmazás megnyitásakor

A főablak az alkalmazás megnyitásakor még tartalmaz egyetlen grafikus elemet sem, viszont a menüsorban található menüpontok segítségével könnyedén változtatni lehet ezen. Ha a felhasználónak nincs korábbi tapasztala a program használatával, akkor érdemes a *Súgó* menüpont kiválasztásával kezdenie, amelyről részletesen szó esik a 2.5 fejezetben.

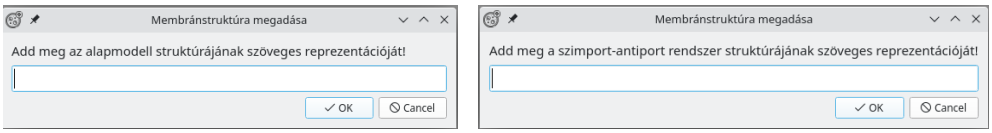
2.4.2. Dialógusablakok

A legfontosabb interakció a felhasználó és a szoftver között a dialógisablakokon keresztül történik.

Membránstruktúra megadása

Egy membránrendszer megalkotásának kezdeti módja a struktúrájának megadásával kezdődik. Mivel egy membránrendszerben a membránok hierarchikusan helyezkednek el, ezért a teljes rétegződést nagyon jól lehet ábrázolni fa alakban, ahol mindenkinek a szülő csúcsa az őt legszűkebben tartalmazó régió. Ezzel egyenértékű az a felírás, amikor egyetlen karakterláncban fejezzük ki ugyanezt, azáltal, hogy egy régiót megfeleltetünk egy nyitó-csukó zárójelpárral. Ilyenkor a két zárójel között elhelyezett objektumok jelentik a régió tartalmát. Azonban nem csak objektumok,

de más régiók is helyet kaphatnak, ezzel kifejezve azt, hogy a már említett régió közvetlen gyerekét szeretnénk megadni.



(a) Dialógusablak alapmodell létrehozásához

(b) Dialógusablak szimport-antiport rendszer létrehozásához

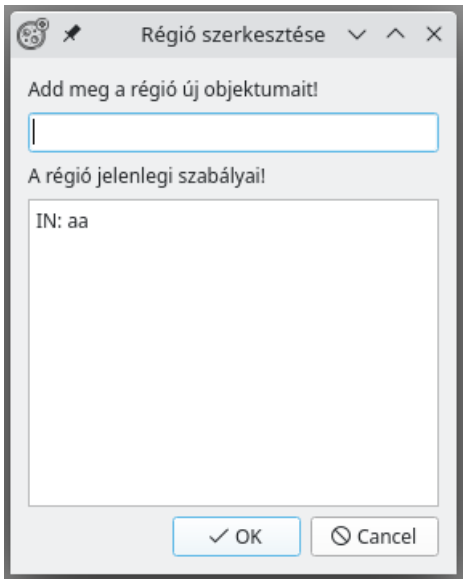
2.2. ábra. Dialógusablakok membránrendszer létrehozásához

A 2.2 ábra a különböző típusú membránrendszerek struktúrájának megadásához használt dialógusablakot mutatja. A megadott karakterláncban a régiók kezdetét és végét jelző zárójelpárok, illetve a bennük előforduló objektumok szerepelhetnek. A helyes formátum feltétele, hogy a zárójelpárok karakterein és az szimport-antiport rendszereknél a kimeneti régió jelzésére használt speciális # karakteren kívül csak az angol ábécé kisbetűi szerepelhetnek a bemenetben, illetve annak meg kell felelnie a helyes zárójelezés szabályainak. Ezt azt jelenti, hogy nem lehetnek átfedések régiók között, azaz olyan karakterláncok, amikor egy külső régió hamarabb kerül lezárásra, mint bármelyik benne lévő régió. Ha ezen feltételeknek megfelel a felhasználó által megadott karakterlánc, akkor a főablak teljes területét lefedő vásznon megjelenik a létrehozott membránrendszer.

Ha a megadott karakterlánc nem a megfelelő formátumú, akkor a felhasználó figyelmeztetésére egy hibaüzenet jelenik meg a képernyőn.

Objektumok módosítása

Miután a felhasználó megkonstruálta a membránrendszer szerkezetét, utána lehetősége nyílik arra, hogy az egyes régióinak tartalmát szerkessze. Mivel minden egyes régióhoz tartozhatnak objektumok és szabályok is egyaránt, ezért ezek módosítása közös dialógusablakban hajtható végre. A dialógusablak megjelenése a szerkeszteni kívánt régió által lefedett területre történő dupla kattintással kezdeményezhető.

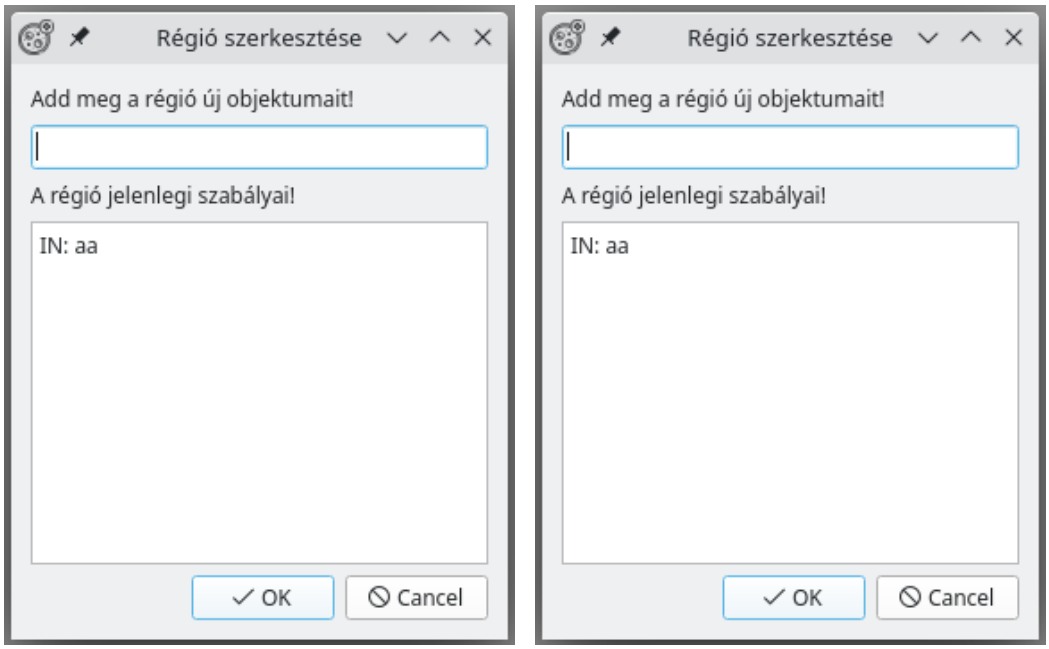


2.3. ábra. Dialógusablak egy régió tartalmának szerkesztéséhez

A régió objektumai közé tetszőleges számban írható a szóköz, mint elválasztó karakter, amely nem kerül figyelembevételre a régió új objektumainak feldolgozásakor. A megengedett karakterek ezen felül továbbra is az angol ábécé kisbetűi. Ha az adott régió a dialógusablak megnyitásakor már rendelkezik objektumokkal, akkor azok alapértelmezett szövegként fognak szerepelni a szövegdobozban. Tehát ha a régió szerkesztésekor az a felhasználó célja, hogy a kiválasztott régió tartalmát újabb objektumokkal egészítse ki, abban az esetben elegendő a hozzáadni kívánt karakterek begépelése.

Szabályok módosítása

A 2.2 segítségével alkotott membránrendszerek még nem fognak szabályokat tartalmazni, ám ezek nélkül nem beszélhetünk még szimulációról. A szabályok fontos szerepet kapnak az információáramlás vezérlésében, hiszen a hatásukra tudnak objektumok átalakulni, be- és kivándorolni a régiók között.



(a) Régió szabályainak szerkesztése alapmodellben

(b) Régió szabályainak szerkesztése szimport-antiport rendszerben

2.4. ábra. Membránrendszerek különböző formátumú szabályokkal

A szabályok felépítése függ a membránrendszer típusától. Minden szabály megadása során először fel kell tüntetni, hogy mik azok az objektumok (olyankor ugyanolyan objektumból akár több is egyszerre), amelyek szükségesek a szabály alkalmazásához.

Szimport-antiport rendszereknél ez a feltétel teljes mértékben elegendő ahhoz, hogy a szabály létrejöjjön, hiszen abban a modellben nincs lehetőség új objektumok létrehozására; a hangsúly inkább az objektumok régiók közötti mozgatására helyeződik. Ahogyan a 2.4b alsó szövegdobozában is látható, szimport-antiport rendszerek esetében két különböző típusú szabály hozható létre. Az első a szimport szabály, amelynek két formája van. Az *IN:* kezdetű szabály alkalmazása esetén a kijelölt objektumok a membránt körülvevő tartományból a jelenleg szerkesztés alatt álló tartományba vándorolnak; az *OUT:* kezdetű szabálynál pedig a szerkesztés alatt álló régióból a membránt körülvevő tartományba kerülnek az objektumok. A másik típushoz az *ún.* antiport szabályokat soroljuk, akkor egyszerre történik meg az előbbiekkel analóg módon be- és kivándorlás. A bevándorló objektumok előtt az *IN:* prefix szerepel, a kivándorló objektumok pedig az *OUT:* után találhatóak. Ezen két rész a karakterláncon belül felcserélhető, illetve köztük és az objektumok között tetszőleges számú szóköz karakter szerepelhet, amelyek a szabály megalkotásakor nem

játszanak szerepet.

Ahogy a 2.4a ábra is mutatja, az alapmodell esetében a szabályok komplexebb felépítéssel rendelkezhetnek. Első nagy különbség, hogy ebben a modellben objektumok átalakulhatnak, illetve eltűnhetnek vagy megsokszorozódhatnak. Ebben a modellben a szabály alkalmazásához szükséges objektumokat a szabály *bal oldalának* nevezzük. A szabály meghatározása ezen objektumok felírásával kezdődik. A szabály alkalmazásának köszönhetően létrejövő objektumok halmazát a nevezzük a szabály *jobb oldalának*, amelyben az objektumok mozgásának figyelembevételével 3 csoportra bonthatunk:

1. Kivándorló objektumok, amelyek a szülő régióba fognak jutni. Ha a legkülső régióban alkalmaztuk a szabályt, akkor az ilyen címkéjű objektumok a környezetbe kerülnek. A szabály megalkotásánál ezen objektumok elé az *OUT:* prefix kerül.
2. Bevándorló objektumok, amelyek véletlenszerűen valamelyik gyerek régióba fognak kerülni (az alapmodell ennél specifikusabban, címke alapján is megengedi az objektumok mozgását). A szöveges formátumban ezen objektumokat az *IN:* szöveg előzi meg.
3. Régión belül létrejövő objektumok, amelyek a szabályhoz tartozó régióba kerülnek. Ezen objektumok elé a karakterláncban a *HERE:* kifejezés kell álljon.

A szabály bal és jobb oldalát a -> szimbólumok választják el, amelyek együttesen egy nyilat reprezentálnak, annak kifejezésére, hogy a bal oldalt a jobb oldal objektumai „váljták fel”. Tehát ekkor a *aa -> IN: bb OUT: cc HERE: dd* szöveghez tartozó alapmodellbeli szabály két a objektumot igényel a végbemeneteléhez (amelyek felemésztődnek a reakció hatására), a régióból kivándorol két b objektum, az egyik gyerek régióba bevándorol két c objektum, illetve régión belül kialakul két új d objektum.

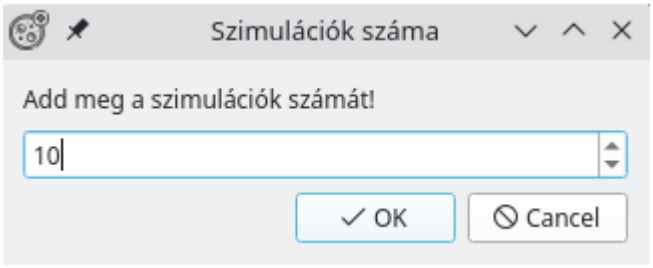
Az alapmodell szabályai kiegészülhetnek többletjelentéssel is. Ebben a modellben ugyanis megengedett lehet egy régió felbomlása, amely egy vagy egyszerre több szabály előidézésére is bekövetkezhet. Ez a szándék jelezhető egy szabály konstruálásakor, ha a bal- és jobb oldalt elválasztó nyilat reprezentáló karakter után közvetlenül egy # karaktert ír a felhasználó. Ilyenkor a régió tartalma és gyerek régiói a szülő régiójába kerülnek.

Szimulációk számának megadása

A korábbi dialógusablakok segítségével a felhasználó lehetőséget szerzett arra, hogy a teljes mértékben a saját elképzeléseinek megfelelő membránrendszer hozza létre és konfigurálja fel objektumokkal és szabályokkal. Ezek után már csak a rendszer számításának szimulálása van hátra. A szimulálás végrehajtására két módot biztosít az alkalmazás:

1. Teljes szimuláció futtatása párhuzamosan tetszőleges számú másolattal
2. Egy szimulációs lépés az aktuális membránrendszeren

Fontos megjegyezni, hogy a teljes szimuláció futtatásának kiválasztása esetén a képernyőn látható membránrendszer nem kerül szimulálásra, annak érdekében, hogy a szimulálni kívánt állapot megőrződjön, így utána változtatások nélkül kezdeményezhető a funkció megismétlése.

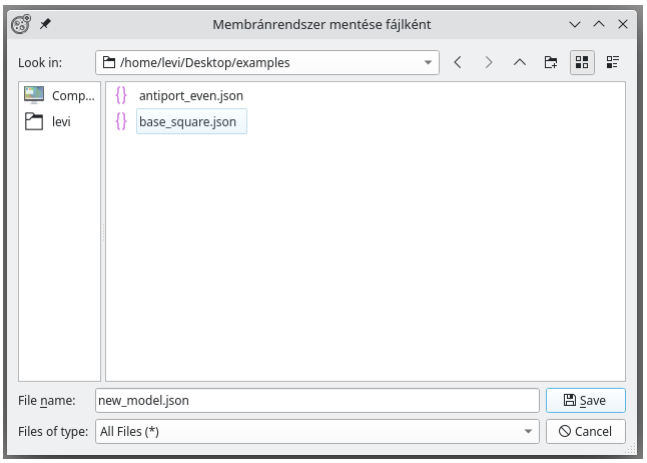


2.5. ábra. Dialógusablak a szimulációk számának kiválasztásához

A szimuláció ismétléseinek számát egy olyan speciális szövegdobozzal lehet beállítani, amely bemenetként csak numerikus értékeket reprezentáló karaktereket fogad el. Alternatívaként a szövegdoboz jobb oldalán található nyilakkal is egyesével állítható az érték, amelynek alsó korlátja *1*, felső korlátja pedig *1000*.

Mentés

Miután a felhasználó kialakított egy olyan membránrendszert, amelyet szeretne elemezni vagy későbbi használat céljából megőrizni, az alkalmazás lehetőséget biztosít a jelenlegi állapot fájlba történő lementésére.



2.6. ábra. Dialógusablak a membránrendszer mentéséhez

A mentés folyamata a *Mentés* gombra történő kattintással vagy a `Ctrl+S` billentyűkombináció lenyomásával kezdeményezhető. A funkció kiválasztása után egy dialógusablak jelenik meg, amelyben a mentés helyére kell navigálnia a fájlrendszerben, majd a fájlnév megadása után az *OK* gomb lenyomására létrejön a *JSON* formátumú fájl a megadott útvonallal.

Mivel a *JSON* formátum könnyen értelmezhető és szerkeszthető, ezért egy jártasabb felhasználója a programnak képes egy lementett membránrendszeren módosításokat végezni a megfelelő szerkesztésekkel.

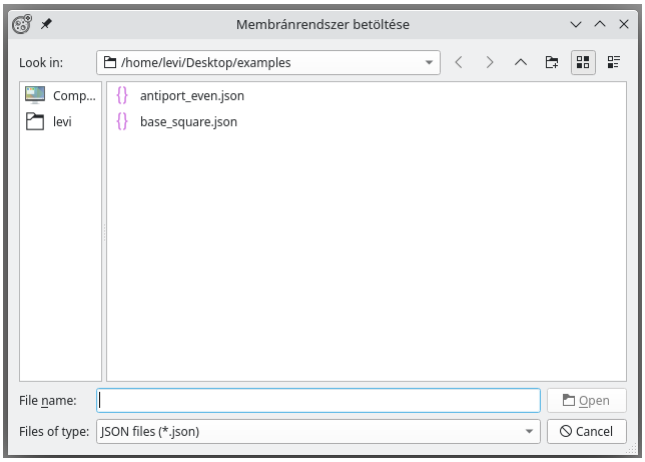
```
{
  "type": "SymportAntiport",
  "structure": "[[[#]]]",
  "rules": {
    "0": [
      "OUT: b",
      "IN: ab"
    ],
    "1": [
      "IN: aa",
      "IN: b OUT: c"
    ],
    "2": [
      "IN: aa"
    ]
  },
  "env_obj": "",
  "env_inf": "a",
  "objects": {
    "0": "b",
    "1": "c",
    "2": ""
  }
}
```

2.7. ábra. Egy lementett membránrendszerhez tartozó JSON fájl [1]

[1] Az ábrán látható konfiguráció a páros számokat generáló szimport-antiport rendszerhez tartozik

Betöltés

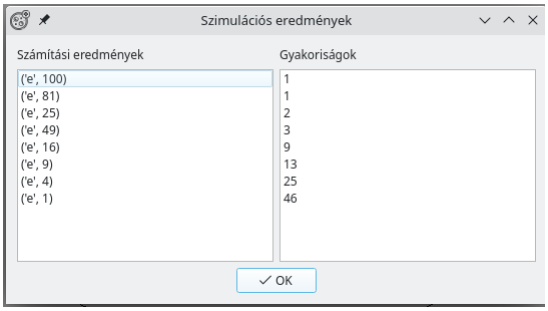
Az alkalmazás biztosítja lementett membránrendszerek betöltését, amelynek hatására betöltődik a kiválasztott fájlhoz tartozó állapot a grafikus felületre. A betöltés nagyon hasonlóan működik a mentéshez, kezdeményezni a *Betöltés* gombra történő kattintással vagy a `Ctrl+O` billentyűkombináció lenyomásával lehetséges, mely után egy dialógusablak segítségével a felhasználó elnavigálhat a megnyitni kívánt fájlt tartalmazó mappához.



2.8. ábra. Dialógusablak egy membránrendszer betöltéséhez

Eredményablak

Miután a felhasználó létrehozott egy membránrendszert és szimulálta azt, nincs más hátra mint értesülnie annak eredményéről. Ezt a funkciót látja el az eredményablak, amely a membránrendszer számításának befejeződése után automatikusan megjelenik és egy listában gyakoriság szerint növekvő sorrendbe rendezve jeleníti meg a különböző számítási eredményeket.



2.9. ábra. Szimulációs eredményeket kilistázó ablak

2.5. Használati útmutató

A szoftver használata a membránrendszerek előzetes ismerete nélkül kriptikus tud lenni, ezért a felhasználó útbaigazítására és a szükséges jellemzőkkel kapcsolatos tudnivalók ismertetésére egy súgó funkció is megtalálható a menüsorban.

A használati útmutató a *Markdown* formátumnak megfelelő formázásban jelenik meg, amely jól strukturált és könnyen olvasható. A tartalma fejezetekre van bontva, amelyek kitérnek a korábban említett dialógusablakok sajátosságaira, az objektumokkal és szabályokkal kapcsolatos elvárásokra és a szoftver helyes használatának módjára.

3. fejezet

Fejlesztői dokumentáció

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis nibh leo, dapibus in elementum nec, aliquet id sem. Suspendisse potenti. Nullam sit amet consectetur nibh. Donec scelerisque varius turpis at tincidunt.

3.1. Tételek, definíciók, megjegyzések

1. Definíció. Mauris tristique sollicitudin ultrices. Etiam tristique quam sit amet metus dictum imperdiet. Nunc id lorem sed nisl pulvinar aliquet vitae quis arcu. Morbi iaculis eleifend porttitor.

Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit, et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna. Phasellus faucibus varius purus, nec tristique enim porta vitae.

1. Tétel. *Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia. Etiam vel odio ante.*

Bizonyítás. Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui. □

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod, faucibus lectus sed, accumsan felis.

Emlékeztető. Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec. Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus.

Fusce in aliquet neque, in pretium sem. Donec tincidunt tellus id lectus pretium fringilla. Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris. Donec pretium et quam a cursus.

Megjegyzés. Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula.

Ut sollicitudin tempus urna et mollis. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam. Donec eget tellus pharetra, semper neque eget, rutrum diam.

3.1.1. Egyenletek, matematika

Duis suscipit ipsum nec urna blandit, $2 + 2 = 4$ pellentesque vehicula quam fringilla. Vivamus euismod, lectus sit amet euismod viverra, dolor metus consequat sapien, ut hendrerit nisl nulla id nisi. Nam in leo eu quam sollicitudin semper a quis velit.

$$a^2 + b^2 = c^2$$

Phasellus mollis, elit sed convallis feugiat, dolor quam dapibus nibh, suscipit consectetur lacus risus quis sem. Vivamus scelerisque porta odio, vitae euismod dolor accumsan ut.

In mathematica, identitatem Euleri (equation est scriptor vti etiam notum) sit aequalitatem 3.1. egyenlet:

$$e^{i \times \pi} + 1 = 0 \tag{3.1}$$

3.2. Forráskódok

Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor. Nam viverra orci id sapien sollicitudin, a aliquam lacus suscipit. Quisque ac tincidunt leo 3.1. és 3.2. forráskód:

```
#include <stdio>

int main()
{
  int c;
  std::cout << "Hello World!" << std::endl;

  std::cout << "Press any key to exit." << std::endl;
  std::cin >> c;

  return 0;
}
```

3.1. forráskód. Hello World in C++

```
using System;
namespace HelloWorld
{
  class Hello
  {
    static void Main()
    {
      Console.WriteLine("Hello World!");

      Console.WriteLine("Press any key to exit.");
      Console.ReadKey();
    }
  }
}
```

3.2. forráskód. Hello World in C#

3.2.1. Algoritmusok

Az 1. algoritmus egy általános elágazás és korlátozás algoritmust (*Branch and Bound algorithm*) mutat be. A 3. lépésben egy megfelelő kiválasztási szabályt kell alkalmazni. Példa forrása: Acta Cybernetica (ez egy hiperlink).

1. algoritmus A general interval B&B algorithm

Funct IBB(S, f)

1: Set the working list $\mathcal{L}_W := \{S\}$ and the final list $\mathcal{L}_Q := \{\}$
2: **while** ($\mathcal{L}_W \neq \emptyset$) **do**
3: Select an interval X from $\mathcal{L}_W$ ▷ Selection rule
4: Compute $lbf(X)$ ▷ Bounding rule
5: **if** X cannot be eliminated **then** ▷ Elimination rule
6: Divide X into $X^j,\ j = 1, \ldots, p$, subintervals ▷ Division rule
7: **for** $j = 1, \ldots, p$ **do**
8: **if** X^j satisfies the termination criterion **then** ▷ Termination rule
9: Store X^j in $\mathcal{L}_W$
10: **else**
11: Store X^j in $\mathcal{L}_W$
12: **end if**
13: **end for**
14: **end if**
15: **end while**
16: **return** $\mathcal{L}_Q$

4. fejezet

Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

A. függelék

Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus,

sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

Ábrák jegyzéke

Táblázatok jegyzéke

Algoritmusjegyzék

Forráskódjegyzék