Az idő kezelésére érzékeny konkurens folyamatok programozásáról

On programming of extremely time sensitive concurrent processes

VÉGH János

MTA Doktora, egyetemi tanár

Miskolci Egyetem, Vegh.Janos@gmail.com

Abstract

When simulating time-aware processes by a computer program, one must be aware of that the simulated time betwen simulated events is proportional neither to the processor time nor the wall-clock time; even, due to different technical implementations, their order can be changed that leads to causality problems. We demonstrate the required special technology and algorithmic thinking on the example of simulating biological neuronal operation, which is extremely sensitive to timing issues.

Keywords: time-aware programming; effects of workload; effects of architecture; algorithm and technology of precise timing; simulating biological structures;

Kivonat

A számítógépes programokkal leírandó folyamatok (idő-helyes szimulációk) esetén figyelembe kell vennünk, hogy két szimulált esemény között a valóságban eltelt (szimulált) idő, és annak számítógépes szimulálására fordított processzor idő, valamit az óra idő egymással nem arányosak, sőt sorrendjük is felcserélődhet, ami oksági problémákat okozhat. A példaként bemutatott, az időzítési problémákra nagyon érzékeny biológiai neurális működés szimulálása speciális technológiát és eltérő algoritmikus gondolkodást is igényel.

Kulcsszavak: idő-helyes programozás; a feladat hatása; az architektúra hatása; a precíz időzítés technológiája és algoritmusa; biológiai struktúrák szimulálása;

1. BEVEZETÉS

Schrödinger nevezetes „What is Life?” könyvének [1] évfordulója kapcsán érdemes felidézni a lényeget érintő kérdését: "How can the *events* in *space and time* which take place *within the spatial boundary* of a *living organism* be *accounted for by physics* and chemistry?". Azaz, ha le akarjuk írni az életet, akkor nem csak matematilag és fizikailag helyes modellt kell alkotnunk (figyelembe véve, hogy egymással összefüggő lassú folyamok eltérő helyeken okoznak változásokat), hanem annak szimulációjakor a helyes időbeli viselkedést is garantálnunk kell. A szimulált viselkedés igényelt időskálája a szub-atomi szintű folyamatok femtoszekundumos idejű történéseitől a geológia vagy kozmosz kutatás évmilliós nagyságú eseményeiig terjed. A számítógép programok végrehajtási idői természetes formájukban egyik időskála szimulálására sem alkalmasak, a szükséges speciális programozási módszerek viszont eltérő algoritmikus gondolkodást igényelnek. A szimulálással kapcsolatos problémák bemutatására az élő anyag, konkrétan a biológiai neuron viselkedés leírása algoritmikus nehézségeinek bemutatását választottuk. A viselkedés bővebb leírását [2] tartalmazza, itt csak annak kifejezetten algoritmikus vonatkozására koncentrálunk. A technikai megvalósítás nehézségeit, a biológia neuronok technikai ábrázolásával kapcsolatos félreértéseket, valamint a kétféle megvalósítási módból következő működésbeli eltéréseket részletesen a [3] irodalom tárgyalja.

2. IDŐ ÉS ESEMÉNY

Az idő szimulálásakor a központi fogalom az esemény, ami – bár eltérő szavakkal – lényegében hasonlóképpen fogalmazódik meg a szimulált és a szimulálandó eseményre vonatkozóan: valami, valamikor, valahol megtörténik, és hatást gyakorol a környezetére; a számítógépes program pedig valamikor egy ennek megfelelő, jól meghatározható utasításcsoportot hajt végre. A szimulált esemény és annak számítógépes

megvalósítása egymáshoz rendelése könnyen megvalósítható, és látszólag időt is könnyen hozzájuk tudunk rendelni. Időből azonban többféléről is beszélnünk kell

- *szimulált* (azaz logikai) idő, amit a biológusok rendelnek az eseményekhez
- *óraidő*, amit a programozó olvas le az órájáról, amikor a program a megfelelő számítógépes eseményeket szimuláló kódhoz ér
- *hasznos processzor idő*, amit a processzorok és szálak az eseményekhez közvetlenül szükséges műveletek elvégzésével töltenek (órajel frekvencia és kód függő)
- *nem közvetlenül hasznosított processzor idő* (rendszer műveletek, átvitelek, várakozások)
- *(idő felbontás*: az az intervallum, amin belül eső időpontokat azonosnak tekintünk)

Eredendő probléma, hogy a rendszerek biológia működése egymástól függetlenül (párhuzamosan) történik, a számítógép viszont szekvenciális (különböző teljesítmény növelő megoldásokkal részben párhuzamosított) módon működik. Emellett a számítógépes hardver és szoftver megoldások, különösen a nagyobb számítógépes konfigurációkat igénylő számítások esetén, már nem teszik lehetővé hogy az események biológiai és számítógéppel szimulált folyamatok végrehajtási ideje arányos legyen egymással. A biológiai rendszerek számára megosztott számítógépes erőforrásokat tudunk biztosítani. A fő probléma a biológiai rendszerek kommunikációjának szimulálása, mivel a biológia lassú párhuzamos átviteleit gyors szekvenciális átvitellel próbálják megvalósítani. A kommunikációs feladatokat a számítógépek nagyrészt rendszerhívások, továbbá ki- és beviteli utasítások formájában valósítják meg. Az eredmény elküldése azzal jár, hogy a számítógép számítási egységei igen gyakran cserélnek adatot számítás közben, ami műveleteket a számítástudomány eredetileg csupán a láncolt számítások elvégzése előtti adatbevitelre és utána eredmény kivitelre vett számba. Ez az eltérés a számítógép viselkedését ez elméletileg elvárthoz képest nagyon eltérővé teszi; minél nagyobb a kommunikáció aránya a hasznos számításhoz képest, annál eltérőbbé. A kétféle „számítás” eltérő megvalósítási módja, mint arra már Neumann János felhívta a figyelmet, a számítási és átviteli idők eltérő aránya miatt már elvileg sem egyszerűen képezhető le egymásra. A technikai kivitel miatt az események időpontjai akár fel is cserélődhetnek, oksági problémákat is okozva.

A számítógép működés egyik sokat kritizált, de szükségszerű, eleme [3] a komponenseket összekötő „busz” (ezt főként a memóriával kapcsolatban a „Neumann-féle szűk keresztmetszet” néven emlegetik), aminek működéséhez az is hozzátartozik, hogy mint közösen használt erőforrást, az átvitel idejére az adatátvitel két partnerének ki kell sajátítani a buszt, csak azután lehet adatot küldeni és fogadni. Sok objektum átviteli igénye esetén minden egyes igény kielégítése előtt egyfajta versenytárgyalás (arbitration) folyik, amelynek ideje az átvitelre várakozó partnerek számával lineárisan nő, így nagy rendszerek esetén a tényleges átviteli idő csupán törtrésze a teljes busz használati időnek. Hasonlóképpen lecsökken a tényleges átviteli sebesség is. Ráadásul a busz használati jogot az operációs rendszertől kell kérni, ami tízezres nagyságú gépi utasítás végrehajtásával jár. A buszt kis rendszerek alkalmi átvitelére tervezték. Nagy rendszerek esetén a kommunikációs igény az üzenetek számának korlátozását igényli (hardveres vagy szoftveres „ritkítás” (pruning)), mert a túlzott kommunikáció a rendszer észszerűtlenül gazdaságtalan működésével vagy akár bénulásával is járhat. A ritkításnak különböző megvalósítási módszerei és alapötletei vannak. Mindegyik az információ csonkításával jár és alkalmazásakor egyedileg meg kell vizsgálni annak hatásait.

Együttműködő objektum rendszerek (például mesterséges neuronok hálózata vagy biológiai rendszerek szimulációja) esetén, a párhuzamos és a párhuzamosított szekvenciális működést össze kell hangolni. Az idő hagyományos kezelésének klasszikus (és szokásos) módszere annak feltételezésén alapszik, hogy a szimulált és a számítási idő egymással arányosak. A számítást megadott hosszúságú időszakaszokban (rácsidő) végzik, és a szakasz végén a számító egységek egymásnak kölcsönösen elküldik eredményeiket vagy pedig a számítást és az eredmények elküldését/feldolgozását folyamatosan végzik, a továbbítást, feldolgozást és időzítést a számítógépes rendszer – más célra készült – erőforrásaira bízva. A számítógép nagyságának növekedésével nyilvánvalóvá vált, hogy a számítógépes rendszer növekedésével ez az arányosság egyre kevésbé teljesül; ennek kiküszöbölésre irányult az „időbélyegzés” használata. Azaz, a küldési módszer lényegében változatlan, de a küldő rendszer az üzenetbe beírja az esemény bekövetkezésének óraidejét. Ez lényegében csupán áttolja a felelősséget az eseményeket feldolgozó egységre, amelyik ugyan tudja keletkezési idő szerinti sorrendben feldolgozni az addig beérkezett eseményeket, de nem oldja meg a még be nem érkezett események feldolgozásának problémáját. Mivel a problémát alapvetően az üzenetek nagy száma okozza, jelentősen javult a helyzet, amikor a biológiai neuronokat szimuláló mesterséges neuronok működését egyetlen üzenet elküldésére korlátozták (Spiking Neural Networks). Ami változtatás hatására az említett problémák csak jelentősen nagyobb számú objektum esetén, de ugyanúgy jelentkeznek. Ennek ára,

hogy a neuron állapotváltozójának információ tartalma elvész; a neuron által közvetített idő információ pedig elveszíti valóság tartalmát: az idő túlnyomórészt a technikai működéstől és csak kevéssé a szimulált biológiai működéstől függ (bár a processzor idő akár nanoszekundum pontossággal is beírható az időbélyegbe, a mérés kezdetét kvázi-véletlenszerű technikai folyamatok határozzák meg, jellemzően milliszekundum pontossággal). A biológiai neuronhálózat a tüzelési idők közötti különbség kb. 200-ad részének megfelelő pontosságú időzítésre képes; továbbá a logikai és az időbélyegben szereplő óraidő közötti kapcsolat esetleges. Az óraidő a biológiai információ átvitelére egyáltalán nem alkalmas; más célú felhasználásának lehetőségét egyedileg, műszaki paraméterektől függően, kell vizsgálni.

3. A NEM-HASZNOS IDŐ ALGORITMIKUS HATÁSA

Mivel a számítógépes végrehajtás gyorsaságát és hatékonyságát óraidővel mérik, a nem-hasznos idők jelentősen befolyásolják a számítás hatékonyságát, és az algoritmikus és tapasztalati hatékonyságok jelentősen eltérnek egymástól. A számítógépes rendszerek hardver műszaki paraméterei csupán a teljesítőképesség felső határát adják meg, annak tényleges kihasználása, a szoftver hatékonyság ennek csekély törtrésze is lehet. Bonyolult (főként: sokat kommunikáló) szoftver sok indokolatlan (nem közvetlenül hasznosan töltött) idő járulékot is tartalmaz. Ezt mutatja a neuronhálózatok kommunikációjára vonatkozó ismert klasszikus algoritmus, amelynek félkövér betűkkel szedett sorai algoritmikusan nem lényegesek, de időbeli járulékuk meghatározza az algoritmus használatának hatékonyságát. (a LaTeX-ben szokásos algoritmikus jelölésmódhoz hasonló módon leírva)

```
\State{t <= 0}
\For{every neuron i}
        \State{Compute timing of next spike}
        \State{Insert event in priority queue}
        \State{Queue handling and I/O}
\EndFor
\State{t <= 0}
\While{queue not empty and t < duration}
        \State{Queue handling and I/O}
        \State{Extract event with lowest timing}
        \State{(event=timing t, neuron i)}
        \State{Schedule neuron}
        \State{Compute state of neuron i at time t}
        \State{Reset membrane potential}
        \State{Compute timing of next spike}
        \State{Insert event in queue}
        \State{Queue handling and I/O}
        \For{every connection i => j}
                \State{Compute state of neuron j at time t}
                \State{Schedule neuron}
                \State{Change state with weight w(i,j)}
                \State{Compute timing of next spike}
                \State{Insert/change/suppress event in the queue}
                \State{Queue handling and I/O}
        \EndFor
\EndWhile
```

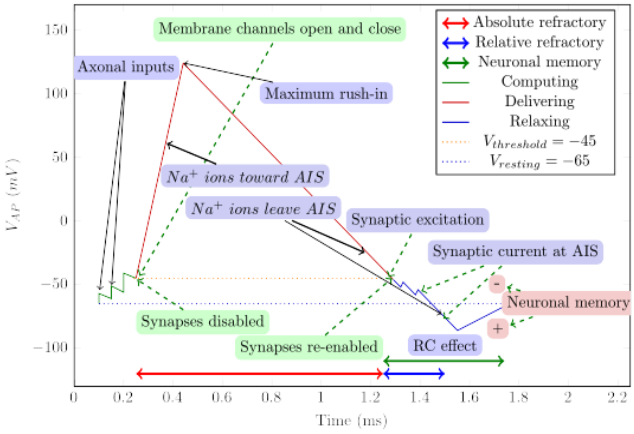
4.

5. BIOLÓGIAI FOLYAMAT IDŐ-HŰ LEÍRÁSA

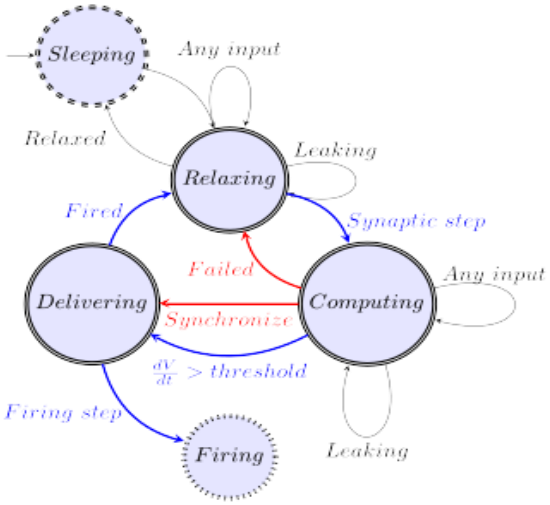
Az élet mibenlétének megfogalmazása még ma is misztikus a kutatás számára. A biológiai neuronokban nem lehet olyan egyszerűen eseményeket definiálni, amint azt a klasszikus természettudományokban megszoktuk (Schrödinger megfogalmazása szerint: a „konstrukció” teljesen más). Az élő anyagban az állapot változó(k) értékét belső folyamatok folyamatosan változtatják, ezért amikor az állapot változó értékét leírjuk, annak értéke csak a pontos biológiai idő ismeretében értelmezhető (a megadott reláció

csak egy bizonyos időpontban érvényes), a matematika szokásos relációi pedig az időt nem tartalmazzák. Emiatt a leíráskor különböző biológiai objektumok eseményeit csak akkor tudjuk egymással kapcsolatba hozni, ha nem csak a változó értéke, hanem annak mérési ideje is megegyezik. Ez a körülmény szigorú korlátokat szab az események és azok időpontjai egymáshoz rendelésével szemben. Pl. a bemutatáshoz használt állapotgép (l. 2. ábra) nagyon hasonlít a programozásban megszokotthoz, de bonyolultsága és pontos értelmezése nagyon eltér.

A modern nagy sebességű áramkörök tervezésekor, a biológiai működéshez nagyon hasonló értelemben, nagyon pontosan kell meghatározni a működés időzítési viszonyait. A feladatra az ennek megoldásában érdekelt vezető elektronikai tervező intézmények létrehoztak egy saját ütemezővel és eseménykezelővel is rendelkező, C^{++} nyelven készített, SystemC könyvtárat [4]. A rendszer saját ütemezője (beleértve az események idősorokba rendezését is) biztosítja, hogy a logikai időben (megválasztható időfelbontással) azonosnak tekintendő események a számítógépes rendszerben (architektúrától és számítási bonyolultságtól függetlenül) azonos (logikai, azaz szimulált) időben hajtódjanak végre, annak ellenére, hogy a számítógépes rendszer processzorai és szálai valójában fizikailag különböző időpontokban dolgozzák fel az eseményeket.

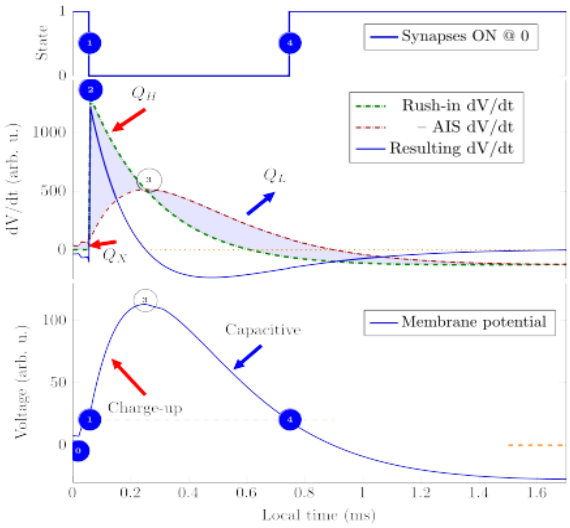


1. ábra: *Az akciós potenciál keltésének sematikus ábrázolása, a biológiailag releváns folyamatok és fogalmak feltüntetésével*

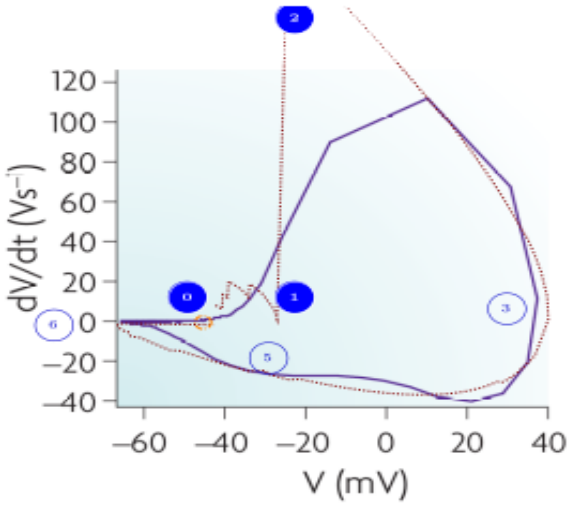


2. ábra: *Az akciós potenciál keltésének, lásd. 1. ábra. folyamatábrája*

A neuronban lezajló folyamatok a fizikának nem szokásos (Schrödinger által 'non-ordinary'-nek nevezett) törvényeivel írhatók le amelyeknek bemutatása itt nem célunk (annak részleteit [5] adja meg); itt csupán az időzítés leírására korlátozódunk. A leírhatóság érdekében, biológiailag és fizikailag is plauzibilis módon, a folyamatot az 1. ábrán bemutatott alfolyamatokból rakjuk össze. A neuronok nevezetes ún. akciós potenciálja három fő fázisra bomlik. Az állapotváltozók a fázisokon belül is folyamatosan változnak, így a 2. ábrán bemutatott, ebből képezett állapot diagram is szokatlanul bonyolult. A biológiai folyamatok jól leírható változásokat eredményeznek, de csak egy kritikus érték túllépése után vezetnek másik fázisba, odáig a folyamat eredményeként a biológiai rendszer állapota csak kisebb mértékben változik. A biológiai folyamat nagyobbrészt folytonos és sok más folyamatot (energia termelés és utánpótlás, állapot szabályozás) is csak szimbolikusan tüntettünk fel. Ráadásul bizonyos körülmények között másféle átmenetek is lehetségesek.



3. ábra: Az algoritmussal előállított akciós potenciál (lent), gradiensek (középütt) és vezérlőjelek (fent)



4. ábra: A feszültség és annak gradiense, kísérletileg és elméletileg. Törött vonal mutatja a integrációs lépés nagysága megválasztásának fontosságát.

A fő algoritmus

```
\State{ //Calculate stage-dependent contributions}
\If{Stage "Relaxing"}
  \If{(RelaxingStopped || !RushinCurrent)}
    \State{dV/dt  <- 0}
  \Else
    \State{dV/dt  <=  rushin derivált (t)}
  \EndIf
\EndIf
\If{Stage "Computing"}
  \State{dV/dt  <=  0}
  \State{dV/dt  <=   PSP derivált(t)}
\EndIf
\If{Stage "Delivering"}
  \State{SynapsesEnabled   <=  (V_Membrane<V_Threshold) }
  \State{dV/dt Rushin <= PSPderivative(t)}
  \State{dV/dt Input <=  PSPderivative(t)_i}
\EndIf
\State{dV/dt_Resulting  <= Sum of inputs  }
\State{dV/dt_AIS  <=  for the output current at time}
\State{V_Membrane  <=  dV/dt_Resulting* dt }
\State{Adjust heartbeat time and set new time}
```

A „heartbeat” kód:

```
  \If{Stage "Relaxing"}
    \If{ReceivedSynapticInput }
      \State{Send 'BeginComputing'}
    \EndIf
    \If{dV/dt_Membrane < 0}
      \If{abs(V_Membrane)< Allowed}
       \State{ V_Membrane <= 0}
       \State{RelaxingStopped <= true}
      \EndIf
    \EndIf
  \EndIf
  \If{Stage "Computing"}
```

```
    \If{V_{Membrane} < 0 $}
      \State{Send 'Failed'}
    \EndIf
  \Else
    \If{V_{Membrane} >= V_{Threshold}}
      \State{SynapsesEnabled <=  false}
      \State{Send 'DeliveringBegin'}
    \EndIf
  \EndIf
  \If{Stage "Delivering"}
    \If{$V_{Membrane} < V_{Threshold}$}
      \State{$RelaxingStopped \Leftarrow true$}
      \State{$SynapsesEnabled \leftarrow true$}
      \State{Start 'RelaxingBegin'}
    \EndIf
\EndIf
  \State{Do main computing in Algorithm~(\ref{alg:Main})}
  \State{Send new heartbeat}
```

A folyamatot a nagyon gyorsan beinjektált töltések által előidézett potenciál gradiens irányítja, amelyik ugrás-szerűen kezdődik, és élesen változik (lásd 3. ábra), míg a kimeneti áram gradiense elegendően simán változik az idő függvényében. Emiatt változtatni kell azt az időközt, amelyek mentén a függvény alakot kiszámoljuk. A biológiai neuron alapvetően három állapottal valósítja meg azt a „számítást”, ami egyetlen neuron esetén (töltés integrálással megvalósított) időmérést jelent. Bonyolultabb műveleteket csupán megfelelően csoportosított és együttműködő neuronok képesek végezni.

A program bizonyos időközönként rápillant a feladatra („heartbeat” kód): lényegében beállítja az integrációs lépés nagyságát és elvégzi az üzemi fázisok beállítását. A meredeken változó gradiensek és a állapot változótól függő szakaszhatárok miatt az algoritmus érzékeny a lépés nagyságára: a kisülés-jellegű rushin esetében a gradiens exponenciálisan esik, illetve a küszöb túllépésének ideje (ami a működés lényegi információja) nem állapítható meg pontosabban, mint a „heartbeat” lépésnagyság. Az SNN esetén éppen ez a legnagyobb probléma: az esemény idő felbontása eleve nem lehet jobb, mint a rendszer jellemzően millisecundum nagyságrendű időszelete, amit az említett késletetések akár a sokszorosára is növelhetnek. Bár az időpecsét értéke akár nanosecundum pontosságú is lehet, annak csak áttételesen van köze a szimulálandó biológiai esemény idejéhez. Azaz, biológiai célra a számítógépes kóddal előállított időzítés nem használható; technikai célra pedig csak akkor, ha az eszköz által igényelt minimális időfelbontás értéke lényegesen hosszabb (például robotika), mint a rendszer által biztosított időszelet hosszúsága.

6. ÖSSZEFOGLALÁS

Egy valós fizikai környezettel kapcsolatban álló rendszernek (biológiai vagy másféle működést szimuláló program, robotok, önvezető autók, stb.) össze kell egyeztetni a környezet és a számítógép program által produkált események idejét. Ez a kétféle rendszer alapvetően eltérő működési elvei miatt igen nehéz feladat, és hagyományos programozási eszközökkel nem is oldható meg megfelelő pontossággal, ha az igényelt időzítési pontosság a számítógépes időszelet nagyságánál jóval kisebb. Az igényelt számítógép kapacitásnak a pontossággal való összehasonlítása sem egyszerű feladat. A fenti feladatok megoldásának nehézségeit és a megoldás lehetőségeit a biológia neuron működése számítógépes szimulációjának megoldásán keresztül mutattuk be.

IRODALMI HIVATKOZÁSOK

[1] Schrödinger, E: Is life based on the laws of psysics? (What is Life?), Cambridge University Press. (1992) 76–85, https://archive.org/details/WhatIsLife-EdwardSchrodinger/ (Utolsó letöltés: 2025.09.01)
[2] Végh J.: Algorithm for describing neuronal electric operation. Algorithms (2025) (elfogadott közlemény, megjelenés alatt) 2025 https://doi.org/10.20944/preprints202506.1290.v1 (Utolsó letöltés: 2025.09.01)
[3] Végh J.: On Implementing Technomorph Biology for Inefficient Computing. Applied Sciences, 15/11(2025)5805 https://www.mdpi.com/2076-3417/15/11/5805 (Utolsó letöltés: 2025.09.01)
[4] Black C. D., Donovan J, Bunton B., Keist A: SystemC: From the Ground Up. Springer, 2010. 978-0-387-69957-8
[5] Végh J.: Can 'non-ordinary laws of physics' describe living matter? Entropy (2025) in review http://dx.doi.org/10.2139/ssrn.5217729