Masarykova univerzita
Fakulta informatiky

Visualisace modelů a biochemického prostoru v SBGN

Bakalárska práca

Michal Štefanič

Brno, jeseň 2018

Masarykova univerzita
Fakulta informatiky

Visualisace modelů a biochemického prostoru v SBGN

Bakalárska práca

Michal Štefanič

Brno, jeseň 2018

Na tomto mieste sa v tlačenej práci nachádza oficiálne podpísané zadanie práce a vyhlásenie autora školského diela.

Vyhlásenie

Vyhlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracovala samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používala alebo z nich čerpala, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Michal Štefanič

Vedúci práce: RNDr. Ph.D. David Šafránek

Poďakovanie

These are the acknowledgements for my thesis, which can span multiple paragraphs.

Zhrnutie

This is the abstract of my thesis, which can
span multiple paragraphs.

Kľúčové slová

SBGN, BCSL, vizualizácia, e-cyanobacterium.org, API …

Obsah

Zoznam tabuliek

Zoznam obrázkov

Úvod

1 Základné pojmy

1.1 Systémová biológia

Pod pojmom systémová biológia si mnohí ľudia nevedia čo predstaviť. Dokonca aj keby by sme sa spýtali dvoch systémových biológov, aby definovali svoj obor, dostali by sme pravdepodobne dve rôzne odpovede. Za tuto diverzitu môže mladý vek systémovej biológie oproti iným príbuzným oborom.

Systémová biológia predstavuje holistický prístup popisujúci komplexitu biologických systémov, ktorý začína pochopením, že biologický systém je niečo viac ako len súčet jeho menších častí. Živý organizmus, ale aj jeho menšie komponenty akými sú bunka, či jadro, sú v systémovej biológii chápané ako systém. Dôležitú úlohu v týchto systémoch majú vzájomné interakcie medzi jeho časťami a *emergentné vlastnosti*, čo sú vlastnosti systému, ktoré nemôžu byť pochopené len na základe súčtu vlastnosti jeho menších časti, ale predstavujú vlastnosti systému ako celku - komplexu. Systémová biológia sa pohybuje na bunkovej a molekulárnej úrovni a spája dokopy viacero vedných oborov ako biológiu, matematiku, informatiku, chémiu či fyziku. Pracuje s veľkým množstvom dát, ktoré je potrebné spracovávať, uložiť a analyzovať. To je jeden z dôvodov, prečo je systémová biológia tak blízko prepojená prepojená s informatikou. Za predpokladu, že by sa všetky dáta spracovávali ručne a následne by na nich boli vykonávane analýzy a experimenty, trvalo by to roky a progres by sa dostavoval veľmi pomaly. To predstavuje jeden z hlavných prínosov informatiky v systémovej biológii. Aby vedci nemuseli vykonávať všetky tieto experimenty v laboratóriách na živých organizmoch, čo by bolo ako nákladne, tak aj zdĺhavé, vytvorí sa na základe dát a informácií získaných či už sekvenovaním DNA alebo inými biochemickými postupmi model biologického systému, ktorý reprezentuje organizmus alebo časť organizmu, a hraje kľúčovú rolu v systémovej biológií. Vytváranie týchto modelov je dôležité pre pochopenie vlastnosti organizmov, správanie sa systémov pri zmenách podmienok z rôznych príčin, či snaha predikovať tieto vlastnosti pomocou analýzy a simulácií. Niektoré experimenty sa ťažko vykonávajú v laboratórnych podmienkach

a určite experimenty by ani nebolo možne simulovať bez pomoci informatiky.

1.1.1 Model v systémovej biológii

Spomínané modely, rovnako ako všetky iné modely, ktoré si s týmto pojmom môžme spojiť (model auta, mesta, slnečnej sústavy), sú abstrakciou reality. Zameriavajú sa na jeden konkrétny aspekt študovaného objektu a často zanedbávajú pohľad na jeho ostatné vlastnosti. V inžinierstve a strojárstve sa často pracuje s matematickým modelom, ktorý sa používa na analýzu a dizajn rôznych technických objektov. Každý takýto objekt ma presne určenú svoju funkciu a preto je možné ľahko vykonávať rôzne výkonnostné testy, a posúdiť tak efektivitu a robustnosť týchto funkcií objektu. Napriek tomu, že biologické systémy nie sú žiadne technické objekty a sú prirodzenou súčasťou prírody, existujú v nej kvôli tomu, že v nej zohrávajú určitú funkciu. Tieto biologické funkcie živých organizmov, vieme podobne ako v strojárstve, za pomoci rôznych postupov zapísať a študovať podobne. Tato analógia s technickým priemyslom ma však svoje hranice. Napríklad prirodzený výber v prírode zatiaľ nevieme simulovať a je zatiaľ čisto záležitosťou prírody. Každopádne sú prípady, pri ktorých si môžeme byť istí funkciou biologických systémov. Zápis týchto modelov je sprostredkovaný pomocou grafov, matematických, či chemických rovníc.

Typy modelov v systémovej biológii:

- statické modely
  - statická analýza biologických sieti pomocou grafu a matíc
- dynamické modely
  - teória dynamických systémov
  - simulácia vývoja modelu v čase

V koncepte tejto práce bude pod pojmom model, vždy myslený dynamický model.

Simulácia modelu
Simulácia modelu je spustenie modelu s počiatočnými parametrami, v určitom prostredí. Simulácia sa snaží abstraktne napodobňovať reálne

správanie sa modelu a s pomocou určitých parametrov a zmien v nastavení modelu následne predikovať určité hypotézy.

Dynamický model mechanicky popisuje sledovaný objekt. Dynamický model umožňuje simulácie a predikcie vývoju v čase a je zložený z dvoch základných komponent:

- množina stavov - v zmysle stavových vektorov, stav vektoru určuje stav systému v danom momente
- systém dynamických rovníc - ktoré sú chápané ako pravidla, či funkcie modelu, ktoré určujú zmenu stavov v čase

Rozlišujeme dva základne modely dynamiky:

- deterministický model
  - makropohľad (populačný)
  - za určitých podmienok generuje jedno správanie
- stochastický model
  - mikropohľad (individuálny)
  - za určitých podmienok môže generovať viacero rôznych správaní

1.1.2 Modelovanie biologických procesov

Modelovanie biologického systému znamená samotnú tvorbu a validáciu modelu. Existuje viacero modelovacích princípov a najznámejším a pre koncept tejto práce dôležitým je modelovanie založené na pravidlách, alebo všeobecne známe viac pod anglickým termínom *ruled-based modelling* je modelovací prístup, v ktorom jednotlivé molekuly a rôzne iné biologické objekty sú reprezentované ako entita a interakcie medzi týmito entitami sú definované ako pravidlá, ktoré transformujú parametre týchto entít. Tento prístup vznikol ako riešenie komplexnosti a veľkosti biologických systémov, definovaných chemickými alebo matematickými reakciami v inom modelovacom princípe - modelovaní založenom na reakciách. Keďže reakcií je v systéme obrovské množstvo, nahradili sa pravidlami, ktoré namiesto jednotlivých biologických konkrétnych objektov pracujú s entitami - typmi. Každý

model obsahuje množinu pravidiel a množinu entít so vstupnými podmienkami, ktoré sa snažia simulovať reálne biologické podmienky náhodným rozložením. Pravidla definujú správanie nie len pre jednu entitu, ale pre skupinu entít. To predstavuje hlavný rozdiel medzi reakciou a pravidlom. Reakcia pracuje s konkrétnym biologickým objektom, pričom pravidlo môže pracovať so skupinou objektov, a tak pravidlo môžme chápať tiež ako abstrakciu reakcie. Týmto prístupom sa interakcie medzi jednotlivými objektami v biologickom systéme dajú definovať oveľa menším počtom pravidiel ako pomocou reakcií.

1.2 E-cyanobacterium.org

Po krátkom úvode do systémovej biológie a modelovaniu biologických systémov sa dostávame k portálu *e-cyanobacterium.org*, ktorý predstavuje webovú platformu na verejné zdieľanie, anotáciu, analýzu a vizualizáciu dynamických modelov a experimentov súvisiacich so *sinicami*. Platforma vznikla a je spravovaná na *Fakulte informatiky Masarykovej univerzity* tímom laboratória SYBILA. Modely na platforme sú reprezentované rôznym tipom abstrakcií, akými sú napríklad biochemické reakčné siete alebo diferenciálne rovnice. Platforma je jedinečná v poskytovaní stručného mapovania matematických modelov do formálneho biochemického popisu. Hlavným cieľom platformy je spojiť svet biologických poznatkov s výhodami matematického popisu dynamických procesov. Samotná platforma je tvorená viacerými modulmi a jedným z hlavných modulov je biochemický priestor (BCS), s ktorým sú všetky ostatné moduly prepojené.

Biochemický priestor predstavuje základ platformy a poskytuje formálny biochemický popis a anotáciu biologického systému. Je založený na hierarchii vybraných biologických procesov a objektov. Veľmi zjednodušene BCS slúži na popis biológie samotnej. Jeho cieľom je poskytnúť vedecký základ pre matematické modely. Na dosiahnutie tohto cieľa BCS využíva databázu biochemických entít. To umožňuje popísať konkrétne entity so všetkými ich vlastnosťami a atribútmi. Každá entita by mala byť správne anotovaná, doplnená podrobnými informáciami a odkazmi na príslušné interné a externé zdroje a databázy. Entity však na samotný popis modelov nestačia, pretože modely

vyžadujú aj určitú hierarchiu procesov popisujúcu interakcie medzi entitami. Preto spolu s priestorovou hierarchiou objektov je vytvorená aj hierarchia procesov prostredníctvom interaktívnych schém. Táto hierarchia nám umožňuje priblížiť sa k jednotlivým pravidlám, ktoré predstavujú atomické operácie, ktoré prebiehajú na entitách. BCS predstavuje ľudsky čitateľný formát, ktorý možno ľahko editovať vo vyhradenom editore a vizualizovať v rámci platformy. BCS ako už bolo spomenuté je definované dvoma časťami - množinou entít a množinou pravidiel. Na zlepšenie použiteľnosti a validácie BCS je definovaný *Biochemical Space Language* (BCSL), ktorý formálne popisuje jednotlivé entity, pravidlá a model samotný. Tento jazyk bude popísaný v ďalšej časti práce. V BCS sa kladie veľký dôraz na správnu anotáciu jednotlivých entít. Každá entita pozostáva z nasledujúcich atribútov:

- **ID entity:** *HCO3*
- **stavy:** *{-, +}*
- **lokácia:** *cyt, liq*
- **názov:** *hydrogén-uhličitan*
- **klasifikácia:** *malá molekula*
- **popis:** *Zohráva dôležitú rolu v mechanizme koncentrácie uhlíka (CCM)*
- **odkazy:** *CHEBI::17544*
- **organizmus:** *Synechococcus elongatus PCC 7942*

Entita v BCS môže zastupovať rôzne biochemické objekty. Od najmenších atomických objektov (*HCO3*), ktoré sa môžu vyskytovať v rôznych stavoch, cez väčšie štruktúrne objekty (*foto-systém*) zložené z atomických objektov, až po komplexy (*proteín*) tvorené ako atomickými či štruktúrnymi objektami tak aj ďalšími vnorenými komplexmi. Špeciálnou entitou sú takzvané kompartmenty (*cytoplazma*), ktoré ktoré reprezentujú priestor v ktorých sa zvyšné entity nachádzajú.

Pravidlá sú definované špeciálnymi rovnicami obsahujúcimi dodatočné informácie. Strany rovnice, na ktorých sa nachádzajú jednotlivé

entity rovnako ako pri reakciách, identifikujú reaktanty a produkty oddelené znamienkom +. Taktiež rozlišujeme typ pravidla, ktoré môže byť buď nevratné (*ireverzibilné*) alebo vratné (*reverzibilné*). Každá jedna entita v pravidle ma priradený svoj kompartment, čo hraje dôležitú úlohu hlavne pri pravidlách, kde sa vyskytuje viacero rôznych kompartmentov. Stechiometria je zahrnutá taktiež, pred každou entitou sa môže nachádzať koeficient udávajúci množstvo. Niektoré pravidlá môžu obsahovať modifikátor - napríklad enzymatické reakcie. Keďže BCS teoreticky dovoľuje nekonečné zanorovanie sa, dovoľuje vzniku pomerne zložitých a veľkých modelov.
Ďalšími modulmi portálu *e-cyanobacterium.org* sú *repozitár modelov*, *repozitár experimentov* a *cyano čísla*

Repozitár modelov je kolekcia matematických modelov, popisujúcich konkrétne častí biologických procesov. Každý model je tvorený súborom matematických rovníc, generovaných z modelu reakčnej siete. Každý objekt modelu je prepojený a mapovaný do BCS. To znamená, že každý objekt modelu je mapovaný na konkrétnu entitu v BCS, a každá reakcia je mapovaná na konkrétne pravidlo z BCS. Model tým pádom obsahuje kompletné informácia a biologické notácie pre všetky reaktanty a reakcie. Model tiež obsahuje vstupné parametre, ktoré umožňujú simuláciu modelu. Model je možné exportovať do *Systems Biology Markup Language* (SBML), tento formát bude v ďalšej časti práce dôkladnejšie popísaný.

Repozitár experimentov je modul na ukladanie a prezentáciu takzvaných *time-series* dát z laboratórnych experimentov. Každý experiment je podložený detailným popisom a dôkladnou anotáciou. Jednotlivé premenné a zložky experimentu môžu byť prepojené s BCS. Výsledky experimentov je možné vizualizovať pomocou grafov.

Cyano čísla je súbor čísel súvisiacich so *sinicami*. Sú to rôzne hodnoty a konštanty, ako počet atómov železa na bunku alebo izoelektrický bod plastocyaninu.

1.2.1 Rozsiahla modelová platforma

Rozsiahla modelová platforma alebo anglicky *Comprehensive Modelling Platform* (CMP) je obecný názov pre platformu akou je *e-cyanobacterium.org*. CMP je projekt, ktorého cieľom je umožniť komunite vedcov zaoberajúcou sa systémovou biológiou využívať všetky moduly a služby, ktoré platforma poskytuje, pre ľubovoľný organizmus, nie len pre *sinicu*. Používatelia si budú môcť nahrať ľubovoľné modely a experimenty, vykonávať simulácie a analýzy, následne ich vizualizovať a exportovať v rôznych formátoch. Platforma je vyvíjaná laboratórium SYBILA a cieľom tejto práce bolo jej ďalšie rozširovanie a obohacovanie v zmysle vizualizácie reakcií v *Repozitáry modelov* a pravidiel v *BCS*.

1.3 Formáty modelov v systémovej biológii

V tejto časti práce opíšeme tri formáty systémovej biológie, ktorými môžu byť modely definované. Nejde o jediné formáty, ktoré existujú, ale ide o formáty relevantné pre koncept tejto práce. Ide o už spomínaný SBML, *System Biology Graphical Notation* (SBGN) a BCSL.

1.3.1 SBML

SBML definuje modely pozostávajúce z entít (biochemických objektov) viazaných a modifikovaných procesmi (biochemickými reakciami). Tu je názorný vymyslený príklad malého súboru biochemických reakcií:

$$S_1 \xrightarrow{k_1[S_1]/([S_1]+k_2)} S_2 \tag{1.1}$$

$$S_2 \xrightarrow{k_3[S_2]} S_3 + S_4 \tag{1.2}$$

Symboly v hranatých zátvorkách (napr. S1) predstavujú koncentrácie molekulárnych objektov, šípky predstavujú reakcie a vzorce nad šípkami predstavujú rýchlosti, s ktorými sa reakcie uskutočňujú. (Aj keď v tomto príklade sú použité koncentrácie, mohol by byť použitý aj iný parameter, napríklad počet molekúl každého chemického objektu.)

SBML model je tvorený väčším počtom rozdielnych komponent: reaktanty, produkty, reakcie, reakčné rýchlosti, ... Aby bolo model možné analyzovať, či simulovať, musia byť explicitne uvedené ďalšie komponenty vrátane kompartmentov, v ktorých sa nachádzajú entity a jednotky v rôznych množstvách. V SBML sú dva dôležité princípy, ktoré spočívajú v tom, že za prvé - modely sú rozložené do explicitne označených prvkov a za druhé - reprezentácia modelu úmyselne neimplementuje model priamo do množiny diferenciálnych rovníc alebo inej špecifickej interpretácie modelu. To uľahčuje rôznym nástrojom pracujúcim s SBML interpretovať a konvertovať model z SBML formátu do akéhokoľvek iného formátu, s ktorým nástroj pracuje.

V SBML sa môžu nachádzať nasledujúce komponenty:

- **definícia funkcie** - pomenovanie matematickej funkcie, ktorá môže byť modelom používaná
- **definícia jednotky** - definovanie novej jednotky merania, alebo zmena definície základnej jednotky SBML
- **typ kompartmentu** - typ prostredia kde sa entity reakcie, ako chemické substancie môžu nachádzať
- **typ entity** - charakterizuje typ entity, ako napríklad ión vápnika, molekulu glukózy, či ATP, ktorá prináleží určitej reakcii
- **kompartment** - množina kompartmentov rôznych typov a konečnej veľkosti, ktoré definujú, kde sa jednotlivé SBML entity modelu môžu vyskytovať. V modely sa môže nachádzať viacero kompartmentov rovnakého typu. Každá entita modelu, musí byť lokalizovaná aspoň v jednom z týchto kompartmentov.
- **entita** - množina entít rovnakého typu, ktoré ktoré sú lokalizované v určitom kompartmente
- **parameter** - vyjadruje určité množstvo, pod vymysleným, symbolickým názvom. Parameter je použitý vo všeobecnom zmysle, aby nerozlišoval, či ide o konštantu, alebo o premennú reakcie. Parameter definovaný na najvyššej vrstve je chápaný ako globálna premenná, ale parameter je taktiež možné si zadefinovať aj pre jednu konkrétnu reakciu, teda ako lokálnu premennú.

- **iniciálne nastavenia** - matematický výraz určujúci počiatočné podmienky modelu
- **pravidlo** - matematický výraz priradený k súboru rovníc, vytvorených na základe reakcií definovaných v modely. Pravidlá definujú ako počítať hodnotu určitej premennej z iných premenných a taktiež je nimi možné definovať mieru zmeny premennej. Spolu s rovnicami reakčnej rýchlosti sa s pravidlami dá určiť správanie sa modelu v čase. Súbor pravidiel obmedzuje model počas celého trvania simulácie.
- **obmedzenie** - zakazujú neplatné stavy a podmienky modelu počas celej dynamickej simulácie. Sú definované matematickými výrazmi, ktoré vypočítajú pravé alebo falošné hodnoty z modelových premenných, parametrov a konštánt.
- **reakcia** - výraz popisujúci určitý proces transformácie, prenosu alebo väzby, ktorý môže meniť množstvo jedného alebo viacerých entít. Reakcia popisuje ako sa určité entity (reaktanty) transformujú na iné entity (produkty). Reakcie majú priradené výrazy, (ľubovoľné matematické funkcie) dominujúce kinetickú rýchlosť, ktorá určuje, ako rýchlo sa uskutočňujú.
- **udalosť** - popisuje okamžitú nespojitú zmenu množiny premenných akéhokoľvek typu (množstvo, veľkosť kompartmentu, hodnotu parametra, ...), za predpokladu, že je splnená určitá spúšťacia podmienka tejto udalosti

Okrem vyššie spomenutých komponent, je ďalšia dôležitá vlastnosť SBML, že každá entita môže obsahovať strojovo čitateľné anotácie, ktoré vyjadrujú vzťahy medzi entitami z modelu a entitami z externých zdrojov, akými sú napríklad biochemické databázy. S dôkladnými anotáciami model nie je len obyčajným matematickým zápisom, ale stáva sa sémanticky obohateným nástrojom na komunikáciu a šírenie vedeckých poznatkov.

1.3.2 SBGN

SBGN je grafický štandard slúžiaci na efektívne ukladanie, výmenu a hlavne vizualizáciu dát o signalizačných dráhach, metabolických

sieťach, či génových regulačných sieťach medzi komunitou biochemikov, biológov a iných teoretikov. Tento štandard niekoľko rokov vytváralo spoločenstvo biochemikov, modelárov a počítačových vedcov. SBGN dáta sú ukladané v súboroch v rozšíriteľnom značkovacom jazyku (XML), kde obsahujú informácie o jednotlivých komponentách a ich priestorovom rozložení, každá komponenta ma svoje súradnice, výšku a šírku. SBGN pozostáva z troch ortogonálnych jazykov, ktoré zachytávajú rôzne pohľady na biologické systémy:

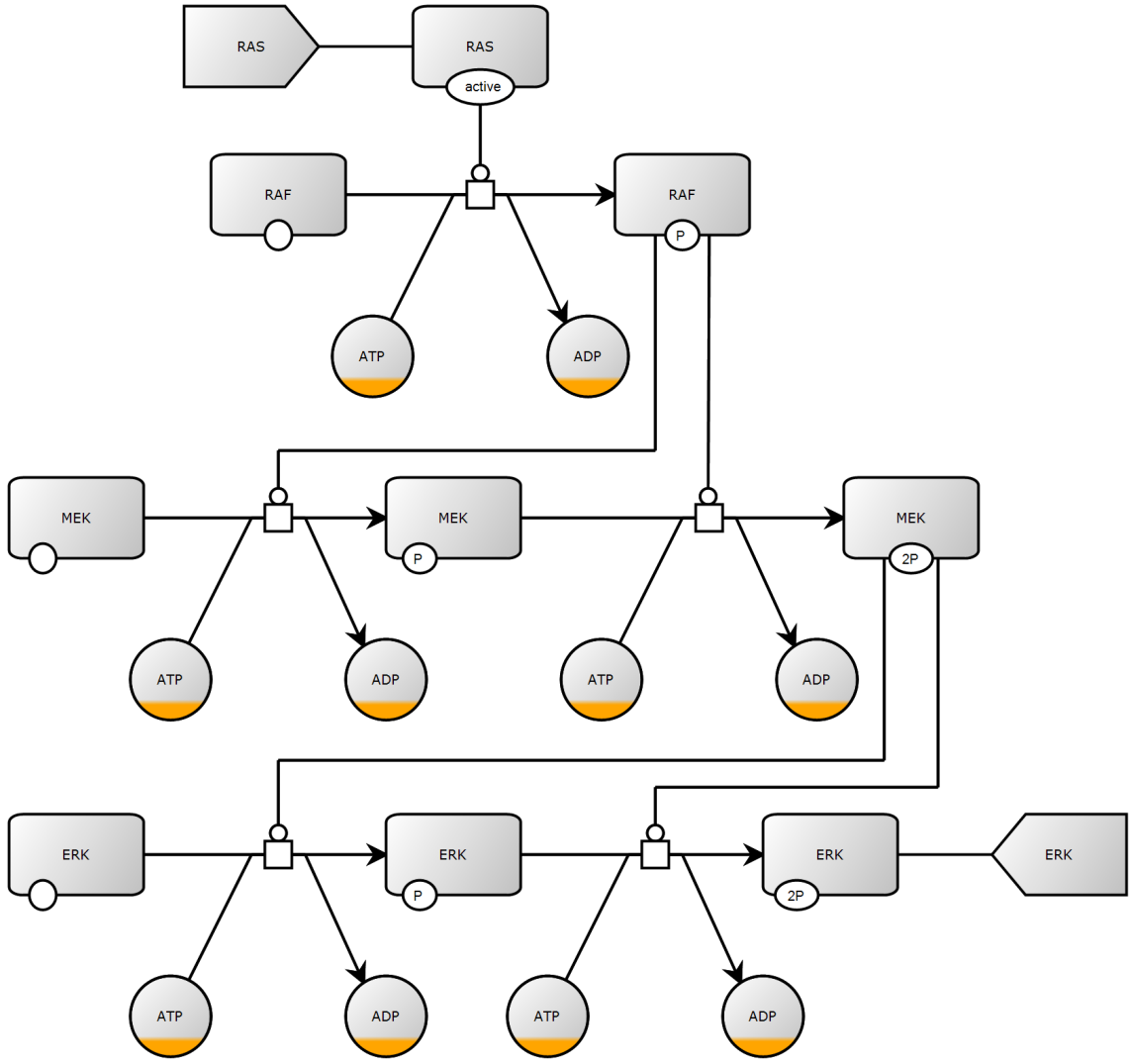
- ***Process description language*** **(PD)** je jazyk, ktorý zobrazuje časové priebehy biochemických interakcií v sieti. Môže byť použitý na zobrazenie všetkých molekulárnych interakcií, ku ktorým dochádza v sieti medzi biochemickými objektami, pričom tá istá entita sa objavuje viackrát v tom istom diagrame.
- ***Entity relationship language*** **(ER)** tento jazyk zobrazuje všetky vzťahy, v ktorých sa entita môže vyskytovať, bez ohľadu na čas. Vzťah môžeme chápať ako pravidlo, popisujúce vplyv entít na iné vzťahy.
- ***Activity flow language*** **(AF)** jazyk slúži na zobrazovanie toku informácií medzi entitami v sieti. Neobsahuje informácie o zmenách stavov jednotlivých entít a je vhodný na zobrazenie odchýliek (perturbácií), či už genetických alebo environmentálnych v prírode.

Každý z jazykov je definovaný komplexnou sadov symbolov s presnou sémantikou a podrobnými syntaktickými pravidlami, týkajúcimi sa konštrukcie a interpretácie grafických štruktúr. Pomocou týchto troch jazykov, sú vedci schopní presne graficky reprezentovať signálne dráhy, metabolické siete a iné biochemické siete. V koncepte tejto práce je dôležitý hlavne prvý z menovaných jazykov - PD, ktorý spomedzi všetkých troch jazykov zároveň patrí medzi najviac používaný.

1.3.3 SBGN - PD

Cieľom PD je zobraziť ako odlišné entity v systéme prechádzajú z jednej formy do druhej. Autorom jazyka bolo od začiatku vývoja jasné,

že je nemožné vytvoriť kompletnú a presnú formu notácie hneď na prvýkrát. Preto sa autori rozhodli rozdeliť vývoj jazyka do viacerých úrovní. Pod konkrétnou úrovňou jedného z jazykov SBGN, sa rozumie sada funkcií, ktoré spolu dobré fungujú a vytvárajú funkcionalitu, ktorú komunita používateľov zvolila za vhodnú na riešenie určitej sady problémov a úloh. Okrem úrovní sa jazyk ďalej delí aj na verzie, v ktorých sú definované menšie vývojové zmeny jazyka (ujasnenie sémantiky, úpravy v piktogramoch), avšak žiadne veľké zmeny, ktoré by ovplyvňovali to, akým spôsobom sú SBGN diagramy generované. Navyše, nové verzie by mali byť spätne kompatibilné s predchádzajúcimi verziami, čo znamená, že nové verzie sú vytvárané v súlade so staršími verziami prináležiacimi do rovnakej úrovne jazyka, aby boli stále validné, čo neplatí pre úrovne jazyka.



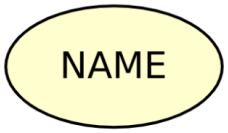
Obr. 1.1: Príklad SBGN diagramu

SBGN pozostáva z troch základných štruktúr, ktorými sú uzly alebo vrcholy entít (*entity pool nodes*), uzly procesov (*process nodes*) a hrany (*arcs*). Aby sme si vedeli predstaviť čo PD popisuje a ako vyzerá jeho grafická podoba, použijeme príklad 1.1, ktorý zobrazuje jednoduchý diagram, popisujúci časť MAPK (*kaskáda mitogénom aktivovanéj*

proteinkinázy). Väčšie štvoruholníky a kruhy predstavujú biologické objekty, akými sú makromolekuly alebo jednoduché chemikálie, ktoré spoločne nazývame uzly entít. Tieto entity sa menia na iné pomocou biochemických procesov, v príklade znázornených ako menšie štvorce, nazývané uzly procesov. Uzly procesov a entít sú v príklade navzájom prepojené pomocou šípok (orientovaných hrán). V tomto konkrétnom príklade máme znázornený iba jeden typ procesu a tým je proces katalýzy biochemických objektov. Šípky naznačujú smer procesu, napríklad nefosforilovaná RAF kináza sa mení na fosforilovanú RAF kinázu v procese katalýzy pomocou RAS katalizátora, čoho vedľajším účinkom je vznik ADP z ATP. V príklade sa vyskytujú aj menšie kruhové objekty, ktoré reprezentujú stavy entít. Teraz si popíšeme niektoré objekty PD. Bolo by zbytočné popisovať úplne celú štruktúru so všetkými objektami SBGN PD jazyka a tak budú popísane len relevantné objekty dôležité k pochopeniu jazyka a cieľa tejto práce.

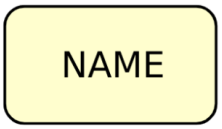
Uzly entít:

- **Nešpecifikovaná entita** - označuje nešpecifikovanú entitu, ktorej typ je neznámy alebo irelevantný pre účely použitia v SBGN diagrame. Znázorňuje sa elipsou (Obr. 1.2).



Obr. 1.2: Príklad nešpecifikovanej entity

- **Makromolekula** - je zahrnutá v mnohých biologických procesoch. Reprezentuje biochemickú substanciu, zloženú z kovalentných väzieb pseudo-identických jednotiek. Príkladom makromolekúl sú proteíny, nukleové kyseliny, či polysacharidy. Makromolekula je znázornená štvoruholníkom so zaoblenými rohmi (Obr. 1.3).



Obr. 1.3: Príklad makromolekuly

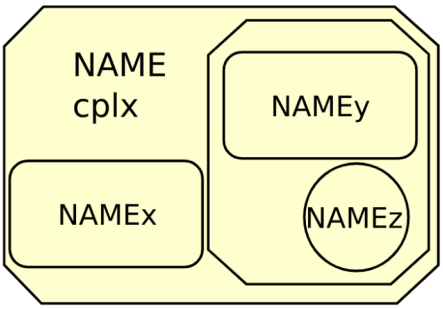
- **Jednoduchá chemikália** - v PD je jednoduchá chemikália opakom makromolekuly, to znamená, že nie je tvorená kovaletnými väzbami. Ide napríklad o atóm, ión, radikál, či soľ. Znázorňuje sa jednoduchým kruhom. (Obr. 1.4).

Obr. 1.4: Príklad jednoduchej chemikálie

- **Zdroj a výlevka** - je užitočné mať možnosť reprezentovať vznik určitej entity z nešpecifikovaného zdroja. Napríklad ak v modely vzniká nejaký proteín, nechceme popisovať každú aminokyselinu, cukor a ostatné zložky proteínu. Rovnako užitočné je mať niečo čím môžme reprezentovať deštrukciu alebo rozpad určitej biochemickej entity, ktorá obrazne myslené, "zmizne do výlevky". Pre tieto účely, je v PD definovaný špeciálny piktogram, ktorý reprezentuje neznámy zdroj alebo výlevku. Znázorňuje sa matematickým symbolom prázdnej množiny, čo je kruh, prečiarknutý čiarou smerujúcou z pravého horného rohu, do ľavého dolného rohu (Obr. 1.5)

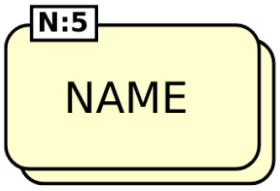
Obr. 1.5: Príklad zdroju/výlevky

- **Komplex** - reprezentuje biochemickú entitu zloženú z jednoduchších entít, napríklad makromolekúl, jednoduchých chemikálií, multimérov, alebo aj iných komplexov. Výsledkom je nezávislá entita, ktorá ma svoju vlastnú identitu, vlastnosti a funkciu. Komplex je zobrazený ako štvoruholníkový kontajner v ktorom sa môžu nachádzať všetky ostatné entity (vrátane ďalšieho komplexu) s orezanými rohmi (Obr. 1.6).



Obr. 1.6: Príklad komplexu

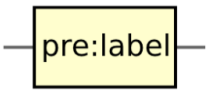
- **Multimér** - je agregácia viacerých identických alebo pseudo-identických entít združených pokope nekonvalentými väzbami. Multimér sa od polyméru líši práve v tom, že neobsahuje kovalentné väzby, ktoré majú byť reprezentované makromolekulami. Pod pojmom pseudo-kovalentné je myslené, že entity sa líšia chemicky, ale zdieľajú určitú spoločnú globálnu charakteristiku, ako štruktúru alebo funkciu. Multimér je zobrazený dvoma prikrytými štvoruholníkmi, s informáciou o množstve entít zobrazenom v menšom štvoruholníku (Obr. 1.7).



Obr. 1.7: Príklad multiméru

Doplňujúce štruktúry uzlov entít:

- **Dodatočná informácia** - ako z názvu vyplýva, táto štruktúra predstavuje dodatočné informácie o funkcii danej entity, nesúvisiacej s jeho rolou v SBGN diagrame. Môže napríklad obsahovať informáciu, charakterizujúcu logickú časť entity v oblasti funkcionality, alebo informáciu o prostredí v ktorom sa entita vyskytuje. Táto informácia je zobrazená štvoruholníkom na hranici uzlu entity, ktorú anotuje (Obr. 1.8).



Obr. 1.8: Príklad informácie

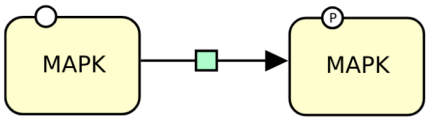
- **Stav** - veľké množstvo biochemických entít sa môže v organizme vyskytovať v rôznych stavoch. Tieto stavy sa môžu meniť za rôznych špecifických podmienok. Stav sa môže meniť napríklad po syntéze, keď zvyšky makromolekúl (aminokyseliny, glycidy, ...) sú kovalentnými väzbami naviazané na iné chemické objekty. Každá entita môže obsahovať jeden alebo viac stavov. Stav sa znázorňuje elipsou nachádzajúcou sa na hranici uzlu entity, ktorého stav popisuje (Obr 1.9).

Obr. 1.9: Príklad stavu

Uzly procesov:

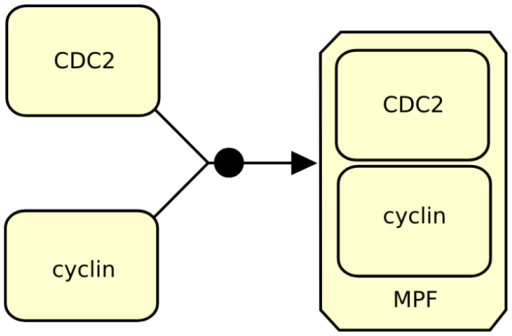
- **Proces** - je základný uzol procesov. Popisuje proces transformácie súboru biochemických entít na súbor iných biochemických entít. Transformácia môže znamenať konverziu (modifikáciu kovalentými väzbami), konformácia (zmenu relatívnej polohy v zložke), translokáciu (presun z jedného kompartmentu do iného). Proces je znázornený štvorčekom, na ktorý sú napojené

dva konektory (krátke hrany) po jednej na protiľahlých stranách, na ktoré sa napájajú hrany konzumácie a produkcie, ktoré budu popísane nižšie. V príklade 1.10 je znázornený proces fosforylácia proteínu MAP kinázy.



Obr. 1.10: Príklad procesu

- **Zjednotenie** - definuje vznik zložitejších komplexov z niekoľko jednoduchších štruktúr nekonvalentými väzbami. Znázorňuje sa čiernym kruhom s dvoma konektormi, na ktoré sa viažu konzumné a produkčné hrany (Obr. 1.11 *zjednotenie cyklínu a CDC2 kinázy do MPF komplexu*). Zjednotenie nie je nikdy reverzibilné - opačný proces je štiepením, definovaným nižšie.



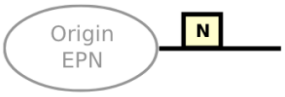
Obr. 1.11: Príklad zjednotenia

- **Štiepenie** - popisuje rozpadnutie nekovalentých väzieb a rozklad zložitejších komplexov na jendoduchšie - opačný proces ku zjednoteniu. Znázorňuje sa podobne ako zjednotenie, krúžok ale nie je vyplnený, naopak obsahuje navyše jeden zamorený kruh (Obr. 1.12).

Obr. 1.12: Príklad rozštiepenia

Hrany:

- **Konzumné** - hrany sa používajú na znázornenie toho, že uzly entít sú konzumované uzlom procesu, ale nie sú ním produkované. Označuje smer reakcie, viaže reaktanty k uzlu procesu. Je znázornený jednoduchou rovnou čiarou bez znakov (Obr. 1.13). Môže obsahovať štítok naznačujúci stechiometriu. Každá hrana spája práve jednu entitu s jedným uzlom procesu.

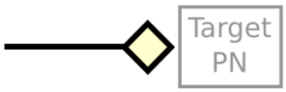


Obr. 1.13: Príklad konzumnej hrany

- **Produkčné** - hrany vychádzajú z procesného uzlu a indikujú vznik produktov. Rovnako ako pri konzumných hranách, vždy práve jedna hrana vychádza z uzlu procesu, do jednej entity. Znázorňujú sa podobne ako hrany konzumné, ale navyše obsahujú hrot šípky indikujúci smer procesu (Obr. 1.14).

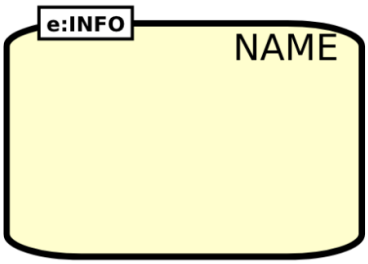
Obr. 1.14: Príklad produkčnej hrany

- **Modulačné** - hrany znázorňujú proces modulácie cieľového procesu inou entitou, či už pozitívne, negatívne ale oboje, v závislosti na podmienkach. Znázorňuje sa rovnou čiarou s diamantom na konci, smerujúcemu k uzlu procesu, ktorý modifikuje (Obr. 1.15).



Obr. 1.15: Príklad modulačnej hrany

Kompartment - reprezentuje logickú alebo fyzickú štruktúru obsahujúcu uzly entít. Jedna entita môže vždy patriť iba do jedného kompartmentu. Preto dve "rovnakéëntity patriace do 2 rôznych kompartmentov sú v skutočnosti dve odlišné entity a mali by byť reprezentované odlišnými entitami. Kompartment je zobrazený ako uzavretý, spojitý priestor, rôzneho tvaru, ktorého hraný by mali byť viditeľne tenšie ako hrany ostatných štruktúr. (Obr. 1.16).



Obr. 1.16: Príklad kompartmentu

Popísali sme si väčšinu štruktúr, ktoré vytvárajú PD jazyk, a ktoré využijeme v ďalšej časti práce. Na to aby sme sa dostali k jadru problému, si ešte musíme predstaviť jeden formát, a tým je BCSL.

1.3.4 BCSL

Tento jazyk, ako sme si už spomínali, vytvoril M. Troják v spolupráci s laboratóriom SYBILA. Je dôležité si uvedomiť rozdiel medzi BCS a BCSL. Hlavnou úlohou BCS je vytvoriť vedomostný základ pre matematické modely. Je tvorený databázou biochemických entít, ktoré sú všetky dôkladne anotované popisom a linkami na externe zdroje a databázy. To umožňuje BCS vytvoriť priestorovú hierarchiu jednotlivých objektov s ich atribútmi a vlastnosťami. Samotné entity a priestorová hierarchia však na popis modelov nestačia, a je nutné zadefinovať

pomocou interaktívnych schém aj hierarchiu procesov, a opísať tak jednotlivé atomické operácie prebiehajúce na entitách. Na zlepšenie použiteľnosti a presnosti BCS vznikol BCSL, ktorý formálne popisuje entity a pravidlá. Sémantika jazyka je založená na princípoch modelovania pomocou pravidiel, podobne ako SBML a jedným z dôvodov vzniku, bolo vytvoriť taktiež jazyk, ktorý by mal byť ľahko čitateľný pre aj pre človeka, nie len pre rôzne nástroje, čo sa pre účely CMP perfektne hodilo. BCSL teda formálne definuje celý BCS. To zabezpečuje definovať všetky entity a pravidlá so všetkými ich vlastnosťami a pomáha prepájať objekty z modelov s objektami BCS. Akonáhle je BCS vytvorený, môžme použiť referenciu na SBML model - spojiť všetky objekty a reakcie modelu s entitami a pravidlami BCS, čo vďaka anotácii BCS, obohacuje objekty modelu o ďalšie informácie. Model definovaný v BCSL má dve výhody - možnosť aplikovať formálnu analýzu na model a úplne zrekonštruovať biologický význam celého modelu.

Formálny základ jazyka BCSL tvoria pravidlá, v ktorých vystupujú agenti. Definujeme konkrétne tri typy agentov - atomický, štruktúrny a komplexný. Ďalej je ešte definovaná signatúra a kompartment.

Signatúra môže byť atomická alebo štruktúrna. Atomická obsahuje množinu možných stavov, v ktorých sa môže atomický agent nachádzať a štruktúrna obsahuje mená atomických agentov, ktoré sa môžu v štruktúrnom agentovi vyskytovať.

Atomický agent je najjednoduchšia a najmenšia jednotka BCSL, slúžiaca na popis biochemických objektov. Atomický agent sa vždy nachádza v nejakom stave, (v prípade, že stav nie je dôležitý, alebo je neznámy definujeme "prázdny stav", ktorý označujeme symbolom ϵ (epsilon)). Povolené stavy agenta definuje signatúra s rovnakým menom, akým je meno atomického agenta. Atomický agent sa definuje ako dvojica, obsahujúca meno agenta a jeho stav, napríklad:

$$A_1 = (\text{chl}, \text{n})$$

čo v BCSL zapisujeme ako: *chl{n}*.

Štruktúrny agent popisuje biochemické objekty, ktoré sú zložene z viacero menších objektov (atomických agentov), pričom zloženie je iba abstraktné a nemusí byť nutne úplne kompletné. Kvôli tomu má štruktúrny agent unikátne meno a môže byť zložený len z atomických agentov, ktoré sa nachádzajú v rovnakom kompartmente ako štruktúrny agent samotný. Typických príkladom štruktúrnych agentov sú proteíny, pozostávajúce z aminokyselín. Aminokyselín sa môže v proteíne nachádzať stovky, ale nie všetky nás v našom modely zaujímajú. Povedzme, že len dve aminokyseliny sú schopné určitej modifikácie (zmeny stavu), preto je logickejšie modelovať len tieto dve aminokyseliny, namiesto celej štruktúry proteínu. Štruktúrny agent je definovaný ako dvojica pozostávajúca z mena štruktúrneho agenta a množiny atomických agentov, ktoré sa v ňom nachádzajú. Príklad štruktúrneho agenta, obsahujúceho troch atomických agentov:

$$S_1 = (\text{ps2}, \{(\text{chl}, \text{n}), (\text{p680}, +), (\text{pheo}, -)\})$$

čo v BCSL zapisujeme ako: *ps2(chl{n}|p680{+}|pheo{-})*

Komplexný agent - definuje zložitejšie, netriviálne objekty, zložene zo všetkých tipov agentov, vrátane komplexných. V SBML alebo SBGN boli komplexy charakterizované pomocou väzieb, BCSL sa ale od toho snaží abstrahovať a pod komplexom myslí existenciu určitých objektoch v akomsi zhluku, či skupine, ktorý zdieľa určité vlastnosti. Komplex definujeme ako dvojicu obsahujúcu množinu agentov, ktorí komplex vytvárajú a kompartment v ktorom sa nachádza. Príklad komplexu:

$$C_1 = (\ (\ (\text{ps2}, \{(\text{chl}, \text{n}), (\text{p680}, +), (\text{pheo}, -)\}), \text{fa}\{-\}\ ), \text{tlm}\ )$$

čo v BCSL zapisujeme ako: *ps2(chl{n}|p680{+}|pheo{-}) . fa{-}::tlm*
Jednotliví agenti sú oddelený bodkou.

Každý samostatne stojací agent, ktorý je v pravidle oddelený znamienkom plus, (to znamená jednotlivé reaktanty a produkty v reakcii), musí mať priradený svoj kompartment. Všetky zvyšní agenti nachádzajúci sa v hlavnom agentovi, sa musia nachádzať v tom istom kompartmente. Kompartment sa v pravidle vždy nachádza na poslednom mieste a je oddelený od zvyšku agenta dvoma po sebe nasledujúcimi

dvojbodkami (NADPH::cyt).

Pravidlo je tvorené z jednotlivých agentov a kompartmentov. Každé pravidlo má dve strany, ktoré však nie vždy obsahujú nejakých agentov (vznik a zánik určitých objektov). Jednotlivé strany pravidla sú oddelené šípkou, a to buď $\Rightarrow$ čo indikuje, že pravidlo je ireverzibilné, alebo $\Leftrightarrow$ pre reverzibilné pravidlá. Samotná formálna definícia je zložitejšia, avšak nie až tak dôležitá pre koncept tejto práce a názorný príklad pravidla, bude postačujúci pre jeho pochopenie.

$$HCO_3\{-\} :: liq + OH\{-\} :: liq \Leftrightarrow CO_3\{2-\} :: liq + H_2O :: liq$$

V pravidle sa nachádzajú dva reaktanty a dva produkty. Všetky agenti sa nachádzajú v rovnakom kompartmente a $\Leftrightarrow$ naznačuje, že pravidlo je reverzibilné.

Model je v BCSL definovaný ako štvorica, obsahujúca pravidlá, atomickú signatúru, štruktúrnu signatúru a množinu komplexných výrazov, definujúcich stav modelu na začiatku. Správanie sa modelu je definované množinou pravidiel. Atomická signatúra definuje pre každého atomického agenta, ktorý sa v pravidlách nachádza, všetky stavy ktoré môže nadobúdať. Podobne štruktúrna signatúra definuje pre každého štruktúrneho agenta, ktorý sa v pravidlách vyskytuje, atomických agentov z ktorého môže byť zložený.

Syntaktické rozšírenia

- V prípade, že štruktúrny agent obsahuje atomických agentov len s prázdnym stavom ϵ, je možné ich z pravidla vynechať.

- Okrem atomických a štruktúrnych signatúr je možné si zadefinovať aj komplexnú signatúru, ktorá obsahuje unikátne mená (alias), komplexných agentov, ktorým môžme v pravidle konkrétny komplexný agent substituovať a nezapisovať celú jeho vnútornú štruktúru.

- Stechiometria - definuje množstvo jednotlivých reaktantov a produktov v pravidle, nachádza sa pred agentom.

- Operátorom "::", ktorý sa použitý pre priradenie agenta do kompartmentu, je tiež možné naznačiť priradenie agenta do komplexu, čo umožňuje prehľadnejšie zanorovanie sa, hlbšie do biochemických štruktúr (*S{u}::KaiC::KaiC3::cyt*).

- Posledným rozšírením, je možnosť si zadefinovať v pravidle premennú, ktorá v pravidle reprezentuje ľubovoľný objekt z danej množiny, čo umožňuje zredukovať počet podobných pravidiel.

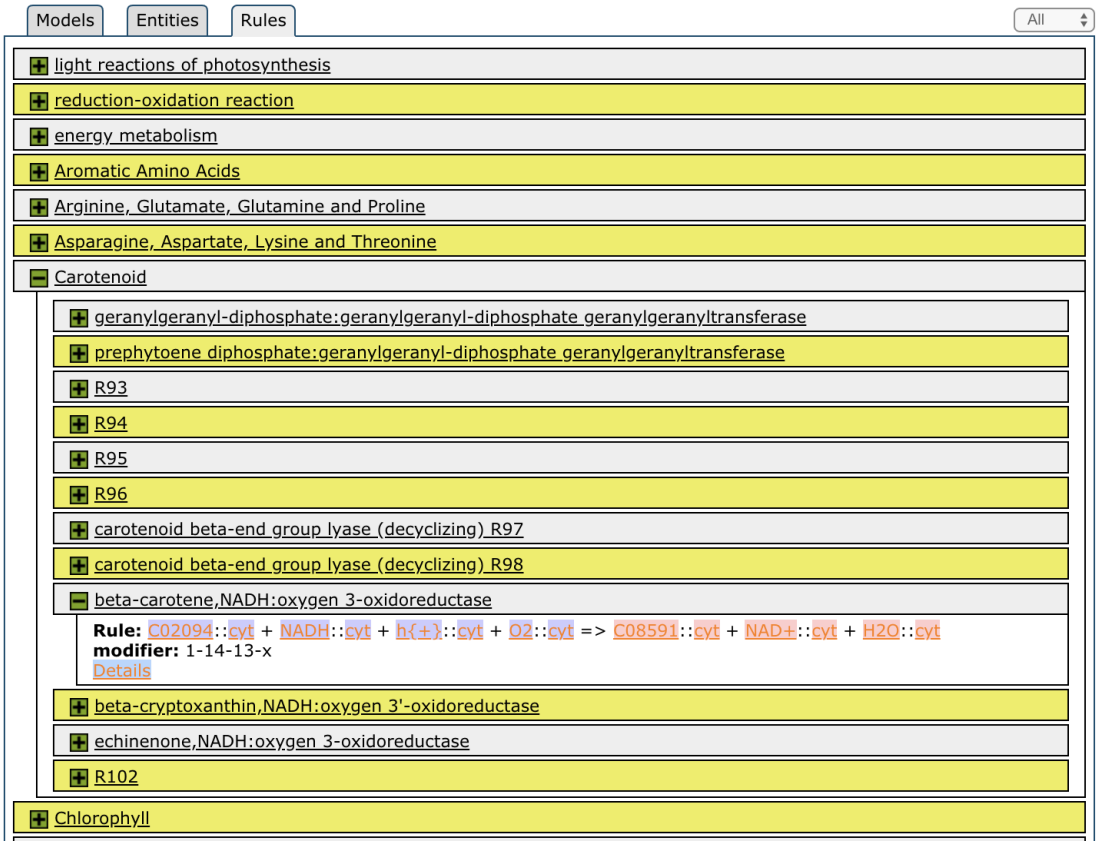
  S{u}::KaiC::?X::cyt $\Rightarrow$ S{p}::KaiC::?X::cyt ; ?X = {KaiC3, KaiBC}

  V tomto príklade je zadefinovaná premenná X, ktorej množina možných hodnôt sa nachádza za pravidlom. X môžme v tomto prípade substitovať za KaiC3 alebo KaiBC. Ak by premenná nebola zadefinovaná, museli by sme toto pravidlo rozpisať na dve samotné pravidla. (V skutočnosti to však reprezentuje dve odlišné pravidla, tento zápis nám len pomáha zredukovať počet pravidiel modelu a tak zjednodušiť celý model).

2 Popis problému

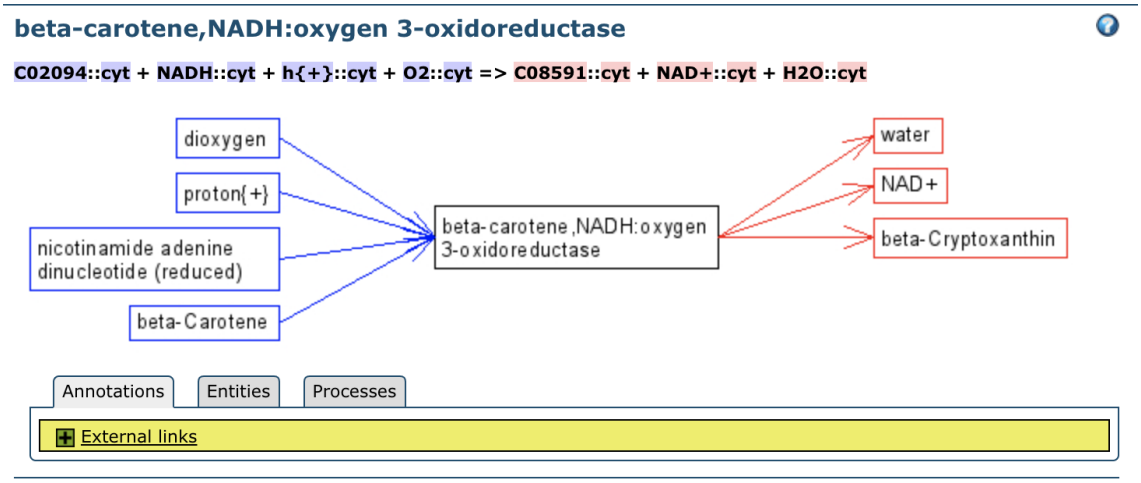
V predchádzajúcej kapitole sme si popísali všetky potrebné pojmy a definície. V tejto kapitole si popíšeme problém, ktorý je objektom tejto práce. Vizualizácia modelov na portály e-cyanobacterium.org je v čase vzniku tejto práce v zastaralom a improvizovanom stave. Cieľom tejto práce bolo vymyslieť a neimplementovať lepší spôsob vizualizácie modelov, so zameraním na pravidla v *BCS* module a reakcie v *Repozitári modelov*, ktorý bude následne použitý pre účely CMP.

V čase vzniku práce je stav portálu nasledovný. Ak si na portály v module BCS, zvolíme sekciu pravidiel konkrétneho modulu, objaví sa nám zoznam všetkých pravidiel, zoradených podľa jednotlivých entít (Obr. 2.1).



Obr. 2.1: Zoznam pravidiel, v module BCS na portály e-cyanobacterium.org

V prípade, že si vyberieme nejaké konkrétne pravidlo, a rozbalíme záložku, zobrazí sa nám samotné pravidlo, modifikátor a tiež možnosť kliknúť na detaily. V prípade, že si zvolíme detaily, presmeruje nás to na detaily pravidla, kde je samotné pravidlo aj vizualizované (Obr 2.2)

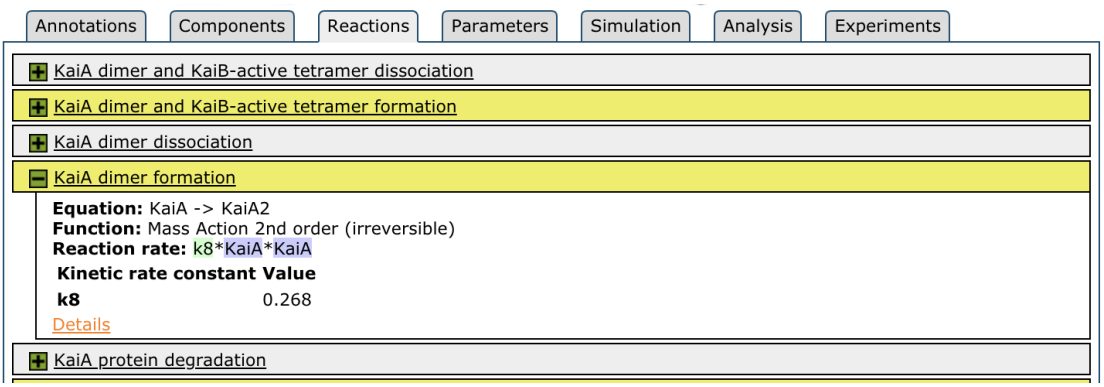


Obr. 2.2: Detaily konkrétneho pravidla, v module BCS na portály e-cyanobacterium.org

Ako môžme vidieť, vizualizácia pravidla je implementovaná len pomocou jednoduchých farebných štvoruholníkov a šipiek. Schéma neznázorňuje štruktúru pravidla, nie je možné identifikovať či ide o atomického, štruktúrneho, či komplexného agenta a taktiež chýba informácia o kompartmente. Úlohou práce je tento spôsob vizualizácie nahradiť a automaticky vizualizovať pomocou vhodných nástrojov v štandarde SBGN.

Pre reakcie, nachádzajúce sa v *Repozitári modelov* (Obr. 2.3), sa možnosť vizualizácie nenachádza vôbec. V prípade, že si zvolíme detaily reakcie, zobrazia sa detaily pravidla z ktorých reakcia vychádza v BCS module, ktorý je opísany vyššie. Reakcie sú reprezentované pomocou štandardu SBML.

V obidvoch prípadoch, ako pre reakcie tak aj pravidlá, je potrebné vytvoriť spôsob automatickej vizualizácie v štandarde SBGN. V prípade reakcií je mapovanie z SBML na SBGN známe, avšak pre pravidlá je potrebné vytvoriť mapovanie z BCSL do SBGN.

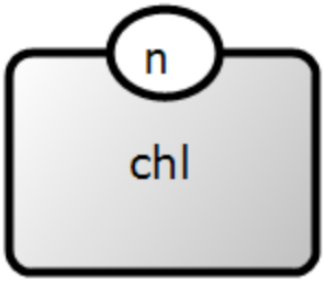


Obr. 2.3: Zoznam reakcií, v Repozitore modelov na portály e-cyanobacterium.org

3 Návrh mapovania BCSL na SBGN

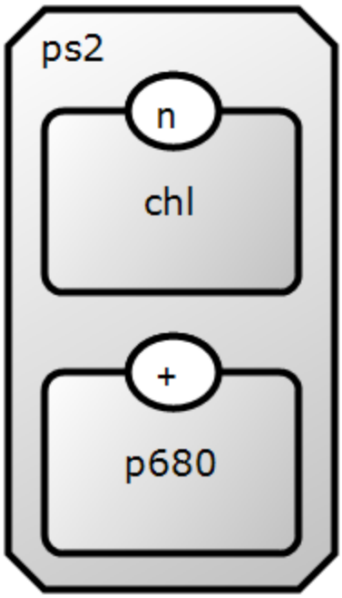
V tejto kapitole si definujeme spôsob mapovania BCSL pravidiel do štandardu SBGN. Prevod z BCSL do SBGN doposiaľ nebol nikde definovaný a nasledujúci návrh mapovania, je jeden z možných spôsobov, akým by bolo možné tuto konverziu uskutočniť.

Začneme od základnej jednotky BCSL, ktorou je atomický agent. Skutočnosť, že atomický agent sa nachádza vždy v určitom stave a že vždy vieme o akú entitu ide, keďže BCS kladie dôraz na dôkladnú anotáciu jednotlivých entít, môžme vylúčiť použitie jednoduchej chemikálie (ktorá v SBGN diagrame nemôže obsahovať stav), a taktiež nešpecifikovanú entitu. Pre atomických agentov sa ako najvhodnejší objekt z SBGN javí makromolekula, s dodatočnou informáciou o stave. Povedzme, že mame atomického agenta A_1 = (chl, n), zapísaného ako *chl{n}*. Jeho grafická podoba v SBGN by vyzerala nasledovne:



Obr. 3.1: Príklad mapovania atomického agenta do SBGN

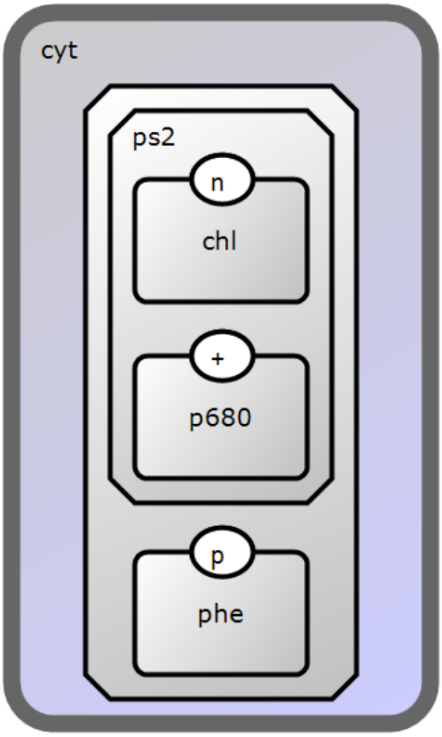
Ďalšou jednotkou BCSL je štruktúrny agent. Vzhľadom k tomu, že štruktúrny agent je tvorený z jednoduchších atomických agentov, čo naznačuje určité zanorenie, ako najvhodnejšia a jediná možnosť sa javí reprezentovať štruktúrneho agenta SBGN komplexom. Komplex bude v ľavom hornom rohu obsahovať informáciu o názve štruktúrneho agenta a vo vnútri SBGN komplexu budu rozmiestnený jednotlivý atomický agenti, reprezentovaný SGBN makromolekulami. Uvažujme štruktúrneho agenta S_1 = (ps2, {(chl, n), (p680, +)}), zapísaného ako *ps2(chl{n}|p680{+})*. Zobrazený v SBGN by vyzeral nasledovne:



Obr. 3.2: Príklad mapovania štruktúrneho agenta do SBGN

Posledným agentom je komplexný agent. Keďže podobne ako štruktúrny agent obsahuje zanorené objekty, (a to aj hlbšie ako do jednej úrovne zanorenia), tiež je najvhodnejšie použiť pre jeho grafické zobrazenie v SBGN komplex. Narozdiel od štruktúrneho agenta, však komplex neobsahuje názov, čo sa skvelé hodí k rozlíšeniu, či ide o štruktúrneho alebo komplexného agenta. Taktiež dobrým rozpoznávacím prvkom je aj to, že štruktúrny agent môže obsahovať len atomických agentov (čiže makromolekuly v SBGN), pričom komplexný agent môže obsahovať všetky ostatné typy agentov. V prípade, že komplex je tiež tvorený len atomickými agentmi, stále je možné rozlíšiť, že ide o komplex absenciou názvu. V prípade, že máme definovanú komplexnú signatúru, čo znamená že určitý komplexný agent môže byť v BCSL pomenovaný, bude SBGN komplex tiež pomenovaný, ale nebude obsahovať žiadne zanorené objekty vo vnútri (informácie o vnútornej štruktúre komplexu sa nachádzajú v komplexnej signatúre). Rozlíšiť komplexný a štruktúrny agent graficky pomocou rozdielnych SBGN objektov v súčastnosti nie je možné, keďže komplex je jediný objekt z SBGN, ktorý môže obsahovať iné objekty. Ak by sa vyššie zmieňované riešenie s absenciou mena komplexu ako rozlišovacie

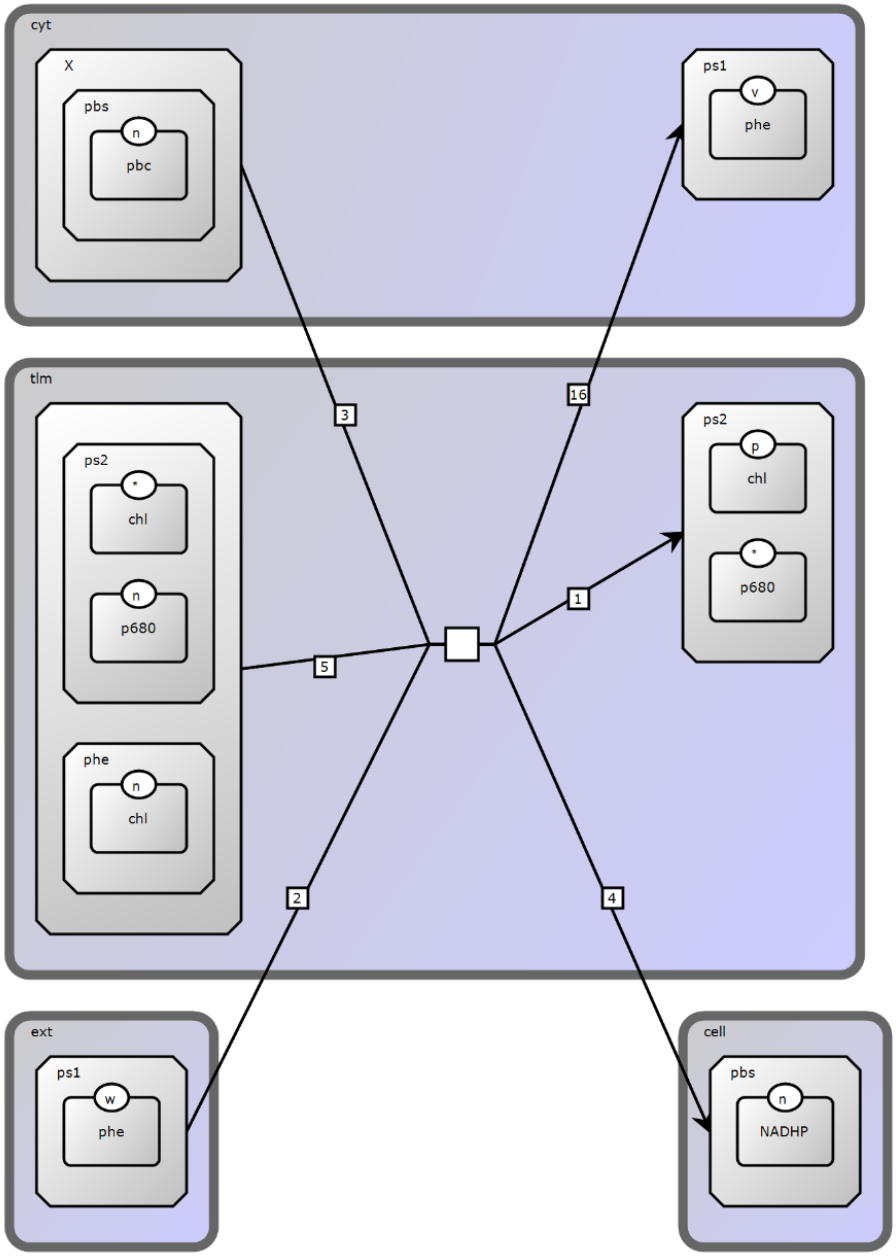
znamenie zdalo nedostatočné, jednou z možností ako striktnejšie definovať, že ide o komplex, je možnosť pridať "Komplex"k názvu. Pre komplexy, ktoré nemajú pomenovanie, použiť "Komplexäko meno, a pri pomenovaných komplexov pridať "Komplex - äko prefix. Pre zobrazenie samotného kompartmentu je v SBGN konkrétny objekt, takže s kompartmentom nie sú žiadne komplikácie. Uvažujme BCSL komplex C_1 = (((ps2, {(chl, n), (p680, +), phe{p}), cyt), zapísaný ako *ps2(chl{n}|p680{+}).phe{p}::cyt*. Grafické zobrazenie aj s kompartmentom v SBGN by mohlo vyzerať nasledovne:



Obr. 3.3: Príklad mapovania komplexného agenta do SBGN

BCSL pravidlá definujú správanie sa modelu a teda zmenu vstupných entít (reaktantov) na iné entity (produkty). Na zobrazenie tejto zmeny je najvhodnejšie využiť uzol procesu z SBGN, a konkrétne proces. Stechiometriu je možné zobraziť pomocou štvoruholníkových dekorátorov, ktoré sa pridajú na hrany, smerujúce z uzlu entít, zastupujúceho jednotlivé entity, do uzlu procesu. V nasledujúcom príklade

(Obr. 3.4) si zobrazíme cele pravidlo (nereálne, slúžiace len na vizuálnu ukážku komplexnosti vytvoreného mapovania).



Obr. 3.4: Príklad zobrazenia kompletného BCSL pravidla do SBGN

V príklade 3.4 je môžme vidieť zobrazené pomerne zložité pravidlo,

$$
\begin{gathered}
3\ \ pbc\{n\} :: pbs :: X :: cyt\ \ + \\
2\ \ phe\{w\} :: ps1 :: ext\ \ + \\
5\ \ ps2(chl\{*\}|p680\{n\}).phe(chl\{n\}) :: tlm \\
\Rightarrow \\
4\ \ NADHP\{n\} :: pbs :: cell+ \\
16\ \ phe\{v\} :: ps1 :: cyt\ \ + \\
ps2(chl\{p\}|p680\{*\}) :: tlm
\end{gathered}
$$

v ktorom sa nachádzajú 4 rozdielne kompartmenty, viacnásobné zanorenie komplexov a taktiež je zahrnutá aj stechiometria.

4 Rešerš možných nástrojov a aplikácií

V tejto kapitole si predstavíme zopár nástrojov, knižníc a aplikácií slúžiacich na vytváranie, manipuláciu, vizualizáciu SBGN dát.

4.1 Cytoscape

Cytoscape je open source desktopová aplikácia na vizualizáciu molekulárnych interakcií a biologických sietí. Cytoscape bol vyvinutý prevažne pre biologické účely a výskum, ale je to platforma pre všeobecná komplexnú sieťovú analýzu a vizualizáciu. Ponúka veľa funkcií pre integráciu dát, analýzu a vizualizáciu.

4.2 MINERVA

MINERVA je webová platforma slúžiaca na vizualizáciu a manažment molekulárnych interakčných sieti uložených v SBGN formáte. MINERVA poskytuje automatickú anotáciu a verifikáciu dát. Funkcie exportu umožňujú sťahovanie konkrétnych oblastí vizualizovaných sietí ako modely kompatibilné so štandardom SBGN pre efektívne opätovné použitie. Vizualizácia je generovaná s pomocou Google maps API.

4.3 CySBGN

CySBGN je dodatočný balíček, ktorý si užívateľ môže stiahnuť a nainštalovať do nástroja Cytoscape. CySBGN umožňuje importovanie, modifikovanie a analýzu SBGN máp v Cytoscape. Ponúka širokú paletu nástrojov dostupných v tejto platforme pre prácu so sieťami vo formáte SBGN. CySBGN podporuje všetky tri jazyky štandardu SBGN a umožňuje automatickú generáciu SBGN diagramov z importovaného SBML modelu. Bohužiaľ, CySBGN nepodporuje všetky typy hrán a uzlov z SBGN PD.

4.4 Biographer

Biographer je webový editor pre reakčné siete, ktorý môže byť integrovaný ako knižnica do nástrojov, ktoré pracujú s informáciami zaoberajúcimi sa sieťami. Softvér umožňuje vizualizáciu založenú na štandarde SBGN. Je schopný importovať siete v rôznych formátoch, ako sú SBML a SBGN a jSBGN (vlastný formát na zdieľanie dát). Obsahuje interaktívne nástroje na úpravu grafov a algoritmy automatického rozloženia grafov. Poskytuje samostatný editor grafov a webový server, ktorý obsahuje rozšírené funkcie, ako sú webové služby pre import a export modelov a vizualizácie v rôznych formátoch. Domovská stránka projektu je však nanešťastie nedostupná a nie je možné sa dostať k obsahu. Zdá sa, že podpora projektu skončila a nie je viac vyvíjaná.

4.5 VISIBIOweb

VISIBIOweb je bezplatná webová služba, ktorá sa používa na vizualizáciu a rozloženie modelov v BioPAX formáte (jazyk podobný SBML, ktorého cieľom je umožniť integráciu, výmenu, vizualizáciu a analýzu údajov o biologických sietiach). Je vytvorená v jazyku JavaScript a podobne ako MINERVA využíva Google maps API a zobrazuje statický obrázok v štandarde SBGN PD. Importovať je možné iba súbory typu ".owl", avšak používatelia sú schopní exportovať grafy v SBGN. Poskytuje službu automatického rozloženia grafu. Táto služba prijíma biologické siete v jednoduchom formáte XML a vráti ich späť klientovi v rovnakom formáte s pridaním informácií o rozložení (vygeneruje súradnice pre všetky SBGN komponenty), čo by sa pre koncept našej práce presne hodilo. Avšak sú tu dva problémy. Prvým je, že automatické generovanie nie je moc presné a komponenty sa často prekrývajú. Druhým, závažnejším problémom je, že podobne ako Biographer, domovská adresa služby je nedostupná a vyzerá to tak, že podpora tejto služby taktiež skončila.

4.6 SBGNViz

SBGNViz je webový prehliadač pre mapy procesov v SBGN-PD. SBGN-Viz dokáže zobraziť formáty BioPAX aj SBGN. Jedinečné vlastnosti SBGNViz zahŕňajú schopnosť zanoriť uzly do ľubovoľnej hĺbky, aby dôkladne reprezentovali molekulárne komplexy, automatické rozloženie jednotlivých komponent, editovanie a možnosť prezrieť si podrobné informácie o entitách. SBGNViz môže byť použitý vo webovom prehliadači bez akejkoľvek inštalácie a môže byť ľahko implementovaný do webových stránok. SBGNViz má dve verzie postavené pomocou jazyka ActionScript a JavaScript. Importovať je možné len validný SBGN súbor vo formáte XML, exportovanie je možné aj vo formátoch XML, PNG alebo JPG.

4.7 LibSBGN

LibSBGN je oficiálna softvérová knižnica pre čítanie, zapisovanie a manipuláciu s SBGN dátami. Knižnica (dostupná v jazykoch C++, Java a Python) uľahčuje vývojárom pridávanie SBGN nástrojov, zatiaľ čo formát súborov uľahčuje výmenu SBGN dát medzi kompatibilnými softvérovými aplikáciami. Knižnica tiež podporuje validáciu SBGN dát, čo zabezpečuje, že výsledné diagramy sú v súlade s podrobnými špecifikáciami SBGN. Knižnica umožňuje nie len vytváranie a zapisovanie SBGN máp vo formáte XML, ale taktiež vygenerovanie samotných SBGN diagramov vo formáte PNG, pomocou externého API.

4.8 Krayon

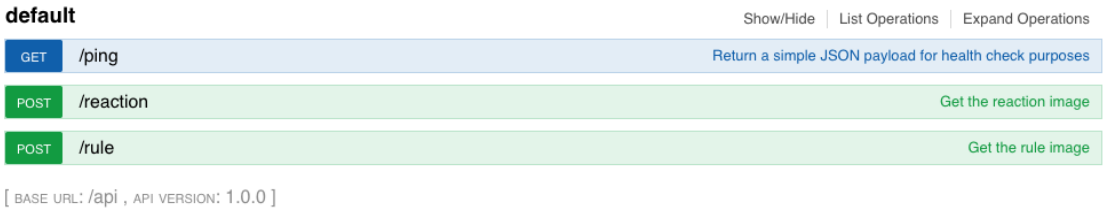
Krayon je desktopová aplikácia vyvíjaná v jazyku Kotlin, s príjemným užívateľským rozhraním. Slúži ako interaktívny editor pre SBGN diagramy. Umožňuje export SBGN diagramu v rôznych formátoch (PDF, PNG, SVG, JPG, GIF), importovať vstupné dáta sa však dá len vo formáte SBGN. Poskytuje jednoduchý interaktívny režim, ktorý pomôže rýchlo vytvoriť platný SBGN diagram.

5 Návrh a implementácia aplikácie

Úlohou práce je vymyslieť a neimplementovať nástroj, ktorý bude schopný plne automaticky, bez akéhokoľvek manuálneho, ľudského zásahu, spracovať reakciu alebo pravidlo, vytvoriť validný SBGN súbor a vygenerovať výsledný SBGN diagram vo formáte PNG alebo SVG, ktorý vráti na výstup. V predchádzajúcej kapitole sme si uviedli niekoľko nástrojov, slúžiacich na prácu a vizualizáciu SBGN dát. Hlavný nedostatok väčšiny spomenutých nástrojov je, že buď ide o online webové platformy, alebo desktopové aplikácie, ktoré potrebujú manuálny zásah na vytváranie, alebo upravovanie SBGN diagramov. Väčšina z nich taktiež na vstup potrebuje už validný SBGN súbor vo formáte XML, ktorý však my nemáme. Vhodnými nástrojmi by boli Biographer alebo VISIBIOweb, ktoré poskytovali API na automatické vytváranie rozloženia jednotlivých komponent v SBGN diagramoch, čo z implementačného hľadiska predstavuje najkomplexnejší problém tejto práce, keďže každá SBGN komponenta obsahuje svoje vlastné súradnice, výšku a šírku, čo sa prejaví na výslednej podobe diagramu. Správne rozloženie týchto komponent zohráva kľúčovú úlohu v pochopení jednotlivých pravidiel a reakcií, čo koniec koncov predstavuje hlavný cieľ tejto práce, aby užívateľské rozhranie CMP bolo obohatené o kvalitnejšiu a prehľadnejšiu vizualizáciu biochemických dát, a tým viedlo k rýchlejšiemu a jednoduchšiemu pochopeniu modelových dát. Vzhľadom na to, že nástroje Biographer a VISIBIOweb sú "mŕtveä zdá sa, že ich podpora skončila, nie je možné využiť ich služby prostredníctvom API. Najvhodnejším riešením sa pre účely práce sa javí použitie oficiálnej LibSBGN knižnice, ktorá má podporu v troch populárnych programovacích jazykoch. V spolupráci s e-cyanobacterium API, ktoré poskytuje informácie o jednotlivých reakciách a pravidlách, dôležitých na validné vygenerovanie SBGN diagramu, použitím LibSBGN knižnice, syntaktického parseru SBGN pravidiel, napísaného v C++ a ďalších nástrojov sme vytvorili aplikáciu, ktorá vytvorí a na výstup vráti validný SBGN diagram.

5.1 Návrh aplikácie

Aplikácia mala jeden hlavný cieľ, a tým bolo plne automatizované vytvorenie validných SBGN diagramov pre reakcie v SBML a pravidlá v BCSL. API je preto tvorené z dvoch HTTP POST požiadavok a jedného GET požiadavku (Obr 5.1).



Obr. 5.1: Návrh HTTP požiadavok navrhovaného API, pomocou nástroja Swagger

GET požiadavok s cestou "/ping" slúži ako indikátor, že API je spustené a riadne beží. Ak zašleme požiadavok na túto cestu, ako odpoveď dostaneme jednoduchý JSON objekt, ktorý obsahuje jediný kľúč ("pong") s hodnotou ("true"), čo značí, že API je spustené a odpovedá na HTTP požiadavky.

POST metóda s cestou "/reaction" slúži ako požiadavok na vytvorenie SBGN diagramu pre SBML reackiu. Telo validného POST požiadavku by mal byť JSON, ktorý obsahuje tri kľúče:

- model_id - číslo reprezentujúce ID modelu, v ktorom sa reakcia nachádza
- reaction_id - číslo reprezentujúce ID reakcie z modelu, ktorá má byť vizualizovaná
- as_svg - indikátor obsahujúci hodnotu ("trueälebo "false"), udávajúci, či výsledný obrázok má byť vo formáte SVG alebo PNG

Pomocou jednotlivých ID hodnôt a e-cyanobacterium API, je aplikácia schopná zistiť o aký model a konkrétnu reakciu modelu ide, a získať jednotlivé komponenty reakcie a ich typy, s ktorými môže ďalej pracovať.

Druhý POST požiadavok s cestou "/rule" , je požiadavok na vytvorenie SBGN diagramu z BCSL pravidla. Podobne ako predchádzajúce volanie, obsahuje telo volania JSON, ktorý však obsahuje len dva kľúče:

- rule - textový reťazec reprezentujúci samotné pravidlo v BCSL, napríklad: $ps2(qa-|qbn) :: tlm \Leftrightarrow ps2(qan|qb-) :: tlm$
- as_svg - rovnaký indikátor ako v predchádzajúcom volaní

Textový reťazec je pomocou aplikácie a externej knižnice spracovaný do syntaktického stromu, s ktorým potom aplikácia ďalej pracuje.

5.2 Použité nástroje

V tejto podkapitole si popíšeme nástroje, ktoré sú v aplikácii použité.

5.2.1 BCSLruleParser

BCSLruleParser je knižnica napísaná v jazyku C++, ktorú vytvoril pre účely laboratória SYBILA a BCSL J. Hradec. Knižnica slúži na syntaktické spracovanie BCSL pravidla do špeciálnej stromovej štruktúry, ktorý zachytáva a popisuje štruktúru pravidla a umožňuje jednoduchšie strojové spracovanie a prácu s BCSL pravidlom. Knižnicu ju možné pomocou mapovania použiť v jazyku PHP a Python.

5.2.2 LibSBGN

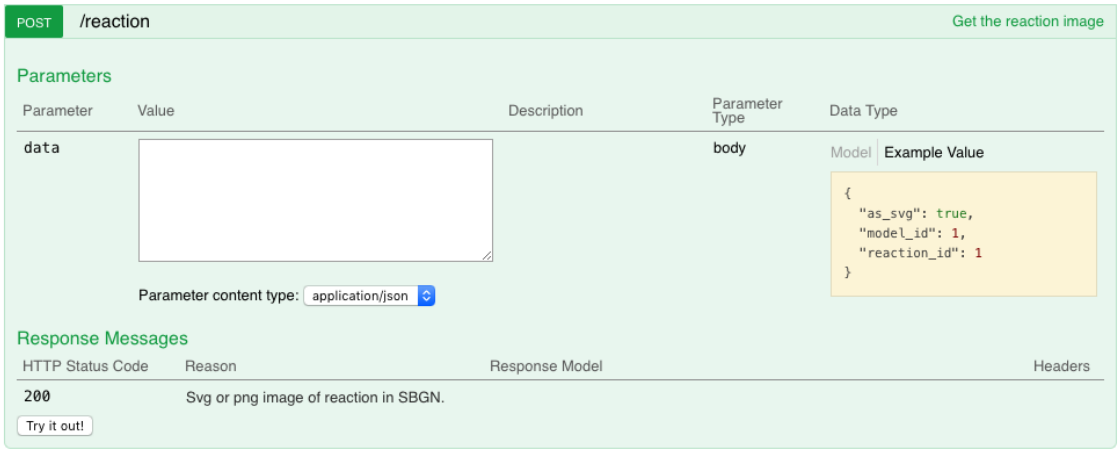
LibSBGN je už vyššie spomenutá knižnica na prácu a manipuláciu SBGN dát. Pomocou jednotlivých tried a metód je možné vytvoriť SBGN objekt, ktorý obsahuje všetky informácie o podobe výsledného diagramu, ako je jeho celková šírka a dĺžka diagramu, či informácie o jednotlivých komponentách, ktoré majú prísne dané súradnice, dĺžku a šírku. Pomocou externého API, je knižnica schopná vygenerovať SBGN diagram vo formáte PNG. Je možné taktiež exportovať SBGN vo formáte XML, ktorý je vhodný na uloženie a zdieľanie obsahu, s možnosťou výsledný SBGN diagram ďalej modifikovať (vygenerovaný obrázok už nie je možné modifikovať).

5.2.3 Potrace

Potrace je nástroj na transformovanie bit-mapových formátov (PBM, PGM, PPM alebo BMP) do jedného z niektorých vektorových formátov (SVG, PDF). Typickým príkladom použitia je vytváranie súborov SVG alebo PDF zo skenovaných údajov, ako sú firemné alebo univerzitné logá, ručne písané poznámky atď. Výsledný obrázok je potom možné vykresliť v akomkoľvek rozlíšení.

5.2.4 Swagger

Swagger je open source softvérový nástroj, tvorený veľkým počtom funkcií, ktoré pomáhajú vývojárom navrhovať, vytvárať, dokumentovať a konzumovať RESTful webové služby. Nástroj Swagger obsahuje podporu pre automatizovanú dokumentáciu, generovanie kódov a generovanie testovacích prípadov. Swagger umožňuje jednoduchý návrh aplikácie súčasne s kontrolou vstupných a výstupných dát, dokumentáciou a pomocou rozhrania, ktoré generuje, umožňuje aj možnosť jednotlivé API služby volať a testovať. V príklade (Obr. 5.2) je zobrazené Swagger rozhranie, definujúce HTTP požiadavok na vygenerovanie SGBN diagramu, pre konkrétnu reakciu. V ľavo je popis a tip parametrov API požiadavku a pravo je príklad dát, s ktorými môžme API zavolať. Rozhranie tiež zobrazuje tip a cestu API požiadavku, dokumentáciu a popis výstupných dát, to všetko na jednom mieste.



Obr. 5.2: Swagger rozhranie definujúce HTTP požiadavok na generovanie SBGN pre reakciu

5.3 Implementácia

V tejto podkapitole si bližšie popíšeme samotný postup implementácie.

Aplikácia je vyvíjaná v jazyku Python. Tento jazyk bol zvolený z viacerých dôvodov. Prvým dôvodom bola skutočnosť, že použitá knižnica LibSBGN obsahuje podporu pre Python a je možné s ňou jednoducho pracovať. Druhým dôvod bola knižnica BCSLruleParser, ktorá poskytuje mapovanie do jazyka Python a je možné ju využiť na spracovanie BCSL pravidiel z textového reťazca do jednoduchšej stromovej formy, čo značne zjednodušuje prácu s pravidlami. Tretí dôvod je fakt, že Python je populárny programovací jazyk s veľkým množstvom knižníc, ktorý sa skvelé hodí na vývoj API tohoto druhu. Umožňuje rýchly ako samotný vývoj, tak aj následné zmeny a vylepšenia aplikácií, má jednoduchú prehľadnú syntax, ktorú programátor, ktorý ovláda iní programovací jazyk rýchlo pochopí a s kvalitnou dokumentáciu a anotáciou kódu predstavuje vhodný jazyk pre vývoj tejto aplikácie. Aplikácia využíva Python 3.6.5. a beží vo virtuálnom prostredí, čo zabezpečuje jednoduchú inštaláciu systémových požiadavok na väčšine populárnych operačných systémoch.

5.3.1 Implementácia pravidiel

Aby sa zachovala správna štruktúra BCSL pravidiel a vytvoril sa validný SBGN diagram, aplikácia využíva viackrát spomínanú stromovú štruktúru pravidla (získanú pomocou knižnice BCSLruleParser), ecyano API na získanie informácií o jednotlivých entitách, a taktiež jednoduchý algoritmus na rozloženie komponent a generovanie súradníc, ktorý sme vymysleli a neimplementovali, a ktorý je postačujúci pre účely a splnenie požiadavok tejto aplikácie. Syntaktický strom rozdeľuje pravidlo na ľavú a pravú stranu, teda na reaktanty a produkty. Vytvorený algoritmus ako prvé vypočíta vnútornú štruktúru jednotlivých reaktantov a produktov. Postupným prehľadávaním uzlov stromu do hĺbky a prechádzaním jednotlivých uzlov, ktoré obsahujú informáciu o jednotlivých agentoch, sa generujú súradnice pre jednotlivé komponenty pravidla. Základná myšlienka je si vždy uchovávať informáciu o súčasných, dostupných súradniciach (*current coordinates*), kde sa ďalšia komponenta môže nachádzať. Povedzme, že mame pravidlo, ktoré obsahuje reaktant *ps2(chl{*}|p680{n}|pheo{n})*, čo je štruktúrny agent s tromi atomickými agentmi. Ako prvé sa vygenerujú súradnice X a Y určujúce pozíciu ľavého, horného rohu, pre najväčšiu komponentu (štruktúrneho agenta ps2). Následné sa *current coordinates* posunú o určitú vzdialenosť, definovanú globálne, pre celé pravidlo, aby nedošlo k tomu, že by sa jednotlivé komponenty neprekrývali a bolo ich od seba možné rozoznať. V ďalšom kroku sa generujú súradnice pre atomických agentov. Keďže tie už v sebe neobsahujú iné komponenty, tak spolu so súradnicami X a Y sa podľa dĺžky mena agenta vygeneruje výška a šírka SBGN entity, prípadne sa vygenerujú súradnice, výška a šírka taktiež pre entitu stavu atomického agenta, ktorá sa vždy nachádza na hornej hrane atomického agenta. Podobne sa vygenerujú súradnice pre zvyšných dvoch atomických agentov, posúvaním hodnoty *current coordinates*, definujúcich voľné súradnice pre ďalšie entity. Nakoniec sa určí aj výška a šírka štruktúrneho agenta, v závislosti na polohe atomických agentov zanorených v ňom. Rovnaký prístup je aplikovaný aj keď sa jedná o komplexného agenta obsahujúceho štruktúrneho agenta, či ďalší komplex, do ľubovoľnej hĺbky. Po tom čo je takto vypočítaná vnútorná štruktúra jednotlivých reaktantov a produktov sa na základe ich šírky a výšky určí poloha entity procesu a následne prebehne zarovnanie reaktantov a produktov podľa jednot-

livých kompartmentov - reaktanty a produkty nachádzajúce sa v tom istom kompartmente sa nachádzajú na rovnakej úrovni a podľa ich finálnych súradníc sa prepočítajú aj súradnice všetkých objektov v nich zanorených. Následne sa podľa najväčšej X-ovej a Y-ovej súradnice určí výsledná veľkosť celého SBGN diagramu pomocou SBGN API sa vygeneruje výsledný obrázok. Zo stromovej štruktúry nie je vždy jasné, či ide o komplexného alebo štruktúrneho agenta. Pre určenie tejto informácie sa volá e-cyano API, pomocou ktorého sa určí o akého agenta ide.

5.3.2 Implementácia reakcií

Pri reakciách je to o niečo jednoduchšie. Jednotlivé reaktanty a produkty v sebe nemôžu obsahovať iné zanorené objekty a všetky sú reprezentované jedným SBGN objektom - makromolekulou. Oproti pravidlám však navyše obsahujú modifikátory. Taktiež sa nezobrazuje kompartment. Tieto dve skutočnosti pomerne zjednodušujú výpočet súradníc pre komponenty reakcie oproti pravidlám. Reaktanty a produkty sa v diagrame nachádzajú na ľavej a pravej strane a jednotlivé modifikátory sú umiestnené hore a dole. To znamená, že reaktantov alebo produktov určí celkovú výšku a počet modifikátorov definuje výslednú šírku diagramu.

Výsledný SBGN súbor vo formáte XML, obsahuje všetky potrebné informácie na vygenerovanie SBGN diagramu vo formáte PNG. Ak vstupný API parameter *as_svg* obsahuje hodnotu *true*, aplikácia prevedie konverziu formátu PNG na SVG s využitím nástroja *Potrace* a Python knižnice PIL, ktorá slúži na manipuláciu a úpravu obrázkových súborov. Ako prvé sa pomocou PIL knižnice prevedie PNG súbor do bit-mapového formátu PPM, čo je potrebný medzikrok aby následne nástroj Potrace, mohol previesť tento súbor do formátu SVG.

V prípade že BCSL pravidlo bolo reverzibilné, sa pri generovaní SBGN súboru vytvoria dva SBGN diagramy. Ak je pravidlo reverzibilné tak v skutočnosti ide o dve samostatné pravidlá a pre lepšiu prehľadnosť je výhodnejšie vygenerovať každé vo vlastnom diagrame. Na výstup sa však vráti len jeden obrázok. Pomocou Python knižnice numpy, ktorá podobne ako PIL slúži na prácu s obrázkovými súbormi,

sa spoja tieto dve obrázky do jedného spoločného a následne je tento súbor vrátený na výstup ako jeden obrázok, prípadne ak je potrebné, je pred odoslaním konvertovaný do SVG formátu.

Aby aplikácia zbytočne nezaberala veľa miesta na disku, tým že by si ukladala súbory s obrázkami, každý obrázok, ktorý už nie je potrebný (PNG súbory a PPM súbory z medzikroku pri konvertovaní), sa z disku zmaže. Navyše pri každom volaní API, sa na začiatku volania premaže celá zložka s uloženými obrázkami. Tým sa zabráni zbytočnému zaberania miesta na disku, čo by aplikácia tohoto typu nemala robiť.

5.4 Pohľad do budúcna a možné vylepšenia

V prípade, že v budúcnosti by vznikla potreba, vizualizovať celé modely, nie len jednotlivé reakcie a pravidlá, by bolo potrebné využitie lepšieho algoritmu na generovanie súradníc. Bolo by možné využiť a upraviť niektorý zo známych, zložitejších, rozsiahlejších algoritmov pre účely platformy, aby boli schopné pracovať s BCSL modelom, alebo druhým potencionálnym rozšírením by mohlo byť editovanie a integrovanie niektorého externého nástroja, na interaktívne zarovnávanie a prezeranie SBGN diagramov, akým je napríklad SBGNViz. To už by však presahovalo koncept a rozsah bakalárskej práce a hodilo by sa ako možná téma na diplomovú prácu.