MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Rozšírenie a refaktorizácia nástroja BioDiVinE

DIPLOMOVÁ PRÁCA

Martin Demko

Brno, jar 2014

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Vedúci práce: RNDr. David Šafránek, Ph.D.

Poďakovanie

Zhrnutie

Kľúčové slová

Obsah

Úvod

Kapitola 1

Základné pojmy

Než začneme zachádzať hlbšie do problematiky tejto práce a opisovať postupy a nástroje v nej použité, je potrebné vysvetliť na počiatku niekoľko pojmov. Tieto sa v práci mnohokrát opakujú a ich včasným uvedením predídeme nepochopiteľnosti textu. Je tiež dôležité poznamenať, že táto kapitola je z veľkej miery doslovne citovaná zo zdrojov uvedených vždy na konci každej podkapitoly.

1.1 LTL - Lineárna Temporálna Logika

Temporálna logika obecne je špeciálna vetva logiky zaoberajúca sa logickou štruktúrou výrokov v čase. Je to formalizmus vhodný pre overovanie vlastností formálnych dynamických systémov resp. matematických modelov.

Lineárna temporálna logika (ďalej len LTL) je najjednoduchšia verzia temporálnej logiky, ktorá neumožňuje vetvenie času ani kvantifikátory. Môžeme ju považovať tiež za konkrétny výpočtový kalkulus pracujúci s tzv. formulami, definovanými nasledujúcou syntaxou:

Atomické propozície (ďalej len AP)

$$A > 0$$
$$B \leq 5.834$$
$$C \neq \text{"}nie\text{"}$$
atď...

Logické operátory

$\neg,\ \vee$	*– základné logické operátory*
$\wedge,\ \rightarrow,\ \leftrightarrow,\ \mathbf{true},\ \mathbf{false}$	*– odvodené logické operátory*

Temporálne operátory

- $\mathbf{X}\phi$ - ne**X**t, *vyjadruje platnosť* ϕ *v ďalšom stave*

- $\mathbf{G}\phi$ - **G**lobal, *vyjadruje trvalú platnosť* ϕ
- $\mathbf{F}\phi$ - **F**uture, *vyjadruje platnosť* ϕ *v niektorom z budúcich stavov*
- $\psi\mathbf{U}\phi$ - **U**ntil, *vyjadruje platnosť* ψ *až do kedy nezačne platiť* ϕ
- $\psi\mathbf{R}\phi$ - **R**elease, *vyjadruje platnosť* ϕ*, až dokedy nezačne platiť* ψ *a to vrátane tohto bodu. Ak* ψ *nikdy nezačne platiť, musí* ϕ *platiť do nekonečna*

  kde ϕ *a* ψ *sú atomické propozície*

Potom platí nasledujúce:

- Ak $p \in AP$, tak p je formula.
- Ak f a g sú formule, tak $\neg g$, $f \vee g$, $f \wedge g$, $f \rightarrow g$, $f \leftrightarrow g$, $\mathbf{X}g$, $\mathbf{F}g$, $\mathbf{G}g$, $f\mathbf{U}g$ a $g\mathbf{R}f$ sú formule.

Takto vytvorená LTL formula môže byť splniteľná nekonečnou postupnosťou pravdivých vyhodnotení jednotlivých $p \in AP$. Túto postupnosť si možno predstaviť ako nekonečné slovo w, pre ktoré platí $w = a_0, a_1, a_2, \ldots$ a kde a_i je pravdivostná hodnota nejakej $p \in AP$. Nech $w(i) = a_i$ a $w^i = a_i, a_{i+1}, \ldots$ je podpostupnosť alebo sufix slova w. Potom formálna definícia relácie splniteľnosti $\vDash$ medzi slovom w a LTL formulov vyzerá:

- $w \vDash p$, ak $p \in AP \bigwedge p = w(0)$
- $w \vDash \neg\phi$, ak ϕ a ψ sú LTL formule $\bigwedge w \nvDash \phi$
- $w \vDash \phi \vee \psi$, ak $w \vDash \phi \bigvee w \vDash \psi$
- $w \vDash \mathbf{X}\phi$, ak $w^1 \vDash \phi$
- $w \vDash \phi\mathbf{U}\psi$, ak $\exists i,\ i \geq 0 \wedge w^i \vDash \psi \bigwedge \forall k,\ 0 \leq k < i \wedge w^k \vDash \phi$

Predchádzajúce platí pre základné logické a temporálne operátory, ktoré majú ale dostatočne expresívnu silu, aby s ich pomocou mohli byť zadefinované ľubovolné LTL formule. Ovšem pre uľahčenie zápisu aj čítania, si môžeme dodefinovať rozšírenú paletu operátorov za predpokladu platnosti predchádzajúcich pravidiel:

- $\phi \wedge \psi \ \equiv \ \neg(\neg\phi \vee \neg\psi)$, ak ϕ, ψ sú LTL formule
- $\phi \rightarrow \psi \ \equiv \ \neg\psi \vee \psi$,
- $\phi \leftrightarrow \psi \ \equiv \ (\phi \rightarrow \psi) \vee (\psi \rightarrow \phi)$,

- $\mathbf{true} \equiv p \vee \neg p$, ak $p \in AP$
- $\mathbf{false} \equiv \neg\mathbf{true}$,
- $\phi\mathbf{R}\psi \equiv \neg(\neg\phi\mathbf{U}\neg\psi)$,
- $\mathbf{F}\phi \equiv \mathbf{true}\mathbf{U}\phi$,
- $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$,

Napriek tomu, že je LTL tou najprimitívnejšou temporálnou logikou, jej prevod do Büchiho automatu je v najhoršom prípade exponenciálne zložitý. Dôvod tohto prevodu bude vysvetlený v kapitole 1.5. [13]

1.2 Büchiho automat

Automat obecne je matematický model stroja s konečným množstvom pamäte spracovávajúci vstup o neznámej veľkosti. Kvôli obmedzeniu pamäte ho nazývame konečným automatom. Vstup sa nazýva slovo a môže byť konečný aj nekonečný. Büchiho automat je potom najjednoduchším konečným automatom nad nekonečným slovom a preto patrí do skupiny ω-automatov.

Formálne je konečný automat $\mathcal{A}$ pätica $(\Sigma, Q, \Delta, Q_0, F)$, pre ktorú platí:

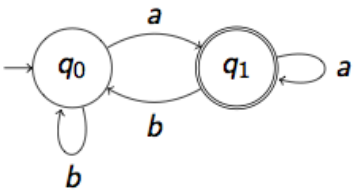
- Σ je konečná abeceda
- Q je konečná množina stavov
- $\Delta \subseteq Q \times \Sigma \times Q$ je relácia nazývaná prechodová funkcia
- $Q_0 \subseteq Q$ je podmnožina množiny stavov, nazývaná iniciálne stavy
- $F \subseteq Q$ je podmnožina množiny stavov, nazývaná akceptujúce stavy

Príklad jednoduchého automatu je daný na obrázku 1.1.

Automat nad konečným slovom akceptuje toto slovo, ak po prejdení posledného znaku slova zodpovedajúceho prechodu medzi dvoma stavami, je tento posledný stav v množine F. Avšak automat nad nekonečným slovom nemôže nikdy prejsť cez posledný znak. Preto takýto automat akceptuje nekonečné slovo len v prípade, že počas prechádzania slova je aspoň jeden stav navštívený nekonečne často a zároveň tento stav patrí aj do množiny F.

Automaty môžeme ešte rozlíšiť na deterministické a nedeterministické. Deterministický automat má jednoznačne určené prechody medzi stavmi.

Tým sa myslí, že zo stavu $q \in Q$ sa pod znakom $s \in \Sigma$ dá prejsť maximálne do jednoho stavu $q^{'} \in Q$. Naproti tomu nedeterministické automaty umožňujú prechod zo stavu q pod slovom s do stavov $Q^{'} \subseteq Q$. Našťastie existuje algoritmus prevodu nedeterministického konečného automatu na deterministický, ale iba nad konečným slovom. Nevýhodou je ale veľký nárast počtu stavov. [14]



Obr. 1.1: Jednoduchý deterministický automat $\mathcal{A} : \Sigma = \{a, b\}, Q = \{q_0, q_1\}, \Delta = \{(q_0, a, q_1), (q_0, b, q_0), (q_1, a, q_1), (q_1, b, q_0)\}, Q_0 = \{q_0\}, F = \{q_1\}$

1.3 Zákon o mass action kinetike

Tento zákon vyjadruje základné pravidlo fungovania chemických reakcií. Konkrétne rýchlosť s akou chemické substancie, či už veľké makromolekuly alebo malé ióny, do seba narážajú a interagujú za tvorby nových chemických látok. Predpokladajme, že substráty A a B spolu reagujú za vzniku novej látky, produktu C,

$$A + B \xrightarrow{k} C. \tag{1.1}$$

Rýchlosť tejto reakcie predstavuje rýchlosť tvorby produktu C, a síce $\frac{d[C]}{dt}$, ktorá stelesňuje počet kolízií za jednotku času medzi reaktantami A a B a zároveň pravdepodobnosť, že tieto kolízie majú dostatočnú energiu na prekonannie aktivačnej energie reakcie. To samozrejme záleží po prvé na koncentrácii reaktantov, ale aj na ich tvare a veľkosti a tiež na teplote a ph roztoku. Kombináciou týchto a aj d'alších faktorov dostávame nelineárnu ordinárnu diferenciálnu rovnicu (z *ang.* ordinary differential equation, skrátene ODE) [26]:

$$\frac{d[C]}{dt} = k[A][B] \tag{1.2}$$

Identifikácia vzťahu 1.1 s rovnicou 1.2 sa nazýva zákon mass action kinetiky (z *ang.* law of mass action) a konštanta k je potom rýchlostná konštanta (z *ang.* rate constant) tejto reakcie.

V skutočnosti nejde o zákon ako taký. Nie je to neporušiteľné pravidlo ako Newtonov gravitačný zákon, ale skôr veľmi užitočný model, ktorý však nemusí byť vo všetkých prípadoch validný. [17]

1.4 Michaelis-Mentenovej a Hillova kinetika

Je nutné začať od enzýmovej kinetiky, pretože práve tá bola hlavným katalyzátorom pre objavenie nových spôsobov modelovania chemických reakcií. Tiež je dobré si uvedomiť, prečo tomu tak bolo. Enzýmová kinetika totiž patrí medzi niekoľko prípadov, v ktorých použitie mass action kinetiky nie je validné. Podľa nej sa so zvyžujúcou koncentráciou substrátu S zvyšuje rýchlosť reakcie zlučovania s enzýmom E lineárne. Zatiaľ čo v *in-vivo* prípade táto rýchlosť postupne konverguje k určitému maximu, cez ktoré sa nedá dostať ani dodatočným zvýšením koncentrácie substrátu S.

Model, ktorý vysvetľoval túto odchýlku od zákona o mass action kinetike bol prvý krát prezentovaný v roku 1913 nemeckým biochemikom Leonardom Michaelisom a kanadskou fyzičkou Maud Mentenovou (z toho *kinetika Michaelis-Mentenovej*). V ich reakcii premieňal enzým E subtrát S na produkt P v dvoch fázach. Prvá bola reverzibilná reakcia zlučovania S s E za vzniku enzým-substrátového komplexu C a druhá predstavovala rozpad komplexu C za vzniku produktu P a uvoľnenia nezmeneného enzýmu E (viď rovnicu 1.3).

$$S + E \underset{k_2}{\overset{k_1}{\rightleftarrows}} C \xrightarrow{k_3} E + P \tag{1.3}$$

Je dôležité si všimnúť, že druhá reakcia nie je reverzibilná.

Existujú dva spôsoby ako analyzovať túto rovnicu a obe sú si veľmi podobné. Ide o rovnovážnu aproximáciu a aproximáciu kvázistacionárneho stavu. Začneme aplikáciou zákona o mass action na tieto reakcie, čo nám vo výsledku dá nasledujúce diferenciálne rovnice vyjadrujúce rýchlosti zmien jednotlivých chemických látok:

$$\frac{d[S]}{dt} = k_2[C] - k_1[S][E], \tag{1.4}$$

$$\frac{d[E]}{dt} = (k_2 + k_3)[C] - k_1[S][E], \tag{1.5}$$

$$\frac{d[C]}{dt} = k_1[S][E] - (k_2 + k_3)[C], \tag{1.6}$$

$$\frac{d[P]}{dt} = k_3[C]. \tag{1.7}$$

Všimnite si, že platí $\frac{d[C]}{dt} + \frac{d[E]}{dt} = 0$ a preto

$$[E] + [C] = [E_0], \tag{1.8}$$

kde $[E_0]$ je absolútna koncentrácia enzýmu v reakcii. [18]

A. Rovnovážna aproximácia

V pôvodnej analýze Michaelis a Mentenová predpokladali, že substrát je v neustálej rovnováhe s enzým-substrátovým komplexom, a teda že platí

$$k_1[S][E] = k_2[C]. \tag{1.9}$$

Potom na základe platnosti vzťahov 1.8 a 1.9, môžeme zadefinovať nasledujúci vzťah:

$$[C] = \frac{[E_0][S]}{K_s + [S]}, \tag{1.10}$$

kde $K_s = \frac{k_2}{k_1}$. Z toho zase vyplíva, že rýchlosť reakcie V, respektíve rýchlosť tvorby produktu P môže byť zadaná takto:

$$V = \frac{d[P]}{dt} = k_3[C] = \frac{k_3[E_0][S]}{K_s + [S]} = \frac{V_{max}[S]}{K_s + [S]}, \tag{1.11}$$

kde $V_{max} = k_3[E_0]$ je maximálna reakčná rýchlosť a predstavuje prípad, keď všetky molekuly enzýmu sú naviazané na substrát.

Pri malej koncentrácii substrátu je rýchlosť reakcie lineárna, ak táto koncentrácia nepresiahne celkové množstvo enzýmu. Avšak pri väčších koncentráciách substrátu je rýchlosť reakcie limitovaná množstvom enzýmu a disociačnou konštantou (v našom prípade k_3). Ak sa koncentrácia substrátu približuje hodnote K_s, znamená to, že reakčná rýchlosť je rovná polovičke svojho maxima.

Poznamenajme však, že vzťah 1.9 v reálnych podmienkach takmer nikdy neplatí a rýchlosť reakcie je tak ovplyvnená aj disociačnou konštantou enzým-substrátového komplexu v smere spätného rozpadu (v našom prípade k_2). Práve z tohto dôvodu tu hovoríme o aproximácii. [18]

B: Aproximácia kvázistacionárneho stavu

Inou alternatívou analýzi tejto enzymatickej reakcie je práve aproximácia kvázistacionárneho stavu (z *ang.* Quasi-steady state aproximation), ktorá je v dnešnej dobe aj najpoužívanejšia. Jej tvorcovia, George Briggs a J.B.S.

Haldane predpokladali, že rýchlosť tvorby enzým-substrátového komplexu aj jeho disociácia sú si od počiatku rovné. Takže platí $\frac{d[C]}{dt} = 0$.

Aby sme dali tejto aproximácii správny matematický základ, je vhodné zadefinovať nasledujúce bezrozmerné [1] premenné:

$$\sigma = \frac{[S]}{[S_0]}, \qquad \chi = \frac{[C]}{[E_0]}, \qquad \tau = k_1[E_0]t,$$
$$\kappa = \frac{k_2 + k_3}{k_1[S_0]}, \qquad \epsilon = \frac{[E_0]}{[S_0]}, \qquad \alpha = \frac{k_2}{k_1[S_0]}, \tag{1.12}$$

s pomocou ktorých dostaneme systém iba dvoch diferenciálnych rovníc:

$$\frac{d\sigma}{d\tau} = -\sigma + \chi(\sigma + \alpha), \tag{1.13}$$
$$\epsilon\frac{d\chi}{d\tau} = \sigma - \chi(\sigma + \kappa). \tag{1.14}$$

V porovnaní s koncentráciou substrátu nejakej reakcie je koncentrácia enzýmu vo väčšine prípadov oveľa menšia. Táto skutočnosť pekne odzrkadluje efektívnosť enzýmov ako katalyzátorov chemických reakcií. Preto je ϵ veľmi malé, typicky v rozmedzí od 10^{-2} do 10^{-7}. Napriek tomu je reakcia 1.14 rýchla a tiež rýchlo spadá do rovnováhy, v ktorej zostáva aj keď sa hodnota premennej σ mení. Preto uvažujeme túto aproximáciu ako $\epsilon\frac{d\chi}{d\tau} = 0$. Toto tvrdenie je ekvivalentné tomu úvodnému, že $\frac{d[C]}{dt} = 0$.

Potom z tejto aproximácie vyplíva:

$$\chi = \frac{\sigma}{\sigma + \kappa}, \tag{1.15}$$
$$\frac{d\sigma}{d\tau} = -\frac{q\sigma}{\sigma + \kappa}, \tag{1.16}$$

kde $q = \kappa - \alpha = \frac{k_3}{k_1[S_0]}$. Rovnica 1.16 popisuje rýchlosť prírastku substrátu a je pomenovaná zákon Michaelis-Mentenovej (z *ang.* Michaelis-Menten law). V znení pôvodných premenných tento zákon vyzerá takto:

$$V = \frac{d[P]}{dt} = -\frac{d[S]}{dt} = \frac{k_3[E_0][S]}{[S] + K_m} = \frac{V_{max}[S]}{[S] + K_m}, \tag{1.17}$$

kde $K_m = \frac{k_2+k_3}{k_1}$. Vzťahy 1.11 a 1.17 sú si už na prvý pohľad veľmi podobné. Jediným rozdielom sú konštanty K_s a K_m. Jedná sa o dva podobné výsledky na základe rôznych predpokladov.

---

1. Existuje viacero spôsobov, ako systém diferenciálnych rovníc previesť na bezrozmerný. O tom však táto práca nepojednáva. Ďalšie informácie ohľadom tejto problematiky je možné nájsť v [19].

Tak ako zákon o mass action kinetike aj Michaelis-Mentenovej zákon 1.17 nie je platný univerzálne. Je však veľmi užitočný a používaný, pretože koeficient K_m je experimentálne dobre pozorovateľný a teda aj ľahko merateľný narozdiel od individuálnych rýchlostných konštánt jednotlivých chemických substancií v reakcii. [18]

C: Enzýmová spolupráca

Reakčná rýchlosť mnohých enzýmov nemá klasickú hyperbolickú krivku, tak ako je predpokladané pri použití Michaelis-Mentenovej kinetiky, ale miesto toho má sigmoidálny charakter. To je spôsobené vlastnosťou týchto enzýmov, ktorá im umožňuje viazať na seba viacero substrátov naraz a zároveň tým ovplyvniť obtiažnosť tohto viazania. Táto vlastnosť sa nazýva kooperácia alebo súčinnosť, či spolupráca.

Tieto enzýmy majú viac ako len jedno viazacie miesto (z *ang.* binding site) a pri naviazaní prvej molekuly dochádza k zmene konformácie vzniknutého komplexu, čo môže ovplyvniť naviazanie ďalšej molekuly pozitívne, ale aj negatívne. Pre každú novú molekulu tento proces pokračuje rekurzívne.

Predpokladajme, že enzým E dokáže viazať až dve molekuly substrátu S, takže sa môže nachádzať v troch rôznych stavoch:

1. voľná molekula enzýmu (E),
2. komplex s jednou naviazanou molekulov substrátu (C_1),
3. komplex s dvoma naviazanými molekulami substrátu (C_2).

Potom samotné reakcie vyzerajú nasledovne:

$$S + E \underset{k_2}{\overset{k_1}{\rightleftarrows}} C_1 \xrightarrow{k_3} E + P, \tag{1.18}$$

$$S + C_1 \underset{k_5}{\overset{k_4}{\rightleftarrows}} C_2 \xrightarrow{k_6} C_1 + P. \tag{1.19}$$

Použitím zákona o mass action kinetike dostaneme najskôr päť diferenciálnych rovníc a po úprave len tri. Následným uplatnením aproximácie kvázistacionárneho stavu dostaneme:

$$[C_1] = \frac{K_2[E_0][S]}{K_1K_2 + K_2[S] + [S]^2}, \tag{1.20}$$

$$[C_2] = \frac{[E_0][S]^2}{K_1K_2 + K_2[S] + [S]^2}, \tag{1.21}$$

kde $K_1 = \frac{k_2+k_3}{k_1}$, $K_2 = \frac{k_5+k_6}{k_4}$ a $[E_0] = [E] + [C_1] + [C_2]$. Reakčná rýchlosť potom vyzerá takto:

$$V = k_3[C_1] + k_6[C_2] = \frac{(k_3K_2 + k_6[S])[E_0][S]}{K_1K_2 + K_2[S] + [S]^2}. \tag{1.22}$$

Ak budeme teraz pre ukážku uvažovať prípad pozitívnej spolupráce, tak naviazanie prvej molekuly S bude relatívne pomalé, ale naviazanie druhej molekuly S už bude rýchlejšie. Tento jav môžeme vyjadriť ako $k_4 \to \infty$ a zároveň $k_1 \to 0$, zatiaľ čo k_1k_4 je konštantná hodnota. V tomto prípade ale naopak platí, že $K_1 \to \infty$ a $K_2 \to 0$, zatiaľ čo K_1K_2 je tiež konštantné. Po aplikácii týchto nových obmedzení na vzťah 1.22 dostávame:

$$V = \frac{k_6[E_0][S]^2}{K_m^2 + [S]^2} = \frac{V_{max}[S]^2}{K_m^2 + [S]^2}, \tag{1.23}$$

kde $K_m^2 = K_1K_2$ a $V_{max} = k_6[E_0]$.

Obecne sa dá povedať, že ak enzým dokáže viazať n molekúl substrátu, existuje tiež n rovnovážnych konštánt $K_1, \ldots, K_n$, pre ktoré bude platiť $K_n \to 0$ a $K_1 \to \infty$, zatiaľ čo K_1K_n bude stále konštanta a obecná rovnica reakčnej rýchlosti je

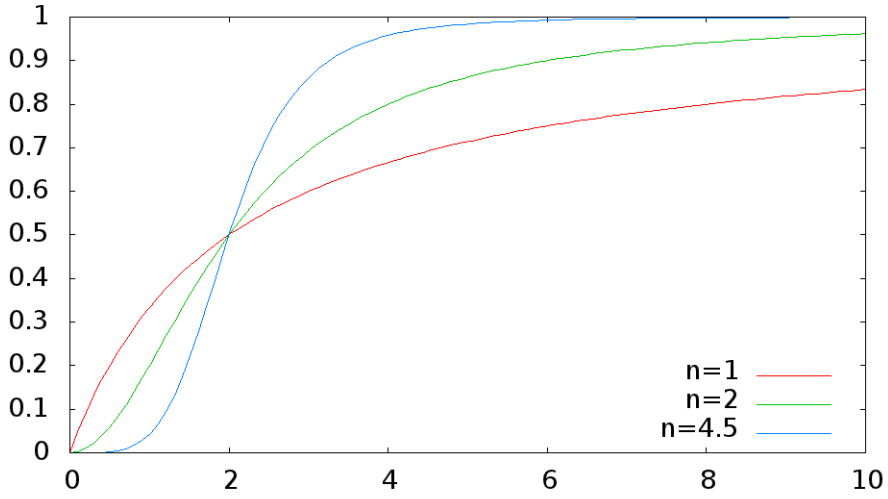
$$V = \frac{V_{max}[S]^n}{K_m^n + [S]^n}, \tag{1.24}$$

kde $K_m^n = \prod_{i=1}^{n} K_i$. Táto rovnica je známa ako Hillova rovnica alebo Hillova kinetika a konštanta K_m^n vyjadruje koncentráciu, pri ktorej je rýchlosť reakcie v polovičke svojho maxima, tj. $\frac{V_{max}}{2}$. Typicky faktor n, vyjadrujúci strmosť reakčnej krivky (obr. 1.2) býva menší ako skutočný počet viazacích miest na enzýme, často to dokonca nie je ani celé číslo. Stojí za zmienku, že v prípade $n = 1$ zodpovedá Hillova kinetika Michaelis-Mentenovej kinetike, preto náš nástroj ponúka iba možnosť zadania Hillovej rovnice (viď kapitolu 2.B). [18]

1.5 Model checking

Model checking alebo overovanie modelov je automatizovaná technika formálnej verifikácie špecifikovaných vlastností konečného stavového systému. Hlavnou výzvou tejto problematiky je úspešne si poradiť s problémom explózie stavového priestoru (z *ang.* state space explosion).

Proces overovania modelov pozostáva z nasledujúcich podúloh:



Obr. 1.2: Faktor n Hillovej kinetiky pre veľkosti $1, 2$ a 4.5 ($K_m = 2$)

Modelovanie Prvou úlohou je prevedenie skúmaného systému do formálneho matematického modelu, ktorý bude akceptovaný vybraným overovacím nástrojom. V niektorých prípadoch je to ľahká úloha, avšak v iných je potreba použiť vhodné abstrakcie za účelom odstránenia irelevantných detailov alebo naopak zvýraznenia istých črtov skúmaného systému.

Špecifikácia Ešte pred samotným overovaním, je nevyhnutné špecifikovať vlastnosti, ktoré má skúmaný systém spĺňať, pretože práve tie budeme ďalej overovať. Špecifikáciou sa myslí použitie nejakého vhodného formalismu. Typicky napríklad temporálnej logiky, ktorá umožňuje skúmať správanie systému v čase. My budeme používať LTL (viď 1.1).

Overovanie Model checking dokáže overiť, či model vyhovuje danej špecifikácii, ale nedokáže rozhodnúť, či daná špecifikácia pokrýva všetky vlastnosti, ktorým skúmaný systém vyhovuje. Toto je veľmi významný problém formálneho overovania modelov.

Výsledkom overovanie modelu je buď tvrdenie áno (v zmysle model spĺňa vlastnosť) alebo nie (model nespĺňa vlastnosť) a v tomto prípade by mal použitý nástroj poskytnúť možnosť trasovania chyby (z *ang.* error trace). Táto chybová trasa grafom sa obvykle používa ako protipríklad k overovanej vlastnosti. S jej pomocou môžeme lep-

šie pochopiť dôvod a miesto vzniku chyby a upraviť systém podľa toho. [15]

Prvé algoritmy pre overovanie modelov používali ako formalizmus matematického modelu daného systému Kripkeho štruktúru. Je to obdoba nedeterministického konečného automatu (viď kapitolu 1.2), ktorého stavy sú označené výrazmi z množiny 2^{AP}, kde AP sú atomické propozície formule f (viď kapitolu 1.1). Všetky tieto stavy sú akceptujúce.

Neskôr sa používal μ-kalkulus, ale v súčasnosti sú najrozšírenejším formalizmom automaty. Konkrétne budeme pojednávať o použití Büchiho automatu (viď kapitolu 1.2). Tento je veľmi vhodný, pretože sa s jeho pomocou dá vyjadriť rovnako model systému, ako aj po vynaložení určitého výpočtového úsilia, špecifikácia vlastnosti.

Prvou fázou tvorby modelu systému je vytvorenie Kripkeho štruktúry, ktorá sa ale veľmi jednoducho prevedie na automat $\mathcal{A}$. Špecifikáciu vlastnosti môžeme vyjadriť ako automat $\mathcal{S}$. Potom sú oba tieto automaty vytvorené nad rovnakou abecedou $\Sigma = 2^{AP}$. Overenie modelu teraz znamená jednoducho zistiť, či platí $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{S})$, kde $\mathcal{L}(\mathcal{A})$ je jazyk zodpovedajúci automatu $\mathcal{A}$ a $\mathcal{L}(\mathcal{S})$ je jazyk zodpovedajúci automatu $\mathcal{S}$ (Jazyk je množina všetkých slov, ktoré je možno vygenerovať príslušným automatom.). To znamená, že každé chovanie modelovaného systému, určeného jazykom $\mathcal{L}(\mathcal{A})$ sa nachádza medzi povolenými chovaniami, určenými jazykom špecifikácie $\mathcal{L}(\mathcal{S})$.

Toto tvrdenie môžeme preformulovať. Nech $\overline{\mathcal{L}(\mathcal{S})}$ je komplement k $\mathcal{L}(\mathcal{S})$, potom pri overovaní modelu dokazujeme, či platí $\mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{S})} = \emptyset$. Ak je tento prienik neprázdny, predstavuje to chovanie, ktoré je protipríkladom k overovanej vlastnosti. Použiť predchádzajúcu formuláciu nám umožňuje vedomosť, že Büchiho automaty sú uzavreté na prienik a doplnok (komplement). Vďaka tomu môžeme urobiť nasledovné:

$$\begin{aligned} \emptyset = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}^{'}) = \mathcal{L}(\mathcal{A} \cap \mathcal{S}^{'}) = \mathcal{L}(\mathcal{M}), \\ \text{kde } \mathcal{M} = \mathcal{A} \cap \mathcal{S}^{'} \text{ a } \mathcal{S}^{'} = \mathcal{S}(\neg\phi), \\ \text{kde } \phi \text{ je LTL formula a } \neg\phi \text{ jej negácia,} \end{aligned}$$

potom $\mathcal{S}^{'}$ je komplement $\mathcal{S}(\phi)$ a v skutočnosti dokazujeme prázdnosť jazyka $\mathcal{L}(\mathcal{M})$ (Pre úplnosť treba dodať, že $\mathcal{S}(\phi)$ je automat $\mathcal{S}$ zkonštruovaný zo špecifikácie vlastnosti ϕ.).

Výhodou použitia automatov pre vyjadrenie modelu skúmaného systému aj špecifikácie vlastnosti je, že stačí skonštruovať automat pre vlastnosť (viď kapitolu 1.5.1) a automat modelu sa konštruuje za behu samotného overovania pomocou algoritmu pre prienik automatov. To znamená,

že v mnohých prípadoch môžeme vyvrátiť splniteľnosť oveľa skôr, než sa vytvorí celý stavový priestor automatu modelu nájdením prvého protipríkladu. Čo značne šetrí čas a priestor. Táto technika sa nazýva overovanie modelu za behu (z *ang*. on-the-fly model checking). [14]

1.5.1 Prevod LTL do BA

Ako sme už predtým niekoľko krát spomenuli, prevod LTL formule na Büchiho automat nie je primitívny algoritmus. V skutočnosti je príliš rozsiahly pre naše potreby zoznamovania sa s problematikou. Z tohto dôvodu ho tu nebudeme rozoberať, avšak uvedieme niekoľko dôležitých informácii, ktoré sa tohto problému týkajú.

Ešte pred začiatkom sa musí samotná LTL formula ϕ previesť do negatívnej normálnej formy (z *ang.* negation normal form). Najskôr sa preformulujú niektoré temporálne operátory:

- $\mathbf{F}\ \phi\ \Rightarrow\ \mathbf{true}\ \mathbf{U}\ \phi$
- $\mathbf{G}\ \phi\ \Rightarrow\ \mathbf{false}\ \mathbf{R}\ \phi$

a tiež logické operátory, tak aby zostali iba $\wedge$, $\vee$ a $\neg$. V poslednom kroku prípravy sú všetky negácie presunuté dovnútra:

- $\neg(\psi\ \mathbf{U}\ \phi)\ \Rightarrow\ (\neg\psi)\ \mathbf{R}\ (\neg\phi)$
- $\neg(\psi\ \mathbf{R}\ \phi)\ \Rightarrow\ (\neg\psi)\ \mathbf{U}\ (\neg\phi)$
- $\neg(\mathbf{X}\ \phi)\ \Rightarrow\ \mathbf{X}\ (\neg\phi)$

Ďalej pokračuje dlhý algoritmus [16], ktorého výsledkom je Büchiho automat $\mathcal{S}$. Jeho konštrukcia má exponenciálnu časovú aj priestorovú zložitosť závislú od veľkosti formule ϕ. Avšak v praxi býva takto skonštruovaný automat ovykle menší.

Je dôležité ešte raz pripomenúť, že pri overovaní modelu vlastne nezisťujeme, či model spĺňa vlastnosť, ale naopak zisťujeme, či model nespĺňa opak vlastnosti. Dôvodom k tomu je, že je výpočtovo ľahšie vytvoriť automat z negácie vlastnosti, ako vytvoriť komplement automatu z pôvodnej vlastnosti, ktorý by mohol mať až dvojnásobne exponenciálnu priestorovú zložitosť. [16]

1.5.2 Farebný model checking

Farebný model checking alebo tiež paralelný sa snaží vysporiadať s parametrizáciou formálneho modelu, ktorá ho rozširuje o nový rozmer. Ak sa

budeme na jednotlivé parametre $p_i \in P$, kde P je množina všetkých neznámych parametrov dívať ako na intervaly alebo skôr ako na množiny hodnôt, vďaka diskretizácii. Potom kombinácia evaluacií všetkých neznámych parametrov tvorí parametrický priestor $\mathcal{P} = \prod_{i=1}^{n}[min(p_i), max(p_i)]$, kde $n = |P|$. Tento rozširuje model systému o parametrizované multi-afinné funkcie $f_i(x, \pi_i)$, kde $\pi_i \in \mathcal{P}$ namiesto bežných multi-afinných funkcií $f_i(x)$, kde $x = (x_1, \ldots, x_n)$ je vektor premenných a $f = (f_i, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ je vektor multi-afinných funkcií (viď kapitolu 2). Hodnota π_i vyjadruje konkrétnu evaluáciu parametrov v celom modeli a označujeme ju ako farba.

Aby sa zabránilo opakovanému vytváraniu automatu pre každú evaluáciu π_i za účelom overenia modelu, bola vytvorená nová pomocná štruktúra pomenovaná parametrizovaná Kripkeho štruktúra (ďalej len PKS). Tá v sebe tradične uchováva celý stavový priestor modelu, ale navyše s novou informáciou, ktorá rozhoduje pod ktorou farbou, resp. konkrétnou evaluáciou parametrov sa dá prejsť z prechodu s do prechodu $s^{'}$. Každý prechod musí byť uschopnený aspon pod jednou farbou, ale zároveň môže byť aj pod všetkými. Celý stavový priestor je tak zjednotenie všetkých jednofarebných stavových priestorov.

Vďaka tomu prebieha overovanie všetkých parametrizácií modelu naraz. Výsledkom je najväčšia množina parametrizácií, v ktorých model spĺňa danú vlastnosť. [1]

Kapitola 2

Biochemický dynamický vstupný model

Táto kapitola je z veľkej miery doslovne citovaná z nižšie uvedených zdrojov.

A. Multi-afinný ODE model

Vstupným modelom sa u nás myslí model biochemických reakcií, ktorý je v našom poňatí braný ako po častiach multi-afinný systém diferenciálnych rovníc. Ale začnime od počiatku a postupne sa dopracujme k tomuto výsledku.

Na základe pravidla o mass action kinetike (viď 1.3) je možné modelovať ľubovolnú biochemickú reakciu alebo dokonca sústavu takýchto reakcií pomocou sústavy nelineárnych ODE [26].

Uvažujme multi-afinný systém vo forme $\dot{x} = f(x)$, kde $x = (x_1, \ldots, x_n)$ je vektor premenných a $f = (f_1, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ je vektor multi-afinných funkcií. Tieto funkcie sú vlastne polynómy, v ktorých je stupeň premenných $x_1, \ldots, x_n$ obmedzený na hodnotu 1. Každá premenná x_i, kde $i \in \{1, \ldots, n\}$ predstavuje koncentráciu špecifickej chemickej látky a je interpretovaná ako $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$. /*Mozno priklad*/

Z dôvodu, že premenné môzeme vyjadriť len ako nezáporné reálne čisla, je možné tiež spojitý stavový priestor našeho matematického systému obmedziť iba na prvý, resp. kladný kvadrant $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x \geq 0\}$.

Ak uvažujeme o premenných ako o nestabilných chemických látkach, ktoré sami od seba degradujú v čase, môžme s kľudom obmedziť náš spojitý stavový priestor $\mathcal{D}$ ešte viac. A síce na n-dimenzionálny obdĺžnik $\mathcal{D} = \prod_{i=1}^{n}[0, max_i] \subset \mathbb{R}^n$, kde max_i je horná hranica koncentrácie premennej x_i. [5][6]

B. Po častiach multi-afinný ODE model

Multi-afinný systém diferenciálnych rovníc dokáže pokryť skoro celú mass action kinetiku s jedinou výnimkou. A tou sú homodiméry a reakcie s

nimi spojené. Dôvodom je predchádzajúce obmedzenie multi-afinných funkcií $f_1, \ldots, f_n$ s ohľadom na stupeň premenných $x_1, \ldots, x_n$.

Teoreticky sme schopný popísať akýkoľvek biochemický model pomocou pravidiel tejto kinetiky. V skutočnosti, ak sa pokúsime formulovať tieto pravidlá pre rozsiahly model, zistíme, že s narastajúcou veľkosťou rastie komplexita týchto pravidiel a navyše k úplnosti modelu je potrebné poznať veľké množstvo čo najpresnejšie vyčíslených parametrov. Práve tento druhý problém môže byť v niektorých prípadoch experimentálne neriešiteľný. Či už ide o veľké enzymatické komplexy, alebo (o látky s veľmi krátkou existenciou / o veľmi rýchlo degradujújce látky).

Práve preto sa ponúkajú možnosti aproximácie, ktoré nie len zmenšujú systém a tým aj dimenzionalitu matematického modelu, ale zjednodušujú aj výpočtovú zložitosť. Takouto možnosťou je aj aproximácia kvázistacionárneho stavu (viď 1.4.B). Napríklad Michaelis-Mentenovej kinetika (viď. 1.4), či obecnejšia Hillova kinetika (viď. 1.4.C) a tiež sigmoidálne prepínače publikované na konferencii CAV (Grosu et al. 2011) [11]. Všetky vyššie zmienené abstrakcie náš nástroj ponúka a budeme ich jednotne označovať ako regulačné funkcie $\rho(x)$, ktoré budeme matematicky definovať neskôr.

Takto redukované difernciálne rovnice majú formu racionálnych polynómov, získaných ako lineárna kombinácia týchto regulačných funkcí, medzi ktoré patria aj Heavisideove alebo schodové funkcie [27]. V skutočnosti výsledný matematický model nie je multi-afinný, ale na druhú stranu je ho možné aproximovať v zmysle po častiach multi-afinného systému. A to tak, že nahradíme všetky regulačné funkcie sústavou vhodných po častiach lineárnych rampových funkcií. Tieto sú definované nasledovne:

$$r^+coor(x_i, \theta_i, \theta_i^{'}, y, y^{'}) \;=\; \begin{cases} y, & \text{pre } x_i < \theta_i, \\ y + (y^{'} - y)\frac{x_i - \theta_i}{\theta_i^{'} - \theta_i}, & \text{pre } \theta_i < x_i < \theta_i^{'}, \\ y^{'}, & \text{pre } x_i > \theta_i^{'}. \end{cases} ;$$

$$r^+(x_i, \theta_i, \theta_i^{'}, a, b) \;=\; r^+coor(x_i, \theta_i, \theta_i^{'}, a\theta_i + b, a\theta_i^{'} + b);$$

kde $i \in \{1, \ldots, n\}$,
$y = x_j, y^{'} = x_j^{'}; j \in \{1, \ldots, n\} \wedge j \neq i$,
$\theta_i, \theta_i^{'} \in \mathbb{R}^+,\ \theta_i < \theta_i^{'} \leq max_i$,
$a, b \in \mathbb{R}$.

Potom klesajúce rampové funkcie sú definované ako kvantitatívny do-

plnok rastúcich:

$$\begin{aligned} r^- coor(x_i, \theta_i, \theta_i^{'}, y, y^{'}) &= 1 - r^+ coor(x_i, \theta_i, \theta_i^{'}, y, y^{'}) \\ r^-(x_i, \theta_i, \theta_i^{'}, a, b) &= 1 - r^+(x_i, \theta_i, \theta_i^{'}, a, b) \end{aligned}$$

Už zmienené regulačné funkcie majú nasledujúce formy:

$$\begin{aligned} hill^+(x_i, \theta_i, d, a, b) &= a + (b-a)\frac{[x_i]^d}{[\theta_i]^d + [x_i]^d}; \\ hill^+(x_i, \theta_i, d, a, b) &= 1 - hill^+(x_i, \theta_i, d, a, b); \end{aligned}$$

$$\begin{aligned} s^+(x_i, e, \theta_i, a, b) &= a + (b-a)\frac{1 + tanh(e(x_i - \theta_i))}{2}; \\ s^-(x_i, e, \theta_i, a, b) &= 1 - s^+(x_i, e, \theta_i, a, b); \end{aligned}$$

$$\begin{aligned} h^+(x_i, \theta_i, a, b) &= a, \text{ ak } x_i < \theta_i;\ b \text{ inak}; \\ h^-(x_i, \theta_i, a, b) &= 1 - h^+(x_i, \theta_i, a, b); \end{aligned}$$

kde $hill^+, hill^-$ sú funkcie Hillovej kinetiky,
s^+, s^- sú sigmoidálne prepínače,
h^+, h^- sú Heavisideove (schodové) funkcie,
$\theta_i \in \mathbb{R}^+,\ \theta_i \leq max_i$,
$i \in \{1, \ldots, n\}$,
$a, b \in \mathbb{R}_0^+$,
$e, d \in \mathbb{R}^+$.

Špeciálnym prípadom je recipročná hodnota sigmoidálnej funkcie:

$$s^+(x_i, e, \theta_i, a, b)^{-1} = s^-(x_i, e, \theta_i + \frac{ln(\frac{a}{b})}{2e}, b^{-1}, a^{-1}),$$

ktorú označujeme ako $s^+inv(x_i, e, \theta_i, a, b)$. Potom klesajúcu recipročnú funkciu označíme obdobne ako doplnok rastúcej:

$$s^-inv(x_i, e, \theta_i, a, b) = 1 - s^+(x_i, e, \theta_i, a, b).$$

Dôkaz možno nájsť v článku *From cardiac cells to genetic regulatory network* na strane 6 [11].

Teraz už môžeme zadefinovať úplný formát nášho po častiach multi-afinného ODE modelu (ďalej len PMA model z *ang.* piece-wise multi-affine ODE model). PMA model $\mathcal{M}$ je daný ako $\dot{x} = f(x)$, kde x je stále vektor premenných $(x_1, \ldots, x_n)$, ale $f = (f_1, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ je tentokrát vektor po častiach multi-afinných funkcií. Nevyhnutnou súčasťou modelu $\mathcal{M}$ je množina prahových hodnôt (z *ang.* threshold) $\theta^i_m \in \mathbb{R}^+$ spĺňajúca $min_i = \theta^i_1 < \theta^i_2 < \ldots < \theta^i_{\eta_i} = max_i$, kde $i \in \{1, \ldots, n\}$, $m \in \{1, \ldots, \eta_i\}$ a platí, že $\eta_i \geq 2$.

Uvažujme Ω ako časť modelu $\mathcal{M}$ tak, že $\Omega = \prod_{i=1}^{n} \{1, \ldots, \eta_i - 1\}$. Funkcia $g : \mathbb{R}^n \to \mathbb{R}^+$ je vtedy po častiach multi-afinná, ak je multi-afinná na každom n-dimenzionálnom intervale $(\theta^1_{j_1}, \theta^1_{j_1+1}) \times \ldots \times (\theta^n_{j_n}, \theta^n_{j_n+1})$, kde $(j_1, \ldots, j_n) \in \Omega$ a zároveň $\forall i, 1 \leq i \leq n, j_i < max_i$. Potom dostávame n-dimenzionálny PMA model pozostávajúci z funkcií f v nasledujúcom tvare:

$$f_i(x) = \sum (s \prod_{j \in I} \rho^j_i(x)), \tag{2.1}$$

kde $i \in \{1, \ldots, n\}$, $s \in \{-1, 1\}$, I je neprázdna podmnožina konečnej množiny indexov všetkých použitých regulačných funkcií a ρ^j_i je príslušná regulačná funkcia, ktorá môže nadobúdať tieto hodnoty:

$$\rho(x_t) = \begin{cases} c, & c \in \mathbb{R} \\ x_k, & k \in \{1, \ldots, n\} \\ r^*(x_k, \theta^k_m, \theta^k_{m+1}, a', b'), & m \in \{1, \ldots, \eta_k - 1\}; a', b' \in \mathbb{R} \\ r^*coor(x_k, \theta^k_m, \theta^k_{m+1}, x_t, x'_t), & t \in \{1, \ldots, n\} \wedge t \neq i \\ s^*(x_k, e, \theta^k_m, a, b), & e \in \mathbb{R}^+;\ a, b \in \mathbb{R}^+_0 \\ s^*inv(x_k, e, \theta^k_m, a, b), & \\ hill^*(x_k, \theta^k_m, d, a, b), & d \in \mathbb{R}^+ \\ h^*(x_k, theta^k_m, a, b) & \end{cases}, \tag{2.2}$$

kde $* \in \{+, -\}$. Ak $\rho^{j_1}_i, \rho^{j_2}_i$ sú regulačné funkcie, ktoré patria do jednoho produktu $s \prod_{j \in I} \rho^j_i(x)$, tak že $j_1, j_2 \in I$, musí platiť $dep(\rho^{j_1}_i) \cap dep(\rho^{j_2}_i) = \emptyset$, kde $dep(\rho)$ je množina premenných x, na ktorých je ρ závyslá. Príklad PMA modelu možno nájsť na obrázku ...

Do modelu je priamo zavedená aj možnosť zadania konštantnej funkcie (v prípade, že $\rho(x_t) = c$) a lineárnej funkcie (v prípade $\rho(x_t) = x_k$). Ďalšou nevyhnutnou súčasťou modelu sú iniciálne podmienky, vyjadrujúce počiatočné koncentrácie jednotlivých látok. Tieto nie sú vyjadrené presne, ale

namiesto toho sú určené intervalom, ktorého hranice musia byť z množiny $\{\theta_1^i, \ldots, \theta_{\eta_i}^i\}$.

Takto vytvorený model je prakticky použiteľný pre ľubovolný biologický systém. [5][6]

2.1 Abstrakcia

A. Optimálna lineárna abstrakcia

V minulej časti sme objasnili, prečo je potrebné prevádzať regulačné funkcie $\rho(x_i)$, kde x_i je premenná na po častiach lineárne rampové funkcie. V tejto časti sa pokúsime objasniť princíp tohto prevodu. Vynára sa hneď niekoľko problémov:

1. Ako vhodne rozdeliť ľubovolnú krivku na zadaný počet ramp.
2. Ako vhodne rozdeliť niekoľko kriviek rovnakej dimenzie na zadaný počet ramp.
3. Ako vybrať správny počet ramp, tak aby stavový systém nebol príliš veľký a zároveň tak, aby abstrakcia nebola príliš hrubá.

Na prvú otázku dáva odpoveď algoritmus dynamického programovania vyvinutý pre počítačovú grafiku, konkrétne pre aproximáciu digitalizovaných polygonálnych kriviek s minimálnou chybou [22]. Tento bol upravený a rozšírený pre viacero kriviek naraz v článku *From Cardiac Cells to Genetic Regulatory Network* (Grosu et al.) [11] a to tak, že zmienená chyba sa počíta pre všetky krivky naraz. Aby to bolo možné, musia byť všetky krivky diskretizované v rovnakých x-ových bodoch. V našom prípade sa krivkami myslia Hillove funkcie (viď 1.4.C) a sigmoidálne funckie popísané, tiež v článku [11].

Účelom tejto abstrakcie je nájsť také body x_i, kde $i \in \{1, \ldots, n\}$ a n je počet diskretizačných bodov, ktoré budú sekať krivky na lineárne segmenty s minimálnou odchýlkou (chybou). Chyba sa počíta ako súčet druhých mocnín vzdialeností dvoch bodov na y-ovej ose. Týmito bodmi sú funkčná hodnota krivky a funkčná honota rampy pre konkrétny bod x_i na úseku ohraničenom bodmi x_a a x_b, kde $x_i \in \langle x_a, x_b \rangle$. Rampa je v tomto zmysle braná ako lineárna spojnica medzi funkčnými hodnotami týchto hraničných bodov. Ak sa nájde vyhovujúca hodnota chyby, potom možno s istotou tvrdiť, že body x_a a x_b sú zároveň hraničnými bodmi nového segmentu. So všetkými takto získanými bodmi už nie je problém previesť každú funkciu na

jej po častiach lineárnu formu. Zároveň treba tieto body uložiť ako nové prahové hodnoty modelu (viď kapitolu 2.B). Príklad abstrakcie možno vidieť na obrázku 2.1.

Odpoveď na tretiu otázku ohľadom správneho počtu rámp je takáto. Užívateľ musí nastaviť požadovaný počet sám, to mu však umožňuje, aby skúšal rôzne úrovne abstrakcie osobne. [11]

B. Obdĺžniková abstrakcia

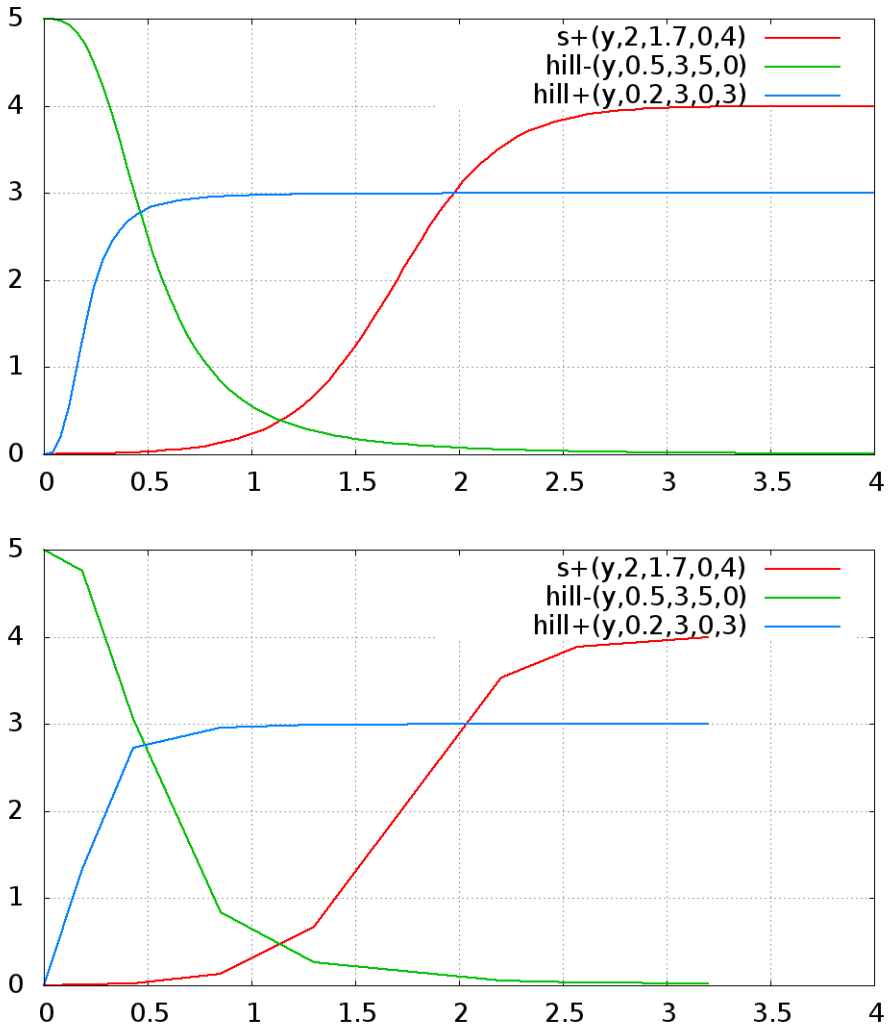
Výhodou multi-afinného systému je, že ho možno abstrahovať do podoby obdĺžnikového prechodového systému (z *ang.* Rectangular Abstraction Transition System, skrátene RATS) [2]. K tomu je potreba pôvodný n-dimenzionálny systém, kde n je počet premenných najskôr rozdeliť na menšie n-dimenzionálne oddiely. Každá premenná má priradenú množinu prahových hodnôt alebo thresholdov, ktoré vyjadrujú významné, či zaujímavé koncentračné hladiny. Každej premennej sú priradené aspoň dve takéto hodnoty a to minimálna a maximálna. Ďalšie môžu byť zadefinované vo vstupnom modeli, alebo môžu pochádzať z predchádzajúcej lineárnej abstrakcie.

Teraz už nie je problém rozdeliť multi-afinný systém podľa týchto prahov na menšie n-rozmerné ohraničené časti nazvané obdĺžniky. Tieto si môžeme predstaviť ako uzly grafu. Ak sú dva obdĺžniky susedmi, môže medzi nimi vzniknúť prechod v zmysle hrany medzi dvoma uzlami. Takto dostávame diskrétny prechodový systém (RATS) z pôvodne dynamického systému.

Prechody sú samozrejme orientované. Hodnotu orientácie dostaneme zo smerových vektorov vypočítaných vo všetkých rohoch n-rozmerného obdĺžnika. A to tak, že ak aspoň jeden vektor v rohoch spoločných so susediacim obdĺžnikom má smer k tomuto susedovi, pridáme odchádzajúcu hranu z tohto tohto uzla do susedného. Ak aspoň jeden vektor v rohoch spoločných so susediacim obdĺžnikom má smer od tohto suseda, pridáme prichádzajúcu hranu zo susediaceho uzla k tomuto uzlu. To znamená, že hrany môžu byť aj obojsmerné. Podmienkou je, že ak nastane situácia, kedy uzol nebude mať žiadnu odchádzajúcu hranu, musíme mu pridať slučku. Tento jav symbolizuje rovnovážny stav v daných podmienkach.

Je známy fakt, že obdĺžniková absrakcia je nadaproximáciou vzhľadom k trajektóriam pôvodného dynamického modelu. [4]

mozno dorobit obrazok podobne ako v [4] pomocou latex/picture



Obr. 2.1: Tri krivky s popisom. V prvom prípade bez použitia abstrakcie a v druhom prevedené na sústavu rámp po použití abstrakcie s parametrami: diskretizačných bodov 500, požadovaný počet segmentov 7.

Kapitola 3

Východiskový stav a podobné nástroje

3.1 BioDiVinE 1.0

Pôvodný nástroj BioDiVinE bol vyvinutý v laboratóriu systémovej biológie (SyBiLa)[24] v spolupráci s laboratóriom paralelných a distribuovaných systémov (ParaDiSe)[21]. Obe sídlom na fakulte informatiky Masarykovej univerzity.

BioDiVinE 1.0 je nástroj vytvorený pre verifikáciu vlastností biochemických modelov zadaných ako systém ODE (viď kapitolu 2). Je to nadstavba nástroja DiVinE [3], určeného na model checking (viď kapitolu 1.5). K tomuto účelu ponúka niekoľko odlišných algoritmov.

Vstupom je, ako už bolo povedané biochemický model v tzv. .bio formáte (podrobnosti v [25]), ktorý môže predstavovať genovú regulačnú sieť alebo vzájomnú interakciu proteínov, či iné. Dôležitou súčasťou je aj súbor s jednou alebo viacerými vlastnosťami zapísanými ako LTL formule (viď kapitolu 1.1). Tieto dva súbory sa musia skombinovať do jednoho použitím príslušného nástroja, pričom vlastnosť sa zároveň prevedie z LTL formule na Büchiho automat (viď kapitolu 1.2). Takto hotový vstup sa môže predať jednomu z overovacích algoritmov.

Je vidno, že postup nie je veľmi užívateľsky prívetivý. Užívateľ musí najskôr použiť jeden nástroj pre vytvorenie vstupného súboru, potom ho ručne premenovať, aby ho ďalší nástroj, tentokrát overovací správne rozpoznal. Výsledky v podobe protipríkladu alebo podrobný výpis priebehu procesu sú vygenerované do nových súborov .trail alebo .report po zadaní príslušného vstupného prepínača vybraného algoritmu. Jedinou výhodou je, že väčšina prepínačov je unifikovaná, pretože nástroje boli vytvorené jedným vývojárskym tímom.

Celý nástroj BioDiVinE pritom obsahuje zbytočne veľa ďalších rozšírení pre iné typy modelov, pretože pôvodný nástroj DiVinE bol prevzatý ako celok aj s týmito predchádzajúcimi rozšíreniami.

Kvôli vyššie spomenutým dôvodom sme si dali za cieľ okrem rozšírenia vstupného modelu a abstrakcie aj refaktorizáciu starého nástroja. [4]

3.2 PEPMC

Nástroj vyvinutý v laboratóriu systémovej biológie (SYBILA)[24] pre overovanie modelov s ohľadom na odhad parametrov (z *ang.* Parameter Estimation by Parallel Model Checking, skrátene PEPMC). Funguje na princípe paralelného overovania modelov alebo tiež farebného overovania modelov (viď kapitolu 1.5.2). Modelom je systém ODE podobne ako v nástroji BioDiVinE 1.0, ale rozšírený o možnosť zadávania po častiach lineárnych ramp funkcií (viď kapitolu 2). Dôležitým rozdielom oproti BioDiVinE-u a súčasne hlavnou podstatou nástroja PEPMC však je parametrizácia modelu a s tým súvisiace prehľadávanie parametrického priestoru.

Existuje myšlienka zlúčiť tieto dva nástroje, bráni tomu však mierne odlišný vstup, datový model a samozrejme rozdielne jadrá programov. Nebudeme tu však tento problém ďalej rozoberať, pretože nie je súčasťou našej témy. [6]

3.3 RoVerGeNe

RoVerGeNe je podobne ako PEPMC nástroj pre analýzu genetických regulačných sietí s neurčitými parametrami. Model siete je zapísaný ako po častiach multi-afinný systém diferenciálnych rovníc. Vlastnosti, proti ktorým sa model overuje, majú formu LTL formule a prametre sú zadané ako intervaly. Nástroj sa snaží overiť, či model spĺňa danú vlastnosť pre všetky parametre.

Na rozdiel od PEPMC však RoVerGeNe dokáže pracovať aj s lineárnymi kombináciami parametrov. To v praxi znamená, že obdĺžniková abstrakcia (viď 2.1.B) u neho nie je použiteľná, pretože vyžaduje, aby n-dimenzionálne štvoruholníky s ktorými pracuje, boli ortogonálne, čiže obdĺžniky. Namiesto toho používa tento nástroj štandardné polyhedrálne operácie balíka MPT [20] a celý je naprogramovaný v Matlabe. [8] [7]

Kapitola 4

BioDiVinE 1.1

V kapitole 3.1 boli predstavené niektoré nedostatky pôvodného nástroja BioDiVinE 1.0, ktoré sme si dali za cieľ odstrániť. Súčasne bol záujem o rozšírenie jeho funkcionality a odstránenie nadbytočných častí kódu.

Prvotný impulz spočíval v zovšeobecnení vstupného modelu, aby poskytoval širší záber možností. Aby nebol obmedzený iba na mass action kinetiku (viď 1.3) a s tým súvisiace nelineárne diferenciálne rovnice. Do úvahy prichádzali samozrejme najbežnejšie používané Michaelis-Mentenovej kinetika a Hillova kinetika (viď 1.4), ale aj Heavisidove funkcie [27], sigmoidálne funkcie z konferencie CAV 2011 [11] a obecné rampové funkcie (viď 2.B), ktoré slúžia aj ako produkt optimálnej lineárnej abstrakcie (viď 2.1.A).

Tým sme nakúsli ďalšiu novú funkcionalitu. Zlúčenie nového rozšíreného vstupného modelu (viď 2) a pôvodne používanej obdĺžnikovej abstrakcie (viď 2.1.B). Prechodom medzi tým je práve vyššie spomenutá optimálna lineárna abstrakcia, ktorá prevedie novozavedené nemulti-afinné funkcie na po častiach multi-afinné, presne ako je potrebné pre obdĺžnikovú abstrakciu.

Pri zápise jednotlivých rovníc modelu sme sa chceli vyhnúť operácii delenia, preto sme sa rozhodli všetky pridávané funkcie zapisovať v podstate unifikovaným spôsobom, ktorý jasne určuje o aký typ funkcie sa jedná, či je rastúca alebo klesajúca a ďalej parametre nevyhnutné pre jej vyčíslenie. Tak isto sme pridali pár noviniek ako definovanie pomenovaných konštánt a voliteľné, užívateľom zadávané parametre pre lineárnu abstrakciu. Konkrétne ide o počet bodov, v ktorých majú byť funkcie evaluované a o počet segmentov, na ktoré majú byť funkcie rozdelené a linearizované.

Z týchto dôvodov sme sa rozhodli vytvoriť úplne nový parser a súčasne s ním aj nový datový model pre uchovanie nových dát. Niektoré prvky boli síce zhodné so starým BioDiVinE-om, ale spôsob akým boli naimplementované v starej verzii, bol veľmi zle rozšíriteľný. Začať v týchto bodoch od znova nám prišlo jednoduchšie a prehľadnejšie.

Avšak kód starého nástroja bol hodne rozsiahly a skladal sa z veľkého

množstva rôznych tried, z ktorých mnohé už pre naše účely neboli potrebné. Preto bolo nutné najskôr celý kód postupne prechádzať a debuggovať aby sme mu lepšie porozumeli. Na základe toho sme boli schopný eliminovať neužitočné a nepoužívané časti kódu. Odstránili sme tak počas fáze refactoringu asi polovicu tried. /* treba ich aj vymenovat? */

Samotné pripojenie našich nových tried pre parser a datový model, ale nebolo triviálne. K tomu všetkému sme potrebovali zachovať časť starého parseru, ktorá mala za úlohu načítať Büchiho automat (viď 1.2) so skúmanou vlastnosťou.

Overovacie algoritmy boli naštastie napísané dostatočne obecne, takže po úspešnom zavedení nových tried do starých sa nevyskytlo veľa problémov. Najdôležitejšie pre ucelenie celého nástroja však bolo vytvorenie konzolového užívateľského prostredia (ďalej len CLI), ktoré do jednoho príkazu ukryje to, čo by inak bolo nutné spraviť v troch krokoch (viď kapitolu 3.1). Okrem toho automaticky overí všetky vlastnosti zadané vo vstupnom súbore s týmito vlastnosťami a následne zmaže dočasné súbory.

Vstupom pre CLI je názov vybraného overovacieho algoritmu (owcty, distr_map, owcty_reversed), voliteľné prepínače a dva súbory. Jedným je cesta k súboru s modelom s príponou .bio a druhým je cesta k súboru s jednou alebo viacerými vlastnosťami zapísanými ako LTL formule (viď 1.1) s príponou .ltl.

Bohužial sme boli donútený odstrániť GUI z pôvodného nástroja, ktoré slúžilo na vytváranie a ukladanie modelov a vizualizáciu výsledkov, pretože sa ukázalo nezlučiteľné s novými rozšíreniami a bolo by nutné vytvoriť úplne novú verziu. Táto činnosť je ponechaná pre budúce možnosti rozšírenia.

Pôvodný BioDiVinE disponoval aj možnosťou paralelne distribuovaného spracovávania prostredníctvom protokolu MPI (z *ang.* Message Passing Interface) [23]. Táto možnosť zostala zachovaná v rámci všetkých pôvodných častí zdrojového kódu.

Kvôli lepšiemu a celistvejšiemu prehľadu všetkého vyššie zmieneného ponúkame pohľad na architektúru nástroja BioDiVinE 1.1 v diagrame 4, kde je červenou farbou zvýraznená oblasť veľkej časti našej práce.

4.1 Vstupný súbor s modelom

Hlavnou náplňou tejto podkapitoly je zoznámiť bližšie užívateľa so syntaxou vstupného modelu. Tým sa myslí súbor s príponou .bio a vyzerá nasledovne:

- VARS : *premenná1*, *premenná2*, ...
- CONSTS : *konštanta1*, *hodnota1*; *konštanta2*, *hodnota2* ; ...
- EQ : *premenná1* = *rovnica1*
- EQ : *premenná2* = *rovnica2*
- THRES : *premenná1*: *prah1*, *prah2*, *prah3*, ...
- THRES : *premenná2*: *prah1*, *prah2*, ...
- VAR_POINTS : *premenná1*: *počet_bodov*, *počet_segmentov*; *premenná2*: *počet_bodov*, *počet_segmentov*
- INIT : *premenná1*: *od*, *do*; *premenná2*: *od*, *do*
- system async;

Niekoľko vysvetliviek a obmedzení:

1. Všetko, čo je písané neskoseným písmom, musí zostať zachované. Naopak to, čo je napísané skoseným písmom, treba nahradiť vlastnými hodnotami.
2. Biele znaky sú kompletne ignorované, dôležité sú iba oddeľovače vyznačené hrubým písmom (: ; , =). Na koncoch riadkov sa nikdy oddeľovač nesmie nachádzať s výnimkou posledného riadku (system async;).
3. Názvy premenných a konštánt musia ako prvý znak obsahovať písmeno nasledované ľubovolnou kombináciou písmen, čísel a znakov _ ~ { }.
4. Čísla majú tvar celých alebo desatinných čísel s desatinnou bodkou.
5. Riadok (CONSTS) nie je povinný. Je tiež možné používať iba nepomenované konštanty (čiže čísla).
6. Riadky (VARS) a (CONSTS) sa musia nachádzať pred ostatnými riadkami – sú úvodné.
7. Riadok (VAR_POINTS) nie je povinný a *počet_bodov* znamená počet bodov, v ktorých budú evaluované regulačné funkcie pre danú premennú, a *počet_segmentov* určuje počet rámp, na ktoré sa tieto funkcie prevedú.

8. Hodnoty *od* a *do* v riadku (INIT) musia byť niektoré z hodnôt v príslušnom riadku (THRES) pre danú premennú, pre ktoré musí platiť $od < do$.

9. Každá premenná ma povinný vlastný riadok (EQ) a (THRES) a musí sa nachádzať aj v riadku (INIT) s príslušnými hodnotami.

10. Riadok (system async;) je povinný a musí sa nachádzať na konci súboru z historických dôvodov.

11. Rovnica v riadku (EQ) je matematicky popísaná v kapitole 2.B konkrétne v zápise 2.1 a v bežnom jazyku nižšie:

    - Parser podporuje unárnu operáciu $-$, binárne operácie $+$, $-$ a $*$ a guľaté zátvorky, ktoré môžu byť vnorené ľubovolne veľa krát.
    - Rovnica môže okrem operácií a zátvoriek obsahovať pomenované aj nepomenované konštanty, premenné a nasledujúce regulačné funkcie (viď zápis 2.2):
      - rpcoor(premenná, prah1, prah2, hodnota1, hodnota2), rmcoor(premenná, prah1, prah2, hodnota1, hodnota2),
      - rp(premenná, prah1, prah2, parameter1, parameter2), rm(premenná, prah1, prah2, parameter1, parameter2),
      - hp(premenná, prah, parameter1, parameter2), hm(premenná, prah, parameter1, parameter2),
      - sp(premenná, faktor, prah, parameter1, parameter2), sm(premenná, faktor, prah, parameter1, parameter2),
      - spinv(premenná, faktor, prah, parameter1, parameter2), sminv(premenná, faktor, prah, parameter1, parameter2),
      - hillp(premenná, prah, faktor, parameter1, parameter2), hillm(premenná, prah, faktor, parameter1, parameter2),
    - Úvodné znaky (r, h, s) môžu byť zadané aj ako veľké písmená.
    - Znak (p) resp. (m) znamená (+) resp. (–) v zmysle rastúcej resp. klesajúcej funkcie.
    - Premenná musí byť jedna zo skôr uvedených premenných v riadku (VARS).
    - Hodnoty (prah, faktor, hodnota, parameter) môžu byť zadané ako pomenované alebo nepomenované konštanty.
    - Prah musí byť jednou z prahových hodnôt premennej.

- **Nie je možné násobiť spolu rovnaké premenné ani premennú s regulačnými funkciami závyslími od tejto premennej ani samotné regulačné funkcie závyslé na rovnakej premennej.** Bolo by to porušenie pravidla o stupni premennej z kapitoly 2.A.

4.2 Vstupný súbor s vlastnosťou

Súbor s vlastnosťou alebo vlastnosťami je detailne popísaný v práci [12]. Tu si iba v krátkosti zhrnieme syntax. Súbor s príponou .ltl môže obsahovať nasledujúce riadky:

- #define *názov_premennej hodnota_premennej*
- #property *vlastnosť_vo_forme_ltl_formule*

a platia pre ne nasledujúce pravidlá:

1. Súbor môže obsahovať ľubovolný počet definícií (riadok #define), mali by sa však odlišovať v názve premennej. Ten môže obsahovať malé písmená, čísla a znak _. Začínať ale musia buď písmenom alebo znakom _.

2. Hodnota premennej je v podstate výraz s logickou hodnotou (čiže atomická propozícia, AP (viď kapitolu 1.1)). Tá môže mať buď priamo hodnotu *true* alebo *false* alebo môže byť tvorená reťazcom obsahujúcim meno nejakej premennej z modelu, relačný operátor (<, >, <=, >=) a číselnú hodnotu (celú alebo reálnu), ktorá sa musí nachádzať medzi prahovými hodnotami premennej z modelu.

3. Počet riadkov (#property) tiež nie je obmedzený, ale užívateľ by mal vedieť, že každý takýto riadok znamená novú vlastnosť, ktorá sa bude testovať na modeli a to znamená výpočtovú réžiu.

4. Samotná formula *vlastnosť_vo_forme_ltl_formule* môže byť tvorená nasledujúcimi hodnotami:

   - Unárnymi temporálnymi operátormi (**X**ϕ, **F**ϕ, **G**ϕ) a binárnymi temporálnymi operátormi (ϕ**R**ψ, ϕ**U**ψ)(viď kapitolu 1.1), kde ϕ a ψ sú výrazy s logickou hodnotou.
   - Výrazmi s logickou hodnotou, ktoré sú ďalej tvorené logickými operátormi (**!** – negácia, **&&** – konjunkcia, **||** – disjunkcia, **->** – implikácia, **<->** – ekvivalencia) a premennými definovanými na začiatku súboru.

- Syntax podporuje aj ľubovolne vnorené guľaté zátvorky.

5. Pre úplnosť treba dodať, že táto syntax podporuje aj pár ďalsích, rozšírených operátorov a takmer ku každému ešte druhý možný zápis, ale náš výpočet je dostatočný pre zápis ľubovolnej LTL formule.

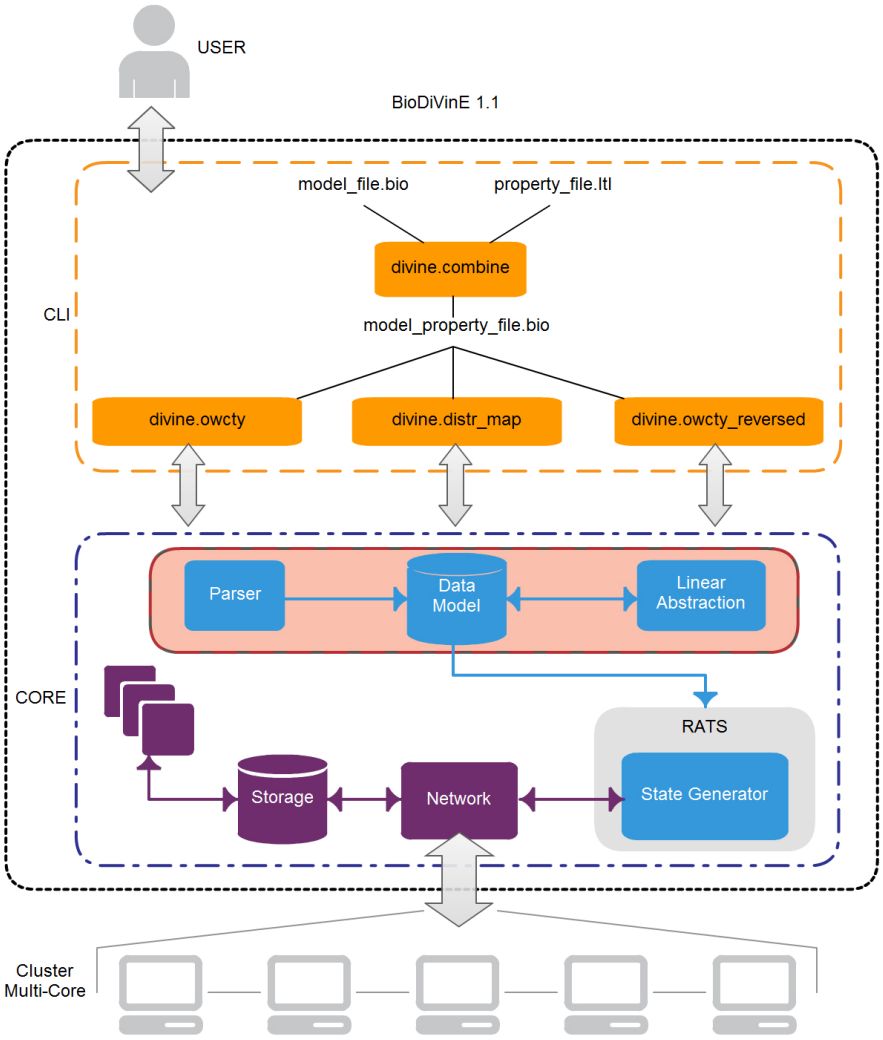
4.3 Priebeh procesu

Po zavolaní CLI spolu so zadanými vstupnými parametrami sa najskôr skontroluje ich počet. Ak je tento menší, program vypíše *help* ponuku /*odkaz na obrazok s nou*/. V opačnom prípade skontroluje existenciu vstupných súborov a predá ich pomocnému nástroju *divine.combine*. Ten zistí počet vlastností a pre každú z nich vytvorí nový súbor a zároveň prevedie vlastnosť z LTL formule na Büchiho automat. Tento výstupný súbor má tiež koncovku .bio, ale je iba pomocný a pred ukončením CLI bude zmazaný. Ďalej sa zistí požadovaný overovací algoritmus. Môže ísť o algoritmy *owcty*, *distr_map* alebo *owcty_reversed*. Žiadne iné nie sú prípustné.

Vybraný algoritmus sa spustí pre každý novovytvorený súbor .bio ako nástroj *divine.názov_algoritmu* a predajú sa mu prípadné parametre. Potom sa vstup predá triede s parserom a vytvorí sa datový model, ktorý prejde lineárnou aproximáciou, ak je to potrebné. Tým sa myslí prípad, keď rovnice modelu obsahujú aspoň jednu Hillovu funkciu alebo aspoň jednu sigmoidálnu funkciu. Heavisidove funkcie sa upravujú zvlášť hneď po načítaní, pretože ich možno previesť na jedinú rampu a teda lineárna aproximácia pre ne nemá zmysel.

Prahové hodnoty vygenerované aproximáciou aj prevodom Heavisidových funkcií sa uložia medzi ostatné. To na jednu stranu zvyšuje výpočtovú réžiu, pretože sa zväčšuje stavový priestor generovaný nasledujúcou obdĺžnikovou aproximáciou. A to lineárne k počtu nových prahových hodnôt. No na strane druhej sa tým zlepšuje presnosť tejto aproximácie a celého výpočtu, pretože sa abstrahovaný model približuje skutočnému spojitému modelu.

Už spomenutá obdĺžniková aproximácia vygeneruje RATS (viď 2.1.B), čiže diskrétny systém pôvodne spojitého modelu, ktorý sa overuje proti vlastnosti v podobe automatu. Ak ju RATS spĺňa, potom možno s istotou tvrdiť, že aj pôvodný model túto vlastnosť spĺňa. Naopak, ak je vygenerovaný protipríklad, tento vôbec nemusí zodpovedať protipríkladu v skutočnom spojitom modeli. Avšak stále platí, že model, či už abstrahovaný alebo skutočný danú vlastnosť nespĺňa. [4] /*dalo by sa este spomenut, ze owcty a owcty_reversed dokazu skumat aj samostatne reachabilitu*/



Obr. 4.1: Diagram architektúry nástroja BioDiVinE 1.1. V červenej oblasti je zhrnutý náš prínos

Kapitola 5

Implementácia

/*asi by sa hodilo napisat v com som pisal parser a scanner a jake su tam subory, dalej popisat triedy datoveho modelu a CLI*/

5.1 Parser

Na úvod je treba povedať, že za výrazom parser sa ukrýva aj druhý pomocný nástroj tzv. scanner. Samotný parser je nástroj na syntaktickú analýzu vstupu a v niektorých jednoduchých príkladoch môže byť dostatočný, ale nie v našom. Aby bolo parsovanie úspešné musí mu predchádzať ešte lexikálna analýza a tú má na starosti práve scanner, ktorý zjednodušene povedané načítava "slová". Tie potom posiela parseru, ktorý z nich skladá "vety", respektíve vzory. Akonáhle je nejaký vzor rozpoznaný, inštrukcie parseru povedia, akú akciu má s dátami vykonať.

Pre vytvorenie parseru a scanneru sú k dispozícii sesterské programy. Najstaršie z nich sú *flex* a *bison*. Tie sa však ukázali byť až príliš zastarané a neschopné vytvoriť jednoduchú štruktúru typu trieda, ktorá by sa dala ľahko začleniť do už existujúceho kódu. Ďalšou možnosťou, ktorú sme zvažovali, boli programy *flex*++ a *bison*++. Tie údajne podporujú triedy, ale nájsť k nim použiteľný užívateľský manuál sa ukázalo byť nemožné. Naopak programy *flexc*++ [10] a *bisonc*++ [9] sú prehľadne zdokumentované a veľmi dobre začleniteľné do zdrojového kódu v jazyku c++, preto sa ukázali byť tou najlepšou voľbou. /*mozno opisat este detailnejsie subory ktorych a to tyka a co priblizne obsahuju*/

5.2 Dátový model

5.3 Konzolové užívateľské prostredie CLI

Kapitola 6

Použitie programu

Kapitola 7

Case study

Kapitola 8

Záver

Literatúra

[1] J. Barnat, L. Brim, A. Krejčí, D. Šafránek, A. Streck, M. Vejnár, and T. Vejpustek. On parameter synthesis by parallel model checking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(3):693–705. IEEE Computer Society Press, May-June 2012.

[2] J. Barnat, L. Brim, I. Černá, S. Dražan, J. Fabriková, and D. Šafránek. On algorithmic analysis of transcriptional regulation by LTL model checking. *Theoretical Computer Science*, 410(33-34):3128–3148, 2009.

[3] J. Barnat, L. Brim, I. Černá, and P. Šimeček. DiVinE – The Distributed Verification Environment. In *Proceedings of 4th International Workshop on Parallel and Distributed Methods in verifiCation*, pages 89–94, July 2005.

[4] J. Barnat, L. Brim, I. Černá, S. Dražan, J. Fabriková, J. Láník, H. Ma, and D. Šafránek. Biodivine: A framework for parallel analysis of biological models. *In Proceedings of 2nd International Workshop on Computational Models for Cell Processes (COMPMOD 2009)*, pages 31–45. EPTCS 6, 2009.

[5] J. Barnat, L. Brim, I. Černá, S. Dražan, J. Fabriková, and D. Šafránek. Computational analysis of large-scale multi-affine ode models. *In Proceedings of High performance computational systems Biology (HiBi 2009)*, pages 81–99. IEEE Computer Society Press, 2009.

[6] J. Barnat, L. Brim, D. Šafránek, and M. Vejnár. Parameter scanning by parallel model checking with applications in systems biology. *In Proceedings of High performance computational systems Biology (HiBi 2010)*, pages 95–104. IEEE Computer Society Press, 2010.

[7] G. Batt and C. Belta. RoVerGeNe: A tool for the Robust Verification of Gene Networks [online], [cit. 2014-05-17]. Dostupné na: <`http://iasi.bu.edu/~batt/rovergene/rovergene.htm`>.

[8] G. Batt, C. Belta, and R. Weiss. Model checking liveness properties of genetic regulatory networks. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *Lecture Notes in Computer Science*, pages 323–338. Springer, 2007.

[9] F.B. Brokken. Bisonc++ (Version 4.09.01) User Guide [online]. Center for Information Technology, University of Groningen, The Netherlands, [cit. 2014-05-21]. Dostupné na: <http://bisoncpp.sourceforge.net/bisonc++.html>.

[10] F.B. Brokken, J.P. van Oosten, and R. Berendsen (until 0.5.3). Flexc++ (Version 2.01.00) User Guide [online]. University of Groningen, The Netherlands, [cit. 2014-05-21]. Dostupné na: <http://flexcpp.sourceforge.net/flexc++.html>.

[11] R. Grosu, G. Batt, F.H. Fenton, J. Glimm, C. Le Guernic, S.A. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. *In Proceedings of CAV'11, the 23rd International Conference on Computer Aided Verification, Lecture Notes in Computer Science*, 6806:396–411. Springer, July 2011.

[12] M. Jaroš. Převod formulí ltl na automaty [online]. Diplomová práca, Fakulta informatiky Masarykovej univerzity, Brno, Česká republika, 2004 [cit. 2014-05-19]. Dostupné na : <http://is.muni.cz/th/4017/fi_m/>.

[13] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. *Model Checking*, chapter 3. Temporal Logics, pages 27–33. MIT Press, 1999.

[14] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. *Model Checking*, chapter 9. Model Checking and Automata Theory, pages 121–140. MIT Press, 1999.

[15] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. *Model Checking*, chapter 1.3. The Process of Model Checking, pages 4–4. MIT Press, 1999.

[16] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. *Model Checking*, chapter 9.4. Translating LTL into Automata, pages 132–138. MIT Press, 1999.

[17] J.P. Keener and J. Sneyd. *Mathematical Physiology*, volume 8 of *Interdisciplinary Applied Mathematics*, chapter 1.1. The Law of Mass Action, pages 3–4. Springer-Verlag, 1998.

[18] J.P. Keener and J. Sneyd. *Mathematical Physiology*, volume 8 of *Interdisciplinary Applied Mathematics*, chapter 1.2. Enzyme kinetics, pages 5–16. Springer-Verlag, 1998.

[19] J.P. Keener and J. Sneyd. *Mathematical Physiology*, volume 8 of *Interdisciplinary Applied Mathematics*, chapter 1.4. Appendix: Math Background, pages 24–30. Springer-Verlag, 1998.

[20] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT) [online], [cit. 2014-05-17]. Dostupné na: <http://control.ee.ethz.ch/~mpt/>.

[21] Laboratórium paralelných a distribuovaných systémov [online]. Fakulta informatiky Masarykovej univerzity, Brno, Česká republika, [cit. 2014-05-17]. Dostupné na: <http://paradise.fi.muni.cz/index.html>.

[22] J.C. Perez and E. Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15(8):743–750. Elsevier Science Inc., August 1994.

[23] M. Snir, S.W. Otto, D.W. Walker, J. Dongarra, and S. Huss-Lederman. *MPI: The Complete Reference*. MIT Press, 1995.

[24] Laboratórium systémovej biológie [online]. Fakulta informatiky Masarykovej univerzity, Brno, Česká republika, [cit. 2014-05-17]. Dostupné na: <http://sybila.fi.muni.cz/home>.

[25] Nástroj BioDiVinE 1.0 [online]. Fakulta informatiky Masarykovej univerzity, Brno, Česká republika, [cit. 2014-05-17]. Dostupné na: <http://sybila.fi.muni.cz/tools/biodivine/v1/index>.

[26] G. Teschl. *Ordinary Differential Equations and Dynamical Systems*, volume 140 of *Graduate Studies in Mathematics*, chapter 1. Introduction, pages 3–31. American Mathematical Society, 2012.

[27] E.W. Weisstein. Heaviside Step Function, From MathWorld–A Wolfram Web Resource [online], [cit. 2014-05-17]. Dostupné na: <http://mathworld.wolfram.com/HeavisideStepFunction.html>.

Príloha