PA054: Formální modely v systémové biologii

David Šafránek

16.4.2010

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Pravděpodobnostní model checking CTMC

Uvažme množinu AP atomických propozic.

Budeme uvažovat CTMC definovaný jako trojici $M = \langle S, Q, L \rangle$ kde:

- S je konečná množina stavů
- $Q : S \times S \to \mathbb{R}_0^+$ je generující matice
- $L : S \to 2^{AP}$ je interpretační funkce

Pravděpodobnostní model checking CTMC

- pro každý stav $s \in S$ značíme $E(s) = \sum_{s' \in S} Q_{ss'}$
  $\Rightarrow$ průměrná čekací doba v s je $\frac{1}{E(s)}$
- stav $s \in S$ je *absorbující* pokud $E(s) = 0$
- pravděpodobnost přechodu $s \rightarrow s'$ značíme $P_{ss'}$:

$$P_{ss'} = \begin{cases} \frac{Q_{ss'}}{E(s)}, & E(s) > 0 \\ 0, & \textit{jinak} \end{cases}$$

- *běh* π v CTMC M budeme uvažovat jako konečnou nebo nekonečnou sekvenci $s_0, t_0, s_1, t_1, \ldots$, kde $s_i \in S$ a $t_i \in \mathbb{R}^+$ t.ž. $\forall i.\ Q_{s_i s_{i+1}} > 0$; zapisujeme:

$$\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots$$

- pro konečné běhy $\pi = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots \xrightarrow{t_{l-1}} s_l$ navíc předpokládáme s_l absorbující

Pravděpodobnostní model checking CTMC

- pro běh π v CTMC M ($\pi \in M$) značíme:
  - $\pi(i)$ ... $i$tý stav běhu, $\pi(i) = s_i$ pro $i \geq 0$
  - $\delta(\pi, i) = t_i$ ... čekací dobu ve stavu s_i
  - $\pi(t)$ ... stav v čase t:

$$\pi(t) = \begin{cases} s_0 & \text{pokud } t < t_0, \\ \pi(i) & \text{pokud } t \geq t_0 \text{ a } i \text{ nejmenší index t.ž. } t \leq \sum_{0 \leq j \leq i} t_j \end{cases}$$

  - pro $\pi = s_0, ..., s_l$ uvažujeme $\delta(\pi, l) = \infty$ a $\pi(t) = s_l$ pro všechna $t > t_1 + \cdots + t_{l-1}$
- $Path(s) = \{\pi \in M | \pi(0) = s\}$ (metrický prostor)

Pravděpodobnostní model checking CTMC

Borelovské pole

- uvažujme $s_0, \ldots, s_k \in S$ t.ž. $Q_{s_i s_{i+1}} > 0$, $(0 \leq i < k)$
- $I_0, \ldots, I_{k-1}$ neprázdné (otevřené) intervaly v $\mathbb{R}_0^+$
- $C(s_0, I_0, \ldots, I_{k-1}, s_k)$ značíme množinu jevů zahrnující všechny běhy $\pi \in Path(s_0)$ t.ž. $\pi(i) = s_i$ (pro $i \leq k$) a $\delta(\pi, i) \in I_i$ (pro $i < k$)
- $\mathcal{F}(Path(s))$ značíme nejmenší σ-algebru na množině $Path(s)$ zahrnující:
  - každou sekvenci $s_0, s_1, \ldots, s_k$ t.ž. $s = s_0$ a $Q_{s_i s_{i+1}} > 0$ (pro $0 \leq i < k$)
  - každou sekvenci $I_0, I_1, \ldots, I_{k-1}$ neprázdných intervalů v $\mathbb{R}_0^+$
- pro $\mathcal{F}(Path(s))$ definujeme pravděpodobnostní metriku induktivně:

$$
\begin{aligned}
k = 0: \quad & \Pr(C(s_0)) = 1 \\
k > 0: \quad & \Pr(C(s_0, I_0, \ldots, I_{k-1}, s_k, I', s') = \\
& = \Pr(C(s_0, I_0, \ldots, I_{k-1}, s_k)) \cdot P(s_k, s') \cdot (e^{-E(s_k)\cdot a} - e^{-E(s_k)\cdot b})
\end{aligned}
$$

kde $a = \inf I'$, $b = \sup I'$, pro $b = \infty$ uvažujeme $e^{-\lambda\cdot\infty} = 0$

Předpokládejme AP množ. atom. propozic, $p \in \langle 0, 1 \rangle$, $\bowtie \in \{\leq, \geq, <, >\}$.

Formule ϕ je *stavovovou formulí CSL* pokud:

- $\phi = a$, pro lib. $a \in AP$
- jsou-li ϕ_1 a ϕ_2 stavové formule CSL, pak:
  - $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ jsou stavové formule CSL
  - $\mathcal{S}_{\bowtie p}(\phi_1)$ je stavová formule CSL
- je-li φ běhová formule CSL, pak:
  - $\mathcal{P}_{\bowtie p}(\varphi)$ je stavovou formulí CSL

Formule φ je pro lib. $t \in \mathbb{R}_0^+$ *běhovou formulí CSL* pokud:

- jsou-li ϕ_1 a ϕ_2 stavové formule CSL, pak:
  - $\mathbf{X}\phi_1$, $\phi_1\mathbf{U}\phi_2$, $\phi_1\mathbf{U}^{\leq t}\phi_2$ jsou běhové formule CSL

Uvažujme CTMC $M = \langle S, Q, L \rangle$ nad množinou atomických propozic AP.

Pro $S' \subseteq S$ označme:

$$\pi_{S'}(s) = \lim_{t \to \infty} \Pr\{\pi \in Path(s) | \pi(t) \in S'\}$$

Platí $\pi_{S'}(s) = \sum_{s' \in S'} \pi_{\{s'\}}(s)$, definitoricky nechť $\pi_\emptyset = 0$.

Dále označme $Prob(s, \varphi) = \Pr\{\pi \in Path(s) | \pi \models \varphi\}$.

Relace splnitelnosti $\models$ pro stavové formule:

- $s \models a$, pokud $a \in L(s)$
- $s \models \neg\phi$, pokud $s \not\models \phi$
- $s \models \phi_1 \wedge \phi_2$, pokud $s \models \phi_1$ a $s \models \phi_2$
- $s \models \mathcal{S}_{\bowtie p}(\phi)$, pokud $\pi_{Sat(\phi)}(s) \in I_{\bowtie p}$
- $s \models \mathcal{P}_{\bowtie p}(\varphi)$, pokud $Prob(s, \varphi) \in I_{\bowtie p}$

Značení:

- $I_{\bowtie p} = \{q \in \langle 0, 1\rangle | q \bowtie p\}$
- $Sat(\phi) = \{s \in S | s \models \phi\}$

Relace splnitelnosti $\models$ pro běhové formule:

- $\pi \models \mathbf{X}\phi$, pokud $\pi(1) \models \phi$
- $\phi_1 \mathbf{U} \phi_2$, pokud $\exists k \geq 0.\, \pi(k) \models \phi_2 \land \forall 0 \leq i < k.\, \pi(i) \models \phi_1$
- $\phi_1 \mathbf{U}^{\leq t} \phi_2$, pokud
  $\exists x \in \langle 0, t \rangle.\, \pi(x) \models \phi_2 \land \forall y \in \langle 0, x).\, \pi(y) \models \phi_1$

- $\mathcal{P}_{\bowtie p}(\mathbf{F}^{\leq t}\phi) = \mathcal{P}_{\bowtie p}(\mathrm{tt}\,\mathbf{U}^{\leq t}\phi)$
- $\mathcal{P}_{\geq p}(\mathbf{G}\phi) = \mathcal{P}_{\leq 1-p}(\mathbf{F}\neg\phi)$

- `http://www.prismmodelchecker.org/`
- pravděpodobnostní model checking pro DTMC, CTMC
- podporuje logiky LTL, PCTL a CSL
- transientní a stabilní analýza:
  - formule tvaru $\mathcal{P}_{=?}(\varphi)$ a $\mathcal{S}_{=?}(\phi)$
  - formule tvaru $\mathcal{P}_{min=?}(\varphi)$ a $\mathcal{P}_{max=?}(\varphi)$
  - enumerace pravděpodobnosti
- symbolické algoritmy (vychází z BDD)
- výpočetně náročné (zejména pro CTMC)

- jazyk Reactive Modules
- podpora datových typů (int, double, ...)
- lokální, globální proměnné
- paralelní moduly
- pojmenované a strážené příkazy (přechody):

```
[label] condition -> rate:(assignement);
```

- (binární) synchronizace shodně pojmenovaných příkazů
  ⇒ typicky se uvažuje rate jen pro jednu ze synchronizovaných akcí, druhá 1
- podpora výrazů (expressions)
  ⇒ lze definovat rate závislý na daném stavu

```
// Example
// N-place queue + server

ctmc

const int N = 10;
const double mu = 1/10;
const double lambda = 1/2;
const double gamma = 1/3;

module queue
     q : [0..N];

     [] q<N -> mu:(q'=q+1);
     [] q=N -> mu:(q'=q);
     [serve] q>0 -> lambda:(q'=q-1);
endmodule

module server
     s : [0..1];

     [serve] s=0 -> 1:(s'=1);
     [] s=1 -> gamma:(s'=0);
endmodule
```