Formalización de la matemática y ciencias de la computación en asistentes de pruebas

Propuesta: Formalización de Redes de Petri

Bernardo Dávalos

1 de noviembre de 2023

Resumen

Este proyecto busca establecer definiciones, enunciar lemas/proposiciones/teoremas y probar dichos enunciados sobre redes de Petri reversibles usando el asistente de pruebas **Lean**, la cual pretende ser pionera en la comunidad de Lean y con el objetivo de ser un atajo al momento de investigar en este tema.

1. Preliminares

Las redes de Petri son un sistema importante teórico para estudiar los estados de la computación, que puede depender de cómo se mire su uso. Podríamos restringir que estas redes sean de ocurrencia para estudiar las propiedades de los conflictos y reversibilidades [1, p. 6-18], estas definiciones deberían poder llevarse a cabo ya que en [2] se define como conjuntos, y la librería `mathlib` otorgaría un atajo a este planteamiento.

2. Definiciones

Se define el conjunto de plazas $\mathcal{P}$ de tipo $Set\ \alpha$ y el conjunto de transiciones $\mathcal{T}$ de tipo $Set\ \beta$. El conjunto de estados es el conjunto potencia $\mathcal{S} = 2^{\mathcal{P}}$ de tipo $Set\ Set\ \alpha$.

def 1 (Petri_net) $N = (\mathcal{P}, \mathcal{T}, \mathcal{R}, s_0)$ ***where***

$\mathcal{P}$	$--\ places$
$\mathcal{T}$	$--\ transitions$
$\mathcal{R} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$	$--\ relation$
$s_0 \subseteq \mathcal{P}$	$--\ initalstate$

notation: ls '$\prec$' rs $\Rightarrow (ls, rs) \in R$

fun 1 (Preset) Sea $p \in \mathcal{P}$ definimos el preset como ${}^\bullet p := \{t \in \mathcal{T} | (t,p) \in R\}$.

infix '$\bullet$' => PetriNet.preset

fun 2 (Poset) Sea $p \in \mathcal{P}$ definimos el poset como $p^\bullet := \{t \in \mathcal{T} | (p,t) \in R\}$.

postfix '$\bullet$' => PetriNet.poset

fun 3 (Enabled_transition) Para cada estado $s \in \mathcal{S}$ el conjunto de transiciones habilitadas está dada por $en(s) := \{t \in \mathcal{T} | ({}^\bullet t \subset s) \wedge (t^\bullet \cap s \subset^\bullet t)\}$.

def 2 (Deadlock) $s \in \mathcal{S}$ está en deadlock si y sólo si $en(s) = \emptyset$.

fun 4 (Fired_transition) Dados un estado $s \in \mathcal{S}$ y una transición $t \in en(s)$ definimos la ejecución de s a través de t como $s[t\rangle = (s \setminus^\bullet t) \cup t^\bullet$.

Denotamos el firing $s[t\rangle_N$ para especificar la red de Petri N sobre la cual se ejecuta el estado y la transición.

def 3 (Firing sequence) Diremos que $s_0[t_1\rangle s_1[t_2\rangle \dots s_{n-1}[t_n\rangle s_n$ es una secuencia ordenada de ejecuciones, donde $t_i \in \mathcal{T}$, $s_i \in \mathcal{S}$ y $s_{i-1}[t_i\rangle = s_i$ para cada $i = 1, \dots, n$ y s_0 es el estado inicial de N. A esta secuencia de transiciones se denota por una lista $[t_1, \dots, t_n]$. Denotamos $s_0[seq\rangle s_n$ (o $s_0[*\rangle s_n$) si existe una secuencia $seq = [t_1, \dots, t_n]$ tal que $s_0[t1\rangle s_1 \dots s_{n-1}[t_n\rangle s_n$ sea una secuencia de ejecuciones.

Si s_0 es un estado inicial, $reach(s_0)$ retorna todos los estados posibles de una red de Petri.

def 4 (Conflict) Dos transiciones t_1, t_2 están en conflicto si ${}^\bullet t_1 \cap^\bullet t_2 = \emptyset$

notation:80 lhs : 81 "# "rhs : 80 $\Rightarrow$ Conflict lhs rhs

def 5 (Occurrence_net) Una red de Petri N es de ocurrencia si y solo si se cumplen las siguientes condiciones:

- Es acíclica.
- Es 1-safe.
- $\bigcup_{p \in s_0} {}^\bullet p = \emptyset$.
- No hay conflictos hacia atrás ($\forall a \in \mathcal{P}.|{}^\bullet a| \leq 1$).
- No tiene auto-conflicto ($\forall t \in \mathcal{T}.\neg(t \# t)$).

def 6 (Concurrent) Para cada $x, y \in \mathcal{P} \cup \mathcal{T}$, decimos que x e y son concurrentes si y solo si $x \neq y \wedge \neg(x \prec y) \wedge \neg(y \prec x) \wedge \neg(x \# y)$.

lemma 1 *Sea O una red de ocurrencia entonces* $\forall t, t' \in en(s0) \leftrightarrow (co\ t\ t' \leftrightarrow^\bullet t \cap^\bullet t' = \emptyset)$

def 7 (Redes reversibles) Dada una red de Petri $N = (\mathcal{P}_N, \mathcal{T}_N, R, s_0)$ su versión reversible es la red de Petri $\overleftarrow{N} = (\mathcal{P}_{\overleftarrow{N}}, \overleftarrow{\mathcal{T}}_{\overleftarrow{N}}, \overleftarrow{R}, s_0)$, donde $\mathcal{P}_{\overleftarrow{N}} = \mathcal{P}_N$, $\mathcal{T}_{\overleftarrow{N}} = \mathcal{T}_N \cup \{\overleftarrow{t} \,|t \in \mathcal{T}_N\}$, y para cada $t \in \mathcal{T}_{\overleftarrow{N}}$

$$\overset{\bullet}{\underset{\overleftarrow{N}}{}}t = \begin{cases} \overset{\bullet}{\underset{N}{}}t & si\ t \in \mathcal{T}_N \\ t\overset{\bullet}{\underset{N}{}} & si\ t \notin \mathcal{T}_N \end{cases} \qquad t\overset{\bullet}{\underset{\overleftarrow{N}}{}} = \begin{cases} t\overset{\bullet}{\underset{N}{}} & si\ t \in \mathcal{T}_N \\ \overset{\bullet}{\underset{N}{}}t & si\ t \notin \mathcal{T}_N \end{cases}$$

$\overleftarrow{R}$ es la clausura simétrica de R (es decir, $\overleftarrow{R}$ es de equivalencia).

3. Enunciados de proposiciones

lemma 2 *Toda red de ocurrencia es reversible*

lemma 3 *Sea $t; t'$ una traza de una red de ocurrencia reversible. Entonces t co t' si y sólo si $\overleftarrow{t}$ co t'.*

lemma 4 *Sea $O = (\mathcal{P}, \mathcal{T}, R, s_0)$ una red de ocurrencia, entonces para cada estado $s \in reach(O, s_0)$ tal que $\exists t \in \mathcal{T}.s[t\rangle = s'$ implica que $s[t\rangle_O s' = s'[t\rangle_{\overleftarrow{O}} s$*

theorem 1 *Sea $O = (\mathcal{P}_O, \mathcal{T}, R, s_0)$ una red de ocurrencia. Entonces para cada $s \in \mathcal{S}$,*

$$s_0[*\rangle_O s_n \longleftrightarrow s_n[*\rangle_{\overleftarrow{O}} s_0$$

Referencias

[1] H. Melgratti, C. Antares, I. Ulidowski. *Reversing place transition nets*, Logical Methods in Computer Science, 1-28, 2020.

[2] J. Kaoten, D. Peled. *Taming confusion for modeling and implementing probabilistic concurrent systems.*

[3] J. Avigad, P. Massot. *Mathematics in Lean*, 2023.

[4] H. Lehmann, M. Leuschel, *Inductive Theorem Proving by Program Specialisation: Generating proofs for Isabelle using Ecce*, Department of Electronics and Computer Science, University of Southampton.