A Kleene Functor for a Subclass of Net Systems

Ramchandra Phawade

Joint work with
Kamal Lodaya and Madhavan Mukund

January 29, 2011

Net system, Language of a Net system

Definition

Fix a finite alphabet A and a finite set of locations Loc.
A net $N = (S, T, \ell, loc, F)$ over A and Loc has

- S a finite set of places;
- T a finite set of transitions
- $\ell : T \to A$ is the labelling function
- $loc : T \to \wp(Loc)$ is location function
- $F \subseteq (S \times T) \cup (T \times S)$ is flow relation.

Net system, Language of a Net system

- A **marking** of a net is a function $M : S \to \mathbb{N}$.
- A **net system** is a pair (N, M_0).
- **1-bounded** net systems - where the range of the marking function is $\{0, 1\}$.
- The **language** accepted by the net system (N, M_0) is:

  the set of maximal firing sequences
- **trace-labelled net** :

  two transitions with the same label also have the same locations.

A trace labelled net system

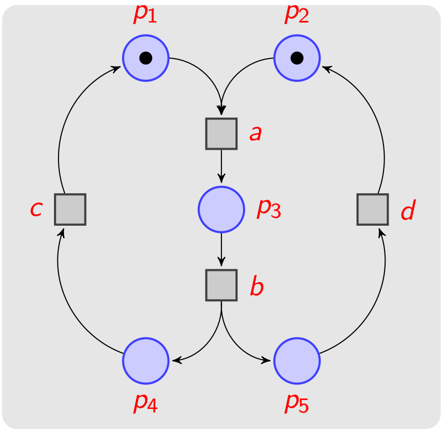

Figure 1

- $T = \{t_1, t_2, t_3, t_4\}$
- $A = \{a, b, c, d\}$
- $l(t_1) = a,\ l(t_2) = b,\ l(t_3) = c,\ l(t_4) = d$
- $Loc = \{1, 2\}$
- $loc(t_1) = loc(t_2) = \{1, 2\}$ and $loc(t_3) = \{1\}, loc(t_4) = \{2\}$

- $\{p_1, p_2\}t_1\{p_3\}t_2\{p_4, p_5\}t_3\{p_6, p_5\}t_4\{p_1, p_2\}$
  $t_1\{p_3\}t_2\{p_4, p_5\}t_3\{p_6, p_5\}t_4\{p_1, p_2\}$
  $\cdots$
  Hence,
  $abcdabcdabcd \cdots \in Lang(N, M_0)$

Language of Net system : maximal firing sequence

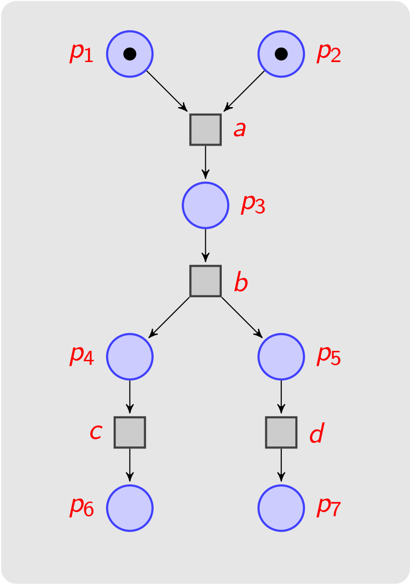

Figure 2

- $T = \{t_1, t_2, t_3, t_4\}$
- $A = \{a, b, c, d\}$
- $l(t_1) = a,\ l(t_2) = b,\ l(t_3) = c,\ l(t_4) = d$
- $Loc = \{1, 2\}$
- $loc(t_1) = loc(t_2) = \{1, 2\}$ and $loc(t_3) = \{1\}, loc(t_4) = \{2\}$
- $\{p_1, p_2\}t_1\{p_3\}t_2\{p_4, p_5\}t_3\{p_6, p_5\}t_4\{p_6, p_7\}$ is a maximal firing sequence. Hence, $abcd \in Lang(N, M_0)$
- $\{p_1, p_2\}t_1\{p_3\}t_2\{p_4, p_5\}$ is a not a maximal firing sequence. Hence, $ab \notin Lang(N, M_0)$.

T-systems, T-automaton

- **T-net** : $N = (S, T, F, l, loc)$
  for $t_1 \neq t_2$, ${}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$ and $t_1^\bullet \cap t_2^\bullet = \emptyset$.
- **T-system** : (N, M_0)
- **Distributed transition** : $|{}^\bullet t| = |t^\bullet| = |loc(t)|$
- **T-automaton** :
  T-system where all transitions are distributed.
- **trace-labelled net** :
  two transitions with the same label also have the same locations.

- Nets given in Figure 1 and Figure 2 satisfy all above

Categorical approach to nets

- petri nets are monoids: Meseguer and Montanari (1990)
  - form symmetric monoidal category.
  - semantics given in terms of case graphs.
  - no expressions
  - uses unbounded petri nets.
- many others but no expressions for nets

C-module

Fix a category $\mathbf{C}(Obj(\mathbf{C}), Arr(\mathbf{C}), \cdot)$.

A $\mathbf{C}$-module $\mathbf{M} = (Obj(\mathbf{M}), Arr(\mathbf{M}), \circledast)$ is

- $Obj(\mathbf{M})$ are in bijection with $Obj(\mathbf{C})$.
- $Arr(\mathbf{M})$
- left action $\circledast$
$$\mathbf{C}(c, a) \circledast \mathbf{M}(a, b) \in \mathbf{M}(c, b)$$
- for $f, g \in Arr(\mathbf{C})$ and $m \in Arr(\mathbf{M})$
$$f \circledast (g \circledast m) = (f \cdot g) \circledast m$$
- Identities of $\mathbf{C}$ act as identity actions on $\mathbf{M}$
$$I \circledast m = m.$$

symocat-module-pair

A **C**-module **M**$=(Obj(\mathbf{C}), Arr(\mathbf{M}), \circledast, ^{\omega})$ is

- a **C**-module **M** with an
- ω-power operation from **C**to **M**satisfying, $f \circledast f^{\omega} = f^{\omega}$.

A special case is when $Arr(\mathbf{M}) = \{f \circledast g^{\omega} \mid f, g \in Arr(\mathbf{C})\}$. Then

- **M** is called a **C** -power
- (**C**,**M**) is called a **symocat-module pair**.
- Similar structures has been used by
  - Perrin and pin in *automata and semigroups*, Infinite Words: Automata, Semigroups, Logic and Games
  - Esik and Kuich, *Finite automata*, Handbook of weighted automata

Symocat-module structure for T-systems: (**Tsys**, **Tlive**)

1. **objects**: *subsets of Loc*.
2. **arrows**:
   - Given $L \subseteq Loc$, the arrows $L \to L$ are acyclic T-automata (N, M_0), such that $|M_0| = |L| =$ the cardinality of the sink places of N. There are no arrows $L \to M$ for L and M different.
   - The *identity* for L denoted by $1_L : L \to L$ is the T-automaton consisting of just $|M_0|$ marked places and no transitions.
   - There is also a *zero* arrow denoted by $z_L : L \to L$, the empty T-automaton. It consists of $|M_0|$ unmarked places and no transitions.

Symocat-module structure for T-systems:(**Tsys**, **Tlive**)

Arrow Composition with an intermediate set of locations L is defined by identifying the sink places of the first (acyclic) T-automaton with the initially marked places of the second.

Let $f : L \to L$ and $g : L \to L$ be two arrows.

- f and g both non-zero and non-identity; result is easy to see. It is concatenation of two T-systems.
- $f \cdot 1_L = f = 1_L \cdot f$
- $f \cdot z_L = z_L = z_L \cdot f$
- composition is associative.

Categorical structure for T-systems: (**Tsys**, **Tlive**)

- *tensor* ($\otimes$) operation:
  Let $L, M \subseteq Loc$.
  $\otimes$: **Tsys** $\times$ **Tsys** $\rightarrow$ **Tsys**:
  $\otimes : \; L \times M \rightarrow L \cup M$.
  $\otimes : (\, f : L \rightarrow L) \times (\, g : M \rightarrow M) \rightarrow (f \otimes g : \; (L \cup M) \rightarrow (L \cup M))$.
  $\otimes$ performs *union* on the objects and *synchronization* on the arrows.
  resultant T-automaton is the union of the two T-automata except that the transitions with common labels are fused.
- This operation has a natural symmetry. Partial distributivity $(f1 \otimes f2) \cdot (g1 \otimes g2) = (f1 \cdot g1) \otimes (f2 \cdot g2)$ holds provided that both sides are not zero.
- 1_L acts as the *idenitity* for *composition* while z_L acts as the *annihilator* for *composition* in **Tsys**(L, L). Also, (**Tsys**$(L, L), \cdot, 1_L)$ is a monoid.

Expressions for nets

- Grabowski (1981)
  - uses $\cdot, +, \|, ^*$ and *rename* operation..
  - semantics of expressions in *pomsets*
  - translation between expressions and 1-bounded systems given.
- Ochmanski (1985)
  - star-connected expressions
  - $\|$ and *concurrent star* operation
  - translation between net systems to expressions is given.
- Garg and Raghunath (1992)
  - uses Grabowski's syntax along with *shuffle closure* operation.
  - translation between expressions and (unbounded) nets given.
  - size of expressions have exponential lowerbound as one component treated as finite automata.

Expression syntax

Let A be a finite alphabet.

Definition

Our expressions come in three syntactic sorts: **sequences** s, **connected expressions** c and **T-expressions** e.

$$\begin{gathered} s ::= \epsilon | a \in A | s_1 s_2 \\ c ::= \emptyset | s | sync(c_1, c_2) \\ e ::= c^{\omega} | e_1 || e_2 \end{gathered}$$

The alphabetic width $wd(e)$ of expression e is defined to be the number of occurrences of letters (from A) in e.

Semantics of expressions

Let *Loc* be a set of locations.
For the connected expressions c it is a language of finite words, for the T-expressions e it is a language of infinite words.
Formally,

- $Lang(s) = \{s\}$.
  Each sequence s is also assigned a *location* l, which is disjoint from other locations.
- $sync(c_1, c_2)$ common letters are $X = \alpha(c_1) \cap \alpha(c_2)$

  $$Lang(sync(c_1, c_2)) = \bigcup\{sync_X(w_1, w_2) \mid w_1 \in L_1,\ w_2 \in L_2\}.$$

  computing locations:
  - c_1 location function loc_1 , locations Loc_1
  - c_2 location function loc_2 , locations Loc_2
  - $sync(c_1, c_2)$ has loc over the locations $Loc_1 \cup Loc_2$
    - ★ $a \in X$, $loc(a) = loc_1(a) \cap loc_2(a)$
    - ★ For the other letters in A, $loc(a)$ is inherited from loc_1 or loc_2

Semantics of expressions

- In c, aIb, if $loc(a)$ and $loc(b)$ are disjoint.
- I is irreflexive and transitive
- equivalence relation $\sim$ on A^* by letting $wabv \sim wbav$, for aIb independent occurrences, and taking the reflexive and transitive closure. This is usually called trace equivalence. We write $[w]$ for the equivalence class of w.

Note that our semantics for the *sync* operator yields unions of equivalence classes under the trace equivalence. We let $[L] = \bigcup\{[w] \mid w \in L\}$.

- Consider expression c^ω.
  Assume a given $loc : \alpha(c) \to \wp(Loc)$. The independence relation is the one computed for the expression c. The semantics of c^ω is the trace equivalence closure:

$$Lang(c^\omega) = [(Lang(c))^\omega], \text{ where } L^\omega = \{w_1 w_2 \cdots \mid \forall i, w_i \in L\}.$$

Categorical structure for expressions:(**Texp**, **Tinf**)

1. objects are *subsets of Loc*.
2. arrows
   - Given $L \subseteq Loc$, The arrows $L \to L$ are connected expressions over L. There are no arrows $L \to M$ for L and M different.
   - The *identity* for L denoted by $\epsilon_L : L \to L$ is the empty sequence. wlg we can assume that $\epsilon_L = \epsilon$.
   - There is also a *zero* arrow denoted by $\emptyset_L : L \to L$, is $\emptyset$ expression. wlg we can assume that $\emptyset_L = \emptyset$.

Categorical structure for expressions:(**Texp**, **Tinf**)

Arrow Composition:
Composition of arrows, of sequences is provided in the syntax. By inductively applying partial distributivity and associativity of the *sync* operation we get composition for all connected expressions.

Let f and g be two arrows in $\mathbf{Texp}(L, L)$.

1. f and g both non-zero and non-identity; result is easy to see. It is concatenation of two c-expressions as explained above.
2. $f \cdot 1_L = f = 1_L \cdot f$ since $1_L = \epsilon$
3. $f \cdot z_L = z_L = z_L \cdot f$ since $z_L = \emptyset$
4. composition is associative.

Categorical structure for expressions:(**Texp**, **Tinf**)

- *Tensor* ($\otimes$) operation:
  Let $L, M \subseteq Loc$.
  $\otimes$: **Texp** $\times$ **Texp** $\rightarrow$ **Texp**:
  $\otimes : \; L \times M \rightarrow L \cup M$.
  $\otimes : (\; f : L \rightarrow L) \times (\; g : M \rightarrow M) \rightarrow (f \otimes g : \; (L \cup M) \rightarrow (L \cup M))$.
  $\otimes$ performs *union* on the objects and *synchronization* on the arrows.
  Arrrows here being expressions, $f \otimes g$ is written as $sync(f, g)$.
- This operation has natural symmetry. Partial distributivity $(f1 \otimes f2) \cdot (g1 \otimes g2) = (f1 \cdot g1) \otimes (f2 \cdot g2)$ holds provided that both sides are defined (that is, not zero).
- ϵ_L acts as the *idenitity* for *composition* while $\emptyset_L$ acts as the *annihilator* for *composition* in **Texp**(L, L). Also, (**Texp**$(L, L), \cdot, \epsilon_L)$ is a monoid.

Expressions to T-automaton (sequences, c-expressions)

Definition

A nonempty 1-bounded T-system where every transition has pre- and post-sets of size 1 is called a **line**. An acyclic connected trace-labelled 1-bounded T-system is called a **dag**.

Lemma (s-expression to path T-automaton)

For $s \in A^*$, we can construct a line T-automaton of size $O(wd(s))$ accepting $Lang(s)$, and it can be computed in linear time

Lemma (c-expression to dag T-automaton)

Let c be a connected expression. Then there exists a dag T-automaton (N, M_0) accepting the same language which is covered by a set Loc of initially marked lines. The size of the constructed system is $O(wd(c))$ and it can be computed in linear time

Functor: $F_1 : \mathbf{Tsys} \to \mathbf{Texp}$

- object L is mapped to itself i.e., $F_1(L) = L$.
- arrow $f : L \to L$ is mapped to $F_1(f) : F_1L \to F_1L$ i.e., $F_1(f) : L \to L$.
- 1_L of $\mathbf{Tsys}(L, L)$ is mapped to a an expression ϵ_L of $\mathbf{Texp}(L, L)$.
- z_L of $\mathbf{Tsys}(L, L)$ is mapped to an expression $\emptyset_L$ of $\mathbf{Texp}(L, L)$.
-
  - Let $f,\ g \in \mathbf{Tsys}(L, L)$, which when composed give $f \cdot g$ in $\mathbf{Tsys}(L, L)$.
  - Now let $F_1(f) = sync(c_1, c_2)$ and $F_1(g) = sync(c_3, c_4)$.
  - By inductively applying partial distributivity and associativity of the *sync* operation we get composition for connected expressions which is of form $sync(c, c') \in \mathbf{Texp}(L, L)$.
- For f in $\mathbf{Tsys}(L, L)$ and g in $\mathbf{Tsys}(M, M)$, $F_1(f \otimes g) = sync(F_1(f), F_1(g))$ which is an arrow in $\mathbf{Texp}(L \cup M, L \cup M)$.

Net systems to Expressions (lines, dags)

Lemma (line T-automaton to s-expression)

Let $(N = (S, T, F), M_0, \ell)$ be a nonempty line. Then there exists an equivalent expression s for its language. The alphabetic width of this expression is $O(|T|)$ and it can be computed in time $O(|N|)$.

Lemma (dag T-automaton to c-expression)

Let $(N = (S, T, F), M_0)$ be a dag. There is a connected expression c for $Lang(N, M_0)$ of alphabetic width $O(|S| \times |T|)$ which can be computed in time $O(|N|^3)$.

Functor: $G_1 : \textbf{Texp} \to \textbf{Tsys}$

- object L is mapped to itself i.e., $G_1(L) = L$.
- arrow $f : L \to L$ is mapped to $G_1(f) : G_1 L \to G_1 L$ i.e., $G_1(f) : L \to L$.

- $1_L = \epsilon_L$ of $\textbf{Texp}(L, L)$ is mapped to an T-automaton 1_L of $\textbf{Tsys}(L, L)$
- $z_L = \emptyset_L$ of $\textbf{Tsys}(L, L)$ is mapped to an T-automaton z_L
- Let $f = sync(c_1, c_2)$ and $g = sync(c_3, c_4)$ be two expressions in $\textbf{Texp}(L, L)$ which when composed (using associativity and distributivity) give $f \cdot g = sync(c, c^{'})$ in $\textbf{Texp}(L, L)$.
- For $f = sync(c_1, c_2)$ in $\textbf{Texp}(L, L)$ and $g = sync(c_3, c_4)$ in $\textbf{Texp}(M, M)$, $G_1(sync(f, g)) = G_1(f) \otimes G_1(g)$ is an arrow in $\textbf{Tsys}(L \cup M, L \cup M)$.

Functor: F_2 : **Tlive** $\rightarrow$ **Tinf**

Theorem

Let (N, M_0) be a trace-labelled live, 1-bounded T-system. Then we can compute in $O(|N|^3)$ time a T-expression of alphabetic width $O(|N|^2)$ for the accepted language.

- object mapping: $F_2(L) = L$
- arrow f of **Tlive**(L, L) is mapped to $F_2(f) : F_2(L) \rightarrow F_2 L$ in **Tinf**(L, L).
- for f in **Tsys**(L, L) and f^ω in **Tlive**(L, L) we get f^ω in **Tlive**(L, L) by defn of arrows in *module*, i.e., $f^\omega = f \circledast f^\omega$.

  Let $F_1(f) = c$.
  F_1 is the underlying functor of F_2.
  Then, $F_1(f) \circledast (F_1(f))^\omega = c \circledast c^\omega = c^\omega$ in **Tinf**(L, L).

Functor: F_2 : **Tlive** $\rightarrow$ **Tinf**

- How do we get the expression of form $e_1||e_2$ in **Tinf**?

If f is a map in **Tsys**(L, L) and g is a map in **Tsys**(M, M) then f^ω is a map in **Tlive**(L, L) and g^ω is a map in **Tlive**(M, M).

The $f \otimes g$ in **Tlive**$(L \cup M, L \cup M)$ is *disjoint union in* **Tlive**$(L \cup M, L \cup M)$ by definition of module **Tlive**.

Therefore if $F_2(f) = c_1$ and $F_2(g) = c_2$ then $c_1^\omega \; || \; c_2^\omega$ is the expression in **Tinf**$(L \cup M, L \cup M)$.

Functor G_2 : **Tinf** $\rightarrow$ **Tlive**

Theorem

Let $e = c^\omega$ be an expression over alphabet A. Then there exists a trace-labelled, live T-automaton (N, M_0) such that $Lang(e) = Lang(N, M_0)$. The size of the system is $O(wd(e))$ and it can be constructed in linear time

- $G_2(L) = L$
- $G_2(f) : G_2(L) \rightarrow G_2(L)$ in **Tlive**(L, L).
- for c in **Texp**(L, L) and c^ω in **Tinf**(L, L) we get c^ω in **Tinf**(L, L) by defn of arrows in *module*, i.e., $c^\omega = c \circledast c^\omega$.

  Let $G_1(c) = f$.
  G_1 is the underlying functor of G_2.
  Then, $G_1(c) \circledast (G_1(c))^\omega = f \circledast f^\omega = f^\omega$ in **Tlive**(L, L).

Functor G_2 : **Tinf** $\to$ **Tlive**

- How do we get two live, 1-bdd disjoint nets in **Tlive**?

  If c_1 is a map in **Texp**(L, L) and c_2 is a map in **Texp**(L, L), then c_1^ω is a map in **Tinf**(L, L), then c_2^ω is a map in **Tinf**(L, L).

  The $c_1 \otimes c_2$ in **Tinf**$(L \cup M, L \cup M)$ is *shuffle operator in* **Tinf**.

  Therefore, if $G_2(c_1) = f$ and $G_2(c_2) = g$ then, *disjoint union* of two nets f^ω and g^ω is the resultant net in **Tlive**$(L \cup M, L \cup M)$.

Conclusion

- From the early work of Elgot, Bloom, Ésik, Ştefănescu and Cazanescu it is known that the Kleene theorem can be seen as functors between the categories of regular expressions and finite automata. The regular expression constructed from an automaton can be exponential in the worst case (Ehrenfeucht and Zeiger).
- We showed that a similar result can be obtained for trace-labelled live and 1-bounded T-systems. Because of the T-system property we get polynomial constructions in both directions.
- We believe this result can be extended to 1-bounded free choice systems (which subsume automata as well as T-systems), but quite likely this will involve symmetric semiringal categories which have two tensor operations.