Logical Semantics of Petri Nets

Def: A Petri net \( N = (M(P), L, \rightarrow_N) \) is a labeled directed graph such that its set of nodes \( M(P) \) is the commutative monoid of multisets on a set \( P \) of places.

A transition of \( N \) has the form:

\[ m \xrightarrow{\ell} m' \text{ where } m, m' \in M(P), \ell \in L \]

We use multiplicative notation for \( M(P) \). For example, if \( P = \{p_1, \ldots, p_k\} \), then \( p_1^3 p_2 p_3 p_5 \in M(P) \), which can be abbreviated to \( p_1^3 p_2 p_5 \). The unit element is denoted \( 1 \in M(P) \).

The logic \( \mathcal{L}(N) \) of Petri Net \( N = (M(P), L, \rightarrow_N) \)

---

**Idle/Reflexivity**

\[
\text{Idle/Reflexivity} \quad \quad m \in M(P) \\
\text{Idle/Reflexivity} \quad \quad m \xrightarrow{\ell} m
\]

**Action**

\[
\text{Action} \quad \quad (m, \ell, m') \in \rightarrow_N \\
\text{Action} \quad \quad m \xrightarrow{\ell} m'
\]
Congruence

\[ m \xrightarrow{\alpha} m', \quad u \xrightarrow{\beta} u' \]

\[ m u \xrightarrow{\alpha \beta} m' u' \]

Transitivity

\[ m \xrightarrow{\alpha} u, \quad u \xrightarrow{\beta} v \]

\[ m \xrightarrow{\alpha \beta} v \]

Therefore, this logic is exactly as the logic \( L(A) \) of an automaton \( A \), except for the addition of the Congruence inference rule, which allows us to describe parallel computations in \( A \), in the same way that Transitivity allows us to describe sequential computations.

As for automata, we call any \( M \xrightarrow{\alpha} m' \) for which we can build a proof tree in the logic \( L(N) \) a computation specification from state \( M \) to state \( m' \) with proof term \( \alpha \). Again, as for automata, this defines a directed graph:

\[ \text{Spec}(N) = (M(P), \text{ProofTerms}, \xrightarrow{\text{spec}(N)}) \]
where \((m, \alpha, m') \in \text{Spec}(N)\) iff \(m \xrightarrow{\alpha} m'\) is a computation specification for \(N\).

The Computations of \(N\)

As for automata, we can ask: (1) When do two computation specifications \(m \xrightarrow{\alpha} m'\) and \(m \xrightarrow{\beta} m'\) describe the same computation from \(m\) to \(m'\)? This question is equivalent to the more interesting question:

(2) What is a concurrent computation of the Petri net \(N\)?

Of course, our answer, as for automata, to question (2) is: a concurrent computation for \(N\) is an equivalence class \([m \xrightarrow{\alpha} m']\) of computation specifications under a natural equivalence relation \(\equiv\) between computation specifications.

How can we define \(\equiv\)? Of course, as for automata, we should also have:

**Associativity**

\(m \xrightarrow{(\alpha; \beta); \gamma} m' \equiv m \xrightarrow{\alpha; (\beta; \gamma)} m'\)

**Identity**

\(m \xrightarrow{\alpha} m' \equiv m \xrightarrow{\alpha} m' \equiv m \xrightarrow{\alpha; m'}\)
What else? Obviously, since $M(P)$ is a commutative monoid
we have: $(m \cdot m')m'' = m(m' \cdot m'')$, $m \cdot m' = m' \cdot m$, $m \cdot \text{null} = m$.

But since proof terms such as $\alpha \beta$ and $\beta \alpha$ only
depend on the decomposition $mm'$ versus $m'm$, and so
on, proof terms themselves should also form a commutative
monoid, i.e., we should have for $m_1 \xrightarrow{\alpha} m'_1$, $m_2 \xrightarrow{\beta} m'_2$, $m_3 \xrightarrow{\gamma} m'_3$:

\begin{equation*}
(\text{Parallel) Associativity} \quad m_1 m_2 m_3 \xrightarrow{(\alpha \beta)\gamma} m'_1 m'_2 m'_3 \equiv m_1 m_2 m_3 \xrightarrow{\alpha(\beta\gamma)} m'_1 m'_2 m'_3 \end{equation*}

\begin{equation*}
(\text{Parallel) Unit} \quad m_1 \xrightarrow{\text{null}} m'_1 \equiv m_1 \xrightarrow{\alpha} m'_1 \end{equation*}

\begin{equation*}
(\text{Parallel) Commutativity} \quad m_1 m_2 \xrightarrow{\alpha \beta} m'_1 m'_2 \equiv m_1 m_2 \xrightarrow{\beta \alpha} m'_1 m'_2 \end{equation*}

Anything else? Yes! We should capture the very intuitive
fact that parallel and sequential compositions "commute with
each other in the following sense: if we have a parallel
composition specification $m_1 m_2 \xrightarrow{\alpha \beta} m'_1 m'_2$ this should
be equivalent to first drop $\alpha$ and then $\beta$ or in vice versa, i.e.,

\begin{equation*}
(\star) \quad \xrightarrow{\alpha m_2} \quad \equiv \quad \xrightarrow{\alpha m'_2} \quad \equiv \quad \xrightarrow{m'_1 \beta} \end{equation*}
We can express this "commutation between sequential and parallel computation" in an even more general way (so that, using the identity of sequential composition we get [Exercise!] the equivalence \(\ast\) as special case as follows: Let 
\[ m_1 \xrightarrow{\alpha'} m'_1 \quad \text{and} \quad m_2 \xrightarrow{\beta'} m'_2 \] 
be also computation specifications. Then we have:

\[
\begin{align*}
    m_1 m_2 & \xrightarrow{\alpha, \beta} m'_1 m'_2 \\
    (M_1 M_2) & \xrightarrow{\alpha, \beta} (M'_1 M'_2) \equiv \frac{M_1 M_2 \equiv M'_1 M'_2}{M_1 m_2 \equiv m'_1 M_2}
\end{align*}
\]

The strict symmetric monoidal category of computations \(\mathcal{J}_N\) of \(N\) is defined as the category:

\[ \mathcal{J}_N = (M(P), \text{ProofTerm}/\equiv, \rightarrow_{\mathcal{J}_N}) \]

where

\[ \frac{m \xrightarrow{\alpha} m'}{\mathcal{J}_N} \iff m \xrightarrow{\alpha} m' \text{ is a computation specification.} \]

Because of the associativity and identity axioms for \(-\rightarrow_{\mathcal{J}_N}\), \(\mathcal{J}_N\) is obviously a category of computations. But it has the additional structure of a strict symmetric monoidal category. What is that?
$$\left(\frac{\text{weight} \times \text{speed}^2}{\text{mass} + \text{mass}}\right) = \left(\frac{\text{weight} \times \text{speed}^2}{\text{mass} + \text{mass}}\right)$$

(2) Fundamental. The + operation on constant previous constants.

Given + varies with time, we have the

Fundamental

In this way, the operation + on objects + and on

we have an answer of the form: \[ \text{min} \left( \frac{\text{speed}^2}{\text{mass}}, \frac{\text{weight}}{\text{mass}} \right) \]

(4) For any \( m \in M \) and \( n \in N \), then in \( G \), there

+ \( A \times A \rightarrow A \) such that:

+ on \( M \) is just the restriction to \( M \) of the operator + on \( M \), which is commutative. In fact, the

commutative

in \( M, \) \( (A, +) \in (A, +) \in (A, +) \) together with +, commutative

where, without any of previous we assume + = + for each

Moreover, commuting in a category \( G \in (M, A, +) \),

Definition (Strict Symmetric Monoid). A strict symmetric

...
**Theorem.** The category of computations \( \mathcal{C}_N \) of a Petri net \( N \) is a strict symmetric monoidal category.

**Proof.** This follows immediately from the Associativity, Identity, Parallel Associativity, Concurrency and Unit, and Functionality axioms defining the equivalence relation \( \equiv \).

**Exercise.** Prove that for any Petri net \( N \), any computation \( m \xrightarrow{c} m' \) has a [in general not unique] interleaving description as a sequential composition of basic transitions:

\[
\begin{align*}
(m_0 \xrightarrow{c_1} m_1) &= (m_0, m_1) \\
(m_1 \xrightarrow{c_2} m_2) &= (m_1, m_2) \\
&m_{n+1} = m_n \\
&\quad \forall 0 \leq n \leq m+2
\end{align*}
\]

where:

\[
(m_i \xrightarrow{c_i} m_{i+1}) \in \mathcal{N} \quad , \quad 1 \leq i \leq m+2
\]

**Hint:** Try induction on the depth of the proof tree for \( m \xrightarrow{c} m' \).