Logical Semantics of Petri Nets

CS 524 Lecture 4
José Meseguer

Def: A Petri net $\mathcal{N} = (M(P), L, \to_{\mathcal{N}})$ is a labeled directed graph such that it sets of nodes $M(P)$ is the commutative monoid of multisets on a set P of places.

A transition of $\mathcal{N}$ has the form:

$$m \xrightarrow{l} m' \quad \text{where } m, m' \in M(P),\ l \in L$$

We use multiplicative notation for $M(P)$. For example, if $P = \{p_1, .., p_k\}$, then $p_1 p_1 p_1 p_3 p_3 p_5 \in M(P)$, which can be abbreviated to $p_1^3 p_3^2 p_5$. The unit element is denoted null $\in M(P)$.

The Logic $\mathcal{L}(\mathcal{N})$ of Petri Net $\mathcal{N} = (M(P), L, \to_{\mathcal{N}})$

Idle/Refexivity

$$\frac{}{m \xrightarrow{m} m} \quad m \in M(P)$$

Action

$$\frac{}{m \xrightarrow{l} m'} \quad (m, l, m') \in \to_{\mathcal{N}}$$

<u>Congruence</u>

$$\frac{m \xrightarrow{\alpha} m' \qquad u \xrightarrow{\beta} u'}{mu \xrightarrow{\alpha\beta} m'u'}$$

<u>Transitivity</u>

$$\frac{m \xrightarrow{\alpha} u \qquad u \xrightarrow{\beta} v}{m \xrightarrow{\alpha;\beta} v}$$

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

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

$$Spec(\mathcal{N}) = (M(P),\ ProofTerms,\ \xrightarrow[Spec(\mathcal{N})]{})$$

where $(m, \alpha, m') \in \underset{Spec(N)}{\longrightarrow}$ iff $m \xrightarrow{\alpha} m'$ is a computation specification for N.

<u>The Computations of N</u>

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

(2) What is a <u>concurrent computation</u> of the Petri net N?

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

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

<u>Associativity</u> $\quad m \xrightarrow{(\alpha;\beta);\gamma} m' \equiv m \xrightarrow{\alpha;(\beta;\gamma)} m'$

<u>Identity</u> $\quad m \xrightarrow{m;\alpha} m' \equiv m \xrightarrow{\alpha} m' \equiv m \xrightarrow{\alpha;m'} m'$

What else? Obviously, since $M(P)$ is a commutative monoid we have: $(m m')m'' = m(m'm'')$, $m m' = m' m$, $m \, null = m$.

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 $m_1 \xrightarrow{\alpha} m_1'$, $m_2 \xrightarrow{\beta} m_2'$, $m_3 \xrightarrow{\gamma} m_3'$

(Parallel) Associativity $\quad m_1 m_2 m_3 \xrightarrow{(\alpha\beta)\gamma} m_1' m_2' m_3' \equiv m_1 m_2 m_3 \xrightarrow{\alpha(\beta\gamma)} m_1' m_2' m_3'$

(Parallel) Unit $\quad m_1 \xrightarrow{\alpha \, null} m_1' \equiv m_1 \xrightarrow{\alpha} m_1'$

(Parallel) Commutativity $\quad m_1 m_2 \xrightarrow{\alpha\beta} m_1' m_2' \equiv m_1 m_2 \xrightarrow{\beta\alpha} m_1' m_2'$

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 computation specification $m_1 m_2 \xrightarrow{\alpha\beta} m_1' m_2'$ this should be equivalent to first doing α and then β or viceversa, i.e.,

(✡)
$$
\begin{tikzcd}
m_1 m_2 \arrow[r, "m_1 \beta"] \arrow[d, "\alpha m_2"'] \arrow[dr, "\alpha\beta"] \arrow[dr, phantom, shift left=4ex, "\equiv"] \arrow[dr, phantom, shift right=4ex, "\equiv"] & m_1 m_2' \arrow[d, "\alpha m_2'"] \\
m_1' m_2 \arrow[r, "m_1' \beta"'] & m_1' m_2'
\end{tikzcd}
$$

We can express this "commutation between sequential and parallel computations" in an <u>even more general way</u> (so that, using the Identity of sequential composition we get [Exercise!] the equivalence (✡) as special case as follows: Let $m_1' \xrightarrow{\alpha'} m_1''$ and $m_2' \xrightarrow{\beta'} m_2''$ be also computation specifications. Then we have:

<u>Functoriality</u>

$$m_1 m_2 \xrightarrow{(\alpha\beta);(\alpha'\beta')} m_1'' m_2'' \equiv m_1 m_2 \xrightarrow{(\alpha;\alpha')(\beta;\beta')} m_1'' m_2''$$

<u>The Strict Symmetric Monoidal Category of Computations $\mathcal{T}_{\mathcal{N}}$ of $\mathcal{N}$</u>

is defined as the category:

$$\mathcal{T}_{\mathcal{N}} = (M(P),\ \text{ProofTerm}/\equiv,\ \underset{\mathcal{T}_{\mathcal{N}}}{\longrightarrow}),\ \text{where}$$

$$m \underset{\mathcal{T}_{\mathcal{N}}}{\xrightarrow{[\alpha]_\equiv}} m' \quad \text{iff} \quad m \xrightarrow{\alpha} m' \text{ is a computation specification.}$$

Because of the <u>Associativity</u> and <u>Identity</u> axioms for $\_;\_$, $\mathcal{T}_{\mathcal{N}}$ is obviously a <u>category</u> of computations. But it has the additional structure of a <u>strict symmetric monoidal category</u>. What is that?

<u>Definition (Strict Symmetric Monoidal)</u>. A <u>strict symmetric monoidal category</u> is a category $\mathcal{C} = (M, A, \to_{\mathcal{C}})$, where, without loss of generality we assume $id_m = m$ for each $m \in M$, so that $M \subseteq A$, together with a commutative <u>submonoid</u> inclusion $(M,+,0) \subseteq (A,+,0)$, [that is, the constant 0 is the <u>same</u> in M and A, and the operation $+$ on M is just the <u>restriction</u> to M of the operation $+ : A \times A \to A$ on A] such that:

(1) For any $m_1 \xrightarrow{\alpha} m_1'$ and $m_2 \xrightarrow{\beta} m_2'$ in $\mathcal{C}$, then we have an arrow of the form: $m_1 + m_2 \xrightarrow{\alpha+\beta} m_1' + m_2'$ in $\mathcal{C}$.
In other words, the operation $+$ on <u>objects</u> M and on ~~arrows~~ <u>labels</u> A extends to a <u>monoid operation</u> $+$ on ~~arrows~~ [labeled] <u>arrows</u> for which the labeled arrow $0 \xrightarrow{0} 0$ is the <u>unit</u>.

(2) <u>Functoriality</u> [the $+$ operation on arrows preserves composition] Given arrows $m_1' \xrightarrow{\alpha'} m_1''$ and $m_2' \xrightarrow{\beta'} m_2''$ we have the equality of arrows:

$$\left(m_1 + m_2 \xrightarrow{(\alpha+\beta);(\alpha'+\beta')} m_1'' + m_2''\right) = \left(m_1 + m_2 \xrightarrow{(\alpha;\alpha')+(\beta;\beta')} m_1'' + m_2''\right)$$

<u>Theorem</u>. The category of concurrent computations $\mathcal{T}_{\mathcal{N}}$ of a Petri net $\mathcal{N}$ is a strict symmetric monoidal category.

<u>Proof</u>. This follows immediately from the Associativity, Identity, Parallel Associativity, Commutativity and Unit, and Functoriality axioms defining the equivalence relation $\equiv$.

<u>Exercise</u>. Prove that for any Petri net $\mathcal{N}$, any computation $m \xrightarrow{\alpha} m'$ has a [in general not unique] <u>interleaving description</u> as a <u>sequential composition</u> of basic transitions:

$$(m \xrightarrow{\alpha} m') = (m = m_0 m_1 \xrightarrow{m_0 l_1} m_0 m'_2 = m'_0 m_2 \xrightarrow{m'_0 l_2} m'_0 m'_3 = m''_0 m_3 \xrightarrow{m''_0 l_3} m''_0 m'_4 = m'''_0 m_4$$

$$m_0^{(n)} m'_{n+1} = m_0^{(n+1)} m_{n+2} \xrightarrow{m_0^{(n+1)} l_{n+2}} m_0^{(n+1)} m'_{n+3} = m')$$

where: $\quad (m_i \xrightarrow{l_i} m'_{i+1}) \in \underset{\mathcal{N}}{\longrightarrow} \quad , 1 \leq i \leq n+2$

<u>Hint</u>: Try induction on the depth of the proof tree for $m \xrightarrow{\alpha} m'$.