T-79.231 Parallel and Distributed Digital Systems

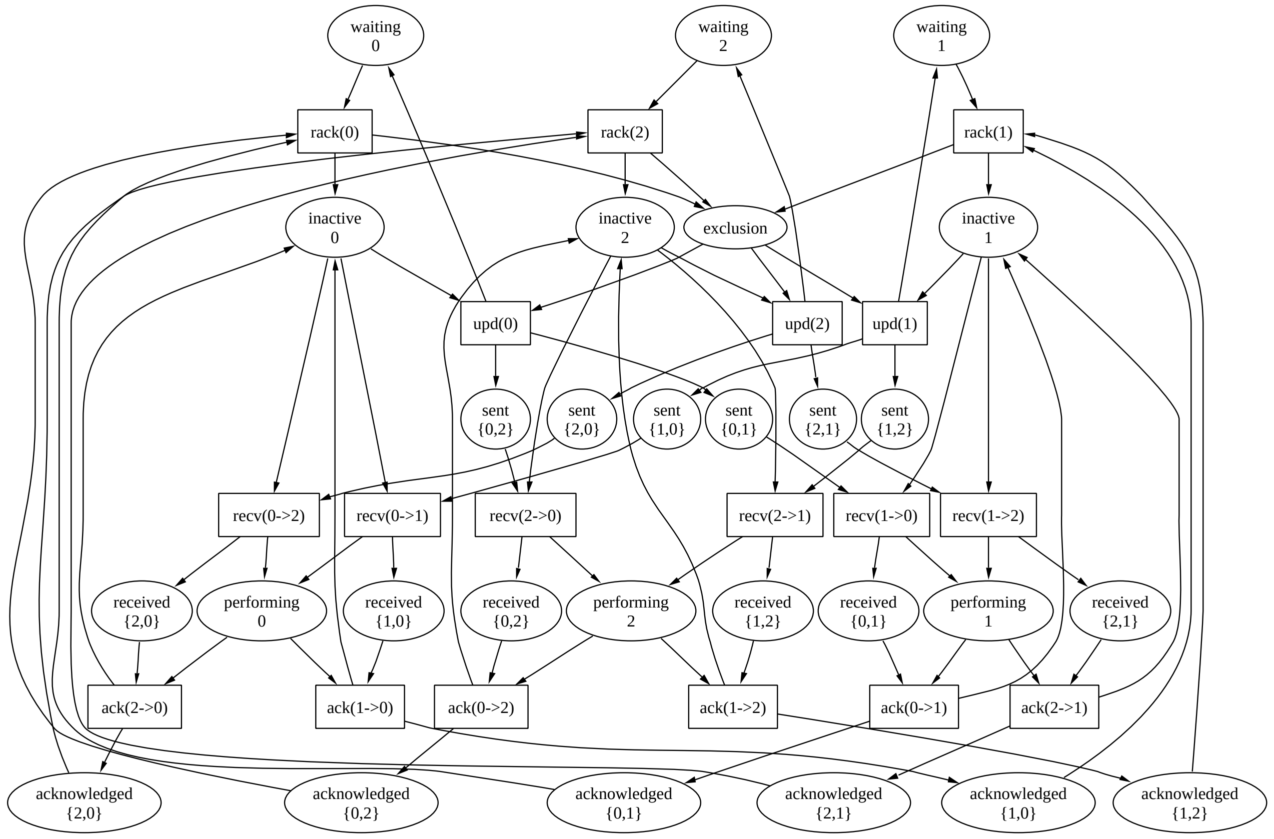
High-Level Nets

Marko Mäkelä

July 16, 2002

High-Level Nets

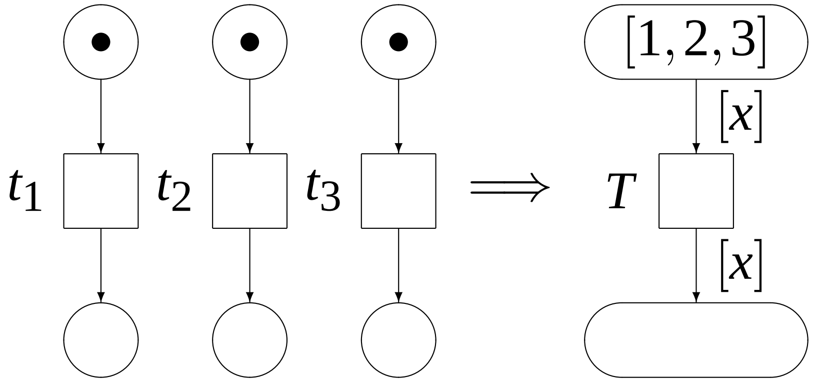
Place/transition systems are not suitable for modelling bigger systems:



Marko Mäkelä

Generalising Place/Transition Nets (1/2)

Another problem place/transition nets have is that the tokens are equal and indistinguishable. The net on the previous slide depicts a database distributed to three processors. What if the tokens were assigned data types and values?



The transitions t_1, t_2 and t_3 are *instances* of the same operation T for different actors x: $t_x \mathrel{\hat{=}} \langle T, x \rangle$.

Generalising Place/Transition Nets (2/2)

The marking (state) of a place/transition system is a mapping from the places $s \in S$ to the numbers of tokens they contain: $S \to \mathbb{N}$.

A transition is enabled if its each input place contains at least as many tokens as the input arc weight indicates. When an enabled transition fires, the input arc weights will be subtracted from the input places and the output arc weights will be added to the output places.

In a high-level net, the marking of a place $s \in S$ is not an integer $n \in \mathbb{N}$ but a mapping from its data type $\mathcal{D}_S$ to integers, $\mathcal{D}_S \to \mathbb{N}$. Thus, the marking indicates the number of tokens of each value separately. The arc weights are generalised to similar mappings, *arc expressions*. The transitions are generalised by introducing *variables* and optional *guard expressions*. Arc and guard expressions may refer to variables.

Markings and arc expressions in a high-level net (1/2)

A place of a high-level net corresponds to a set of places in the *underlying* place/transition net, according to the data type domain of the high-level place. Similarly, a high-level transition corresponds to a number of low-level transitions, one for each possible valuation for the variables.

Formally, the marking of a high-level net is a mapping $S \rightarrow (\mathcal{D} \rightarrow \mathbb{N})$ where $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$ is the union of the data types, a.k.a. *sorts*.

The marking of a place/transition system $M : S \rightarrow \mathbb{N} : M(s_k) \mapsto n_{s_k}$ can be written as $\bigcup_{s \in S} \{\langle s, n_s \rangle\}$. It is more difficult to write a high-level marking $M' : S \rightarrow (\mathcal{D} \rightarrow \mathbb{N}) : M'(s_k) \mapsto M'_{s_k}$ where $M'_{s_k}(d_l) \mapsto n_{\langle s_k, d_l \rangle}$ in a similar way: $\bigcup_{s \in S} \bigcup_{d \in \mathcal{D}_s} \big\{ \big\langle s, \langle d, n_{\langle s,d \rangle} \rangle \big\rangle \big\}$.

Markings and arc expressions in a high-level net (2/2)

A *multi-set* or a *bag* differs from a set by being able to contain several copies of a single item. Formally, the multi-sets on the set A are of the form $A \to \mathbb{N}$. The multi-set $A(a_k) \mapsto n_k$ can be written either as a formal sum $n_1`a_1 + n_2`a_2 + \cdots$ or with the bracket notation $[\underbrace{a_1, a_1}_{n_1}, \underbrace{a_2, a_2, a_2}_{n_2}, \ldots]$. In the sum, items with zero multiplicity are omitted.

Arc expressions and high-level markings can be represented in a compact way as multi-sets. For singleton multi-sets, the brackets are often omitted: $[x]$ is written x.

There are several variants of high-level nets, such as predicate/transition nets (Pr/T nets) and coloured nets. In the following, we shall focus on *algebraic system nets* that are based on many-sorted algebras.

Understanding high-level nets (1/3)

The data model of high-level nets (e.g. algebraic, many sorted or coloured nets) greatly differs from place/transition nets. The major additions are:

- data types $\mathcal{D}$ (also known as colours or algebraic sorts)
- high-level places p that contain multi-sets of typed tokens
  - the marking (or state) of a high-level place p is a multi-set of the place's data type: $M(p) = (\mathcal{D} \rightarrow \mathbb{N})$
  - in the underlying low-level net, there is a place for each $\langle p, d \rangle$ pair, $d \in \mathcal{D}$

Understanding high-level nets (2/3)

The transitions in a high-level net have input and output arcs, just like transitions in low-level nets. High-level arcs are labelled with expressions that must evaluate to multi-sets that match the data types of the attached places.

Usually, the arc expressions refer to variables. In theory, the value of a variable is selected nondeterministically from the domain of the variable. In practice, for efficiency reasons, high-level Petri net analysis tools usually derive the values for variables from the input arc expressions and the markings of the input places.

In the underlying low-level net, there is a transition for each valuation that can be assigned to the variables of a high-level transition. It is possible to restrict these valuations by specifying conditions (guards or gates) on them, e.g. $x \neq y \wedge z < y$.*

*Although equality conditions, such as $x = y + z$, are possible, there often is a more efficient way: just eliminate one of the variables, e.g. replace all occurrences of x with $y + z$.

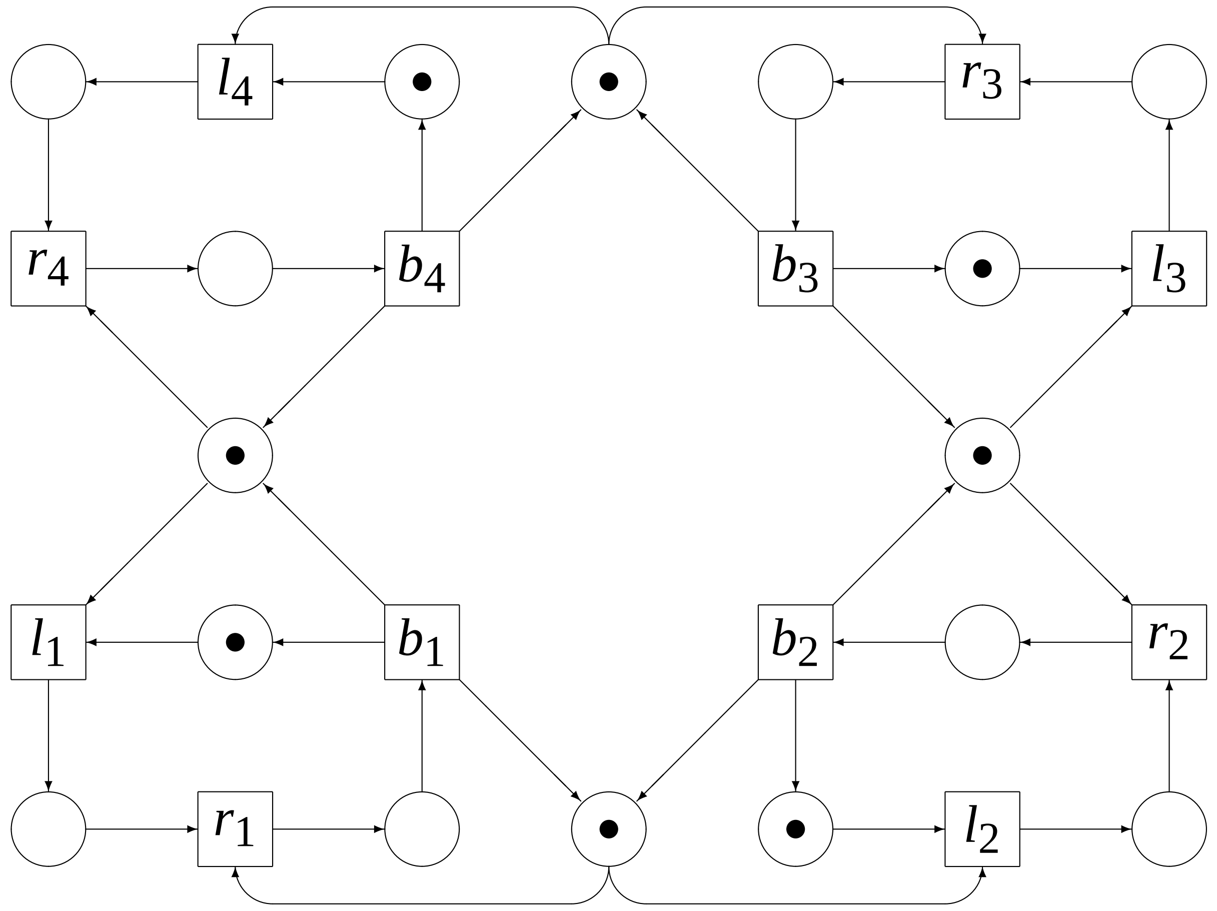
Understanding high-level nets (3/3)

High-level Petri nets can be viewed as computing systems that operate on shared data (the markings of the high-level places). In a sense, the high-level places can be understood as global variables and the high-level transitions as some kind of guarded assignment statements that operate on the data.

When studying or constructing Petri net models, it is often helpful to recognise different flows in the system being modelled: the control flows of the local entities, and various information flows. In many cases, these flows are clearly visible in the graphical representation of the net. Sometimes the places that represent a flow are folded.

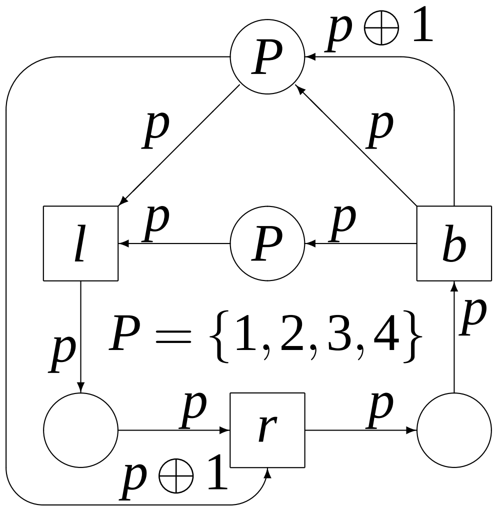
Next we shall introduce the system of *dining philosophers*. There are *n* philosophers and forks at a round table. Each philosopher needs two forks in order to eat: first the left one and then the right one.

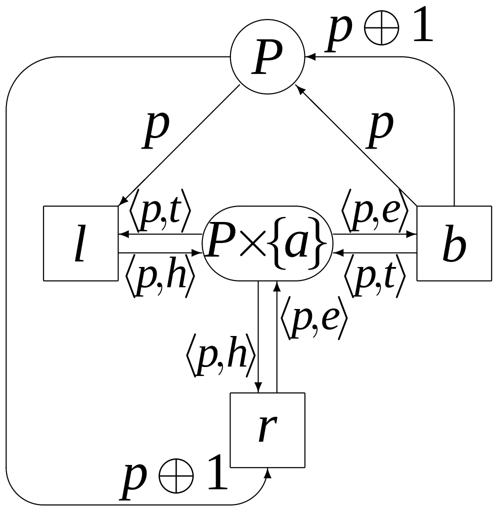
Folding nets: dining philosophers (1/2)



Folding nets: dining philosophers (2/2)

The model becomes much more compact when the places representing philosophers and forks are folded. The two flows of the system are still visible after the transformation:





It is a matter of taste whether folding the philosopher status places improves readability. The more places are folded, the more flexible is the operation of the transitions. In the extreme case, any model can be folded to one place and one transition.

Algebraic system nets (1/3)

In algebraic system nets, there are two kinds of data types: *basic sorts* and corresponding *multi-set sorts*. The variables of transitions belong to the basic sorts, while the arc inscriptions and markings are sorted over multi-sets. There are two predefined basic sorts: truth values $\mathbb{B} = \{\bot, \top\}$ and natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$. For simplicity, we shall identify algebraic sorts with their *support*, the set of values belonging to the sort.

In many-sorted algebras, functions are *operations* and expressions are *terms*, including

- variables (belonging to a sort): x, and
- operations, whose parameters are sorted terms: $f(), g(x,y), g(f(), g(y,x))$.

Constants are parameterless operations. For a term $t \in \mathbf{T}(X)$, the *evaluation* $\overline{\beta}(t)$ is derived from the assignment $\beta : x \mapsto \cdots$ as follows. The evaluation of a variable $x \in X$ is looked up from the assignment, $\overline{\beta}(x) = \beta(x)$, and the evaluation of an operation term $g(x,y)$ is obtained by applying the operation g to the parameters $\overline{\beta}(x)$ and $\overline{\beta}(y)$.

Algebraic system nets (2/3)

The quadruple $\Sigma = \langle N, \mathcal{A}, X, i \rangle$ is an *algebraic system net* if

- $N = \langle S, T, F \rangle$ is a finite net,
- the elements of S are variables over multi-set sorts,
- $\mathcal{A}$ is a many-sorted multi-set algebra ($BSIG$-algebra),
- X is a variable set of $\mathcal{A}$, $X \cap S = \emptyset$, and
- $i : S \cup T \cup F \to \mathbf{T}^{BSIG}(X)$ is an *inscription* of the net, as follows:
  - the *initial marking inscription* $i(s) \in \mathbf{T}^{BSIG}(\emptyset)$ has the sort of the place $s \in S$,
  - the inscription of the arc $f \in F$ ($f = \langle s, t \rangle$ or $f = \langle t, s \rangle$) *(arc inscription)* $i(f)$ is sorted according to the pre- or post-place s of the transition t, and
  - the *guard* $i(t)$ of a transition $t \in T$ is sorted over $\mathbb{B}$.

Algebraic system nets (3/3)

Let there be a net $N = \langle S, T, F \rangle$ and an algebraic system net $\Sigma = \langle N, \mathcal{A}, X, i \rangle$. Let $\beta_\emptyset : \emptyset \to \mathcal{D}$ be the empty assignment.

The *initial marking* of the net Σ is $M_0 : S \to (\mathcal{D} \to \mathbb{N})$ where $M_0(s) = \overline{\beta}_\emptyset(i(s))$.

The *pre- and post-substitutions* of the transition $t \in T$ are $t^-, t^+ : S \to \mathbf{T}(X)$:

$$t^-(s) = \begin{cases} i(s,t) & \text{if } \langle s,t \rangle \in F \\ [] & \text{otherwise,} \end{cases} \qquad t^+(s) = \begin{cases} i(t,s) & \text{if } \langle t,s \rangle \in F \\ [] & \text{otherwise.} \end{cases}$$

The firing rule of algebraic system nets

1. The β-*instance* t_β of a transition $t \in T$ is M-*enabled*, $M[t_\beta\rangle$ if $\forall s \in S : M(s) \geq \overline{\beta}(t^-(s))$ and $\overline{\beta}(i(t)) = \top$. (Firing condition)

2. An M-enabled transition instance t_β may *fire*, producing the *successor marking* $M' := [M\rangle t_\beta$ for which $M'(s) = M(s) - \overline{\beta}(t^-(s)) + \overline{\beta}(t^+(s))$. (Firing rule)

3. The *successor markings* of $\mathcal{M}$ are $\mathcal{M}[\rangle := \bigcup_{M \in \mathcal{M}} \bigcup_t \bigcup_\beta \left\{ \{[M\rangle t_\beta\} \;\middle|\; M[t_\beta\rangle \right\}$.

4. The *reachable markings* of $\mathcal{M}$ are $[\mathcal{M}\rangle := \mathcal{M}[\rangle^* = \mathcal{M} \cup \mathcal{M}[\rangle \cup \mathcal{M}[\rangle[\rangle \cup \ldots$.

Modelling with high-level nets

- Next, we shall model the alternating bit protocol in different ways.

- When modelling with high level nets, one should decide

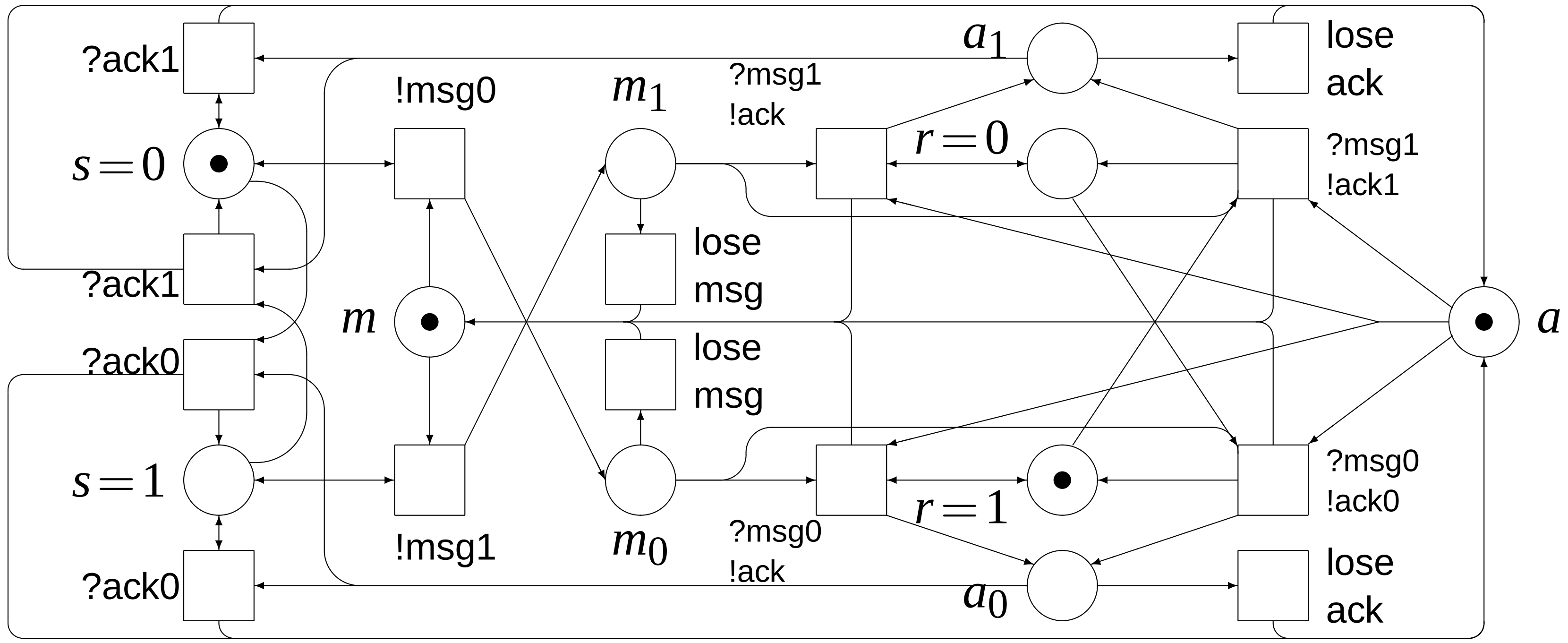
  1. which parts of the system are modelled with the net structure and

  2. which parts are modelled with the inscriptions.

- Since each place/transition system is also a high-level net, the model presented in the first lecture represents the case of modelling everything with the net structure.

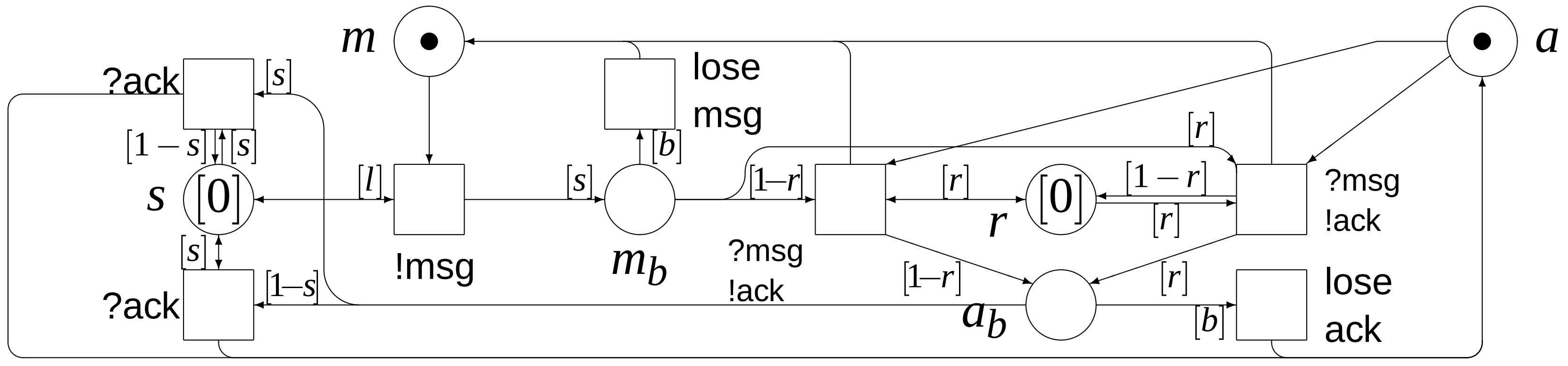
- In the other extreme, the net consists of one place and one transition.

Alternating bit protocol (1/4): place/transition system



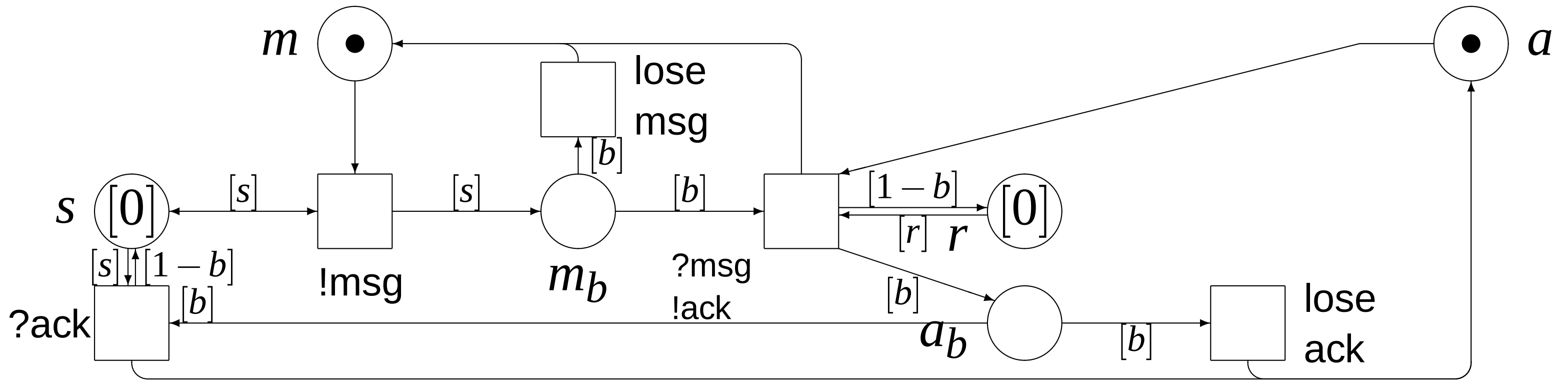
Alternating bit protocol (2/4): folding the bit

Clearly, the place/transition net contains symmetries. Let us fold the variable and channel places representing the state of the alternating bit.



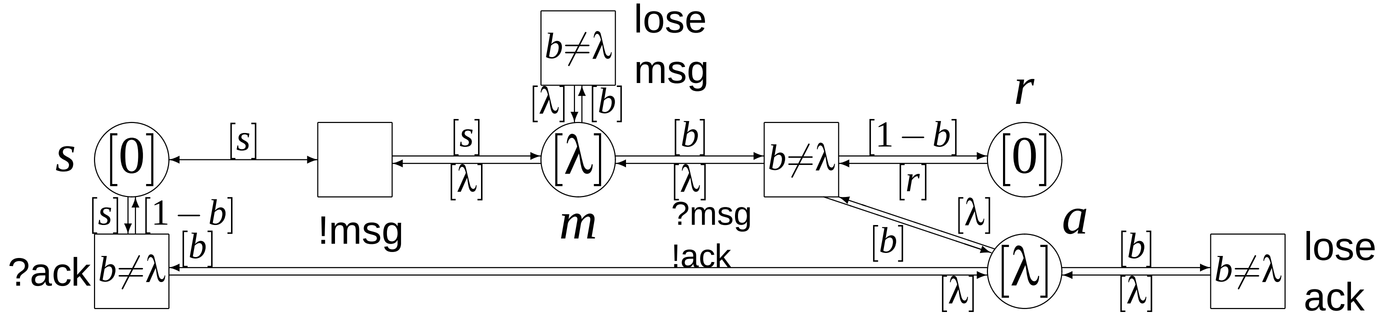
Alternating bit protocol (3/4): folding the reception

The transitions for receiving unexpected and expected messages can be folded:



Alternating bit protocol (4/4): folding the channels

Next, we fold the data channel (s_b and s) and the acknowledgement channel (k_b and k). The domain of these places is $\{\lambda, 0, 1\}$, where λ represents the empty channel.



The transitions must now be guarded. (It is customary to omit trivial guards $\top$.)

Unfolding a high-level net

An algebraic system net can be unfolded to a place/transition net.

1. A place s is unfolded by constructing a low-level place s_d for each possible token value $d \in \mathcal{D}_s$.

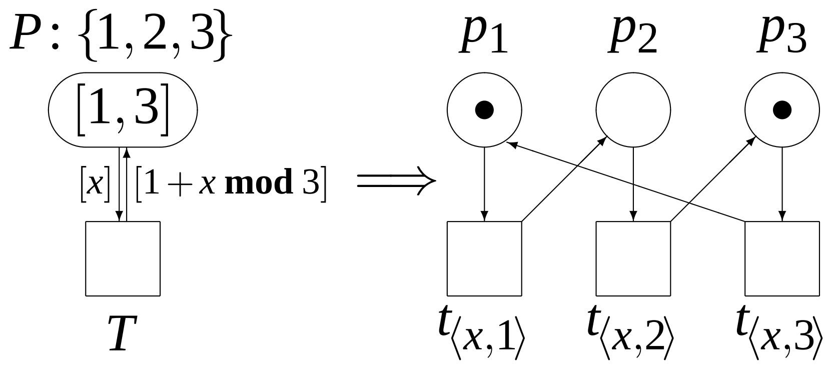
2. A transition t corresponds to a number of low-level transitions t_β; one for each valid assignment β such that $\overline{\beta}(i(t)) = \top$.

3. The arcs are unfolded by connecting the places s_d and transitions t_β: $W(s_d, t_\beta) = \overline{\beta}(t^-)(s)(d)$ and $W(t_\beta, s_d) = \overline{\beta}(t^+)(s)(d)$.

4. Finally, unconnected places are removed, and the initial marking is unfolded.

An unfolded net can be infinite or very large.

Unfolding a high-level net: example and remarks



In addition to unconnected places, an unfolded net may contain *dead transitions* that will never be enabled. Generally, they can only be removed by computing the reachability graph. The number of dead transitions can be reduced by defining redundant guards.

If there are no transition guards, the unfolded nets will be very symmetrical.

If there are large data domains or many variables, the unfolded net will become very large.