To maintain a finite set of link names we must realise when such a name becomes unused, that is the binding $\nu$ is obsolete. The positions are statically determinable and we mark them with additional calculus terms. Since this exceeds the MA syntax we define a refined MA calculus rMA and prove its bisimilarity with MA in Section 3.1.

We then define the one-bounded Petri net that reflects an arbitrary rMA term and show how to translate rMA into this so-called MA-PN in Section 3.2. Already this net is finite if and only if the MA term is bounded. Finally, we establish a second bisimulation in Section 3.3, this time linking rMA and MA-PN so that we can conclude the bisimilarity of MA and MA-PN by the transitivity of bisimulations. The construction is not yet polynomial since the possible name combinations in calls are exponential. This flaw is easily removed by a substitution net [MKH12] but we defer this addition to the appendix in order to avoid clutter.
3.1 From MA to rMA

The introduction of the used and unused commands exceeds the MA calculus's means. We therefore define the refined MA calculus, shortly called rMA calculus, which is capable of expressing the additional commands. We give and analyse the preprocessing algorithm which bridges the gap between the two formalisms. The proof that rMA introduces no new behaviour is done by establishing a bisimulation. It is our first step towards a bisimulation between MA and our Petri nets.

3.1.1 The Refined MA Calculus

Since rMA is a rather simple extension of MA we will only give new definitions where necessary and otherwise expect the given MA definition to extend to rMA as well.

Definition 3.1 (Syntax). $P$ is an rMA process term if it is built according to the following rules: $P := 0 \mid \pi.P \mid \tau.P \mid \nu.P \mid n[P] \mid \nu(n).P \mid \pi(n).P \mid K(\bar{a})$, where $\pi = \{in, out, open\}$ and $\tau = \{used(n), unused(n)\}$ for a name $n$.

This is essentially the same definition as Definition 1.2, only the new commands are added. We give the same binding priority as in, otherwise adopting the priorities given for the MA calculus. Additionally, we need to adapt the transition relation in order to deal with the new actions.

Definition 3.2 (Transition relation for rMA). The rMA transition relation is that of the MA calculus with the additional rules:

use: $\text{used}(n).P \rightarrow \tau.P$

unuse: $\text{unused}(n).P \rightarrow \tau.P$

Thus, the new commands are simple silent actions where the prefix is consumed but no modification on the tree is necessary. They play no role in rMA's semantics and do not influence the system behaviour. However, we know that the Petri net will take a quite different view on them.

We adapt the structural congruence given in Definition 1.3 accordingly by the addition of $P \equiv Q \Rightarrow \tau.P \equiv \tau.Q$ with $\tau = \{\text{used}(n), \text{unused}(n)\}$.

3.1.2 Preprocessing MA

The bridge towards rMA is the preprocessing algorithm for the introduction of used and unused commands to an MA process $P = (D, I)$. We consider the initial term $I$ as the right-hand side of a parameterless equation $D_i \in D$ so that we then simply deal with all defining equations. The set $rn(D_i)$ consists of those names which are freshly restricted in $D_i$.

We need a predicate $\text{occur}(n, P)$ which tests syntactically whether the name $n$ occurs in the term $P$. It can be implemented linearly in $\text{length}(P)$ which is always in $O(\text{length}(D_i))$. The actual algorithm $\text{pre}(P)$ is performed independently for each $D_i \in D$ by the following replacement:

for each $\pi.n.P$:
   if not occur(n, P) then set $\pi.n.\text{unused}(n).P$

for each $n[P]$:
   if not occur(n, P) then set $n[\text{unused}(n).P]$

for each $P[Q]$:
   for each $n \in \bar{D_i} \cup \text{rn}(D_i)$:
      if occur(n, P) and occur(n, Q) then set $\text{used}(n).(P|Q)$

for each $!P$:
   for each $n \in \bar{D_i} \cup \text{rn}(D_i)$:
      if occur(n, P) then set $\text{used}(n).P$

Since capability and spawn address a single name we can set at most one unused command behind each one of them. On the contrary, it may be necessary to put several used commands before a parallel composition or a replication – one for each name which is spread to $P$ and $Q$ or used in $P$. 