Annotating Coloured Petri Nets

Bo Lindstrøm and Lisa Wells

Department of Computer Science, University of Aarhus,
IT-Parken, Aabogade 34, DK-8200 Aarhus N, Denmark
{blind,wells}@daimi.au.dk

Abstract. Coloured Petri nets (CP-nets) can be used for several fundamentally different purposes like functional analysis, performance analysis, and visualisation. To be able to use the corresponding tool extensions and libraries it is sometimes necessary to include extra auxiliary information in the CP-net. An example of such auxiliary information is a counter which is associated with a token to be able to do performance analysis. Modifying colour sets and arc inscriptions in a CP-net to support a specific use may lead to creation of several slightly different CP-nets – only to support the different uses of the same basic CP-net. One solution to this problem is that the auxiliary information is not integrated into colour sets and arc inscriptions of a CP-net, but is kept separately. This makes it easy to disable this auxiliary information if a CP-net is to be used for another purpose. This paper proposes a method which makes it possible to associate auxiliary information, called annotations, with tokens without modifying the colour sets of the CP-net. Annotations are pieces of information that are not essential for determining the behaviour of the system being modelled, but are rather added to support a certain use of the CP-net. We define the semantics of annotations by describing a translation from a CP-net and the corresponding annotation layers to another CP-net where the annotations are an integrated part of the CP-net.

1 Introduction

Coloured Petri nets (CP-nets or CPNs) were formulated by Kurt Jensen [8, 9] with the primary purpose of specifying, designing, and analysing concurrent systems. The tools Design/CPN [3, 5] and CPN Tools [4] have been developed to give tool-support for creating and analysing CP-nets. Ongoing practical use of CP-nets and Design/CPN in industrial projects [10] have identified the need for additional facilities in the tools.

One industrial project described in [2] illustrated that CP-nets can be used for performance analysis by predicting the performance of a web server using a CPN model. As part of this project, the Design/CPN Performance Tool [12] was developed as an integrated tool extension supporting data collection during simulations. Later, work was done to extend and generalise these data collection facilities to serve as a basis for a common so-called monitoring framework [13]. Other projects have shown that visualisation of behaviour using so-called message sequence charts (MSCs) [7] is very useful in combination with CP-nets. As a consequence, a library [14] has been developed for creating MSCs during simulations. Other similar libraries are Mimic [15], which is used for visualisation, and Comms/CPN [6], which is used for communicating with external processes.

The fact that a CPN model can be used for several fundamentally different purposes like functional analysis, performance analysis, and visualisation means that it is desirable that the tool extensions and libraries can be used without having to modify the CPN model itself. It should be possible to use a CPN model, for e.g. performance analysis, without having to add extra places, transitions, and colour sets purely for the purpose of collecting data. Optimally, the auxiliary information should not be integrated into colour set and arc inscriptions of a CPN model, but should be kept separately, so that it is easy to disable this information if the CPN model is to be used for something else.

Up to this point it has only been partially possible to use a CPN model for different purposes without having to change the CPN model itself. With the current tools, it is indeed possible to do, e.g. performance analysis without adding transitions and places for the sole purpose of doing the performance analysis. Unfortunately however, it is often necessary to add extra information to colour sets and arc inscriptions to hold, e.g. performance-related information such as the time at which a certain event happened.

This paper presents work on separating auxiliary information from a CPN model by proposing a method which makes it possible to associate auxiliary information, called *annotations*, with tokens without modifying the colour sets of the CPN model. Annotations are pieces of information that are not essential for determining the behaviour of the system being modelled, but rather are added to support a certain use of the CPN model. A CP-net that is equipped with annotations is referred to as an *annotated CP-net*. In an annotated CP-net, every token carries a token colour, and some tokens carry both a token colour and an annotation. A token that carries both a colour and an annotation is called an *annotated token*. Just like a token value, an annotation may contain any type of information, and it may be arbitrarily complex.

Annotations are defined in *annotation layers*. Defining annotations in layers makes it possible to make modular definitions of both a CP-net and one or more layers of auxiliary information that can be used for varying purposes. By defining several different layers of annotations, it is possible to maintain several versions of a CP-net and thereby to use the same basic CP-net for various purposes by adding, removing, or combining annotation layers. An advantage of the annotation layers is that they are defined so that they affect the behaviour of the original CP-net in a very limited and predictable way. Every marking of an annotated CP-net is the same as a marking in the original CP-net, if annotations are removed.

In the following, we will assume that the reader is familiar with CP-nets as defined in [8]. The first half of this paper provides an informal introduction to annotations and an example of how annotations can be used in practice. The second half of the paper provides a formal definition of annotations and proof of the fact that annotations affect the behaviour of a CP-net in a very limited way. In this paper, we will only discuss how to annotate non-hierarchical, untimed CP-nets. However, timed and hierarchical CP-nets can also be annotated using similar techniques.

The paper is structured as follows. Section 2 presents the well-known resource allocation system CP-net, which will be used as a running example throughout the paper, and discusses existing ways of including auxiliary information in CP-nets. In Sect. 3 we informally introduce our proposal for how to annotate CP-nets. Section 4 discusses how multiple annotation layers can be used for visualisation using MSCs. In Sect. 5 we give the formal definitions for annotating CP-nets. Finally, in Sect. 6 we conclude and give directions for future work.

2 Motivation

It is seldom the case that the exact same CP-net can be used for a variety of different purposes, as it is frequently necessary to make small modifications to a CP-net in order to obtain a CP-net that is appropriate for a given purpose. Consider for example the resource allocation system that is found in Jensen's volumes on CP-nets [8, 9]. At least three variations of the resource allocation CP-net can be found in these volumes: a *basic* version (shown in Fig. 1) suitable for full state space analysis; an *extended* version (shown in Fig. 2) which is extended with cycle counters for the p and q processes; and a *timed* version with cycle counters and timing information which could be used for performance analysis.

The basic version in Fig. 1 purely models the basic aspects of the resource allocation system, and thereby only models the parts of the system that are common for any use of the CP-net. However, even for such a simple system as the resource allocation system, it is indeed necessary to have slightly different versions of the same CP-net in order to support different kinds of use. In other words, modifications of the basic CP-net are made only to support a certain use, and the modifications may limit other uses of the modified CP-net because the modifications may change the behaviour.

An example of a situation where the basic version does not contain sufficient information is when we need to be able to count how many cycles each of the p and q processes make in the resource allocation system. The extended CP-net in Fig. 2 shows how the basic CP-net can be extended with such auxiliary information. First of all, the colour set U has been extended to the product colour set UxI to include an integer for the counter in every process token. In addition, the arc inscriptions have been modified to pass on and to update the cycle counters. The cycle counters for the p and q processes are increased each time a p or q token passes the transition T5. The initial marking has also been modified to include the initial values of the cycle counters. Using this extended CP-net it is possible to determine the number of cycles a process has completed by inspecting the counter of the corresponding tokens.

The version extended with the cycle counters is not useful for all kinds of analysis. This is due to the fact that the cycle counters for the p and q processes increase each time the p and q tokens pass

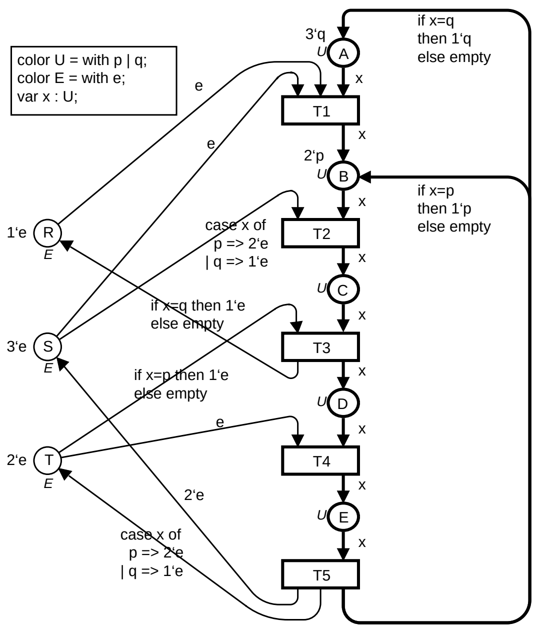


Fig. 1. The basic CP-net for the resource allocation system.

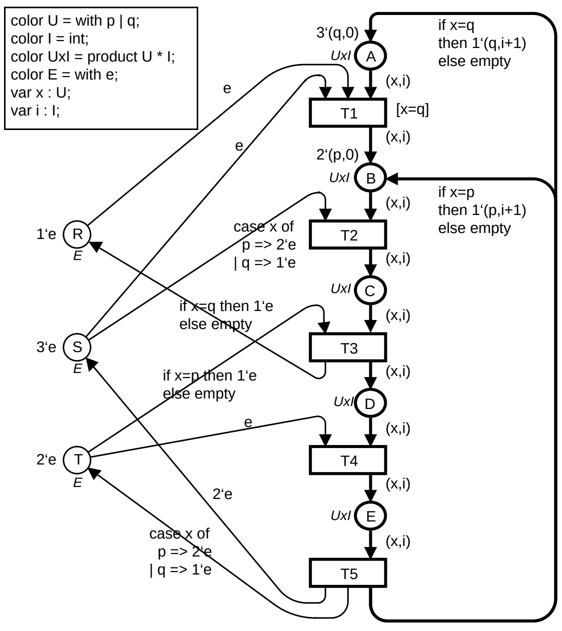


Fig. 2. The CP-net for the resource allocation system extended with a counter.

transition T5, thus resulting in an infinite state space. Therefore, the extended version with cycle counters may be inappropriate for certain kinds of state space analysis. The effect of the cycle counters on the state space can be factored out using equivalence classes, however, it may be annoying to have to remember to manually take care of such auxiliary information before doing state space analysis. In contrast, the state space for the basic CP-net without the cycle counters is finite. This means that the full state space can be generated and analysed, e.g. to prove that the system never reaches a deadlocked state.

Analysing the performance of the resource allocation system is another kind of analysis that requires auxiliary information to be maintained for the tokens in the CP-net. The timed CP-net from [9] could be used to measure the average processing times for each of the two processes. This timed CP-net can be created by modifying the CP-net in Fig. 2 by changing the colour set UxI to a timed colour set, and by adding an auxiliary component to the colour set to be used for recording the time when a process restarts a cycle,[1] i.e the time at which a q process is removed from place A or the time at which a p process is removed from place B. This value can then be used to calculate the processing time for a given process when it passes the T5 transition. If the timed CP-net should be used for a purpose where the auxiliary information should be ignored, it should often be removed. In the tool Design/CPN, it is easy to disable time, i.e. to consider a timed CP-net as an untimed CP-net. However, auxiliary components that have been added to the colour sets also need to be removed by manually modifying the colour sets and arc inscriptions.

From the examples presented above it should now be clear that when using CP-nets for different purposes it is often necessary to maintain different versions of a CP-net with slightly different behaviour. The reason for maintaining different versions is, as mentioned, that it may be necessary to be able to include auxiliary information in tokens. However, the auxiliary information may be extraneous or even disastrous for other uses, e.g. consider the effects of the cycle counters on the size of the state space. Including extra information in a CP-net often requires modification of colour sets, arc expressions, and initialisation expressions.

[1] The colour set U could be modified to consist of pairs (u,t) where u∈U is a process and t∈TIME is the time at which the process started processing.

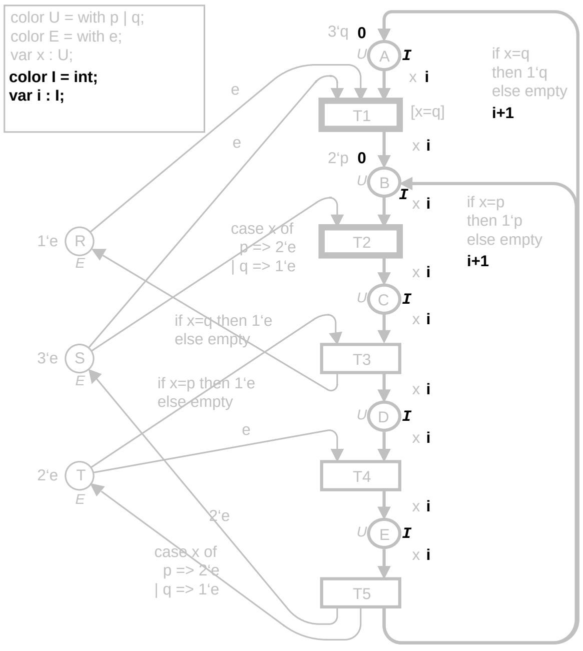
3 Informal Introduction to Annotated CP-nets

In this section we will informally present a method for augmenting tokens in a CP-net with extra or auxiliary information that affects the behaviour of the CP-net in a very limited and predictable manner. To do this we introduce the concept of an *annotation* which is very similar to a token colour in that an annotation is an additional data value that can be attached to a token. An *annotation layer* is used to define annotations and how these annotations are to be associated with tokens in a particular CP-net. An annotation layer cannot be defined independently from a specific CP-net. Therefore, it is always well-defined to refer to the unique CP-net for which an annotation layer is defined. We will refer to this unique CP-net as the *underlying* CP-net of an annotation layer. An *annotated CP-net* is a pair consisting of an annotation layer and its underlying CP-net. We define the semantics of annotations by describing a translation from an annotated CP-net to a CP-net without annotations, referred to as the *matching CP-net*. In practice, the annotations are integrated into the matching CP-net when the translation is made. Section 3.1 gives an informal introduction to annotations and annotation layers. Section 3.2 describes the intuition of how to translate an annotated CP-net to a matching CP-net. Section 3.3 discusses the behaviour of the matching CP-net, and it discusses how the behaviour of the matching CP-net is similar to the behaviour of the underlying CP-net. The formal definition of annotated CP-nets follows in Sect. 5.

3.1 Annotation Layer

To get an intuitive understanding of how annotations can be used, let us see how the cycle counters that were discussed in Sect. 2 can be added as annotations. Recall that Fig. 2 shows how the CP-net from Fig. 1 can be modified to include the cycle counters as part of the token colours.

Figure 3 contains an annotated CP-net for the basic CP-net for the resource allocation system from Fig. 1. In Fig. 3 the elements from the annotation layer are shown in black, whereas the underlying CP-net is shown in grey. The annotation layer contains *auxiliary declarations* and *auxiliary net inscriptions*, where the auxiliary net inscriptions consist of auxiliary arc expressions, auxiliary colour sets, and auxiliary initialisation expressions.



color U = with p | q;
color E = with e;
var x : U;
color I = int;
var i : I;
3'q 0
2'p 0
A
B
C
D
E
R
S
T
U
I
T1
T2
T3
T4
T5
[x=q]
x i
i+1
if x=q then 1'q else empty
if x=p then 1'p else empty
e
1'e
3'e
2'e
case x of p => 2'e | q => 1'e
if x=q then 1'e else empty
if x=p then 1'e else empty

Fig. 3. Annotated CP-net for the resource allocation system.

The colour set I is declared in the first line of the auxiliary declarations. Places A, B, C, D and E have auxiliary colour set I which means that tokens on these places will be annotated tokens that carry integer annotations. A token that carries the annotation n has completed the cycle n times. Places with auxiliary colour sets are called *annotated places*. The token value for a token on an annotated place has both a token colour and an annotation. Not all places will contain annotated tokens, therefore, some places will not have an associated auxiliary colour set.

All of the annotated places that have an initialisation expression in the underlying CP-net must also have an *auxiliary initialisation expression* in the annotation layer. Places A and B have the auxiliary initial expression 0. This expression means that all tokens on places A and B will have annotation 0 in the initial marking.

All arcs that are connected to annotated places have an *auxiliary arc expression*. In Fig. 3, most auxiliary arc expressions consist of the variable i which has type I. Variable i is declared in the annotation layer, therefore, it may only be used within the annotation layer, i.e. it cannot be used in the underlying CP-net. In contrast, variables, colour sets, functions, etc. that are declared in the underlying CP-net may be used both in the underlying CP-net and in the annotation layer. However, certain conditions must be fulfilled in order to ensure that using the same elements in both the annotation layer and the underlying CP-net does not affect the behaviour of the underlying CP-net. These conditions will be discussed further in Sect. 5.2.

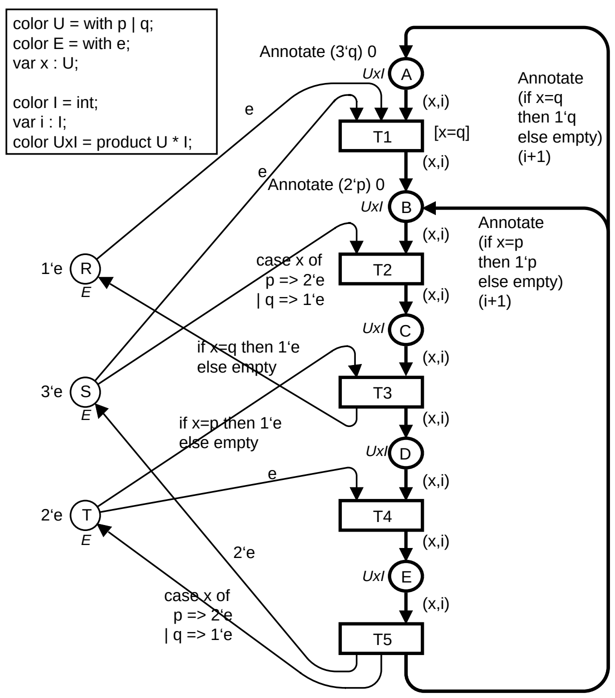
Let us consider the intuition behind the auxiliary arc expressions on the arcs surrounding transition T5. In the underlying CP-net, T5 can occur whenever there is one token on place E, and this must still be true in an annotated version of the CP-net. Informally, the interpretation of the two types of arc expressions surrounding T5 is that when transition T5 occurs with, e.g. binding $<$x=q, i=5$>$, one token with colour q and annotation 5 will be removed from place E. One token with colour q and annotation 5+1=6 will be added to place A, and the empty multi-set of annotated tokens will be added to place B. On the other hand, if T5 occurs with binding $<$x=p, i=3$>$, then one token with colour p and annotation 3+1=4 will be added to place B, and no tokens will be added to place A. In both bindings, multi-sets of (non-annotated) e tokens are also added to places T and S, which are non-annotated places.

The intuition behind the auxiliary inscriptions that have been discussed until now is fairly straightforward. There are, however, some restrictions on the kinds of auxiliary arc expressions that are allowed in order to ensure that annotations have only limited influence on the behaviour of the underlying CP-net. All of the auxiliary arc expressions on arcs from annotated places to transitions consist only of variables, and this is not accidental. For example, the auxiliary arc expression i on the arc from C to T3 must not be replaced with, e.g. the constant 4, which would require that when removing a token from C the annotation must be 4. Allowing such an auxiliary arc expression would mean that the behaviour of the matching CP-net and the underlying CP-net would no longer be similar. Sections 5.2 and 5.3 discuss the restrictions about which kinds of auxiliary arc expressions are allowed.

3.2 Translating an Annotated CP-net to a Matching CP-net

Rather than defining the semantics for annotated CP-nets, we will define the semantics of annotations by describing how an annotated CP-net can be translated to an ordinary CP-net, which is referred to as the *matching* CP-net. The discussion above should have provided a sense of what kinds of annotations the tokens should have and of how an annotated CP-net for the resource allocation system should behave. The annotation layer (referred to as $\mathcal{A}$ and shown in black in Fig. 3) and the underlying CP-net (referred to as CPN and shown in Fig. 1) constitute an annotated CP-net for the resource allocation system. In this section, we will show how the various auxiliary inscriptions from $\mathcal{A}$ and the inscriptions from CPN are translated to inscriptions in the matching CP-net (referred to as CPN* and shown in Fig. 4). The general rules for translating an arbitrary annotation layer and its underlying CP-net are presented in Sect. 5.3. In the following, we shall say that a place/arc is annotated/non-annotated in a matching CP-net (like Fig. 4) if it is annotated/non-annotated in the corresponding annotated CP-net (like Fig. 3).

The rules are simple for translating an annotated CP-net to a matching CP-net. CPN and CPN* have the same net structure. The colour sets for non-annotated places in CPN* are unchanged with respect to CPN; in the example, the colour sets for places R, S and T are unchanged. The colour sets for the annotated places are now product colour sets which are products of the original colour sets and the auxiliary colour sets. The colour set for annotated places A-E in CPN* is UxI which is a product of colour set U (the colour



color U = with p | q;
color E = with e;
var x : U;
color I = int;
var i : I;
color UxI = product U * I;
Annotate (3'q) 0
UxI
A
(x,i)
T1
[x=q]
(x,i)
Annotate (if x=q then 1'q else empty) (i+1)
e
e
Annotate (2'p) 0
B
(x,i)
T2
Annotate (if x=p then 1'p else empty) (i+1)
1'e
R
E
case x of p => 2'e | q => 1'e
C
if x=q then 1'e else empty
T3
3'e
S
if x=p then 1'e else empty
D
T4
2'e
T
2'e
E
case x of p => 2'e | q => 1'e
T5

Fig. 4. Matching CP-net for the resource allocation system.

set of places A-E in CPN) and auxiliary colour set I (the auxiliary colour sets from $\mathcal{A}$). The tokens on an annotated place p in CPN* are said to have an annotated colour (c,a), where c is a colour (from the colour set of p in CPN), and a is an annotation (from the auxiliary colour set of p in $\mathcal{A}$). The set of colour sets for CPN* is the union of the set of colour sets from CPN, the set of auxiliary colour sets from $\mathcal{A}$, and the set of product colour sets for annotated places.

In the previous section, the intuitive meaning of several auxiliary expressions was that a given annotation should be added to all elements in a multi-set of colours. This is the meaning of, for example, all of the auxiliary initial expressions. Let us define a function `Annotate` that, given an arbitrary multi-set and an arbitrary annotation, will annotate all of the elements in the multi-set with the annotation. This function is used in several net inscriptions in CPN*.

Let us consider how the initialisation expressions are created for CPN*. The initialisation expressions for non-annotated places (R, S and T) are unchanged, and evaluating these expressions yields non-annotated multi-sets (`` 1`e,3`e ``, and `` 2`e `` respectively). The initial markings for annotated places in CPN* must be multi-sets of annotated colours. Place A has the initialisation expression `` Annotate (3`q) 0 ``, which evaluates to `` 3`(q,0) `` tokens which correspond to the desired multi-set of three annotated tokens, each with colour q and annotation 0. Note, in particular, that if the annotations are removed from the multi-set `` 3`(q,0) ``, then we obtain the multi-set `` 3`q `` which is exactly the multi-set that is obtained when evaluating the initialisation expression for A in the underlying CP-net. Similarly, place B has an initial marking of two tokens, each with colour p and annotation 0. The initial markings of the remaining places are empty.

The arc expressions of CPN and the auxiliary arc expressions of $\mathcal{A}$ are combined in a similar manner to create arc expressions for CPN*. If the type of an arc expression in CPN, `expr`, is a single colour, then the arc expression in CPN* is the pair `(expr, aexpr)`, where `aexpr` is the auxiliary arc expression in $\mathcal{A}$. The arc expressions for most annotated arcs in Fig. 4 have this form. When the type of an arc expression is a multi-set of colours, then the arc expression for CPN* is `Annotate expr aexpr`. The arc expressions for the arcs from transition T5 to places A and B were created in this manner. The next section discusses how the behaviour of the matching CP-net is similar to the behaviour of the underlying CP-net.

3.3 Behaviour of Matching CP-nets

In a matching CP-net some places contain annotated tokens, other places contain non-annotated tokens, and occurrences of binding elements can remove and add both regular, non-annotated tokens and annotated tokens. Figure 5 shows a marking of the matching CP-net, CPN*. The marking of place A contains two tokens – one with colour (q,4), the other with colour (q,5). This corresponds to a marking in the annotated CP-net of Fig. 3 where A has one token with colour q and annotation 4 and another token with colour q and annotation 5. Similarly, place B contains three tokens with colours (p,5), (p,6) and (q,1). This corresponds to a marking in the annotated net where B has one token with colour p and annotation 5, another token with colour p and annotation 6, and a third token with colour q and annotation 1. Finally, places S and T each contain two non-annotated tokens with colour e.

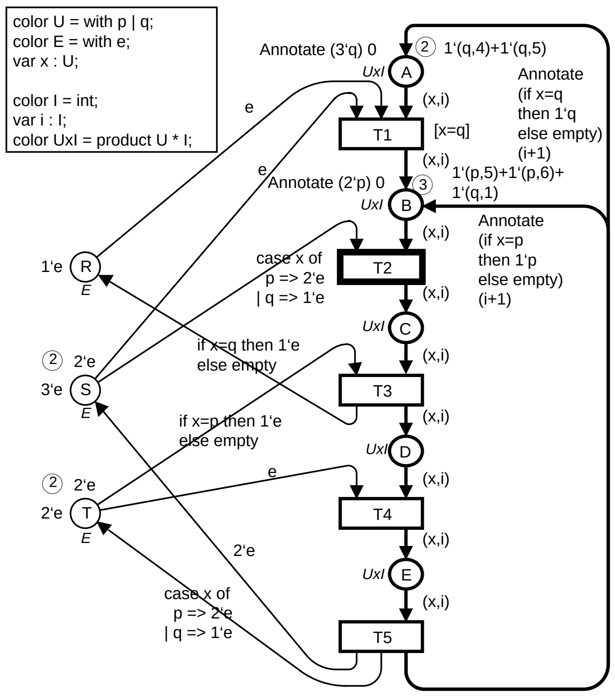


Fig. 5. Marking of the matching CP-net for the resource allocation system.

We say that the behaviour of the matching CP-net *matches* the behaviour of its underlying CP-net. Informally this means that every occurrence sequence in the matching CP-net is also an occurrence sequence in the underlying CP-net if annotations are ignored. Furthermore, for every occurrence sequence in the underlying CP-net, it is possible to find at least one matching occurrence sequence in the matching CP-net which is identical to the occurrence sequence from the underlying CP-net when annotations are ignored. Consider, for example, a marking, M, of the basic CP-net from Fig. 1, in which there are two e tokens on places S and T, two q tokens on place A, and two p tokens and one q token on place B. The binding element (T2, ⟨x=p⟩) is enabled in M. The marking of the matching CP-net in Fig. 5 is the same as marking M if annotations are ignored. The occurrence of either (T2, ⟨x=p, i=5⟩) or (T2, ⟨x=p, i=6⟩) in the matching CP-net will result in markings that are equal M′ where M[(T2, ⟨x=p⟩)⟩M′ when annotations are ignored. A formal definition of matching behaviour can be found in Sect. 5.4.

4 Using Annotation Layers in Practice

This section discusses how to use several annotation layers for the basic CP-net of the resource allocation system presented in Fig. 1 in Sect. 2. The purpose of this section is to illustrate that multiple annotation layers can be added on top of each other without changing the original model, and to illustrate some of the uses of annotations. We will discuss an example of how annotations can be used for visualising simulation results. In particular we will consider how message sequence charts (MSCs) can be created using annotations.

A MSC can be used, e.g. to visualise the use of resources. Figure 6 depicts a MSC for the basic CP-net of the resource allocation system. The MSC contains two vertical lines which represent the activities of allocating and deallocating resources in the resource allocation system. An arrow represents the dependency between the allocation and deallocation of an S or an R resource.

The MSC in Fig. 6 visualises a sequence of allocations of resources by the p and q processes. The arrows for p processes are dashed. The MSC shows that first the q process makes a full cycle where it first allocates an R and an S resource when T1 occurs, and an additional S resource when T2 occurs. The R resource is deallocated when T3 occurs, and the two S resources are deallocated when T5 occurs. The last five arrows show a situation where two p processes interleave with a q process. First the q process allocates an R and an S resource, but then two cycles of p processes appear (the two dashed arrows) before the q process continues the cycle. This interleaving is explicitly visualised by the arrows started by T1 and ended by T3 and T5, and crossing the arrows representing the two p processes.

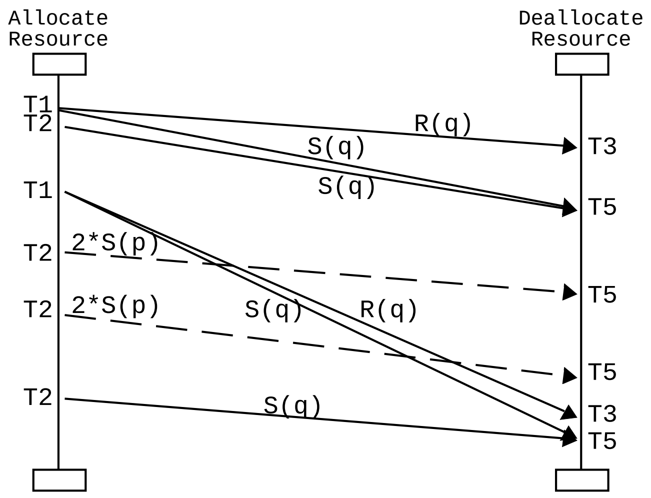


Fig. 6. Message sequence chart for the resource allocation system.

A MSC can be generated automatically from a simulation of a CP-net. However, first it is necessary to specify which occurrences of binding elements in the CP-net should generate which arrows in the MSC. Normally, an arrow in a MSC is created when a single transition occurs. However, arrows as illustrated above correspond to two events: one for creating the start-point and one for creating the end-point of an arrow. Such arrows are defined by means of these two points. First the start-point of the arrow is given. Then some other events may appear, and then the event leading to ending the arrow is given. For CP-nets this means that the occurrence of one transition may define the start-point of an arrow while the occurrence of another transition may define the end-point of an arrow. We call such arrows *two-event arrows*.

When using two-event arrows, it is often necessary to annotate a token to hold information of which arrows have been started but not ended yet. In other words, a token must hold the arrow-id of the start-point of the arrow when the first transition occurs and keep it until a transition supposed to end the arrow consumes the token. To avoid modifying the colours of the the CP-net, annotations can be used. Figure 7 depicts how the basic CP-net for the resource allocation system can be annotated to generate the MSC in Fig. 6. The contents of the annotation layer are shown in black, while the underlying CP-net is shown in grey. The annotation colour `MSC` is an integer which has the purpose of holding the start-point id of an arrow, while the annotation colour `MSCs` is a list of MSC ids. We do not give the details of the functions `msc_start` and `msc_stop` here, however, they are used to set the start-point and end-point of each arrow. In addition each of the functions return a list of arrow-ids of the non-stopped arrows. This list becomes an annotation for the underlying colour. As this example illustrates, we allow auxiliary arc expressions to have side effects. However, the side effects may not affect the behaviour of the underlying CP-net.

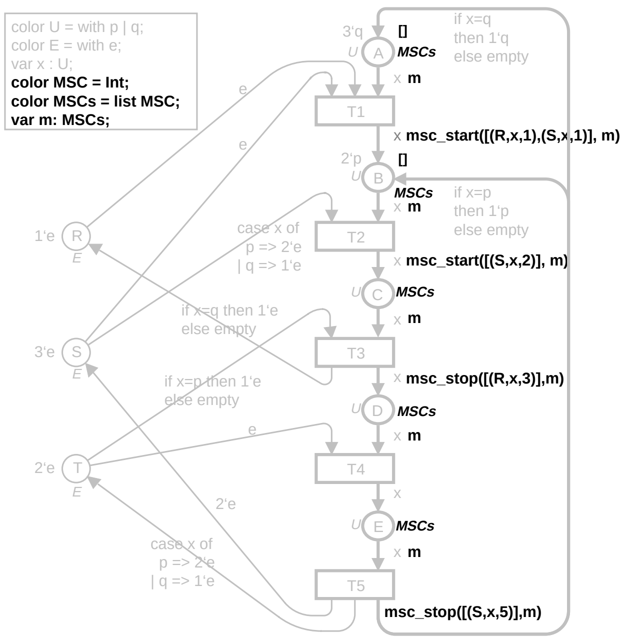


Fig. 7. Resource allocation system with MSC annotation layer.

The annotation layer in Fig. 3 from Sect. 3.1 adds a cycle counter to the CP-net. The value of the cycle counter could be included on the arrows in Fig. 6 in addition to the type of resource and process. To obtain this, an annotation layer that resembles the MSC Annotation Layer in Fig. 7 can be added on top of the cycle counter annotation layer in Fig. 3. We will refer to this new annotation layer as Cycle MSC Annotation Layer. In the Cycle MSC Annotation Layer it is possible to refer to the annotations of the Cycle Counter Annotation Layer from the MSC Annotation Layer.

Figure 8 depicts some of the possible ways to add annotation layers on top of each other. Notice that an alternative to adding the Cycle MSC Annotation Layer on top of the Cycle Counter Annotation Layer, is to add the original MSC Annotation Layer on top of the Cycle Counter Annotation Layer. This makes sense even though the annotations in Cycle Counter Annotation Layer are not used in the MSC Annotation Layer.

If we had not been able to use an annotation layer for creating the MSC, we would have had to create a new CP-net by adding and modifying the colours of the basic CP-net. For example, the colour sets of the places A, B,C, D, and E should also hold the `MSCs` colour set. In addition, so-called code-segments possibly had to be added to execute the the `msc_start` and `msc_stop` function calls, and the arc-expressions had to be modified to include the MSC variable `m`. In other words, we had to modify the CP-net model itself to generate the MSCs. If the information for updating MSCs is included directly in the CP-net, then it would be difficult to disable the updating of the MSCs, and there is no guarantee that the modifications would not affect the behaviour of the underlying CP-net in unexpected ways.

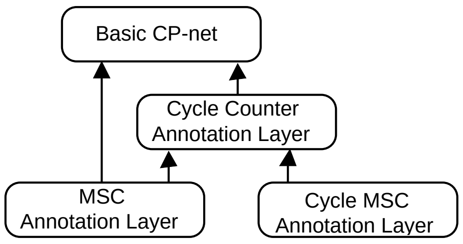


Fig. 8. Structure of annotation layers for a basic CP-net.

5 Formal Definition of Annotated CP-nets

In this section, we will formally define annotated CP-nets. We will start by introducing some new terminology. We will then define annotation layers, and we will discuss how an annotation layer and a CP-net can be translated into a matching CP-net. We want to define the annotation rules so that they are straightforward to use and understand. To achieve this it turns out to be convenient only to allow annotation of those input arcs of transitions where the arc expressions are uniform with multiplicity one (i.e. always evaluate to a single token colour). For output arcs there are no similar restrictions. If an input arc expression is uniform with multiplicity larger than one, it is usually easy to split the arc into a number of arcs that each have multiplicity one. The requirement can be formally expressed as:

Requirement 1. Let CPN=(Σ, P, T, A, N, C, G, E, I) be a CP-net as defined in Def. 2.5 in [8]. Let $P_{\mathcal{A}} \subseteq P$ be the set of *annotated places*. The following must hold in order to be able to annotate CPN:

$\forall\ p \in P_{\mathcal{A}}: \forall\ a \in A$ such that N(a)=(p,t), E(a) must be uniform with multiplicity 1.

This requirement may seem very restrictive. However, in our experience, the kinds of arc inscriptions that are currently not possible to annotate are rarely used in practice. Therefore, the definitions presented here should prove to be useful for annotating many of the CP-nets that are used in practice.

Section 5.1 introduces terminology regarding multi-sets of annotated colours. Section 5.2 defines annotation layers by describing the auxiliary net inscriptions that are allowed in the annotation layer. Section 5.3 presents rules for translating a CP-net and an annotation layer into the matching CP-net, and it discusses the relationship between markings, binding elements, and steps in a matching CP-net and its underlying CP-net. Section 5.4 defines matching behaviour. Finally, Sect. 5.5 discusses the use of multiple annotation layers.

5.1 Multi-sets of Annotated Colours

In the previous section we used expressions such as: `` 1`(p,5)+1`(p,6)+1`(q,1) `` to denote the marking of annotated places[2] in a matching CP-net. This indicates that the marking consists of three tokens with colours `(p,5)`,`(p,6)` and `(q,1)`. However, within the context of annotated CP-nets, this marking can also be interpreted to represent a multi-set of annotated tokens: two tokens with colour `p` and annotations `5` and `6`, and one token with colour `q` and annotation `1`. Multi-sets of annotated elements are ordinary multi-sets[3] of so-called *annotated elements*.

[2] The first paragraph in Sect. 3.2 explains what we mean when we refer to an annotated place in a matching CP-net.
[3] Multi-sets as defined in Def. 2.1 in [8]

Definition 1. For a non-empty set of elements, S, and a non-empty set of annotations, AN, an *annotated element* (from S) is a pair (s,a), where s∈S and a∈AN. π((s,a))=s is the projection of the annotated element (s,a) onto the non-annotated element s.

A *multi-set of annotated elements* over S×AN is a multi-set over S×AN.

If am is a multi-set of annotated elements (of S), then am *determines* an ordinary (non-annotated) multi-set $\mathtt{am}_\pi$ over S, where $\mathtt{am}_\pi$(s) = $(\sum_{a\in AN} \mathtt{am}(s,a))`\, s$. π(am) is the *projection* of am onto the non-annotated multi-set determined by am.

If am is multi-set over S×AN, m is a multi-set over S, and π(am)=m, then am is said to *cover* m, and we say that m is *covered* by am.

In Sect. 3.2 we informally defined the function Annotate that will add a given annotation to all elements in a given multi-set. Let us now formally define Annotate.

Definition 2. Given an annotation a∈AN and a multi-set m=$\sum_{s\in S} \mathtt{m}(s)`\, s$ over a set S, the function Annotate is defined to be:

$$\mathtt{Annotate}\;\mathtt{m}\;\mathtt{a} = \sum_{s\in S} \mathtt{m}(s)`(s,\mathtt{a})$$

which is a multi-set of annotated elements of S, i.e. a multi-set with type $(S\times AN)_{MS}$.

As a consequence of Defs. 1 and 2, $\pi(\mathtt{Annotate}\;\mathtt{m}\;\mathtt{a}) = \mathtt{m}$, for all multi-sets m and all annotations a.

5.2 Annotation Layer

We are now ready to define an annotation layer. An annotation layer is used solely to determine how to add annotations to tokens for a subset of the places in a CP-net. An annotation layer consists of elements that are similar to their counterparts in CP-nets. An annotation layer contains auxiliary net inscriptions, and each auxiliary net inscription is associated with an element of the net structure of the underlying CP-net. When translating an annotation layer and its underlying CP-net to the matching CP-net, these auxiliary net expressions will be combined with their counterparts from the underlying CP-net to create colour sets, initialisation expressions and arc expressions for the matching CP-net. There are, however, additional requirements for each of the concepts. An explanation of each item in the definition is given immediately below the definition. A similar remark applies for many of the other definitions in this paper.

Definition 3. Let CPN=(Σ, P, T, A, N, C, G, E, I) be a CP-net. An *annotation layer* for CPN is a tuple $\mathcal{A}$=($\Sigma_\mathcal{A}$, $P_\mathcal{A}$, $A_\mathcal{A}$, $C_\mathcal{A}$, $E_\mathcal{A}$, $I_\mathcal{A}$) where

i. $\Sigma_\mathcal{A}$ is a finite set of non-empty sets, called *auxiliary colour sets*, where $\Sigma\subseteq\Sigma_\mathcal{A}$.
ii. $P_\mathcal{A}\subseteq P$ is a finite set of *annotated places*.
iii. $A_\mathcal{A}\subseteq A$ is the finite set of *annotated arcs*, where $A_\mathcal{A}$= A($P_\mathcal{A}$).
iv. $C_\mathcal{A}$ is an *auxiliary colour* function. It is a function from $P_\mathcal{A}$ into $\Sigma_\mathcal{A}$.
v. $E_\mathcal{A}$ is an *auxiliary arc expression* function. It is defined from $A_\mathcal{A}$ into expressions such that:
$\forall\, a\in A_\mathcal{A}$: Type($E_\mathcal{A}$(a))=$C_\mathcal{A}$(p(a)) $\wedge$ Type(Var($E_\mathcal{A}$(a)))$\subseteq\Sigma_\mathcal{A}$
vi. $I_\mathcal{A}$ is an *auxiliary initialisation* function. It is a function from $P_\mathcal{A}$ into closed expressions such that:
$\forall\, p\in P_\mathcal{A}$: Type($I_\mathcal{A}$(p))=$C_\mathcal{A}$(p).

i. The set of *auxiliary colour sets* is the set of colour sets that determine the types, operations and functions that can be used in the auxiliary net inscriptions. The auxiliary colour sets determine the type of annotations that the tokens on the annotated places carry. All colour sets from the underlying CP-net can be used as auxiliary colour sets. Additional auxiliary colour sets may be declared within an annotation layer.

ii. The set of annotated places are the only places that are allowed to contain annotated tokens.

iii. The annotated arcs are exactly the surrounding arcs for the places in $P_\mathcal{A}$.

iv. The *auxiliary colour function*, $C_{\mathcal{A}}$, is a function from $P_{\mathcal{A}}$ into $\Sigma_{\mathcal{A}}$, and is defined analogously to the colour function for CP-nets. Thus, for all $p \in P_{\mathcal{A}}$, $C_{\mathcal{A}}(p)$ is the auxiliary colour set of p.

v. Auxiliary arc expressions are only allowed to evaluate to a single annotation of the correct type. If the arc expression of an arc is missing in CPN, then we require that its auxiliary arc expression is also missing in $\mathcal{A}$.

vi. The auxiliary initialisation function maps each annotated place, p, into a closed expression which must be of type $C_{\mathcal{A}}(p)$, i.e. a single annotation from $C_{\mathcal{A}}(p)$. If the initial expression of place p is missing in CPN, then we require that its auxiliary initial expression is also missing in $\mathcal{A}$.

5.3 Translating Annotated CP-nets to Matching CP-nets

We will now define how to translate an annotated CP-net, (CPN, $\mathcal{A}$), to a new CP-net, CPN*, which is called a matching CP-net. CPN* and CPN have the same net structure. Net inscriptions for non-annotated places, non-annotated arcs, and transitions in CPN* are unchanged with respect to CPN. In contrast, net inscriptions for annotated places and annotated arcs in CPN* are obtained by combining net inscriptions from CPN with their counterpart auxiliary net inscriptions in $\mathcal{A}$. A matching CP-net is defined below.

Definition 4. Let (CPN, $\mathcal{A}$) be an annotated CP-net, where CPN=(Σ, P, T, A, N, C, G, E, I) and $\mathcal{A}$=($\Sigma_{\mathcal{A}}$, $P_{\mathcal{A}}$, $A_{\mathcal{A}}$, $C_{\mathcal{A}}$, $E_{\mathcal{A}}$, $I_{\mathcal{A}}$) is a annotation layer. We define the *matching CP-net* to be CPN*=(Σ^*, P*, T*, A*, N*, C*, G*, E*, I*) where

i. $\Sigma^* = \Sigma_{\mathcal{A}} \cup \{C(p) \times C_{\mathcal{A}}(p) \mid p \in P_{\mathcal{A}}\}$.
ii. $P^* = P$
iii. $T^* = T$
iv. $A^* = A$
v. $N^* = N$
vi. $C^*(p) = \begin{cases} C(p) & \text{if } p \notin P_{\mathcal{A}} \\ C(p) \times C_{\mathcal{A}}(p) & \text{if } p \in P_{\mathcal{A}} \end{cases}$
vii. $G^* = G$
viii. $E^*(a) = \begin{cases} E(a) & \text{if } a \notin A_{\mathcal{A}} \\ \text{Annotate } E(a)\ E_{\mathcal{A}}(a) & \text{if } a \in A_{\mathcal{A}} \end{cases}$
ix. $I^*(p) = \begin{cases} I(p) & \text{if } p \notin P_{\mathcal{A}} \\ \text{Annotate } I(p)\ I_{\mathcal{A}}(p) & \text{if } p \in P_{\mathcal{A}} \end{cases}$

i. $\{C(p) \times C_{\mathcal{A}}(p) \mid p \in P_{\mathcal{A}}\}$ is the set of product colour sets for the annotated places in CPN*.

ii. + iii. + iv. + v. The places, transitions, arcs, and node function in CPN* are unchanged with respect to CPN.

vi. Defining the colour function C* is straightforward. The colour set for a non-annotated place in CPN* is the same as its colour set in CPN. The colour set for an annotated place p in CPN* is $C(p) \times C_{\mathcal{A}}(p)$.

vii. The guard function in CPN* is unchanged with respect to CPN.

viii. The arc expression for a non-annotated arc a in CPN* is the same as the arc expression for a in CPN. If the arc expression for a is missing in CPN, then its arc expression will also be missing in CPN*. This is shorthand for empty, as usual for CP-nets. The arc expression for an annotated arc a in CPN* is derived from the arc expression for a in CPN and the auxiliary arc expression for a in $\mathcal{A}$. The expression `Annotate (E(a)) (`$E_{\mathcal{A}}$`(a))` will yield a multi-set with type $(C(p(a)) \times Type(E_{\mathcal{A}}(p(a))))_{MS}$ which is exactly $(C(p(a)) \times C_{\mathcal{A}}(p(a)))_{MS}$, as required. If an arc expression (for an annotated arc) evaluates to a single colour in CPN, then we allow the arc expression for CPN* to be the pair `(E(a), `$E_{\mathcal{A}}$`(a))` where the first element is the arc expression from CPN, and the second element is the auxiliary arc expression from $\mathcal{A}$. This is shorthand for the multi-set ``1`(E(a), ``$E_{\mathcal{A}}$`(a))`.

ix. If p is not an annotated place, then the initial expression of p in CPN* is unchanged with respect to CPN. If the initial expression of a place is missing in CPN, then its initial expression will also be missing in CPN*. A missing initial expression is shorthand for the empty. For an annotated place p, the expression `Annotate (I(a)) (`$\mathtt{I}_{\mathcal{A}}$`(a))` will yield a multi-set with type $(\mathrm{C(p(a))}\times\mathrm{Type}(\mathrm{I}_{\mathcal{A}}(\mathrm{p(a)})))_{MS}$ which is exactly $(\mathrm{C(p(a))}\times\mathrm{C}_{\mathcal{A}}(\mathrm{p(a)}))_{MS}$, as required. I*(p) is a closed expression for all p, since I(p) is a closed expression for all p, and $\mathrm{I}_{\mathcal{A}}$(p) is closed for all p$\in$$\mathrm{P}_{\mathcal{A}}$. When the type of the initial expression for an annotated place in the underlying CP-net is a single colour, then we allow the the initial expression in CPN* to be the pair `(I(p),`$\mathtt{I}_{\mathcal{A}}$`(p))` that is uniquely determined by the initial expression of p in CPN and the auxiliary initial expression of p in $\mathcal{A}$. This is shorthand for ``1`(I(p),``$\mathtt{I}_{\mathcal{A}}$`(p))`.

Covering Markings, Bindings and Steps We will now define what it means for markings, bindings and steps of a matching CP-net to cover the markings, bindings and steps of its underlying CP-net.

Definition 5. Let (CPN, $\mathcal{A}$) be an annotated CP-net with matching CP-net CPN*. We then define three projection functions π that map a marking M* of CPN* into a marking M of CPN, a binding b* of a transition t in CPN* into a binding b of t in CPN, and a step Y* of CPN* into a step Y of CPN, respectively.

i. $\forall p\in P^*: (\pi(M^*))(p) = \begin{cases} M^*(p) & \text{if } p\notin P_{\mathcal{A}} \\ \pi(M^*(p)) & \text{if } p\in P_{\mathcal{A}} \end{cases}$

ii. $\forall$ v$\in$Var(t): $(\pi(b^*))(v)=b^*(v)$, where Var(t) are the variables of t in CPN.

iii. $(\pi(Y^*)) = \sum_{(t,b^*)\in Y^*} (Y^*(t,b^*))\text{`}\big(t,\pi(b^*)\big)$

If $\pi(M^*)$=M, $\pi(b^*)$=b, and $\pi(Y^*)$=Y, then we say that M*, b*, and Y* *cover* M, b, and Y, respectively. We also say that M, b, and Y are *covered* by M*, b*, and Y*, respectively.

i. Given a marking of a matching CP-net, π will remove the annotations from the tokens on annotated places, and it will leave the markings of non-annotated places unchanged. A marking of a matching CP-net covers a marking of its underlying CP-net, if the two markings are equal when annotations in the first marking are ignored.

ii. Given a binding of a transition in CPN*, π removes the bindings of the variables in Var*(t)$\setminus$Var(t), i.e. π removes the bindings of the variables of t that are not found in CPN. A binding of a transition in CPN* covers a binding of the corresponding transition in CPN when the variables that are found in both CP-nets are bound to the same value.

iii. For each binding element (t,b*) in Y*, π removes the bindings of the variables of t that are not found in CPN.

We define similar functions that map the set of markings ($\mathbb{M}^*$), the set of steps ($\mathbb{Y}^*$), the set of token elements (TE*), and the set of binding elements (BE*) of CPN* into the corresponding sets in CPN:

$\pi(\mathbb{M}^*)=\{\pi(M^*): M^*\in\mathbb{M}^*\}$
$\pi(TE^*)=\big\{(p, c^*) \mid p\notin P_{\mathcal{A}} \text{ and } c^*\in C^*(p)\big\} \cup \big\{(p, \pi(c^*)) \mid p\in P_{\mathcal{A}} \text{ and } c^*\in C^*(p)\big\}$
$\pi(\mathbb{Y}^*)=\{\pi(Y^*): Y^*\in\mathbb{Y}^*\}$
$\pi(BE^*) = \{\big(t,\pi(b^*)\big) \mid (t,b^*)\in BE^*\}$

Sound Annotation Layers Definition 3 defines the syntax for elements in annotation layers, but it does not guarantee that annotations do not affect the behaviour of the underlying CP-net. Instead of specifying which kinds of auxiliary arc inscriptions are allowed, we will define a more general property that has to be satisfied.

Definition 6. Let (CPN,$\mathcal{A}$) be an annotated CP-net with matching CP-net CPN*. $\mathcal{A}$ is a *sound* annotation layer if the following property is satisfied:

$$\forall \mathrm{M}\in\mathbb{M}, \forall\ \mathrm{Y}\in\mathbb{Y}, \forall \mathrm{M}^*\in\mathbb{M}^*: \mathrm{M}[\mathrm{Y}\rangle \wedge \pi(\mathrm{M}^*)=\mathrm{M} \Rightarrow \exists\ \mathrm{Y}^*\in\mathbb{Y}^*: \pi(\mathrm{Y}^*)=\mathrm{Y} \wedge \mathrm{M}^*[\mathrm{Y}^*\rangle$$

where $\mathbb{M}$ and $\mathbb{Y}$ are the set of markings and the set of steps, respectively, for CPN, and $\mathbb{M}^*$ and $\mathbb{Y}^*$ are the analogous sets for CPN*.

Assume that the step Y is enabled in the marking M in the underlying CP-net. Let M* be a marking of CPN* that covers M. Definition 6 states that it must be possible to find a step Y* that covers Y, and Y* must be enabled in M*. The soundness of an annotation layer is essential for showing that for every occurrence sequence in the underlying CP-net, there is at least one matching occurrence sequence in the matching CP-net which is identical to the occurrence sequence from the underlying CP-net when annotations are ignored.

The auxiliary arc expressions on input arcs to a transition will be used, in part, to determine if the transition is enabled in a given state of the matching CP-net. By limiting the kinds of auxiliary arc expressions that are allowed on input arcs to transitions, it is possible to guarantee that annotations cannot restrict the enabling of a transition in the matching CP-net with respect to what is allowed in the underlying CP-net. Exactly which kinds of auxiliary arc expressions should be allowed may be decided by the implementors of tools supporting CP-nets. It is also the responsibility of tool implementors to prove that their allowable set of auxiliary arc expressions fulfil Def. 6. An example of an allowable auxiliary arc expression for arc a is a single variable v. However, v must also fulfil the following: v may not found in any arc expressions for the arcs surrounding t(a), and v may not be found in any other auxiliary arc expression for input arcs to t(a).

5.4 Matching Behaviour

In the previous sections we have stated that the behaviour of a matching CP-net *matches* the behaviour of its underlying CP-net. Informally this means that every occurrence sequence in a matching CP-net corresponds to an occurrence sequence in the underlying CP-net, and for every occurrence sequence in the underlying CP-net, it is possible to find at least one corresponding occurrence sequence in the matching CP-net. If a matching CP-net is derived from a CP-net and a *sound* annotation layer, then the following theorem shows how the behaviour of the matching CP-net matches the behaviour of its underlying CP-net.

Theorem 1. *Let (CPN,$\mathcal{A}$) be an annotated CP-net with a* <u>*sound*</u> *annotation layer. Let CPN* be the matching CP-net derived from (CPN,$\mathcal{A}$). Let M_0, $\mathbb{M}$, and $\mathbb{Y}$ denote the initial marking, the set of all markings, and the set of all steps, respectively, for CPN. Similarly, let M_0^*, $\mathbb{M}^*$, and $\mathbb{Y}^*$ denote the same concepts for CPN*. Then we have the following properties:*

i. $\pi(\mathbb{M}^*)=\mathbb{M} \wedge \pi(M_0^*)=M_0$.
ii. $\pi(\mathbb{Y}^*)=\mathbb{Y}$.
iii. $\forall\ M_1^*, M_2^*\in\mathbb{M}^*, \forall\ Y^*\in\mathbb{Y}^*: M_1^*[Y^*\rangle M_2^* \Rightarrow \pi(M_1^*)[\pi(Y^*)\rangle\pi(M_2^*)$
iv. $\forall\ M_1,M_2\in\mathbb{M}, \forall\ Y\in\mathbb{Y}, \forall\ M_1^*,M_2^*\in\mathbb{M}^*:$
$M_1[Y\rangle M_2 \wedge \pi(M_1^*)=M_1 \Rightarrow \exists Y^*\in\mathbb{Y}^*: \pi(Y^*)=Y \wedge M_1^*[Y^*\rangle M_2^* \wedge \pi(M_2^*)=M_2$

i. The markings of a matching CP-net cover the markings of its underlying CP-net. The markings of the underlying CP-net are covered by the markings of the matching CP-net. The initial marking of a matching CP-net covers the initial marking of its underlying CP-net.

ii. The steps of a matching CP-net cover the steps of its underlying CP-net. The steps of the underlying CP-net are covered by the steps of the matching CP-net.

iii. An occurrence sequence of length one in the matching CP-net covers an occurrence sequence of length one in its underlying CP-net. In other words, if marking M$_2^*$ is reached by the occurrence of Y* in marking

M_1^* in CPN*, then $\pi(M_2^*)$ will be reached by the occurrence of $\pi(Y^*)$ in $\pi(M_1^*)$ in CPN.

iv. An occurrence sequence of length one in the underlying CP-net can be covered by an occurrence sequence of length one in the matching CP-net. If M_2 is reached by the occurrence of Y in M_1 in CPN, and if marking M_1^* in CPN* covers M_1, then it is always possible to find a step Y^* in CPN*, such that Y^* covers Y and is enabled in M_1^*. If the occurrence of Y^* in M_1^* yields the marking M_2^*, then M_2^*will cover M_2.

The proof for Theorem 1 can be found in Appendix A.

5.5 Multiple Annotation Layers

The previous sections have discussed how to create a single annotation layer for a CP-net. The purpose of introducing an annotation layer is to make it possible to separate annotations from the CP-net, and to annotate a CP-net for several different purposes like, e.g. performance analysis and MSCs. However, if only one annotation layer exists, then it is not possible to easily disable, e.g. only the annotations for performance analysis, while still using the annotations for MSCs. The reason is, that all annotations have to be written in the one and only annotation layer. This motivates the need for multiple layers of annotations. When multiple annotation layers are allowed, then independent annotations can be written in separate annotation layers, and thereby making it easy to enable and disable each of the independent annotation layers.

Definition 7 defines multiple annotation layers. Multiple annotation layers are defined using the fact that a single annotation layer,$\mathcal{A}_1$, and a CP-net, CPN, is translated to another CP-net, CPN$_1^*$. Seen from another annotation layer,$\mathcal{A}_2$, CPN$_1^*$ is essentially the same as CPN aside from the added annotations, and can therefore be annotated with an annotation layer$\mathcal{A}_2$. The consequence of this definition is that$\mathcal{A}_2$ can refer to annotations in$\mathcal{A}_1$. In general, annotations in annotation layer$\mathcal{A}_i$ can refer to annotations in annotation layer$\mathcal{A}_j$ when $j \leq i$.

Definition 7. Let CPN be a CP-net and let$\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n$ be annotation layers for CPN. Let τ be the translation from an annotation layer$\mathcal{A}$ and a corresponding CP-net CPN to CPN*, as defined in Sect. 5.3. Then CPN* with multiple annotation layers is defined by:

$$CPN^* = \tau(\ldots\tau(\tau(\text{CPN},\mathcal{A}_1),\mathcal{A}_2),\mathcal{A}_n)$$

6 Conclusion

In this paper we have discussed annotations for CP-nets where annotations are used to add auxiliary information to tokens. Auxiliary information is needed to support different uses of a single CP-net, such as for performance analysis and visualisation, thus the information should not have influence on the dynamic behaviour of a CPN model. One of the advantages of using annotations instead of manually extending the colour sets in a CPN model is that annotations are specified separately from the colour sets and arc inscriptions. That means that it is easy to enable and disable annotations from being part of the simulation. This is a great advantage when using a model for several purposes such as functional analysis, performance analysis, and visualisation. In addition, it is a great advantage that the behaviour of the matching CP-net matches the behaviour of the underlying CP-net in a very specific and predictable way.

Related work is considered in, e.g. Lakos' work on abstraction [11], where behaviour-respecting abstractions of CP-nets have been investigated, and a so-called colour refinement is proposed. This colour refinement is used to specify more detailed behaviour in sub-modules by extending colour sets to larger domains. The refined colours are only visible in the sub-modules, and the refined colours will typically contain information that is necessary for modelling the behaviour of the system in question. This colour refinement somewhat corresponds to our way of extending colour sets by adding annotations to colours. We are not aware of any other work that addresses the problem of introducing *auxiliary* information into a CP-net (or any other type of simulation model) while at the same time preserving the behaviour of the

CP-net. Nor do we know of any other method that can be used to automatically enable or disable different kinds of instrumentation when analysing different aspects of one particular model.

Ex*Spect* [1] is another tool for CP-nets. The tool provides libraries of so-called building blocks that provide support for, e.g., creating message sequence charts and performance analysis. Each building block is similar to a substitution transition and its subpage in Design/CPN. In Ex*Spect* all information that is necessary for updating a MSC or for collecting performance data is included in token colours. Reading the relevant data from token values and processing it is also encoded directly into the model via the building blocks. For example, the building block that can be used to calculate performance measures contains a place which holds the current result. When a certain transition occurs, a new value can be read from a binding element, and the result on this place is updated accordingly. While the building blocks are very easy to use, no attempt is made to separate auxiliary information from a CP-net, and the behaviour of the CP-net also reflects behaviour that is probably not found in the system being modelled.

There are many issues that can be addressed in future work regarding annotations. The techniques that have been presented here have not yet been used in practice. Clearly, it is important that support for annotations be implemented in a CPN tool in order to investigate the practicality and usefulness of the proposed method. Future work includes additional research on dealing with arc inscriptions that do not evaluate to a single colour on input arcs to transitions. In addition, further work is required to improve our proposal of how to add annotations to multi-sets of tokens. The definition of annotation layers states that it is only possible to add one particular annotation to all elements in a multi-set that is obtained by evaluating either an initial expression or an arc expression on an output arc from a transition. This is unnecessarily restrictive, and it should be generalised to make it possible to add different annotations to different elements in a multi-set. Practical experience with annotations may also show that the definition of annotation layers should be extended to include the possibility of defining guards in annotation layers.

In this paper we have only considered how to add annotations to existing arcs expressions, and thereby only considered how to annotate existing tokens. However, it might be useful also to be able to add net structure to the annotation layers. As an example, a place could be added only to the annotation layer with a token to hold a counter with the number of occurrences of a transition. Allowing additional net structure at the annotation layers would make it possible to take advantage of the powerfulness of the graphical notation of CP-nets when encoding the logics of the annotations.

We have only discussed separating the auxiliary annotations and the CP-net from each other. This could be generalised to also allow splitting a CP-net into layers where more layers can be combined to specify the full behaviour of a CP-net. In other words, the specification of the behaviour in a CP-net could be split in more layers. As an example, reconsider the resource allocation CP-net in Fig. 1 in Sect. 2. The loop handling the resource on the place R (R, T1, B, T2, C, and T3) is to some extent independent from the remaining model (even though it has impact on the behaviour). This loop could be separated from the remaining CPN model into a new layer to emphasise the fact that the loop is an extra requirement that can be added to the system. This facility could turn out to be very useful when a modeller is simplifying a CP-net to, e.g. be able to generate a sufficiently small state space to be able to analyse it. It would be a matter of moving the parts of the net structure that should not be included when generating the state space to another layer, and then only conduct the analysis on the remaining parts of the CP-net. This could be obtained by disabling the layer with the unneeded behaviour, and the state space could be generated. The advantage is that now a single model exists with layers specifying different behaviour which can be enabled or disabled – instead of having several similar models. Finally, such layers can also make it easier to develop tools where more people can work on a model concurrently, when they operate on different layers.

Acknowledgements We would like to thank Kurt Jensen who has read several drafts of this paper and has provided invaluable comments and feedback. We would also like to thank Søren Christensen, Louise Elgaard and Thomas Mailund who have also read and commented on previous drafts of this paper. Finally, we would also like to thank the anonymous reviewers for their comments and suggestions.

References

1. W. v. d. Aalst, P. d. Crom, R. Goverde, K. v. Hee, W. Hofmann, H. Reijers, and R. v. d. Toorn. ExSpect 6.4 – An Executable Specification Tool for Hierarchical Coloured Petri Nets. In M. Nielsen and D. Simpson, editors,

Proceedings of the 21st International Conference on Application and Theory of Petri Nets, volume 1825 of *LNCS*, pages 455–464. Springer-Verlag, 2000.
2. CAPLAN Project, Online: http://www.daimi.au.dk/CPnets/CAPLAN/.
3. S. Christensen, J. Jørgensen, and L. Kristensen. Design/CPN – A Computer Tool for Coloured Petri Nets. In E. Brinksma, editor, *Proceedings of TACAS'97*, volume 1217, pages 209–223. Springer-Verlag, 1997.
4. CPN Tools, Online: http://www.daimi.au.dk/CPnets/CPN2000/.
5. Design/CPN, Online: http://www.daimi.au.dk/designCPN/.
6. G. Gallasch and L. Kristensen. Comms/CPN: A Communication Infrastructure for External Communication with Design/CPN. In K. Jensen, editor, *Third Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 79–93. University of Aarhus, Department of Computer Science, 2001. Online: http://www.daimi.au.dk/CPnets/workshop01/.
7. ITU-T Recommendation Z.120, Message Sequence Chart (MSC), 1996.
8. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997. 2nd corrected printing.
9. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 2, Analysis Methods*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997. 2nd corrected printing.
10. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 3, Practical Use*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997.
11. C. Lakos. On the Abstraction of Coloured Petri Nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 42–61. Springer-Verlag, 1997.
12. B. Lindstrøm and L. Wells. *Design/CPN Performance Tool Manual*. Department of Computer Science, University of Aarhus, Denmark, 1999. Online: http://www.daimi.au.dk/designCPN/man/.
13. B. Lindstrøm and L. Wells. Towards a Monitoring Framework for Discrete-Event System Simulations. In *To appear in Proceeding of Workshop on Discrete Event Systems*, October 2002.
14. Message Sequence Charts in Design/CPN, Online: http://www.daimi.au.dk/designCPN/libs/mscharts/.
15. J. L. Rasmussen and M. Singh. *Mimic/CPN: A Graphic Animation Utility for Design/CPN*. Department of Computer Science, University of Aarhus, Denmark. Online: http://www.daimi.au.dk/designCPN/libs/mimic/.

A Proof of Matching Behaviour

Theorem 1 (same as in Sect. 5.4) *Let (CPN,$\mathcal{A}$) be an annotated CP-net with a* <u>sound</u> *annotation layer. Let CPN* be the matching CP-net derived from (CPN,$\mathcal{A}$). Let M_0, $\mathbb{M}$, and $\mathbb{Y}$ denote the initial marking, the set of all markings, and the set of all steps, respectively, for CPN. Similarly, let M_0^*, $\mathbb{M}^*$, and $\mathbb{Y}^*$ denote the same concepts for CPN*. Then we have the following properties:*

i. $\pi(\mathbb{M}^*)=\mathbb{M} \wedge \pi(M_0^*)=M_0$.
ii. $\pi(\mathbb{Y}^*)=\mathbb{Y}$.
iii. $\forall\, M_1^*, M_2^*\in\mathbb{M}^*, \forall\, Y^*\in\mathbb{Y}^*: M_1^*[Y^*\rangle M_2^* \Rightarrow \pi(M_1^*)[\pi(Y^*)\rangle\pi(M_2^*)$
iv. $\forall\, M_1,M_2\in\mathbb{M}, \forall\, Y\in\mathbb{Y}, \forall\, M_1^*,M_2^*\in\mathbb{M}^*$:
$M_1[Y\rangle M_2 \wedge \pi(M_1^*)=M_1 \Rightarrow \exists Y^*\in\mathbb{Y}^*: \pi(Y^*)=Y \wedge M_1^*[Y^*\rangle M_2^* \wedge \pi(M_2^*)=M_2$

Proof: The proof is a simple consequence of earlier definitions and Jensen's definitions for CP-nets [8]. Let TE, (t,b), and BE denote the set of all token elements, a binding element, and the set of all binding elements, respectively, for CPN. Similarly, let TE*, (t,b*), BE* denote the same concepts for CPN*.

Before showing that the above properties hold, we will show that the following holds for all annotated arcs:

$$\forall\,(t,b^*), \forall\, a\in A_{\mathcal{A}}\cap A(t): \pi\big(E^*(a)\langle b^*\rangle\big)=E(a)\langle\pi(b^*)\rangle. \qquad (\dagger)$$

Let (t,b*) and $a\in A_{\mathcal{A}}\cap A(t)$ be given.

$$\pi\big(E^*(a)\langle b^*\rangle\big) \overset{Def.\ 4.viii}{=} \pi\Big(\big(\text{Annotate } E(a)\ E_{\mathcal{A}}(a)\big)\langle b^*\rangle\Big) \overset{Defs.\ 1\&2}{=} E(a)\langle b^*\rangle \overset{Def.\ 5.ii}{=} E(a)\langle\pi(b^*)\rangle$$

Property i. We will show that $\mathbb{M}=\pi(\mathbb{M}^*)$. It is straightforward to show that $\pi(\mathbb{M}^*)=(\pi(TE^*))_{MS}$, and the proof is therefore omitted. From Def. 2.7 in [8] we have that $\mathbb{M}=TE_{MS}$. Thus it is sufficient to show that TE=π(TE*). The definition of π(TE*) gives us:

$$\pi(TE^*)=\big\{(p, c^*) \mid p\notin P_{\mathcal{A}} \text{ and } c^*\in C^*(p)\big\} \cup \big\{(p, \pi(c^*)) \mid p\in P_{\mathcal{A}} \text{ and } c^*\in C^*(p)\big\}$$

which by the definition of C* (Def. 4.vi) is equivalent to:

$\pi(TE^*)=\{(p, c) \mid p\notin P_{\mathcal{A}} \text{ and } c\in C(p)\} \cup \{(p, \pi(c^*)) \mid p\in P_{\mathcal{A}} \text{ and } c^*\in C(p)\times C_{\mathcal{A}}(p)\}$

which by the definition of the projection of annotated elements (Def. 1) is equivalent to:

$\pi(TE^*)=\{(p, c) \mid p\notin P_{\mathcal{A}} \text{ and } c\in C(p)\} \cup \{(p, c) \mid p\in P_{\mathcal{A}} \text{ and } c\in C(p)\}$

the two sets can be combined and we have:

$\pi(TE^*)=\{(p, c) \mid p\in P \text{ and } c\in C(p)\} \overset{Def.\ 2.7 in\ [8]}{=} TE$

To show that $\pi(M_0^*)=M_0$, we will show that $\forall p\in P^*$: $(\pi(M_0^*))(p)=M_0(p)$.
Consider non-annotated places:

$\forall p\notin P_{\mathcal{A}}$: $(\pi(M_0^*))(p) \overset{Def.\ 5.i}{=} M_0^*(p) = I^*(p) \overset{Def.\ 4.ix}{=} I(p) = M_0(p)$

Consider annotated places:

$\forall p\in P_{\mathcal{A}}$: $(\pi(M_0^*))(p) \overset{Def.\ 5.i}{=} \pi(M_0^*(p)) = \pi(I^*(p)) \overset{Def.\ 4.ix}{=} \pi\big(\text{Annotate } I(p)\ I_{\mathcal{A}}(p)\big) \overset{Defs.\ 1\&2}{=} I(p) = M_0(p)$

Property ii. We must show that $\mathbb{Y}=\pi(\mathbb{Y}^*)$. It is straightforward to show that $\pi(\mathbb{Y}^*)= (\pi(BE^*))_{MS}$, therefore the proof is omitted. From Def. 2.7 in [8] we have that $\mathbb{Y}=BE_{MS}$, therefore it is sufficient to show that $BE=\pi(BE^*)$, which we will do by showing: $(t,b')\in\pi(BE^*)\Leftrightarrow (t,b')\in BE$.

Let us show $\Rightarrow$: Let $(t, b')\in\pi(BE^*)$ be given. There exists $(t,b^*)\in BE^*$ such that $(t, \pi(b^*))=(t, b')$ (by definition of $\pi(BE^*)$). b^* is a binding of t in CPN*, therefore for all $v\in Var^*(t)$, where $Var^*(t)$ is the set of variables for t in CPN*, $b^*(v)\in Type(v)$, and b^* fulfils the guard of t in CPN*, i.e. $G^*(t)\langle b^*\rangle$.

From Def. 5.ii we have that for all $v\in Var(t)$, $(\pi(b^*))(v)=b^*(v)$, and we know that $b^*(v)\in Type(v)$. Since $G^*(t)=G(t)$ (Def. 4.vii) and $Var(G(t))\subseteq Var(t)$, we can conclude that $G(t)\langle\pi(b^*)\rangle$, i.e. $\pi(b^*)$ fulfills the guard of t in CPN. From the definition of a binding (Def. 2.6 in [8]), we have that $\pi(b^*)$ is a binding for t in CPN, therefore $(t, \pi(b^*))=(t,b')$ is a binding element for CPN, i.e. $(t, b')\in BE$.

Let us show $\Leftarrow$: Let $(t, b')\in BE$ be given. Using arguments that are similar to the above it is straightforward to show that b' fulfills the guard for t in CPN*, i.e. $G^*(t)\langle b'\rangle$. The binding b' does not bind the variables in $Var^*(t)\setminus Var(t)$. Define a new function b^* on $Var^*(t)$:

$$b^*(v) = \begin{cases} b'(v) & \text{if } v\in Var(t) \\ \text{an arbitrary value from } Type(v) & \text{if } v\in Var^*(t)\setminus Var(t) \end{cases}$$

According to Def. 2.6 in [8], b^* is a binding for t in CPN*. Therefore, (t, b^*) is a binding element for CPN*. By definition of b^*, we have that $\pi(b^*)=b'$, and as a result, $(t, b')\in\pi(BE^*)$.

Property iii. We must show that $\forall\ M_1^*, M_2^*\in\mathbb{M}^*, \forall\ Y^*\in\mathbb{Y}^*: M_1^*[Y^*\rangle M_2^* \Rightarrow \pi(M_1^*)[\pi(Y^*)\rangle\pi(M_2^*)$

We will first show that $\pi(M_1^*)[\pi(Y^*)\rangle$. By the *enabling rule* (Def. 2.8 in [8]) we have that:

$$\forall p\in P^*: \sum_{(t,b^*)\in Y^*} \sum_{a\in A(p,t)} E^*(a)\langle b^*\rangle \le M_1^*(p) \qquad (*)$$

Consider non-annotated places and non-annotated arcs. Since $E^*=E$ for all non-annotated arcs (by Def. 4.viii), and $M_1^*=\pi(M_1^*)$ for all non-annotated places (by Def. 5.i), it follows from $(*)$ that:

$$\forall p\notin P_{\mathcal{A}}: \sum_{(t,b^*)\in Y^*} \sum_{a\in A(p,t)} E(a)\langle b^*\rangle \le (\pi(M_1^*))(p)$$

which by the fact that $\pi(b^*)=b^*$ for all variables in $Var(E(a))$ (by Def. 5.ii) and the definition of $\pi(Y^*)$ (Def. 5.iii) is equivalent to:

$$\forall p\notin P_{\mathcal{A}}: \sum_{(t,\pi(b^*))\in\pi(Y^*)} \sum_{a\in A(p,t)} E(a)\langle\pi(b^*)\rangle \le (\pi(M_1^*))(p) \qquad (**)$$

Consider annotated places and annotated arcs. From Def. 1 and $(*)$, it follows that:

$$\forall p\in P_{\mathcal{A}}: \pi\Big(\sum_{(t,b^*)\in Y^*} \sum_{a\in A(p,t)} E^*(a)\langle b^*\rangle\Big) \le \pi(M_1^*(p))$$

which by Defs. 1 and 5.i is equivalent to:

$$\forall p\in P_{\mathcal{A}}\colon \sum_{(t,b^*)\in Y^*}\sum_{a\in A(p,t)} \pi\big(E^*(a)\langle b^*\rangle\big) \leq (\pi(M_1^*))(p)$$

which by (†) and the definitions of $\pi(b^*)$ and $\pi(Y^*)$ is equivalent to:

$$\forall p\in P_{\mathcal{A}}\colon \sum_{(t,\pi(b^*))\in\pi(Y^*)}\sum_{a\in A(p,t)} E(a)\langle\pi(b^*)\rangle \leq (\pi(M_1^*))(p)$$

which together with (∗∗) and the enabling rule gives us that $\pi(M_1^*)[\pi(Y^*)\rangle$.

Next we have to prove that the marking reached when Y^* occurs in M_1^* covers the marking that is reached when $\pi(Y^*)$ occurs in $\pi(M_1^*)$, i.e. that $\pi(M_1^*)[\pi(Y^*)\rangle\pi(M_2^*)$. A proof similar to the above can be used to show this, and the proof is therefore omitted.

Property iv. We must show that $\forall\ M_1,M_2\in\mathbb{M},\ \forall\ Y\in\mathbb{Y},\ \forall\ M_1^*,M_2^*\in\mathbb{M}^*$:

$$M_1[Y\rangle M_2 \wedge \pi(M_1^*)=M_1 \Rightarrow \exists Y^*\in\mathbb{Y}^*\colon \pi(Y^*)=Y \wedge M_1^*[Y^*\rangle M_2^* \wedge \pi(M_2^*)=M_2$$

Let $M_1[Y\rangle M_2$ in CPN be given. It is straightforward to show that it is always possible to find $M_1^*\in\mathbb{M}^*$ such that $\pi(M_1^*)=M_1$, thus the proof is omitted. Since CPN* is a matching CP-net that is derived from an annotated CP-net with a sound annotation layer, and $\pi(M_1^*)=M_1$, Def. 6 tells us that there exists $Y^*\in\mathbb{Y}^*$ such that $\pi(Y^*)=Y$ and $M_1^*[Y^*\rangle$.

We have only left to show that the marking reached after Y occurs in M_1 is covered by the marking reached when Y^* occurs in M^*. Since $M_1[Y\rangle M_2$ in CPN, the *occurrence rule* (Def. 2.9 in [8]) gives us that:

$$\forall\ p\in P\colon M_2(p) = \Big(M_1(p) - \sum_{(t,b)\in Y}\sum_{a\in A(p,t)} E(a)\langle b\rangle\Big) + \sum_{(t,b)\in Y}\sum_{a\in A(t,p)} E(a)\langle b\rangle \qquad (\diamond)$$

Since $M_1^*[Y^*\rangle$ in CPN*, the occurrence rule gives us that:

$$\forall p\in P^*\colon M_2^*(p) = \Big(M_1^*(p) - \sum_{(t,b^*)\in Y^*}\sum_{a\in A(p,t)} E^*(a)\langle b^*\rangle\Big) + \sum_{(t,b^*)\in Y^*}\sum_{a\in A(t,p)} E^*(a)\langle b^*\rangle \qquad (\diamond\diamond)$$

In other words, $M_1^*[Y^*\rangle M_2^*$. We must now show that $\pi(M_2^*)=M_2$.

We will show that $\pi(M_2^*)=M_2$ for non-annotated places. We have found M_1^*, such that $\pi(M_1^*)=M_1$. We have that $M_1^*=\pi(M_1^*)$ and $M_2^*=\pi(M_2^*)$ for non-annotated places (by Def. 5.i). For all non-annotated arcs $E^*=E$ (by Def. 4.viii). It follows from these facts and ($\diamond\diamond$) that:

$$\forall p\notin P_{\mathcal{A}}\colon (\pi(M_2^*))(p) = \Big(M_1(p) - \sum_{(t,b^*)\in Y^*}\sum_{a\in A(p,t)} E(a)\langle b^*\rangle\Big) + \sum_{(t,b^*)\in Y^*}\sum_{a\in A(t,p)} E(a)\langle b^*\rangle$$

which by the fact that $\pi(b^*)=b^*$ for all variables in Var(E(a)) (by Def. 5.ii) and the fact that $\pi(Y^*)=Y$ is equivalent to:

$$\forall p\notin P_{\mathcal{A}}\colon (\pi(M_2^*))(p) = \Big(M_1(p) - \sum_{(t,b)\in Y}\sum_{a\in A(p,t)} E(a)\langle b\rangle\Big) + \sum_{(t,b)\in Y}\sum_{a\in A(t,p)} E(a)\langle b\rangle \qquad (\diamond\diamond\diamond)$$

We will show that $\pi(M_2^*)=M_2$ for annotated places. From the definition of π for multi-sets and markings (Defs. 1 and 5.i) and from ($\diamond\diamond$), it follows that :

$$\forall p\in P_{\mathcal{A}}\colon (\pi(M_2^*))(p) = \Big((\pi(M_1^*))(p) - \sum_{(t,b^*)\in Y^*}\sum_{a\in A(p,t)} \pi\big(E^*(a)\langle b^*\rangle\big)\Big) + \sum_{(t,b^*)\in Y^*}\sum_{a\in A(t,p)} \pi\big(E^*(a)\langle b^*\rangle\big)$$

which by (†) and the fact that $\pi(M_1^*)=M_1$ is equivalent to:

$$\forall p\in P_{\mathcal{A}}\colon (\pi(M_2^*))(p) = \Big(M_1(p) - \sum_{(t,b^*)\in Y^*}\sum_{a\in A(p,t)} E(a)\langle b^*\rangle\Big) + \sum_{(t,b^*)\in Y^*}\sum_{a\in A(t,p)} E(a)\langle b^*\rangle$$

which by the definitions of $\pi(b^*)$ and $\pi(Y^*)$, and the fact that all variables in E(a) are bound by $\pi(b^*)$ is equivalent to:

$$\forall p\in P_{\mathcal{A}}\colon (\pi(M_2^*))(p) = \Big(M_1(p) - \sum_{(t,b)\in Y}\sum_{a\in A(p,t)} E(a)\langle b\rangle\Big) + \sum_{(t,b)\in Y}\sum_{a\in A(t,p)} E(a)\langle b\rangle$$

which together with ($\diamond$) and ($\diamond\diamond\diamond$) gives us that $\pi(M_2^*)=M_2$.

□