HYBRIDGE

Distributed Control and Stochastic Analysis of Hybrid Systems
Supporting Safety Critical Real-Time Systems Design

WP9: Perform risk assessment of distributed control system

Compositional specification of a multi-agent system by Dynamically Coloured Petri Nets

M.H.C. Everdij, M.B. Klompstra,
H.A.P. Blom, B. Klein Obbink [1]

22 November 2004

Version: *0.5*
Task number: *9.2*
Deliverable number: *D9.2*
Contract: *IST-2001-32460 of European Commission*

[1] Organisation

DOCUMENT CONTROL SHEET

Title of document: *Compositional specification of a multi-agent system by Dynamically Coloured Petri Nets*

Authors of document: *M.H.C. Everdij, M.B. Klompstra, H.A.P. Blom, B. Klein Obbink*

Deliverable number: *D9.2*

Contract: *IST-2001-32460 of European Commission*

Project: *Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design (HYBRIDGE)*

DOCUMENT CHANGE LOG

Version #	Issue Date	Sections affected	Relevant information
0.1	30-04-2004	All	Initial draft
0.2	30-06-2004	Section 6	Draft
0.3	31-08-2004	Section 6	Updated Petri Net and counting of arcs
0.4	10-09-2004	Abstract, Sections 1, 5, 7	Abstract updated, new section 1 inserted, section 5 (formerly 4) and section 7 (formerly 6) updated
0.5	22-11-2004	Sections 2, 5, 6, 7	Review comments incorporated

Version 1.0		Organisation	Signature/Date
Authors	M.H.C. Everdij	NLR	
	M.B. Klompstra	NLR	
	H.A.P. Blom	NLR	
	B. Klein Obbink	NLR	
Internal reviewers	P. Lezaud	CENA	
	A. Van der Schaft	TWEN	

Abstract

In WP9 of the HYBRIDGE project, a risk assessment for a distributed ATM operation is executed, based on a stochastic hybrid model for the operation. In this second report of WP9 it is explained how the selected ATM operation, called Free Flight, is specified in the form of a Petri net. For safety-critical operations in the nuclear and chemical industries, Petri nets have proven to be useful for the compositional specification of appropriate accident risk assessment models. For air traffic operations, the development of such model is more challenging due to the high distribution of and complex interactions between the multiple agents involved. The specific problems are: A) Structure of a low-level Petri net changes due to adding an interconnection with another low-level Petri net; B) Duplication of arcs and transitions within a low level Petri net; C) Cluttering of interconnections. The paper develops adequate solutions for each of these problems. The solution approaches are first explained graphically, and next formally. The approach developed is illustrated for an air traffic operation example.

Contents

1. Introduction

Background of HYBRIDGE project

The 21st century finds Europe facing a number of remarkable changes, many of which involve large complex real-time systems the management and control of which undergoes a natural trend of becoming more and more distributed while at the same time the safety criticality of these systems for human society tends to increase. However good the control design for these systems will be, humans are the only ones carrying responsibility for the operational safety. This implies that control system designs for safety critical operations have to be embedded within sound safety management systems such that the level of safety stays under control of humans. The objective of HYBRIDGE is to develop the methodologies to accomplish this, and to demonstrate their use in support of advanced air traffic management design.

In addition to direct application to air traffic management, these contributions form the nucleus for further research and development into a complex, uncertain system theory, and into application of this theory to distributed control of other real time complex systems such as communication, computer and power networks [HYBRIDGE Project].

Background of WP9

The objective of Work package 9 is to demonstrate how the various developments in other HYBRIDGE work packages contribute to a safe advanced air traffic operation which makes explicit use of distributed control over multiple aircraft. In doing so, developments from other work packages are combined either in the definition of the air traffic operation to be assessed, or in the execution of the risk assessment.

The work is organised in the following sequence of tasks:

- Task 9.1: Identify the advanced air traffic operation, including a systematic identification of all non-nominal situations and hazards. For the advanced air traffic operation we will consider one in which it is expected that the proper co-ordination between air and ground in conflict resolution is an essential condition for realising significant capacity improvements. Potential applications are closely spaced runway situations at a busy airport and collaborative airborne separation assurance in dense en-route or TMA traffic areas.
- Task 9.2: Develop a mathematically unambiguous stochastic hybrid model for the operation considered, and specify all model assumptions made, all model parameters and their values that are introduced. In view of the complexity of the development of such a mathematical model an existing model instantiation in Dynamically Coloured Petri Net (DCPN) form will be used as starting point, and all improvements will be developed in an iterative way: extend model specification, update assumptions and update list of model parameters and their values.
- Task 9.3: Develop appropriate risk decomposition and uncertainty assessment approaches. For this, use is made of the methods developed in WP8. Subsequently extend already available accident risk evaluation software according to the model instantiation and risk decomposition of Tasks 9.2 and 9.3.

- Task 9.4: Perform the risk assessment with support of stochastic analysis and Monte Carlo simulations for the instantiated models and their software implementation, and assess how sensitive the risk result is for changes in the values of the most relevant parameters.

Background of Task 9.2 and this report

The main goal of Task 9.2 is to specify a Petri net for the advanced air traffic operation that is identified as demonstration example for WP9 [Klein Obbink, 2004]. Because of the complexity of air traffic operations, the compositional specification of a Petri net is challenging. This report shows how this challenge is taken care of and illustrates this for a part of the complete Petri net specification [Bakker et al., 2004].

2. Compositional specification challenge

The aim of this paper is to extend the compositional specification power of Petri nets for application to a multi-agent hybrid system. The motivating type of application is accident risk assessment of safety-critical operations in general, and of air traffic operations in particular. For safety-critical operations in e.g. nuclear and chemical industries, it is common practice that accident risk assessment models are being developed to provide valuable feedback to the process of design and certification of a change (e.g. Royal Society, 1983; Labeau et al, 2000). Accident risk assessment could play a similar valuable role in the design of novel air traffic operations.

By the very nature of air traffic management, the various decision-makers are highly distributed: per aircraft there is a crew of pilots, and per air traffic control centre there are many human operators. In addition, the safety related decision-making process involves interactions of these humans with each other and with:

- a random and often unpredictable environment, e.g. varying wind, thunderstorms, etc.,
- a large set of procedural rules and guidelines,
- many technical and automation support systems,
- decision-makers at airline operation centres.

These aspects make accident risk assessment for air traffic operations a very challenging application area, the decision making process of which is significantly more complex than it is of operations in other safety-critical industries as is illustrated in Figure 1. This makes the specification of an unambiguous mathematical model of air traffic operations a very challenging task.

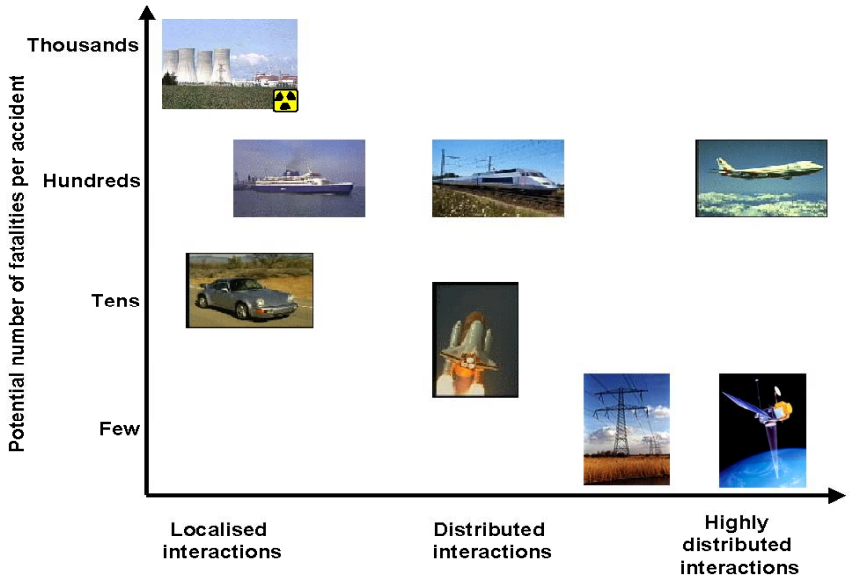

Fig. 1. Potential fatalities and distribution level of air traffic and other safety-critical operations.

The most advanced approaches that have been developed in literature to model accident risk of safety-critical operations in nuclear and chemical industries make use of the compositional specification power of Petri nets to instantiate a model, and subsequently use stochastic analysis and Monte Carlo simulation (e.g. Labeau et al, 2000) to evaluate the model. Since

their introduction in the 1960s, Petri nets have shown their usefulness for many practical applications in different industries (e.g. David & Alla, 1994). Various Petri net extensions and generalisations, new analysis techniques, and numerous supporting computer tools have been developed, which further increased their modelling opportunities, though falling short for air traffic operations. In order to capture the characteristics of air traffic operations through a Petri net, Everdij & Blom (2003a) introduced Dynamically Coloured Petri Net (DCPN) and proved that there exists a close relationship with the larger class of stochastic processes and analysis techniques needed for air traffic operations (Everdij & Blom, 2003b). Basically, a DCPN is an extension of Coloured Stochastic Petri Net (e.g. Haas, 2002), in the sense that in DCPN the token colours evolve in time (dynamically) as solutions of differential equations while the tokens reside in their places. Since its introduction, the DCPN formalism has been successfully used in practical air traffic applications, (e.g. Blom et al, 2003b, 2003c). However, it was found that when being used for modelling more and more complex multi-agent hybrid systems, the compositional specification power of Petri nets reaches its limitations. More specifically, the following problems were identified:

A. **Need for a hierarchy from low level Petri nets to the complete Petri net.** For the modelling of a complete Petri net for complex systems, a hierarchical approach is necessary in order to be able to separate local modelling issues from global or interaction modelling issues.
B. **Duplication of arcs and transitions within a low level Petri net.** Often the addition of an interconnection between two low-level Petri nets leads to a duplication of transition and arcs in the receiving Petri net.
C. **Cluttering of interconnections.** The number of interconnections between the different low level Petri nets tends to grow quadratically with the size of the Petri net.

In literature, approaches have been developed to address problem A. These approaches are outlined below.

Huber et al (1990) introduced Hierarchical Coloured Petri Nets. These Hierarchical CPNs allow a set of subnets, called pages, to be related to each other, in such a way that together they constitute a single model. The pages interact with each other in a well-defined way. A page can also be substituted by a place or a transition, in order to show its role in the larger model, or to postpone its detailed modelling until later. In addition to these substitution transitions and places, Hierarchical CPN allow invocation transitions (CPN is temporarily extended with a new instance of an invocation subpage), place fusion (a set of places is folded into a single place) and transition fusion (a set of transitions is folded into a single transition). The pages that interact solve problem A.

More recent approaches also address problem A; they consider elementary Petri nets that have input (or entry) and output (or exit) places through which these Petri nets are coupled with other Petri nets. One example approach is $B(PN)^2$ (Basic Petri Net Programming Notation), introduced by Best and Hopkins (see e.g. Fleischhack & Grahlmann, 1997). The compositional denotational semantics of $B(PN)^2$ programs can be given in terms of M-nets (modular multilabelled nets), which form an algebra of composable high-level Petri nets. These Petri net components have at least one entry place and at least one exit place. Several composition operations (e.g. parallel composition, sequential composition) are defined to couple the Petri nets. Communication is performed by transition synchronisation. Another example approach is by Kindler (1997), who introduced the concept of Petri net Components and showed how systems can be composed from components. These components have input

and output places and components can be connected at these input and output places. Kindler also provides the compositional semantics.

Also addressing problem A are Fernandes et al (1997) and Fota et al (1997), who consider sub-Petri nets that model parallel systems, and draw these sub-Petri nets in separate boxes. Places and transitions in different sub-Petri nets are coupled by arcs to model interactions. Fernandes et al (1997) uses Synchronous Interpreted Petri Nets (SIPN) as basis and shows how the interactions can be used to model synchronisation or priority of the parallel systems. Fernandes also allows hierarchy: a macroplace can be exploded (or imploded) to form (or hide) a complete sub-Petri net. Fota et al (1997) uses Generalized Stochastic Petri nets (GSPNs), refers to the sub-Petri nets as modules, and adopts the requirement that there should be exactly one token in each module; transitions in a module are not allowed to consume a token from another module without returning one immediately. Therefore, Fota introduced three module coupling mechanisms: 1) marking tests; 2) common transitions; 3) interconnection blocks. In addition, in order to improve the compactness of the module, Fota recommends two rules, called optimisation rules: 1) avoidance of immediate internal transitions; 2) module folding using memories.

For addressing problem B, some ideas from literature are useful. In order to avoid the duplication of transitions, one might apply transition fusion as proposed by Huber et al (1990), or module folding of Fota et al (1997).

The aim of this paper is to combine and adopt the approaches from literature that solve problem A and to develop approaches to solve problems B and C, and to organise these new developments into a compositional specification approach for DCPN. In addition, also the effectiveness of the approach is illustrated for the modelling of an air traffic example. The relations between our approach and those found in literature are explained in Sections 4 and 5, and are summarised below.

To solve problem A, the compositional specification of a DCPN for a complex process or operation starts with developing a Local Petri Net (LPN) for each agent that exists in the process or operation (e.g. air traffic controller, pilot, navigation and surveillance equipment). Counterparts of LPNs in literature are the modules of Fota et al (1997), the pages of Huber et al (1990) and the components of Kindler (1997). An essential difference is that our LPNs (and Fota's modules) are connected with each other such that the number of tokens residing in an LPN is not influenced by these interconnections, while Huber and Kindler do not pose this restriction.

We use two types of interconnections between nodes and arcs in different LPNs:

- Enabling arc (or inhibitor arc) from one place in one LPN to one transition in another LPN. These types of arcs have been used widely in Petri net literature, including Fota et al (1997) for inhibitor arcs and Fernandes et al (1997) for both types.
- Interaction Petri Net (IPN) from one (or more) transition(s) in one LPN to one (or more) transition(s) in another LPN. These IPNs are similar to the interconnection blocks of Fota et al (1997). If an IPN consists of one place only, then the connection of two LPNs through an IPN also has some similarity with place fusion, see e.g. Huber et al (1990) or Kindler (1997), except that our IPN will not change the number of tokens in its connecting LPNs.

Each LPN is surrounded by a box. This boxing idea has also been used by e.g. Kindler (1997) or Fernandes et al (1997). Next, to solve problems B and C, we identify additional interconnections between LPNs that allow, with well-defined meanings, arcs to initiate and/or to end on the edge of the box surrounding an LPN. To the authors' knowledge this element has no counterpart in Petri net literature; however, it is based on how Harel (1987) composes statecharts. The meaning of these interconnections from or to an edge of a box allows several arcs or transitions to be represented by only one arc or transition. In that sense, there is a relation with transition fusion used by Huber et al (1990) and with module folding used by Fota et al (1997).

This paper is organised as follows: Section 3 outlines DCPN. Next, Section 4 outlines how a DCPN can be specified in a logical sequence for each entity of an agent, and explains how the entities of agents are connected without changing the structure of low level entities. This solves problem A above. Section 5 defines some new Petri net clustering types which avoid the internal duplication problem (problem B) and the problem of cluttering interconnections (problem C). It is noted that these clustering types can also be applied to other Petri net extensions than DCPN. Section 6 extends the DCPN definition of Section 3 to include these new clustering types. In Section 7 the approach developed is illustrated for an air traffic operation example. Finally, Section 8 gives concluding remarks.

3. Dynamically Coloured Petri Net

This section gives a definition of DCPN. Subsection 3.1 describes the DCPN elements, while Subsection 3.2 describes the DCPN evolution rules. These elements and evolution rules together form the DCPN definition. For a more formal DCPN definition and a simple DCPN example we refer to (Everdij & Blom, 2003a).

3.1 DCPN elements

A Dynamically Coloured Petri Net (Everdij & Blom, 2003a) is given by the following tuple:

DCPN = (P, T, A, N, S, C, V, G, D, F, I), where:

- P is a set of places
- T is a set of transitions which consists of a set of guard transitions (T_G), a set of delay transitions (T_D) and a set of immediate transitions (T_I).
- A is a set of arcs which consists of a set of ordinary arcs (A_o), a set of enabling arcs (A_e) and a set of inhibitor arcs (A_i).
- N is a node function which maps each arc to an ordered pair of one transition and one place; multiple arcs between the same place and transition are allowed.
- S is a set of colour types for the tokens occurring in the net (a colour is the value of an object or process in Petri net terminology). Each colour type is to be of the form $\mathbb{R}^n$.
- C is a colour function which maps each place to a colour type in S.
- V is a set of place-specific colour functions which describe what happens to (i.e. define the rate of change of) the colour of a token while it resides in its place. For each place, the colour function is determined by a differential equation, which is locally Lipschitz continuous.
- G is a set of Boolean-valued transition guards associating each transition in T_G with a guard function. This guard function is continuously evaluated when the transition has a token in each of its input places, i.e., when there is at least one token per input arc of the transition present. The guard function must evaluate to True before the transition is allowed to fire (i.e. remove and produce tokens). This happens when the colours of the input tokens of the transition (which can change value through time) reach particular transition-specific value combinations.
- D is a set of transition delays associating each transition in T_D with a delay function. This delay function is continuously evaluated when the transition has a token in each of its input places. The delay function determines for how long the transition must wait before it is allowed to fire (i.e. remove and produce tokens). The firing rate depends on the colours of the input tokens of the transition (which can change value through time) and is determined by a Poisson point process.
- F is a set of (probabilistic) firing functions. For each transition it describes the quantity and colours of the tokens produced by the transition at its firing. A transition produces 0 or 1 token per outgoing arc; this quantity and the colour of the produced tokens is according to a transition-specific probabilistic mapping rule that may depend on the colours of the input tokens.
- I is an initial marking which defines the set of tokens initially present, i.e., it specifies in which places they initially reside, and the colours they initially have.

Note that according to this description, the guard and the delay evaluations of the transitions need to be continuously updated, which seems calculation-intensive. However, since the token colour functions for all places are given by (deterministic) ordinary differential equations, the evolution of the colours of the input tokens to these transitions can be predicted, hence the timing of the transition enablings can also be predicted.

The set of places *P*, the set of transitions *T*, the set of arcs *A* and the node function *N* are defined in a Petri net graph. Figure 2 shows the graphical representation of the elements in *P*, *T* and *A*. The node function *N* describes how these components are connected into a Petri net graph.

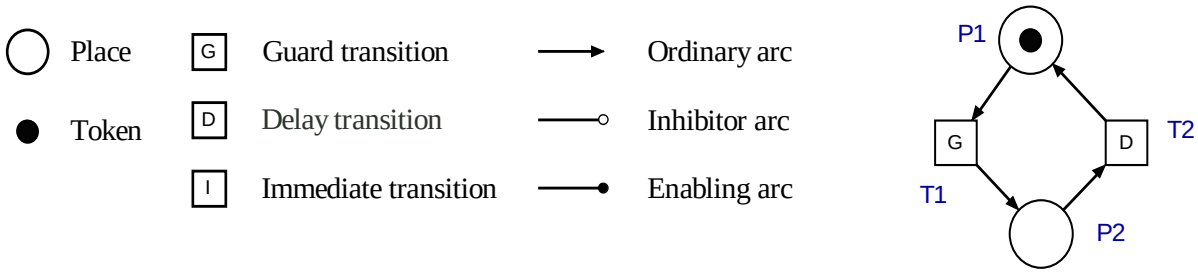

Fig. 2. Notation for places, tokens, transitions and arcs in Petri net graphs. On the right-hand-side there is a simple example Petri net graph, with two places and two transitions and a token in place P1.

3.2 DCPN evolution

Tokens and the associated colour values in a DCPN evolve over time quite similar as in a Coloured Stochastic Petri Nets (e.g. Haas, 2002). The main additions are that the colour of a token while it is residing in a place is an element of $\mathbb{R}^n$ (where n is place-specific) and may evolve according to a differential equation that is governed by the colour function V of the specific place where the token resides, and that guard and delay transitions take the evolving colour values into account.

Tokens can be removed from places by transitions that are connected to these places by incoming ordinary arcs. A transition can only remove tokens if two conditions are both satisfied. If this is the case, the transition is said to be *enabled*. The first condition is that the transition must have at least one token per ordinary arc and one token per enabling arc in each of its input places and have no token in the input places to which it is connected by an inhibitor arc. When the first condition holds, the transition is said to be *pre-enabled*. The second condition differs per type of transition. For immediate transitions the second condition is automatically satisfied if the transition is pre-enabled. For guard transitions the second condition is specified by the set of transition guards *G* and for delay transitions it is specified by the set of transition delays *D*, see their description in Subsection 3.1.

When these two conditions are satisfied, the transition removes the tokens from the input places by which it is connected through an ordinary arc. It does not remove the tokens from places by which it is connected through an enabling arc. Subsequently, the transition produces a token for some or all of its output places, specified by the firing function *F*. The colour of a produced token (which must be of the correct type, indicated by what *C* defines for the output place), and the place for which it is produced is also specified by the firing function *F*. The

evaluation of G, D and F may be dependent on the colours of the input tokens of the corresponding transition.

In order to avoid ambiguity, for a DCPN the following rules apply when two transitions are enabled simultaneously:

- R_0 The firing of an immediate transition has priority over the firing of a guard or a delay transition.
- R_1 If one transition becomes enabled by two or more disjoint sets of input tokens at exactly the same time, then it will fire these sets of tokens independently, at the same time.
- R_2 If one transition becomes enabled by two or more non-disjoint sets of input tokens at exactly the same time, and the firing of one set disables the other, then the set that is fired is selected randomly.
- R_3 If two or more transitions become enabled at exactly the same moment by disjoint sets of input tokens, then they will fire at the same time.
- R_4 If two or more transitions become enabled at exactly the same moment by non-disjoint sets of input tokens, then the transition that will fire is selected randomly, with the same probability for each transition.

Note that in Rule R_2 there is a conflict between tokens fighting for the same transition and in Rule R_4 there is a conflict between transitions fighting for the same set of tokens; these rules settle these fights by appointing a random winner. In Rules R_1 and R_3 there is no conflict between tokens fighting for the same transition or for transitions fighting for the same tokens; these firings can occur independently and these rules only take care of the timing of these individual firings. For a motivation of why these rules are chosen like this, see (Everdij & Blom, 2003a).

4. Local Petri Nets-based specification of a DCPN

The compositional specification of a Dynamically Coloured Petri Net for a complex process with many different interacting agents such as exist in air traffic operations (e.g. air traffic controllers, pilots, navigation and surveillance equipment), is a bottom-up process. Prior to starting this compositional process, per agent the relevant low level functional entities have to be identified based on expert domain knowledge of that agent. For this paper we assume that these low-level functional entities are given per agent. The compositional specification idea is then first to specify one small Petri net per functional entity of an agent, and refer to this as a Local Petri Net (LPN). Next, the interactions between these LPNs are specified. Note that our LPN definition has similar counterparts in Petri net literature. For example, Fota et al (1997) considers Generalized Stochastic Petri Nets to be composed of Modules in a similar way as we propose for DCPN to be composed of LPNs. Huber et al (1990) proposes Hierarchical Coloured Petri Nets to be composed of Pages, while Kindler (1997) considers the composition of Petri Net Components. An essential difference is that our LPNs (and Fota's modules) are connected with each other such that the number of tokens residing in an LPN (or module) is not influenced by these interconnections, while Huber and Kindler do not pose this restriction.

The specification of the various elements of one LPN is explained in Subsection 4.1; this has to be accomplished for all LPNs. Subsection 4.2 describes how the interconnections between these LPNs are established.

4.1 Specification of Local Petri Net

Specification of elements P**,** T**,** A**,** N

First, places (drawn as circles) are identified for the LPN. These places may represent operational or physical conditions (nominal modes and non-nominal modes). Next, the transitions are identified: If between two places, say P_1 and P_2, a switch might occur, one transition (rectangle) is drawn, with two arcs (arrows) connecting the places with the transition. The places are gathered in the set of places P, the transitions are gathered in the set of transitions T, and the arcs are gathered in the set of arcs A. The node function N describes for each arc which place and transition it connects.

Specification of elements S**,** C **and** V

A complex stochastic dynamic process such as in air traffic operations cannot be described by places and transitions alone. Usually, some (piecewise) continuous valued timed processes are identified, which can be influenced by and which can influence the LPN places and transitions. In DCPN, continuous valued processes are associated with tokens that reside in the places. In general, if a token resides in a particular place, its value changes according to a differential equation that is associated with that place. In this step, all places are checked on whether a continuous valued process can be associated with it. Note that on the other hand, it may happen that identified continuous valued processes lead to a necessary introduction of new places. For example: one continuous-valued process appears to alternately follow two different differential equations; in that case two places need to be introduced, each associated with one of these differential equations. All continuous valued process types are collected in the set S. The mapping of each place of the Petri net to one of these types is described by C. The set of place-specific colour functions V describes how the colour of a token changes

while it is staying in a place. For each place, V specifies the coefficients of a differential equation which describes the rate of change of the token colour: if V_P is the token colour function for place P that has colour type C_P, then the colour $c_t^P \in C_P$ of a token in place P at time t satisfies: $\frac{d}{dt} c_t^P = V_P(c_t^P)$.

Specification of elements G, D, F and I in Local PN terms

Next, for each transition, one should determine whether it is a guard transition, a delay transition or an immediate transition. A guard transition fires based on the combined colours of its input tokens reaching some value. A delay transition models a duration, e.g. of an action. An immediate transition fires without delay. The guard transitions are collected in the set T_G, the delay transitions are collected in T_D and the immediate transitions are collected in the set T_I. Subsequently, the guards G and the delays D are specified in detail. The firing function F describes the colours of the tokens fired by a transition into its output places, given the colours of the tokens in the input places. Finally, the initial marking I describes which place(s) of the LPN initially contain one or more token(s) and describes the initial colour values of these tokens, hence it describes the initial state of the process modelled by the DCPN.

4.2 Interconnections between LPNs

The interconnections between the LPNs have to be specified in a way that allows to start at the lowest level and then step by step going up to the highest level, and such that an interconnection at a higher level does not imply a significant change at a lower level. The typical exception on this is caused by non-local influences on G, D and F. In order to improve into this desired direction, in this subsection some specific types of interconnections are identified.

Following Fota et al (1997) one step in enabling a systematic bottom-up specification of a Petri net is to ensure that each LPN always contains exactly one token. For air traffic types of applications it often is useful to allow multiple tokens to be within one LPN, e.g. one for each aircraft. Hence we relaxed the Fota-principle to the following requirement: all interconnections between LPNs shall be such that the number of tokens in an LPN is not directly influenced by these interconnections. Subsequently we identified two types of interconnections that satisfied our above requirement:

- Enabling arc (or inhibitor arc) from one place in one LPN to one transition in another LPN.
- Interaction Petri Net (IPN) from one (or more) transition(s) in one LPN to one (or more) transition(s) in another LPN.

Enabling and inhibitor interconnections are illustrated in Figures 3 and 4, respectively. Note that in these figures, each LPN is surrounded by a box. This boxing idea has also been used by e.g. Kindler (1997) or Fernandes et al (1997).

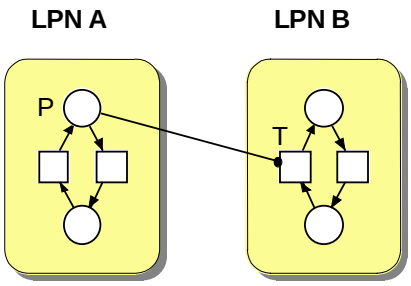

Fig. 3. Illustration of an enabling arc from one place in LPN A to one transition in LPN B.

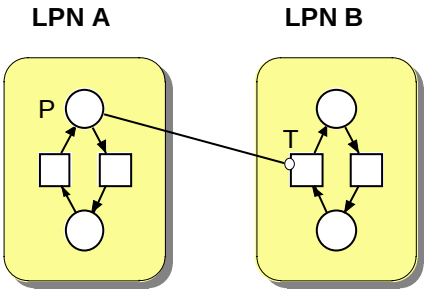

Fig. 4. Illustration of an inhibitor arc from one place in LPN A to one transition in LPN B.

Enabling and inhibitor arcs are used to describe how agents modelled by individual LPNs influence each other. The transition at the tip of the arc (i.e. transition T in LPN B in Figures 3 and 4) can only fire if the process modelled by LPN A is in a particular state or marking, and when it fires, it may use the information existing in this marking of LPN A. For example, it may appear that the guard or delay of transition T is dependent of the colour of the token in place P. In those cases, the Petri net graph needs to be extended with (enabling) arcs from place P to transition T in order to get access to this information (and the guard or delay of the transition, which in the previous subsection has only be defined locally, needs to be adapted). Since tokens are not consumed through enabling arcs at a transition firing, the state of LPN A is not changed through this firing. Fernandes et al (1997) uses enabling arcs like this to model synchronisation. Fota et al (1997) uses GSPN which do not support enabling arcs although they do support inhibitor arcs; however, Fota does allow tokens of other modules be consumed and immediately placed back, which is similar to using an enabling arc.

An Interaction Petri Net (IPN) consists of at least one place, and zero or more transitions. It connects, by means of ordinary arcs, one or more transition(s) in one LPN with one or more transition(s) in another LPN. If there are transitions in the IPN, and if these transitions are connected with other LPNs, then only enabling or inhibitor arcs can be used for the connections of these transitions with other LPNs. An example of an IPN is illustrated in Figure 5. It can be easily verified that an IPN does not influence the number of tokens in the LPNs it connects.

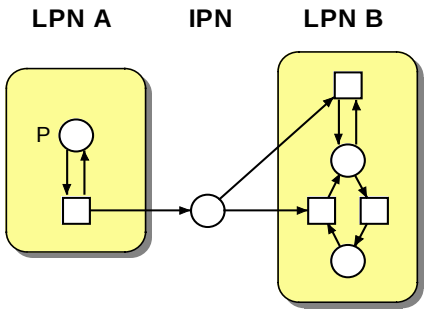

Fig. 5. Illustration of an Interaction Petri Net from one transition in LPN A to two transitions in LPN B.

Interaction Petri Nets are used when enabling or inhibitor arcs are insufficient to model the interconnection between two agents. For example, it can hold on to state information from its input LPN (i.e. LPN A in Figure 5) while the state of LPN A itself evolves further. Also, IPNs can be used to connect two transitions, while enabling or inhibitor arcs always connect a place with a transition. Note that our IPNs are similar to the Interconnection blocks of Fota et al (1997). The connection of two LPNs through an Interaction Petri Net also has some similarity with Place Fusion, see e.g. Huber et al (1990) or Kindler (1997), except that our Interaction Petri Net will not change the number of tokens in its connecting LPNs.

5. Extension with interconnection mapping types

Interconnections between LPNs through enabling (or inhibitor) arcs and IPNs might lead to a combinatorial growth of the number of interconnections with the size of the Petri net. To avoid this combinatorial growth as much as possible, in this section hierarchical clustering and interconnection mapping approaches are graphically developed, based on how Harel (1987) composes statecharts:

1. Interconnection mapping types I and II are defined to avoid possible duplication of transitions and arcs within LPNs caused by specifying interconnections between LPNs.
2. Interconnection mapping types III, IV and V are defined to avoid cluttering of interconnections between places and transitions of different LPNs.
3. Interconnection mapping types VI and VII define interconnections from or to hierarchical clusters of LPNs, which reduce the cluttering of interconnections.
4. Combinations of interconnection mapping types, and an additional interconnection mapping type VIII that avoids a duplication of transitions and arcs within an LPN and duplication of arcs between LPNs.

In Section 6, the DCPN definition is extended to include these interconnection mapping types.

5.1 Avoid duplication of transitions and arcs within an LPN

Figure 6 shows an example where interconnections between LPNs lead to duplication of transitions and arcs within one of these LPNs. A transition from place P3 to place P4 occurs if either P1 *or* P2 contains a token. To model this, it is necessary to use two transitions T1 and T2 between P3 and P4. The use of only one transition between P3 and P4 would model an 'and' relation (i.e. both P1 *and* P2 contain a token) instead of an 'or' relation.

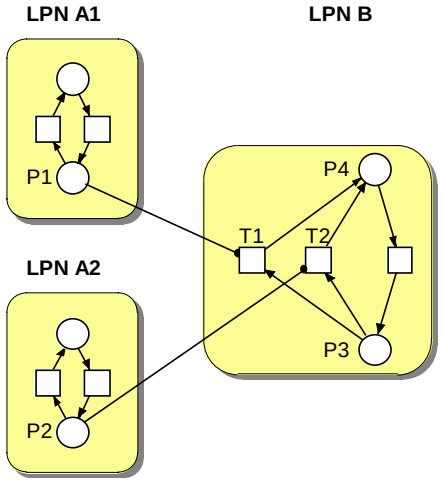

Fig. 6. Illustration of duplication of transitions within an LPN.

In most cases, the duplicated transitions and arcs do not have an essentially different meaning, and they are mostly introduced to be able to make use of colours of tokens residing in other LPNs. In particular, these duplicated transitions have the same guard or delay and the same firing function. This makes that duplication leads to reduced readability. This subsection presents some interconnection mapping types to avoid such duplication.

LPN interconnection mapping type I

A set of *s* enabling arcs initiating on *s* places, merging into one arc and ending on one transition, means that this transition is duplicated *s* times, and that *s* enabling arcs are drawn between the *s* places and the *s* resulting transitions. This type of arc is called *merging arc*. The transition at the end of the merging arc should be in a different LPN than the *s* places that are at the beginning of the arc. Figure 7 shows an example of this interconnection mapping type. Note that in order to avoid confusion when using this interconnection mapping type, the *s* duplicated transitions should have the same guard or delay function and the same firing function and their input places should have the same colour type. Interconnection mapping type I is not defined with inhibitor or ordinary arcs instead of enabling arcs.

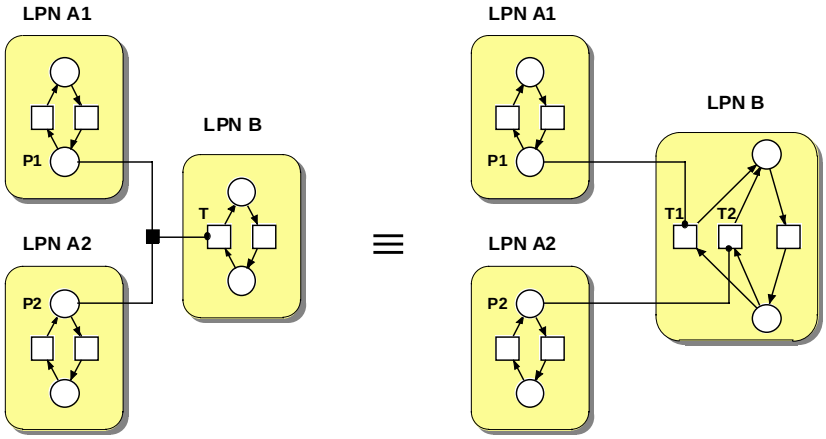

Fig. 7. LPN interconnection[2] mapping type I. The point where several arcs merge into one arc is represented by a small black square.

LPN interconnection mapping type II

An enabling arc initiating on the edge of an LPN box and ending on a transition in another LPN box, means that enabling arcs initiate from all places in the first LPN and end on duplications of this transition in the second LPN. Figure 8 shows an example of this interconnection mapping type. The duplicated transitions should have the same guard or delay function and the same firing function and their input places should have the same colour type. Interconnection mapping type II is not defined with inhibitor or ordinary arcs instead of enabling arcs.

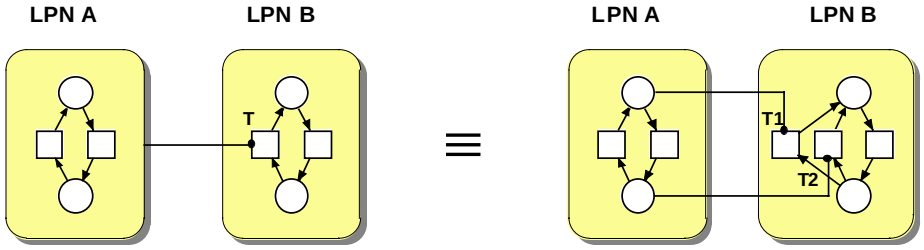

Fig. 8. LPN interconnection mapping type II.

[2] The interconnection mapping types are illustrated by figures where the application of the interconnection mapping type is shown in the left part of the figure.

5.2 Avoid cluttering of interconnections between LPNs

The interconnection mapping types in the previous subsection avoid the duplication problem, but not the cluttering due to the many enabling arcs and IPNs between places and transitions of different LPNs. If several LPNs are interconnected in one graph the result becomes unreadable. This subsection presents some interconnection mapping types to avoid this:

- Interconnection mapping type III can be applied to avoid enabling arcs cluttering.
- Interconnection mapping types IV and V can be applied to avoid IPNs cluttering.

LPN interconnection mapping type III

An enabling arc ending on the edge of an LPN box, means that enabling arcs end on each transition in this LPN. Figure 9 shows an example of this interconnection mapping type. Interconnection mapping type III can also be used with inhibitor arcs instead of enabling arcs. It cannot be used with ordinary arcs, due to the restriction that the number of tokens in an LPN should remain the same.

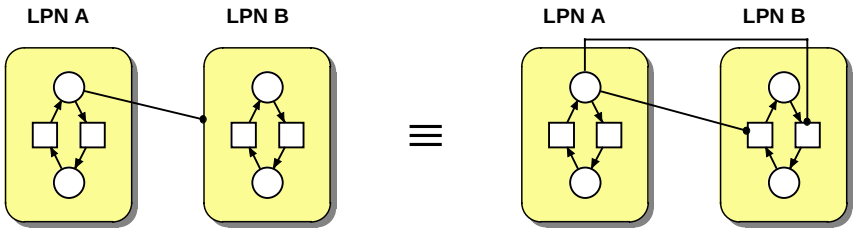

Fig. 9. LPN interconnection mapping type III.

LPN interconnection mapping type IV

An ordinary arc initiating on the edge of an LPN box and ending on a place within an IPN means that ordinary arcs initiate from all transitions in this LPN. Figure 10 shows an example of this interconnection mapping type. Interconnection mapping type IV is not defined with enabling or inhibitor arcs instead of ordinary arcs.

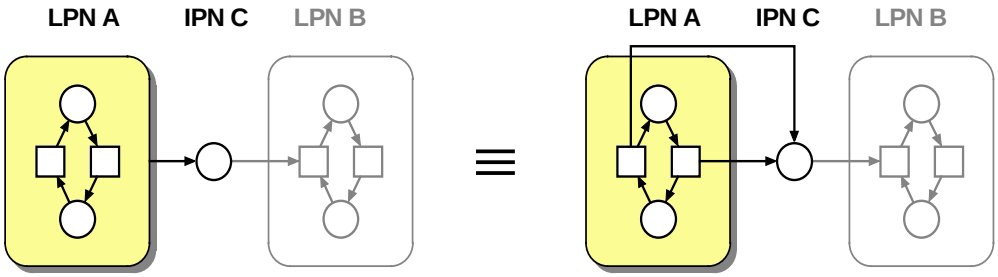

Fig. 10. LPN interconnection mapping type IV.

LPN interconnection mapping type V

An ordinary arc ending on the edge of an LPN box and starting from a place within an IPN means that ordinary arcs end on each transition in this LPN. Figure 11 shows an example of this interconnection mapping type. Interconnection mapping type IV is not defined with enabling or inhibitor arcs instead of ordinary arcs.

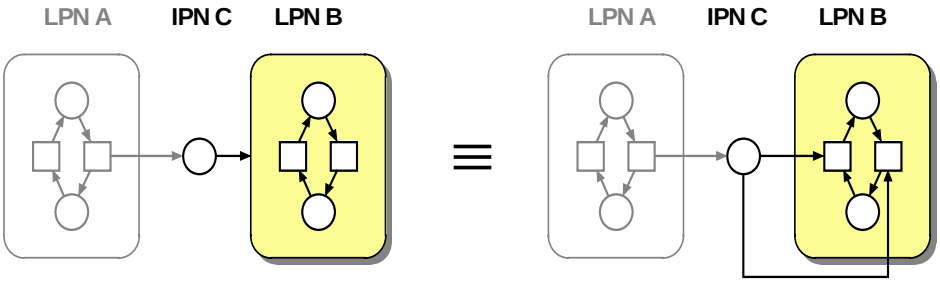

Fig. 11. LPN interconnection mapping type V.

5.3 Clustering of LPNs

In this subsection, we define enabling arcs that go from or to a cluster of LPNs. This is done following the next two interconnection mapping types. Figures 12 and 13 show examples of these interconnection mapping types.

LPN interconnection mapping type VI
Suppose there is one LPN A and a set of n LPNs B_i (i=1,2, ... n) which is enclosed by a large box. An enabling arc initiating on the edge of LPN A and ending on the edge of the large box with the set of LPNs B_i, means that the enabling arc represents n actual enabling arcs, initiating on the edge of LPN A and ending on the edge of each LPN B_i. Interconnection mapping type VI can also be defined from a place to a large box of LPNs (by means of an enabling arc), or from a place within an IPN to a large box of LPNs (by means of an ordinary arc). It is not defined with inhibitor arcs instead of enabling arcs. Note that the right hand side of Figure 12 makes use of a combination of interconnection mapping types II and III. For more examples of such combinations, see Subsection 5.4.

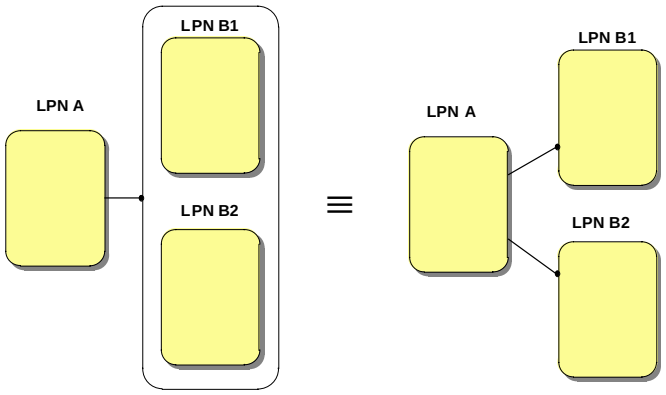

Fig. 12. LPN interconnection mapping type VI.

LPN interconnection mapping type VII
Suppose there is a set of n LPNs A_i (i=1,2, ... n) which is enclosed by a large box and one LPN B. An enabling arc initiating on the edge of the large box with the set of LPNs A_i and ending on the edge of LPN B, means that the enabling arc represents n actual enabling arcs, initiating on the edge of each LPN A_i and ending on the edge of LPN B. Interconnection mapping type VII can also be defined from a large box to a transition. It is not defined with ordinary or inhibitor arcs instead of enabling arcs.

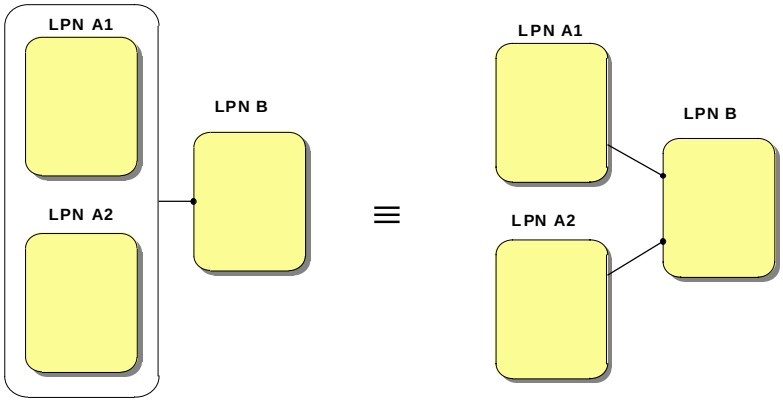

Fig. 13. LPN interconnection mapping type VII.

5.4 Combinations of interconnection mapping types and additional type

Interconnection mapping types can also be combined, such as interconnection mapping type I with II, type II with III, type IV with V, or type VI with VII. An illustration of combination of II with III is given below.

LPN interconnection mapping types II and III combined
An enabling arc initiating on the edge of an LPN box and ending on the edge of another LPN box, means that enabling arcs initiate from all places in the first LPN and end on duplications of all transitions in the second LPN. Figure 14 shows an example of this combination of interconnection mapping types II and III.

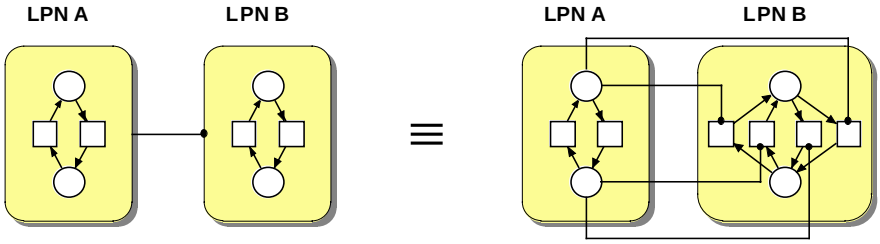

Fig. 14. LPN interconnection mapping types II and III combined.

Finally, we introduce an additional interconnection mapping type which avoids duplication of transitions and arcs within an LPN, and consequently cluttering of arcs between LPNs:

LPN interconnection mapping type VIII
An ordinary arc initiating on the edge of an LPN box and ending on a transition inside the same box, means that ordinary arcs initiate from all places in the LPN box to duplications of this transition. The duplicated transitions should have the same guard or delay function and the same firing function and their set of input places should have the same set of colour types. Figure 15 illustrates how this avoids both the duplication of transitions and arcs within an LPN, and the duplication of arcs between LPNs.

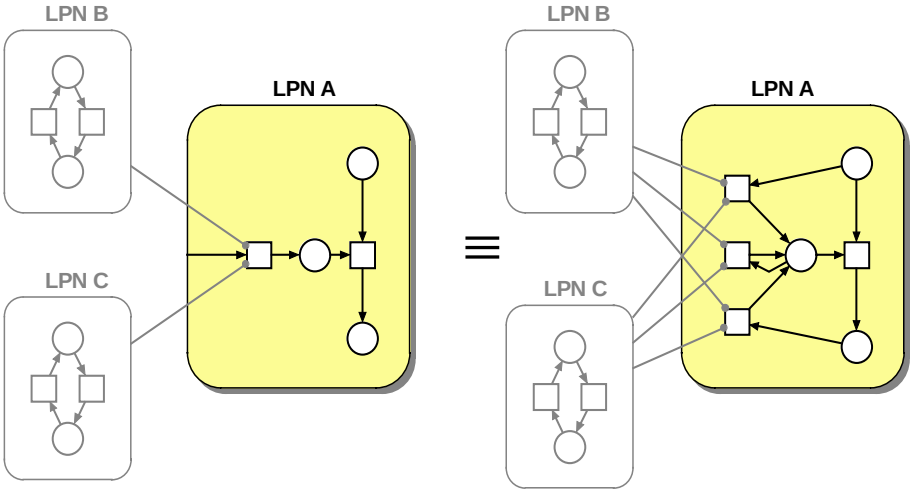

Fig. 15. LPN interconnection mapping type VIII, which avoids duplication of arcs and transitions within an LPN and duplication of arcs between LPNs.

Remark: The interconnection mapping types introduced in this section could also be used for other types of Petri nets than DCPN, provided that these other types of Petri nets support the same graphical elements as DCPN, such as enabling arcs. If this is not the case, the interconnection mapping types might still be used, but then the restriction that the number of tokens in an LPN cannot be changed by the interconnections must be removed.

6. Extension of DCPN with interconnection mapping types I through VIII

This subsection extends the DCPN definition of Subsection 3.1 to include the interconnection mapping types identified in Section 5. The extension is referred to as DCPN^{imt}.

DCPN^{imt} is a tuple (P, T, B, A^{imt}, L, N^{imt}, S, C, V, G, D, F, I), where P, T, S, C, V, G, D, F, I are as in the definition of DCPN (Subsection 3.1), and the other elements are outlined below:

A^{imt} is the set of arcs in the DCPN^{imt}. It equals the set of arcs $A = A_o \cup A_e \cup A_i$ as defined in Subsection 3.1, but extended with a set of merging arcs (A_m). In other words, $A^{imt} = A \cup A_m$.

A merging arc is a set of $s \geq 2$ enabling arcs merging into one arc, where s can be different for each merging arc. The merging point is denoted by a small black square.

B is a set of boxes which consists of a set B_L of LPN-boxes and a set B_C of Cluster-boxes.

Each box in B is drawn as a rectangle with rounded corners. Note that at this definition level, each element of B is just an empty box. The box function L (see definition next) will specify the actual contents (i.e. the places and transitions or other boxes) of each box in B. There, each LPN-box in B_L will be associated with an LPN, and each cluster-box in B_C will be associated with a cluster of LPNs.

L is a box function which specifies the contents of each box in B: L maps each place in P to zero or one box in B, each transition in T to zero or one box in B, and each box in B_L to zero or one box in B_C. Places (and transitions) that form IPNs are not mapped to an LPN-box but can be mapped to a cluster-box, and at least two LPN-boxes should be mapped to each cluster-box.

For each LPN-box in B_L, the box function specifies which places in P and which transitions in T are drawn in it to form an LPN; for each cluster-box in B_C it specifies which (at least two) LPN-boxes in B_L are drawn in it to form a cluster of LPNs. Some places (and transitions) are not inside any LPN-box; these form the IPNs. It is however possible that IPNs are part of a cluster-box (although they are not part of an LPN-box). Similarly, not all LPN-boxes need to be inside a cluster-box.

N^{imt} is a node function which maps each arc in A_m to an ordered pair of which the first component is a set of places (but not in IPNs) or boxes, and the second component is a transition. Furthermore, N^{imt} maps each arc in $A = A_o \cup A_e \cup A_i$, to an ordered pair of nodes, where a node is a place, a transition, an LPN-box or a cluster-box. Multiple arcs between the same pair of nodes are allowed (but not both an inhibitor arc and another type of arc).
There are the following restrictions:

- Ordinary arcs can only be drawn from a place to a transition within the same LPN-box, from a transition to a place within the same LPN-box, from a place in an IPN to a transition, from a transition to a place in an IPN, from a place in an IPN to an

LPN-box, from an LPN-box to a place in an IPN, from a place in an IPN to a cluster-box, or from an LPN box to a transition in the same LPN box.

- Enabling arcs can only be drawn from a place to a transition within the same LPN-box, from a place to a transition in a different LPN-box or in an IPN, from a place (but not in an IPN) to an LPN- or cluster-box, from an LPN- or cluster-box to a transition, or between two boxes (i.e. LPN-LPN, LPN-cluster, cluster-LPN or cluster-cluster).
- Inhibitor arcs can only be drawn from a place to a transition within the same LPN-box, from a place to a transition in a different LPN-box or in an IPN, or from a place (but not in an IPN) to an LPN box.
- Merging arcs can only be drawn from a set of places (but not in IPNs) or boxes, to a transition that is in another LPN than these places or boxes. The input places of a merging arc should be of the same type.

Note that the guards G, delays D, and firing functions F defined for DCPN^{imt} are equal to those defined for DCPN. However, since the elements G, D, and F use the colours of the transition input tokens as input, their evaluation is a little more complicated in the sense that from the DCPN^{imt} graph it is not immediately obvious which places are the input and output places of the transitions. These input and output places become clear if the DCPN^{imt} graph is extended to a DCPN graph, i.e. the cluttered one without the interconnection mapping types. Some rules that avoid this for the most-often used interconnection mapping types are given below. Here, only the between-LPN interconnections are considered. The pre-enabling, enabling or firing of each transition is also dependent on the colours of the input tokens along the within-LPN connections, but to keep the description brief, these are not considered here.

- If a transition has an incoming merging arc (see e.g. interconnection mapping type I), it is pre-enabled if it has a token in at least one of the places also connected to this merging arc. The transition is enabled if it is enabled by this input token as described in Subsection 3.1 (e.g. its guard evaluates to true, or its delay has passed). If there are tokens in several of these input places, the transition guard or delay function uses their colours in parallel for its evaluation.
- If a transition has an (enabling) incoming arc connected with an LPN-box (see e.g. interconnection mapping type II), then it is pre-enabled if there is at least one token somewhere in this input LPN-box (and this is usually the case). It is enabled if it is enabled by this input token as described in Subsection 3.1.
- If the LPN-box in which a transition resides has an input place (see e.g. interconnection mapping type III), then the transition is pre-enabled if there is a token in this input place, and its guard or delay uses the colour of this token for its evaluation as described in Subsection 3.1.
- If the LPN-box in which a transition resides has an input LPN-box (see e.g. interconnection mapping type II combined with III), then the transition is pre-enabled if there is at least one token somewhere in the input LPN-box (and this is usually the case) and its guard or delay uses the colour of this token for its evaluation as described in Subsection 3.1.

7. Free Flight air traffic example

The compositional specification approach described has been used to specify an initial $DCPN^{imt}$ for a risk assessment the of Free Flight based air traffic operation adopted in (Klein Obbink, 2004). Free Flight –sometimes referred to as Self Separation Assurance– is a concept where pilots are allowed to select their trajectory freely at real time, at the cost of acquiring responsibility for conflict prevention. It changes ATM in a fundamental way: the centralised control becomes a distributed one, responsibilities and tasks transfer from ground to air, ATC sectorization and routes are removed and new technologies are brought in. Before such concept can be implemented, it is necessary to determine its level of safety. The aim of this section is to illustrate the $DCPN^{imt}$ specification developed in (Bakker et al., 2004) for a collision risk model for Free Flight.

7.1 LPNs of the Free Flight air traffic example

In the Free Flight air traffic example, the airspace is an En-Route Airspace without fixed routes or an active ATC specifying routes. All aircraft flying in this airspace are assumed to be properly equipped and enabled for Free Flight: the pilots can try to optimise their trajectory, due to the enlarged freedom to choose path and flight level. The pilots are only limited by their responsibility to maintain airborne separation, in which they are assisted by a system called ASAS (Airborne Separation Assistance System). This can be considered as a system processing the information flows from the data-communication links between aircraft, the navigation systems and the aircraft guidance and control systems. ASAS detects conflicts, determines conflict resolution manoeuvres and presents the relevant information to the aircrew.

The number of agents involved in the Free Flight operation is huge and ranges from the Control Flow Management Unit to flight attendants. In the setting chosen for an initial risk assessment, the following agents are taken into account:

- A Pilot-Flying in each aircraft,
- A Pilot-Non-Flying in each aircraft,
- A number of systems and entities per aircraft, like the aircraft's position evolution and the Conflict Management Support systems,
- A number of global systems and entities, like the communication frequencies and the satellite system.

As explained in the beginning of Section 4, LPNs are specified for each relevant functional entity of each agent. It was judged sufficient to specify the following number of LPNs for the agents:

- 6 LPNs for each Pilot-Flying,
- 2 LPNs for each Pilot-Non-Flying,
- 36 LPNs for the systems and entities of each aircraft,
- 7 LPNs for the environment.

The actual number of LPNs in the whole model then depends on the number N of aircraft involved, and equals $7 + N \times (6 + 2 + 36)$.

7.2 Interconnected LPNs of "Pilot Flying"

This subsection illustrates, for the specific Free Flight air traffic example, a Petri Net model for the Pilot Flying as agent. A graphical representation of all LPNs the Pilot-Flying consists of, is given in Figure 16 below. The Human-Machine-Interface where sound or visual clues might indicate that attention should be paid to a particular issue, is represented by a LPN that does not belong to the Pilot-Flying as agent and is therefore not depicted in the Figure. Similarly, the arcs to or from any other agent are not shown in Figure 16. Because of the very nature of Petri Nets, these arcs can easily be added during the follow-up specification cycle. To get an understanding of the different LPNs, a good starting point might be the LPN "Current Goal" (at the bottom of the figure) as it represents the objective the Pilot-Flying is currently working on. Examples of such goals are "Collision Avoidance", "Conflict Resolution" and "Horizontal Navigation". For each of these goals, the pilot executes a number of tasks in a prescribed or conditional order, represented in the LPN "Task Performance". Examples of such tasks are "Monitoring and Decision", "Execution" and "Execution Monitoring". If all relevant tasks for the current goal are considered executed, the pilot chooses another goal, thereby using his memory (where goals deserving attention might be stored, represented by the LPN "Goal Memory") and the Human-Machine-Interface. His memory where goals deserving attention might be stored is represented as the LPN "Goal Memory" in Fig. 16.

So, the LPNs "Current Goal", "Task Performance", and "Goal Memory" are important in the modelling of which task the Pilot-Flying is executing. The other three LPNs are important in the modelling on how the Pilot-Flying is executing the tasks. The LPN "State SA", where SA stands for Situation Awareness, represents the relevant perception of the pilot about the states of elements in his environment, e.g. whether he is aware of an engine failure. The LPN "Intent SA" represents the intent, e.g. whether he needs to leave the Free Flight Airspace. The LPN "Cognitive mode" represents whether the pilot is in an opportunistic mode, leading to a high but error-prone throughput, or in a tactical mode, leading to a moderate throughput with a low error probability.

There are many interactions (which, in some cases, are complex) between these individual LPNs, which are depicted as enabling arcs and IPNs with one place only. The use of the new interconnection mapping types makes that the figure is still readable. Interconnection mapping types I, IV, V, VI and VII have not been used, while type II has been used 2×, type III 4×, and type VIII 3×. Table 1 shows that without the use of these interconnection mapping types the figure really would be cluttered with duplicated transitions and arcs within LPNs, and with connections drawn between LPNs.

Table 1. Numbers of interconnection mapping types and Petri net elements before and after application of interconnection mapping types. The number of places (i.e. 19 places within LPNs and 8 places between LPNs) does not change due to the interconnection mapping types.

Number of elements	In Figure 16	Without interconnection mapping types
Within LPNs	27 transitions 66 arcs	279 transitions 642 arcs
Between LPNs	16 ordinary arcs 7 enabling arcs 1 inhibitor arc	293 ordinary arcs 1023 enabling arcs 7 inhibitor arcs
Total	117	2244

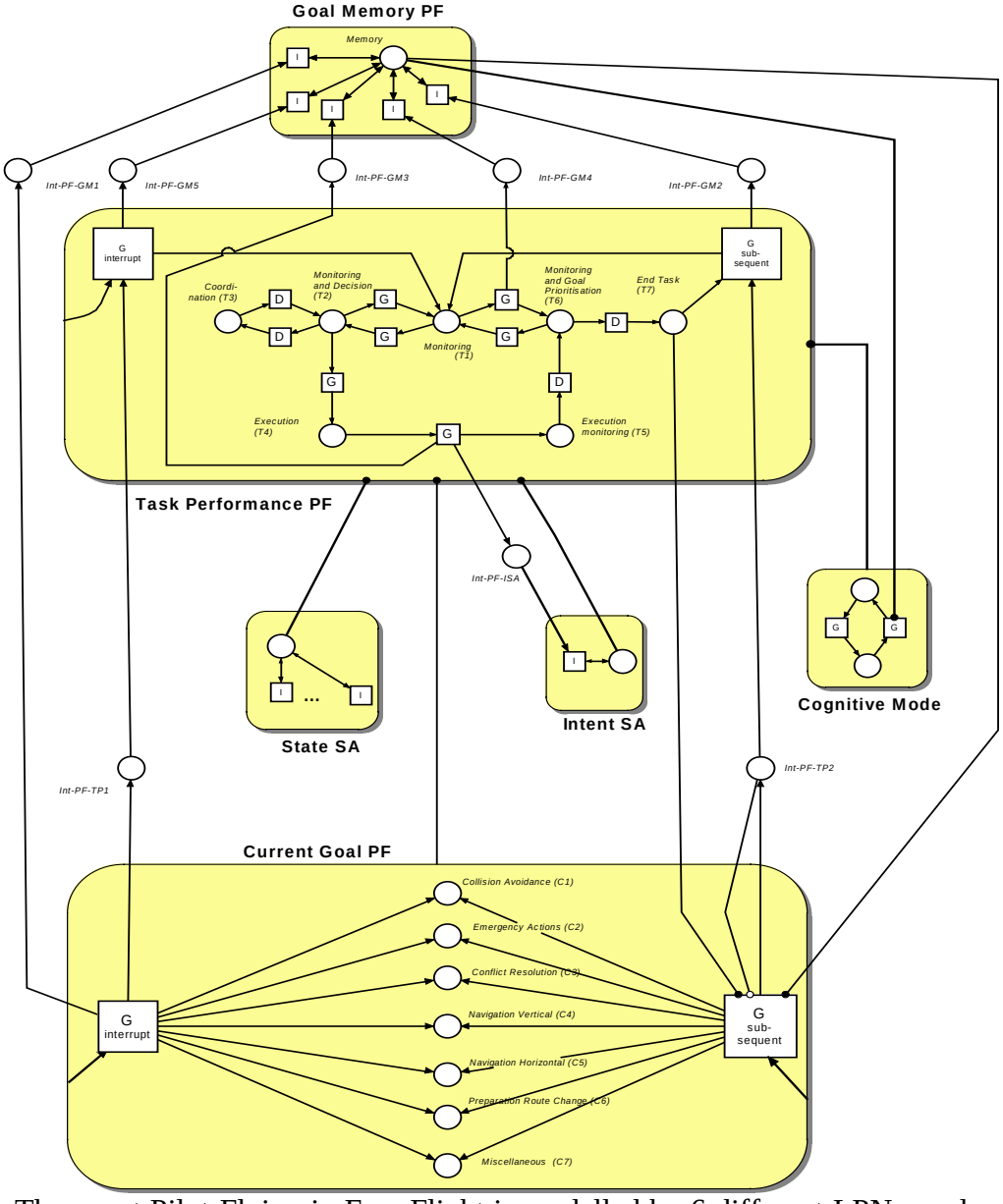

Fig. 16. The agent Pilot-Flying in Free Flight is modelled by 6 different LPNs, and a number of ordinary and enabling arcs and some IPNs, consisting of one place and input and output arcs.

8. Concluding remarks

For the compositional specification of a multi-agent hybrid system this paper has introduced a hierarchical extension of the compositional specification power of Petri nets, which avoids the need for all kinds of low level changes once making connections at a higher model level. Moreover the problem of combinatorial growth of the number of interconnections with the size of the Petri net is remedied. The effectiveness of the DCPN based compositional specification is illustrated for an air traffic example of a taxiing aircraft crossing an active runway.

After a DCPN^{imt} has been specified for a particular application, the next step is to analyse it, and to investigate and assess particular characteristics of the application. This can be done in various ways; for example, the DCPN^{imt} can be directly used as basis of a computer (e.g. Monte Carlo) simulation. Instead of this, or in addition to this, a particular DCPN property can be used: In (Everdij & Blom, 2003a, 2003b) it has been proven that, under a few conditions, a DCPN is equivalent to a particular powerful subclass of hybrid state Markov process, named Piecewise Deterministic Markov Process (PDP, see Davis, 1984, 1993). Due to this equivalence, typical PDP properties can be used to analyse the DCPN^{imt}, even without elaborating the particular transformation from DCPN^{imt} to PDP for the application considered, and Monte Carlo simulations can be run which make use of PDP mathematical properties. By those means, a collision risk model for Free Flight is instantiated using Petri-nets and the new interconnection mapping types. The usage of the new interconnection mapping types improves simplicity, readability and resilience against modelling errors.

For the Free Flight application it was necessary to add Brownian motion terms to the place specific colour functions V of the DCPN specified. As explained in [Everdij and Blom, 2004], by doing so, the resulting DCPN extension becomes a Stochastically Coloured Petri Net (SCPN).

9. References

1. Bakker, G.J., B. Klein Obbink, M.B. Klompstra, H.A.P. Blom, DCPN specification of a free flight air traffic operation, working document, HYBRIDGE WP9.2, Version 0.3, August 2004.
2. Blom, H.A.P., M.B. Klompstra, G.J. Bakker, Accident risk assessment of simultaneous converging instrument approaches, Air Traffic Control Quarterly, Vol. 11, No. 2, pp. 123-155, 2003b.
3. Blom, H.A.P., S.H. Stroeve, M.H.C. Everdij, M.N.J. Van der Park, Human cognition performance model to evaluate safe spacing in air traffic, Human Factors and Aerospace Safety, Vol. 2, pp. 59-82, 2003c.
4. Cacciabue, P.C., Modelling and simulation of human behaviour in system control, Advances in Industrial control, Springer-Verlag, London, 1998.
5. David, R., H. Alla, Petri Nets for the modeling of dynamic systems - A survey, Automatica, Vol. 30, No. 2, pp. 175-202, 1994.
6. Davis, M.H.A., Piecewise Deterministic Markov Processes: a general class of non-diffusion stochastic models, Journal Royal Statistical Soc. (B), Vol. 46, pp. 353-388, 1984.
7. Davis, M.H.A., Markov models and optimization, Chapman and Hall, 1993.
8. Everdij, M.H.C., H.A.P. Blom, Piecewise Deterministic Markov Processes represented by Dynamically Coloured Petri Nets, NLR-TP-2000-428, National Aerospace Laboratory NLR, Revised edition, 2003a.
9. Everdij, M.H.C., H.A.P. Blom, Petri Nets and Hybrid state Markov Processes in a power-hierarchy of dependability models, Proc. IFAC Conference on Analysis and Design of Hybrid Systems, Saint-Malo Brittany, France, pp. 355-360, June 2003b. Preprint available on http://www.nlr.nl/public/hosted-sites/hybridge/
10. Everdij, M.H.C., H.A.P. Blom, Modelling hybrid state Markov processes through Dynamically and Stochastically Coloured Petri Nets, HYBRIDGE WP2.4 report, National Aerospace Laboratory NLR, 2002, 2004.
11. Fernandes, M., M. Adamski, and A.J. Proença. VHDL Generation from Hierarchical Petri Net Specifications of Parallel Controller. IEE Proceedings: Computers and Digital Techniques, Vol. 144, pp. 127-137, March 1997.
12. Fleischhack, H., B. Grahlmann, A Petri net semantics for $B(PN)^2$ with procedures, Parallel and Distributed Software Engineering, 1997.
13. Fota, N.O., M. Kaaniche and K. Kanoun, A modular and incremental approach for building complex stochastic Petri net models, Proc. First Int. Conf. on Mathematical Methods in Reliability, 1997.
14. Haas, P.J., Stochastic Petri Nets, Modelling, Stability, Simulation, Springer-Verlag, New York, 2002.
15. Harel, D., Statecharts: a visual formalism for complex systems. Science of Computer Programming, Vol. 8, pp. 231-274, 1987.
16. Hollnagel, E., Human reliability analysis, context and control. Academic Press, London, 1993.
17. Huber, P., K. Jensen, R.M. Shapiro, Hierarchies in Coloured Petri Nets, G. Rozenberg (ed.): Advances in Petri nets 1990. Lecture notes in Computer Science, Vol. 483. Springer, Berlin Heidelberg New York 1990, pp. 313-341.
18. Kindler, E., A compositional partial order semantics for Petri net components, 18th International Conference on Application and Theory of Petri nets, Pierre Azema and Gianfranco Balbo (ed.), LNCS, 1248, Springer-Verlag, June 1997.

19. Klein Obbink, B., Description of advanced operation: Free Flight, HYBRIDGE WP9.1 report, National Aerospace Laboratory NLR, 2002, 2004
20. Labeau, P.E., C. Smidts, S. Swaminathan, Dynamic reliability: towards an integrated platform for probabilistic risk assessment, Reliability Engineering & Systems Safety, Vol. 68, 2000, pp. 219-254.
21. Royal Society, Risk assessment, Report of a Royal Society Study Group, London, 1983..
22. Stroeve, S.H., H.A.P. Blom, M.N.J. Van der Park, Multi-agent situation awareness error evolution in accident risk modelling, Proc. of the 5th Eurocontrol / FAA ATM R&D Seminar, Budapest, Hungary, 2003.