Modular system development by composing Petri nets on interfaces

Lucia Pomello

Dipartimento di Informatica, Sistemistica e Comunicazione

University of Milano – Bicocca

Moscow, 21/04/2014

Joint work with

- Luca Bernardinello – DISCo, University of Milano–Bicocca
- Carlo Ferigato – JRC European Commission, Ispra
- Görkem Kılınç – Phd student, Univ. Milano – Bicocca
- Elisabetta Mangioni – Phd, now CNR, Milano
- Elena Monticelli – former Master Thesis student.
- Stefano Scacabarozzi – former Master Thesis student.

Modular system development by composing Petri nets on interfaces

Outline

Section 1

Composing on interfaces, the intuition

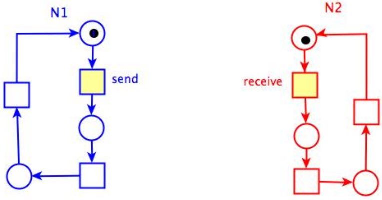
Composition operations for Petri nets

$$N = (P, T, F, M_0)$$

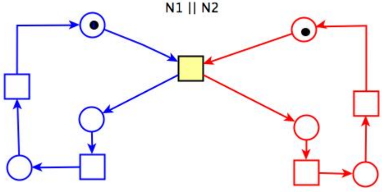
typical ways of composing nets

- synchronous
- asynchronous
- mixed

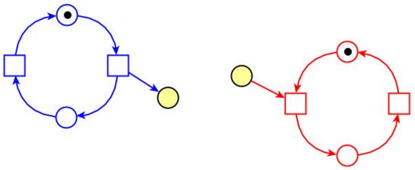
Synchronous Composition ||



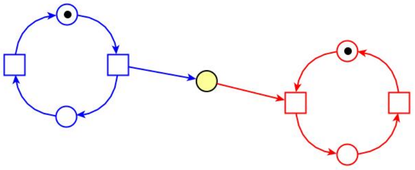
merging transitions (synchronization)



Asynchronous Composition

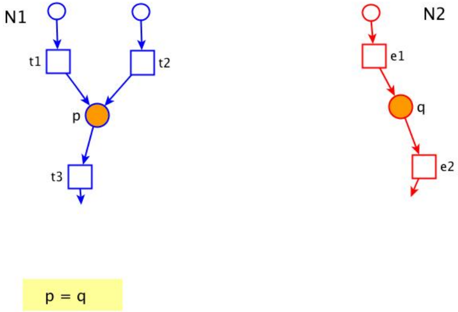


merging places (channels)



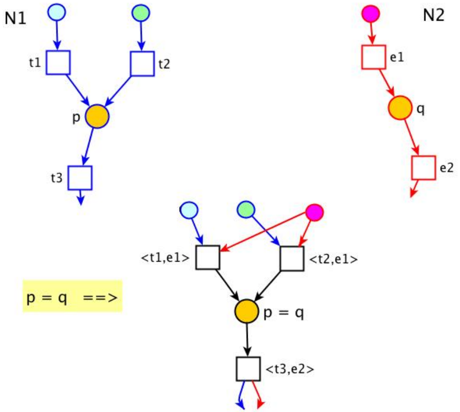
Composing on interfaces, the intuition...

Elementary net systems $N = (B, E, F, c_{in})$:
- B: conditions (boolean propositions) - E: events

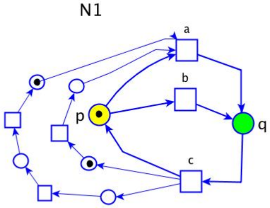


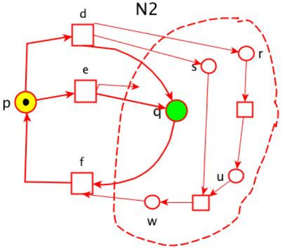
Composing on interfaces, the intuition...

Elementary net systems $N = (B, E, F, c_{in})$:
- B: conditions (boolean propositions) - E: events

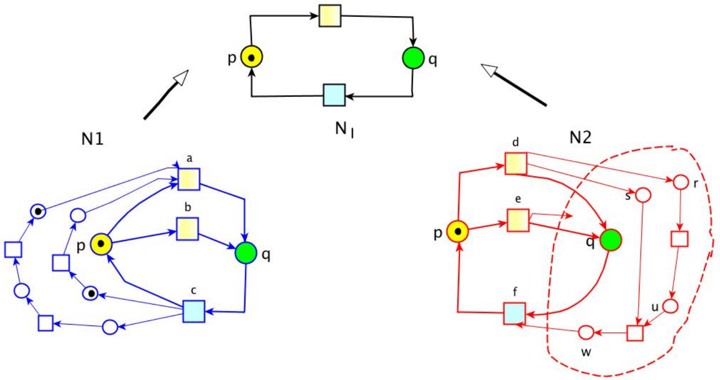


Composing on interfaces, the intuition...

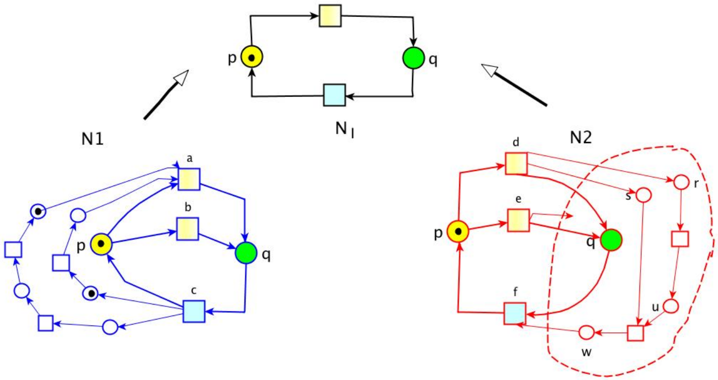




Composing on interfaces, by means of morphisms

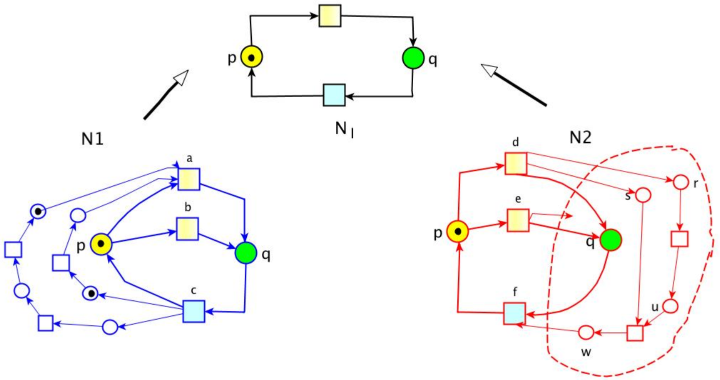


Composing on interfaces, by means of morphisms



abstracting from details

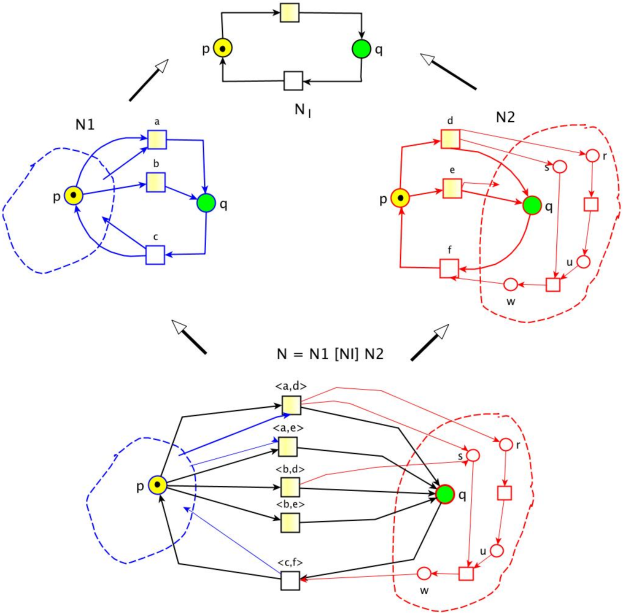
Composing on interfaces, by means of morphisms



abstracting from details

if q then $(s \wedge (r \vee u)) \vee w$

Composing on interfaces, by means of morphisms



Section 2

$\hat{N}$-morphisms: abstraction and refinement

the origins

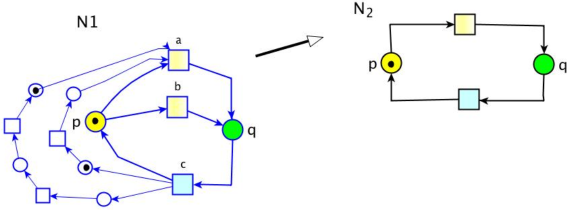
N-morphisms
[Nielsen, Rozenberg, Thiagarajan '92]

(intuitively) they *preserve* behaviours, i.e.:

if $\phi : N_1 \rightarrow N_2$ is a *N-morphism* then N_2 is *partially simulating* N_1.

$\hat{N}$-morphisms for EN systems [1 2]

$N_i = (B_i, E_i, F_i, c_{i,in}), \quad i = 1, 2$

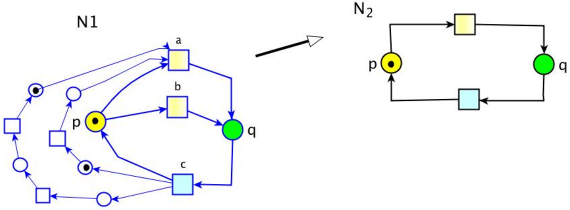


[1] L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in LNCS 3099, 77-96, Springer 2004.

[2] L.Bernardinello, E. Monticelli, L. Pomello, *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.

$\hat{\text{N}}$-morphisms for EN systems [1] [2]

$N_i = (B_i, E_i, F_i, c_{i,in}), \quad i = 1, 2$



$(\beta, \eta) : N_1 \rightarrow N_2$ is an $\hat{\text{N}}$-morphism iff

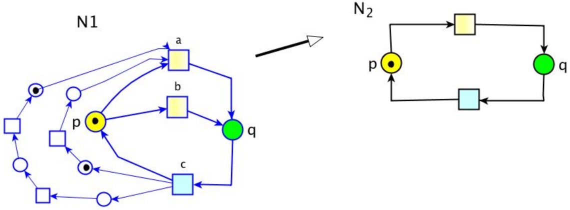
$\beta \subseteq B_1 \times B_2$ relation, $\quad \eta : E_1 \rightarrow E_2$ partial surjective map :

[1] L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in LNCS 3099, 77-96, Springer 2004.

[2] L.Bernardinello, E. Monticelli, L. Pomello, *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.

$\hat{\text{N}}$-morphisms for EN systems [1] [2]

$N_i = (B_i, E_i, F_i, c_{i,in}), \quad i = 1, 2$



$(\beta, \eta) : N_1 \to N_2$ is an $\hat{\text{N}}$-morphism iff

$\beta \subseteq B_1 \times B_2$ relation, $\quad \eta : E_1 \to E_2$ partial surjective map :

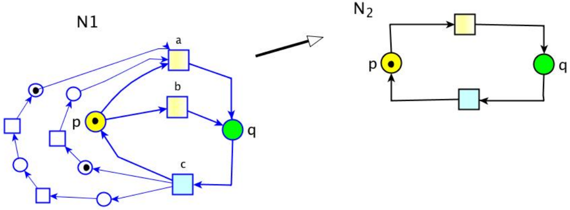
- $\beta^{-1} : B_2 \to B_1$ total injective map
- $\forall (b_1, b_2) \in \beta : b_1 \in c_{1,in} \Leftrightarrow b_2 \in c_{2,in}$

[1] L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in LNCS 3099, 77-96, Springer 2004.

[2] L.Bernardinello, E. Monticelli, L. Pomello, *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.

$\hat{\text{N}}$-morphisms for EN systems [1] [2]

$N_i = (B_i, E_i, F_i, c_{i,in}), \quad i = 1, 2$



$(\beta, \eta) : N_1 \rightarrow N_2$ is an $\hat{\text{N}}$-morphism iff

$\beta \subseteq B_1 \times B_2$ relation, $\eta : E_1 \rightarrow E_2$ partial surjective map :

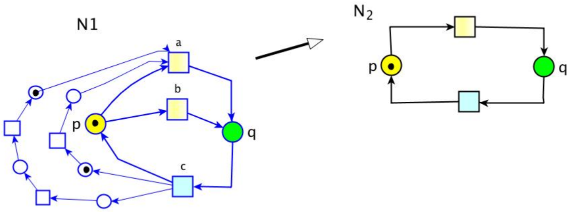
- $\beta^{-1} : B_2 \rightarrow B_1$ total injective map
- $\forall (b_1, b_2) \in \beta : b_1 \in c_{1,in} \Leftrightarrow b_2 \in c_{2,in}$
- $\eta(e_1) = e_2 \Rightarrow \beta({}^\bullet e_1) = {}^\bullet e_2$ and $\beta(e_1{}^\bullet) = e_2{}^\bullet$

[1] L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in LNCS 3099, 77-96, Springer 2004.

[2] L.Bernardinello, E. Monticelli, L. Pomello, *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.

$\hat{\text{N}}$-morphisms for EN systems [1 2]

$N_i = (B_i, E_i, F_i, c_{i,in}), \quad i = 1, 2$



$(\beta, \eta) : N_1 \to N_2$ is an $\hat{\text{N}}$-morphism iff

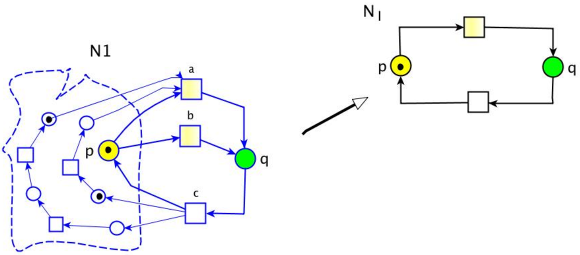
$\beta \subseteq B_1 \times B_2$ relation, $\quad \eta : E_1 \to E_2$ partial surjective map :

- $\beta^{-1} : B_2 \to B_1$ total injective map
- $\forall (b_1, b_2) \in \beta : b_1 \in c_{1,in} \Leftrightarrow b_2 \in c_{2,in}$
- $\eta(e_1) = e_2 \;\Rightarrow\; \beta({}^\bullet e_1) = {}^\bullet e_2$ and $\beta(e_1{}^\bullet) = e_2{}^\bullet$
- $\eta(e_1) = \perp \;\Rightarrow\; \beta({}^\bullet e_1) = \beta(e_1{}^\bullet) = \emptyset$

[1] L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in LNCS 3099, 77-96, Springer 2004.

[2] L.Bernardinello, E. Monticelli, L. Pomello, *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.

N̂-morphisms for EN systems



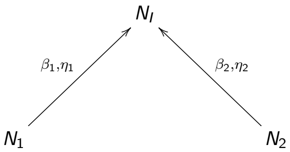
The counterimage of N_I, after T-simplification, is isomorphic to N_I.

N̂-morphisms: refinement /abstraction

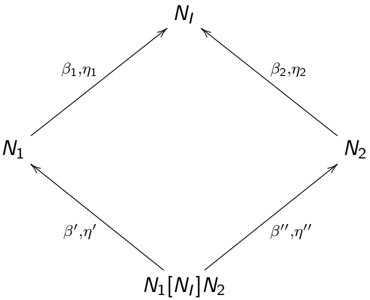
Section 3

Composing on interfaces by $\hat{N}$-morphisms

Composing two nets on an interface by $\hat{N}$-morphisms

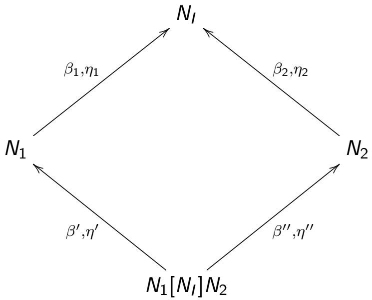


Composing two nets on an interface by N̂-morphisms



N̂-morphisms dictate the identification (composition) of elements

Composing two nets on an interface by N̂-morphisms

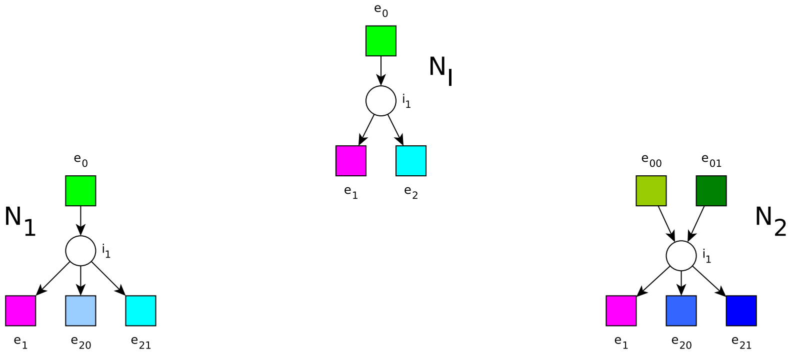


N̂-morphisms dictate the identification (composition) of elements

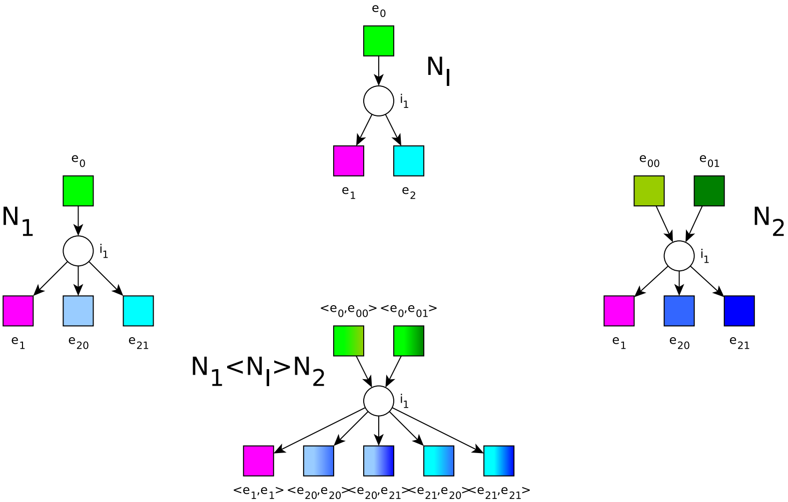
Proposition

The diagram commute

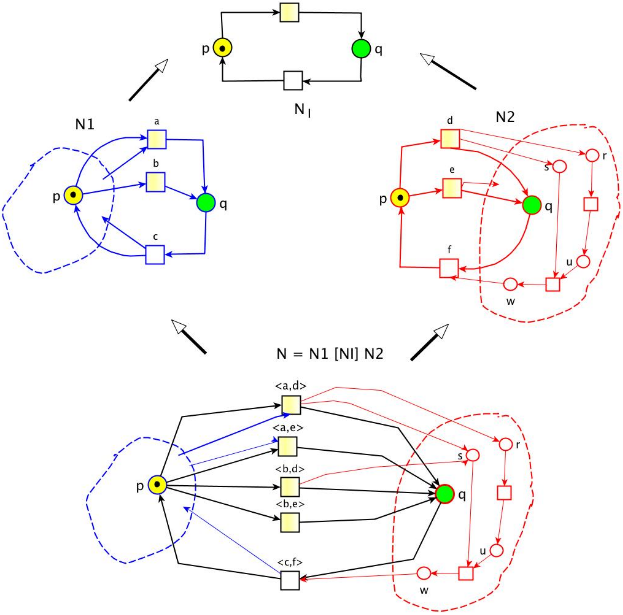
The composition: how to compose events



The composition: how to compose events

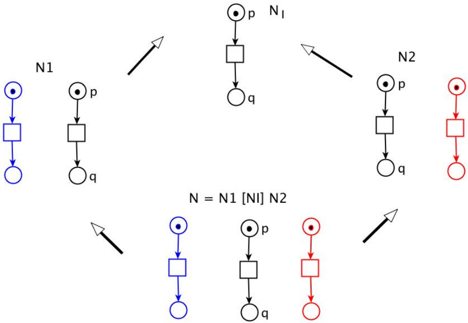


Composing on interfaces, by N̂-morphisms



Composition by N̂-morphisms is **not** a Pullback

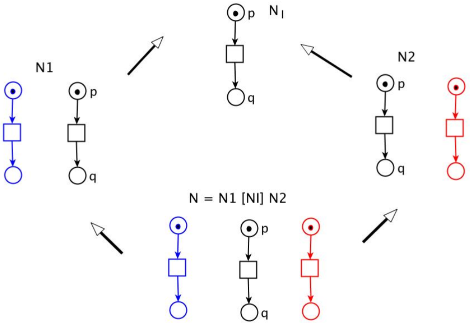
Example:



[3] M. A. Bednarczyk, L. Bernardinello, B. Caillaud, W. Pawlowski, L. Pomello, *Modular system development with pullbacks*, in ATPN2003, LNCS 2679, 140 - 160, Springer, 2003.

Composition by N̂-morphisms is **not** a Pullback

Example:



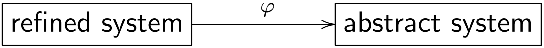
A pullback composition has been defined on a bit different morphisms/ composition [3]

[3] M. A. Bednarczyk, L. Bernardinello, B. Caillaud, W. Pawlowski, L. Pomello, *Modular system development with pullbacks*, in ATPN2003, LNCS 2679, 140 - 160, Springer, 2003.

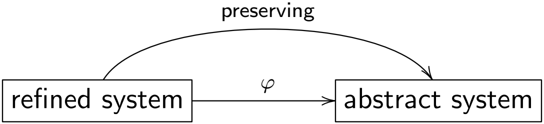
Section 4

Properties preserved/reflected

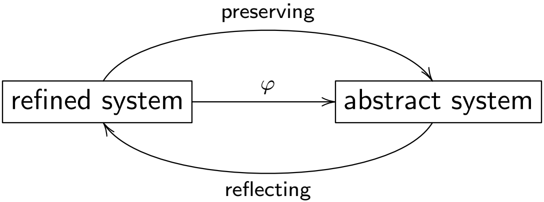
Preserving/reflecting properties



Preserving/reflecting properties



Preserving/reflecting properties



$\hat{\text{N}}$-morphisms: properties preserved/reflected

$(\beta, \eta) : N_1 \rightarrow N_2$

- S-invariants are reflected:
  if I_2 is an S-invariant of N_2, then $I_1 = \beta^{-1}(I_2)$ is an S-invariant of N_1

$\hat{\text{N}}$-morphisms: properties preserved/reflected

$(\beta, \eta) : N_1 \rightarrow N_2$

- S-invariants are reflected:
  if I_2 is an S-invariant of N_2, then $I_1 = \beta^{-1}(I_2)$ is an S-invariant of N_1
- S-invariants are **not** preserved

$\hat{\text{N}}$-morphisms: properties preserved/reflected

$(\beta, \eta) : N_1 \to N_2$

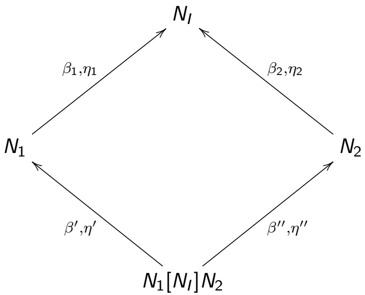
- S-invariants are reflected:
  if I_2 is an S-invariant of N_2, then $I_1 = \beta^{-1}(I_2)$ is an S-invariant of N_1
- S-invariants are **not** preserved
- T-invariants are preserved:
  if J_1 is a T-invariant of N_1, then $J_2 = \eta(J_1)$ is a T-invariant of N_2

$\hat{N}$-morphisms: properties preserved/reflected

$(\beta, \eta) : N_1 \to N_2$

- S-invariants are reflected:
  if I_2 is an S-invariant of N_2, then $I_1 = \beta^{-1}(I_2)$ is an S-invariant of N_1
- S-invariants are **not** preserved
- T-invariants are preserved:
  if J_1 is a T-invariant of N_1, then $J_2 = \eta(J_1)$ is a T-invariant of N_2
- T-invariants are **not** reflected

Preserving properties



- It is possible that N_I, N_1, N_2 are live, but $N_1[N_I]N_2$ is not live;
- however, ...

reflecting sequences

Definition

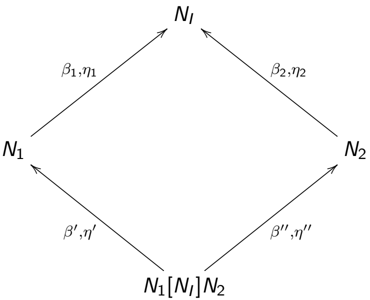
FS(N) firing sequences of N,

$(\beta, \eta) : N \rightarrow N'$ $\hat{\text{N}}$-morphism

N reflects the sequences of N' under (β, η) iff

$\forall v \in \text{FS}(N'), \exists w \in \text{FS}(N)$ such that: $\hat{\eta}(w) = v$

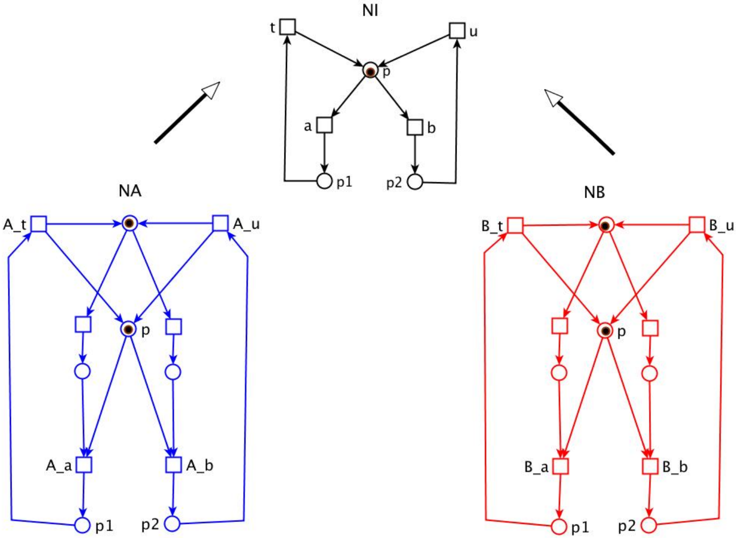
Preserving properties



If N_1 and N_2 *reflect the sequences* of N_I,respectively,

then $N_1[N_I]N_2$ reflects the sequences of N_1, N_2 and N_I.

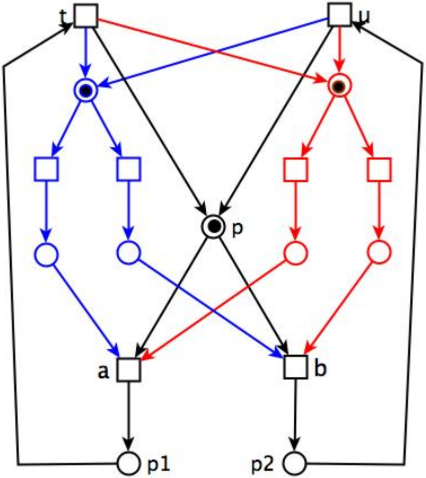
Deadlock-freeness?



N_I, N_A and N_B are deadlock-free and even live.

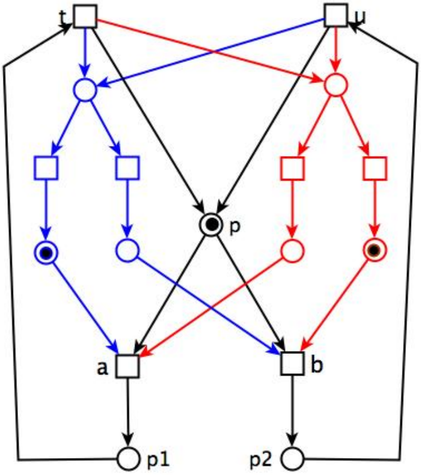
Deadlock-freeness?

$N_A[N_I]N_B$



Deadlock-freeness?

$N_A[N_I]N_B$



$N_A[N_I]N_B$ is dead

Weak Bisimulation[4]

if N' and N'' are *weakly bisimilar* ($N' \approx^{BIS} N''$)

then N' is deadlock-free **iff** N'' is deadlock-free

[4] R. Milner, A Calculus of Communicating Systems, 1980

Weak Bisimulation[4]

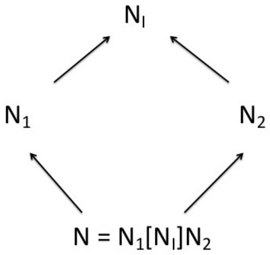
if N' and N'' are *weakly bisimilar* ($N' \approx^{BIS} N''$)

then N' is deadlock-free **iff** N'' is deadlock-free

Remark
Weak Bisimulation is verified considering the reachability graphs.

[4] R. Milner, A Calculus of Communicating Systems, 1980

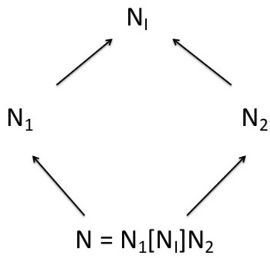
Deadlock-freeness?



Theorem

$$N_1 \approx^{BIS} N_I \quad \Rightarrow \quad N \approx^{BIS} N_2$$

Deadlock-freeness?

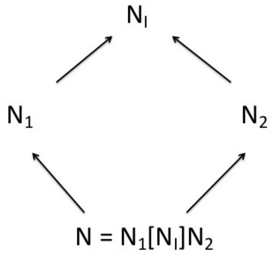


Theorem

$$N_1 \approx^{BIS} N_I \quad \Rightarrow \quad N \approx^{BIS} N_2$$

if $N' \approx^{BIS} N''$ **and** N' is deadlock-free, **then** N'' is deadlock-free

Deadlock-freeness?



Theorem

$$N_1 \approx^{BIS} N_I \quad \Rightarrow \quad N \approx^{BIS} N_2$$

if $N' \approx^{BIS} N''$ **and** N' is deadlock-free, **then** N'' is deadlock-free

Corollary

$$(N_1 \approx^{BIS} N_I \quad \wedge \quad N_2 \textit{ deadlock-free}) \Rightarrow N \textit{ deadlock-free}$$

Section 5

Refinement and composition based on a *new* notion of morphisms: α-*morphisms*

Section 5

Refinement and composition based on a *new* notion of morphisms: α-*morphisms*

for Elementary Net systems, covered by sequential components

Main aim

Refinement/abstraction and composition

preserving/reflecting properties

by considering behaviours, **only locally**

Main aim

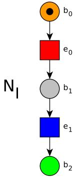
Refinement/abstraction and composition

preserving/reflecting properties

by considering behaviours, **only locally**

Composition on Interfaces using α-morphisms: the idea [5] [6]

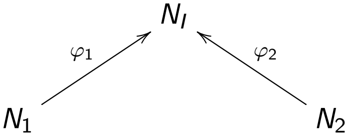
N_I

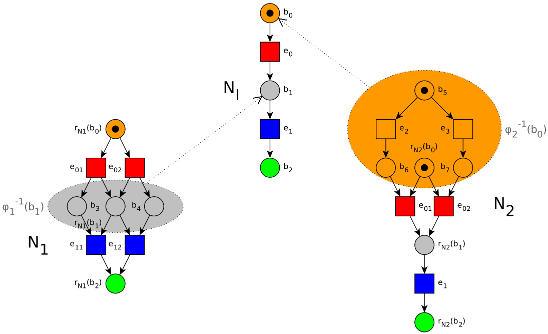


[5] Luca Bernardinello, Elisabetta Mangioni and Lucia Pomello, Local State Refinement and Composition of Elementary Net Systems: An Approach Based on Morphisms, ToPNoC VIII, 2013

[6] Elisabetta Mangioni, Modularity for System Modelling and Analysis, PhD Thesis, Univ. Milano-Bicocca, 2013

Composition on Interfaces using α-morphisms: the idea [5] [6]



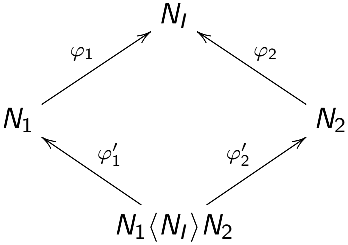


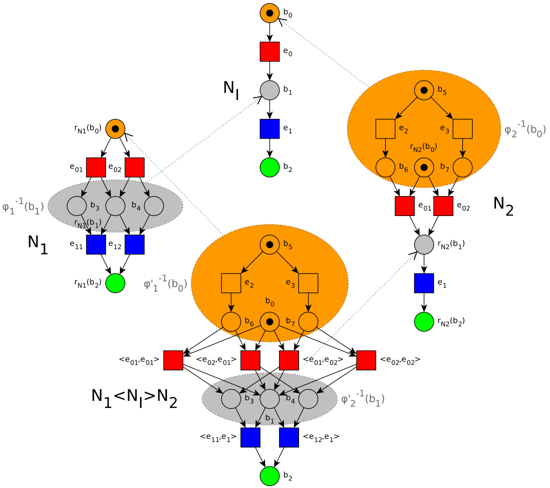
---

[5] Luca Bernardinello, Elisabetta Mangioni and Lucia Pomello, Local State Refinement and Composition of Elementary Net Systems: An Approach Based on Morphisms, ToPNoC VIII, 2013

[6] Elisabetta Mangioni, Modularity for System Modelling and Analysis, PhD Thesis, Univ. Milano-Bicocca, 2013

Composition on Interfaces using α-morphisms: the idea [5] [6]





---

[5] Luca Bernardinello, Elisabetta Mangioni and Lucia Pomello, Local State Refinement and Composition of Elementary Net Systems: An Approach Based on Morphisms, ToPNoC VIII, 2013

[6] Elisabetta Mangioni, Modularity for System Modelling and Analysis, PhD Thesis, Univ. Milano-Bicocca, 2013

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \rightarrow N_2$ α-morphism :

- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \rightarrow N_2$ α-morphism :

- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble
- Sequential components are reflected
  - counter image of a sequential component is covered by sequential components

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \rightarrow N_2$ α-morphism :

- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble
- Sequential components are reflected
  - counter image of a sequential component is covered by sequential components
- Sequential components are not preserved

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \to N_2$ α-morphism :

- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble
- Sequential components are reflected
  - counter image of a sequential component is covered by sequential components
- Sequential components are not preserved
- Reachable markings are preserved

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \rightarrow N_2$ α-morphism :

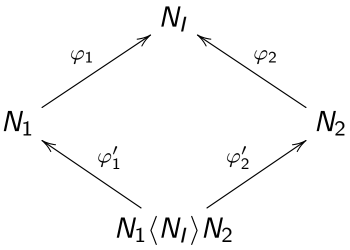
- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble
- Sequential components are reflected
  - counter image of a sequential component is covered by sequential components
- Sequential components are not preserved
- Reachable markings are preserved
- Reachable markings are reflected **iff**
  1. the initial marking of each bubble is at the start of the bubble itself
  2. no deadlock internal to a bubble and
     each final marking of a bubble enables the "same" set of events enabled by its image

Properties preserved and reflected by α-morphisms

$\varphi : N_1 \rightarrow N_2$ α-morphism :

- "Good behaviour" of a "bubble"
  - if an entering event to a bubble can fire then the bubble is empty
  - if an outgoing event from a bubble fires, it empties the bubble
- Sequential components are reflected
  - counter image of a sequential component is covered by sequential components
- Sequential components are not preserved
- Reachable markings are preserved
- Reachable markings are reflected **iff**
  1. the initial marking of each bubble is at the start of the bubble itself
  2. no deadlock internal to a bubble and
     each final marking of a bubble enables the "same" set of events enabled by its image
  3. a "Local unfolding" condition

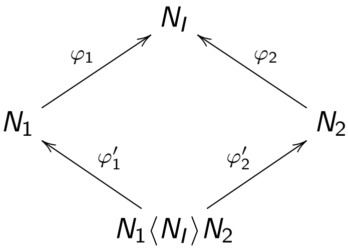
Composition based on α-morphisms, formal results



Proposition

- *α-morphism: $N_1 \to N_I$ and* $\mathbf{1} + \mathbf{2} + \mathbf{3}$
  $\Rightarrow$ *N_1 is weakly bisimilar to N_I;*

Composition based on α-morphisms, formal results



Proposition

- *α-morphism: $N_1 \to N_I$ and* $\mathbf{1} + \mathbf{2} + \mathbf{3}$
  $\Rightarrow$ *N_1 is weakly bisimilar to N_I;*

- *α-morphism: $N_1 \to N_I$ and* $\mathbf{1} + \mathbf{2} + \mathbf{3}$ *and α-morphism: $N_2 \to N_I$*
  $\Rightarrow$ *$N_1 \langle N_I \rangle N_2$ is weakly bisimilar to N_2.*

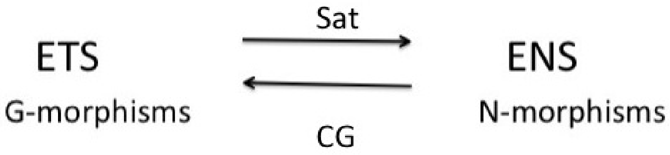
Section 6

Application to modular synthesis (on the basis of $\hat{N}$-morphisms)

Synthesis

$A = (S, E, T)$
Elementary Transition Systems

N = (B, E, F)
Elementary Nets Systems



[Nielsen, Rozenberg, Thiagarajan '92]

Modular synthesis

ETS and Ĝ-morphisms

$$
\begin{array}{ccc}
A_I & \xleftarrow{\;h_1\;} & A_1 \\
{\scriptstyle h_2}\uparrow & & \\
A_2 & &
\end{array}
$$

Modular synthesis

ETS and Ĝ-morphisms

$$\begin{array}{ccc} A_I & \xleftarrow{h_1} & A_1 \\ {\scriptstyle h_2}\uparrow & & \\ A_2 & & \end{array}$$

ENS and N̂-morphisms

$$\begin{array}{ccc} N(A_I) & \xleftarrow{N(h_1)} & N(A_1) \\ {\scriptstyle N(h_2)}\uparrow & & \\ N(A_2) & & \end{array}$$

Modular synthesis

ETS and Ĝ-morphisms

$$\begin{array}{ccc} A_I & \xleftarrow{h_1} & A_1 \\ {\scriptstyle h_2}\uparrow & & \\ A_2 & & \end{array}$$

ENS and N̂-morphisms

$$\begin{array}{ccc} N(A_I) & \xleftarrow{N(h_1)} & N(A_1) \\ {\scriptstyle N(h_2)}\uparrow & & \uparrow{\scriptstyle n_1} \\ N(A_2) & \xleftarrow{n_2} & N \end{array}$$

Modular synthesis

ETS and Ĝ-morphisms

$$
\begin{array}{ccc}
A_I & \xleftarrow{h_1} & A_1 \\
{\scriptstyle h_2}\uparrow & & \uparrow{\scriptstyle g_1} \\
A_2 & \xleftarrow{g_2} & A
\end{array}
$$

ENS and N̂-morphisms

$$
\begin{array}{ccc}
N(A_I) & \xleftarrow{N(h_1)} & N(A_1) \\
{\scriptstyle N(h_2)}\uparrow & & \uparrow{\scriptstyle n_1} \\
N(A_2) & \xleftarrow{n_2} & N
\end{array}
$$

$CG(N)$ is isomorphic to A

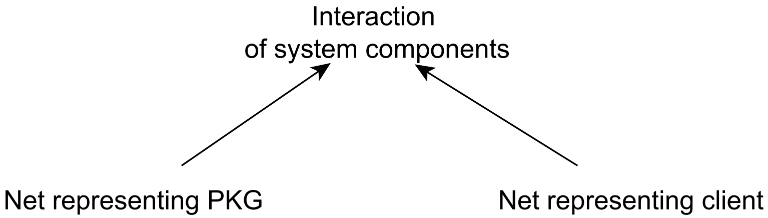
[Bernardinello, Ferigato, Pomello 02]

Section 7

A case study

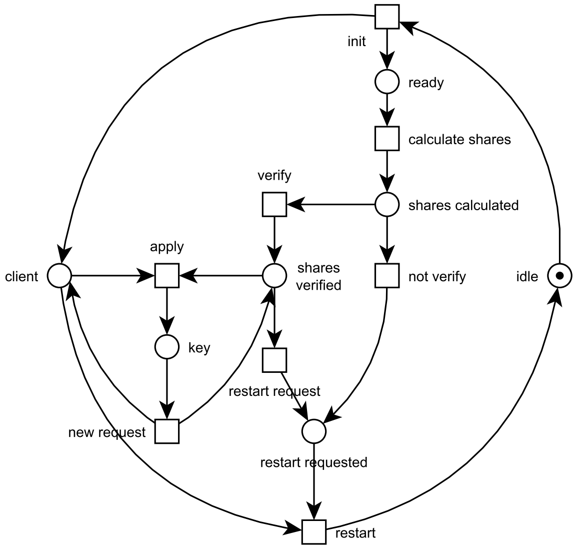
Modeling and Analyzing a Distributed Private Key Generation Protocol

Modeling Distributed Private Key Generation Protocol [7]



[7] L. Bernardinello, G. Kılınç, E. Mangioni, L. Pomello, *Modeling Distributed Private Key Generation by Composing Petri Nets*, Int. Workshop PNSE'13, 2013 (to appear in TopNoC, Springer).

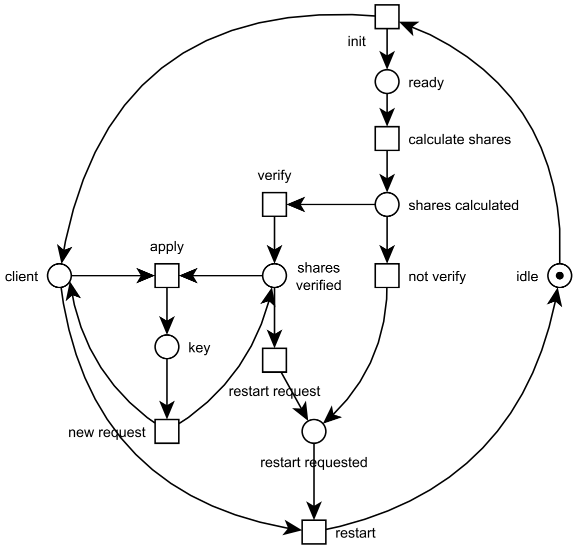
Interface Net, N_I



Interface

- an abstract view of the system
- represents the communication between components

Interface Net, N_I



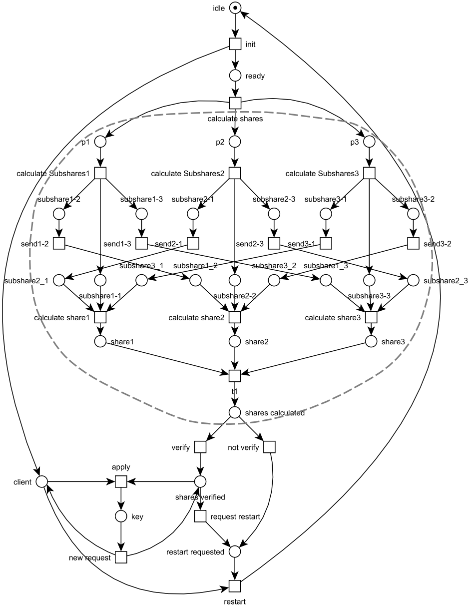
Interface

- an abstract view of the system
- represents the communication between components

Properties

- live
- reversible
- covered by sequential components

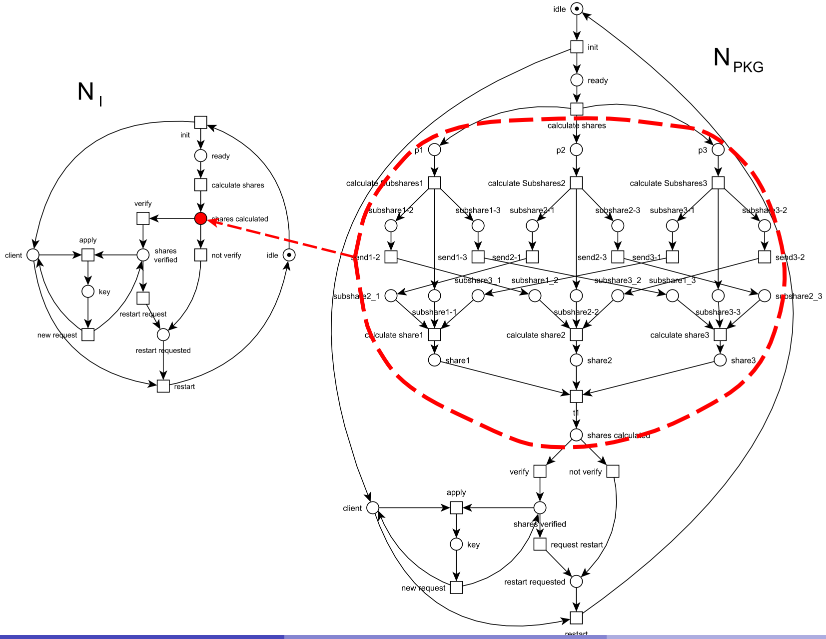
Net Representing PKG, N_{PKG}



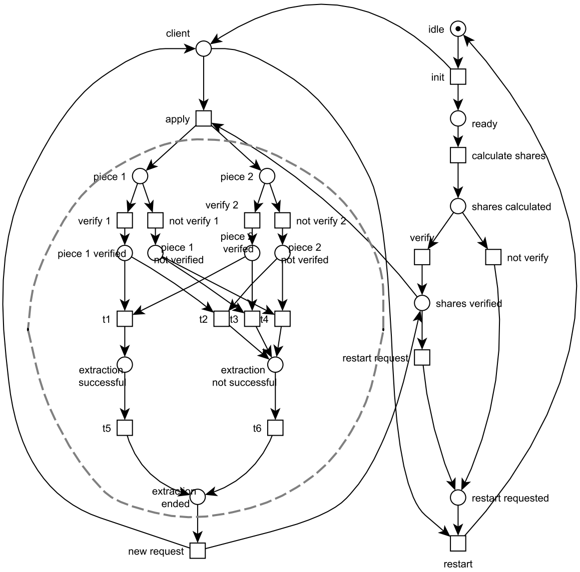
Properties

- live
- reversible
- covered by sequential components

α-morphism

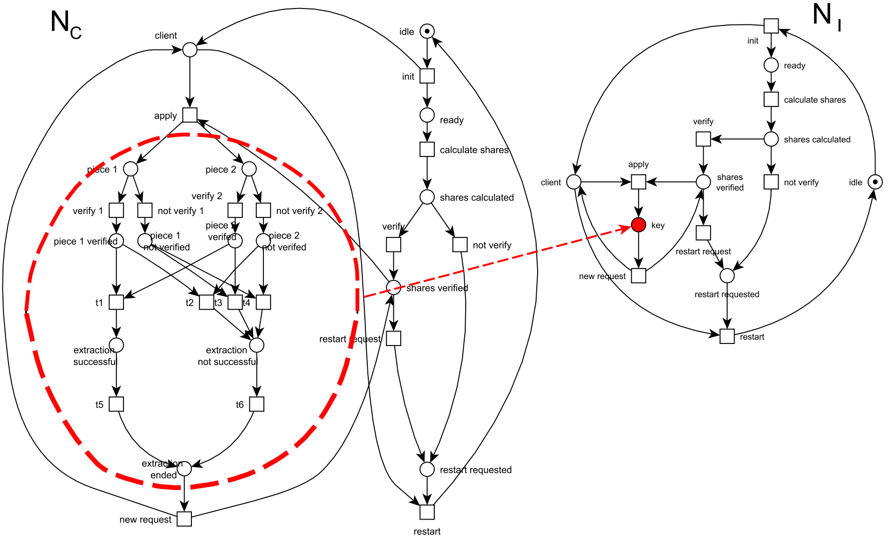


Net Representing the Client, N_C



Properties

- live
- reversible
- covered by sequential components



N C
client
apply
piece 1
piece 2
verify 1
not verify 1
verify 2
not verify 2
piece 1 verified
piece 1 not verified
piece 2 verifed
piece 2 not verifed
t1
t2
t3
t4
extraction successful
extraction not successful
t5
t6
extraction ended
new request
idle
init
ready
calculate shares
shares calculated
verify
not verify
shares verified
restart request
restart requested
restart
N I
init
ready
calculate shares
verify
shares calculated
apply
client
shares verified
not verify
idle
key
restart request
new request
restart requested
restart

- There is an α-morphism both from N_{PKG} to N_I and from N_C to N_I.
- Additional requirements are satisfied.

- There is an α-morphism both from N_{PKG} to N_I and from N_C to N_I.
- Additional requirements are satisfied.

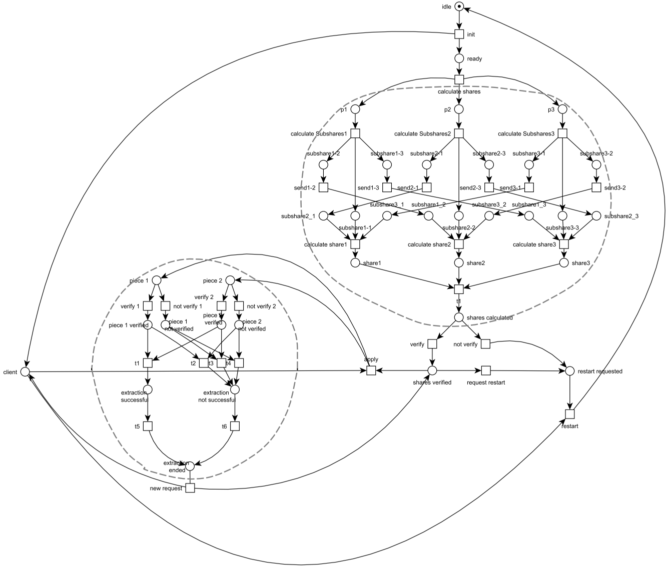
- Reflection of reachable markings property is held.
- Weakly bisimulation property is held.
  - N_{PKG} is weakly bisimilar to N_I.
  - N_C is weakly bisimilar to N_I.

- There is an α-morphism both from N_{PKG} to N_I and from N_C to N_I.
- Additional requirements are satisfied.

⇓

- Reflection of reachable markings property is held.
- Weakly bisimulation property is held.
  - N_{PKG} is weakly bisimilar to N_I.
  - N_C is weakly bisimilar to N_I.
  - Consequently, $N_{PKG}\langle N_I \rangle N_C$ is weakly bisimilar to N_I.

Analysis on the Composed Net . . .



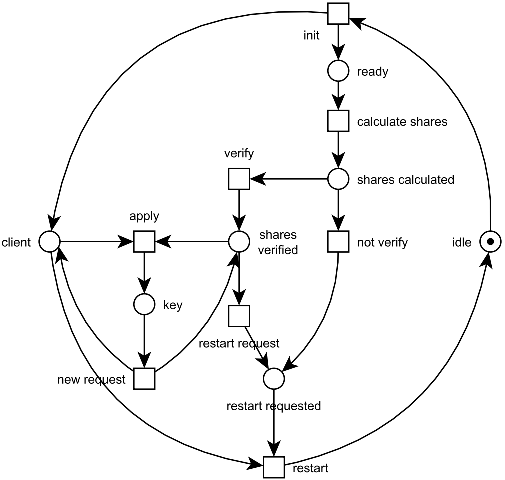
Ex:

A property to be analyzed:

“Shares cannot be verified while distribution or extraction process is continuing”

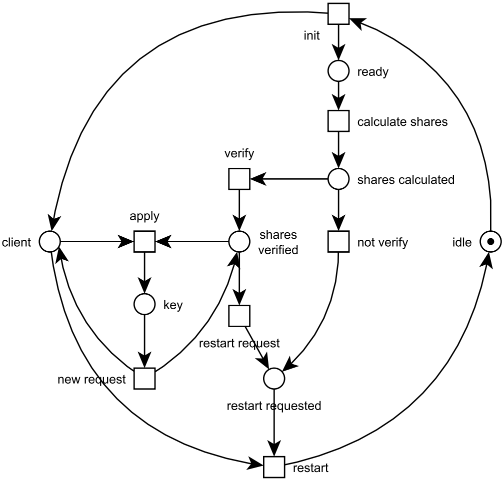
Analysis on the Composed Net can be done directly on the Interface

even without computing the Composed Net



Analysis on the Composed Net can be done directly on the Interface

even without computing the Composed Net



A property to be analyzed: "Shares cannot be verified while distribution or extraction process is continuing"

Corresponding CTL formulae

EXPATH EVENTUALLY shares verified AND key AND shares calculated

Section 8

Remarks and Conclusions

Remarks and Conclusions

- $\hat{N}$-morphisms and the composition on $\hat{N}$-morphisms have been defined also for P/T nets
- an other notion of morphisms for marked graphs has been studied (paper just submitted to PNSE 2014)

Remarks and Conclusions

- $\hat{\text{N}}$-morphisms and the composition on $\hat{\text{N}}$-morphisms have been defined also for P/T nets
- an other notion of morphisms for marked graphs has been studied (paper just submitted to PNSE 2014)

Future?

- Define and study α-morphisms and the other just defined notion for more general classes (e.g.: P/T nets, high level nets, Coloured nets,..)

Remarks and Conclusions

- $\hat{\text{N}}$-morphisms and the composition on $\hat{\text{N}}$-morphisms have been defined also for P/T nets
- an other notion of morphisms for marked graphs has been studied (paper just submitted to PNSE 2014)

Future?

- Define and study α-morphisms and the other just defined notion for more general classes (e.g.: P/T nets, high level nets, Coloured nets,..)
- Morphisms and compositionality on Petri Hypernets or on Nested nets

References

- M. A. Bednarczyk, L. Bernardinello, B. Caillaud, W. Pawlowski, L. Pomello, *Modular system development with pullbacks*, in ATPN2003, LNCS 2679, 140 - 160, Springer, 2003.
- L. Bernardinello, C. Ferigato, L. Pomello, *Towards Modular Synthesis of EN Systems*, in B. Caillaud, P. Darondeau, L. Lavagno, X. Xie (eds.), "Synthesis and Control of Discrete Event Systems", 103-113, Kluwer, 2002.
- L. Bernardinello, G. Kılınç, E. Mangioni, L. Pomello, *Modeling Distributed Private Key Generation by Composing Petri Nets*, Int. Workshop PNSE'13, 2013 (to appear in TopNoC, Springer).
- Luca Bernardinello, Elisabetta Mangioni, Lucia Pomello, *Composition of Elementary Net Systems based on α-morphisms*, in 2nd Int. Workshop on Petri Nets Compositions Proc.(CompoNet'12), 2012, http://ceur-ws.org/Vol-853/componet2.pdf
- Luca Bernardinello, Elisabetta Mangioni, Lucia Pomello, *Local State Refinement and Composition of Elementary Net Systems: An Approach Based on Morphisms*, ToPNoC VIII, 2013.
- L. Bernardinello, E. Monticelli, L. Pomello *On preserving structural and behavioural properties by composing net systems on interfaces*, Fundamenta Informaticae, Vol. 80, N.1-3, 31- 47, 2007.
- E. Mangioni, *Modularity for System Modelling and Analysis*, PhD Thesis, Univ. Milano-Bicocca, 2013.
- L. Pomello, L. Bernardinello, *Formal Tools for Modular System Development*, in ATPN 2004, LNCS 3099, 77-96, Springer 2004.

THANK YOU !

Спасибо большое !

THANK YOU !

Спасибо большое !

Arrivederci!...